UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI **INGEGNERIA**

Corso di Laurea triennale in

Ingegneria Informatica e dell'automazione

IDENTIFICAZIONE E CONTROLLO DI UN ATTUATORE DI PRESSIONE

IDENTIFICATION AND CONTROL OF A PRESSURE ACTUATOR

Relatore: Chiar.mo
Prof. **Giuseppe Conte**

Tesi di Laurea di:
Alessandro De Toni

A.A. **2019/2020**

Indice

INTRODUZIONE

Lo scopo di questa tesi è quello di descrivere il lavoro svolto durante il tirocinio proposto dall'azienda "AEA srl. - Gruppo Loccioni" e di completarne la sostanza con l'enunciazione dei fondamenti teorici applicati.

A causa delle problematiche dovute alla COVID-19, la maggior parte del lavoro è stato svolto da casa, ma, a questo, sono state aggiunte delle visite in azienda per verificare la consistenza dei risultati ottenuti.

Come riassume sinteticamente il titolo, l'elemento cardine di questo elaborato è un attuatore di pressione, la cui funzione è quella di regolare, appunto, la pressione in ingresso ad un iniettore automobilistico. Più in generale, esso costituisce un piccolo tassello di un sistema ben più complesso che si occupa di testare la qualità costruttiva dei sopracitati iniettori.

Perciò, come affrontato in dettaglio nel capitolo 1, ci si propone di sviluppare e implementare una legge di controllo adeguata al funzionamento del componente, previa la determinazione del suo modello matematico.
Un ruolo particolarmente importante è svolto dagli strumenti software di cui si è usufruito per ottenere i risultati desiderati. Essi assumono, sempre più, una posizione preponderante in tutto ciò che concerne l'automazione: generare simulazioni, scalabilità, facilità di manutenzione, costituiscono, ormai, delle potenzialità di cui non si può più fare a meno.

Questo in Loccioni è ben noto ed è per questo motivo che si è colta l'occasione per un primo approfondimento di queste tematiche attraverso l'utilizzo degli ambienti Matlab e Simulink.

Già nella prima fase, si è fatto uso di un particolare toolbox di Matlab (System Identification toolbox) per la determinazione di un modello polinomiale (ARX - Autoregressive Exogenous model) che descrivesse in modo accurato il sistema attuatore. Il processo che ha condotto a questo prodotto si diparte dal classico approccio "black box" che ha richiesto l'acquisizione di numerosi campionamenti della coppia ingressi/uscite appartenente al componente reale. In vista di ciò, si è fatto uso della letteratura riguardante l'identificazione dei processi dinamici e, in concordanza con essa, si è cercato di fornire un ingresso il più simile possibile a un rumore bianco.
Gran parte del tempo dedicato a questo processo è stato investito nel discernere quale, tra i diversi modelli calcolati attraverso il toolbox, fosse il più adatto a rappresentare l'attuatore. Fondamentali, in questo senso, sono state le nozioni relative alla stabilità (secondo Lyapunov) dei sistemi dinamici, nonché quelle statistiche che rientrano nell'analisi dei residui, e cioè in quell'indagine che si propone di determinare se tutti i comportamenti manifestati dal componente reale si presentino o meno nella simulazione originata dal modello matematico.

Successivamente, si è riportato l'ARX, sotto forma di funzione di trasferimento, all'interno di Simulink e si è proceduto alla progettazione di un regolatore PI che comunicasse la corretta posizione all'azionamento, che muove l'attuatore, in vista dell'ottenimento della pressione desiderata in uscita. Al suo interno è stato implementato l'algoritmo della "back-calculation" per impedire il fenomeno della crescita del segnale di controllo dovuta all'azione integrale, comunemente denominato "Windup".

Terminato il suo sviluppo, il controllore è stato implementato nel sistema di controllo informatico "TwinCAT". Questo software è presente sul calcolatore che controlla la macchina di cui fa parte l'oggetto di questa tesi e il suo ruolo è quello di gestirne il comportamento tramite PLC.

Infine, per verificare la correttezza e la qualità del lavoro svolto, il controllore è stato effettivamente messo in funzione all'interno del macchinario e sono stati prelevati i dati di funzionamento per determinare se le performance raggiunte fossero conformi con quelle richieste dall'azienda.

Capitolo 1

PRESENTAZIONE DEL PROBLEMA

1.1 Definizione del problema e obiettivi

Gli obiettivi a partire dai quali si articola questo elaborato sono molteplici: innanzitutto, è richiesto di identificare il modello matematico di un attuatore, il cui scopo è quello di regolare la pressione di una camera, contenente sostanze allo stato liquido, utili al testing del corretto funzionamento di alcuni iniettori automobilistici. Sulla base di questo, occorre, poi, ricercare, scegliere, implementare diversi algoritmi di controllo in MATLAB che dovranno essere integrati all'interno dell'ambiente TwinCAT (software PLC real time).

Tutto ciò è di interesse per l'azienda Loccioni, perché l'utilizzo di software come MATLAB e Simulink costituisce un argomento poco esplorato ma di grandi potenzialità. Tra queste si annoverano la semplicità di progettazione e sintesi di un regolatore, la grande varietà di applicazioni e toolbox, continuamente aggiunti, da cui è possibile attingere per sopperire ai bisogni più disparati, la possibilità di generare e gestire simulazioni permettendo di velocizzare la fase di testing e di godere di maggior sicurezza. Infine, la loro grande diffusione, ha condotto alla creazione di una solida community con cui

è possibile confrontarsi per dubbi, spunti e problemi. A questo si aggiunge la volontà di tentare un primo approccio all'identificazione dei componenti proprietari, una pratica ancora poco approfondita dall'azienda.

1.2 L'attuatore

L'oggetto di questa tesi, come già citato, è un moltiplicatore elettrico, nello specifico si tratta di un prodotto "custom", progettato e commissionato direttamente da Loccioni.
Al variare della posizione di uno stelo all'interno di un cilindro (con una corsa contenuta nell'intervallo [0,20] mm), esso incrementa o diminuisce la pressione del liquido contenuto in una camera.

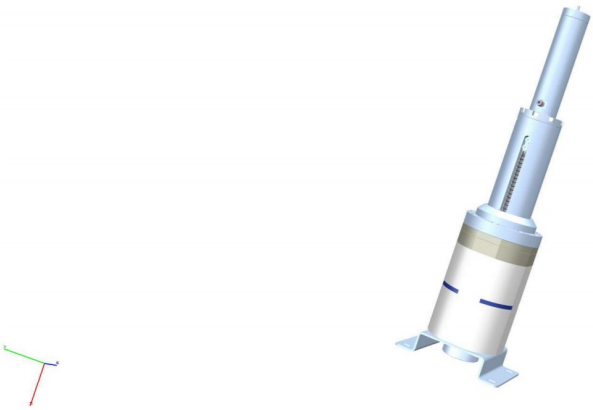


Figura 1.1: Prospetto anteriore del moltiplicatore elettrico

Il suddetto componente viene azionato tramite un motore sincrono trifase a magneti permanenti "SITEM MTRI 190.100.LCC", la cui posizione è misurata dall' encoder "SKS/SKM36" prodotto da "SICK".

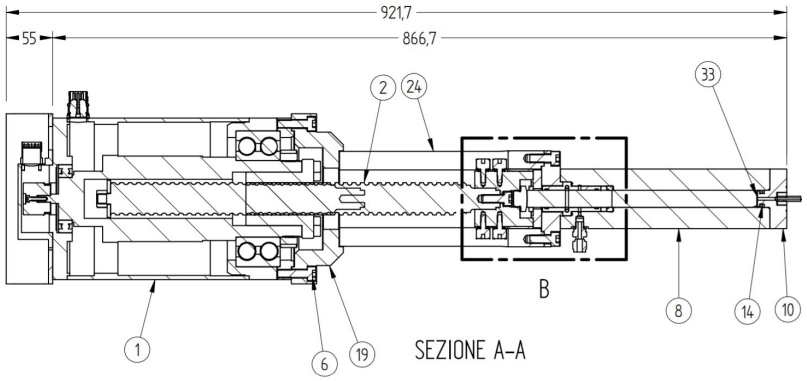


Figura 1.2: Progetto parziale dell'attuatore, motore situato in corrispondenza di 1

Interessante per l'argomento di questa tesi è anche il sensore di pressione "UNIK 5000" di "GE - General Electric", il quale fornisce un output in corrente nel range 4-20 mA, può misurare dai 70 mBar ai 700 Bar e può raggiungere una frequenza massima di campionamento di 3.5 kHz.

Si fa presente, infine, che l'attuatore altro non è che un piccolo componente facente parte di una struttura complessa, la cui unità centrale di comando è un PC embedded della serie CX prodotto da Beckhoff.

Capitolo 2

IDENTIFICAZIONE

2.1 Il processo di identificazione

Identificare un sistema significa stimarne un modello a partire dai dati sperimentali. Più nello specifico, ciò che risulta interessante è stabilire una rappresentazione matematica della sua dinamica da cui estrarre una descrizione in termini di variabili di stato.

Ci sono diversi approcci per affrontare tale problematica.
Banalmente, si può pensare di osservare attentamente un sistema e di descriverne i comportamenti attraverso le leggi della fisica, oppure, come in questo caso, di trascurare completamente le sue caratteristiche fisiche e di affidarsi ai dati sperimentali relativi agli ingressi forniti e alle uscite prelevate, come insegna il metodo black box.

É chiaro, allora, che i dati ricavati dal sistema ricoprono un ruolo fondamentale nel processo di identificazione. In particolare, è necessario eccitarlo con un ingresso forzante che gli permetta di esprimere in uscita tutta la sua dinamica (si parla di ingresso persistentemente eccitante) e di utilizzare una frequenza di campionamento che sia (per il teorema di Shannon) almeno il doppio del limite della sua banda di frequenze.

Procedimento di stima Poco sopra, ho espresso quanto sia importante che l'ingresso sia persistentemente eccitante e, oltre alla spiegazione intuitiva, ve n'è una più rigorosa che necessita di introdurre come avviene il procedimento di stima del modello nel caso di un'identificazione di tipo parametrico[5].

Si elencano, quindi, i passi che caratterizzano il procedimento di stima:

- *Raccolta dati* - Si fornisce un ingresso $u(t)$ al sistema S e se ne preleva l'uscita $y(t)$ nell'intervallo di tempo $[0, N]$, l'insieme $(u(t), y(t))$ è detto vettore delle osservazioni $\phi(t)$. Attraverso $\phi(t)$ si costruisce la matrice $S(N) = \sum_{t=\tau}^{N} \phi(t)\phi(t)^T$.

- *Scelta della famiglia di modelli* - In letteratura sono state introdotte numerose famiglie di modelli $\mathbf{M}$ che forniscono una struttura prefissata a cui ricondurre il sistema in base alle sue caratteristiche (p. es. distinzione tra serie temporale o sistema di controllo, posizione dell'errore, sistemi lineari o non lineari ecc.). In generale si può dire che la struttura definita da $\mathbf{M}$ abbia l'obiettivo di esprimere $y(t)$ attraverso una combinazione dei valori che l'uscita e l'ingresso (o, semplicemente l'uscita se si tratta di serie temporali) assumono in istanti precedenti a t [1]. I parametri che descrivono questa combinazione sono collezionati in un unico vettore θ, che, per l'appunto, viene chiamato vettore dei parametri. Lo specifico modello, corrispondente a un certo vettore θ appartenente alla famiglia $\mathbf{M}$, viene allora indicato con la notazione $M(\theta)$.

- *Stima del vettore dei parametri* θ - Scelta la famiglia $\mathbf{M}$, è il momento di trovare il modello che le appartiene, che descrive nel modo più accurato possibile S, attraverso la determinazione di θ.

[1] Questo concetto diverrà più concreto nella sezione 2.2.2 dove l'uscita verrà espressa come una combinazione lineare dei valori che y e u assumono, rispettivamente, negli na e negli nb istanti precedenti.

Per questo motivo, si costruisce un predittore $\hat{M}(\theta)$ associato al modello $M(\theta) \in \mathbf{M}$, attraverso $\hat{M}(\theta)$ si determina l'espressione generica in funzione di θ di una stima $\hat{y}(t)$ dell'uscita del sistema reale $y(t)$ e, da queste, l'errore di predizione $\varepsilon_\theta(t) = y(t) - \hat{y}(t)$.
A questo punto, una via possibile è quella di utilizzare il metodo dei minimi quadrati, determinare l'espressione del *funzionale errore quadratico medio* $J_N(\theta) = 1/N \sum_{t=\tau}^{N} \varepsilon(t)^2$ e trovare una stima del vettore dei parametri come punto di minimo $\theta_{min,N}$ di $J_N(\theta)$.

- *Validazione* - Trovato il modello $M(\theta_{min,N})$ del sistema S, occorre accertarsi che esso sia effettivamente un buon modello. Ciò risulta necessario poiché $J_N(\theta)$ fornisce una misura dell'aderenza del modello ai dati su cui esso si basa, ma non ci assicura che con altri dati, l'uscita simulata sia comparabile a quella reale.
  Inoltre, si può incorrere nel fenomeno della *sovraparametrizzazione*, il funzionale errore quadratico medio, infatti, diminuisce all'aumentare della complessità del modello (in sostanza il numero di parametri che lo descrivono). Questo, si deve al fatto che i nuovi gradi di libertà, aggiunti aumentando la complessità, vengono utilizzati per modellare anche l'andamento dei disturbi randomici che agiscono sui dati, introducendo, di fatto, una componente estranea al sistema.

  Per evitare che ciò avvenga e per verificare la qualità del modello vi sono diverse tecniche:

  - *Analisi poli/zeri* - si rappresentano i poli e gli zeri sul piano di Gauss di $M(\theta_{min,N})$ e si verifica che tra di essi non siano presenti delle sovrapposizioni.
  - *Valutazione della bianchezza dell'errore di predizione* - si sottopone l'errore di predizione al test di Anderson, in modo da assicurarsi che la complessità sia abbastanza elevata da esprimere tutta la dinamica del sistema. Se, infatti, l'errore di predizione non è

un rumore bianco, significa che stiamo trascurando una parte significativa del modello, che appunto, non essendo modellata, si presenta sotto forma di errore.
Per un ulteriore approfondimento dell'argomento si rimanda alle sottosezioni 2.1.1 e 2.2.1.

- *Cross validazione* - tecnica che consiste nel dividere in due il set di dati campionato, utilizzando la prima parte per il processo di identificazione e la seconda parte per valutare la bianchezza dell'errore di predizione. Quest'ultimo si ottiene come una differenza tra l'uscita reale campionata e l'uscita stimata dal modello, al quale sono inviati come ingresso, gli ingressi utilizzati per ottenere la seconda parte dei dati.
- *Indici FPE, AIC, MDL* - sono indici, funzioni di $J_N(\theta)$ e della complessità n (p. es. $FPE = \frac{N+n}{N-n} J_N(\theta)$), che presentano un fattore di penalizzazione crescente al crescere di n. Essi forniscono un criterio oggettivo di arresto dell'algoritmo di ricerca del modello, impedendogli, così, di raggiungere complessità eccessivamente elevate.
- *Valutazione dell'incertezza dei parametri* - al termine del procedimento di stima si ottiene un modello con parametri noti in un certo range di variabilità. Perché il modello sia buono, occorre che la varianza, riferita ai parametri, sia la minore possibile, o comunque, caratterizzata da un ordine di grandezza non comparabile con quello del termine corrispondente.

Date le considerazioni introdotte nelle pagine precedenti, ora è possibile introdurre il concetto rigoroso di ingresso persistentemente eccitante.

Definizione - Un segnale $u(t)$ si dice persistentemente eccitante (pe) di ordine n se[6]:

1. il seguente limite esiste:

$$r_u(\tau) = \lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N} u(t+\tau)u^T(t); \tag{2.1}$$

2. la matrice:

$$R_u(n) = \begin{pmatrix} r_u(0) & r_u(1) & \cdots & r_u(n-1) \\ r_u(-1) & r_u(0) & \cdots & r_u(n-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_u(1-n) & r_u(2-n) & \cdots & r_u(0) \end{pmatrix} \tag{2.2}$$

   è definita positiva.

Brevemente, la matrice $R_u(n)$, altro non è che una sottomatrice di $S(N)$, la cui invertibilità decreta la convergenza di θ a $\theta_{min,N}$ attraverso la risoluzione delle equazioni normali associate all'errore quadratico medio, come sarà introdotto nella sottosezione 2.2.2.

Quindi, in sostanza, tramite $R_u(n)$ è possibile sapere se il vettore di parametri θ, che permette di identificare il sistema S, tramite il modello $M(\theta)$ convergerà o meno a $\theta_{min,N}$.

Volendo dare una lettura dal punto di vista statistico, $R_u(n)$, equivale alla matrice di covarianza riferita all'ingresso $u(t)$ e a una sua realizzazione ad istanti precedenti o successivi. Ne segue che se i due segnali hanno media nulla e sono tra loro incorrelati ($Cov(u(t), u(t-k)) = 0, conk \neq 0$), $u(t)$ costituisce un ingresso persistentemente eccitante.

2.1.1 Rumore bianco

Il segnale ideale per la modellazione è costituito da una sequenza di numeri casuali, tali che l'informazione relativa al suo andamento in un certo intervallo di tempo non permette in alcun modo di predire il suo andamento in intervalli di tempo successivi. In termini matematici, ciò si traduce in un segnale avente media nulla e matrice di covarianza multipla della matrice identità. Un segnale di questo tipo esiste, aggiunge alle caratteristiche appena descritte il fatto di essere costante su tutto lo spettro di frequenze ed è chiamato vettore di rumore bianco $\omega(t)$.

$$\mu_\omega(t) = \mathbf{E}\{w\} = 0 \tag{2.3}$$

$$R_{\omega\omega}(t, t+\tau) = \mathbf{E}\{w(t)w(t+\tau)^T\} = \sigma^2\delta(t-t+\tau)\forall\tau^2 \tag{2.4}$$

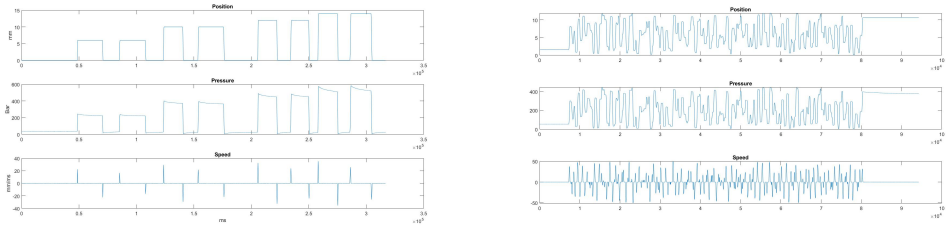
É chiaro, allora, che il vettore di rumore bianco si annoveri tra i segnali persistentemente eccitanti.

