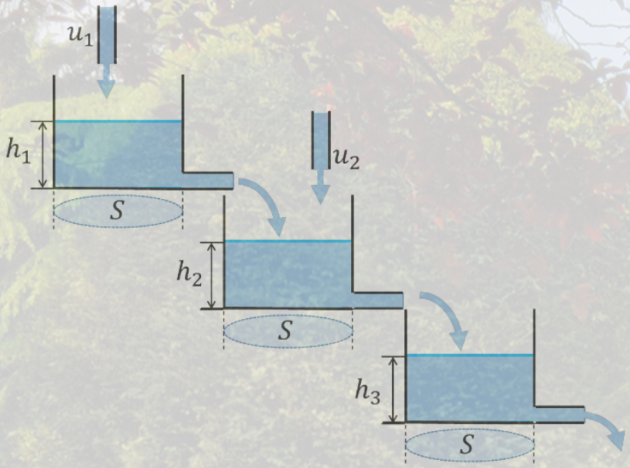
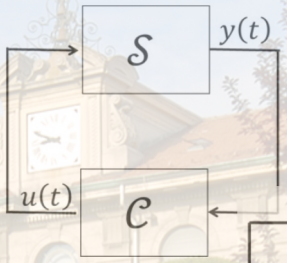
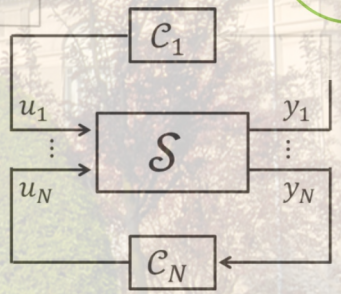
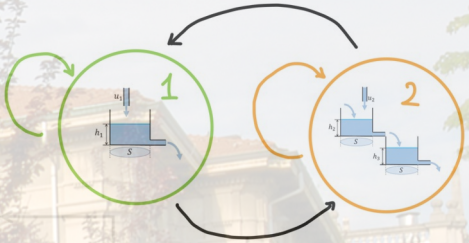
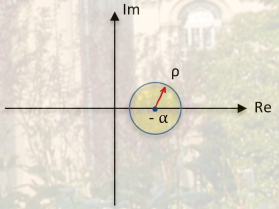
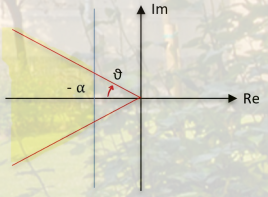












POLITECNICO
MILANO 1863

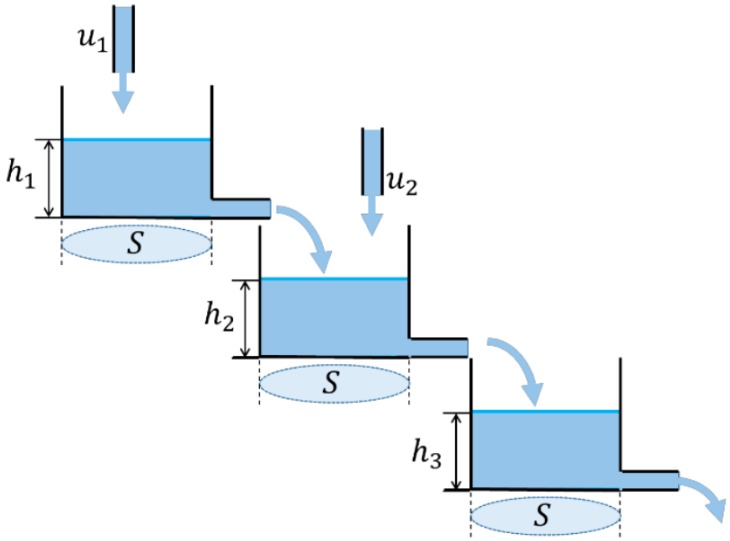
NETWORKED CONTROL

PROJECT WORK

Codazzi Andrea, Puglisi Alessandro, Sonnino Sole Ester

System description

The system is made by 3 tanks:



The linearized model of the system is:

$$\begin{cases} S\dot{h}_1 = & -kh_1 + u_1 \\ S\dot{h}_2 = & kh_1 - kh_2 + u_2 \\ S\dot{h}_3 = & kh_2 - kh_3 \end{cases}$$

For S= 1 m^2 and k = 1 m^2/s

And, defining $x = [h_1,h_2,h_3]^T$ and $u = [u_1,u_2]^T$

We obtain the continuous-time state space model:

$$\dot{x} = Ax + Bu$$

System decomposition

According to the system description, and to guarantee that:

- All sub-systems have Inputs and outputs
- The local control problem is not "too large"
- Each component of u, y is associated to one sub-system

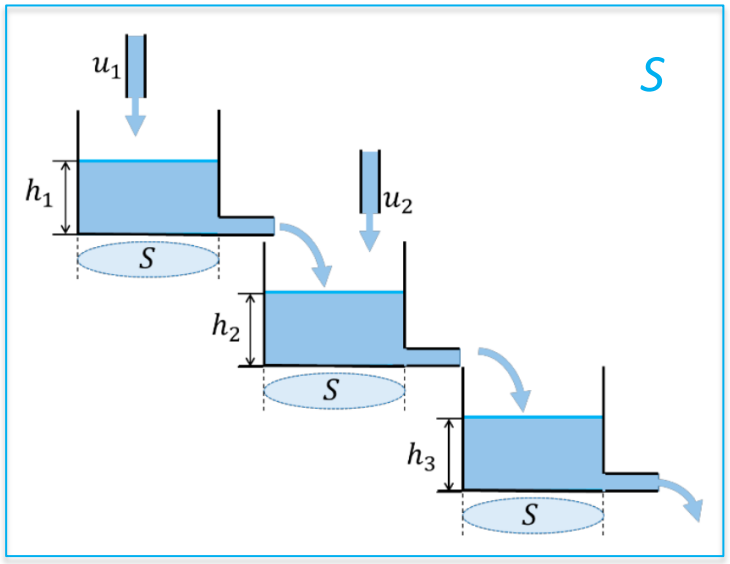
From:

$\dot{x} = Ax + Bu$

$y = Cx + Du$

$$A = \begin{bmatrix} -1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad D = [0 \quad 0]$$



System decomposition

According to the system description, and to guarantee that:

- All sub-systems have Inputs and outputs
- The local control problem is not "too large"
- Each component of u, y is associated to one sub-system

N = 2

To:

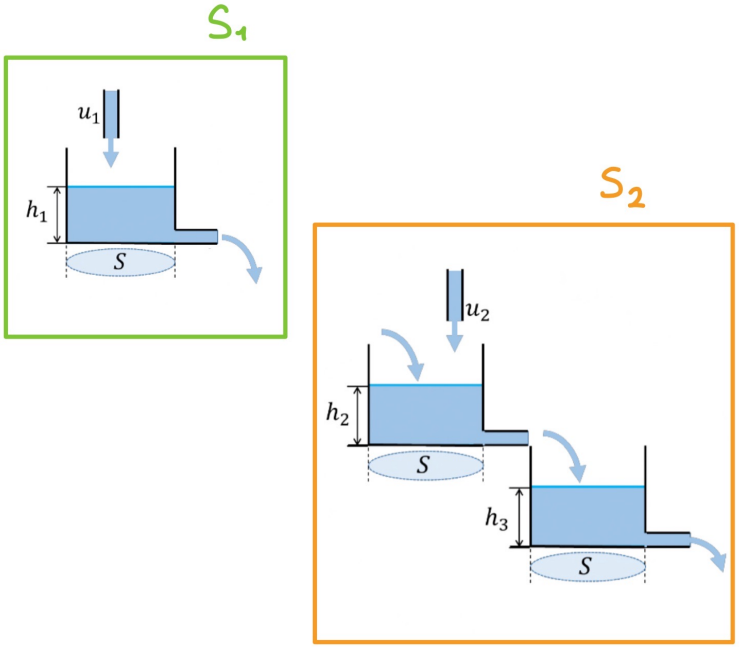
$\dot{x} = Ax + B_1 u_1 + B_2 u_2$

$y_1 = C_1 x$

$y_2 = C_2 x$

$$A = \begin{bmatrix} -1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix} \quad B_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad B_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$C_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \quad C_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Open loop analysis I
Continuous time system

Eigenvalues of the matrix A:

The eigenvalues of A are the values of lambda such that the characteristic polynomial is 0: $\det(\lambda * I - A) = p_A(\lambda) = 0$

In our system: The eigenvalues af A are: $\lambda_1 = -1,\ \lambda_2 = -1, \lambda_3 = -1$

(here A is triangular, we can read them directly from matrix diagonal elements)

The open loop system is asymptotically stable if and only if the eigenvalues of the matrix A are such that: $Re(\lambda_i) < 0\ , \forall i$

A matrix with this property is denoted as Hurwitz stable

In our system: **The (open loop) system is asymptotically stable**

Spectral abscissa:

The spectral abscissa is the real value of the eigenvalue of A with maximum real part. In our case the specral abscissa is: $\rho = -1$

Open loop analysis II
Discrete time system

Discretization of the system:

The definition of the discrete time system is easly obtained from the continuous time model, once the sampling time h has been defined.

Given:

A: state matrix (cont-time)

B: input matrix (cont-time)

C: output state matrix (cont-time)

D: output input matrix (cont-time)

h: sampling time

We can define the discrete time system as:

$$S_{DT}: \begin{cases} x_{k+1} = Fx_k + Gu_k \\ y_k = Hx_k + Lu_k \end{cases}$$

where $k \in \mathbb{N}$, $F = e^{Ah}$, $G = \int_0^h e^{Av} dv B$, $H = C$, $L = D$.

Open loop analysis II
Discrete time system

Choice of the samplig time h

To discretize the system we need to define a value for the sampling time h.

In order to find the most appropriate value of h we've done some adjustments a posteriori, taking into account the discrete-time closed loop dynamic and the control action, more specifically we've seen that using a small sampling time (0.01<h<0.1) the control action is too aggressive, with large peaks (too reactive) and as a consequence the states also show big undesired oscillations.

If instead we consider an higher value of **h=0.2**, the control action will be smoother. And we are able to control properly the system in closed loop.

Moreover, looking at the simultations of the open loop continuous time system (and to the spectral abscissa), we can see that the settling time of the state variables is around 5s, so the sampling time that we've chosen is appropriate in order to capture its dynamic.

Open loop analysis II
Discrete time system

Eigenvalues of the matrix F:

The eigenvalues of F are the values of lambda such that the characteristic polynomial is 0: $\det(\lambda * I - F) = p_F(\lambda) = 0$

The eigenvalues of F are: $\lambda_1 = 0.8187,\ \lambda_2 = 0.8187,\ \lambda_3 = 0.8187$

The open loop system is asymptotically stable if and only if the eigenvalues of the matrix F are such that: $|\lambda_i(F)| < 1,\ \forall i$

A matrix with this property is denoted as Shur stable

In our system: **The (open loop) system is asymptotically stable**

We can see that there is a strict correlation between the eigenvalues of matrix A and the ones of matrix F: $e^{\lambda_i(A)h} = \lambda_i(F)$

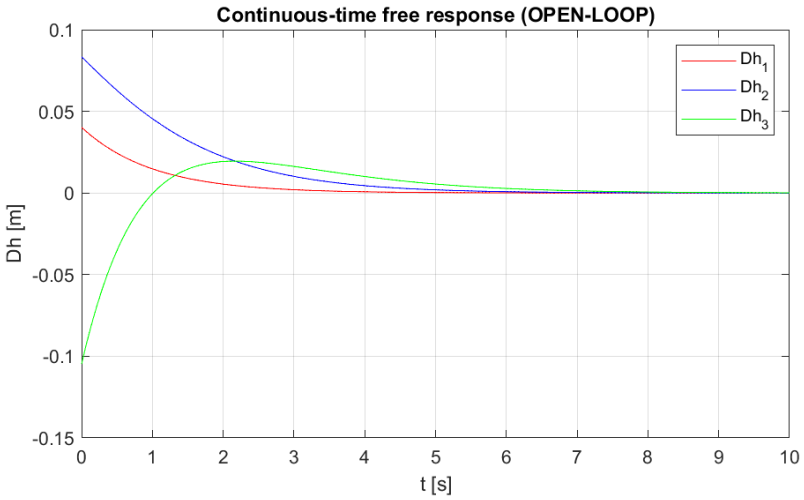
Spectral radius:

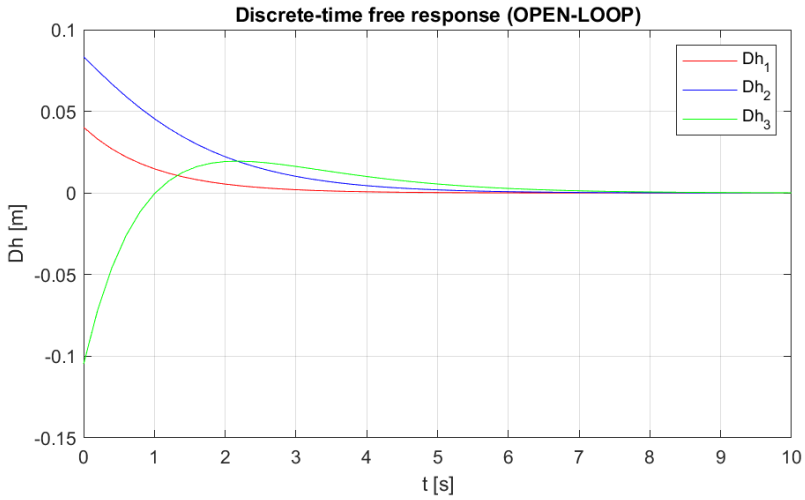
The spectral radius is the modulus of the eigenvalue of F with maximum absolute value. In our case the spectral radius of F is: $\rho = 0.8187$

Open loop analysis
Simulation

Simulating the system dynamic in open loop, for different initial random states:

$x_0 = [0.040, 0.083, -0.104]^T$

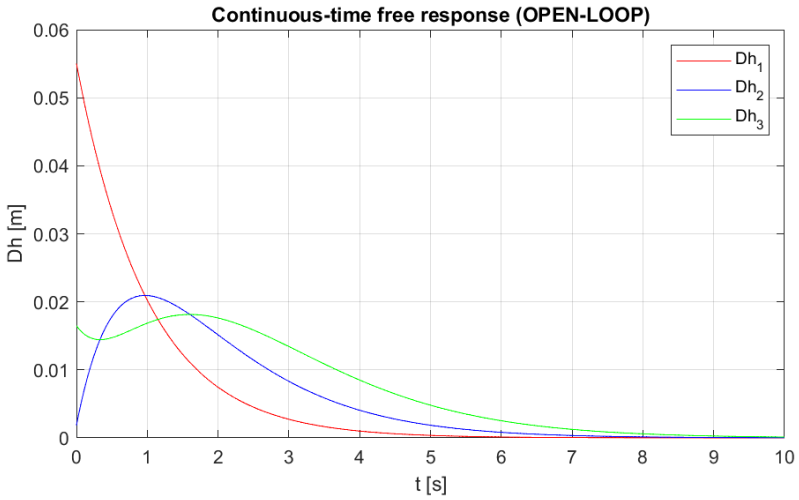


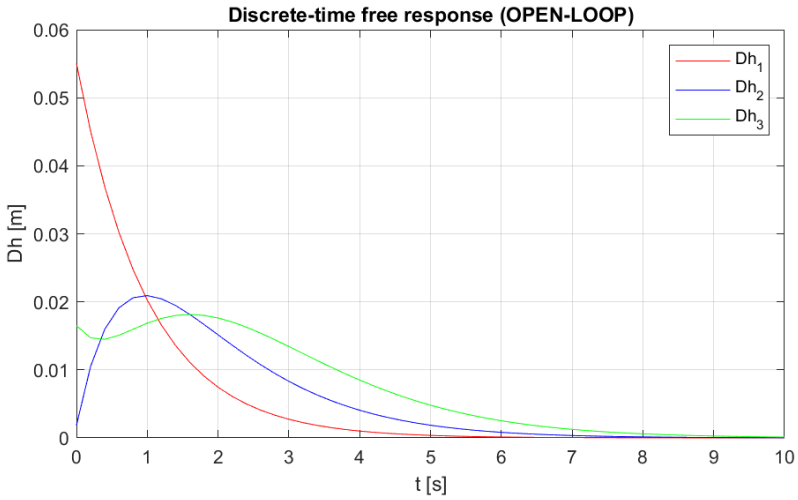


Open loop analysis
Simulation

Simulating the system dynamic in open loop, for different initial random states:

x_0 = [0.055, 0.0018, 0.0165]^T



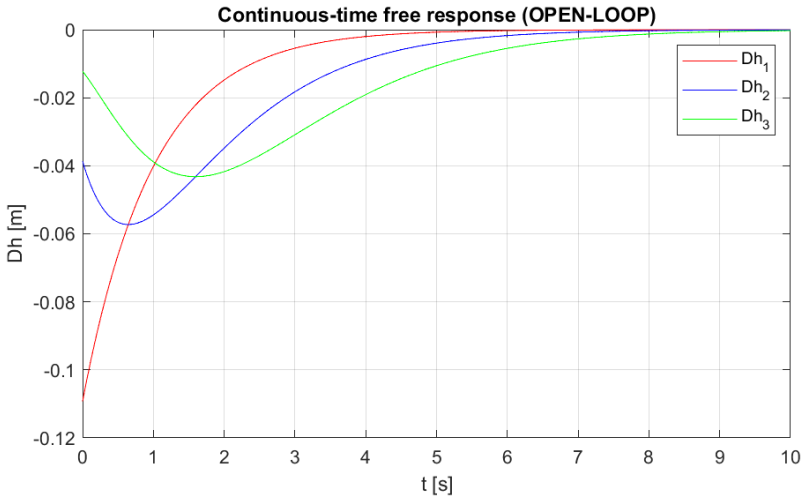


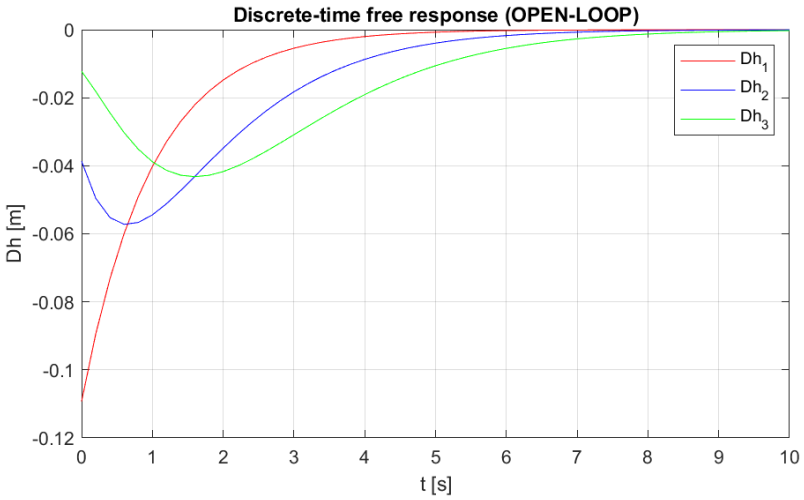
Open loop analysis
Simulation

Simulating the system dynamic in open loop, for different initial random states:

$x_0 = [-0.109, -0.039, -0.012]^T$

h = 0.2 s represent well the continuous time dynamics
(this is not the main reason of its choice)

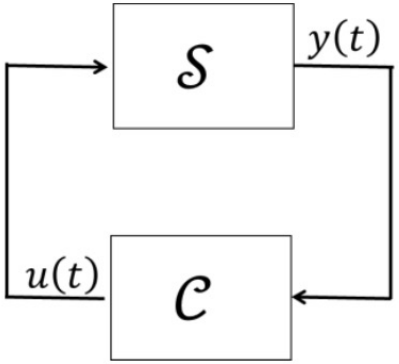




Always **converging** (as expected), but in some cases shows undesired **long oscillations** (creating waves inside the tanks) and the dynamic seems a little bit **slow**.

State feedback contol
Centralized structure

Consider a system having the following structure:



Where the dynamic of system S is described by:

$$\mathcal{S}: \begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) \end{cases}$$

(Where the matrices are the ones described before)

Apply to the system a **static state-feedback control law** of the type: $u(t) = K_x x(t)$
where K_x is the control gain (mxn matrix)

```
ContStructure_centralized = [1  1; 1  1]
```

The **closed loop dynamics** can be described by: $\dot{x}(t) = (A + BK_x)x(t)$

If the pair (A,B) is controllable we can **design a control gain K_x such that the matrix $(A+BK_x)$ is Hurwitz stable**

State feedback contol
CFM

Continuous time

The **centralized fixed modes** are the elements of the set:

$$\Lambda_{CFM} = \bigcap_{K_y \in \mathbb{R}^{m \times p}} \text{spec}(A + BK_yC)$$

So they are the eigenvalues of A that do not change position under a static output-feedback control law.

It can be proved that the modes of A that cannot be controlled and/or observed correspond to the CFM.

Discrete time

If a mode λ is a CFM, the discrete-time system (obtained by, e.g.,sampling-and-hold discretization with sampling time h)

$$\begin{cases} x(k+1) = Fx(k) + Gu(k) \\ y(k) = Hx(k) \end{cases}$$

has a corresponding (discrete-time) centralized fixed mode in $e^{\lambda h}$.

Definition: The **discrete-time CFM** are the elemets of the set:

$$\Lambda_{CFM}^{\text{discrete-time}} = \bigcap_{K_y \in \mathbb{R}^{m \times p}} \text{spec}(F + GK_yH)$$

State feedback contol
CFM- computation

Continuous time

1. Compute the spectrum of matrix A, i.e., $\text{spec}(A)$.
2. Define $\Lambda_{CFM} = \text{spec}(A)$.
3. Select the $K_y \in \mathbb{R}^{m \times p}$ randomly.
4. Compute $\text{spec}(A + BK_yC)$.
5. Set $\Lambda_{CFM} = \text{spec}(A + BK_yC) \cap \Lambda_{CFM}$.
6. Repeat procedure from step 3 "a number of times".

Running the algorithm on our system we'll see that **there are no continuous-time CFM**

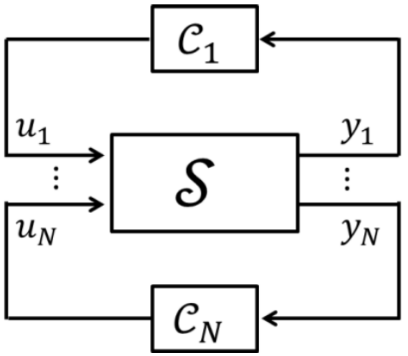
Discrete time

1. Compute the spectrum of matrix F, i.e., $\text{spec}(F)$.
2. Define $\Lambda_{CFM}^{\text{discrete-time}} = \text{spec}(F)$.
3. Select the matrix $K_y \in \mathbb{R}^{m \times p}$ randomly.
4. Compute $\text{spec}(F + GK_yH)$.
5. Set $\Lambda_{CFM}^{\text{discrete-time}} = \text{spec}(F + GK_yH) \cap \Lambda_{CFM}$
6. Repeat procedure from step 3 'a number of times'

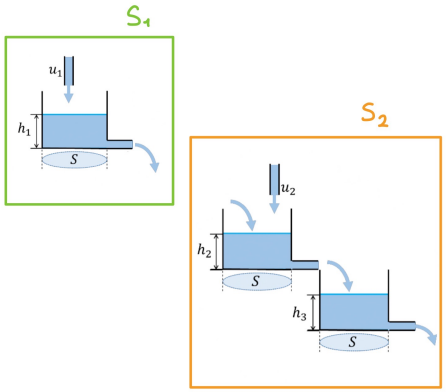
Running the algorithm on our system we'll see that **there are no discrete-time CFM**

State feedback contol
Decentralized structure

Consider a **decentralized control system**, where controller C_i is committed to control the output y_i by acting on input u_i , for each i=1,…,N



Where *S*:



$$\dot{x} = Ax + B_1 u_1 + B_2 u_2$$
$$y_1 = C_1 x$$
$$y_2 = C_2 x$$

$$A = \begin{bmatrix} -1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix} \quad B_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad B_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$C_1 = [1 \quad 0 \quad 0] \quad C_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

State feedback contol
Decentralized structure

For each channel the control law is: $u_i(t) = K_x^i x_i(t)$

Since x_i is a subvetor of x, in the considered system:

$$x = [x_1 \;\; x_2]^T$$

$$x_1 = [1 \;\; 0 \;\; 0]\, x = C_1 x$$

$$x_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} x = C_2 x$$

Collectively we write:

$$C = [C_1 \;\; C_2]^T = I$$

$$K_x^d = \text{diag}(K_x^1, K_x^2)$$

(It's like an output feedback control law. Because here outputs and states coincide.)

```
ContStructure_decentralized = [1  0; 0  1]
```

The **closed loop dynamics**, therefore, is:

$$\dot{x}(t) = Ax(t) + BK_x^d Cx(t) = (A + BK_x^d)x(t)$$

State feedback contol
DFM

Continuous time

Definition: DFM are elements of the set:

$$\Lambda_{DFM} = \bigcap_{K_y^d \in \mathcal{K}^d} \text{spec}(A + BK_y^d C)$$

With respect to the CFM, where K_y is free to take any structure, for the DFM the structure of K_y is constrained to be block diagonal (K^d) , therefore:

$$\Lambda_{CFM} \subseteq \Lambda_{DFM}$$

Discrete time

If a mode λ is a DFM, the discrete-time system (obtained by, e.g., sampling-and-hold discretization with sampling time h)

$$\begin{cases} x(k+1) = Fx(k) + Gu(k) \\ y(k) = Hx(k) \end{cases}$$

DOES NOT NECESSARILY have a corresponding (discrete-time) decentralized fixed mode in $e^{\lambda h}$.

