Suite of Feedback Controllers for Unicycle Robots

Roberta Di Tucci
Tommaso Marroni
Gabriele Perfumo

SAPIENZA
UNIVERSITÀ DI ROMA

Department of Computer, Control, and Management Engineering
Control Engineering

Autonomous and Mobile Robotics (AMR)
Final Project (FP7)

Summary

Introduction

Goal of the project

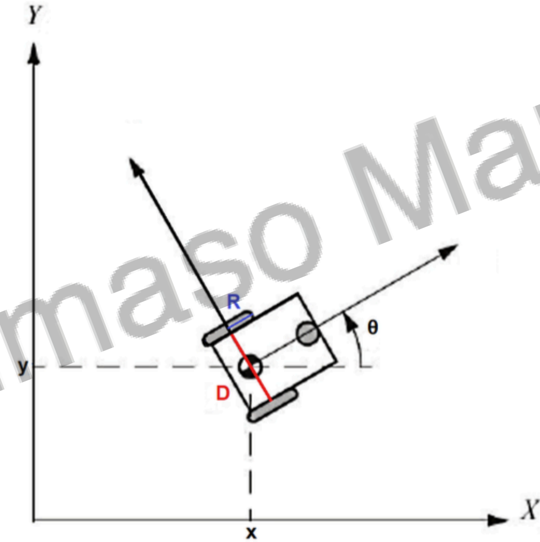
- The goal of the project is the control of a **differential-drive mobile robot** using a set of different types of control schemes.
- All controllers are implemented and simulated in Simulink.
- Controllers: 3 for trajectory tracking, 3 for posture stabilization.

Introduction

Model of the differential-drive mobile robot

The following equations describe the state-space representation for the chosen robot:

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



$$v = \frac{R(\omega_r + \omega_l)}{2}$$
$$\omega = \frac{R(\omega_r - \omega_l)}{D}$$

$$\rightarrow \begin{cases} \dot{x} = \frac{R}{2}\cos\theta\omega_r + \frac{R}{2}\cos\theta\omega_l \\ \dot{y} = \frac{R}{2}\sin\theta\omega_r + \frac{R}{2}\sin\theta\omega_l \\ \dot{\theta} = R\frac{\omega_r}{D} - R\frac{\omega_l}{D} \end{cases}$$

Introduction

Control problems

The controllers taken into consideration are used to solve either one of these problems:

Tracking problem: Make the robot follow a given reference trajectory $(x_d(t), y_d(t))$

Regulation problem: Make the robot reach a desired state configuration (x_f, y_f, θ_f) starting from a given initial one (x_i, y_i, θ_i)

Introduction

Localization

Computing the correct control actions requires the knowledge of the position and the orientation of the robot in order to use a feedback loop. In this work, two ways of doing so are used:

GPS: An external system sends information to the robot about its configuration.

Odometry: Information is obtained by numerical integration of the kinematic model over a sampling interval (using **2° order Runge-Kutta** algorithm**)**.

$$x_{k+1} = x_k + v_k T_s \cos\left(\theta_k + \frac{\omega_k T_s}{2}\right)$$

$$y_{k+1} = y_k + v_k T_s \sin\left(\theta_k + \frac{\omega_k T_s}{2}\right)$$

$$\theta_{k+1} = \theta_k + \omega_k T_s$$

Trajectory Tracking

- Linear Approximation

- RAS Controllers
  - taken from [1]

- CST Controller
  - taken from [2]

[1] Luigi Freda, Giuseppe Oriolo, *Vision-based interception of a moving target with a nonholonomic mobile robot*, Elsevier, Robotics and Autonomous Systems 55 (2007) 419-432.

[2] Giuseppe Oriolo, Alessandro De Luca, *WMR Control Via Dynamic Feed- back Linearization: Design, Implementation, and Experimental Validation*, IEEE Transactions on Control Systems Technology, Vol. 10, No. 6, 835-852, November 2002.

Linear Approximation

- *Linearize the error dynamics around the reference trajectory*
- Control law:

$$v = v_d \cos e_3 - k_1 e_1$$
$$\omega = \omega_d + k_2 e_2 + k_3 e_3$$

- e_1, e_2 and e_3 (rotated component of the error):

$$e = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_d - x \\ y_d - y \\ \theta_d - \theta \end{pmatrix}$$

Linear Approximation

- k_1, k_2 as follows:

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

- v_d and ω_d are obtained from the desired trajectory using flatness:

$$\theta_d = \mathrm{atan2}(\dot{\mathrm{y}}_\mathrm{d}, \dot{\mathrm{x}}_\mathrm{d}) + \mathrm{k}\pi$$
$$v_d = \pm\sqrt{\dot{x_d}^2 + \dot{y_d}^2}$$
$$\omega_d = \frac{\ddot{y_d}\dot{x_d} - \ddot{x_d}\dot{y_d}}{\dot{x_d}^2 + \dot{y_d}^2}$$

- Asymptotic stability is **not guaranteed**, unless v_d and ω_d are constant.
  - even for these two types of references, the asymptotic stability is **not global**.

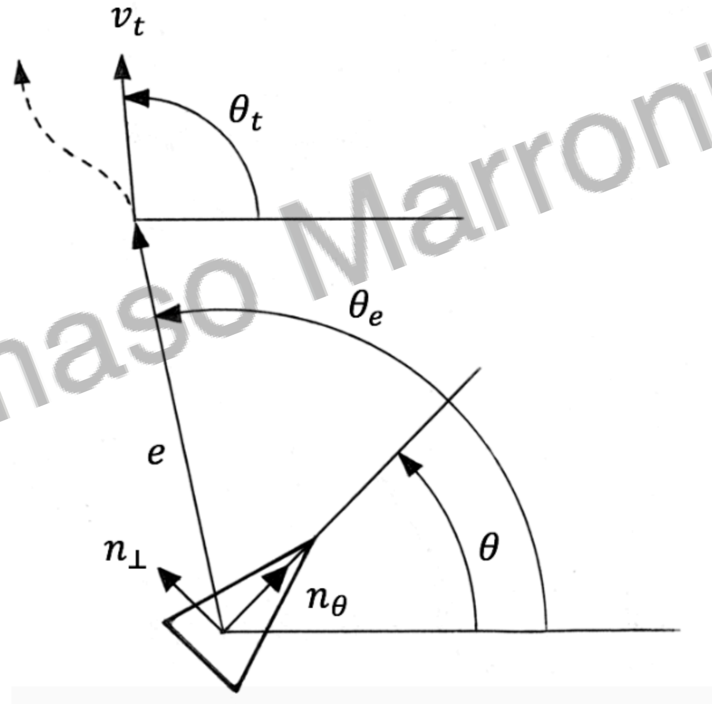
RAS

- First controller (RAS 1):

$$v = k_1 e^T n_\theta + v_t^T n_\theta$$
$$\omega = k_2(\theta_e - \theta) + \dot{\theta}_e$$

- Second controller (RAS 2)

$$v = k_1 e^T n_\theta + v_t^T n_\theta$$
$$\omega = k_2(\theta_e - \theta) + k_3 e^T n_\perp + \dot{\theta}_e$$



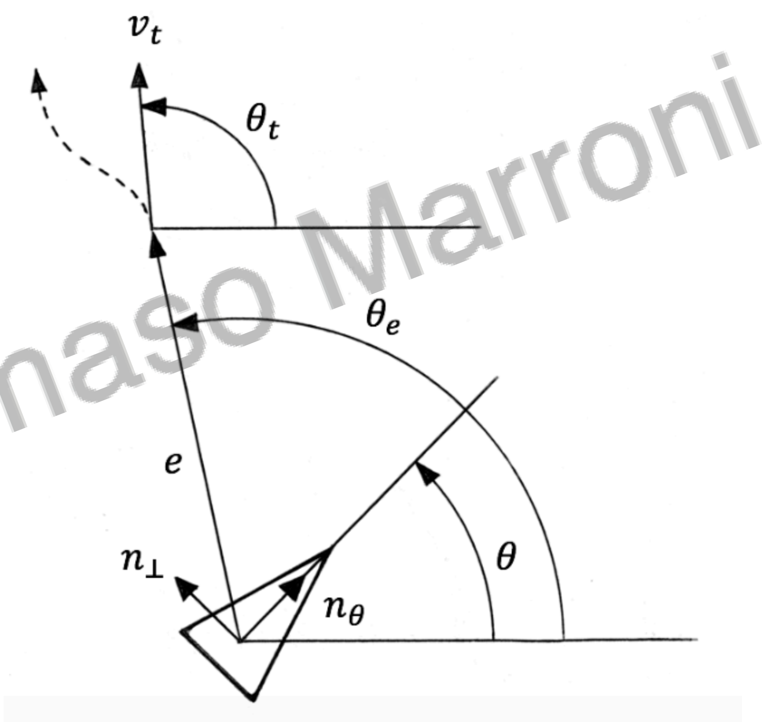
RAS

- Main difference from [1]: usage of a given reference instead of a system composed of a camera and a moving target.

