UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI INGEGNERIA

Corso di Laurea triennale in **INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE**

Tesi di laurea triennale

CONTROLLO DATA-DRIVEN BASATO SU LINEARIZZAZIONE DINAMICA

DATA-DRIVEN CONTROL BASED ON DYNAMIC LINEARIZATION

Relatore: Prof. Giuseppe Orlando
Correlatore: Prof. Gianluca Ippoliti

Candidato: Carmen Andreozzi

A.A. **2020/2021**

Ringraziamenti

Innanzitutto, ringrazio il Professor Orlando e il Professor Ippoliti perché grazie al loro aiuto e alla loro disponibilità sono stata in grado di affrontare l'esperienza del tirocinio e della stesura della tesi. Grazie, inoltre, per avermi trasmesso la vostra passione, motivo per il quale ho deciso di intraprendere con voi questo percorso importante.

Ringrazio tutta la mia famiglia che mi è stata accanto con tutto l'affetto e la premura di cui avevo bisogno durante questi anni, anche nei momenti più difficili. Grazie perché senza di voi non sarei arrivata dove sono oggi e non avrei mai creduto in me stessa come lo avete fatto voi ogni giorno. Mi ritengo una persona davvero fortunata poiché non mi sono mai sentita sola e ho sempre avuto da parte vostra tutto l'appoggio che mi serviva.

Ringrazio Fabrizio, il mio ragazzo, che ha sempre creduto in me dal primo momento che mi ha conosciuta. Ogni tuo gesto e ogni tua parola sono stati determinanti per aiutarmi a oltrepassare gli ostacoli che ho incontrato. Sei la persona più speciale che io abbia mai conosciuto e ti ringrazio perché ci sei sempre stato per me.

Ringrazio le mie amiche da una vita: Marianna, Valeria, Damiana, Chiara e Alessia. Siete sempre state la mia ancora di salvezza e la spalla su cui piangere nei momenti difficili. Ogni volta che avevo bisogno di voi eravate sempre presenti, nessuna esclusa. Siete le amiche che tutti vorrebbero e non potrei chiedere di meglio.

Grazie, infine, ai miei compagni di corso e agli amici che ho conosciuto all'università, in particolare Alex, Francesca, Kevin, Piero, Antonio e Teresa. Senza di voi tutto ciò non sarebbe stato possibile e spero che la nostra amicizia duri negli anni.

Indice

Introduzione

Le teorie di controllo moderne consistono in un preciso numero di passaggi da seguire, sia per quanto riguarda i sistemi lineari che quelli non lineari. Il passaggio principale prevede di modellare il processo che deve essere controllato per ottenere una rappresentazione matematica da cui partire.

La teoria di controllo che si basa su questi passaggi è chiamata *model-based control (MBC) theory*.

Questo documento ha lo scopo di analizzare i limiti principali della tecnica appena citata e di presentare un nuovo metodo chiamato *data-driven control (DDC)*.
Questo metodo può essere considerato complementare alla teoria MBC; infatti, si basa sulla progettazione del controllore a prescindere dalla conoscenza del modello matematico del processo.

Nel corso dell'elaborato saranno inoltre presentate le implementazioni degli algoritmi in *Matlab* e le simulazioni in *Simulink*.

1 Model-based Control

1.1 Generalità

La tecnica di controllo MBC fa parte della teoria di controllo moderna, nata nel 1960 grazie a Kalman, considerato il padre della teoria dei sistemi e del controllo.
Con lo sviluppo di questa tecnica sono nati diversi metodi che si basano tutti sullo stesso funzionamento. Ad esempio, per i sistemi lineari si può parlare di controllo robusto o dell'assegnazione dei poli e degli zeri, mentre per i sistemi non lineari si ricorre alla progettazione del controllore basata sulla teoria di *Lyapunov*.

In generale, la progettazione del sistema di controllo si basa su due passi fondamentali da seguire:

1. la modellazione o l'identificazione del processo da controllare;
2. la progettazione del controllore basata sul modello matematico del processo precedentemente ottenuto.

Come introdotto in precedenza, la teoria MBC si basa sulla conoscenza *a priori* del modello matematico del processo da controllare. Senza questa conoscenza, la tecnica MBC risulta completamente inutile.

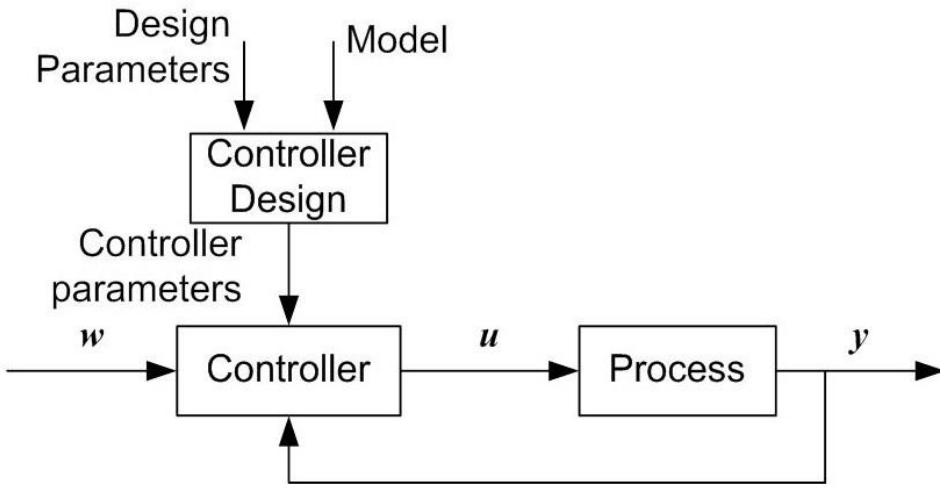

Fig. 1: Teoria MBC

Il principio su cui si basa questa tecnica porta innumerevoli e inevitabili problemi, dovuti ai profondi cambiamenti a cui sono sottoposte le industrie

ogni giorno. Infatti, a causa dello sviluppo tecnologico che diventa sempre più evidente, i processi da controllare sono sempre più complessi.

Innanzitutto, la modellazione matematica di un problema reale porta necessariamente con sé delle approssimazioni. Di conseguenza il modello ottenuto conterrà degli errori a causa della mancanza di modellazione di particolari dinamiche che caratterizzano il fenomeno che si sta studiando. Per cercare di risolvere questa mancanza, si è cercato di "rafforzare" la teoria del controllo robusto rendendola in grado di includere anche gli errori di modellazione. Tuttavia, il problema non può essere risolto del tutto.

In secondo luogo, sta diventando sempre più complesso rappresentare analiticamente un processo a causa delle strutture tempo-varianti, delle non-linearità, etc. Inoltre, anche se fosse possibile rappresentare il processo, si tratterebbe di una rappresentazione troppo complessa proprio per la presenza di questi fattori.

Si può affermare, infine, che la teoria MBC risulterebbe completamente inutile se il processo non fosse disponibile analiticamente.

Le osservazioni precedentemente illustrate permettono di evidenziare la difficoltà dell'utilizzo di teorie di controllo che prevedono la conoscenza a priori del modello matematico.

Allo stesso modo, è evidente che progettare un controllore che sia basato su un modello matematico inesatto potrebbe portare a comportamenti indesiderati e all'instabilità.

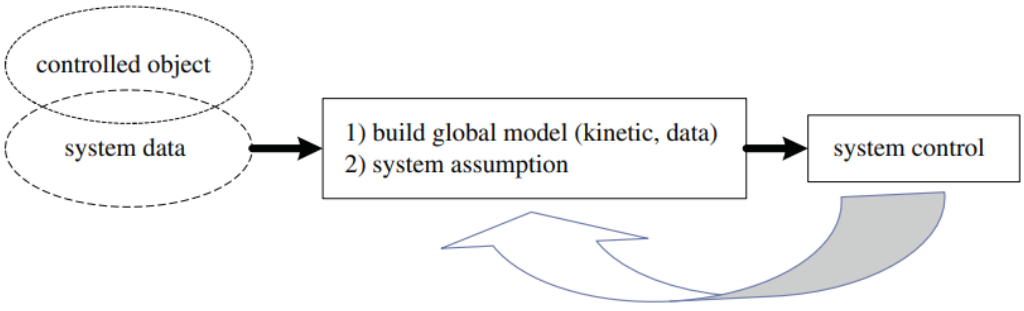

Fig. 2: Funzionamento MBC

La Fig. 2 mostra l'architettura principale della teoria MBC. Dal diagramma si può osservare come il sistema di controllo dipenda dalle assunzioni effettuate a partire dal processo.

Dopo aver analizzato brevemente le caratteristiche relative a questa tecnica di controllo e averne evidenziato le problematiche principali, si può affermare che la conoscenza *a priori* del processo non è sempre la modalità migliore da cui partire per progettare un sistema di controllo. E' per questo motivo che nel corso dell'elaborato si analizzerà più approfonditamente la tecnica DDC, nata per risolvere quelle problematiche per le quali la tecnica MBC risulta poco vantaggiosa.

2 Data-Driven Control

2.1 Generalità

Con i cambiamenti che le industrie stanno costantemente subendo, è nata la necessità di sfruttare tecniche di controllo più consone agli sviluppi che si sono registrati nel corso del tempo, motivo per il quale i sistemi da controllare presentano delle complessità sempre più forti.

I processi industriali attuali producono delle quantità enormi di dati ed è proprio per questo motivo che si è pensato di utilizzare queste informazioni per la progettazione dei sistemi di controllo. Pertanto, si è diffusa sempre di più la tecnica di controllo *data-driven*.

Definizione 1: Il controllo *data-driven* consiste nelle teorie di controllo secondo le quali il controllore è progettato direttamente usando i dati I\O (sia on-line che off-line) del sistema controllato senza utilizzare esplicitamente o implicitamente il modello matematico del processo da controllare, la cui stabilità, convergenza e robustezza sono garantite secondo rigorose analisi matematiche e assunzioni effettuate.

Grazie a questo approccio, completamente diverso da quello classico, è possibile risolvere tutte le problematiche precedentemente analizzate relative alla tecnica MBC. La disponibilità di grandi quantità di dati a costo praticamente nullo permette di intuire la convenienza di sfruttare questi ultimi per ottenere un sistema di controllo ad alto livello di efficacia.

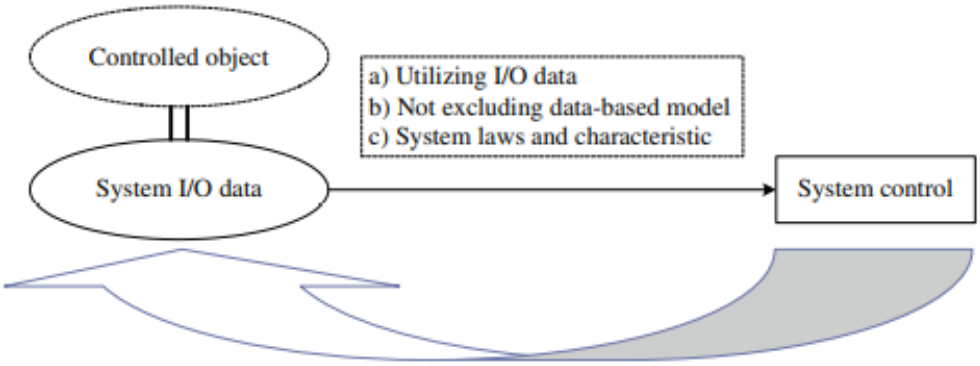

Fig. 3: Funzionamento DDC

Nella Fig.3 è descritta schematicamente l'architettura alla base del metodo DDC, il quale non valuta nessuna considerazione inerente al processo, bensì mette le sue basi sui dati I\O del sistema controllato, garantendo il soddisfacimento di tutte le proprietà come la robustezza, la convergenza e la stabilità.

Le caratteristiche analizzate ci permettono di affermare che il mondo del controllo moderno è diviso in due grandi categorie: la teoria MBC e la teoria DDC.
In generale, una tecnica di controllo che si può considerare efficiente dovrebbe essere in grado di gestire un problema di controllo a prescindere dai dati che ha a disposizione. Da questo si deduce che le due teorie analizzate non possono rispondere a questa esigenza, dal momento che presentano entrambe dei limiti. Per questo motivo le due tecniche possono considerarsi l'una complementare all'altra, in quanto seguono due modalità di approccio totalmente opposte. Infatti, non è possibile né tantomeno conveniente applicare ambedue le tecniche allo stesso problema di controllo.

2.2 I metodi DDC

Nel corso degli anni sono stati sviluppati diversi metodi basati sul data-driven control nel dominio del tempo. Di seguito ne saranno analizzati alcuni.

- *Proportional-integral-derivative control (PID)*: è il primo metodo data-driven che è stato utilizzato. Come indica il nome, si basa su un'azione proporzionale, una integrale e, infine, una derivativa. Attraverso tutte queste azioni (o solo alcune di esse) il controllore riesce a regolare l'uscita del sistema per portarla a un valore desiderato.

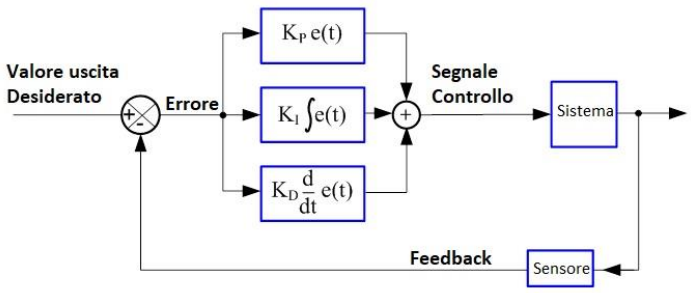

Fig. 4: Schema a blocchi PID

La Fig.4 aiuta a comprendere il funzionamento di questo metodo, il quale si basa su un sistema in retroazione negativa per poter calcolare l'errore tra la grandezza misurata e quella desiderata.

- *Iterative feedback tuning (IFT)*: è un altro approccio data-driven che utilizza delle iterazioni per ottimizzare i parametri di controllo di un determinato sistema a ciclo chiuso.

- *Virtual reference feedback tuning (VRFT)*: è un metodo data-driven utilizzato per i sistemi lineari tempo invarianti (LTI).

- *Unfalsified control (UC)*: questo metodo consiste nel considerare ricorsivamente dei parametri di controllo inesatti, in modo tale da ricavare quelli corretti e ottenere il controllore.

- *Model-free adaptive control (MFAC)*: è il metodo di controllo che sarà analizzato nel corso di questo documento. E' stato ideato per sistemi non lineari a tempo discreto e si basa su un particolare metodo di linearizzazione che sarà descritto più avanti.

3 Linearizzazione dinamica

3.1 Generalità

In questo documento si analizzerà una tecnica di controllo utilizzata per sistemi discreti lineari e non lineari. Perciò è necessario introdurre una linearizzazione per i sistemi non lineari, in modo da semplificare la progettazione del sistema di controllo, le varie analisi e le applicazioni.

Esistono diversi metodi di linearizzazione, come ad esempio la linearizzazione basata sugli sviluppi di Taylor: essa consiste nel considerare un intorno di un punto x_0 della funzione da linearizzare. Se la funzione è derivabile, può essere considerata al suo posto la funzione linearizzata nel punto x_0.

