



# Università Politecnica delle Marche

---

## FACOLTA' DI INGEGNERIA

Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione

### **Problematiche di analisi, simulazione e controllo di un moltiplicatore di pressione idraulico**

Analysis, simulation and control problems of an hydraulic  
pressure booster

Relatore:  
Chiar.mo Prof.  
Giuseppe Conte

Presentata da:  
Marco Liberati

Sessione Autunnale  
Anno Accademico 2018/2019



# Indice

<b>Introduzione</b>	7
<b>Capitolo 1</b>	
<b>Descrizione modello moltiplicatore di pressione</b>	9
1.1 Il moltiplicatore di pressione	10
1.2 La servovalvola proporzionale	12
1.3 Modello complessivo	13
1.4 Implementazione in Simulink	17
<b>Capitolo 2</b>	
<b>La logica Fuzzy</b>	21
2.1 Definizioni	22
2.2 Funzioni di appartenenza	23
2.3 Operazioni sugli insiemi	26
2.4 Regole Fuzzy	28
2.5 Struttura di un sistema Fuzzy	29

## Capitolo 3

<b>Descrizione controllori PID e Fuzzy PID</b>	35
3.1 Controllore PID	35
3.2 Controllore Fuzzy PID	38

## Capitolo 4

<b>Implementazione logica Fuzzy</b>	40
4.2 Implementazione controllore Fuzzy PI in Simulink	45

## Capitolo 5

<b>Simulazioni controllori PI e fuzzy PI</b>	48
5.1 Verifica robustezza controllori	50

## Capitolo 6

<b>Descrizione componenti Beckhoff</b>	52
6.1 CX5140	52
6.2 Software TwinCAT	55
6.3 Protocollo Ads	57
6.4 Guida integrazione Simulink e TwinCAT	57

## **Capitolo 7**

<b>Implementazione hardware</b>	67
7.1 Implementazione in TwinCAT	70
7.2 Interfaccia grafica	71
7.7 Avvio progetto TwinCAT con interfaccia grafica	75

## **Capitolo 8**

<b>Risultati implementazione reale</b>	77
8.1 Implementazione finale	80

<b>Conclusioni</b>	86
--------------------	----

<b>Appendice A</b>	87
--------------------	----

<b>Bibliografia</b>	88
---------------------	----

<b>Ringraziamenti</b>	89
-----------------------	----



# Introduzione

Uno degli obiettivi fondamentali dell'industria automobilistica è quello di ridurre il consumo di carburante da parte delle automobili con motori termici tradizionali, migliorando il processo di combustione del carburante in modo da aumentarne la qualità e soddisfare i livelli di emissione imposti, mantenendo comunque elevate prestazioni. Un ruolo chiave nel funzionamento del motore, in relazione a questa problematica, è svolto dall'iniettore e, proprio per questo, attualmente si presta molta attenzione alle fasi di calibrazione e test degli iniettori. In particolare, avendo l'iniettore il compito di introdurre l'esatta quantità di carburante calcolata dalla centralina controllo in relazione ad ogni condizione operativa del motore, è necessario che esso rispetti una determinata distanza e un angolo di iniezione specifico per poter ottenere risultati ottimali.

In questo contesto, il Gruppo Luccioni negli anni ha sviluppato un banco test apposito per verificare il corretto funzionamento dell'iniettore. La procedura di test consiste di cinque fasi che corrispondono all'esecuzione di cinque differenti tipi di misura e verifica per il corretto funzionamento dell'iniettore. Le fasi sono denominate, rispettivamente, run-in, calibrazione, flussaggio, tenuta e spray. Nella fase di run-in, l'iniettore viene semplicemente alimentato e azionato a basse pressioni per essere riscaldato per le fasi successive. Si passa poi alla fase di calibrazione, dove l'iniettore viene alimentato con l'exsol in pressione ed azionato a varie frequenze. In questa fase, l'iniettore viene montato su un apposito supporto e una valvola d'intercettazione convoglia il fluido di prova su di esso per verificarne la tenuta. Una volta ottenuto l'injection rate voluto, si regola il precarico della molla, calibrando così l'iniettore. Nella fase di flussaggio, l'iniettore viene fatto funzionare per verificare che la calibrazione effettuata sia corretta. Successivamente si passa alla prova di tenuta, nella quale viene testata la perdita dell'iniettore quando è alimentato con un fluido in pressione. L'ultima fase è quella dello spray, nel corso della quale si analizza il getto di fluido uscente dall'iniettore. Per eseguire questo ultimo test, si utilizza con una lampada stroboscopica che, illuminando l'iniettore, permette di evidenziare il getto di

materiale emesso. La geometria del getto è fondamentale per ottimizzare la combustione del carburante.

Nel corso di questa tesi, è stata presa in considerazione la fase di tenuta, durante la quale l'iniettore viene alimentato mediante un moltiplicatore di pressione idraulico per aumentare la pressione del fluido e verificare la presenza di perdite. Nel corso del Capitolo 1, si andrà perciò a descrivere la struttura del moltiplicatore di pressione e, con essa, il modello matematico del sistema di test ricavato attraverso l'utilizzo del software Simulink di Matlab. Una volta ottenuto il modello, il lavoro si è concentrato principalmente sulla sintesi e sull'implementazione di un controllore Fuzzy PI che permettesse di migliorare la procedura di test. La struttura Fuzzy PI del controllore è stata scelta per migliorare le prestazioni di robustezza in presenza di disturbi e incertezze parametriche. La logica Fuzzy sulla quale si basa il controllore, la sua descrizione, l'implementazione e la validazione in simulazione sono descritte nei Capitoli 2, 3, 4, 5. Il modello del controllore è stato poi esportato dall'ambiente Simulink all'ambiente Twincat, in modo da poter poi procedere ad una implementazione del controllore stesso su una struttura hardware composta da un dispositivo Beckhoff CX5140 con controllore embedded. Una descrizione dell'hardware e della modalità di implementazione è contenuta nei Capitoli 6 e 7. I risultati dei test effettuati sfruttando tale configurazione, illustrati e discussi nel Capitolo 8, mostrano l'efficacia del controllore e la semplicità di implementazione su dispositivi industriali. Per la maggiore capacità di ottenere elevate prestazioni rispetto ai controllori classici in presenza di disturbi e incertezze parametriche, esso si qualifica come una valida soluzione di controllo del sistema di test.



# Capitolo 1

## Descrizione modello moltiplicatore di pressione

Il moltiplicatore di pressione analizzato viene utilizzato per effettuare test in automatico sugli iniettori, compiendo differenti prove su di esso verificandone le prestazioni. Il sistema idraulico può essere rappresentato come segue:

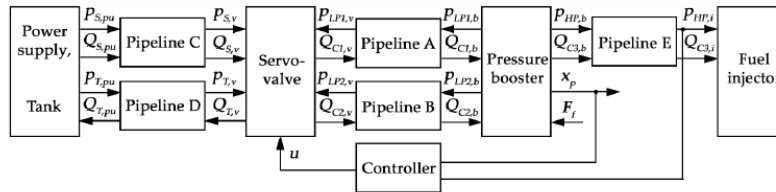


Figura 1 Modello del moltiplicatore di pressione

In particolare, il moltiplicatore di pressione è composto dai seguenti elementi:

- **L'unità di alimentazione** (power supply) trasforma potenza elettrica in potenza idraulica attraverso l'utilizzo di una pompa e una riserva di olio idraulico contenuta all'interno del serbatoio (tank).
- La **servovalvola** (servo-valve) che attraverso un segnale di comando specifico regola la portata di fluido all'interno del moltiplicatore e, di conseguenza la pressione nelle due camere di bassa pressione del moltiplicatore di pressione.
- Le **tubazioni** (pipelines) sono considerate solo se presentano una lunghezza elevata.
- Il **moltiplicatore di pressione** (pressure booster) il quale, aumenta la pressione fino al valore desiderato impostato come riferimento.
- L' **iniettore** (fuel injector) testato utilizzando fluidi con proprietà simili ai principali combustibili come benzina e gasolio.
- Un **controllore** (controller) che conoscendo la posizione del pistone attraverso un sensore LVDT e la pressione misurata, controlla la pressione nella camera di alta pressione.

## 1.1 - Il moltiplicatore di pressione

Il moltiplicatore di pressione è un dispositivo che permette di fornire una pressione in utilizzo maggiore di quella disponibile in ingresso; di fatto si tratta di un torchio idraulico costituito da due camere, una di alta pressione ed una di bassa pressione. La pressione prodotta risulta molto stabile e priva di oscillazioni e può quindi, essere utilizzato per testare un componente come l'iniettore nel quale, le condizioni al contorno devono rimanere il più costante possibile per non alterare la misura.

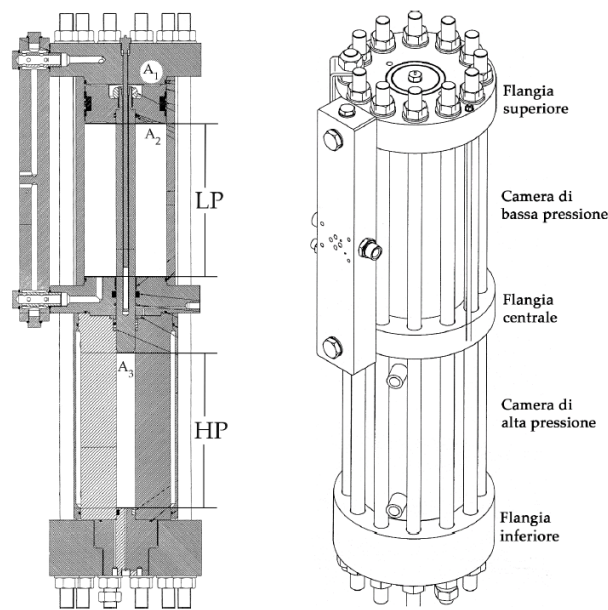
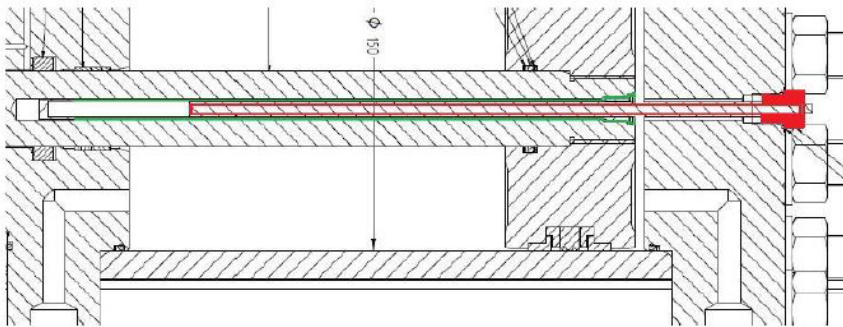


Figura 2 Sezione e modello 3D del moltiplicatore di pressione

Come si nota dalla figura, il moltiplicatore è composto da due camere, una di bassa pressione LP, divisa a sua volta in LP1 e LP2 ubicate rispettivamente sopra e sotto lo stelo e, una di alta pressione HP. Quando un fluido idraulico ISO VG46 viene introdotto nella camera LP1 al di sopra dello stelo, si esercita una forza  $F_1$  sull'area  $A_1$  del pistone mentre, una forza  $F_2$ , diversa da  $F_1$ , viene applicata sull'area  $A_2=A_1-A_3$ . La forza  $F_2$ , è tale da muovere lo stelo trasmettendo a sua volta una forza  $F_1-F_2$  al fluido di test nella camera HP, aumentando la pressione al suo interno. Essendo l'area  $A_3$  dello stelo minore dell'area  $A_1$ , la pressione del fluido di test risulterà moltiplicata per il rapporto tra le sezioni. La misura della posizione avviene utilizzando

un sensore ICT100 della Penny & Giles, che non è altro che un sensore LVDT (Linear Variable Displacement Transducer) progettato per essere inserito all'interno di ambienti pressurizzati. Il sensore viene utilizzato per sapere quando la camera ad alta pressione è vuota, in modo tale da poter attivare il riempimento. Nella fase di riempimento l'olio idraulico entra nella camera di bassa pressione LP1 provocando l'abbassamento dello stelo, in modo che il fluido possa entrare attraverso il condotto di alimentazione nella camera ad alta pressione. Successivamente per mettere in pressione il fluido nella camera di alta pressione, viene introdotto olio idraulico all'interno della camera di bassa pressione LP2. L'uscita del fluido nella camera di alta pressione avviene attraverso il canale di mandata. Il controllo dei flussi di olio è effettuato attraverso una valvola 4 vie 4 posizioni che analizzeremo successivamente.



**Figura 3** Sezione del sensore LVDT

## 1.2 - La servovalvola proporzionale

La servovalvola proporzionale installata è un modello D765 della casa produttrice Moog. Viene utilizzata per regolare l'entrata e l'uscita di olio idraulico nelle camere di bassa pressione del moltiplicatore (LP1 e LP2).

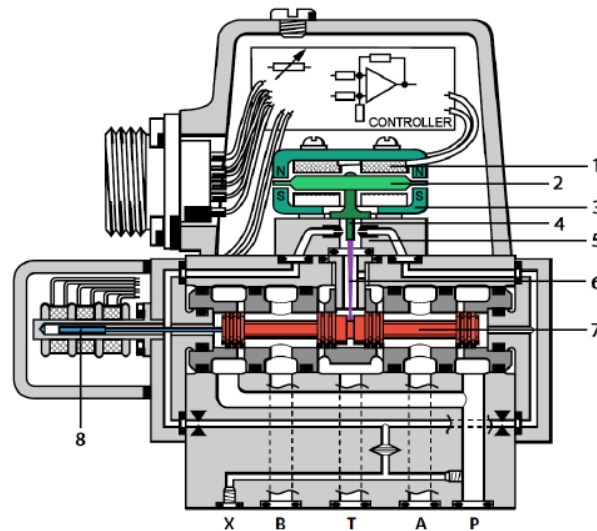


Figura 4 Valvola proporzionale Moog D765

Il funzionamento della servovalvola parte dalla ricezione di un setpoint di portata che non è altro che un segnale di comando in tensione. Una volta ricevuto il segnale, il controller eccita una coppia di solenoidi (1) che polarizzano un'armatura metallica (2) all'interno di un campo magnetico generato dalla presenza di due magneti permanenti (3). L'armatura a causa delle forze magnetiche inizia a ruotare e la linguetta (4), solidale con essa, ruotando, chiude un ugello e ne apre un altro nel circuito di sbilanciamento (5). L'apertura e la chiusura di un ugello provoca una differenza di pressione che comporta il movimento laterale della spoletta della valvola (7). Il movimento fa sì che le due porte di controllo (A e B) siano collegate con le porte di alimentazione e di ritorno (P e T). Il controllore viene utilizzato per regolare la corrente in base alla differenza tra il setpoint e la posizione della spoletta ottenuta tramite il sensore LVDT. In base alla corrente il controllore fa sì che la valvola in uscita produca una determinata portata di fluido che, nel nostro caso, sarà inserito all'interno delle camere di bassa pressione del moltiplicatore.

### 1.3 - Modello complessivo

Lo schema introdotto nella figura 1 è stato semplificato nel seguente modo:

- Le tubazioni, non avendo lunghezze esagerate, sono state considerate statiche e quindi con un volume costante di flusso all'interno.
- Le pressioni delle porte P e T ( $P_S$  e  $P_T$ ) e il flusso di alimentazione  $Q_S$  sono stati considerati costanti.
- La dinamica relativa all'attivazione dell'iniettore e quindi alla fuoriuscita di liquido non è stata considerata. Al suo posto è stata inserita una costante nulla per descrivere l'iniettore come un tappo. Di conseguenza è stato considerato nullo anche il flusso uscente dalla camera di alta pressione ( $Q_{c3}$ ).

Il sistema finale è costituito quindi dal moltiplicatore di pressione, la servovalvola proporzionale e il controllore.

Forniamo una breve descrizione del modello complessivo già precedentemente realizzato per comprendere al meglio il funzionamento e soprattutto la sua implementazione all'interno del software Simulink.

Prima di introdurre la descrizione dettagliata delle varie equazioni presenti nel sistema, introduciamo i parametri geometrici rappresentanti il moltiplicatore di pressione preso in questione.

Descrizione	Parametro	Valore	Unità di misura
Area del pistone	$A_1$	$(\pi/4) \cdot 0,015^2$	$m^2$
Area dello stelo	$A_2$	$(\pi/4) \cdot 0,0035^2$	$m^2$
Area del pistone lato stelo	$A_3$	$A_2 = A_1 - A_3$	$m^2$
Volume morto camera LP1	$V_{01}$	$38.07 \cdot 10^{-6}$	$m^3$
Volume morto camera LP2	$V_{02}$	$V_{01} = V_{02}$	$m^3$
Volume morto camera HP	$V_{03}$	$0,5 \cdot 10^{-5}$	$m^3$
Lunghezza camera LP	$S_{LP}$	0.28	$m$
Lunghezza camera HP	$S_{HP}$	0.282	$m$
Massa del pistone più stelo	$m$	10.27	$kg$

Il valore del bulk modulus  $\beta_2$  del fluido ISO 4113 utilizzato per il test, delle forze di attrito e il valore del bulk modulus  $\beta_1$  dell'olio idraulico ISO VG46, sono derivati da un'analisi precedente effettuata su un modello molto simile di moltiplicatore di pressione.

Bulk modulus fluido di test ISO 4113	$\beta_2$	$1,5 \cdot 10^9$	$P_a$
Forza di Coulomb	$F_c$	2	$N$
Area del pistone lato stelo	$\mu$	0,01275	$N_s / m$
Bulk modulus olio idraulico ISO VG46	$\beta_1$	$1,69823 \cdot 10^9$	$P_a$

Il modello è composto da un vettore delle variabili di stato  $\mathbf{x} \in \mathbb{R}^5$  e un vettore degli ingressi  $\mathbf{u} \in \mathbb{R}^3$  che descrivono le seguenti equazioni differenziali:

$$\mathbf{x} = [x_1 \ x_2 \ x_3 \ x_4 \ x_5]^T$$

$x_1 = x_p$	Posizione del pistone
$x_2 = \dot{x}_p$	Velocità del pistone
$x_3 = p_{LP1}$	Pressione nella camera LP1
$x_4 = p_{LP2}$	Pressione nella camera LP2
$x_5 = p_{HP}$	Pressione nella camera HP

$$\mathbf{u} = [u_1 \ u_2 \ u_3]^T$$

Flusso di controllo alla camera LP1	$u_1 = Q_{c1}$
Flusso di controllo alla camera LP2	$u_2 = Q_{c2}$
Flusso uscente dalla camera HP	$u_3 = Q_{c3}$

$$\left\{ \begin{array}{l} \dot{x}_1 = x_2 = f_1(x, u) \\ \dot{x}_2 = \frac{1}{m} [(x_3 - x_4 \bar{A}) A_1 - \text{sgn}(x_2) F_c - \mu x_2 - x_5 A_3] = f_2(x, u) \\ \dot{x}_3 = \frac{\beta_1}{V_{01} + A_1 x_1} (u_1 - x_2 A_1) = f_3(x, u) \\ \dot{x}_4 = \frac{\beta_1}{V_{02} + A_2 (S_{LP} - x_2)} (u_2 + x_2 A_2) = f_4(x, u) \\ \dot{x}_5 = \frac{B_2}{V_{03} + A_3} (S_{HP} - x_2) (u_3 + x_2 A_3) = f_5(x, u) \end{array} \right.$$

Per quanto riguarda le prime due equazioni del sistema finale, esse rappresentano la **dinamica del pistone** che, note le pressioni nelle due camere di bassa pressione, ci permette di calcolare la forza agente sul pistone con la seguente formula:

$$\mathbf{F}_P = p_{LP1} \mathbf{A}_1 - p_{LP2} \mathbf{A}_2 \quad \text{oppure} \quad \mathbf{F}_P = \mathbf{A}_2 (p_{LP1} - p_{LP2} \bar{\mathbf{A}}) \quad \text{con} \quad \bar{\mathbf{A}} = \frac{A_2}{A_1}$$

Considerando il pistone in movimento è necessario aggiungere altre due forze opposte che sono:

- la forza di attrito ( $\mathbf{F}_f = \text{sgn}(\dot{x}_p) \mathbf{F}_c + \mu \dot{x}_p$ ), costituita dalla forza di Coulomb ( $\mathbf{F}_c$ ) dipendente dal verso del moto e quindi dal  $\text{sgn}(\dot{x}_p)$  e, dalla forza di attrito viscoso con coefficiente  $\mu$ .
- la forza ( $\mathbf{F}_3 = p_{HP} \mathbf{A}_3$ ) generata a causa dell'aumento di pressione all'interno della camera di alta pressione.

Sommando tutte le equazioni descritte, otteniamo esattamente la seconda formula espressa all'interno del sistema.

$$\dot{x}_2 = \frac{1}{m} [(x_3 - x_4 \bar{\mathbf{A}}) \mathbf{A}_1 - \text{sgn}(x_2) \mathbf{F}_c - \mu x_2 - x_5 \mathbf{A}_3]$$

Per quanto riguarda la prima, come si può notare, essa non è altro che l'applicazione di una derivata per ricavare la velocità nota la posizione.

$$\dot{x}_1 = x_2$$

La terza e quarta equazione invece, rappresentano la **dinamica della pressione nella due camere di bassa pressione** generata dalla servovalvola proporzionale. Le equazioni descrivono la variazione dei flussi netti in ognuna delle due camere  $Q_{C1}$  e  $Q_{C2}$ . La variazione deriva dal movimento del pistone e quindi da una variazione di volume ( $\dot{V}_1 = \dot{x}_p A_1$  e  $\dot{V}_2 = -\dot{x}_p A_2$ ) e, dalla compressibilità dell'olio idraulico attraverso il bulk modulus  $\beta_1$ .

$$Q_{C1} = \dot{V}_1 + \frac{V_1}{\beta_1} \dot{p}_{LP1} \quad Q_{C2} = \dot{V}_2 + \frac{V_2}{\beta_1} \dot{p}_{LP2}$$

Considerando anche i volumi morti  $V_{01} = V_{02}$ , che rappresentano le tubature di conduzione dell'olio idraulico nelle due camere di bassa pressione, otteniamo che i nuovi volumi saranno rappresentati come segue:

$$V_1 = V_{01} + x_p A_1 \quad V_2 = V_{02} + (s_{LP} - x_p) A_2$$

Possiamo quindi riscrivere le equazioni in modo tale da ottenere le stesse rappresentate all'interno del sistema.

$$\dot{p}_{LP1} = \frac{\beta_1}{V_{01} + x_p A_1} (Q_{C2} - \dot{x}_p A_1)$$

$$\dot{p}_{LP2} = \frac{\beta_1}{V_{02} + (s_{LP} - x_p) A_2} (Q_{C2} + \dot{x}_p A_2)$$

Infine, l'ultima equazione rappresenta la **dinamica della pressione nella camera di alta pressione**. La formula risulta essere simile all'equazione che descrive la dinamica della seconda camera di bassa pressione; è rappresentata nel seguente modo:

$$\dot{p}_{HP} = \frac{\beta_2}{V_{03} + (s_{HP} - x_p) A_3} (Q_{C3} + \dot{x}_p A_3)$$

I sensori installati ci permettono di rilevare le misure dell'alta pressione ( $p_{HP}$ ) e della posizione del pistone ( $x_p$ ). Inoltre, derivando il secondo termine possiamo risalire alla velocità del pistone ( $\dot{x}_p$ ). Dalle equazioni appena proposte si nota come la dinamica del sistema risulti non lineare a causa della non linearità degli ingressi  $u_1$  e  $u_2$ .



## 1.4 - Implementazione in Simulink

Per implementare le equazioni ricavate precedentemente si è deciso di utilizzare un software della MathWorks, denominato Simulink, all'interno del quale sono state riprodotte le equazioni del modello complessivo e finale del sistema descritto. Il modello è rappresentato da un primo blocco (in azzurro) rappresentante la servovalvola proporzionale Moog, che invia i valori relativi ai flussi in ingresso ( $Q_{c1}$  e  $Q_{c2}$ ), alle due camere di bassa pressione del moltiplicatore di pressione, opportunamente convertiti. Il moltiplicatore di pressione (in giallo) a sua volta elabora le portate ottenute in input dalla valvola e, restituisce in uscita i valori delle pressioni delle due camere di bassa pressione ( $p_{LP1}$  e  $p_{LP2}$ ), il movimento ( $x$ ), la velocità del pistone ( $v$ ) e la pressione all'interno della camera di alta pressione ( $p_{HP}$ ). Le pressioni delle camere di bassa pressione devono essere inviate come ingresso alla valvola per poter gestire il flusso successivamente. L'iniettore è stato considerato come un tappo quindi, troviamo il valore 0 come ingresso al flusso in uscita della camera di alta pressione del moltiplicatore ( $Q_{c3}$ ). Infine, il modello è composto anche da un controllore (in arancio) che contiene un classico controllore PI realizzato precedentemente attraverso un blocco standard della libreria di Simulink. Successivamente il controllore verrà sostituito da un controllore Fuzzy PI descritto nei prossimi capitoli.