Definition: The discrete-time decentralized fixed modes are the elements of the set:

$$\Lambda_{DFM}^{\text{discrete-time}} = \bigcap_{K_y^d \in \mathcal{K}_y^d} \text{spec}(F + GK_y^d H)$$

State feedback contol
DFM- computation

Continuous time

1. Compute the spectrum of matrix A, i.e., $\text{spec}(A)$.
2. Define $\Lambda_{DFM} = \text{spec}(A)$.
3. Select the block-diagonal matrix $K_y^d \in \mathcal{K}_y^d$ randomly.
4. Compute $\text{spec}(A + BK_y^d C)$.
5. Set $\Lambda_{DFM} = \text{spec}(A + BK_y^d C) \cap \Lambda_{DFM}$.
6. Repeat procedure from step 3 "a number of times".

Running the algorithm on our system we'll see that **there are no continuous-time DFM**

Discrete time

1. Compute the spectrum of matrix F, i.e., $\text{spec}(F)$.
2. Define $\Lambda_{DFM}^{\text{discrete-time}} = \text{spec}(F)$.
3. Select the block-diagonal matrix $K_y^d \in \mathcal{K}_y^d$ randomly.
4. Compute $\text{spec}(F + GK_y^d H)$.
5. Set $\Lambda_{DFM}^{\text{discrete-time}} = \text{spec}(F + GK_y H) \cap \Lambda_{DFM}$
6. Repeat procedure from step 3 'a number of times'

Running the algorithm on our system we'll see that **there are no discrete-time DFM**

State feedback contol
Distributed structure

A distributed controller is characterized by the fact that each input u_i is a function, not only of the corresponding output y_i, but also of pieces of information from other channels compatibly with the **information structure constraints** $(\mathcal{N}_i)$.

Static state feedback control law:

Consider the states of the system: $x = [x_1 \ x_2]^T$

For each controller the control law is: $$u_i(t) = \sum_{j \in \mathcal{N}_i} K_x^{ij} x_j(t)$$

In matrix form we can write: $$u_i(t) = [K_x^{i1} \qquad K_x^{i2}]x(t)$$

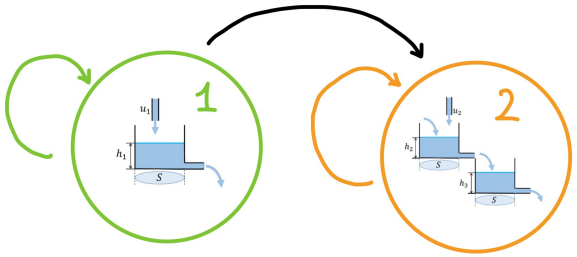
$$u(t) = \begin{bmatrix} u_1(t) \\ \cdot \\ u_2(t) \end{bmatrix} = \begin{bmatrix} K_x^{11} & K_x^{12} \\ \cdot & \\ K_x^{21} & K_x^{22} \end{bmatrix} x(t) = K_x \ x(t)$$

State feedback contol
Distributed structure - information structure constraint

Information structure constraint:

We have 2 significant options:

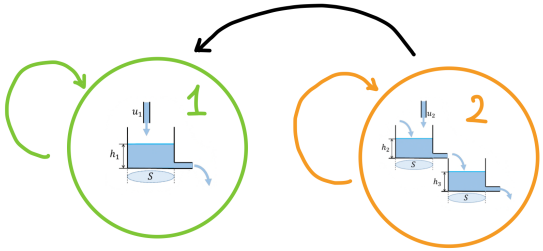
DISTRIBUTED STRING 1



ContStructure_String = [1 0;1 1]

In this case we act ‘a posteriori’,
u_2 is chosen accordingly to y_1
(this seems phiscally more reasonable, becouse y_1 directly influence the second sub-system dynamic, simulations in continuous prove us right!)

DISTRIBUTED STRING 2



ContStructure_String = [1 1;0 1]

In this case we act ‘a priori’,
u_1 is chosen accordingly to y_2
(works well sometimes, but simulation will show that it’s not a robust choice in stressful conditions)

State feedback contol
DiFM

Continuous time

Definition: The DiFM are the elements of the set:

$$\Lambda_{DiFM} = \bigcap_{K_y \in \mathcal{K}_y^{\mathcal{N}}} \text{spec}(A + BK_yC)$$

Because the matrix K_y in the distributed structure has less DOF of the one in the centralized case, it follows that: $\Lambda_{CFM} \subseteq \Lambda_{DiFM}$.
On the other hand it has more DoF of the block diagonal K_y of the decentralized case, so: $\Lambda_{DiFM} \subseteq \Lambda_{DFM}$
Overall: $\Lambda_{CFM} \subseteq \Lambda_{DiFM} \subseteq \Lambda_{DFM}$

Discrete time

If a mode λ is a DiFM, the discrete-time system (obtained by, e.g., sampling-and-hold discretization with sampling time h)

$$\begin{cases} x(k+1) = Fx(k) + Gu(k) \\ y(k) = Hx(k) \end{cases}$$

DOES NOT NECESSARILY have a corresponding (discrete-time) distributed fixed mode in $e^{\lambda h}$.

Definition: The discrete-time DiFM are the elements of the set:

$$\Lambda_{DiFM}^{\text{discrete-time}} = \bigcap_{K_y \in \mathcal{K}_y^{\mathcal{N}}} \text{spec}(F + GK_yH)$$

State feedback contol
DiFM- computation

Continuous time

1. Compute the spectrum of matrix A, i.e., $\mathrm{spec}(A)$.
2. Define $\Lambda_{DiFM} = \mathrm{spec}(A)$.
3. Select the structured matrix $K_y \in \mathcal{K}_y^{\mathcal{N}}$ randomly.
4. Compute $\mathrm{spec}(A + BK_yC)$.
5. Set $\Lambda_{DiFM} = \mathrm{spec}(A + BK_yC) \cap \Lambda_{DiFM}$
6. Repeat procedure from step 3 'a number of times'

Running the algorithm on our system we'll see that **there are no continuous-time DiFM**

Discrete time

1. Compute the spectrum of matrix F, i.e., $\mathrm{spec}(F)$.
2. Define $\Lambda_{DiFM}^{\mathrm{discrete-time}} = \mathrm{spec}(F)$.
3. Select the structured matrix $K_y \in \mathcal{K}_y^{\mathcal{N}}$ randomly.
4. Compute $\mathrm{spec}(F + GK_yH)$.
5. Set $\Lambda_{DiFM}^{\mathrm{discrete-time}} = \mathrm{spec}(F + GK_yH) \cap \Lambda_{DiFM}$
6. Repeat procedure from step 3 'a number of times'

Running the algorithm on our system we'll see that **there are no discrete-time DiFM**

Control goals

Realizing a simple stabilizing control law results useless, because the system is already asymptotically stable.

Moreover if we implement this controller it can deteriorate the perfomaces of the system, so we've directly implemented a controller that was able to improve them.

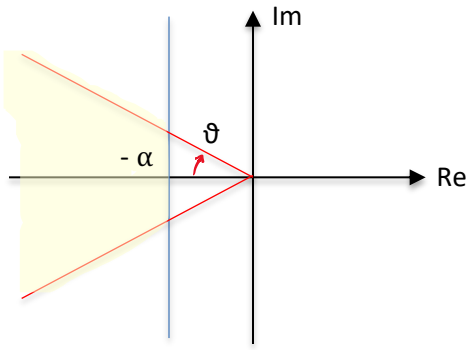
Our goal will be to implement a controller such that we obtain an optimal trade-off between:

- **Faster response** of the system
- **Limited oscillations** of the controlled variables
- Minimization of the **control action**

Control goals

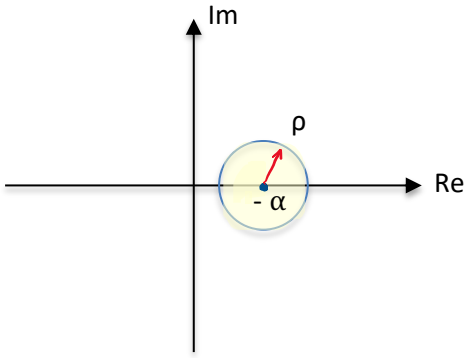
Continuous time

- Eigenvalues of the system are forced to have a real part which is smaller than a certain value (reduce the settling time of dominant mode)
- The damping factor is kept over a certain treshold (reduce oscillations as much as possible)



Discrete time

- Eigenvalues of the system are forced to be in a region inside the unitary circle, whose radius is smaller than 1, in order to obtain a faster response
- Move the region rightwards with respect to the origin in order to reduce the oscillations



Control goals
LMIs

1. Place the Eigenvalues in a desired region (faster response and limited oscillations)

Continuous time

- Guarantee that all the eigenvalues have real part such that: $Re(\lambda)<-\alpha$
  The corrensponding LMI is: $$YA^T + AY + L^T B^T + BL + 2\alpha Y < 0$$

- Confine the eigenvalues in a region $\mathcal{D}$ defined starting from the desired damping:
  $$\mathcal{D} = \{z \in \mathbb{C}: f_D(z) < 0\}$$
  with $f_D(z)=\Lambda+z\Theta+z*\Theta T$.
  We can define:
  $$f_D(z) = \begin{bmatrix} \sin(\theta)(z+z^*) & \cos(\theta)(z-z^*) \\ \cos(\theta)(z^*-z) & \sin(\theta)(z+z^*) \end{bmatrix}$$
  The LMI that guarantees this result is:

$$\begin{bmatrix} \sin(\theta)\{(A+BK_x)Y + Y(A+BK_x)^T\} & \cos(\theta)\{(A+BK_x)Y - Y(A+BK_x)^T\} \\ \cos(\theta)\{-(A+BK_x)Y + Y(A+BK_x)^T\} & \sin(\theta)\{(A+BK_x)Y + Y(A+BK_x)^T\} \end{bmatrix} < 0$$

Discrete time

- The goal is to place the eigenvalues of the discrete time system in a disk of center α and radius ρ.

  To do so we simply have to guarantee that the matrix $\frac{1}{\rho}(A + BK_x + \alpha I)$ is Shur stable.

  The correnspondin LMI is:

$$\frac{1}{\rho^2}(A + BK_x + \alpha I)P(A + BK_x + \alpha I)^T - P < 0,$$

  which leads to:

$$(A + BK_x)P(A + BK_x)^T + \alpha P(A + BK_x)^T + \alpha(A + BK_x)P - (\rho^2 - \alpha^2)P < 0$$

  applying the Shur complement we'll obtain:

$$\begin{bmatrix} (\rho^2 - \alpha^2)P - APA^T - AL^TB^T - BLA^T - \alpha(PA^T + AP + L^TB^T + BL) & BL \\ L^TB^T & P \end{bmatrix} > 0$$

2. Minimization of the control action

It would be desiderable to **minimize the control effort**, which is obtained by minimizing the size of the control gain K_x (obtained either using Y and L as optimization variables, in continuous time, or P and L in discrete time).

It is possible by imposing $\|K_x\| \leq \alpha_x$

We recall that $\|K_x\| \leq \|L\|\|Y^{-1}\|$ (in discrete time $\|K_x\| \leq \|L\|\|P^{-1}\|$), so we can enforce limitations on $\|L\|$ and $\|Y^{-1}\|$

Limit $\|L\|$

$$\begin{bmatrix} \alpha_L I & L^T \\ L & I \end{bmatrix} \geq 0$$

Limit $\|Y^{-1}\|$

$$\begin{bmatrix} \alpha_Y I & I \\ I & Y \end{bmatrix} \geq 0$$

Cost function to be minimized:

$$J = a_L \alpha_L + a_Y \alpha_Y$$

A good weigth trade off is to select $a_L = 0.01$, $a_Y = 10$

Controller Design
Continuous time

To reach our goal we proceed by 'iterative tuning', comparing results.

A good dumping factor can be obtained for $\vartheta_{max} \leq 30^{\circ}$,

While to speed up we try to push the spectral abscissa on the left, via $\alpha_{max} \geq 1$

Good trade-off?

Fix $\vartheta_{max} = 30^{\circ}$, α_{max} = variable > 1

Select a random initial condition, (fix that $x_0 = [-0.15, 0.092, -0.086]^T$) and repeat simulation increasing α_{max}

Once we reach a good behaviour, we simulate for new randomly selected x_0, to see if it is a good choice in general, than we fix that α and reduce ϑ as much as we can, to obtain a controller more robust against oscillations

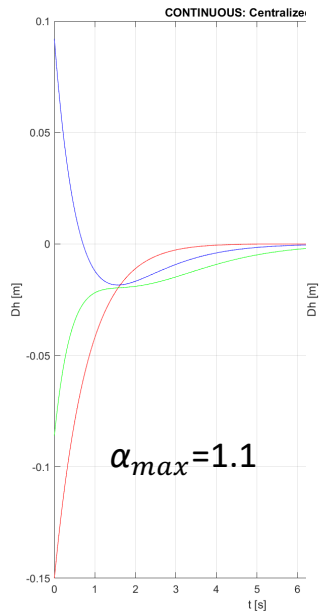
α_{max} = α , $\vartheta_{max} =$ variable < 30°

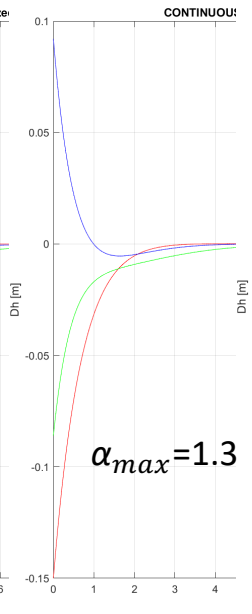
(all done analyzing also to the **control action**, to find the best performance/control effort trade-off)

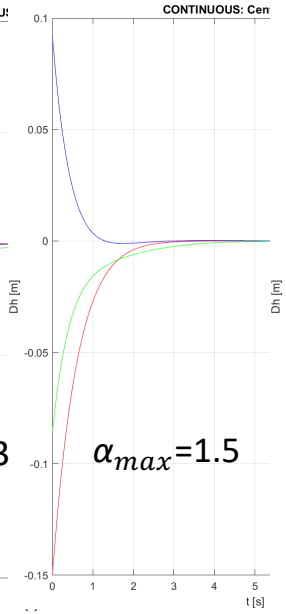
Controller Design
Continuous time

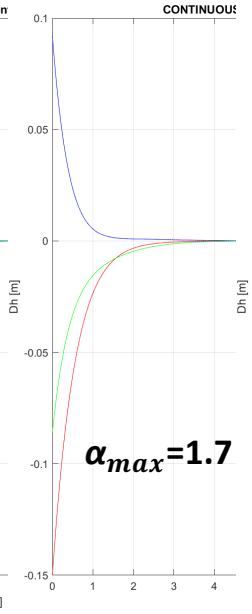
$\vartheta_{max} = 30^{\circ}$, α_{max} = variable

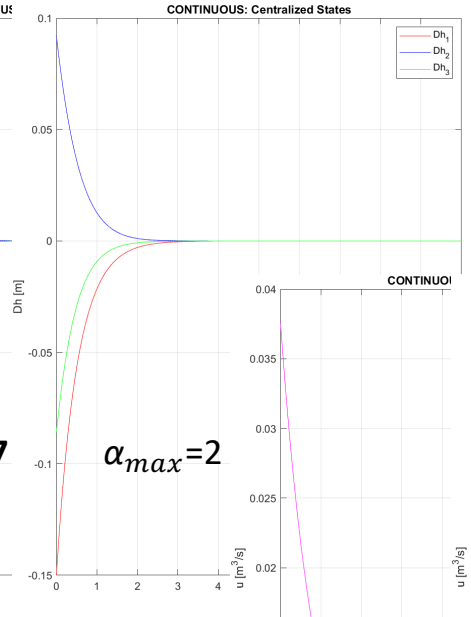
CENTRALIZED

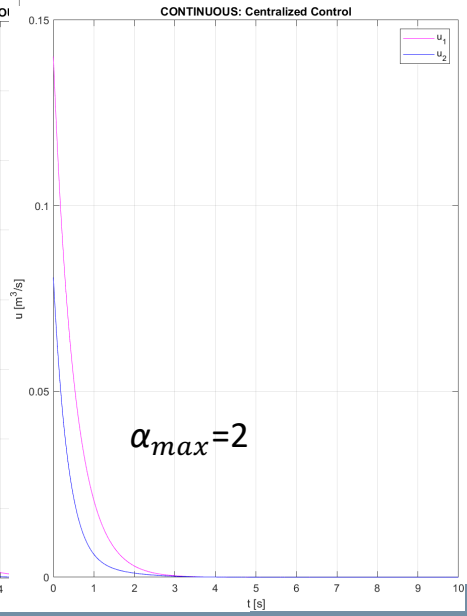








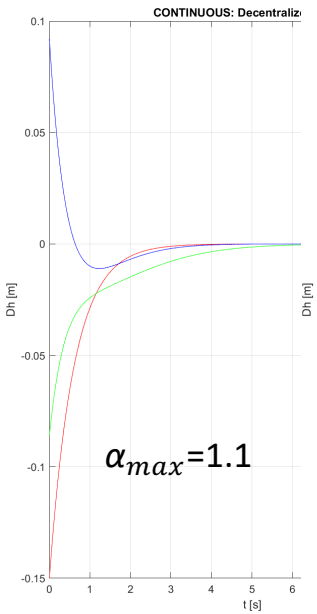


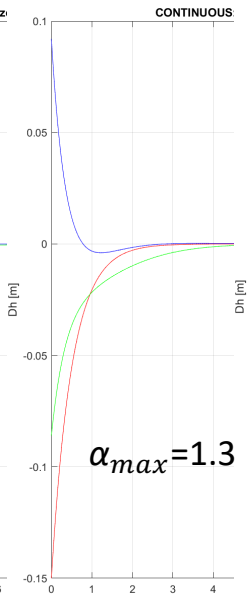


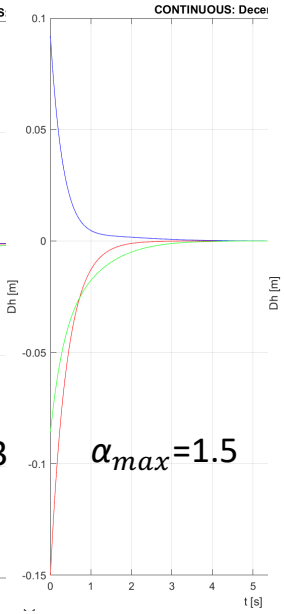
Controller Design
Continuous time

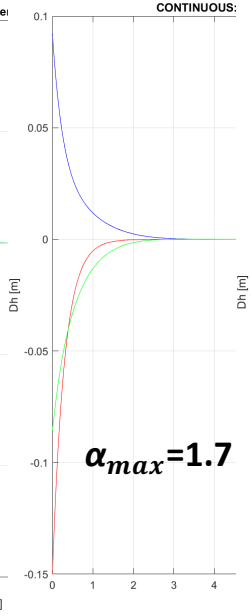
$\vartheta_{max} = 30^{\circ}$, α_{max} = variable

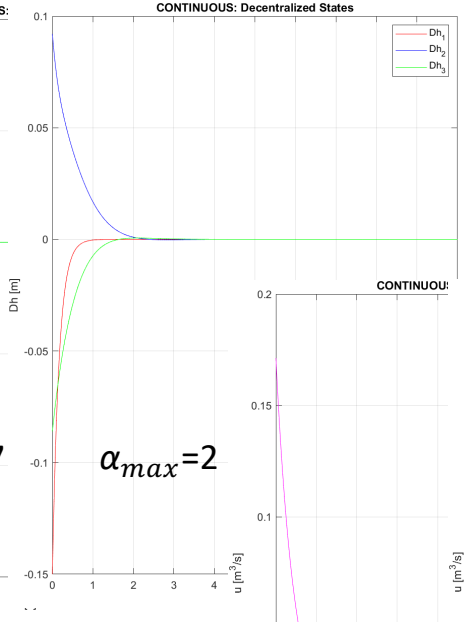
DECENTRALIZED

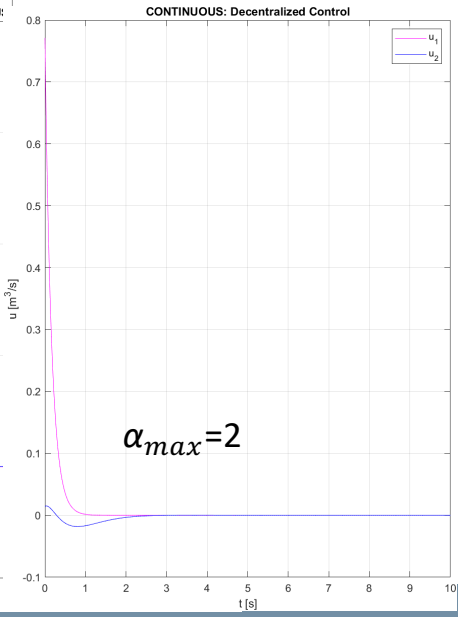








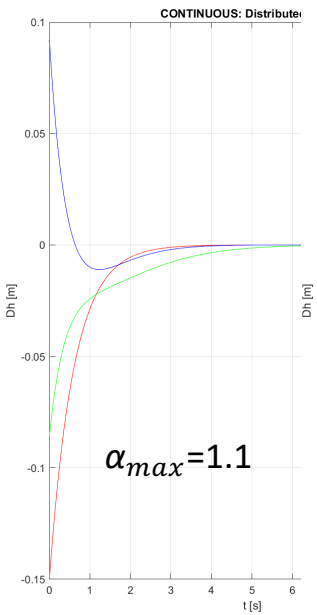


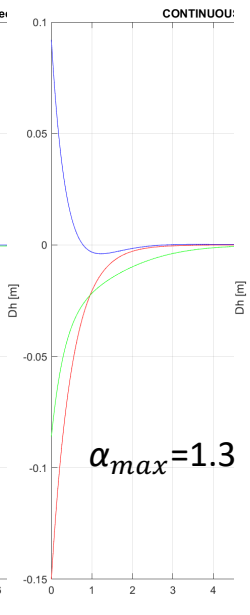


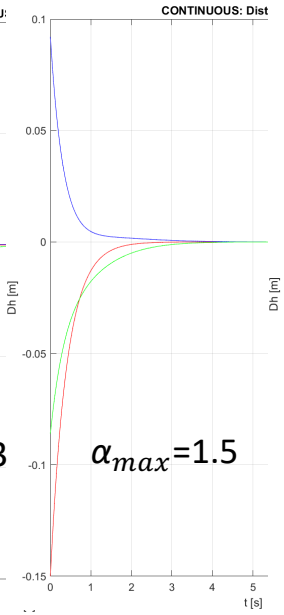
Controller Design
Continuous time

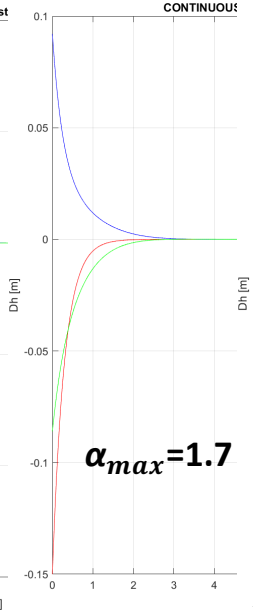
$\vartheta_{max} = 30^{\circ}$, α_{max} = variable

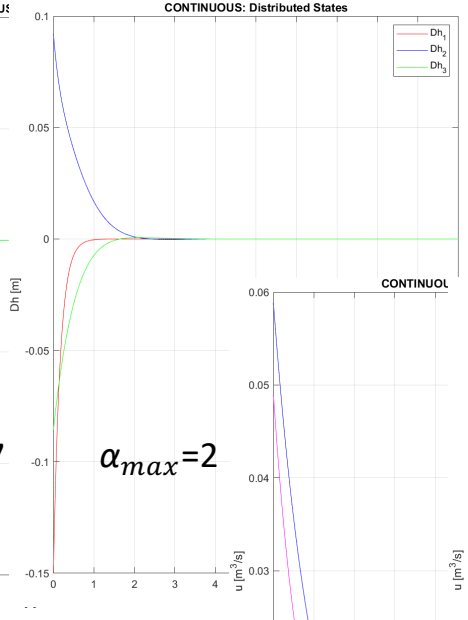
DISTRIBUTED 1

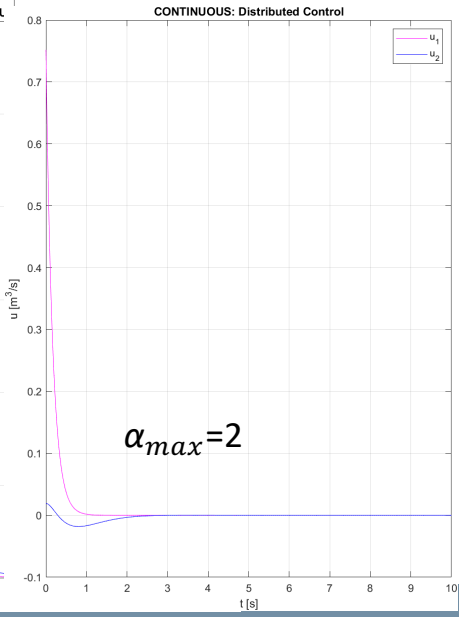








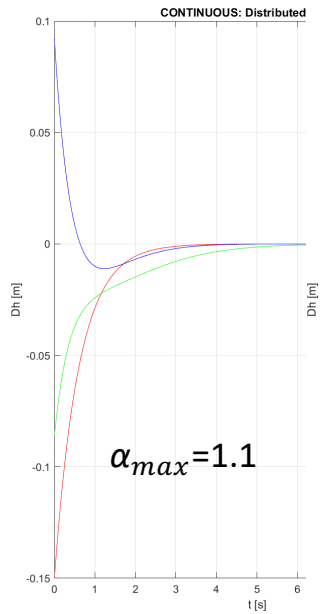


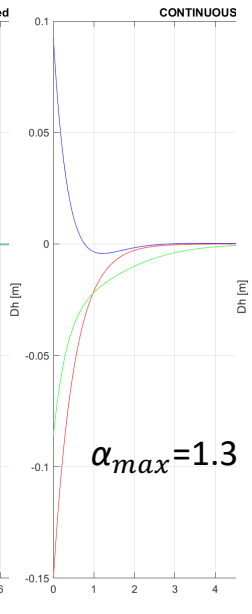


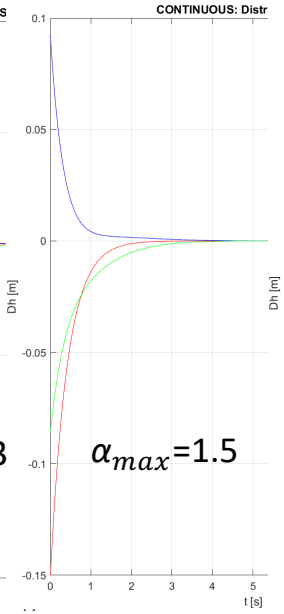
Controller Design
Continuous time

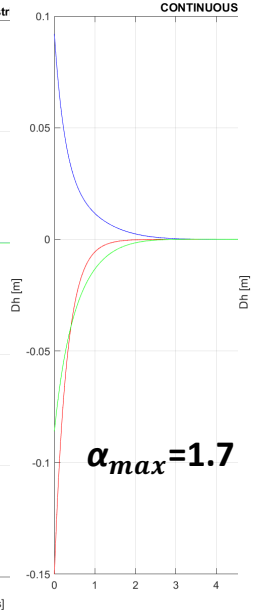
$\vartheta_{max} = 30^{\circ}$, α_{max} = variable

