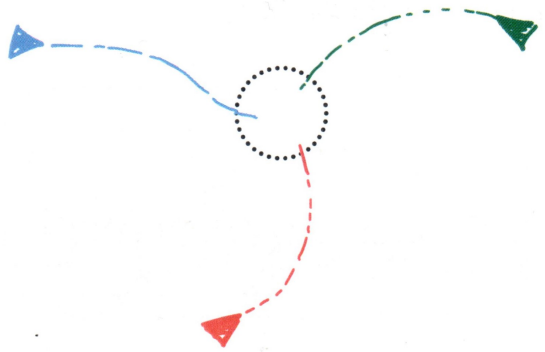
Final Project

Robotics & Control 2 - A.A. 2023/2024

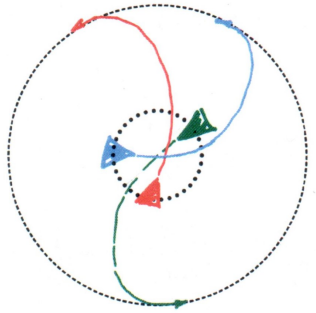
Adami Alessandro - 2089579
Peron Davide- 2082148
Vitetta Emanuele – 2082149

Overview

1) Introduction
2) Rendezvous point [Phase 1]
3) Regulation [Phase 2]
4) Trajectory tracking [Phase 3]
5) Switching logics



(a) Phase 1-2



(b) Phase 3

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

[1] Introduction

1.1) Basics

1.2) Unicycle Model

1.3) Further details

1.4) Evaluation metrics

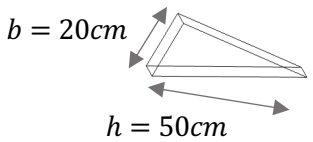
DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

[1.1] Basics

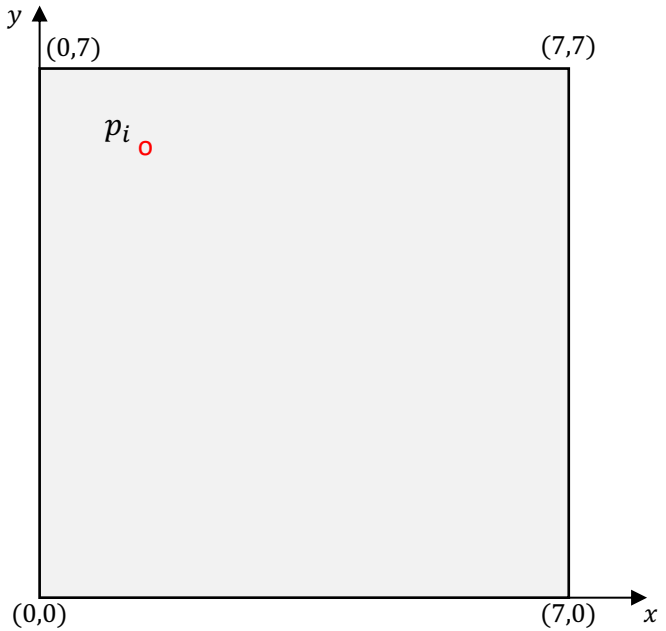
3 Agents, each one defined by an initial position

$$p_i = \begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix}, i = 1,2,3$$

Each agent has a triangular shape of height 10cm



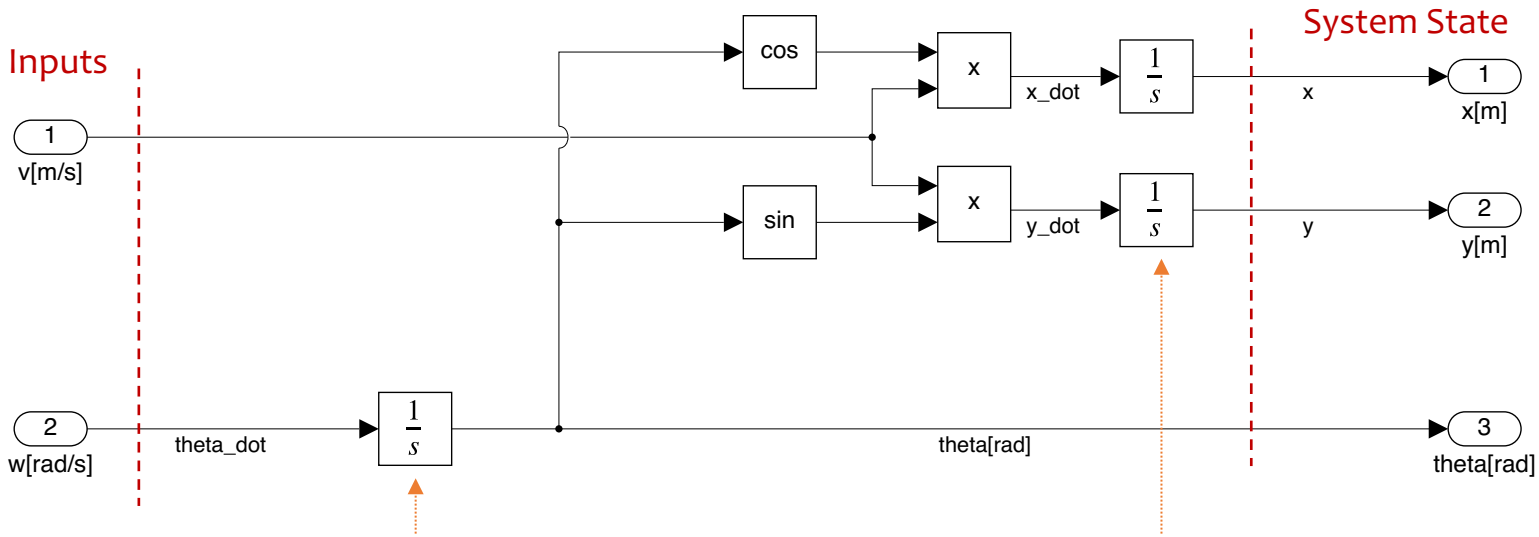
A 7x7 meters room where agents move.



[1.2] Unicycle Model

Dynamic Equations

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



The 3 integratators are initialized with each agent's initial conditions.

[1.3] Further Details

(1) In the whole simulations, we used the *Butterworth derivatives*, characterized by

$$B(s) = \frac{fs}{s^2 + 2\delta fs + f^2}$$

Where the cut-off frequency $f = 2\pi\omega_b$ is determined by $\omega_b = 50\ [rad\backslash s], \delta = {}^{1}/_{\sqrt{2}}$ instead of the. default Simulink block. This choice made all the resulting smoother, cancelling a lot of noise due to numerical approximations.

(2) During simulations, the $atan2$ function is often used. To avoid discontinuities, such function is always followed by an $unwrap$ Simulink block.

[1.4] Performance Metrics

During the realization of the different controllers, we considered some metrics that are used to make comparisons homogenous:

1) A well *bounded workspace* was set. All the controllers aim to keep agents inside such borders.
2) Agents can actuate inputs belonging to certain finite bounds. To emulate this behavior, some *saturation blocks* have been inserted to limit both linear and angular velocity. It has been chosen $\mathrm{v} = 1\ [m\backslash \mathrm{s}],\ \omega = 1.5\ [rad\backslash \mathrm{s}]$.
3) *Convergence time* both to the rendez-vous point and to the desired trajectory are considered.

[2] Rendezvous Point

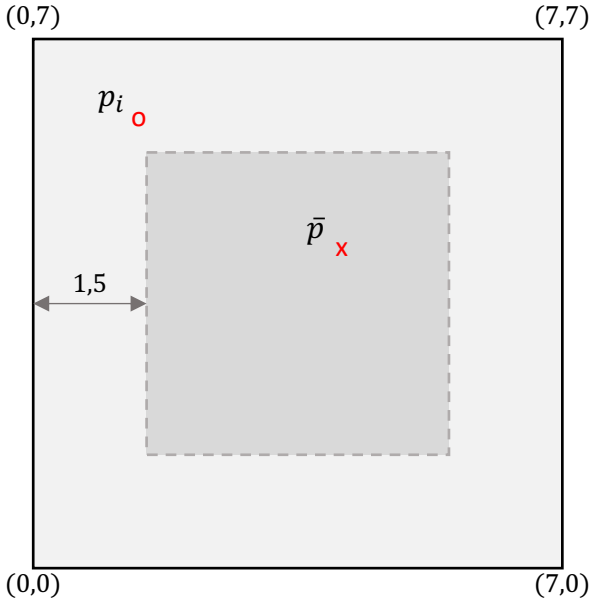
Primitive consensus matrix P (column and row stochastic) to have *average consensus*.

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0.25 & 0.5 & 0.25 \\ 0.25 & 0.25 & 0.5 \end{bmatrix} \overset{\text{Pk}}{\rightarrow} \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}$$

Rendezvous point

$$\bar{p} = \lim_{n \to \infty} P^n x_0$$

Rendezvous point must be inside the smaller dashed square. Hence if any of the components exceed the bounds, this component is forced to the boundary value.



[3] Regulation

The final objective of the regulation task is to bring each agent's center of mass (CoM) to arrive at a final posture $[x_d \quad y_d \quad \theta_d]^T$. This can be done in two ways

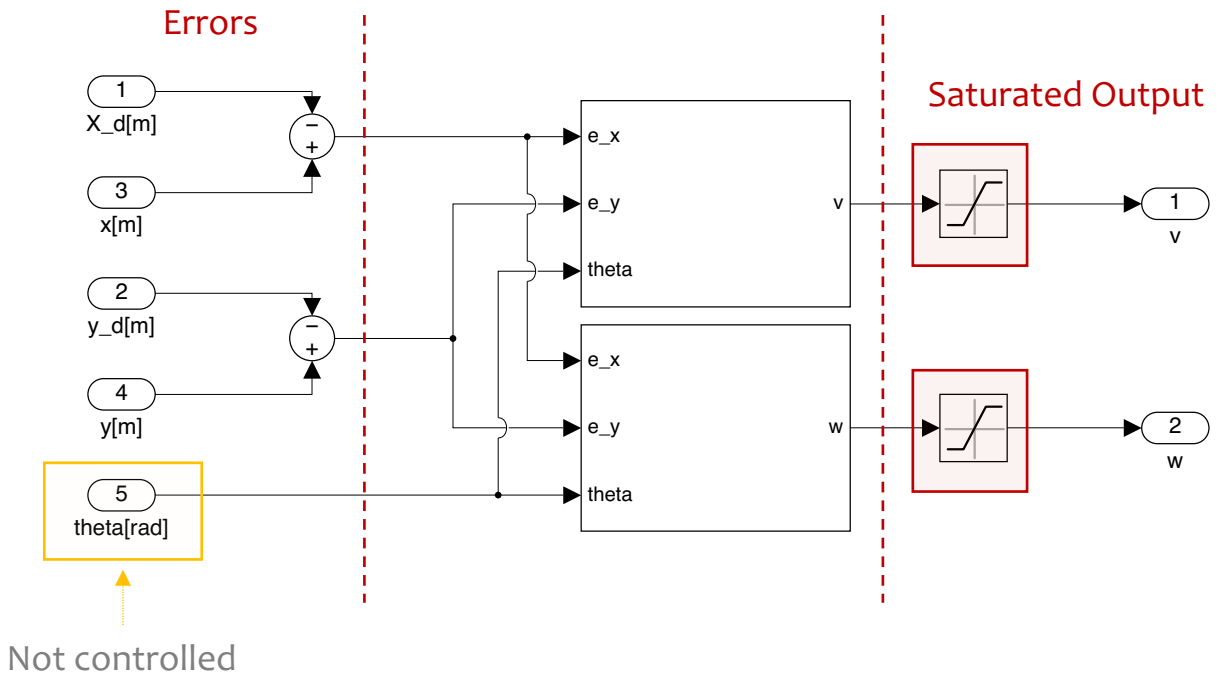
1) *Position* regulation – only (x, y) position is considered;
2) *Posture* regulation – the final orientation is also considered.

It is fundamental to remark that from now on, all the choice for controllers' *gains* have been made to *avoid saturations* of the control signal. Such gains are based on the specific choice of the starting point, rendez-vous point and trajectories.

[3.1] Position regulation

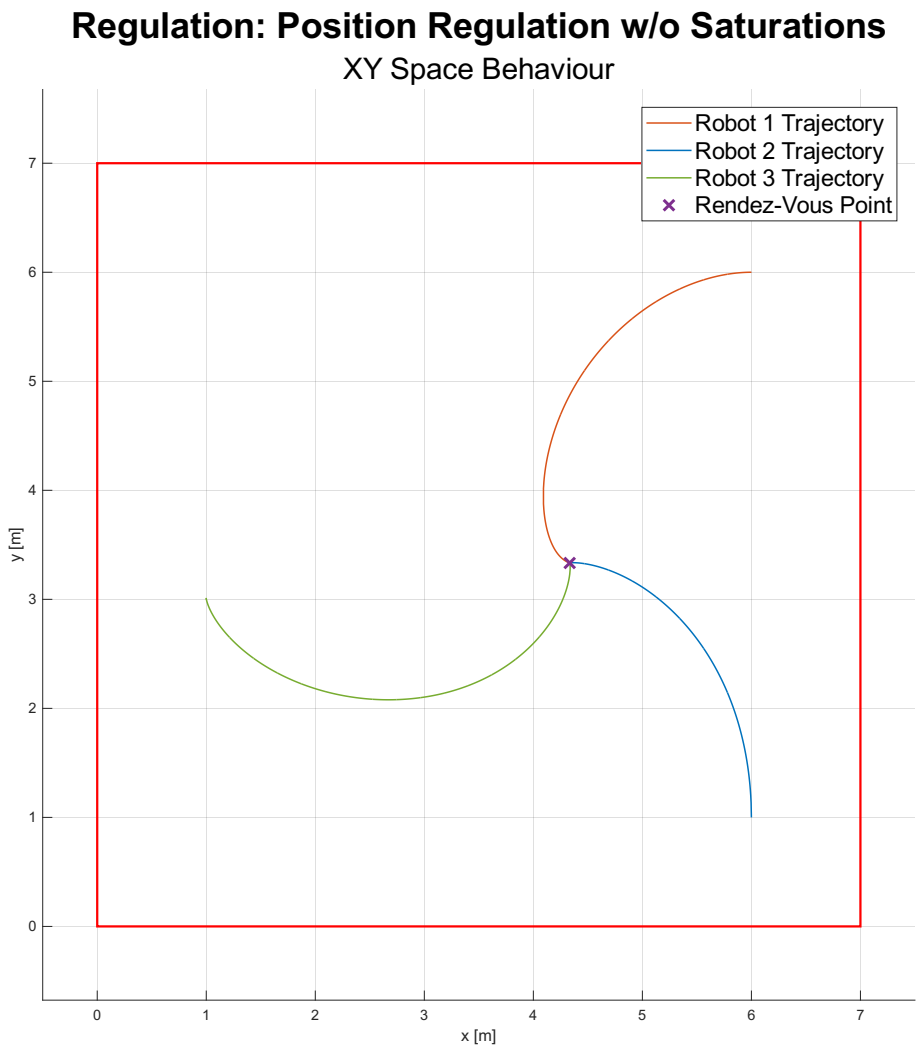
Cartesian errors in *world frame* $\begin{bmatrix} e_x \\ e_y \end{bmatrix} = \begin{bmatrix} x - x_d \\ y - y_d \end{bmatrix}$

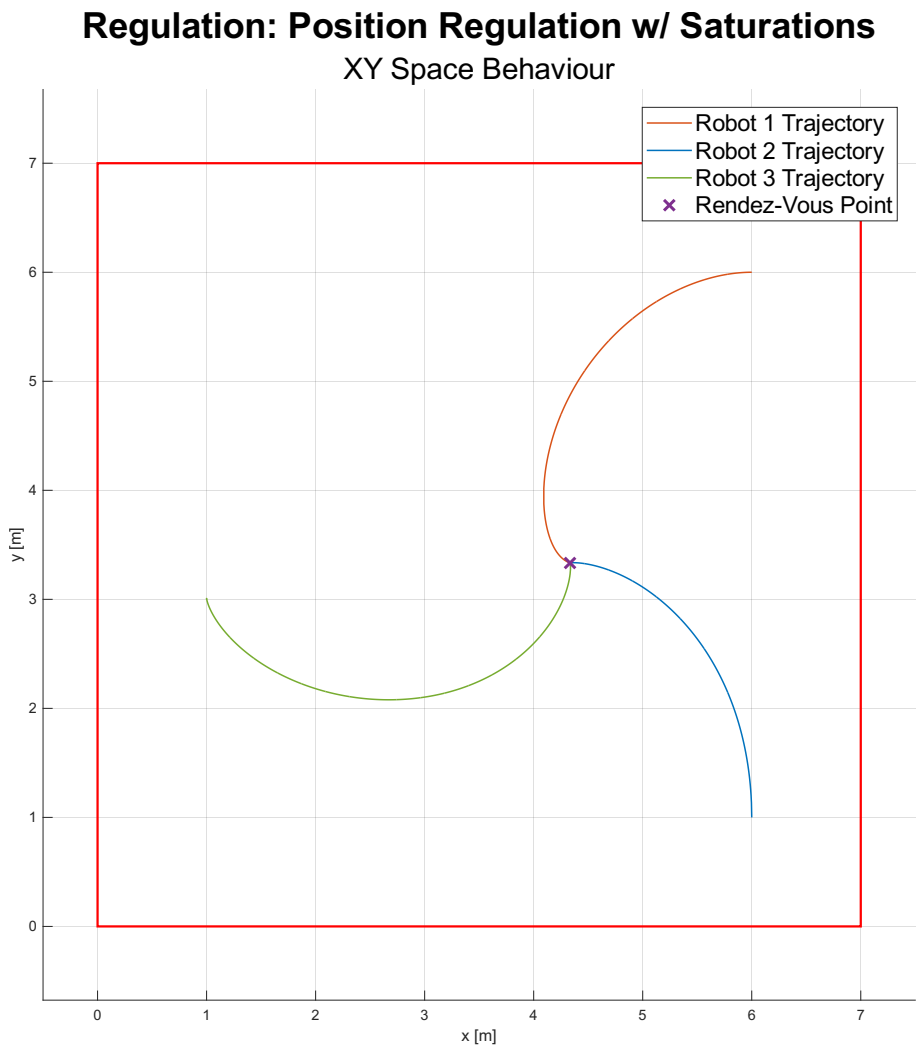
Controller equations $\begin{cases} v = -k_v\left(e_x \cos(\theta) + e_y \sin(\theta)\right) & k_v > 0 \\ \omega = k_\omega\left(atan2\left(e_y, e_x\right) + \pi - \theta\right) & k_\omega > 0 \end{cases} \Rightarrow \begin{cases} k_v = 0.4 \\ k_\omega = 0.5 \end{cases}$



Not controlled

[3.1] Results

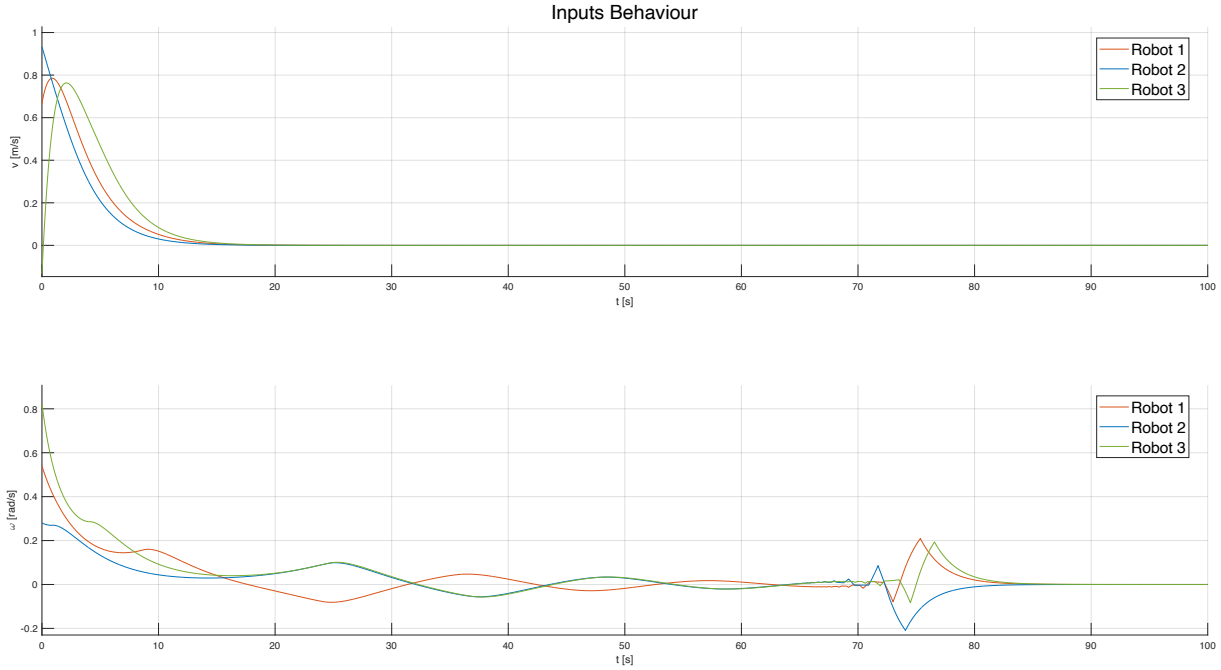

Regulation: Position Regulation w/o Saturations
XY Space Behaviour
Robot 1 Trajectory
Robot 2 Trajectory
Robot 3 Trajectory
Rendez-Vous Point
y [m]
x [m]
0
1
2
3
4
5
6
7


Regulation: Position Regulation w/ Saturations
XY Space Behaviour
Robot 1 Trajectory
Robot 2 Trajectory
Robot 3 Trajectory
Rendez-Vous Point
y [m]
x [m]
0
1
2
3
4
5
6
7

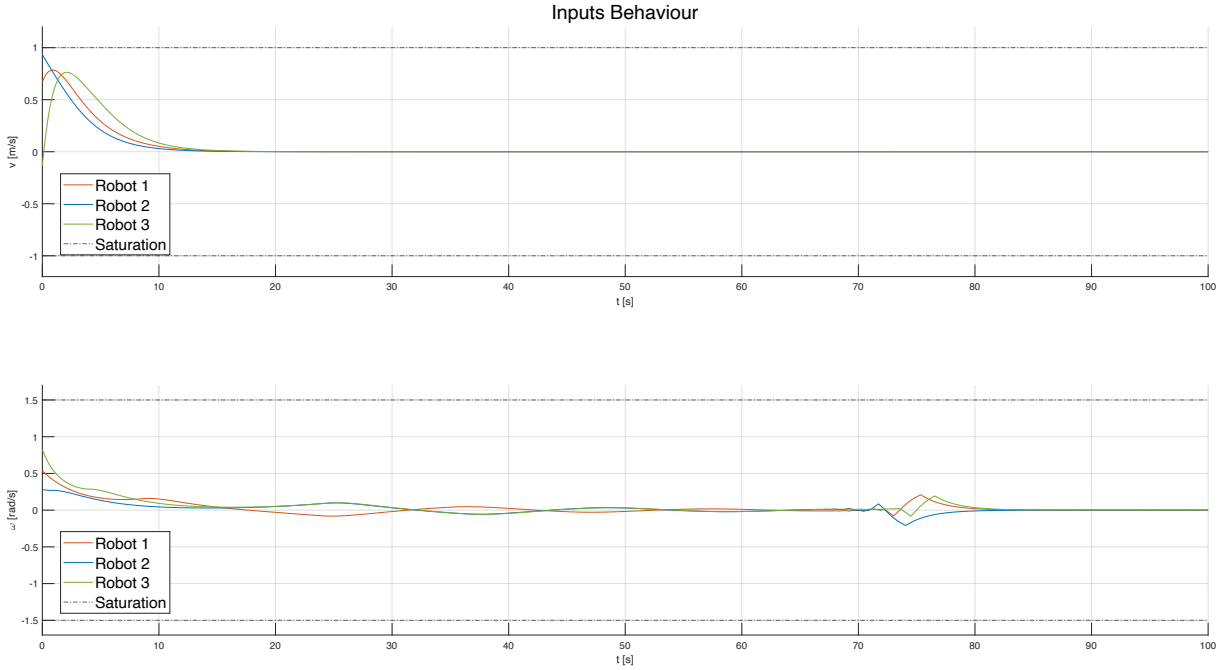
[3.1] Results

Saturation is never reached (with our choice of parameters). Hence the two trajectories coincide.