Ovviamente, un tale segnale costituisce solo un'idealizzazione, visto che non esiste alcun sistema in grado di generare uno spettro uniforme, per tutto il range di frequenze compreso tra zero e infinito. Tuttavia, è comunque possibile utilizzare un rumore quasi bianco a banda limitata (cioè restringendo il range di frequenze con spettro uniforme alle frequenze di lavoro tipiche del sistema che vogliamo modellare) senza che questa approssimazione infici l'accuratezza del modello [5].

[2] infatti, la funzione delta di Dirac assume un valore non nullo se e solo se $\tau = 0$. Ciò corrisponde agli elementi presenti sulla diagonale di $R_{\omega\omega}$.

2.2 Analisi Dati

Scelta dell' ingresso Nella sezione 2.1.1 ho evidenziato come l'ingresso ideale sia un rumore bianco a banda limitata. Tuttavia, come in questo caso, non è sempre possibile inviare un ingresso di questo tipo. Per questo motivo, inizialmente, si è tentato di porsi nelle normali condizioni di lavoro forzando l'attuatore con 3 coppie di rampe che coprissero, prima l'intervallo [0,6] mm, poi quelli di estremi [0,10] mm e [0,12] mm, infine quello di [0,14] mm. In ogni coppia, inoltre, le due rampe si distinguono per la velocità con cui raggiungono il set-point.
Come ci si poteva aspettare, i risultati si sono rivelati tutt'altro che soddisfacenti. Quello che si è provato, allora, è stata la generazione di N numeri casuali sempre compresi nell'intervalo [0,12] mm. Questo, infatti, costituisce il miglior compromesso disponibile, visto che la libreria TwinCAT che gestisce il motore "SITEM MTRI" impone che esso possa ricevere un nuovo comando solo successivamente al raggiungimento della posizione precedentemente richiesta. Altri segnali canonici, come un PWM, sono stati scartati per lo stesso motivo. Con questo tipo di segnale (N numeri randomici) sono state eseguite numerose acquisizioni, di cui l'acquisizione 4 sarà la più importante, in quanto ha portato al modello scelto in fase finale.



(a) Primo tentativo (b) Acquisizioni successive

Figura 2.1: Grafici IN/OUT con annessa velocità motore

Tempo di campionamento Come si intuisce dall'introduzione, la frequenza massima di campionamento ammissibile, a causa dei limiti hardware, è di 3.5 kHz.
Tuttavia, il PLC che gestisce il sistema, e in generale, il software TwinCAT che permette di eseguire le acquisizioni delle misure rilevate dai sensori, lavorano con un tick time di 1 ms.
Per questo motivo il tempo di campionamento è stato settato a 1 kHz.

Pulizia dei Dati Si Introduce, qui, il procedimento utilizzato per eliminare gli errori dei sensori e per ottenere un plot adeguato delle grandezze rilevanti per l'identificazione. Ciò si ripete pressappoco allo stesso modo per tutti i campionamenti eseguiti dall'azienda. Per tutte le acquisizioni di dati, è stato fornito un file Excel di questo tipo, ottenuto attraverso la funzionalità "Scope" di TwinCAT:

Column1	Column2	Column3	Column4	Column5	Column6
Column1	Column2	Column3	Column4	Column5	Column6
Name	ElectricIntensifierData				
File	C:\Users\g.valentini\Desktop\tempi\ElectricIntensifierData.t				
Starttime of expo	132348926862860000	25 May 2020	17:04:46.286		
Endtime of expor	132348930029790000	25 May 2020	17:10:02.979		
Name	ActPos	Name	ActVelo	Name	HeadPressureRaw
SymbolCommen		SymbolComme		SymbolComme	
Data-Type	REAL64	Data-Type	REAL64	Data-Type	INT16
SampleTime	1	SampleTime	1	SampleTime	1
VariableSize	8	VariableSize	8	VariableSize	2
SymbolBased	True	SymbolBased	True	SymbolBased	True
IndexGroup	16448	IndexGroup	16448	IndexGroup	61472
IndexOffset	622552	IndexOffset	622576	IndexOffset	598390
SymbolName	MOTORS.HEAD_PRESS_MOTOR_REG_SPRAY_ST1.N	SymbolName	MOTORS.HEAD_PRESS_MOTOR_REG_SPRAY_ST1.Nc	SymbolName	CPU3.CPU3_HW_ANA_IN_GEN[4]
NetID	192.168.0.1.1.1	NetID	192.168.0.1.1.1	NetID	192.168.0.1.1.1
Port	851	Port	851	Port	851
Offset	0	Offset	0	Offset	0
ScaleFactor	1	ScaleFactor	1	ScaleFactor	0,021362956633198
BitMask	0xffffffffffffffff	BitMask	0xffffffffffffffff	BitMask	0xffffffffffffffff
Unit	(None)	Unit	(None)	Unit	(None)
Unit ScaleFactor	1	Unit ScaleFactc	1	Unit ScaleFactc	1
Unit Offset	0	Unit Offset	0	Unit Offset	0
0	0	0	-1,7895163334880586E-07	0	1665
1	0	1	-1,4912636112400488E-07	1	1657
2	0	2	-1,4912636112400488E-07	2	1665
3	0	3	-1,2427196760333742E-07	3	1661
4	0	4	-1,2427196760333742E-07	4	1664
5	0	5	-1,0355997300278118E-07	5	1672
6	0	6	-1,0355997300278118E-07	6	1671
7	0	7	-8,6299977502317651E-08	7	1647
8	0	8	-8,6299977502317651E-08	8	1665
9	0	9	-7,1916647918598044E-08	9	1660
10	0	10	-7,1916647918598044E-08	10	1656
11	-7,63E-05	11	-0,0063583932638732639	11	1665
12	-7,63E-05	12	-0,0063583932638732639	12	1665
13	0	13	0,0010596722801056117	13	1682
14	0	14	0,0010596722801056117	14	1646
15	0	15	0,0008830602334234309	15	1631
16	0	16	0,0008830602334234309	16	1672
17	0	17	0,0007358835278511926	17	1671
18	0	18	0,0007358835278511926	18	1667
19	0	19	0,0006132362732092666	19	1664
20	0	20	0,0006132362732092666	20	1665
21	0	21	0,0005110302276743845	21	1658
22	0	22	0,0005110302276743845	22	1649
23	0	23	0,0004258585230619904	23	1657
24	0	24	0,0004258585230619904	24	1662
25	0	25	0,0003548821025516586	25	1649

Figura 2.2: Prima parte di un' acquisizione in formato Excel

Brevemente, `ActPos`, `ActVelo`, `HeadPressureRaw` sono i nomi utilizzati all'interno del PLC per indicare posizione, velocità e pressione, mentre i nomi a fianco del campo `SymbolName`, sono le variabili TwinCAT che si interfacciano con i diversi sensori (Encoder, sensore di pressione).

Dal file si possono notare alcune cose importanti: la pressione presenta un fattore di scala diverso da 1 (campo `Unit ScaleFactor`) e già nei primi 25 ms compaiono evidenti errori di misura di natura infinitesima (`-7.63E-05` sotto la colonna della posizione).

Inoltre, è stato utilizzato un tempo di campionamento pari a 1 ms in conformità con il tempo di aggiornamento delle variabili di IN/OUT settato all'interno del PLC.

Per procedere alla correzione del dataset è stato creato un programma MATLAB molto semplice:

Analisi_dati.m

```
%Data import
T = readtable('ElectricIntensifierData.xlsx');

%OPERATIONS on data
%1)Useless rows removal and new variable names
ElectricIntensifierData = T(27:end-1, [2:2:end]);
ElectricIntensifierData.Properties.VariableNames = {'Position', 'Speed', 'Pressure'};

%2)Data cleaning and Pressure scaling
ElectricIntensifierData = standardizeMissing(ElectricIntensifierData, '-7,63E-05');
ElectricIntensifierData = standardizeMissing(ElectricIntensifierData, '7,63E-05');
ElectricIntensifierData.Position = fillmissing(ElectricIntensifierData.Position, 'previous');
ElectricIntensifierData.Pressure = ElectricIntensifierData.Pressure*0.021362956633198;
summary(ElectricIntensifierData)

```

Figura 2.3: Codice pulizia dati, parte 1

Nel quale, inizialmente viene prelevata la parte del file contenente i valori assunti da posizione, velocità e pressione, al variare del tempo, che viene, poi, assegnata al tipo di dato "table" nativo di MATLAB. Seguono una serie di istruzioni per eliminare gli errori di misura e per ridimensionare la pressione da "punti" a Bar.

A questo, è stata aggiunta qualche riga di codice per graficare i dati:

```
21 -    x = 1:length(ElectricIntensifierData.Position);
22
23 -    subplot(3,1,1)
24 -    plot(x' ,ElectricIntensifierData.Position);
25 -    title('Position')
26 -    ylabel('mm')
27
28 -    subplot(3,1,2)
29 -    plot(x' ,ElectricIntensifierData.Pressure);
30 -    title('Pressure')
31 -    ylabel('Bar')
32
33 -    subplot(3,1,3)
34 -    plot(x' ,ElectricIntensifierData.Speed);
35 -    title('Speed')
36 -    xlabel('ms')
37 -    ylabel('mm/ms')
```

Figura 2.4: Codice pulizia dati, parte 2

2.2.1 Il Test di Anderson

La conferma del fatto che i forzamenti utilizzati, seppur costituiti da N numeri casuali, non rientrassero nella categoria dei rumori bianchi è fornita dal Test di Anderson.

Si supponga che $w(t)$ sia un segnale a media nulla $\mathbf{E}[w(t)] = 0$ di cui si vuole conoscere la bianchezza.
É noto che c.n.e.s affinché $w(t)$ sia un rumore bianco è che la matrice di covarianza sia multiplo della matrice identità.
Allora, quello che si propone di fare il test di Anderson è costruire uno stimatore $\hat{r}(\tau)$ della covarianza normalizzata $\rho(\tau)$ e verificare che $\rho(\tau) = 0, \forall\tau \neq 0$, attraverso la stima [3] $\hat{r}(\tau) = \frac{\hat{R}(\tau)}{\hat{R}(0)}$ (per la relazione che descrive $\hat{R}(\tau)$ si rimanda alla 2.1).

[3] Ci si aspetta che $\hat{r}(\tau) \simeq 0, \quad \forall\tau \neq 0$.

Per il teorema del limite centrale, per N che tende a ∞, la variabile $\sqrt{N}\hat{r}(\tau)$ tende a $N(0,1)$, dove $N(0,1)$ rappresenta una distibuzione normale di media nulla e varianza unitaria.

Ne segue che, scelto un livello di significatività, per esempio, del 5%, il segnale $w(t)$ è un rumore bianco se [7]:

$$|\hat{r}(\tau)| \leq \frac{1.96}{\sqrt{N}} \quad \forall \tau \geq 1 \tag{2.5}$$

Quello che si fa, quindi, è prendere una finestra di k ritardi ($\tau \in [1,k]$) e verificare che lo stimatore fuoriesca dall'intervallo $[-\frac{1.96}{\sqrt{N}}, \frac{1.96}{\sqrt{N}}]$ al massimo per $k * 5\%$ volte.

Graficamente ciò viene rappresentato attraverso un correlogramma, il quale tornerà utile al momento dell'analisi dei residui.

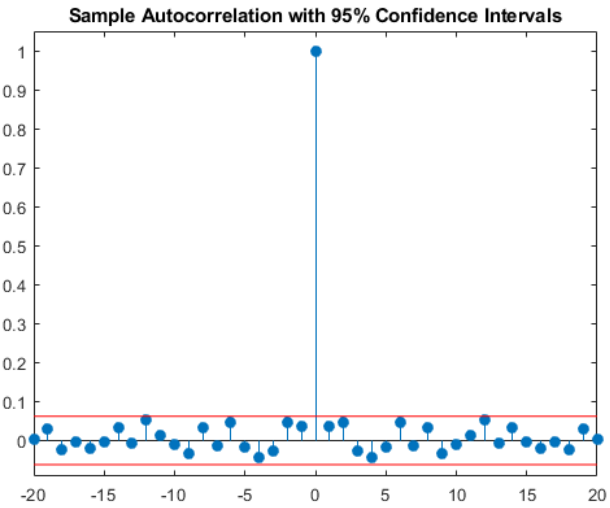


Figura 2.5: Correlogramma ottenuto attraverso MATLAB con un livello di significatività del 5% e N = 1000.

Per effettuare il test di bianchezza, sui segnali inviati in ingresso, è stato utilizzato un algoritmo[5] scritto in MATLAB che implementa le deduzioni teoriche appena presentate.

Come c'era d'aspettarsi, nessuno di essi si è rivelato essere un rumore bianco.

Si Riporta qui sotto l'esempio riferito al set di dati che ha condotto al miglior modello.

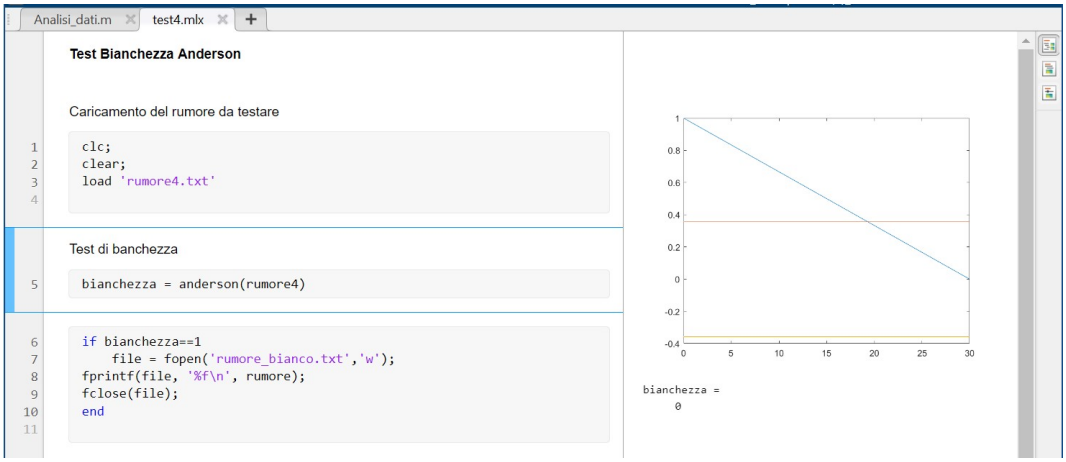


Figura 2.6: Test di bianchezza riferito all'acquisizione numero 4

2.2.2 ARX - "Autoregressive Exogenous Model"

L'approccio che sarà utilizzato per determinare le relazioni matematiche che descrivono il sistema è quello di ricondursi a una struttura già nota che ne rispecchi le caratteristiche.

Premessa É chiaro che il processo analizzato appartiene alla categoria dei SISO e che non si tratta di una serie temporale visto che è possibile eccitarlo con un forzamento esterno. Sicuramente, inoltre, vista la mancanza di vibrazioni significative che potrebbero far variare la posizione in ingresso, si può affermare l'assenza di disturbo in quella componente. Diversamente si può dire per l'uscita, sulla quale vi sarà un errore dovuto all'imprecisione dei sensori. Infine, come sarà presentato successivamente si può assumere che il modello sia lineare poiché le due grandezze esaminate hanno andamento molto simile.
Tutte queste caratteristiche conducono alla categoria dei "Modelli ad Errore d'equazione", i quali son descritti dall'equazione:

$$y(t) = G(z)u(t-1) + W(z)\xi(t) \tag{2.6}$$

che più nello specifico diventa:

$$\begin{aligned} y(t) = \; & a_1 y(t-1) + a_2 y(t-2) + \cdots + a_{na} y(t-na) + \\ & + b_1 u(t-1) + b_2 u(t-2) + \cdots + b_{nb} u(t-nb) + \\ & + w(t) \end{aligned} \tag{2.7}$$

e possono essere rappresentanti con il seguente diagramma:

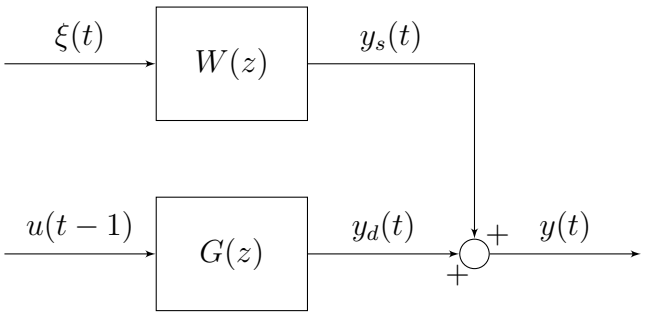


Figura 2.7: Diagramma a blocchi modello a errore d'equazione

Dove,

- $\xi(t)$ è un rumore bianco
- $u(t-1)$ è l'ingresso
- $W(z)$ è propria, $G(z) * z^{-1}$ è strettamente propria, ed entrambe hanno denominatori monici.
- $y_d(t)$ è detta componente deterministica della risposta
- $y_s(t)$ è detta componente stocastica della rispsota
- $y(t)$ costituisce l'uscita misurata durante le acquisizioni
- $w(t)$ è chiamato residuo d'equazione

Tra i modelli ad errore d'equazione (che altro non sono che modelli lineari), si annoverano gli Autoregressive Exogenous (ARX) e gli Auturegressive Moving Average Exogenous (ARMAX) che saranno oggetto del procedimento di stima utilizzato in questa tesi.

