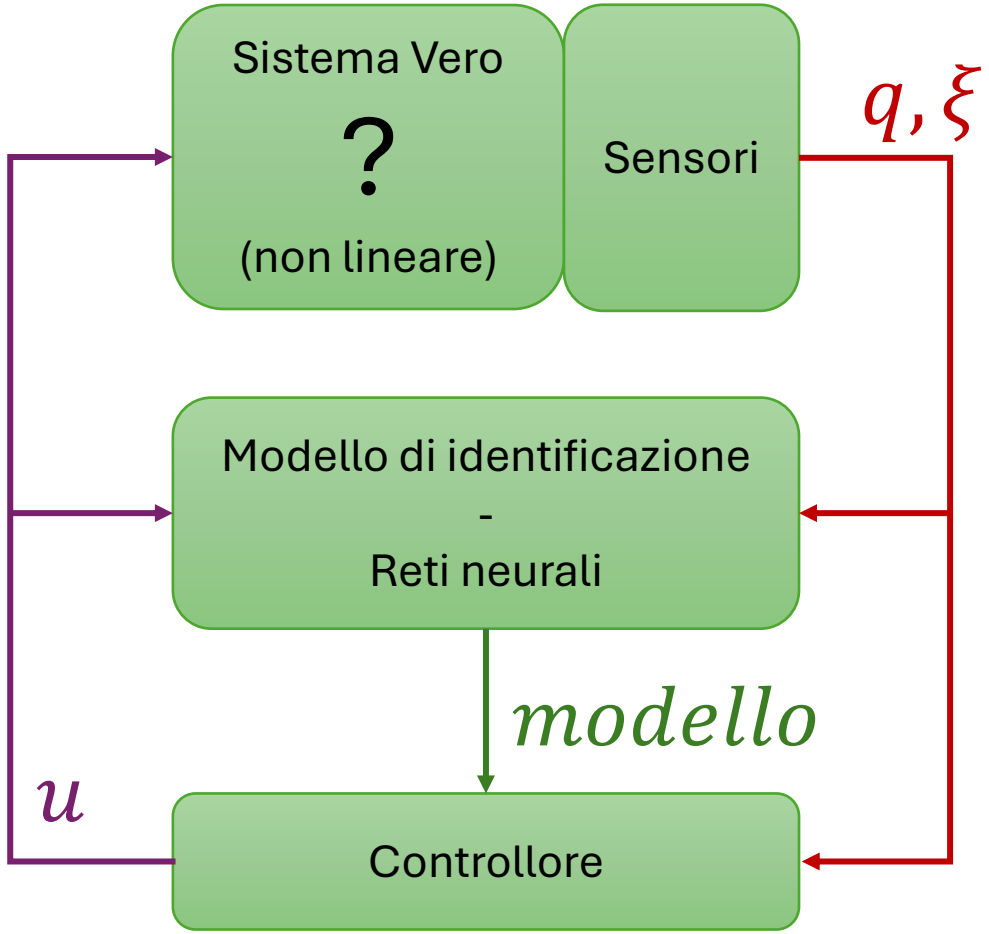
INTRODUZIONE

Idea

L'obiettivo è quello di sviluppare un controllore per un sistema di cui, oltre a non conoscere il valore numerico di eventuali parametri in gioco, **non ne conosciamo neanche la forma e il tipo di non linearità**.



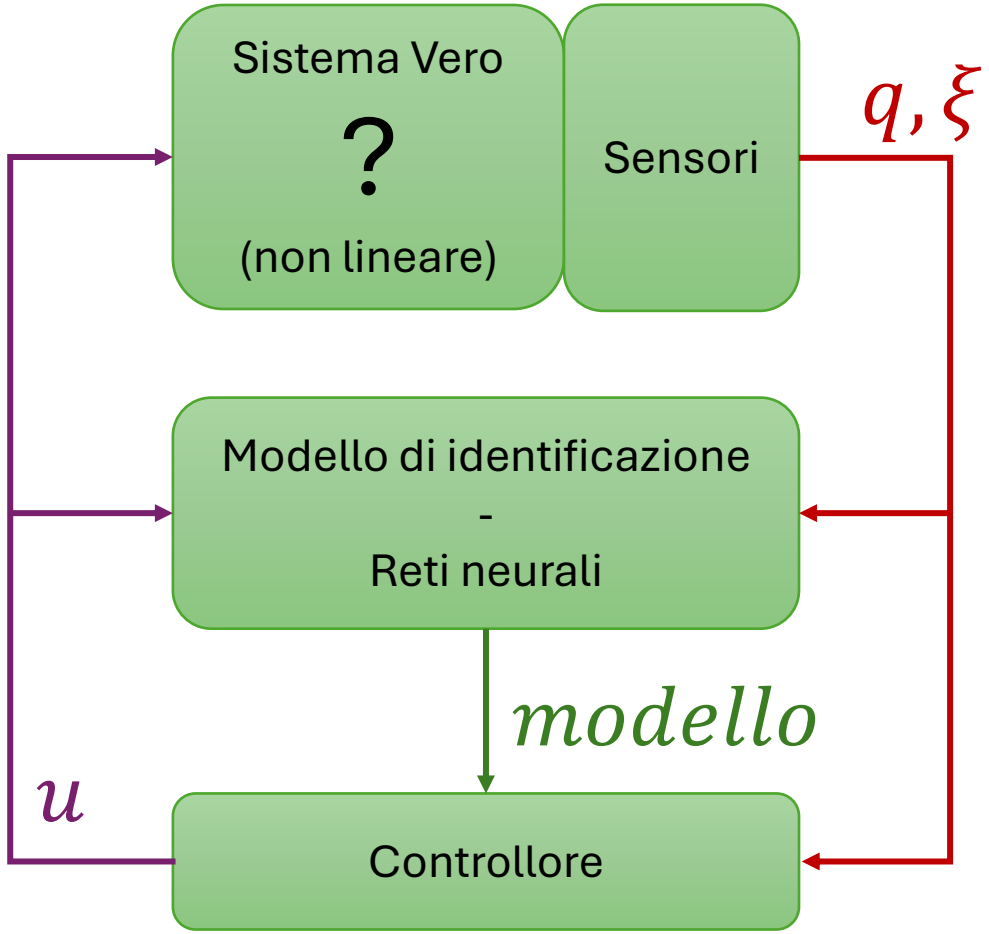
INTRODUZIONE

Articoli di riferimento

J.-J.E. Slotine and R. M. Sanner – *«Neural Networks for Adaptive Control and Recursive Identification: A Theoretical Framework»* (1993)

Marios M. Polycarpou and Petros A. Ioannou – *«Modelling, Identification and Stable Adaptive Control of Continuous-Time Nonlinear Dynamical Systems Using Neural Networks»* (1992)

Farzaneh Abdollahi, H. Ali Talebi and Rajnikant V. Patel – *«Stable Identification of Nonlinear Systems Using Neural Networks: Theory and Experiments»* (2006)

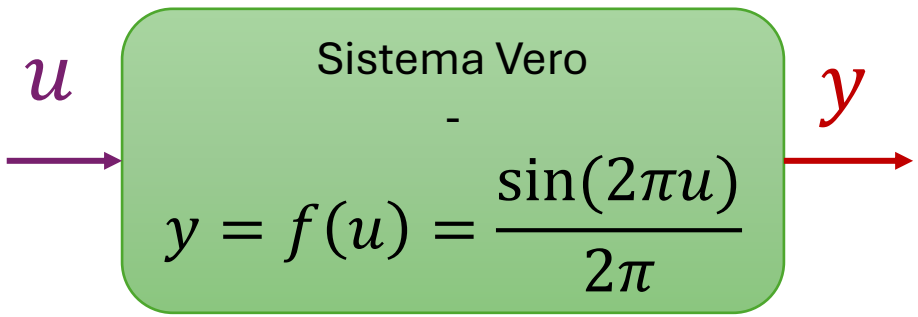


ESERCIZIO 1 – FUNZIONE SCALARE

Sistema vero

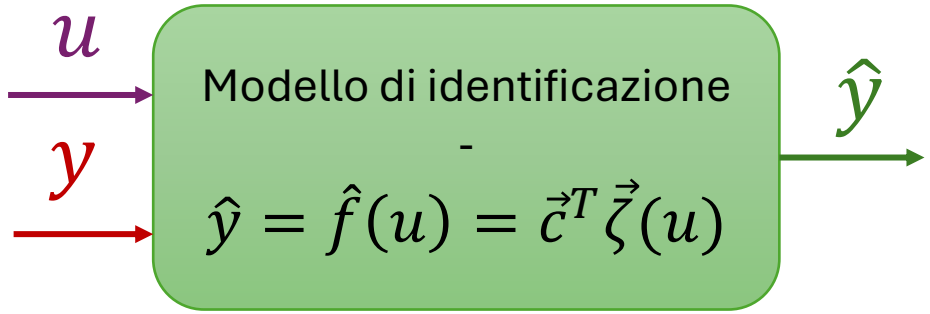
Al fine di introdursi alla problematica il primo esercizio proposto è quello di **identificare una singola funzione scalare non lineare**.

Il sistema vero (incognito) è il seguente:



Modello di identificazione

Per approssimare il modello vero è stata scelta una **Base di Funzioni Gaussiane Radiali (GRBF)**, ciascuna scalata con un peso.



$$\zeta_i(u) = e^{-\frac{(u-m_i)^2}{\sigma^2}}$$

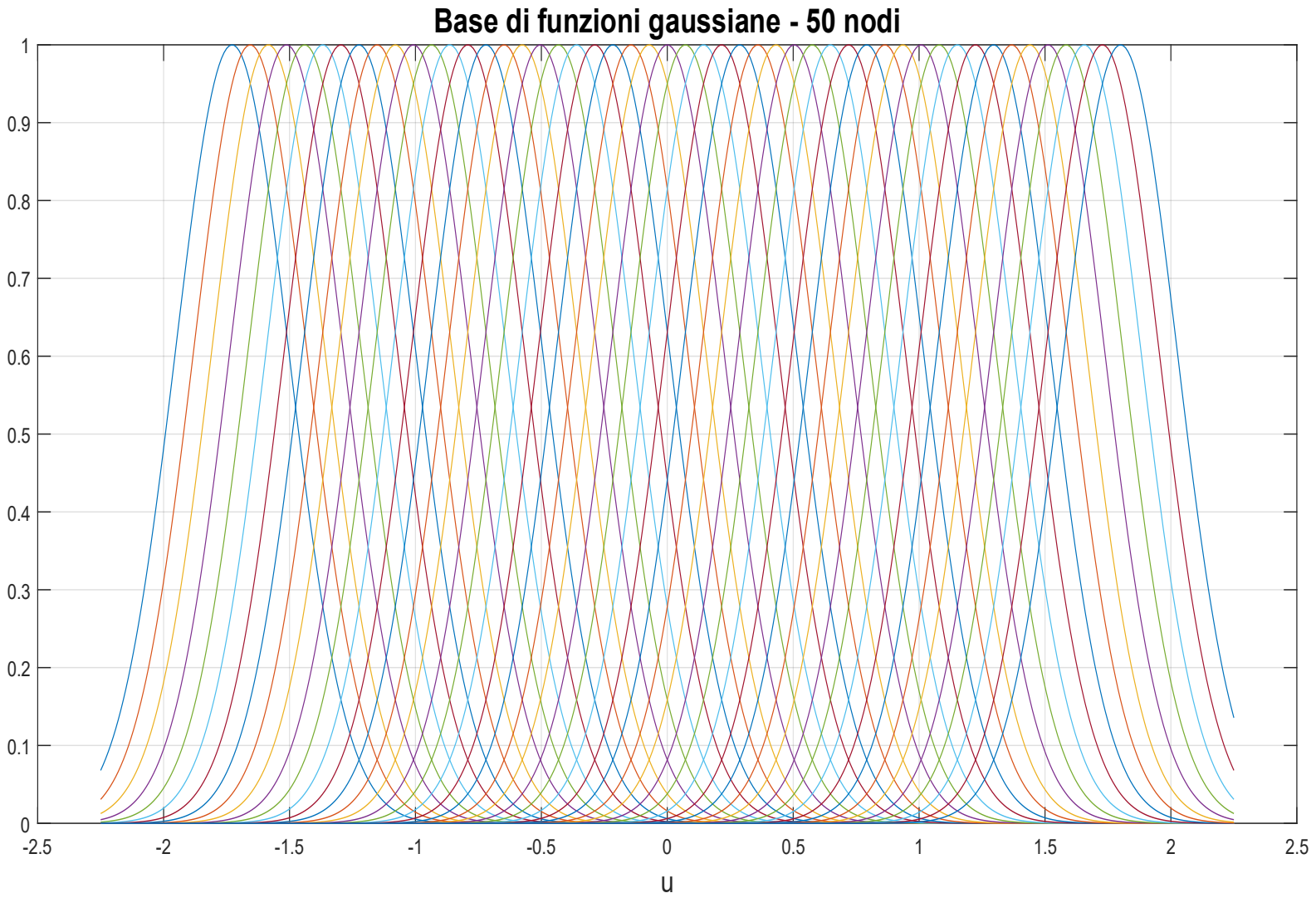
ESERCIZIO 1 – FUNZIONE SCALARE

Visualizzazione della base

In figura sono riportate le 50 funzioni di base che, combinate linearmente attraverso i pesi, approssimeranno il sistema vero.