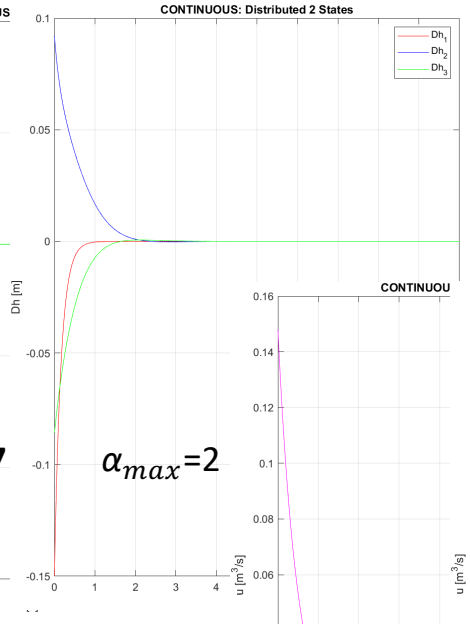
DISTRIBUTED 2

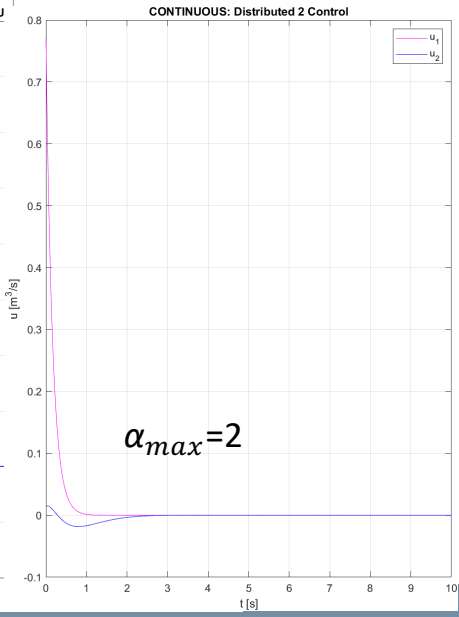












Controller Design
Continuous time

A good trade off between performance and control effort is to select α_{max} = 1.7

At this point to have a more robust control for more condition we fixed that α_{max} and reduce ϑ_{max} .

for the previous x_0 the advantages of it seems negligible, but for new randomly selected more critical x_0, we found that a robust choice is for ϑ_{max} = 15°

Controller design:

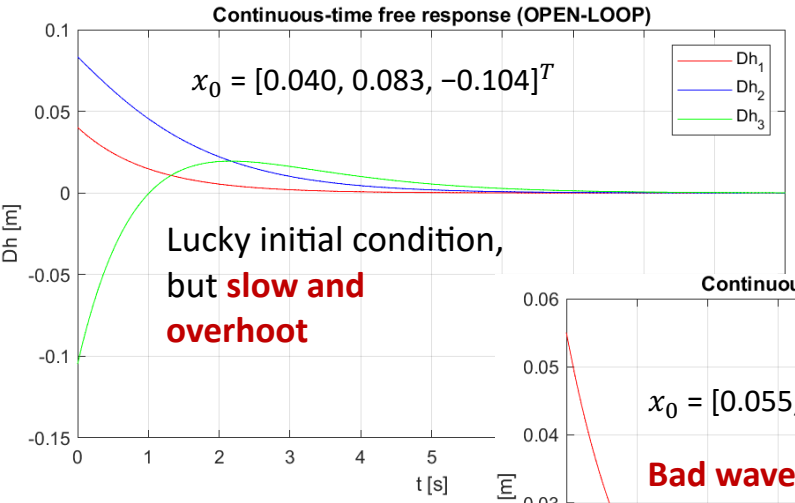
α_{max} = 1.7, ϑ_{max}= 15

$$K_C = \begin{bmatrix} -1.20 & -0.02 & 0.26 \\ -1.19 & -1.58 & -0.74 \end{bmatrix} \qquad K_{De} = \begin{bmatrix} \mathbf{-4.55} & 0 & 0 \\ 0 & -1.96 & -1.11 \end{bmatrix}$$

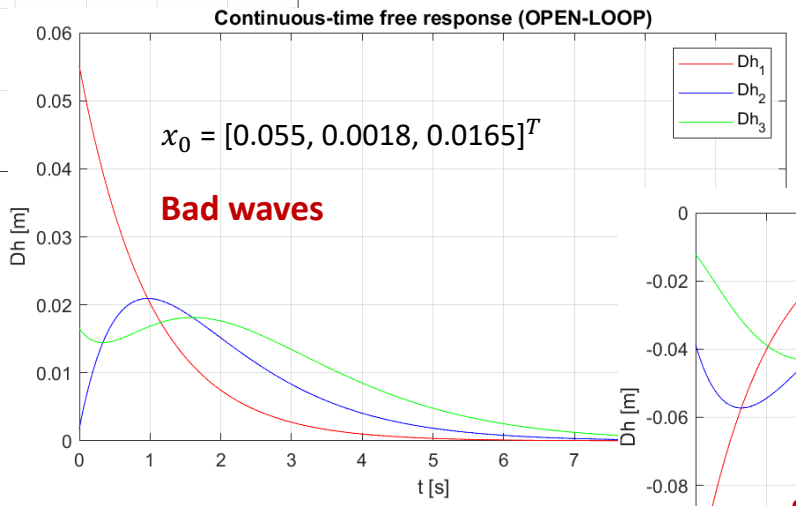
$$K_{Di1} = \begin{bmatrix} -0.85 & 0 & 0 \\ -0.93 & -1.71 & -0.95 \end{bmatrix} \qquad K_{Di2} = \begin{bmatrix} \mathbf{-4.2} & 0.37 & 0.36 \\ 0 & -1.82 & -1.04 \end{bmatrix}$$

Now let's test the performance of the closed loop system with this controller design, using the same random initial conditions analyzed in open loop

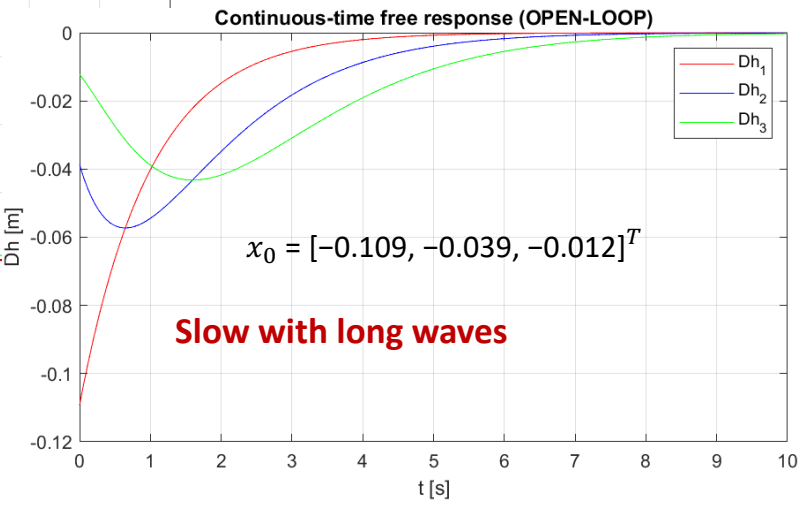
Controller Design
Continuous time



Lucky initial condition, but **slow and overhoot**



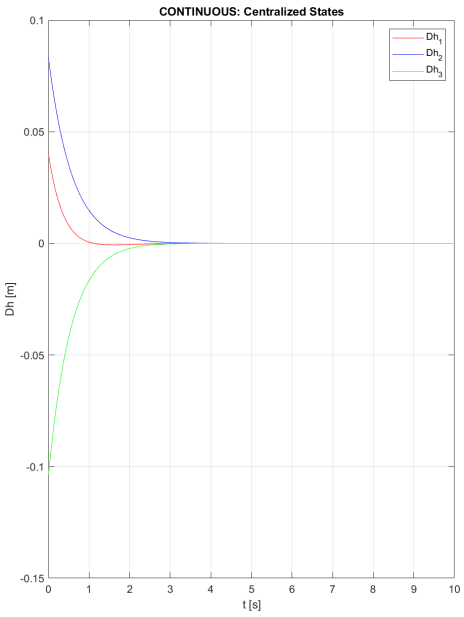
Remember the open loop free responce:

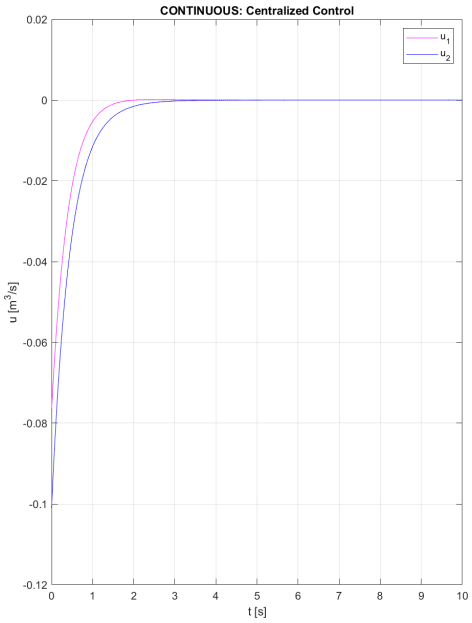


Simulation
Continuous time

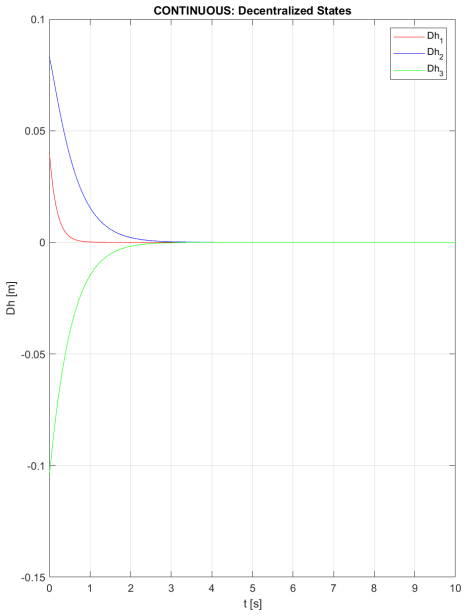
α_{max} =1.7 , ϑ_{max}= 15 $x_0 = [0.040, 0.083, -0.104]^T$

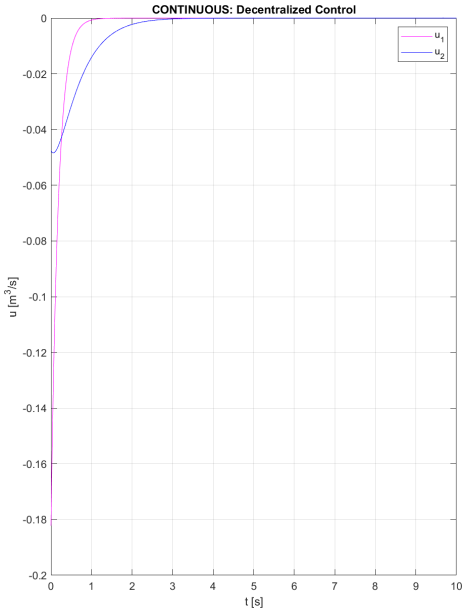
CENTRALIZED





DECENTRALIZED

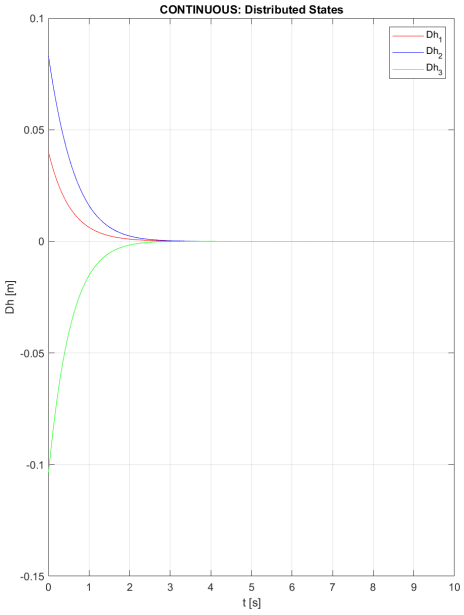


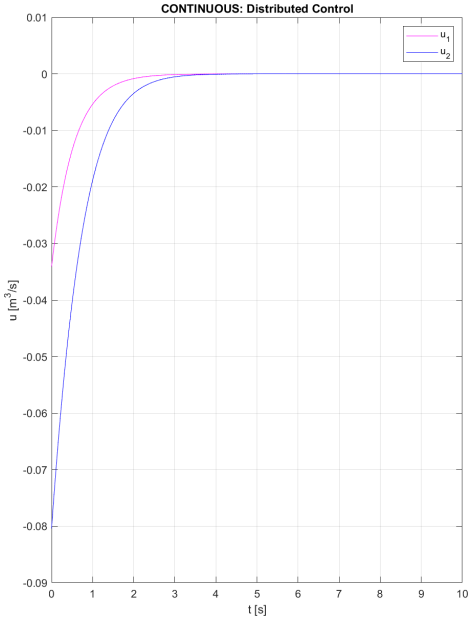


Simulation
Continuous time

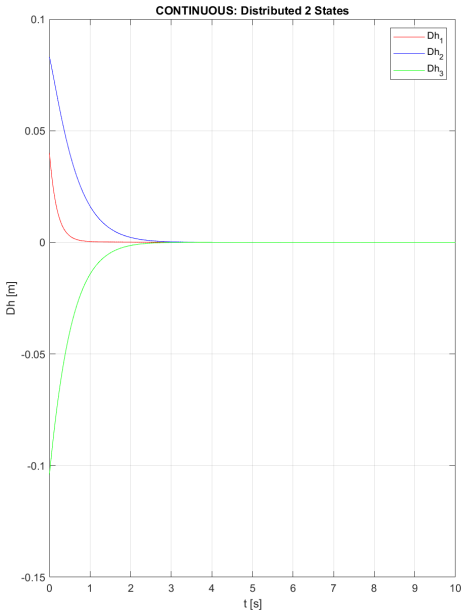
α_{max} =1.7 , ϑ_{max}=15 $x_0 = [0.040, 0.083, -0.104]^T$

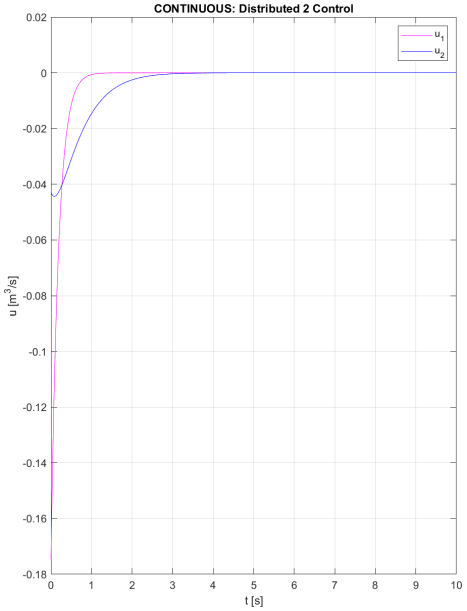
DISTRIBUTED 1





DISTRIBUTED 2

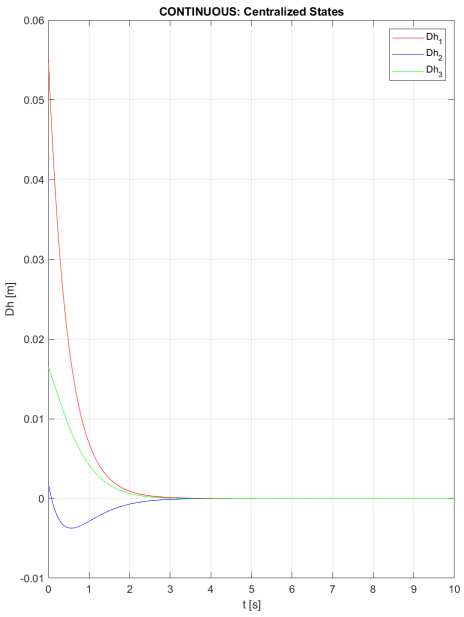


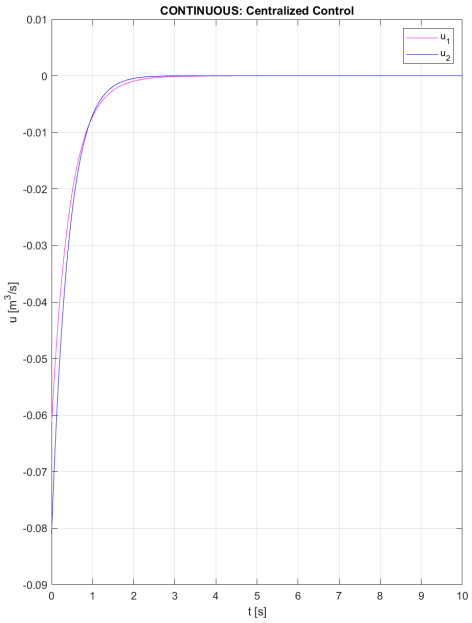


Simulation
Continuous time

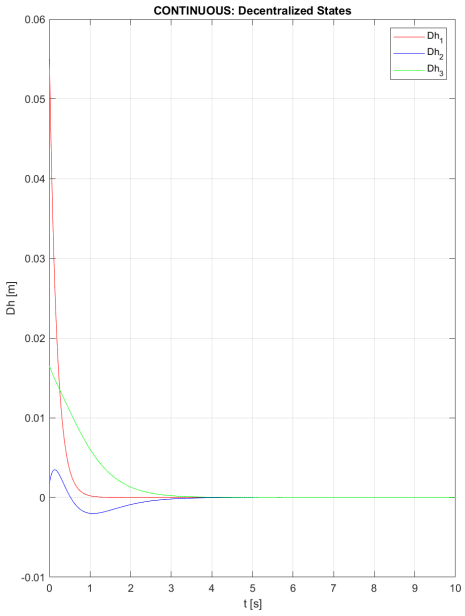
α_{max} = 1.7, ϑ_{max}=15 $x_0 = [0.055, 0.0018, 0.0165]^T$

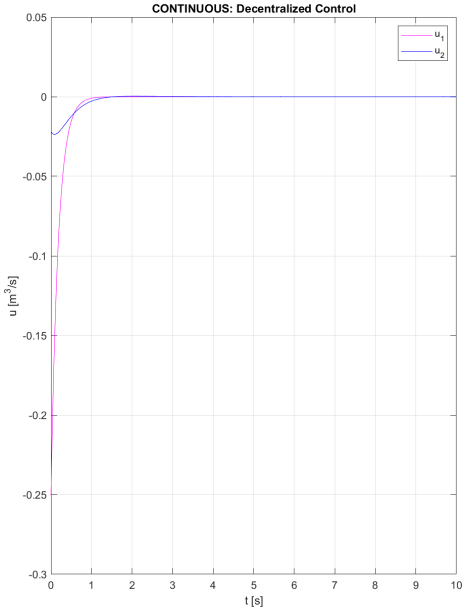
CENTRALIZED





DECENTRALIZED

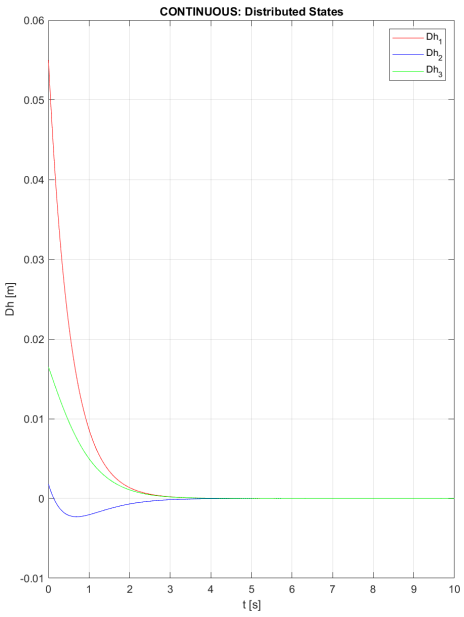


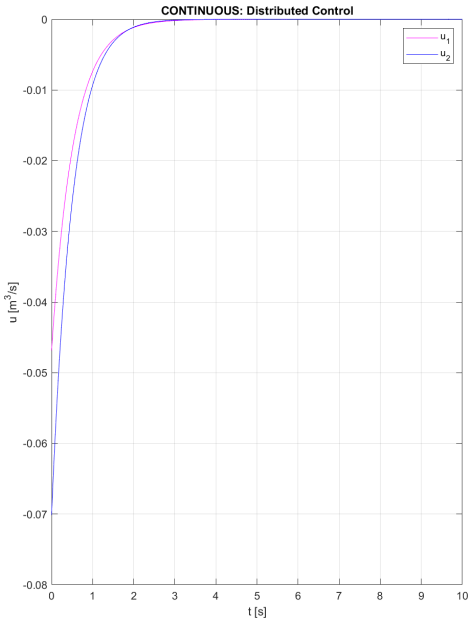


Simulation
Continuous time

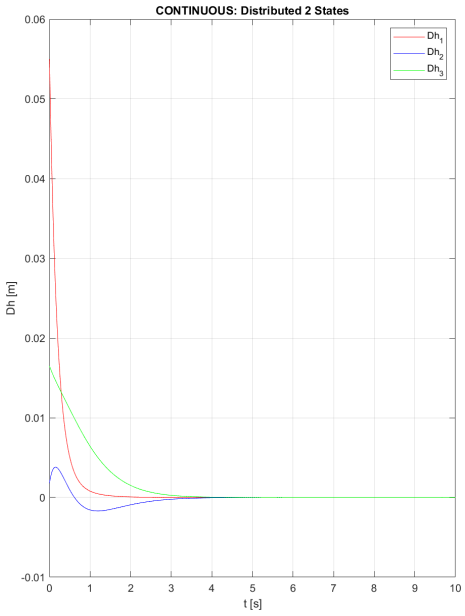
α_{max} =1.7 , ϑ_{max}=15 x_0 = [0.055, 0.0018, 0.0165]^T

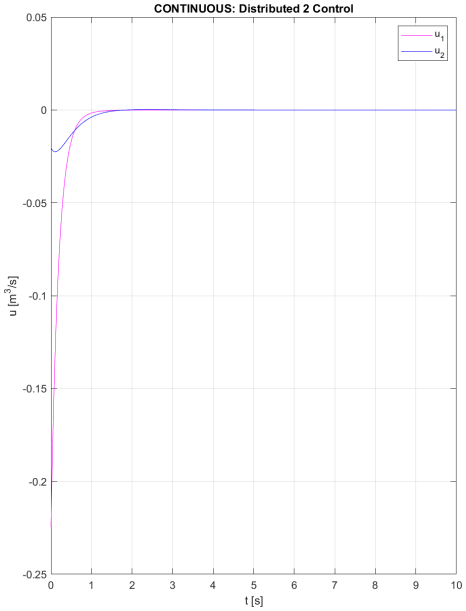
DISTRIBUTED 1





DISTRIBUTED 2

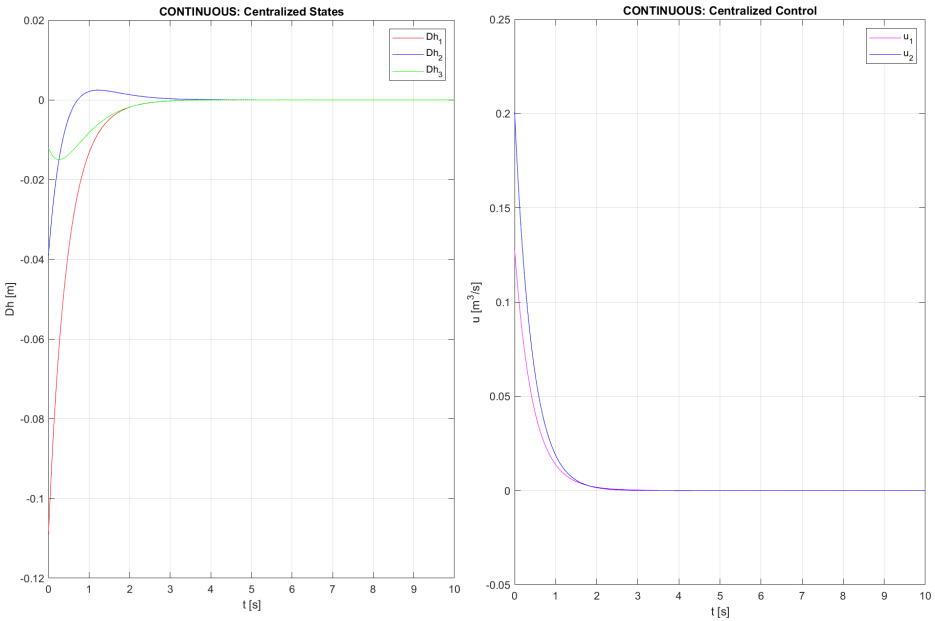




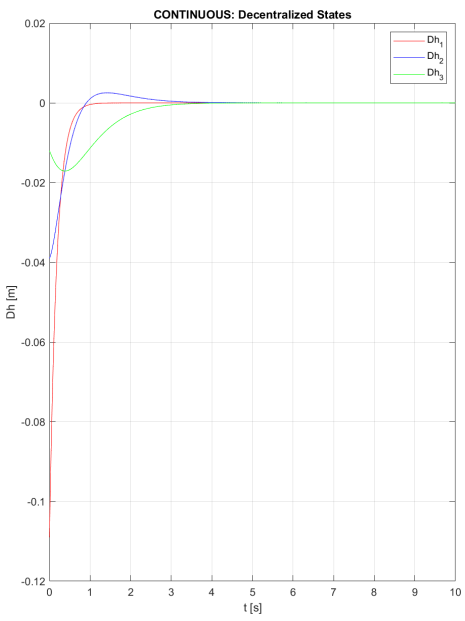
Simulation
Continuous time

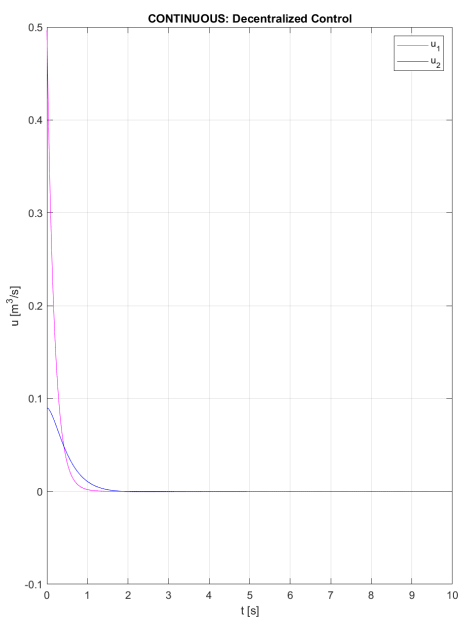
α_{max} =1.7 , ϑ_{max}=15 $x_0 = [-0.109, -0.039, -0.012]^T$

