CyberCarpet acceleration-level control of position

SAPIENZA
UNIVERSITÀ DI ROMA

Locomotion and Haptic Interfaces
Control Problems in Robotics A.A. 19-20

Adelina Selimi
1843827

Introduction

The CyberWalk platform systems are used for virtual reality applications, to keep the walker in a certain point while he/she is moving in the virtual reality.

Cyberwalk platforms:

- ball-array
- belt-array

The ball-array will be here considered (CyberCarpet).

Aim: positioning the user at the center of the platform satisfying different constraints and using a control law at acceleration level.

The user does not have to feel the motion of the platform, as much as possible.

CyberCarpet platform

The CyberCarpet platform is a ball array platform.

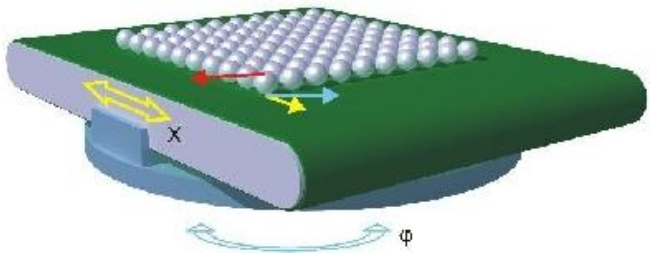
Composition:

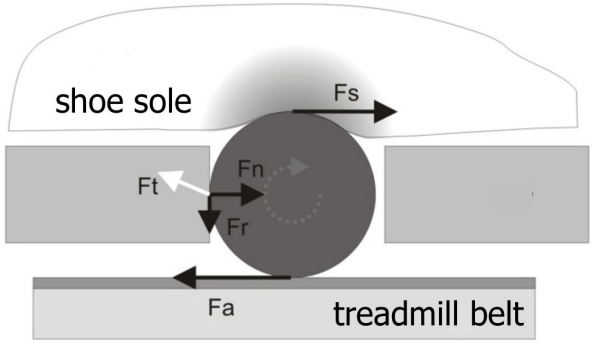
- treadmill belt, bidirectional
- turntable which moves on the main axes
- grid with an array of balls

When the treadmill moves, a force is applied to each sphere on the grid, an equivalent force is then applied on the top of the sphere with an opposite direction.

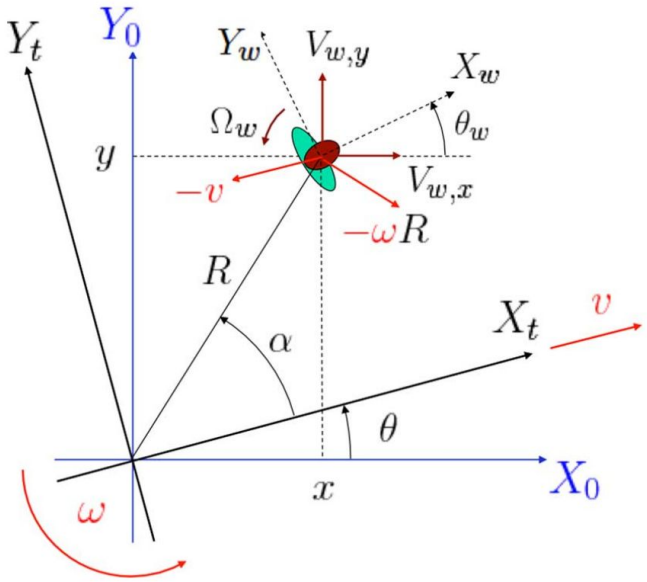
In order to transmit this force:

- high friction between the sphere and the treadmill belt
- low friction between the sphere and the grid.





Kinematic model



$$\dot{x} = -v\,cos(\theta) + y\,\omega + V_{w,x}$$
$$\dot{y} = -v\,sin(\theta) - x\,\omega + V_{w,y}$$
$$\dot{\theta} = \omega$$
$$\dot{\theta}_w = -\omega + \Omega_w$$
$$\dot{v} = a$$
$$\dot{\omega} = a_\omega$$

Applying a velocity v to the treadmill, the human will move in the opposite direction due to the balls.

Applying a velocity ω to the treadmill, the human will change orientation in the opposite direction due to the balls.

Notice: $\theta + \theta_w$ = constant if the walker is standing still

They cannot be controlled independently and for this reason the variable θ_w is taken off from the control problem.

Disturbance observer

$$\dot{x} = -v\,cos(\theta) + y\,\omega + V_{w,x}$$
$$\dot{y} = -v\,sin(\theta) - x\,\omega + V_{w,y}$$
$$\dot{\theta} = \omega$$
$$\dot{v} = a$$
$$\dot{\omega} = a_{\omega}$$

The walker velocities along x and along y can be considered as disturbances to the platform.

The intentional walker velocity is unknown obviously but two disturbance observers are implemented to estimate these two velocities.

$$\dot{\xi}_x = -v\,cos\theta + y\,\omega + k_w(x - \xi_x)$$
$$\tilde{V}_{w,x} = k_w(x - \xi_x)$$

$$\dot{\xi}_y = -v\,cos\theta - x\,\omega + k_w(y - \xi_y)$$
$$\tilde{V}_{w,y} = k_w(y - \xi_y)$$

The two estimates use a copy of the platform system.

The gains k_w have to be high enough to assure a fast estimation.

Control specifications and constraints

Control specifications:

- keep the walker close to the center of the platform (0, 0)
- The absolute orientation of the walker instead does not matter
- control at acceleration level

Control constraints:

- perceptual constraints on the velocities
- bounds on the achievable velocities
- bounds on the acceleration inputs

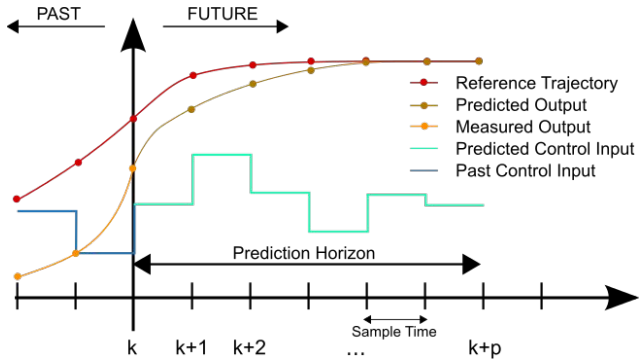
The output of the system is considered equal to the state vector, since all the states are considered accessible.

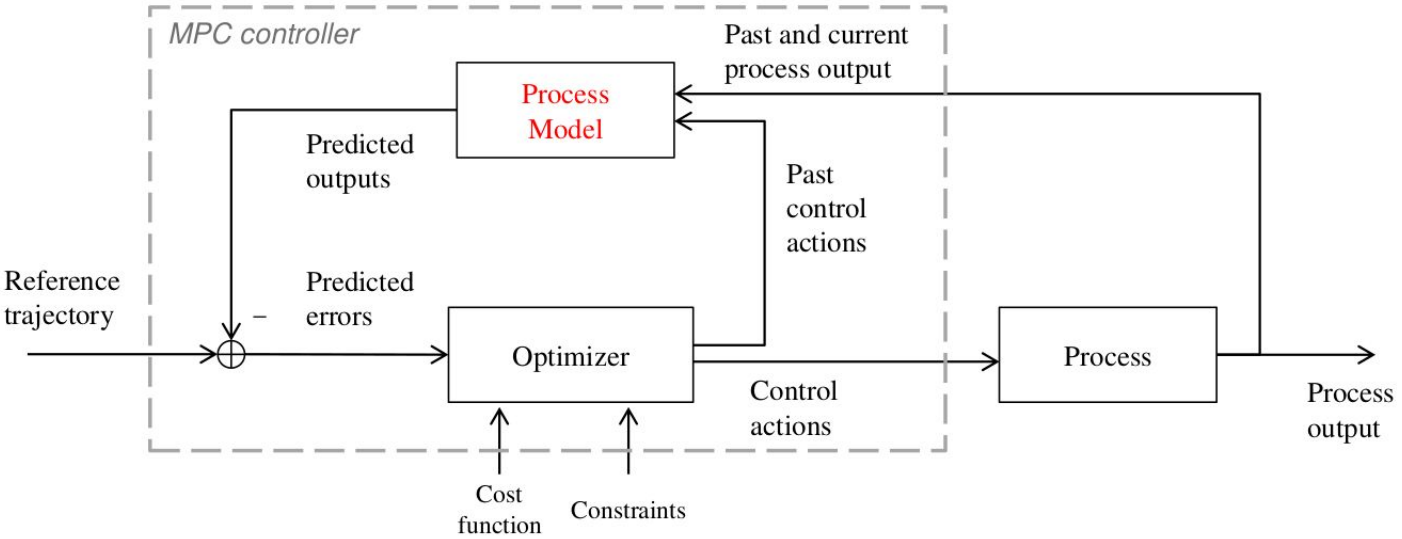
Why control at acceleration level?