Risulta evidente un aspetto fondamentale: Il modello di identificazione funzionerà soltanto per un **range di ingressi stabilito a priori**.

La conoscenza di questo range rappresenta quindi una **conoscenza a priori necessaria** per lo sviluppo del modello.



ESERCIZIO 1 – FUNZIONE SCALARE

Backpropagation – Aggiornamento dei Pesi

L'obiettivo della rete sarà minimizzare la funzione di costo

$$\begin{cases} e_y \triangleq \hat{y} - y \\ Cost \triangleq \frac{1}{2} e_y^2 \end{cases}$$

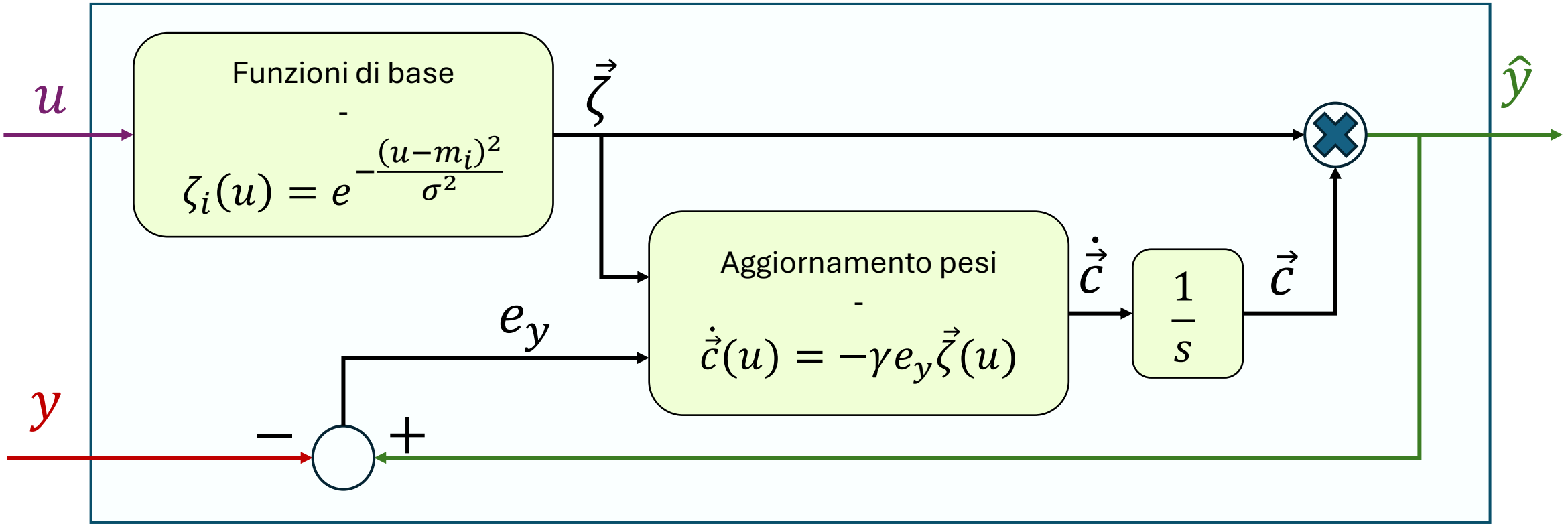
per fare questo stabiliamo una legge di aggiornamento dei pesi così fatta

$$\dot{\vec{c}}(u) = -\gamma \frac{\overrightarrow{\delta Cost}}{\delta \vec{c}} = -\gamma \frac{\delta Cost}{\delta e_y} \frac{\overrightarrow{\delta e_y}}{\delta \vec{c}} = -\gamma e_y \vec{\zeta}(u)$$

Dove γ è un parametro detto *«Learning rate»*

ESERCIZIO 1 – FUNZIONE SCALARE

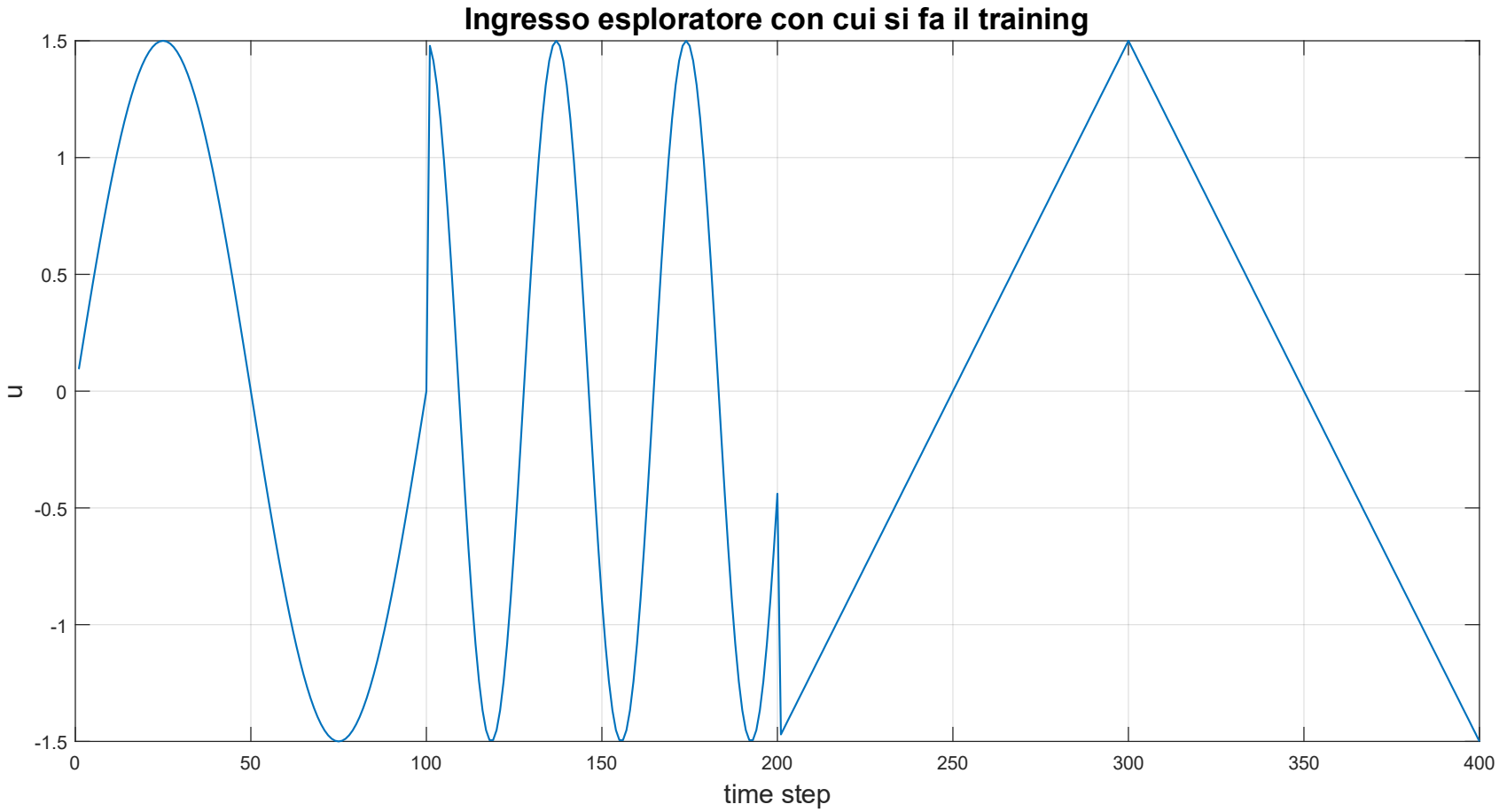
Schema del modello di identificazione



ESERCIZIO 1 – FUNZIONE SCALARE

Fase di training - ingresso

Nella fase di training della rete si sottopone al sistema una funzione di ingresso che esplori quanto più a fondo possibile il range in cui ci interessa effettuare l'identificazione.

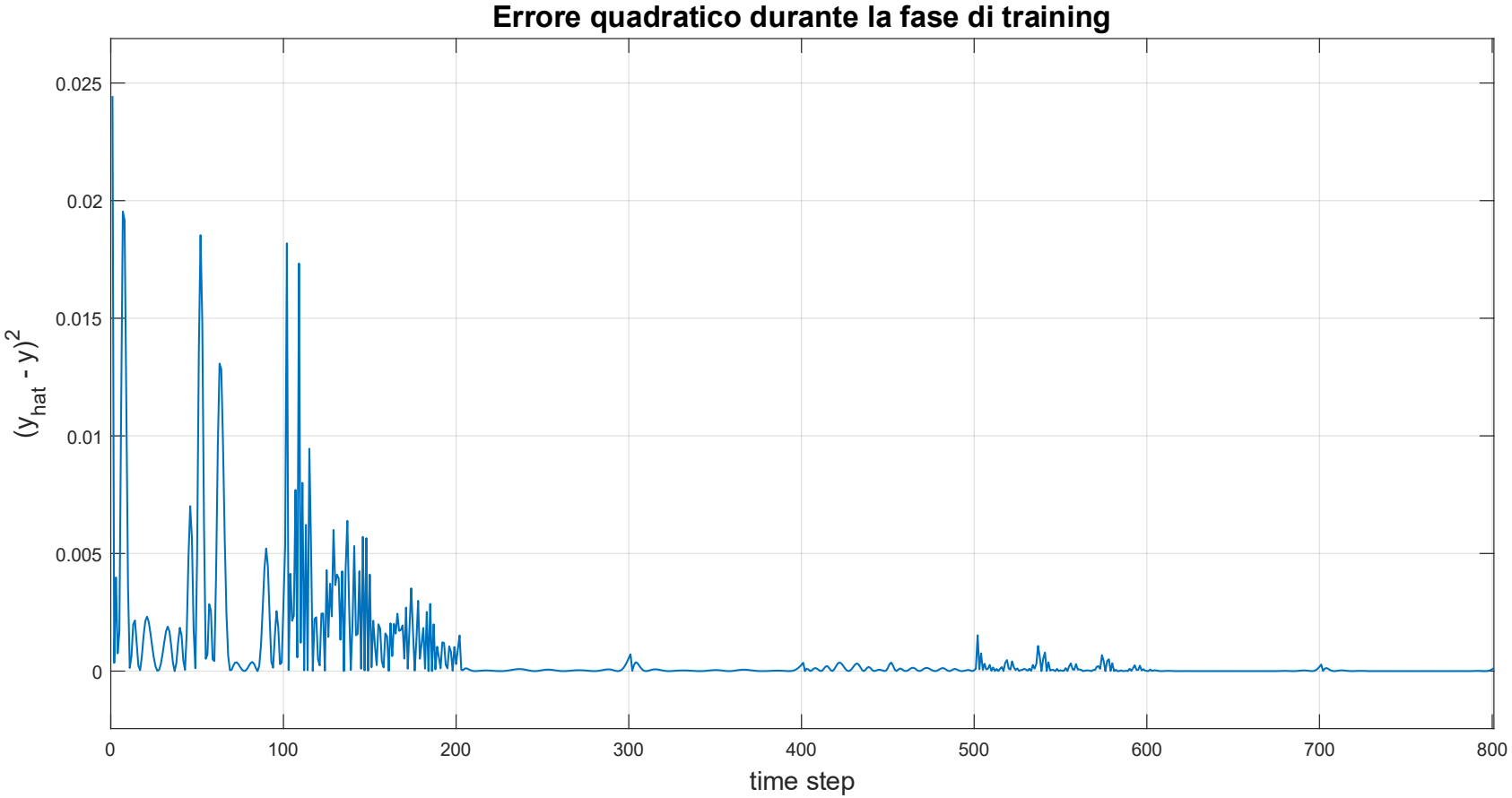


ESERCIZIO 1 – FUNZIONE SCALARE

Fase di training - errore

Questo è l'andamento dell'errore quadratico (ovvero la funzione obiettivo della rete) durante la fase di training.

Dopo soli due cicli della funzione di ingresso vista precedentemente la rete di identificazione sembra essere arrivata a convergenza.