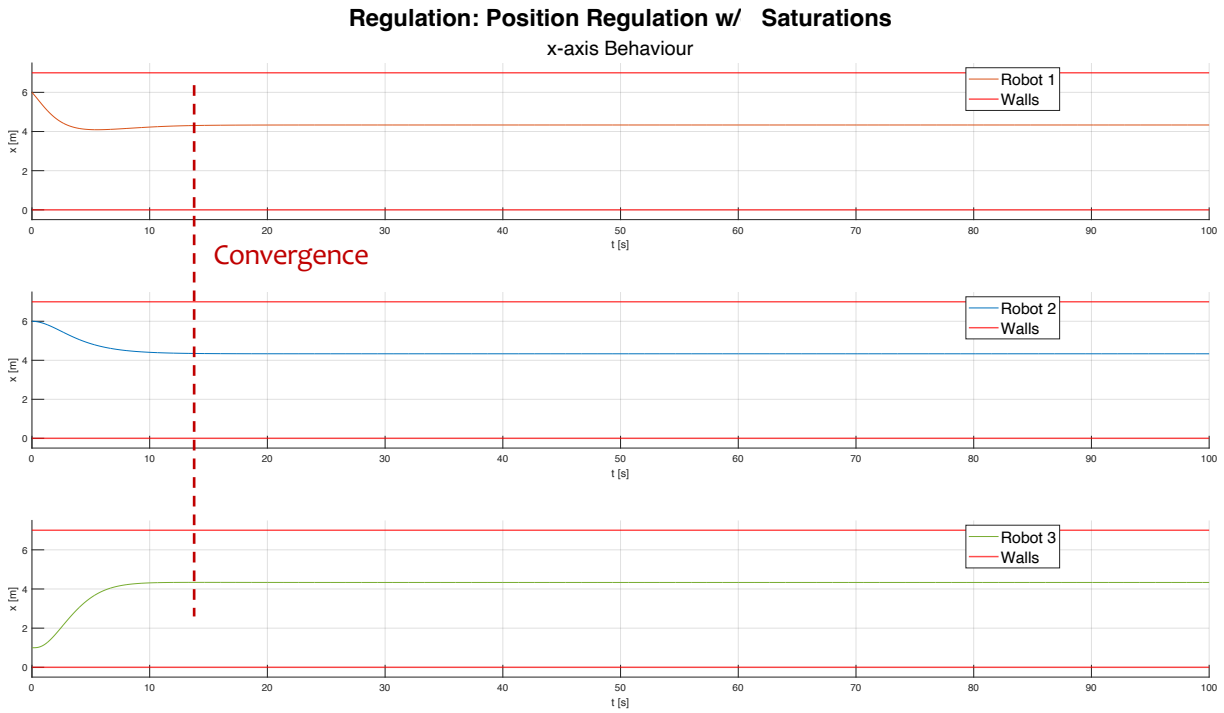
Regulation: Position Regulation w/o Saturations

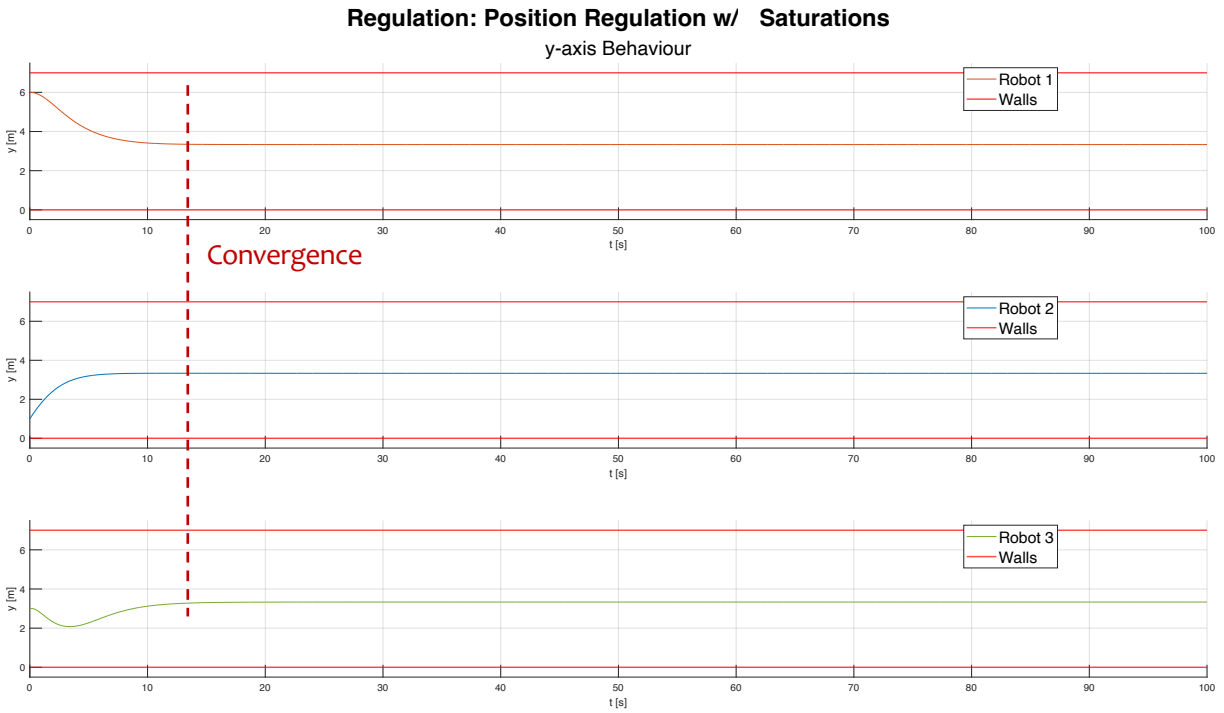


Regulation: Position Regulation w/ Saturations

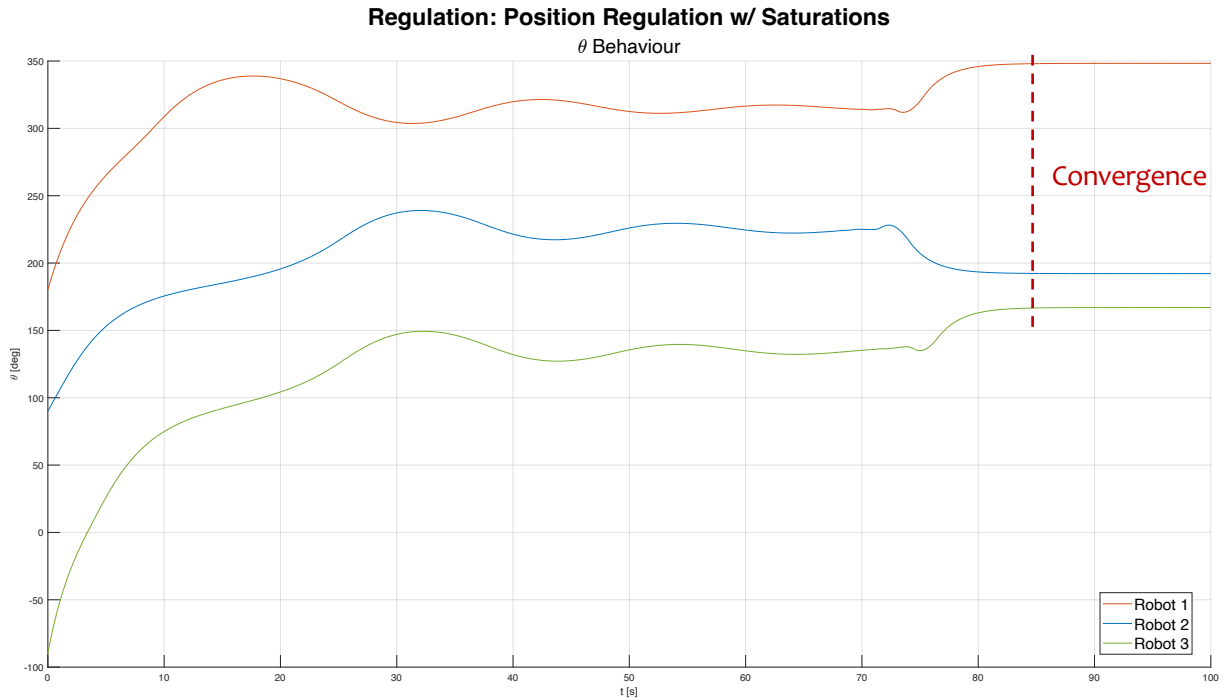


[3.1] Results


Regulation: Position Regulation w/ Saturations
x-axis Behaviour
Robot 1
Walls
x [m]
t [s]
Convergence
Robot 2
Walls
x [m]
t [s]
Robot 3
Walls
x [m]
t [s]


Regulation: Position Regulation w/ Saturations
y-axis Behaviour
Robot 1
Walls
y [m]
t [s]
Convergence
Robot 2
Walls
y [m]
t [s]
Robot 3
Walls
y [m]
t [s]

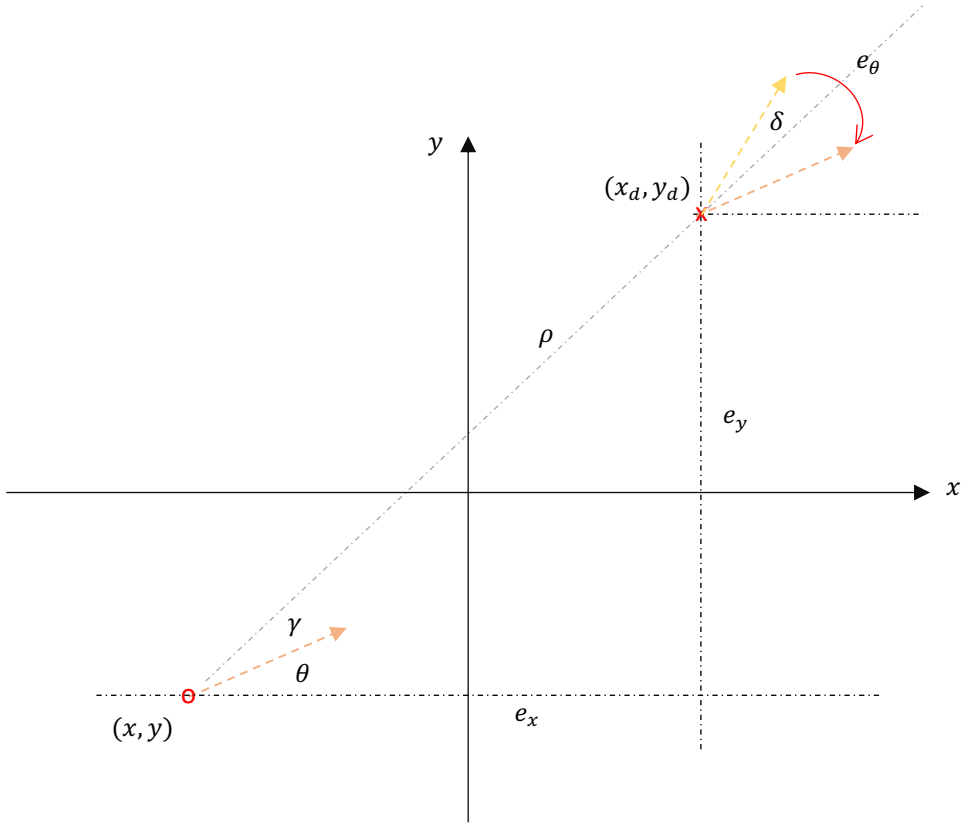
[3.1] Results


Regulation: Position Regulation w/ Saturations
θ Behaviour
θ [deg]
t [s]
Convergence
Robot 1
Robot 2
Robot 3
350
300
250
200
150
100
50
0
-50
-100
0
10
20
30
40
50
60
70
80
90
100

[3.2] Posture regulation

Errors in *world frame* $\begin{bmatrix} e_x \\ e_y \\ e_\theta \end{bmatrix} = \begin{bmatrix} x - x_d \\ y - y_d \\ \theta - \theta_d \end{bmatrix}$

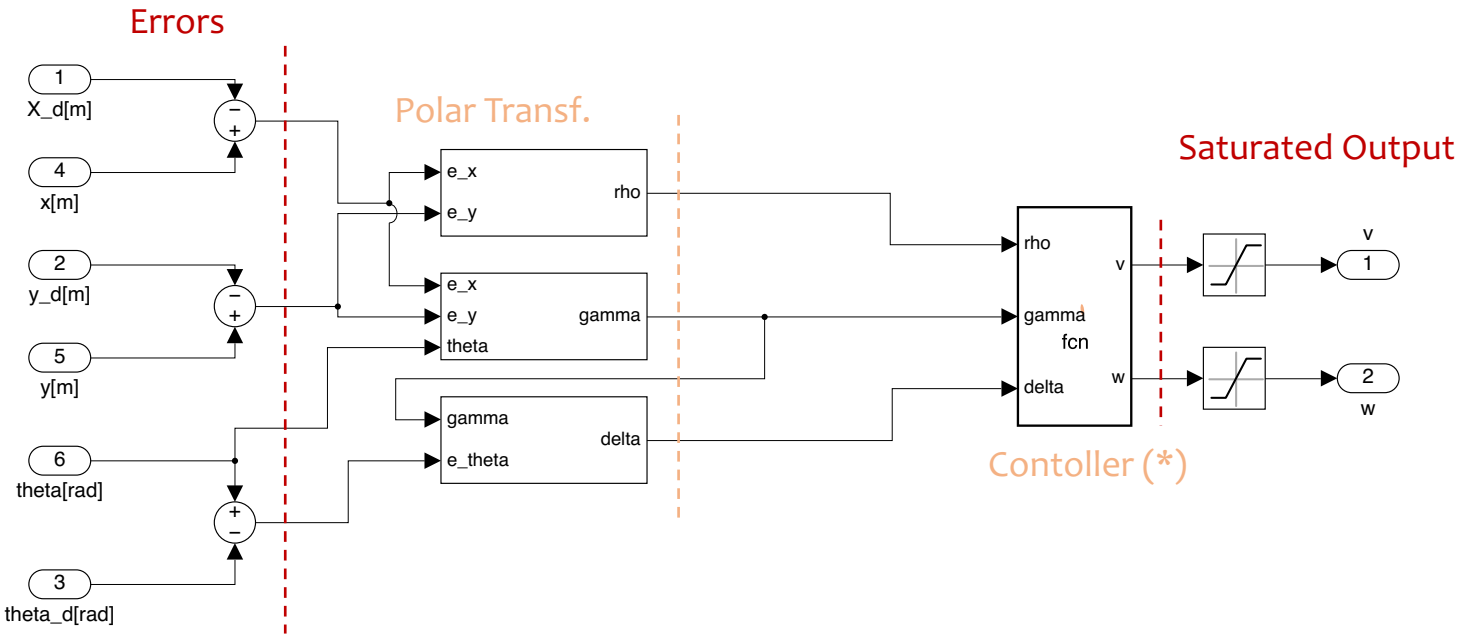
Polar transformation $\begin{cases} \rho = \sqrt{e_x^2 + e_y^2} \\ \gamma = atan2(e_y, e_x) + \pi - \theta \\ \delta = \gamma + e_\theta \end{cases}$



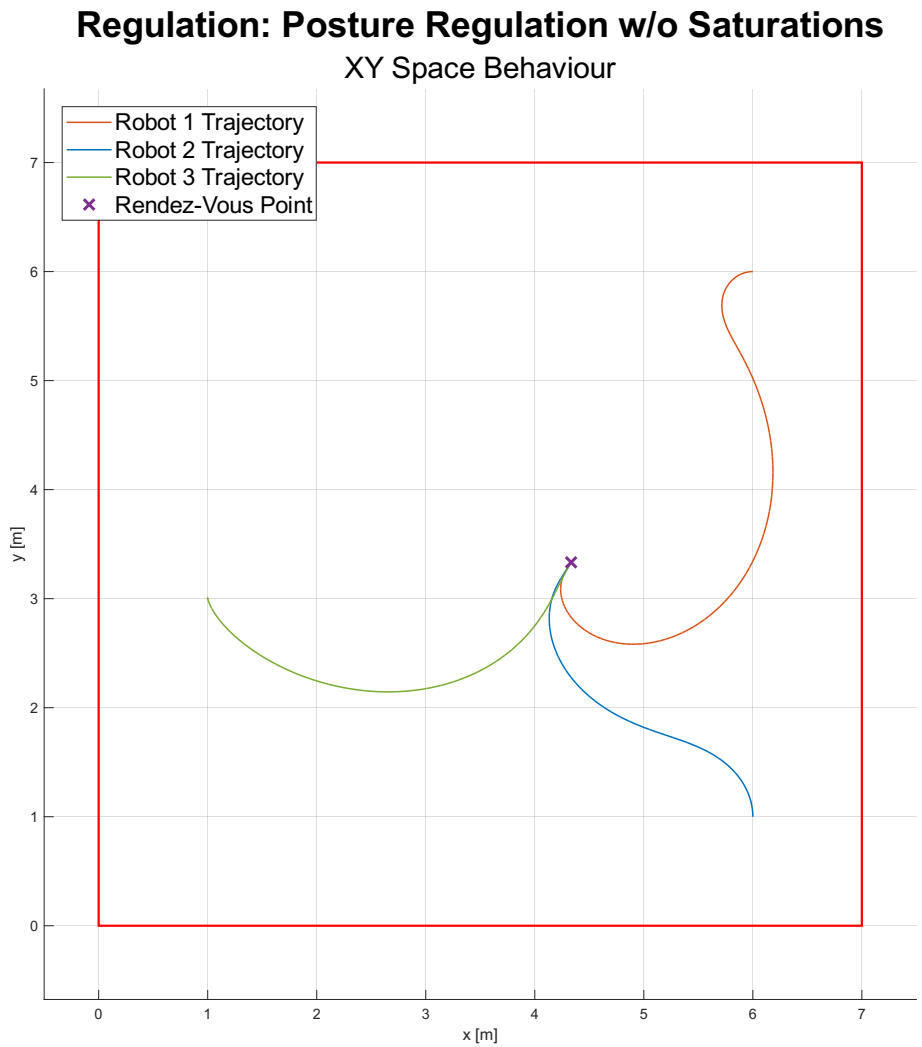
[3.2] Posture regulation

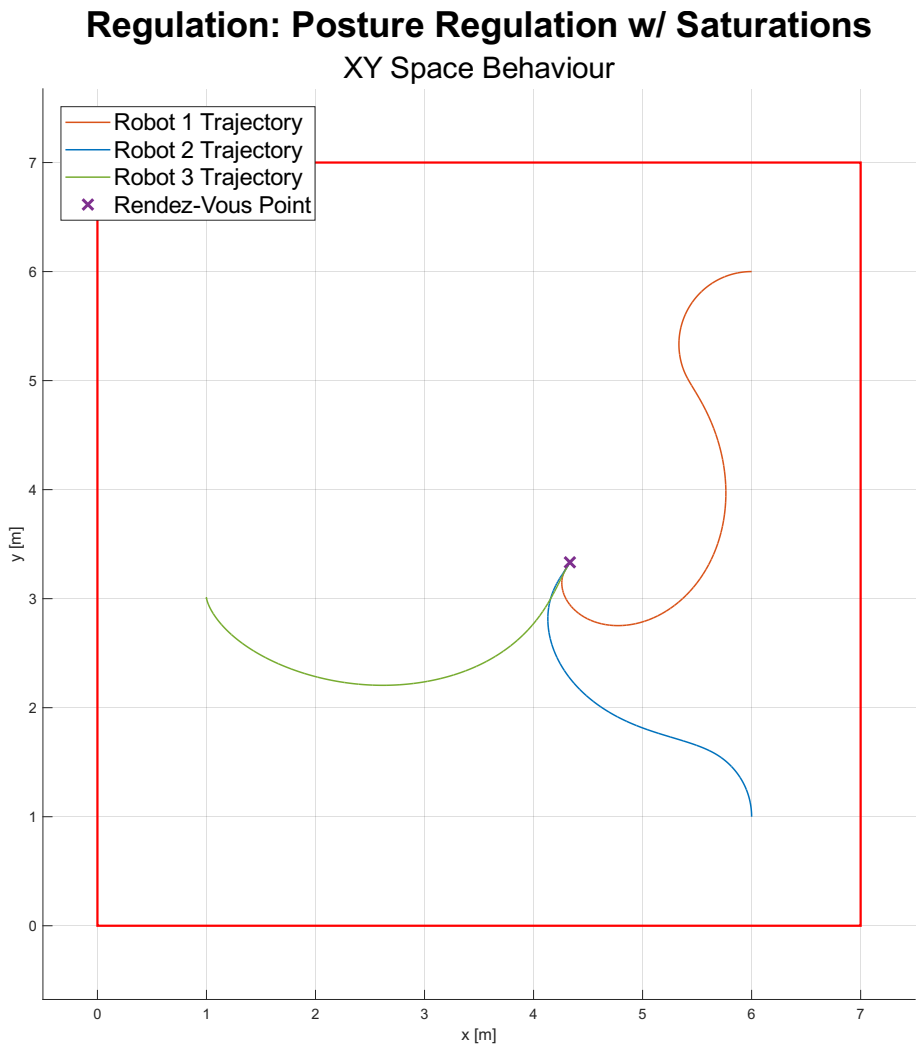
Controller equations $\begin{cases} v = k_1 \rho \cos(\gamma) \\ \omega = k_2 \gamma + k_1 \dfrac{\sin(\gamma)\cos(\gamma)}{\gamma}(\gamma + k_3 \delta) \end{cases}$ $\Longrightarrow$ $\begin{cases} k_1 = 0.7 \\ k_2 = 1.5 \\ k_3 = 3.0 \end{cases}$

(*) (v, ω) both forced to $(0,0)$ near to the singularity $\rho = 0$.

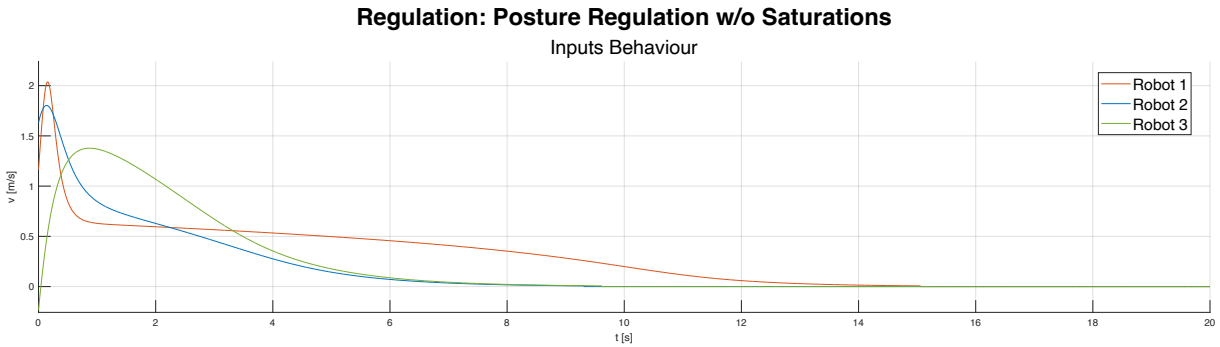


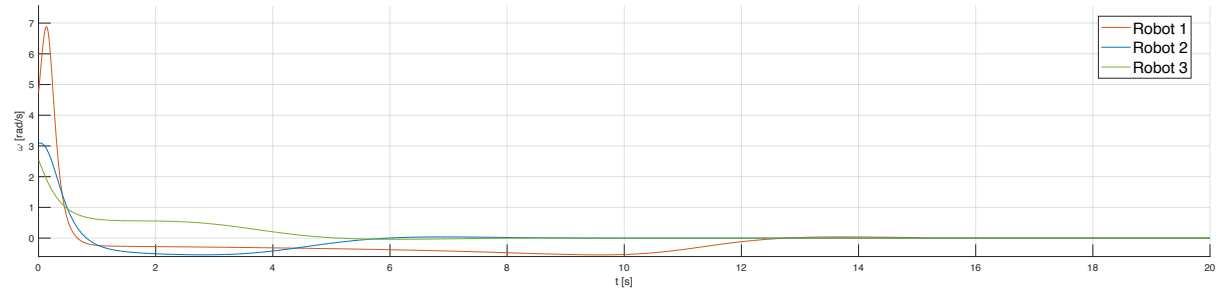
[3.2] Results

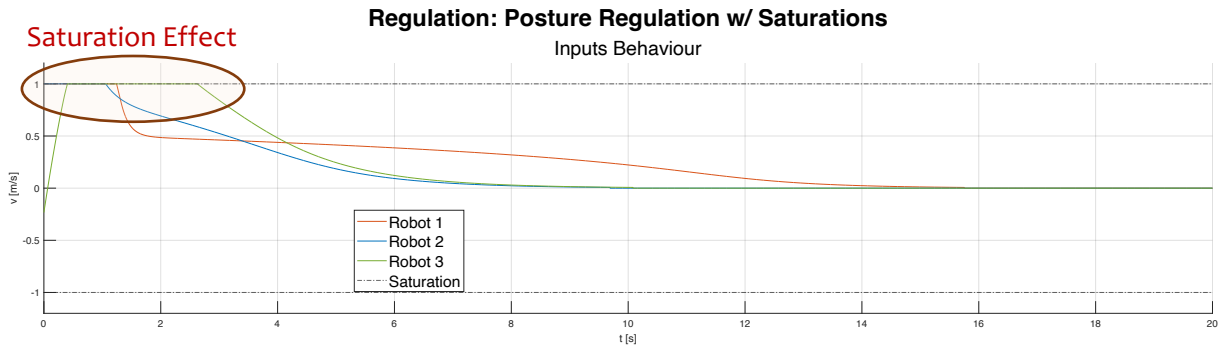

Regulation: Posture Regulation w/o Saturations
XY Space Behaviour
Robot 1 Trajectory
Robot 2 Trajectory
Robot 3 Trajectory
Rendez-Vous Point
y [m]
x [m]
0
1
2
3
4
5
6
7

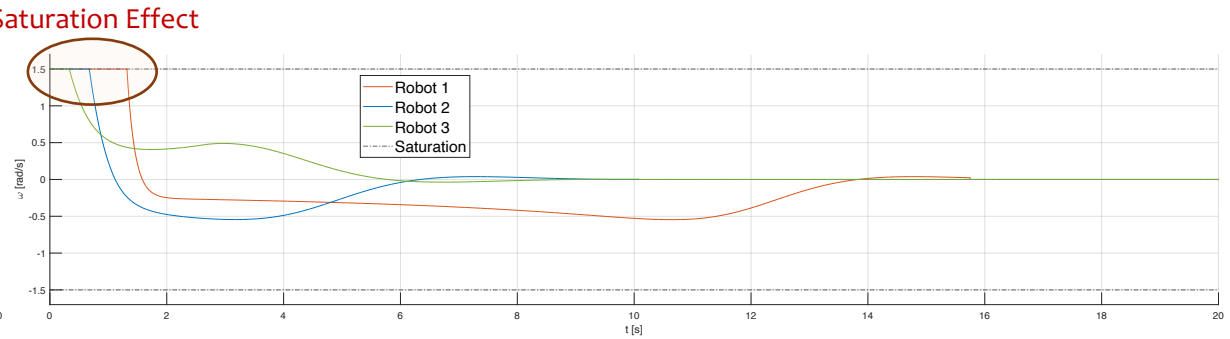

Regulation: Posture Regulation w/ Saturations
XY Space Behaviour
Robot 1 Trajectory
Robot 2 Trajectory
Robot 3 Trajectory
Rendez-Vous Point
y [m]
x [m]
0
1
2
3
4
5
6
7

[3.2] Results

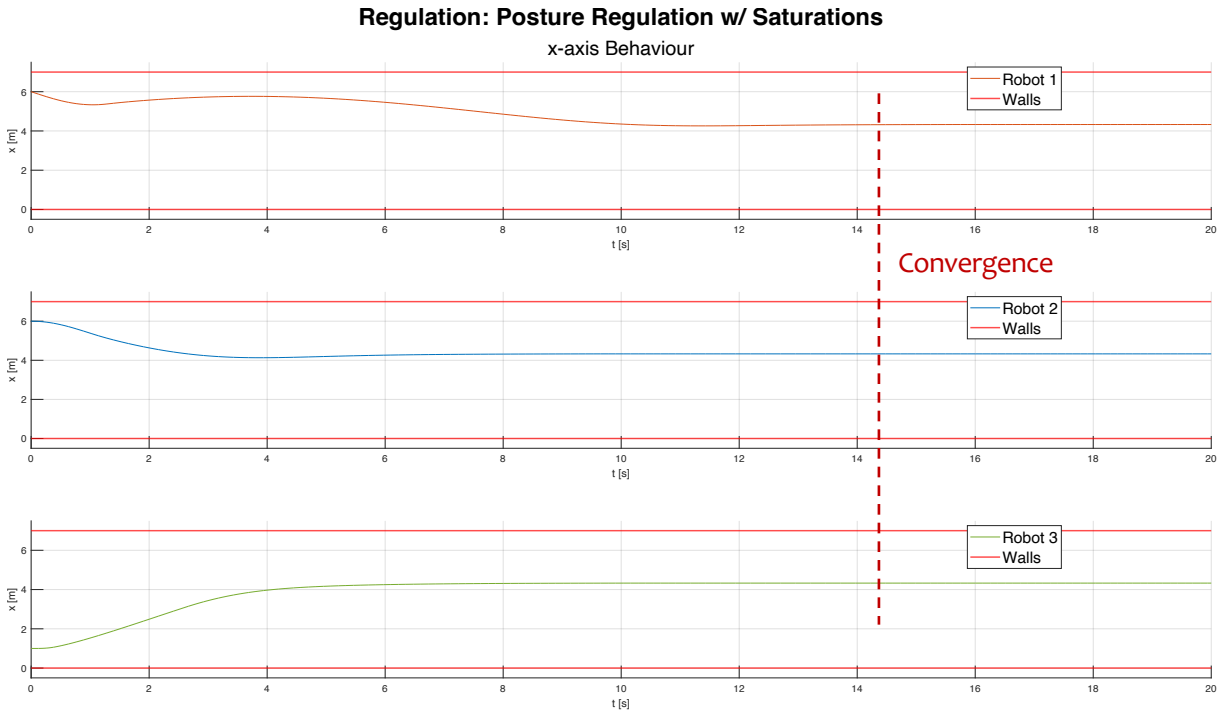

Regulation: Posture Regulation w/o Saturations
Inputs Behaviour
Robot 1
Robot 2
Robot 3
v [m/s]
t [s]