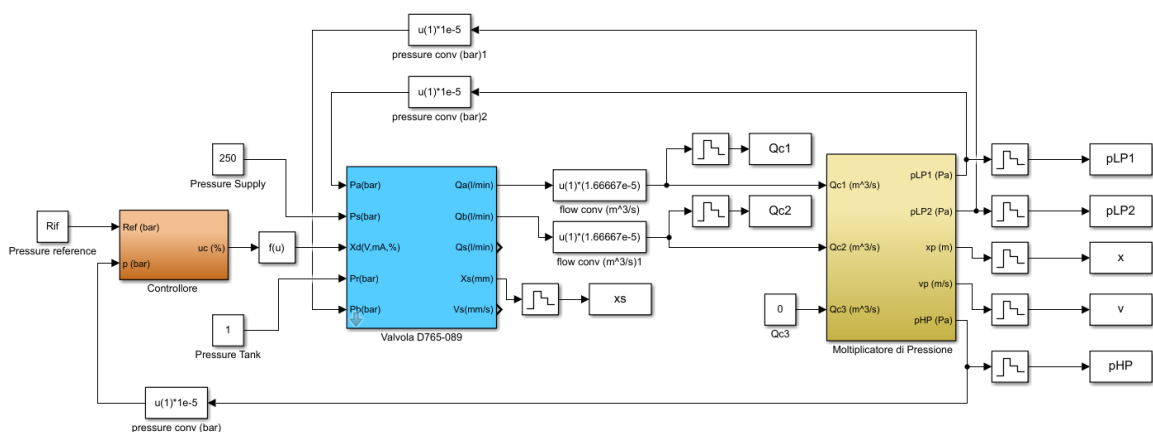


Figura 5 Modello complessivo in ambiente Simulink

Il moltiplicatore di pressione è composto dalle equazioni  $f_1(x, u)$  e  $f_2(x, u)$  descritte nel blocco dinamica pistone mentre, nei blocchi dinamica della pressione delle camere LP1, LP2 e HP troviamo l'implementazione delle equazioni  $f_3(x, u)$ ,  $f_4(x, u)$  e  $f_5(x, u)$ .

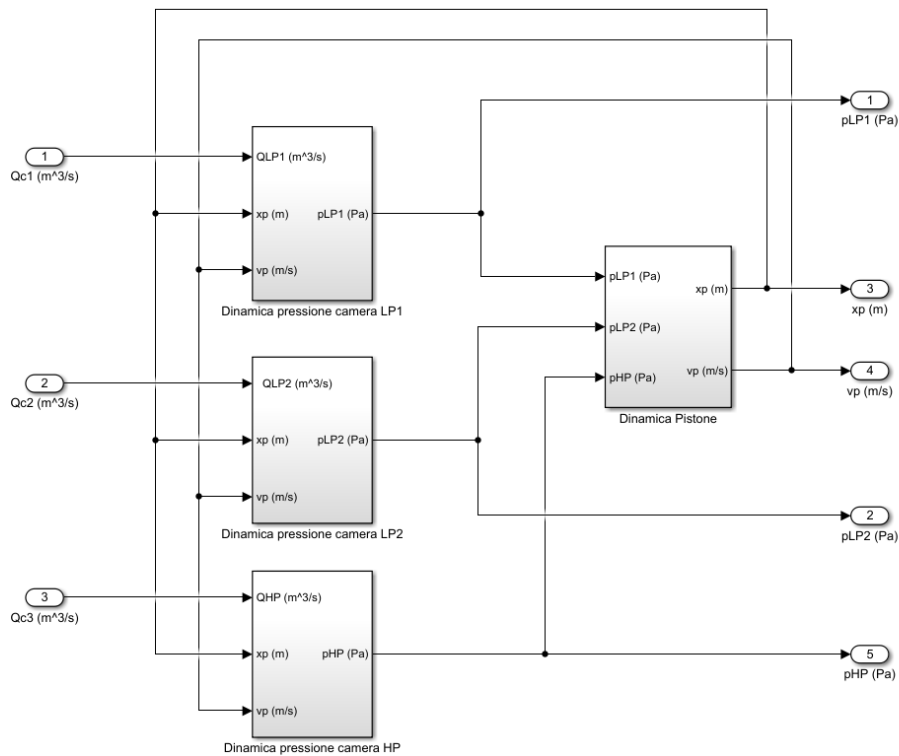


Figura 6 Equazioni del moltiplicatore di pressione in Simulink

Il modello della servovalvola Moog D765-098 è stato implementato dalla casa costruttrice e perciò forniamo direttamente una descrizione degli ingressi e delle uscite.

Ingressi		Uscite	
$p_a(\text{bar})$	Pressione alla porta A	Flusso alla porta A	$Q_a(\text{l} / \text{min})$
$p_s(\text{bar})$	Pressione di alimentazione (porta P)	Flusso alla porta B	$Q_b(\text{l} / \text{min})$
$x_d(\text{V})$	Segnale di comando	Flusso di alimentazione (porta P)	$Q_s(\text{l} / \text{min})$
$p_r(\text{bar})$	Pressione di ritorno (Porta T)	Posizione della spoletta	$x_s(\text{mm})$
$p_b(\text{bar})$	Pressione alla porta B	Velocità della spoletta	$v_s(\text{mm} / \text{s})$

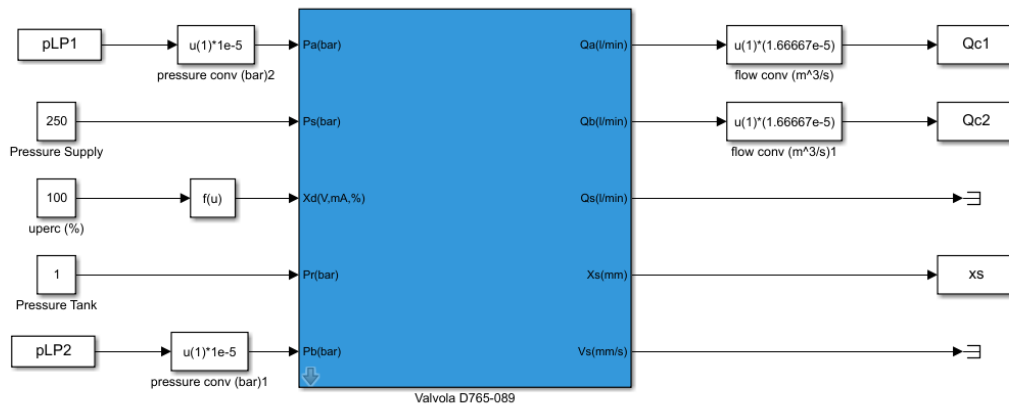


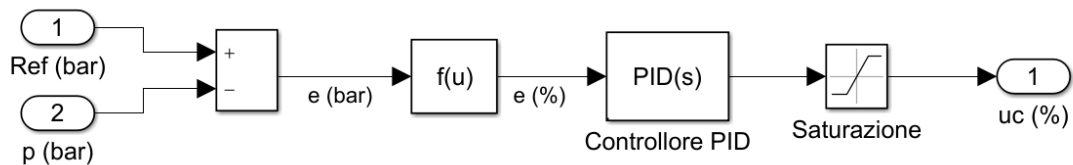
Figura 7 Rappresentazione valvola Moog D765-089 in Simulink

Nel nostro caso, per i valori in ingresso, è stata considerata una pressione di alimentazione dell'olio idraulico di  $250 \text{ bar}$  e una pressione di ritorno di  $1 \text{ bar}$ . La pressione di ritorno può essere impostata al valore della pressione atmosferica poiché le tubazioni, come descritto precedentemente, sono state considerate come sistemi statici. Le porte A e B corrispondono alle pressioni  $p_{LP1}$  e  $p_{LP2}$ , uscenti dal modello del moltiplicatore e rappresentanti le pressioni nelle due camere di bassa pressione. Come ultimo ingresso troviamo il segnale di controllo  $u$  che, essendo in forma percentuale, viene convertito in un segnale di tensione  $x_d$  attraverso la funzione  $f(u)$ , prima di essere inserito all'interno del modello della valvola. Riguardo le uscite, troviamo i flussi  $Q_a$  e  $Q_b$  che rappresentano i flussi di controllo  $Q_{c1}$  e  $Q_{c2}$  delle due camere di bassa pressione presenti all'interno del moltiplicatore di pressione, opportunamente convertiti da  $l/min$  a  $m^3/s$ . Inoltre, è possibile conoscere il movimento effettuato dalla spoletta attraverso la variabile  $x_s$  e la sua velocità attraverso  $v_s$ . Il flusso di alimentazione  $Q_s$  della porta P non viene utilizzato nel modello.

Riportiamo ora le principali caratteristiche della servovalvola proporzionale estratte dal catalogo tecnico.

<b>Pressioni di esercizio</b>	
<b>Porta X, P, A, B</b>	315 bar
<b>Porta T</b>	210 bar
<b>Portata nominale (<math>\pm 10\%</math>)</b>	50 l / min (a 70 bar)
<b>Tempo di risposta</b>	4 ms
<b>Tensione di alimentazione</b>	$\pm 15 V DC$
<b>Segnale di comando</b>	$\pm 10 V DC$

Il controllore come già anticipato, è stato realizzato utilizzando il blocco standard della libreria Simulink. All'interno, troviamo una differenza tra il riferimento e la pressione misurata dal sensore, dalla quale ricaviamo l'errore. L'errore a sua volta viene percentualizzato rispetto al fondo scala di 1280 Bar all'interno della funzione  $f(u)$  e, successivamente inserito all'interno del controllore PID, opportunamente tarato e saturato, per ottenere lo sforzo di controllo necessario in uscita.



**Figura 8** Controllore PID

Prima di descrivere il funzionamento del toolbox di Simulink utilizzato per realizzare il nuovo controllore basato sulla logica Fuzzy, nel capitolo successivo introduciamo una descrizione riguardante gli insiemi Fuzzy e tutte le proprietà collegate ad essi.

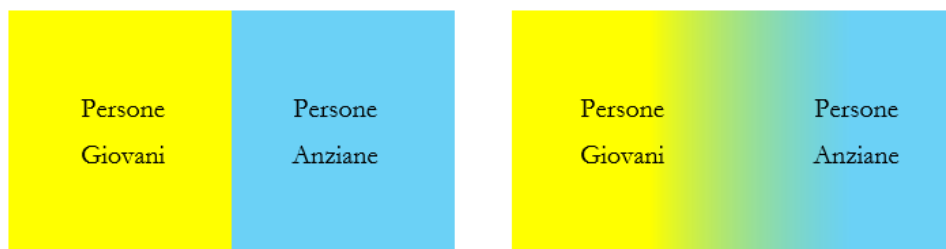
## Capitolo 2

### Logica Fuzzy

In questo capitolo chiariremo il concetto di “Logica Fuzzy” che risulta fondamentale per la comprensione dell’elaborato. La logica Fuzzy si basa sulla “Fuzzy Set Theory”, introdotta in una pubblicazione del 1965 da Lofti A. Zadeh, e poi formalizzata in maniera definitiva nel 1973. L’obiettivo definitivo della logica introdotta da Zadeh è quello di descrivere le modalità di ragionamento della mente umana e quindi di superare la rigidità della logica classica. La mente umana infatti, quando si presenta un problema nella vita di tutti i giorni, tenta di schematizzarlo semplificando il problema per poi trovare una soluzione in base a informazioni espresse in termini vaghi o sfumati che vengono appunto tradotti in inglese con “*Fuzzy*”. Nella maggior parte dei casi, le classi di oggetti incontrati nella vita reale, risultano soggettivi e impossibili da quantificare o classificare secondo precisi criteri di appartenenza. L’idea della logica Fuzzy è che una grandezza non possa assumere solo valori booleani (vero o falso), e quindi far parte di un insieme perfettamente definito (*Crisp*). Al contrario, secondo la stessa logica, una grandezza può assumere un insieme di valori indicanti il livello di “verità” di una certa espressione e perciò, il suo grado di appartenenza a una determinata classe.

Per comprendere al meglio la logica Fuzzy, riprendiamo un esempio che si basa sulla classificazione di persone in base alla loro età. Consideriamo che una persona possa appartenere a uno dei due insiemi: *Giovane* o *Anziana*. Secondo la logica classica, bisognerebbe inserire dei valori di soglia affinché si stabilisca il limite tra gli insiemi, in modo tale da suddividere il campo di valori in due fasce, dove ogni persona potrà appartenere ad una sola di queste. Stabilendo la soglia di divisione tra gli insiemi a 50 anni, una persona di 20 anni viene definita *Giovane* mentre, una con un’età di 55 anni *Anziana*. In questo caso però avremo che una persona di 49 anni viene definita *Giovane* e appartiene allo stesso insieme di quella di 20 anni, anche se presenta una differenza maggiore con quella *Giovane*, precisamente di 29 anni, e una differenza di 6 anni e quindi minore con la persona *Anziana*. Dal punto di vista logico tutto ciò non risulta soddisfacente. Formulando lo stesso esempio considerando la logica Fuzzy e quindi,

considerando gli insiemi Fuzzy, esso risulterà molto più vicino alla mente umana. Infatti, non avremmo più solamente persone *Giovani* o *Anziane* ma, ogni persona apparterrà ad un insieme secondo un certo grado. Avremo quindi persone *abbastanza Anziane* o *piuttosto Giovani*. Riprendendo l'esempio precedente, una persona di 49 anni non verrà più considerata giovane al 100% ma apparterrà all'insieme delle persone anziane con un grado del 35%. Quella di 55 anni avrà un grado di appartenenza maggiore (45%), così come quella di 20 anni apparterrà all'insieme delle persone anziane con un grado del 5%. In questo caso ognuna di queste persone viene considerata *Giovane* o *Anziana* con un certo grado di appartenenza. Preso quindi un insieme di persone, all'aumentare dell'età diminuirà l'appartenenza all'insieme delle persone *Giovani* e aumenterà l'appartenenza all'insieme delle persone *Anziane*. Nelle immagini sottostanti viene descritta la differenza tra un insieme definito (*Crisp*) e un insieme sfumato (*Fuzzy*).



**Figura 9** Insieme Crisp ed insieme Fuzzy

## 2.1 - Definizioni

La teoria degli insiemi Fuzzy fornisce gli strumenti matematici per effettuare ragionamenti quando le informazioni disponibili sono insicure o imprecise. Tuttavia, non sono altro che un'estensione degli insiemi classici.

Sia  $X$  uno spazio di punti (oggetti) e  $x$  un generico elemento di  $X$ . Se prendiamo un insieme tradizionale  $A$  in  $X$  esso può essere identificato con una funzione con dominio  $X$  e codominio l'insieme dei due elementi  $\{0,1\}$ :

$$A(x) = \begin{cases} 1 & \text{se } x \in A \\ 0 & \text{se } x \notin A \end{cases}$$

La funzione descritta è denominata funzione caratteristica dell'insieme A e ci permette di distinguere se un elemento appartiene o no all'insieme e cioè, se si trova o no all'interno della soglia determinata. Analogamente, un insieme Fuzzy è caratterizzato da una funzione di appartenenza  $f_A(x)$ , la quale associa ad ogni punto  $x \in X$  un numero reale nell'intervallo  $[0,1]$  ( $f_A(x): X \rightarrow [0,1]$ ). Attraverso il valore acquisito da  $f_A(x)$  si rappresenta il “grado di appartenenza” di x ad A, naturalmente essendo un insieme  $[0,1]$  più il valore si avvicina ad 1, più alto sarà il grado di appartenenza a quel determinato insieme. In particolare,  $A(x) = 0$  significa che l'elemento x non appartiene all'insieme A,  $A(x) = 1$  che l'elemento appartiene interamente ad A, e un valore intermedio che l'elemento appartiene solo in parte all'insieme.

Attraverso la funzione di appartenenza quindi si può separare il caso in cui si lavori in un ambiente Fuzzy, nel quale  $f_A(x)$  può assumere qualsiasi valore all'interno dell'intervallo  $[0,1]$ , o in un ambiente classico o definito (crisp), nel quale  $f_A(x)$  potrà assumere solo i valori 0 o 1.

## 2.2 – Funzioni di appartenenza

Nel caso esaminato precedentemente le funzioni di appartenenza dei due insiemi “persone giovani” e “persone anziane” potrebbero avere la seguente forma.

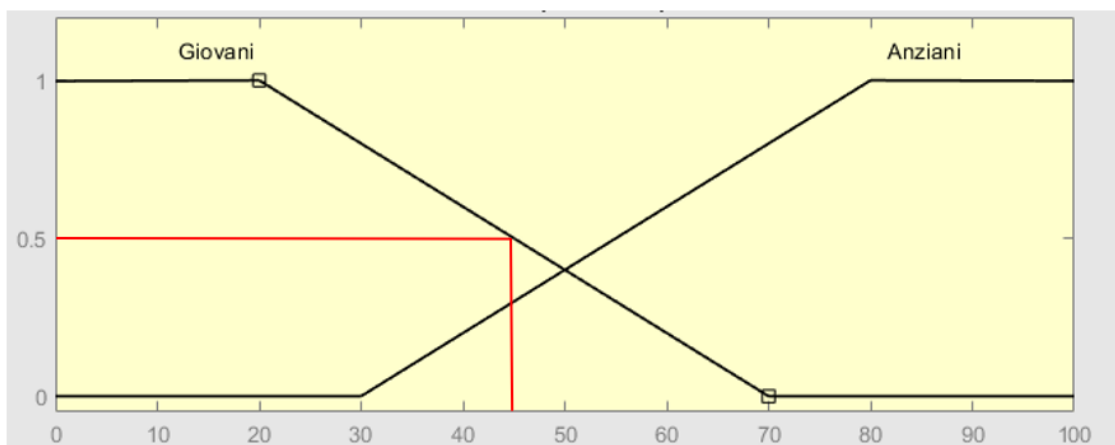
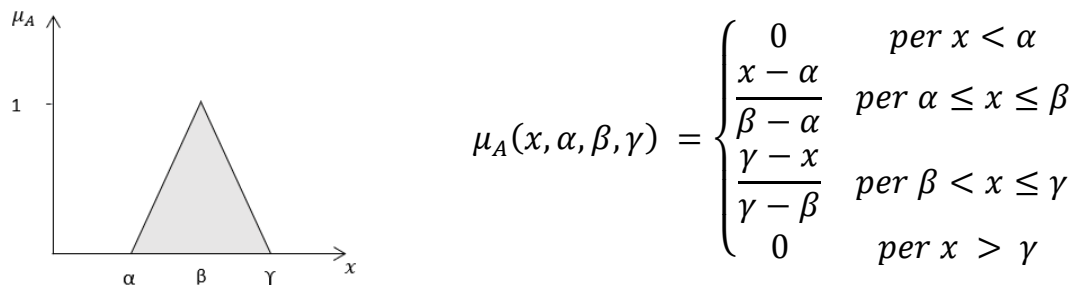


Figura 10 Esempio funzione di appartenenza

In questo caso l'ascissa corrisponde al grado di appartenenza mentre, l'ordinata all'età della persona presa in oggetto. Ad esempio, una persona di 45 anni apparterrà all'insieme delle persone *Giovani* al 50%. Una notazione molto usata per rappresentare la funzione di appartenenza è  $\mu_A: X \rightarrow [0,1]$  che nel nostro caso diventa:  $\mu_A(45) = 0,5$

Si possono definire funzioni di appartenenza diverse tra loro che variano a seconda della loro applicazione. Successivamente andiamo a descrivere tre tipi di funzioni anticipando però, che le più usate rimangono le funzioni triangolari e trapezoidali.

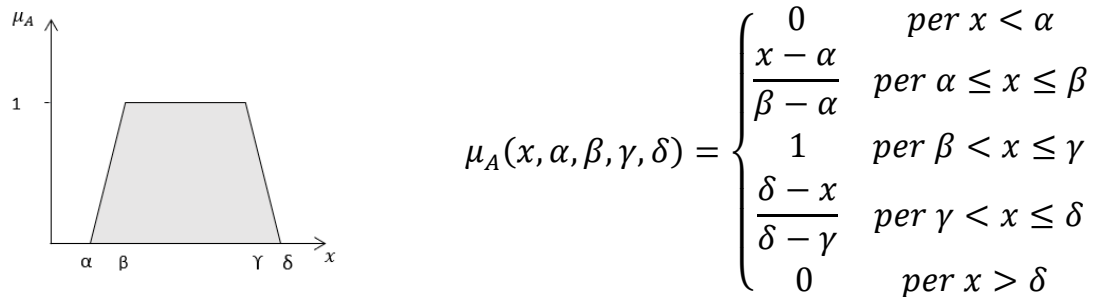
- La **funzione di appartenenza triangolare** è definita da due estremi  $\alpha$  e  $\delta$  e un punto di massimo  $\beta$ . La sua caratteristica principale è che un solo elemento presenta un grado di appartenenza massimo e quindi pari a 1, mentre gli altri hanno un grado inversamente proporzionale alla distanza dal punto di massimo. Per calcolare il grado di appartenenza in ogni punto, considerando la funzione in figura si utilizzano le seguenti equazioni:



**Figura 11** Funzione di appartenenza triangolare con relative equazioni

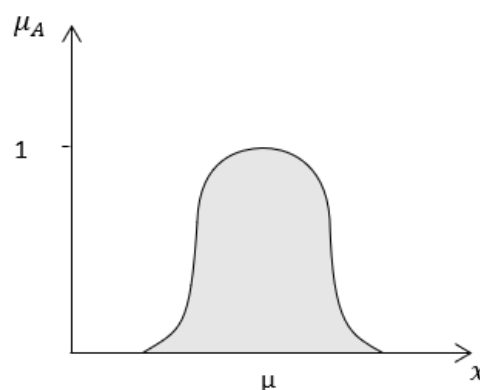


- La **funzione di appartenenza trapezoidale** invece è rappresentata da due parametri,  $\alpha$  e  $\delta$ , che rappresentano la base maggiore del trapezio e altri due,  $\beta$  e  $\gamma$ , che rappresentano la base minore. La differenza principale è che la forma trapezoidale, rispetto a quella triangolare, ammette più elementi con grado massimo di appartenenza all'insieme in questione.



**Figura 12** Funzione di appartenenza trapezoidale con relative equazioni

- La **funzione di appartenenza a campana** invece si presenta come variante alla funzione a triangolo ottenuta, usando archi di parabola oppure una gaussiana. Presenta un unico elemento con grado massimo, proprio come la funzione di appartenenza triangolare ma, i restanti elementi non avranno un grado di appartenenza derivato da segmenti ma bensì da una funzione.



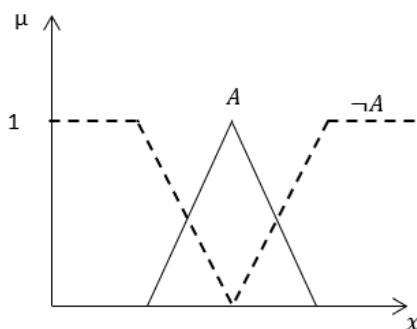
**Figura 13** Funzione di appartenenza gaussiana

La scelta della funzione di appartenenza è fondamentale in un sistema Fuzzy perché determina le caratteristiche dei processi di fuzzificazione degli ingressi e defuzzificazione delle uscite. Con il primo processo, cioè la fuzzificazione, si associa ad ogni elemento in ingresso il proprio grado di appartenenza all'insieme Fuzzy. La defuzzificazione invece, calcola un valore da associare alla variabile di uscita a partire dal risultato precedente. Le funzioni di appartenenza devono essere tali da non lasciare alcuna lacuna all'interno dell'insieme e quindi tali da ricoprire l'intera area. Per evitare che alcune parti dell'universo del discorso rimangano scoperte si possono sovrapporre parzialmente le aree delle funzioni di appartenenza. In questi casi per decretare il grado di appartenenza e successivamente i valori di uscita bisogna introdurre le operazioni sugli insiemi Fuzzy.

### 2.3 - Operazioni sugli insiemi Fuzzy

Gli operatori utilizzati per combinare più insiemi Fuzzy tra di loro sono chiamati connettivi Fuzzy od operatori di aggregazione. Gli operatori sono gli stessi operatori insiemistici utilizzati nei normali insiemi: l'intersezione (AND), l'unione (OR) e il complemento (NOT).