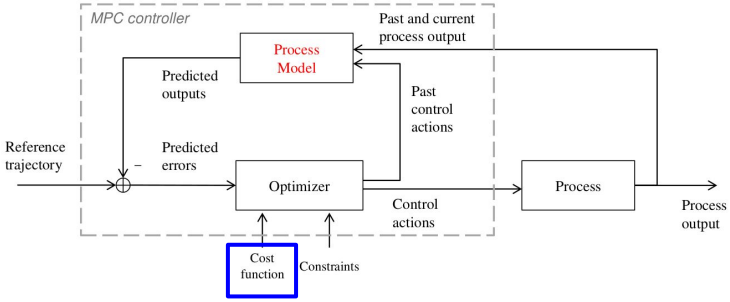
- softer transient
- no discontinuities in velocity
- analysis of the dynamic effects on the walker is straightforward, which allows to study also perceptual constraints on the acceleration

Nonlinear model predictive control

The presence of the perceptual constraints and the knowledge of an appropriate model of the system suggest the possibility of implementing an optimization problem. It can be done for example by designing a model predictive control.





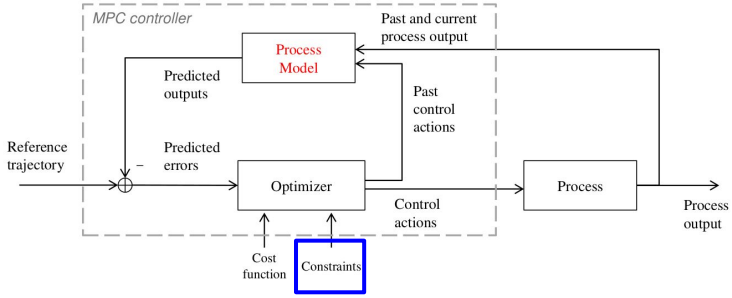


At a time instant the plant is sampled and, through a minimization of the cost function, a control sequence is calculated. Only the first step of the control strategy is implemented, then the procedure repeats by sampling again the new state and calculating the new control inputs and the new predicted outputs.

One of the most used cost function is a quadratic cost function:

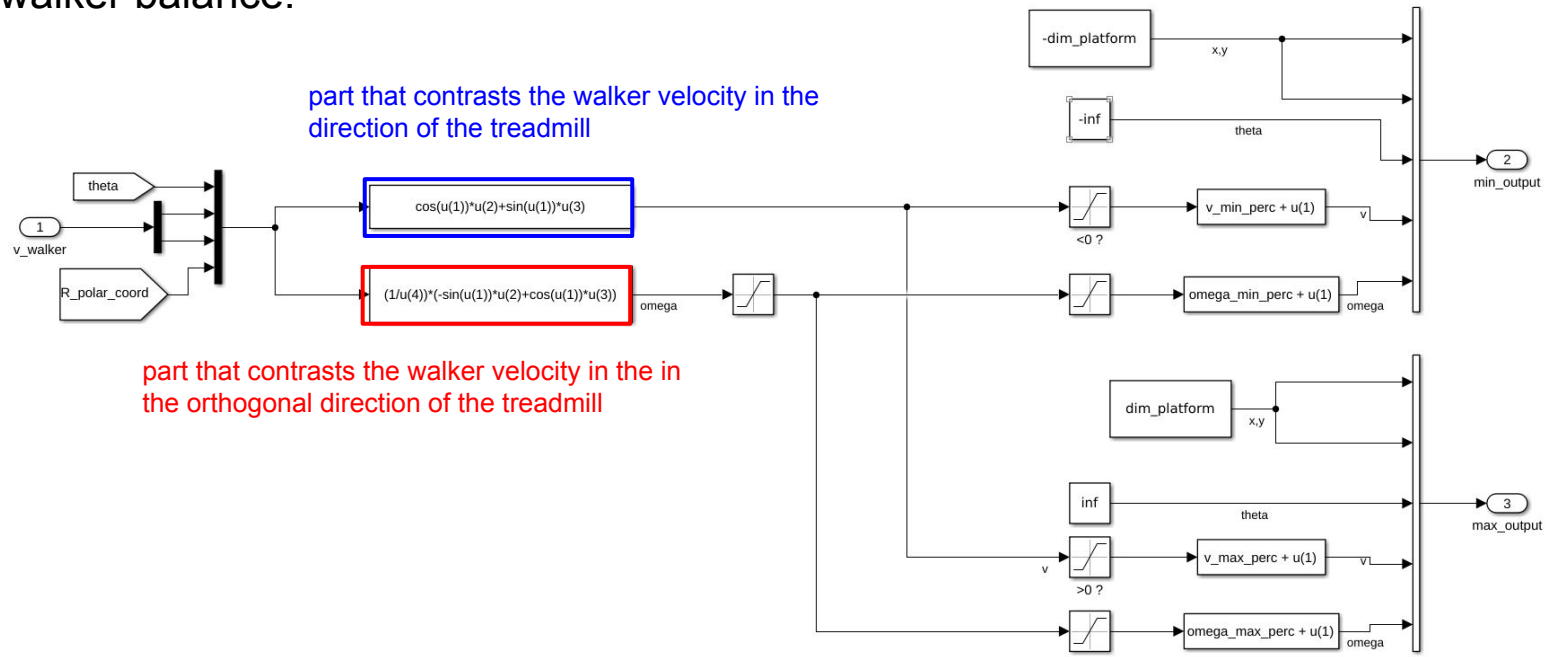
$$J = \sum_{i=1}^{N} w_{x_i}(r_i - x_i)^2 + \sum_{i=1}^{M} w_{\Delta u_i}(\Delta u_i)^2 + \sum_{i=1}^{M} w_{u_i}(u_i)^2$$

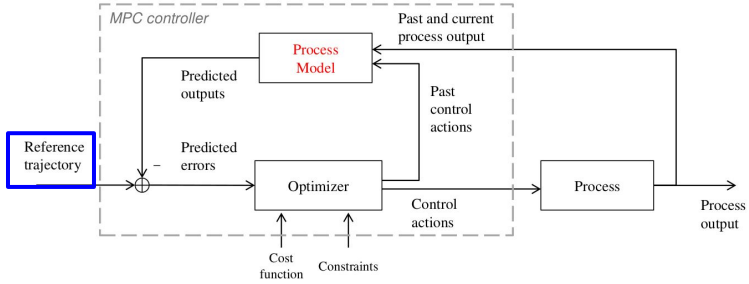
- w_x : weighting coefficients for the output variables [x, y, θ, v, ω], since θ does not have to be controlled its coefficient can be zero. Moreover since the main objective is to obtain (x, y) equal to (0, 0), their coefficients can be imposed higher than the ones for the velocities;
- w_u : weighting coefficients for the input variables [a, a_ω], useful to avoid large inputs;
- $w_{\Delta u}$: weighting coefficients for the input variables used to penalize their rate of change.



The constraints on the accelerations are implemented as bounds of the controller outputs. The constraints on the velocities are related to the perceptual constraints.

Notice: the perceptual constraints are applied to the velocities except for the part of the control that contrasts the walker velocity since it does not influence the walker balance.

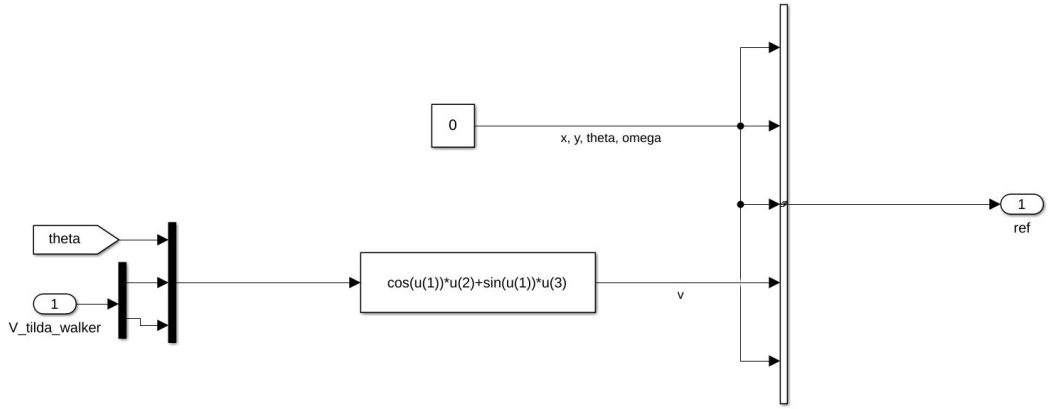




The NMPC needs the values of reference for the outputs.