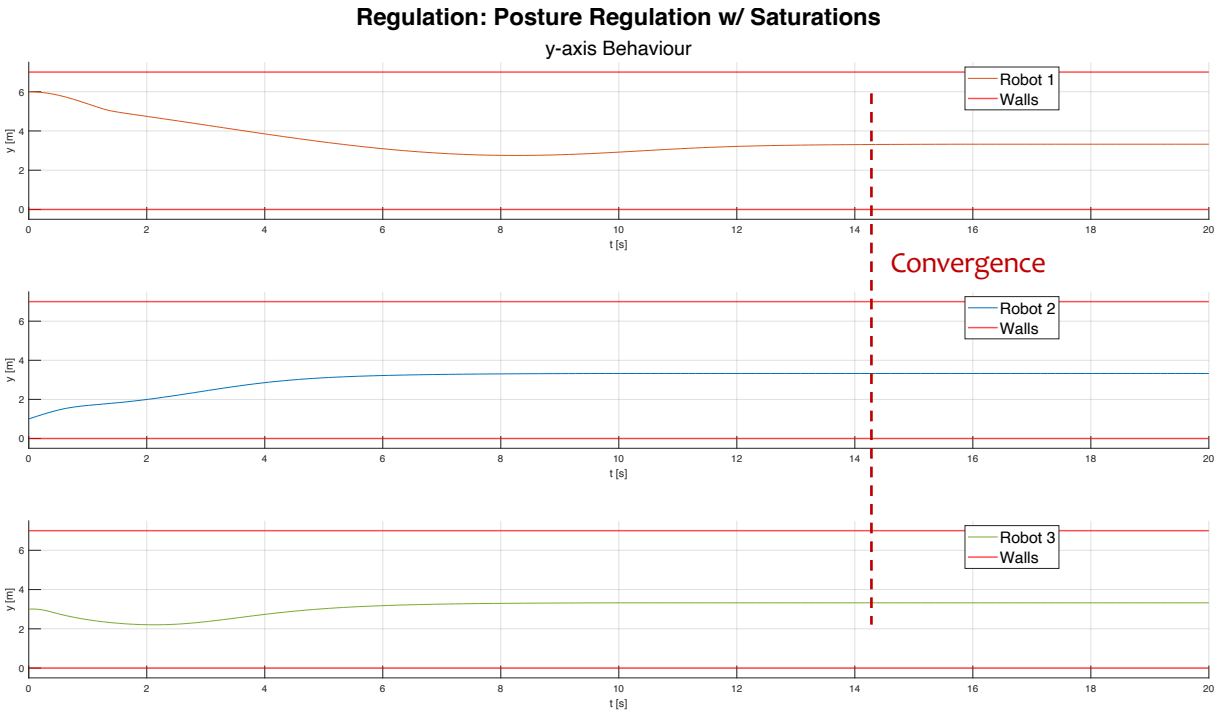

Robot 1
Robot 2
Robot 3
ω [rad/s]
t [s]


Saturation Effect
Regulation: Posture Regulation w/ Saturations
Inputs Behaviour
Robot 1
Robot 2
Robot 3
Saturation
v [m/s]
t [s]


Saturation Effect
Robot 1
Robot 2
Robot 3
Saturation
ω [rad/s]
t [s]

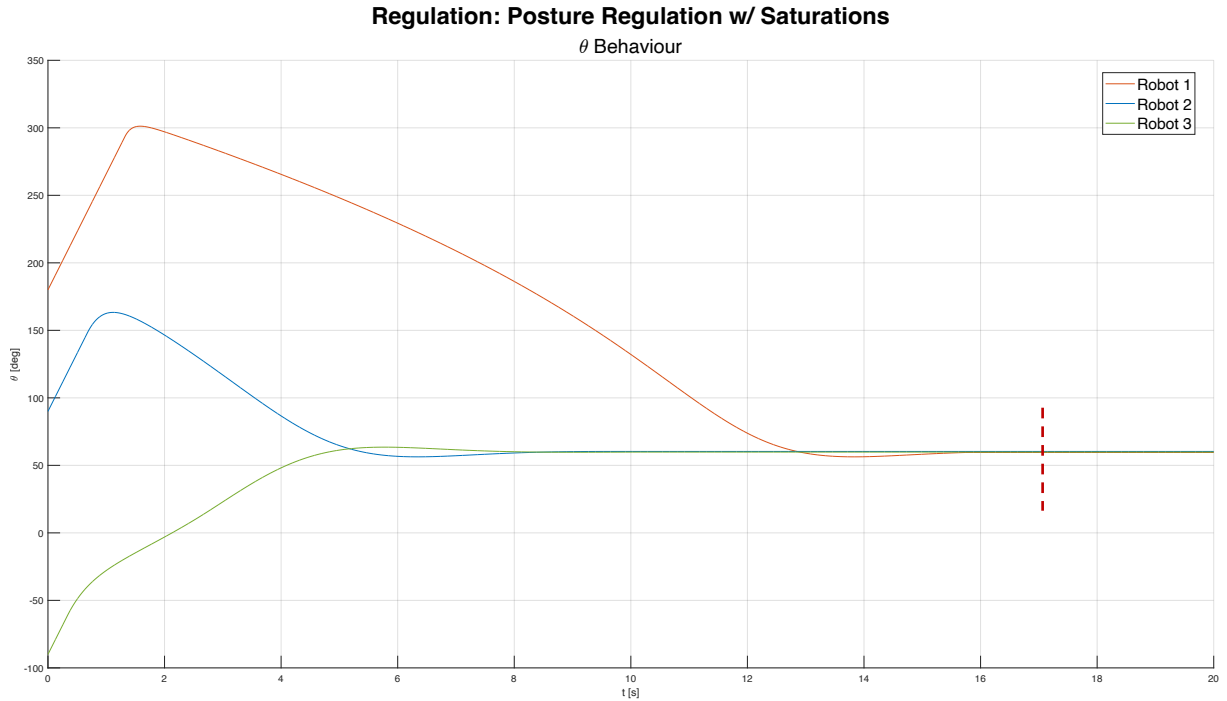
[3.2] Results


Regulation: Posture Regulation w/ Saturations
x-axis Behaviour
Robot 1
Walls
Robot 2
Walls
Robot 3
Walls
x [m]
t [s]
Convergence


Regulation: Posture Regulation w/ Saturations
y-axis Behaviour
Robot 1
Walls
Robot 2
Walls
Robot 3
Walls
y [m]
t [s]
Convergence

[3.2] Results

Convergence occurs at the same value of θ, but takes a longer time wrt to position.



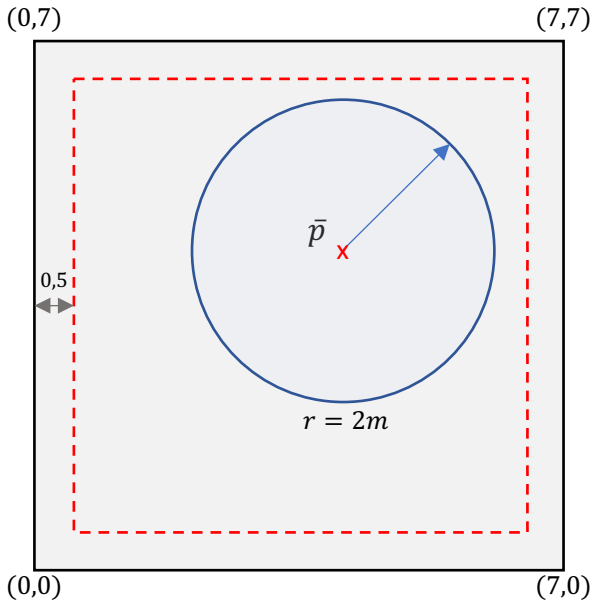
[4] Trajectory tracking

1) Trajectory definition
2) Output Feedback
    1) Change of variables
    2) Further Derivatives
3) State Feedback
    1) Non-Linear Controller
    2) Linearization

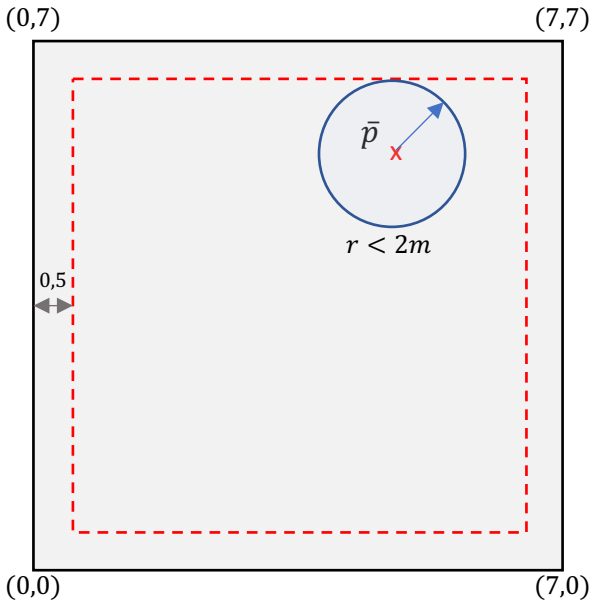
DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

[4.1] Trajectory definition

After reaching the rendezvous point (pose obtained in position regulation task), agents are supposed to follow a circular trajectory, centered in $\bar{p}$. Such trajectory must be far at least $0.5\ m$ from the room borders. Furthermore, such trajectory has $2\ m$ of default radius (a). If $\bar{p}$ is too close to one of the borders, the trajectory is automatically adjusted to be the widest possible (b).

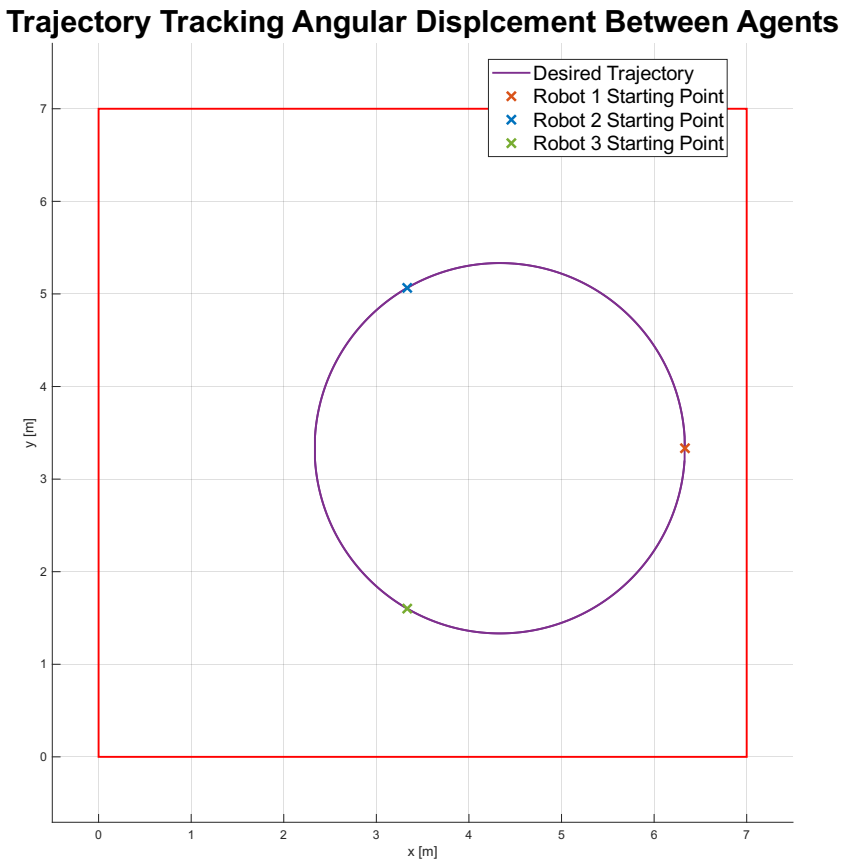


(a) Default case



(b) Adjusted case

[4.1] Trajectory definition



We want to define a circular trajectory (x_d, y_d) of radius r around the rendez-vous point $\bar{p} = (x_r, y_r)$. Such trajectory is defined by

$$\begin{cases} x_d = x_r + r\cos(\omega t + \phi_0) \\ y_d = y_r + r\sin(\omega t + \phi_0) \end{cases}$$

Where ω $(0.25\ [rad/s])$ is an angular velocity and defines how fast we are rotating around the centre, while ϕ is the initial phase. It is worth noticing that 3 different trajectories are generated for the different agents. Furthermore:

1) ω, r are the same for different trajectories;

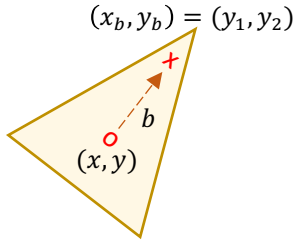
2) To avoid agents to collide on the same point, ϕ_0 assumes different values, in particular $0, {}^{2\pi}/_{3}, {}^{4\pi}/_{3}\ [rad]$ have been used.

3) All the trajectories and their derivatives have been defined online and then inserted in the code by using a *repeating sequence* block.

[4.2] Output Feedback – Change of Var.

Change of coordinates $\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = T(\theta) \begin{bmatrix} v \\ \omega \end{bmatrix}$

where $\mathrm{T}(\theta) = \begin{bmatrix} \cos(\theta) & -b\sin(\theta) \\ \sin(\theta) & b\cos(\theta) \end{bmatrix}, b \neq 0$



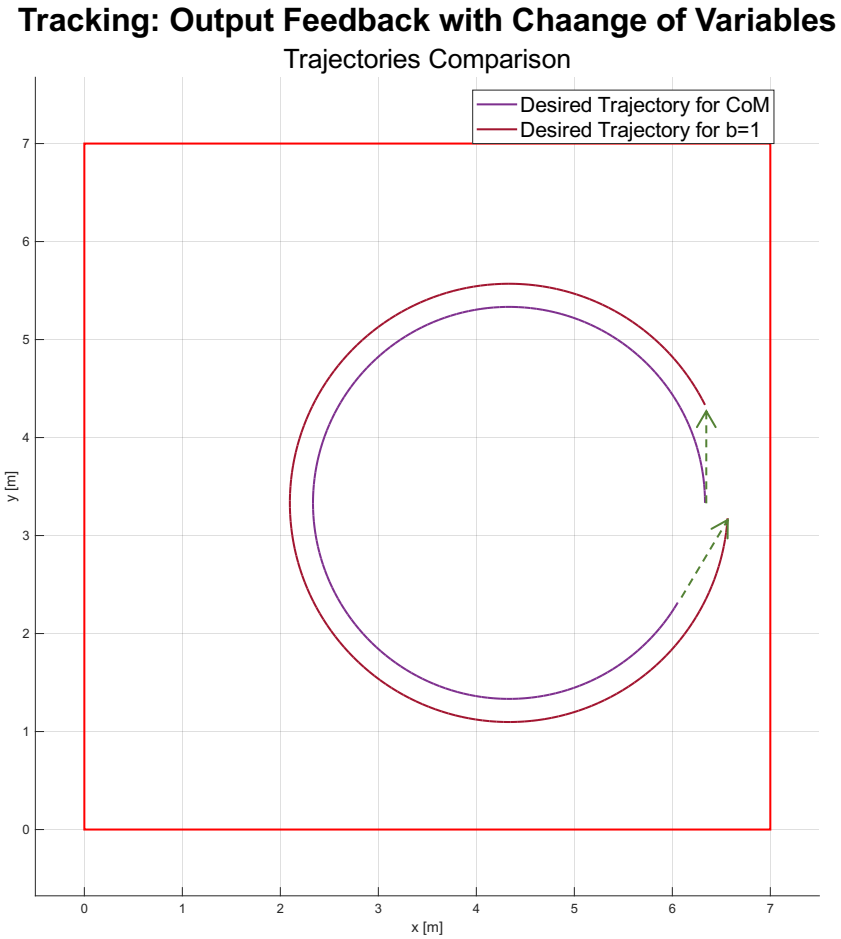
Values read by the system $\begin{cases} \dot{y}_1 = u_1 \\ \dot{y}_2 = u_2 \\ \omega = -\frac{\sin(\theta)}{b} u_1 + \frac{\cos(\theta)}{b} u_2 \end{cases}$

1) Decoupled dynamics in the two componets (y_1, y_2) referring yo the cartesian position
2) The orientation, not controlled, depends clearly on both the cartesian components
3) The smaller the absolute value of b is, the greter numerical problems are.

[4.2] Trajectory definition

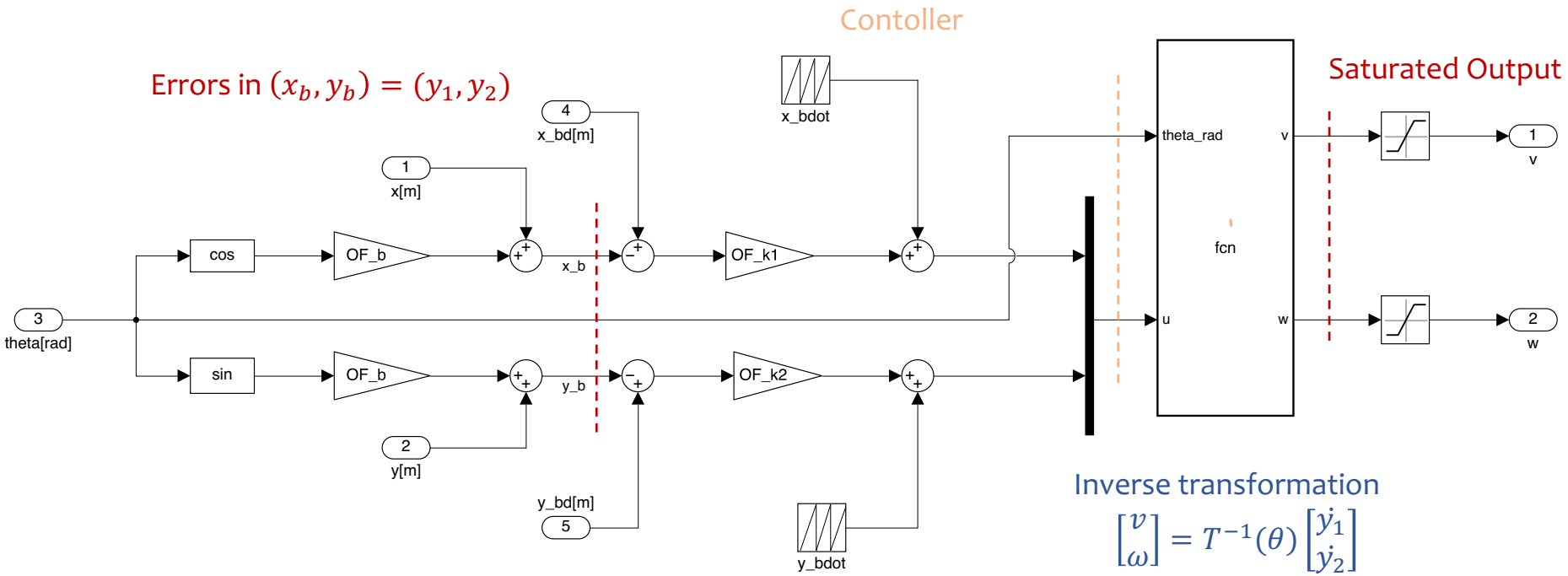
In this case we are not going to track (x_d, y_d) anymore. As can be seen from the previous equations and figure, we are introducing a change of coordinates. This leads to the need of defining another trajectory (x_b, y_b). The necessary equations are

$$\begin{cases} x_{bd} = x_d + b\cos(\theta) \\ y_{bd} = y_d + b\sin(\theta) \end{cases}$$



[4.2] Control Scheme

$$\text{Controller} \begin{cases} u_1 = k_1(y_{1d} - y_1) + \dot{y_{1d}} \\ u_2 = k_2(y_{2d} - y_2) + \dot{y_{2d}} \\ k_1, k_2 > 0 \end{cases} \Longrightarrow \begin{cases} k_1 = 0.5 \\ k_2 = 0.5 \end{cases}$$

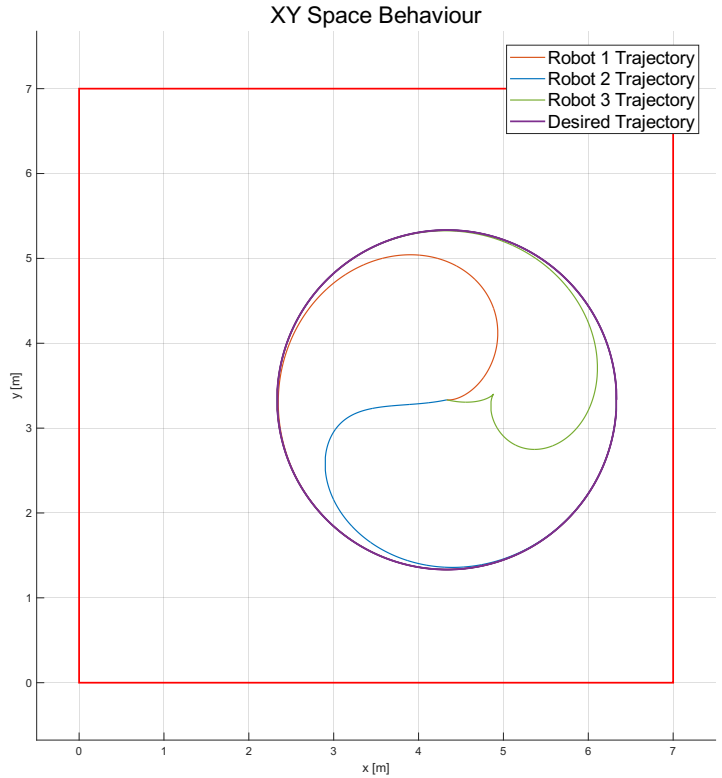


Inverse transformation

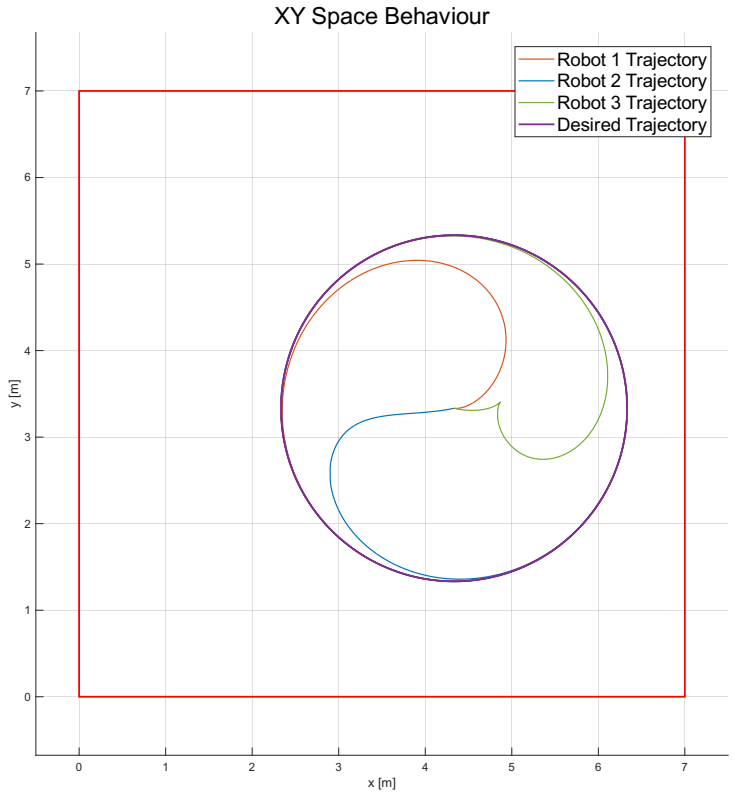
$$\begin{bmatrix} v \\ \omega \end{bmatrix} = T^{-1}(\theta) \begin{bmatrix} \dot{y_1} \\ \dot{y_2} \end{bmatrix}$$

[4.2] Results

Tracking: Output Feedback with Change of Variables w/o Saturations

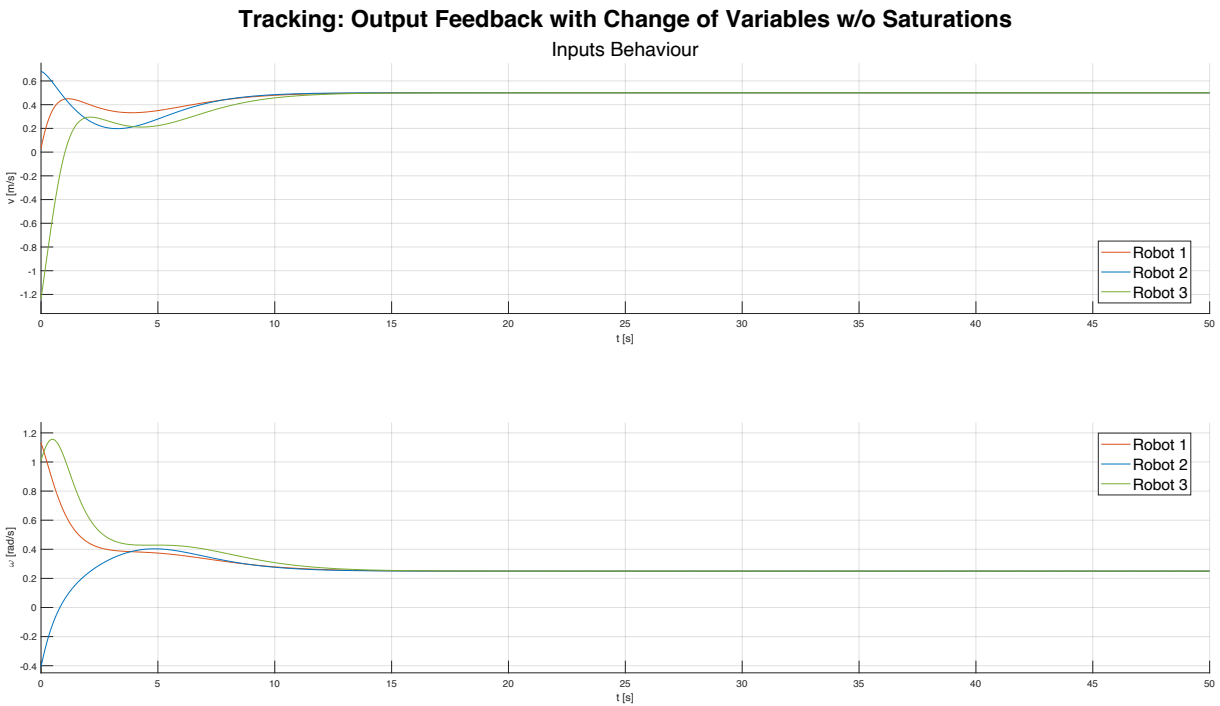


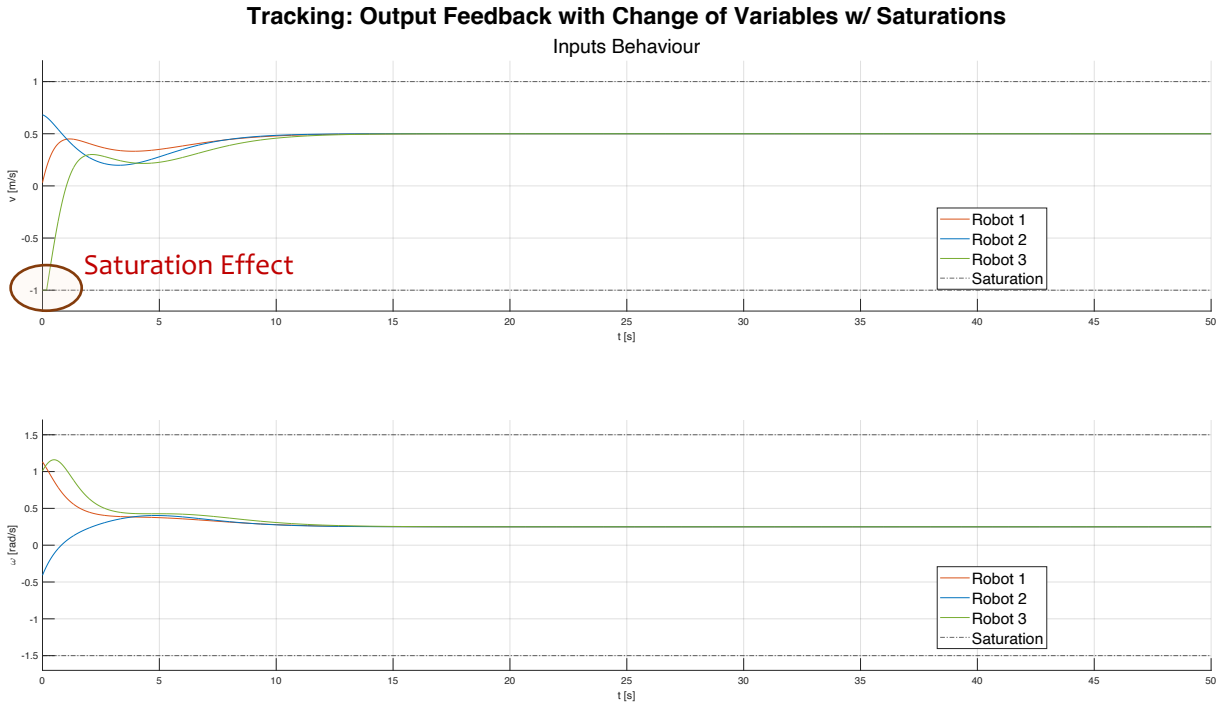
Tracking: Output Feedback with Change of Variables w/ Saturations