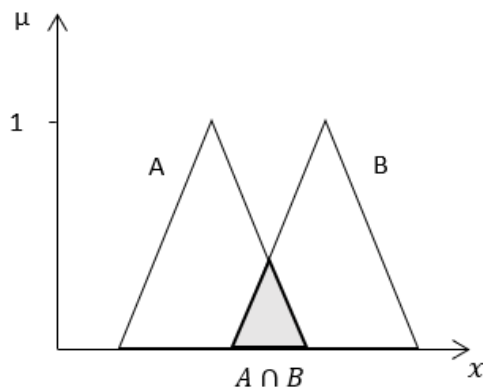
Negli insiemi Fuzzy l'**operatore di complemento** indica il grado di non appartenenza all'insieme. Se consideriamo A un insieme Fuzzy, il suo complemento  $\neg A$  è:



$$\mu_{\neg A}(x) = 1 - \mu_A(x)$$

**Figura 14** Operatore di complemento

L'**operatore di intersezione** tra due insiemi A e B viene rappresentata come l'insieme Fuzzy più grande che contiene sia A che B. Quindi l'intersezione tra due insiemi A e B con funzioni di appartenenza rispettivamente  $f_A(x)$  e  $f_B(x)$  è rappresentata da un insieme C con funzione di appartenenza  $f_C(x)$  definita come:



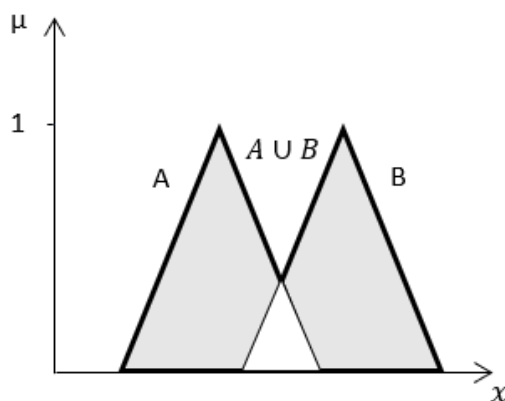
$$f_C(x) = \min[f_A(x), f_B(x)] \text{ con } x \in X$$

o in forma abbreviata

$$f_C(x) = f_A(x) \wedge f_B(x)$$

**Figura 15** Operatore di intersezione

L'**operatore di unione** tra due insiemi A e B è definito come l'insieme Fuzzy più piccolo che contiene sia A che B. Se prendiamo i due insiemi con funzione di appartenenza rispettivamente  $f_A(x)$  e  $f_B(x)$  l'unione è rappresentata da un insieme C con funzione di appartenenza  $f_C(x)$  definita come:



$$f_C(x) = \max[f_A(x), f_B(x)] \text{ con } x \in X$$

o in forma abbreviata

$$f_C(x) = f_A(x) \vee f_B(x)$$

**Figura 16** Operatore di unione

## 2.4 – Regole Fuzzy

La base della conoscenza di un sistema Fuzzy è costituita sia dalle funzioni di appartenenza delle variabili che abbiamo precedentemente introdotto, sia dall'insieme delle regole d'inferenza Fuzzy che rappresentano il passaggio tra le conoscenze di tipo empirico e la loro elaborazione numerica. Le regole risultano essere una descrizione formale del sistema ma, quando vengono inserite nello stesso contesto degli insiemi Fuzzy e delle funzioni di appartenenza permettono di fornire un modello del sistema puramente numerico.

Le regole vengono solitamente espresse attraverso costrutti *IF – THEN*, all'interno del quale possiamo trovare anche operatori logici. Una regola composta da un solo input e un solo output viene espressa nella seguente forma:

**IF X is A THEN Y is B**

La seguente regola può essere rappresentata anche come un'implicazione, che nella logica Fuzzy, non rappresenta un'implicazione logica classica che mette in relazione gli elementi attraverso una propria tabella di verità ma, una relazione Fuzzy sugli insiemi A e B presi in considerazione.

$$\mu_{A \rightarrow B}(x, y) = \mu_a(x) \mathfrak{I} \mu_b(y)$$

Dove  $\mathfrak{I}$  è l'operatore di implicazione. Esistono due funzioni d'implicazione che risultano essere le più utilizzate e le più semplici **min** e **product** che utilizzano come operatore di implicazione rispettivamente la norma triangolare di minimo e quella di prodotto algebrico.

- Implicazione fuzzy **min**:  $\mu_{A \rightarrow B}(x, y) = \min(\mu_a(x), \mu_b(y))$
- Implicazione fuzzy **product**:  $\mu_{A \rightarrow B}(x, y) = \mu_a(x) * \mu_b(y)$

Solitamente le regole però presentano più ingressi o più uscite e perciò bisogna introdurre i connettivi logici **and** e **or** per quanto riguarda gli ingressi e il connettivo **also** per le uscite. La forma generale di una regola avente, *n* ingressi e *m* uscite, è la seguente:

**IF**( $x_1$  is  $A_{k_1}$ )**AND**...**AND**( $x_1$  is  $A_{k_i}$ )**OR**...**OR**( $x_n$  is  $A_{k_n}$ )**THEN** ( $y_1$  is  $B_{k_1}$ )**ALSO**...**ALSO**( $y_m$  is  $B_{k_m}$ )

## 2.5 - Struttura di un sistema Fuzzy

Un sistema Fuzzy è costituito dalle seguenti unità: la base di conoscenza, la fuzzificazione, l'interferenza e la defuzzificazione, dove le ultime tre rappresentano le unità di calcolo.

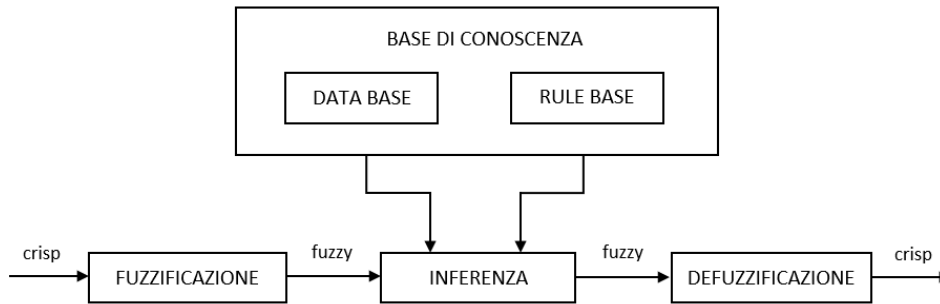
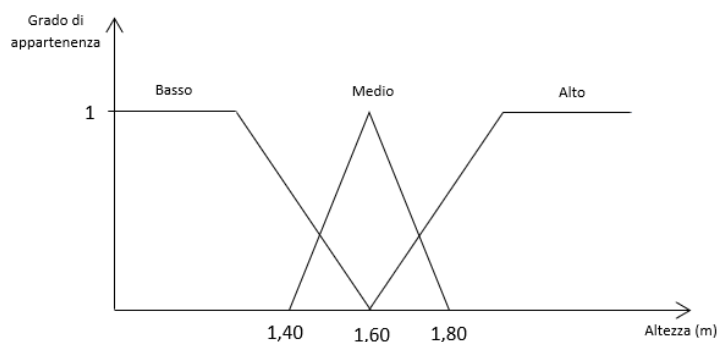


Figura 17 Struttura sistema Fuzzy

Attraverso la **base di conoscenza**, che contiene tutte le informazioni sul sistema, le altre tre unità di calcolo riescono ad elaborare gli ingressi in modo da ottenere le uscite. All'interno le informazioni possono essere suddivise in data base e rule base, dove il primo comprende la descrizione di tutte le variabili con le loro funzioni di appartenenza e il secondo le regole linguistiche d'inferenza.

La **fuzzificazione** viene utilizzata per convertire un dato numerico in un dato fuzzy. L'operazione è necessaria poiché i dati di ingresso vengono espressi sotto forma di insiemi limitati (Crisp) mentre, il sistema lavora su insiemi sfumati (Fuzzy). La conversione viene effettuata utilizzando le funzioni di appartenenza degli insiemi Fuzzy relativi alla variabile presa in considerazione. Ad ogni valore limitato viene attribuito un grado di appartenenza relativo ad un termine linguistico della variabile. Per termine linguistico di una variabile si intendono i valori non numerici e quindi linguistici che una variabile può assumere. Ad esempio, presa la variabile "altezza", si possono assegnare i seguenti valori: "alta", "bassa", "media". Ad ognuno di questi valori viene assegnata una funzione di appartenenza e l'insieme di tutte le funzioni deve ricoprire l'universo in cui è stata considerata la variabile. È importante ricordare che i valori possono essere rappresentati da funzioni di appartenenza diverse tra loro.



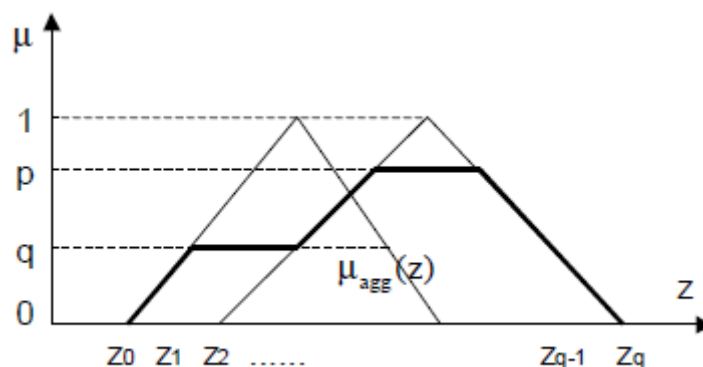
**Figura 18** Termini linguistici con funzioni di appartenenza

Il motore d'**inferenza** utilizza le informazioni contenute nella base di conoscenza per determinare lo stato delle uscite corrispondenti a determinate configurazione di ingressi. La **defuzzificazione** prende un insieme Fuzzy ottenuto attraverso l'inferenza e ne determina un valore crisp e quindi limitato, che rappresenti nel miglior modo l'uscita finale del sistema. In particolare, durante la fuzzificazione si convertono valori Fuzzy d'uscita dall'inferenza in valori numerici utilizzabili.

Esistono diverse strategie di defuzzificazione e spetta al progettista scegliere quale sia più consona alle sue esigenze poiché non esiste una strategia principale e standard da utilizzare in tutti i casi. Le più utilizzate rimangono il metodo del baricentro e il metodo di massimo.

- **Metodo del centro di gravità (o centroide o baricentro)**

È il metodo più usato e si suppone che l'aggregazione delle regole produca la seguente funzione di appartenenza  $\mu_{agg}(z)$ ,  $z \in [z_0, z_q]$ :



**Figura 19** Metodo del centro di gravità

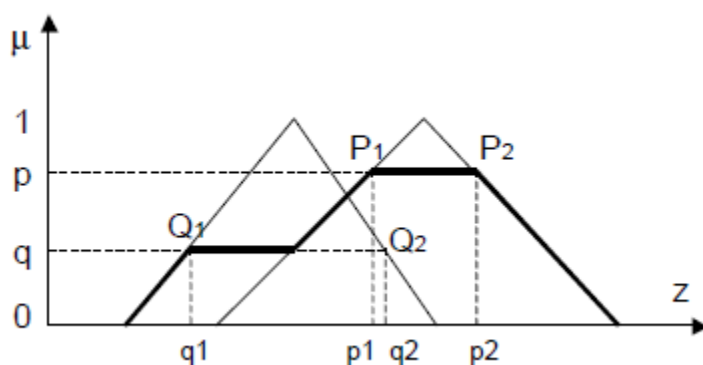
Una volta suddiviso l'intervallo  $[z_0, z_q]$  in  $q$  sottointervalli uguali mediante i punti  $z_1, z_2, \dots, z_{q-1}$  possiamo ricavare il valore numerico cercato  $\tilde{z}$ , che è rappresentato più specificatamente dalla media pesata dei numeri  $z_k$ , dove il peso è  $\mu_{agg}(z_k)$ :

$$\tilde{z} = \frac{\sum_{k=1}^{q-1} z_k \mu_{agg}(z_k)}{\sum_{k=1}^{q-1} \mu_{agg}(z_k)}$$

Esistono due modi di interpretare il valore numerico, una è l'interpretazione geometrica, e cioè  $\tilde{z}$  rappresenta l'ascissa del centro  $(\tilde{Z}, \tilde{u})$  dell'area sotto la curva  $\mu_{agg}(z)$  limitata dall'asse  $z$ , e l'altra è l'interpretazione fisica, dove si presuppone che l'area fosse ottenuta ritagliando un pezzo di legno o plastica e, il centro di tale area sarebbe il centro di gravità.

- **Metodo della media dei massimi**

Se consideriamo la stessa funzione di appartenenza precedente  $\mu_{agg}(z)$ ,  $z \in [z_0, z_q]$ , notiamo che presenta due segmenti paralleli all'asse  $z$ . Proiettando il segmento  $P_1P_2$  (alla massima altezza) sull'asse  $z$  possiamo calcolare il punto medio del segmento che è dato da  $\tilde{z}$ .



$$\tilde{z} = \frac{P_1 + P_2}{2}$$

Figura 20 Metodo della media dei massimi

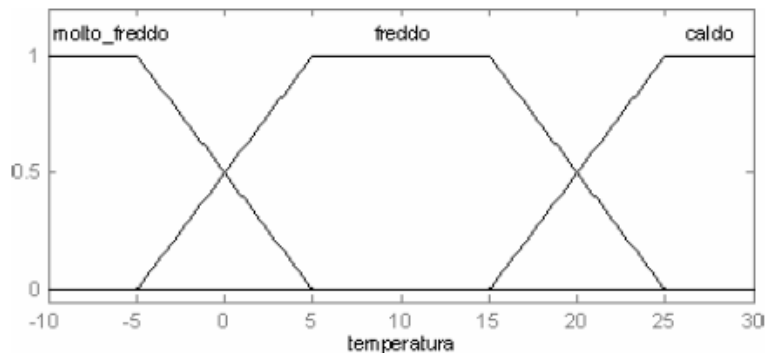
- **Metodo della media pesata**

È una generalizzazione del metodo precedente che considera tutte le proiezioni dei segmenti paralleli all'asse z.

$$\tilde{z} = \frac{p \frac{p_1 + p_2}{2} + q \frac{q_1 + q_2}{2}}{p + q} = w_1 \frac{p_1 + p_2}{2} + w_2 \frac{q_1 + q_2}{2}$$

Nel dettaglio  $\tilde{z}$  è rappresentata dalla media pesata dei punti medi dei segmenti  $[p_1, p_2]$  e  $[q_1, q_2]$  rispetto ai pesi  $w_1 = \frac{p}{p+q}$  e  $w_2 = \frac{q}{p+q}$ , dove  $p$  e  $q$  rappresentano le altezze dei due segmenti. Naturalmente, la formula può essere estesa anche ai casi in cui i segmenti paralleli all'asse z sono molteplici.

Introduciamo un esempio del procedimento da effettuare. Se consideriamo tre insiemi Fuzzy “molto freddo”, “freddo” e “caldo” con le rispettive funzioni di appartenenza definite nell'immagine seguente, è possibile “fuzzificare” e quindi, individuare un qualsiasi valore numerico relativo alla temperatura.

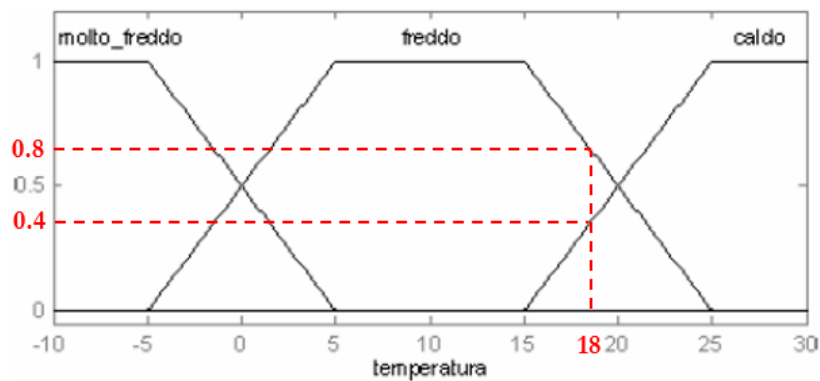


**Figura 21** Funzioni di appartenenza della variabile “temperatura”

Se consideriamo in termini numerici l'espressione “ho una temperatura di 18°C” sappiamo che questa equivale all'espressione in termini Fuzzy “Fa freddo all'60% e caldo al 40%”. In particolare, avendo in ingresso il valore 18, la variabile numerica della temperatura assumerà il valore 18, mentre la variabile Fuzzy d'ingresso, denominata “temperatura”, assumerà il valore:

$$[\mu_{moltofreddo}(18), \mu_{freddo}(18), \mu_{caldo}(18)] = [0, 0.6, 0.4]$$

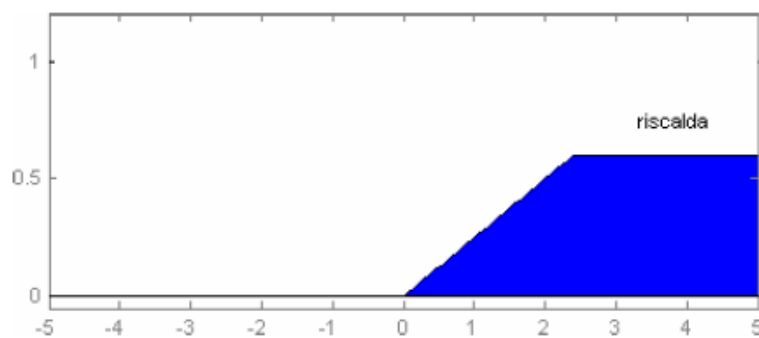




**Figura 22** Esempio di fuzzificazione

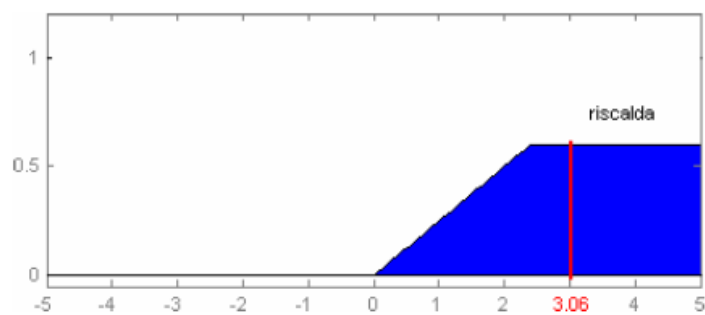
L'operazione di defuzzificazione calcola il valore numerico di una variabile linguistica a partire dal suo valore linguistico. Un'applicazione di questo procedimento si può trovare nella risoluzione di un problema di controllo dove, alla fine, invece del valore linguistico si desidera ottenere il valore numerico per regolare il controllore. La defuzzificazione opera in maniera inversa rispetto alla fuzzificazione poiché viene applicata all'uscita invece che all'ingresso del sistema. Successivamente introdurremo il controllo Fuzzy PI implementato utilizzando la logica Fuzzy e descriveremo, come i valori linguistici di uscita vengono decisi in base alle regole applicate sui valori linguistici di ingresso.

Per comprendere al meglio la spiegazione introduciamo il seguente esempio. Consideriamo di controllare una caldaia all'interno di un'abitazione e che il risultato ottenuto dall'applicazione della logica Fuzzy sia l'insieme "riscalda la casa" rappresentato nell'immagine successiva.



**Figura 23** Risultato dell'elaborazione Fuzzy

Il risultato ottenuto ci evidenzia che per aumentare la temperatura all'interno dell'abitazione è necessario attivare la caldaia. Il problema fondamentale è che non conosciamo il valore da impostare alla caldaia affinché la temperatura aumenti. Il procedimento per ricavare il valore numerico è appunto, la defuzzificazione. In questo caso se utilizzassimo il metodo del centro di gravità otterremo come risultato che il valore da impostare alla caldaia per aumentare la temperatura, si trova sull'ascissa del baricentro della funzione di appartenenza, come rappresentato in figura.



**Figura 24** Rappresentazione centro di gravità

# Capitolo 3

## Descrizione controlli PID e Fuzzy PID

Nell'ingegneria del controllo, un sistema controllato è caratterizzato principalmente dal suo comportamento dinamico che, determina l'ambito richiesto per risolvere un compito di controllo e, dalla sua risposta al gradino, utilizzata per riflettere questo comportamento dinamico. La risposta al gradino misura la variabile controllata dopo una variazione della variabile misurata rilevando la reazione della variabile di controllo rispetto ai cambiamenti di quella misurata. Le prestazioni del controller sono ottenute attraverso lo studio e l'analisi delle caratteristiche transitorie della risposta a gradino come tempo di salita, tempo di assestamento, errore di stato, ampiezza di picco, superamento, ecc.... I convenzionali controlli PID sono stati ben sviluppati e sono ampiamente utilizzati per l'automazione industriale e il controllo di processo a causa della loro semplicità di funzionamento e della facilità di progettazione. I sistemi non lineari e i sistemi particolarmente complessi e vaghi che non hanno modelli matematici precisi, presentano però alcune lacune. Per controllare sistemi complessi e non lineari si implementano controllori Fuzzy PID; all'interno dei quali i parametri dei guadagni proporzionale, integrale e derivativo vengono calcolati utilizzando regole Fuzzy.

### 3.1 - Controllore PID

Il controllore PID è il più utilizzato e il più popolare nel settore industriale per la sua semplicità di utilizzo e di implementazione. La popolarità è dovuta al fatto che i regolatori PID non richiedono una dettagliata conoscenza del sistema e possono essere configurati e tarati automaticamente utilizzando semplici regole. Lo scopo alla base di questo controllore è di rendere la differenza di errore, che si trova tra il segnale desiderato (riferimento) e il segnale effettivo, la più piccola possibile, ovvero porla a zero.

Matematicamente un controllore PID può essere descritto come la somma di tre contributi che verrà poi inserita in ingresso al sistema da controllare:

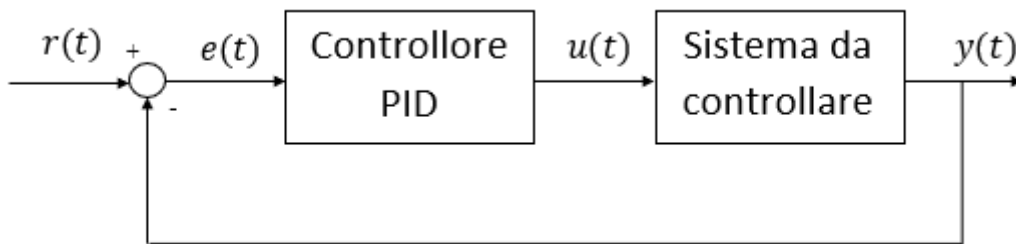
$$u(t) = K_p e(t) + K_i \int_0^t e_\tau d\tau + K_d \frac{d}{dt} e(t)$$

$K_p$ ,  $K_i$  e  $K_d$  rappresentano rispettivamente i guadagni proporzionali, integrali e derivati. Il termine proporzionale, come si può dedurre dal nome, produce un valore di output che è proporzionale al valore di errore attuale. Questo termine ha lo scopo di contrastare direttamente l'errore ed infatti è connesso direttamente allo stato della variabile di processo. In particolare, più l'uscita del sistema si distacca dal riferimento maggiore dovrà essere il suo rapporto per poter contrastare il movimento. Viceversa, se l'uscita si avvicina al riferimento, la sua intensità dovrà diminuire. Il termine integrale è proporzionale al valore medio dell'errore commesso e quindi alla durata di esso. Viene aggiunto al controllore per ridurre ed infine eliminare l'errore che si genera al riferimento quando abbiamo un controllo costituito da un solo proporzionale. Aggiungendo questo termine si accelera il movimento del sistema verso il riferimento, ottenendo un'ottima convergenza a regime. Il termine derivativo ha lo scopo di anticipare l'andamento dell'errore negli istanti successivi, facendo sì che il sistema abbia una maggiore prontezza e risulti più veloce alle variazioni. Attraverso lo studio della derivata dell'errore possiamo prevedere se il segnale di uscita si discosta dal segnale di riferimento oppure si avvicina intervenendo tempestivamente e correggendo il controllore. Una variazione positiva della derivata dell'errore comporta un aumento dell'errore nel tempo e quindi un maggiore distacco dal riferimento. Al contrario una variazione negativa ci indica che il sistema si sta avvicinando al riferimento. Chiaramente, nel controllore non devono essere presenti tutti i guadagni ma è possibile utilizzare delle combinazioni, come ad esempio, un controllore P, PI o PD.

L'errore rappresentato nella formula precedente è calcolato considerando la differenza tra il valore misurato ( $r(t)$ ) e l'uscita del sistema ( $y(t)$ ).

$$e(t) = r(t) - y(t)$$

Introduciamo, ora, una rappresentazione generale a blocchi di un sistema ad anello chiuso contenente un controllore PID:



**Figura 25** Sistema ad anello chiuso con controllore PID

Possiamo andare a studiare gli effetti sulla risposta provocati dall'aggiunta dei guadagni proporzionale, integrale e derivativo al controllore.