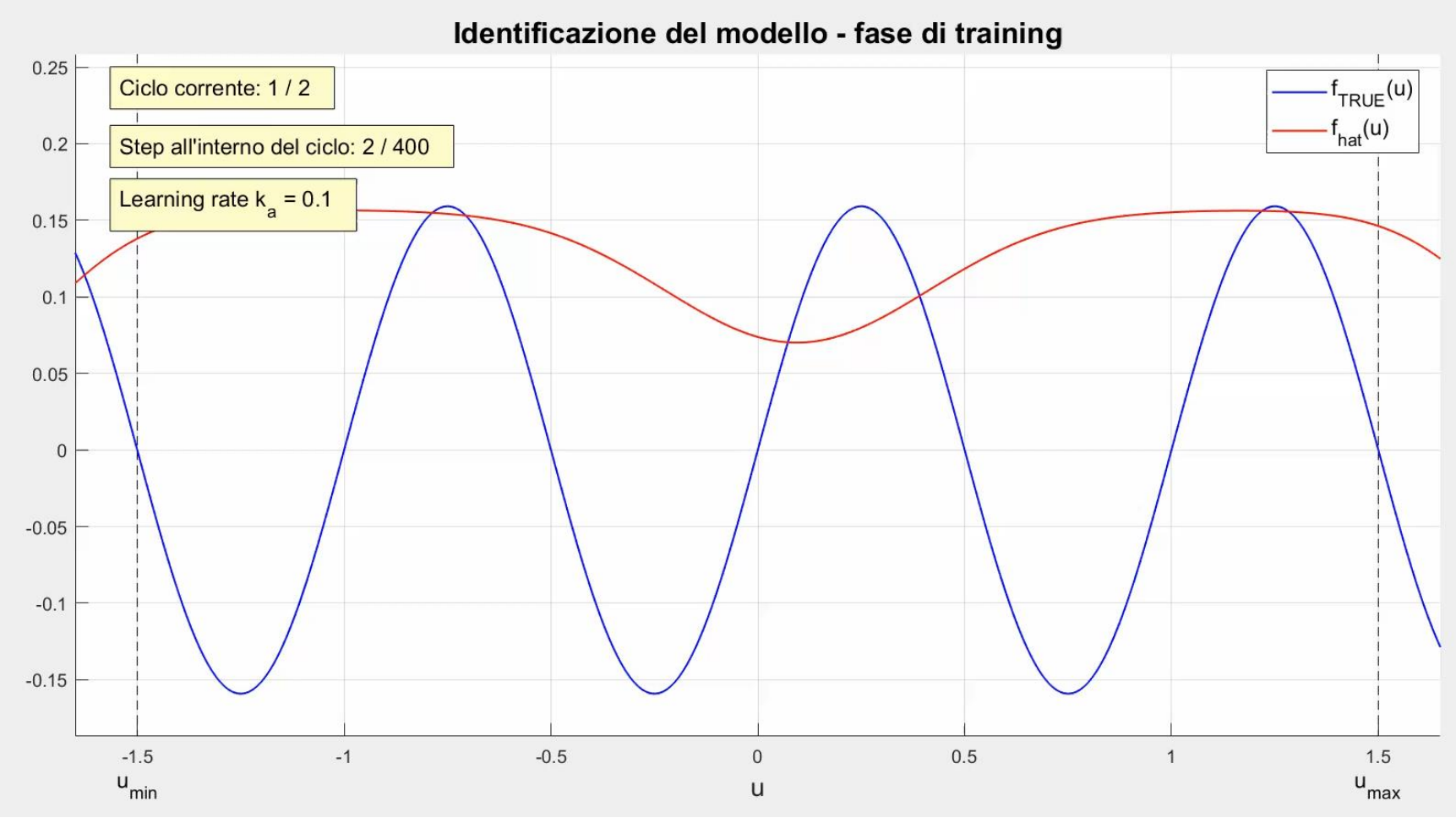


ESERCIZIO 1 – FUNZIONE SCALARE

Fase di training - animazione

Per visualizzare meglio cosa succede in fase di training, ho creato un'animazione per vedere dopo ogni passo temporale come appare la funzione approssimata.

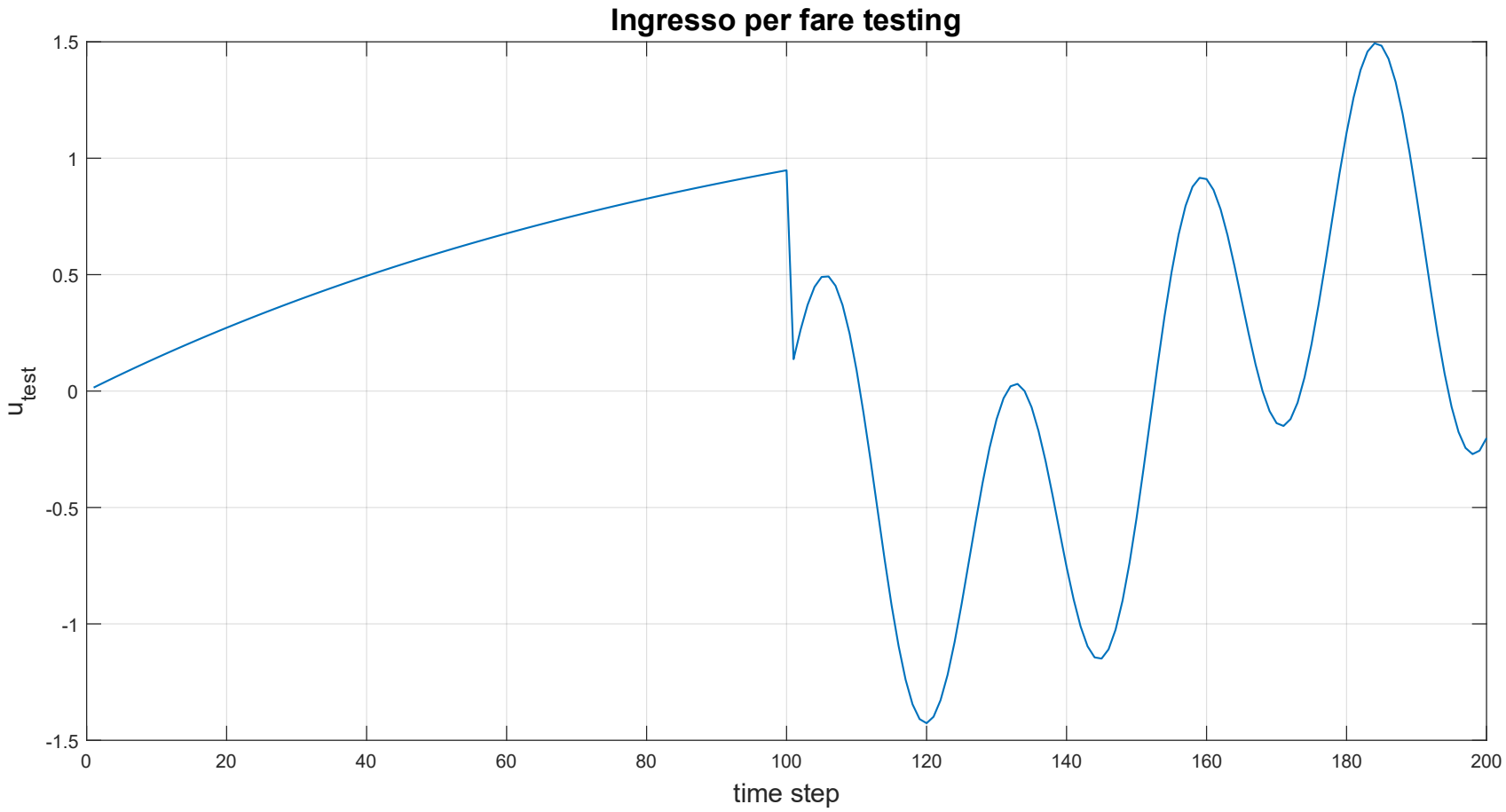
Notare che questo tipo di visualizzazione è possibile solo grazie alla semplicità del problema, ovvero che la funzione è una ed è scalare.



ESERCIZIO 1 – FUNZIONE SCALARE

Fase di testing - ingresso

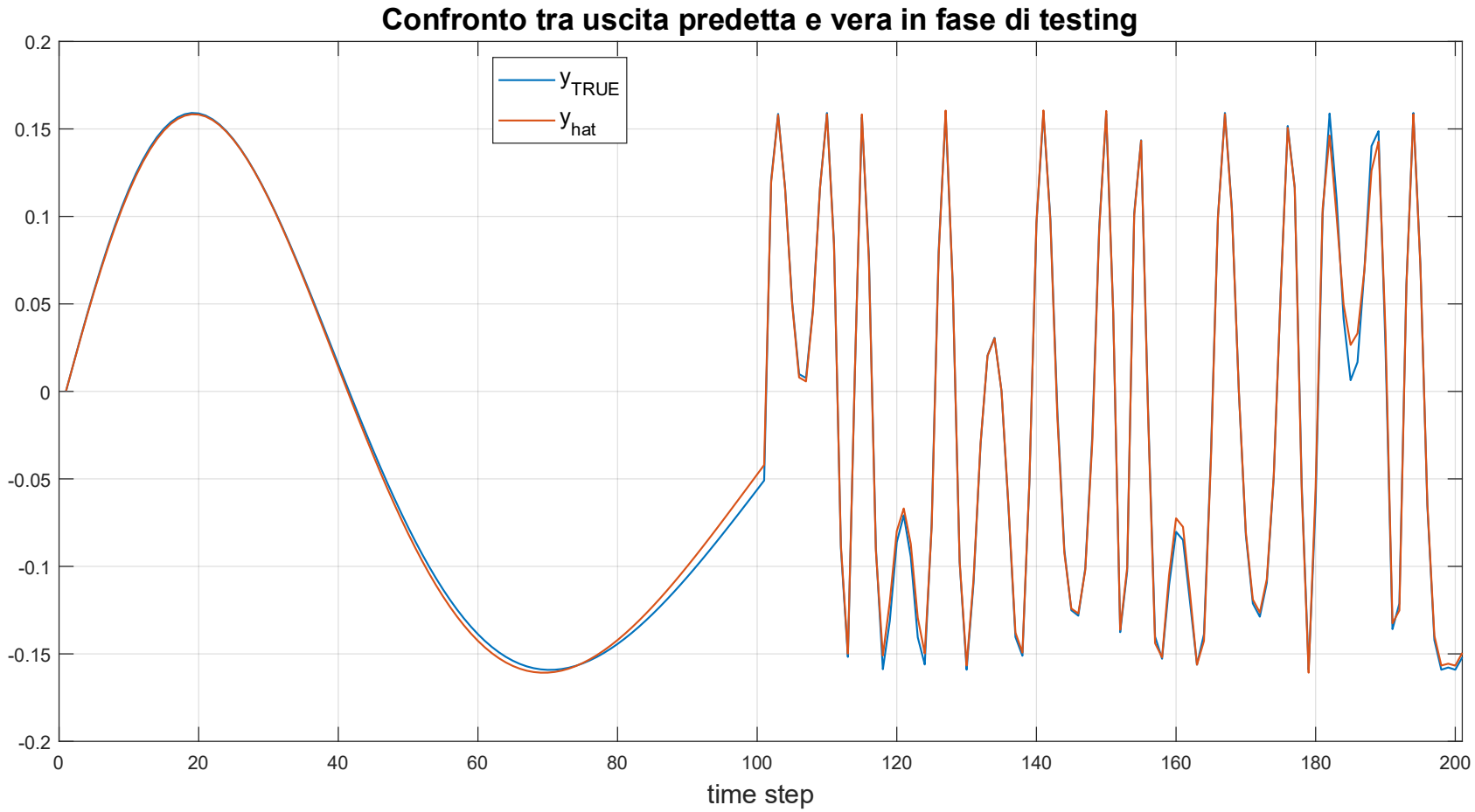
Nella fase di testing i pesi sono fissi ai valori ottenuti alla fine della fase di training, e si sottopone il modello di identificazione e il sistema vero a una sequenza di ingresso che sia diversa da quella della fase di training così da validare l'addestramento del modello.



ESERCIZIO 1 – FUNZIONE SCALARE

Fase di testing - errore

Come è evidente dalla figura il modello si comporta bene anche in fase di testing.