ARX Questa sottoclasse ridefinisce l'equazione 2.7 esprimendo il residuo d'equazione come un semplice rumore bianco [3]:

$$\begin{aligned} y(t) = \; & a_1 y(t-1) + a_2 y(t-2) + \cdots + a_{na} y(t-na) + \\ & + b_1 u(t-1) + b_2 u(t-2) + \cdots + b_{nb} u(t-nb) + \\ & + \xi(t) \end{aligned} \tag{2.8}$$

l'informazione relativa alla parte autoregressiva (cioè la parte che lega $y(t)$ al suo passato) è contenuta nel polinomio $A(z) = 1 - a_1 z^{-1} - \cdots - a_{na} z^{-na}$ mentre la parte esogena (cioè quella che lega l'uscita agli ingressi) viene descritta tramite $B(z) = b_1 + b_2 z^{-2} + \cdots + b_{nb}^{-nb}$, dove na e nb sono gli ordini delle due parti e rappresentano quanto il sistema viene influenzato dal passato.

Ne segue che la 2.7 può essere riscritta come: $A(z)y(t) = B(z)u(t-1) + \xi(t)$ e, di conseguenza, $G(z) = \frac{B(z)}{A(z)}$, $W(z) = \frac{1}{A(z)}$.

Infine, si introduce il vettore dei parametri $\theta = [a_1, \ldots, a_{na}, b_1, \ldots, b_{nb}]$.

ARMAX Rispetto a quanto fatto sopra, viene fatto lo stesso, ma questa volta, il residuo d'equazione assume un espressione più complessa fornendo maggiore flessibilità per quanto concerne la modellazione del disturbo [3]:

$$\begin{aligned} y(t) = \; & a_1 y(t-1) + a_2 y(t-2) + \cdots + a_{na} y(t-na) + \\ & + b_1 u(t-1) + b_2 u(t-2) + \cdots + b_{nb} u(t-nb) + \\ & + \xi(t) + c_1 \xi(t-1) + c_2 \xi(t-2) + \cdots + c_{nc} \xi(t-nc) \end{aligned} \tag{2.9}$$

Anche qui, $A(z) = 1 - a_1 z^{-1} - \cdots - a_{na} z^{-na}$, $B(z) = b_1 + b_2 z^{-2} + \cdots + b_{nb}^{-nb}$. Ma a questi, si aggiunge un nuovo polinomio $C(z) = 1 + c_1 z^{-1} + \cdots + c_{nc} z^{-nc}$.

Perciò, si ha che $G(z) = \frac{B(z)}{A(z)}$, $W(z) = \frac{C(z)}{A(z)}$,
mentre $\theta = [a_1, \ldots, a_{na}, b_1, \ldots, b_{nb}, c_1, \ldots, c_{nc}]$

Richiamando il vettore delle osservazioni $\phi(t)$ introdotto nella sezione 2.1, la 2.8 assume, quindi, la forma:

$$y(t) = \phi^T(t)\theta + \xi(t) \tag{2.10}$$

2.2.3 Stima dell'ordine a priori

Prima di passare al processo di identificazione, è bene comprendere, a priori, di quale complessità consta il sistema.

Per giungere a questo risultato ci si avvale di una particolare matrice costruita con i dati sperimentali [5]:

$$M(n) = \begin{pmatrix} y(n+1) & y(n) & \cdots & y(1) & u(n) & u(n-1) & \cdots & u(1) \\ y(n+2) & y(n+1) & \cdots & y(2) & u(n+1) & u(n) & \cdots & u(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ y(N) & y(N-1) & \cdots & y(N-n) & u(N-1) & u(N-2) & \cdots & u(N-2) \end{pmatrix} \tag{2.11}$$

Ora, quello che si vuole, è cercare il più piccolo n per cui l'equazione $y(t) = a_1 y(t-1) + \cdots + a_{na} y(t-na) + b_1 u(t-1) + \cdots + b_{nb} u(t-nb)$ sia soddisfatta. Ciò corrisponde alla ricerca di quell'n t.c. le colonne di $M(n)$ siano linearmente dipendenti mentre quelle di $M(k)$, non lo siano per ogni $k < n$.

Questo si traduce nel calcolare il rango della matrice $M(n)M(n)^T$ e, iterativamente, nel verificare la sua singolarità, comparandone il rango con $2n+1$: se esso è minore di $2n+1$ la matrice è singolare.
Nei fatti, però, la relazione che descrive $y(t)$ coincide con la 2.7, introducendo un disturbo e implicando che $M(n)M(n)^T$ non sia singolare per alcun n.

Per fortuna, viene in aiuto un' importante proprietà delle matrici singolari per cui 0 è autovalore di un endomorfismo se e solo se esso è singolare.

Perciò, $M(n)M(n)^T$ sarà tanto più vicina alla singolarità quanto più il suo autovalore minimo $\lambda_{min,n}$ sarà prossimo allo 0.

É chiaro, allora, che $\lambda_{min,n}$ costituisce una misura dell'aderenza ai dati. Come introdotto durante la spiegazione del procedimento di stima, però, all'aumentare della complessità, si ha un incremento dell'aderenza. Ne consegue che al crescere di n, $\lambda_{min,n}$ decresce, rendendo inutili tutte le considerazioni appena riportate.

Per evitare che ciò avvenga, si calcola l'autovalore minimo per ordini crescenti e si sceglie l'ordine n per cui la differenza $|\lambda_{min,n} - \lambda_{min,n-1}|$ risulta massima.

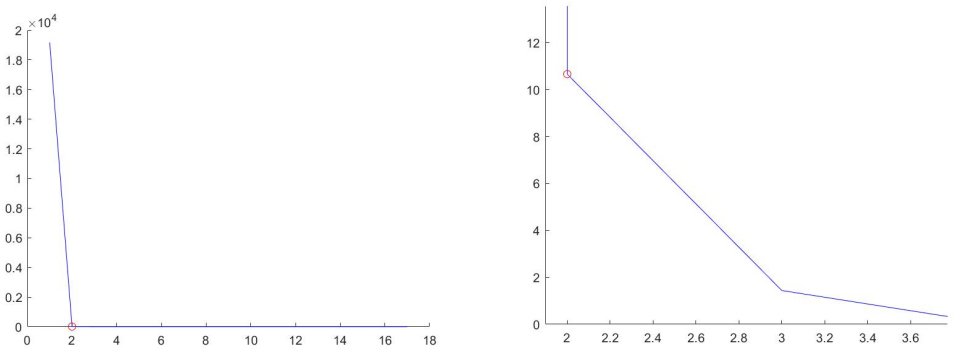
Implementazione pratica Per la valutazione dell'ordine relativa ai dati che mi sono stati forniti è stato utilizzato un programma MATLAB scritto insieme al collega Angelo D'Agostino Bonomi durante il corso di Modellistica e Identificazione dei processi dinamici.

Il codice segue di pari passo i fondamenti teorici introdotti sopra, salvo aggiungere un criterio d'arresto con una tolleranza di 10^{-4}.

```
%inizializzazione
u = ElectricIntensifierData.Position;
y = ElectricIntensifierData.Pressure;

N=length(u);
i=1;
n=1;
autovalori = 0;
tol = 1e-4; %criterio d'arresto dell'algoritmo

while i==1

    %Popolazione matrice dati sperimentali
    v_y=[n+1:N];
    v_u=[n:N-1];
    My=y(v_y');
    Mu=u(v_u');
    a = 1;

    for j=n:-1:1
        v_y = v_y-a;
        My=[My y(v_y')];
    end

   for j=n-1:-1:1
       v_u=v_u-a;
       Mu=[Mu u(v_u')];
   end

   M=[My Mu];
   [righe, colonne]= size(M);

    if (rank(M) == colonne)

        i=1;
        n=n+1;

    else i=0;
    end


   if i==1

       Mtot= M'*M;
       minimo = Autovalore_minimo(Mtot);

       if(minimo <= tol)
           i = 0;
       end

       %inizializzazione del vettore di autovalori (1a iterazione)
        if autovalori == 0
           autovalori = minimo;

       %aggiornamento vettore di autovalori
        else autovalori = [autovalori minimo];

        end
    end
end

%calcolo della variazione massima tra autovalori di iterazioni successive
[salto, idx] = max(autovalori - [autovalori(2:length(autovalori)) 0]);
idx=idx+1;

%visualizzazione del valore degli autovalori per ogni iterazione
hold on
plot (autovalori,'b-')
plot (idx,autovalori(idx),'ro')
hold off
fprintf("l'ordine minimo stimato da cui iniziare la modellazione è: %i\n", idx);

%-------------------------------------------------------------------------%

%funzione che calcola l'autovalore minore presente nella matrice considerata
function v = Autovalore_minimo(Matrice)
autovalori = eig(Matrice);
autovalori= abs(autovalori);
v = min(autovalori);
end
```


Figura 2.8: Codice implementativo valutazione a priori dell'ordine

Ordine stimato Applicando il programma ai dati forniti, si stima che n debba aggirarsi attorno a 3, perché se è vero che il maggior salto si presenta in corrispondenza di $n = 2$, come mostrato nel seguente grafico che descrive l'andamento dell'autovalore minimo all'aumentare dell'ordine, è vero anche che $\lambda_{min,2}$ si discosta in maniera ritenuta significativa dallo 0.



(a) Vista completa (b) Ingrandimento attorno a $n = 2$

Figura 2.9: Illustrazione ordine stimato

2.3 ARX 331 BIS - il miglior modello, con qualche compromesso

Per giungere alla determinazione del miglior modello lineare, per il sistema in esame, è stato utilizzato il "System Identification toolbox", un particolare toolbox di MATLAB. Questo, tramite una semplice interfaccia permette la pre-elaborazione dei dati ottenuti attraverso il campionamento, la stima automatica dei parametri di diverse tipologie modelli (tra cui quelli ad errore di equazione) e fornisce importanti strumenti utili allo studio della bontà, oltre che alla scelta, dei modelli ottenuti.
Durante la trattazione pratica del procedimento di stima, verrà presentata solo l'identificazione basata sull'acquisizione 4.

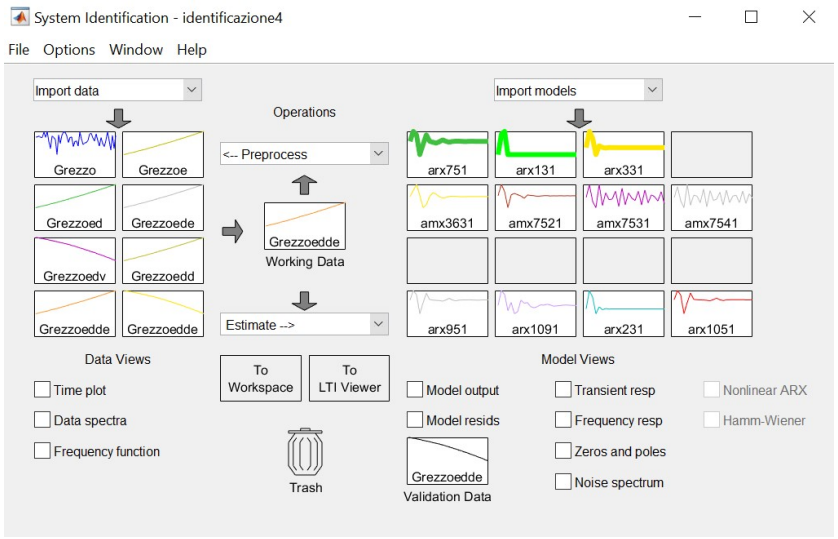


Figura 2.10: Graphic User Interface del toolbox

2.3.1 System Identification toolbox

Saranno, qui, illustrati tutti i passaggi necessari per l'identificazione, elaborati attraverso il system identification toolbox. Il primo passo è chiaramente quello di importare la coppia ingressi/uscite. Perciò, dopo la pulizia dei dati, per ogni acquisizione, sono state importate posizione e pressione.

Considerazioni preliminari deducibili dai dati Confrontando l'andamento dell'ingresso, al variare del tempo con quello dell'uscita, si può dedurre che si tratta di un sistema semplice, il cui comportamento è lineare. Ciò si evince dal fatto che l'uscita segue perfettamente l'ingresso, ad eccezione di qualche caduta, che ricorre nel momento in cui la posizione assume un valore costante. La natura di questi cali è, probabilmente, da imputare alla presenza di perdite, le quali assumono un valore significativo in corrispondenza di elevati livelli di pressione.
Inoltre, considerate le maggiori ampiezze raggiunte, si prevede un guadagno rilevante in uscita.

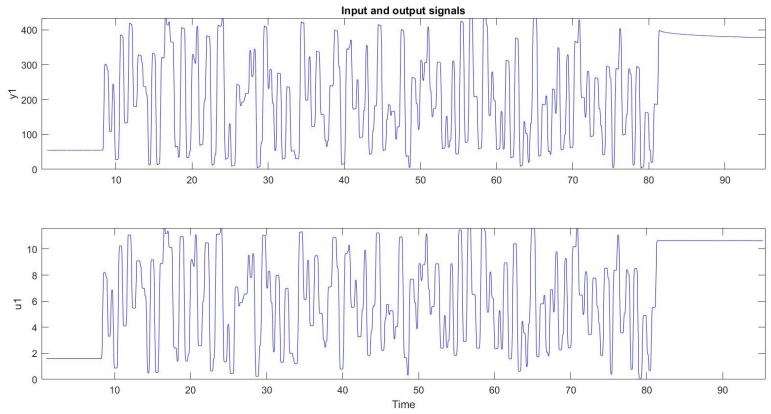


Figura 2.11: Rappresentazione, nel dominio del tempo, dell'andamento di posizione (u1) e pressione (y1)

Pre-elaborazione dei dati Come da prassi, sono stati rimossi:

- La media per evitare che il punto d'equilibrio a cui essa corrisponde rientri nelle equazioni del modello
- I trends (tendenze particolari che si evidenziano nell'evoluzione temporale di una variabile) per evitare che alcuni comportamenti predominanti ne oscurino altri

Infine, i campioni sono stati divisi in 2 parti uguali in modo da utilizzare la prima parte per la ricerca del modello e la seconda parte per la cross validazione [3].
Notare che, non sempre eliminare media e trends porta a risultati migliori, può accadere, infatti, che la loro cancellazione precluda la descrizione da parte del modello di importanti comportamenti del sistema reale.

Stima modelli a errore d'equazione Si prosegue alla stima dei candidati (i.e. la stima di θ) dei quali dovrà essere successivamente effettuata un'attenta selezione. Di norma, prima si cerca di ricondurre il sistema a un ARX, poi, se i risultati non sono soddisfacenti, si tenta la strada degli ARMAX a partire dagli ordini del miglior ARX ottenuto precedentemente.

Nella sezione 2.1 si era parlato di come fosse possibile ricavare θ minimizzando il funzionale errore quadratico medio $J_N(\theta)$. Senza entrare eccessivamente nel dettaglio, attraverso le espressioni associate al predittore $\hat{M}(\theta)$ e al modello $M(\theta)$, si riformula l'errore di predizione nel modo seguente:
$\xi(t) = y(t) - \phi^T(t)\theta$ [6].

Ora è sufficiente sostituire $\xi(t)$ all'interno di $J_N(\theta)$, calcolarne la derivata e porla uguale a 0.
Si ottengono, così, $na+nb$ equazioni dette *Equazioni Normali*, della forma[5]:

$$\sum_{\tau}^{N} \phi(t)y(t) = (\sum_{\tau}^{N} \phi(t)\phi(t)^T)\theta \tag{2.12}$$

Osservato che $S(N) = \sum_{\tau}^{N} \phi(t)\phi(t)^T$, il punto di minimo è dato da:

$$\theta = S(N)^{-1} \sum_{\tau}^{N} \phi(t)y(t) \tag{2.13}$$

Ne segue, che, come già introdotto nella trattazione degli ingressi persistentemente eccitanti, c.n.e.s. affinché $\theta_{min,N}$ sia unico, è che la matrice $S(N)$ sia non singolare.

Il calcolo di θ di cui si è appena discusso è lo stesso su cui si basa il toolbox quando si preme sul tasto "Estimate...", presente nella GUI, riferendosi ai modelli polinomiali.

La stima ha prodotto il seguente risultato: ogni rettangolo dell'istogramma altro non è che un ARX e più ci si sposta verso destra, più la complessità cresce. I rettangoli colorati, invece, come scritto nella legenda, rappresentano la miglior scelta secondo gli indici MDL, AIC e Best Fit.

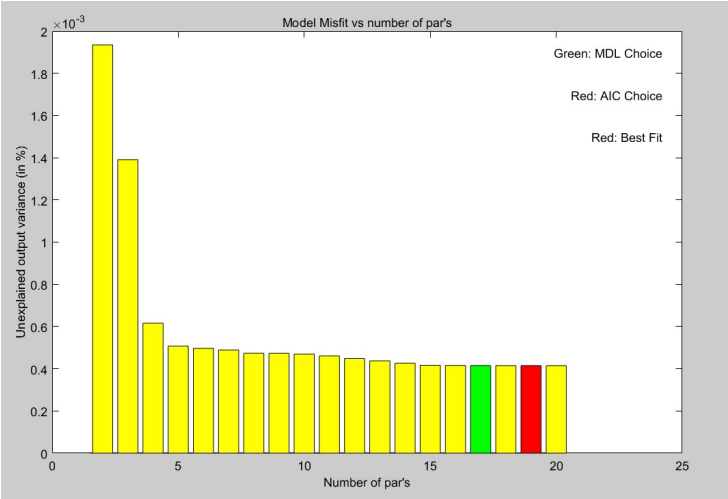


Figura 2.12: Valori di $J_N(\theta)$ al crescere dell'ordine n

Di questi, sono stati prelevati i modelli in cui si presenta il maggior salto e quelli indicati dagli indici.

Selezione dei modelli con caratteristiche migliori Si riportano i diversi criteri che hanno condotto alla scrematura della popolazione iniziale di modelli, fino ad ottenere il miglior candidato.

- *Fitting* – Sono stati eliminati tutti i modelli che dimostravano un livello di accuratezza in termini di simulazione dell'uscita troppo basso.

