POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Elettrica

Tesi di Laurea Magistrale

Simulazione e test HiL del controllo di azionamenti elettrici in ambienti MATLAB e PLECS

Relatore
Prof. Gianmario Pellegrino

Candidato
Emanuele Giamello

Anno accademico 2018-2019

Abstract

Questa tesi di laurea ha come obiettivo principale quello di confrontare e valutare l'efficienza e la precisione di due programmi di modellistica e simulazione: *Simulink* (MathWorks) e *PLECS* (Plexim) utilizzati per la simulazione del controllo di azionamenti elettrici.

Il lavoro di tesi è suddiviso in due parti: la prima parte si concentra su un confronto tra i due ambienti citati e tra i diversi componenti disponibili. A partire da dei modelli Simulink forniti durante il corso *"Controllo digitale di convertitori e azionamenti"* vengono sviluppati dei modelli con diverse caratteristiche di convertitore su entrambi i software con un confronto finale in termini di risultati e tempi di calcolo con l'obiettivo di determinare i vantaggi e gli svantaggi di un determinato modello o ambiente anche in un'ottica didattica.

Nella seconda parte viene introdotto un modello Hardware-in-the-loop sviluppato a partire dai dati raccolti da un motore sincrono a magneti permanenti ad oggi utilizzato per le esercitazioni del corso già citato. In particolare viene presentato il simulatore real time *"RT Box"* di proprietà Plexim, progettato quindi per lavorare con il software PLECS. Sul simulatore viene caricato un modello contenente le componenti di potenza (inverter e motore) mentre il codice C di controllo è implementato su un microcontrollore STM connesso alla RT Box.

Ringraziamenti

Ed eccomi giunto ai ringraziamenti, l'ultima pagina di un percorso che per me è iniziato quattordici anni fa quando la maestra in quinta elementare mi chiese: "cosa vuoi fare da grande?". La mia risposta fu, ovviamente: *"l'ingegnere"*.

Iniziando dai ringraziamenti ufficiali ringrazio il mio relatore, il professor Pellegrino, per il tempo dedicatomi durante lo svolgimento di questo progetto.

Un ringraziamento speciale va poi alla mia famiglia, ed in particolare ai miei genitori che mi hanno sempre sostenuto ed incoraggiato; se oggi sono qui è in gran parte merito loro.

Ringrazio gli amici, quelli di sempre, i pochi rimasti dal liceo e che ancora oggi sono qui al mio fianco ma anche i nuovi amici e le persone che sono entrate da poco nella mia vita ma l'hanno resa subito speciale.

Ringrazio quegli amici che con me hanno condiviso le gioie e i dolori del Politecnico, con i quali e grazie ai quali ho imparato tanto...persino un po' di pugliese.

Le ultime righe di questi ringraziamenti vanno ad una persona per me speciale, una persona che dal mio primo giorno di università mi ha chiamato con orgoglio "ingegnere" e che più di tutti vorrebbe essere al mio fianco oggi. Le ultime righe di questi ringraziamenti vanno a mio nonno Mario a cui dedico questa tesi.

Indice

Elenco delle tabelle

Elenco delle figure

Capitolo 1

Introduzione

L'idea alla base di questa tesi è quella di confrontare e verificare l'efficacia, intensa in termini di qualità e di velocità, di due ambienti di modellistica e simulazione nell'ambito del controllo degli azionamenti elettrici. I due software sono *Simulink* di proprietà Mathworks, probabilmente più noto nell'ambiente ingegneristico e pensato per modellare e simulare una vasta gamma di sistemi non necessariamente elettromeccanici e *PLECS* di proprietà Plexim progettato con una particolare attenzione per il mondo degli azionamenti elettrici.