ESERCIZIO 1 – FUNZIONE SCALARE

Fine Esercizio 1

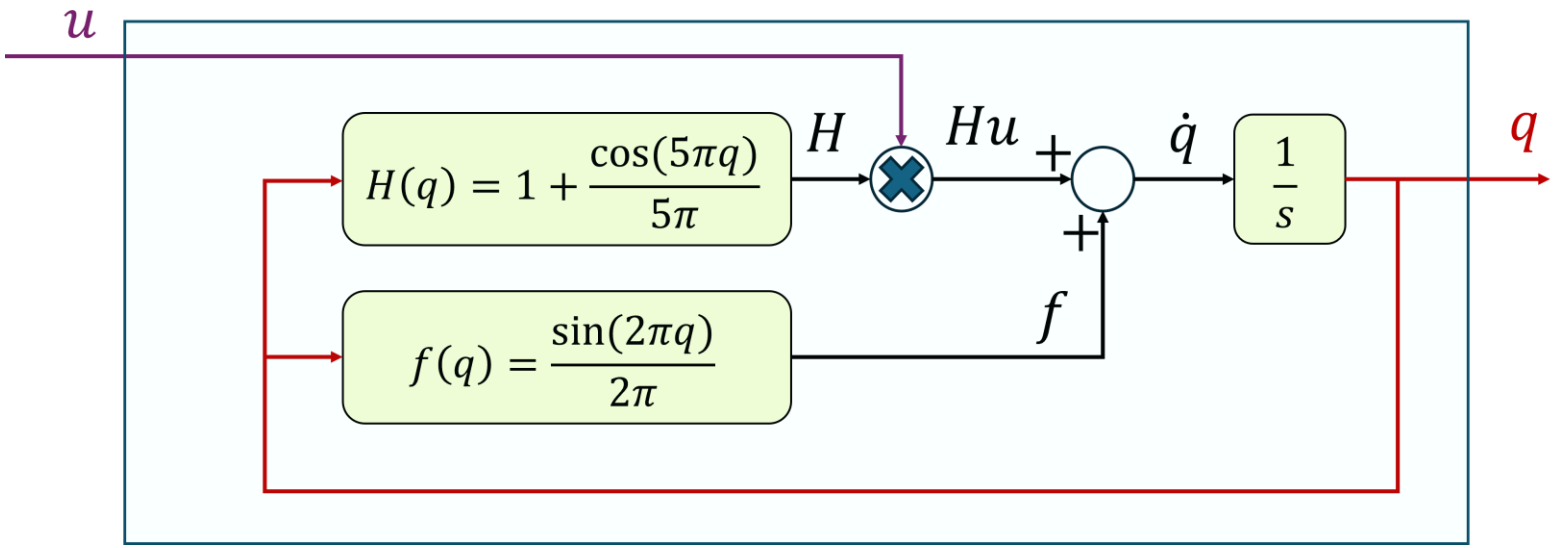
ESERCIZIO 2 – SISTEMA DINAMICO SCALARE

Sistema vero

In questo esercizio il sistema dinamico da controllare è nella forma di stato affine nel controllo

$$\dot{q} = f(q) + H(q)u$$

L'obiettivo è fare un **controllo di movimento** del sistema, senza sapere nulla riguardo le due funzioni $f(q)$ e $H(q)$, se non il range in cui vive q.



ESERCIZIO 2 – SISTEMA DINAMICO SCALARE

Generazione del riferimento

La funzione da inseguire $q_m(t)$ è stata generata a sua volta da un modello dinamico, così da conoscere precisamente la sua derivata temporale $\dot{q_m}(t)$

$$\begin{cases} \dot{q_m} = -A_m q_m + b_m r \\ r = \dfrac{1}{b_m}(A_m q_m + k_r(q_r(t) - q_m)) \end{cases}$$

In questo modo devo soltanto scegliere $q_r(t)$, e lo farò in modo che sia chiaro l'intervallo in cui questa varia

$$q_r(t) = \frac{q_{limit}}{2}[\sin(a_r \omega_r t) + \sin((1 + a_r \omega_r t))]$$

ESERCIZIO 2 – SISTEMA DINAMICO SCALARE

Controllore ideale

Definisco l'errore

$$e_m \triangleq q - q_m$$

Definisco una Lyapunov

$$V = \frac{1}{2} e_m^2$$

$$\dot{V} = e_m(f + Hu + A_m q_m - b_m r)$$

Costruisco l'ingresso u in modo che annulli i termini «scomodi» e renda $\dot{V}$ negativa definita

$$u_{ide} = \frac{1}{H}(-f - A_m q_m + b_m r - k_m e_m)$$

Così che $\dot{V} = -k_m e_m^2$.

ESERCIZIO 2 – SISTEMA DINAMICO SCALARE

Controllore neurale

Ovviamente la legge di controllo ideale non è applicabile perché le funzioni $f(q)$ e $H(q)$ non sono note. Chiamo $N_f(q, \overrightarrow{c_f})$ e $N_H(q, \overrightarrow{c_H})$ le versioni approssimate dalle reti neurali di queste funzioni, da cui la legge di controllo reale:

$$u_{neural} = \frac{1}{N_H}\left(-N_f - A_m q_m + b_m r - k_m e_m\right)$$

Si nota subito che essendo N_H al denominatore sarà necessario occuparsi del fatto che questa funzione non si annulli mai nel range di interesse della q.

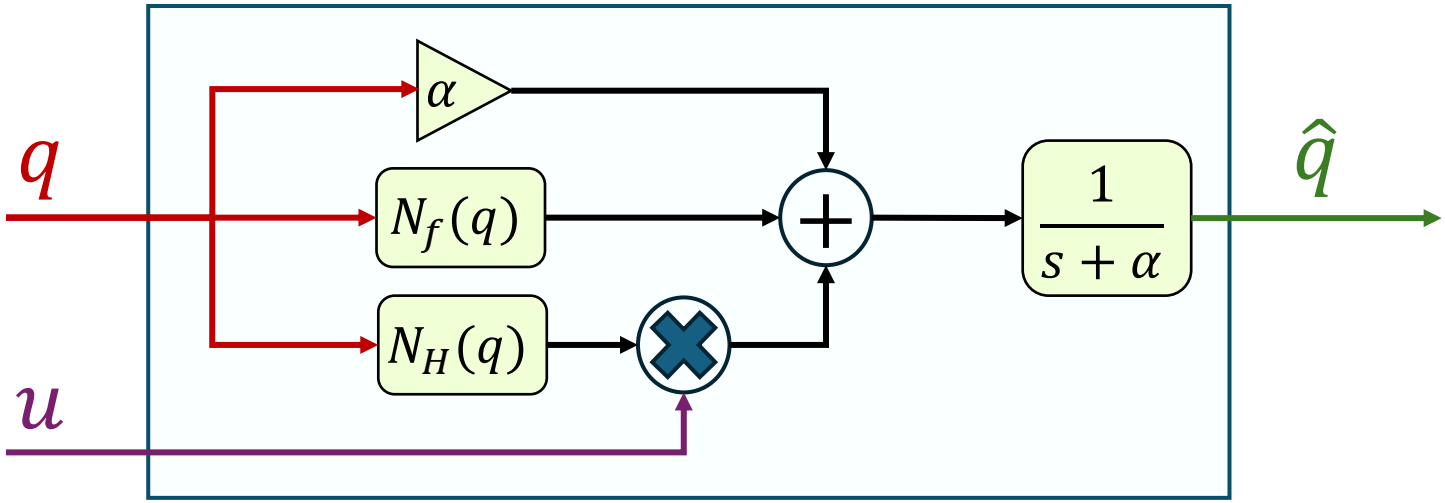
ESERCIZIO 2 – SISTEMA DINAMICO SCALARE

Modello di identificazione (1)

Scelgo un modello di identificazione della forma

$$\dot{\hat{q}} + \alpha\hat{q} = \alpha q + N_f(q) + N_H(q)u$$

Dove α è una costante reale positiva.



ESERCIZIO 2 – SISTEMA DINAMICO SCALARE

Modello di identificazione (2)

Questo modello prende il nome di «***Error Filtering Identification Scheme***» ed è presentato nell'articolo di Policarpou e Iannou. Possiamo scrivere, senza introdurre approssimazioni:

$$\dot{q} = N_f\left(q, c_f^*\right) + N_H(q, c_H^*)u + model\_error$$

Dove con c_f^* e c_H^* sono indicati i pesi teorici ottimi che minimizzano $model\_error$ per una data struttura delle reti neurali. Da questa equazione deriva un primo possibile modello di identificazione:

$$\dot{\hat{q}} = N_f\left(q, c_f\right) + N_H(q, c_H\ )u$$

Tuttavia, andando a modificare il modello con i termini aggiuntivi αq, $\alpha\hat{q}$:

$$\dot{\hat{q}} + \alpha\hat{q} = \alpha q + N_f(q) + N_H(q)u$$

Si ottiene una dinamica dell'errore di stima

$$\dot{e}_q = -\alpha e_q + \left(N_f - N_f^*\right) + (N_H - N_H^*)u - model\_error$$

Il termine aggiuntivo gioca un ruolo positivo per la convergenza dell'errore di stima.

ESERCIZIO 2 – SISTEMA DINAMICO SCALARE

Backpropagation – aggiornamento dei pesi – rete approssimante N_f

La struttura della rete è la solita dell'esercizio precedente, base di funzioni radiali gaussiane pesate

$$\begin{cases} N_f(q, \overrightarrow{c_f}) = \overrightarrow{c_f}^T \overrightarrow{\zeta_f}(q) \\ \zeta_{f,i}(q) = e^{-\frac{(q-m_{f,i})^2}{\sigma_f^2}} \end{cases}$$

La legge di aggiornamento dei pesi è la solita, con la differenza che la norma del vettore dei pesi è limitata a un valore stabilito a priori M_f.
Definendo l'errore di predizione

$$e_q \triangleq \hat{q} - q$$

Si ha la legge di aggiornamento dei pesi

$$\dot{\overrightarrow{c_f}}(q) = \begin{cases} -\gamma_f \overrightarrow{\zeta_f}(q) e_q & if \ \|\overrightarrow{c_f}\| < M_f \\ -\gamma_f \overrightarrow{\zeta_f}(q) e_q + \gamma_f \frac{e_q \overrightarrow{c_f}^T \overrightarrow{\zeta_f}(q)}{\|\overrightarrow{c_f}\|^2} \overrightarrow{c_f} & if \ \|\overrightarrow{c_f}\| \geq M_f \end{cases}$$

ESERCIZIO 2 – SISTEMA DINAMICO SCALARE

Backpropagation – aggiornamento dei pesi – rete approssimante N_H

La legge di aggiornamento dei pesi per la rete N_H cambia in quanto è coinvolto anche l'ingresso quando si deriva la funzione obiettivo rispetto ai pesi

$$\dot{\overrightarrow{c_H}}(q) = \begin{cases} -\gamma_H \overrightarrow{\zeta_H}(q) e_q u & if\ \|\overrightarrow{c_H}\| < M_H \\ -\gamma_H \overrightarrow{\zeta_H}(q) e_q u + \gamma_H \dfrac{e_q \overrightarrow{c_H}^T \overrightarrow{\zeta_H}(q) u}{\|\overrightarrow{c_H}\|^2} \overrightarrow{c_H} & if\ \|\overrightarrow{c_H}\| \geq M_H \end{cases}$$

Per garantire l'invertibilità di N_H la sua struttura è leggermente diversa, ho aggiunto un 1:

$$N_H(q, \overrightarrow{c_H}) = 1 + \overrightarrow{c_H}^T \overrightarrow{\zeta_H}(q)$$

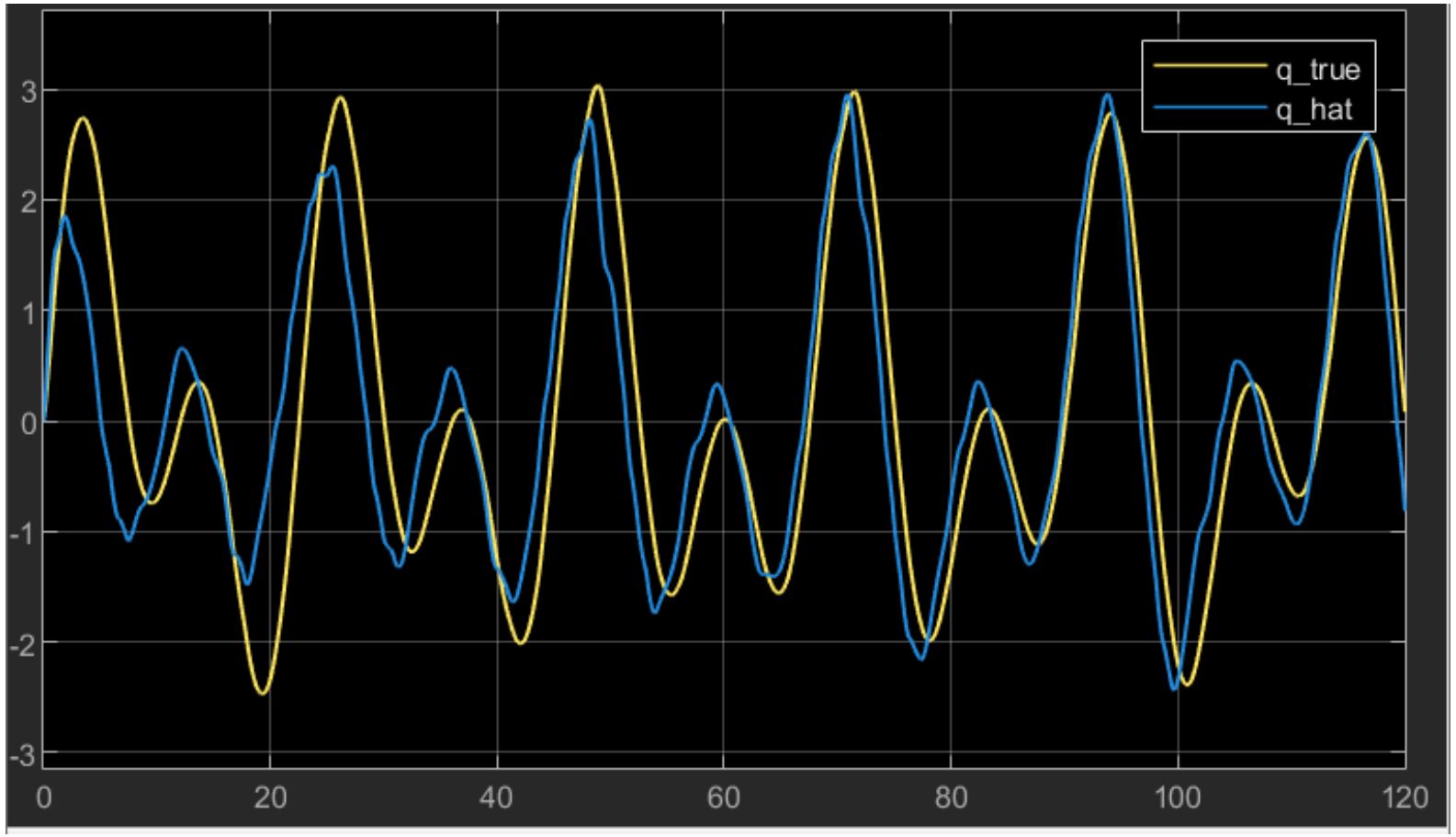
Così che, vincolando la norma dei pesi ponendo $M_H < 1$, si ha la garanzia che $N_H > 0$.