- *Stabilità* – Osservato il grafico raffigurante i poli e gli zeri di ogni modello sono stati eliminati tutti quelli instabili, cioè quelli per cui almeno un polo uscisse dalla circonferenza unitaria. Inoltre, sono stati eliminati a priori tutti quei modelli per cui si verificasse la condizione di fisica irrealizzabilità: $nb > na$. [4]

- *Analisi dei residui* – È stato esaminato l'errore residuale del modello. Purtroppo, a causa dell'inadeguatezza del segnale in ingresso, l'autocorrelazione di ogni candidato fuoriesce dall'intervallo di confidenza, rendendo praticamente inutile questo tipo di disamina.

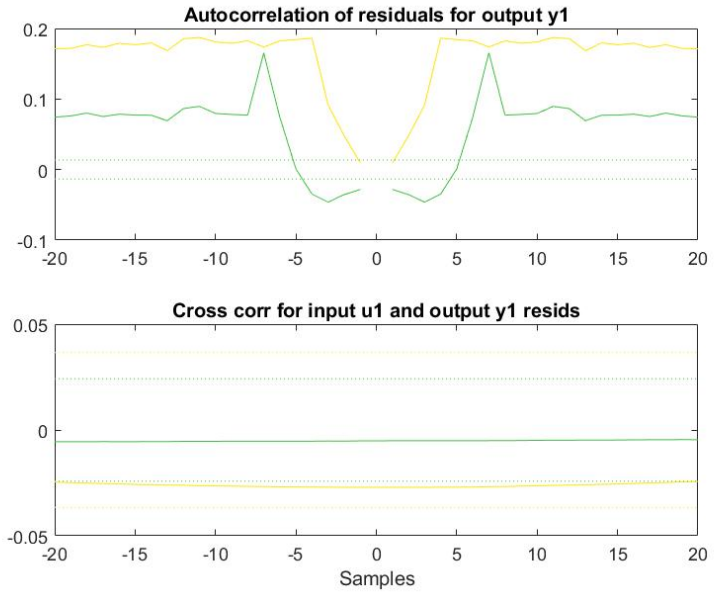


Figura 2.13: Autocorrelazione e Cross-Correlazione ARX 751 e 331

[4] $G(z) = \frac{B(z)}{A(z)}$, risulterebbe impropria.

Va meglio per quanto riguarda la cross-correlazione: per la maggior parte dei modelli, l'uscita $y(t)$ generata dall'ingresso $u(t-k)$ è descritta propriamente per ogni $k \in [-20, 20]$.

- Sovrapposizione e Complessità – Sono stati eliminati tutti quei modelli che mostrassero evidenti sovrapposizioni polo-zero, indici di possibili cancellazioni e, quindi, di una complessità troppo elevata.
  In accordo con il "Parsimony Principle" [6], lo stesso è stato fatto per tutti i candidati il cui ordine esageratamente elevato non comportasse miglioramenti di accuratezza significativi.

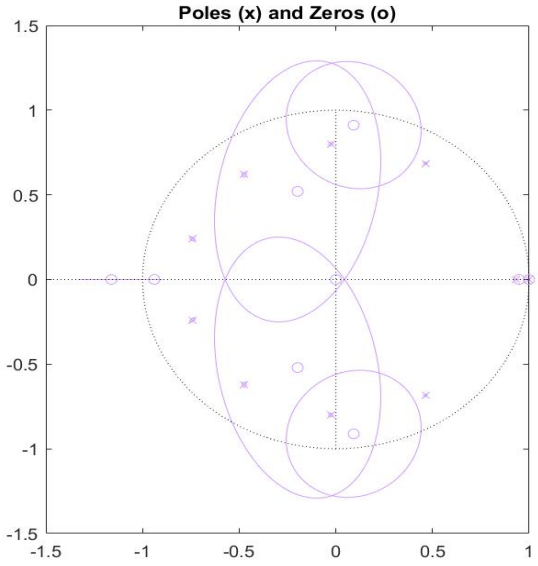
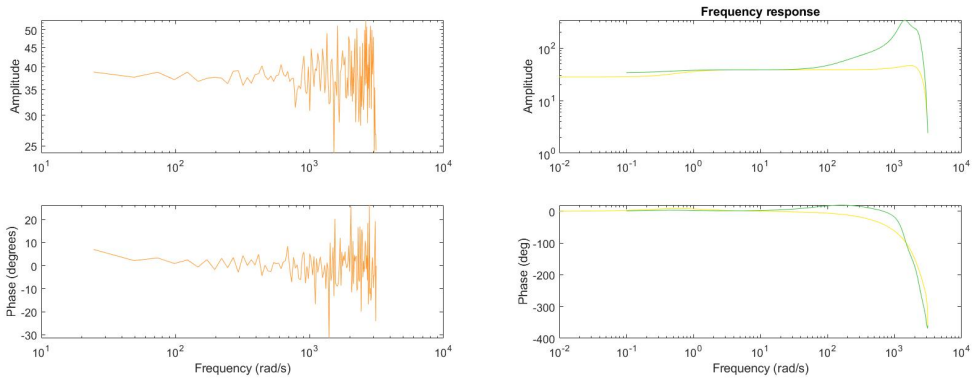


Figura 2.14: Esempio di possibili sovrapposizioni polo-zero come mostrano le intersezioni degli intervalli di confidenza che attorniano poli e zeri (ARX 1091)

Dopo questa scrematura, relativa all'acquisizione 4, restano 2 possibili candidati: ARX 331 e ARX 751. Per compiere una scelta definitiva si sono fatte le seguenti valutazioni:

- *Somiglianza della risposta in frequenza* – Comparando il diagramma di Bode dei due ARX con quello dei dati campionati si nota come il 331 si comporti meglio del 751, in quanto l'ampiezza si aggira attorno ai 40 dB senza mai superare i 50dB, proprio come accade per la risposta in frequenza dei dati campionati.



(a) Dati campionati

(b) ARX 751, in giallo; ARX 331, in verde

Figura 2.15: Comparazione delle risposte in frequenza

- *Miglior compromesso Fitting/Ordine* – Qualche pagina sopra si è evidenziato come il sistema paia essere molto semplice, fornendo un ulteriore motivazione (oltre a quella di una possibile sovraparametrizzazione) per contenere la complessità.

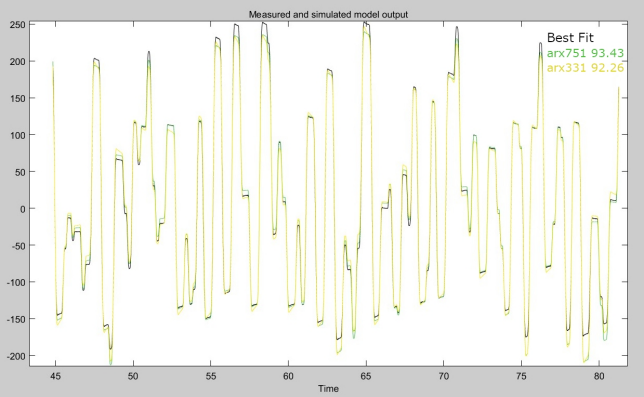


Figura 2.16: Uscite simulate comparate con l'uscita reale. In alto a destra, percenutale di fitting

Pare già ovvia, allora, la scelta dell'ARX 331 in favore dell' ARX 751, la quale viene giustificata anche da una differenza esigua in termini di accuratezza. Per tutti gli altri dati disponibili è stato eseguito grosso modo lo stesso procedimento ma nessun set ha permesso di ottenere risultati simili a quelli raggiunti con l'acquisizione 4. In ogni caso, questi set sono stati comunque utili per testare il comportamento del modello in diverse situazioni.

Per completezza, sono stati eseguiti dei tentativi di stima polinomiale, anche con la classe ARMAX, in prossimità dell'ordine 3. Tuttavia, l'unico effetto che se n'è ricavato è stato un aumento della complessità.

Presentazione dell' ARX 331: Il modello ottenuto è un ARX a tempo discreto di equazione caratteristica, $A(z)y(t) = B(z)u(t) + e(t)$
Dove:

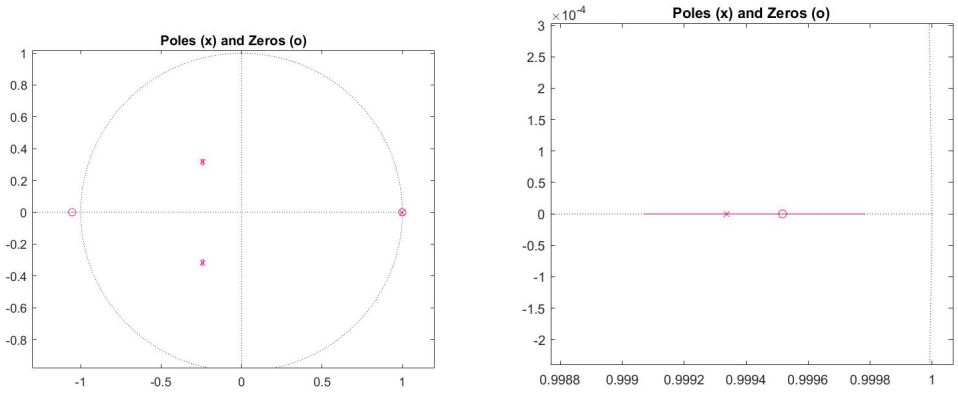
$$A(z) = 1 - 0.5166(\pm 0.0052)z^{-1} - 0.3248(\pm 0.0056)z^{-2} - 0.1575(\pm 0.0052)z^{-3}$$
$$B(z) = 30.97(\pm 0.1814)z^{-1} + 1.66(\pm 0.0840)z^{-2} - 32.6(\pm 0.1809)z^{-3}$$

Di ordine pari a 3 e, nello specifico: $na = 3; nb = 3; nk = 1$ [5].

[5] nk, rappresenta il ritardo: k corrisponde al primo coefficiente b_k, della parte esogena, diverso da zero

Caratteristiche del modello:

- *Incertezza dei parametri* - la varianza di ogni parametro è sempre di alcuni ordini di grandezza inferiore al parametro stesso, quindi il modello si dimostra solido poiché non sono possibili cancellazioni.
- *Risposta in frequenza* – si è già discussa la somiglianza con il diagramma di bode ricavato dall'acquisizione.
- *Fitting* – L'uscita simulata si comporta molto bene, raggiungendo una percentuale di fitting, rispetto all'uscita misurata nell'acquisizione 4, del 92.26% [6]
- *Poli e zeri* – Il modello risulta, ovviamente, stabile. Come in quasi tutti i modelli però, si presenta una possibile sovrapposizione polo-zero nelle vicinanze di 1 forse dovuta a errori numerici o di misura.



(a) (b)

Figura 2.17: Poli e zeri ARX 331

[6]Il system identification toolbox ricava la misura di quanto l'uscita predetta dal modello si avvicini a quella misurata attraverso il calcolo dell'NRSME (normalized-root-mean-square deviation): $\frac{\sqrt{J_N(\theta)}}{\mathbf{E}\{y(t)\}}$. Più l'NRSME è alto, migliore è la predizione ottenuta.

2.3.2 ARX 221 - Cancellazione manuale sovrapposizione polo-zero

Vista la frequenza del presentarsi di questa sovrapposizione nei modelli, ipotizzando che possa derivare da un errore sistematico numerico o di misura, si è deciso di provare a eliminarla manualmente dividendo i polinomi A(z) e B(z) per i corrispettivi poli e zeri.

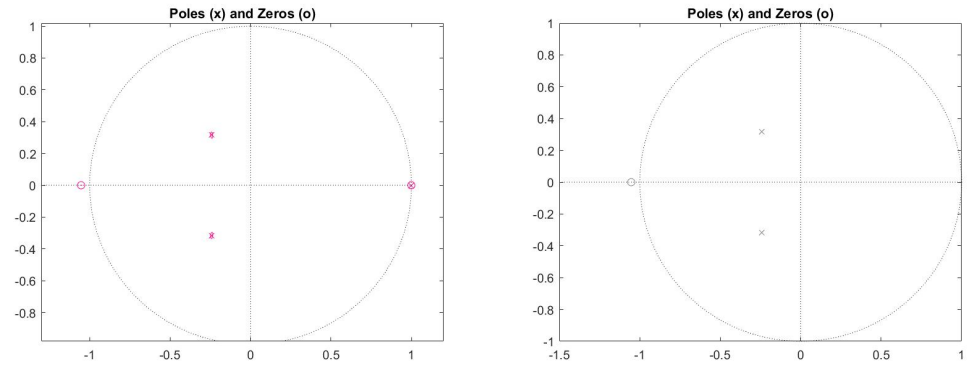
Quello che si ottiene è un ARX 221 a tempo discreto descritto dai polinomi:

$$\frac{A(z)}{1-0.9993z^{-1}} = \frac{1-0.5166z^{-1}-0.3248z^{-2}-0.1575z^{-3}}{1-0.9993z^{-1}} = -1-0.4844z^{-1}-0.1597z^{-2}$$

$$\frac{B(z)}{1-0.9995z^{-1}} = \frac{30.97z^{-1}+1.66z^{-2}-32.6z^{-3}}{1-0.9995z^{-1}} = -30.9741z^{-1}-32.619z^{-2}$$

Confronto tra ARX 331 e ARX 221: Si riportano nelle prossime pagine alcune immagini relative alle caratteristiche salienti dei due modelli, a sinistra inserirò figure sempre appartenenti all'ARX 331 mentre a destra quelle appartenenti all'ARX 221.

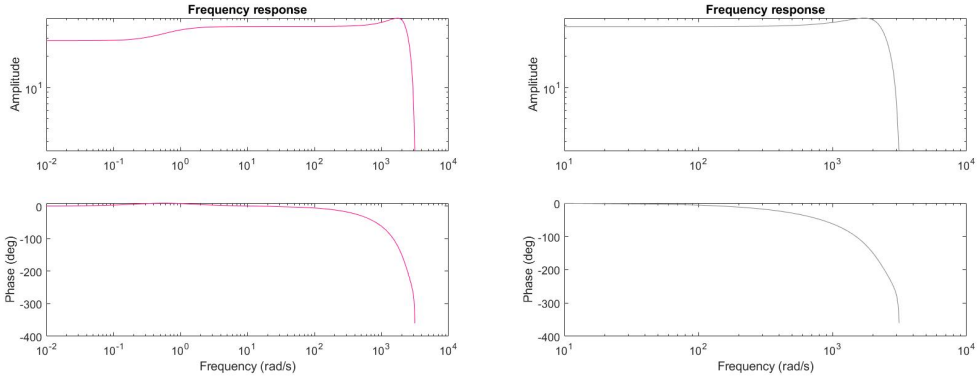
- *Poli e zeri* - I grafici confermano i risultati sperati, l'ARX 221 è una versione dell'ARX 331 senza sovrapposizione.



(a) (b)

Figura 2.18: Poli e zeri ARX 331, ARX 221

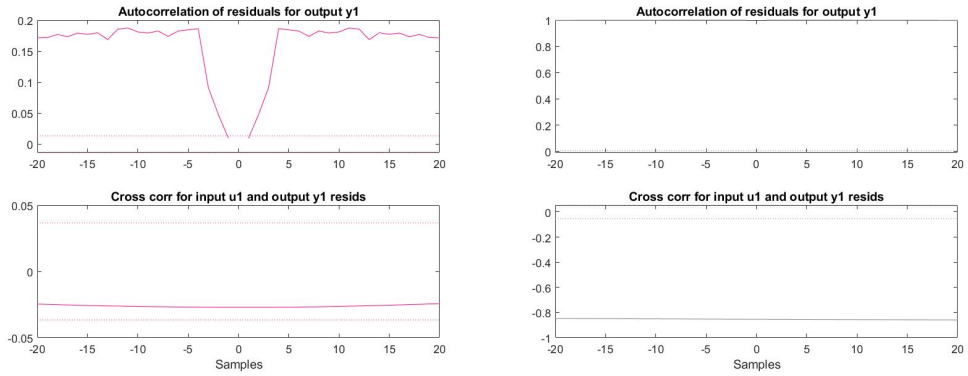
- *Risposte in frequenza* - Apparentemente molto simili, tuttavia si nota come venga meno l'informazione per frequenze minori di 10^1 rad/s.



(a) (b)

Figura 2.19: Risposte in frequenza ARX 331, ARX 221

- *Residui* - la semplificazione ha persino peggiorato le caratteristiche, già pessime, dei residui.



(a) (b)

Figura 2.20: Autocorrelazione, Cross-Correlazione ARX 331, ARX 221

- *Fitting* - l'accuratezza dell'uscita originata dal nuovo modello relativa ai dati dell'acquisizione 4 peggiora leggermente scendendo dal 92.26% al 91.69%. Inoltre la caduta di pressione dopo il raggiungimento del setpoint non viene più simulata.

- *Accuratezza dei modelli in corrispondenza di altri set di dati* - È qui che la mossa fatta appena sopra acquisisce un senso. Il test di fitting eseguito su acquisizioni diverse da quella su cui si basano i modelli rende totalmente inadeguato l'ARX 331 rispetto all'ARX221.

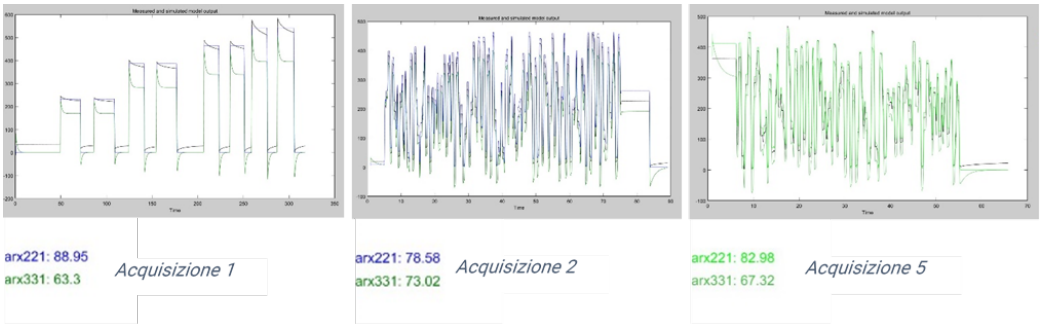


Figura 2.21: Accuratezza ARX 331 e ARX 221 rispetto ad acquisizioni diverse dall'acquisizione 4

Tuttavia, l'ARX pare essere una semplificazione troppo accentuata. Il fatto che la caduta di pressione non venga replicata minimamente induce a pensare che si potesse fare di meglio.