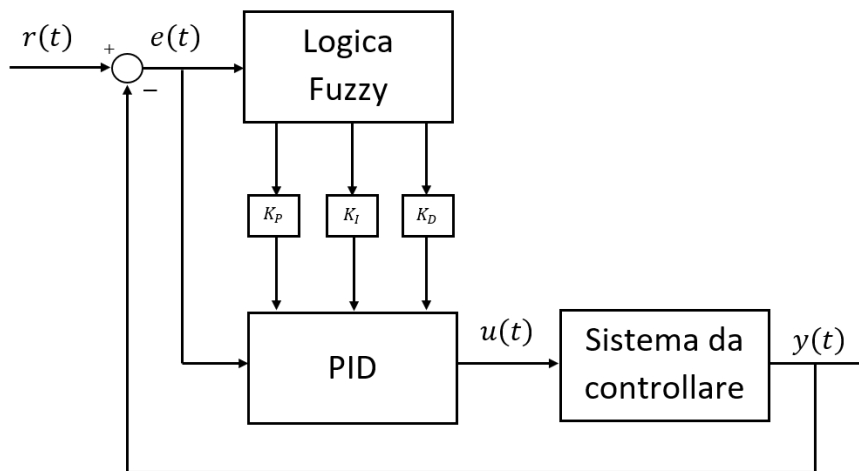
<b>Guadagni PID</b>	<b>Tempo di salita</b>	<b>Sovraelongazioni</b>	<b>Tempo di assestamento</b>	<b>Errore di stabilizzazione dell'uscita</b>
$K_p$	Diminuisce	Aumentano	Invariato	Diminuisce
$K_i$	Diminuisce	Aumentano	Aumenta	Eliminato
$K_d$	Invariato	Diminuiscono	Diminuisce	Invariato

Per una buona progettazione di un controllore PID, per prima cosa è necessario inserire un regolatore composto unicamente da un guadagno proporzionale. Una volta che tale controllore è stato tarato e quindi raggiunge il riferimento, è possibile inserire gli altri due guadagni. Di solito viene inserito prima il guadagno integrale, e infine il guadagno derivativo, andando, naturalmente, ogni volta a tarare i valori per

fare in modo che il sistema raggiunga il riferimento in tempi ragionevoli e con minime sovraelongazioni. Il controllore PID, però, non funziona bene in alcuni casi : in sistemi con grandi incertezze o variazioni, sistemi MIMO dove il coordinamento è importante, sistemi con modalità oscillatorie leggermente smorzate, ecc.... Per superare queste limitazioni è necessario utilizzare dei metodi di controllo più sofisticati basati sulla conoscenza come il controllo Fuzzy PID.

### 3.2 - Controllore Fuzzy PID

I controllori Fuzzy PID svolgono lo stesso compito dei classici controllori ma, riescono a gestire problemi di controllo complessi attraverso euristiche e modelli matematici forniti dalla logica Fuzzy anziché, da modelli matematici prodotti da equazioni differenziali. A differenza della logica classica, si dice che la logica Fuzzy sia tollerante all'incertezza e all'imprecisione. Ciò semplifica l'implementazione di controllori su modelli non lineari rispetto alle altre tecniche classiche o convenzionali. Integrando adeguatamente la logica Fuzzy al controllore PID è possibile ottenere una maggiore precisione statica e una risposta dinamica più rapida. Il controllore Fuzzy PID utilizza quindi il sistema di inferenza Fuzzy (FIS) per ottimizzare i parametri dei guadagni proporzionali ( $K_p$ ), integrali ( $K_i$ ) e derivativi ( $K_d$ ) in base alle regole definite. Lo schema a blocchi rappresentante l'applicazione di un controllore Fuzzy PID ad un processo generico è rappresentato in figura.



**Figura 26** Sistema ad anello chiuso con controllore Fuzzy PID

Come si può notare lo schema risulta simile a quello proposto per il convenzionale controllo PID, con l'unica differenza che i parametri relativi ai guadagni  $K_p$ ,  $K_i$  e  $K_d$  non sono costanti nel tempo ma, derivano dall'applicazione della logica Fuzzy all'errore tra il riferimento e l'uscita del sistema. In questo modo il Fuzzy PID può esercitare un controllo che si avvicini in maniera significativa all'operato della mente umana. Operando in questo modo si riesce ad avere dei parametri scalati in modo opportuno man mano che il sistema si avvicina al riferimento, evitando così elevate sovraelongazioni o tempi di salita enormi. Tutto ciò non è possibile utilizzando un controllore PID e proprio per questo in sistemi complessi e non lineari si preferisce l'utilizzo del controllo Fuzzy PID. Inoltre, come vedremo successivamente, il controllo Fuzzy PID, al contrario del classico PID, risulta essere estremamente robusto alle variazioni del sistema, riuscendo a garantire una stabilità a regime con sovraelongazioni contenute e tempi di salita minimi anche per grandi variazioni di parametri.

# Capitolo 4

## Implementazione Logica Fuzzy

Per quanto riguarda l'implementazione del controllore, esso è stato realizzato interamente su Matlab/Simulink utilizzando un toolbox dedicato all'implementazione della logica Fuzzy. Il toolbox è installabile attraverso Matlab ed è visibile nelle App dello stesso. Il primo passo da effettuare per implementare una logica Fuzzy consiste nell'aprire il toolbox "Fuzzy Logic Designer" direttamente dal menu "App" di Matlab. La finestra principale del toolbox permette di impostare le definizioni dei vari metodi come l'AND, l'OR, l'implicazione, l'aggregazione e la defuzzificazione.

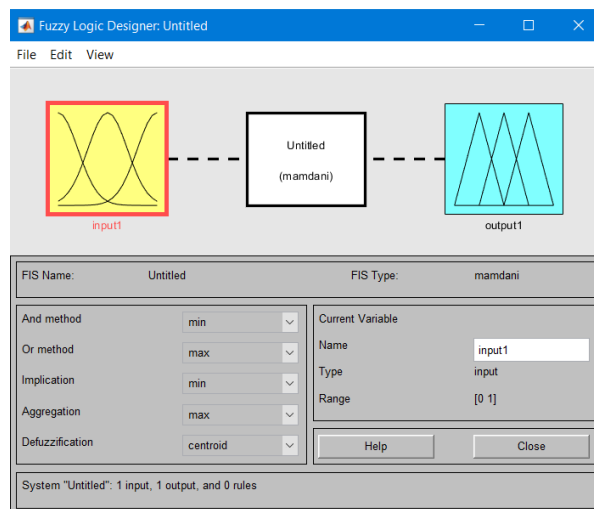
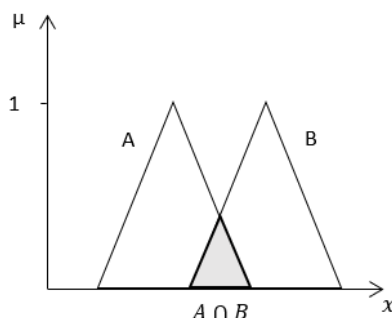


Figura 27 Finestra principale del toolbox "Fuzzy Logic Designer"

Nel mio caso ho deciso di utilizzare per il metodo AND e per l'implicazione la funzione "min" in modo tale che, come definito precedentemente, la funzione di appartenenza risultante  $f_C(x)$  prendesse l'insieme più grande che contiene gli insiemi fuzzy A e B.

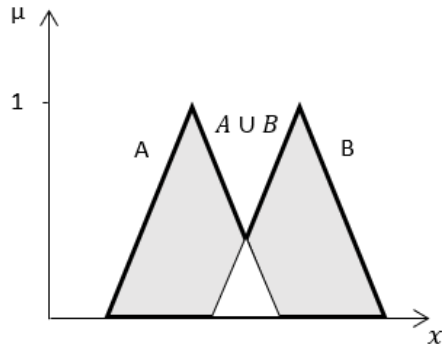


$$f_C(x) = \min[f_A(x), f_B(x)] \text{ con } x \in X$$

Figura 28 Metodo del centro di gravità



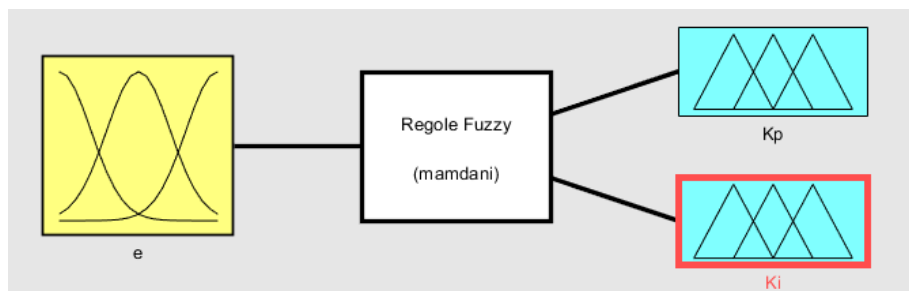
Mentre per il metodo OR e per l'aggregazione ho deciso di utilizzare la funzione “*max*” in modo che la funzione risultante  $f_C(x)$  prendesse l'insieme più piccolo che contiene gli insiemi fuzzy A che B.



$$f_C(x) = \max[f_A(x), f_B(x)] \text{ con } x \in X$$

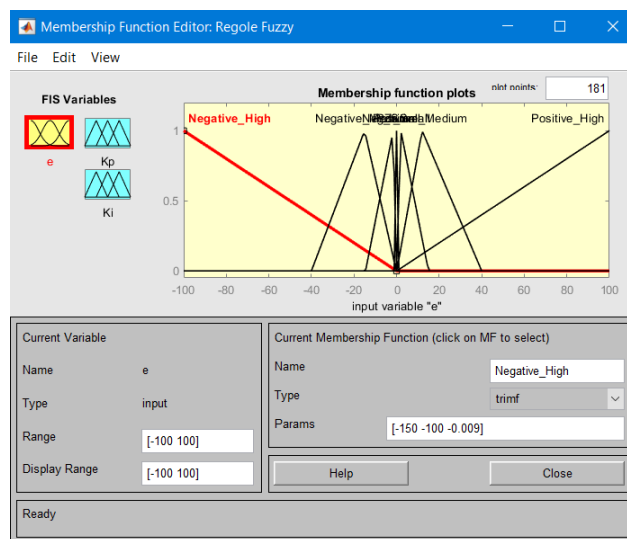
**Figura 29** Metodo del centro di gravità

Infine, per la defuzzificazione ho deciso di utilizzare il metodo del centro di gravità poiché è uno dei più semplici ed utilizzati nei controlli. Il metodo consiste nel prendere come valore l'ascissa del baricentro della figura formata dalle funzioni di appartenenza e risulta essere la miglior soluzione di compromesso poiché tiene conto anche dei contributi dati dalle regole meno influenti. Una volta impostati i vari metodi, ho realizzato gli input e gli output di cui avevo bisogno per la logica Fuzzy. Ho utilizzato un solo input che rappresenterà l'errore in percentuale e due output che corrisponderanno rispettivamente ad il guadagno proporzionale ( $K_p$ ) ed al guadagno integrale ( $K_i$ ) del controllore.



**Figura 30** Rappresentazione Input e output della logica Fuzzy

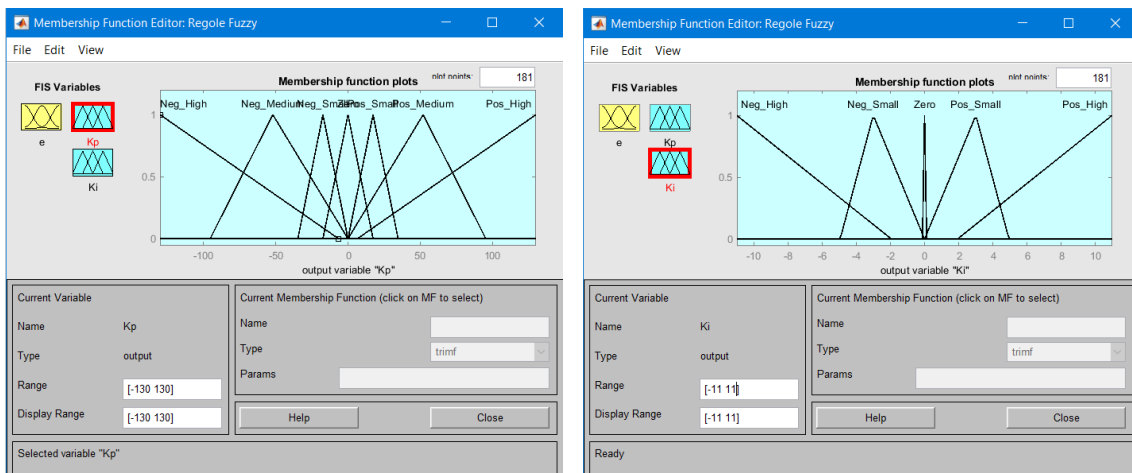
Per aggiungere un nuovo output o un nuovo input la procedura da seguire è la seguente. Per prima cosa si seleziona “Edit” e nel menu a tendina l’opzione “Add variable...”, successivamente “input” o “output” a seconda della variabile che si vuole realizzare. Una volta inserite le variabili si possono andare a settare i range nei quali bisogna operare e con essi anche i parametri di ogni funzione di appartenenza. Per modificare le variabili di input o di output bisogna effettuare un doppio click sulla variabile in modo tale da aprire una finestra di questo tipo:



**Figura 31** Finestra di editor dell’input

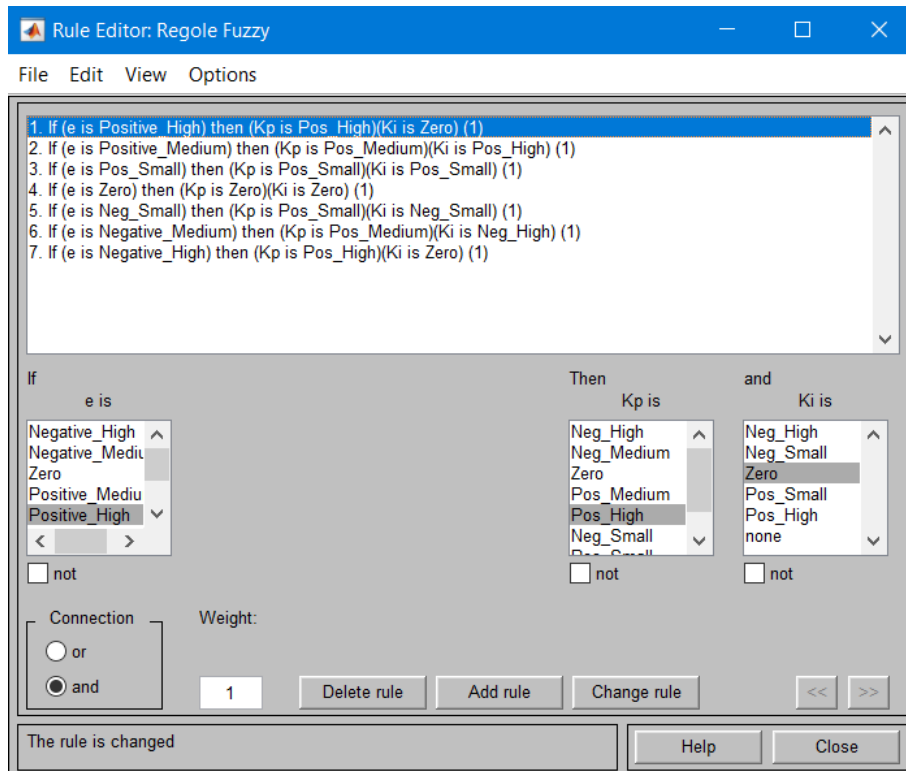
Dalla finestra di editor delle variabili è possibile aggiungere o rimuovere regioni, modificandone nome, range e parametri. Prendiamo come esempio la variabile di input che nel mio caso risulta essere l’errore in forma percentuale. Ho deciso di utilizzare un range che vada da [-100 100] poiché essendo l’errore calcolato in percentuale sul fondo scala del sistema, non potrà mai eccedere questo range. Per ottenere le regioni ho testato in simulazione il controllo modificando range e parametri, visualizzabili in “Params”, a seconda dell’output ottenuto. Nel mio caso per rappresentare l’errore percentuale ho ottenuto 7 regioni: “Negative High”, “Negative medium”, “Negative Small”, “Zero”, “Positive Small”, “Positive Medium” e “Positive High”. Più specificatamente, se ottengo un errore del 100% avrò un grado di appartenenza 1 alla regione “Positive High”, se invece ho un errore del 20% avrò

due gradi di appartenenza differenti relativi alle regioni “Positive Small” e “Positive Medium”. Per quanto riguarda gli Output ho deciso di implementare i coefficienti di proporzionale e integrale di un controllore PI. Anche i coefficienti sono stati realizzati in base a prove effettuate su simulazioni ottenendo i seguenti range:  $[-130 \ 130]$  per il guadagno proporzionale e  $[-10 \ 10]$  per il guadagno integrale. I range sono notevolmente diversi perché, come sappiamo dalla teoria dei controllori PI, è necessario un maggiore apporto del proporzionale che corregge direttamente l'errore piuttosto dell'integrale che esegue la propria azione nel tempo e quindi porta il processo esattamente al punto di riferimento richiesto. Un'altra differenza è che nel proporzionale ho realizzato 7 regioni come nel caso dell'errore, mentre per l'integrale ho avuto bisogno di solo 5 regioni eliminando quindi la zona intermedia rappresentata da “Neg Medium” e “Pos Medium”.



**Figura 32** Finestre di editor degli output

Una volta realizzate le varie regioni e settato i range sono andato ad implementare le regole su cui si baserà la logica Fuzzy. Per aprire la finestra è necessario selezionare “Edit” e successivamente la voce “Rules..”.



**Figura 33** Finestra di editor delle regole Fuzzy

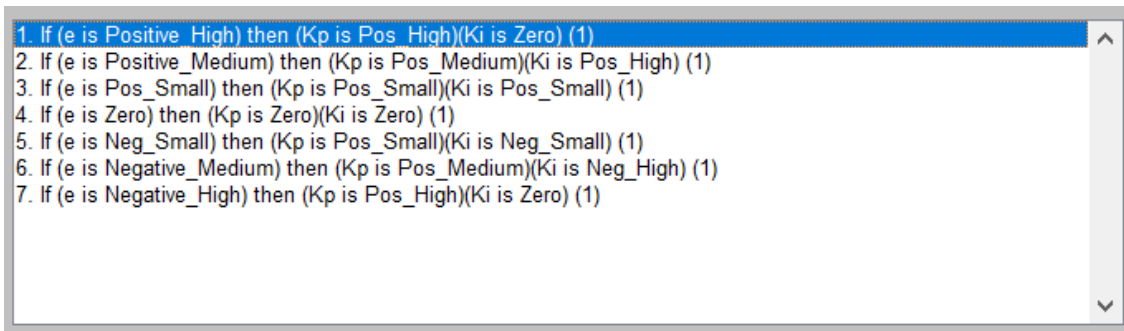
Nella finestra troveremo in basso a sinistra le variabili di input con le varie regioni precedentemente realizzate, mentre a destra le variabili di output sempre con le relative regioni. Per inserire una nuova regola è necessario selezionare quale regione di input e quale di output si intende prendere in considerazione, selezionare in basso a sinistra in base alle proprie preferenze il metodo AND o OR e infine, selezionare “Add Rule” per far comparire la regola nello spazio iniziale sotto forma di:

**IF** “variabile input” **IS** “regione di appartenenza” **THEN** “variabile output” **IS** “regione di appartenenza”

Le regole possono essere interpretate quindi nel seguente modo: se la variabile di input si trova all’interno del range della regione di appartenenza selezionata, allora la variabile di output apparterrà alla seguente regione di appartenenza e quindi avrà un

valore che viene calcolato secondo la defuzzificazione, che nel mio caso è quella del centro di gravità.

Più nel dettaglio le regole che ho deciso di implementate sono le seguenti:



```
1. If (e is Positive_High) then (Kp is Pos_High)(Ki is Zero) (1)
2. If (e is Positive_Medium) then (Kp is Pos_Medium)(Ki is Pos_High) (1)
3. If (e is Pos_Small) then (Kp is Pos_Small)(Ki is Pos_Small) (1)
4. If (e is Zero) then (Kp is Zero)(Ki is Zero) (1)
5. If (e is Neg_Small) then (Kp is Pos_Small)(Ki is Neg_Small) (1)
6. If (e is Negative_Medium) then (Kp is Pos_Medium)(Ki is Neg_High) (1)
7. If (e is Negative_High) then (Kp is Pos_High)(Ki is Zero) (1)
```

**Figura 34** Regole Fuzzy implementate

La prima rappresenta il caso in cui l'errore è molto positivo e quindi ha un valore molto diverso da 0 ("Positive High"). In questo caso mi trovo molto lontano dal riferimento e quindi avrò bisogno di un alto proporzionale ("Pos High") e un integrale prossimo allo zero ("Zero"). Successivamente se mi sto avvicinando al riferimento, e quindi l'errore è contenuto nella regione "Positive Medium", avrò una diminuzione del proporzionale ("Pos Medium") e un aumento dell'integrale ("Pos High"). Nel caso "Pos Small" dell'errore avrò che il proporzionale diminuisce ("Pos Small") e con esso anche l'integrale (Pos Small). A riferimento invece avrò che l'errore è nullo ("Zero") e quindi non avrò bisogno né del proporzionale ("Zero") né dell'integrale ("Zero"). Gli stessi casi si ripetono nel caso in cui l'errore sia nella parte negativa.

## 4.1 – Implementazione controllore Fuzzy PI in Simulink

Una volta realizzate le regole, l'implementazione della logica Fuzzy è terminata e si può procedere con la realizzazione del modello del controllore in Simulink.

Per prima cosa inserito nel progetto Simulink il blocchetto denominato “Fuzzy Logic Controller” che mi permette di inserire in “FIS name” il file implementato precedentemente attraverso il toolbox, denominato “Fuzzy\_Logic.fis”.

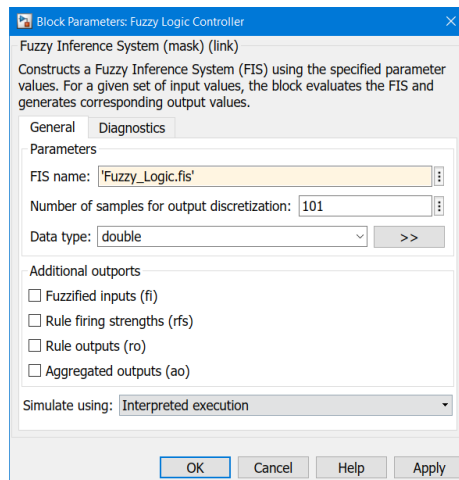


Figura 35 Finestra Fuzzy Logic Controller in Simulink

Il controllo finale implementato è composto quindi dall'acquisizione dell'errore tramite una semplice sottrazione tra il riferimento e la pressione misurata. L'errore viene poi rappresentato in percentuale all'interno della funzione  $f(u)$  per essere messo in relazione con la valvola. Infatti, in output devo ottenere un voltaggio da impartire alla valvola Moog in modo tale da poter aumentare la pressione e ridurre l'errore.

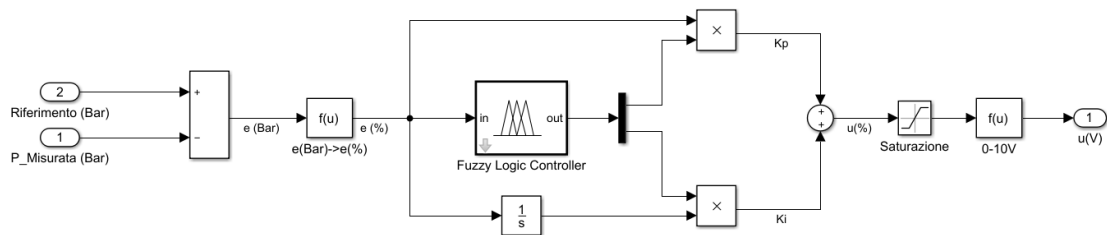
$$ERRORE_{0/0} = \frac{ERRORE_{BAR} * 100}{1280}$$

Il valore 1280 rappresenta il fondo scala del sistema. Infatti, esso in catena aperta non riesce a raggiungere una pressione massima di 1280 Bar.

L'errore percentuale viene inserito nel blocco "Fuzzy Logic Control" relativo alla logica Fuzzy dal quale, otterremo in output i valori dei guadagni proporzionale ( $K_p$ ) e integrale ( $K_i$ ) del controllore. Ottenuti i guadagni, ho implementato un normale controllo PI moltiplicando il guadagno proporzionale per l'errore e quello integrale per l'integrale dell'errore. Successivamente ho saturato l'uscita del controllore per evitare problemi di wind up e infine proporzionata rispetto alla tensione  $\pm 10$  Volt secondo la seguente formula:

$$u_c = \frac{u_c * 10}{100}$$

La formula è espressa nel seguente modo perché è proporzionata all'errore iniziale espresso in percentuale. Il sistema completo viene rappresentato nella figura seguente e sarà inserito al posto del controllore PI descritto nel capitolo 1.