- Position: (0, 0);
- Linear velocity: the values depend on the user. Indeed if the user is not walking and standing still, then the final desired velocities are equal to zero. But if the user is walking, the controller has to contrast this velocity and this can be done by imposing as references;
- Angular velocity: it is set to zero but if the walker is changing the orientation (Ω_w nonzero) then the final ω will not be zero, it will assume a value close to Ω_w , violating the reference

$$v_r = \begin{bmatrix} cos\theta & sin\theta \end{bmatrix} \tilde{V}_w$$
$$\omega_r = 0$$



Comparison with cascaded control design

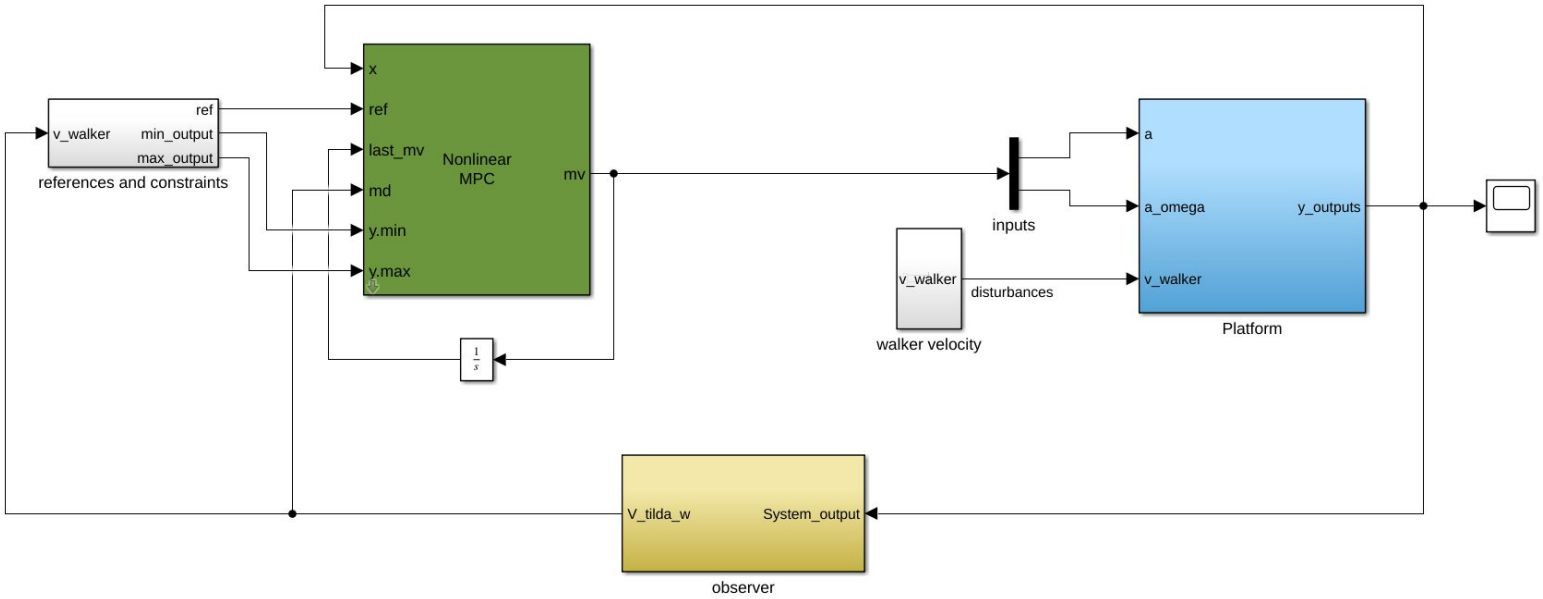
Main differences and similarities between the input output linearization control (expanded to acceleration level using a cascaded system) and the NMPC:

- The cascaded second order control law is easy to implement and only the gains have to decided, the NMPC instead has more parameters to be defined;
- In the NMPC the control problem assumes a form of optimization control problem and at each step it has to be solved, more computation is needed;
- Singularities and chattering problems had to be discussed before implementing the cascaded law;
- In the cascaded control a scaling factor for the gains had to be introduced in order not to violate perceptual constraints. For NMPC these constraints are introduced simply as constraints of the optimization problems;

- Both the controllers need an accurate model of the system;
- Both need to estimate the intentional walker velocity;
- Both the controller tend to align the platform with the walker velocity direction.

Simulation

A simulation of the overall system has been implemented using Simulink. This simulation is composed mainly by three blocks, which are the one for the kinematic system of the platform, the observer for the walker velocities and the last one is the nonlinear model predictive controller.



Dimension of the platform: 3 × 3 (m × m).

Perceptual constraints: v_{max} = 0.5 m/s and ω_{max} = 0.1 rad/s.

Bounds on the inputs: a_{max} = 4 m/s^2 and $a_{\omega,max}$ = 4 rad/s^2 .

The dark green block is the nonlinear MPC. It is a block defined in the library *Model Predictive Control Toolbox* of Matlab, and its parameters have been defined in a Matlab script creating a *nlobj* object. The most important choice of parameters used in the following simulations are summarized in the figure below:

```
%%
% Sample time and the prediction and control horizons of the controller
nlobj.PredictionHorizon = 10;
nlobj.ControlHorizon = 2;
nlobj.Ts = 0.5;
%%
% Objective function coefficients
nlobj.Weights.ManipulatedVariables = [0.01 0.01];
nlobj.Weights.ManipulatedVariablesRate = [0.1 0.1];
nlobj.Weights.OutputVariables = [10 10 0 1 1];
%%
% Constraints on acceleration inputs
nlobj.MV = struct('Min',{-4;-4},'Max',{4;4});
```

The solver to use for the nonlinear optimization has been used the default solver used by Matlab which is a sequential quadratic programming (**SQP**) algorithm. For the prediction and control horizon the choice is multiple. The prediction horizon value indicates how far the controller looks into the future and the control horizon indicates how many control moves have to be computed (for the rest of the prediction horizon the controller output is kept constant). The choice of these two values affect the computation effort.

Simulation results

In order to test the results obtained from this control scheme, four different simulations for different walking patterns have been done, changing also the constraints for some cases.

Notice: in all the simulations it is considered $\theta(0) = \theta_w(0)$ and that the walking patterns are defined in the absolute reference frame directions.

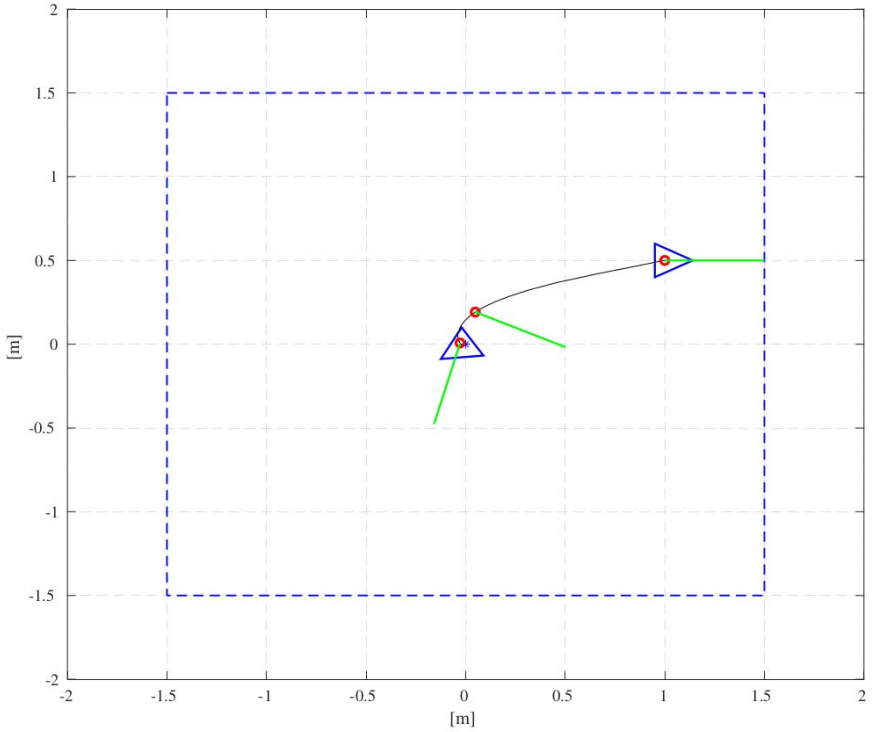
The walking patterns are:

1. Walker is standing still, off center position;
2. Constant walker velocity along x and along y:
   - $a_{max} = 0.6$ m/s^2 and $a_{\omega,max} = 0.6$ rad/s^2;
   - $a_{max} = 4$ m/s^2 and $a_{\omega,max} = 4$ rad/s^2.
   - ❏ Perceptual constraints $v_{max} = 0.5$ m/s and $\omega_{max} = 0.1$ rad/s;
   - ❏ Perceptual constraints $v_{max} = 0.9$ m/s and $\omega_{max} = 0.4$ rad/s.
3. The walker moves along a circular path;
4. The walker moves along a square path starting from the center
   - Ts = 0.5;
   - Ts = 0.2.