CENTRALIZED



DECENTRALIZED

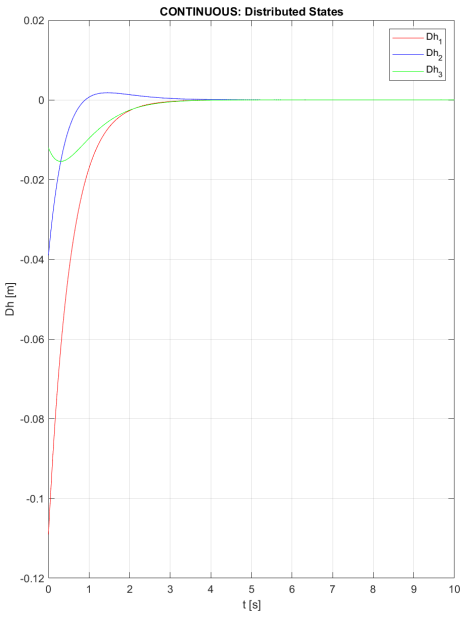


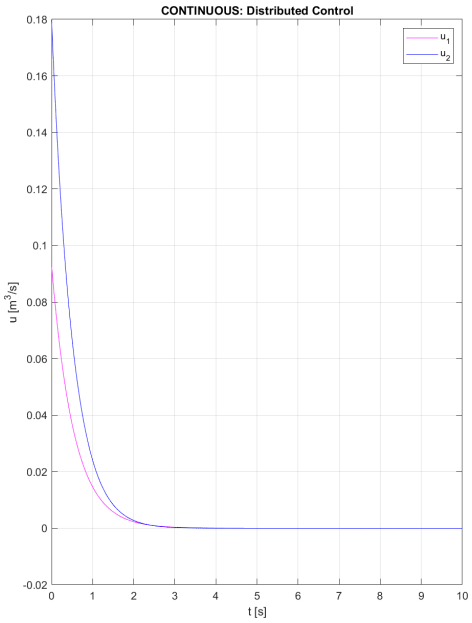


Simulation
Continuous time

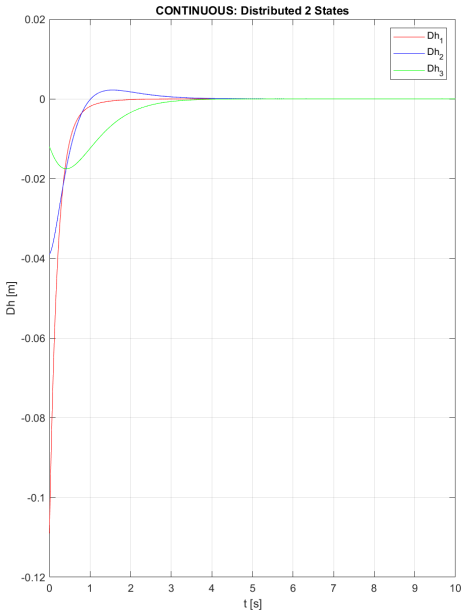
α_{max} = 1.7, ϑ_{max}=15 $x_0 = [-0.109, -0.039, -0.012]^T$

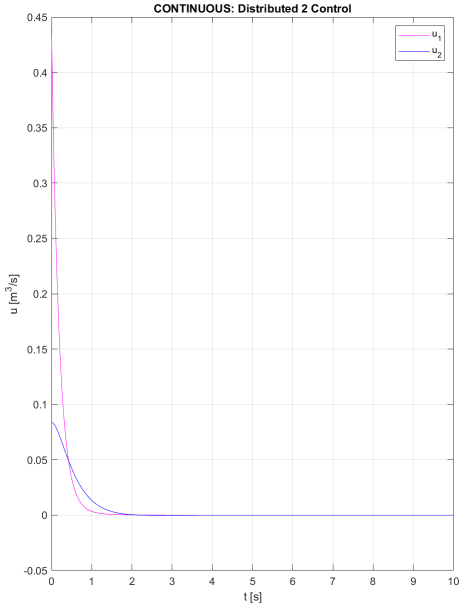
DISTRIBUTED 1





DISTRIBUTED 2



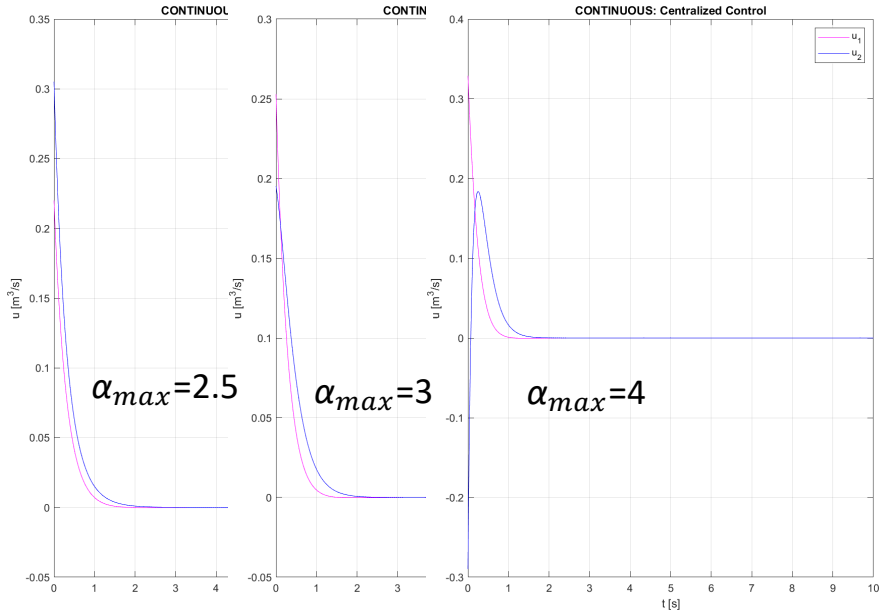


Simulation
Continuous time

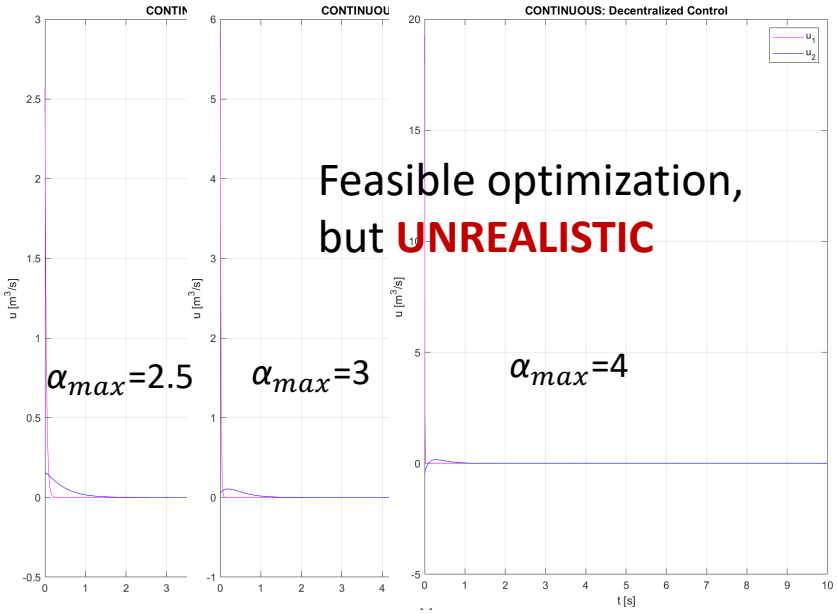
Even if we are happy with this performance, If our goal is to speed up even more (mantaining robustness against oscillations), some interesting aspects show up when we push even further the performance. If we look to the control action, in particular:

$\alpha_{max} > 2$, $\vartheta_{max}=15$ $\quad x_0 = [0.047, -0.011, -0.048]^T$

CENTRALIZED



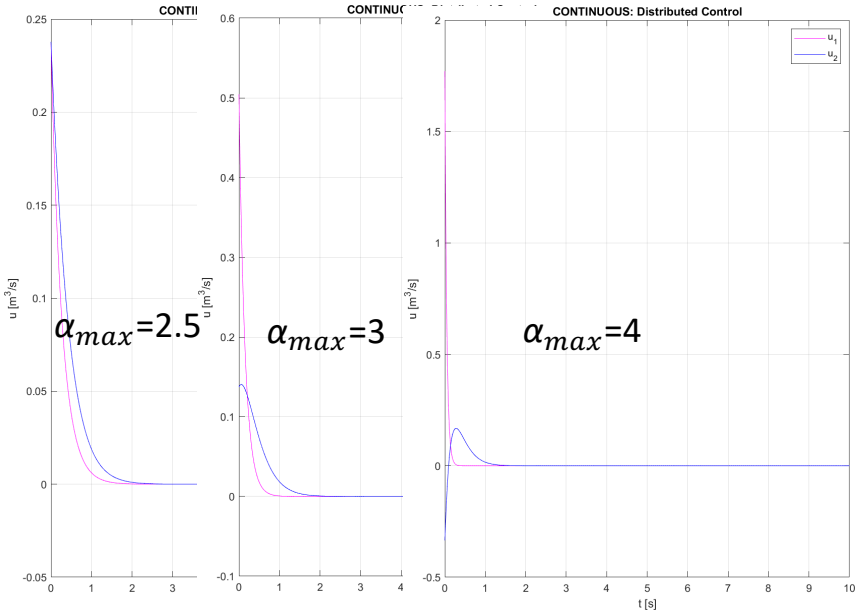
DECENTRALIZED



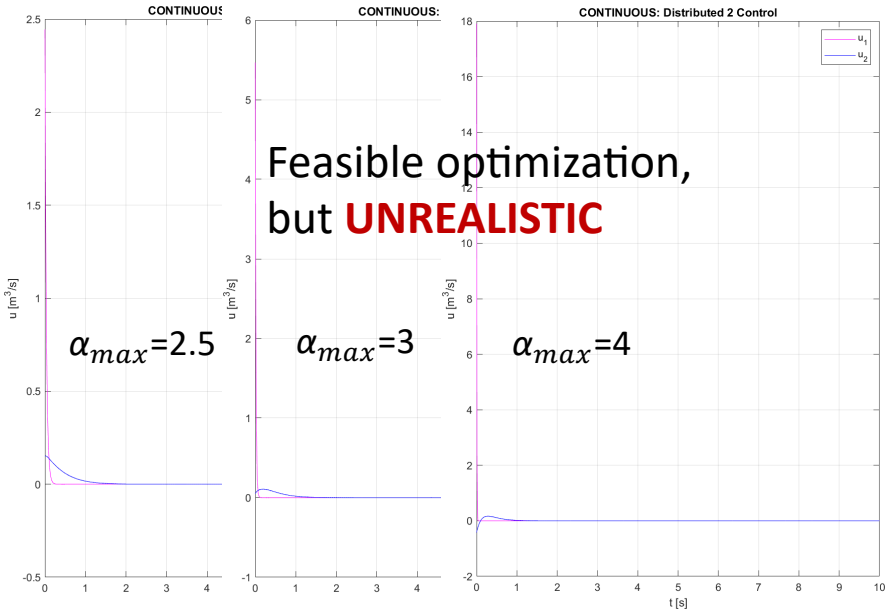
Simulation
Continuous time

$\alpha_{max} > 2$, $\vartheta_{max}=15$

DISTRIBUTED 1



DISTRIBUTED 2



If we speed up too much the system, **Decentralized** and **Distributed 2** schemes get worse very quickly, with an unrealistic control action, While **Centralied** and **Distributed 1**, even if becomes more aggressive, obtain the same performance (states behaviour is the same) with control action one order of magnitude less !

Comments
Continuous time

From our simulations, we conclude that:

- Centralized and Distributed 1 are the best option, allow to obtain very good performance with a realistic control action.
- Decentralized and Distributed 2 degenerate quickly in terms of control effort, if we ask too much (even if our optimization algorith minmize its norm).
- If the control action limitations are strict, we would have to relax our performance request, maybe reducing a bit α_{max} respect our trade-off choice of 1.7
- If instead control action is not too much restrictive, we can push the performance, but being carefully not to request too much, or the oscillation robustness is lost in some cases, due to a reactive control action that causes undesired big overshoots.
- In terms of problem feasibility through our LMI optimization algorithm, we can push the problem to the boundary of physics, α_{max} *=10*, ϑ_{max}=1 request can be solved, but obviously ask for a whole lake poured inside each tank in 1 second…

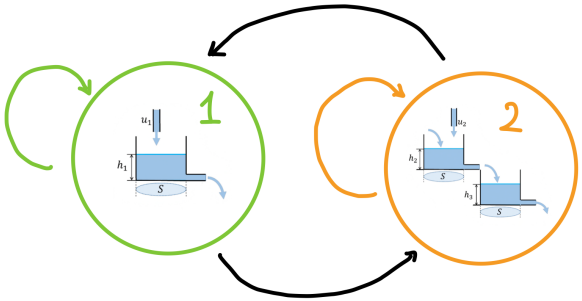
Comments
Continuous time

- Because in general Centralized control guarantees the best trade-off in terms of control effort and performance, for our system composed of just 2 sub-system, in the hypotesis of a possible full comunication, the best choice would be a distributed all-to-all comunication controller.

  (same controller gain design as centralized one, but more robust against point of failure and more flexible)

With the following information trasmission graph, and :

ContStructure_String = [1 1; 1 1]



$\mathcal{H}_2$ control
Continuous time

Negative aspects of previous procedure:

- We have to place the system poles by hand, and we don't have control on each mode, or in the control effort, we just prescribe the region.
- It can be boring and time consuming to find the best pole region for our system dynamic, because we have to make some simulations to see if we reach our desired objective.

What about an **optimal control strategy**?

If we solve the problem through LQ optimal control (casted from $\mathcal{H}_2$ minimization)

- The control law obtained is more robust, we can also model possible additional disturbances on the system, and obtain a controller robust against them.
- Our effort is just to implement the function that computes the optimal control gain, with the usual LMI definition, then to design the controller we have just to tune the best input/state deviation trade-off, with the proper choice of the matrices of weigths Q and R.

$\mathcal{H}_2$ control
Continuous time

We define a new performance output z, and a noise variable w

$$\dot{x} = Ax + Bu + B_w w$$

$$z = C_z x + D_z u$$

And to go from $\mathcal{H}_2$ to LQ control, we define the performance output matrices as:

$$C_z = \begin{bmatrix} \sqrt{q_1} & 0 & 0 \\ 0 & \sqrt{q_2} & 0 \\ 0 & 0 & \sqrt{q_3} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad D_z = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \sqrt{r_1} & 0 \\ 0 & \sqrt{r_2} \end{bmatrix}$$

To make the analysis simpler, we consider as weigths just q, r

$B_w = I_3$ or zeros(3)

(The noise matrix B_w is an identity if we want to model additional disturbances, for example leakage of water in each tank, or a zero matrix 3x3 if we don't want to model any disturbance)

At this point, we just have to implement the LMI as:

$$AY + BL + YA^T + L^T B^T + B_w B_w^T < 0$$

$$\begin{bmatrix} S & CY + DL \\ (CY + DL)^T & Y \end{bmatrix} \geq 0$$

With cost function

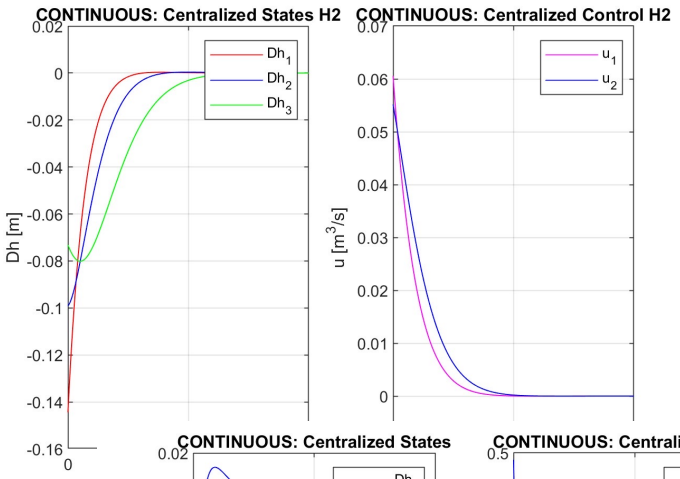
$$J = trace(S)$$

$\mathcal{H}_2$ Simulations
Continuous time

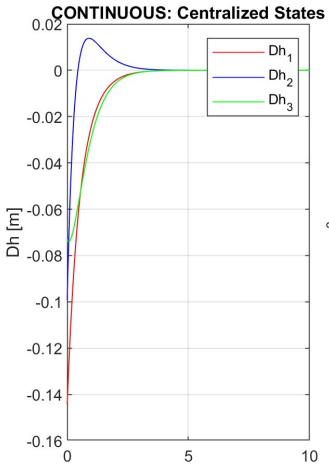
Now we are ready to test the new controller, first with control limitations **r>q**

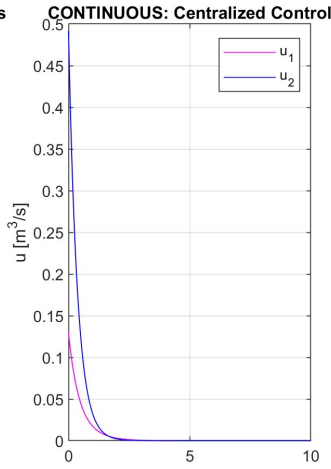
q=50, r=100 vs α_{max} = 1.7, ϑ_{max}=10 $(B_w = I_3)$ $x_0 = [-0.14, -0.09, -0.07]^T$

CENTRALIZED

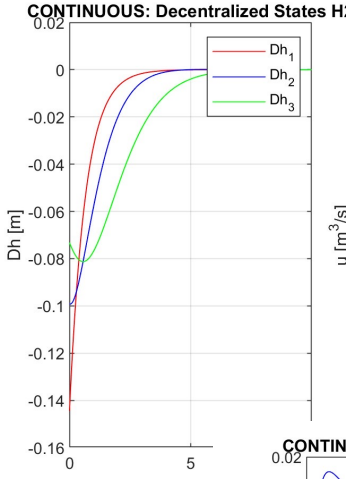


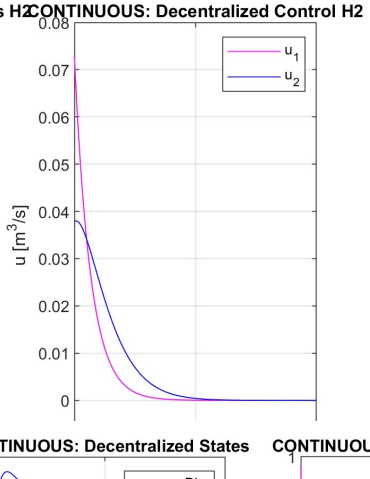
$\mathcal{H}_2$



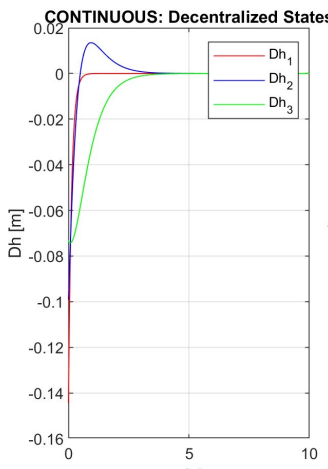


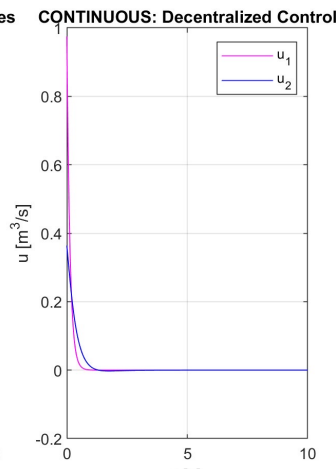
DECENTRALIZED





Pole place

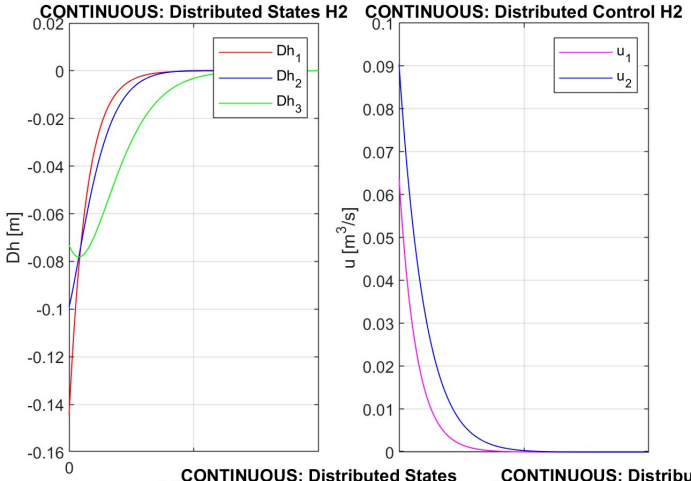




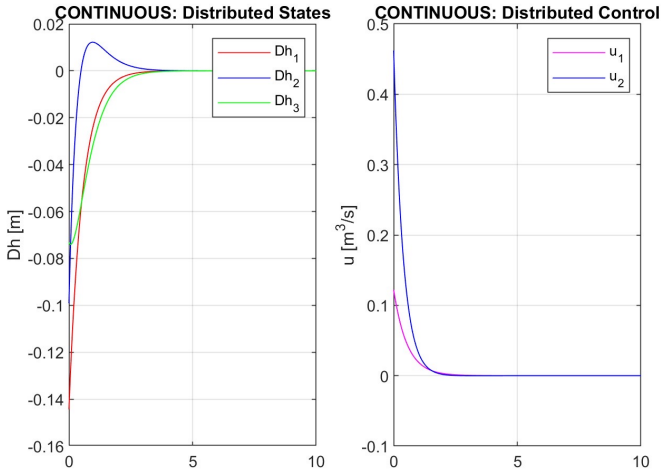
$\mathcal{H}_2$ Simulations
Continuous time

q=50, r=100 vs α_{max} = 1.7, ϑ_{max}=10 $(B_w = I_3)$ $x_0 = [-0.14, -0.09, -0.07]^T$

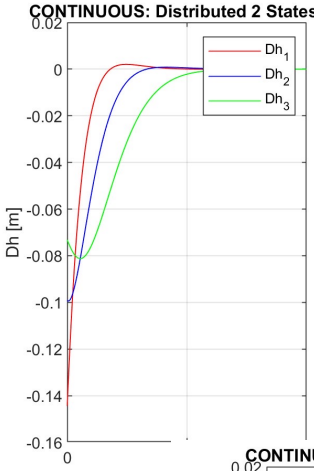
DISTRIBUTED 1

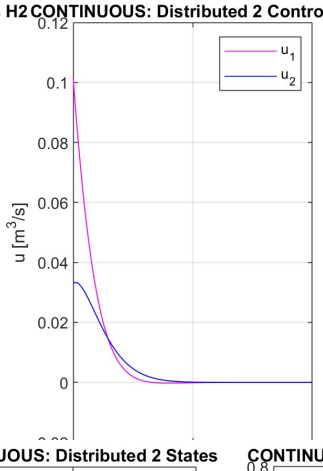


$\mathcal{H}_2$

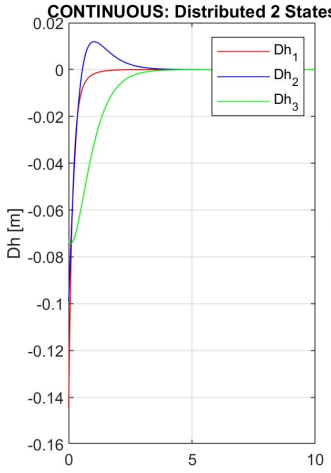


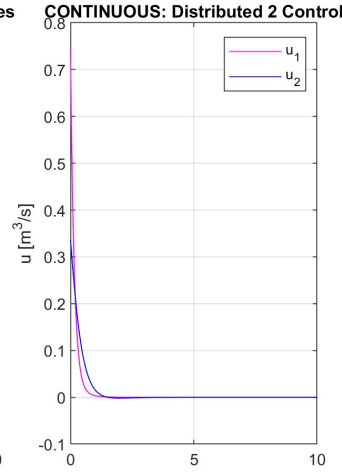
DISTRIBUTED 2





Pole place



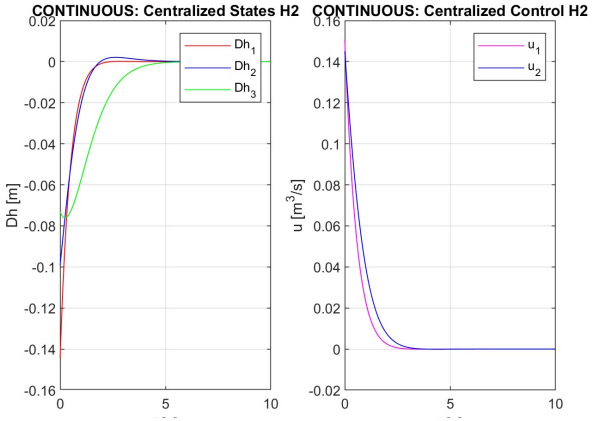


$\mathcal{H}_2$ Simulations
Continuous time

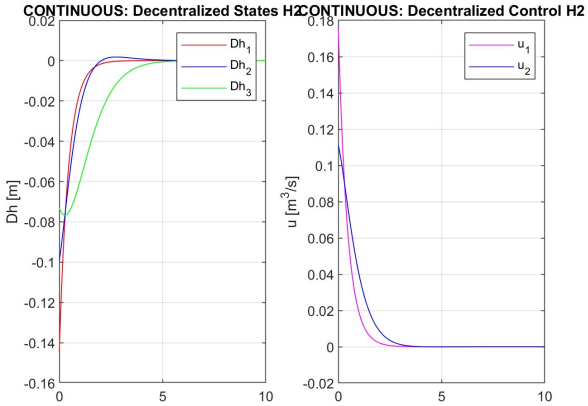
q=100, r=50 vs α_{max} = 1.7, ϑ_{max}=10 $(B_w = I_3)$ $x_0 = [-0.14, -0.09, -0.07]^T$