**Figura 36** Rappresentazione completa del controllo Fuzzy PI

# Capitolo 5

## Simulazioni controllore PI e Fuzzy PI

Prima di analizzare il controllo ottenuto attraverso la logica Fuzzy, introduciamo il precedente controllo PI realizzato sul modello in modo tale da poter poi eseguire un confronto tra i due. La simulazione che ho effettuato presenta un riferimento di 300 Bar ed ha una durata di 20 secondi con un tempo di campionamento fisso posto a 1ms.

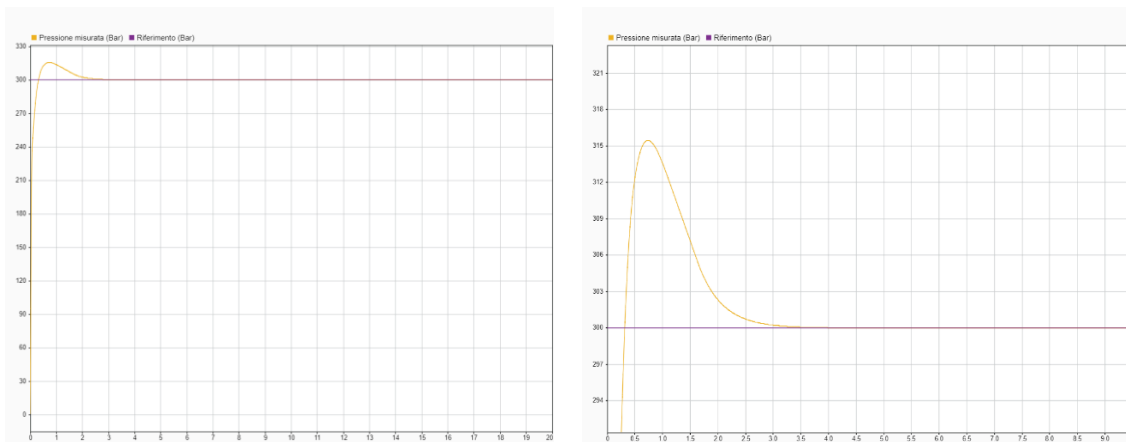


Figura 37 Grafici controllo PI con riferimento a 300 bar

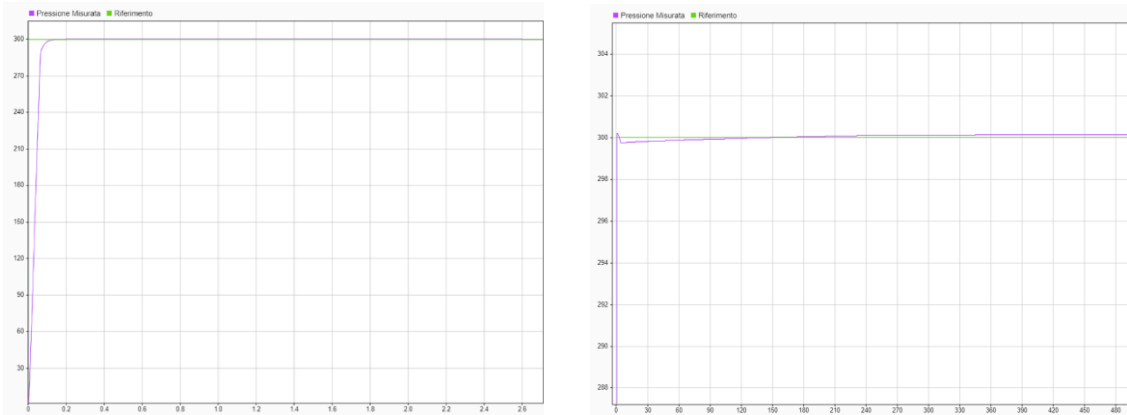
Dalle immagini si può notare come il controllo riesca a far convergere la pressione al valore di regime ma presentando sovralongazioni molto elevate. Eseguendo un'analisi attraverso la funzione “stepinfo” predefinita in Matlab, che mi permette di ottenere per un qualsiasi sistema il tempo di salita, il tempo di assestamento e la sovralongazione, otteniamo i valori rappresentati nella tabella.

Tempo di salita	0.1486 s
Tempo di assestamento	0.1601 s
Sovralongazione	15.4570 Bar



Il grafico evidenzia una sovraelongazione di 15 bar che su un fondo scala di 1280 bar corrisponde ad un errore dell'1,1718%.

Successivamente sono andato a studiare il controllo Fuzzy PI implementato, eseguendo una simulazione esattamente identica a quella effettuata per il PI e cioè, con una durata di 20 secondi campionando il tempo ogni millisecondo e con un riferimento a 300 bar.



**Figura 38** Grafici controllo Fuzzy PI con riferimento a 300 bar

Anche in questo caso ho analizzato il sistema utilizzando la funzione “stepinfo” ottenendo i seguenti valori:

Tempo di salita	0.0501 s
Tempo di assestamento	0.0615 s
Sovraelongazione	0.1961 Bar

Dalle due simulazioni posso concludere che il controllo Fuzzy PI risulta molto più efficiente ed efficace del classico controllo PID poiché permette di avere un'ottima risposta al transitorio mantenendo sovraelongazioni nell'arco di 0,2 bar che corrispondono ad un errore sul fondo scala dello 0,015625%. Riesce a raggiungere il riferimento con tempi nettamente inferiori riducendo il tempo di assestamento del 61,58%. Il riferimento in questo caso viene raggiunto molto lentamente poiché il

controllore tende ad oscillare di un'ampiezza di  $\pm 0,2$  bar. I valori risultano comunque accettabili in ottica di uno sviluppo reale perché coerenti con la risoluzione del sensore di  $\pm 2,3$  bar.

## 5.1 - Verifica robustezza controllore

Dopo aver ottenuto degli ottimi risultati sono andato a verificare la robustezza del controllore variando i parametri standard del moltiplicatore di pressione del 5, 20, 50 e 70 per cento sia positivamente che negativamente. Per parametri standard intendo tutti quei valori costanti presenti all'interno del modello come, ad esempio, i volumi e quindi le aree delle camere di alta e bassa pressione o i valori dei bulk modulus (moduli di compressibilità) dei fluidi all'interno. In questo caso oltre ad analizzare la simulazione con riferimento a 300 Bar ho ampliato l'analisi considerando le variazioni anche per riferimenti a 500 e 700 Bar. L'analisi è stata effettuata con lo script inserito all'interno dell'appendice A che permette di eseguire per ogni riferimento, una simulazione per ogni percentuale di cambiamento, e restituisce in output l'errore massimo raggiunto, la sovraelongazione, il tempo di salita e il tempo di assestamento, utilizzando la funzione introdotta precedentemente "stepinfo".

{'Riferimento'}	{'errore' }	{'Tempo di salita'}	{'Tempo di assest...'}	{'Sovraelongazione'}	{'p' }
{[ 300]}	{[ 0.0654]}	{[ 0.0501]}	{[ 11.2426]}	{[ 0.1961]}	{[ 1]}
{[ 300]}	{[ 0.0559]}	{[ 0.0601]}	{[ 0.1813]}	{[ 0.1676]}	{[1.2000]}
{[ 300]}	{[ 0.0459]}	{[ 0.0753]}	{[ 0.2143]}	{[ 0.1378]}	{[1.5000]}
{[ 300]}	{[ 0.0411]}	{[ 0.0854]}	{[ 0.2345]}	{[ 0.1233]}	{[1.7000]}
{[ 300]}	{[ 0.0792]}	{[ 0.0400]}	{[ 17.7687]}	{[ 0.2376]}	{[0.8000]}
{[ 300]}	{[ 3.7195]}	{[ 0.0248]}	{[ 20.8654]}	{[ 11.1584]}	{[0.5000]}
{[ 300]}	{[26.2421]}	{[ 0.0101]}	{[ 21.5479]}	{[ 78.7262]}	{[0.2000]}
{[ 500]}	{[ 0.0132]}	{[ 0.0836]}	{[ 0.1497]}	{[ 0.0662]}	{[ 1]}
{[ 500]}	{[ 0.0112]}	{[ 0.1006]}	{[ 0.1754]}	{[ 0.0561]}	{[1.2000]}
{[ 500]}	{[ 0.0091]}	{[ 0.1261]}	{[ 0.2110]}	{[ 0.0456]}	{[1.5000]}
{[ 500]}	{[ 0.0081]}	{[ 0.1430]}	{[ 0.2338]}	{[ 0.0405]}	{[1.7000]}
{[ 500]}	{[ 0.0162]}	{[ 0.0667]}	{[ 0.1138]}	{[ 0.0811]}	{[0.8000]}
{[ 500]}	{[ 2.3939]}	{[ 0.0412]}	{[ 0.0928]}	{[ 11.9694]}	{[0.5000]}
{[ 500]}	{[15.8948]}	{[ 0.0159]}	{[ 0.0823]}	{[ 79.4738]}	{[0.2000]}
{[ 700]}	{[ 0.0041]}	{[ 0.1176]}	{[ 0.1726]}	{[ 0.0290]}	{[ 1]}
{[ 700]}	{[ 0.0035]}	{[ 0.1416]}	{[ 0.2038]}	{[ 0.0244]}	{[1.2000]}
{[ 700]}	{[ 0.0028]}	{[ 0.1776]}	{[ 0.2486]}	{[ 0.0197]}	{[1.5000]}
{[ 700]}	{[ 0.0025]}	{[ 0.2016]}	{[ 0.2776]}	{[ 0.0175]}	{[1.7000]}
{[ 700]}	{[ 0.0051]}	{[ 0.0936]}	{[ 0.1342]}	{[ 0.0356]}	{[0.8000]}
{[ 700]}	{[ 1.7074]}	{[ 0.0575]}	{[ 0.1015]}	{[ 11.9517]}	{[0.5000]}
{[ 700]}	{[13.4999]}	{[ 0.0216]}	{[ 0.0973]}	{[ 94.4992]}	{[0.2000]}

Figura 39 Risultati simulazioni controllore Fuzzy PI

Dai risultati verifico come il controllore risulti robusto fino a variazioni positive del 70% e variazioni negative del -20%, ottenendo errori sempre al di sotto di 0,25 bar che risultano essere lo 0,01953% di errore rispetto alla pressione massima che può raggiungere il moltiplicatore di 1280 Bar. Successivamente è stata ripetuta la stessa simulazione per il controllore PI ottenendo i seguenti risultati:

{'Riferimento'}	{'errore'}	{'Tempo di salita'}	{'Tempo di assest...'	{'Sovraelongazione'}	{'p' }
{[ 300]}	{[3.9992]}	{[ 0.1615]}	{[ 3.9471]}	{[ 11.9976]}	{[ 1]}
{[ 300]}	{[4.4823]}	{[ 0.1880]}	{[ 3.8702]}	{[ 13.4469]}	{[1.2000]}
{[ 300]}	{[5.1446]}	{[ 0.2244]}	{[ 3.8031]}	{[ 15.4339]}	{[1.5000]}
{[ 300]}	{[5.5515]}	{[ 0.2466]}	{[ 3.8320]}	{[ 16.6544]}	{[1.7000]}
{[ 300]}	{[3.4748]}	{[ 0.1330]}	{[ 3.8600]}	{[ 10.4243]}	{[0.8000]}
{[ 300]}	{[2.5844]}	{[ 0.0861]}	{[ 3.0880]}	{[ 7.7532]}	{[0.5000]}
{[ 300]}	{[1.4964]}	{[ 0.0336]}	{[ 1.5146]}	{[ 4.4893]}	{[0.2000]}
{[ 500]}	{[4.1196]}	{[ 0.1212]}	{[ 4.8793]}	{[ 20.5980]}	{[ 1]}
{[ 500]}	{[4.6709]}	{[ 0.1406]}	{[ 4.8457]}	{[ 23.3544]}	{[1.2000]}
{[ 500]}	{[5.4420]}	{[ 0.1683]}	{[ 4.5729]}	{[ 27.2102]}	{[1.5000]}
{[ 500]}	{[5.9260]}	{[ 0.1861]}	{[ 4.3648]}	{[ 29.6299]}	{[1.7000]}
{[ 500]}	{[3.5308]}	{[ 0.1009]}	{[ 4.7788]}	{[ 17.6540]}	{[0.8000]}
{[ 500]}	{[2.5529]}	{[ 0.0669]}	{[ 4.2977]}	{[ 12.7647]}	{[0.5000]}
{[ 500]}	{[1.3914]}	{[ 0.0260]}	{[ 2.7555]}	{[ 6.9568]}	{[0.2000]}
{[ 700]}	{[4.9086]}	{[ 0.1297]}	{[ 5.3002]}	{[ 34.3602]}	{[ 1]}
{[ 700]}	{[5.6382]}	{[ 0.1537]}	{[ 5.2739]}	{[ 39.4674]}	{[1.2000]}
{[ 700]}	{[6.6865]}	{[ 0.1889]}	{[ 5.1218]}	{[ 46.8056]}	{[1.5000]}
{[ 700]}	{[7.3633]}	{[ 0.2121]}	{[ 5.0007]}	{[ 51.5431]}	{[1.7000]}
{[ 700]}	{[4.1452]}	{[ 0.1051]}	{[ 5.2102]}	{[ 29.0166]}	{[0.8000]}
{[ 700]}	{[2.9102]}	{[ 0.0666]}	{[ 4.7892]}	{[ 20.3712]}	{[0.5000]}
{[ 700]}	{[1.4918]}	{[ 0.0250]}	{[ 3.5330]}	{[ 10.4429]}	{[0.2000]}

**Figura 40** Risultati simulazioni controllore PI

Dalla tabella notiamo come il controllore PI non risulti robusto a variazioni significative dei parametri. Il suo comportamento era prevedibile perché sappiamo che esso sia utilizzato per la sua semplicità e per la sua facile realizzazione ma che non risulti adatto per controlli su sistemi non lineari o complessi. Possiamo quindi concludere che utilizzando un controllo PI è necessario tarare i parametri proporzionale, integrativo e derivativo ad ogni variazione del riferimento non risultando robusto alle variazioni e presentando ingenti sovraelongazioni.

# Capitolo 6

## Descrizione componenti Beckhoff

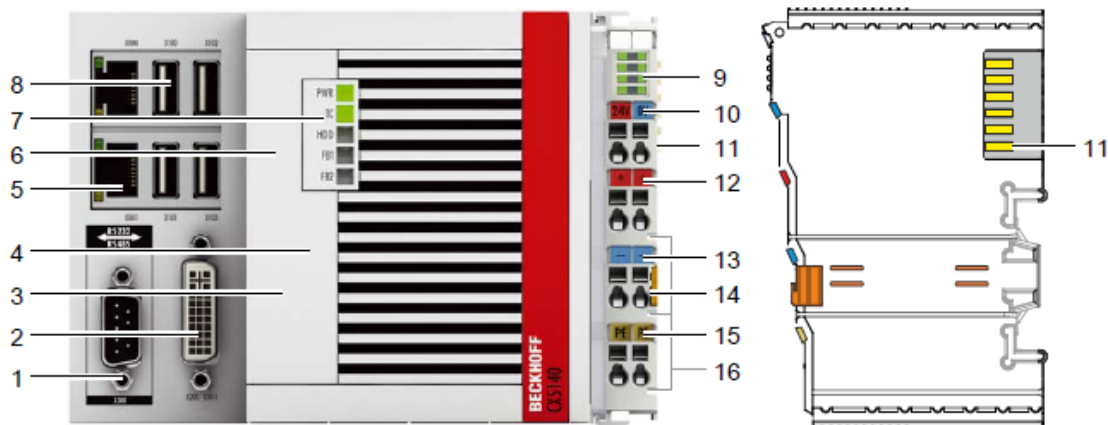
La multinazionale Beckhoff realizza sistemi aperti per automazione con tecnologia di controllo basata su PC. La gamma di prodotti copre i principali settori dell'industria come PC industriali, componenti per bus di campo e I/O, Motion Control e software di automazione. Uno tra questi dispositivi, in particolare il modello Beckhoff CX5041, è stato utilizzato nella realizzazione del progetto.

### 6.1 - CX5140

Con la serie CX dei computer Embedded, Beckhoff ha combinato la tecnologia del PC e I/O modulari per formare un'unità di alto livello. La CX5140 comprende al suo interno

- Uno slot per la scheda CFast
- Uno slot per la scheda MicroSD
- Due porte indipendenti Ethernet
- Quattro porte USB 2.0
- Una porta DVI per la visualizzazione a schermo

All'interno della CX troviamo anche un UPS che permette, in assenza di corrente, di memorizzare fino a 1MB di dati persistenti sulla scheda CFast o sulla MicroSD. Possiede inoltre un processore Intel Atom E3845 single-core da 1.91GHz e 4GB di Ram DDR3 e uno dei seguenti sistemi operativi: Microsoft Windows 10 IoT Enterprise LTSC, Microsoft Windows Embedded Standard 7 P o Microsoft Windows Embedded Compact 7.



**Figura 41** PC Embedded CX5140

Sul lato destro della CX troviamo i terminali di alimentazioni (11); in particolare, troviamo un terminale per acquisire l'alimentazione di 24V e il rispettivo ground. Nella parte sinistra invece troviamo i due terminali EtherCat (5) che permettono la comunicazione del PLC con i vari dispositivi, come ad esempio un computer. Insieme ai terminali troviamo le porte USB (8), i led informativi (7) e infine le due uscite video, la DVI-I per la connessione di monitor analogici o digitali (2), e la DVI-D per la connessione a display digitali (1).

La programmazione può essere effettuata in remoto tramite Ethercat. Tale possibilità, adottata nel caso in cui il sistema sia dotato di "Windows CE.NET", permette di programmare il tutto tramite un computer fisso o portatile collegato alla CX tramite Ethernet. Per l'utilizzo dei componenti Beckhoff, si utilizza il software proprietario TwinCAT che è in grado di gestire con semplicità tutte le funzionalità della CX e degli altri componenti Beckhoff, ciò permette di collegare le uscite fisiche direttamente a variabili logiche facilmente gestibili d'ambienti di sviluppo software, tra i quali si annovera il software Visual Studio © progettato da Microsoft. Il CX è caratterizzato da una struttura modulare: infatti è costituito da vari moduli, ognuno con caratteristiche ben precise, con la possibilità di variare la configurazione dell'embedded PC a seconda delle esigenze di impiego. Al modulo base possono essere aggiunti diversi altri tipi di moduli e terminali. Nel caso presentato come esempio, i moduli aggiunti sono:

- Il terminale di ingresso **EL3702** che gestisce segnali nell'intervallo da -10V a +10V. La tensione viene digitalizzata a una risoluzione di 16 bit e, viene trasmessa, isolata elettricamente, al controller. La frequenza di campionamento massimo è di 100ksps (campionamento di 10  $\mu$ s) con una larghezza di banda del segnale di ingresso consigliata di 0-30 kHz. Il terminale di ingresso EL3702 è disponibile anche per segnali nell'intervallo da -150mV a +150mV. Inoltre, il lampeggio dei led indica lo scambio dei dati con il sensore o il terminale inserito.

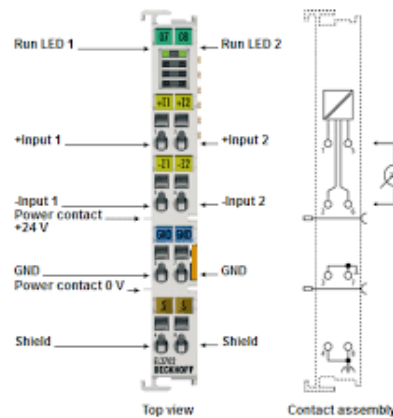


Figura 42 Modulo EL3702

- Il terminale di uscita **EL4134** genera segnali nell'intervallo da 0V a 10V o da -10V a 10V. La tensione viene digitalizzata a una risoluzione di 16 bit, e isolata elettricamente. Il ground dei terminali risulta in comune e il lampeggio dei led indica lo scambio di dati con il terminale inserito.

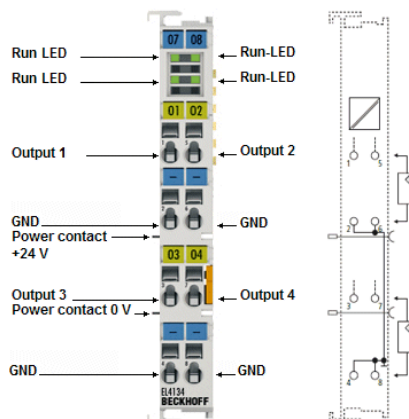
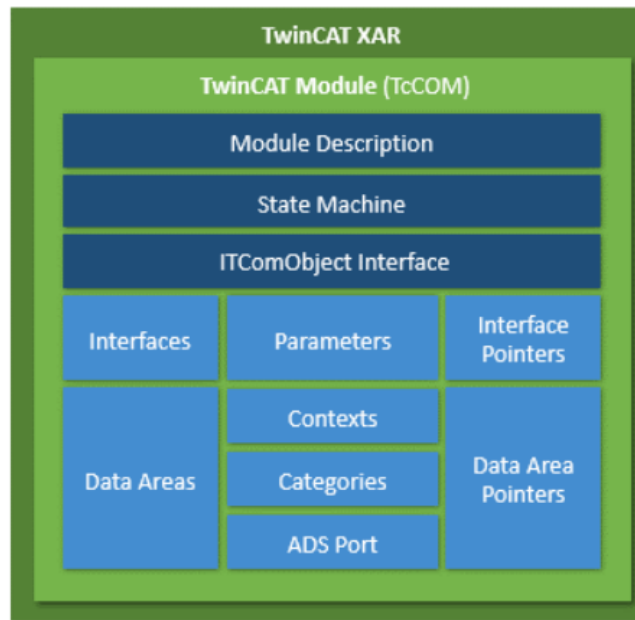


Figura 43 Modulo EL4134

## 6.2 – Software TwinCAT

Per l'utilizzo dei componenti Beckhoff si utilizza il software proprietario TwinCAT che permette di convertire ogni tipo di pc industriale o PC embedded Beckhoff, in un controllore "Real-Time". L'architettura Twincat consiste in un sistema "run-time" che esegue programmi di controllo in tempo reali interfacciandosi con programmi Microsoft per la visualizzazione dei dati e per l'esecuzione dei comandi. Un valore aggiunto offerto dal software Twincat è di permettere al programmatore di scrivere istruzioni sia in modalità "offline", se non si dispone della connessione Ethernet con il dispositivo, sia "online". Il software può essere gestito tramite un ambiente di sviluppo come ad esempio, Visual Studio progettato dalla Microsoft che, ci permette di gestire le funzionalità della CX e degli altri componenti Beckhoff. Un progetto standard realizzabile attraverso Visual Studio ci mette a disposizione le seguenti opzioni: System, Motion, PLC, Safety, C++ e I/O. Nel nostro caso di studio andremo a descrivere più dettagliatamente l'opzione System. Il menù system contiene tutte le informazioni relative al programma e al progetto, quali le licenze software, i vari "Task" (tempi ciclo) realizzati per eseguire il progetto in Real-Time, gli indirizzi relativi ai componenti Beckhoff connessi e infine i TcCOM Objects. Proprio questi ultimi sono parte fondamentale del lavoro descritto successivamente. Il componente TcCOM Object descrive il modo in cui vari componenti software sviluppati e compilati in modo indipendente possono cooperare tra loro. TcCOM si basa su COM (Component Object Model del mondo Microsoft Windows) anche se per poter facilitare l'interazione dei moduli, utilizza delle definizioni aggiuntive come ad esempio il modulo macchina a stati. Se uno dei moduli TcCOM viene istanziato nel runtime TwinCAT, si registra con un'istanza di sistema centrale, l'ObjectServer. Ciò lo rende raggiungibile e parametrizzabile per altri moduli e anche per strumenti generali. I moduli possono essere compilati in modo indipendente e quindi possono anche essere sviluppati, testati e aggiornati, in modo indipendente. I moduli possono essere semplici, ad esempio composti da una funzione base come il filtro passa-basso, oppure possono essere molto complessi internamente e contenere l'intero sistema di controllo per una macchina. Esistono molte applicazioni di moduli e per questo non viene fatta distinzione tra moduli che rappresentano le funzioni base di un sistema,

come compiti in tempo reale, driver di bus o un sistema di runtime PLC. Lo schema seguente mostra un modulo TwinCAT comune con le sue proprietà principali. I blocchi blu scuro definiscono le proprietà standard, quelli blu chiaro le opzionali.



**Figura 44** Proprietà TcCOM

Ogni modulo TcCOM ha alcuni parametri di descrizione generale. Questi includono un ClassID, che fa riferimento alla classe del modulo. Ogni modulo possiede una macchina a stati che ne descrive lo stato di inizializzazione e i mezzi con cui può essere modificato dall'esterno. Quindi descrive gli stati che si verificano durante l'avvio e l'arresto del modulo. Ciò riguarda la creazione, la parametrizzazione e la produzione dei moduli insieme agli altri moduli. Inoltre, ogni modulo contiene dei parametri che possono essere letti e scritti durante l'inizializzazione o successivamente in fase di esecuzione. Le interfacce consistono in un insieme definito di metodi (funzioni), che offrono moduli attraverso i quali possono essere contattati da altri moduli. Le interfacce sono caratterizzate da un ID univoco e ogni modulo deve supportare almeno l'interfaccia ITComObject oltre alle altre interfacce richieste.