ESERCIZIO 2 – SISTEMA DINAMICO SCALARE

Es2 – Risultati
Errore di predizione

In figura è riportato l'andamento della q predetta e della q reale.

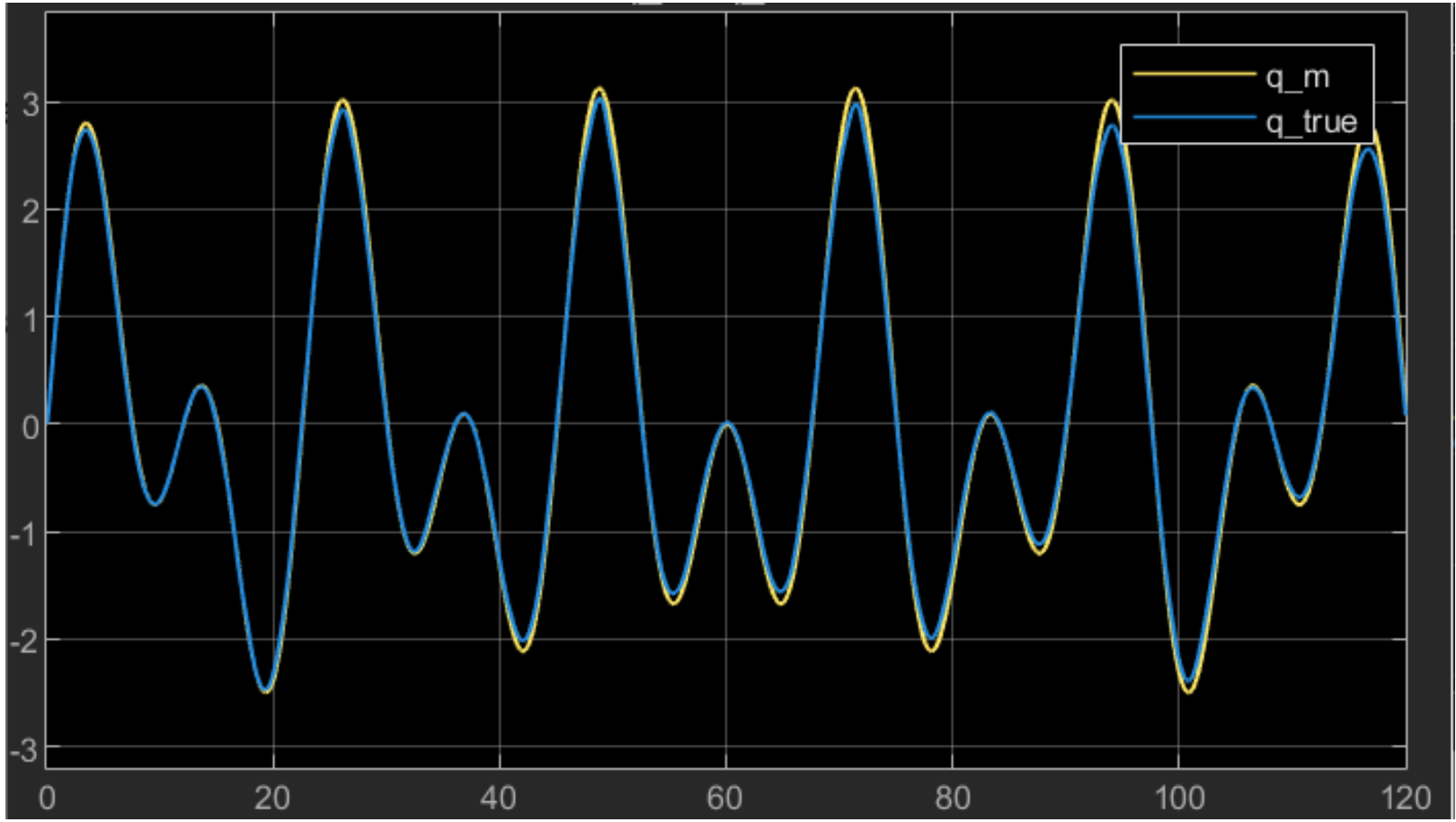
La stima è buona e migliora con l'avanzare del tempo, come ci si aspetta da una rete neurale che sta apprendendo.



ESERCIZIO 2 – SISTEMA DINAMICO SCALARE

Es2 – Risultati
Errore di inseguimento

In figura è riportato l'andamento della q_m (il movimento da inseguire) e della q reale.



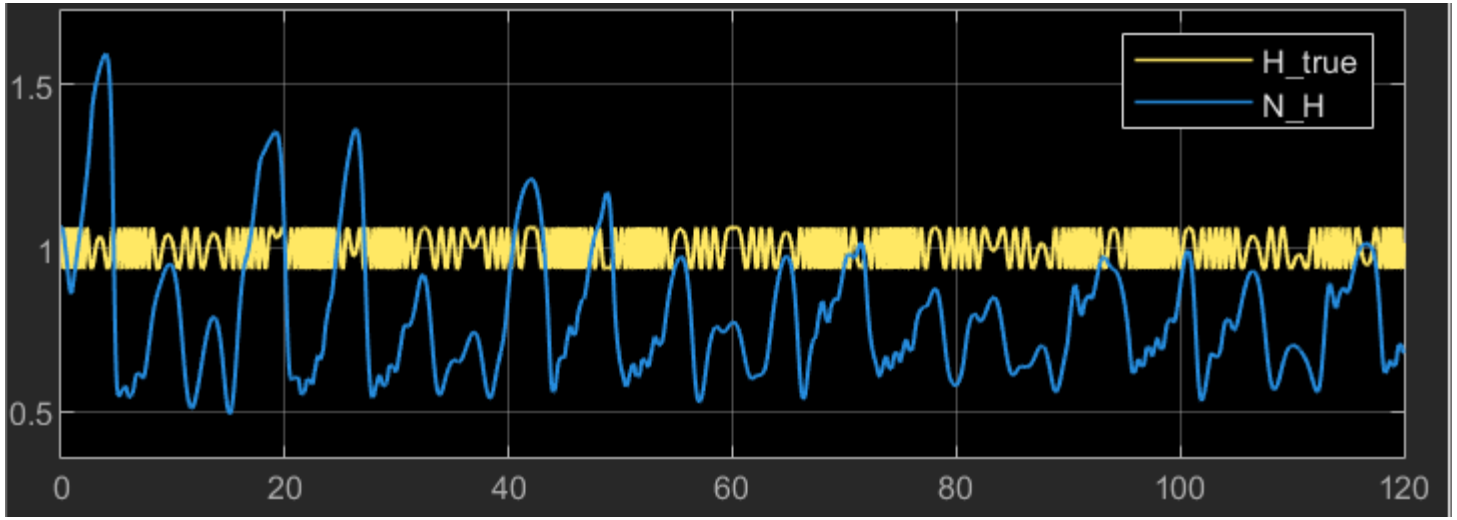
ESERCIZIO 2 – SISTEMA DINAMICO SCALARE

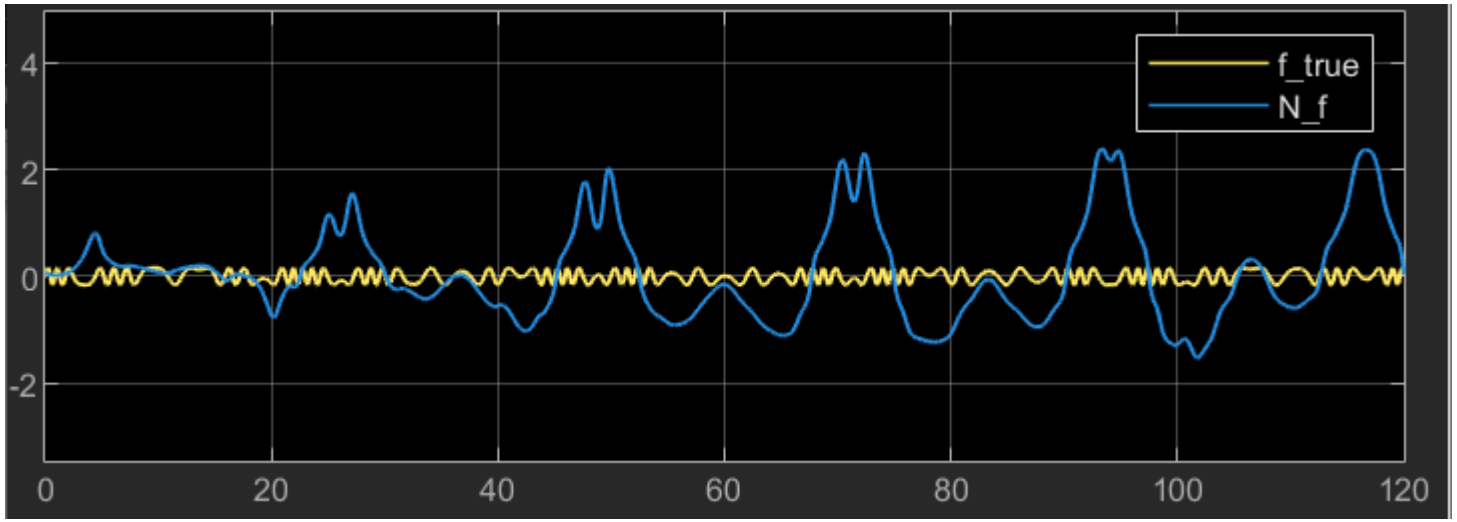
Es2 – Risultati
Funzioni approssimate

In figura sono riportati gli andamenti nel tempo delle funzioni f e H a confronto con le loro versioni approssimate dalle reti neurali N_f e N_H.

È interessante vedere che queste stime sono completamente sbagliate, ma questo non compromette il funzionamento del controllore in questo caso.

Seguono ulteriori commenti a riguardo nella slide successiva.





ESERCIZIO 2 – SISTEMA DINAMICO SCALARE

Es2 – Risultati
Commento sull'errore di approssimazione (parte 1)

Il fatto che le approssimazioni delle due funzioni N_f e N_H siano individualmente sbagliate, non compromette il funzionamento complessivo del modello di identificazione.

A differenza dell'esercizio precedente, qua abbiamo due leve (e non una) su cui agire per minimizzare la funzione obiettivo (errore di predizione quadratico).

In altre parole, lo stesso valore $\dot{q}(q)$ può essere ottenuto a partire da combinazioni diverse delle funzioni $f(q)$ e $H(q)$. Se questa ridondanza non viene gestita in qualche modo, la rete si assesta su una combinazione ammissibile «a piacere» (ovvero non controllata da chi progetta la rete), che è quello che è successo in questo caso.

ESERCIZIO 2 – SISTEMA DINAMICO SCALARE

Es2 – Risultati
Commento sull'errore di approssimazione (parte 2)

Questo comportamento della rete può essere visto negativamente, se uno degli obiettivi è effettivamente quello di identificare il sistema, in quanto ovviamente si fallisce. Ma lo stesso comportamento può essere un vantaggio se l'unico obiettivo è quello di controllare il sistema in esame.

Si prenda ad esempio un caso in cui la funzione reale $H(q)$ per qualche valore della variabile di configurazione si annulla: applicando una tecnica di controllo che prevede l'inversione di questa si potrebbe incorrere in una singolarità.