- Presence of component depending on e in ω of RAS 2.

  → faster alignment

- Unicycle can follow only trajectories described by $x_d(t)$, $y_d(t)$ of class $\bar{C}^2$.



CST

- *Nonlinear controller based on Dynamic Feedback Linearization*.
- Dynamic compensator:

$$\dot{\xi} = u_1 \cos\theta + u_2 \sin\theta$$
$$v = \xi$$
$$\omega = \frac{u_2 \cos\theta - u_1 \sin\theta}{\xi}$$

- The feedback (providing exponential stabilization):

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

- $\xi = 0$ is a singularity for the dynamic feedback compensator.

Regulation

- *Drive the unicycle to a desired configuration* q_d.
- Tracking a trajectory which stops in q_d wouldn't work:

  – Error-dynamics-based controllers require persistent trajectories.

  – Need a point B as output when performing I/O linearization.

I/0 Linearization leads point B to destination

Regulation

- Cartesian Regulation
- Posture Regulation
- CST Regulation
  - taken from [2]

[2] Giuseppe Oriolo, Alessandro De Luca, *WMR Control Via Dynamic Feed- back Linearization: Design, Implementation, and Experimental Validation*, IEEE Transactions on Control Systems Technology, Vol. 10, No. 6, 835-852, November 2002.

Cartesian Regulation

- *Drive the unicycle to the desired Cartesian position* (x_d, y_d)

- Model of the control:

$$v = k_1 \boldsymbol{e_p}^T \boldsymbol{n}$$
$$\omega = k_2(\mathrm{atan2}(y_e, x_e) - \theta)$$

- With this control law robot reaches the target configuration through a forward motion.

Posture Regulation

- *Drive the unicycle to the* desired configuration (x_d, y_d, θ_d)
- Model of the control:

$$v = k_1 \rho \cos\gamma$$

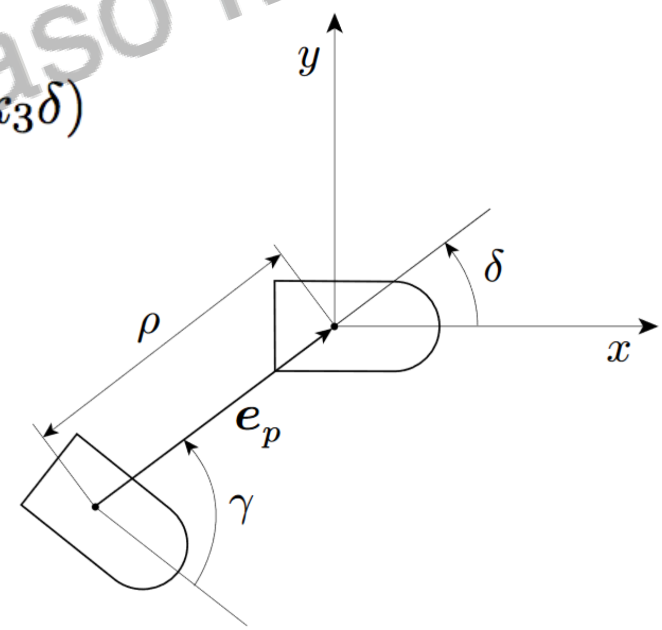
$$\omega = k_2\gamma + k_1 \frac{\sin\gamma\cos\gamma}{\gamma}(\gamma + k_3\delta)$$

- Polar coordinates:

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

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

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



CST Regulation

- Solve posture stabilization problem
- Model of the control:
  - Compensator:

$$\dot{\xi} = u_1 \cos\theta + u_2 \sin\theta$$
$$v = \xi$$
$$\omega = \frac{u_2 \cos\theta - u_1 \sin\theta}{\xi}$$

  - Feedback:

$$u_1 = -k_{p1}x - k_{d1}\dot{x}$$
$$u_2 = -k_{p2}y - k_{d2}\dot{y}$$

CST Regulation

- Exponential convergence to the origin is guaranteed if $q_0 \in Q/Q^*$, with Q^* such that:

$$Q^* = \{q \in Q : (x = 0, \cos\theta \geq 0) \; OR \\ (y = 0, \cos\theta = -1) \; OR \\ (x = y = 0)\}$$

and:

$$k_{d1}^2 - 4k_{p1} = k_{d2}^2 - 4k_{p2} > 0$$

$$k_{d2} - k_{d1} > 2\sqrt{k_{d2}^2 - 4k_{p2}}$$

$$\begin{cases} \xi_0 < 0 \;\; (backward\ motion) & if\ q_0 \in RHP/Q^* \\ \xi_0 > 0 \;\; (forward\ motion) & if\ q_0 \in LHP/Q^* \end{cases}$$

CST Regulation

- Conditions to avoid singularity in $\xi = 0$:

$$\xi_0 \neq 2\frac{k_{p1}x_0 \sin\theta_0 - k_{p2}y_0 \cos\theta_0}{k_{d2} - k_{d1}}$$

- This condition guarantees that eigenvalues characterizing the transient are real.

- When $q_0 \in Q^*$ the controller brings robot to the wrong orientation.

  → a via point is needed

  → global exponential stability.

Simulations

Trajectories for tracking

The following trajectories were used to test the efficiency of the tracking controllers:

- **Linear**:

$$\begin{cases} x_d(t) = t \\ y_d(t) = 0 \end{cases}$$

- **Circular**:

$$\begin{cases} x_d(t) = \cos(t) \\ y_d(t) = \sin(t) \end{cases}$$

- **Linear-Circular succession**:

$$\begin{cases} x_d(t) = t \\ y_d(t) = 0 \end{cases} \quad , t < t_1$$

$$\begin{cases} x_d(t) = t_1 + \cos(t - 4 - \frac{\pi}{2}) \\ y_d(t) = -1 - \sin(t - 4 - \frac{\pi}{2}) \end{cases} \quad , t \geq t_1$$

Simulations

Trajectories for tracking

- **Cubic**:

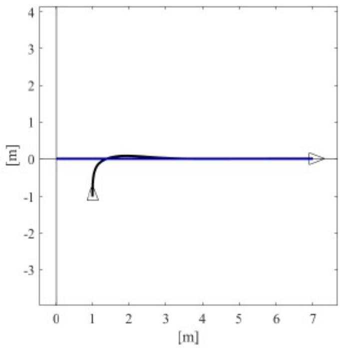
$$\begin{cases} x_d(t) = t \\ y_d(t) = t^3 \end{cases}$$

- **Spiral:**

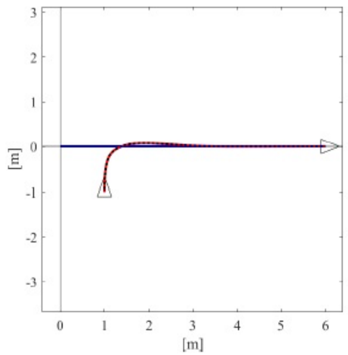
$$\begin{cases} x_d(t) = t\cos t^2 \\ y_d(t) = t\sin t^2 \end{cases}$$