Now with faster state convergence, q> r

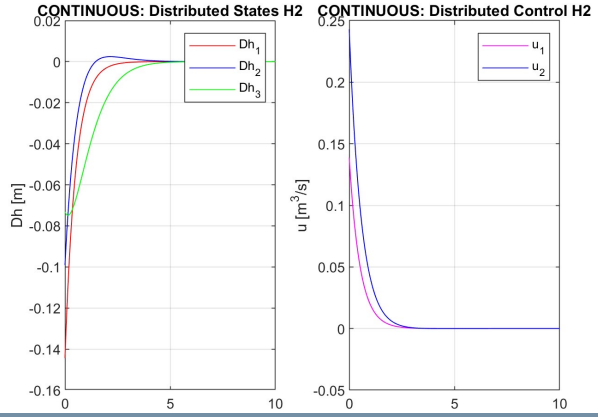
CENTRALIZED



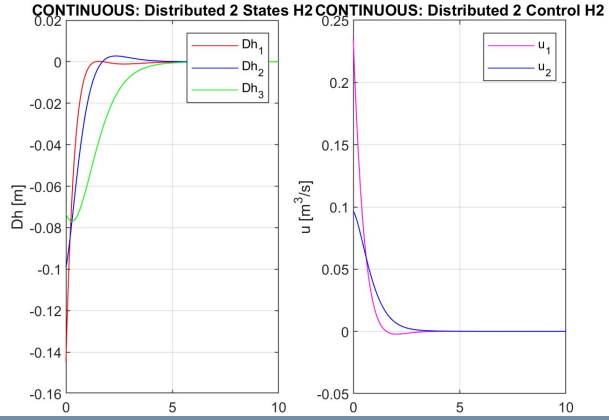
DECENTRALIZED



DISTRIBUTED 1



DISTRIBUTED 2



$\mathcal{H}_2$ Comments
Continuous time

As we expect, we observe that:

- When **r > q**, control weigth is higher, so optimization focus on limiting the control action, with the guarantee of convergence with a slower dynamic. This conservative condition allow us to have no overshoot, but slows down the system a lot.
- When **q > r**, state weigth is higher, so the optimization puts more control action effort to speed up convergence (anyway we don't exagerate with the request, so the control action is still acceptable). It is a little bit slower than the previous control design, but with an overshoot almost null, and a control action much more limited.
- When **q=r**, same relative weigth, we are in the middle, the system is still a little bit slower, not so much interesting.
- For $\boldsymbol{B_w}$ **= zeros(3)**, the system without disturbance becomes 'less reactive' and even slower, we don't show this results. Even if not present, include some noise modelling with $B_w = I_3$ can help to achieve better robustness.

$\mathcal{H}_2$ Comments
Continuous time

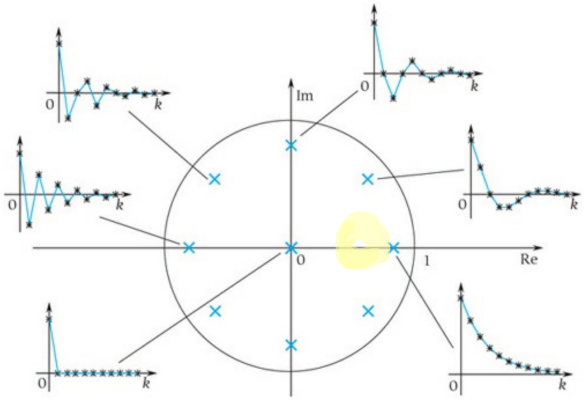
- Notice also that in general, even pushing forward the performance, there is not an evident 'better' control structure, so we are free to choice accordingly to our resources.
- In general by a simple choice of Q and R we can afford our goal according to the control problem specifications.
- Another aspect that we don't explore is the possibility of giving different weigths to each state or control variable, using different q_i and r_i. This allows us to obtain a fine control on the states and on the control variables. If for example we want u_1 to have more restrictive limitations than u_2 we just have to chose $r_1 > r_2$. Same reasoning for the choice of q_i weigths.

Controller Design
Discrete time

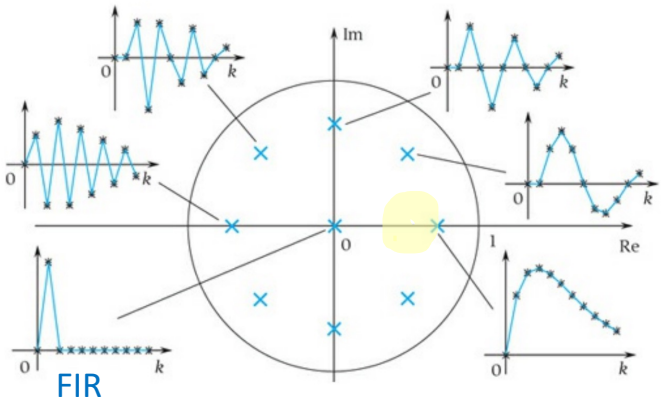
As done in continuous time, to reach good performance we search for the best placement by tentatives.

In particular, we chose the **best pole regions**, according to the modes behaviour and the overshoot reduction region:

When F diagonalizable

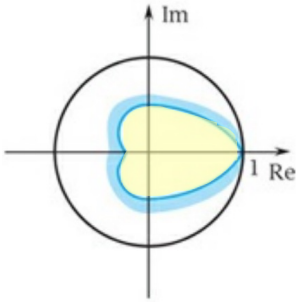


When F non diagonalizable



FIR

Region where $\xi \geq \bar{\xi}$



Let's simulate what happens, in terms of **state performance and control action**, if using our LMI we try to move the poles inside a region with desired modes behaviour

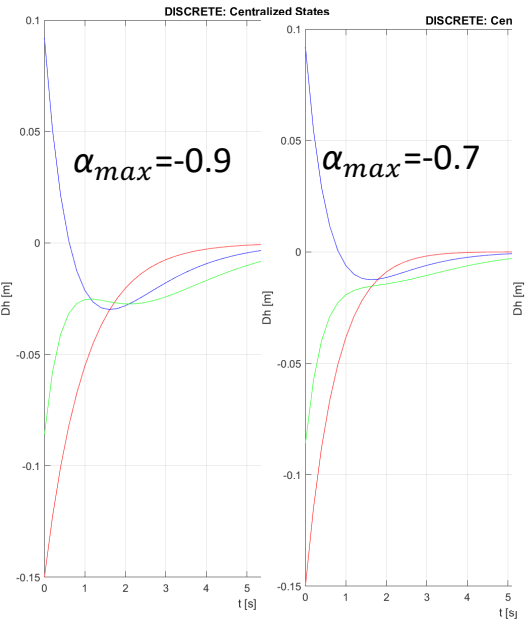
Simulation
Discrete time

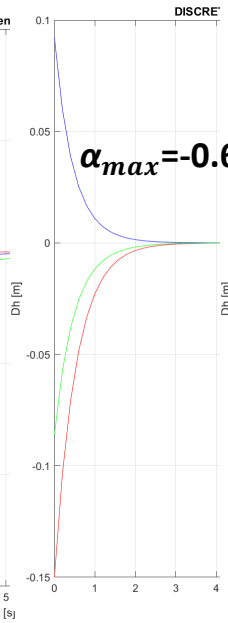
Again we randomly pick an initial state x_0 = [−0.109, −0.039, −0.012]^T and Try :

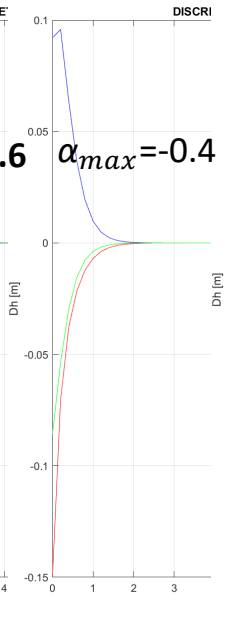
Fix **ρ_{max} = 0.1** (remain in desired ξ region) **$|\alpha_{max}|$= variable ≤ 0.9**

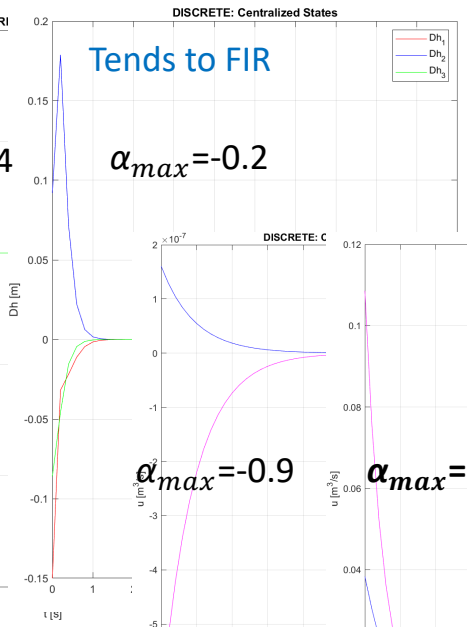
(From now on, consider **h=0.2** as discussed before. Than we will motivate furthermore this choice)

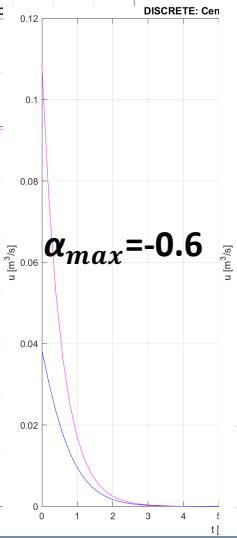
CENTRALIZED

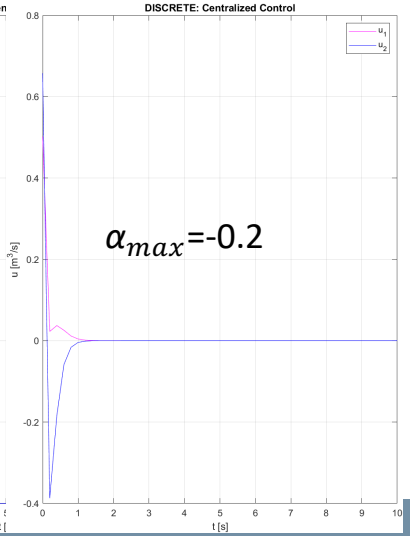








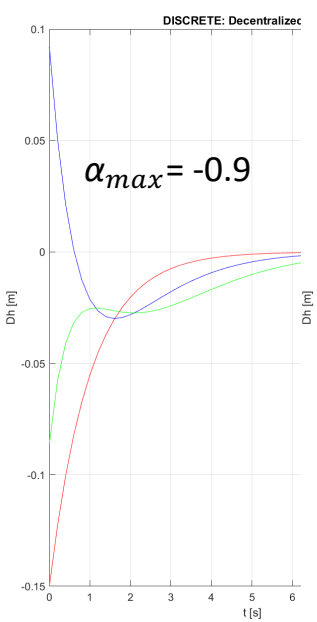


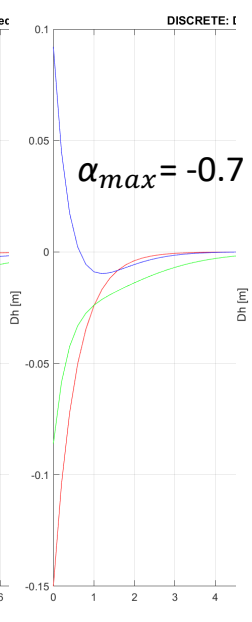


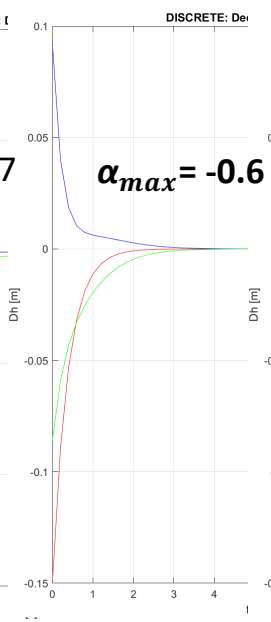
Controller Design
Discrete time

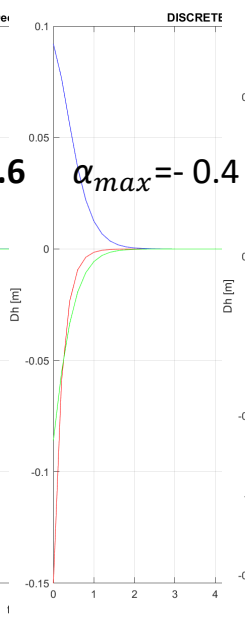
ρ_{max} = 0.1, α_{max} = variable

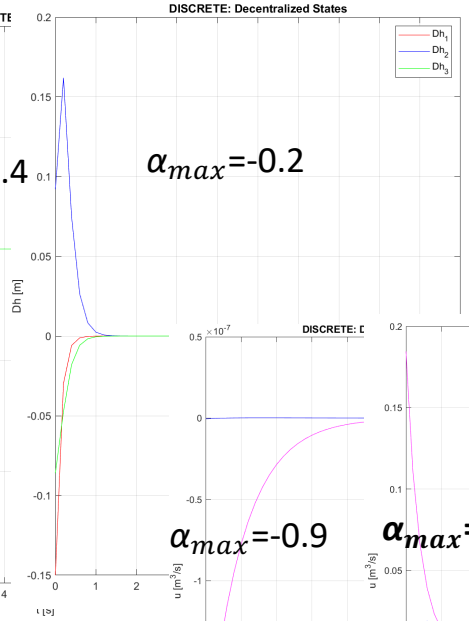
DECENTRALIZED

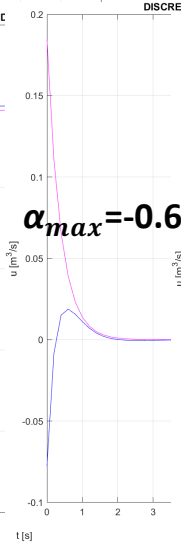


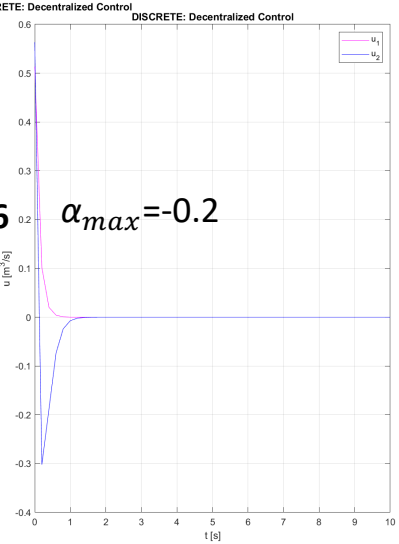








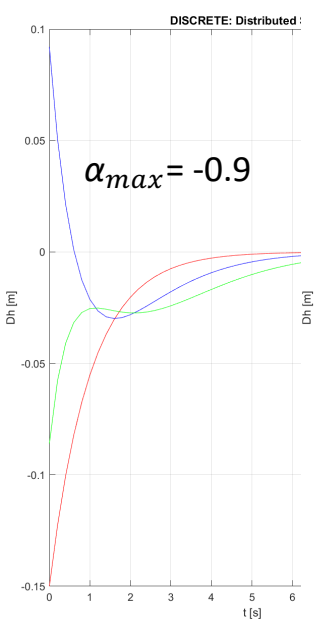


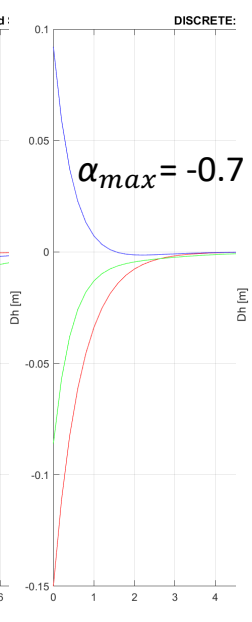


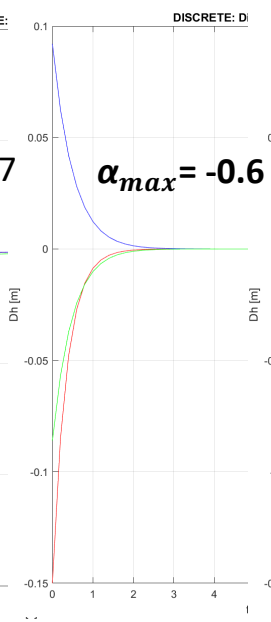
Controller Design
Discrete time

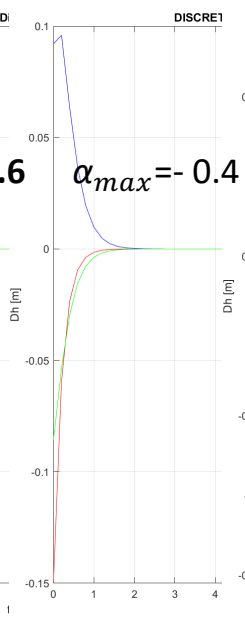
ρ_{max} = 0.1, α_{max} = variable

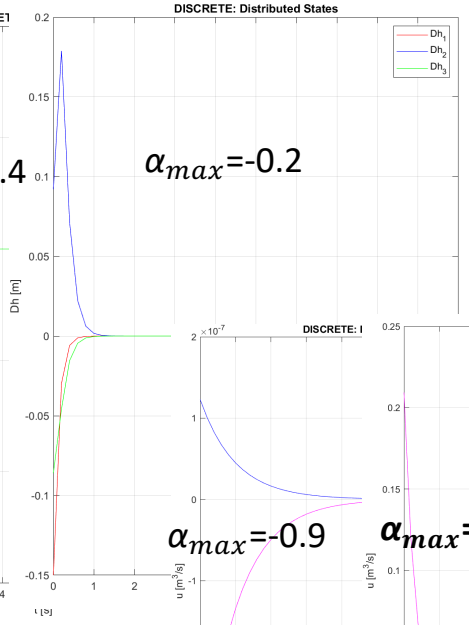
DISTRIBUTED 1

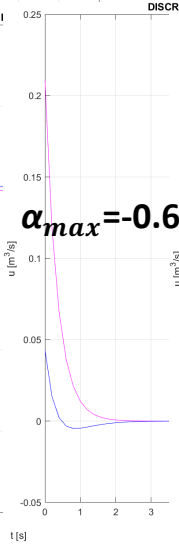


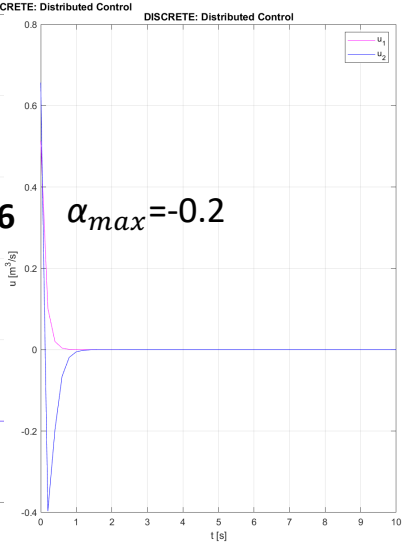








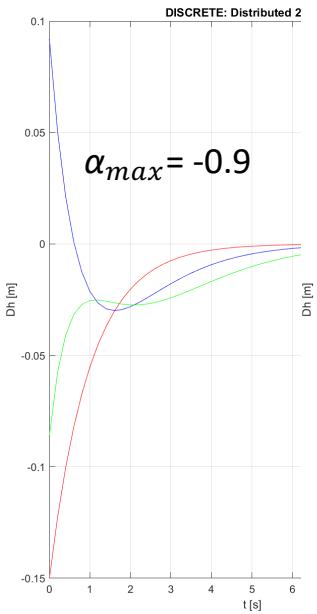


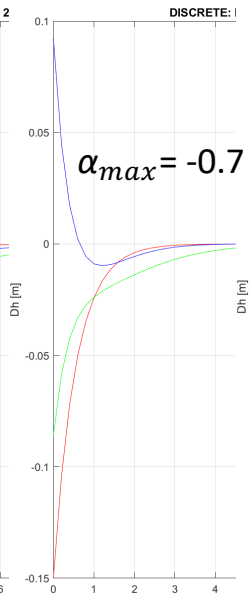


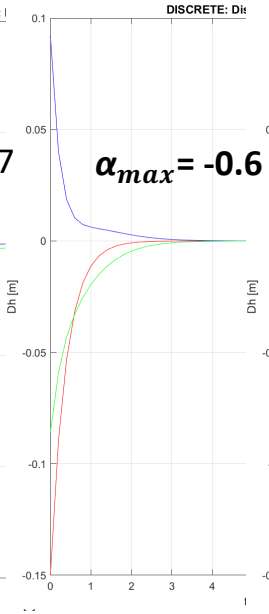
Controller Design
DIscrete time

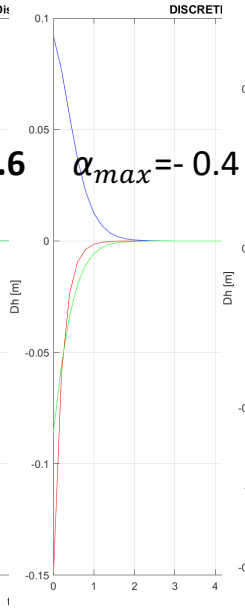
ρ_{max} = 0.1, α_{max} = variable

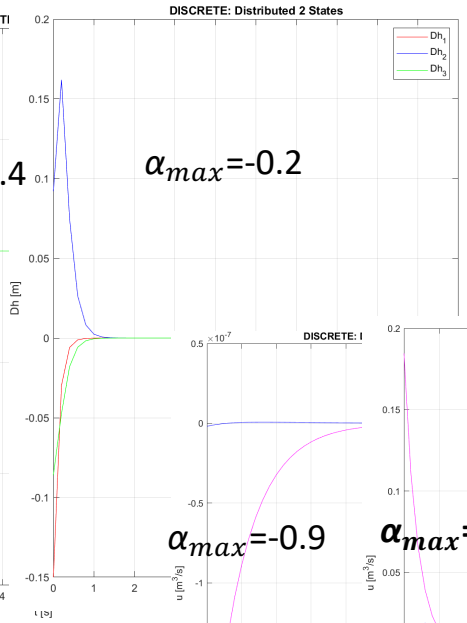
DISTRIBUTED 2

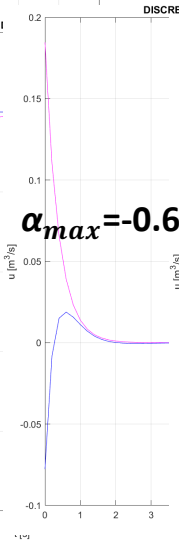


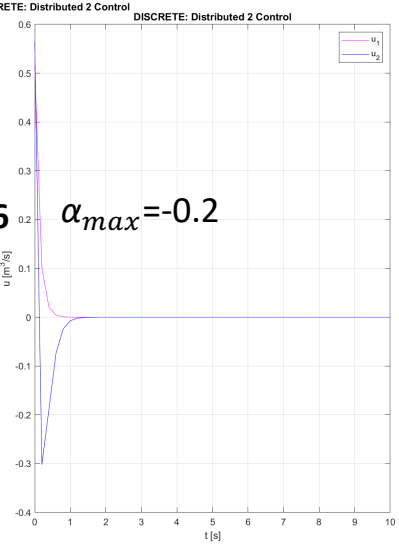












Controller Design
Discrete time

Overall, repeating those simulations for other random initial conditions we can conclude that a good performance/control trade-off is obtained for:

ρ_{max} = 0.1, α_{max} = -0.6

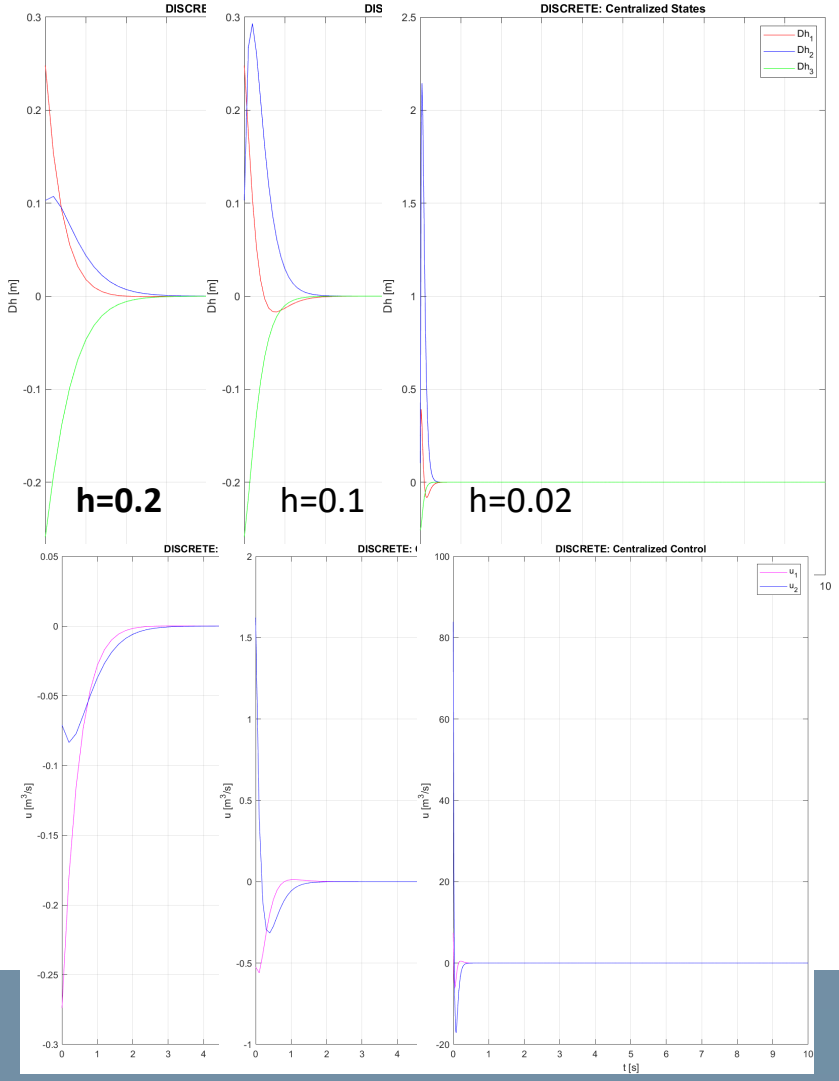
(No reason to limit ρ_{max} furthermore, it is already enoght robust against overshoot)

What about the **sampling time h**? maybe reducing it we can improve the performance.