Questo metodo di linearizzazione, come molti altri, non è adatto alla tecnica di controllo che si vuole analizzare in questo documento, poiché nella maggior parte dei casi è necessario il modello matematico esatto del processo controllato, oppure si ottengono linearizzazioni troppo complesse. Infatti, prendendo come esempio gli sviluppi di Taylor, considerando solo i primi termini si otterrebbe un'approssimazione troppo semplice che non tiene conto dei comportamenti della funzione in punti più o meno lontani da quello preso in considerazione. Al contrario, se non si vuole perdere nessuna informazione, e quindi si considera un elevato numero di termini, si avrebbe un'approssimazione troppo complessa e sconveniente.

Per questo motivo, si introduce una tecnica chiamata *linearizzazione dinamica (DL)* sviluppata proprio per i sistemi non lineari a tempo discreto e dalla quale è possibile ottenere un sistema esattamente equivalente.

3.2 DL per sistemi non lineari a tempo discreto

La tecnica di linearizzazione dinamica (DL) presenta numerosi vantaggi: in particolare, si tratta di un metodo data-driven per sistemi lineari non noti e per i quali non è necessaria la conoscenza del modello matematico. Inoltre, il risultato che si ottiene dalla linearizzazione presenta una forma poco complessa e descritta da pochi parametri.

La linearizzazione dinamica è stata estesa dalla forma compatta (*CFDL: compact-form dynamic linearization*), alla forma parziale (*PFDL: partial-form dynamic linearization*) e a quella completa (*FFDL: full-form dynamic linearization*).

Questa tecnica, inoltre, si basa sul nuovo concetto chiamato *PPD* che è una struttura tempo-variante che non deve essere necessariamente conosciuta a priori e che può essere stimata utilizzando l'analisi delle coppie ingresso-uscita del sistema a ciclo chiuso che si vuole controllare.

Di seguito si valuteranno i vari tipi di DL partendo dalla PFDL, per poi analizzare più approfonditamente la FFDL. Successivamente verrà introdotta la legge di controllo che si baserà proprio sulla FFDL.

3.2.1 PDFL

Consideriamo un sistema NARMAX (*nonlinear auto-regressive moving average model with exogenous inputs*) descritto nella seguente forma:

$$y(k+1) = f\left(y(k), \dots, y\left(k-n_y\right), u(k), \dots, u(k-n_u)\right) \quad (1)$$

Dove $u(k) \in R$ e $y(k) \in R$ sono rispettivamente l'ingresso di controllo e l'output del sistema all'istante k; n_y e n_u sono due interi positivi mentre $f(\dots) : R^{n_y+n_u+2} \rightarrow R$ è una funzione non lineare non nota.

A questo punto, per poter procedere con la linearizzazione, è necessario effettuare delle assunzioni, grazie alle quali si può applicare il teorema fondamentale.

Assunzione 1: Le derivate parziali di $f(\dots)$ esistono e sono continue.

Assunzione 2: La funzione non lineare $f(\dots)$ soddisfa la *condizione di Lipschitz*:

$$\left|f\left(\boldsymbol{y}(k), \boldsymbol{u}(k)\right) - f(\boldsymbol{y}(k-1), \boldsymbol{u}(k-1))\right| \leq b\|\boldsymbol{u}(k) - \boldsymbol{u}(k-1)\|$$

per ogni condizione tale che $\|\boldsymbol{u}(k) - \boldsymbol{u}(k-1)\| \neq 0$. Si ha che b è una costante positiva mentre $\boldsymbol{u}$ è un vettore che varia per ogni forma di linearizzazione dinamica.

L'assunzione 2, la più importante, è quella che descrive il comportamento dell'output del sistema al variare dell'input di controllo. In particolare, questa condizione impone un *upper bound* sull'uscita, per far sì che essa sia limitata per variazioni limitate dell'ingresso.

Teorema 1: Si consideri un sistema non lineare di tipo (1) che soddisfa le assunzioni 1 e 2 con $\boldsymbol{u} = \boldsymbol{U}_L(k) = [u(k), \dots, u(k-L+1)]^T$. Allora, esiste un vettore tempo variante $\Phi_{p,L}(k) \in R^L$ chiamato *pseudo gradiente (PG)* tale che il sistema (1) può essere trasformato nel seguente modello PFDL:

$$\Delta y(k+1) = \varphi_{p,L}^T(k)\Delta \boldsymbol{U}_L(k) \qquad (2)$$

Dove L è un intero chiamato *control input linearization length constant (LLC)*; $\varphi_{p,L}(k) = \left[\varphi_1(k), \dots, \varphi_L(k)\right]^T$ e $\left\|\varphi_{p,L}(k)\right\| \leq b$ è limitato per ogni istante k.

Si può notare che al variare della costante LLC si possono ottenere diverse forme di DL.

3.2.2 CFDL

Ponendo $L = 1$, il modello PFDL diventerà di tipo CFDL. Ne segue la seguente variazione del Teorema 1:

Teorema 1: Considerando un sistema non lineare di tipo (1) che soddisfa le assunzioni 1 e 2, se $|\Delta u(k)| \neq 0$ allora esiste un parametro tempo-variante $\Phi_c(k) \in R$ chiamato *pseudo partial derivative (PPD)* tale che il sistema (1) può essere trasformato nel seguente modello CFDL:

$$\Delta y(k+1) = \varphi_c(k)\Delta u(k)$$

con $\varphi_c(k)$ limitato per ogni istante k.

PPD è un valore tempo-variante che si ottiene mediante una stima, poiché non è conosciuto a priori. Questo parametro serve per "catturare" tutte le possibili proprietà del sistema non lineare. In particolare, se il periodo di campionamento e $\Delta u(k) = u(k) - u(k-1)$ sono sufficientemente piccoli si ha che la variazione nel tempo di PPD sarà molto lenta. Per questo motivo ottenere una stima numerica di questo parametro non risulta eccessivamente complesso.

Per quanto riguarda il significato geometrico, PPD rappresenta la derivata della funzione in un punto compreso tra $u(k)$ e $u(k+1)$.

Se il comportamento di PPD risultasse troppo difficile da stimare, è consigliabile applicare i metodi PFDL o FFDL poiché il vettore PG presenta un comportamento più semplice rispetto al PPD, se si considera lo stesso problema di controllo.

3.2.3 FFDL

Ponendo $L = L_y + L_u$, con L_y e L_u due interi positivi chiamati *pseudo-ordini*, si ottiene il modello FFDL. Di seguito verrà analizzata nel dettaglio questa tecnica, che sarà quella utilizzata nel corso dell'elaborato per effettuare le simulazioni relative ad alcuni processi.

Consideriamo un sistema non lineare non noto a tempo discreto, single-input single-output (SISO):

$$y(k+1) = f\left(y(k), y(k-1), \dots, y\left(k-n_y\right), u(k), u(k-1), \dots, u(k-n_u)\right) \quad (1)$$

dove $u(k) \in R$ e $y(k) \in R$ sono rispettivamente l'ingresso di controllo e l'uscita del sistema all'istante k; $n_y, n_u \in Z_+$ sono gli ordini dell'uscita e dell'ingresso che non sono noti a priori, $f(\dots) : R^{n_y+n_u+2} \rightarrow R$ una funzione non lineare anch'essa non nota.

Anche in questo caso dobbiamo definire le due assunzioni fondamentali per procedere con la linearizzazione:

Assunzione 1: Le derivate parziali di $f(\dots)$ esistono e sono continue.

Assunzione 2: La funzione non lineare $f(\dots)$ soddisfa la *condizione di Lipschitz*:

$$|y(k_1+1) - y(k_2+1)| \leq b \left\| \boldsymbol{H}_{L_y,L_u}(k_1) - \boldsymbol{H}_{L_y,L_u}(k_2) \right\|$$

dove $\boldsymbol{H}_{L_y,L_u}(k) = [y(k), \dots, y(k - L_y + 1), u(k), \dots, u(k - L_u + 1)]^T \in R^{L_y+L_u}$ è un vettore che contiene gli input di controllo e gli output del sistema.

In particolare, i segnali presi in considerazione sono valutati su delle precise finestre temporali: per quanto riguarda gli ingressi, l'intervallo di tempo è $[k - L_u + 1, k]$, per le uscite invece si ha una finestra del tipo $[k - L_y + 1, k]$.

Come accennato, $L_y (1 \leq L_y \leq n_y)$ e $L_u (1 \leq L_u \leq n_u)$ sono chiamati *pseudo-ordini* del sistema.

Anche in questo caso, dopo aver specificato le assunzioni che devono essere soddisfatte, si procede enunciando il teorema principale.

In particolare, prima di procedere con l'enunciato, definiamo $\Delta \boldsymbol{H}_{L_y,L_u}(k) = \boldsymbol{H}_{L_y,L_u}(k) - \boldsymbol{H}_{L_y,L_u}(k-1)$.

Teorema 1: Si consideri un sistema non lineare di tipo (1) che soddisfa le assunzioni 1 e 2, per ogni $1 \leq L_y \leq n_y$ e $1 \leq L_u \leq n_u$ fissati e $\left\| \Delta \boldsymbol{H}_{L_y,L_u}(k) \right\| \neq 0$. Allora, esiste un vettore tempo-variante $\boldsymbol{\Phi}_{f,L_y,L_u}(k) \in R^{L_y+L_u}$, chiamato PG vector, tale che il sistema (1) può essere trasformato nel seguente modello FFDL:

$$\Delta y(k+1) = \boldsymbol{\Phi}^T_{f,L_y,L_u}(k) \Delta \boldsymbol{H}_{L_y,L_u}(k) \qquad (2)$$

con $\left\| \boldsymbol{\Phi}_{f,L_y,L_u}(k) \right\| \leq b$ per ogni istante k, dove $\boldsymbol{\Phi}_{f,L_y,L_u}(k) = [\Phi_1(k), \dots, \Phi_{L_y}(k), \Phi_{L_y+1}(k), \dots, \Phi_{L_y+L_u}(k)]^T$.

Anche in questo caso, il sistema (1) è esattamente equivalente al sistema (2), poiché nessuna informazione viene omessa nel processo di linearizzazione.

Dopo aver analizzato le tre tecniche di linearizzazione dinamica, si possono fare alcune considerazioni:

- il modello che si ottiene effettuando questo metodo di linearizzazione è equivalente a quello di partenza, infatti nessuna informazione utile viene omessa. Tutte le simulazioni e gli esperimenti svolti a partire da questa tecnica dimostrano l'equivalenza tra i due sistemi.

- CFDL viene utilizzata per sistemi non lineari più semplici, mentre i metodi FFDL e PFDL sono utilizzati per sistemi che presentano una maggiore complessità. In particolare, PFDL, come si può notare dal teorema 1, analizza la relazione che lega l'ingresso di controllo e l'output del sistema considerando una finestra temporale del tipo $[k - L + 1, k]$.
  Nel caso del modello FFDL, invece, si ha una finestra temporale del tipo $[k - L_y - L_u + 1, k]$.

- Per quanto riguarda il significato fisico, i parametri introdotti precedentemente caratteristici delle tre tecniche di linearizzazione, non assumono nessun significato, a differenza di altre tecniche di controllo in cui si prendono in considerazione, ad esempio, i poli e gli zeri, che invece sono molto importanti dal punto di vista fisico.

- Dopo aver analizzato tutti i vantaggi della DL, è opportuno analizzare anche i limiti di questa tecnica di linearizzazione. In particolare, utilizzando questi metodi si ottiene una rappresentazione del sistema che consiste in una descrizione del legame tra l'ingresso e l'uscita. Dal punto di vista fisico, non si può dedurre nessun significato concreto, inoltre, è una modalità utilizzata solo per sistemi a ciclo chiuso.

4 Model-Free Adaptive Control

4.1 Generalità

La tecnica *Model-Free Adaptive Control (MFAC)* viene utilizzata per i sistemi non lineari a tempo discreto. Questo metodo si basa sulla linearizzazione dinamica, infatti grazie alla DL è possibile ottenere un modello linearizzato del sistema che, come già detto in precedenza, è esattamente equivalente a quello di partenza e può essere utilizzato per definire la legge di controllo. Si tratta di una tecnica altamente conveniente sia per la sua versatilità, poiché può essere utilizzata in molteplici applicazioni, che per il suo basso costo computazionale.

Il principio di funzionamento non si basa su nessun modello noto del processo, bensì sfrutta le informazioni I\O relative agli istanti precedenti. Analizzando questi dati è possibile ricavare il controllore e garantire tutte le proprietà relative alla stabilità e alla robustezza.

Si consideri la funzione di costo relativa all'ingresso di controllo:

$$J\big(u(k)\big) = |y_d(k+1) - y(k+1)|^2 + \lambda |u(k) - u(k-1)|^2 \qquad (3)$$

dove $\lambda > 0$ è un fattore di peso, infatti al variare di questo parametro si avranno cambiamenti nella variazione dell'ingresso di controllo. Inoltre, è stata introdotta $y_d(k+1)$ che rappresenta l'uscita desiderata.

A questo punto è possibile sostituire l'equazione (2) relativa al sistema linearizzato nella funzione (3) appena definita, per poi minimizzare l'equazione ottenuta rispetto a $u(k)$. Si ottiene perciò il controllore:

$$\Delta u(k) = \qquad (4)$$

$$\begin{cases} \dfrac{\Phi_{L_y+1}(k)\left[\rho_{L_y+1}\big(y_d(k+1) - y(k)\big) - \sum_{i=1}^{L_y} \rho_i\, \Phi_i(k)\Delta y(k-i+1)\right]}{\lambda + \left|\, \Phi_{L_y+1}(k)\right|^2} \\ \dfrac{-\,\Phi_{L_y+1}(k) \sum_{i=L_y+2}^{L_y+L_u} \rho_i\, \Phi_i(k)\Delta u\big(k - L_y - i + 1\big)}{\lambda + \left|\, \Phi_{L_y+1}(k)\right|^2}, L_u \geq 2 \\ \dfrac{\Phi_{L_y+1}(k)\left[\rho_{L_y+1}\big(y_d(k+1) - y(k)\big) - \sum_{i=1}^{L_y} \rho_i\, \Phi_i(k)\Delta y(k-i+1)\right]}{\lambda + \left|\, \Phi_{L_y+1}(k)\right|^2}, L_u = 1 \end{cases}$$

con $\rho_i \in (0,1], i = 1, \dots, L_y + L_u$ chiamati *step factors*, ossia dei parametri utilizzati per rendere l'algoritmo ancora più adattabile ed efficace.

Come già accennato in precedenza, il funzionamento a livello pratico di questo algoritmo si basa sulla stima del vettore PG, il quale corrisponde ad una struttura tempo-variante. Si introduce perciò una nuova funzione di costo che permette di ottenere una stima di questo vettore a partire dalle informazioni di I\O.

$$J\left(\boldsymbol{\Phi}_{f,L_y,L_u}(k)\right) = \left|y(k) - y(k-1) - \boldsymbol{\Phi}^T_{f,L_y,L_u}(k)\Delta\, \boldsymbol{H}_{L_y,L_u}(k-1)\right|^2 + \mu \left\|\boldsymbol{\Phi}_{f,L_y,L_u}(k) - \widehat{\boldsymbol{\Phi}}_{f,L_y,L_u}(k-1)\right\|^2 \quad (5)$$

In questa funzione è stato aggiunto un nuovo parametro $\mu > 0$ con lo scopo di ponderare la differenza tra il vettore PG e la sua stima. $\widehat{\boldsymbol{\Phi}}_{f,L_y,L_u}(\dots) \in R^{L_y+L_u}$ è proprio la stima di $\boldsymbol{\Phi}_{f,L_y,L_u}(\dots)$.