Perciò, si è provato ad eseguire un'identificazione basata sempre sull'acquisizione 4, ma senza sottrarre media e trend ipotizzando che avendo fatto ciò precedentemente si erano trascurati comportamenti importanti del sistema. Il risultato di questa operazione è ancora una volta un ARX 331, che, per convenzione, è stato chiamato ARX 331 BIS.

2.3.3 Presentazione ARX 331 BIS

Il modello è molto simile al 331 presentato sopra, gode di una risposta in frequenza fedele, stabilità, nessuna sovrapposizione (se non quella in prossimità di 1) ma si comporta di gran lunga meglio nella maggior parte delle situazioni.

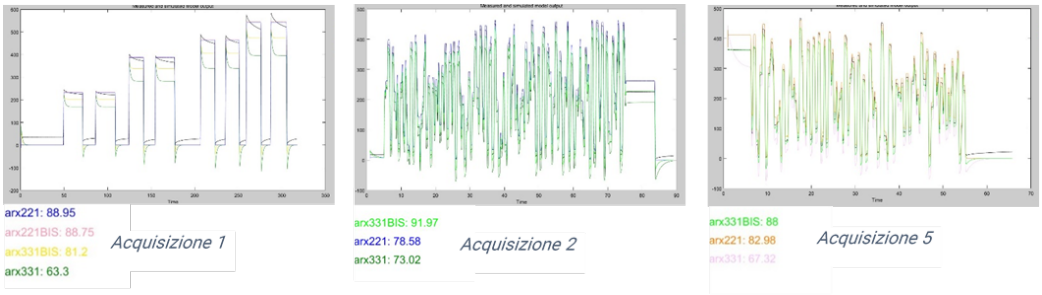


Figura 2.22: Accuratezza ARX 331 e ARX 331BIS rispetto ad acquisizioni diverse dall'acquisizione 4

È stato replicato il tentativo di eliminare manualmente la sovrapposizione, ma visto che, anche in questo caso, la caduta di pressione continuava a non essere modellata, si è decretato l'ARX 331 BIS come modello definitivo del sistema.

Parametri ARX 331 BIS:

$$A(z) = 1 - 0.4927(\pm 0.0045)z^{-1} - 0.3317(\pm 0.0048)z^{-2} - 0.1735(\pm 0.0045)z^{-3}$$
$$B(z) = 31.96(\pm 0.1611)z^{-1} + 1.65(\pm 0.07927)z^{-2} - 33.54(\pm 0.1609)z^{-3}$$

La scelta del 331 BIS è stata appoggiata anche dai tutor aziendali, che pur non del tutto contenti del risultato finale hanno riconosciuto l'impossibilità di fare di meglio vista l'inadeguatezza degli ingressi utilizzati per i campionamenti.

Capitolo 3

CONTROLLO

3.1 PID - Proporzionale, Integrale, Derivativo

Ancora oggi, in ambito industriale, la tipologia di controllori più utilizzata è quella dei PID. Questa grande diffusione si deve alla loro semplicità di determinazione (anche qualora il modello matematico del processo da controllare non sia del tutto preciso) e alla loro versatilità che consente una regolazione soddisfacente per un'ampia gamma di processi [4].

Inoltre, la taratura dei parametri che li caratterizzano, è resa ulteriormente semplificata dalle tecniche automatiche che sono state sviluppate nel corso degli anni. In questa tesi se ne fa ampio utilizzo come descritto nella sezione 3.2.1.

Struttura Concettualmente, la struttura dei PID è basata su alcune considerazioni empiriche riferite ai contributi necessari per annullare l'errore tra uscita e riferimento $e(t) = r(t) - y(t)$. Di seguito si elencano le suddette considerazioni riguardo ai 3 contributi:

- *Proporzionale* - Come è intuitivo pensare, la variabile di controllo u dovrà variare in modo proporzionale rispetto all'errore e così da imporre un'azione più accentuata se quest'ultimo è elevato, oppure più lieve se l'uscita è prossima al riferimento.

- *Integrale* - Per migliorare il comportamento asintotico, e, in particolare, far sì che l'errore si annulli asintoticamente a fronte di riferimenti o disturbi additivi costanti, si introduce un contributo proporzionale all'integrale di e. In questo modo, si dispone di un termine che attua una regolazione incentrata sulla storia passata dell'errore.

- *Derivativo* - l'ultimo contributo, invece, si occupa di anticipare l'andamento di e attraverso il calcolo della sua derivata. Di norma, questo termine è aggiunto nei casi in cui non vi siano problemi di stabilità, per incrementare la banda passante.

La somma dei 3 contributi appena presentati fornisce la legge di controllo caratteristica di un PID:

$$u(t) = K_P e(t) + K_I \int_{t_0}^{t} e(\tau) d\tau + K_D \frac{de(t)}{dt} \tag{3.1}$$

dove K_P, K_I e K_D sono costanti positive o nulle, nonché i coefficienti corrispondenti (rispettivamente) all'azione proporzionale, integrale e derivativa.

É chiaro che una tale relazione non può che portare a un regolatore ideale, in quanto, applicando la trasformata di Laplace alla funzione di trasferimento associata, si osserva che l'azione derivativa introduce uno zero, rendendo il sistema improprio.

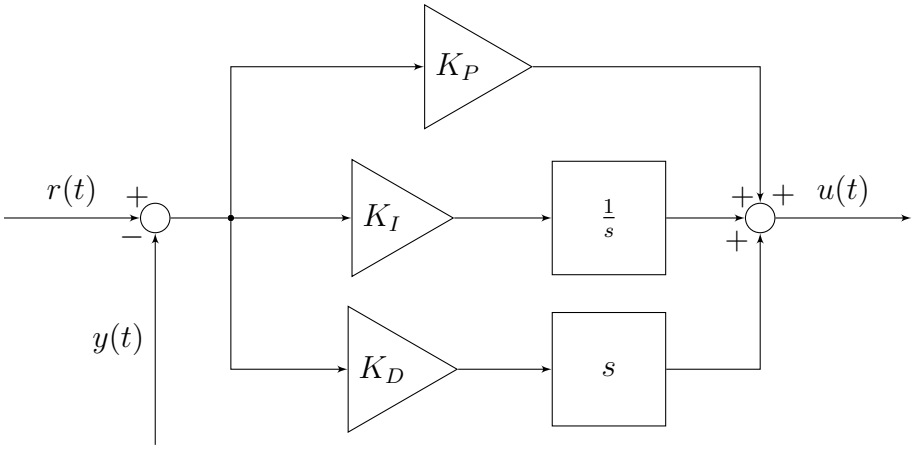


Figura 3.1: Schema a blocchi di un PID ideale

Per rendere il PID realizzabile si introduce un polo esterno alla banda di frequenze di interesse per il controllo. In questo modo la funzione di trasferimento diventa:

$$R_{PID}(s) = K_P + \frac{K_I}{s} + \frac{K_D s}{1 + \frac{K_D}{K_P N} s} \tag{3.2}$$

Dove $s = -N\frac{K_P}{K_D}$ è il polo introdotto per le condizioni di fisica realizzabilità.

3.1.1 Saturazione e Windup

Idealmente, l'azione di controllo potrebbe raggiungere qualsiasi valore, cosa che, nella realtà, ovviamente non è possibile. L'attuatore, infatti è sottoposto ad alcuni limiti fisici che portano $u(t)$ alla saturazione.

Perciò, se indichiamo con u(t) l'uscita del controllore e con m(t) la posizione generata dall'azionamento che muove il moltiplicatore elettrico si ha:

$$m(t) = \begin{cases} -u_M \ , u(t) < -u_M \\ u(t) \ , |u(t)| \leq u_M \\ u_M \ , u(t) > u_M \end{cases} \tag{3.3}$$

Il risultato è un comportamento non lineare che inficia sulle prestazioni del sistema di controllo.

In particolare, quello che avviene è che se la legge di controllo oltrepassa i limiti fisici dell'attuatore, questo sarà portato a esercitare un'azione costante pari al valore di saturazione. Allo stesso tempo, però, l'errore non sarà ancora nullo, portando il termine integrale a crescere sempre più.

Se il sistema è stabile, dopo un certo intervallo di tempo, l'errore diminuirà fino ad annullarsi e, con esso, ci si aspetta che anche $u(t)$ diminuisca. Tuttavia, l'azione dell'attuatore rimarrà costante fino al momento in cui il termine integrale non si sarà scaricato abbastanza da far scendere il forzamento al di sotto del livello di saturazione. Il fenomeno appena descritto prende il nome di carica integrale o "integral wind-up" [4].

Per evitare che il forzamento attuato dall'azionamento risenta di integral wind-up esistono alcune tecniche di desaturazione che giocano tutte sull'alimentare il regolatore anche con il segnale a valle della saturazione.

Tecniche anti wind-up Tra le diverse tecniche si annoverano quella di "Integrazione Condizionale" e della "Back Calculation" che saranno illustrate qui di seguito:

- *Integrazione Condizionale* - Questo metodo consiste nel fermare l'azione integrale nel momento in cui $u(t)$ raggiunge i limiti di saturazione. [2]

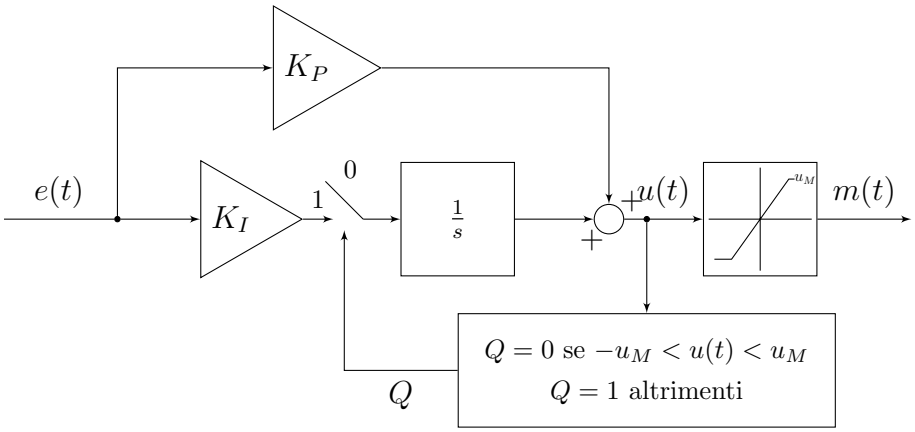


Figura 3.2: Schema rappresentativo di un PI con Integrazione Condizionale

Come mostra la figura 3.2, viene effettuato un controllo sull'output generato dal controllore: se questo rientra nei limiti di saturazione, l'errore viene correttamente inviato al blocco integratore, altrimenti è settato a 0 annullando, di fatto, il contributo del termine integrale. In questo modo $u(t)$ dipende solamente dal termine proporzionale che non soffre di wind-up.

- *Back Calculation* - Diversamente da quanto accade nella tecnica precedente, qui viene prelevata l'azione di controllo prima e dopo la saturazione. Viene applicata una sottrazione per determinare l'errore di saturazione $e_s(t) = m(t)–u(t)$ (che quindi è un numero negativo), il cui valore viene pesato attraverso il guadagno K_b e sommato all'errore $e(t)$ che rientra nel calcolo del termine integrale. La funzione della quale è calcolato l'integrale risulta quindi: $e_i = K_I e + K_b e_s$.

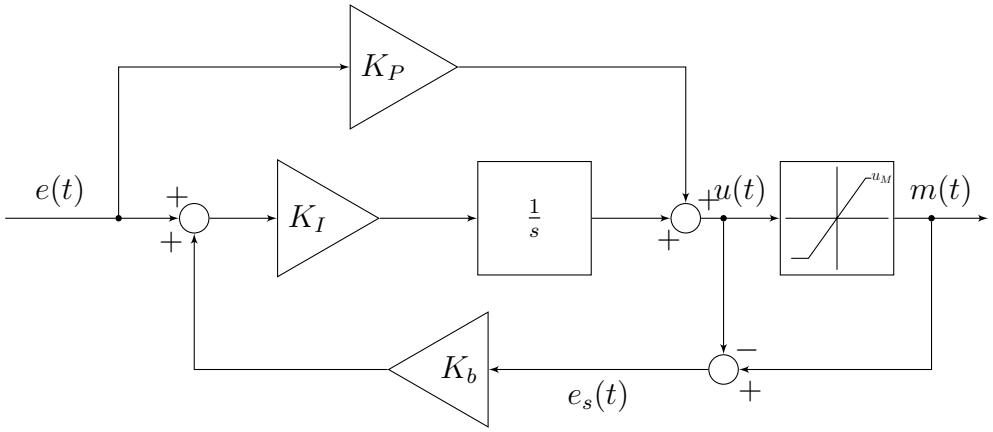


Figura 3.3: Schema rappresentativo di un PI con Back Calculation

In questo modo, se $u(t)$ eccede i limiti imposti dalla saturazione, l'errore $e_s(t)$ riportato a monte riduce l'azione integrale facendo rientrare il segnale di controllo nel suo comportamento lineare.

3.2 Modello Simulink

Sono appena state illustrate le caratteristiche di un generico regolatore PID e, come è intuibile, il motivo che vi risiede dietro è che proprio un regolatore di questo tipo sarà utilizzato per controllare il processo, oggetto di questa tesi.

Prima di procedere alla sintesi del PID, però, occorre introdurre l'ambiente dentro il quale ciò avverrà. Si parla, quindi, di MATLAB Simulink, un programma di modellazione grafica, ormai noto ai più, che permette di eseguire simulazioni e sviluppare algoritmi di controllo in modo sicuro ed efficiente.

Il primo passo della modellazione Simulink del processo (attuatore) è stato quello inserire l'ARX 331 BIS sotto la forma di una funzione di trasferimento all'interno del progetto.

Considerato che $A(z)y(t) = B(z)u(t) + e(t)$, allora

$$F(z) = \frac{y(t)}{u(t)} = \frac{B(z)}{A(z)} = \\ = \frac{31.96(\pm 0.1611)z^{-1} + 1.65(\pm 0.07927)z^{-2} - 33.54(\pm 0.1609)z^{-3}}{1 - 0.4927(\pm 0.004565)z^{-1} - 0.3317(\pm 0.004851)z^{-2} - 0.1735(\pm 0.004568)z^{-3}}$$

e quindi, raccogliendo z^{-3} al numeratore e al denominatore, si ottiene:

$$F(z) = \frac{31.96(\pm 0.1611)z^2 + 1.65(\pm 0.07927)z - 33.54(\pm 0.1609)}{z^3 - 0.4927(\pm 0.004565)z^2 - 0.3317(\pm 0.004851)z - 0.1735(\pm 0.004568)}$$

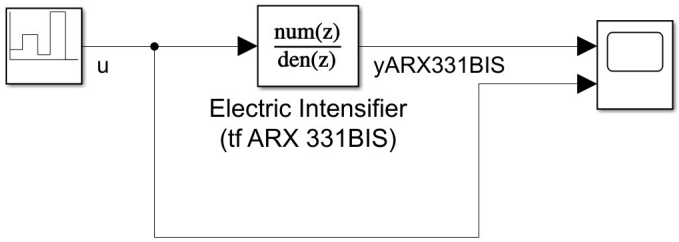


Figura 3.4: Modello Simulink del processo

Per accertarsi che tutto fosse corretto è stata inviata una sequenza di gradini (tramite l'apposito blocco) di ampiezze 4, 10, 2, 20, 0, che hanno confermato che la funzione di trasferimento si comporta in modo conforme al processo reale, come mostra il grafico riportato qui di seguito:

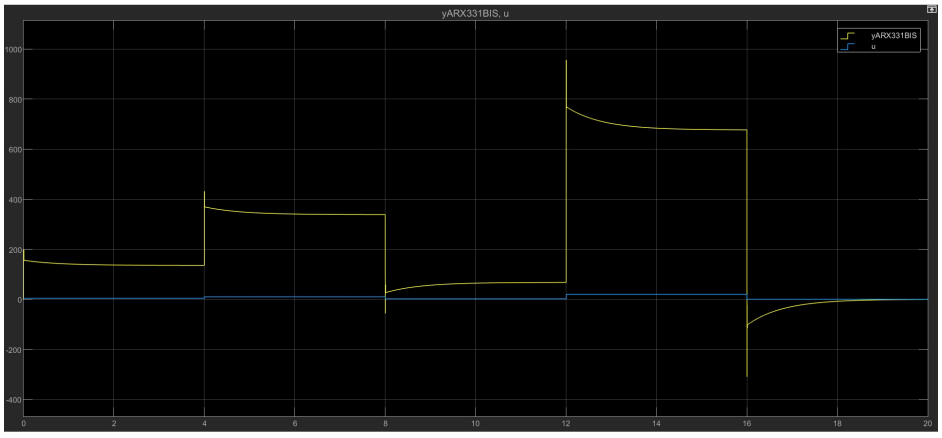


Figura 3.5: Rappresentazione ingresso(BLU)/uscite(GIALLO) del processo simulato

Notare che il fondo scala, in posizione, dell'attuatore (20 mm) corrisponde a circa 700 Bar come da specifica e che la caduta di pressione si manifesta correttamente.

3.2.1 PID tuning

Si è già discusso delle grandi potenzialità, di cui, software come MATLAB e Simulink sono dotati e che ne hanno fatto materia e principio fondante del tirocinio da cui ha origine questa tesi.

In questa sezione viene presentata una delle funzioni che rendono Simulink tanto utile: la sintesi automatica dei controllori PID. Esso fornisce un ottimo sistema di taratura dei PID per i quali risulta necessario solamente introdurre il blocco "Discrete PID controller, PID(z)", collegarlo alla funzione di trasferimento e disegnare l'anello di retroazione. A questo punto, settato il tempo di campionamento a 1 ms, in coerenza con gli altri blocchi, è sufficiente premere il tasto "tune" per determinare i parametri in base alla prontezza desiderata.