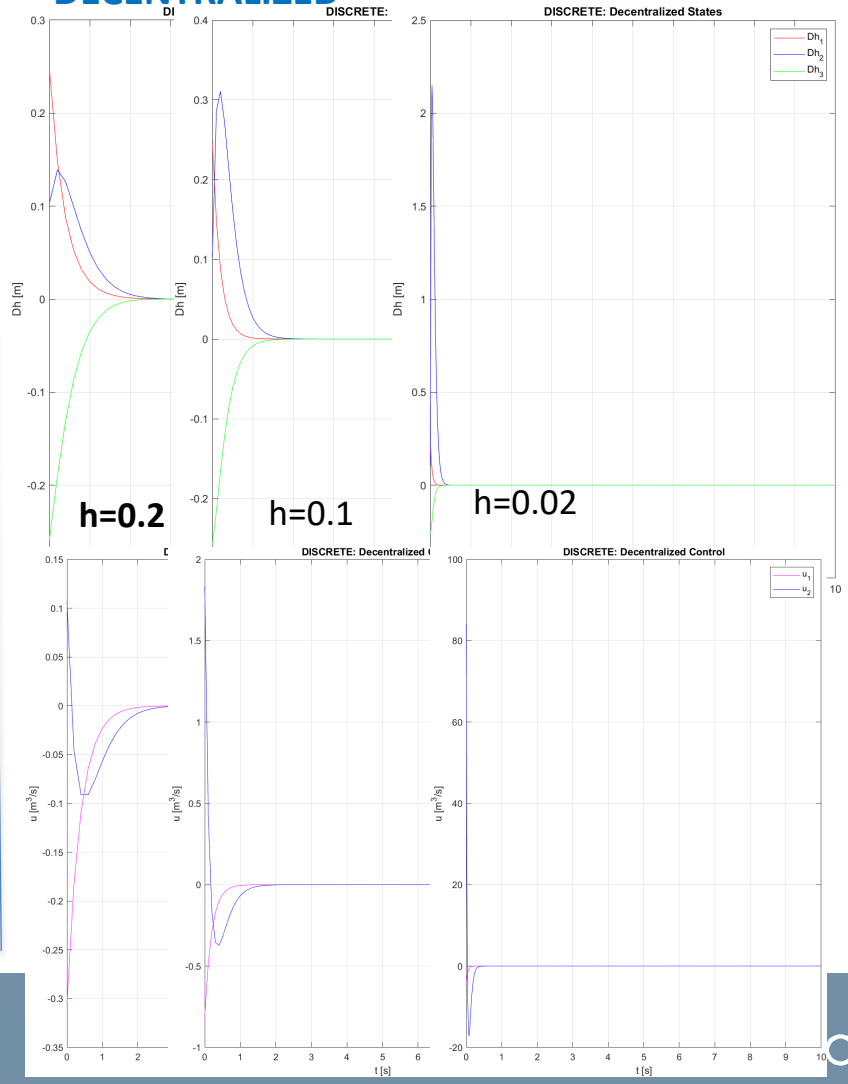
Let's fix that ρ_{max}, α_{max} and simulate the system decreasing h with a general random initial condition: x_0 = [0.248,0.103,−0.259]^T

Simulation
Discrete time

CENTRALIZED

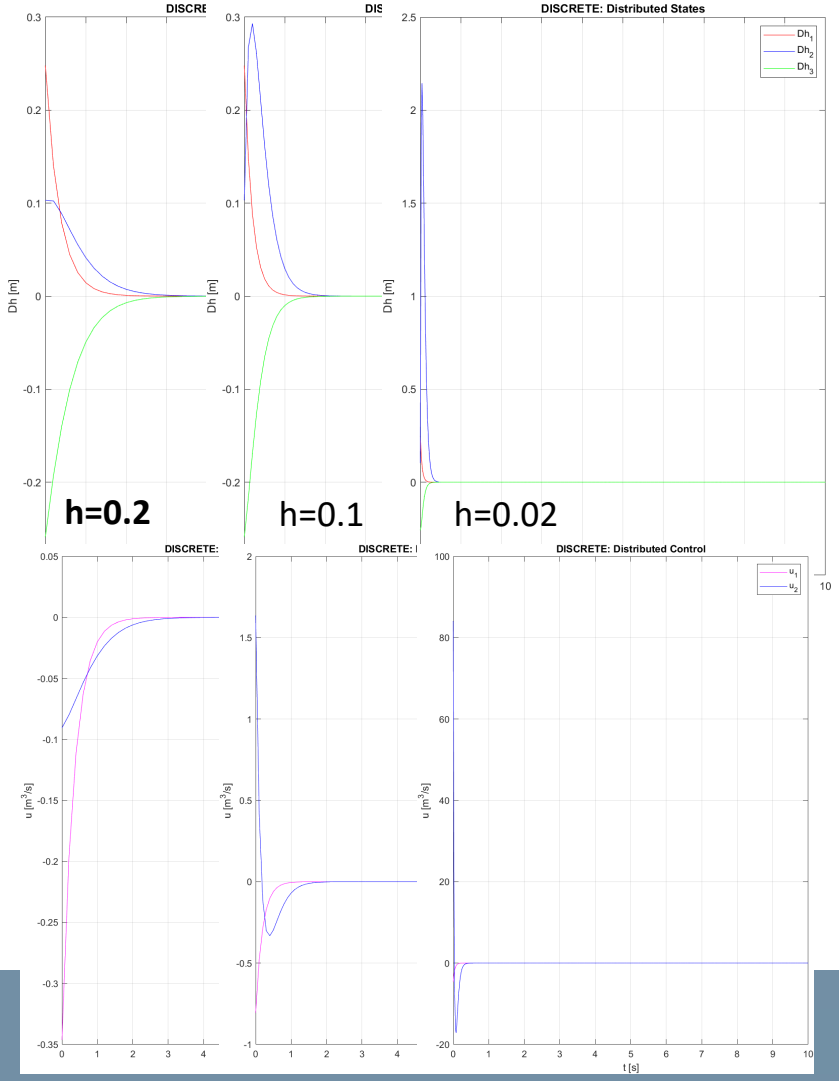


DECENTRALIZED

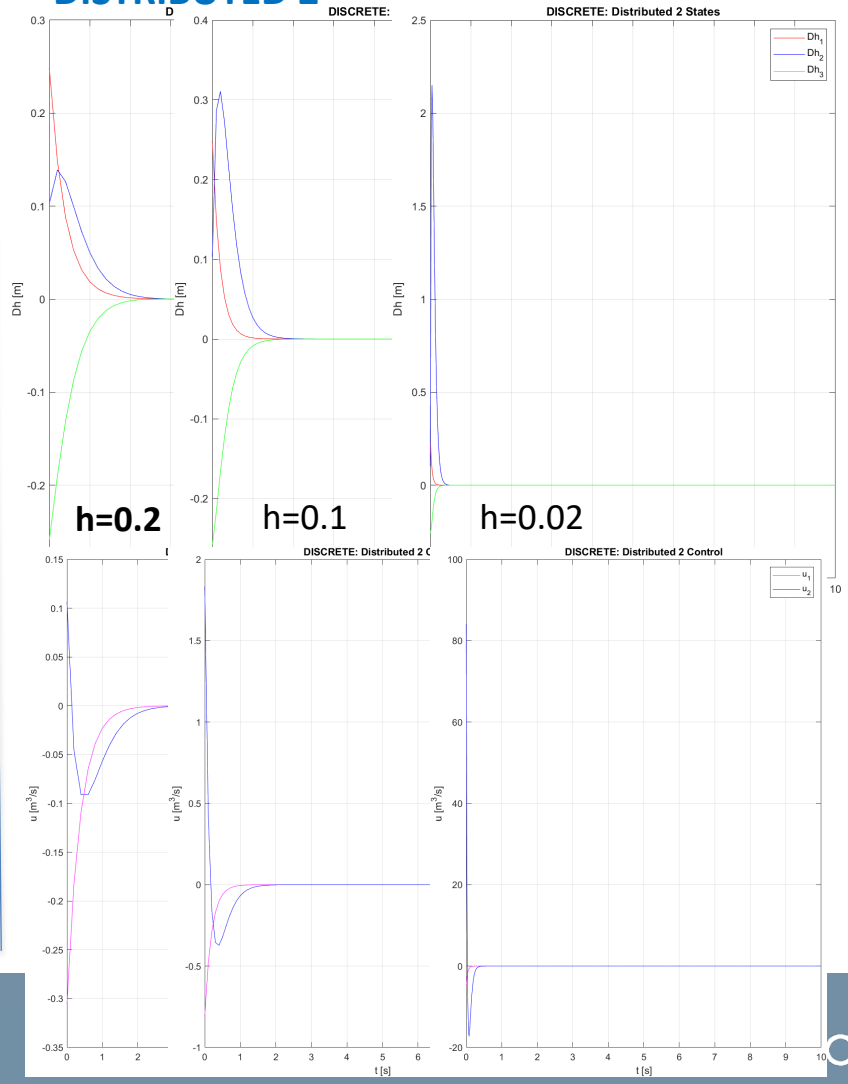


Simulation
Discrete time

DISTRIBUTED 1



DISTRIBUTED 2



Controller Design
Discrete time

For h < 0.2 the performance get worse, the control action becomes not realistic and too reactive. Also, if we reduce h too much we have numerical problems.

Controller design:

α_{max} = -0.6, ρ_{max} = 0.1, h = 0.2

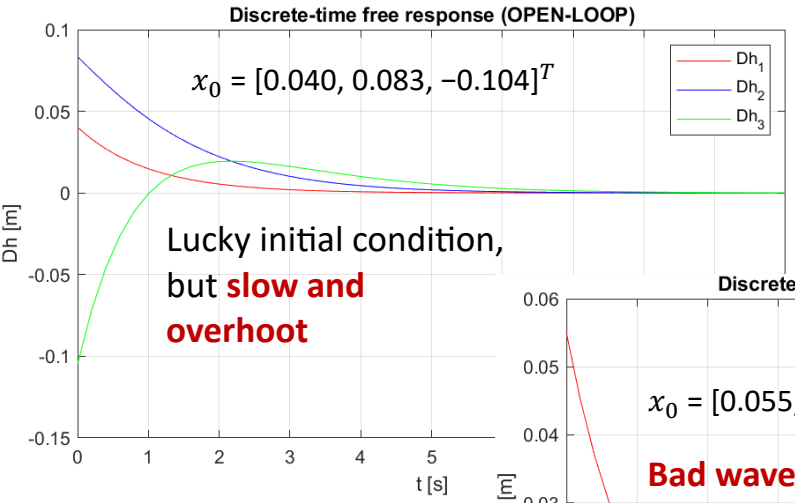
$$K_C = \begin{bmatrix} -0.88 & -0.09 & 0.17 \\ -0.68 & -1.70 & -1.06 \end{bmatrix} \qquad K_{De} = \begin{bmatrix} -1.23 & 0 & 0 \\ 0 & -1.95 & -1.18 \end{bmatrix}$$

$$K_{Di1} = \begin{bmatrix} -1.39 & 0 & 0 \\ -0.77 & -1.82 & -1.11 \end{bmatrix} \qquad K_{Di2} = \begin{bmatrix} -1.23 & 6 * 10^{-7} & 3 * 10^{-7} \\ 0 & -1.95 & -1.18 \end{bmatrix}$$

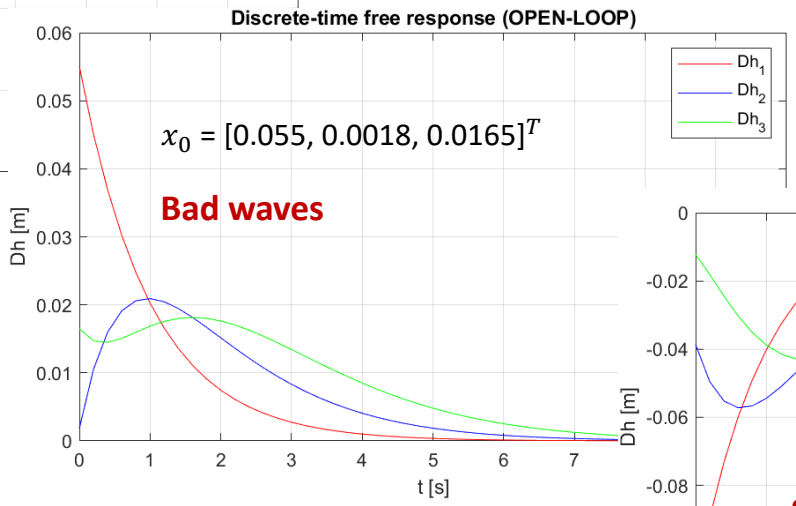
Interestingly Distributed 2 gain is equal to Decentralized one (again not so useful information structure)

Now Let's test the performance of the closed loop system with this controller design, using the same random initial conditions analyzed in open loop

Controller Design
Discrete time

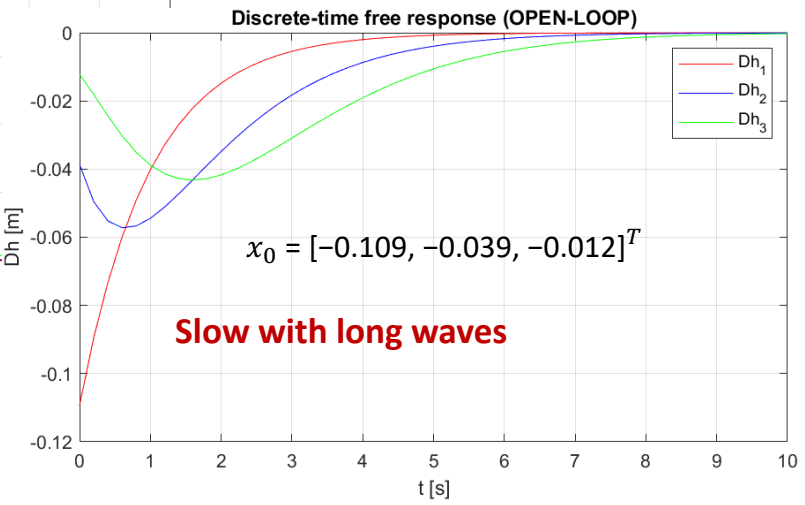


Lucky initial condition, but **slow and overhoot**



Bad waves

Remember the open loop free responce:
(exactly equal to continuous time)

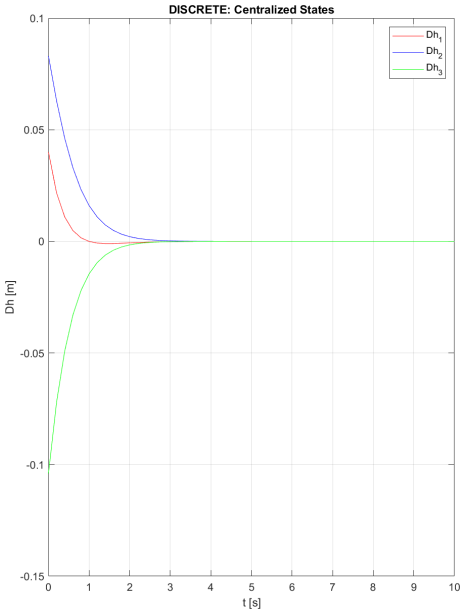


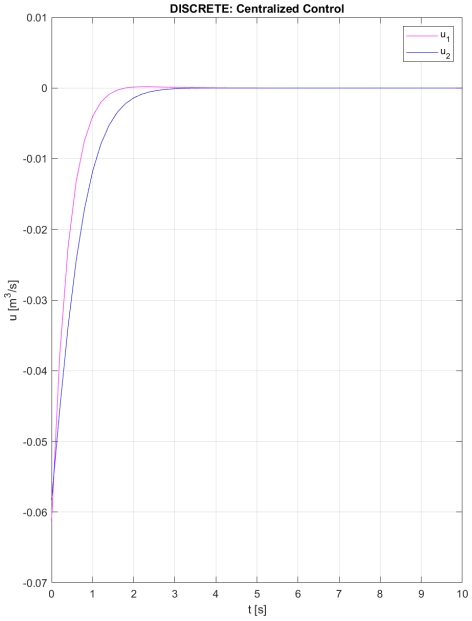
Slow with long waves

Simulation
Discrete time

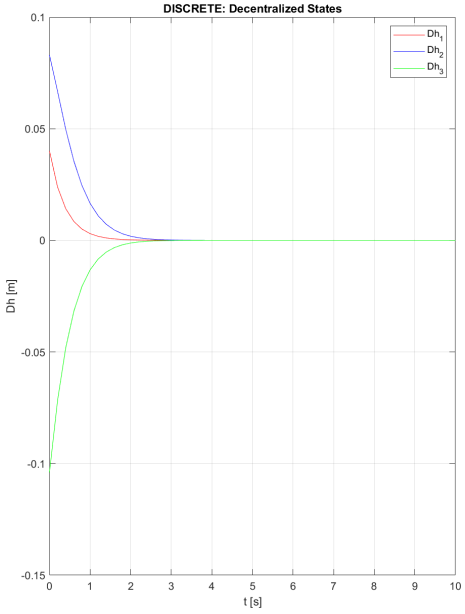
α_{max} =-0.6 , ρ_{max} = 0.1, h = 0.2 $x_0 = [0.040, 0.083, -0.104]^T$

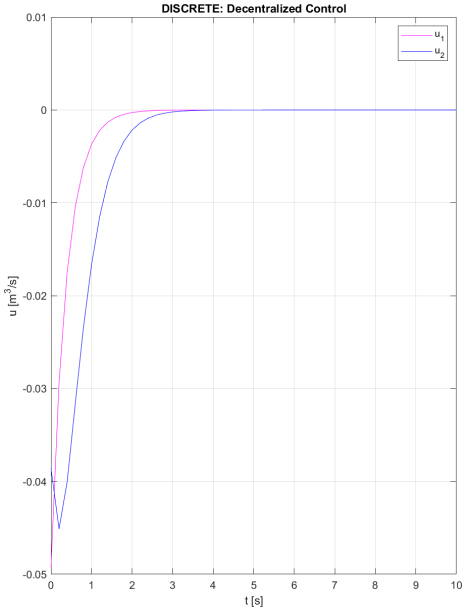
CENTRALIZED





DECENTRALIZED

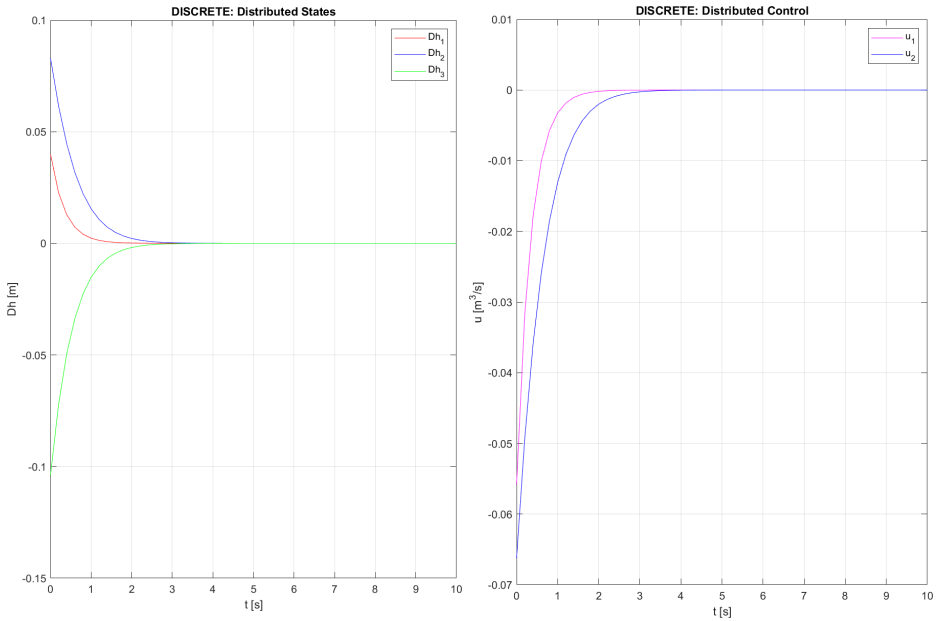




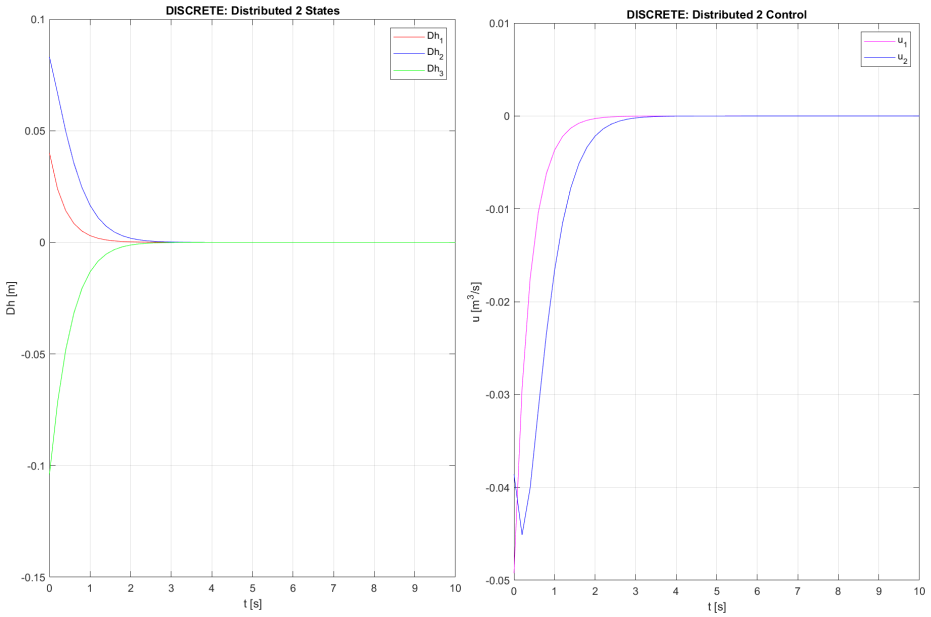
Simulation
Discrete time

α_{max} =-0.6 , ρ_{max} = 0.1, h = 0.2 $x_0 = [0.040, 0.083, -0.104]^T$

DISTRIBUTED 1



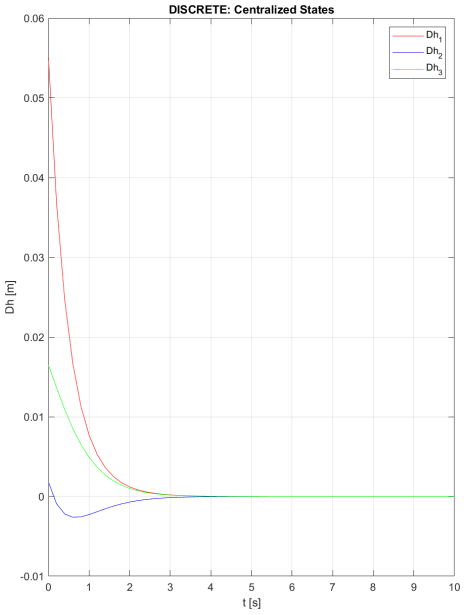
DISTRIBUTED 2

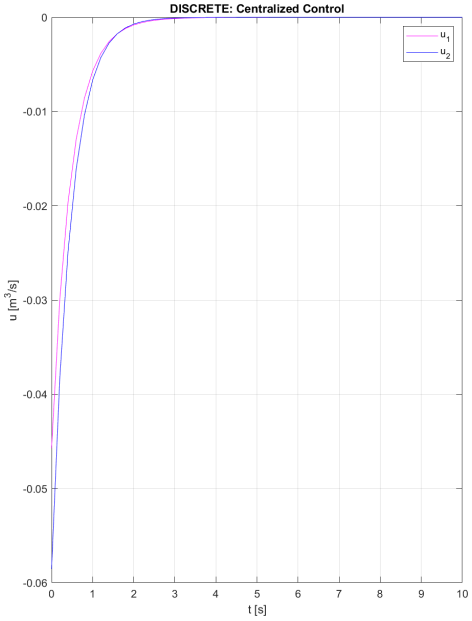


Simulation
Discrete time

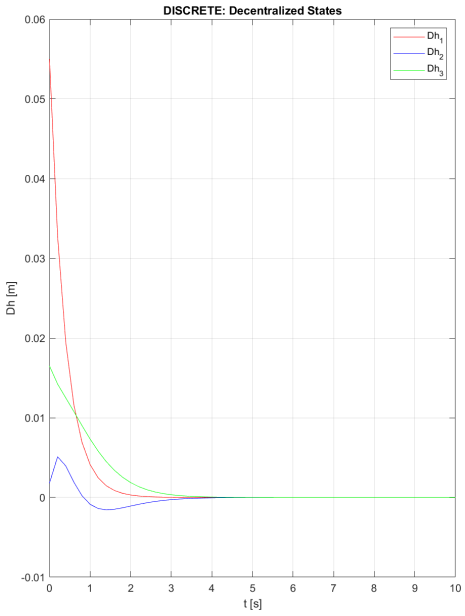
α_{max} =-0.6 , ρ_{max} = 0.1, h = 0.2 x_0 = [0.055, 0.0018, 0.0165]^T