L'obiettivo di questa tesi è principalmente didattico, il confronto dei due ambienti di simulazione e la successiva integrazione di un sistema Hardware-in-the-loop sono effettuati a partire da modelli di azionamenti e sistemi di controllo utilizzati durante il corso *"Controllo digitale di convertitori e azionamenti"* tenuto dal Professor. Pellegrino per il corso di laurea magistrale in Ingegneria Elettrica del Politecnico di Torino; sono presentati dei modelli alternativi a quelli ad oggi utilizzati con lo scopo di valutare i pro e i contro di un ambiente o di un determinato tipo di modello; questa tesi non è però esclusivamente rivolta al mondo didattico, l'utilizzo di questi software è sicuramente radicato nel mondo universitario ma ritengo che anche nel mondo industriale la possibilità di modellare e simulare in modo accurato la realtà stia diventando sempre più importante, in particolare per esempio grazie ai sistemi Hardware-in-the-loop, anche e soprattutto quando l'obiettivo principale può diventare la velocità o il contenimento dei costi di realizzazione di un progetto. Ritengo quindi che capire ed analizzare quali siano le potenzialità o i vantaggi di questi software ma anche la loro accessibilità e facilità di utilizzo sia un obiettivo utile e attuale anche in quel mondo industriale che negli ultimi anni si è definito "Industry 4.0".

La prima parte dell'elaborato è quindi dedicata al confronto dei due ambienti e delle componenti fornite dalle rispettive librerie utili alla realizzazione di un modello di azionamento elettrico. In particolare viene analizzato il comportamento di diversi modelli di convertitore, a partire da un inverter ai valori medi fino ad introdurre un modello realizzato con interruttori "reali".

Nella seconda parte della tesi viene invece presentato un modello Hardware-in-the-loop. Anche in questo caso il punto di partenza è didattico: viene infatti realizzato e simulato in real-time il modello di un azionamento di un motore sincrono a magneti permanenti attualmente utilizzato dagli studenti per realizzare e testare gli algoritmi di controllo introdotti durante il corso del Prof. Pellegrino. La simulazione è realizzata grazie al real-time computer *RT Box* prodotto dalla Plexim ed ovviamente compatibile con il loro ambiente di modellistica PLECS e, grazie ad una serie di Input/Output analogici e digitali, con un qualunque microcontrollore (MCU), in questo caso il codice di controllo è implementato sulla scheda Nucleo STM32.

1.1 PLECS

Prima di procedere viene presentato brevemente l'ambiente PLECS. Citando letteralmente il sito del produttore, il software viene descritto come: *"the tool of choice for high-speed simulations of power electronic systems"* [16].

L'idea alla base di PLECS è infatti quella di creare un programma di modellistica e simulazione specificatamente progettato per il mondo elettrico/elettronico, riconducendo il processo di modellistica il più possibile ad uno schema circuitale. La libreria inclusa nel software consente di modellare tutti gli aspetti dei sistemi di power elettronics e del loro controllo. Troviamo infatti componenti essenziali per il mondo elettrico ma anche magnetico, termico e meccanico; il tutto presentato tramite una finestra drag and drop ed un editor schematico.

La libreria più fornita è ovviamente quella elettrica: PLECS offre una vasta gamma di componenti a partire da quelli passivi fino ad una serie di circuiti di potenza basati su componenti ideali ma anche in grado di includere specifiche di produzione di componenti reali in modo da simulare correttamente i fenomeni parassitici e termici. É inoltre presente una collezione di modelli di macchine elettriche con vari gradi di dettaglio e tutti interamente modificabili. Sono presenti anche componenti non lineari ed una serie di modelli di trasformatori pronti all'uso.

Anche per quanto riguarda la modellistica termica e magnetica viene riproposto lo schema circuitale tramite l'utilizzo di circuiti equivalenti. Viene infine fornito un insieme di componenti meccaniche progettato per esempio per interagire facilmente con i modelli delle macchine elettriche in modo da modellare e simulare un sistema di controllo completo. Inoltre, grazie all'utilizzo di blocchi appositamente progettati per il controllo dei convertitori di potenza e soprattutto grazie al blocco *C-Script* che consente di implementare direttamente in PLECS degli algoritmi di controllo in ANSI-C è facilmente possibile realizzare qualunque schema di controllo digitale.