[4.2] Results

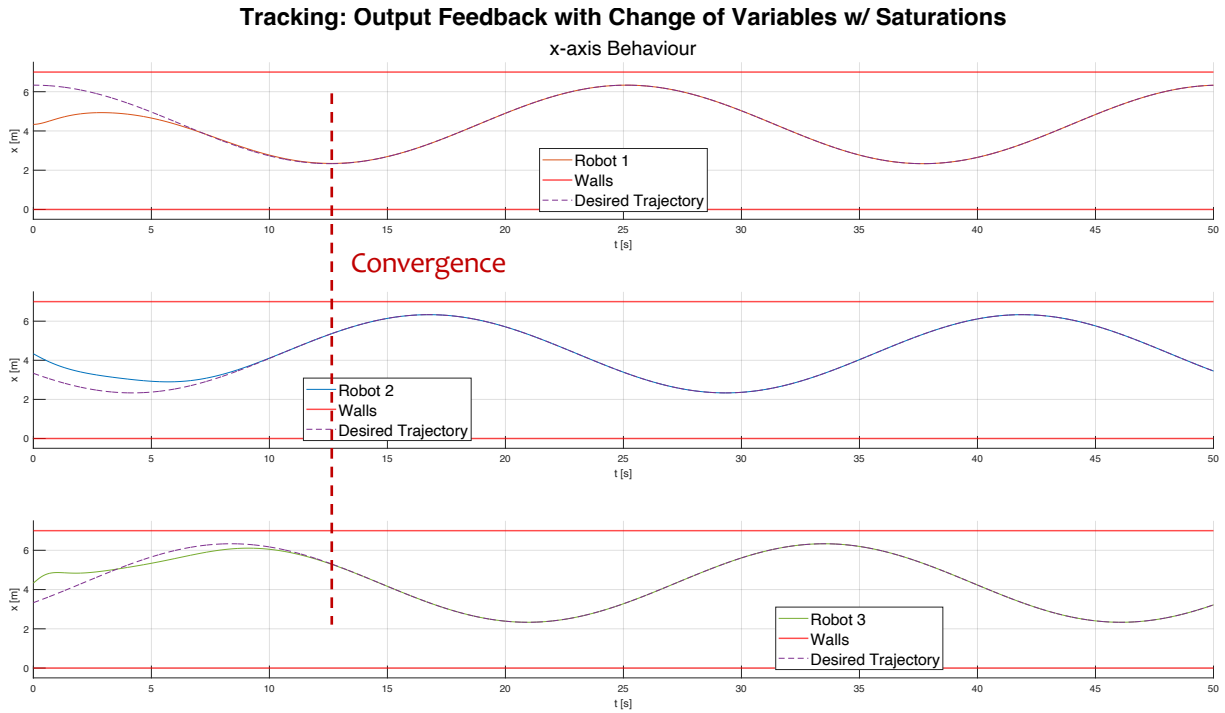
Small saturation, hence quasi-equal trajectories.

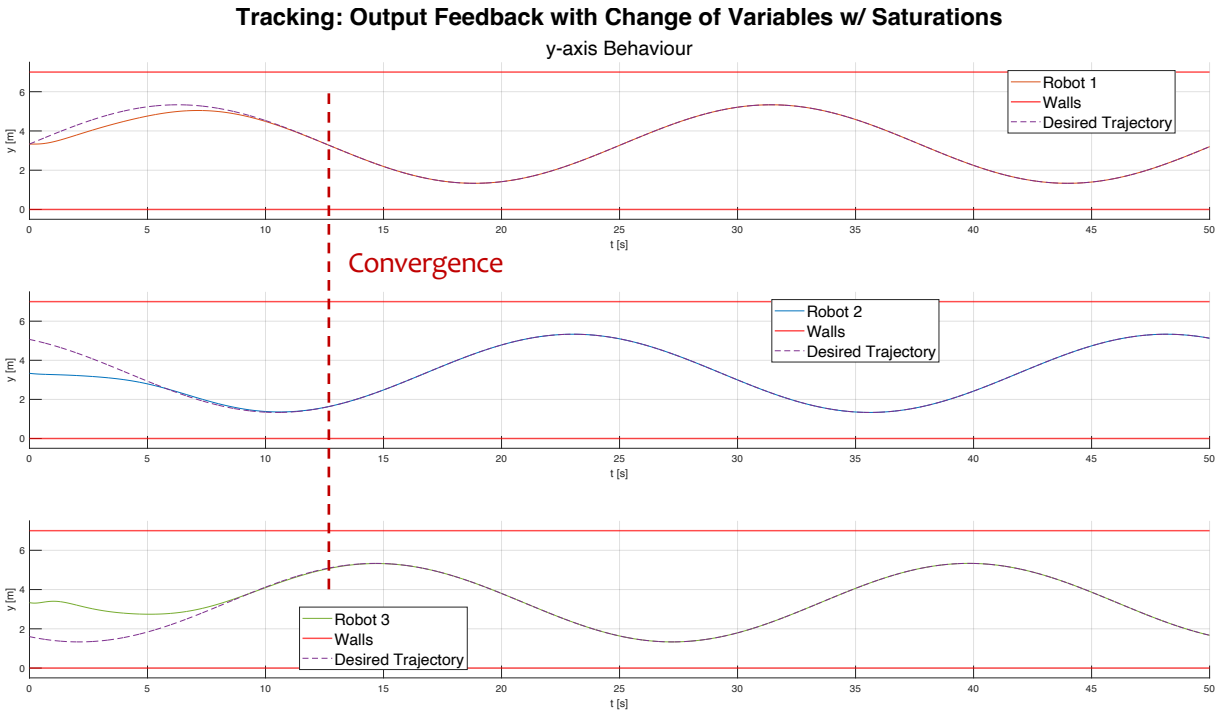




[4.2] Results

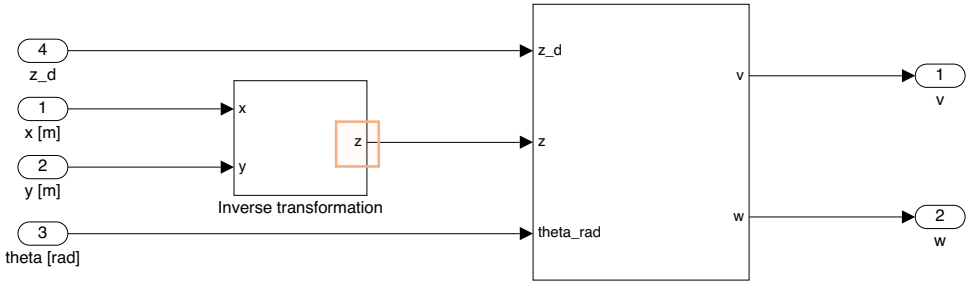
Convergence in approximately 13 seconds.





[4.3] Output Feedback – Further. Der.

Augmented state (takes velocities into account) $z = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}$



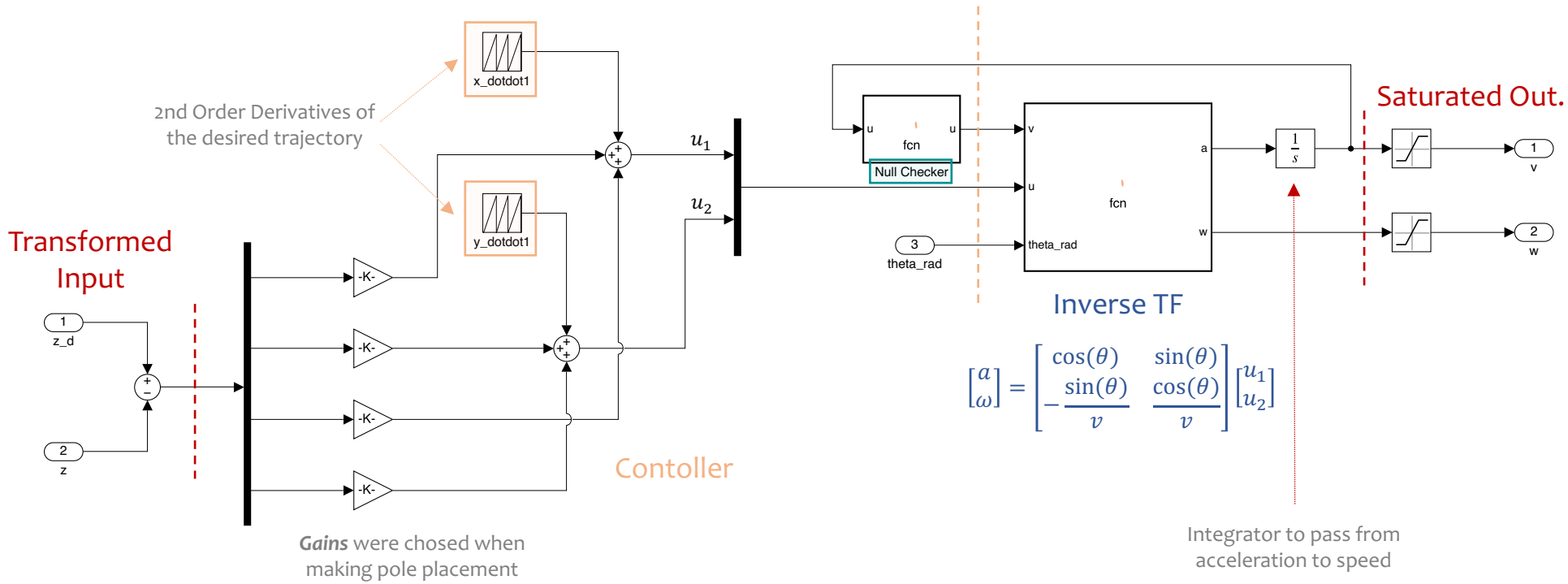
1) This controller works in *acceleration* and not in velocity!
2) Velocity is part of the dynamic (may generate singularities)
3) Butterworth Filter is used to compute derivatives
4) When generating the augmented state, we try two alternatives:

   first order *derivatives* may be *saturated* or not.

[4.3] Output Feedback – Further Der.

Controller $\begin{cases} u_1 = k_1 e_1 + k_3 e_3 + \ddot{x}_d \\ u_2 = k_2 e_2 + k_4 e_4 + \ddot{y}_d \end{cases} \Rightarrow \begin{cases} k_1 = \frac{15}{16}, k_2 = \frac{1}{2} \\ k_3 = 2, k_4 = \frac{3}{2} \end{cases} \Rightarrow$ Gains chosen to place poles in $\{-0.5, -0.75, -1, -1.25\}$

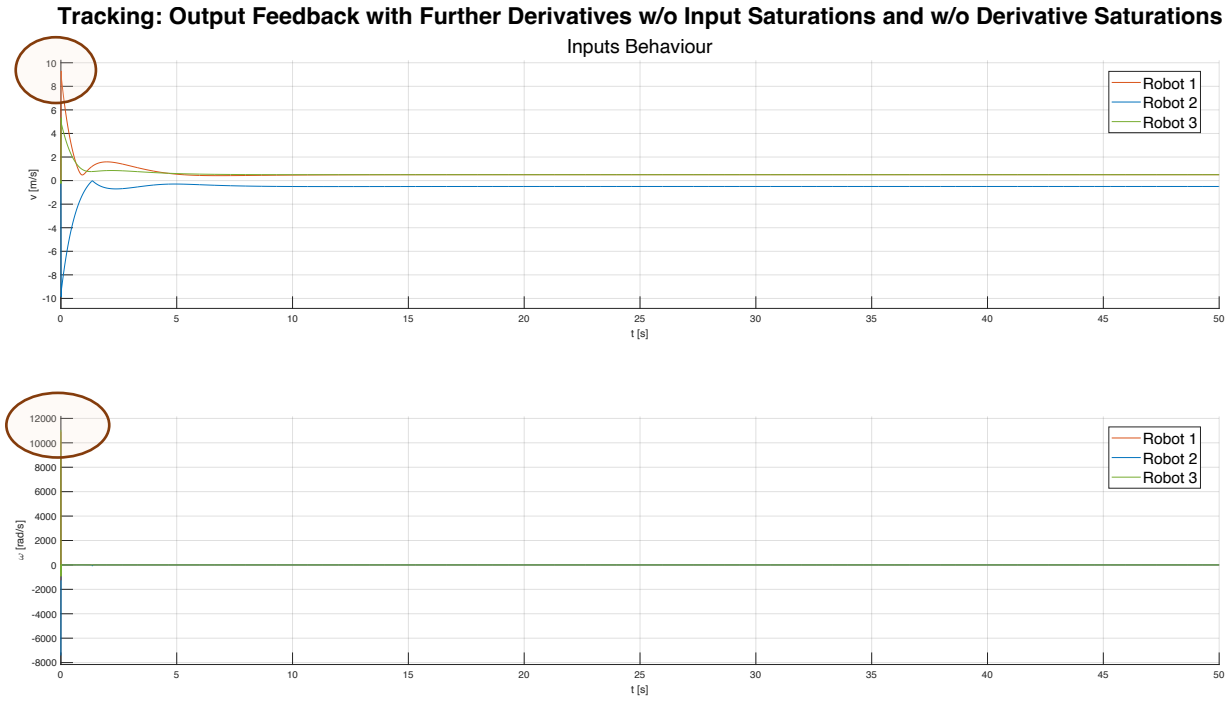
Null Checker (speed singularities) $f(x) = \begin{cases} x, & |x| \geq 10^{-4} \\ 10^{-4}, & otherwise \end{cases}$

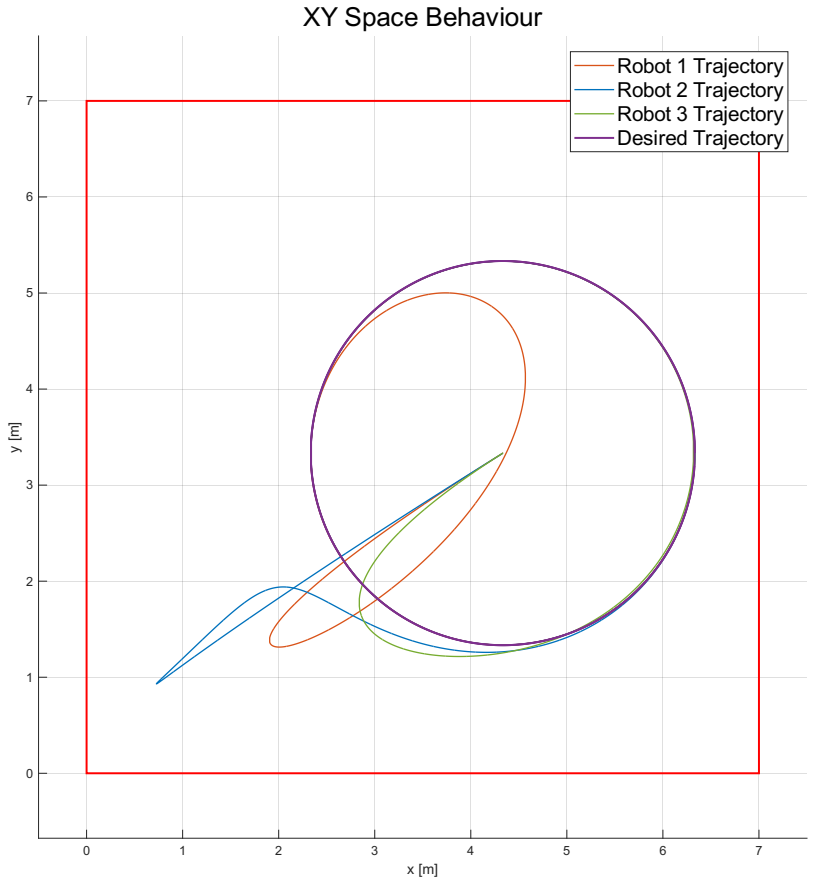


Gains were chosed when making pole placement

Integrator to pass from acceleration to speed

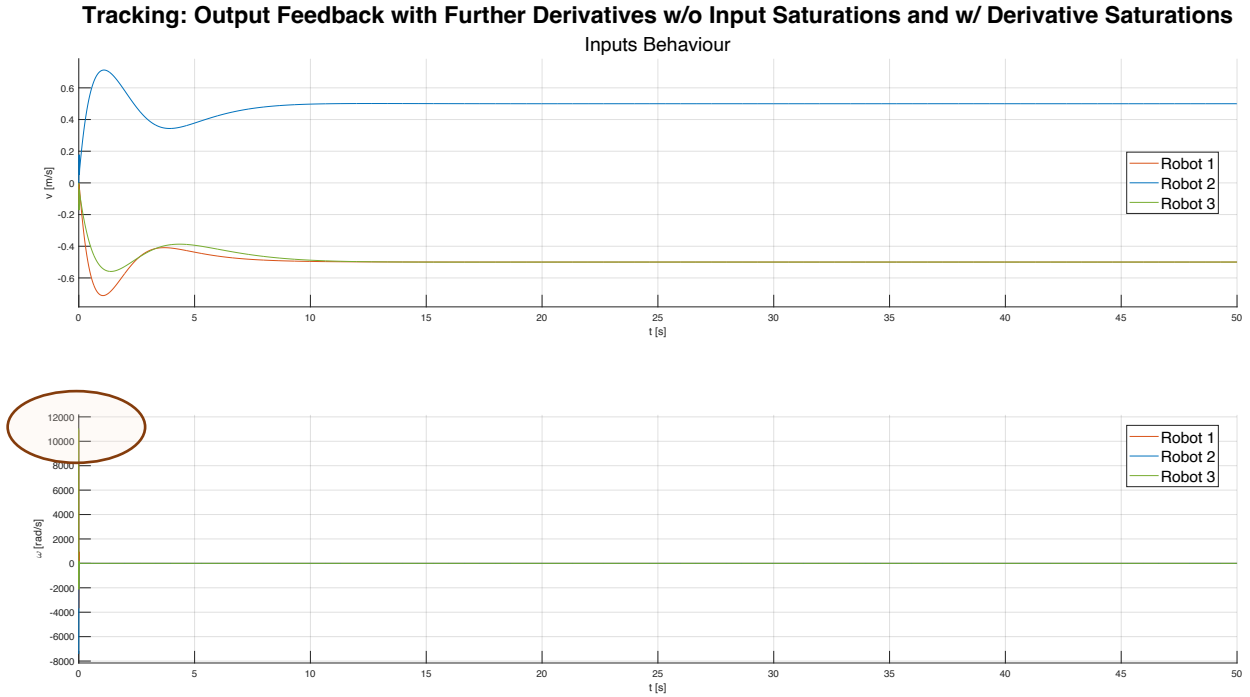
[4.3] Results – No Saturations

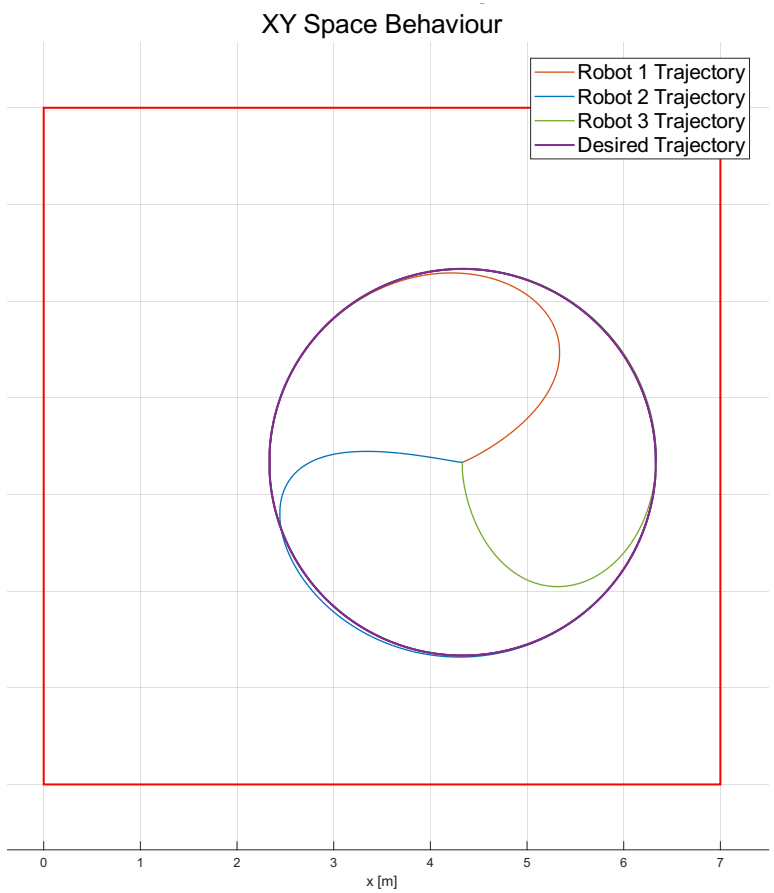




Both linear and angular velocities reach not physically realizable values.

[4.3] Results – Deriv. Saturated

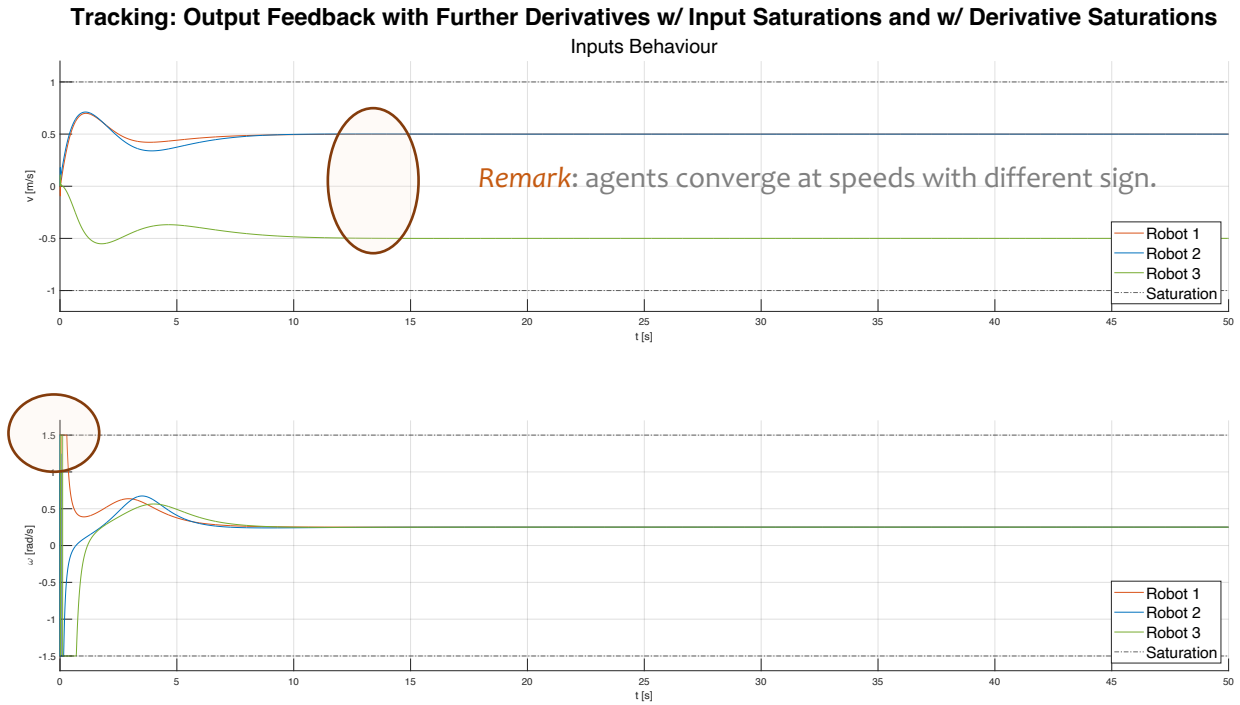




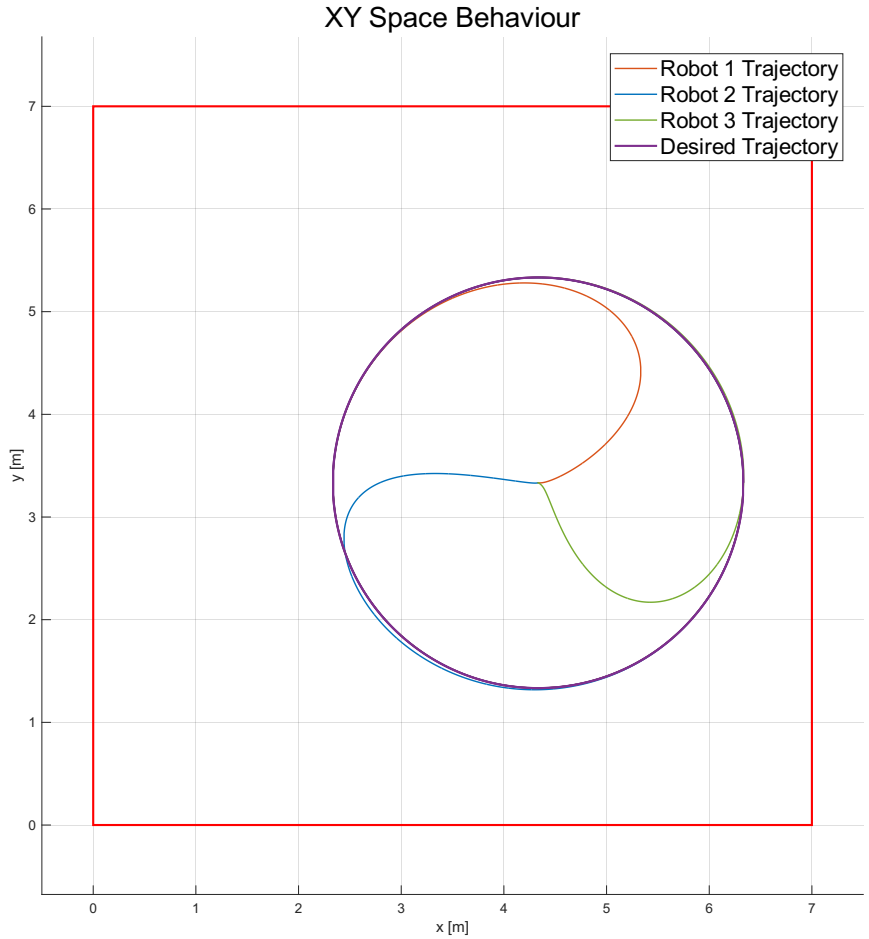
Adding a saturation on the derivatives $\left(d_{sat} = 1 \left[\frac{m}{s}\right]\right)$ only the angular velocity goes over the spec limit.