Simulations

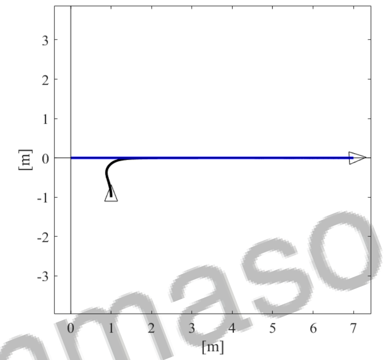
Linear trajectory



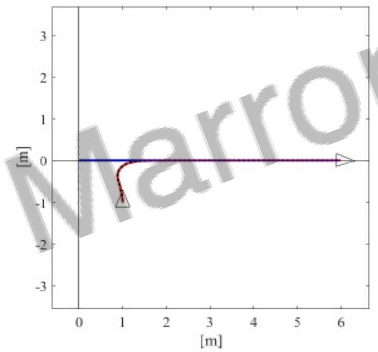
Lin. Approx. GPS



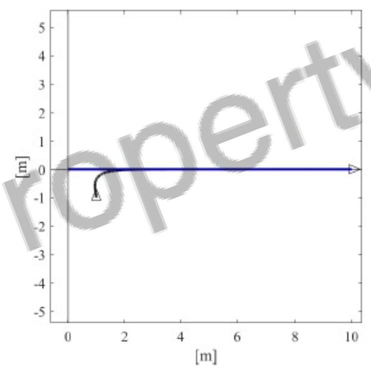
Lin. Approx. R-K



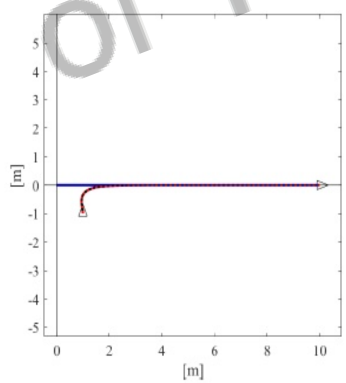
RAS 1 GPS



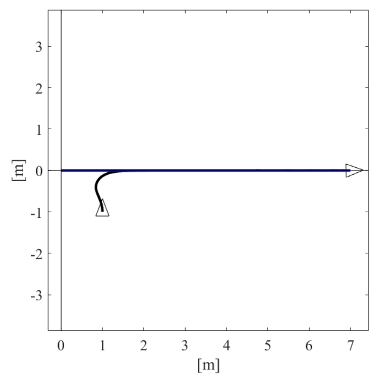
RAS 1 R-K



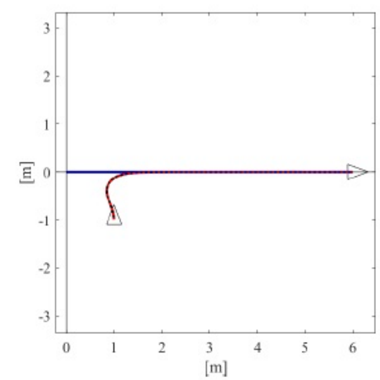
CST GPS



CST R-K



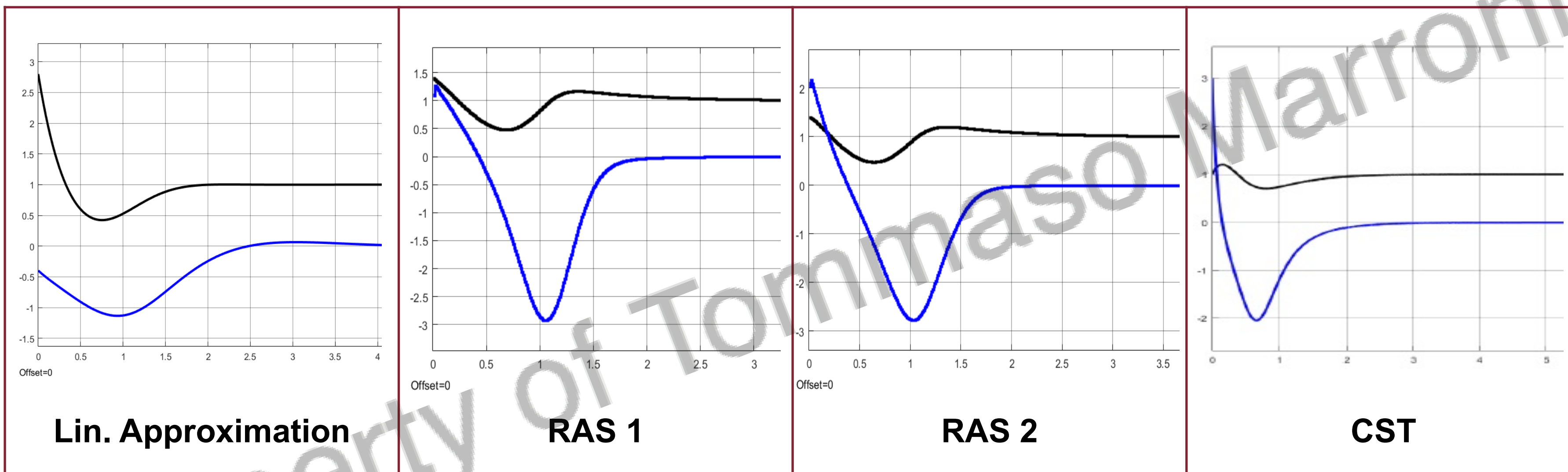
RAS 2 GPS



RAS 2 R-K

Simulations

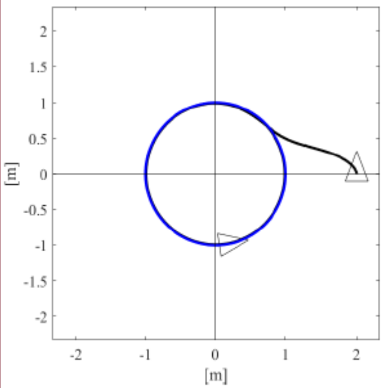
Control actions



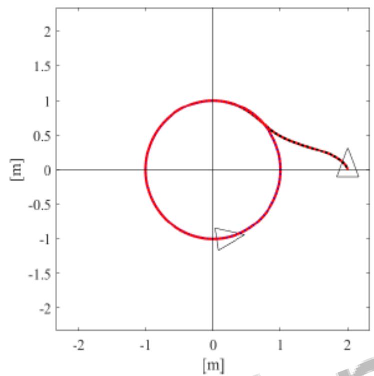
v (black) and ω (blue) in the GPS case.