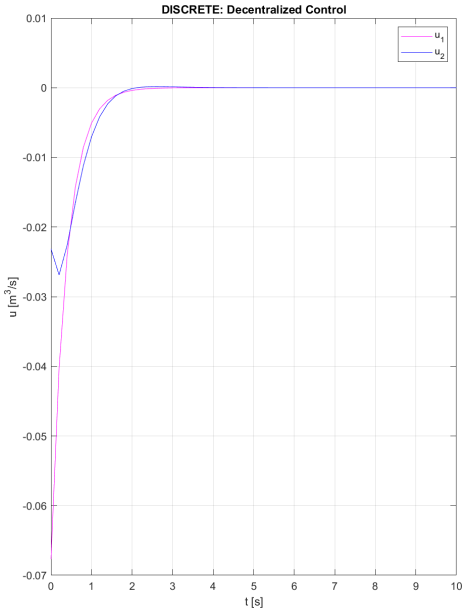
CENTRALIZED





DECENTRALIZED

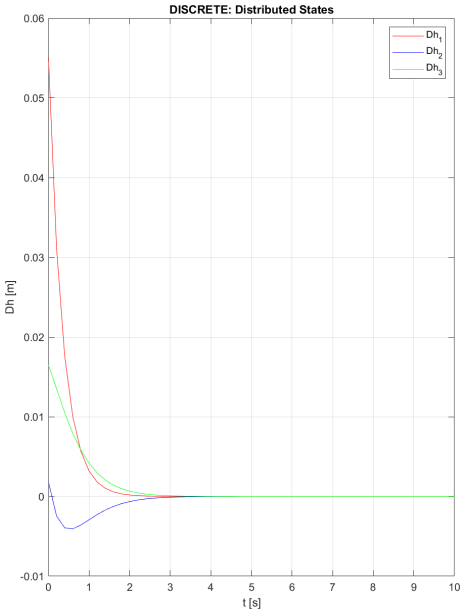


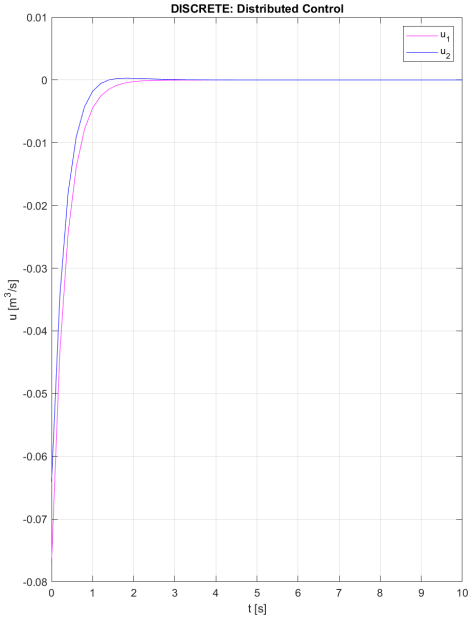


Simulation
Discrete time

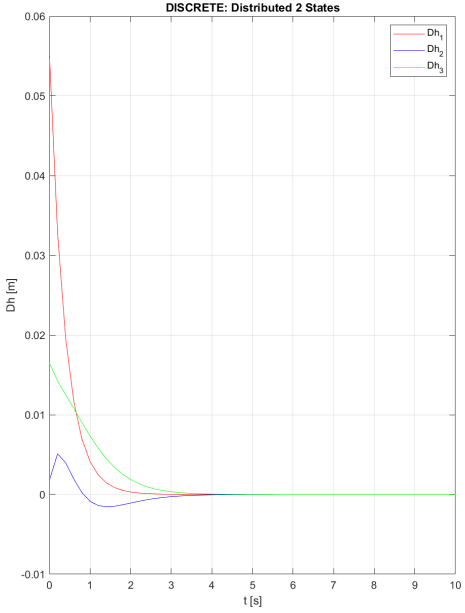
α_{max} =-0.6 , ρ_{max} = 0.1, h = 0.2 $x_0 = [0.055, 0.0018, 0.0165]^T$

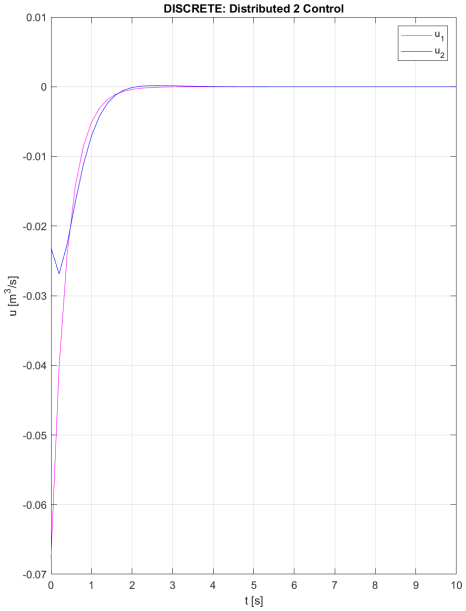
DISTRIBUTED 1





DISTRIBUTED 2

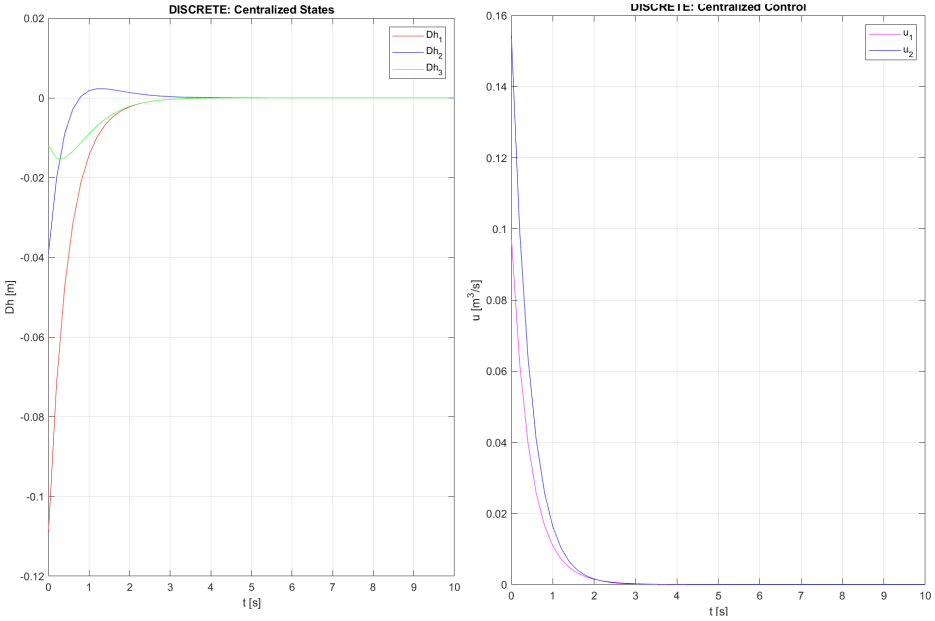




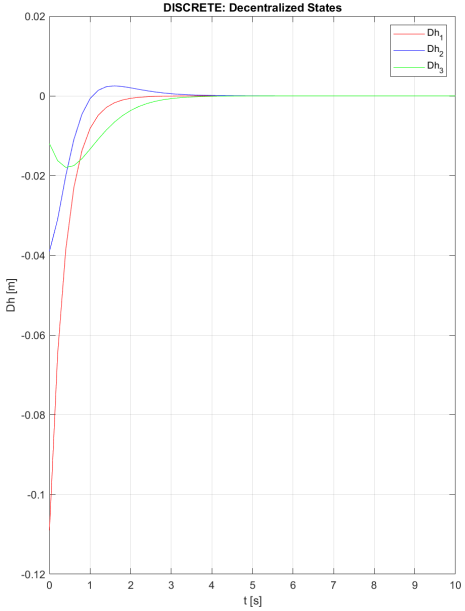
Simulation
Discrete time

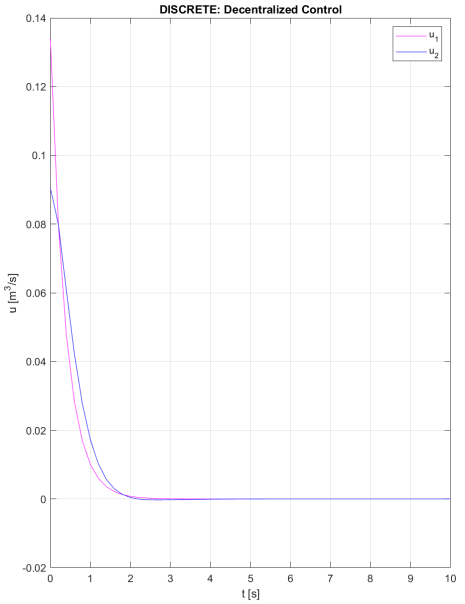
α_{max} =-0.6 , ρ_{max} = 0.1, h = 0.2 $x_0 = [-0.109, -0.039, -0.012]^T$

CENTRALIZED



DECENTRALIZED

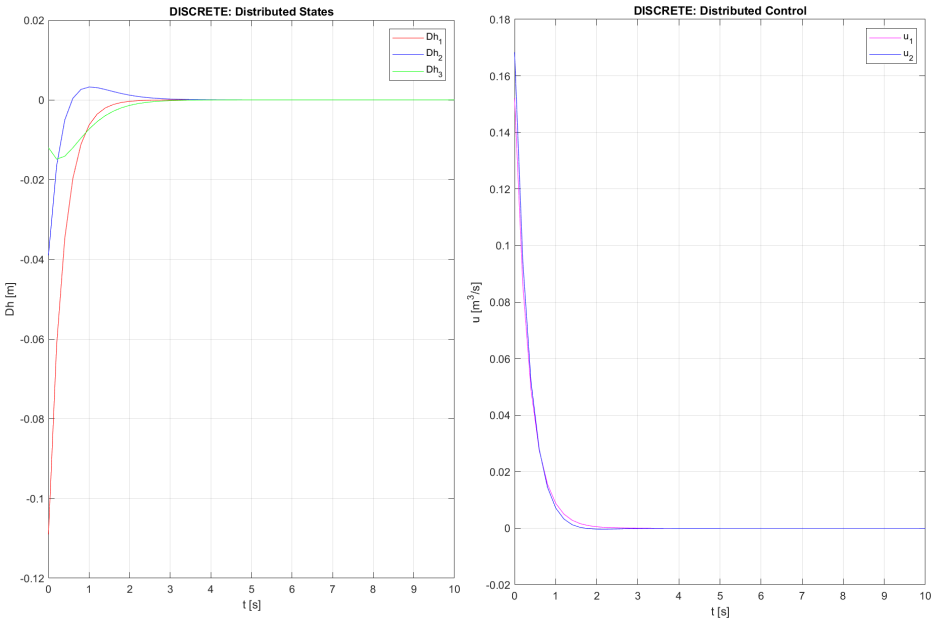




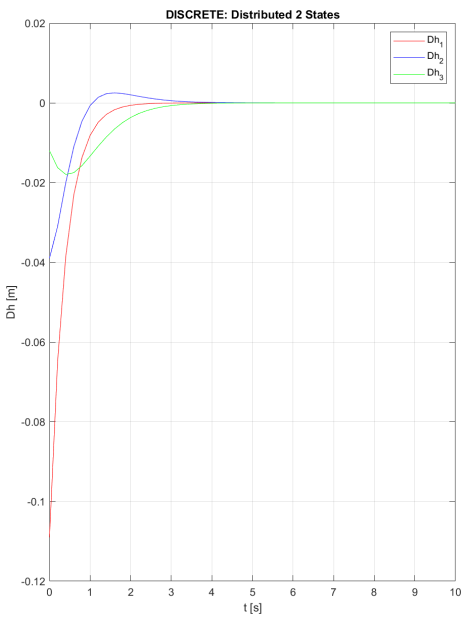
Simulation
Discrete time

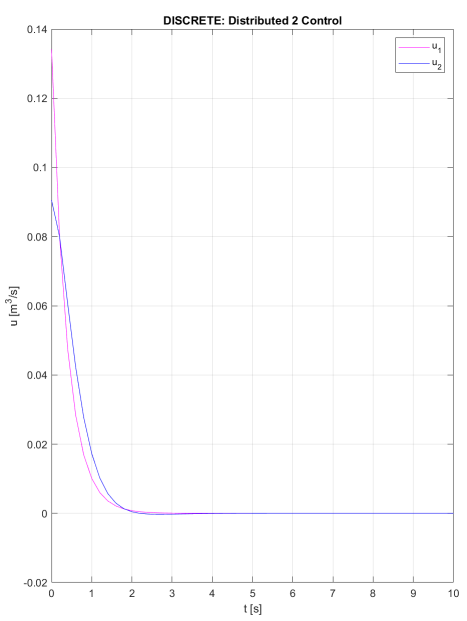
α_{max} =-0.6 , ρ_{max} = 0.1, h = 0.2 $\quad x_0 = [-0.109, -0.039, -0.012]^T$

DISTRIBUTED 1



DISTRIBUTED 2





Comments
Discrete time

From our simulations, we conclude that:

- In dicrete time we can see an advantage in terms of control effort with respect to the continuous time control. In fact, with same initial conditions we can achieve same performance but with a control action less aggressive.
- If we push performance further, control action becomes too reactive and performance degenerates. But differently from cont. time, we don't see evident state performance/control effort difference between the control schemes.

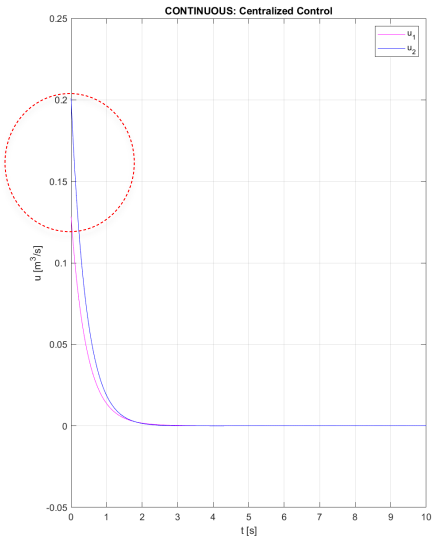
  So, even the less demanding scheme in terms of implementation (Decentralized) is a good choice for this system when controlling in discrete time.
- If we can accept big, but fast, responce overshoot, and we have access to a modest control action, we can even obtain behaviour similar to a FIR system (by designing a controller such that α_{max}= -0.2, ρ_{max}= 0.05 , the problem is still feasible).

Alternative Control Design
Limit control action rate

Notice that, we are dealing with a **real phisycal system**…

In our control framework we observe the system (linearized) and for a variation of the tanks' height respect the nominal conditions Dh, we act by a variation on the water input inside the tanks (to avoid any confusion, we call u that variation of u respect nominal value $\bar{u}$), in order to bring that variations Dh to 0.

Let's look to any control action (same reasoning in Discrete time):



Control variables vary from 0 to 0.2 (and 0.12) in 0 sec, and in 1 sec we are again back to 0. Even if an electric valve can be instantaneus, we can **assume to have some control rate limitation** (actuator dynamic), taking into account that Δu is limited, and that control action at t=0 start from 0, not from the instantaneus value chosen by the control law.

This constraint allow us to simulate better a real system, but also (sometimes) to improve the perfomance through a minimization of Δu.

Which leads to a distributed action over time, converging without aggressive behaviour.

Alternative Control Design
Limit control action rate

OPTION 1, LMI: (discrete time)

$\Delta u_k = K(F + GK - I)x_k$

To minimze Δu_k , we should minimize $\|K(F + GK - I)\|$

But if we also implement the minimization of the control effort, $\|K\|$ will be already minimized. We need to minimize 2-norm of $(F + GK - I)$

Proceeding similarly to the procedure for the minimization of control effort:

$\|F + GK - I\| = \|(FP + GL - P)P^{-1}\| \leq \|FP + GL - P\|\|P^{-1}\|$

Where $\|P^{-1}\|$ again is minimized by the control effort minimization.

So we focus on $\|FP + GL - P\|$:

$\min\{\alpha_D\}$

s.t. $\|FP + GL - P\| \leq \alpha_D$ (This can be casted by the shur complement into an LMI)

So overall, use this additional LMI and rewrite the overall cost as:

J = $a_D \alpha_D + a_Y \alpha_Y + a_L \alpha_L$ (with the proper trade-off choise of the weigth)

Alternative Control Design
Limit control action rate

Similarly, **continuous time**:

$u(t) = Kx(t)$ (state-feedback)

$\dot{u}(t) = K\dot{x}(t) = K\big(Ax(t) + Bu(t)\big) = K(A + BK)x(t)$

Same reasoning as before, we want to minimize $\|A + BK\|$…

Anyway, this option doesn't capture our modelling goal, not obtaining the proper cost trade-off to minimize well the control rate toghether with the control effort.

That's why we opted for

OPTION 2, State expansion:

Consider the control vector u as part of the state vector, expanding the state-matrix, and take as input of the system a virtual input v, representing the derivative of u.

(nothing forbid us to apply the virtual input v computed by our control scheme, and integrate it numerically to compute the real actuation input!)

Alternative Control Design
state expansion

From the initial system:

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases}$$

We expand the dynamic:

$$\begin{cases} \dot{x} = Ax + Bu \\ \dot{u} = v \\ y = Cx \end{cases}$$

$$A = \begin{bmatrix} -1 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$C = I_3$$

And, redefine the system vectors as:

$$x = [x_1 \quad u_1 \quad x_2 \quad x_3 \quad u_2]^T$$

$$u = [v_1 \quad v_2]^T$$

(Exactly the same reasoning is done for discrete time state-space description)

So the final state-space model become:

$$\begin{cases} \dot{x} = A_e x + B_e u \\ y = C_e x \end{cases}$$

$$A_e = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad B_e = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \qquad C_e = I_5$$

Alternative Control Design
state expansion

At this point, the system decomposition comes straigth foreward

$$A_e = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad B_e = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \quad C_e = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

B_1 B_2 C_1 C_2

Finally, we use exactly the same script as before, but with this new system matrix both in continuous and dicrete time. Through the control action minimization, this time we are minimizing u= $[\, v_1 \;\; v_2]^T$ control effort, exactly $\dot{u}$ as desired (or Δu in discrete).

Remember also to impose as initial condition:

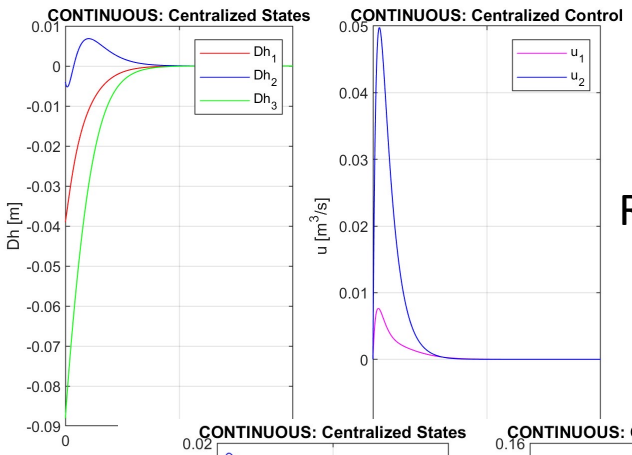
$x_0 = [h_{1,0}, 0, h_{2,0}, h_{3,0}, 0]^T$ (to start from 0 control action).

The tuning procedure to chose the best control design parameter is exactly as before, we can even try using the same optimal trade-off found previously.

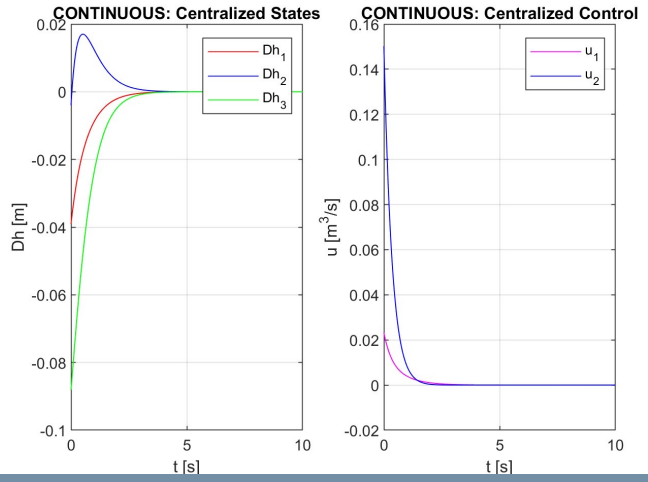
Simulation - Continuous time
Limit control action rate

α_{max} =1.7 ,ϑ_{max}=10 x_0=[−0.039,−0.04,−0.088]^T

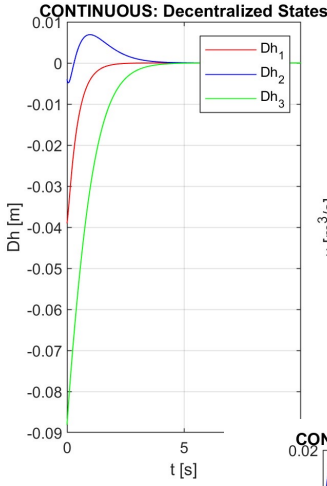
CENTRALIZED

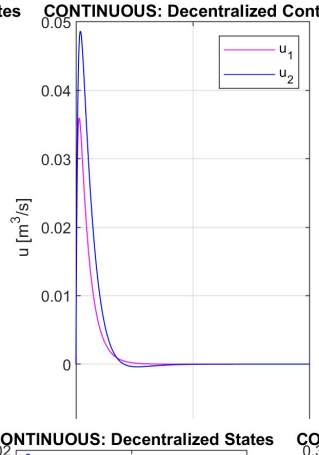


Rate Limit

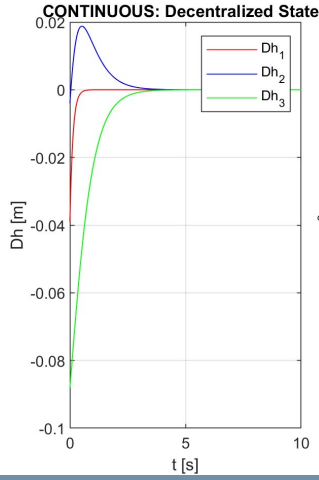


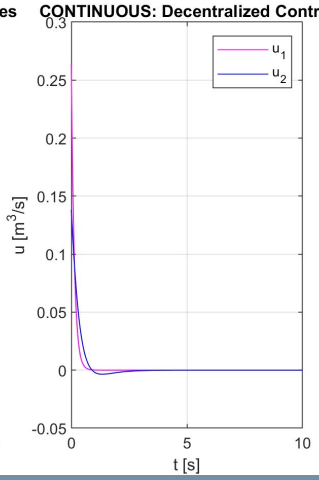
DECENTRALIZED





NO Rate Limit

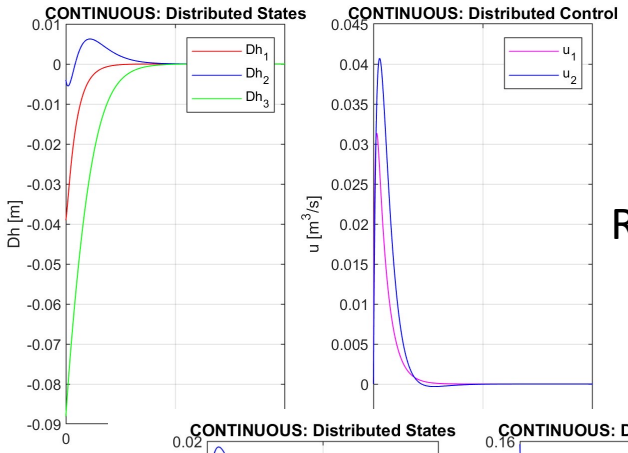




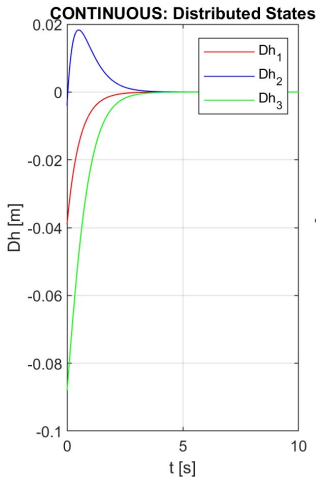
Simulation - Continuous time
Limit control action rate

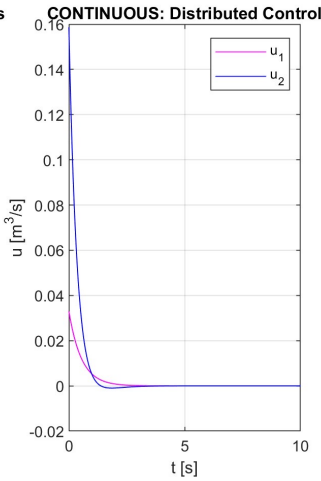
α_{max} =1.7, ϑ_{max}=10 x_0=[−0.039,−0.04,−0.088]^T