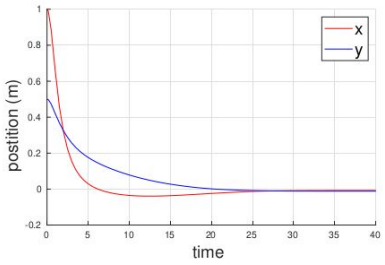
Scenario 1 - part 1

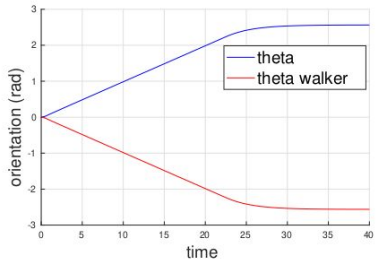
The initial position of the walker is: (1, 0.5)

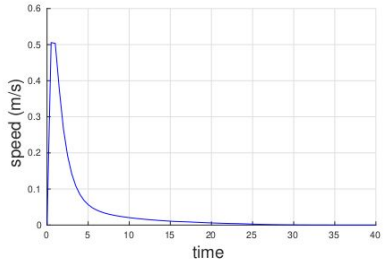
The two blue triangles are used to represent the initial and final positions and orientations of the walker in the platform. The blue square marks the limits of the platform. Then the red circle is the position of the walker for some instants of time and the green segment is used to show the orientation of the user

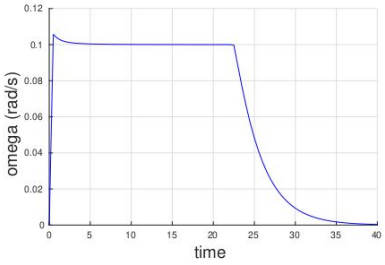


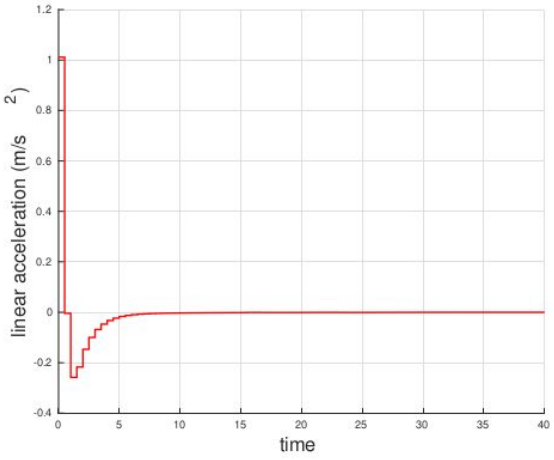
Scenario 1 - part 2

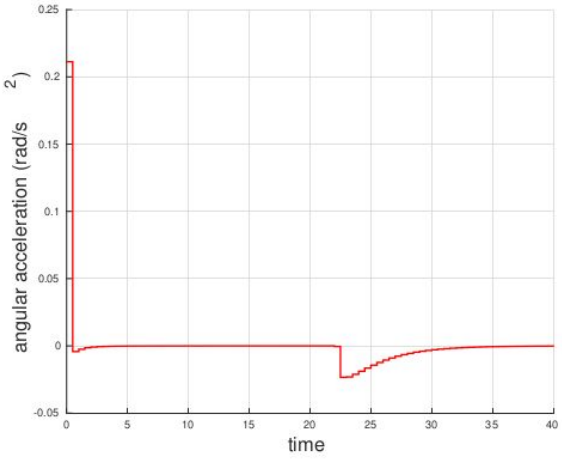






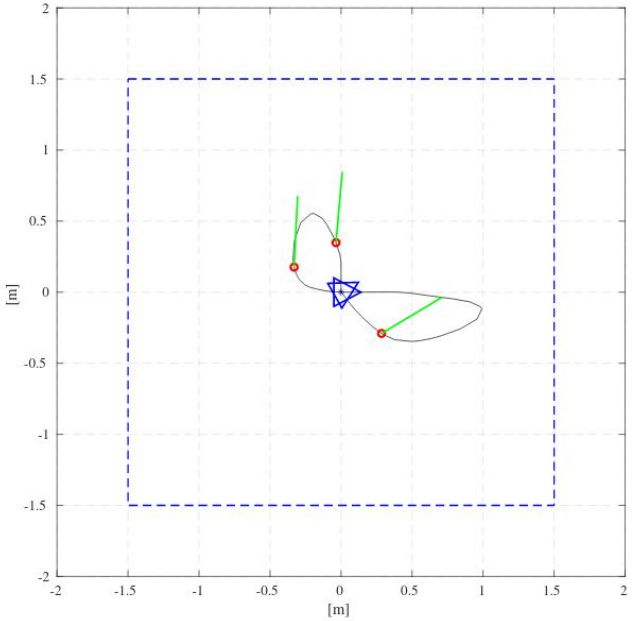


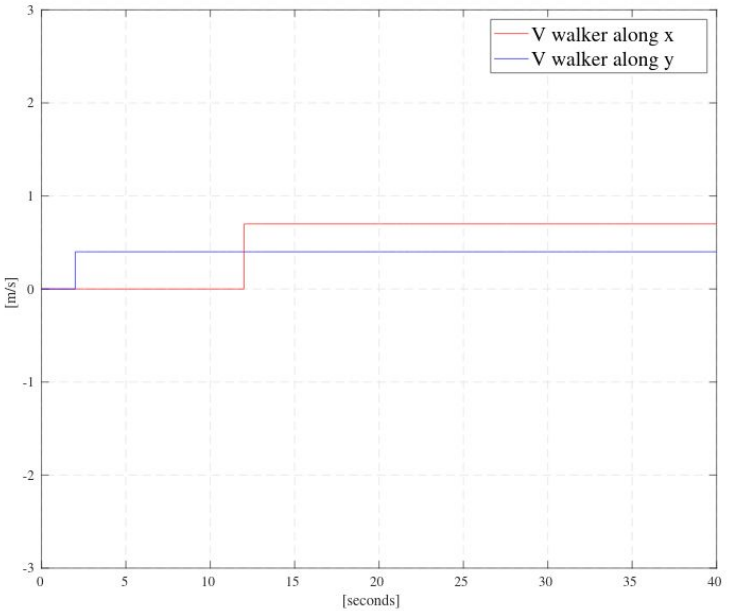




The walker is successfully recentered, while the orientation is not set to a particular reference since the final orientation does not matter.
The recentering takes the controller around ten seconds.

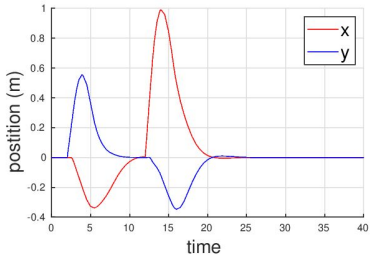
Scenario 2 - part 1

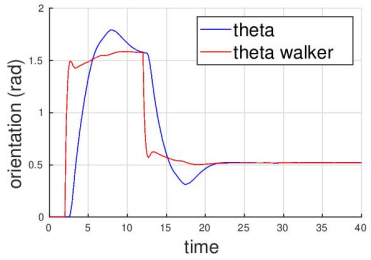


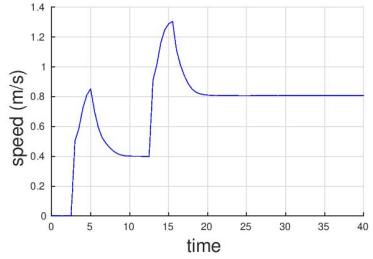


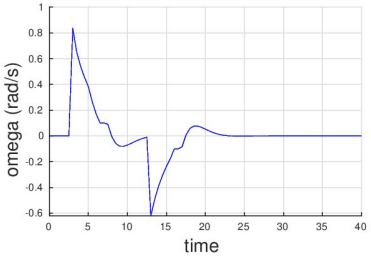
The walker is positioned at the platform center, after some seconds he/she starts moving with V_y = 0.4 m/s at time 2 seconds, at time 12 seconds with a velocity V_x = 0.7 m/s.