A questo punto è possibile definire il vettore delle stime, considerando una minimizzazione della funzione J rispetto a $\boldsymbol{\Phi}_{f,L_y,L_u}(k)$.

Il vettore $\widehat{\boldsymbol{\Phi}}_{f,L_y,L_u}(k)$ assume la seguente forma:

$$\widehat{\boldsymbol{\Phi}}_{f,L_y,L_u}(k) = \widehat{\boldsymbol{\Phi}}_{f,L_y,L_u}(k-1) + \frac{\eta\Delta\, \boldsymbol{H}_{L_y,L_u}(k-1)\big(y(k) - y(k-1)\big)}{\mu + \left\|\Delta\, \boldsymbol{H}_{L_y,L_u}(k-1)\right\|^2} - \frac{\eta\Delta\, \boldsymbol{H}_{L_y,L_u}(k-1)\widehat{\boldsymbol{\Phi}}^T_{f,L_y,L_u}(k-1)\Delta\, \boldsymbol{H}_{L_y,L_u}(k-1)}{\mu + \left\|\Delta\, \boldsymbol{H}_{L_y,L_u}(k-1)\right\|^2} \quad (6)$$

in cui compare un nuovo parametro $\eta \in (0,2]$ e il vettore delle stime $\widehat{\boldsymbol{\Phi}}_{f,L_y,L_u}(k) = [\widehat{\boldsymbol{\Phi}}_1(k), \dots, \widehat{\boldsymbol{\Phi}}_{L_y}(k), \widehat{\boldsymbol{\Phi}}_{L_y+1}(k), \dots, \widehat{\boldsymbol{\Phi}}_{L_y+L_u}(k)]^T$.

$$\widehat{\boldsymbol{\Phi}}_{f,L_y,L_u}(k) = \widehat{\boldsymbol{\Phi}}_{f,L_y,L_u}(1) \text{ if } \left\|\widehat{\boldsymbol{\Phi}}_{f,L_y,L_u}(k)\right\| \leq \varepsilon$$

$$\text{or } \left\|\Delta \boldsymbol{H}_{L_y,L_u}(k-1)\right\| \leq \varepsilon$$

$$\text{or } sign(\widehat{\boldsymbol{\Phi}}_{f,L_y,L_u}(k)) \neq sign(\widehat{\boldsymbol{\Phi}}_{f,L_y,L_u}(1))$$

(7)

La costante ε è generalmente molto piccola e può assumere un valore arbitrario.

A questo punto, una volta ottenuta la stima del vettore PG, si procede semplificando l'equazione (4) riscrivendo il modello del controllore nel seguente modo:

$$\Delta u(k) = \begin{cases} \hat{\xi}_{L_y+1}(k)[\rho_{L_y+1}\left(y_d(k+1) - y(k)\right) - \sum_{i=1}^{L_y} \rho_i \widehat{\Phi}_i(k)\Delta y(k-i+1) \\ \qquad - \sum_{i=L_y+2}^{L_y+L_u} \rho_i \widehat{\Phi}_i(k)\,\Delta u\big(k-L_y-i+1\big)],\ L_u \geq 2 \\ \hat{\xi}_{L_y+1}(k)[\rho_{L_y+1}\left(y_d(k+1) - y(k)\right) - \sum_{i=1}^{L_y} \rho_i \widehat{\Phi}_i(k)\Delta y(k-i+1)\,, \\ \qquad Lu = 1 \end{cases}$$

(8)

dove $\hat{\xi}_{L_y+1}(k) = \frac{\Phi_{L_y+1}(k)}{\lambda + \left|\Phi_{L_y+1}(k)\right|^2} \in R$ viene utilizzato per ottenere una migliore leggibilità del sistema.

E' stato così ottenuto il modello MFAC basato sulla tecnica FFDL.

Gli pseudo-ordini L_y e L_u, se troppo piccoli, potrebbero causare problemi quando si effettua la stima del vettore PG perché si avrebbe un aumento della complessità dal punto di vista del comportamento del sistema; tuttavia, si

otterrebbe un miglioramento per quanto riguarda la complessità matematica. Di conseguenza, ciò che conviene fare è partire da valori di L_y e L_u sufficientemente piccoli per poi aumentarli in base al comportamento del sistema.

Inoltre, assumono una grande importanza anche tutti i parametri che sono stati introdotti finora, come λ, ρ, η, etc. La scelta di questi valori, infatti, è fondamentale poiché da essi dipende l'uscita del sistema controllato, che deve essere il più possibile fedele a quella desiderata.

5 Risultati preliminari

5.1 Generalità

Nel corso dell'elaborato sono state analizzate le tre tecniche di linearizzazione dinamica. In particolare, sono state brevemente descritte la forma compatta (CFDL) e quella parziale (PFDL), mentre la forma completa (FFDL) è stata esaminata nel dettaglio.

Di seguito sono presentati degli studi pratici relativi al modello FFDL-MFAC prendendo in considerazione come esempi due processi dall'articolo *"On Model-Free Adaptive Control and Its Stability Analysis" – Zhongsheng Hou, Shuangshuang Xiong.*

L'algoritmo dello schema di controllo FFDL-MFAC è stato implementato inizialmente in *Matlab*, un ambiente di programmazione utile per il calcolo numerico. Dopodichè è stato ulteriormente esaminato effettuando delle simulazioni in *Simulink.*

Simulink è un ambiente di programmazione i cui comandi sono associati a quelli di *Matlab*. Per accedervi è necessario scrivere sulla *command window* di *Matlab* il comando simulink.

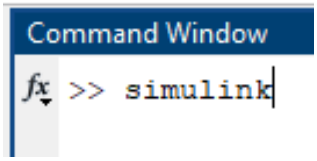

Fig. 5: Avvio Simulink

Viene principalmente utilizzato per effettuare delle simulazioni tramite la costruzione di schemi a blocchi relativi a sistemi dinamici lineari e non lineari.

Il vantaggio principale di *Simulink* è la facilità nel costruire lo schema a blocchi; infatti, attraverso un'interfaccia grafica si hanno a disposizione diversi blocchi che l'utente deve selezionare e collegare. Inoltre, vengono messe a disposizione numerose librerie che possono essere consultate

facilmente per trovare il blocco che si vuole utilizzare. Tutti i risultati ottenuti possono essere salvati nel workspace di *Matlab*.

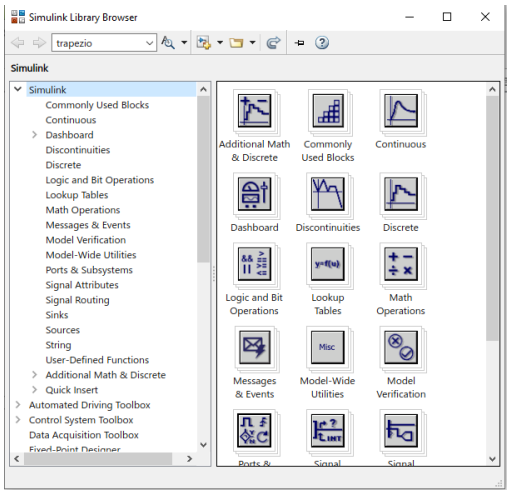

Fig. 6: Librerie Simulink

5.2 Esempio 1

Consideriamo il seguente processo lineare:

$$y(k+1) = \begin{cases} 2y(k) + u(k) + 3u(k-1), & 0 \le k < 400 \\ 1.67y(k) - 0.67y(k-1) + 0.2u(k) + 0.18u(k-1), & 400 \le k < 800 \end{cases}$$

L'uscita desiderata è così definita:

$$y_d(k+1) = \begin{cases} 1, & 0 \le k < 400 \\ 3, & 400 \le k < 800 \end{cases}$$

Di seguito è possibile consultare la tabella relativa ai parametri fondamentali.

Symbol	MFAC C2
L_y/n	$L_y = 1$
L_u/m	$L_u = 2$
λ	2.5
μ/c	$\mu = 1$
$\eta/a(t)$	$\eta = 0.3$
ρ	0.82
$\hat{\boldsymbol{\theta}}_1(k)$	Null
$\hat{\boldsymbol{\phi}}_{f,L_y,L_u}(1)$	[3.7,2,5]
$(u(0),u(1),u(2))$	(0,0,0)
$(y(0),y(1),y(2))$	(1,1,1)

Fig. 7: Tabella dei parametri esempio 1

Dalla Fig. 7 si possono ricavare informazioni riguardo gli pseudo-ordini, i vari fattori di peso e le condizioni iniziali.

5.2.1 Implementazione in Matlab

Il codice che implementa l'algoritmo dello schema di controllo per il processo preso in considerazione è riportato di seguito con le relative spiegazioni.

```
%Esempio1 FFDL-MFAC


time = (1:800);
Ly = 1;
Lu = 2;
L=Lu+Ly;
lambda = 2.5;
mu = 1;
etha = 0.3;
rho = 0.82;
phi_1= [3.7,2,5];
S_phi= zeros(L,800);
S_phi(:,3)= phi_1;
yd = zeros(1,800);
y = zeros(1,800);
u = zeros(1,800);
H = zeros(L,800);
err = yd - y;

x = [0,0];
deltaU = zeros(Lu,1);
deltaY = zeros(Ly,1);
deltaH = zeros(L,800);

for i = (1:3)
    u(i)= 0;
    y(i)= 1;
end

```

Fig. 8: Definizione parametri in Matlab esempio 1

Per iniziare, sono stati definiti i parametri con i relativi valori della Fig. 7. In secondo luogo sono state inizializzate tutte le strutture dati (array e matrici) che saranno poi definite in seguito dall'algoritmo di controllo. In particolare, sono stati inizializzati i vettori degli ingressi e delle uscite e la matrice H. A partire dalla riga 26 sono state definite le condizioni iniziali di $u(k)$ e di $y(k)$, anch'esse specificate nella tabella iniziale.

```
31 -  for k = (1:800)
32 -        if k<400
33 -            yd(k)= 1;
34 -        end
35 -        if k>=400
36 -            yd(k)= 3;
37 -        end
38
39 -  end
40
```

Fig. 9: Definizione uscita desiderata in Matlab esempio 1

In questa parte, tramite un semplice ciclo *for*, è stata definita l'uscita desiderata. Quest'ultima servirà per effettuare un confronto con l'uscita effettiva del sistema in modo da assicurare un corretto funzionamento del sistema di controllo.

```
41 - for k = (4:799)
42 -     Y = y(k);
43 -     U = [u(k);u(k-1)];
44 -     H(:,k) = [Y;U];
45
46 -     deltaH(:,k) = H(:,k)- H(:,k-1);
47
48 -     S_phi(:,k)= S_phi(:,k-1)+(etha*deltaH(:,k-1))*(y(k)-y(k-1))/(mu+norm(deltaH(:,k-1))^2)
49 -     -((etha*deltaH(:,k-1))*S_phi(:,k-1).'*deltaH(:,k-1))/(mu+norm(deltaH(:,k-1))^2);
50
51 -     Xi = (S_phi(2,k))/(lambda + abs(S_phi(2,k))^2);
52
53 -     u(k) = u(k-1)+ Xi*(rho*(yd(k+1)-y(k))- (rho*S_phi(1,k)*deltaH(1,k))- (rho*S_phi(3,k)*deltaH(3,k)));
54
55 -     if k<400
56 -        y(k+1) = 2*y(k) + u(k) + 3*u(k-1);
57 -     end
58 -     if k>=400
59 -         y(k+1) = 1.67*y(k) - 0.67*y(k-1) + 0.2*u(k) + 0.18*u(k-1);
60 -     end
61 - end
62
```

Fig. 10: Algoritmo di controllo FFDL-MFAC e defizione processo in Matlab esempio 1

A questo punto troviamo l'implementazione vera e propria dello schema di controllo, anch'esso all'interno di un ciclo *for*, che esclude i primi tre istanti già definiti dalle condizioni iniziali nel codice riportato in precedenza. In questa parte viene dichiarata la matrice H, precedentemente inizializzata a

zero, e la sua variazione che servirà in seguito per l'algoritmo di controllo. Successivamente viene definita la matrice contenente le stime del vettore PG e, alla riga 53, la legge di controllo $u(k)$. Come ultimo passaggio, prima di chiudere il ciclo, è stato rappresentato il processo il cui ingresso è proprio l'input di controllo trovato.

```
63 -    figure;
64 -    plot(time,yd,'b');
65 -    hold on
66 -    plot(time,y,'g');
67 -    grid
68 -    legend('Uscita desiderata','Uscita MFAC C2')
69
```

Fig. 11: Costruzione grafici in Matlab esempio 1

Infine, ricaviamo i due grafici che serviranno per effettuare il confronto tra l'andamento dell'uscita desiderata e quello dell'uscita effettiva del sistema. Per ottenere i grafici in *Matlab* esistono dei comandi appositi:

- *figure:* con questo comando viene creata una nuova finestra per il tracciamento di un grafico.
- *plot(X,Y,S):* questo comando permette di tracciare dei grafici utilizzando svariati simboli e colori. Nel codice presentato, come primo argomento viene inserito l'asse delle ascisse, come secondo argomento l'asse delle ordinate, mentre come terzo argomento viene aggiunto il colore.
- *hold on:* attraverso questo comando si richiede di mantenere il grafico appena tracciato per poterlo confrontare con un altro che sarà creato successivamente.
- *grid:* si aggiunge la griglia al grafico.
- *legend:* permette di aggiungere una legenda alla figura. In questo caso sono state aggiunte due stringhe che definiscono i nomi dei due grafici tracciati.

A questo punto, mandando in esecuzione il codice, è possibile ottenere le figure desiderate. Tramite il comando *hold on* i grafici sono stati sovrapposti in modo tale da ottenere un confronto ancora più evidente.