Esistono due versioni del software: "PLECS Blockset" e "PLECS Standalone". Entrambe presentano la stessa libreria con componenti compatibili tra loro. La versione "Blockset" è integrabile con Simulink/MATLAB e consente di usare i modelli e i componenti di PLECS con il solver, i blocchi di controllo e lo Script

Editor di Simulink. Al contrario la versione "Standalone" si presenta come un piattaforma di simulazione indipendente con un solver proprietario e ottimizzato. Durante lo svolgimento di questa tesi è stata utilizzata la versione "Standalone".

La Plexim produce anche un simulatore real-time chiamato *"RT Box"* che può essere programmato e utilizzato direttamente da PLECS tramite l'utilizzo di un apposita libreria facilmente integrabile con le componenti e i modelli già citati [16]. Le caratteristiche e le specifiche del simulatore sono approfondite nella seconda parte di questo elaborato.

Capitolo 2

Modelli e simulazioni

La prima parte di questa tesi è, come già detto, dedicata alla presentazione di diversi modelli di due controlli di azionamenti elettrici. In particolare, nelle prossime pagine sono analizzati e confrontati dei modelli di controllo di un motore sincrono a magneti permanenti e di un asincrono realizzate su Simulink e PLECS. Lo scopo è quello di valutare la convenienza di un determinato modello o software rispetto al modello benchmark già citato nell'introduzione e presentato poco più avanti. Verranno confrontati l'accuratezza della simulazione e la sua precisione ma anche la velocità di calcolo.

In particolare vengono presentati tre modelli diversi dal punto di vista dell'inverter: il primo modello è definito "Average", il secondo "Logico" ed infine un modello "Fisico". I modelli *Average* e *Logico* non includono il deadtime o le cadute di tensione delle componenti elettroniche di potenza che sono quindi eliminati anche dal modello benchmark durante il confronto; bisogna inoltre disattivare la compensazione delle cadute di deadtime all'interno del codice di controllo. Sia le impostazioni all'interno del blocco inverter sia la compensazione del deadtime tramite codice sono ripristinate per il confronto con i modelli *Fisici*.

Durante il corso *"Controllo digitale di convertitori e azionamenti"* per la simulazione dei modelli benchmark è stato utilizzato un solver di tipo Fixed-step, quando possibile in questo elaborato viene introdotto anche un solver Variable-step. Un solver di tipo fixed utilizza per tutto l'intervallo simulato lo stesso valore temporale per gli step di simulazione, un solver variable al contrario è in grado di adattare l'ampiezza del passo in funzione della rapidità degli eventi simulati, se non avvengono cambiamenti rapidi lo step size viene incrementato mentre viene ridotto qualora sia necessario simulare eventi che si sviluppano su tempi minori del passo di simulazione; ovviamente questo implica la necessità di una maggiore potenza di calcolo ma può anche portare ad un minor numero di step complessivi. Diventa quindi interessante confrontare l'impatto del solver in termini di tempi di calcolo.

2.1 Modelli Benchmark

In questa sezione sono presentati i componenti e i risultati di simulazione dei modelli benchmark sulla base di cui vengono poi sviluppati i tre "nuovi" modelli. Come già detto i motori modellati sono due: uno sincrono a magneti permanenti (SPM) e uno asincrono (IM).