### **6.3 – Protocollo Ads**

Un computer remoto è in grado di dialogare e quindi di scambiare messaggi con i moduli PLC attraverso un'interfaccia Ads tramite router di messaggi. Il protocollo Ads (Automation Device Specification) è un sistema basato su messaggi che permette la comunicazione e quindi lo scambio di dati tra PLC e moduli o tra moduli e moduli collegati tra loro. Nel caso in cui si debba comunicare con un altro PC, il protocollo Ads utilizza un indirizzo TCP/IP per mettere in comunicazione un PC con un PLC in rete. Per identificare i router del PC e del PLC messi in comunicazione tramite scambio di messaggi si utilizza l'AdsAmsNetID che non è altro che un numero creato all'installazione di TwinCAT, che normalmente è costruito a partire dall'indirizzo TCP/IP del dispositivo con l'aggiunta di "1.1" come se fosse una maschera di rete. La CX presa in considerazione nel resto del progetto, ha un Internet Protocol Address di 5.51.149.132 e il suo AdsAmsNetID è 5.51.149.132.1.1, anche se quest'ultimo può comunque essere scelto liberamente purché rispetti le regole della maschera di sottorete. Il protocollo ADS è stato introdotto perché verrà utilizzato all'interno dell'interfaccia grafica del progetto. Infatti, tramite l'utilizzo di questo protocollo è stato possibile interfacciare il progetto PLC in run all'interno della CX con l'interfaccia grafica realizzata tramite Visual Studio utilizzando C# come linguaggio di programmazione.

### **6.4 - Guida integrazione Simulink e TwinCAT**


La seguente guida permette di esportare interamente un progetto realizzato in Simulink in TwinCAT e renderlo eseguibile su qualunque dispositivo Beckhoff. L'integrazione tra i due ambienti è stata fondamentale per il mio lavoro poiché in TwinCAT non è presente nessun blocco relativo alla logica Fuzzy e quindi in questo modo ho potuto verificare il corretto funzionamento del controllore sul moltiplicatore di pressione presente al Kite Lab in Loccioni.

## Pre-requisiti

- Matlab/Simulink
- Simulink Coder (Matlab Coder)
- Visual Studio
- Visual Studio C++ Compiler
- WinDDK7, WinDDK8.1, WinDDk10
- TwinCAT 3.1 + TE1400

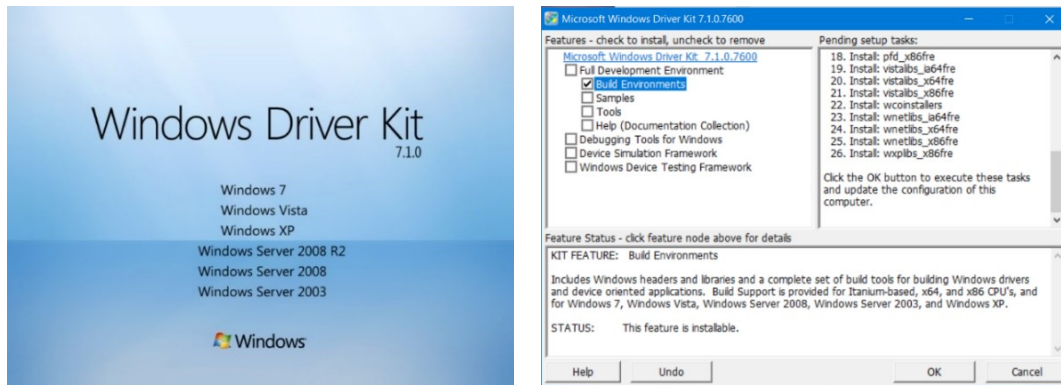
## Installazione Microsoft Windows Driver Kit

Una volta verificato di possedere tutti i prerequisiti necessari, il primo passo è quello di installare “Microsoft Windows Driver Kit” (WDK), che verrà utilizzato per abilitare lo sviluppo dei driver del kernel di Windows. Infatti, i driver C++ di Twincat sono basati sui WDK.

**N.B.** è necessario installare i windows driver kit sia per quanto riguarda Windows 7 (WDK7) sia riguardo alla versione di windows installata all'interno del proprio computer. Per scoprire quale versione di Windows è in esecuzione sul dispositivo, è necessario premere i tasti  + **R**, aprire “winver” e leggere la versione corrente.

### • Installazione Microsoft Windows Driver Kit 7

1. Scaricare WDK7 dal seguente indirizzo: <https://www.microsoft.com/en-us/download/details.aspx?id=11800>
2. Eseguito il download, o si monta su CD il file ISO scaricato oppure si utilizza un software per virtualizzare la lettura del CD.
3. Avviare “KitSetup.exe” dal CD come amministratore



**Figura 45** Scherma Windows Driver Kit 7

4. Selezionare l'opzione "Build Environments" e successivamente **OK** per continuare.
5. Dopo aver accettato la licenza Microsoft EULA e selezionato la cartella di installazione attivare l'installazione con il tasto **OK**.
6. Navigare in "Start" → "Pannello di controllo" → "Sistema" e selezionare "Impostazioni di sistema avanzate"
7. Selezionare "avanzate" e cliccare su "Variabili d'ambiente"
8. Nell'area relativa alle "variabili di sistema", selezionare "nuova" inserendo in "Nome variabile" WINDDK7 e in "Valore variabile" C:\WinDDK\7600.16385.1
9. Selezionare **OK** e riavviare il computer per completare l'installazione

- **Installazione Microsoft Windows Driver Kit versione propria di windows**

L'installazione relative ai Windows Driver Kit della versione corrente di Windows all'interno del computer, ha come prerequisito, la presenza di Visual studio poiché l'installazione dei WDK deve essere effettuata dal suo menù di configurazione.

Al seguente link è possibile scaricare Visual Studio oppure unicamente i WDK, se si possiede già una versione di visual studio

<https://docs.microsoft.com/en-us/windows-hardware/drivers/other-wdk-downloads>

Per eseguire l'installazione quindi, una volta aperto il menù di installazione e configurazione di Visual Studio è necessario solamente selezionare “Sviluppo di applicazioni desktop con C++” e, successivamente, nella parte destra selezionare i WDK di cui abbiamo bisogno (nel menù di visual studio al posto della denominazione WDK troviamo SDK perché rappresentano i Software Development Kit compatibili con i Windows Driver Kit).

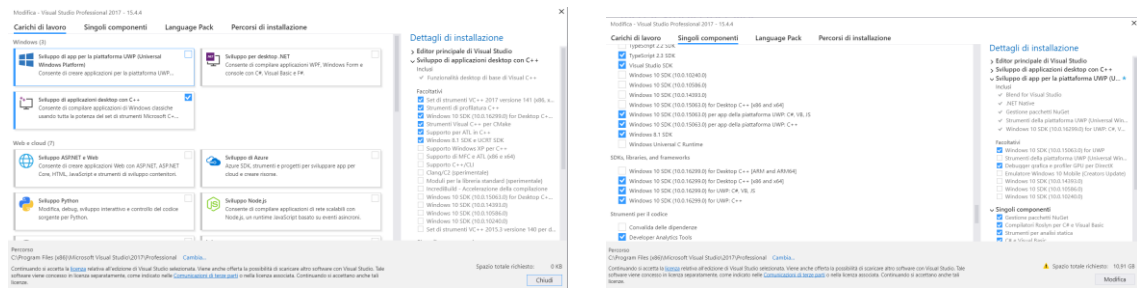


Figura 46 Schermata Visual Studio installazione Windows Driver Kit

## • Creazione Certificato

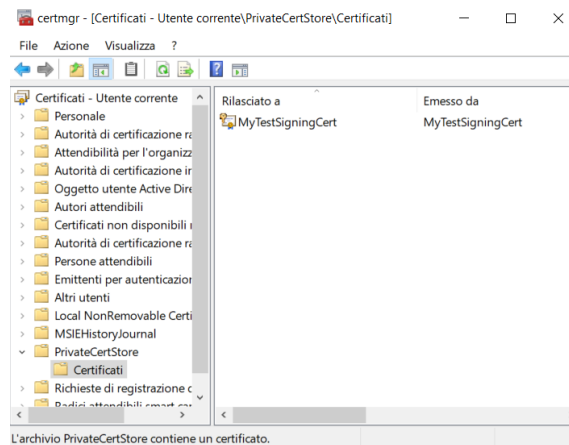
I moduli C++ di TwinCat devo essere firmati da un certificato per poter essere eseguiti. Di norma i certificati creati e quindi non verificati non possono essere eseguiti a meno che il sistema operativo non si trovi nella modalità test in modo tale da non utilizzare i certificati creati su sistemi produttivi.

Per creare un certificato è necessario aprire il prompt dei comandi in modalità amministratore ed eseguire il seguente comando:

```
makecert -r -pe -ss PrivateCertStore -n CN=MyTestSigningCert MyTestSigningCert.cer
```

Per verificare la corretta creazione del certificato bisogna seguire la seguente procedura:

1. Aprire la finestra di dialogo “esegui” utilizzando i tasti **Windows + R**
2. Inserire il nome “certmgr.msc” e selezionare OK per aprirlo
3. Nella finestra, selezionare “PrivateCertStore” → “Certificati” e visualizzeremo il certificato appena creato

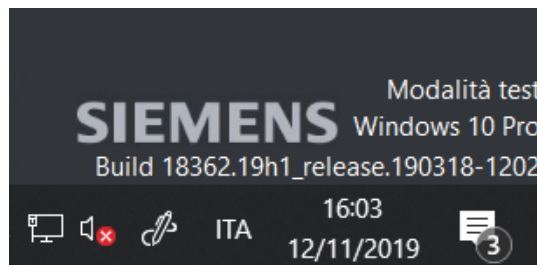


**Figura 47** Finestra verifica creazione certificato

Successivamente è necessario attivare la modalità test, sia nel proprio computer che all'interno di un qualsiasi PC Embedded utilizzato per eseguire il progetto, aprendo il prompt dei comandi in modalità amministratore e digitando:

```
bcdedit /set testsigning yes
```

Una volta eseguito è necessario riavviare il PC e, all'accensione se non ci sono stati errori comparirà la seguente scritta in basso a destra:



**Figura 48** Attivazione Modalità Test

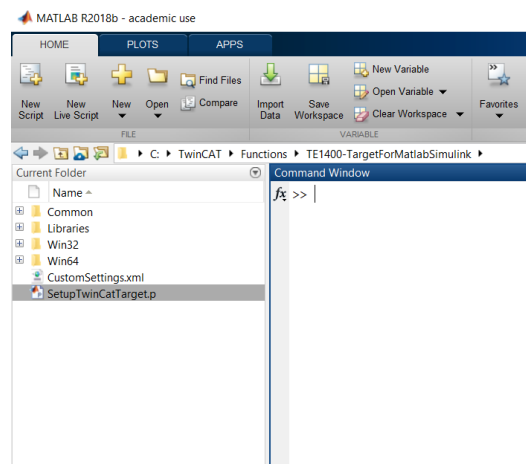
**N.B.** Potrebbe essere necessario disattivare all'interno del Bios il “SecureBoot”.

- **Installazione TE1400**

Il target Simulink di TwinCAT offre un sistema per utilizzare il Simulink Coder all'interno di TwinCAT, abilitando la generazione di moduli runtime di TwinCAT3. Il target è scaricabile dal seguente indirizzo:

<https://www.beckhoff.com/TE1400/>

Naturalmente l'installazione prevede come prerequisito TwinCAT3 installato sul proprio computer. Una volta eseguita l'installazione di TE1400 è possibile eseguire il file "SetupTwinCatTarget.p" all'interno di Matlab. Il file si trova all'indirizzo "C:\TwinCAT\Functions\TE1400-TargetForMatlabSimulink" e per eseguirlo è necessario selezionarlo con il tasto destro ed eseguire il "run" all'interno di Matlab. Durante il run verrà richiesto di selezionare il compilatore e sarà necessario selezionare "Microsoft Visual C++".



**Figura 49** Esecuzione SetupTwinCatTarget

**N.B.** Se il procedimento venisse eseguito senza aver aperto Matlab in modalità amministratore sarà necessario ripeterlo ogni qual volta si voglia eseguire l'esportazione.

Una volta eseguita questa operazione possiamo aprire il nostro progetto Simulink e selezionare "Model Configuration Parameters". Successivamente si aprirà una finestra nella quale dobbiamo selezionare "Code Generation", per poi inserire "TwinCAT.tlc" in "System target file" e inserire la spunta in "Generate code only".

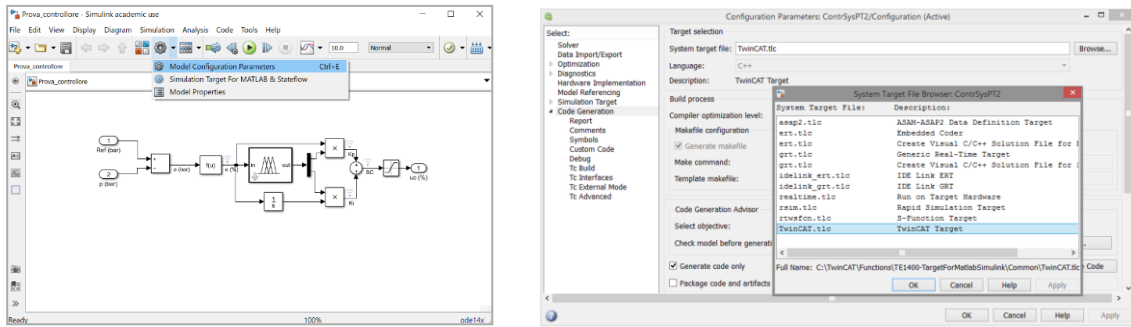


Figura 50 Apertura Model Configuration Parameters

Ora possiamo andare in “TC Build” e inserire il certificato in “Signing Certificate for x64 Windows Loader”

**N.B.** Il solver deve essere impostato su “Fixed-Step”.

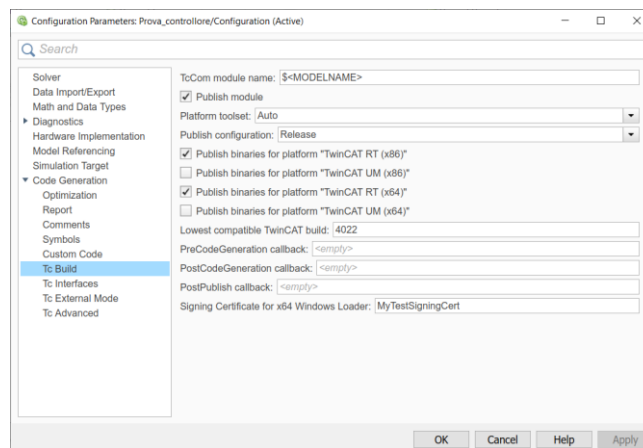


Figura 51 Finestra Tc Build

Una volta eseguite le seguenti operazioni si può eseguire il build selezionando “Build Model”.

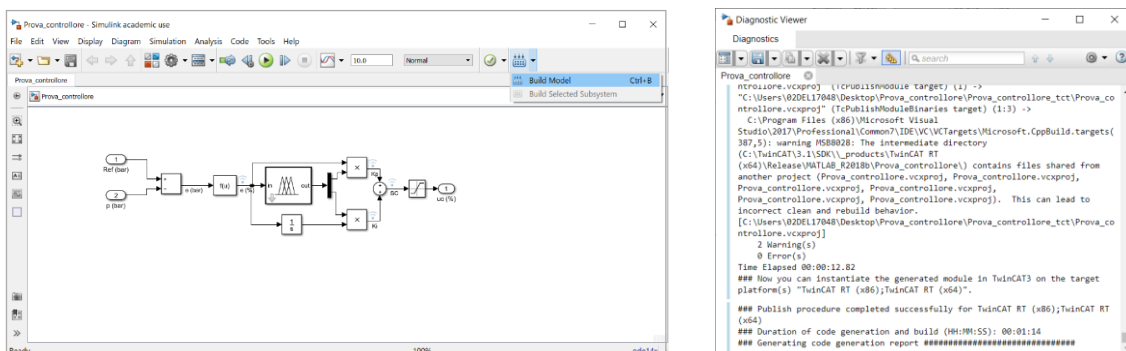
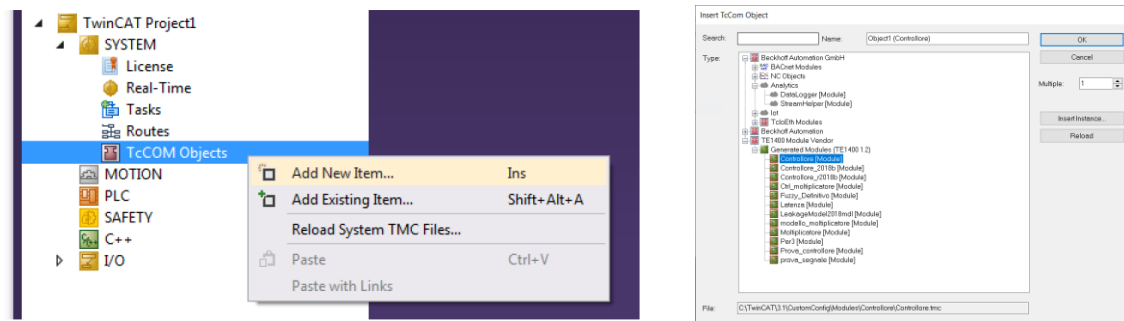


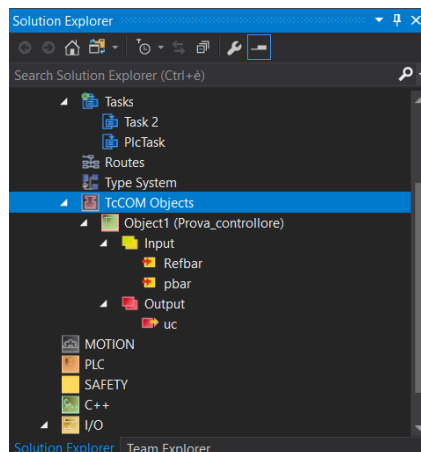
Figura 52 Esecuzione Build

Una volta che il build è terminato si può aprire l'oggetto creato all'interno di Visual Studio in un progetto TwinCAT XAE. All'interno del progetto è sufficiente selezionare con il tasto destro del mouse su "TcCOM Objects" e successivamente su "Add New Item...". Si aprirà una finestra all'interno del quale bisogna selezionare: "TE1400\_Module\_Vendor/Generated Modules(TE1400 1.2)/Nome del progetto".



**Figura 53** Apertura TcCom Objects in Twincat

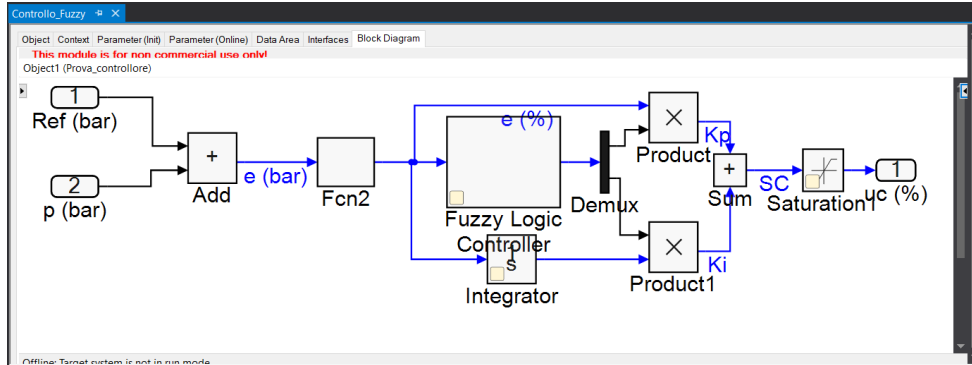
Selezionato l'oggetto da inserire, compariranno in automatico i riferimenti di input/output esattamente identici a quelli realizzati in Simulink.



**Figura 54** Input e Output dell'oggetto importato



Inoltre, selezionando su “Object1(nome del progetto)” è possibile visualizzare nella finestra “Block Diagram” lo schema a blocchi realizzato in Simulink



**Figura 55** Visualizzazione oggetto importato

Successivamente occorrà creare un nuovo Task da applicare al nostro oggetto che abbia lo stesso tempo di campionamento del progetto realizzato e compilato in Simulink. Per eseguire tale operazione, bisogna selezionare dal menu, con il tasto destro del mouse su “Tasks”: selezionare “Add New Item...” e successivamente su OK per poi inserire in “Cycle ticks” il valore del tempo di campionamento del progetto Simulink

**Figura 56** Creazione Task

Successivamente nel menu dell'oggetto importato selezionare "Context" in "Task" inserire il Task appena realizzato.

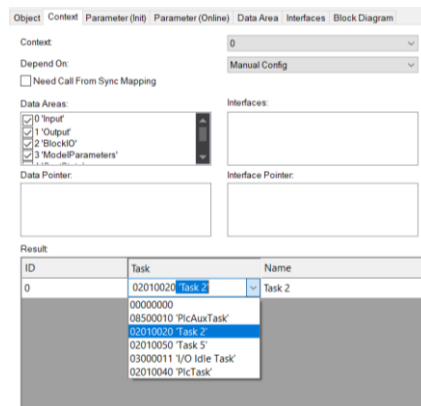


Figura 57 Inserimento Task nell'Object

Ora è possibile collegare un qualsiasi PC Embedded della Beckhoff Real Time ed eseguire il progetto come un normale progetto TwinCAT. Ricordo che il PC deve essere obbligatoriamente Real Time per poter eseguire l'oggetto importato da Simulink.

# Capitolo 7

## Implementazione hardware

Prima di descrivere il codice implementato in TwinCAT andiamo a descrivere le componenti hardware utilizzate e le varie modifiche effettuate sul progetto Simulink esportato. Per l'implementazione hardware ho utilizzato un PC embedded CX5140, collegato tramite EtherCAT, all'interno del quale Beckhoff ha combinato la tecnologia de PC con il sistema I/O modulare in modo da formare un dispositivo adatto per compiti di controllo con prestazioni molto elevate simili a quelle presenti nei PC industriali.



Figura 58 CX5140

Per quanto riguarda le connessioni con il moltiplicatore di pressione ho inserito, un modulo EL3702 per acquisire il segnale analogico di input inviato dal sensore di pressione e, un modulo EL4134 per inviare il segnale analogico di uscita alla servovalvola in modo da aumentare la pressione nel sistema. Al modulo analogico di input collegato alla CX è stato inserito l'ingresso del sensore di misura della pressione nell'input positivo del canale numero 1 (+Input1) e il rispettivo ground nel canale GND, come mostrato in figura.

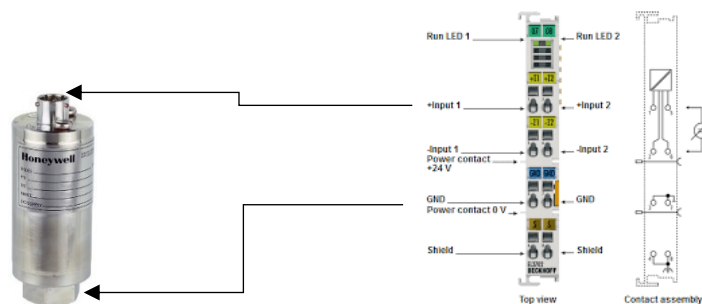


Figura 59 Connessione modulo EL3702 con sensore di pressione

Siccome i valori acquisiti risultano essere in Volt è stato necessario eseguire una scalatura per ottenere gli equivalenti valori in Bar. Il sensore presenta un fondo scala di 2250 Bar mentre l'acquisizione del modulo avviene a  $\pm 10V$  che corrispondono a valori decimali di  $\pm 32767$ . Perciò la scalatura viene effettuata attraverso la seguente formula:

$$\text{Valore di pressione in Bar} = \text{Valore acquisito in Volt} * \frac{2250}{32767}$$

Per quanto riguarda il modulo analogico di output invece è stato collegato alla servovalvola utilizzando il canale numero 1 di output (Output1) e il rispettivo ground (GND).