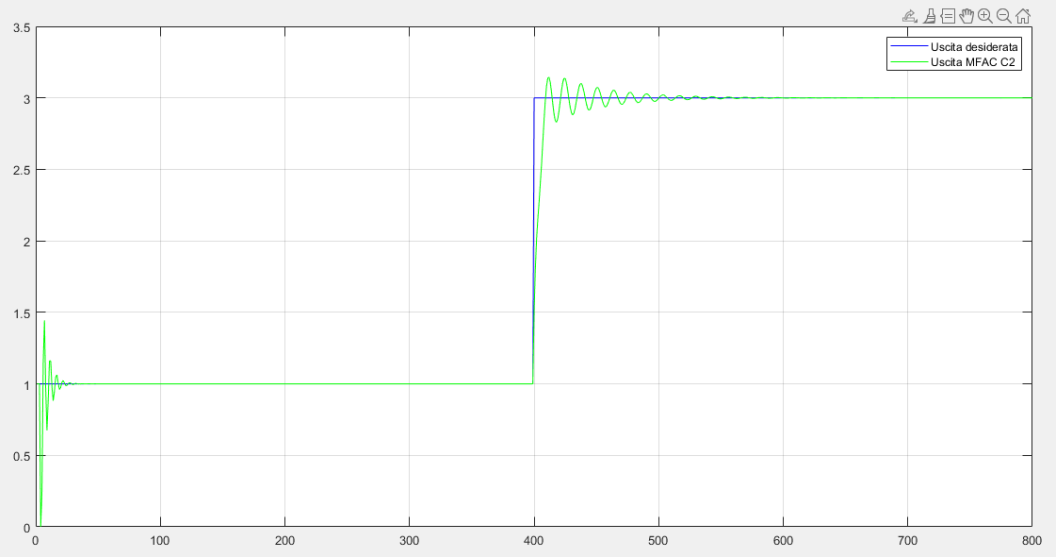

Fig. 12: Uscita desiderata (blu) e uscita effettiva (verde) esempio 1

Dalla Fig. 6 è possibile analizzare il comportamento del sistema controllato, considerando che l'uscita desiderata è stata tracciata in blu mentre quella effettiva è verde. Come si evince dal grafico, l'uscita del sistema presenta delle oscillazioni quando si verificano cambiamenti bruschi della curva; tuttavia, il comportamento che assume negli 800 campioni presi in considerazione è accettabile.

5.2.2 Implementazione in Simulink

Utilizzando *Simulink* per simulare il comportamento del sistema, come primo passo si devono definire tutti i valori della Fig. 7 relativi ai parametri caratteristici in un file *.m* che, una volta mandato in esecuzione, avvierà la simulazione.

Il comando *sim('MODEL', PARAMETERS)* serve per effettuare questo passaggio e riceve come argomenti il nome del modello da simulare e il numero di parametri da considerare.

```
1       %parametri
2 -     lambda = 2.5;
3 -     mu = 1;
4 -     etha = 0.3;
5 -     rho = 0.82;
6 -     epsi = 10^-5;
7 -     phi1 = 3.7;
8 -     phi2 = 2;
9 -     phi3 = 5;
10
11
12      %simulazione
13 -    sim('simContrEs1',800)
```

Fig. 13: Definizione parametri per l'avvio della simulazione in Simulink esempio 1

A questo punto, partendo dall'uscita desiderata e considerando lo schema di controllo FFDL-MFAC, il modello ottenuto è il seguente:

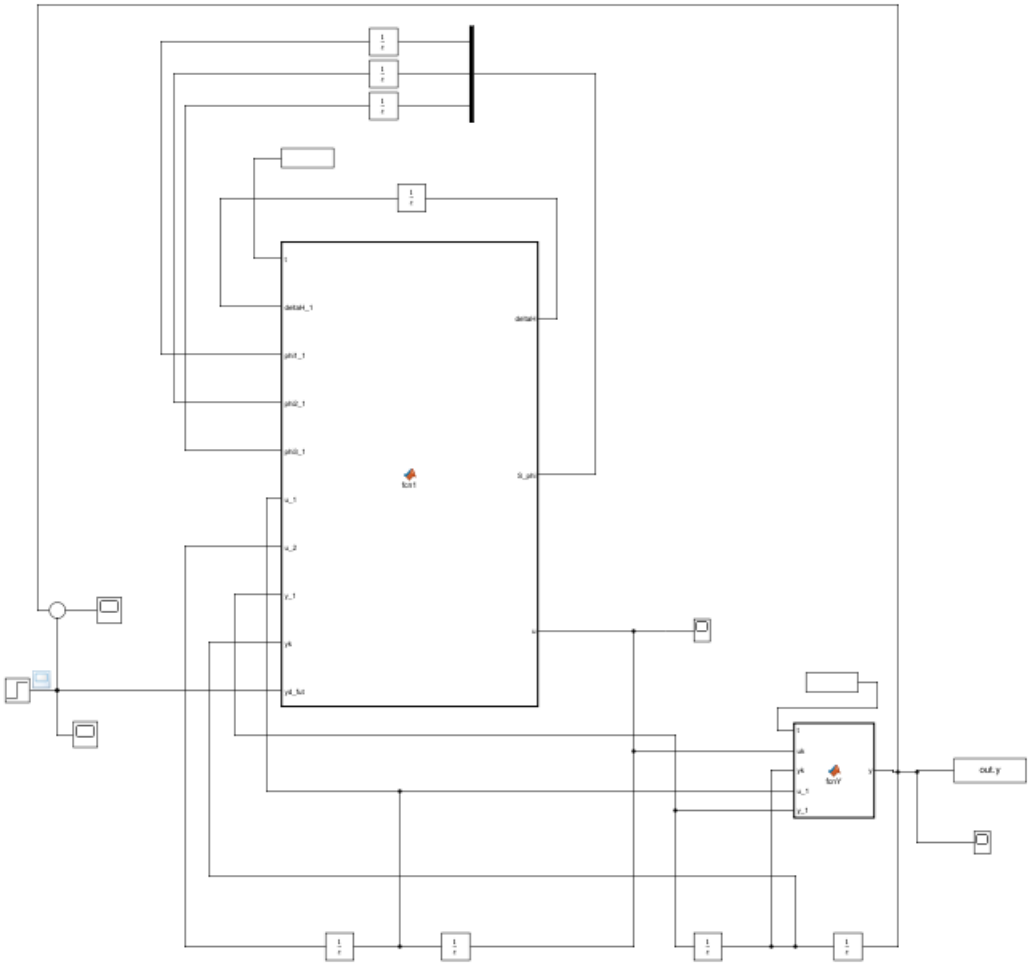

Fig. 14: Schema a blocchi in Simulink esempio 1

Di seguito sono analizzati i singoli componenti nel dettaglio:

- Il primo elemento che deve essere definito è l'uscita desiderata, che viene inserita nella simulazione tramite un blocco *step* (a sinistra).

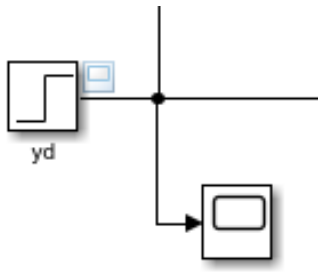

Fig. 15: Blocchi "Step" e "Scope"

Il blocco a destra, invece, è chiamato *scope* e serve per visualizzare il grafico del segnale a cui è collegato.

- Il blocco centrale rappresenta una *Matlab Function*, infatti tramite questo si implementa l'algoritmo di controllo.

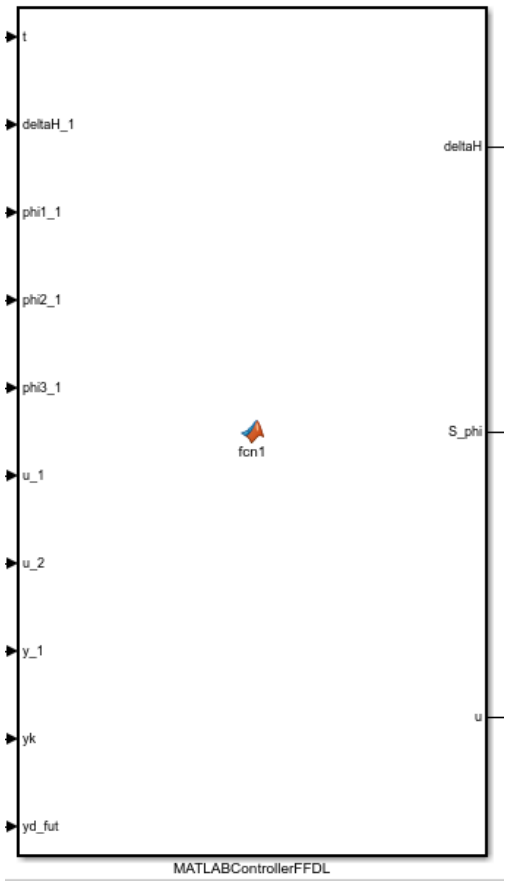

Fig. 16: Blocco "Matlab Function" per l'algoritmo di controllo FFDL-MFAC esempio 1

Si possono osservare tutte le variabili che corrispondono agli ingressi del blocco e che si riferiscono agli argomenti della funzione. Inoltre, le frecce uscenti rappresentano gli output.

In questo caso gli output della funzione sono l'ingresso di controllo, la stima del vettore PG e la variazione della matrice *H*.

Di seguito è riportato il codice, che può essere visualizzato facendo doppio click sul blocco tramite l'interfaccia di *Simulink*.

```
function [deltaH,S_phi,u] = fcn1(t,deltaH_1,phi1_1,phi2_1,phi3_1,u_1,u_2, ...
                y_1,yk,yd_fut,lambda, mu, etha, rho,epsi, phi1, phi2,phi3)

 deltaY = yk-y_1;
 S_phi_1 = [phi1_1;phi2_1;phi3_1];
 S_phi = S_phi_1+(etha*deltaH_1)*(deltaY)/(mu+norm(deltaH_1)^2)
             -((etha*deltaH_1)*S_phi_1.'*deltaH_1)/(mu+norm(deltaH_1)^2);
if norm(S_phi)<=epsi || norm(deltaH_1)<=epsi || sign(S_phi(1))~= sign(phi1)
    S_phi(1) = phi1;
    S_phi(2) = phi2;
    S_phi(3) = phi3;
end

 Xi = (S_phi(2))/(lambda + abs(S_phi(2)^2));

 if t<=3
     u=0;
 else
    u = u_1+ Xi*(rho*(yd_fut-yk)-(rho*S_phi(1)*deltaY)-(rho*S_phi(3)*(u_1-u_2)));
 end

 H = [yk;u;u_1];
 H_1= [y_1;u_1;u_2];
 deltaH = H - H_1;
```

Fig. 17: Algoritmo di controllo FFDL-MFAC in Simulink esempio 1

In queste righe di codice sono definite le condizioni iniziali che servono per implementare l'algoritmo. Le uscite della funzione vengono retroazionate e, dopo averle sottoposte a un ritardo, vengono inserite come input della funzione stessa.

- Il blocco che ci permette di introdurre un ritardo si chiama *Unit Delay*.

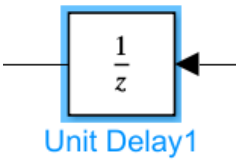

Fig. 18: Blocco "Unit Delay"

Questo blocco prende in ingresso un segnale e ha come uscita lo stesso segnale ritardato di un istante di tempo.

Ad esempio, nello schema a blocchi della Fig.19, un segnale che è stato sottoposto a un ritardo è proprio l'ingresso di controllo. Il blocco *Unit Delay* riceverà come ingresso $u(k)$ e avrà come uscita $u(k-1)$.
Per ottenere, ad esempio, il segnale $u(k-2)$ basterà inserire due blocchi *Unit Delay* in successione.

- Nello schema a blocchi è necessario aggiungere il processo che sappiamo essere dipendente dalle uscite e dagli ingressi agli istanti precedenti.
  Questi segnali sono gli input di un nuovo blocco che rappresenta un'altra *Matlab Function*, inserita proprio per descrivere il processo controllato.

  Graficamente, il blocco relativo al processo si presenta nel seguente modo:

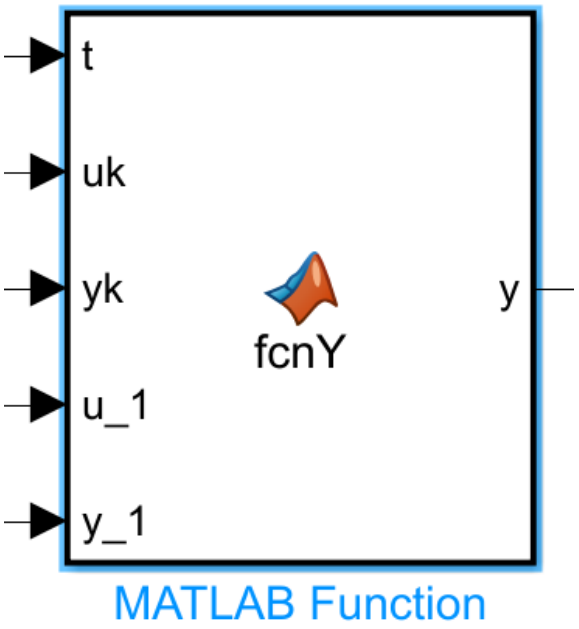

Fig. 19: Blocco "Matlab Function" per la definizione del processo esempio 1

Gli argomenti della funzione sono proprio i segnali sopraccitati, mentre l'output è l'uscita del sistema controllato.

Tra gli argomenti relativi agli ingressi è stato aggiunto *t* che definisce gli istanti di tempo. In particolare, per definire questa variabile, è necessario aggiungere un blocco chiamato *Digital Clock* che specifica il passo di campionamento.

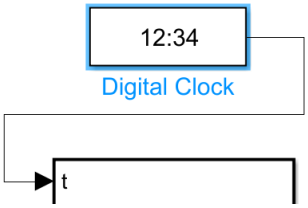

Fig. 20: Blocco "Digital Clock"

Di seguito è riportato il codice relativo alla *Matlab Function* che descrive il processo controllato.

```
function y = fcnY(t,uk,yk,u_1,y_1)

if (t<=3)
    y=1;
else if (t>3) & (t<400)
    y = 2*yk + uk + 3*u_1;
else
    y = 1.67*yk - 0.67*y_1 + 0.2*uk + 0.18*u_1;
    end

end
```

Fig. 21: Definizione processo in Simulink esempio 1

A questo punto, sfruttando gli *scope* che sono stati inseriti nello schema a blocchi, è possibile visualizzare tutti i grafici relativi ai vari segnali per poter valutare l'efficacia del sistema di controllo. La prima cosa da fare è mandare in esecuzione il file *.m* che contiene la definizione di tutti i parametri fondamentali dell'algoritmo e poi cliccare sui vari *scope* su *Simulink* per osservare l'andamento dei grafici.