Simulations

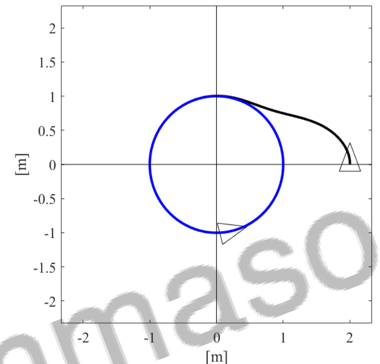
Circular trajectory



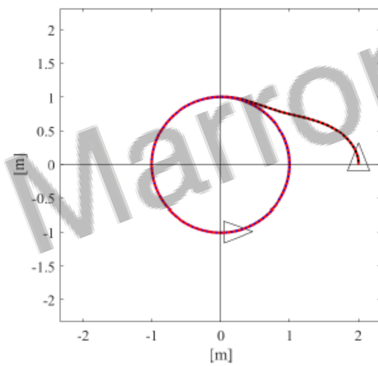
Lin. Approx. GPS



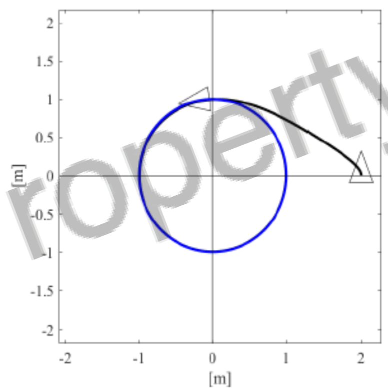
Lin. Approx. R-K



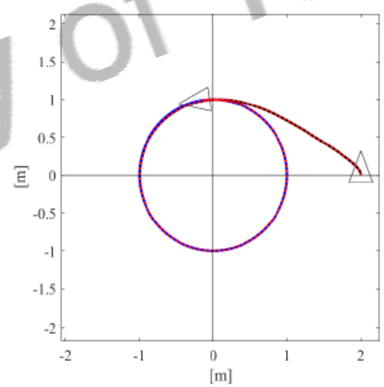
RAS 1 GPS



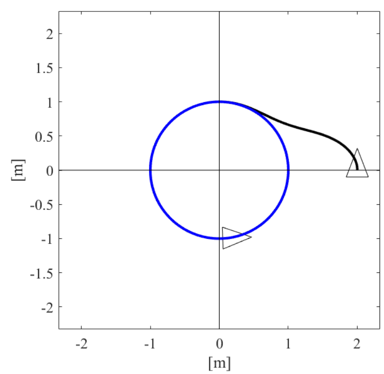
RAS 1 R-K



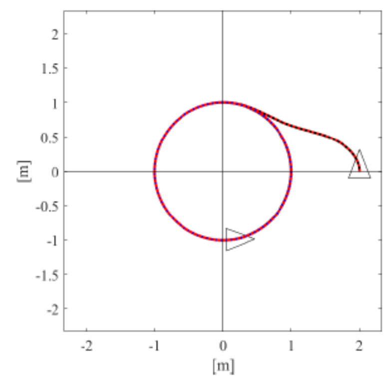
CST GPS



CST R-K



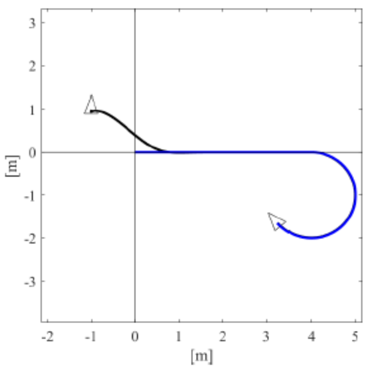
RAS 2 GPS



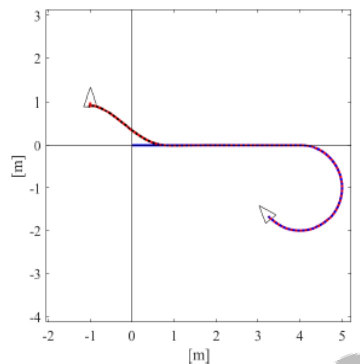
RAS 2 R-K

Simulations

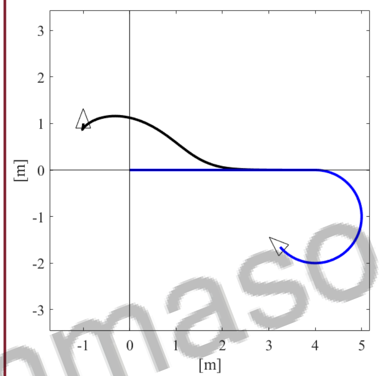
L-C trajectory



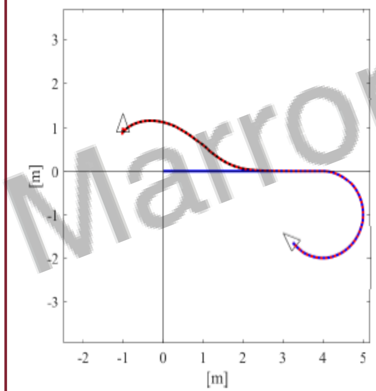
Lin. Approx. GPS



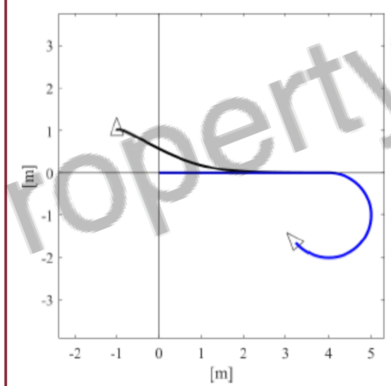
Lin. Approx. R-K



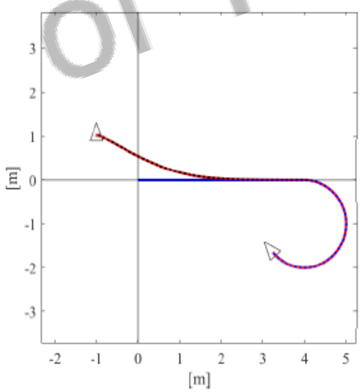
RAS 1 GPS



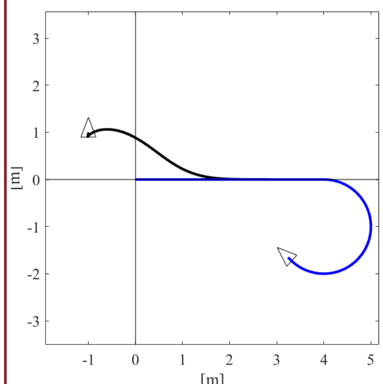
RAS 1 R-K



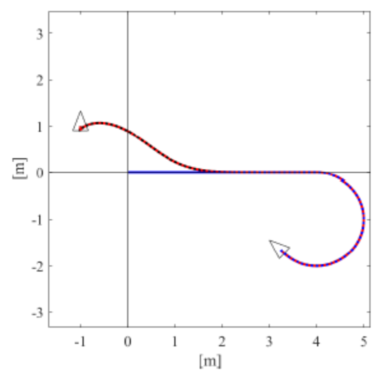
CST GPS



CST R-K



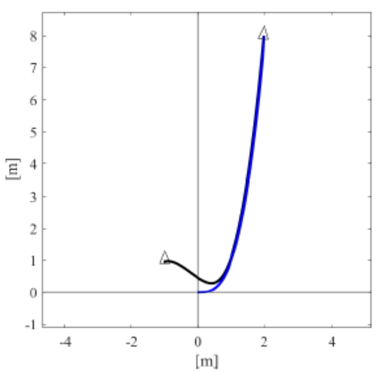
RAS 2 GPS



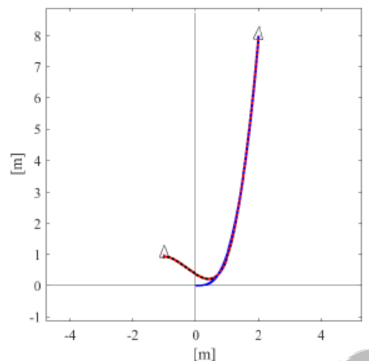
RAS 2 R-K

Simulations

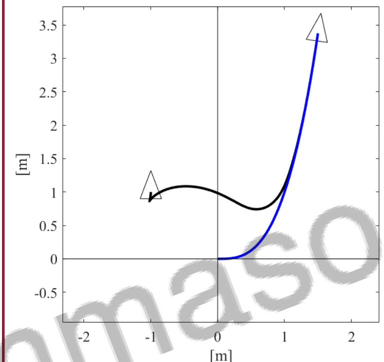
Cubic trajectory



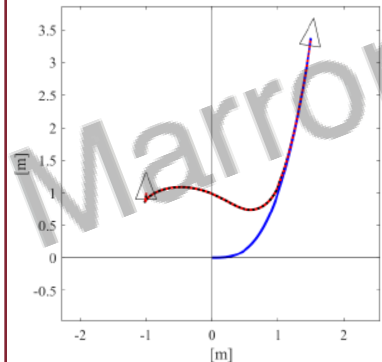
Lin. Approx. GPS



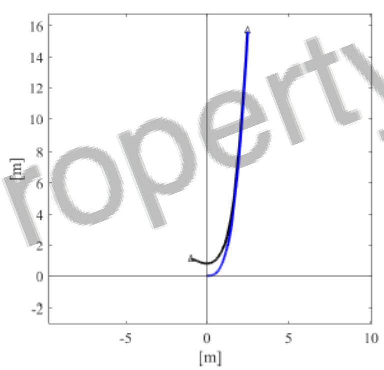
Lin. Approx. R-K



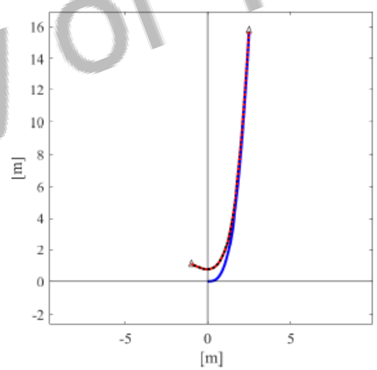
RAS 1 GPS