2.1.1 Modello benchmark SPM

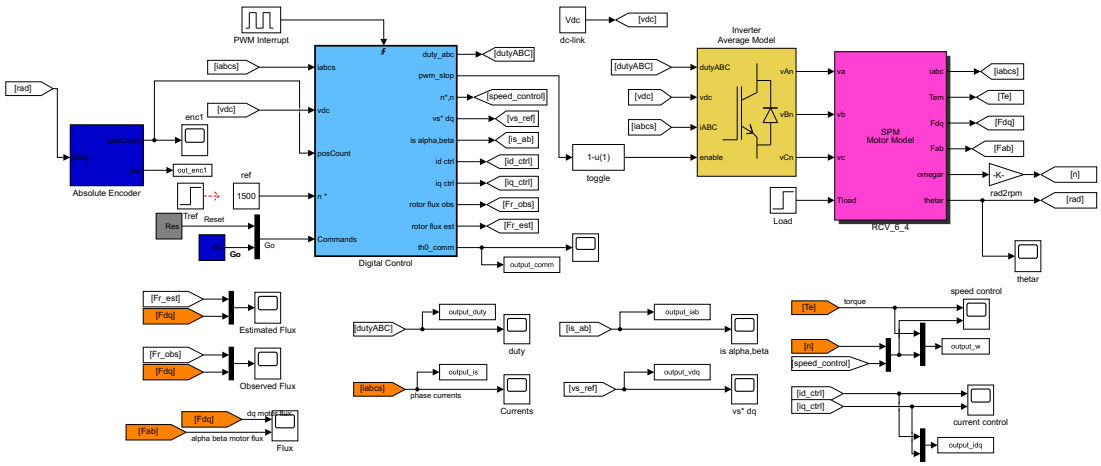


Figura 2.1: Modello Benchmark Motore SPM

Il modello benchmark del motore SPM si basa sui valori riportati nel codice 2.1 che viene richiamato ad inizio simulazione. I valori elettrici e meccanici sono ottenuti a partire dalle misure effettuate sul motore *RCV 6_4* [13].

Per il motore SPM è stato implementato un controllo in anello chiuso di velocità riassunto nello schema a blocchi di Figura 2.2 [3, p. 449].

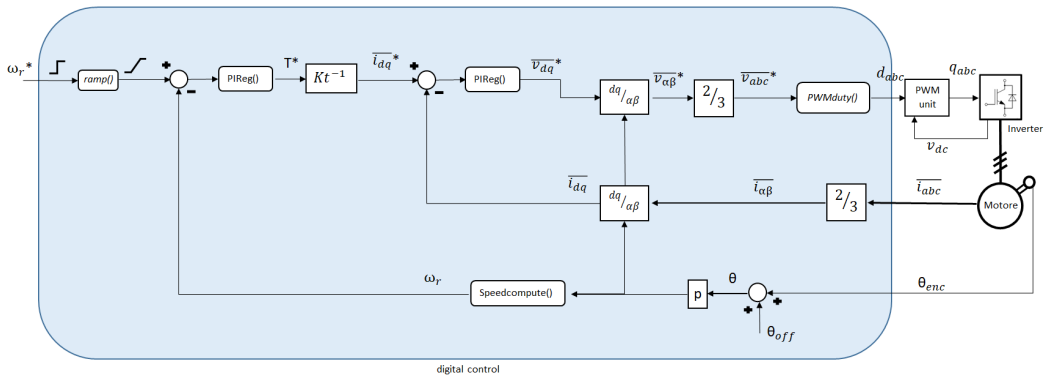


Figura 2.2: Controllo in anello chiuso di velocità

```
% PWM and sampling time
Tsw  = 100e-6;          % control task [s] = PWM task
% dc link voltage
Vdc = 310;
% mechanical quantities
encOffset = 1.0472; % rad
th0 = (pi/4)-pi;
J    = 1.0e-3;      % (not measured)

% motor data
Rs  = 1.5;           %
p   = 2;
Rfe = inf;
Fm = 0.261;     % PM flux (Vs)
Leq   = 0.0046; % average inductance (H)
Ld    = Leq;
Lq    = Leq + 2 * dLeq;

%Lookup table magnetic model
fd = linspace(0,Fm+Leq*20,40);
fq = linspace(-Lq*20,Lq*20,20);
ID_dato_fd_fq = (fd-Fm)/Ldpos;
ID_dato_fd_fq(fd<Fm) = (fd(fd<Fm)-Fm)/Ldneg;
IQ_dato_fd_fq = (fq)/Lq;

% Mechanical losses torque = Tmec_a0 + Tmec_a2 * w^2
P0 = 10;          % ventilation loss at nmax (W)
nmax = 6000;    % maximum operating speed (rpm)
Bm = 0;     % damping constant (Nm/(rad/sec))
Tc = 0.1;         % friction loss (Nm)
Kv = P0/(nmax*pi/30)^3; % ventilation loss coefficient (W/(rad/s)^3 = Nm/(rad/s)^2)
```