La prima figura analizzata è quella relativa all'uscita desiderata $y_d(k+1)$:

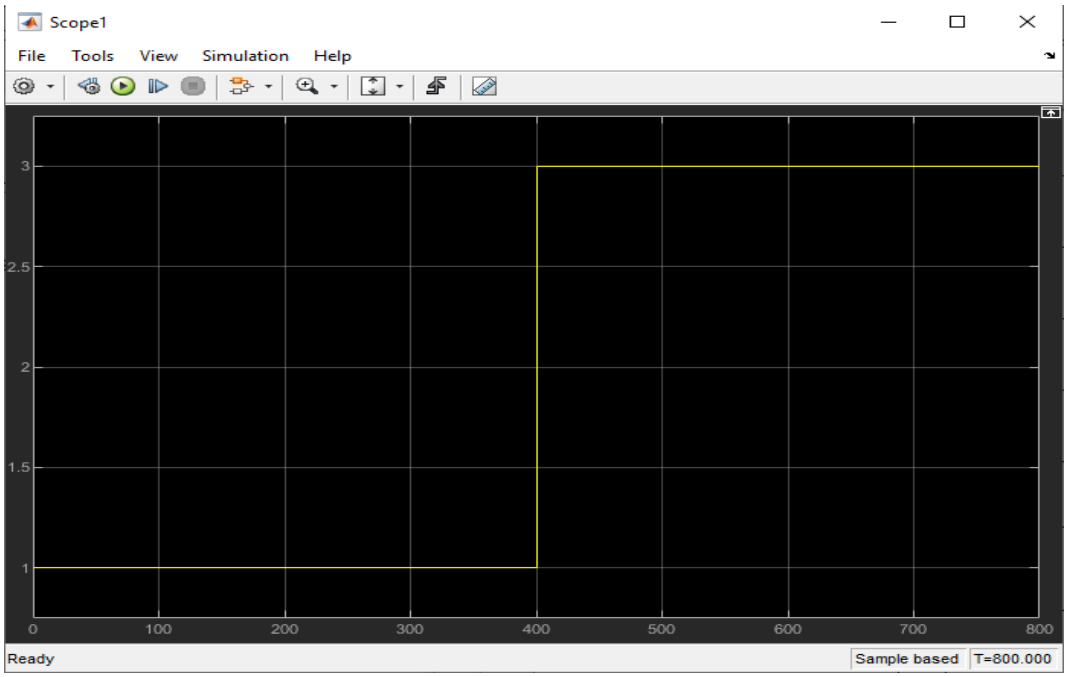

Fig. 22: Uscita desiderata esempio 1

Una volta analizzato l'andamento dell'uscita desiderata, visualizziamo quello relativo all'uscita effettiva del sistema $y(k)$.

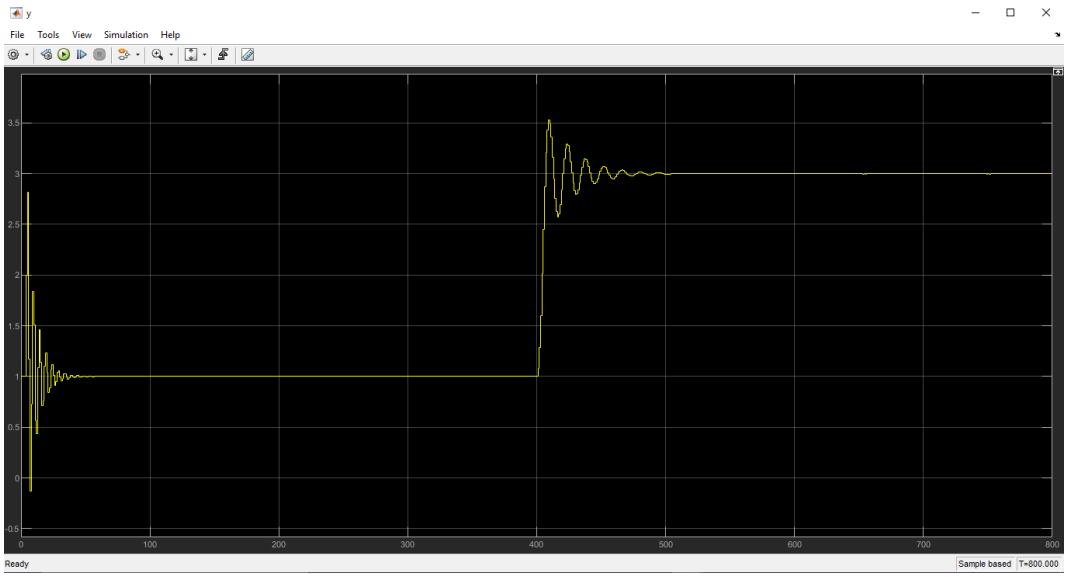

Fig. 23: Uscita effettiva esempio 1

Per una corretta valutazione del funzionamento, è stata aggiunta nello schema a blocchi la rappresentazione della variabile errore, ossia la differenza tra

l’uscita desiderata e quella effettiva. Questa variabile deve essere molto piccola in modo da assicurare un corretto funzionamento dello schema di controllo.

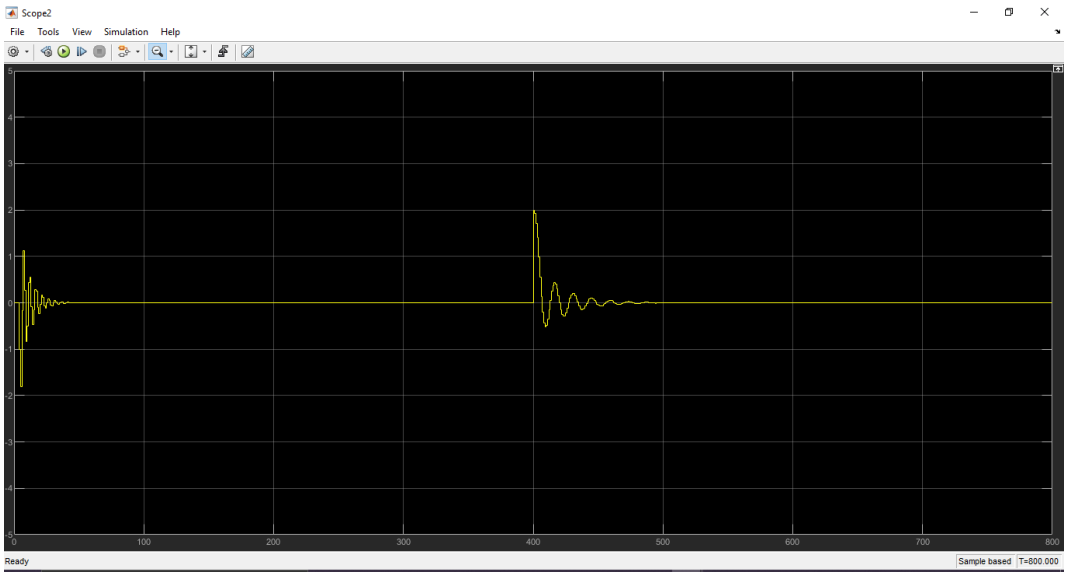

Fig. 24: Errore esempio 1

Si può notare come le oscillazioni maggiori si trovino nei punti relativi ai cambiamenti più bruschi della curva; tuttavia, nel resto dei campioni esaminati l’errore è molto piccolo. Da questo deduciamo che l’andamento è accettabile.

Come ultimo segnale, esaminiamo quello dell’ingresso di controllo $u(k)$:

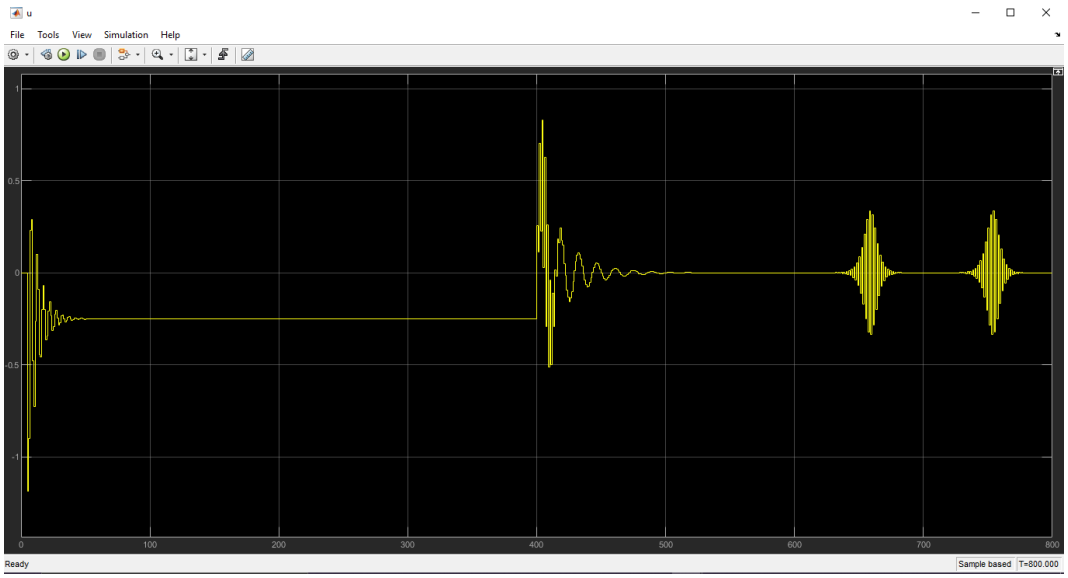

Fig. 25: Ingresso di controllo esempio 1

5.3 Esempio 2

Per verificare ancora più dettagliatamente il funzionamento dell'algoritmo, consideriamo ora il seguente processo, questa volta non lineare:

$$y(k+1) = 0.6y(k) - 0.1y(k-1) + 1.8u(k) - 1.8u^2(k) + 0.6u^3(k)$$
$$-0.15u(k-1) + 0.15u^2(k-1) - 0.05u^3(k-1)$$

Definiamo di seguito l'uscita desiderata:

$$y_d(k+1) = \begin{cases} 0.5, & 0 \le k < 100 \\ 1.0, & 100 \le k < 200 \\ 2.0, & 200 \le k < 300 \\ 1.5, & 300 \le k < 400 \end{cases}$$

Anche in questo caso sono già stati definiti i parametri caratteristici dell'algoritmo e sono riportati nella figura seguente:

	MFAC
Orders	$L_y = 1, L_u = 1$ $\hat{\boldsymbol{\phi}}_{f,1,1}(1) = [0.4, 0.4]$
Initial values	$y(0) = y(1) = 0$ $u(0) = u(1) = 0.3$
Parameters	Weighting factors: $\lambda = 1, \mu = 0.4$ Step factors: $\rho = 1, \eta = 1$

Fig. 26: Tabella dei parametri esempio 2

Per quanto riguarda l'algoritmo, l'implementazione è identica a quella già presentata in precedenza relativamente al primo esempio.

5.3.1 Implementazione in Matlab

Di seguito è riportato il codice in *Matlab* con i giusti valori dei parametri:

```
%Esempio2 FFDL-MFAC


time = (1:400);
Ly = 1;
Lu = 1;
L=Lu+Ly;
lambda = 1;
mu = 0.4;
etha = 1;
rho = 1;
phi_1= [0.4,0.4];
S_phi= zeros(L,400);
S_phi(:,2)= phi_1;
yd = zeros(1,400);
y = zeros(1,400);
u = zeros(1,400);
H = zeros(L,400);
err = yd - y;

x = [0,0];
deltaU = zeros(Lu,1);
deltaY = zeros(Ly,1);
deltaH = zeros(L,400);

for i = (1:2)
    u(i)= 0;
    y(i)= 0.3;
end

```

Fig. 27: Definizione parametri in Matlab esempio 2

Anche in questo caso sono stati assegnati tutti i valori dei parametri e inizializzate a zero le varie strutture che saranno definite dallo schema di controllo.

```
for i=1:400
    if i<100
        yd(i)=0.5;
    end
    if (i>=100)&(i<200)
        yd(i)=1.0;
    end
    if (i>=200)&(i<300)
        yd(i)=2.0;
    end
    if (i>=300)&(i<400)
        yd(i)=1.5;
    end
end

```

Fig. 28: Definizione uscita desiderata in Matlab esempio 2

In questo ciclo *for* è stata definita l'uscita desiderata che, tramite degli *if* interni al ciclo, assegna il valore alla funzione per un totale di 400 campioni.

```
for k = (3:399)
    Y = y(k);
    U = u(k);
    H(:,k) = [Y;U];

    deltaH(:,k) = H(:,k)- H(:,k-1);

    S_phi(:,k)= S_phi(:,k-1)+(etha*deltaH(:,k-1))*(y(k)-y(k-1))/(mu+norm(deltaH(:,k-1))^2)
            -((etha*deltaH(:,k-1))*S_phi(:,k-1).'*deltaH(:,k-1))/(mu+norm(deltaH(:,k-1))^2);
    Xi = (S_phi(2,k))/(lambda + abs(S_phi(2,k))^2);

    u(k) = u(k-1)+ Xi*(rho*(yd(k+1)-y(k))- (rho*S_phi(1,k)*deltaH(1,k)));

    y(k+1)=0.6*y(k)-0.1*y(k-1)+1.8*u(k)-1.8*(u(k))^2+0.6*(u(k))^3
                        -0.15*u(k-1)+0.15*(u(k-1))^2-0.05*(u(k-1))^3;
end

figure;
plot(time,yd,'b');
hold on
plot(time,y,'g');
grid
legend('Uscita desiderata','Uscita MFAC C2')
```

Fig. 29: Algoritmo di controllo, definizione processo e costruzione grafici in Matlab esempio 2

Lo schema di controllo è identico a quello analizzato per il primo esempio; ciò che cambia è il processo controllato.

Con gli stessi comandi già descritti precedentemente, tracciamo i due grafici relativi all'uscita desiderata (tracciata in blu) e all'uscita effettiva del sistema controllato (tracciata in verde).

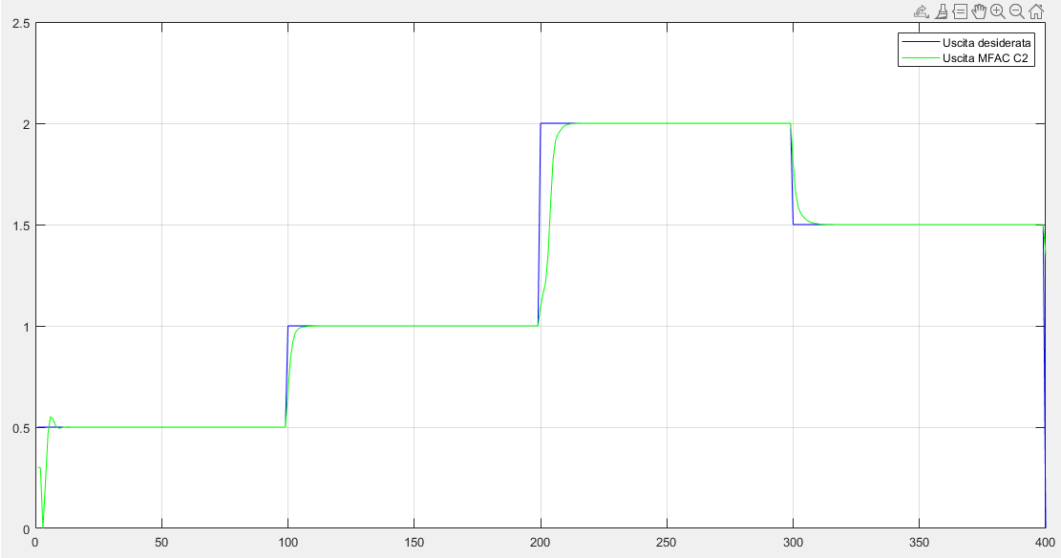

Fig. 30: Uscita desiderata (blu) e uscita effettiva (verde) esempio 2

Si deduce che l'uscita del sistema è accettabile poiché segue abbastanza fedelmente quella desiderata; perciò, si può confermare l'efficienza dell'algoritmo FFDL-MFAC.

5.3.2 Implementazione in Simulink

Anche per il secondo processo preso in analisi è stata effettuata una simulazione in *Simulink* e, come nel caso precedente, è stato creato un file contenente tutti i parametri dell'algoritmo con i relativi valori assegnati.

```
%parametri
lambda = 1;
mu = 0.4;
etha = 1;
rho = 1;
epsi= 10^-5;
phi1=0.4;
phi2=0.4;

%simulazione
sim('simContrEs2',400)
```

Fig. 31: Definizione parametri per l'avvio della simulazione in Simulink esempio 2

Di seguito è riportato lo schema a blocchi relativo al secondo processo esaminato, che sarà analogo a quello dell'Esempio 1.