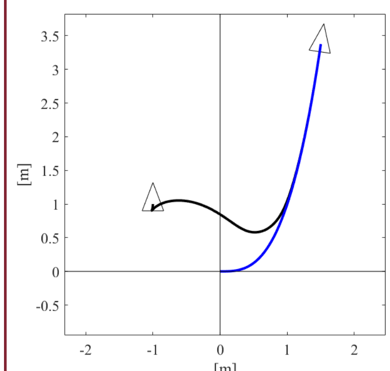
RAS 1 R-K



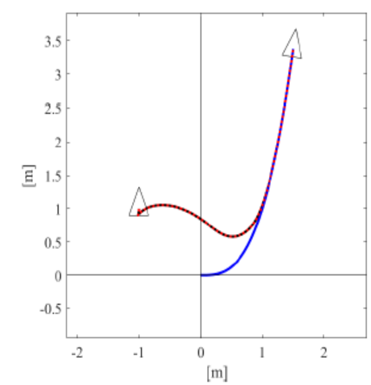
CST GPS



CST R-K



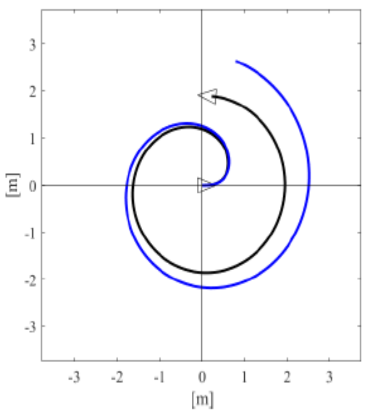
RAS 2 GPS



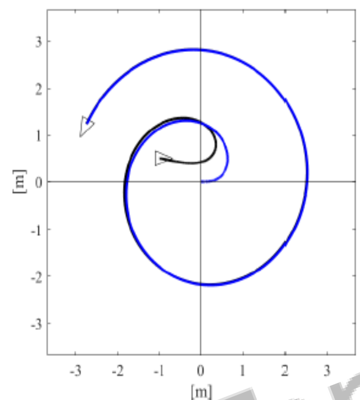
RAS 2 R-K

Simulations

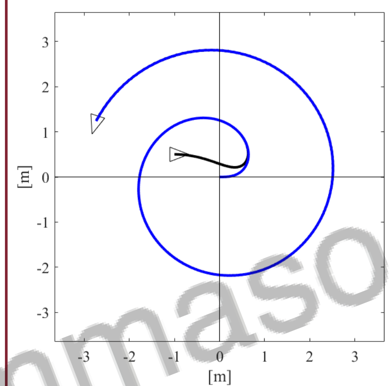
Spiral trajectory



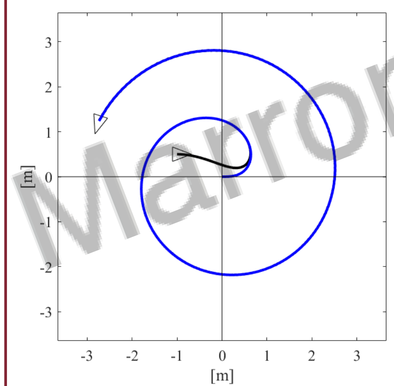
Lin. Approx. GPS



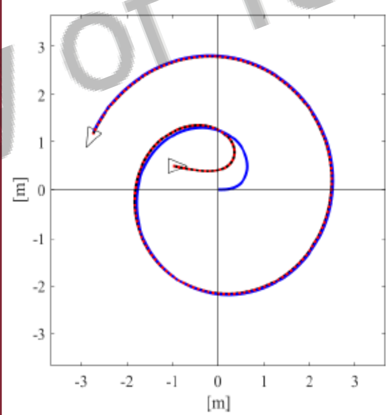
CST GPS



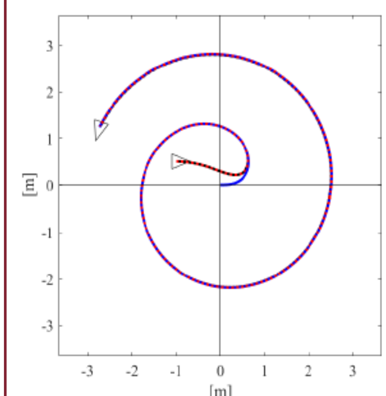
RAS 1 GPS



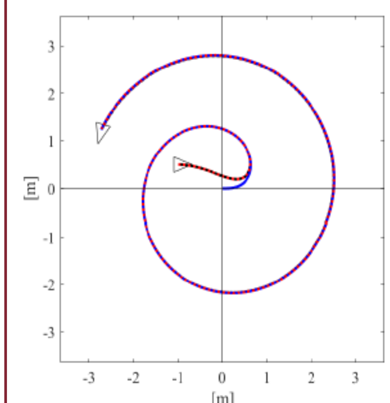
RAS 1 GPS



CST R-K



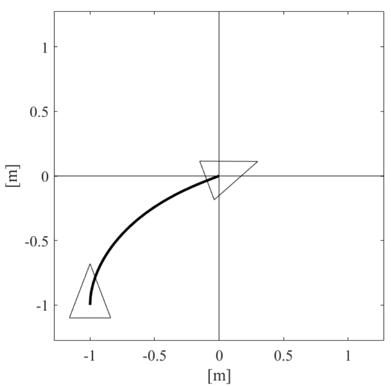
RAS 1 R-K



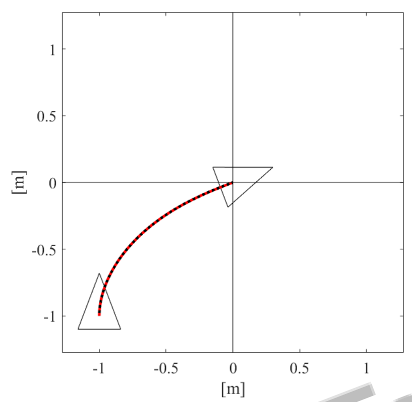
RAS 2 R-K

Simulations

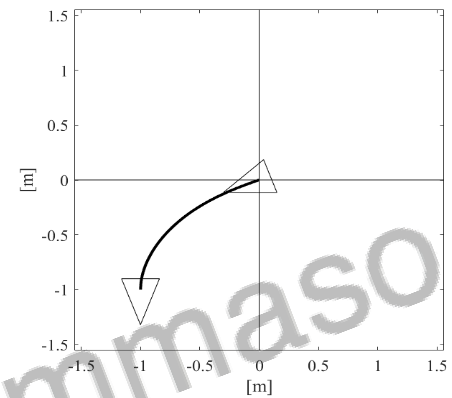
Cartesian Regulation



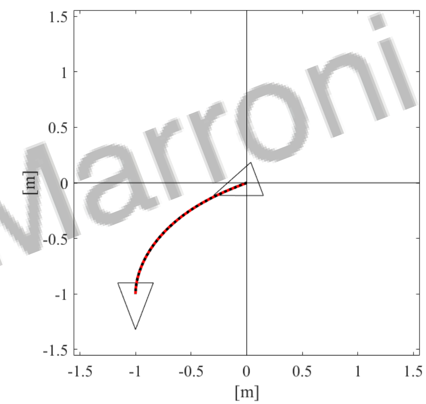
Cart. Reg. FWD GPS



Cart. Reg. FWD R-K



Cart. Reg. BWD GPS

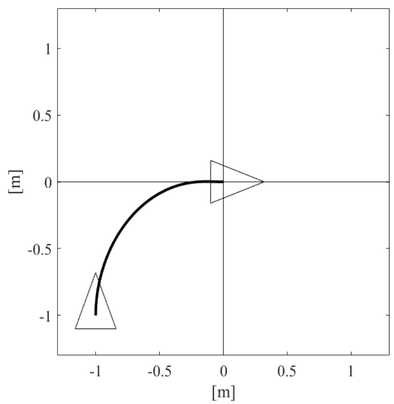


Cart. Reg. BWD R-K

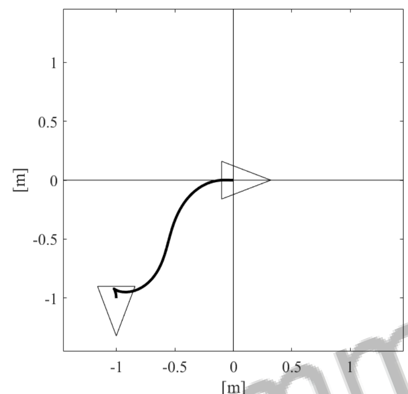
- To avoid useless reversals and reach desired configuration in FWD or BWD motion:
  - Add the term $-\pi sign(atan2(y, x) - \theta + \pi)$ to ω when $e_p^T n \leq 0$.
- To impose only FWD or BWD motion:
  - Multiply v by $sign(e_p^T n)$.
  - Multiply v by $-sign(e_p^T n)$ and add π to ω.