Codice 2.1: Inizializzazione Motore SPM. File: "init_SPM.m"

Inverter

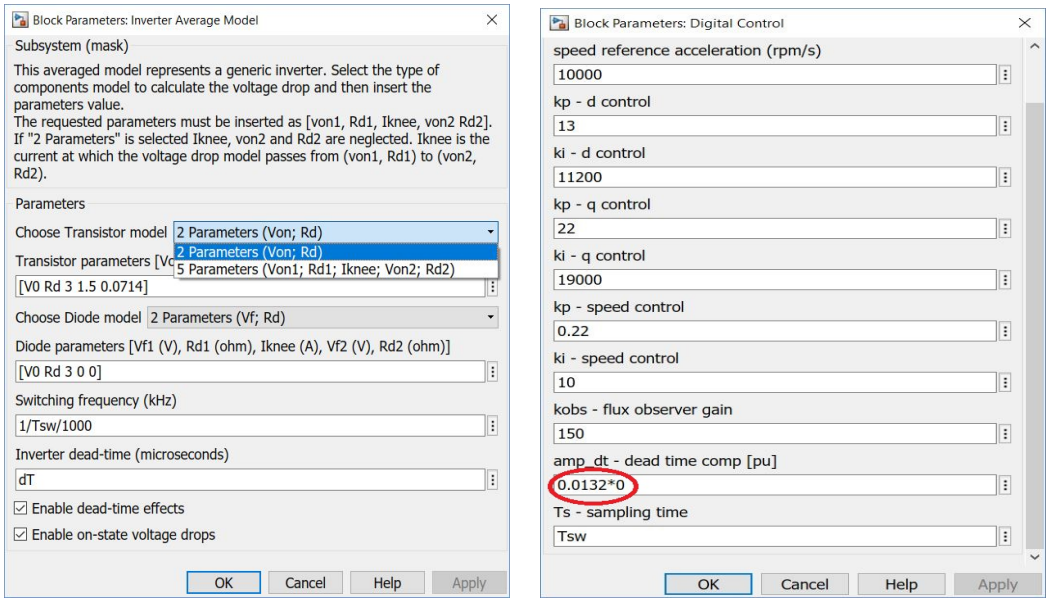
L'inverter è lo stesso in entrambi i modelli benchmark ed è basato su un modello ai valori medi scritto in linguaggio C che riceve in ingresso la tensione al DC Link, i duty cycle direttamente dal controllo e le correnti i_{abc} del motore. Come evidente in Figura 2.3a è possibile selezionare, per entrambi i componenti elettronici, un modello a due o cinque parametri. Il modello a 5 parametri inserisce una soglia di corrente (*Iknee*) al di sopra della quale cambiano i valori di Ron e Vf; il modello a 2 parametri invece utilizza solamente un valore di Ron e Vf (vedi il codice 2.2 per i valori numerici) per tutto il range di correnti ammesso. Siccome né in Simulink né in PLECS viene fornito un inverter circuitale con il comportamento a 5 parametri il modello didattico viene utilizzato con il modello a 2 parametri. Come già detto per il confronto con i modelli *AVG* e *Fisico* bisogna disattivare il dead-time e le cadute di tensione, per fare ciò è sufficiente aprire la maschera del blocco inverter e togliere la spunta vicina ai due parametri appena citati. Bisogna inoltre disattivare la compensazione degli effetti del deadtime all'interno del codice di controllo; è sufficiente impostare a zero il valore di *"amp_dt"* nella maschera del blocco *Digital Control* (Figura 2.3b), in modo tale che la compensazione del duty cycle sia nulla (vedi codice 2.3).

```
% Converter
V0 = 1;                 % power semiconductors ON treshold [V]
Rd = 0.1e-3;            % power semiconductors resistance [Ohm]
dT  = 1e-6;             % dead time [s]
```