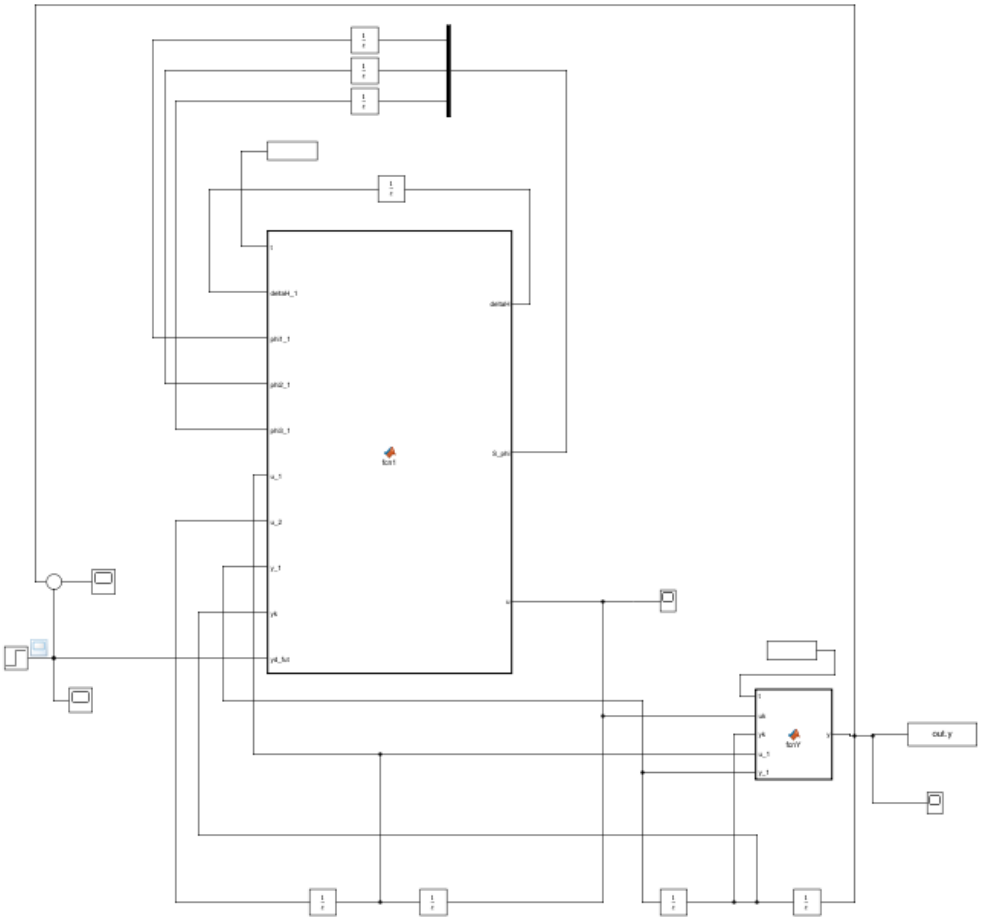

Fig. 32: Schema a blocchi in Simulink esempio 2

I blocchi utilizzati sono gli stessi che sono stati analizzati in precedenza poiché lo schema di controllo utilizzato è il medesimo. Tuttavia, sono state effettuate delle modifiche per quanto riguarda l'andamento dell'uscita desiderata e i codici relativi ai blocchi *Matlab Function*.

Per l'uscita desiderata è stato utilizzato un blocco diverso dal precedente, poiché i due segnali assumono andamenti diversi.

In particolare, il blocco utilizzato si chiama *Repeating Sequence Stair* e si utilizza per definire una sequenza di gradini.

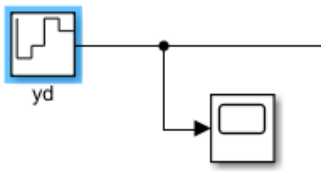

Fig. 33: Blocchi “Repeating Sequence Stair” e “Scope”

Gli altri blocchi sono gli stessi dell’esempio precedente, per cui le spiegazioni possono essere consultate nel sottocapitolo relativo all’Esempio 1.

Di seguito sono riportati i codici dei due blocchi *Matlab Function*, che riportano rispettivamente l’algoritmo di controllo e la definizione del processo controllato.

```
function [deltaH,S_phi,u] = fcn(t,deltaH_1,y_1,yk,yd_fut,u_1,phi1_1,phi2_1,
                                lambda, mu, etha, rho, phi1, phi2, epsi)



 deltaY = yk-y_1;
 S_phi_1 = [phi1_1;phi2_1];
 S_phi = S_phi_1+(etha*deltaH_1)*(deltaY)/(mu+norm(deltaH_1)^2)
                -((etha*deltaH_1)*S_phi_1.'*deltaH_1)/(mu+norm(deltaH_1)^2);
if norm(S_phi)<=epsi || norm(deltaH_1)<=epsi || sign(S_phi(1))~= sign(phi1)
    S_phi(1) = phi1;
    S_phi(2) = phi2;
end

 Xi = (S_phi(2))/(lambda + abs(S_phi(2)^2));

 if t<=2
     u=0.3;
 else
     u = u_1+ Xi*(rho*(yd_fut-yk)-(rho*S_phi(1)*deltaY));
 end


 H = [yk;u];
 H_1= [y_1;u_1];
 deltaH = H - H_1;
```

Fig. 34: Algoritmo di controllo FFDL-MFAC in Simulink esempio 2

```
function y = fcn1(t,yk,y_1,u,u_1)

        if t<=2
            y=0;

        else

        y = 0.6*yk-0.1*y_1+1.8*u-1.8*((u)^2)+0.6*((u)^3)
                    -0.15*u_1+0.15*((u_1)^2)-0.05*((u_1)^3);
        end
```

Fig. 35: Definizione processo in Simulink esempio 2

Anche in questo caso è stata aggiunta una variabile *t* definita da un clock per specificare il passo di campionamento.

A questo punto si può avviare la simulazione eseguendo il file contenente i parametri assegnati. Dopodiché, sarà possibile visualizzare tramite gli *scope* i grafici per esaminare il funzionamento del sistema di controllo.

Di seguito sono riportati quattro grafici: l'uscita desiderata $y_d(k+1)$, l'uscita effettiva del sistema $y(k)$, la variabile errore che servirà per analizzare l'efficacia dell'algoritmo e, infine, l'andamento dello sforzo di controllo $u(k)$.

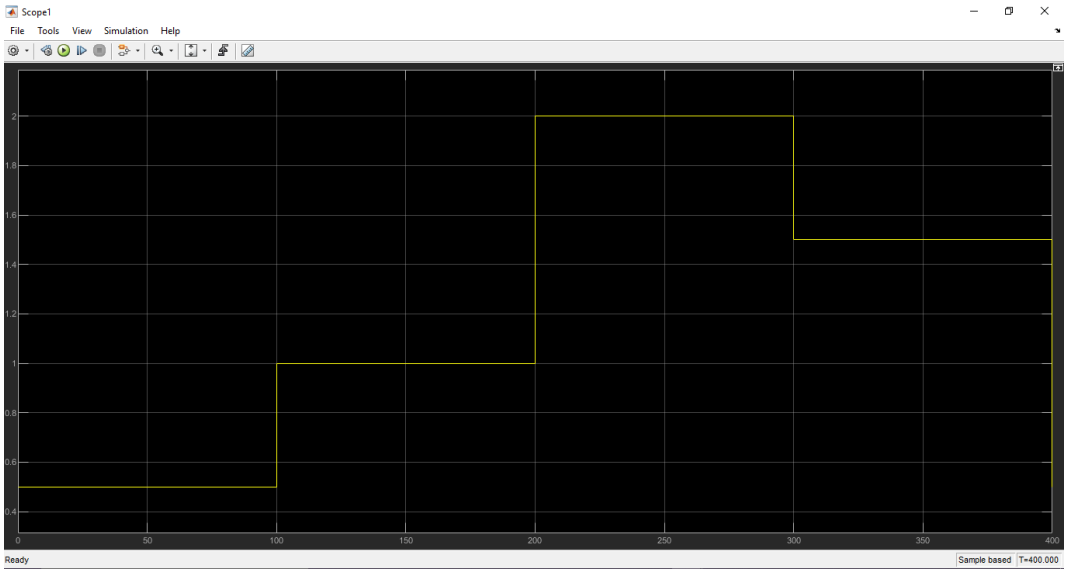

Fig. 36: Uscita desiderata esempio 2

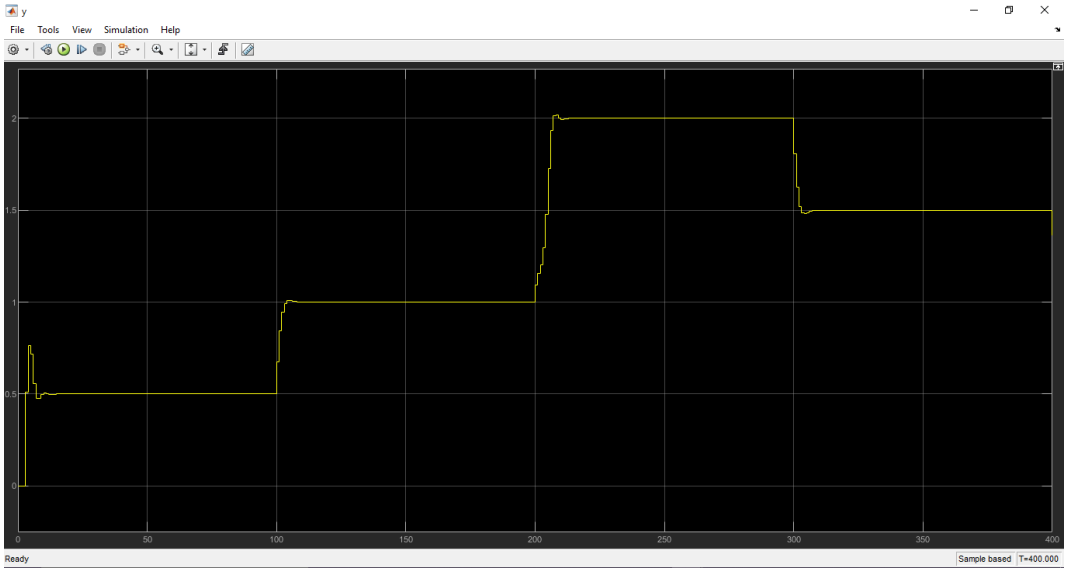

Fig. 37: Uscita effettiva esempio 2

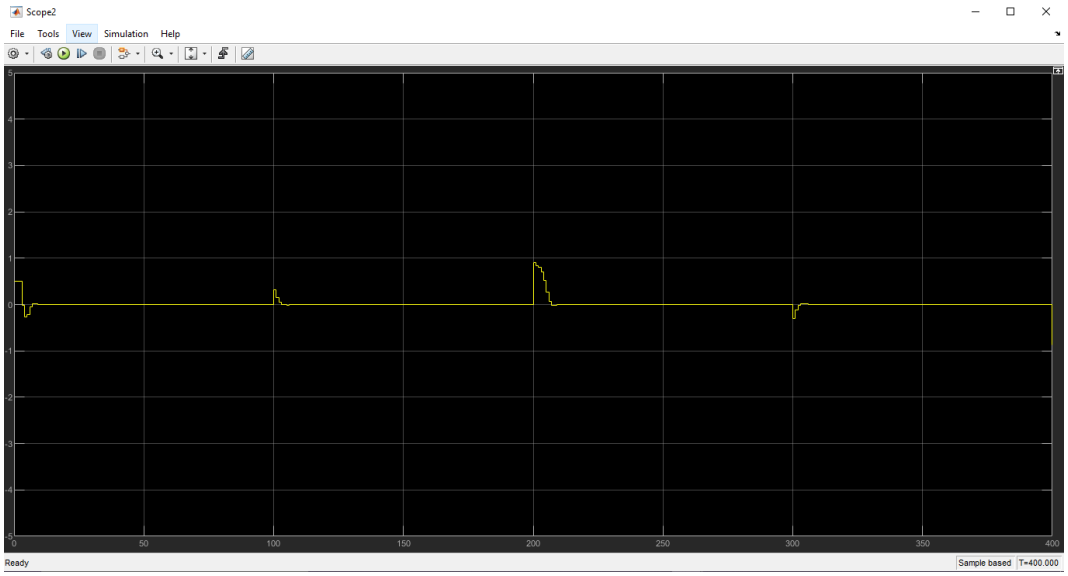

Fig. 38: Errore esempio 2

Da questo grafico è possibile notare come la differenza tra l'uscita desiderata e quella effettiva sia molto piccola. Le uniche oscillazioni evidenti, come nell'esempio precedente, sono relative ai punti in cui la curva varia bruscamente.

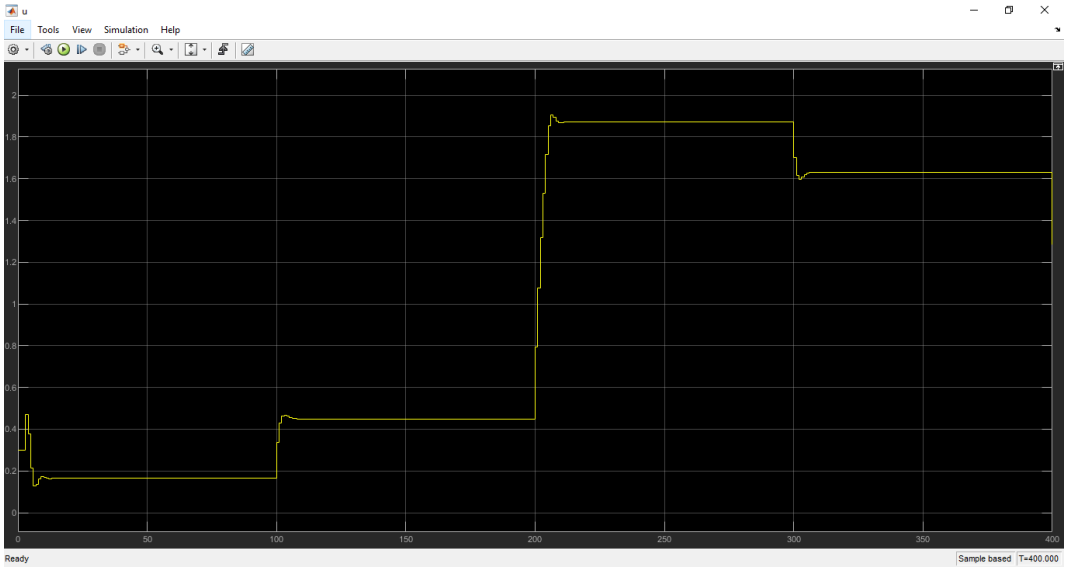

Fig. 39: Ingresso di controllo esempio 2

6 Motori sincroni a magneti permanenti

6.1 Generalità

Il motore sincrono a magneti permanenti (PMSM – Permanent Magnet Syncronous Motor) è un tipo di motore che sta ricoprendo un ruolo sempre più rilevante per quanto riguarda il controllo del movimento. Questa tipologia di motore è utilizzata per piccole e medie potenze, infatti presenta una piccola inerzia e una banda passante elevata. Inoltre, è molto impiegato a livello industriale ed è parecchio costoso a causa delle performance molto efficienti.