Simulations

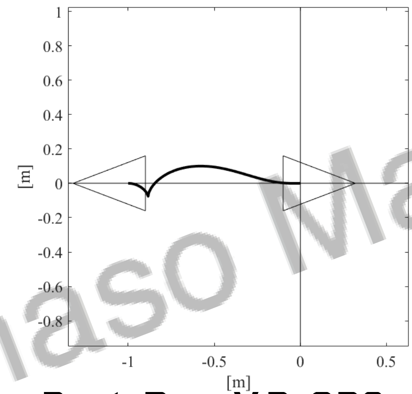
Posture Regulation



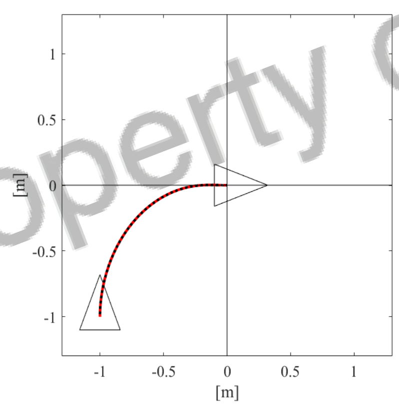
Post. Reg. FWD GPS



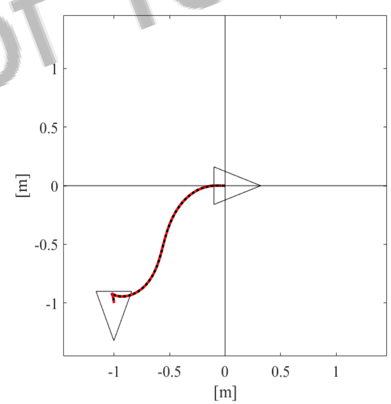
Post. Reg. BWD GPS



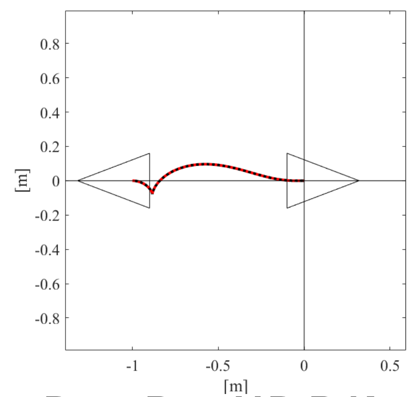
Post. Reg. V.P. GPS



Post. Reg. FWD R-K



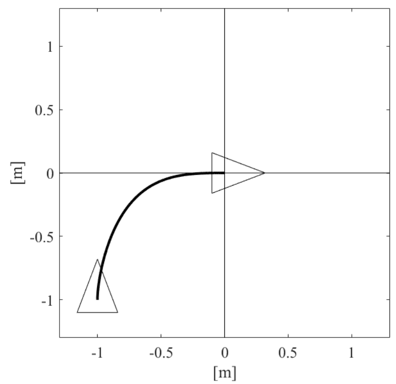
Post. Reg. BWD R-K



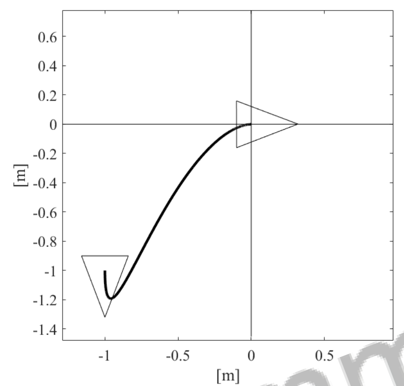
Post. Reg. V.P. R-K

Simulations

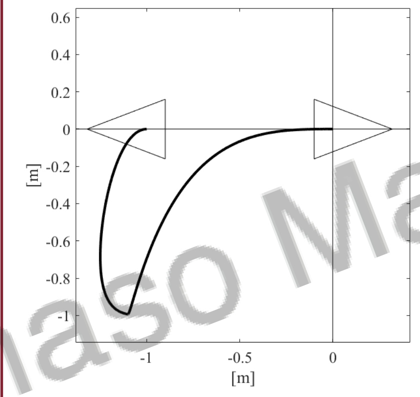
CST Regulation



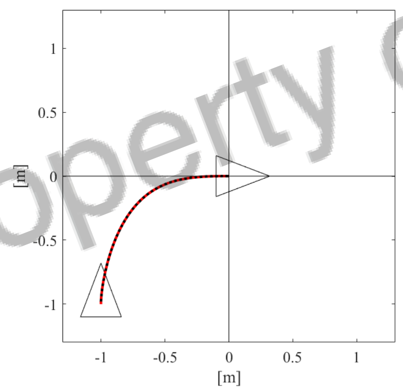
CST Reg. FWD GPS



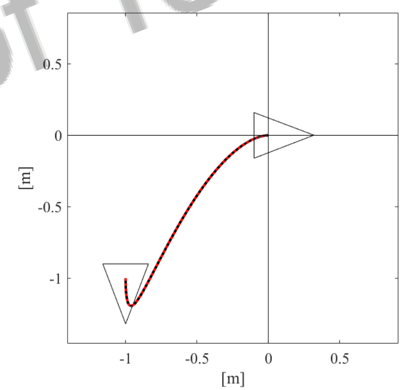
CST Reg. BWD GPS



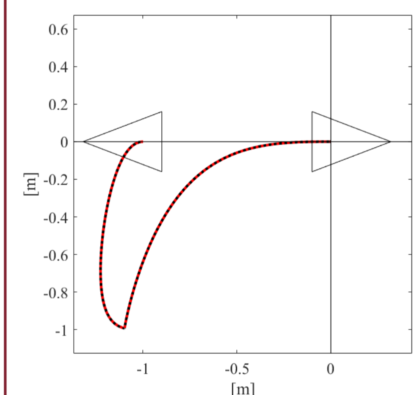
CST Reg. V.P. GPS



CST Reg. FWD R-K



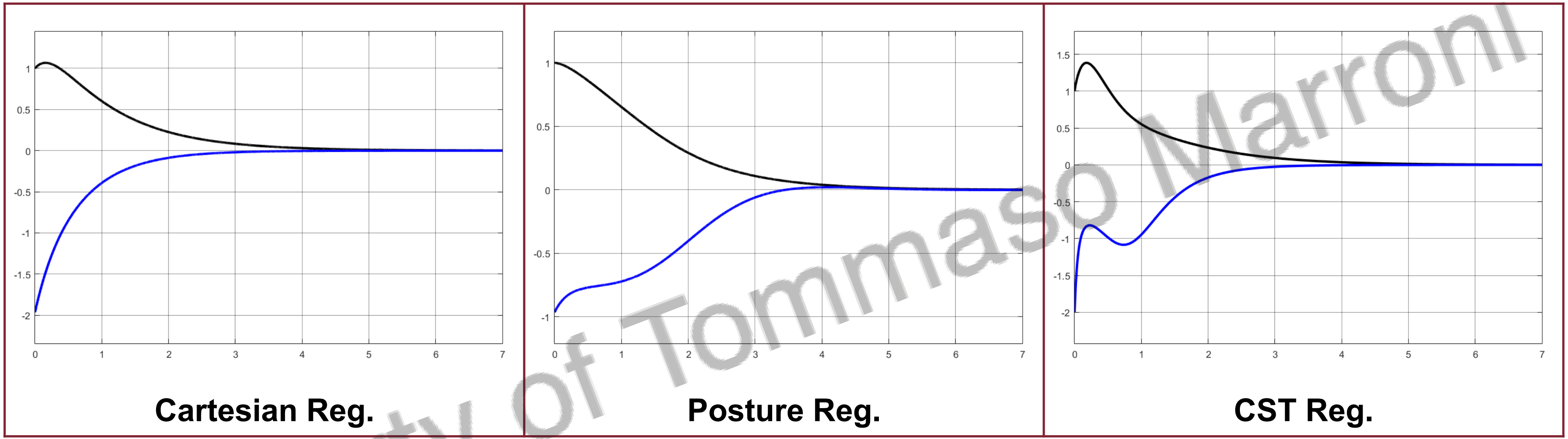
CST Reg. BWD R-K



CST Reg. V.P. R-K

Simulations

Control actions



blue: ω
black: v

Additional Notes

Issue n° 1 [Linear Approximation]

Problem regarding the implementation of $\theta_d - \theta$.

Solution: remove *atan2* and implement a new method to compute angle differences.