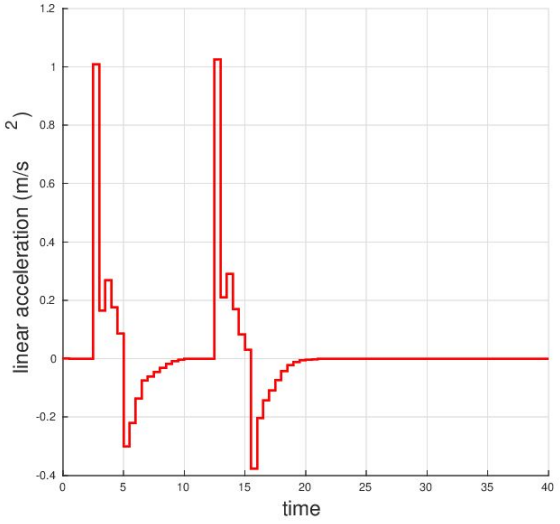
Scenario 2 - part 2

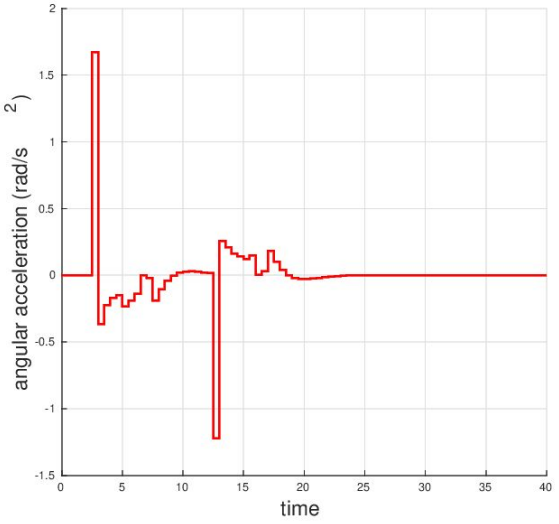










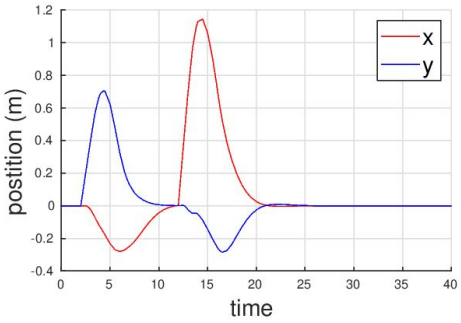


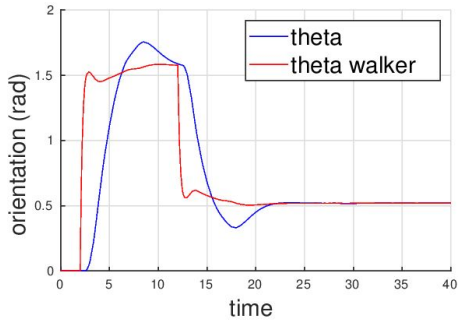
Scenario 2 - change of acceleration bounds

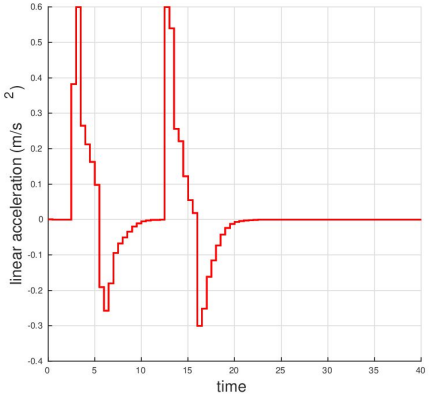
Consider now lower bounds for the accelerations:

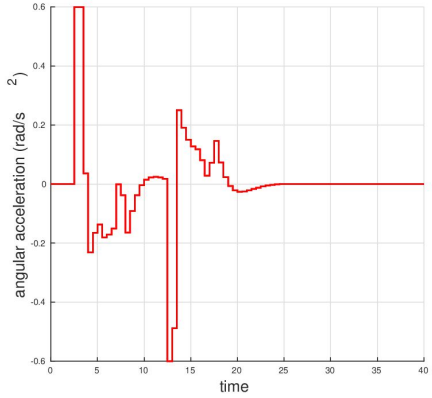
$a_{max} = 0.6\ m/s^2$ and $a_{\omega,max} = 0.6\ rad/s^2$

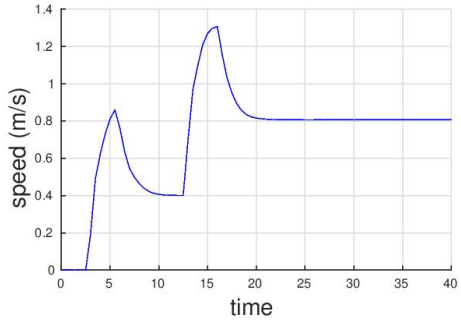
In this case, for the controller it takes (a little bit, ≈0.5 sec) longer to recenter the walker and he/she gets closer to the limits of the platform.