L'impiego di questo motore ha portato numerosi vantaggi a differenza del motore a collettore; infatti, le caratteristiche principali sono l'assenza di scintillio dovuto a collettore-spazzole e la semplificazione dei problemi di raffreddamento, poiché in questo tipo di motore gli avvolgimenti si trovano soltanto nello statore. Infatti, il motore sincrono a magneti permanenti è costituito dagli avvolgimenti di armatura che si trovano nello statore e dal flusso di eccitazione generato da magneti permanenti, i quali si trovano nel rotore.

Fig.40: Motore Sincrono a Magneti Permanenti

Si avrà perciò che l'**induttore** è rotante ed è costituito dal magnete, mentre l'**indotto** è fisso sullo statore.

In particolare, le modalità costruttive di questo tipo di motore prevedono:

- uno **statore** di forma cilindrica composto da materiale ferromagnetico e costituito da denti e cave, le quali rappresentano le sedi degli avvolgimenti di armatura.
- un **rotore**, anch'esso di forma cilindrica, in cui sono disposti i magneti permanenti in modo da ottenere un'alternanza di poli.

Il rotore si trova all'interno dello statore in modo da ottenere una fessura d'aria molto piccola, generalmente di qualche millimetro, chiamata **traferro** $\boldsymbol{\delta}$.

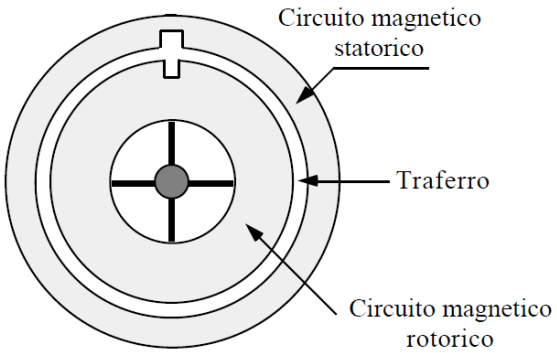

Fig. 41: Traferro

Consideriamo le equazioni elettriche che descrivono la dinamica del motore sincrono a magneti permanenti prendendo in considerazione un sistema di riferimento *(d,q)*:

$$\frac{di_d}{dt} = -\frac{R}{L}i_d + \omega_e i_q + \frac{1}{L}u_d \qquad (1)$$

$$\frac{di_q}{dt} = -\frac{R}{L}i_q + \omega_e i_q - \frac{1}{L}\lambda_0\omega_e + \frac{1}{L}u_q \qquad (2)$$

con i_d e i_q che rappresentano le correnti statoriche relative rispettivamente all'asse d e all'asse q. Per quanto riguarda u_d e u_q, esse rappresentano le tensioni statoriche, anche in questo caso in relazione agli assi d e q del sistema di riferimento. R è la resistenza degli avvolgimenti dello statore

mentre L rappresenta l'induttanza. λ_0 è il collegamento del flusso generato dal magnete permanente mentre ω_e è la velocità angolare elettrica riferita al rotore, che sappiamo essere rotante in questo tipo di motore.

Si possono definire perciò la coppia elettrica τ_e e la potenza meccanica P nel seguente modo:

$$\tau_e = K_t i_q; \qquad P = \tau_e \omega_r \tag{3}$$

in cui K_t è la costante di coppia che può essere definita come $K_t = \frac{3}{2}\lambda_0 N_r$ con N_r che rappresenta il numero di coppie polari. Infine, ω_r è la velocità angolare meccanica del rotore. Un'ipotesi che viene effettuata per semplificare la rappresentazione del motore, è l'assenza della coppia di riluttanza, per questo motivo si ha una proporzionalità che lega la coppia sviluppata e la corrente i_q.

A questo punto è possibile definire le equazioni che descrivono il movimento meccanico del motore sincrono a magneti permanenti:

$$J\frac{d\omega_r}{dt} + B\omega_r = \tau_e - \tau_l \tag{4}$$

$$\frac{d\theta_r}{dt} = \omega_r \tag{5}$$

nelle quali sono definite l'inerzia meccanica J del motore e del carico, il coefficiente di attrito viscoso B, la coppia di carico τ_l e la posizione angolare meccanica θ_r riferita al rotore del motore.

La posizione e la velocità angolari dal punto di vista meccanico sono legate a quelle elettriche mediante delle relazioni di seguito definite:

$$\omega_e = N_r \omega_r; \qquad \theta_e = N_r \theta_r \tag{6}$$

in cui si evince un legame dipendente dal numero di espansioni polari.

6.2 FFDL-MFAC applicato ad un PMSM

Nel corso dell'elaborato è stata accuratamente esaminata la tecnica di controllo FFDL-MFAC; perciò, di seguito sarà applicata al processo che descrive il motore sincrono a magneti permanenti, dopo averne analizzato le equazioni caratteristiche.

Come già specificato in precedenza, la tecnica di controllo FFDL-MFAC è nata per controllare i sistemi a tempo discreto. E' opportuno, perciò, discretizzare il modello descritto per poter applicare l'algoritmo.

L'equazione che si ottiene tramite una discretizzazione si presenta nella seguente forma:

$$\omega_e(k+1) = A_\omega \omega_e(k) + B_\omega(K_t i_q - \tau_l) \qquad (7)$$

in cui, considerando un passo di campionamento T_c, si hanno $A_\omega = e^{-\frac{B}{J}T_c}$ e $B_\omega = \frac{1}{J}\int_0^{T_c} e^{-\frac{B}{J}\tau}\, d\tau$.

E' opportuno tenere conto anche di eventuali incertezze che potrebbero modificare la dinamica del sistema, perciò si considerano i seguenti parametri:

$$A_\omega = \bar{A}_\omega + \Delta A_\omega; \qquad B_\omega = \bar{B}_\omega + \Delta B_\omega$$

$$|\Delta A_\omega| \leq \rho_{A_\omega}; \qquad |\Delta B_\omega| \leq \rho_{B_\omega} \qquad (8)$$

In questo modo, si ipotizza che i parametri iniziali possano discostarsi dal loro valore nominale di una quantità limitata che rappresenta l'incertezza.

A questo punto è possibile controllare il processo che descrive il motore tramite la tecnica di controllo FFDL-MFAC considerando come variabile da controllare la velocità angolare rotorica ω_e e come ingresso di controllo la corrente statorica i_q.

Come prima cosa, è necessario determinare la funzione che descrive l'uscita desiderata. In questo caso, dall'esempio preso in considerazione nell'articolo *"A Quasi Sliding Mode Approach for Robust Control and Speed Estimation*

of PM Synchronous Motors" – Maria Letizia Corradini, Gianluca Ippoliti, Sauro Longhi and Giuseppe Orlando, l'uscita desiderata assume il seguente andamento:

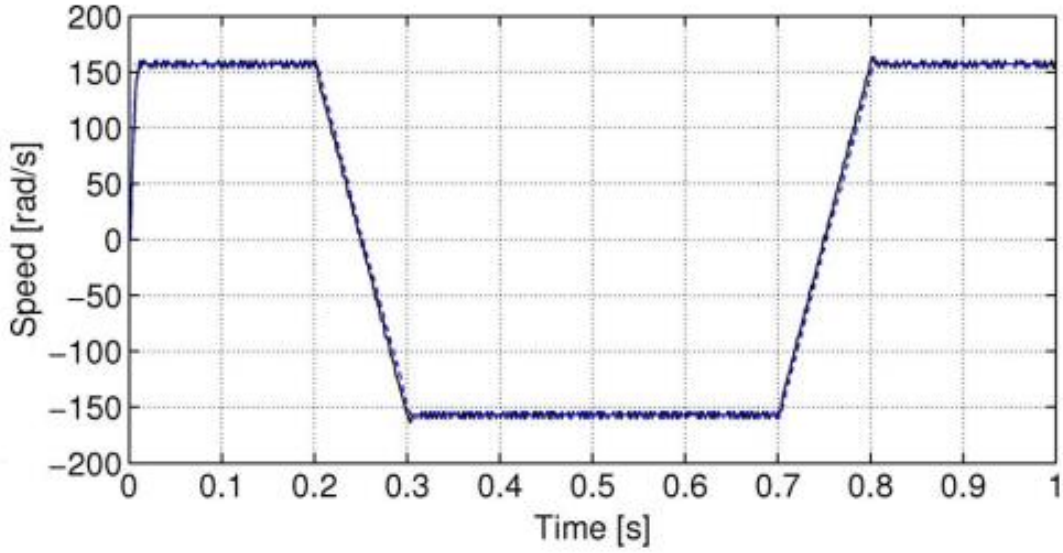

Fig. 42: Uscita desiderata PMSM

Per l'esempio preso in esame sono già stati definiti i valori relativi ai parametri $\bar{A}_\omega$, $\bar{B}_\omega$ e K_t, in particolare:

$$\bar{A}_\omega = 0.9999$$

$$\bar{B}_\omega = 909.0877$$

$$K_t = 0.0378$$

Nel processo a cui verrà applicata la tecnica di controllo è presente anche il termine relativo alla coppia di carico τ_l che, per semplicità, sarà considerata nulla.

A questo punto è possibile, come per gli esempi già visti in precedenza, effettuare una simulazione in *Simulink* per analizzare l'efficienza dell'algoritmo anche nel caso di un motore sincrono a magneti permanenti.

L'intero schema a blocchi si presenta nella seguente forma:

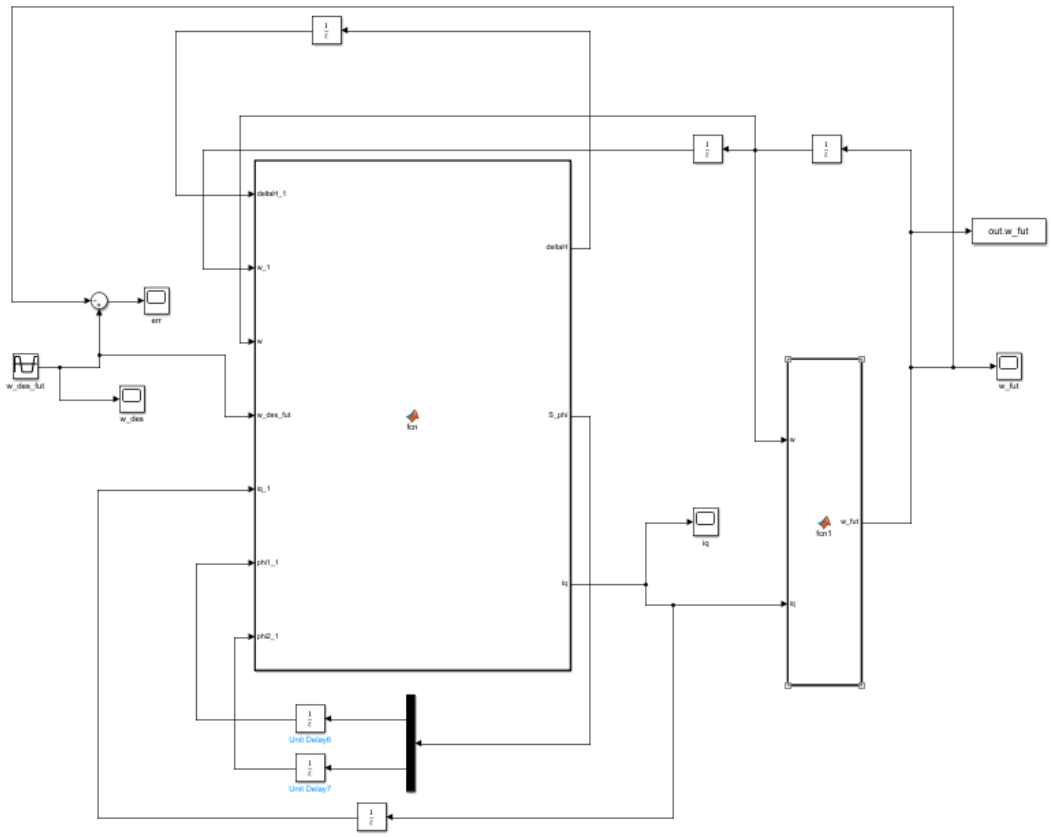

Fig. 43: Schema a blocchi in Simulink PMSM

Lo schema è analogo a quelli già descritti in precedenza; infatti, sono stati utilizzati i medesimi blocchi. L'unico blocco che è stato sostituito è quello relativo all'uscita desiderata poiché assume un andamento diverso dai segnali precedenti, che consistevano in semplici sequenze di scalini.

Il blocco utilizzato per descrivere l'uscita desiderata si chiama *Repeating Sequence Interpolated*:

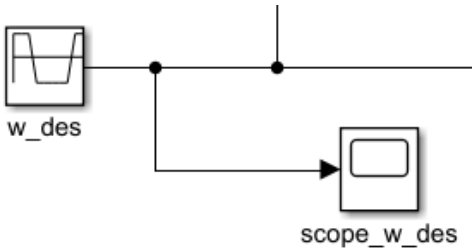

Fig. 44: Blocchi "Repeating Sequence Interpolated" e "Scope"

Di seguito sono riportati i due codici inseriti nella simulazione attraverso il blocco *Matlab function* relativi rispettivamente all'implementazione dell'algoritmo di controllo e alla definizione del processo considerando i parametri definiti precedentemente.

```
function [deltaH,S_phi,iq] = fcn(deltaH_1,w_1,w,w_des_fut,iq_1, ...
                    phi1_1,phi2_1,lambda, mu, etha, rho, phi1, phi2, epsi)

 deltaW = w-w_1;
 S_phi_1 = [phi1_1;phi2_1];
 S_phi = S_phi_1+(etha*deltaH_1)*(deltaW)/(mu+norm(deltaH_1)^2)-
               ((etha*deltaH_1)*S_phi_1.'*deltaH_1)/(mu+norm(deltaH_1)^2);
if norm(S_phi)<=epsi || norm(deltaH_1)<=epsi || sign(S_phi(1))~= sign(phi1)
    S_phi(1) = phi1;
    S_phi(2) = phi2;
end

 Xi = (S_phi(2))/(lambda + abs(S_phi(2)^2));
 iq = iq_1 + Xi*(rho*(w_des_fut-w)-(rho*S_phi(1)*deltaW));

 H = [w;iq];
 H_1= [w_1;iq_1];
 deltaH = H - H_1;
```

Fig. 45: Algoritmo di controllo FFDL-MFAC in Simulink PMSM

Questa funzione che implementa l'algoritmo di controllo FFDL-MFAC ha come output la variazione della matrice H, la matrice delle stime del vettore PG e l'ingresso di controllo i_q che servirà successivamente per definire il processo da controllare. Infatti, quest'ultimo è dipendente dall'ingresso di controllo e dalle uscite valutati negli istanti precedenti, perciò il segnale i_q appena calcolato dall'algoritmo costituirà uno degli input della seconda *Matlab function*.

```
function w_fut = fcn1(w,iq)

    A_w = 0.9999;
    B_w = 909.0877;
    Kt = 0.0368;
    tau_l=0;

    w_fut=A_w*w + B_w*(Kt*iq+tau_l);

end
```

Fig. 46: Definizione processo in Simulink PMSM

In questa funzione è stato definito il processo da controllare che, come si evince osservando gli argomenti della funzione stessa, dipende dall'ingresso di controllo e dall'uscita all'istante precedente. Prima di ciò, sono stati definiti i valori dei parametri caratteristici del processo.