- Set $n_d = \begin{pmatrix} \cos\theta_d \\ \sin\theta_d \end{pmatrix}$ and $n = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}$.
- $n \cdot n_d = \cos\alpha \quad \Rightarrow \quad \alpha = \arccos(n \cdot n_d)$
- The sign of α is obtained from the z-component of the vectorial product between n and n_d.
- Final equation: $\alpha = sign(n \times n_d)_z \cdot \arccos(n \cdot n_d)$

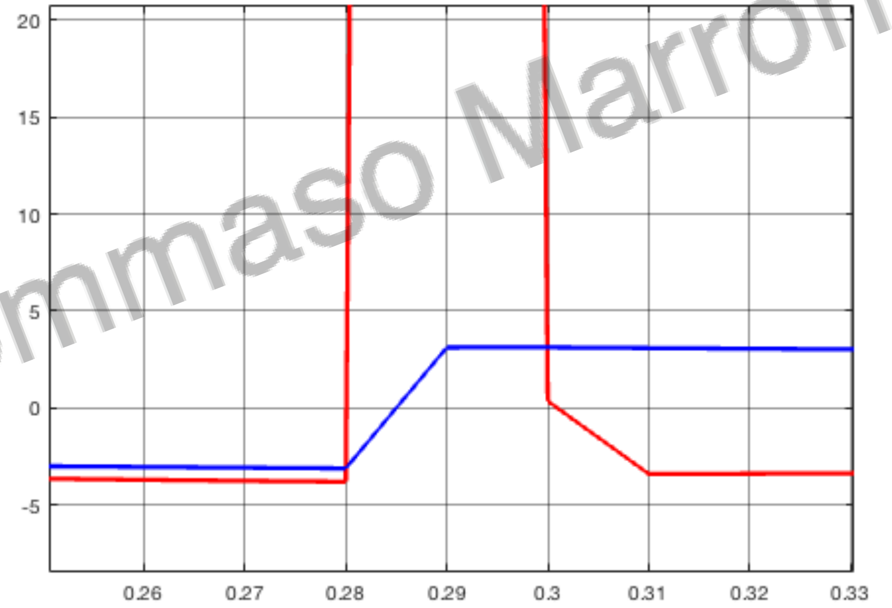
Same problem has been faced with RAS controller ($\theta_e - \theta$, θ_e and θ_t).

Additional Notes

Issue n° 2 [RAS]

Both controllers (RAS 1 and 2) use $\dot{\theta}_e$ and $\dot{\theta}_t$.

<u>Problem</u>: when the angle passes from $-\pi$ to π (or from π to $-\pi$) the derivative grows (decreases) excessively.

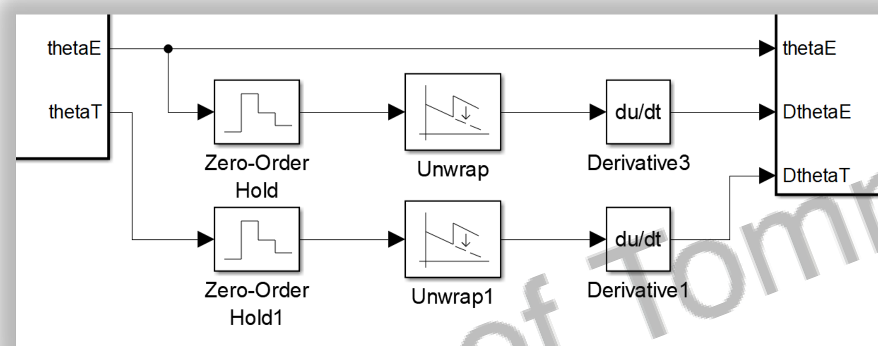


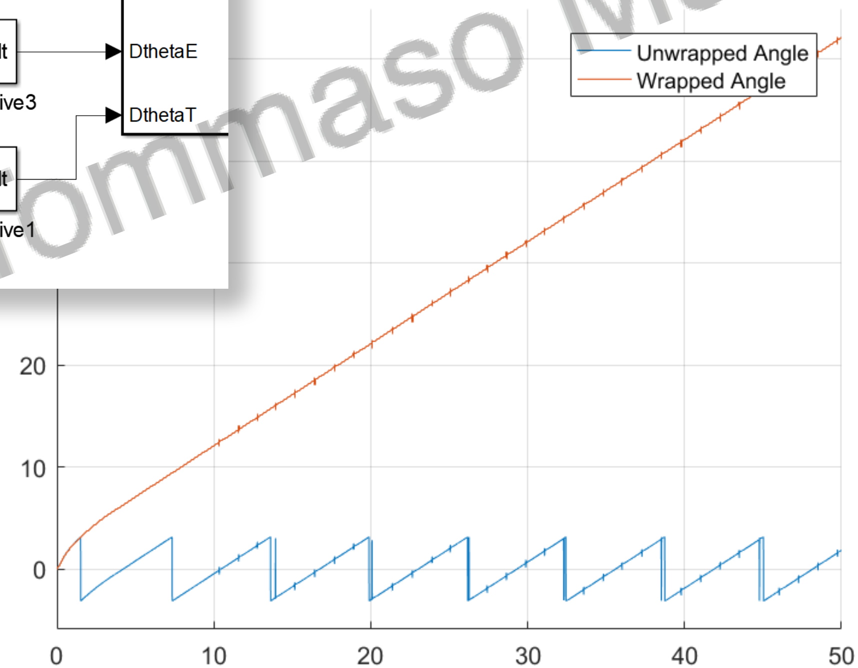
RED: $\dot{\theta}_e$, BLUE: θ_e

Additional Notes

Issue n° 2 [RAS]

<u>Solution</u>: unwrap the angles whose derivative must be computed.





Additional Notes

Issue n° 3 [RAS]

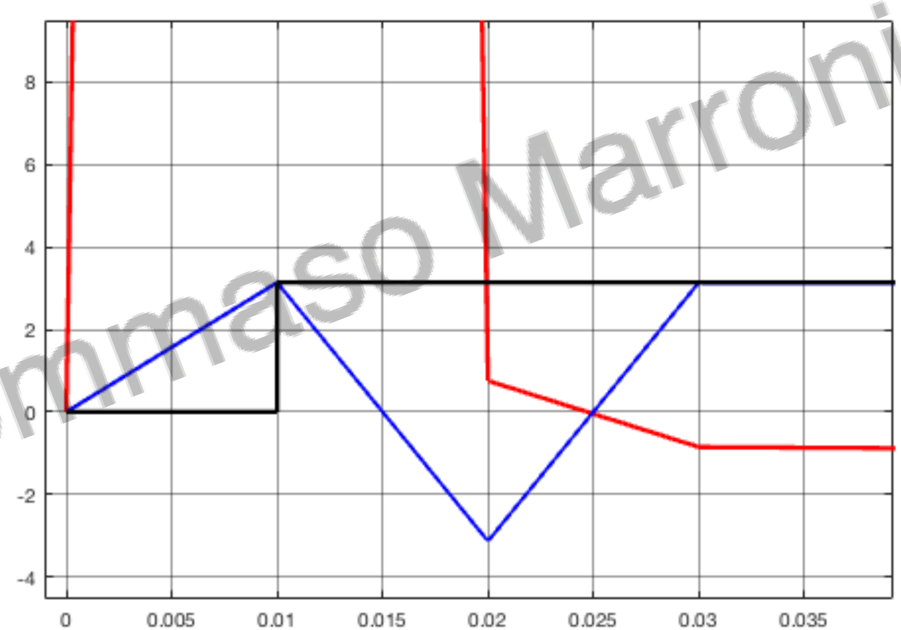
- Problem: ZOH mantains, for the first T_s seconds, the angle θ_e (or θ_t) equal to zero.

↓

Unjustified peak in the derivative term.

↓

Wrong initial control action.



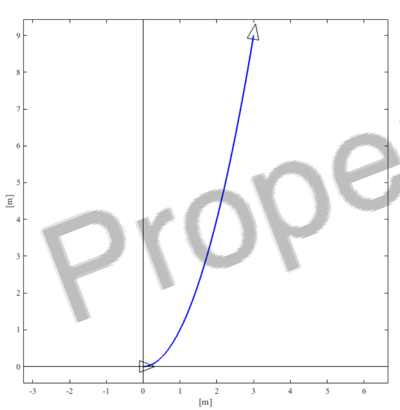
RED: $\dot{\theta}_e$, BLUE: θ_e before unwrap, BLACK: θ_e

- Solution: Neglect the derivative term for the first $2T_s$ seconds.