Tuttavia, se vincoliamo la funzione approssimante $N_H(q)$ a stare lontana dallo 0 con una qualche tecnica (nel mio caso ho sommato 1 e limitato la norma dei pesi), si potrebbe aggirare la singolarità nella legge di controllo.

Questo però non cambia il fatto che il sistema reale in quella particolare configurazione sia incontrollabile.

ESERCIZIO 2 – SISTEMA DINAMICO SCALARE

Fine Esercizio 2

ESERCIZIO 3 – RR PLANARE

Sistema vero

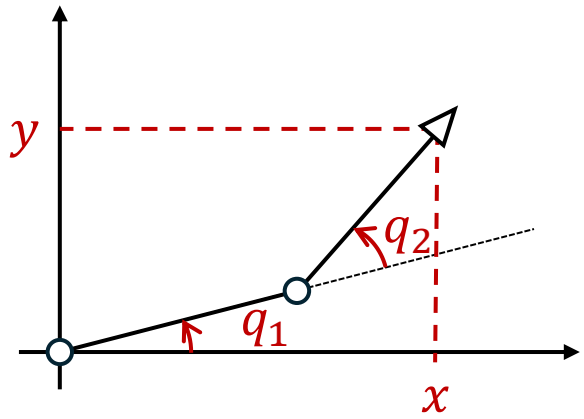
In questo esercizio il sistema dinamico da controllare è un classico manipolatore RR planare

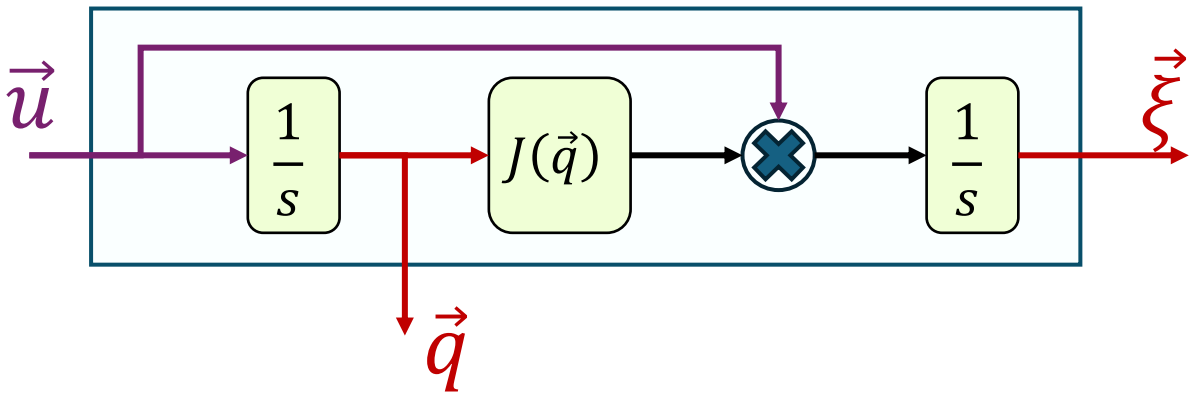
$$\dot{\vec{\xi}} = J(\vec{q})\dot{\vec{q}} = J(q)\vec{u}$$

Con $\vec{q} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$; $\vec{\xi} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$J(\vec{q}) = \begin{bmatrix} -L_1 s(q_1) - L_2 s(q_1 + q_2) & -L_2 s(q_1 + q_2) \\ +L_1 c(q_1) + L_2 c(q_1 + q_2) & +L_2 c(q_1 + q_2) \end{bmatrix}$$

L'obiettivo è fare un **controllo di movimento** dell'end-effector, senza sapere nulla riguardo il jacobiano $J(\vec{q})$, se non il range in cui vive $\vec{q}$.





ESERCIZIO 3 – RR PLANARE

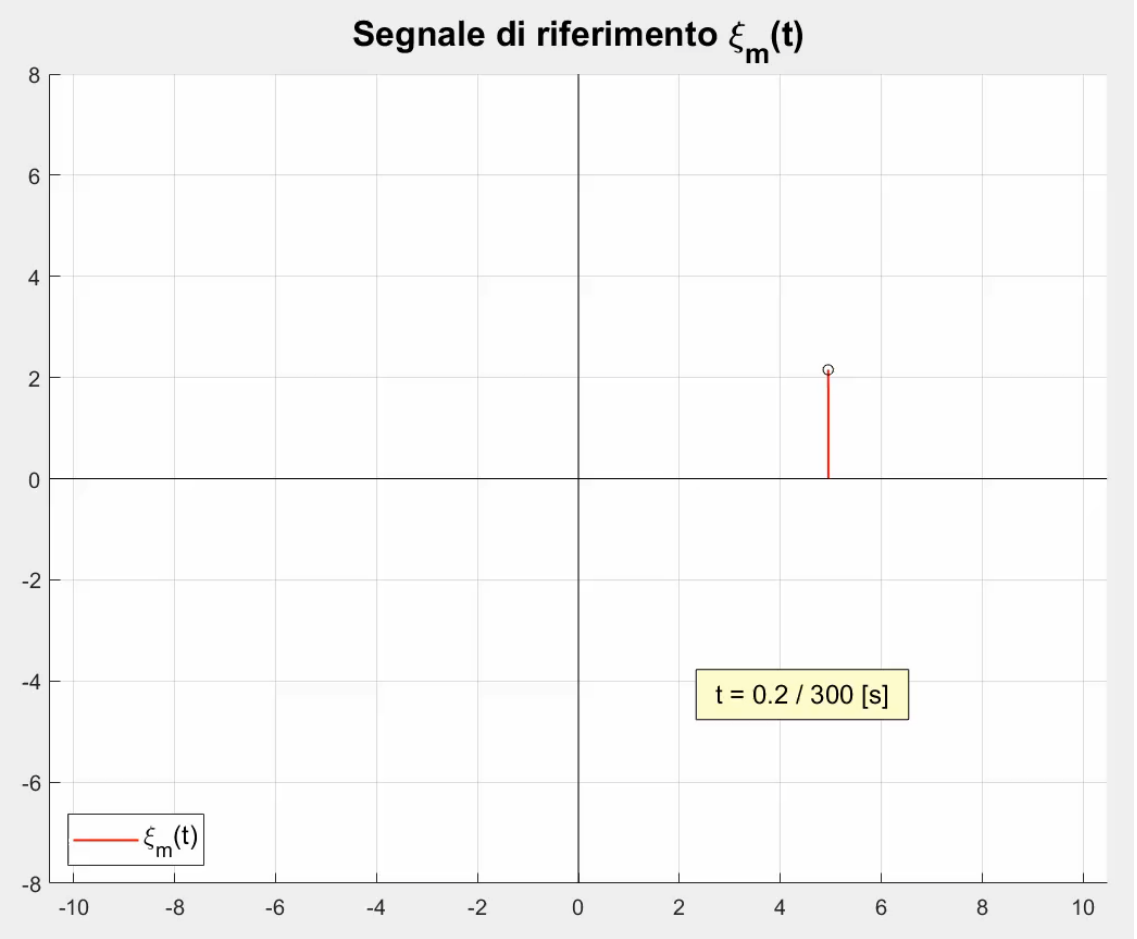
Generazione del riferimento

Per generare il segnale di riferimento $\overrightarrow{\xi_m}(t)$ è stato utilizzato lo stesso approccio dell'esercizio precedente

$$\begin{cases} \dot{\overrightarrow{\xi_m}} = -A_m \overrightarrow{\xi_m} + B_m \vec{r} \\ \vec{r} = B_m^{-1}(A_m \overrightarrow{\xi_m} + k_r(\overrightarrow{\xi_r}(t) - \overrightarrow{\xi_m})) \end{cases}$$

In questo modo devo soltanto scegliere $\overrightarrow{\xi_r}(t)$

$$\overrightarrow{\xi_r}(t) = \begin{bmatrix} \frac{L_1 + L_2}{\sqrt{2}} \cos(\omega_{r,1} t) \\ \frac{L_1 + L_2}{2\sqrt{2}} \left(1 + sin(\omega_{r,2} t)\right) \end{bmatrix}$$



ESERCIZIO 3 – RR PLANARE

Controllore ideale

Definisco l'errore

$$\overrightarrow{e_m} \triangleq \vec{\xi} - \overrightarrow{\xi_m}$$

Costruisco una Lyapunov

$$V = \frac{1}{2}\overrightarrow{e_m}^T\overrightarrow{e_m}$$

$$\dot{V} = \overrightarrow{e_m}^T\left(J\vec{u} + A_m\overrightarrow{\xi_m} - B_m\vec{r}\right)$$

Costruisco l'ingresso $\vec{u}$ in modo che annulli i termini «scomodi» e renda $\dot{V}$ negativa definita

$$\overrightarrow{u_{ide}} = J^{-1}\left(-A_m\overrightarrow{\xi_m} + B_m\vec{r} - k_m\overrightarrow{e_m}\right)$$

Così che $\dot{V} = -k_m\overrightarrow{e_m}^T\overrightarrow{e_m}$.

ESERCIZIO 3 – RR PLANARE

Controllore neurale

Ovviamente la legge di controllo ideale non è applicabile perché il Jacobiano $J(\vec{q})$ non è noto. Chiamo $N_J(\vec{q}, \overrightarrow{c_J})$ la sua versione approssimata dalle reti neurali, da cui il controllore «reale»:

$$\overrightarrow{u_{neural}} = N_J^{-1}\left(-A_m\overrightarrow{\xi_m} + B_m\vec{r} - k_m\overrightarrow{e_m}\right)$$

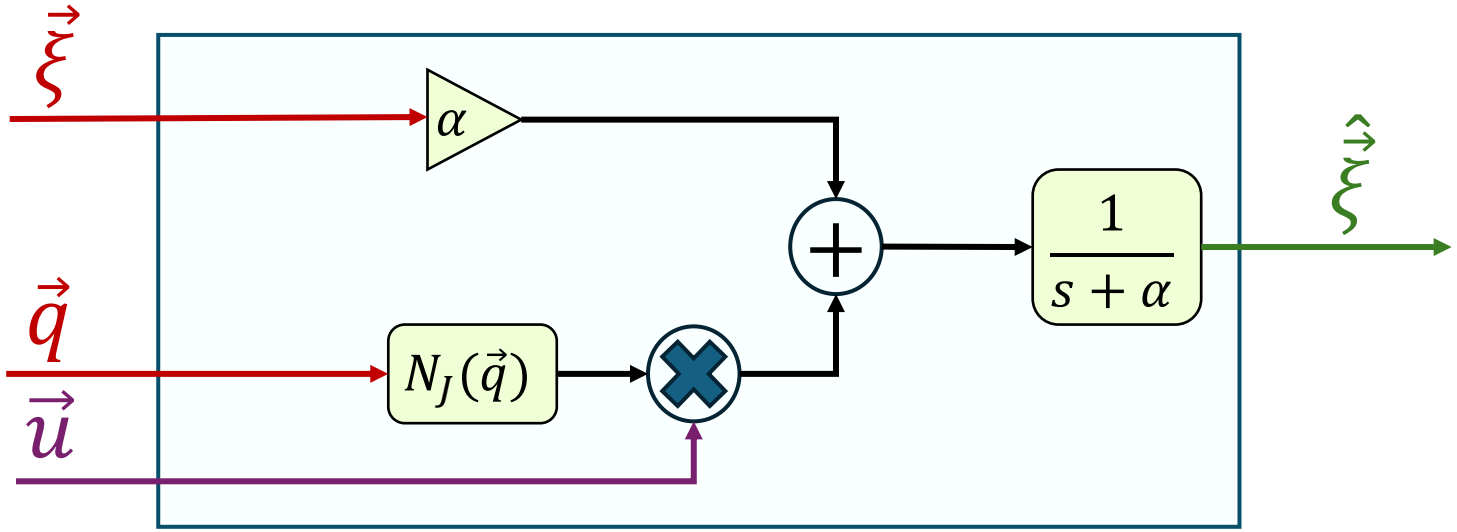
ESERCIZIO 3 – RR PLANARE

Modello di identificazione

Scelgo un modello di identificazione della forma

$$\dot{\hat{\vec{\xi}}} + \alpha\hat{\vec{\xi}} = \alpha\vec{\xi} + N_J(\vec{q})\vec{u}$$

Dove α è una costante reale positiva.