This saturation was added because at the controller activation the agent start moving, hence its velocity changes as a step.

[4.3] Results – Full Saturation



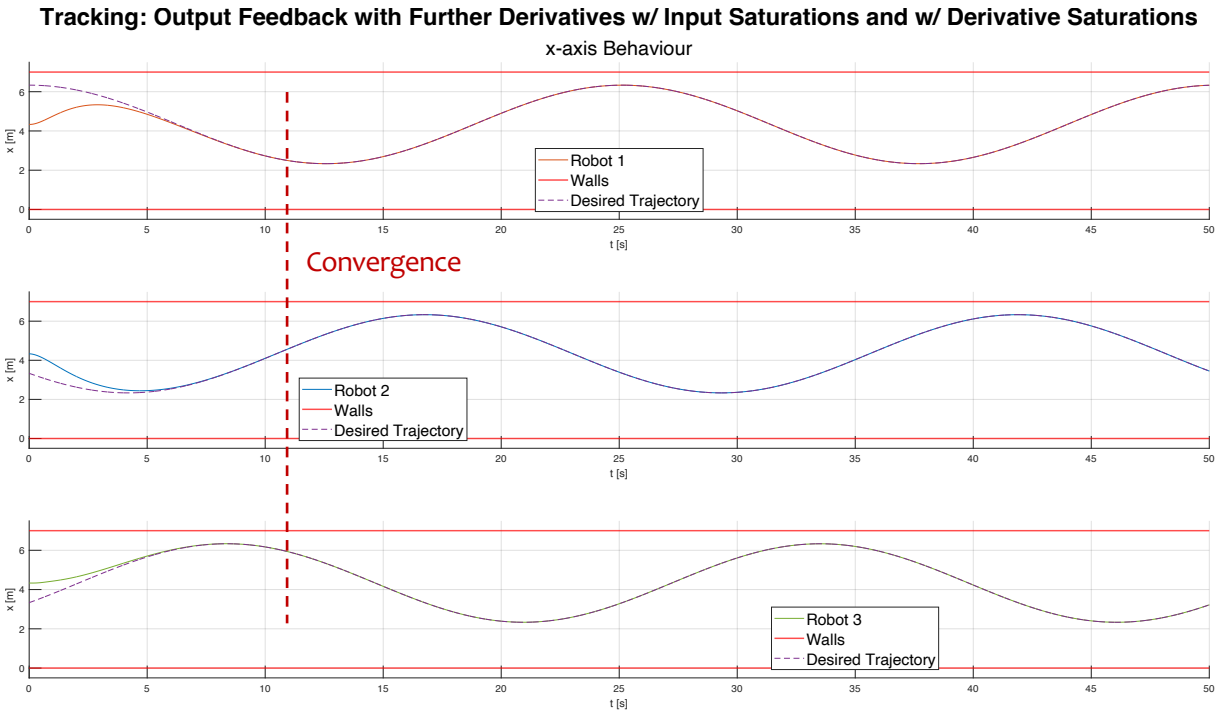
Remark: agents converge at speeds with different sign.

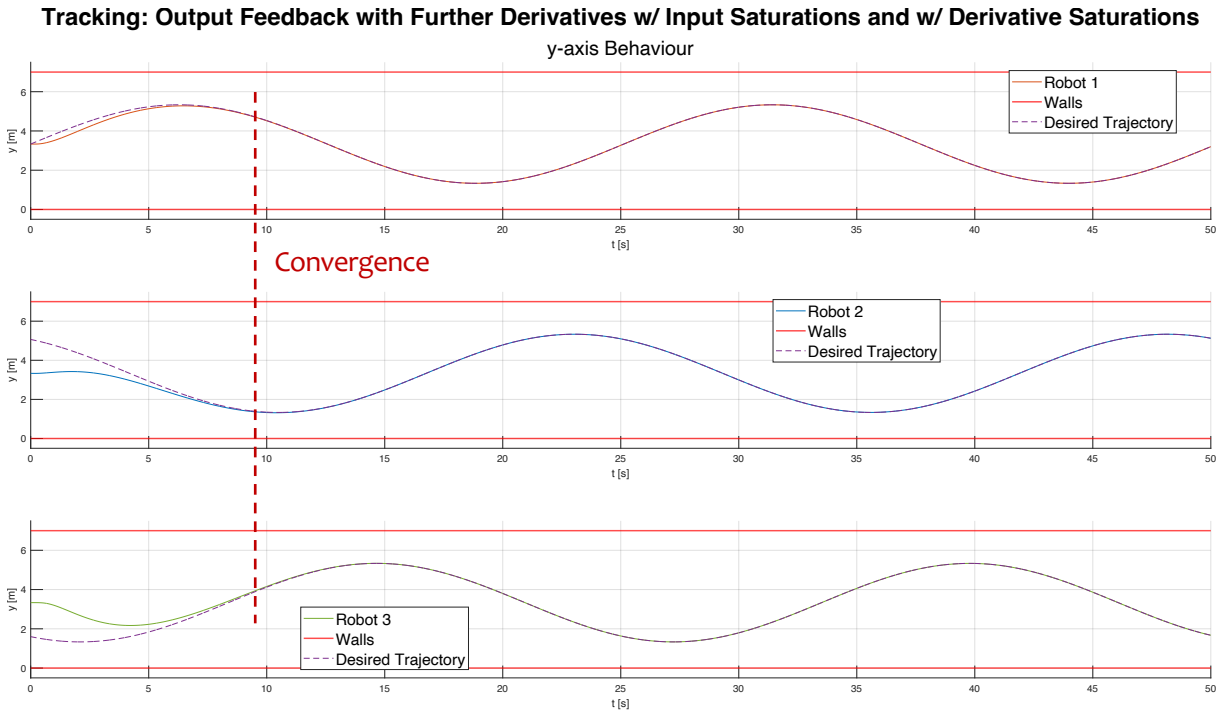


By adding saturations also on the output of the contoller we make the control action physically realizable without changing the final result.

[4.3] Results – Full Saturation

Convergence in approximately 11 seconds.





[4.4] State Feedback

Error dynamic in Body frame | $\dot{e} = \begin{bmatrix} 0 & \omega_d & 0 \\ -\omega_d & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} e + \begin{bmatrix} 0 \\ \sin(e_3) \\ 0 \end{bmatrix} v_d + \begin{bmatrix} 1 & -e_2 \\ 0 & e_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$

[4.5] Non-Linear Controller

[4.6] Linearization around small $e \sim 0$

In both cases we use the following transformations:

World to Body (W2B) transformation $e_B = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} e_W$

Inverse Transformation (Inv) $\begin{cases} v = v_d \cos(e_3) - u_1 \\ \omega = \omega_d - u_2 \end{cases}$

[4.5] State Feedback – Non-linear ctrl.

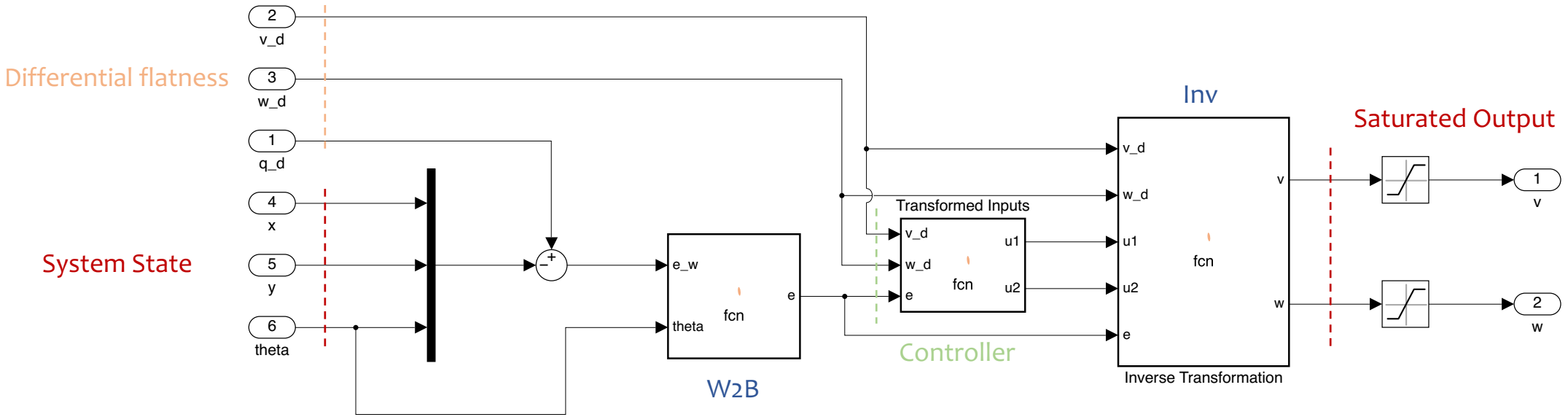
$$\text{Controller} \begin{cases} u_1 = -k_1 e_1 \\ u_2 = -k_2 v_d \frac{\sin(e_3)}{e_3} e_2 - k_3 e_3 \end{cases}$$

$$\text{Gains} \begin{cases} k_1 = k_3 = 2\xi\sqrt{b v_d + \omega_d^2}. \quad \xi \in (0,1) \\ k_2 = b > 0 \end{cases} \Longrightarrow \begin{cases} b = 0.1 \\ \xi = 0.6 \end{cases}$$

Differential flatness

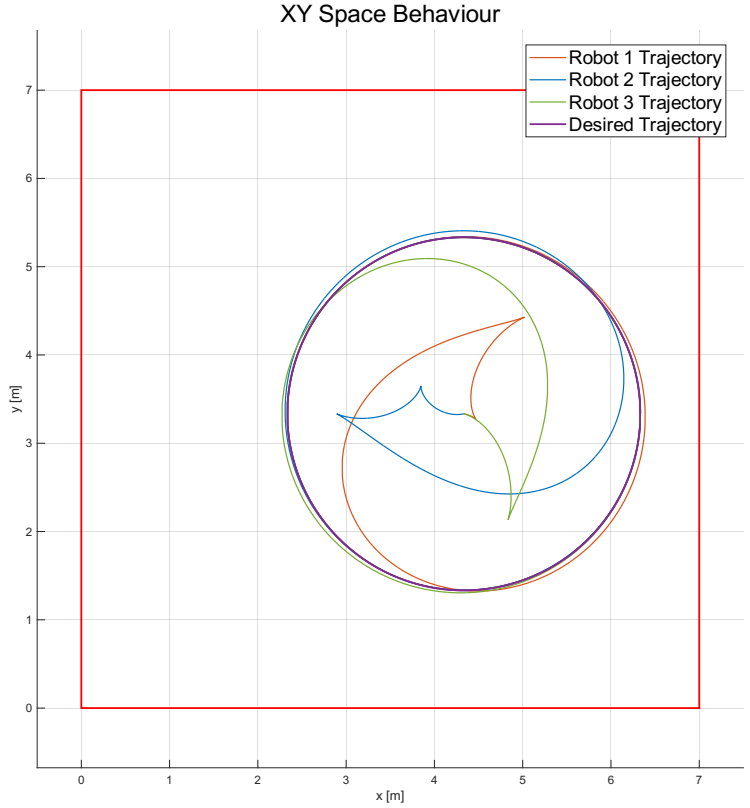
$$\text{State} \begin{cases} x_d \\ y_d \\ \theta_d = atan2(\dot{y}_d, \dot{x}_d) + k\pi \end{cases} \qquad \text{Input} \begin{cases} v_d = \pm\sqrt{\dot{x}_d^2 + \dot{y}_d^2} \\ \omega_d = \frac{\dot{x}_d \ddot{y}_d - \ddot{x}_d \dot{y}_d}{\dot{x}_d^2 + \dot{y}_d^2} \end{cases}$$

[4.5] State Feedback – Non-linear ctrl.

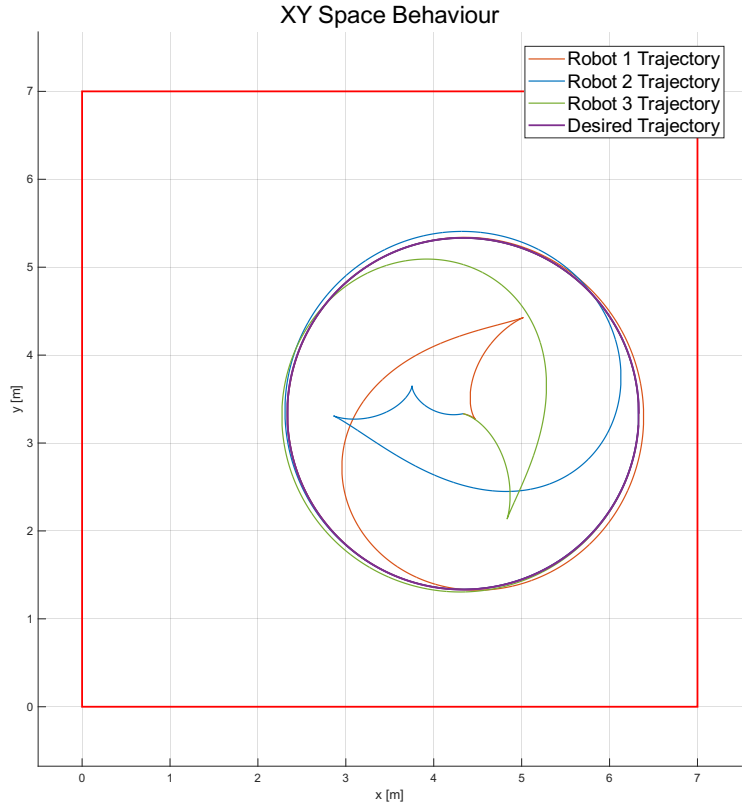


[4.5] Results

Tracking: State Feedback with Exact Linearization w/o Saturations



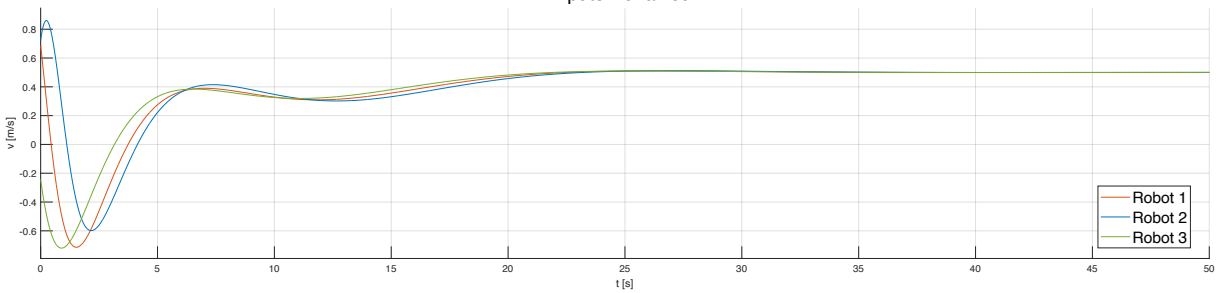
Tracking: State Feedback with Exact Linearization w/ Saturations

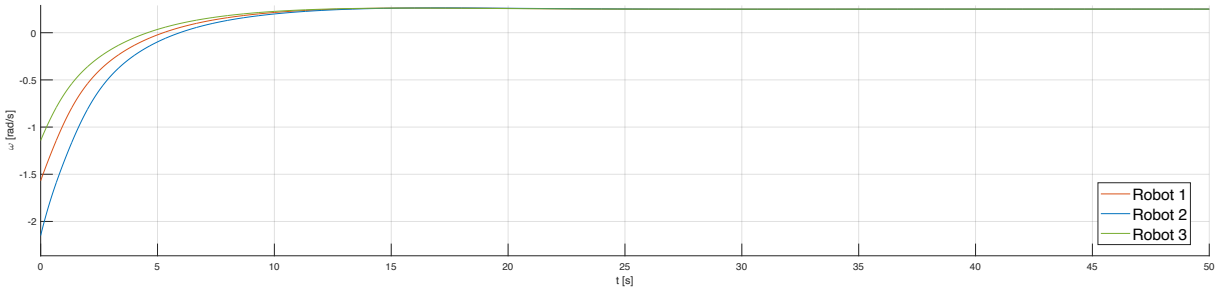


[4.5] Results

Tracking: State Feedback with Exact Linearization w/o Saturations

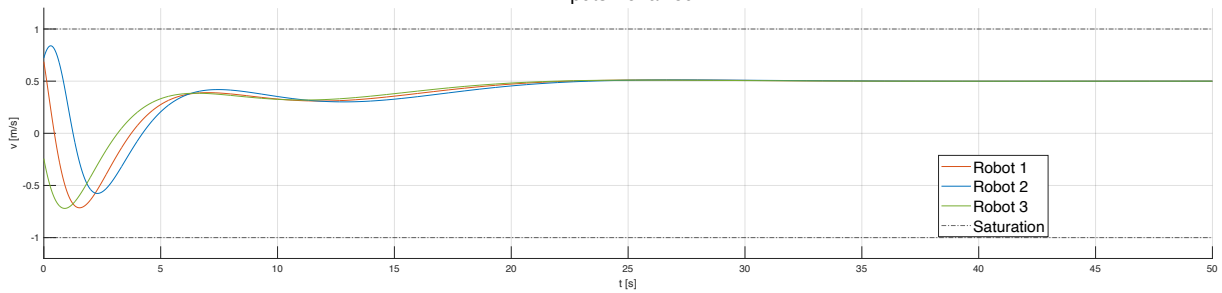
Inputs Behaviour

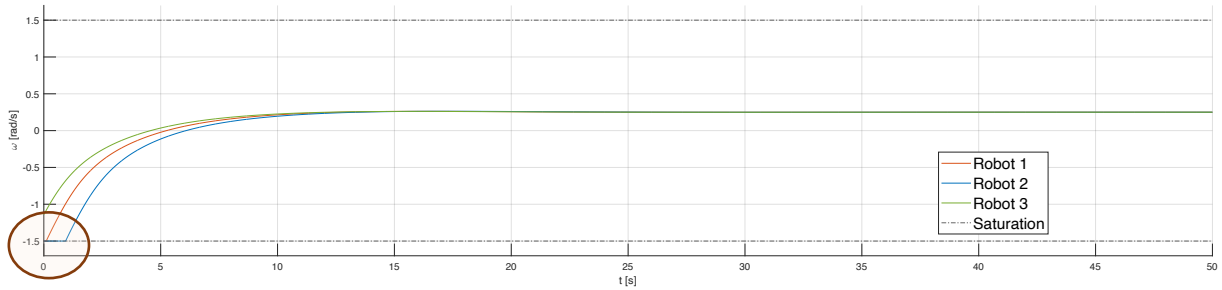




Tracking: State Feedback with Exact Linearization w/ Saturations

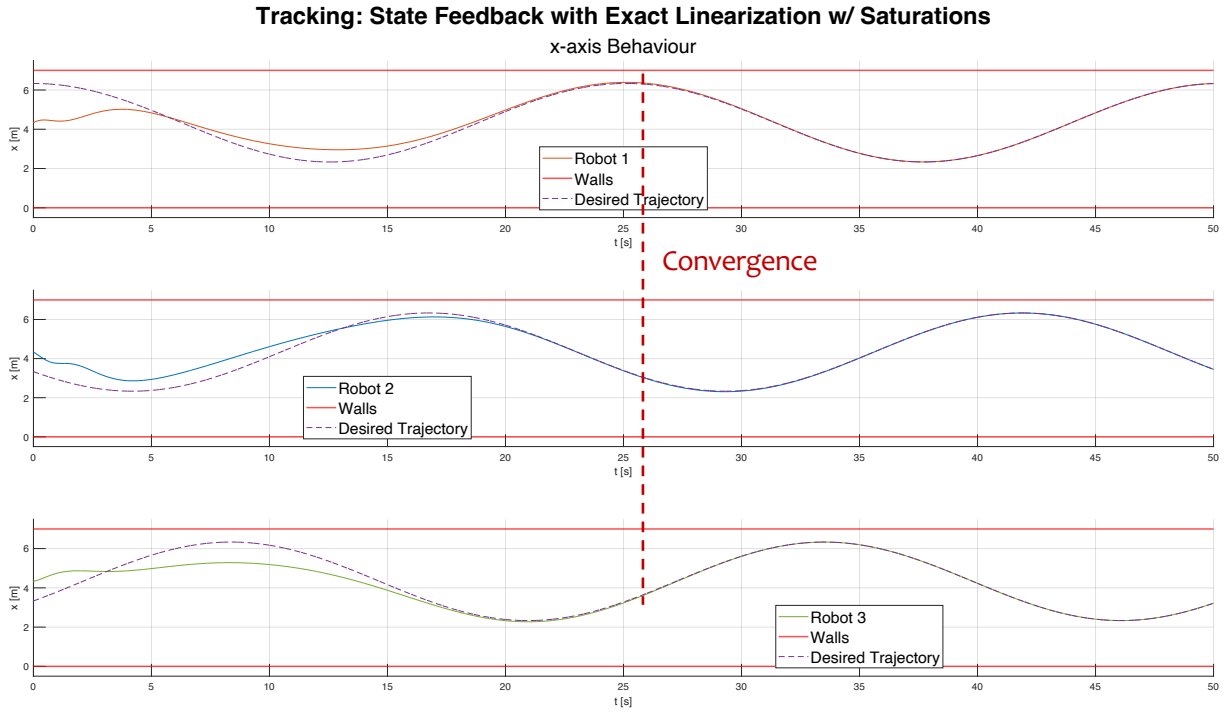
Inputs Behaviour

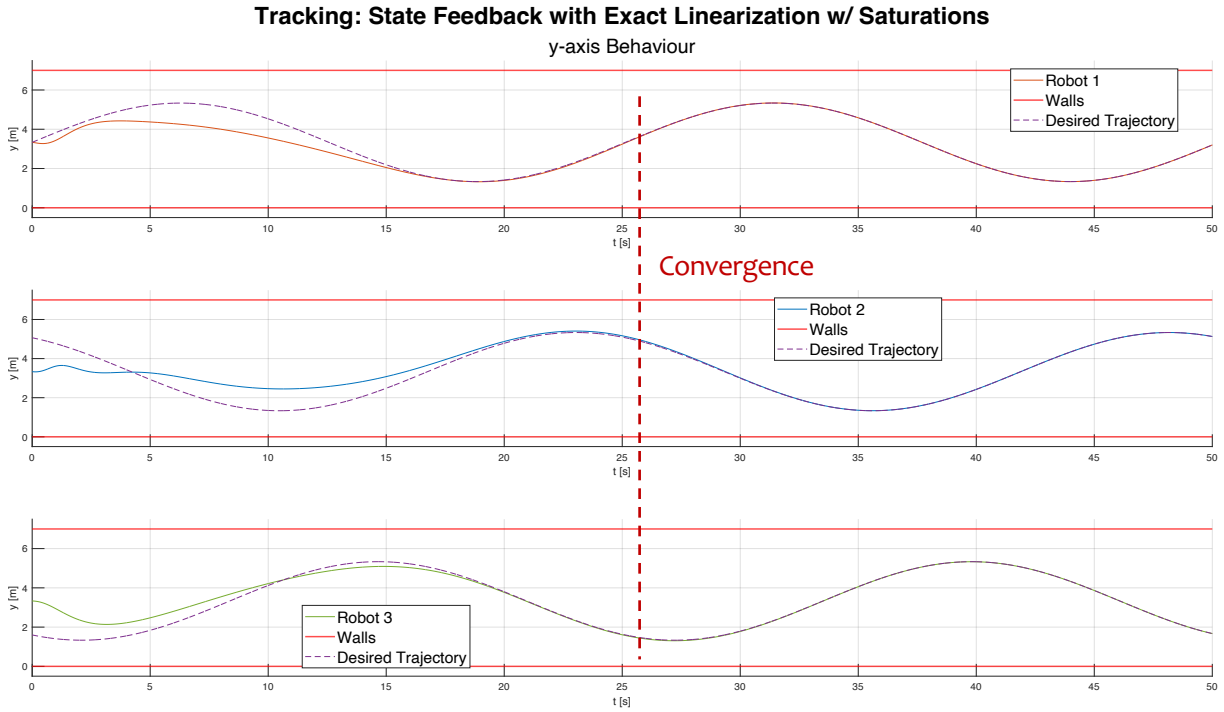




[4.5] Results

Convergence in approximately 26 seconds.





[4.6] State Feedback – Linearization

Tacking again the error dynamic into consideration, it follows that:

$$\dot{e} = \begin{bmatrix} 0 & \omega_d & 0 \\ -\omega_d & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} e + \begin{bmatrix} 0 \\ \sin(e_3) \\ 0 \end{bmatrix} v_d + \begin{bmatrix} 1 & -e_2 \\ 0 & e_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$\dot{e} = \begin{bmatrix} 0 & \omega_d & 0 \\ -\omega_d & 0 & v_d \\ 0 & 0 & 0 \end{bmatrix} e + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

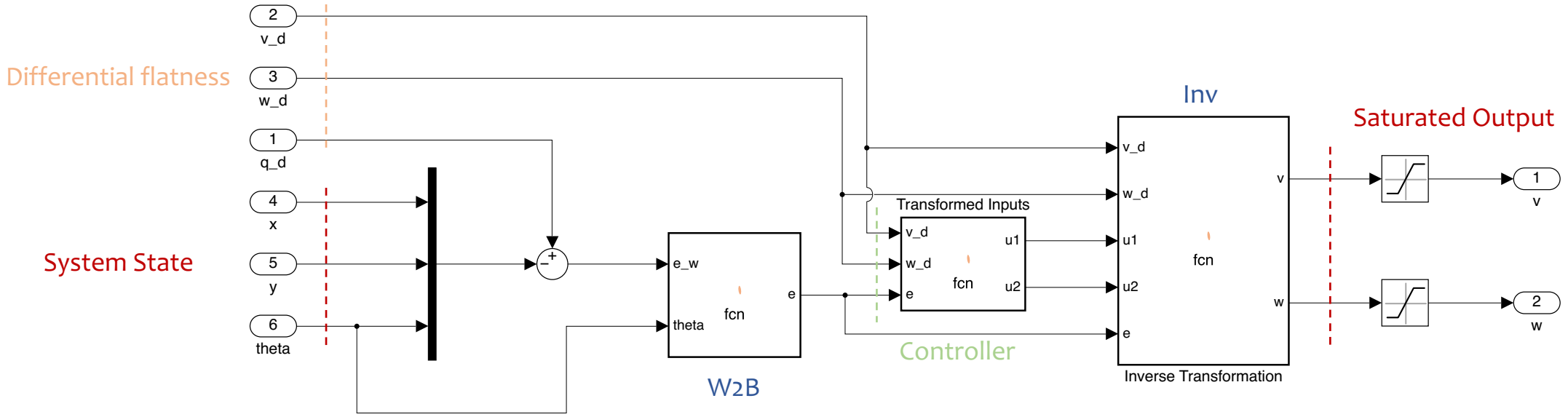
$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} -k_1 & 0 & 0 \\ 0 & -k_2 & -k_3 \end{bmatrix} e$$

$$\dot{e} = \begin{bmatrix} -k_1 & \omega_d & 0 \\ -\omega_d & 0 & v_d \\ 0 & -k_2 & -k_3 \end{bmatrix} e$$

[4.5] State Feedback – Non-linear ctrl.