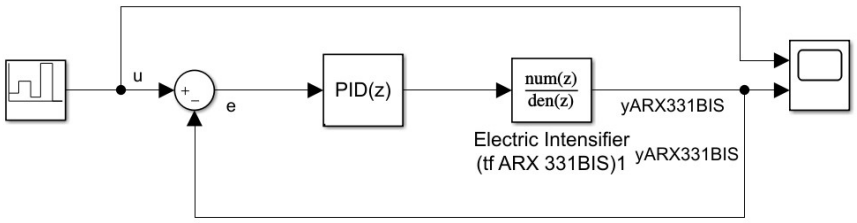


Figura 3.6: Modello Simulink del PI associato al processo

I risultati sono ottimi, il software, pur avendo la possibilità di introdurre anche il termine derivativo, non lo ha ritenuto necessario giudicando il PI un controllore già sufficiente per gestire la dinamica del sistema.

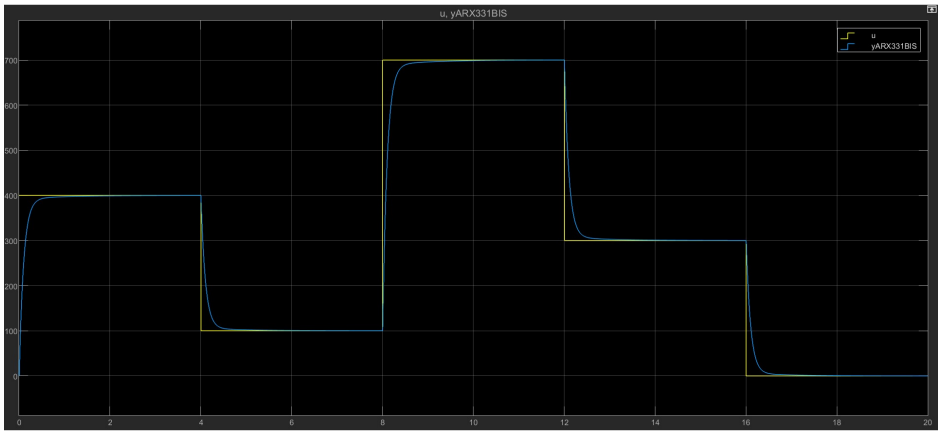


Figura 3.7: Uscita controllata "yARX331BIS" in relazione all'uscita desiderata "u"

Parametri: Il software ha stimato i seguenti parametri come ottimi:

- $K_P = 0.000128401570326013$
- $K_I = 0.256803140652026$
- $t_s = 0.228$ s

Implementazione dell'anti wind-up: Per sopperire al problema della scarica integrale, sono stati introdotti all'interno del blocco PID i limiti di saturazione $[0, 20]$ mm in conformità con le specifiche fornite dai datasheet e con i valori massimi stabiliti perché il banco lavori in completa sicurezza.

Simulink permette di implementare due tecniche di desaturazione: Back Calculation e Clamping (una rivisitazione più complessa dell'integrazione condizionale). Delle due è stata scelta la prima, lasciando K_b al valore di default ($K_b = 1$), riservando la possibilità di cambiarlo nel caso in cui si rivelasse incorretto.

Esternamente, il modello Simulink si presenta identico al precedente salvo inserire un simbolo che indica la saturazione all'interno del blocco PID(z). Internamente, invece, viene reso attivo il blocco che si occupa dell'anti wind-up.

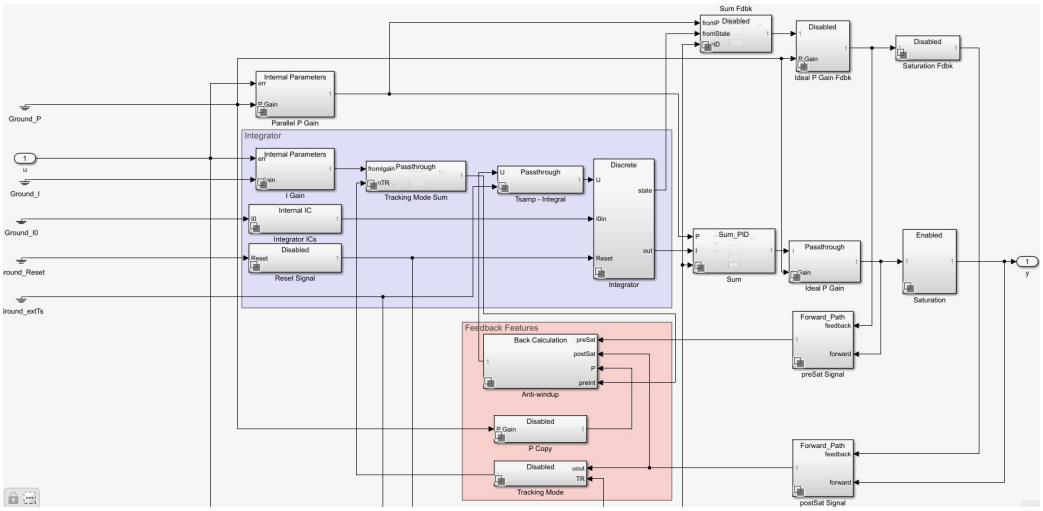


Figura 3.8: Schema interno PI

Come si nota, all'interno del PI, sono presenti due macroaree evidenziate da colori differenti: l'area blu, rappresentante l'azione di controllo integrale e l'area rossa, in cui vengono esercitate le contromisure per il fenomeno del windup.

Entrando all'interno del blocco "Back Calculation" si può osservare come lo schema sia conforme alla descrizione riportata nella sottosezione 3.1.1.

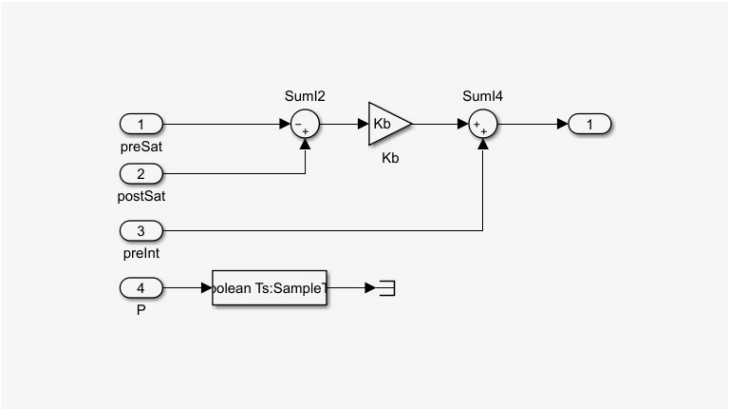


Figura 3.9: Implementazione Simulink della Back Calculation

A questo punto, sono presenti tutti i prerequisiti necessari per procedere all'implementazione sulla macchina attraverso il software PLC real-time "TwinCAT".

Capitolo 4

IMPLEMENTAZIONE IN TwinCAT

4.1 "The Windows Control and Automation Technology"

La documentazione fornita da Beckhoff afferma che [1],

The Windows Control and Automation Technology è un sistema di controllo per automazione industriale PC-based, che trasforma ogni PC in un sistema real-time, integrando al suo interno l'ambiente di sviluppo.

Per compiere questa trasformazione esso si pone allo stesso livello di architettura software del sistema operativo, in particolare, un core della CPU viene isolato e dedicato al suo funzionamento mentre i restanti sono lasciati in condivisione con Windows. In questo modo, TwinCAT, può attribuire ai task realtime una priorità maggiore di quella del S.O. evitando che essi vengano abortiti, rischiando di causare gravi problemi di sicurezza. È chiaro che avendo la precedenza sul S.O., è possibile che, per errore, la macchina su cui è installato venga compromessa. Quindi, per la fase di progettazione del PLC è stata utilizzata una macchina virtuale.

Le potenzialità di questo software sono tante e disparate, oltre alla ovvia compatibilità con il grande parco di applicativi sviluppati per Windows.
Se ne riportano alcune che saranno rilevanti per l'implementazione completa del regolatore:

- Scrittura ed esecuzione di PLC attraverso il linguaggio IEC 61131 orientato agli oggetti.
- Comunicazione con le periferiche hardware tramite bus di campo EtherCAT (o altri).
- Monitoraggio e controllo real-time delle variabili di interesse.

L'esigenza di un sistema real-time: Non è mai stato posto l'accento sulla questione, ma è chiaro che il macchinario su cui è installato l'attuatore di pressione necessiti di rispondere con tempi ben definiti agli eventi esterni che si presentano, quali che siano l'inserimento di un nuovo iniettore da testare, la richiesta di un incremento della pressione di inlet o il fermo della macchina.

Per ottenere una reattività conforme con quella di un sistema real-time, TwinCAT implementa un sofisticato sistema di priorità che gli permettono di soddisfare i task più importanti in concomitanza con il "Cycle Time" deciso dagli sviluppatori.

Per trasformare un personal computer in un sistema real-time, viene definito un "Base Time" diviso in diverse percentuali tra TwinCAT e Windows. Quando TwinCAT termina la sua percentuale, o completa l'esecuzione del task PLC corrente, la CPU viene passata a Windows che la lascerà solo una volta finito il Base Time [1].

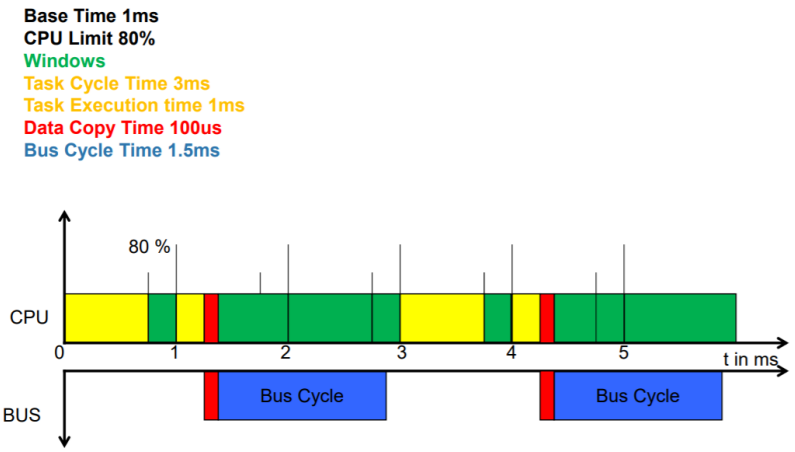


Figura 4.1: Gestione real-time di TwinCAT

Diversamente accade per l'aggiornamento dei dati di I/O, il quale avviene in parallelo all'esecuzione dei task in due modalità: La prima, che occupa tempo di calcolo della CPU, impone che i dati siano copiati sulla DPRAM della scheda fieldbus, la seconda, accede direttamente alla memoria del computer attraverso DMA ("Direct Memory Access"). Nella figura 4.3 è riportato un esempio di quanto appena scritto.

4.2 Da Simulink a TwinCAT

Per implementare il controllore all'interno del PLC presente sul PC embedded CX series montato sul banco è stato necessario rimodulare il modello Simulink nel modo seguente:

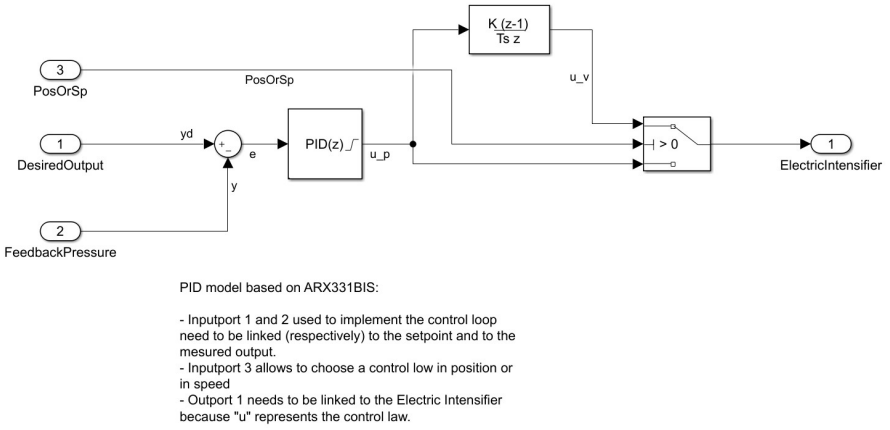


Figura 4.2: Versione finale del modello Simulink del controllore PI

Lo scope e la sequenza di gradini sono stati sostituiti con delle porte di IN/OUT necessarie per interfacciarsi con le variabili presenti nel PLC. In particolare, la porta IN1 si occupa di fornire il set-point al loop di controllo, la porta IN2 chiude l'anello di retroazione prelevando la pressione del sistema mentre OUT1 restituisce l'azione di controllo da inviare all'azionamento collegato all'attuatore. Il PLC, di norma, usufruisce di un regolatore standard Beckhoff in velocità. Per non precludere questa possibilità è stata introdotta una porta IN3 sulla quale è possibile trasmettere un booleano per scegliere tra controllo in posizione `u_p` o in velocità `u_v`. La variabile `u_v` è ottenuta derivando `u_p` tramite il metodo delle differenze finite [1].

[1] i.e. $f'(x) \simeq \frac{f(x+\Delta T)-f(x)}{\Delta T}$

TcCOM Object: Per integrare un modello Simulink in TwinCAT è necessario generare un codice C/C++ attraverso il compilatore MATLAB. Questo codice si interfaccia con il PLC attraverso il framework TcCOM basato sul Component Object Model "COM" di Microsoft, mentre Visual Studio si occupa di generarne il file binario. Il prodotto finale è chiamato TcCOM object ed è perfettamente integrato nell'ambiente di sviluppo TwinCAT. Per ottenere questo risultato, che può sembrare banale, è stato necessario l'impiego di diverso tempo. Le documentazioni Beckhoff non sono molto intuitive e siccome le operazioni da eseguire sono a basso livello è essenziale seguire attentamente alcuni passi nell'ordine corretto (cosa spesso omessa nelle guide). Si ritiene, però poco, rilevante entrare nei dettagli, per questo motivo viene riportato direttamente il risultato finale.

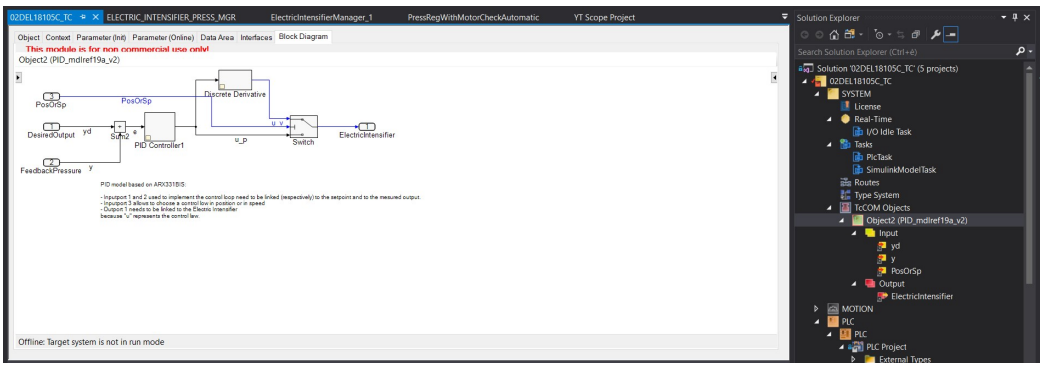


Figura 4.3: Modello Simulink integrato all'interno del PLC

4.3 Variazioni al PLC

Il PLC montato sulla CX-44CEB2, a cui è affidato un task (di "cycle ticks" 1 ms), si compone di una struttura complessa, il cui fulcro è, ovviamente, il main. Ogni millisecondo avviene la lettura degli ingressi, l'esecuzione sequenziale delle istruzioni all'interno del main e la scrittura delle uscite. I paradigmi di programmazione di TwinCAT sono chiamati POU, "Program Organization Unit". I POUs si suddividono in Programs (PRG), Function Blocks (FB) e Functions. Non rientra negli scopi di questa tesi enunciare le differenze tra queste 3 tipologie. Basti sapere che i PRG sono costituiti da istruzioni, variabili e richiami a FB o Functions, il loro scopo è quello di eseguire una serie di operazioni senza che sia necessario istanziarli. I FB, d'altro canto, sono delle unità a cui è assegnata una funzione specifica ma che necessitano di essere istanziate. Normalmente prelevano delle variabili in ingresso e restituiscono delle elaborazioni in uscita[1]. Nel main si osservano una serie di Programs, di cui, l'unico di interesse è `ELECTRIC_INTENSIFIER_PRESS_MGR()`, poiché esso si occupa della gestione dell'attuatore di pressione:

```
PROGRAM MAIN
VAR
END_VAR
```

```
WatchDog();
PRG_SAFETY();
INTERLOCK_MANAGER();
AUTO_SIGNAL_ASSIGN();

AXIS_MANAGER();
AXIS_SLIDE_MOTOR_MANAGER();
CALIBRATION_MOTOR_MANAGER();
CALIBRATION_TOOLING();

LEAK_MOTOR_MANAGER();
ELECTRIC_INTENSIFIER_PRESS_MGR();

PRG_FLOWMETER_FREQ_MNG();
```

Figura 4.4: Main del PLC

All'interno di questo PRG è attuata l'azione di controllo: vengono prelevati i segnali campionati dai sensori di posizione e di pressione, viene calcolato il forzamento in velocità attraverso un controllore PI, viene inviato il comando di movimento al motore.

Collegamento del regolatore all'hardware: Si è già visto come il modello Simulink sia presente all'interno del progetto TwinCAT e si era introdotto il fatto che sono state predisposte le opportune porte IN/OUT per interfacciarlo con le variabili del PLC. Questo paragrafo, si propone, quindi, di darne una spiegazione dettagliata al lettore.

Il regolatore è stato fornito di 4 porte che sono state opportunamente associate a 4 variabili aggiunte per adempiere a diversi scopi.

Porte di INPUT:

1. `yd` – è stata collegata alla variabile `pressure_desired` definita all'interno di `ELECTRIC_INTENSIFIER_PRESS_MGR()`, la quale sarà forzata durante l'esecuzione realtime in modo manuale tramite l'IDE di Visual Studio TwinCAT.