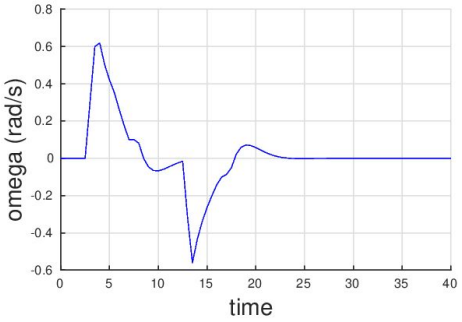




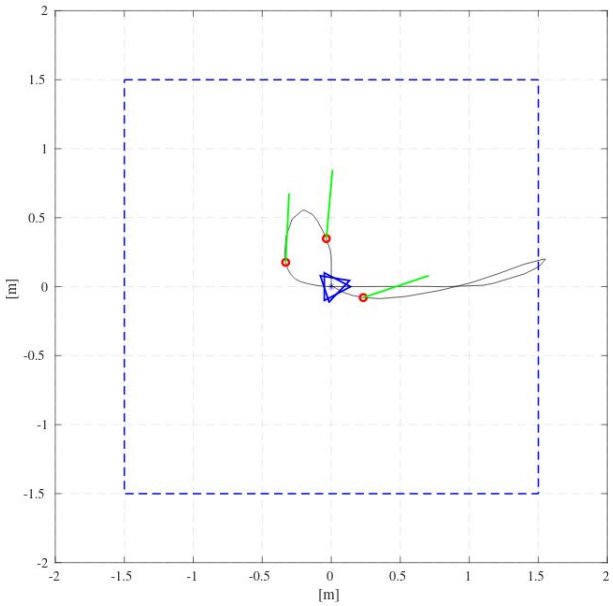




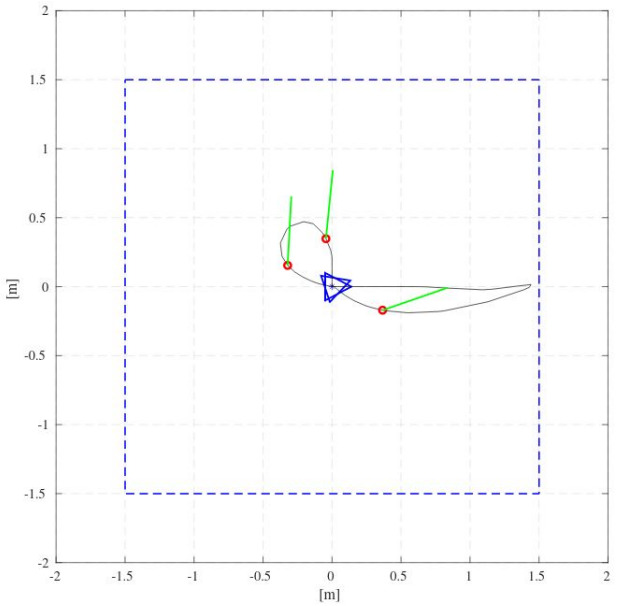




Scenario 2 - change of perceptual constraints



v_{max} = 0.5 m/s ω_{max} = 0.1 rad/s



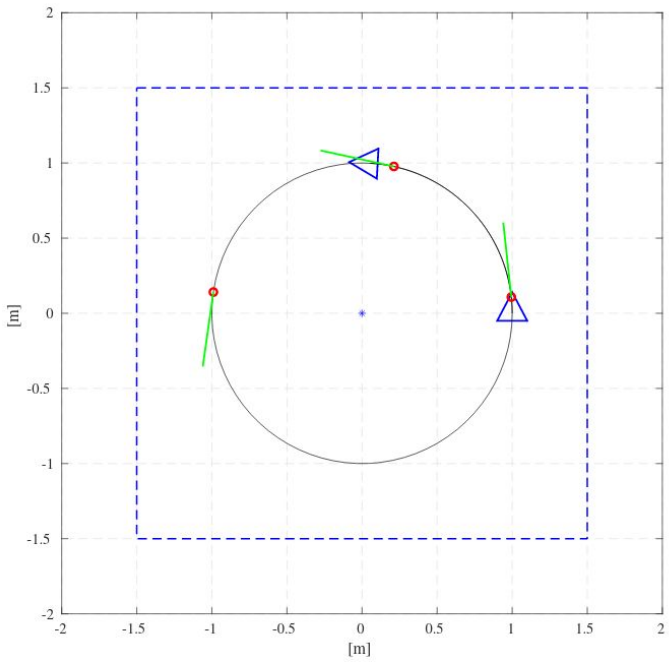
v_{max} = 0.9 m/s ω_{max} = 0.4 rad/s

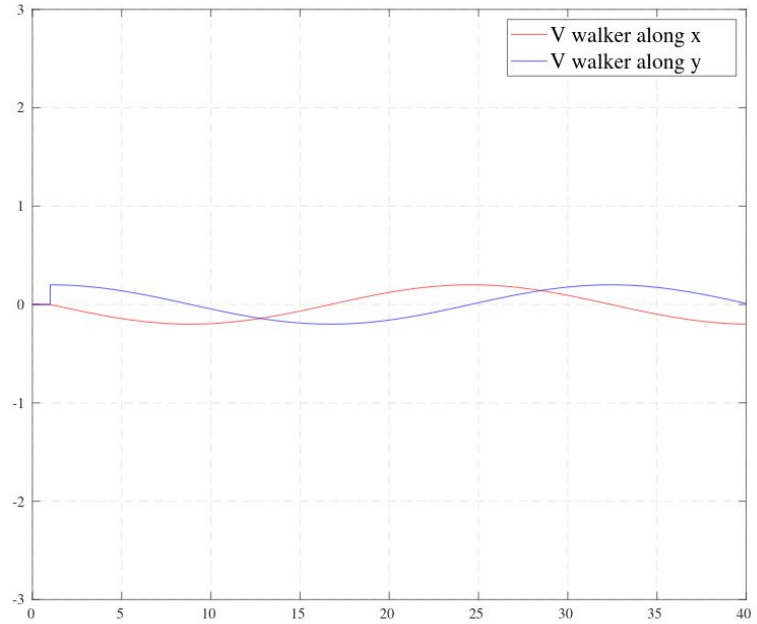
What happens if the walker velocity is too fast?

The walker may get off the platform limits. This depends on different factors, like the limits on the control inputs, too long sampling time and perceptual constraints.

In the figure above the velocities are: V_y = 0.4 m/s at time 2 seconds, at time 12 seconds with a velocity V_x = 1.2 m/s .

Scenario 3 - part 1

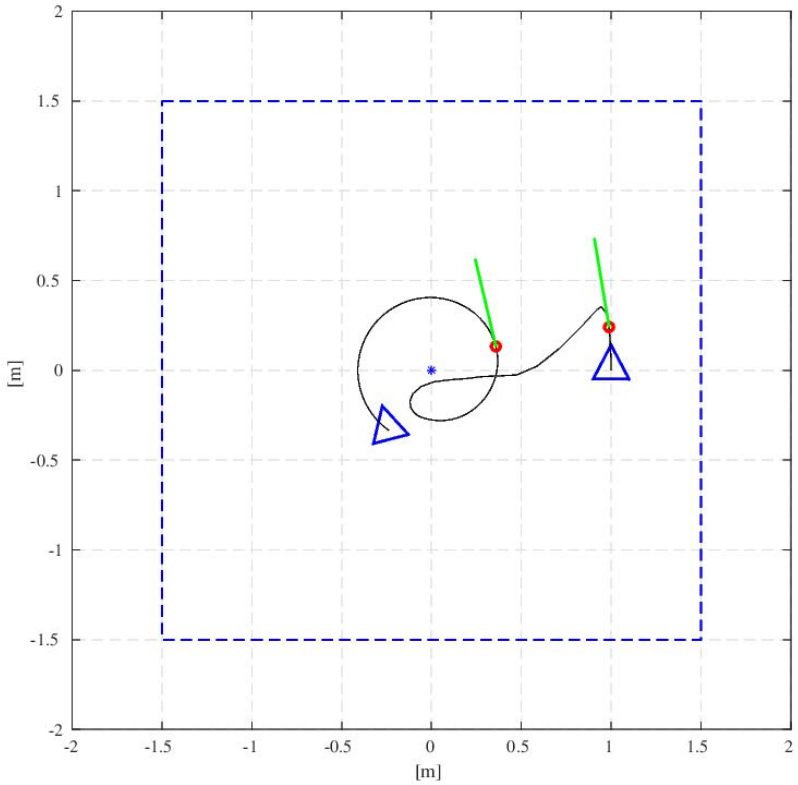




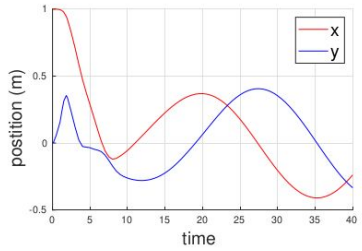
It is supposed that the radius of the circumference is r = 1 [m] and that the angular speed is Ω_w = 0.2 [rad/s]

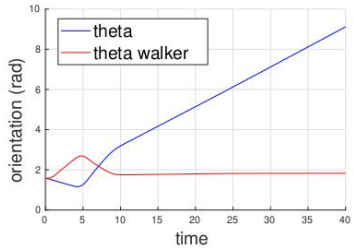
Scenario 3 - part 2

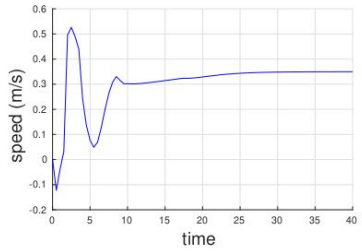
The control action keeps the walker closer to the center, reducing the radius of rotation. Indeed the walker is still going along a circular path, but thanks the action of the control inputs the radius is little. Changing the perceptual constraints, or with shorter sampling time for the controller the radius is even shorter.

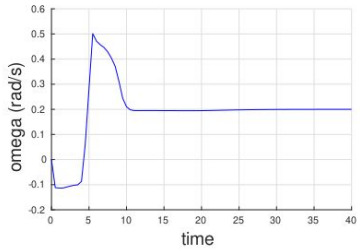


Scenario 3 - part 3

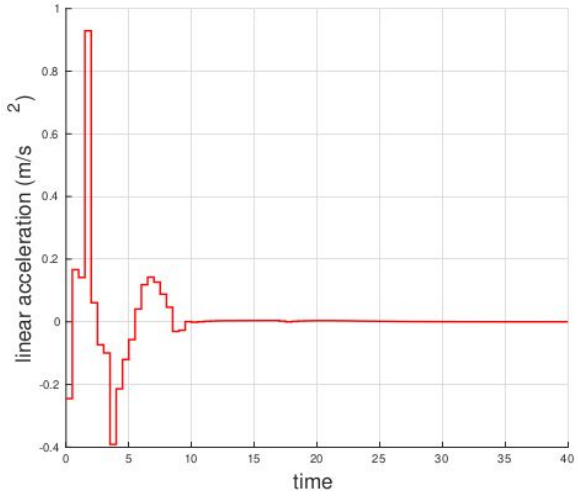


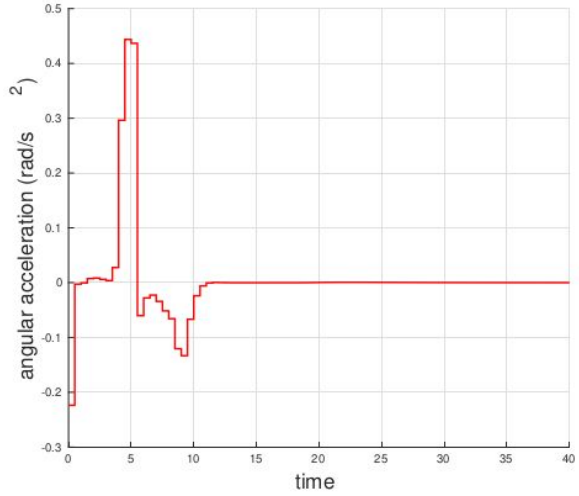






It can be noticed that the angular velocity after a transient it assumes the same value of the angular velocity of the walker. For different perceptual values, or for different time sampling, the controller is able to make ω = 0.2[rad/s] in a faster way (as result the walker would be closer to the center).





Comparison cascaded control - scenario 3 - part 1

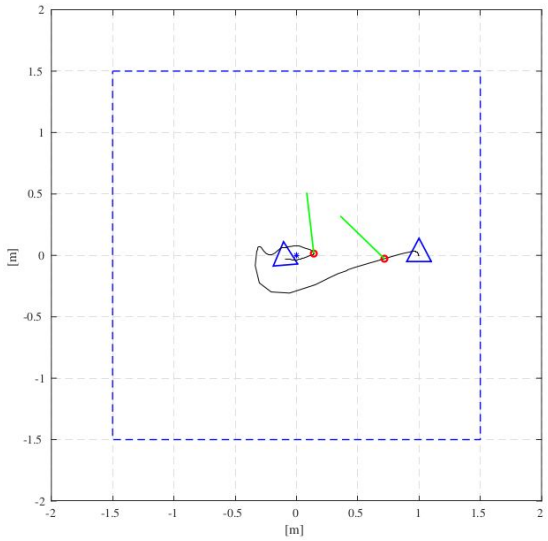
Second order control law obtained through cascaded system method:

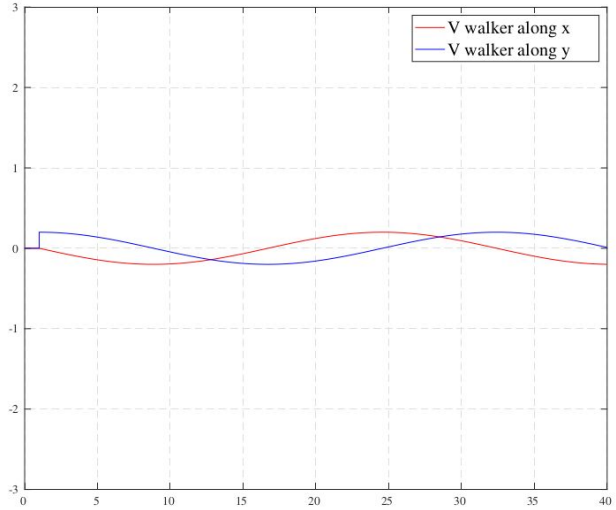
$$\begin{pmatrix} a \\ a_\omega \end{pmatrix} = \frac{d\, f(x, y, \theta, \tilde{V}_w)}{d\, t} - K_a \left(\begin{pmatrix} v \\ \omega \end{pmatrix} - f(x, y, \theta, \tilde{V}_w)) \right)$$

$$f(x, y, \theta, \tilde{V}_w)) = \begin{pmatrix} k(x^2 + y^2) sgn(x\, cos\theta + y\, sin\theta) \\ k(y\, cos\theta - x\, sin\theta) sgn(x\, cos\theta + y\, sin\theta) \end{pmatrix}$$

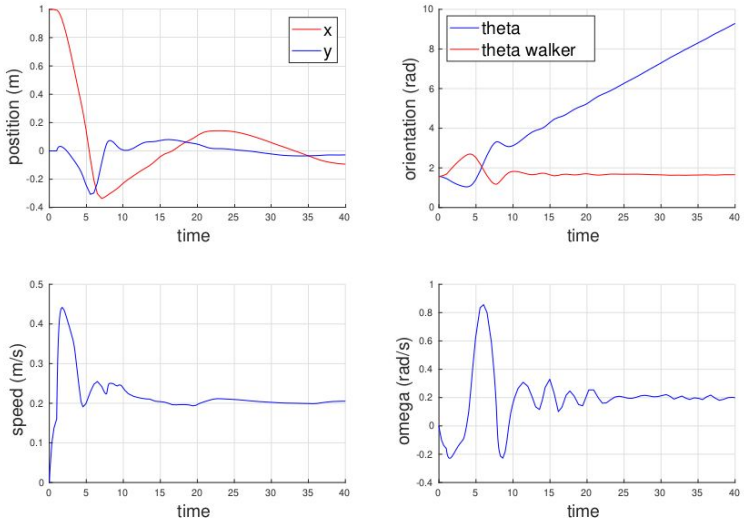
```
k     = 1;
k_a   = 2;
```

Scenario 3 trajectory under control:

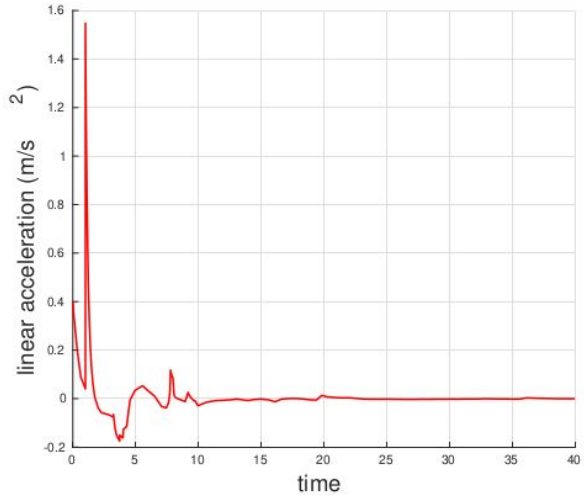


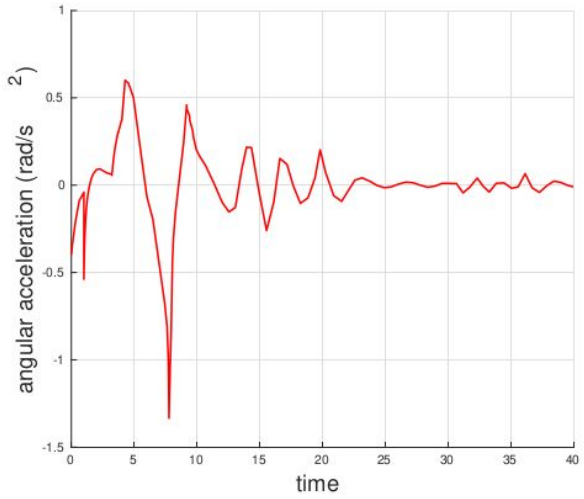


Comparison cascaded control - scenario 3 - part 2



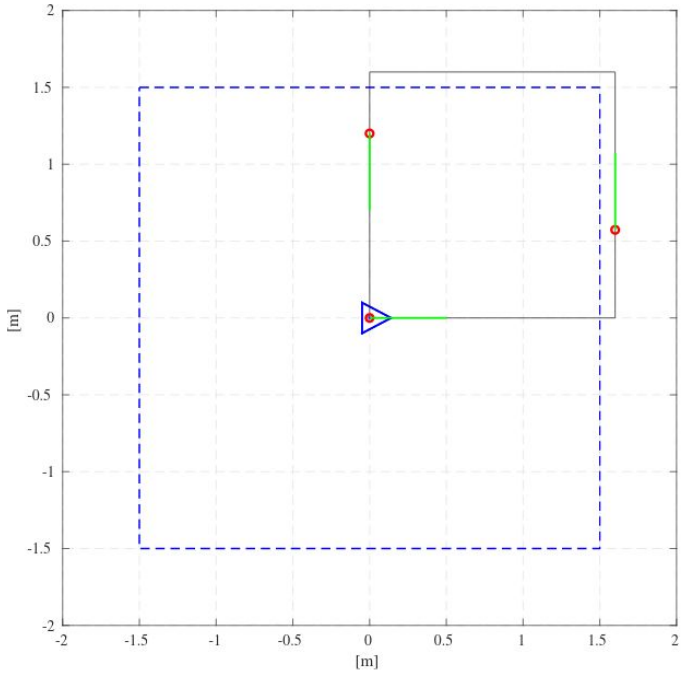
It tends to align the platform with $\tilde{V}_w$, it does not capture the motion. The convergence to the estimate of the walker velocity oscillates around the nominal value since the estimated velocity is a low-pass filtered version of the actual velocity (due to the observers).