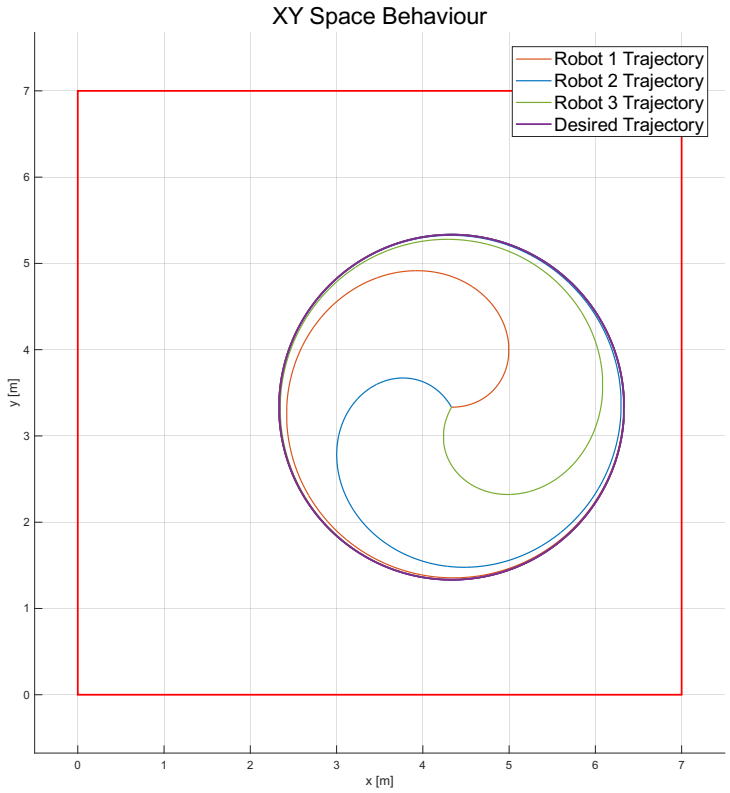
$$\text{Controller} \begin{cases} u_1 = -k_1 e_1 \\ u_2 = -k_2 e_2 - k_3 e_3 \end{cases} \qquad \text{Gains} \begin{cases} k_1 = k_3 = 2\xi a \;\; \xi \in (0,1), a > 0 \\ k_2 = \frac{a^2 - \omega_d^2}{v_d} \end{cases} \Longrightarrow \begin{cases} a = 10 \\ \xi = 0.005 \end{cases}$$

To have a defined input, v_d has to be different from zero. In this case we are following a circular trajecotry, hence this condition is always satified.

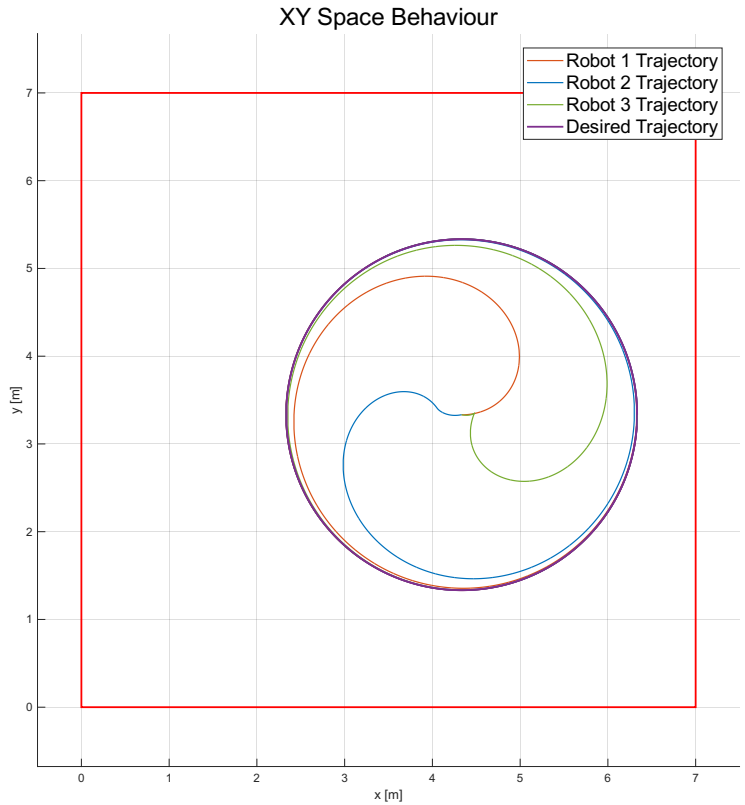


[4.6] Results

Tracking: State Feedback with Approximate Linearization w/o Saturations



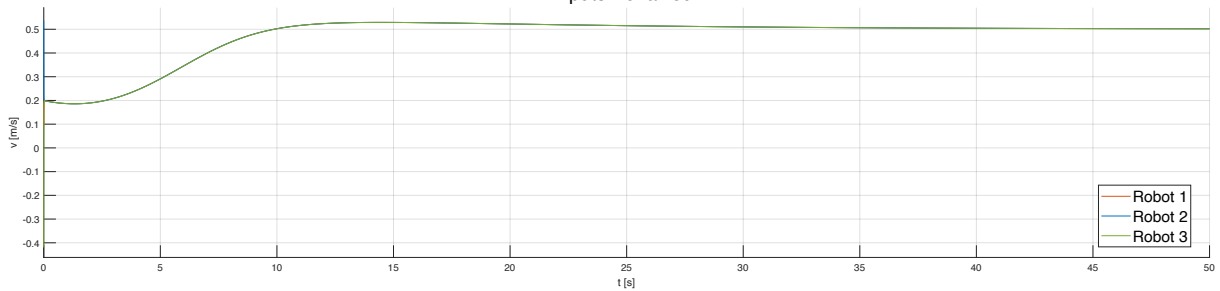
Tracking: State Feedback with Approximate Linearization w/ Saturations



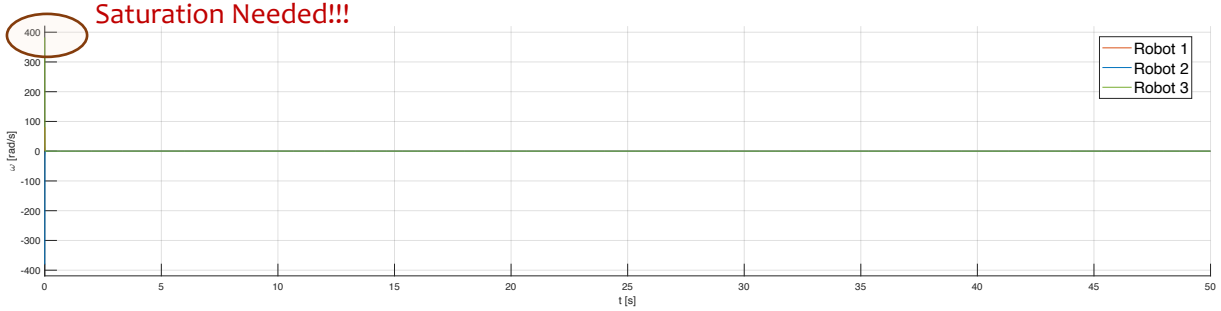
[4.6] Results

Tracking: State Feedback with Approximate Linearization w/o Saturations

Inputs Behaviour

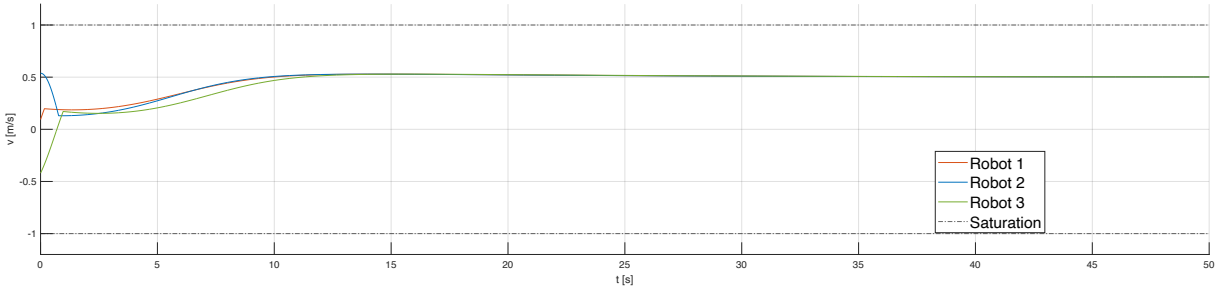


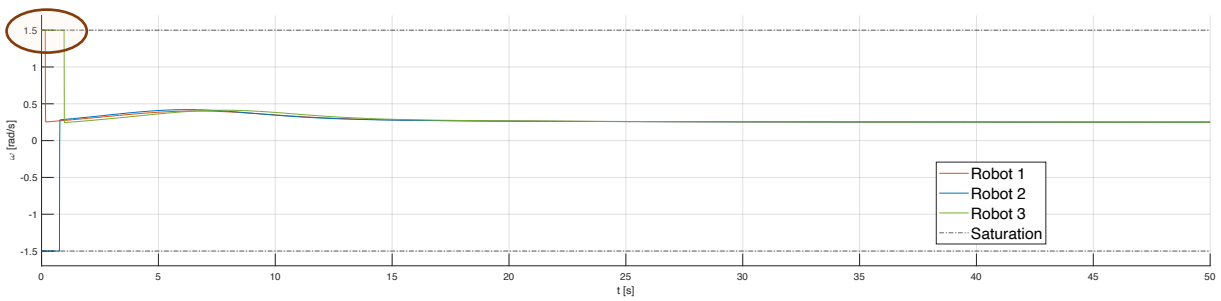
Saturation Needed!!!



Tracking: State Feedback with Approximate Linearization w/ Saturations

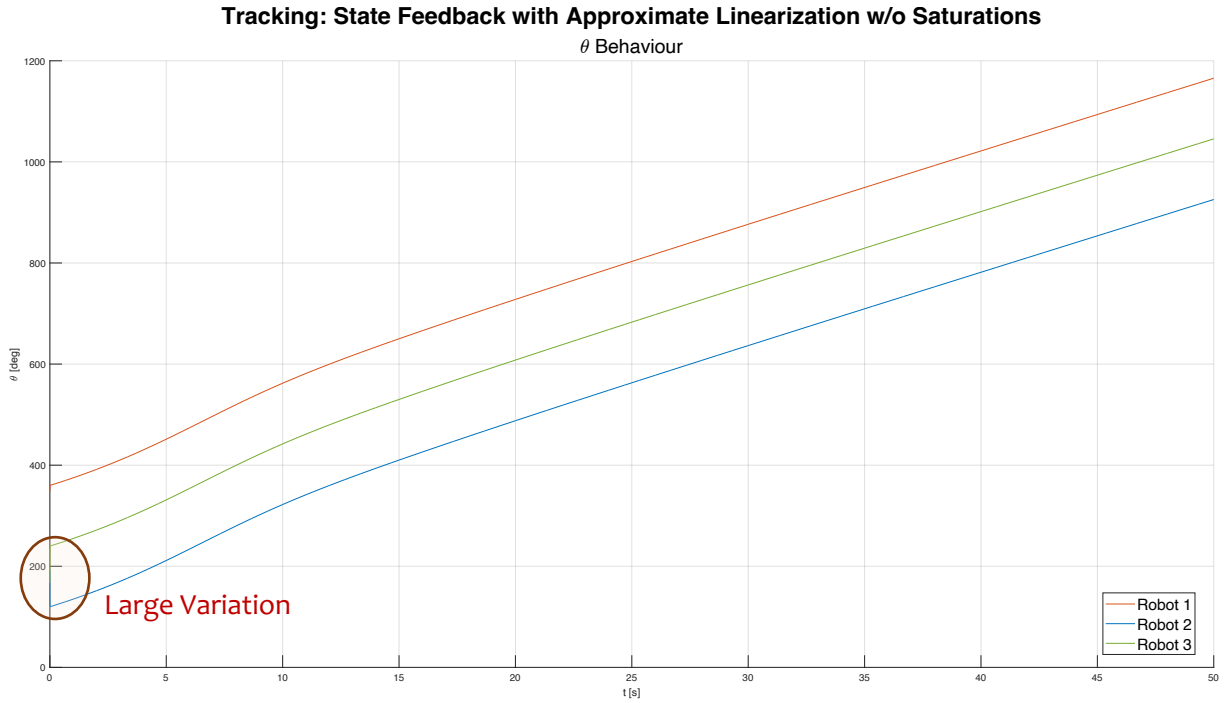
Inputs Behaviour

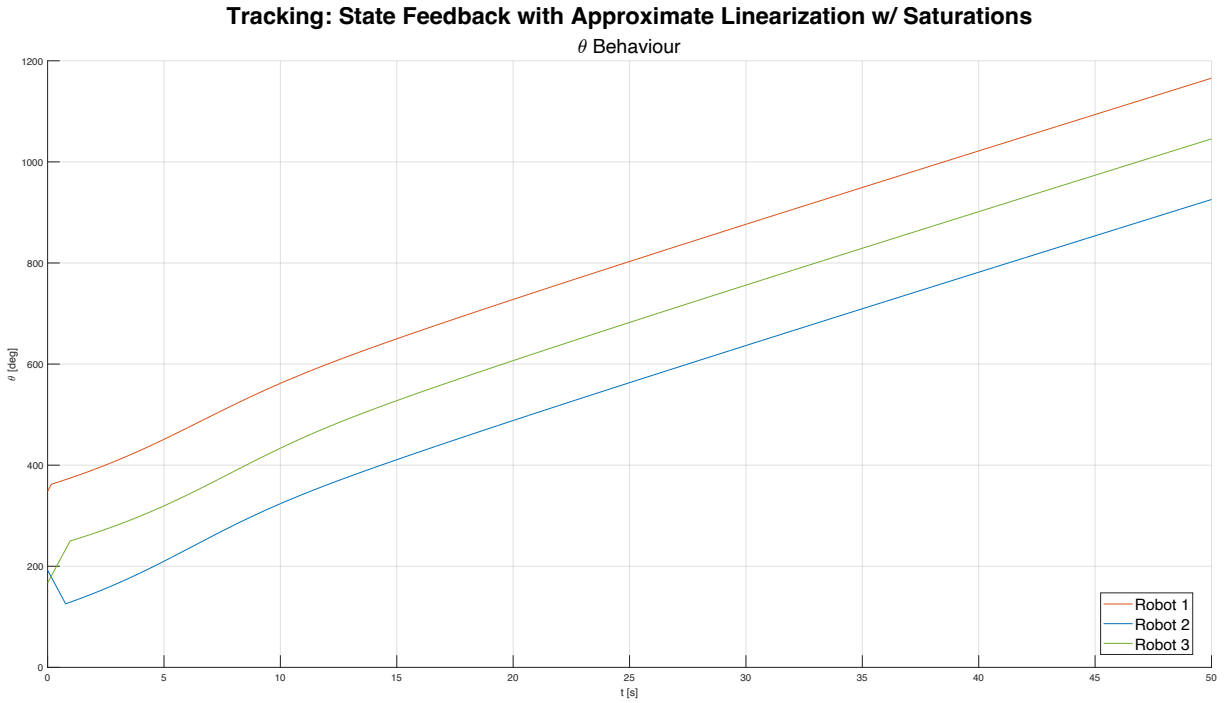




[4.6] Results

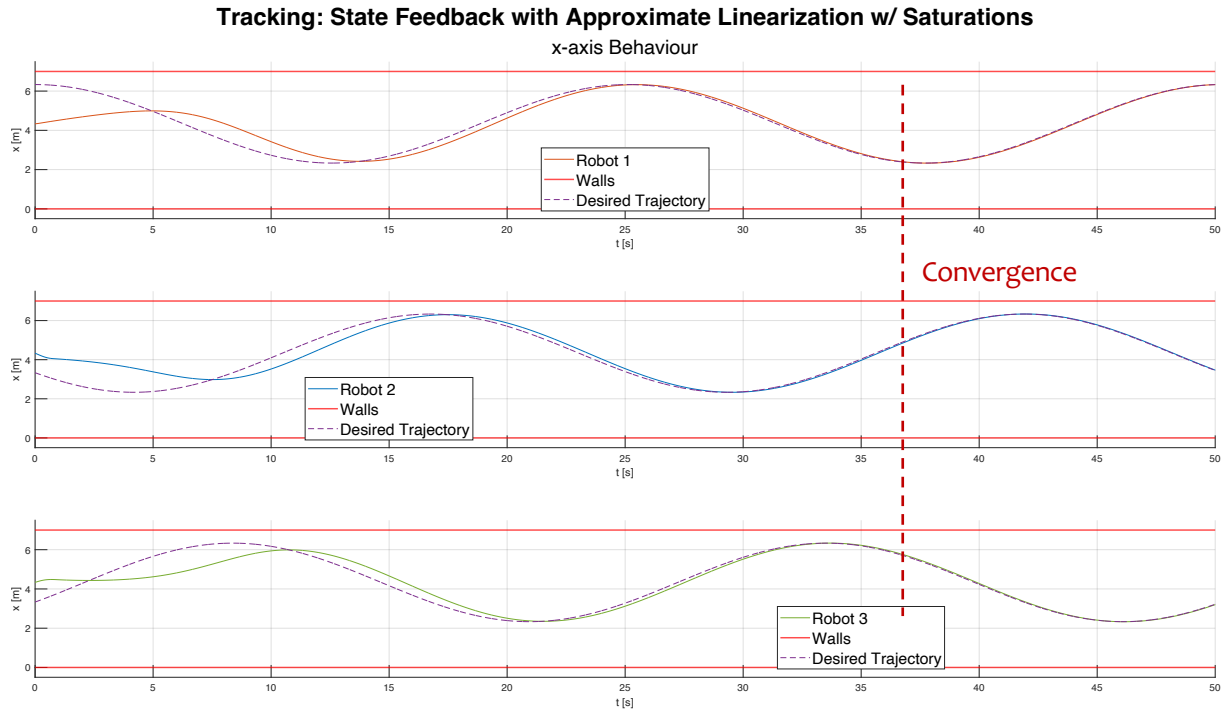
The difference can be observed in a relevant way only when considering orientation, not position.

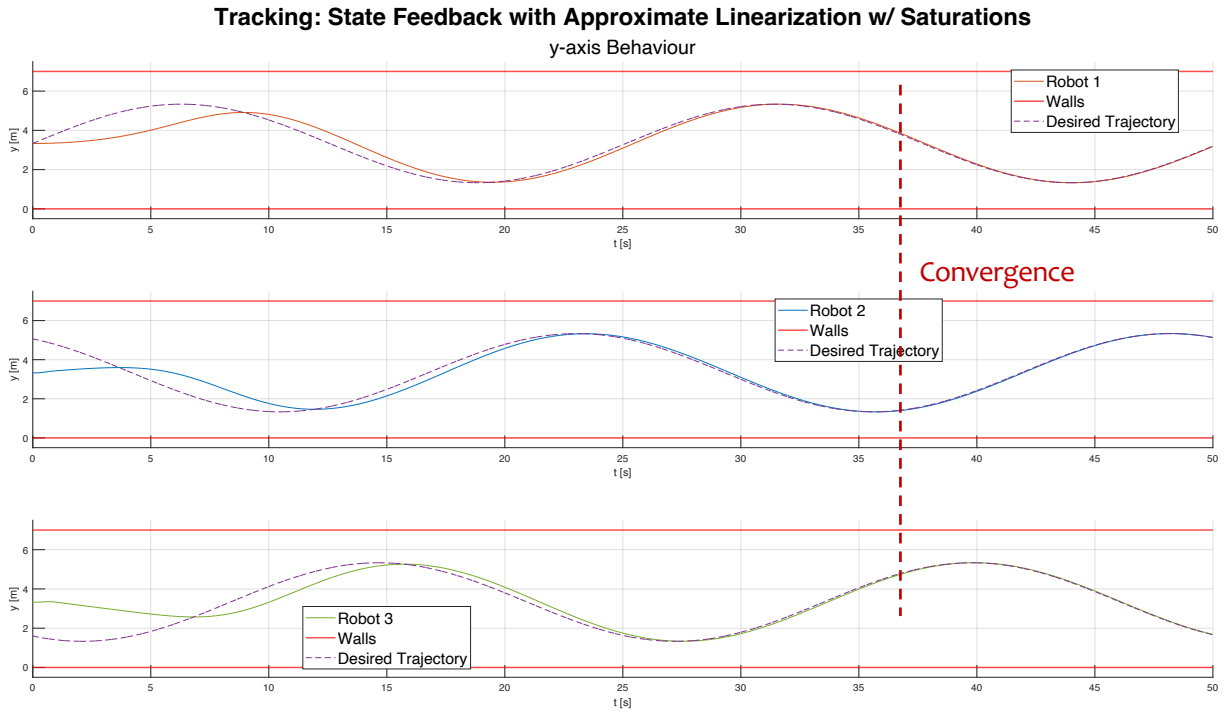




[4.6] Results

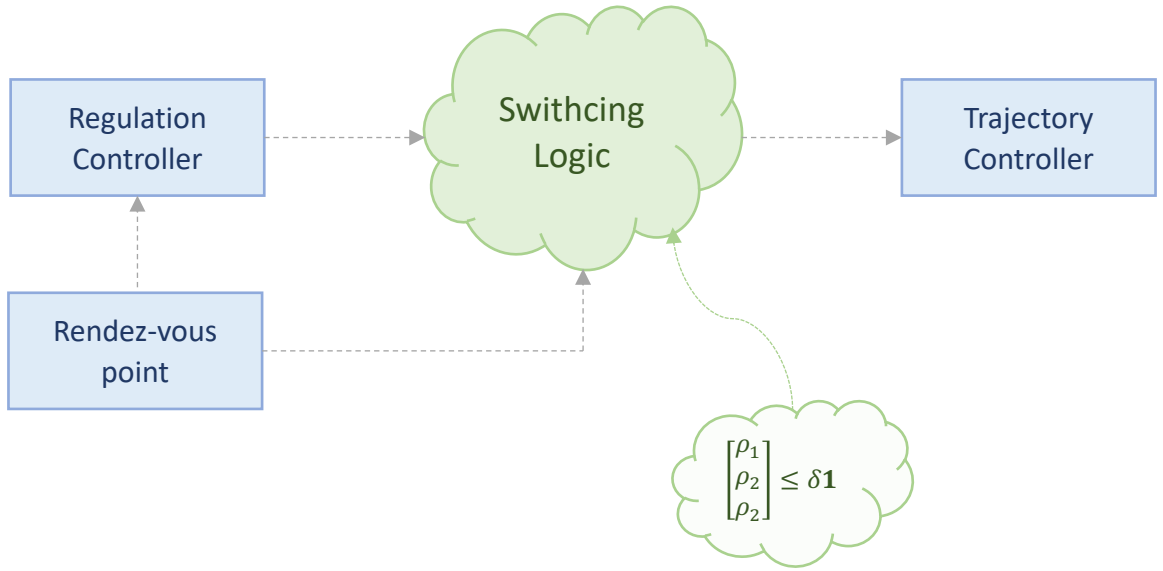
Convergence in approximately 37 seconds.





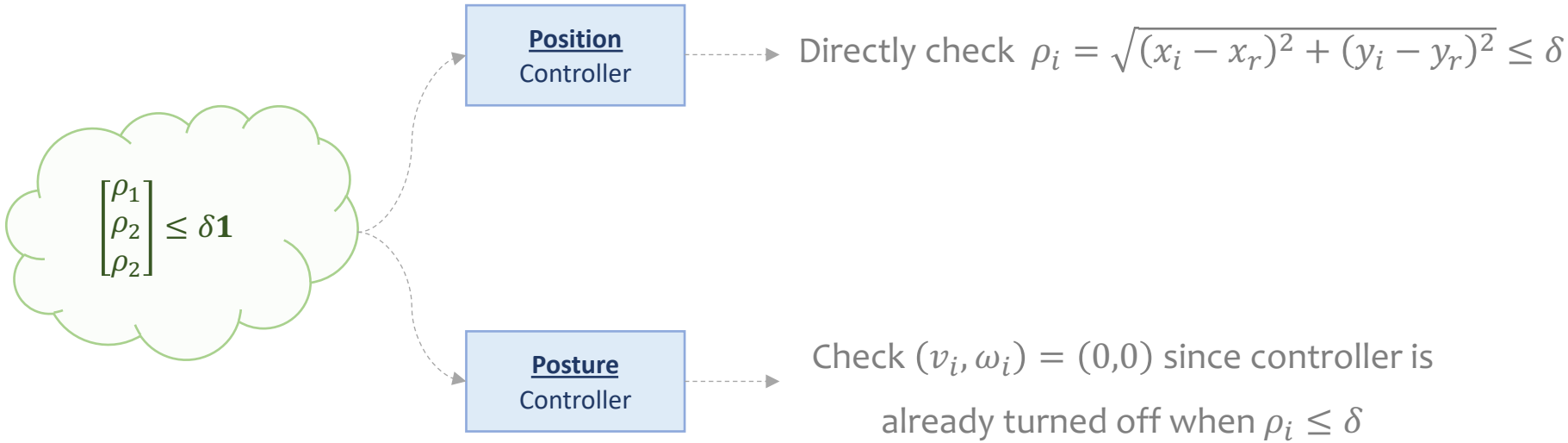
[5] Switching Controller

Until this point, each task (regulation/tracking) was executed independentely. The main purpose of this section is to provide an explanation of how the task may be executed in the same simulation. It is important to remark that some tradeoffs are necessary.

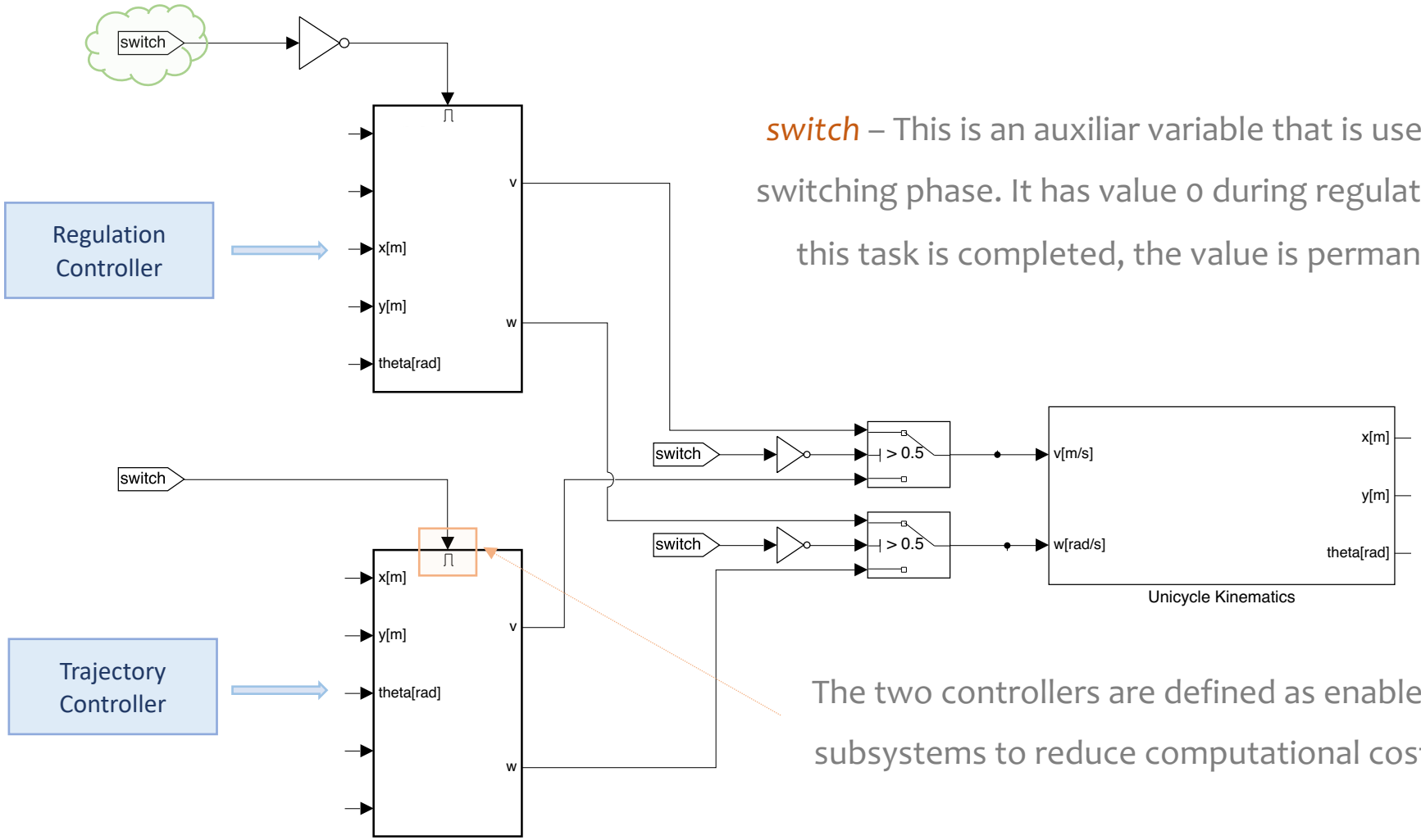


The regulation controller is stopped and the trajectory controller starts working when the CoM of all three agents is inside a circle of radius $\delta = 10^{-2}\ [m]$, centered in the rendez-vous point.