Come già descritto precedentemente, per avviare la simulazione è necessario creare un nuovo file contenente i valori dei parametri dell'algoritmo di controllo. L'assegnazione dei valori numerici è stata effettuata tramite una lunga serie di tentativi che hanno portato ad un andamento dell'uscita desiderato.

```
%parametri
lambda = 9.8;
mu = 0.001;
etha = 1.1;
rho = 1;
epsi= 10^-5;
phi1=0.978;
phi2=0.3;

%simulazione
sim('simMotore',1)
```

Fig. 47: Definizione parametri per l'avvio della simulazione in Simulink PMSM

Anche in questo caso, dopo aver mandato in esecuzione il file, è possibile analizzare tutti gli andamenti dei segnali desiderati, utilizzando gli appositi blocchi chiamati *Scope*. Come primo segnale, si analizza il segnale relativo all'uscita desiderata prodotto dal blocco *Repeating Sequence Interpolated*:

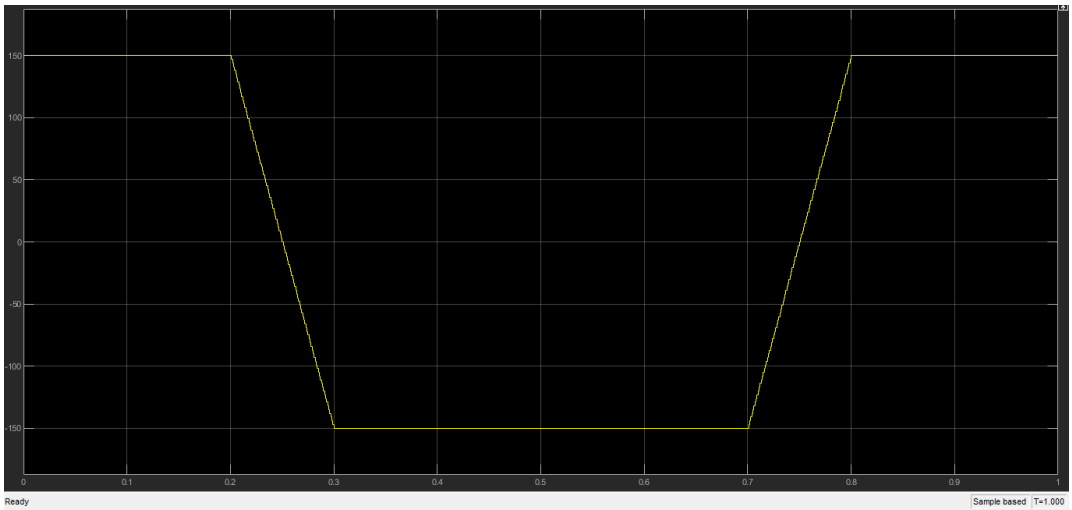

Fig. 48: Uscita desiderata PMSM

Successivamente, è riportata l'uscita effettiva del sistema:

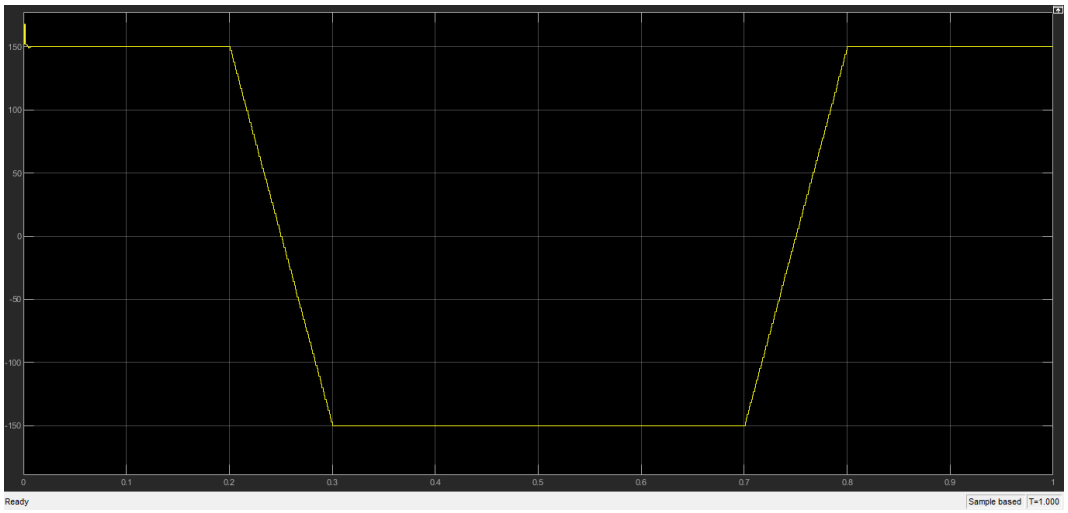

Fig. 49: Uscita effettiva PMSM

Come si evince da un primo confronto tra i due grafici, gli andamenti dei due segnali sono molto simili.

Per ottenere un'analisi più dettagliata del funzionamento dell'algoritmo è opportuno considerare la variabile errore che è data dalla differenza tra l'uscita desiderata e quella effettiva del sistema. Il corretto funzionamento della tecnica di controllo applicata si deduce dall'andamento di questa nuova variabile, poiché deve assumere valori molto piccoli in ogni punto. Di seguito è riportato il grafico relativo all'errore, che è stato aggiunto nello schema a blocchi tramite un nodo sommatore (*Sum*) che serve per effettuare la differenza tra i segnali.

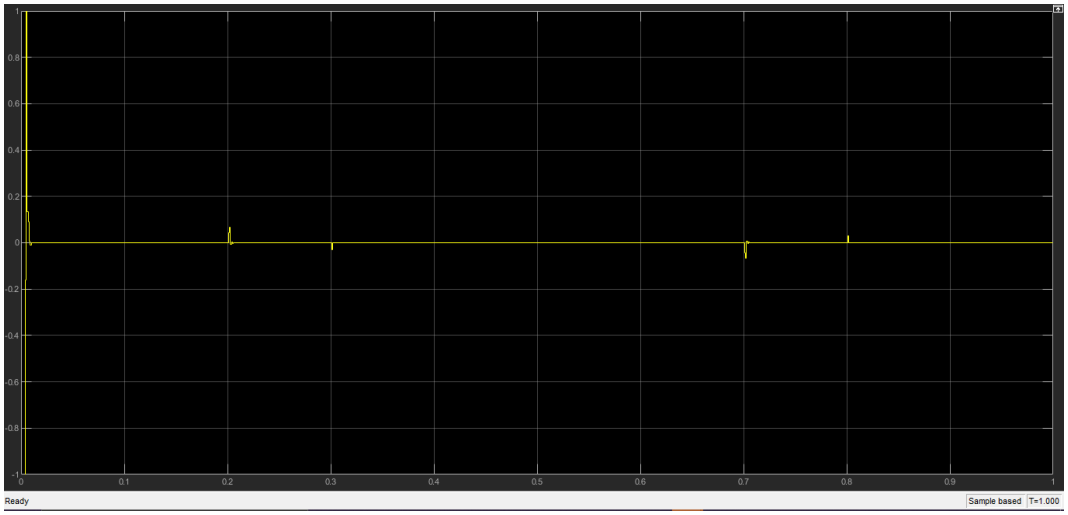

Fig. 50: Errore PMSM

Dal grafico si può notare un'oscillazione accentuata negli istanti iniziali, mentre per gli istanti successivi si ha una convergenza a zero. Sono presenti delle lievi oscillazioni nei punti in cui la curva dell'uscita varia bruscamente. Dall'osservazione di questo grafico è possibile affermare che, anche in questo caso, l'algoritmo di controllo FFDL-MFAC si è dimostrato una tecnica efficiente, considerando i valori molto piccoli dell'errore.

L'ultimo segnale mostrato è quello relativo all'ingresso di controllo i_q:

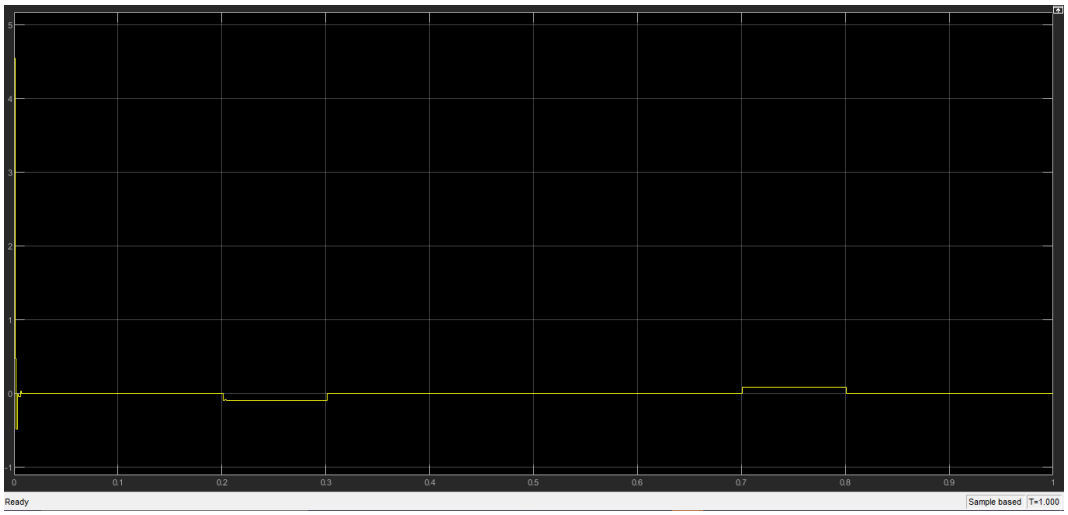

Fig. 51: Ingresso di controllo PMSM

7 Conclusioni

L'elaborato aveva come obiettivo quello di spiegare il più accuratamente possibile le caratteristiche più importanti della teoria di controllo Data-Driven, esaminando i vantaggi e gli svantaggi principali che si possono riscontrare effettuando un confronto con la teoria Model-Based.

In particolare, la tecnica di controllo Data-Driven che è stata esaminata nel dettaglio è la teoria MFAC basata sulla linearizzazione dinamica FFDL. Sono state inoltre descritte brevemente le altre due tecniche di linearizzazione che possono essere applicate, ossia le tecniche PFDL e CFDL.

La teoria di controllo FFDL-MFAC è stata inizialmente analizzata a livello teorico, introducendo tutti i concetti necessari da un punto di vista più generale, per poi essere applicata ad alcuni processi come, ad esempio, un motore sincrono a magneti permanenti.

Grazie alle applicazioni effettuate in Matlab e Simulink è stato possibile verificarne il corretto funzionamento e l'efficacia. Infatti, sono stati presi in analisi sia processi lineari che non lineari e, in entrambi i casi, i risultati ottenuti sono stati più che soddisfacenti.

Nell'ultimo caso, quello relativo al motore, i parametri caratteristici dell'algoritmo sono stati scelti effettuando numerosi tentativi. Il controllore, infatti, dipende dai valori assunti da questi parametri ed è stato possibile vedere come l'inesattezza di uno di questi valori possa causare l'instabilità del sistema.

In conclusione, ciò che è stato appreso è che la tecnica di controllo esaminata in questo elaborato si è rivelata semplice e poco dispendiosa dal punto di vista computazionale, ma allo stesso tempo capace di garantire tutte le proprietà fondamentali come ad esempio la stabilità, la convergenza e la robustezza del sistema.

Elenco delle figure

Tabella delle variabili Matlab

VARIABILI	SIMBOLI
λ	lambda
μ	mu
η	etha
ρ	rho
$\widehat{\boldsymbol{\Phi}}_{f,L_y,L_u}(1)$	phi_1
$\widehat{\boldsymbol{\Phi}}_{f,L_y,L_u}(k)$	S_phi
$y_d(k+1)$	yd
$y(k)$	y
$u(k)$	u
$\boldsymbol{H}_{L_y,L_u}(k)$	H
$e(k)$	err
$\Delta u(k)$	deltaU
$\Delta y(k)$	deltaY
$\Delta\boldsymbol{H}_{L_y,L_u}(k)$	deltaH
$\hat{\xi}_{L_y+1}(k)$	Xi

Tabella delle variabili Simulink

VARIABILI	SIMBOLI
λ	lambda
μ	mu
η	etha
ρ	rho
ε	epsi
$\widehat{\boldsymbol{\Phi}}_{f,L_y,L_u}(1)$	phi1
$\widehat{\boldsymbol{\Phi}}_{f,L_y,L_u}(2)$	phi2
$\widehat{\boldsymbol{\Phi}}_{f,L_y,L_u}(3)$	phi3

$y(k)$	yk
$y(k-1)$	y_1
$\Delta y(k)$	deltaY
$\boldsymbol{H}_{L_y,L_u}(k)$	H
$\widehat{\boldsymbol{\Phi}}_{f,L_y,L_u}(k)$	S_phi
$\widehat{\boldsymbol{\Phi}}_{f,L_y,L_u}(k-1)$	S_phi_1
$\Delta\boldsymbol{H}_{L_y,L_u}(k)$	deltaH
$u(k)$	u
$u(k-1)$	u_1
$\Delta u(k)$	deltaU
$y_d(k+1)$	yd_fut
$\omega(k)$	w
$\omega(k-1)$	w_1
$\Delta\omega(k)$	deltaW
$i_q(k)$	iq
$i_q(k-1)$	iq_1
$\omega_d(k+1)$	w_des_fut

Bibliografia e Sitografia

[1] Zhongsheng Hou, Ronghu Chi and Huijun Gao: *An Overview of Dynamic-Linearization-Based Data-Driven Control and Applications.*

[2] Zhongsheng Hou and Shuangshuang Xiong: *On Model-Free Adaptive Control and Its Stability Analysis.*

[3] Danna Wang: *A Novel Controller Dynamic Linearization-Based Model Free Adaptive Predictive Control for A Class Of Discrete Nonlinear Systems.*

[4] Zhong-Sheng Hou and Zhuo Wang: *From model-based control: Survey, classification and perspective.*

[5] Maria Letizia Corradini, Gianluca Ippoliti, Sauro Longhi and Giuseppe Orlando: *A Quasi-Sliding Mode Approach for Robust Control and Speed Estimation of PM Synchronous Motors.*

[6] Gianluca Ippoliti: *Motori Brushless – Tecnologie per l'Automazione e la Robotica, Ingegneria Informatica e dell'Automazione, Univpm.*

[7] http://www.colombi.pc.it/prodotti/motori-elettrici/motore-sincrono-a-magneti-permanenti.html

[8] https://www.ne555.it/controllo-pid-pi-microcontrollore/sistema-pid-schema-a-blocchi/