ESERCIZIO 3 – RR PLANARE

Struttura della rete neurale

In questo esercizio le differenze sostanziali dagli esercizi precedenti sono legate alla dimensione degli spazi, infatti in questo caso la «funzione» da approssimare è

$$J(\vec{q})\colon\ \mathbb{R}^2 \rightarrow\ \mathbb{R}^{2x2}$$

Questa novità può essere scomposta in due sotto-novità da affrontare separatamente:

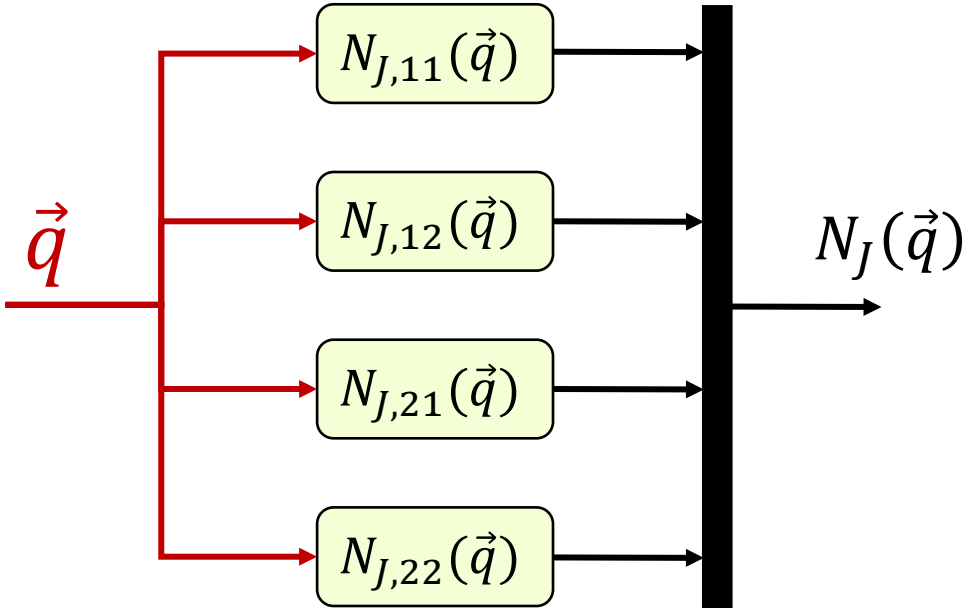
1. Lo spazio di arrivo non è scalare

2. Lo spazio di partenza non è scalare

ESERCIZIO 3 – RR PLANARE

Struttura della rete neurale – spazio di arrivo non scalare

Questo si affronta introducendo tante reti quante sono le entrate della matrice



ESERCIZIO 3 – RR PLANARE

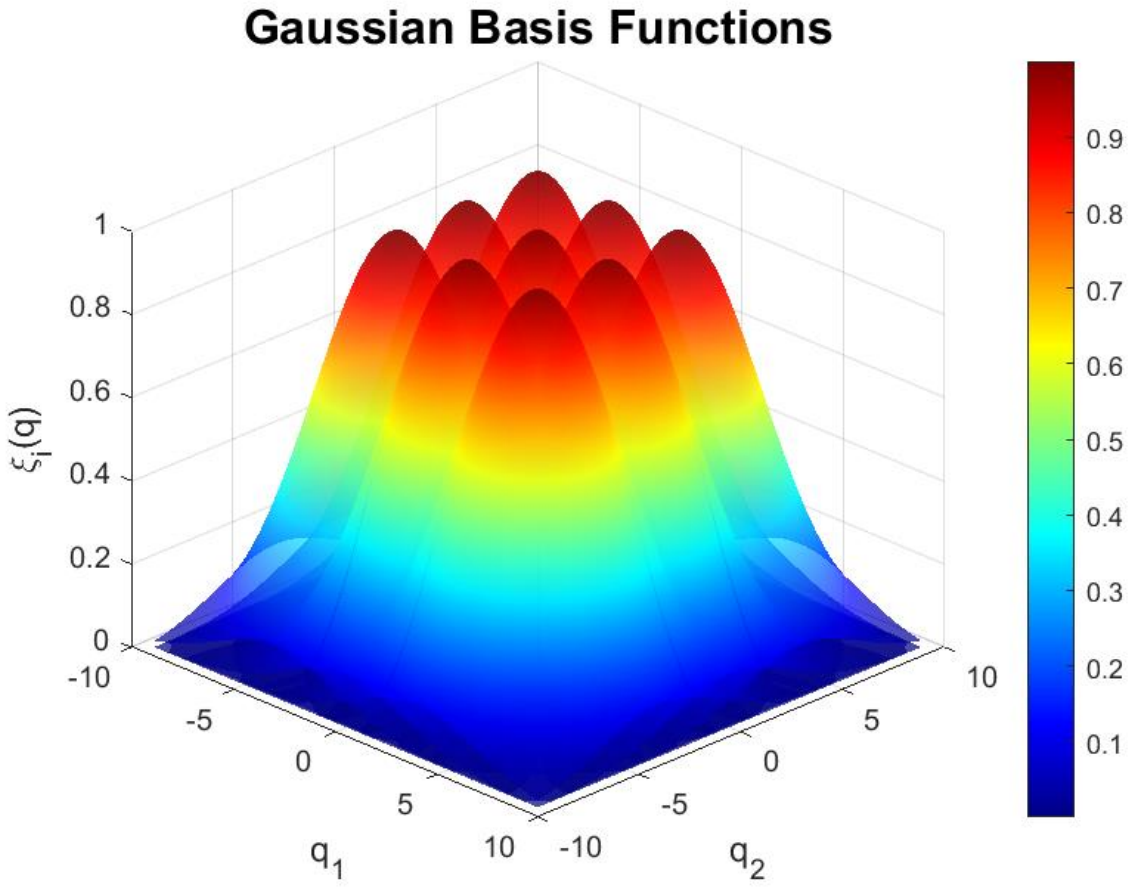
Struttura della rete neurale – spazio di partenza non scalare

In questo caso le funzioni di base devono essere gaussiane vettoriali e non più scalari

$$\zeta_{ij_k}(\vec{q}) = e^{-\frac{1}{2}(\vec{q}-\overrightarrow{m_k})^T \Sigma^{-1}(\vec{q}-\overrightarrow{m_k})}$$

Questo introduce una problematica detta «**maledizione della dimensionalità**»: sia n_q il numero di gaussiane **per asse**, se lo spazio di partenza ha dimensione $\mathbb{R}^p$ il numero di gaussiane (e quindi di pesi da aggiornare) diventa:

$$n° \, pesi = (n_q)^p$$



ESERCIZIO 3 – RR PLANARE

Backpropagation – aggiornamento dei pesi

Definendo l'errore di predizione

$$\overrightarrow{e_\xi} \triangleq \hat{\vec{\xi}} - \vec{\xi} = \begin{bmatrix} e_{\xi 1} \\ e_{\xi 2} \end{bmatrix}$$

Si ha la legge di aggiornamento dei pesi

$$\dot{\overrightarrow{c_{ij}}}(q) = \begin{cases} -\gamma \overrightarrow{\zeta_{ij}}(\vec{q}) e_{\xi i} u_j & if \ \left\| \overrightarrow{c_{ij}} \right\| < M \\ -\gamma \overrightarrow{\zeta_{ij}}(\vec{q}) e_{\xi i} u_j + \gamma \dfrac{e_{\xi i} \overrightarrow{c_{ij}}^T \overrightarrow{\zeta_{ij}}(q) u_j}{\left\| \overrightarrow{c_{ij}} \right\|^2} \overrightarrow{c_{ij}} & if \ \left\| \overrightarrow{c_{ij}} \right\| \geq M \end{cases}$$

ESERCIZIO 3 – RR PLANARE

Es3 – Fase di apprendimento

Mentre nell'esercizio precedente si è usato fin da $t = 0$ il controllore neurale, in questo caso non si può fare perché la struttura della rete è tale per cui all'istante iniziale le entrate $N_{ij}(\vec{q})$ sono tutte uguali tra loro, e quindi la matrice $N_J(\vec{q})$ non è invertibile.

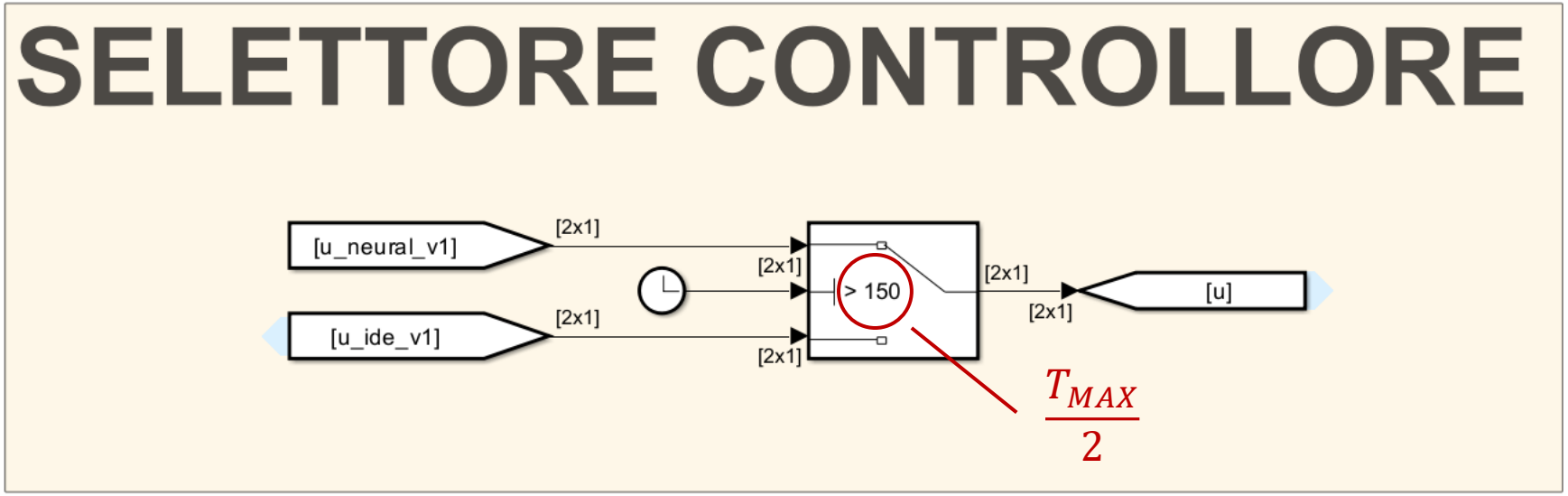
Inoltre, è di poco senso pratico usare il controllore neurale a $t = 0$ in quanto non avendo avuto occasione di apprendere il modello, si tradurrebbe in un controllo casuale dettato dalle condizioni iniziali arbitrarie dei pesi delle reti, che quasi sicuramente si tradurrebbe in comportamenti instabili.

Una strategia potrebbe essere quella di prevedere una fase di apprendimento in cui gli ingressi $\vec{u}(t)$ sono costruiti a tavolino in modo da esplorare lo spazio delle configurazioni $\vec{q}$.

ESERCIZIO 3 – RR PLANARE

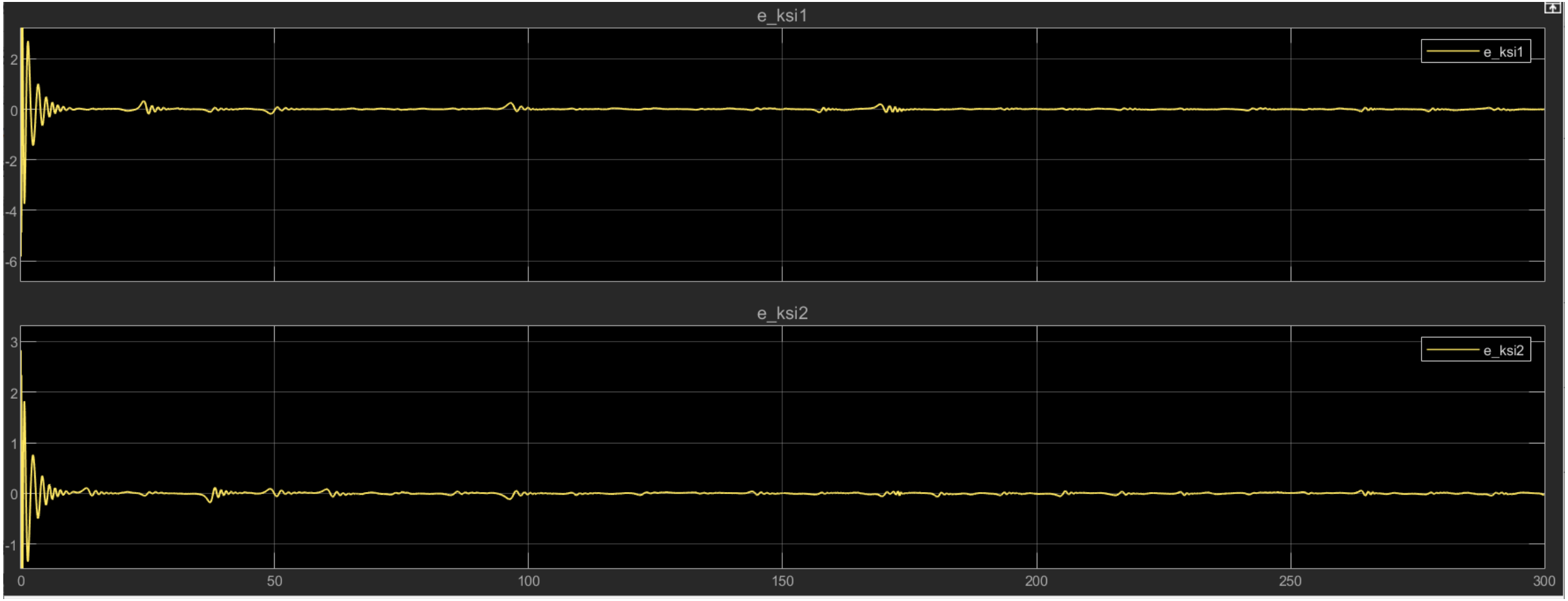
Es3 – Fase di apprendimento

Per simulare la fase di apprendimento in questo esercizio ho deciso di fare sì che per la prima metà del tempo di simulazione il controllore è quello ideale.