[5.1] Switching Logic



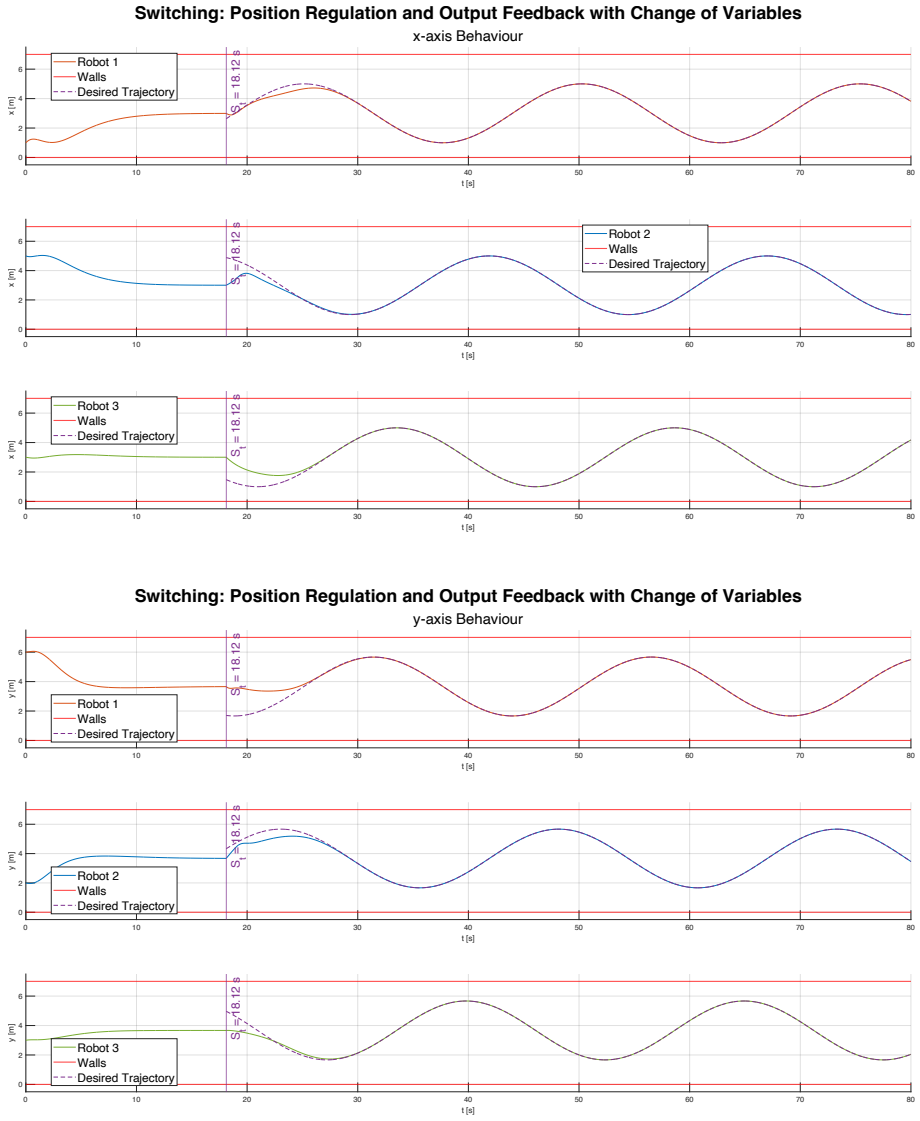
[5.1] Switching Controller

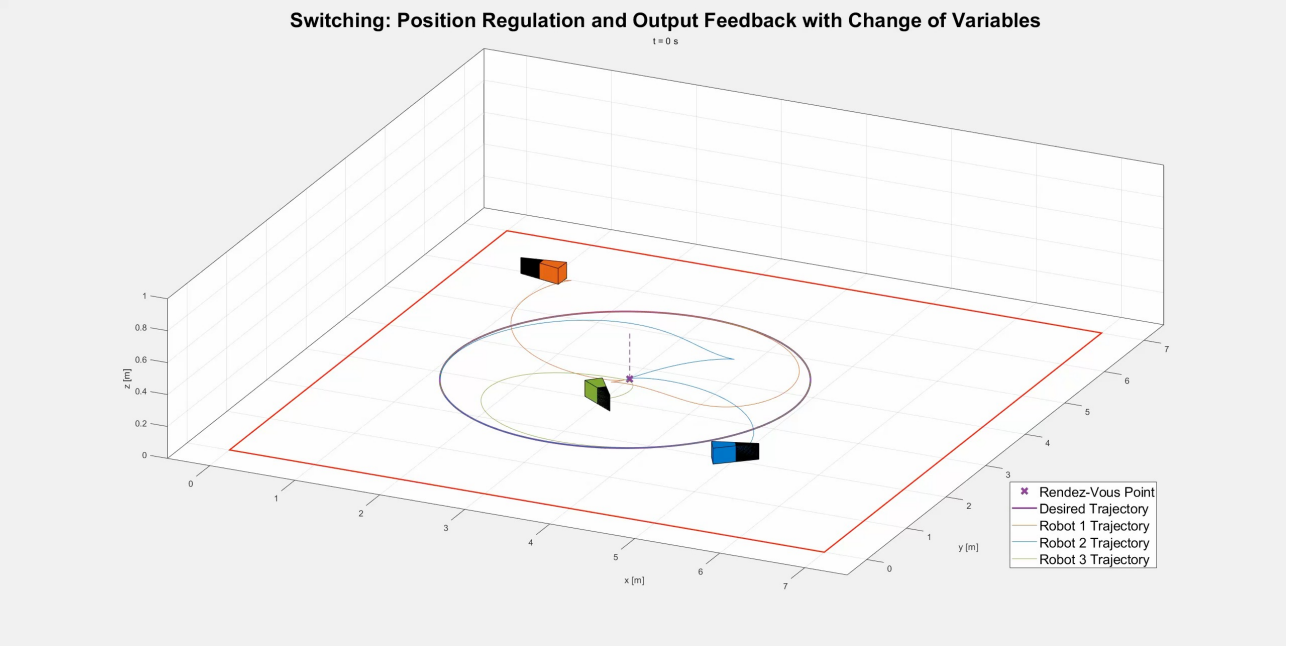


switch – This is an auxiliar variable that is used to handle the switching phase. It has value 0 during regulation phase. Once this task is completed, the value is permanently set to 1.

The two controllers are defined as enabled subsystems to reduce computational cost.

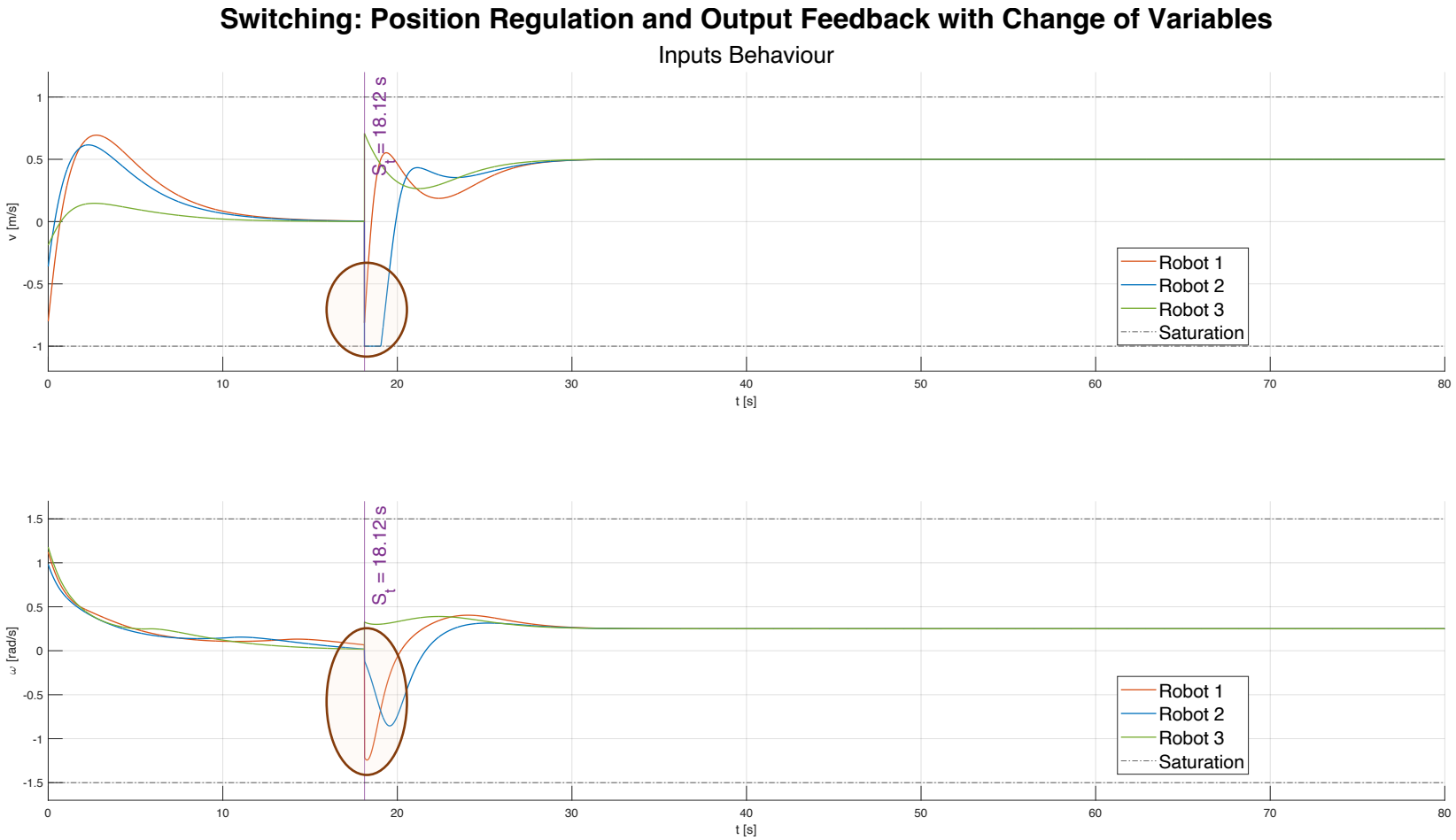
Position Regulation & OF w/ change of variables





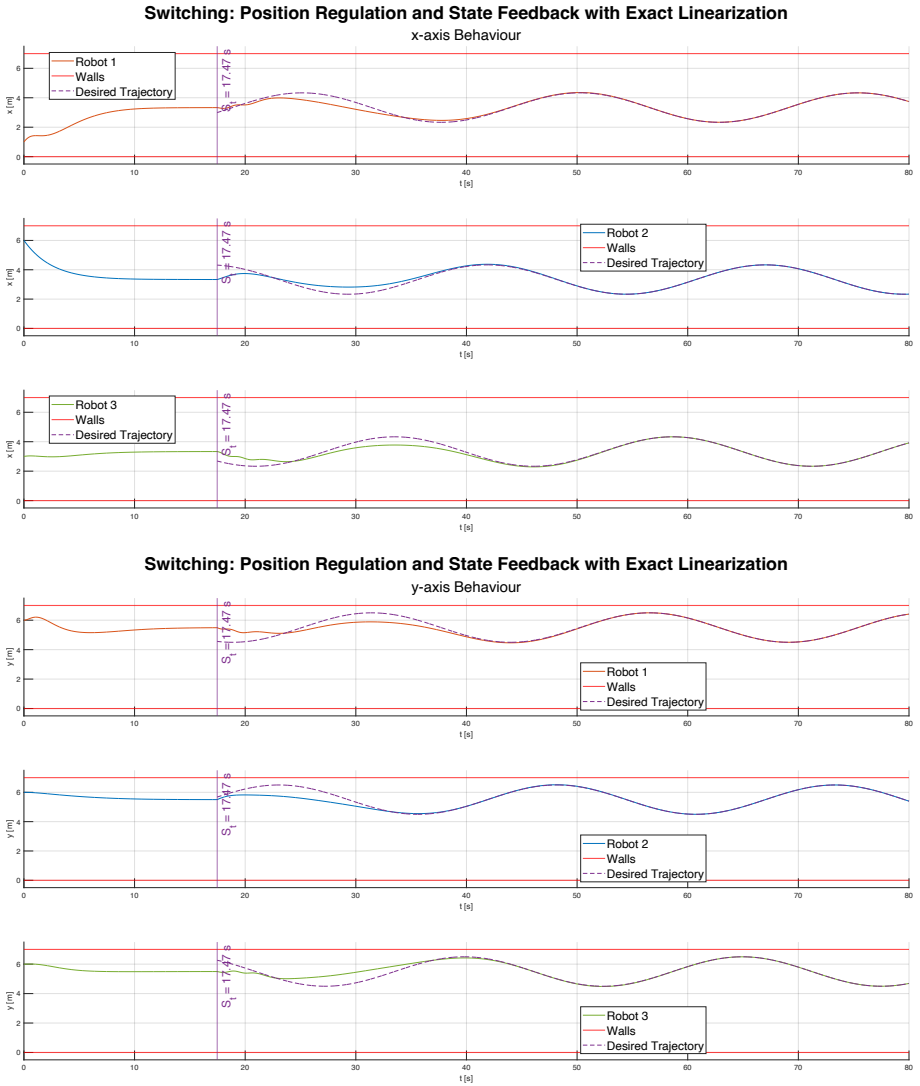
Position Regulation & OF w/ change of variables

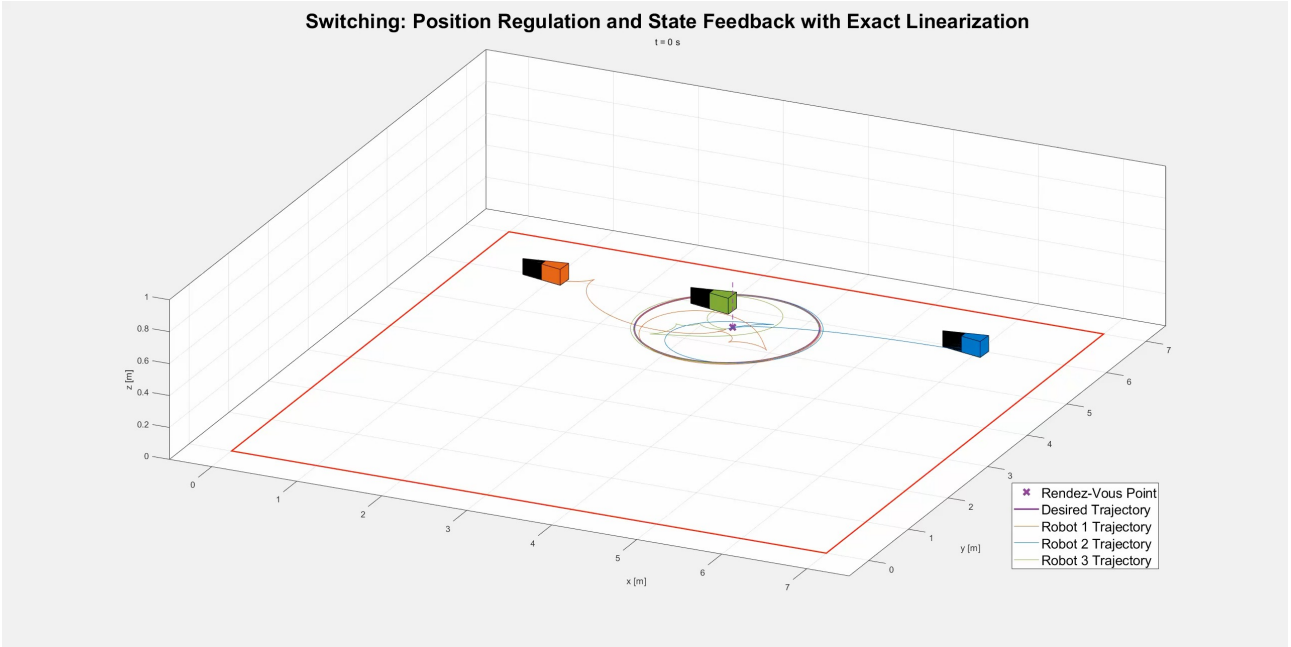
Once the three agents reach stationary conditions, they are supposed to have the same control input in velocity.



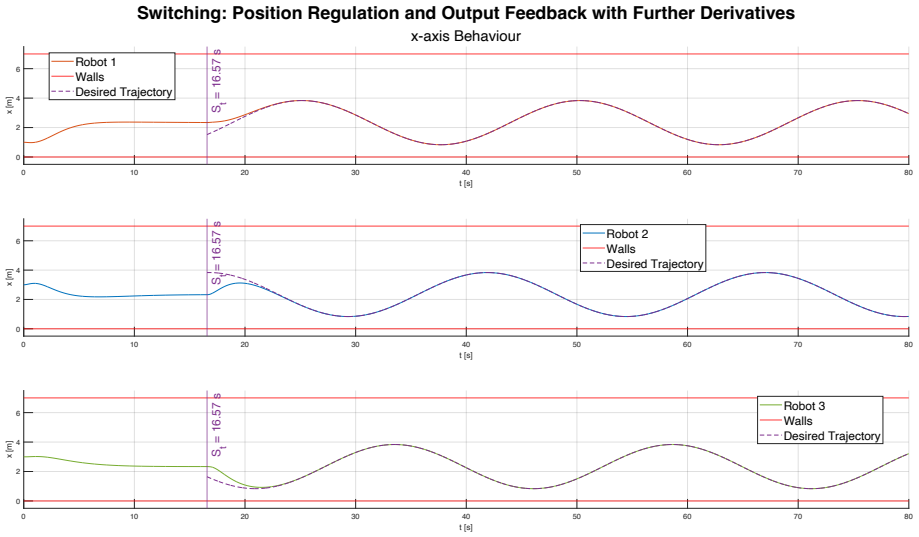
Remark: the control action has a strong variation after the switching time. This is due to istantaneous echange between the two executed tasks.

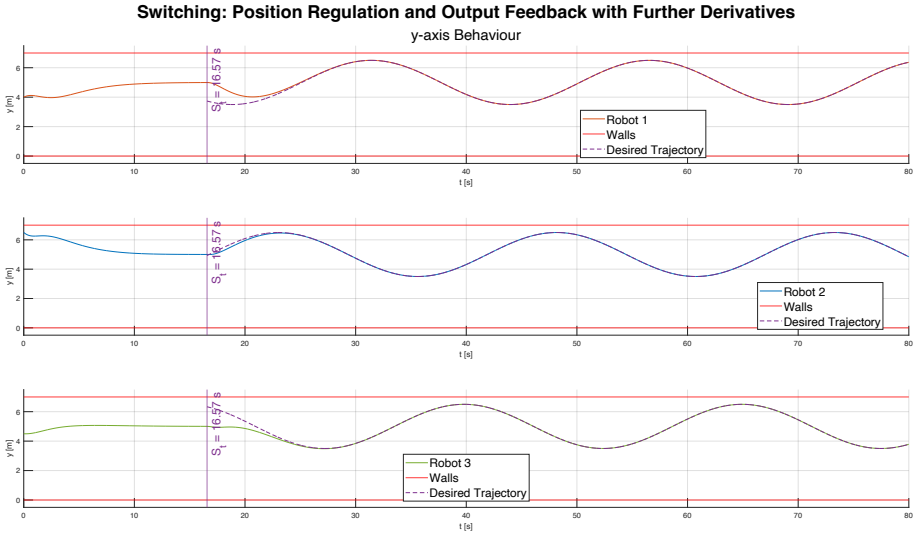
Position Regulation & SF w/ Non-Linear Ctrl.