2. `y` – è stata collegata alla variabile `pressure_feedback` definita all'interno di `ELECTRIC_INTENSIFIER_PRESS_MGR()` e utilizzata per raccogliere il segnale trasmesso dal sensore di pressione.

3. `PosOrSp` – è stata collegata a una variabile costante, sua omonima, definita all'interno di `SOFTWARE`, (un file xml presente nel progetto e contenente la definizione di alcune variabili globali). È stata assegnata a `FALSE` in modo da indurre il controllo in posizione, ma può essere cambiata a `TRUE` per quello in velocità.

Porte di OUTPUT:

1. `ElectricIntensifier` – è stata collegata alla variabile `PidOutFromModel` anch'essa definita all'interno di `SOFTWARE`. Attraverso di essa verrà inviata l'azione di controllo all'attuatore.

```
VAR_OUTPUT
    pressure_feedback AT %Q* : LREAL;
    pressure_desired AT %Q*: LREAL;
END_VAR
```

Figura 4.5: Esempio di definizione di variabili di OUTPUT secondo l'IEC 61131

Notare che le variabili collegate alle porte di input sono definite come variabili output, l'esatto opposto accade per le porte di output.

Ridimensionamento dei sensori: In prima istanza, considerato che al momento della calibrazione tutte le misure sono tradotte in "punti" (cioè in una rappresentazione intera standard) prima che esse raggiungano il regolatore Beckhoff, mentre il controllore Simulink è sviluppato tenendo in considerazione numeri reali con le rispettive unità di misura, è stata eseguita una rimappatura delle acquisizioni originate dai sensori. Per adempiere a questo scopo è stato creato un FB denominato `FB_SCALE` che esegue un semplice calcolo:

$$out\_var = (in\_var - in\_min)\,\frac{out\_max - out\_min}{in\_max - in\_min} + out\_min \tag{4.1}$$

Dove $in\_var \in [in\_min, in\_max]$ e $out\_var \in [out\_min, out\_max]$

Una breve ricerca all'interno del computer che permette di interfacciarsi col banco su cui è montato l'oggetto di questa tesi, mostra come, di norma, la pressione prelevata dal sensore venga prima convertita in mA e poi in punti. Osservando i range adottati durante la calibrazione, la pressione è stata espressa in Bar nel modo in cui segue:

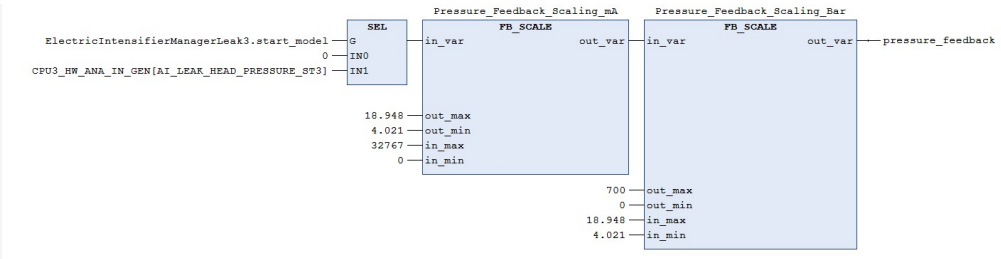
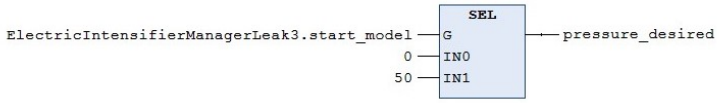


Figura 4.6: Scalatura della pressione da punti a Bar

L'esatto opposto accade per l'azione di controllo che è stata tradotta da posizione a punti (ma di questo se ne discuterà successivamente). Come si nota dalle figure, alcuni elementi non sono ancora stati introdotti: In entrata agli `FB_SCALE` si trova un particolare blocco della libreria Beckhoff, è un blocco selettore (`SEL`). La sua funzione è quella di valutare la variabile booleana in `G`, se questa è falsa viene trasmesso `IN0`, altrimenti, `IN1`. Tramite `start_model`, allora, è possibile trasmettere al modello il segnale inviato dal sensore, lungo il collegamento hardware associato alla variabile `CPU3_HW_ANA_IN_GEN[AI_LEAK_HEAD_PRESSURE_ST3]`.

Sempre attraverso `start_model` è possibile avere accesso alla nuova variabile `pressure_desired` che, come intuibile, sarà utilizzata per esprimere l'uscita desiderata nonché permettere al regolatore Simulink di generare un output mettendo il booleano `bRegulatorInAutoMode` a `TRUE`.



(a) Comando valore iniziale di `pressure_desired`

(b) Comando di inizio funzione del regolatore

Figura 4.7: Diversi utilizzi di `start_model`

ElectricIntensifierManagerLeak3: La gestione dell'attuatore all'interno del PRG `ELECTRIC_INTENSIFIER_PRESS_MGR()` viene svolta da un particolare Function Block denominato `ElectricIntensifierManager`. Vi sono 3 istanze di questo FB, una per ogni moltiplicatore elettrico presente sul banco. La terza, corrispondente all'attuatore su cui verranno eseguiti i test, è stata opportunamente cambiata per il controllo tramite modello Simulink.

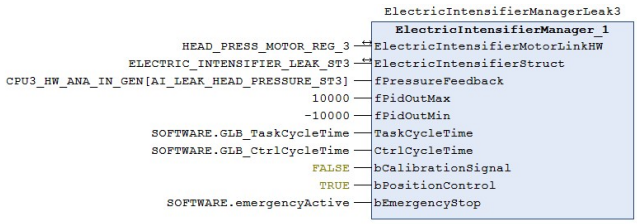


Figura 4.8: Function Block `ElectricIntensifierManager`

Esternamente, l'unica variazione presente è il valore in ingresso a `bPositionControl`, settato, ora, a `TRUE`, per il controllo in posizione. Entrando all'interno del blocco si nota subito l'aggiunta della possibilità di scelta tra azione di controllo in velocità o in posizione, nonché di un ridimensionamento da mm/s a punti e di un blocco di saturazione: un'ulteriore sicurezza per evitare che la pressione superi i limiti consentiti.

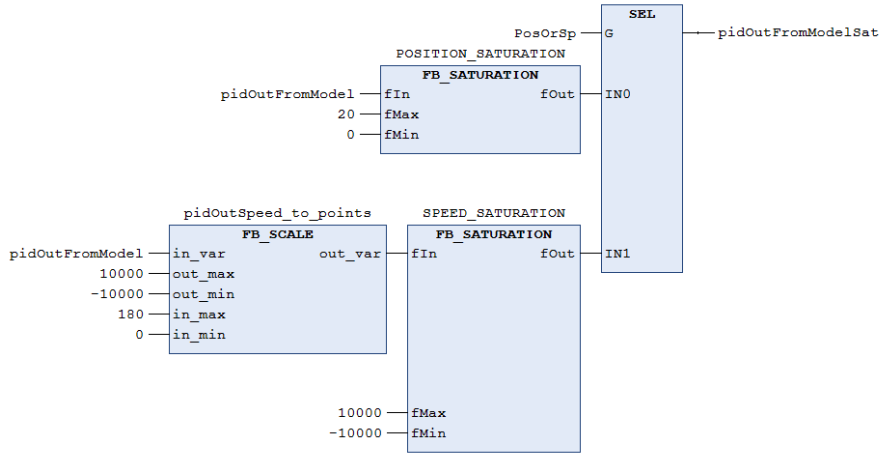


Figura 4.9: Conversione forzamento generato dal PI da punti a mm/s

I parametri relativi al ridimensionamento sono stati scelti calcolando il massimo valore raggiunto dalla velocità in un'escursione di posizione 0-20 mm e considerando i limiti di saturazione presenti precedentemente. Per quanto riguarda la posizione è stato utilizzato l'intervallo $[0, 20]$ mm per le motivazioni già citate. L'uscita saturata `pidOutFromModelSat` viene ora inviata al Function Block `PressReg WithMotorCheckAutomatic`, anch'esso modificato per il controllo in posizione.

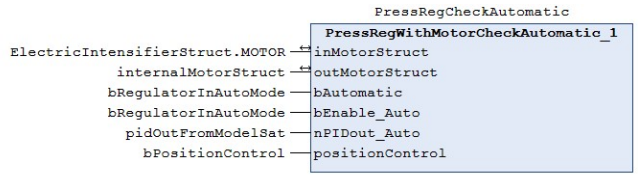


Figura 4.10: Function Block `PressRegCheckAutomatic`

Notare che l'azione di controllo all'interno del blocco viene chiamata `nPIDout_Auto` e il booleano che determina il controllo in posizione viene passato a `positionControl`. Nel momento in cui `start_model` viene forzato a `TRUE`, il codice entra nel primo `IF`. Ora, normalmente salterebbe direttamente all'`ELSE` per gestire la velocità del motore, tuttavia, essendo `positionControl` a `TRUE`, si entra nel secondo `IF`. Qui, una particolare struttura aggiorna la posizione da raggiungere al valore indicato dal PI. Successivamente, ci si chiede se il motore sia pronto a ricevere un comando o se sia in funzione. Nel primo caso, viene attivato il movimento assoluto e qui risiede il primo cambiamento: nella versione precedente il movimento era relativo, ciò permetteva di ricevere valori di posizione negativi, ora, visto che l'azione di controllo si esprime in un range di [0,20] mm è stato inserito un movimento di tipo assoluto.

```
IF (bAutomatic) THEN

    //elaborazione uscita PID

    IF positionControl THEN
        //determina la posizione a cui si deve muovere il motore
        outMotorStruct.CONTROL_WORD.TARGET_POSITION := nPIDout_Auto;

        //comando in posizione
        IF (inMotorStruct.STATUS_WORD.IN_TARGET_POS) THEN
            IF (NOT outMotorStruct.CONTROL_WORD.MOVE_ABS) THEN
                outMotorStruct.CONTROL_WORD.MOVE_ABS:= bEnable_Auto;
            ELSE
                outMotorStruct.CONTROL_WORD.MOVE_ABS := FALSE;
            END_IF
        ELSE
            outMotorStruct.CONTROL_WORD.MOVE_ABS := FALSE;
        END_IF

        //reset altri comandi
        outMotorStruct.CONTROL_WORD.MOVE_JOG_POS:=FALSE;
        outMotorStruct.CONTROL_WORD.MOVE_JOG_NEG:=FALSE;

    ELSE
        //determina la velocità a cui si deve muovere il motore
        outMotorStruct.CONTROL_WORD.OVERRIDE := (nPIDout_Auto / 1000);

        //comando in velocita (JOG)
        IF outMotorStruct.CONTROL_WORD.TARGET_POSITION >0 THEN
            outMotorStruct.CONTROL_WORD.MOVE_JOG_POS:=TRUE;
            outMotorStruct.CONTROL_WORD.MOVE_JOG_NEG:=FALSE;
        ELSIF outMotorStruct.CONTROL_WORD.TARGET_POSITION <0 THEN
            outMotorStruct.CONTROL_WORD.MOVE_JOG_POS:=FALSE;
            outMotorStruct.CONTROL_WORD.MOVE_JOG_NEG:=TRUE;
        ELSE
            outMotorStruct.CONTROL_WORD.MOVE_JOG_POS:=FALSE;
            outMotorStruct.CONTROL_WORD.MOVE_JOG_NEG:=FALSE;
        END_IF

        //reset altri comandi
        outMotorStruct.CONTROL_WORD.MOVE_REL := FALSE;

    END_IF
```

Figura 4.11: Prima parte del codice definito all'interno di `PressRegCheckAutomatic`

l'istruzione `outMotorStruct.CONTROL_WORD.MOVE_ABS = bEnable_auto` si occupa di attivare il movimento, settando `MOVE_ABS` al valore di `start_model`, che si conosce essere `TRUE`. A questo punto vengono eseguite una serie di operazioni: si esce dal blocco e si entra nel FB successivo, `PressActuatorMotorManager`, all'interno del quale si trovano implementate diverse modalità di movimento, tra cui anche quella assoluta.

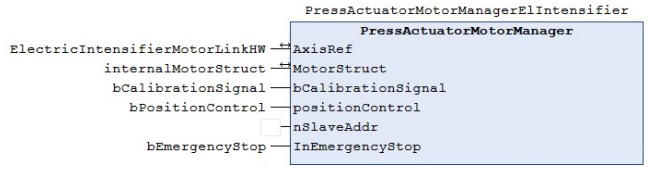


Figura 4.12: Function Block `PressActuatorMotorManager`

`MC_MoveAbsolute` appartiene alla libreria Beckhoff e si occupa del raggiungimento della posizione assoluta nella sua interezza. Tramite `AxisRef` viene confrontata la posizone corrente con quella desiderata, quando le due si equivalgono viene aggiornata la variabile `MOVE_ABS_DONE` per indicare la fine del movimento.

Tornando a `PressRegWithMotorCheckAutomatic`, nel ciclo successivo, il motore si trova nella posizione corretta ma `MOVE_ABS` è ancora a `TRUE`. Allora, si entra nel primo `ELSE`, `outMotorStruc.CONTROL_WORD.MOVE_ABS` viene messo a `FALSE` fermando del tutto l'azionamento.

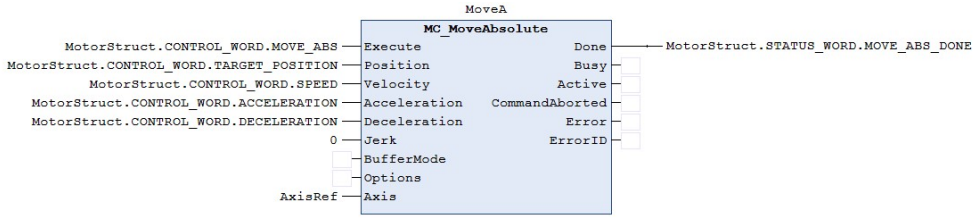


Figura 4.13: Function Block `MC_MoveAbsolute`

Infine, al di sotto delle righe di codice che gestiscono il motore, si trovano alcune istruzioni. Può capitare, infatti, che alcune operazioni vengono eseguite solo a livello software e non raggiungano direttamente il motore reale ma vengano salvate all'interno di `inMotorStruct`.
A un certo punto, però, le informazioni contenute in questa struttura devono raggiungere l'azionamento. Per questo motivo, esse vengono assegnate alla struttura `outMotorStruct` come segue.

```
ELSE
    //comando in posizione
    outMotorStruct.CONTROL_WORD.MOVE_ABS:=inMotorStruct.CONTROL_WORD.MOVE_ABS;
    outMotorStruct.CONTROL_WORD.TARGET_POSITION := inMotorStruct.CONTROL_WORD.TARGET_POSITION;

    //comando in velocita (JOG)
    outMotorStruct.CONTROL_WORD.MOVE_JOG_POS := inMotorStruct.CONTROL_WORD.MOVE_JOG_POS;
    outMotorStruct.CONTROL_WORD.MOVE_JOG_NEG := inMotorStruct.CONTROL_WORD.MOVE_JOG_NEG;
    outMotorStruct.CONTROL_WORD.OVERRIDE := 20; //massima velocita possibile cosi da far avere 100 su override

END_IF
//serve aggiornare lo stato del motore anche sulla struttura reale per refresh INPUT
inMotorStruct.STATUS_WORD := outMotorStruct.STATUS_WORD;
//serve aggiornare lo stato dei comandi del motore anche sulla struttura reale per inviare i comandi al motore reale
outMotorStruct.CONTROL_WORD.ENABLE:=inMotorStruct.CONTROL_WORD.ENABLE;
outMotorStruct.CONTROL_WORD.RESET:=inMotorStruct.CONTROL_WORD.RESET;
outMotorStruct.CONTROL_WORD.HALT:=inMotorStruct.CONTROL_WORD.HALT;
//outMotorStruct.CONTROL_WORD.MOVE_ABS:=inMotorStruct.CONTROL_WORD.MOVE_ABS;
outMotorStruct.CONTROL_WORD.HOME:=inMotorStruct.CONTROL_WORD.HOME;
outMotorStruct.CONTROL_WORD.CURRENT_LIMIT:=inMotorStruct.CONTROL_WORD.CURRENT_LIMIT;
outMotorStruct.CONTROL_WORD.SPEED:=inMotorStruct.CONTROL_WORD.SPEED;
outMotorStruct.CONTROL_WORD.ACCELERATION:=inMotorStruct.CONTROL_WORD.ACCELERATION;
outMotorStruct.CONTROL_WORD.DECELERATION:=inMotorStruct.CONTROL_WORD.DECELERATION;
```

Figura 4.14: Seconda parte del codice definito all'interno di `PressRegCheckAutomatic`

Capitolo 5

ANALISI DELLE PRESTAZIONI

5.1 Caratteristiche dell'hardware usato nei test

Terminata la fase di sviluppo del controllore si è proceduto alle verifiche pratiche sul macchinario.
Purtroppo, non è stato possibile disporre del banco iniziale, su cui sono state eseguite le acquisizioni per l'identificazione, poiché il cliente, a cui era destinato, ha richiesto che gli venisse spedito.

In alternativa, le verifiche e l'implementazione del PLC sono state svolte su un altro banco, analogo al precedente per alcuni aspetti fondamentali, ma con alcune differenze che verranno illustrate in questa sezione.

Per estrapolare queste difformità si è analizzato il grafico IN/OUT (posizione/pressione) ottenuto dal nuovo banco.