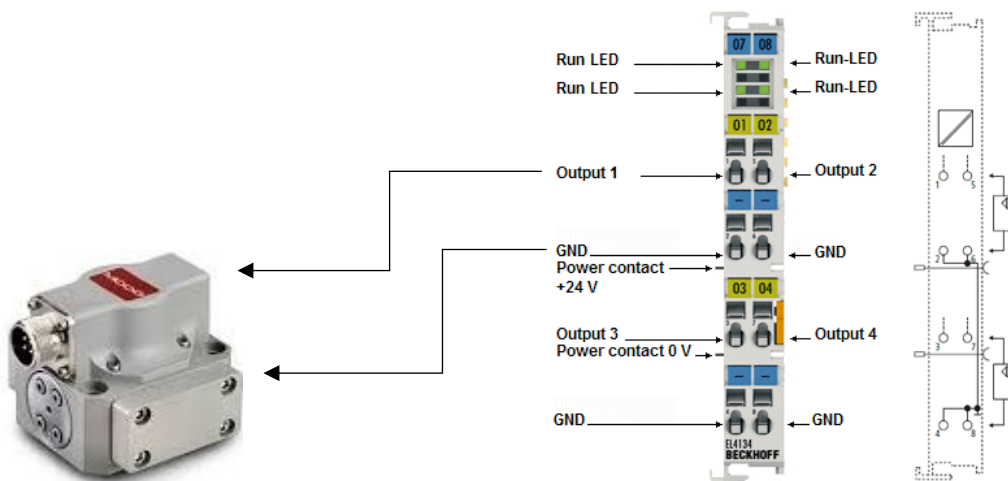
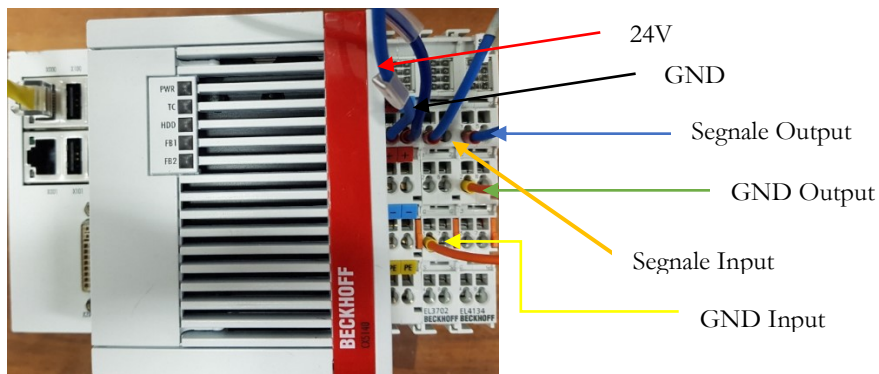


Figura 60 Connessione modulo EL4134 con la valvola Moog

Anche per l'invio del voltaggio è stato necessario scalare i valori come descritto dalla seguente formula:

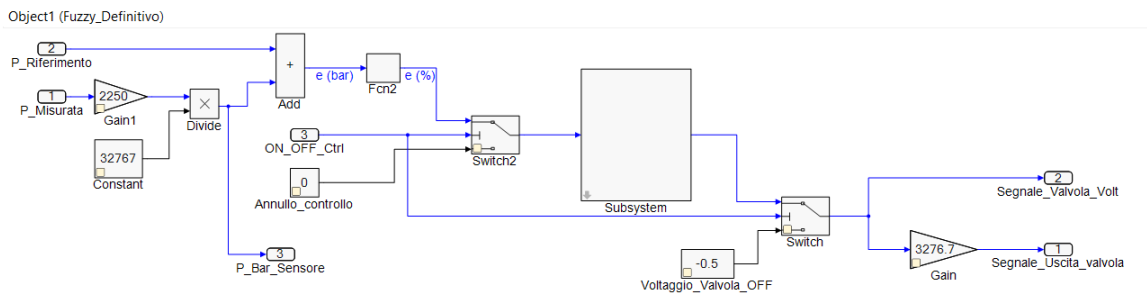
$$\text{Valore inviato alla servovalvola} = \text{Valore } -10/10V \text{ dal controllore} * \frac{32767}{10}$$

I collegamenti finali risultano essere quindi rappresentati nella seguente immagine:



**Figura 61** Collegamenti moduli CX

Ho dovuto quindi modificare anche il controllore precedentemente descritto inserendo le formule per scalare i valori. Il controllo è stato modificato in Simulink ed esportato nuovamente esattamente come descritto nella guida.



**Figura 62** Rappresentazione nuovo controllore Fuzzy PI

È stato inserito un primo “Gain1” collegato ad una divisione per implementare la formula del segnale analogico di ingresso e un secondo “Gain” per la formula del segnale di uscita. Sono stati inseriti anche due switch di controllo che verranno descritti successivamente quando verrà introdotta l’interfaccia grafica.

## 7.1 - Implementazione in TwinCAT

Una volta importato il progetto da Simulink sono andato a collegare i moduli analogici di input e output alla CX5140 effettuando prima una scansione degli stessi, in modo da poterli visualizzare nel progetto TwinCAT. Per effettuare il riconoscimento dei moduli ho selezionando con il tasto destro del mouse “Devices”, e poi “Scan”, ottenendo in “Term2” il modulo analogico di ingresso e in “Term3” il modulo analogico di uscita.



Figura 63 Scansione moduli Beckhoff

Prima di collegare i moduli al progetto ho dovuto creare delle variabili globali all'interno di un progetto PLC poiché per il collegamento finale, era necessario eseguire delle conversioni affinché i valori letti e inviati dall'oggetto importato da Simulink corrispondessero con i valori dei moduli collegati alla CX. Nell'immagine sottostante sono rappresentate le due righe di codice relative alle conversioni da LREAL a INT, per quanto riguarda l'uscita analogica e, da INT a LREAL, per l'ingresso analogico. Le altre variabili globali verranno descritte ed utilizzate nell'interfaccia grafica.

```
2  VAR_GLOBAL
3      Rif AT %Q* : LREAL;
4      Reale AT %I* : LREAL;
5      Ctrl AT %I* : LREAL;
6      ON_OFF AT %Q* : LREAL;
7      Uscita_Ctrl AT %I* : LREAL;
8      Ctrl_convertito AT %Q* : INT;
9      Ingresso_sens AT %I* : INT;
10     Uscita_sens AT %Q* : LREAL;
11 END_VAR
```

Figura 64 Variabili Globali

```
1  GVL.Ctrl_convertito:= LREAL_TO_INT(GVL.Uscita_Ctrl);
2  GVL.Uscita_sens:=INT_TO_LREAL(GVL.Ingresso_sens);
```

Figura 65 Conversioni all'interno del Main

Per concludere sono andato a collegare le variabili in questo modo:

- La variabile di uscita analogica del modulo è stata collegata alla variabile globale Ctrl.Convertito che verrà convertita in un intero e salvata all'interno della variabile Uscita.Ctrl
- La variabile globale Uscita.Ctrl è stata collegata all'uscita dell'oggetto importato da Simulink denominata Segnale\_Uscita\_Valvola
- La variabile analogica di ingresso del modulo è stata collegata alla variabile globale Ingresso.Sens che verrà convertita in LREAL e salvata nella variabile globale Uscita.Sens
- La variabile globale Uscita.Sens è stata infine collegata alla variabile di ingresso dell'oggetto importato da Simulink denominata P\_Misurata

Le variabili rimanenti vengono utilizzate nell'interfaccia grafica che ho implementato per poter ricevere e inviare dati alla CX.

## 7.2 - Interfaccia Grafica

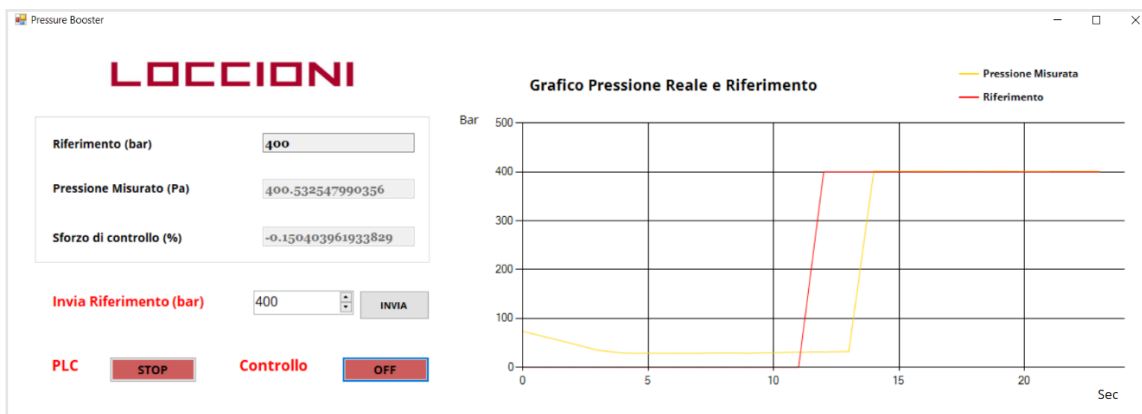


Figura 66 Interfaccia grafica di comando

L'interfaccia permette di visualizzare i parametri relativi al riferimento, alla pressione reale misurata dal sensore e allo sforzo di controllo. Inoltre, permette di inviare un segnale di riferimento, di azionare o disattivare il run del PLC con il pulsante "Stop/Start" e di azionare o disattivare il controllo Fuzzy PI con il pulsante "On/Off". Entrambi i pulsanti cambiano colore e scritta a seconda dello stato in cui

si trova il sistema. Nell'immagine notiamo che il tasto del PLC abbia la scritta "Stop", questo significa che il sistema è in run e che se volessimo fermare l'esecuzione del progetto è necessario premere il pulsante. Automaticamente diventerà di colore verde con la scritta "Start" in modo da poterlo premere di nuovo per avviare il progetto. Lo stesso vale per il controllo poiché in questo caso ci troviamo con il controllore attivo e per disattivarlo è necessario premere il pulsante "Off", anche in questo caso una volta premuto esso diventerà verde con la scritta "On" per poter riattivare il controllo successivamente. Nel caso della disattivazione del controllore il valore dato all'uscita non viene posto a 0, ma a -0,5V per evitare che la servovalvola raggiunga un equilibrio idraulico non a pressione nulla. Inserendo un valore negativo si tende a portare indietro lo stelo del moltiplicatore di pressione e quindi a decomprimere il fluido. Per poter effettuare l'attivazione e lo stop del controllore ho dovuto inserire i due switch introdotti precedentemente nell'immagine 62. Gli switch una volta selezionato il bottone "On/Off", permettono di settare la variabile globale "ON\_OFF" a 0 o 1 a seconda della situazione e di inviare un comando al controllore. Il valore dell'errore viene perciò impostato a 0, in modo tale che non esegua calcoli inutili e, il valore di uscita alla valvola a -0,5V per terminare la compressione.

Per poter utilizzare i parametri acquisiti tramite PLC e quindi attraverso la CX, all'interno di un progetto Visual Studio programmato in C# ed utilizzato esclusivamente per lo sviluppo dell'interfaccia grafica, è necessario seguire i seguenti passaggi. Per prima cosa si deve connettere la CX con il progetto in C# utilizzando la funzione in figura.

```
AdsClient.Connect("5.51.149.132.1.1", 851);
```

**Figura 67** Codice connessione Ads con PLC

La funzione utilizza una connessione tramite protocollo Ads prendendo in considerazione l'indirizzo AdsAmsNetID (5.51.149.132.1.1) e la porta (851) del PLC. Successivamente si inseriscono i toolbox che si desiderano avere all'interno della propria applicazione come, ad esempio, i due bottoni e le varie textbox che troviamo nel mio lavoro. Una volta fatto ciò, si possono modificare i parametri a piacimento



sia nel menù delle “properties” sia manualmente all’interno del codice. Per poter trasferire dati dal progetto PLC all’interfaccia grafica è necessario realizzare un handle che identifica la connessione con la variabile del PLC. Nella figura troviamo l’esempio relativo alla variabile globale ON\_OFF utilizzata per attivare o disattivare il controllore.

```
hON_OFF_Ctrl = AdsClient.CreateVariableHandle("GVL.ON_OFF");
```

**Figura 68** Codice realizzazione handle

In questo esempio l’handle è stato creato utilizzando la funzione “CreateVariableHandle” collegata con il Client tramite Ads, e denominato hON\_OFF\_Ctrl.

```
AdsClient.WriteAny(hON_OFF_Ctrl, ON_OFF_Ctrl);
```

**Figura 69** Codice invio valore al PLC

Successivamente, sempre attraverso protocollo Ads e quindi attraverso l’handle creato, utilizzando la funzione “WriteAny” è possibile inviare il valore della variabile “ON\_OFF\_Ctrl” realizzata in C#, che andrà a scrivere la variabile globale di output creata all’interno del PLC.

```
2  VAR_GLOBAL  
3  
4      ON_OFF AT %Q* : LREAL;  
5  
6  END_VAR
```

**Figura 70** Codice creazione variabile globale di output

Al contrario per poter ricevere e quindi visualizzare dati processati dal PLC all’interno dell’interfaccia grafica si utilizza la funzione “ReadAny” sempre realizzando un handle per permettere la connessione tramite protocollo Ads.

```
P_Reale = AdsClient.CreateVariableHandle("GVL.Reale");
Reale_textBox2.Text = ((double)AdsClient.ReadAny(P_Reale, typeof(double))).ToString();
```

**Figura 71** Codice ricezione valori da PLC

Nel caso in esempio si acquisisce il valore della pressione reale che viene memorizzato nella variabile globale del PLC “GVL.Reale” e inviato alla TextBox realizzata in C# che mi permette di visualizzare il valore della pressione.

In questo caso se volessimo un’acquisizione in tempo reale è necessario realizzare un timer che permetta di acquisire il valore ad ogni suo scatto.

```
public void timer1_Tick(object sendere, EventArgs e)
{
    P_Reale = AdsClient.CreateVariableHandle("GVL.Reale");
    Reale_textBox2.Text = ((double)AdsClient.ReadAny(P_Reale, typeof(double))).ToString();
}
```

**Figura 72** Funzione per acquisizione in tempo reale

Il timer naturalmente deve essere inizializzato scegliendo un intervallo a piacere che rappresenta ogni quanti millisecondi si vuole acquisire il valore e cioè ogni quanto tempo vogliamo attivare la funzione “Timer1\_Tick()”.

```
public void InitTimer()
{
    timer1 = new System.Windows.Forms.Timer();
    timer1.Tick += new EventHandler(timer1_Tick);
    timer1.Interval = 200;
    timer1.Start();
}
```

**Figura 73** Codice inizializzazione timer

Nel mio caso il timer è stato settato per scattare ogni 200 millisecondi. Naturalmente il Timer deve essere inizializzato e successivamente attivato utilizzando le seguenti funzioni.

```
InitTimer();
timer1.Start();
```

**Figura 74** Codice attivazione timer

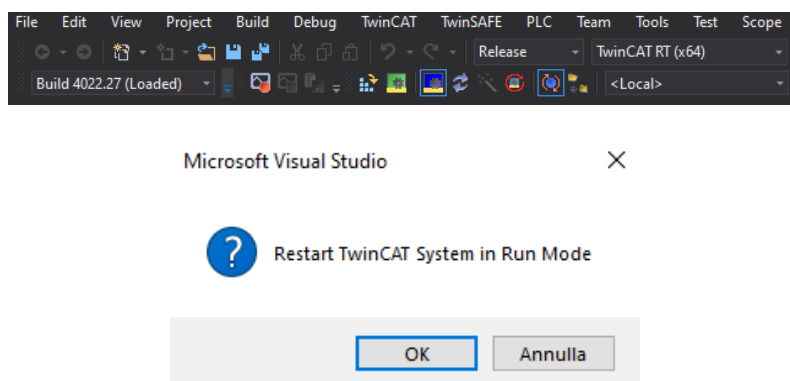
Inoltre, per utilizzare lo start e lo stop del PLC e quindi configurare il Login o il Logout ho utilizzato una funzione standard di TwinCAT (“WriteControl”) che permette di cambiare lo stato del in “Run” oppure “Stop”

```
AdsClient.WriteControl(new StateInfo(AdsState.Stop, AdsClient.ReadState().DeviceState));  
AdsClient.WriteControl(new StateInfo(AdsState.Run, AdsClient.ReadState().DeviceState));
```

**Figura 75** Codice login-logout PLC

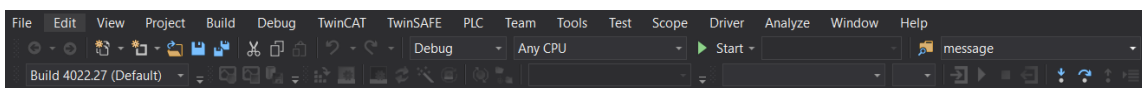
### 7.3 - Avvio progetto TwinCAT con interfaccia Grafica

Una volta eseguiti i vari collegamenti e realizzata l’interfaccia grafica è possibile selezionare l’opzione “Active configuration” nell’ambiente TwinCAT e successivamente “Ok” nella schermata “Restart TwinCAT System in Run Mode” in modo tale da configurare la CX in modalità “Run”.



**Figura 76** Attivazione Configurazione e Run Mode

Una volta riavviato TwinCAT in modalità Run è necessario effettuare il Login selezionando l’apposito bottone. Successivamente possiamo selezionare l’opzione “Start” nel progetto relativo all’interfaccia grafica in modo tale che esso si avvii.



**Figura 77** Attivazione interfaccia grafica

La schermata dell’interfaccia, come già descritto, ci permette quindi di inserire un riferimento all’interno dell’apposito spazio e di inviarlo alla CX selezionando il tasto “Invia”. Successivamente per azionare il controllo e quindi attivare il blocco relativo

alla logica Fuzzy è necessario selezionare il bottone “ON”. Una volta selezionato, il bottone diventerà rosso con una scritta “OFF” poiché se si desidera bloccare il controllo è necessario ripremere lo stesso bottone. La stessa procedura si verifica nel bottone “Start/Stop” per lo start o lo stop del login di Twincat. Nella parte superiore possiamo visualizzare il cambiamento dei valori della pressione acquisita dal sensore, del riferimento e del voltaggio inviato alla valvola MOOG in tempo reale. Ho introdotto anche un grafico che, come descritto nella legenda, permette di visualizzare l'andamento della pressione misurata rispetto al riferimento.

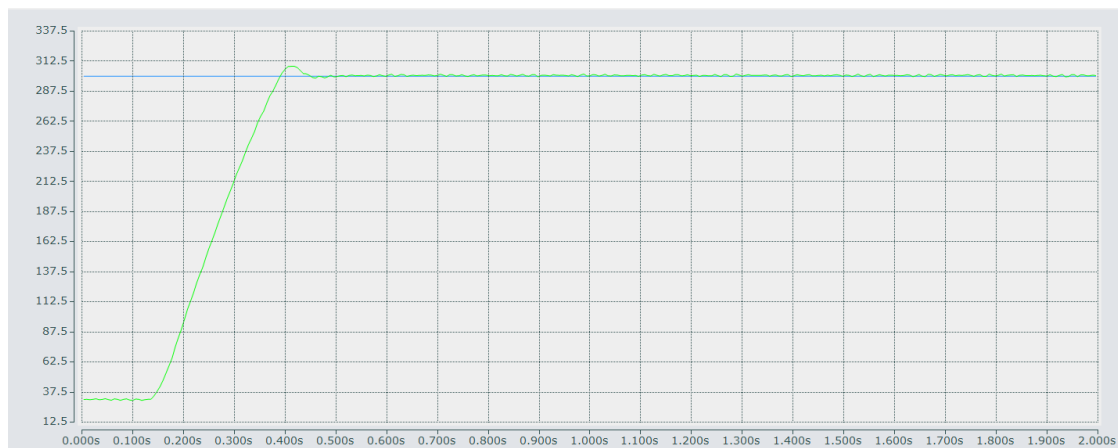
## Capitolo 8

### Risultati implementazione reale

Nella prova finale ho scelto di utilizzare inizialmente un riferimento a 300 bar come effettuato in fase di simulazione. L'unica differenza con la simulazione è la diminuzione della portata dell'olio del moltiplicatore a 100 Bar anziché i 250 Bar originali. La diminuzione di pressione è dovuta a motivi di sicurezza poiché per una prima implementazione, e quindi un primo test, è necessario verificare il funzionamento dei vari collegamenti e del controllore. Inoltre, il moltiplicatore di pressione su cui è stato effettuato il tuning del controllore non era disponibile al Kite Lab dell'impresa Loccioni e quindi, è stato utilizzato un moltiplicatore simile. Le differenze tra i due sono dovute alle aree delle camere di pressione. In particolare, l'area relativa alla prima camera di bassa pressione è più piccola del 16% mentre, quella relativa alla camera di alta pressione risulta maggiore del 41%. Di conseguenza essendo aumentate le aree è aumentato il peso relativo allo stelo del 51%.

Riportiamo le grandezze che sono variate:

Descrizione	Parametro	Valore Iniziale	Nuovo Valore
Area del pistone	$A_1$	$(\pi/4) \cdot 0,015^2 m^2$	$(\pi/4) \cdot 0,0125^2 m^2$
Area dello stelo	$A_2$	$(\pi/4) \cdot 0,0035^2 m^2$	$(\pi/4) \cdot 0,006^2 m^2$
Lunghezza camera LP	$S_{LP}$	0.28 m	0.3 m
Lunghezza camera HP	$S_{HP}$	0.282 m	0.32 m
Massa del pistone più stelo	$m$	10.27 $k_g$	20.49 $k_g$



**Figura 78** Simulazione riferimento 300 Bar e pressione olio 100 Bar

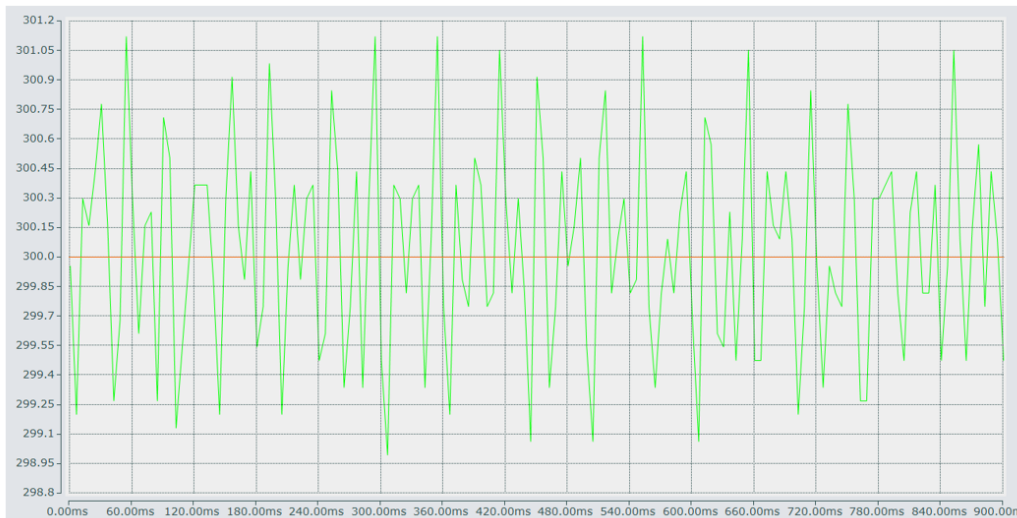
Dal grafico relativo ad una prima simulazione possiamo notare come il controllore vada a riferimento con tempi ragionevoli e con una piccola sovraelongazione iniziale. Il tutto è confermato analizzando i risultati ottenuti dalla stessa funzione utilizzata in fase di simulazione.

Tempo di salita	0,0339 s
Tempo di assestamento	0,0785 s
Sovraelongazione	6,5966 Bar

Notiamo come l'errore di sovraelongazione sia di 6,57 bar che rapportato al riferimento e trasformato in percentuale ci fornisce come risultato 2,1989%.

$$\text{Sovraelongazione}(\%) = \text{Sovraelongazione}(\text{Bar}) \cdot \text{Riferimento} / 100$$

Quindi il sistema compie una sovraelongazione iniziale che però viene attenuata immediatamente diminuendo il tempo di assestamento. La sovraelongazione è dovuta al cambiamento del moltiplicatore. Precedentemente la robustezza del controllore è stata verificata variando tutti i coefficienti positivamente o negativamente ma, in questo caso, abbiamo una variazione positiva su due coefficienti e una negativa su un terzo valore. La robustezza è comunque garantita dal fatto che le sovraelongazioni rimangono contenute e che il controllore converge a regime con tempi di salita e di assestamento estremamente ridotti.



**Figura 79** Pressione misurata a riferimento

Effettuando un ingrandimento sul grafico notiamo come a regime il sistema risulti stabile poiché mantiene oscillazioni in un range di  $\pm 2$  Bar. Le oscillazioni sono ottime considerando che l'accuratezza del sensore è dello 0,1% e che su un fondo scala di 2250 bar corrisponde ad un errore di  $\pm 2,25$  Bar. Il sistema rimanendo al di sotto dell'accuratezza del sensore ci permette di considerare il segnale perfettamente a riferimento poiché le oscillazioni generate rappresentano il rumore bianco che fa sì che non si registri nessun movimento e che la valvola mantenga una portata costante e quindi una pressione a riferimento.