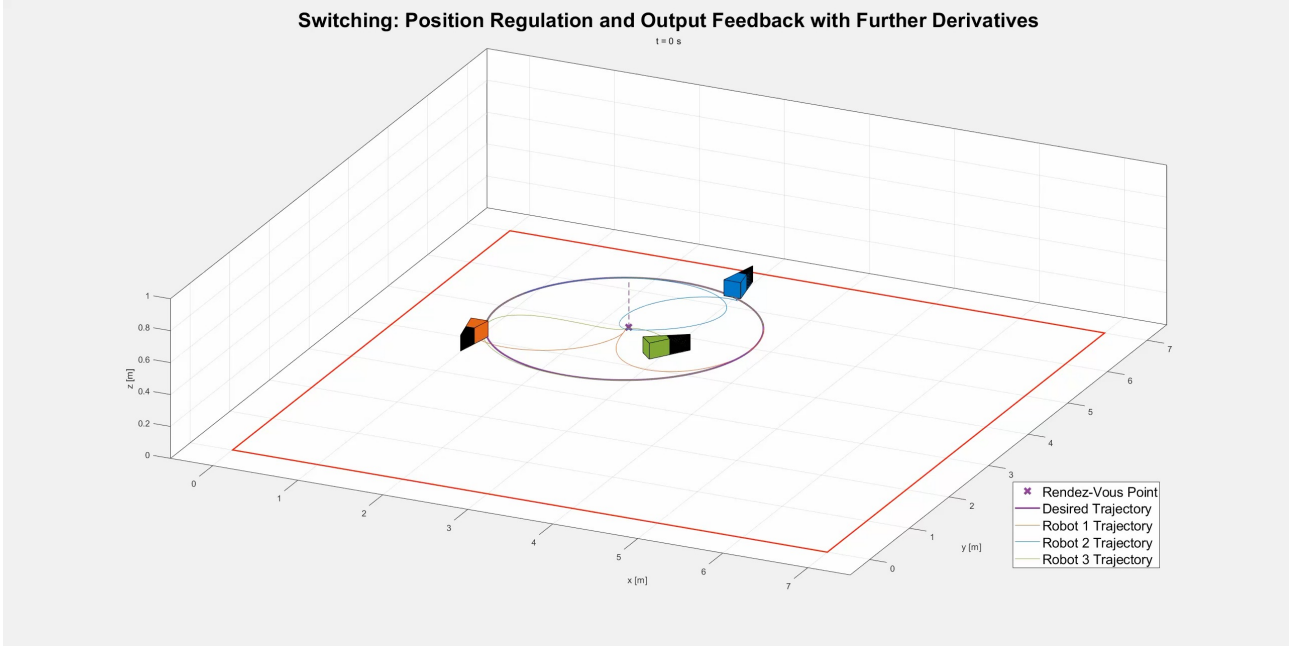


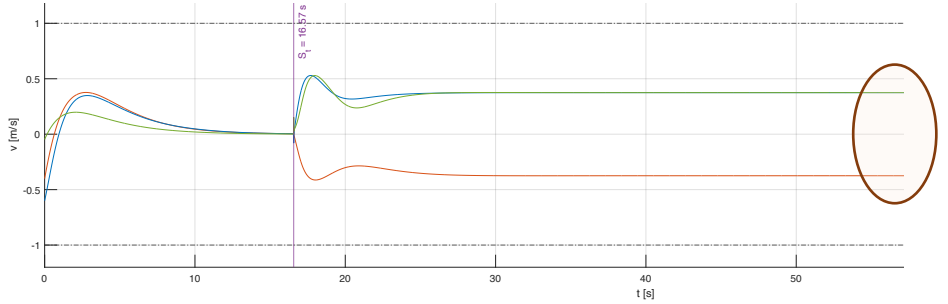


Position Regulation & OF w/ Further Derivatives



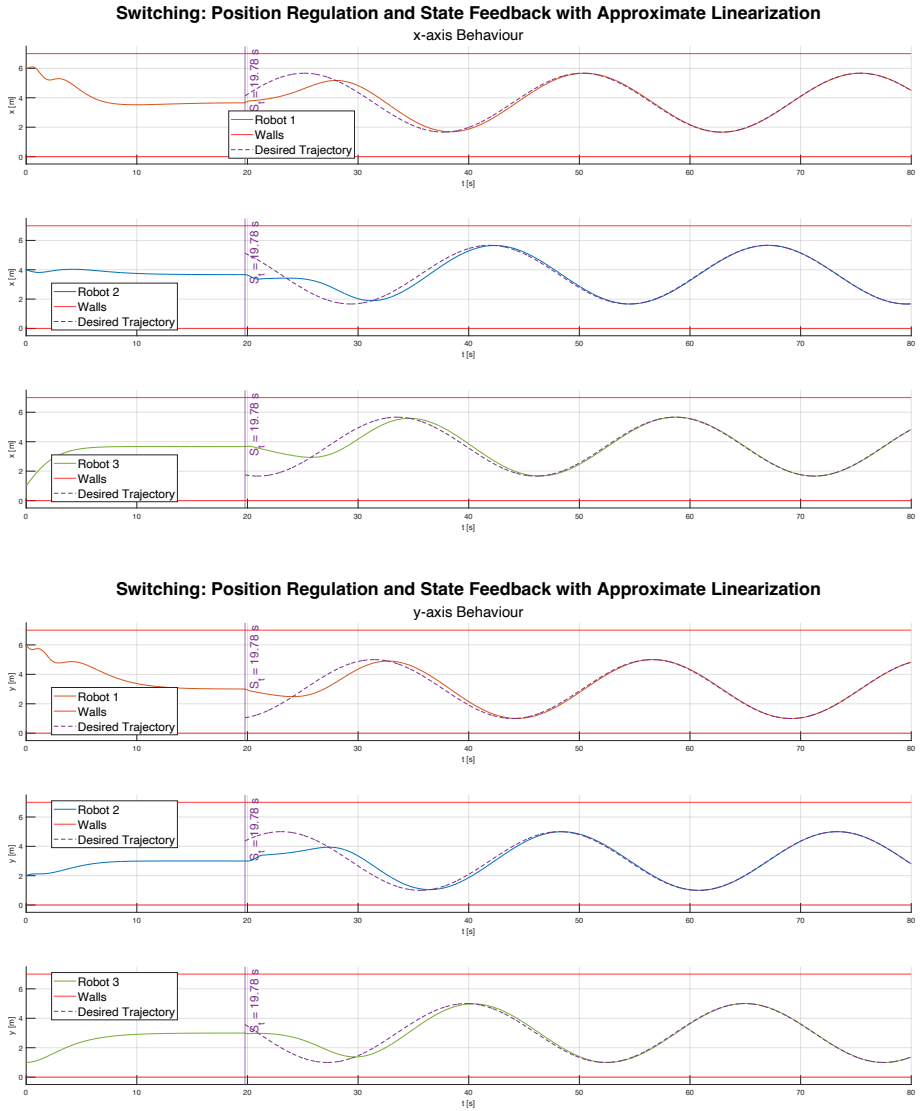


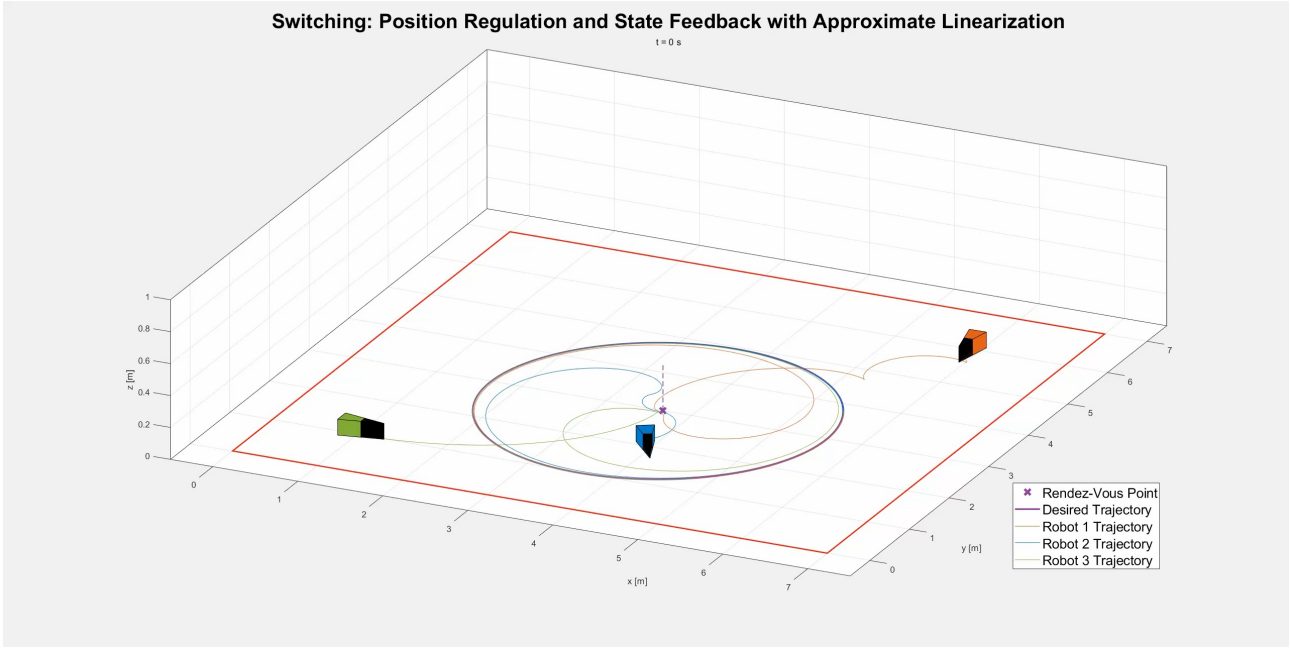




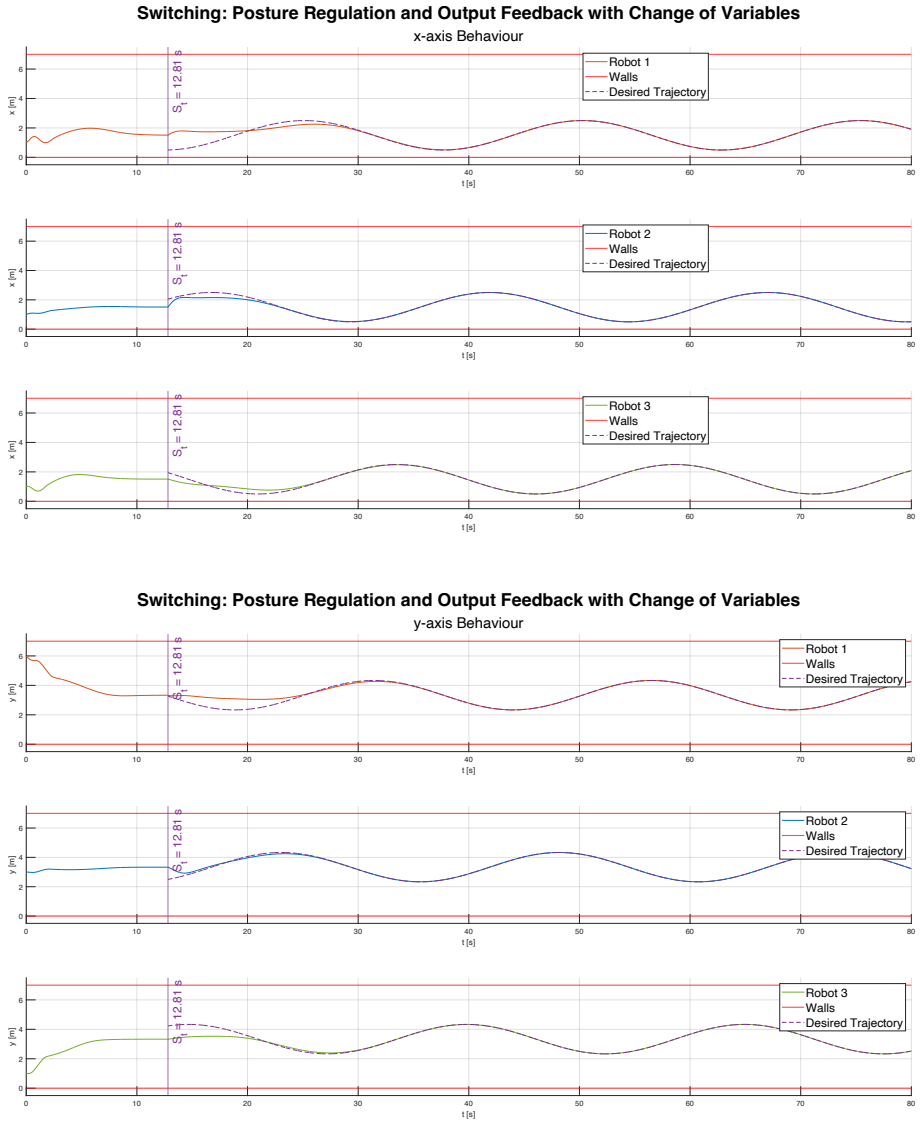
Remark: linear velocities have opposite sign.

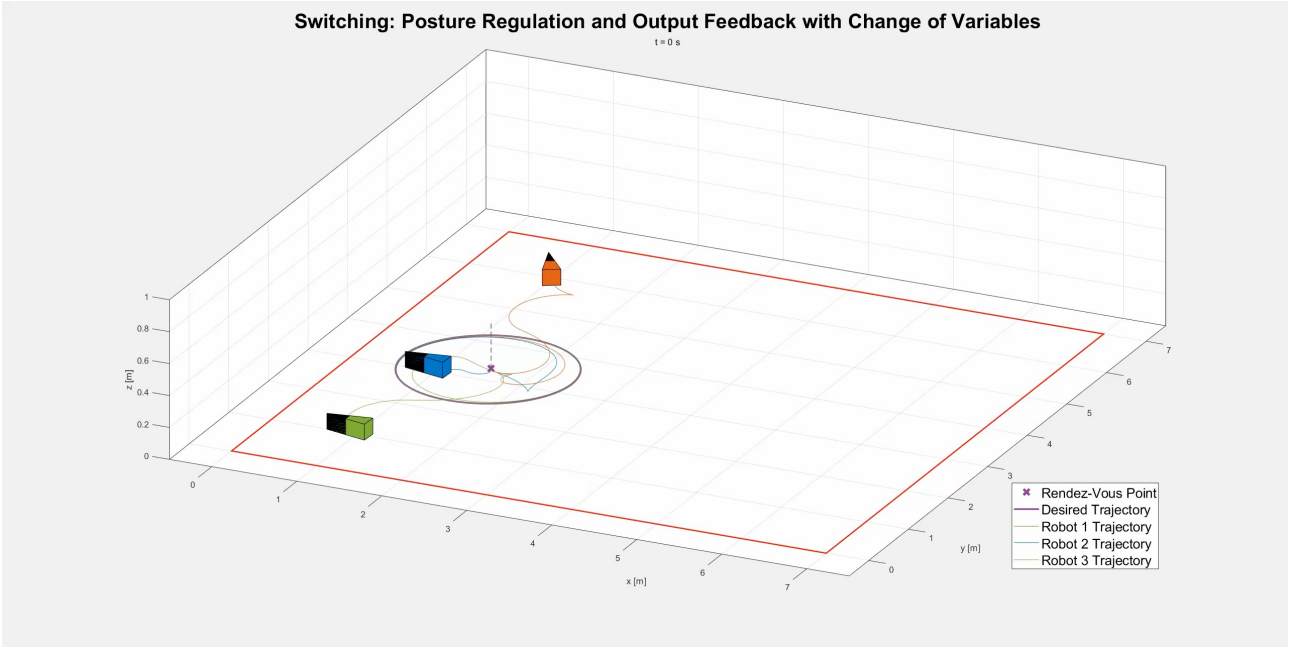
Position Regulation & SF w/ Linearization



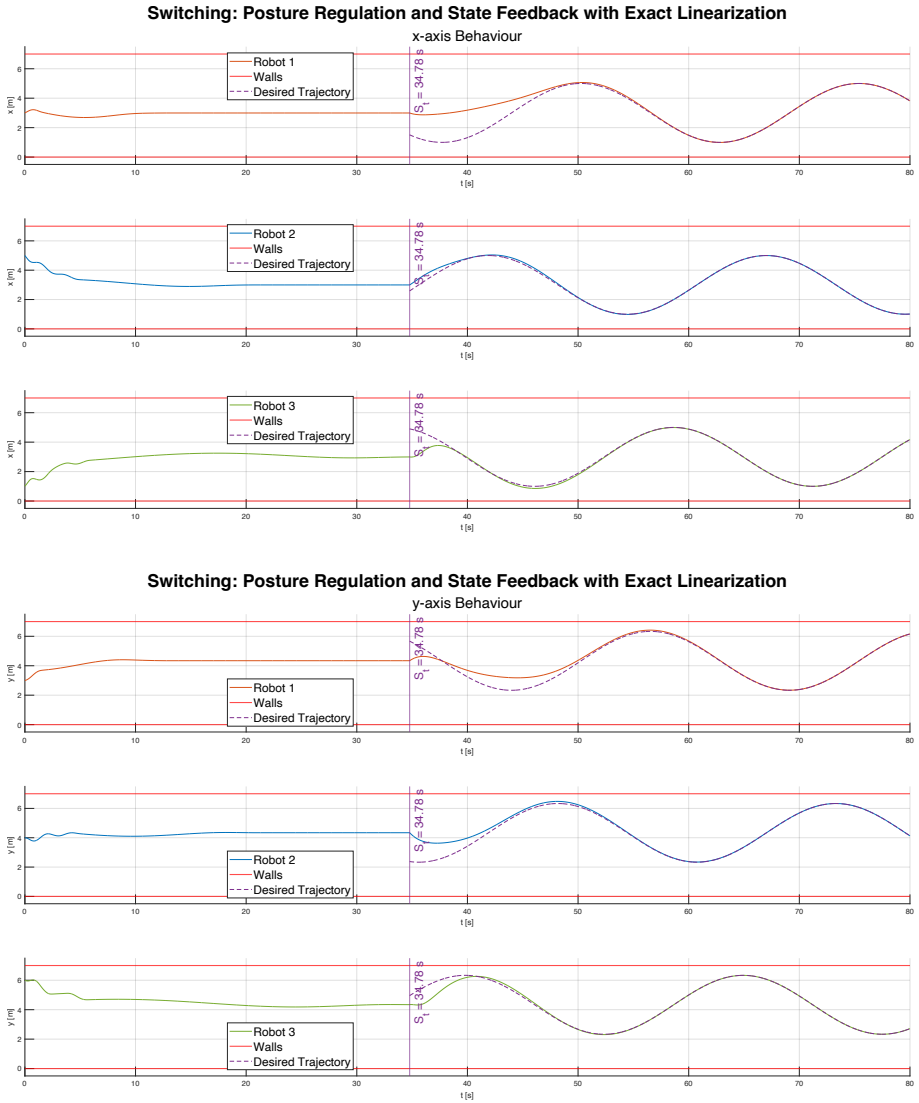


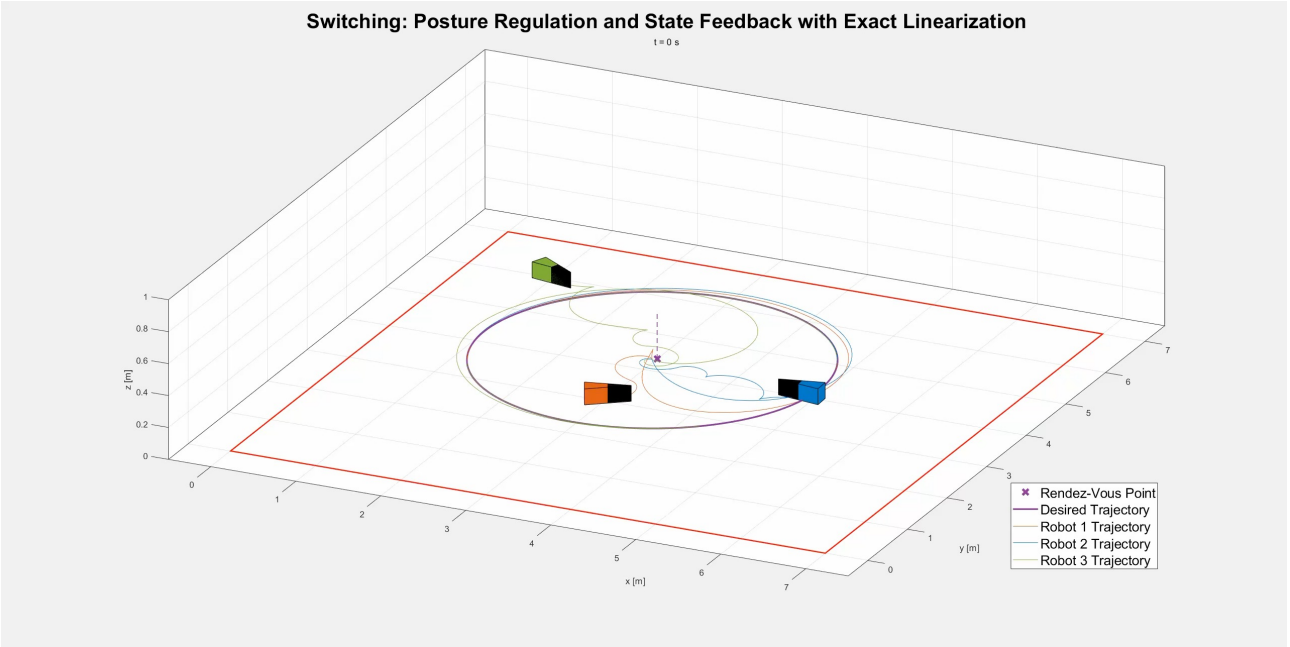
Posture Regulation & OF w/ Change of variables



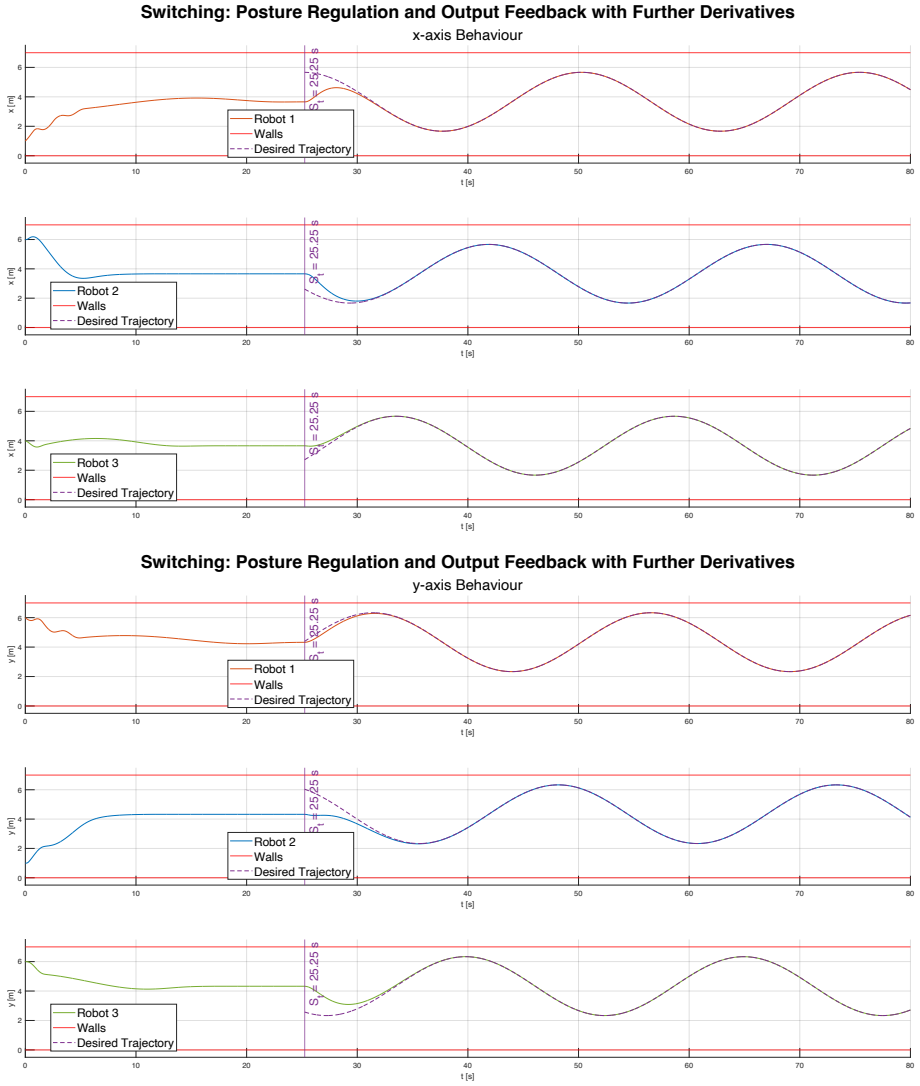


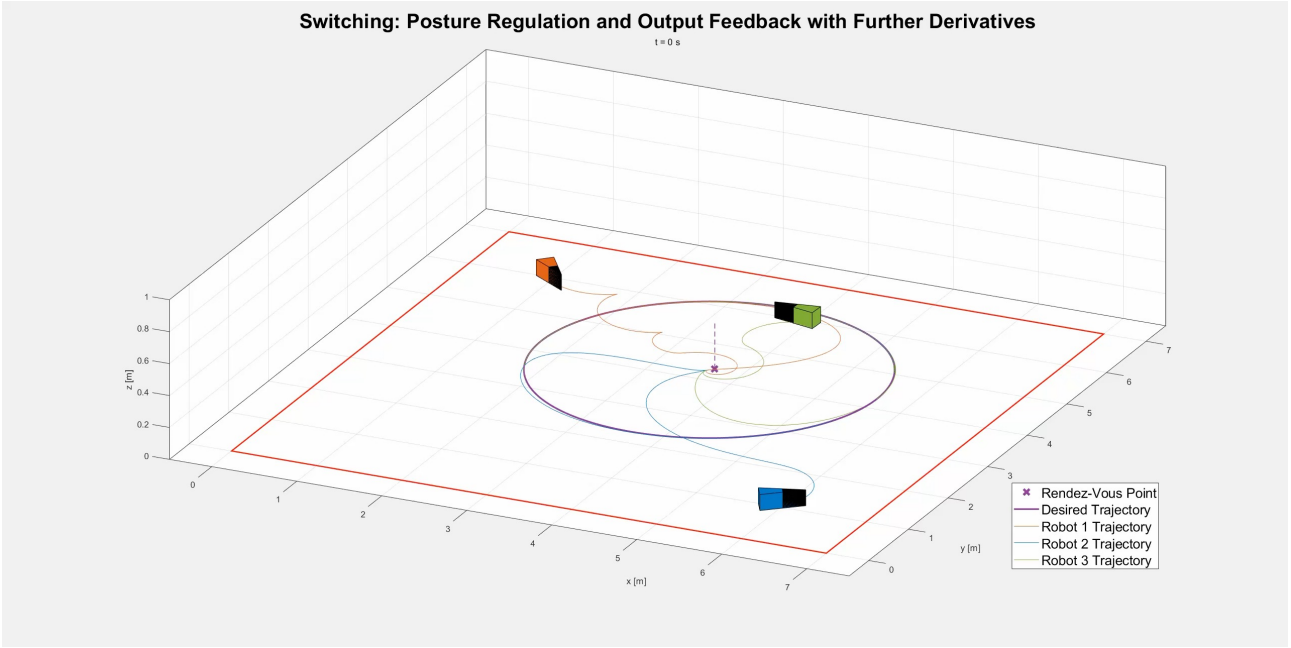
Posture Regulation & SF w/ Non-Linear ctrl.



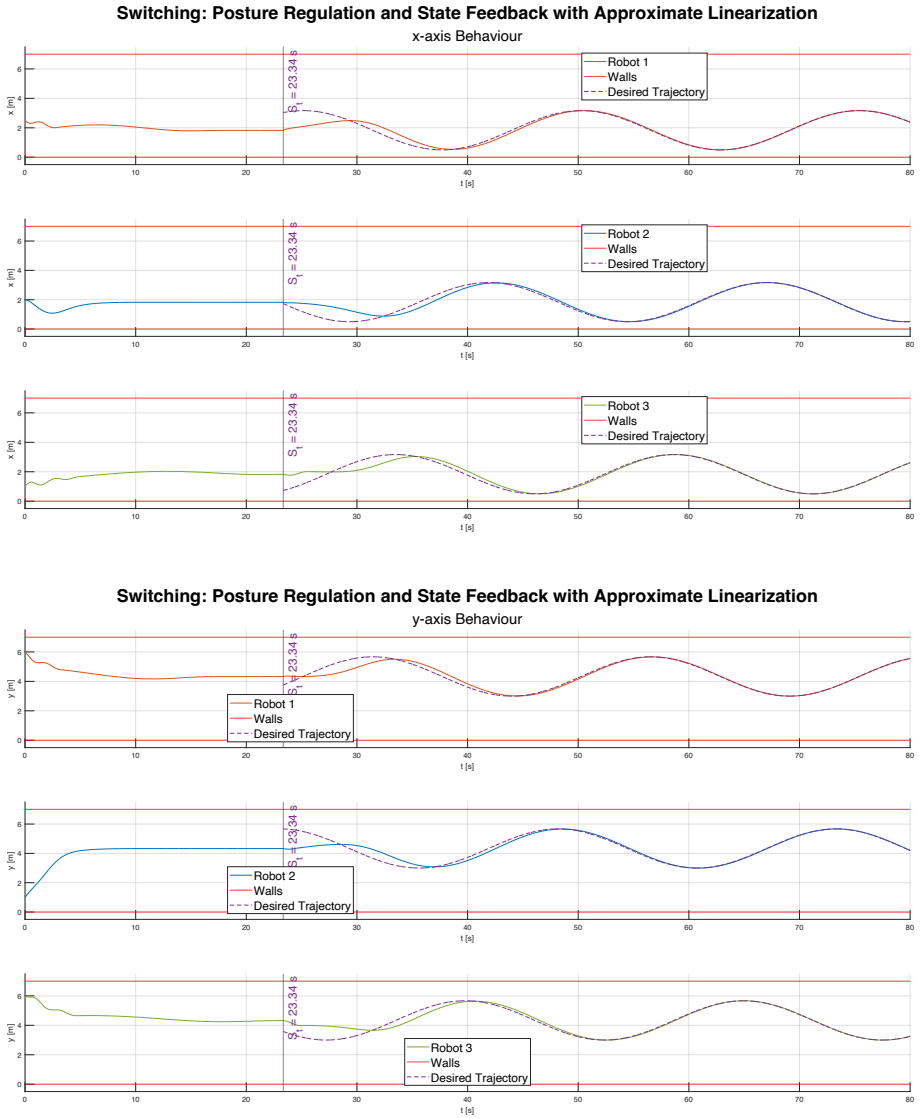


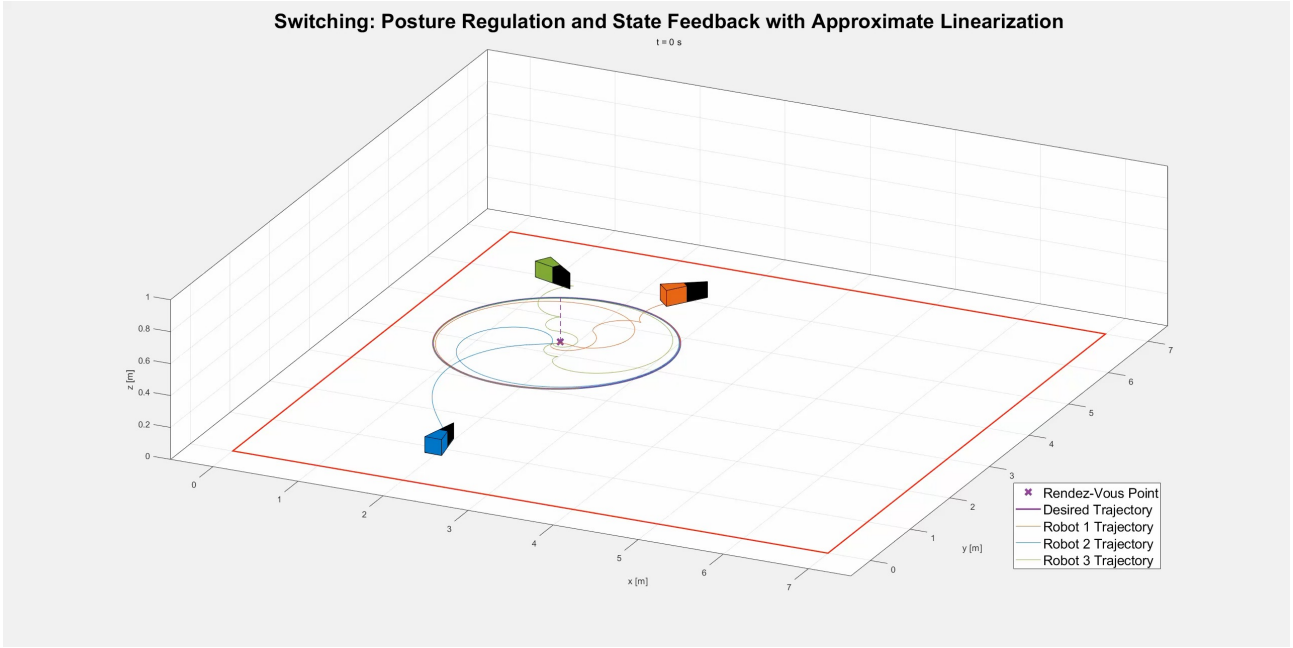
Posture Regulation & OF w/ Further Derivatives





Posture Regulation & SF w/ Linearization





Conclusions

1) All the designed controller guarantee ***perfect convergence*** both to ther rendez-vous point and to the desired trajectory.
2) Considering the tested starting points, agents never touch the room wall. This is mainly due to the use of ***low gains***, that guarantee a smooth control action.
3) All controllers guarantee an optimal behaviour even in presence of ***saturations*** blocks, that make the application closer to ones in real world.
4) It is woth noticing that the choice between the use of position or posture control depends on wich **tradeoff** is desired (faster position convergence vs. slower positon convergence but same arrive angle).
5) Looking at our results (only a particular case), it is possible to notice that output feedback converges faster than state feedback.

Future developments

1) Try the application in a **real scenario** with a differential drive robot (e.g. adding a map to obtain wheel linear velocities v_L, v_R from v, ω).
2) Implement **collision avoidance**, with a particular attention close to the rendez-vous point.