ESERCIZIO 3 – RR PLANARE

Es3 – Risultati – Errore di predizione

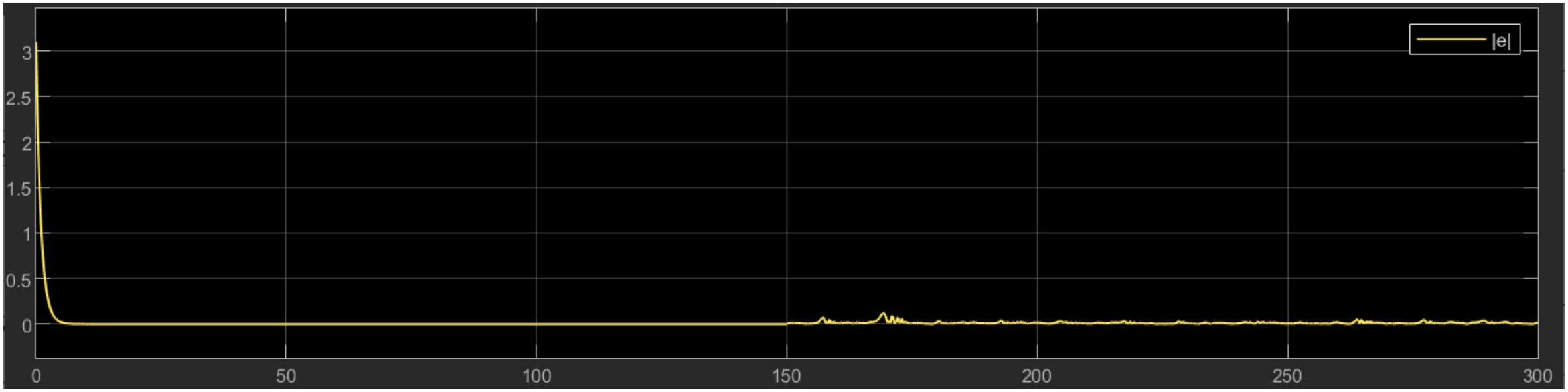


ESERCIZIO 3 – RR PLANARE

Es3 – Risultati – Errore di inseguimento

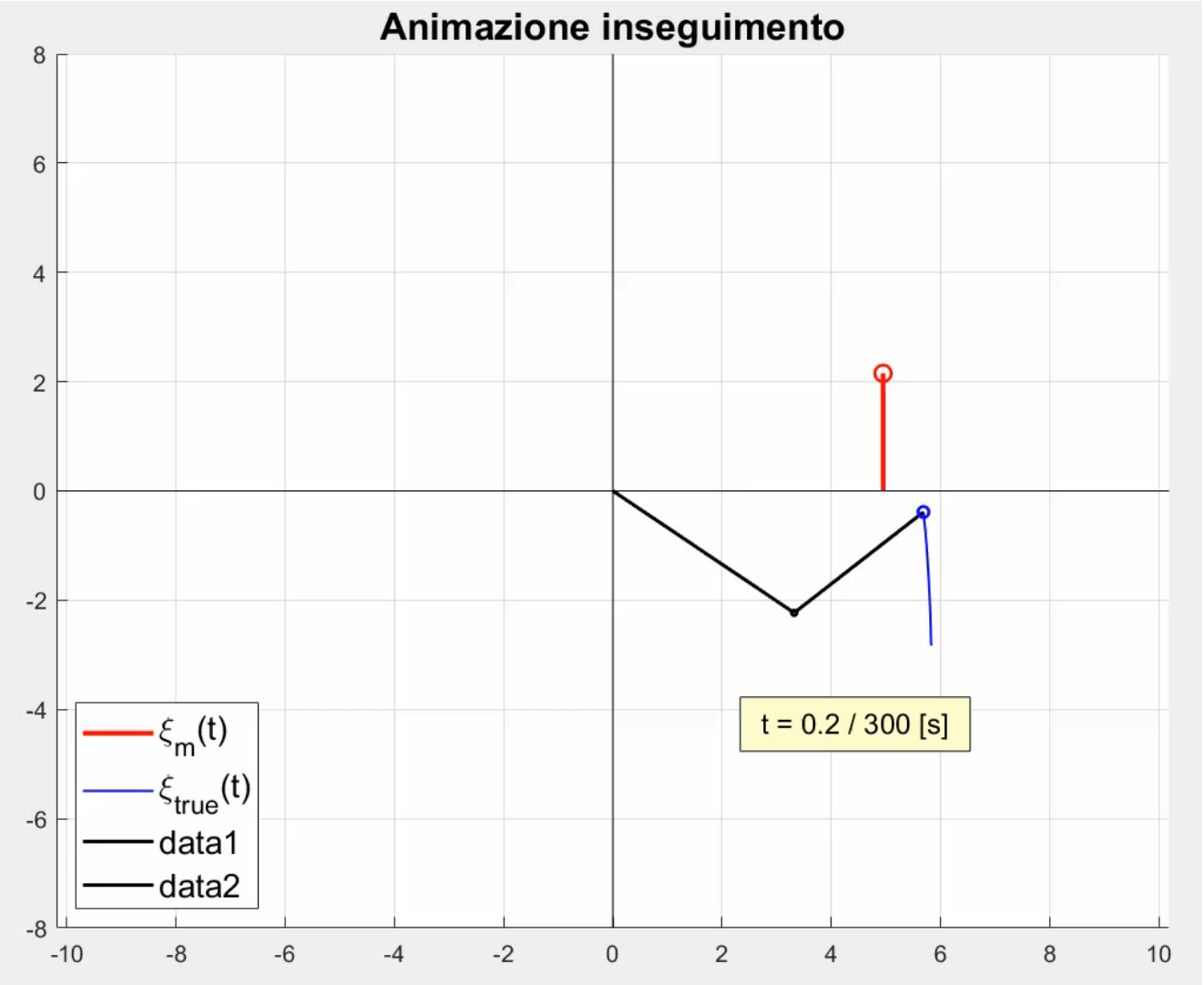
È riportato l'andamento della norma dell'errore di inseguimento.

Si noti che in questo grafico l'unica parte interessante è quella da $t = 150$ in poi, in quanto è la fase in cui sta funzionando il controllore neurale.



ESERCIZIO 3 – RR PLANARE

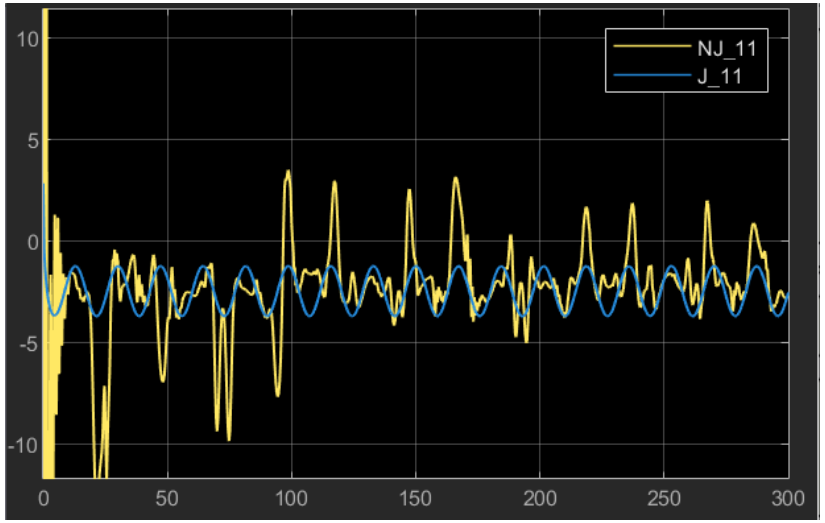
Es3 – Risultati – Animazione

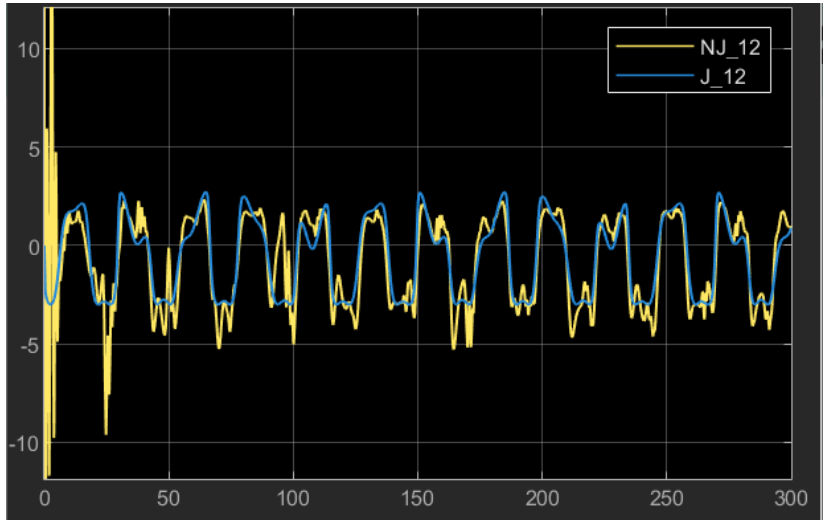


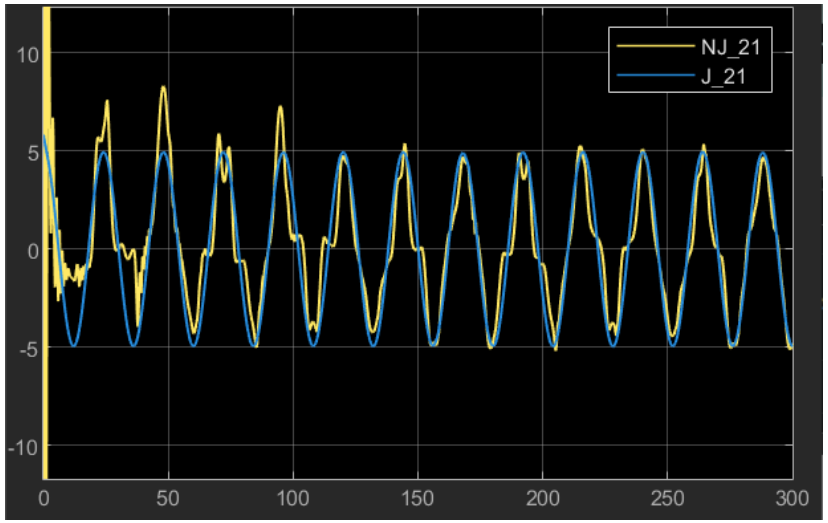
ESERCIZIO 3 – RR PLANARE

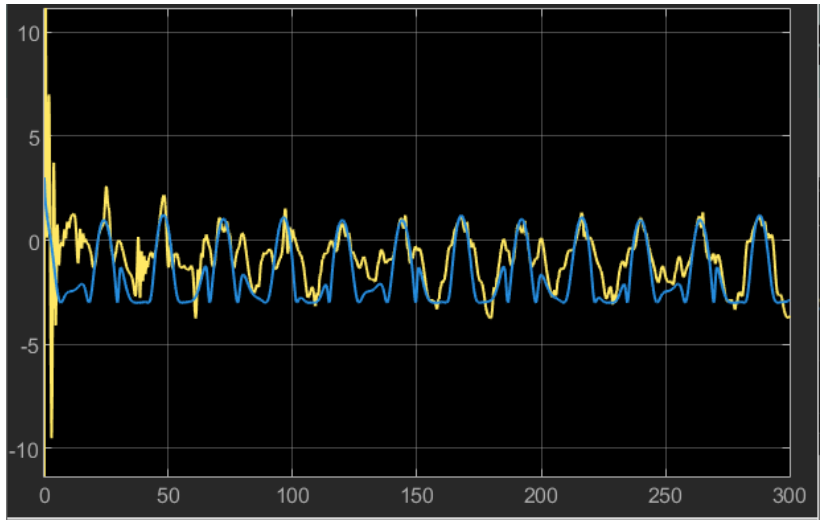
Es3 – Risultati – Errore di identificazione

Anche in questo esercizio si nota che se pur l'identificazione non è perfetta si riesce a raggiungere un errore di inseguimento più che soddisfacente.









ESERCIZIO 3 – RR PLANARE

Fine Esercizio 3