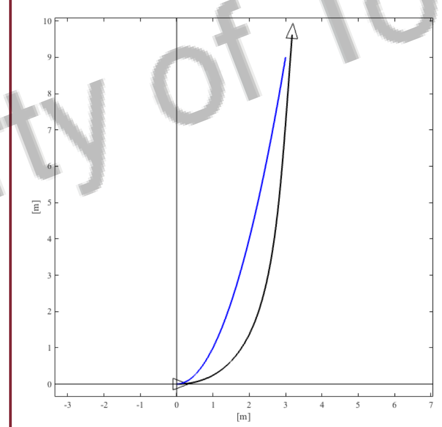
Additional Notes

Issue n° 4 [CST]

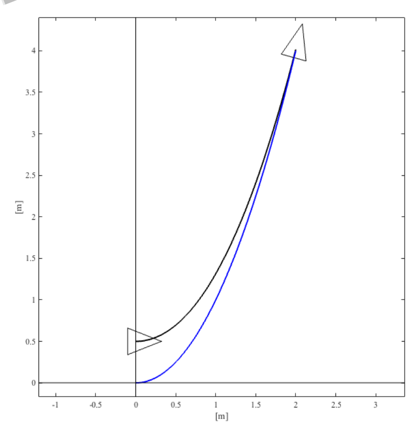
- In general, using <u>derivative blocks</u> in Simulink can lead to complications.
- In particular, in CST:
  - Robot doesn't converge in case of spiral trajectory;
  - Phase shift at steady state in circular trajectory (and steady state error);
  - Longer transients;
  - Less effective control actions;



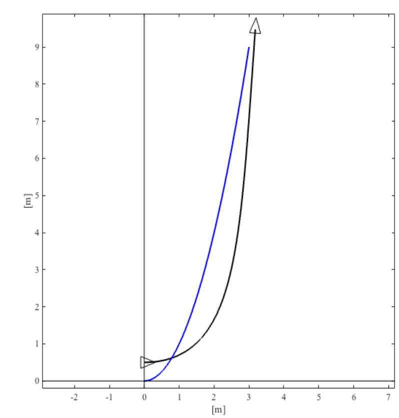
No Deriv. Blocks



Deriv. Blocks

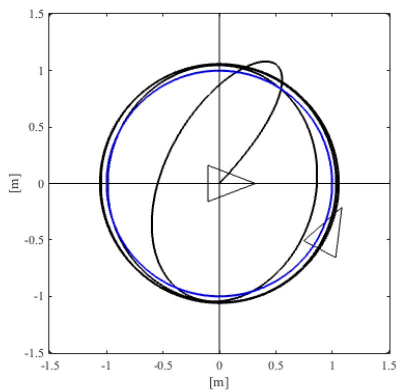


No Deriv. Blocks

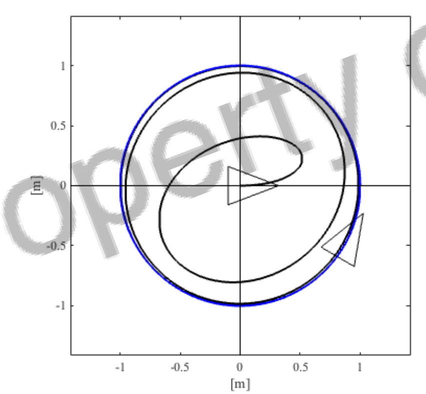


Deriv. Blocks

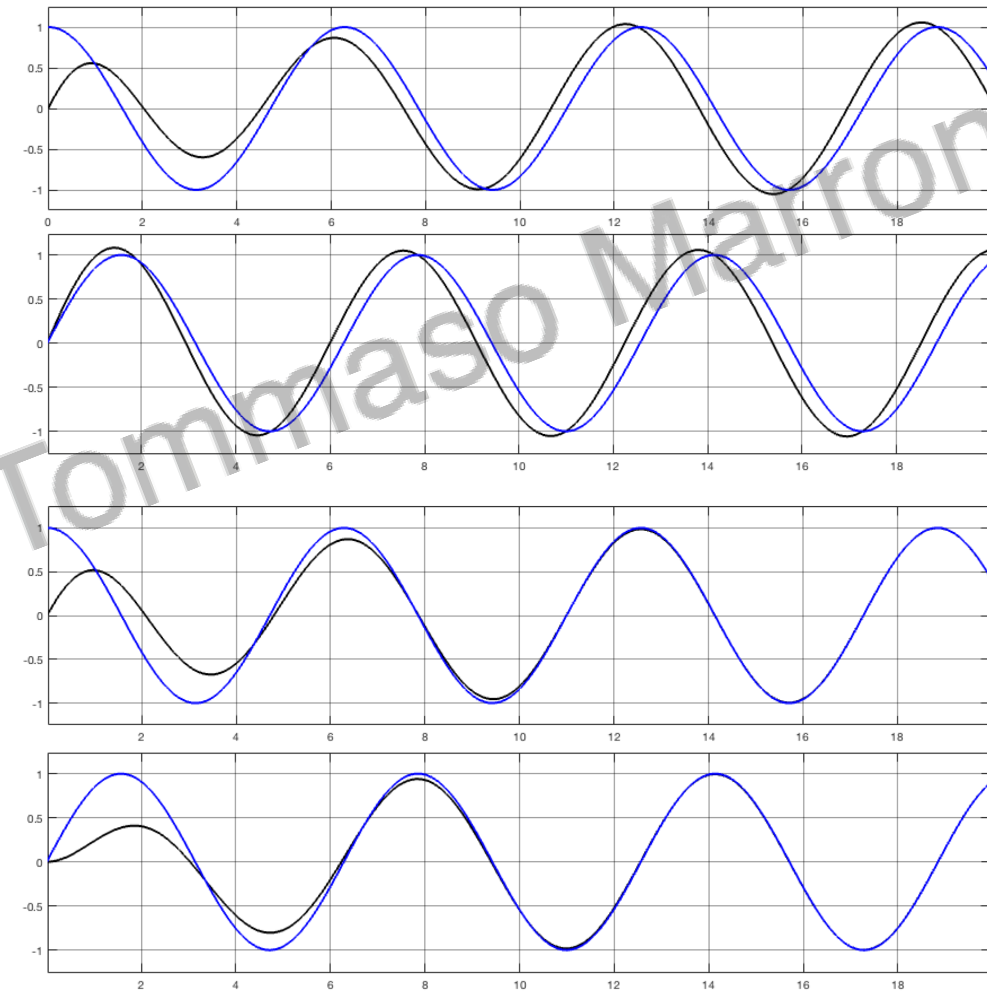
Additional Notes



Deriv. Blocks



No Deriv. Blocks

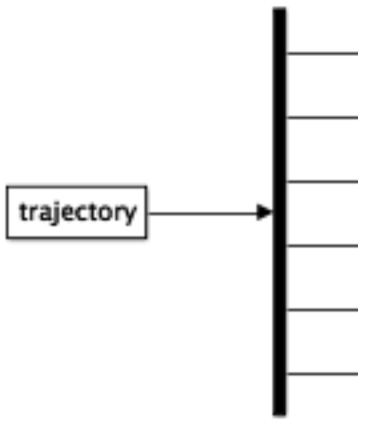


Additional Notes

Issue n° 4 [CST]

```
syms t xd yd Dxd Dyd DDxd DDyd;

%example
xd = sin(t);
yd = cos(t);

Dxd = diff(xd, t);
Dyd = diff(yd, t);
DDxd = diff(Dxd, t);
DDyd = diff(Dyd, t);

t = (0:Ts:T)';

xd = double(subs(xd));
yd = double(subs(yd));
Dxd = double(subs(Dxd));
Dyd = double(subs(Dyd));
DDxd = double(subs(DDxd));
DDyd = double(subs(DDyd));

trajectory.time = t;
trajectory.signals.values = [xd, yd, Dxd, Dyd, DDxd, DDyd];
trajectory.signals.dimensions = 6;
```

- Solution: compute derivatives mathematically using symbolic variables.



- N.B: we have noticed that, in general, using the derivative blocks worsens the performances of all the controllers (also Linear Approximation and RAS)

Conclusion

- The goal of this project was to show the efficiency of the previous controllers through simulations.

- Controllers are able to follow the chosen trajectories.
  - The only exception is linear approximation controller which has some limitations.

- For a more accurate study of the mathematical properties and stability proofs, see [1] and [2].

References

[1] Luigi Freda, Giuseppe Oriolo, *Vision-based interception of a moving target with a nonholonomic mobile robot*, Elsevier, Robotics and Autonomous Systems 55 (2007) 419-432.

[2] Giuseppe Oriolo, Alessandro De Luca, *WMR Control Via Dynamic Feedback Linearization: Design, Implementation, and Experimental Validation*, IEEE Transactions on Control Systems Technology, Vol. 10, No. 6, 835-852, November 2002.

[3] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, Giuseppe Oriolo, *Robotics: Modelling, Planning and Control*, Springer, 3rd Edition, 2009.