DISTRIBUTED

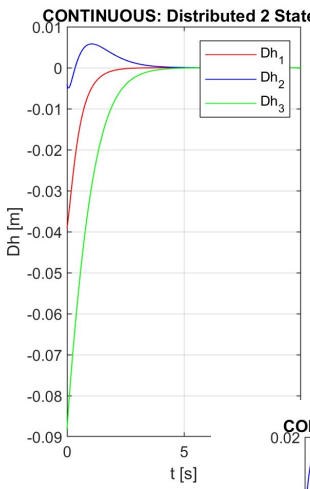


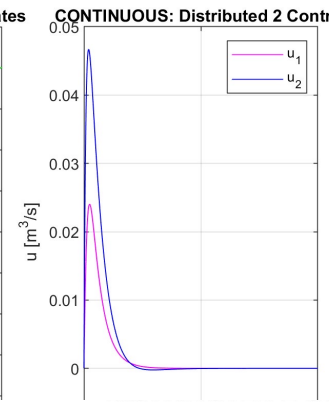
Rate Limit



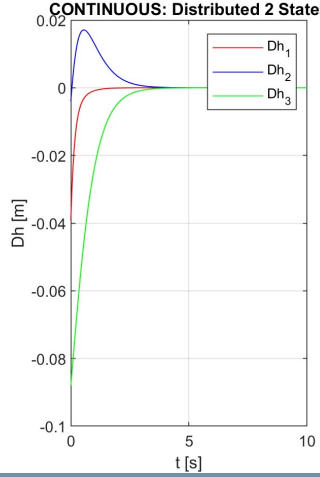


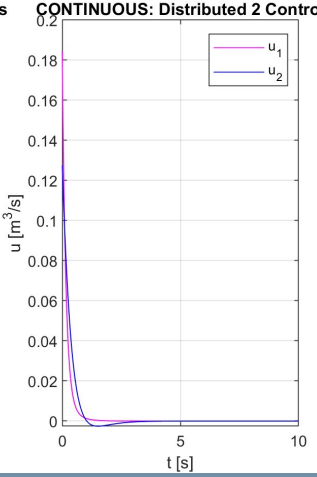
DISTRIBUTRED 2





NO Rate Limit

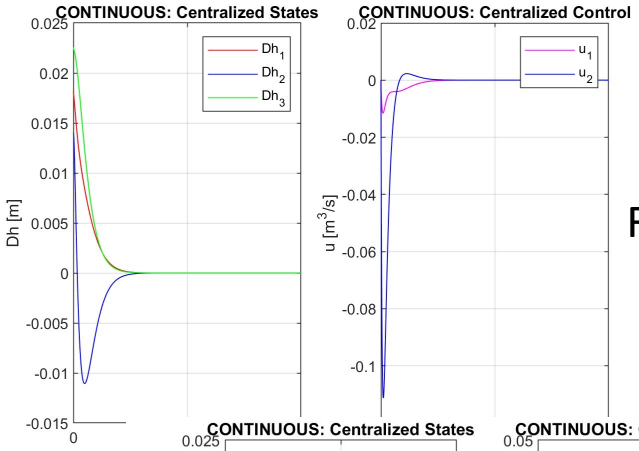




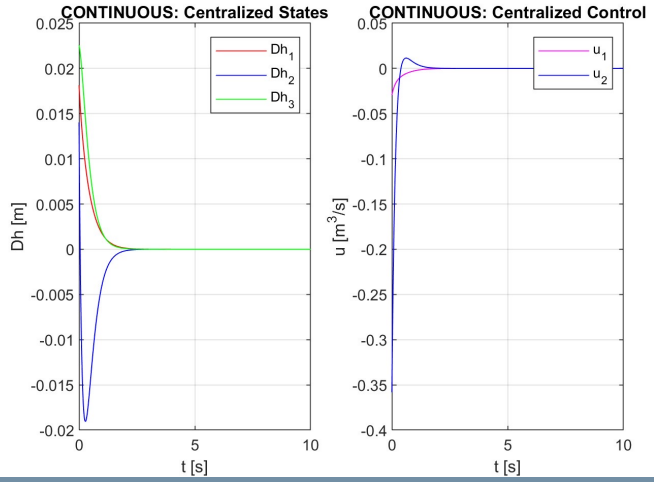
Simulation - Continuous time
Limit control action rate

α_{max} = 3 ,ϑ_{max}=10 x_0=[0.018,0.014,0.023]^T

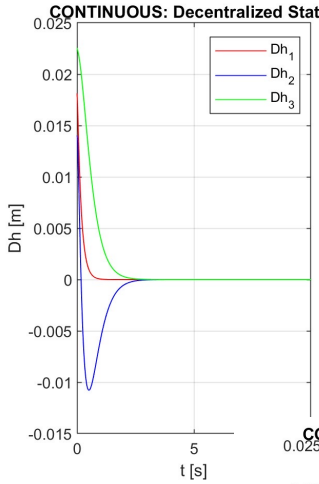
CENTRALIZED

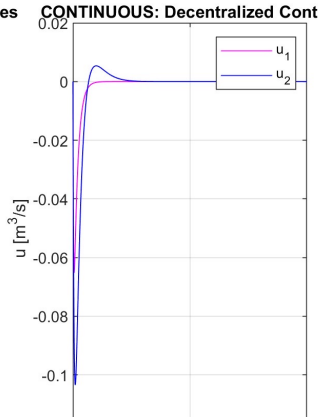


Rate Limit

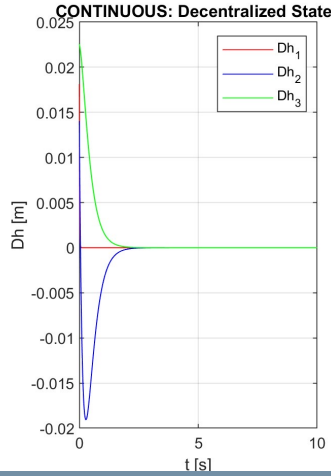


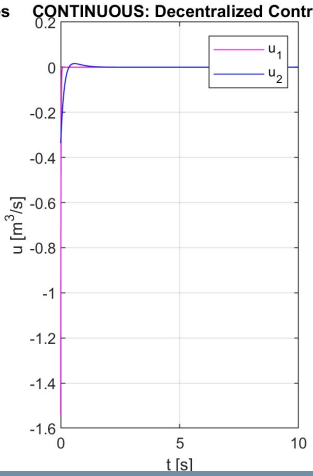
DECENTRALIZED





NO Rate Limit

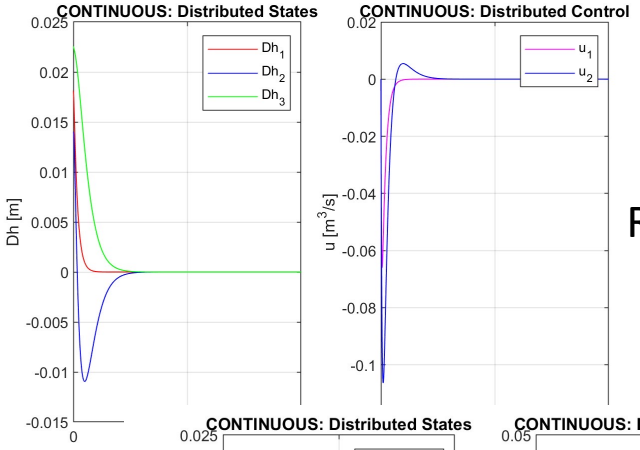




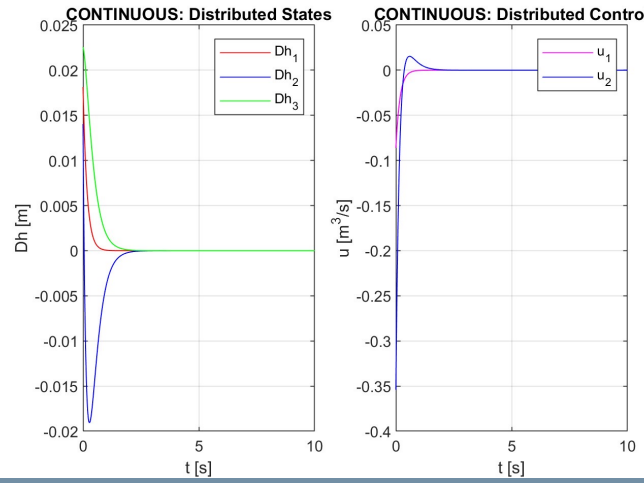
Simulation - Continuous time
Limit control action rate

α_{max} = 3, ϑ_{max}=10 x_0=[0.018,0.014,0.023]^T

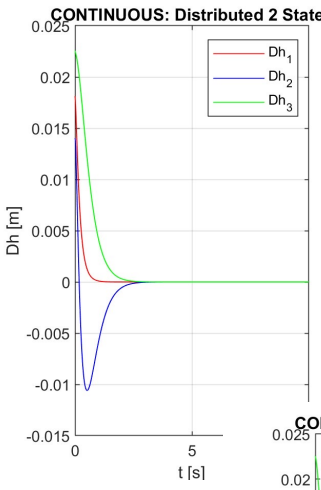
DISTRIBUTED

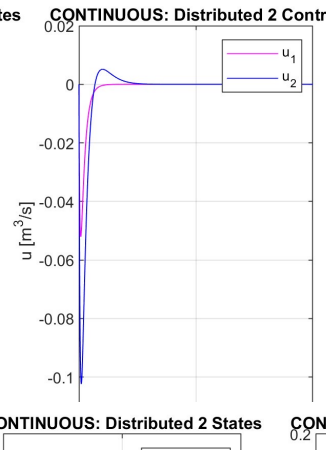


Rate Limit

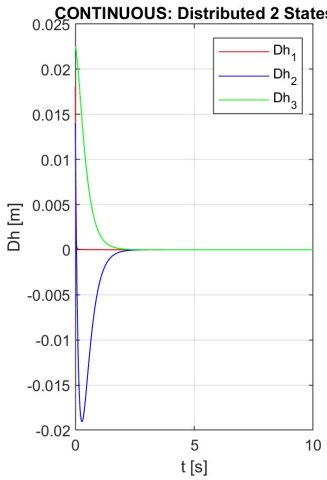


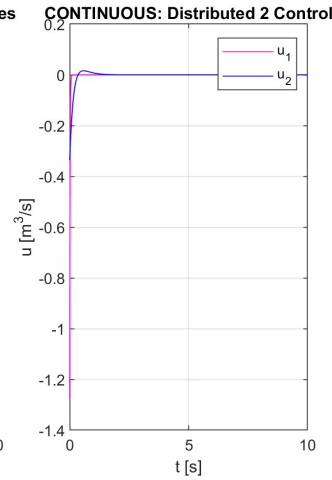
DISTRIBUTRED 2





NO Rate Limit





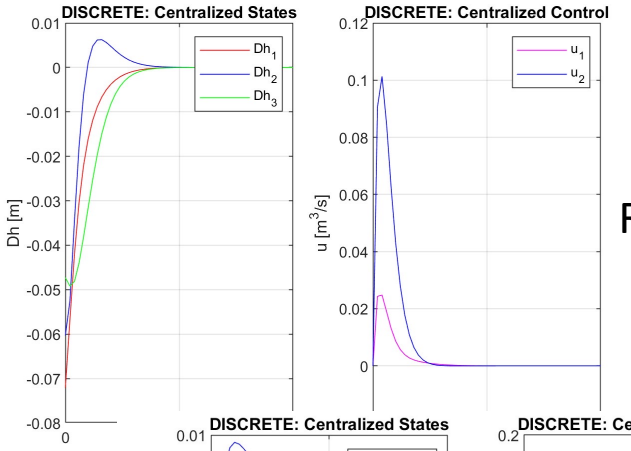
Comments - Continuous time
Limit control action rate

- Simulating by pushing even further the performance (for those cases when we need a really short settling time but with a feasible action), for $\alpha_{max} > 3$, the previous evidence of better performance of Centralized and Distributed 1 control structure here disappears. With this design, in the previous 'worst' schemes (Dec, Dist 2) in many cases we are able to reduce the control action peak by a factor of 30. So even the decentralized scheme has a good performance/control trade-off.
- Even asking for the same performance of before, we are able to reduce a lot the control effort, so this solution is perfect for those cases where we have saturation limits on actuators
- Notice that in most of the cases we are even able to reduce the maximum amount of overshoot

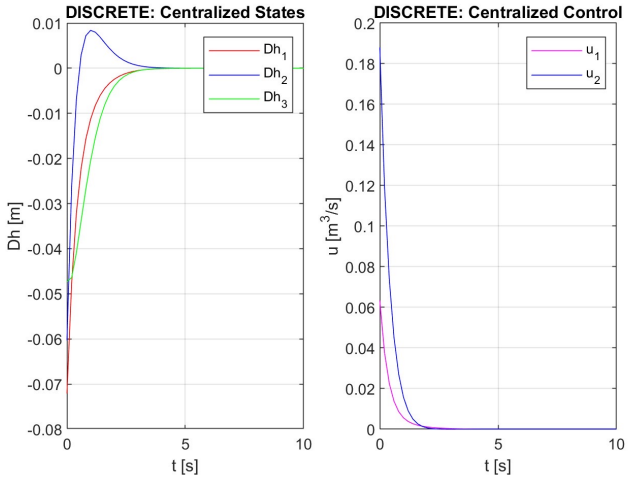
Simulation - Discrete time
Limit control action rate

α_{max} =−0.55 , ρ_{max} = 0.15, h = 0.2, x_0=[−0.072,−0.060,−0.047]^T

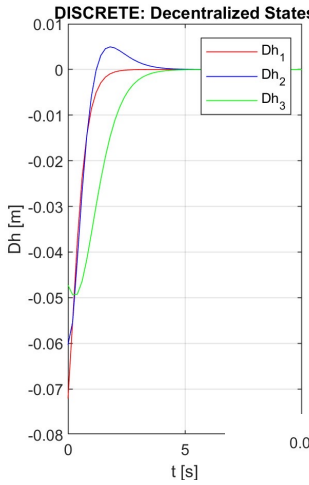
CENTRALIZED

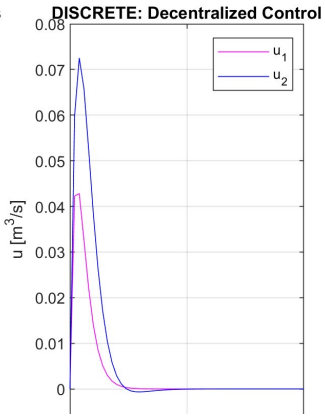


Rate Limit

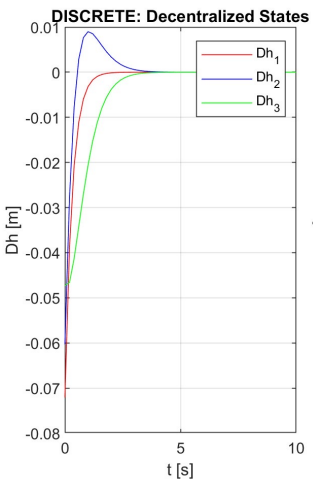


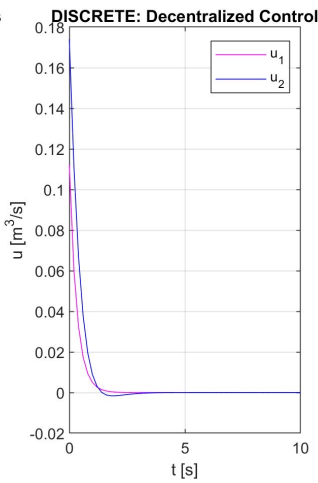
DECENTRALIZED





NO Rate Limit

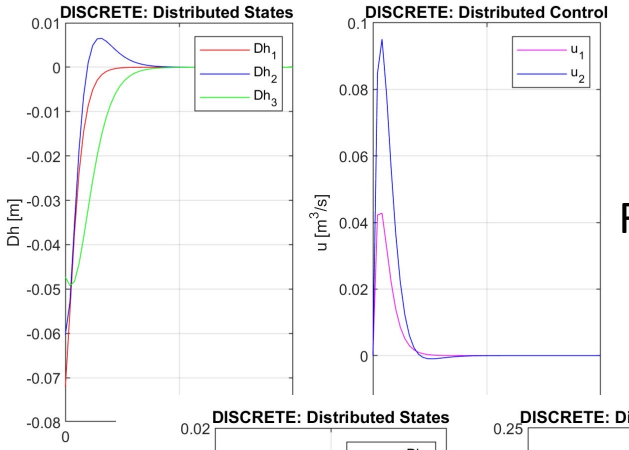




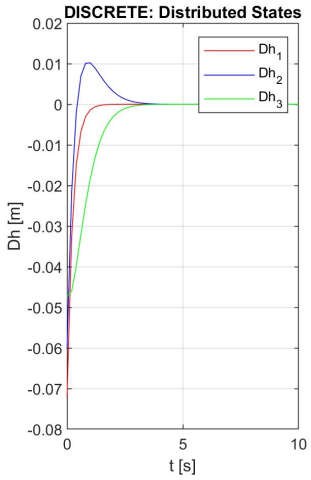
Simulation - Discrete time
Limit control action rate

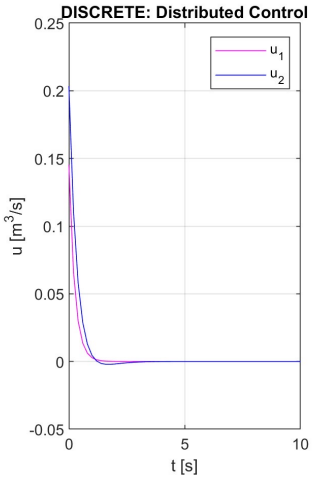
α_{max} =−0.55 , ρ_{max} = 0.15, h = 0.2, x_0=[−0.072,−0.060,−0.047]^T

DISTRIBUTED

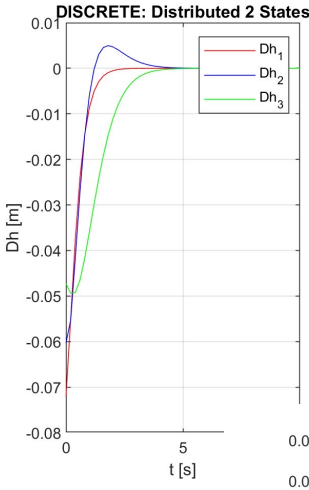


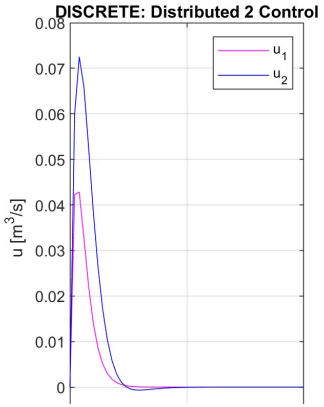
Rate Limit



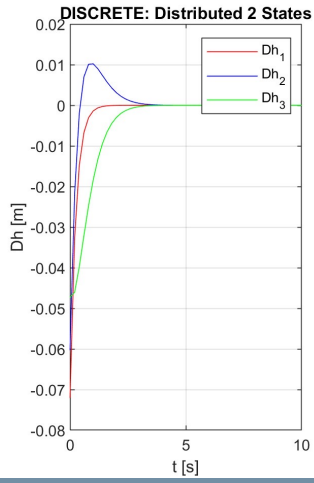


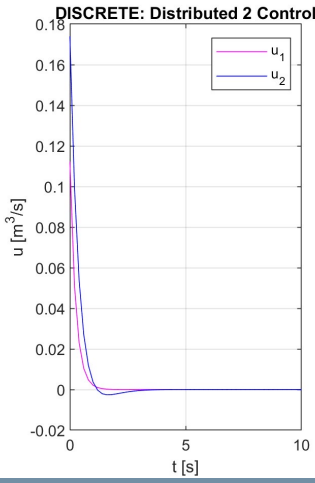
DISTRIBUTRED 2





NO Rate Limit

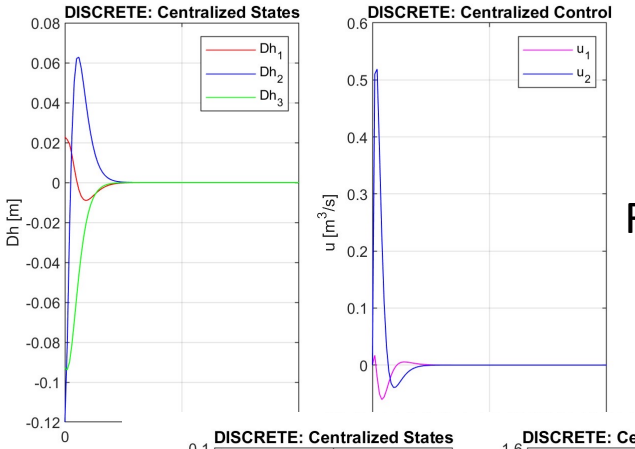




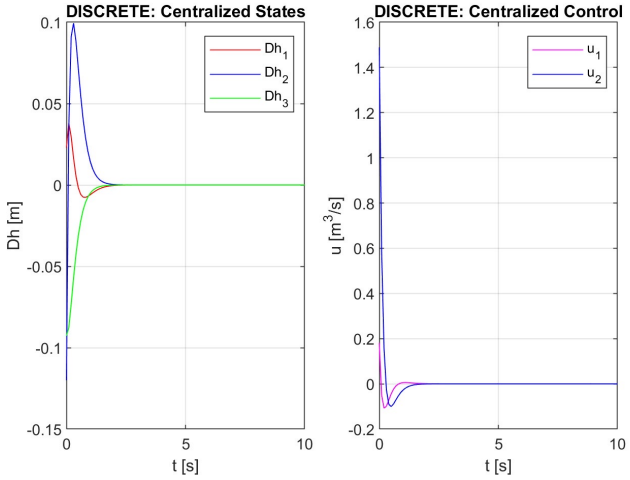
Simulation - Discrete time
Limit control action rate

α_{max} =−0.6 , ρ_{max} = 0.1, h = 0.1, x_0=[−0.023,0.12,−0.093]^T

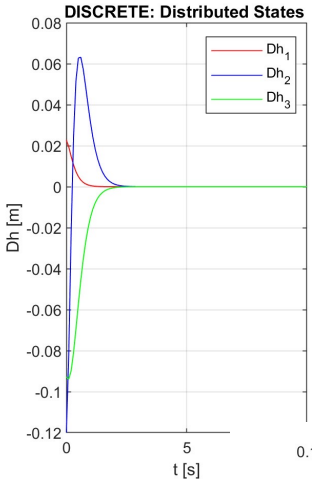
CENTRALIZED

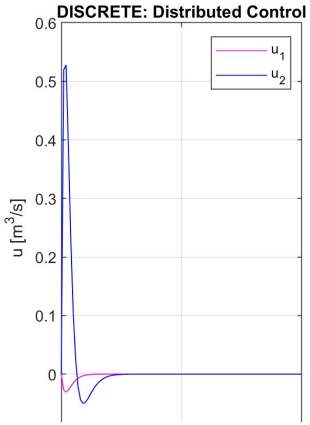


Rate Limit

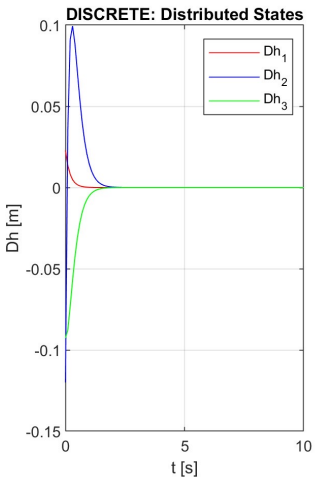


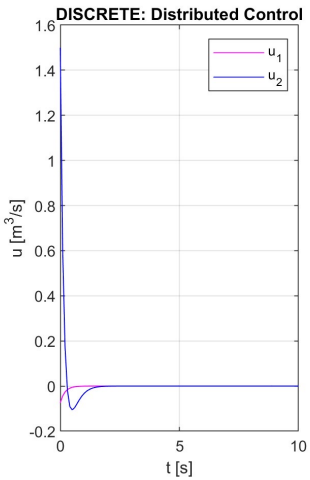
DISTRIBUTRED





NO Rate Limit





Comments - Discrete time
Limit control action rate

- With other simulations, improving the performance in terms of α_{max} we observe that we are able in most of the cases to reduce a lot the control variable peaks, making the requested control action feasible in case of actuator's limitations.
- There are some cases where this solution underperforms respect to the ones described before. Because of the control action rate limitations, it can become a bit slower
- This control strategy being less reactive can work even for h=0.1, in fact it guarantees the same performance of classical controllers, but with less effort. Anyway the control effort is still not negligible, this is not a good advantage.
- Again as in the previous discrete control design, there is not so much difference between the performance and effort of each control structure.
- Overall this strategy is very good if we have huge limitations on the actuators, in terms of max/min saturation and rate limitations, but it doesn't improve too much the performance that we could already obtain, except for some cases.

The results obtained in continuous time were more performing

(maybe with a better tuning something better can be obtained even in discrete)

Final Comments

- Overall, thanks to the absence of fixed modes for all structures, we have been able to reach our goals both in continuous and discrete time using simple LTI state-feedback controllers.
- The control strategies described focus on different aspects of our possible problem specification and on the available resources:
    - By **pole region placement** we have the freedom on the dominant mode, so we can prescribe the overall dynamic as we desire, but we don't have many power on the control effort or in the specific modes (this require to test the system to check the results).
    - By $\mathcal{H}_2$ the controller obtained is the optimal one respect to the figure of merit we define, so according to the best control/state trade-off for each mode or control variable. Using this strategy we can avoid to perform the task of deciding where to place the poles and at the same time the minimization of control effort is guaranteed.
    - The **Control Rate Limitation** Strategies is very good when we have big limitations in terms of control action

Final Comments

- In the discrete time domain the choice of h is very important, and discrete controller allow us to speed up a lot the system, with still acceptable control action if the controller is well designed.
- This shows us also how powerfull LMI design strategies can be. We can cast all our controller design as an LMI problem and solve it easily.

Thanks for the attention