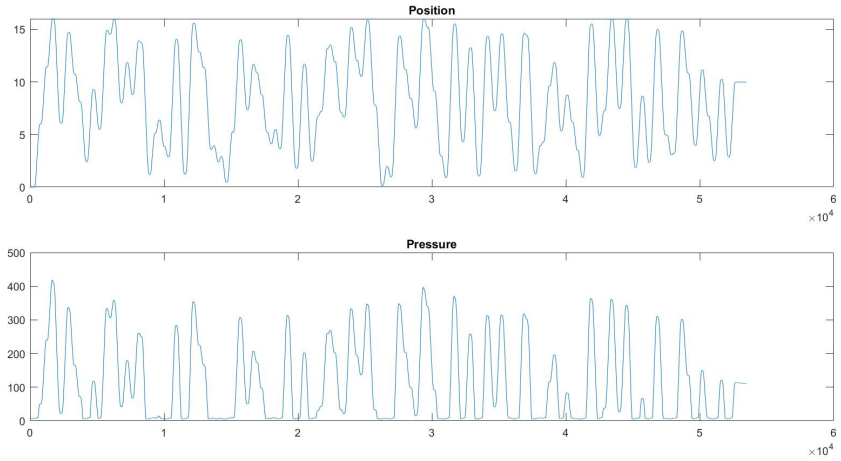


Figura 5.1: Acquisizione ingressi e uscite del nuovo attuatore

É immediato notare che:

1. Se prima era sufficiente porsi a circa 11 mm per ottenere una pressione di 400 Bar, ora ne servono 16.
2. La caduta di pressione, soprattutto quando si raggiungono livelli elevati, è più accentuata.
3. Al di sotto dei 6 mm non si osserva alcuna variazione di pressione.

Risulta rilevante soprattutto il terzo punto di questa breve lista. Se così è, infatti, ci si trova di fronte a un comportamento non lineare che rende inadeguato l'utilizzo della classe di modelli ARX. É probabile, però, che si tratti di un problema al sensore di pressione che presenta una sensibilità eccessivamente bassa nel primo tratto di funzionamento dell'attuatore.

Purtroppo, non vi è stata la possibilità di approfondire la questione, si è comunque tentata un'identificazione con i mezzi di cui si poteva disporre.
Come ci si poteva aspettare, i risultati sono stati tutt'altro che buoni, con l'ARX 331 BIS, ottenuto precedentemente, che possiede un fitting davvero basso.
Ulteriore dimostrazione delle differenze che sono presenti tra i due macchinari.

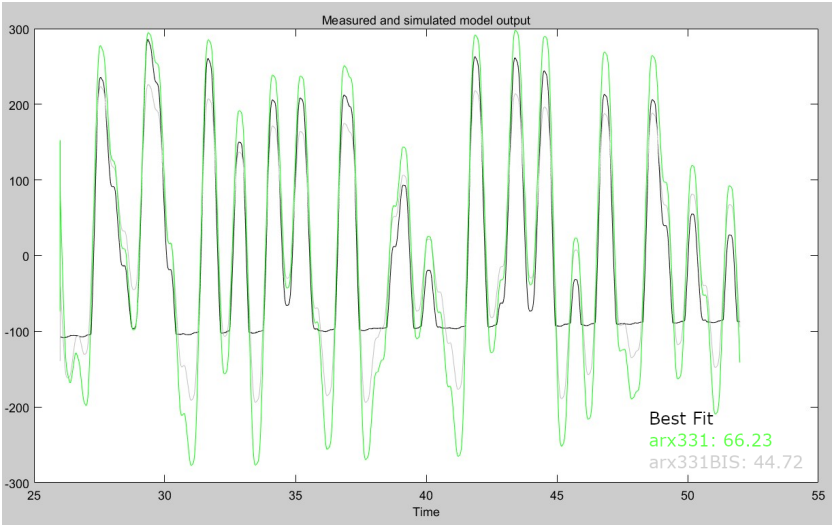


Figura 5.2: Accuratezza ARX 331 nuovo banco comparata con l'ARX 331 BIS

Come si vedrà nella successiva sezione, ciò comporterà una nuova taratura del PI, questa volta manuale.

5.2 Test e correzione della taratura del regolatore

In prima istanza, si è osservato il comportamento del regolatore senza collegarne l'uscita all'attuatore, riservando il compito della regolazione al PI standard appartenente alla libreria Beckhoff. Dopo aver verificato che il PI inducesse un andamento sicuro e stabile, si è proceduto a connetterlo in modo diretto all'hardware.

Per l'analisi e l'osservazione delle grandezze di interesse si è fatto uso del "TwinCAT measurement project", un' importante funzione di TwinCAT che permette, tra le cose, di campionare i valori delle variabili selezionate, al variare del tempo.

Perché tutto fosse fatto nella maggior sicurezza possibile si è iniziato con piccoli valori desiderati di pressione.

Il primo risultato, però, come preannunciato, non è stato molto soddisfacente:

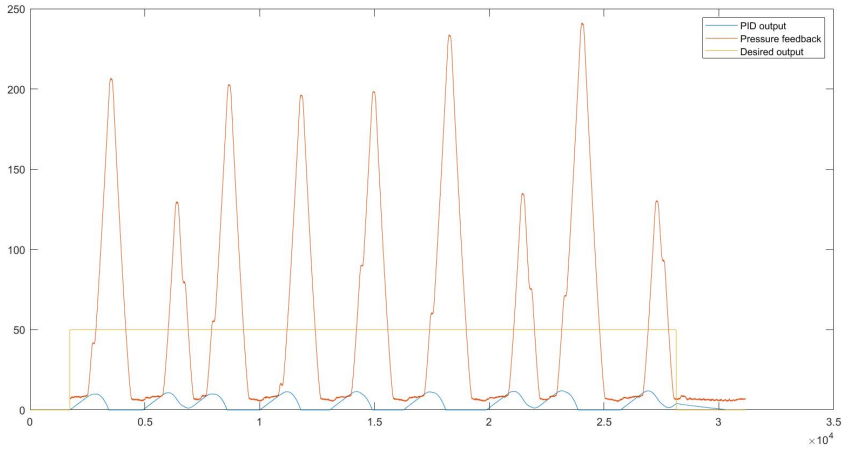


Figura 5.3: Primo tentativo di controllo

L'azione del PI si è dimostrata del tutto irregolare con pericolosi picchi decisamente oltre il valore desiderato. Considerate tali oscillazioni si è pensato di tarare nuovamente i parametri del PI per compensare le differenze presenti nel nuovo banco.

Siccome l'output del PI non raggiunge mai il limite di saturazione, si è concluso che il fenomeno non derivasse dall'erronea scelta del coefficiente K_b. Inoltre, le forti oscillazioni rappresentano il risvolto negativo di un termine integrale troppo accentuato. Per questo, si è deciso di diminuirne il contributo abbassando il guadagno K_I di un ordine di grandezza: $K_I = 0.0256803140652026$.

I risultati sono stati sorprendenti, con una risposta a regime permanente estremamente buona. Tuttavia, il tempo di salita, soprattutto nel primo tratto non è soddisfacente, anche se probabilmente ciò è dovuto principalmente al fatto che il livello di pressione iniziale assegnato al PI è 0, mentre nella camera dell'attuatore vi è un residuo di pressione che oscilla attorno ai 7 Bar.

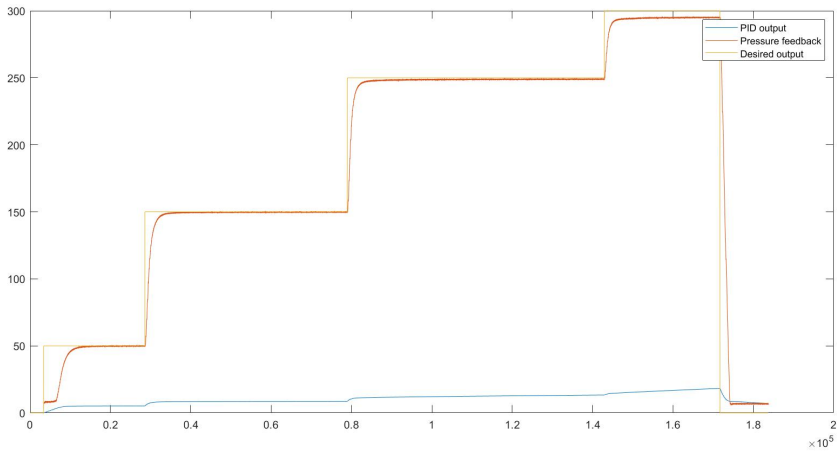


Figura 5.4: Secondo tentativo di controllo

Il tempo di salita nel primo tratto si attesta a 6.28 secondi mentre il tempo di assestamento (con soglia del 2%) è di 16.8 secondi. Per ottenere una prontezza maggiore è stato aumentato il coefficiente proporzionale in modo progressivo, ottenendo buoni miglioramenti.

5.3 Analisi dei risultati

La versione finale del regolatore raggiunge il 90% del valore desiderato, in uscita, in 3 secondi e impiega 19.8 secondi a rientrare nel 2% del valore di regime se l'escursione è di 50 bar.
Essa è dotata dei seguenti parametri:

- $K_P = 0.0128401570326013$
- $K_I = 0.0256803140652026$
- $K_b = 1$
- $u(t) \in [0, 20]$

Il tempo di assestamento potrebbe preoccupare ma non è del tutto rilevante in quanto costituisce una conseguenza della scarsa precisione dei sensori, ad avvalorare questa ipotesi è il fatto che la pressione rilevata oscilla fortemente attorno al valore reale.

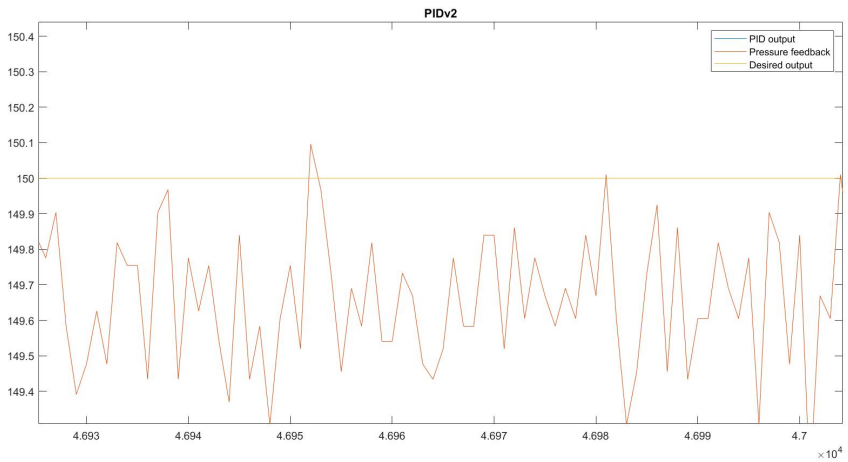


Figura 5.5: Oscillazione della pressione per la scarsa precisione dei sensori

Si noti, inoltre che, a causa della gestione tramite set-point degli azionamenti, se l'escursione richiesta è maggiore, il tempo di salita decresce. Il regolatore, infatti, percependo un errore maggiore richiede subito una posizione elevata, e, siccome deve attendere che il motore raggiunga quella posizione prima di poter inviare un altro comando, si ottiene un incremento della pressione alla velocità massima consentita dal motore.

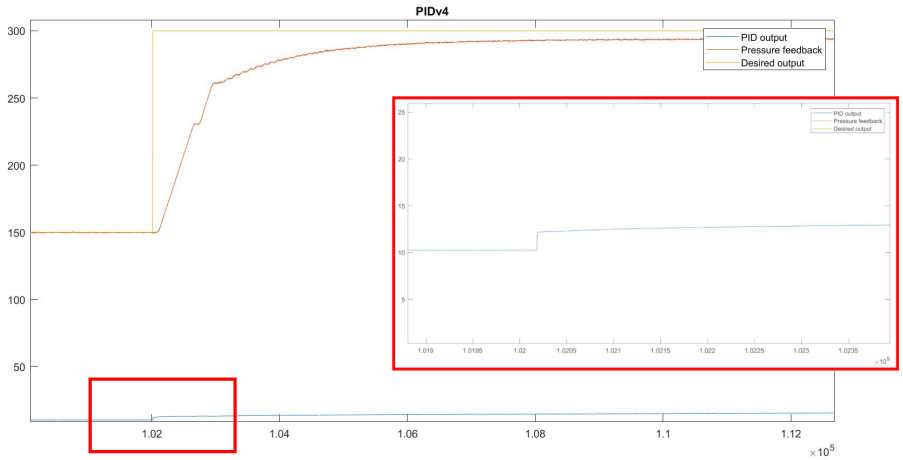


Figura 5.6: Oscillazione della pressione per la scarsa precisione dei sensori

Infine, l'immagine qui sopra, introduce un fenomeno di cui ancora non si è discusso: evidentemente, nella camera dell'attuatore del nuovo banco vi è una perdita che non permette di mantenere la pressione al di sopra dei 295 Bar. Questa conclusione è accreditata dal fatto che anche il controllore PI in velocità, della libreria Beckhoff, non riesce portare il sistema ai 300 Bar richiesti se non aumentando costantemente la posizione dell'attuatore.

Capitolo 6

RIEPILOGO E CONCLUSIONI

Si riepilogano, infine, le caratteristiche salienti del prodotto finale di questa tesi.

modello matematico del processo L'attuatore di pressione è stato modellato con un AutoRegressive Exogenous model, a tempo di discreto, descritto dai seguenti polinomi:

$$A(z) = 1 - 0.4927(\pm 0.0045)z^{-1} - 0.3317(\pm 0.0048)z^{-2} - 0.1735(\pm 0.0045)z^{-3}$$
$$B(z) = 31.96(\pm 0.1611)z^{-1} + 1.65(\pm 0.07927)z^{-2} - 33.54(\pm 0.1609)z^{-3}$$

e dalla funzione di trasferimento:

$$F(z) = \frac{31.96(\pm 0.1611)z^2 + 1.65(\pm 0.07927)z - 33.54(\pm 0.1609)}{z^3 - 0.4927(\pm 0.004565)z^2 - 0.3317(\pm 0.004851)z - 0.1735(\pm 0.004568)}$$

di ordini $na = 3, nb = 3$ e di ritardo (nk) pari a 1.

Il tempo di campionamento utilizzato è stato di 1 ms.

Il modello gode di stabilità e buona somiglianza con il processo reale sia per quanto riguarda la risposta in frequenza che per l'accuratezza, con valori di fitting superiori all 80% in tutti i casi esaminati.

Regolatore Per l'azione di regolazione è stato progettato un PI dotato di anti wind-up, sfruttando il metodo della Back Calculation.

Esso viene descritto dai parametri:

- $K_P = 0.0128401570326013$
- $K_I = 0.0256803140652026$
- $K_b = 1$
- $u(t) \in [0, 20]$ (intervallo di saturazione)

E sono state rilevate le seguenti performance:

- $t_s = 3$s
- t_a[1] $= 19.8$s

In conclusione, i tutor aziendali hanno ritenuto che le performance del PI sviluppato fossero coerenti con quelle desiderate: il regolatore si dimostra stabile e pronto abbastanza per il soddisfacimento dei suoi scopi e, in generale, i risultati sono conformi con quelli del PI precedente, fornito dalla libreria Beckhoff.

Pare chiaro che le performance non siano in alcun modo comparabili con quelle simulate, in cui il tempo di salita t_s era stimato essere 0.228 s. Tuttavia, si deve tenere conto che quella simulazione si basava su un modello ARX331 BIS che, è vero, si comporta in modo simile al processo reale, ma costituisce, comunque, un'approssimazione significativa. Ancora più importanza assume il fatto che il banco, su cui sono stati effettuati i test, mostra significative differenze rispetto a quello iniziale, sulla base del quale, invece, è stato determinato il modello polinomiale.

[1]Dove si considera per tempo di assestamento il tempo necessario a rientrare nel 2% del valore di regime

Ringraziamenti

I primi ringraziamenti vanno senz'altro ai miei genitori che mi hanno permesso di studiare senza che ci fossero impedimenti di alcun genere. Ringrazio mia madre per avermi fatto comprendere quanto fosse importante studiare, per avermi condotto verso un percorsi di studi che non ha mai deluso le mie aspettative e per aver infuso in me tanta fiducia. Ringrazio mio padre per aver alimentato la mia curiosità rispondendo con interesse alle tante domande che continuamente gli presentavo e per avermi trasmesso la capacità di affrontare in modo oggettivo e concreto i problemi che mi sono stati posti davanti.

Ringrazio Maria Grazia per la sua presenza costante, per l'affetto e per avermi donato molti tra i momenti più belli della mia vita.

Ringrazio gli amici per essere quella fonte di svago e divertimento che rendono l'esistenza molto più di un semplice susseguirsi di avvenimenti.

Ringrazio gli ingegneri Alessio Grasselli e Luca Cesarini e il responsabile delle risorse umane, in Loccioni, Francesco De Stefano per essersi spesi nella proposta di un tirocinio che fosse interessante e formativo, per avermi dato fiducia e per avermi seguito in questo percorso sempre con un bel sorriso stampato sulla faccia.

Ringrazio il mio relatore, prof. Giuseppe Conte, per avermi preso in carico in modo corretto e puntuale, per essere stato un ottimo insegnante nonché un grande punto di riferimento per la mia carriera.

Ringrazio mio zio, Marco, per avermi introdotto a questo corso di laurea che si è rilevato azzeccato e conforme con i miei sogni e desideri. Grazie perché, insieme a Matteo, mi hai trasmesso quella passione per la tecnologia senza la quale non avrei intrapreso questa strada.

Infine, ringrazio tutti i parenti per essere stati una casa accogliente e felice. Grazie per essere sempre stati volenterosi di festeggiare o di passare del tempo insieme. Grazie per la prontezza che avete dimostrato nel tendere una mano. Vi sento vicini.

Bibliografia

[1] *Documentazione TwinCAT fornita da AEA srl.* Beckhoff Automation. Verl, Germania.

[2] Anirban Ghoshal e Vinod John. "Anti-windup Schemes for Proportional Integral and Proportional Resonant Controller". In: (giu. 2010).

[3] Lennart Ljung. "System Identification Toolbox for use with MATLAB". In: 21 (gen. 2011).

[4] Nicola Schiavoni Paolo Bolzern Riccardo Scattolini. *Fondamenti di controlli automatici.* McGraw Hill, 2004.

[5] David Scaradozzi. "Materiale didattico del corso di Modellistica e Identificazione dei processi dinamici". 2019/2020.

[6] Torsten Söderström e Petre Stoica. *System Identification.* Prentice Hall, 1989.

[7] Petre Stoica. "A test for whiteness". In: *IEEE Transactions on Automatic Control* 22.6 (1977), pp. 992–993.

Elenco delle figure