Per completezza ho ripetuto la simulazione del controllore diminuendo nel modello la pressione dell'olio a 100 Bar e aggiornando i parametri rispetto al nuovo moltiplicatore. Analizzando il risultato ho ottenuto i seguenti valori:

Tempo di salita	0,0828 s
Tempo di assestamento	0,1096 s
Sovraelongazione	0,0714 Bar

Dalla tabella risulta che in simulazione la sovraelongazione è praticamente nulla mentre, nella realtà essa aumenta fino a 6 Bar. L'aumento è dovuto alla non linearità del sistema che non risulta identico al modello. Inoltre, i tempi di salita e assestamento sono addirittura minori nel caso reale, questo perché il sensore di pressione in simulazione registra 0 Bar nel momento in cui la valvola è spenta. Al contrario, nella

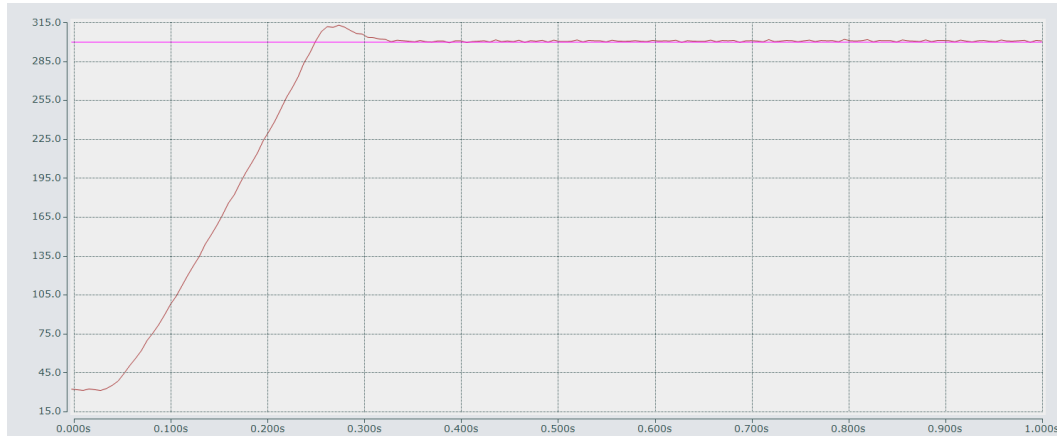
realizzazione reale la pressione registrata non scende mai al di sotto dei 30 Bar comportando una diminuzione dei tempi di salita e di conseguenza dei tempi di assestamento.

## 8.1 - Risultati implementazione finale

Una volta verificato in sicurezza che il controllo funzionasse, ho aumentato la pressione dell'olio del moltiplicatore a 250 Bar in modo da ottenere il massimo delle prestazioni. I test sono stati effettuati con riferimenti di 300, 500 e 700 Bar poiché l'iniettore non supporta pressioni superiori ai 1000 bar. In questo caso però è stata ridotta la portata massima della pompa da  $\pm 10V$  a  $\pm 5V$  per evitare problemi di rottura e per i soliti motivi di sicurezza.

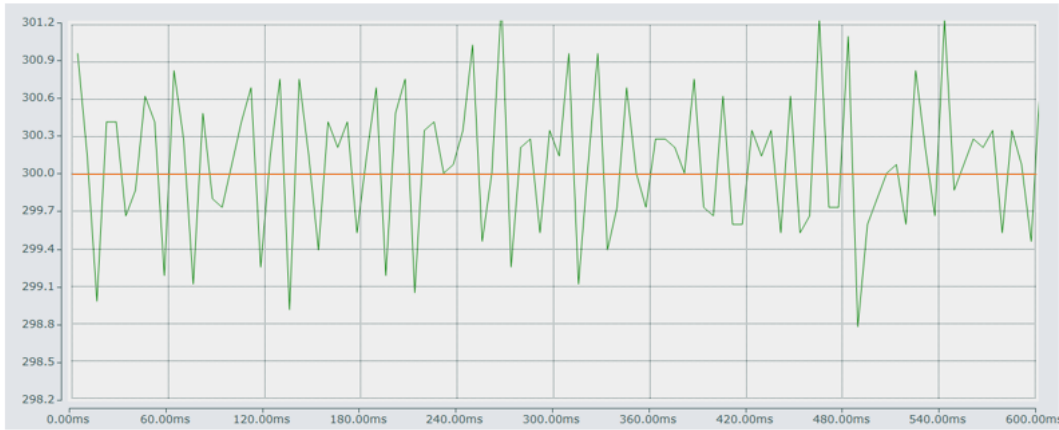
Nelle immagini successive vengono rappresentate le pressioni misurate dei vari riferimenti analizzati, visualizzando anche i rispettivi comportamenti a regime.

- Riferimento a 300 bar con  $\pm 5V$  di uscita massima



**Figura 80** Pressione misurata e Riferimento a 300 Bar

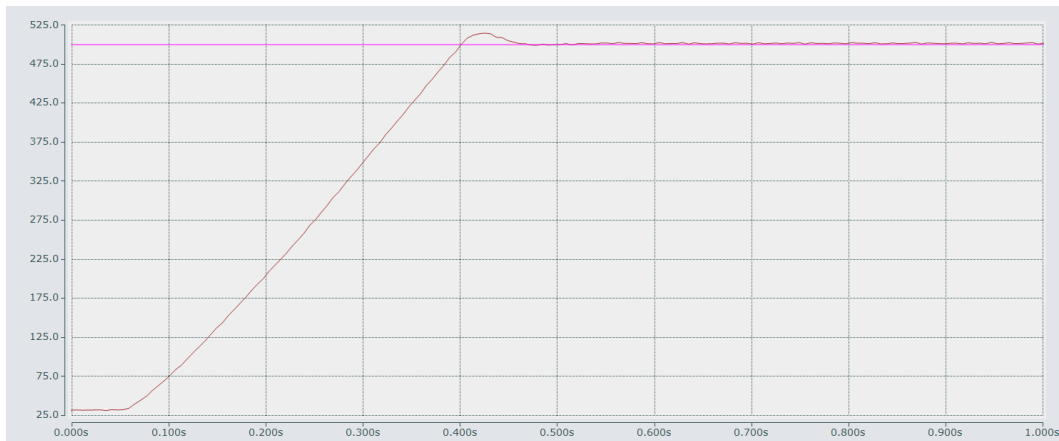




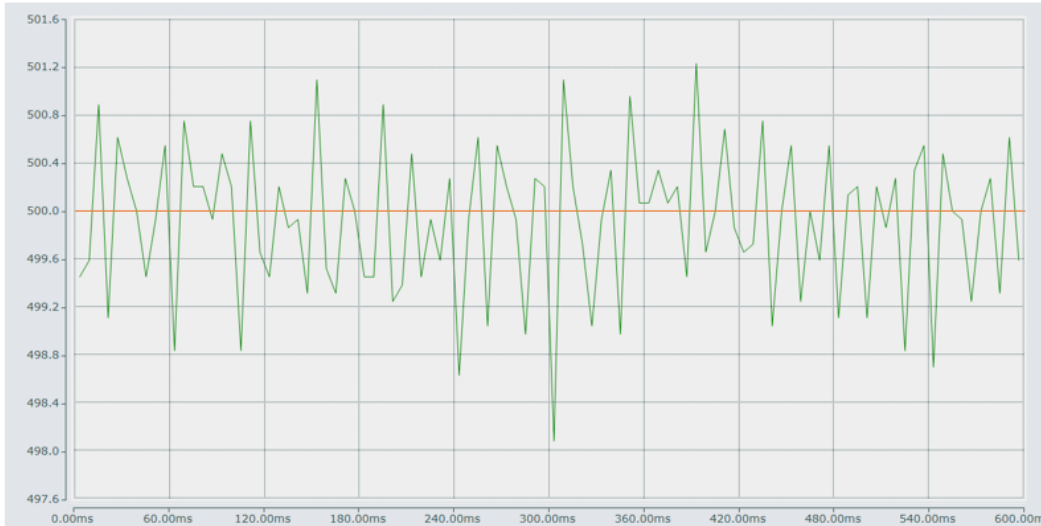
**Figura 81** Pressione misurata a regime

Tempo di salita	0,0259 s
Tempo di assestamento	0,0487 s
Sovraelongazione	13,8066 Bar

- Riferimento 500 bar con  $\pm 5V$  di uscita massima



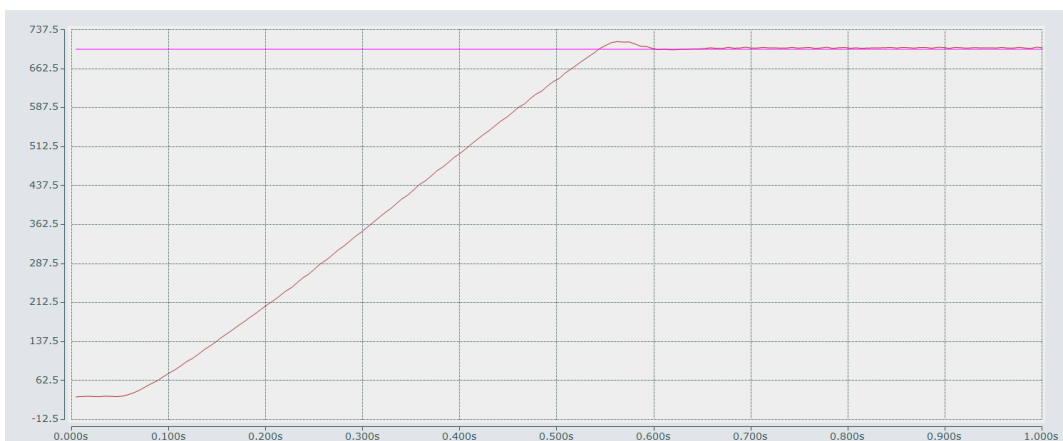
**Figura 82** Pressione misurata e Riferimento a 500 Bar



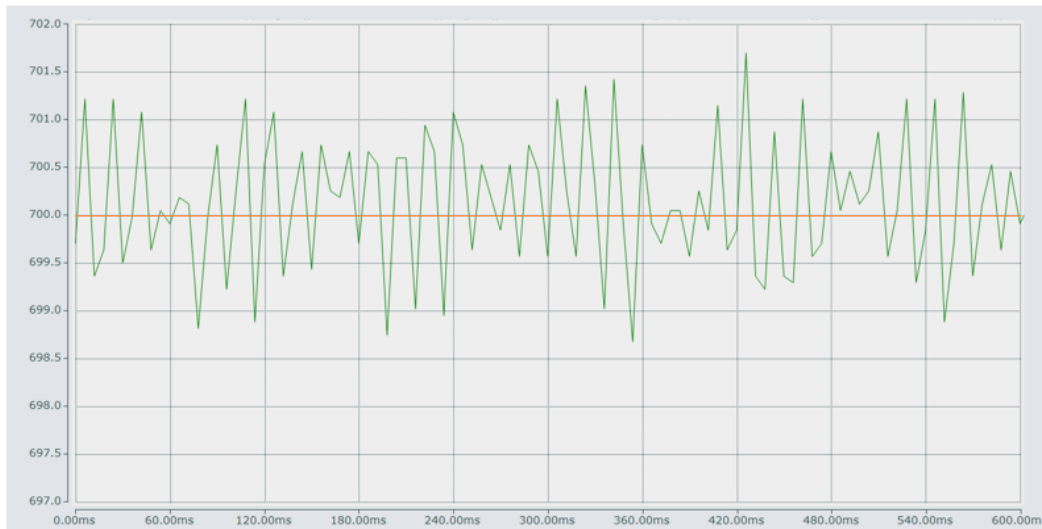
**Figura 83** Pressione misurata a regime

Tempo di salita	0,0430 s
Tempo di assestamento	0,0568 s
Sovraelongazione	16,3045 Bar

- Riferimento 700 bar con  $\pm 5V$  di uscita massima



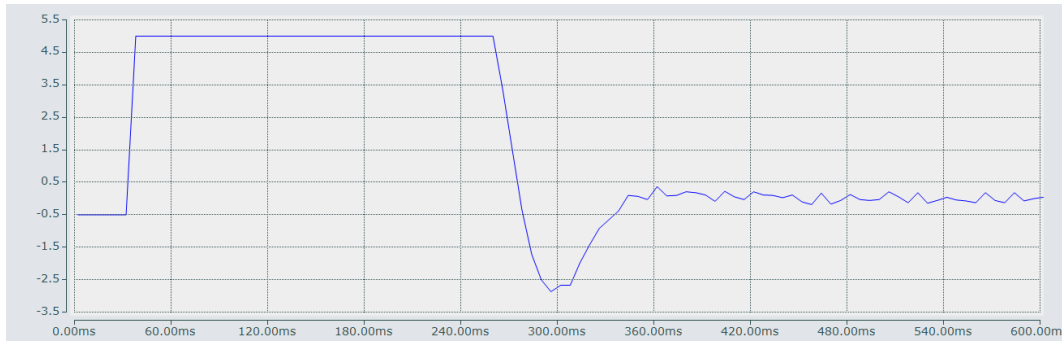
**Figura 84** Pressione misurata e Riferimento a 700 Bar



**Figura 85** Pressione misurata a regime

Tempo di salita	0,0613 s
Tempo di assestamento	0,0780 s
Sovraelongazione	14,1331 Bar

I risultati ottenuti aumentando la pressione dell'olio sono coerenti con la prima simulazione e non prevedono oscillazioni a regime maggiori di quelle ottenute precedentemente. Posso quindi concludere che il controllo funziona correttamente su questo tipo di moltiplicatore anche se presenta delle piccole sovraelongazioni che, sono causate dalla non coerenze dei valori utilizzati in fase di tuning con quelli del moltiplicatore utilizzato. Inoltre, vorrei soffermarmi sull'analisi dello sforzo di controllo. Prendiamo come esempio l'andamento della prova effettuata con riferimento a 300 Bar.



**Figura 86** Sforzo di controllo riferimento 300 Bar

Come volevasi dimostrare l'andamento dello sforzo di controllo è coerente con il controllo effettuato. Infatti, presenta un ingente aumento iniziale quando il sistema si trova molto lontano dal riferimento. Successivamente avvicinandoci al valore desiderato il controllore diminuisce lo sforzo di controllo, effettuando una piccola sottoelongazione per poi stabilizzarsi a riferimento. A regime troviamo delle piccole oscillazioni dovute alla non perfetta stabilità del controllore. Come visto precedentemente sappiamo che esso oscilla tra  $\pm 2$  Bar che risultano comunque poco significativi perché la servovalvola mantiene un funzionamento intorno a  $\pm 0,5V$  che, non presenta problemi per future rotture dovute all'azionamento e lo spegnimento continuo della stessa. Infatti, per evitare l'equilibrio idraulico, nella fase di spegnimento del controllore, ho inserito una tensione negativa alla valvola di  $-0,5V$ . In questa situazione la valvola risulta spenta e non in fase di ricarica. Lo stesso avviene quindi durante il controllo non influenzando negativamente sui risultati finali. Anche in questo caso ho dovuto ripetere le simulazioni aggiornando i valori dei parametri derivati dal nuovo moltiplicatore ottenendo i seguenti risultati.

<b>Riferimento</b>	<b>Tempo di salita</b>	<b>Tempo di assestamento</b>	<b>Sovraelongazione</b>
300 Bar	0.0179 s	0.0251 s	22.3341Bar
500 Bar	0.0310 s	0.0398 s	17.6149 Bar
700 Bar	0.0460 s	0.0563 s	10.7872 Bar

I risultati nel caso reale sono più o meno coerenti con quelli ottenuti in simulazione. Possiamo notare come ci siano alcune differenze riguardanti le sovraelongazioni e la netta diminuzione dei tempi in fase di simulazione. Ciò è dovuto all'impossibilità di replicare il sistema perfettamente in simulazione. Come descritto nel primo capitolo nella fase di implementazione del modello sono state utilizzate delle semplificazioni che fanno sì che il sistema si discosti leggermente da quello reale. I risultati sono comunque ottimi perché il controllore, durante l'implementazione reale, riesce a convergere a regime con tempi minimi e con sovraelongazioni coerenti con le simulazioni. Ricordiamo che il controllore è stato esportato direttamente da Simulink in TwinCAT ed eseguito senza ulteriori modifiche. La stessa cosa non è stata possibile utilizzando il controllore PID poiché presentava già in fase di simulazione ingenti sovraelongazioni che, avrebbero sicuramente creato problemi ed eventuali rotture durante la fase realizzativa. A regime, come già introdotto, è stato verificato un andamento coerente con l'accuratezza del sensore poiché il segnale rimane sempre contenuto all'interno del range di  $\pm 2,25 \text{ bar}$  tale da poter considerare le oscillazioni non misurabili e quindi pari al rumore bianco del sensore.

## Conclusioni

Posso concludere che per quanto riguarda il sistema proposto, il controllo Fuzzy PI risulta robusto alle variazioni rispetto al classico controllo PI, che al contrario deve essere tarato ogni qual volta si effettuano modifiche al modello o al sistema. Ho verificato come il controllo Fuzzy, una volta implementato su simulazioni da modello, risulta perfettamente eseguibile in un caso reale senza nessun problema di taratura. Infatti, una volta collegata la CX ho potuto effettuare test ottenendo risultati coerenti con le simulazioni fin da subito. I test hanno affermato l'effettiva superiorità del controllo Fuzzy PI rispetto al classico controllo PI, non solo per l'elevata robustezza ma, anche per le ottime performance ottenute rispetto ai tempi di salita e di assestamento che risultano essere nettamente migliori. Infine, il lavoro effettuato mi ha permesso di verificare che un qualsiasi modello Simulink è completamente esportabile, e quindi eseguibile, in ambiente TwinCAT ottenendo i medesimi risultati. L'introduzione di una guida che mi permetta l'esportazione è un ottimo passo per quanto riguarda la validazione dei modelli poiché, permette di eseguire test reali su modelli implementati precedentemente in Simulink, software che appunto da solo non gode di proprietà implementative su hardware.

Un eventuale sviluppo futuro potrebbe essere quello di tarare in simulazione il controllo Fuzzy PI rispetto al moltiplicatore utilizzato nella fase implementativa per poter verificare l'equivalenza dei risultati. Successivamente si potrebbe aumentare la tensione della valvola fino a  $\pm 10V$  per verificare l'andamento finale del sistema.

# Appendice A

```
1 - addpath(genpath('valvola_G765')) %Aggiunta modello valvola
2 - Ts = 1e-3; %Tempo di campionamento
3 - p2=[1 1.2 1.5 1.7 0.8 0.5 0.2]; %Percentuali di variazione
4 - Rif=0;
5 - rif=[300 500 700]; %Riferimenti
6 - Risultati=cell(length(rif),5); %Tabella finale
7 - Risultati{1,1}='Riferimento';
8 - Risultati{1,2}='errore';
9 - Risultati{1,3}='Tempo di salita';
10 - Risultati{1,4}='Tempo di assestamento';
11 - Risultati{1,5}='Sovraelongazione';
12 - Risultati{1,6}='p';
13 - f=1;
14 - j=0;
15
16 - for i=1:length(rif)
17 -     Rif=rif(i);
18 -     for t=1:length(p2)
19 -         p=p2(t);
20 -         bulk1 = (1.69823e+09)*p; % (Pa) bulk modulus Olio Idraulico VG46
21 -         bulk2 = (1.5e9)*p; % (Pa) bulk modulus fluido test ISO4113
22 -         A1 = (((150e-3)^2)*pi/4)*p; % (m^2) Area pistone
23 -         A3 = (((35e-3)^2)*pi/4)*p; % (m^2) Area stelo
24 -         A2 = A1 - A3;
25 -         Abar = A2/A1;
26 -         rapp = A1/A3;
27 -         sLP = (280e-3)*p; % (m) stroke camera LP
28 -         sHP = (282e-3)*p; % (m) stroke camera HP
29 -         V01 = (38.17e-6)*p; % (m^3) volume morto camera LP1
30 -         V02 = V01; % (m^3) volume morto camera LP2
31 -         V03 = (0.5e-6)*p; % (m^3) volume morto camera HP
32 -         m = 10.27*p; % (Kg) massa pistone + stelo
33 -         Fc = 2*p; % (N) Forza di Coulomb
34 -         mu = 0.01275*p; % (N/m/s) coefficiente di attrito viscoso
35
36 -         s = 1+t+j;
37
38 -         sim('modello_equaz_diff',20)
39 -         n = 1;
40 -         Valori = pHP2.Data;
41 -         Tempo= pHP2.Time;
42 -         S=stepinfo(Valori, Tempo, rif(i),'SettlingTimeThreshold',0.1);
43 -         Risultati{s,6}=p;
44 -         Risultati{s,1}=rif(i);
45 -         Risultati{s,5}=(S.Overshoot)*rif(i)/100;
46 -         Risultati{s,3}=S.RiseTime;
47 -         Risultati{s,2}=S.Overshoot;
48 -         Risultati{s,4}=S.SettlingTime;
49
50 -     end
51
52 -     j=j+length(p2);
53 - end
```

# Bibliografia

- [1] Yager R.R. e Filev D.P., *Essentials of Fuzzy Modelling and Control*, Wiley-Interscience, 1994
  
- [2] Driankov D, Hellendoorn H., Reinfrank M., *An introduction to Fuzzy Control*, Springer-Verlag, 1993
  
- [3] D. Dobois e H. Prade, *Fuzzy Sets and Systems: Theory and Applications*, Academic Press, 1980
  
- [4] Patyra M.J. e Mlynek D.M., *Fuzzy Logic, Implementation to Fuzzy Control*, Springer-Verlag, 1996
  
- [5] *Moog D765 Series Servovalves*, Datasheet
  
- [6] *Manual CX51x0 Embedded-PC*, Beckhoff Automation GmbH & Co
  
- [7] *TwinCAT 3 Matlab Simulink Integration*, Beckhoff Automation GmbH & Co
  
- [8] *Datasheet EL37x2, Analog Input Terminals*, Beckhoff Automation GmbH & Co
  
- [9] *Datasheet EL41xx, Analog Output Terminals*, Beckhoff Automation GmbH & Co



## Ringraziamenti

Sono due anni che aspetto questo momento, quell'attimo in cui ti soffermi a guardare il capitolo finale della tesi sperando di riuscire a scrivere qualcosa senza neanche accorgerti che quest'ultimo capitolo non rappresenta solo l'epilogo della tesi, ma la conclusione di un cammino iniziato ben 5 anni fa. Guardandomi indietro mi rendo conto che sono cambiate tante cose nella mia vita dal modo di pensare fino al modo di affrontare le sfide. Quindi scrivere i ringraziamenti è solo un modo per esternare i propri sentimenti verso coloro che durante il percorso hanno sempre creduto in te, anche quando a volte non ci credevi neanche tu.

Inizierei con il ringraziare il professor Giuseppe Conte per avermi aiutato e spronato a completare nella maniera migliore possibile questo lavoro e l'azienda Loccioni che mi ha permesso di svolgere questo lavoro all'interno dell'impresa accogliendomi come un loro collaboratore.

Passando ora alla mia famiglia, il primo pensiero va ai miei genitori senza i quali non sarei qui ora e non avrei potuto coronare i miei molteplici sogni e a mia sorella Alessia che per tutta la durata del percorso di laurea non ha fatto altro che ripetermi quanto sia facile conseguire la laurea in ingegneria, anche se in fondo sa che, nonostante tutto, lei non ce l'avrebbe mai fatta. Non posso dimenticare i miei nonni che con i loro sorrisi e pianti per ogni esame superato accrescevano in me la voglia di andare avanti e fare sempre meglio, lo Zio che, anche se lontano, mi è sempre vicino e sempre pronto ad aiutarmi. Zia Romina che spero riesca a guarire e a partecipare alla cerimonia di laurea, perché anche lei, seppur lontana geograficamente, è sempre stata vicina sentimentalmente.

Un grazie particolare va a Martina, che mi ha accompagnato per tutto questo viaggio restandomi vicina, avendo fiducia in me anche e soprattutto nei momenti di maggiore difficoltà. È piccola, è vero, ma determinata e so che potrò sempre contare su di lei. Un grazie particolare va ai ragazzi dell'università, Bracco, Fabio, Luca, Paki e Vito, il gruppo più strano e più folle che abbia mai visto, ma anche il più affiatato, sul quale ho capito fin dall'inizio di poter fare pieno affidamento.

Inoltre, vorrei ringraziare tutti i miei amici, citandoli uno ad uno, ma sarebbe impresa troppo impegnativa dato il numero smisurato, anche se sono sicuro che tanti se ne dispiaceranno di non poter passare alla storia. Scherzi a parte, vi ringrazio per l'aiuto grazie al quale ho potuto staccare la spina dagli impegni universitari.

Non me ne vogliano gli altri, ma un pensiero particolare lo vorrei dedicare a nonna "Ia" che purtroppo ci ha lasciato proprio a metà del percorso. Ad ogni esame pensava che da solo non sarei riuscito a superarlo e allora mi diceva che avrebbe pregato uno di quei suoi santi affinché lo passassi. Alla fine, nonna, so che non mi hai mai abbandonato e hai "obbligato" i Santi a ripassare matematica, algebra e fisica. Finalmente ce l'ho fatta! Ti vorrò bene per sempre.