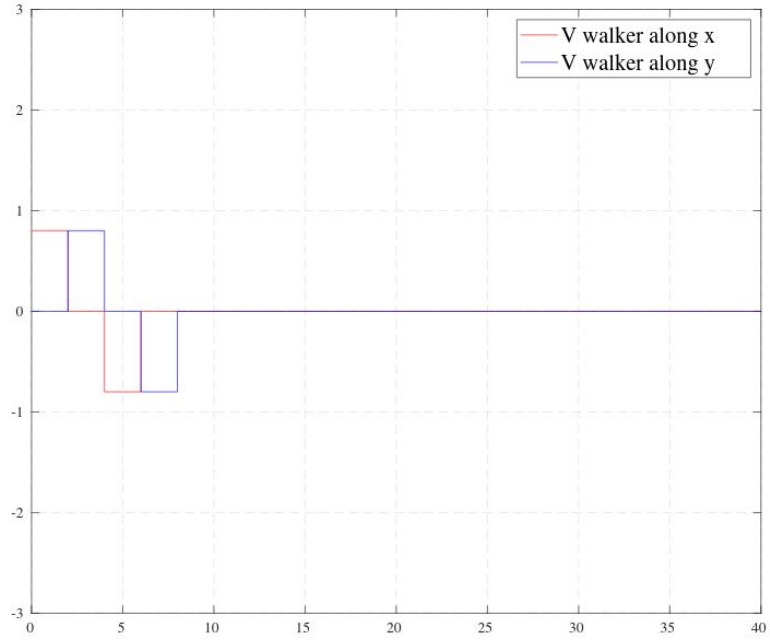




Scenario 4 -part 1

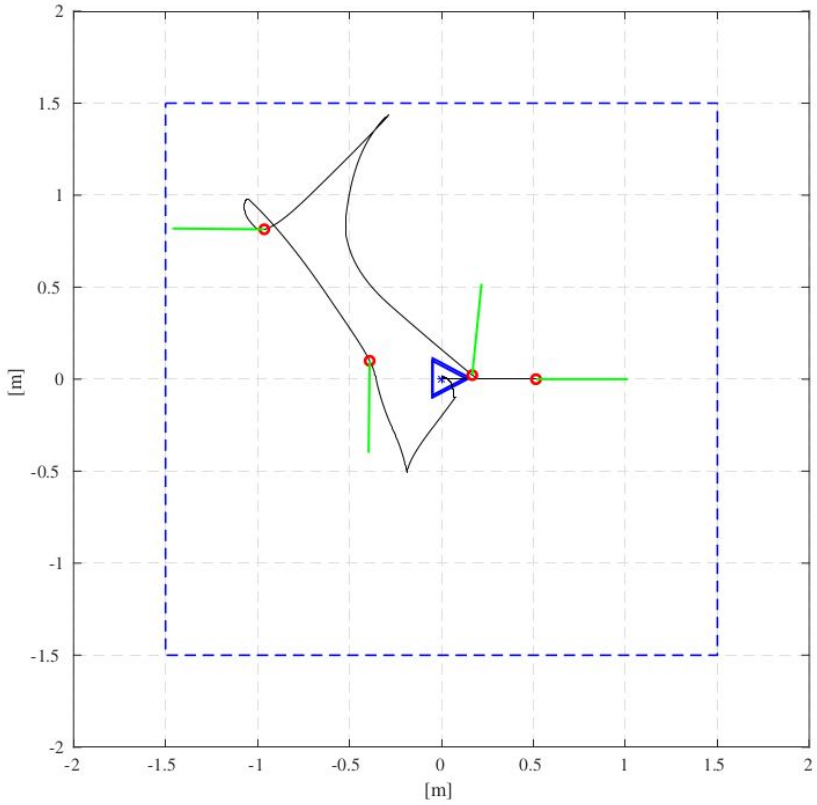
In the last scenario a square path is considered, without control action it looks like:



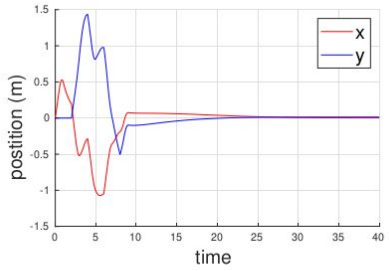


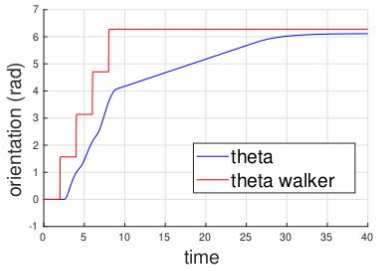
Scenario 4 - part 2

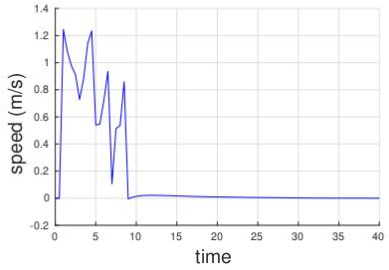
The control action is able to keep the walker close to the platform center, but he/she gets very close to the limits of the platform.

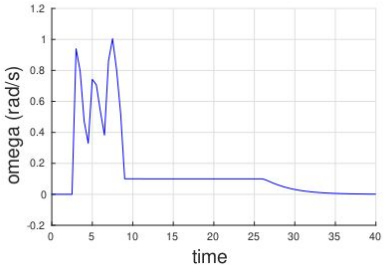


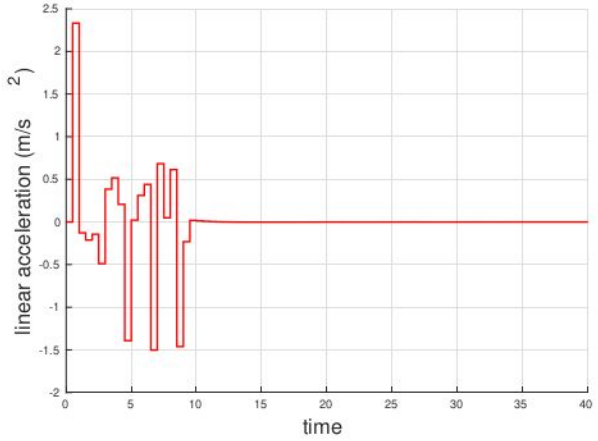
Scenario 4 - part 3

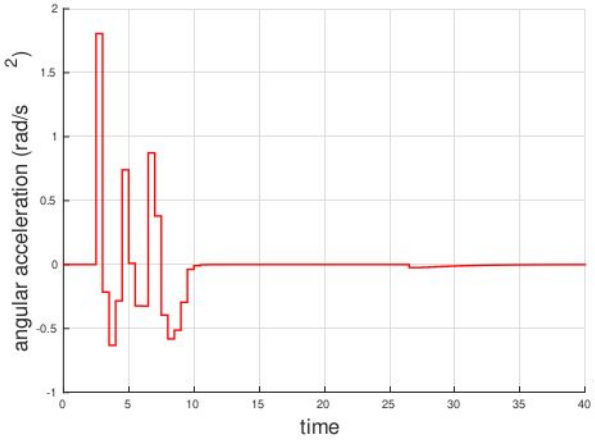




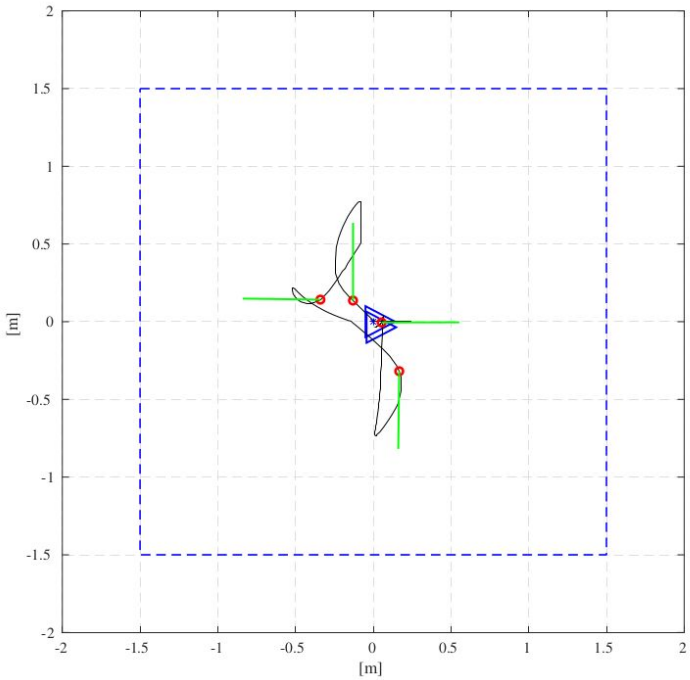


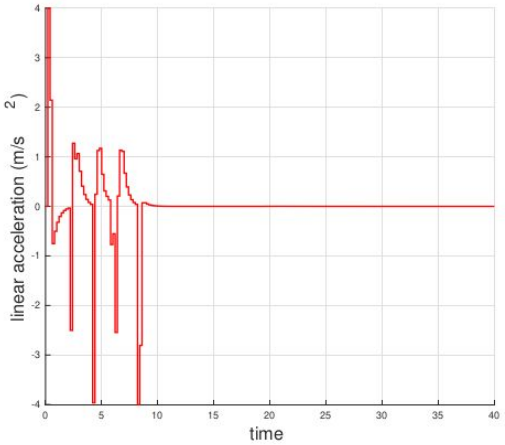


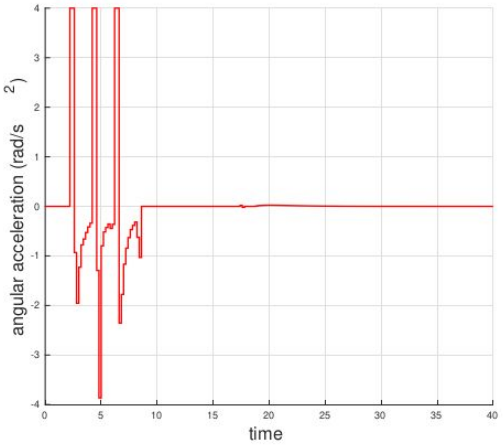




Scenario 4 - change of Ts, Ts = 0.2







If we consider a lower sampling time, the controller is able to contrast in a faster way the walker velocities and keep him/her closer to the platform center. For this last scenario a simulation has been done considering T_s = 0.2.

Conclusion

A different control scheme for the CyberCarpet platform has been proposed.

Different simulations have been done, for different paths and different constraints. The results satisfy the control specifications, for further development the constraints should be defined based on the walker dynamics, perceptual constraints and platform limits. Once this is done the nonlinear MPC parameters can be defined definitely. Also a comparison with the cascade control scheme has been proposed, showing the pros and cons of the two schemes.

The nonlinear MPC is very powerful and provides the best control solution that can be achieved. In general linear MPC controls are more computationally efficient than nonlinear MPC. For future works, the nonlinear MPC can be converted, for different operating points (in Matlab it can be done using the function *convertToMPC*) and then a gain-scheduled MPC can be implemented using the linear MPC objects. If the controller obtained is comparable with the performances of the nonlinear one then it may be preferred to be used. Finally experiments on a real implementation of the system have to be done.

Thank you for the attention!