Codice 2.2: Parametri Inverter

```
void DTComp(Xabc *isabc,Xabc *duty_abc, float amp_dt) {

if  ((*isabc).a>0)   (*duty_abc).a+=amp_dt;
else          (*duty_abc).a+=-amp_dt;

if  ((*isabc).b>0)   (*duty_abc).b+=amp_dt;
else               (*duty_abc).b+=-amp_dt;

if  ((*isabc).c>0) (*duty_abc).c+=amp_dt;
else               (*duty_abc).c+=-amp_dt;

}
```

Codice 2.3: Compensazione Deadtime



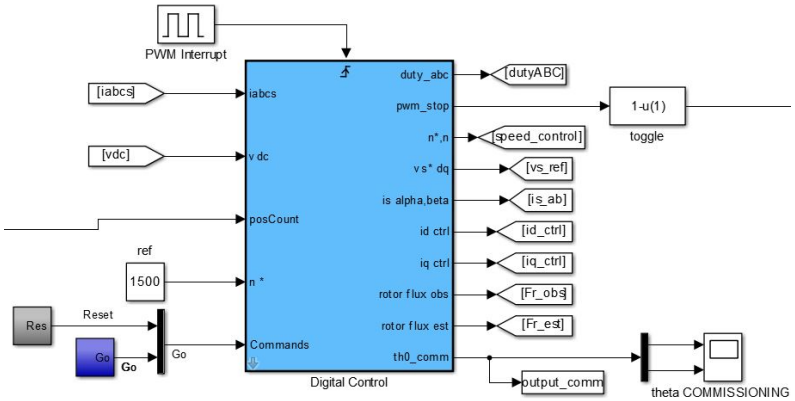
(a) Impostazioni inverter Benchmark (b) Compensazione dead-time

Figura 2.3: Impostazioni inverter e dead-time

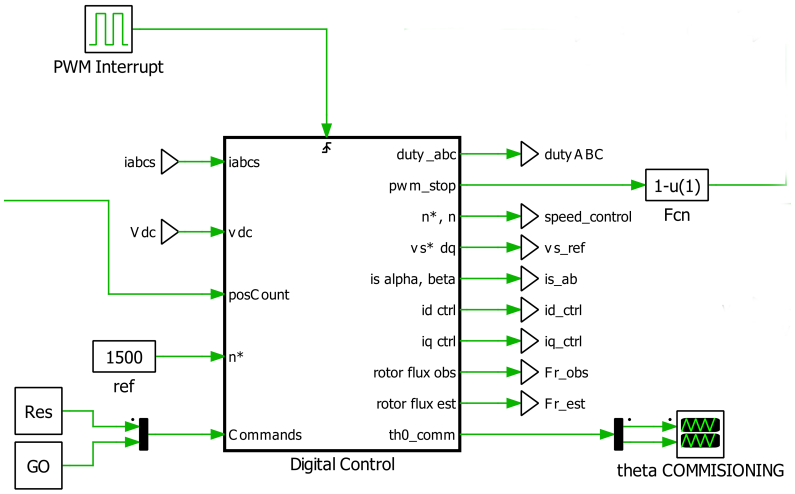
Controllo digitale

L'ultimo componente da presentare è denominato nel modello benchmark *"Digital Control*, il suo scopo è quello di simulare il comportamento di un microcontrollore e contiene quindi l'algoritmo di controllo scritto in linguaggio C. Questo è l'unico blocco che rimarrà sempre invariato, cambiando ovviamente il codice di controllo in base al tipo di motore, per ogni modello analizzato. Siccome a partire dal benchmark sono realizzati altri modelli, sia in Simulink che in PLECS, in questa sottosezione vengono presentati il blocco di controllo del modello benchmark (e di tutti gli altri modelli Simulink) e la sua componente duale realizzata in PLECS. Le impostazioni qui riportate sono riferite al motore SPM ma sono speculari nel caso dell'asincrono.

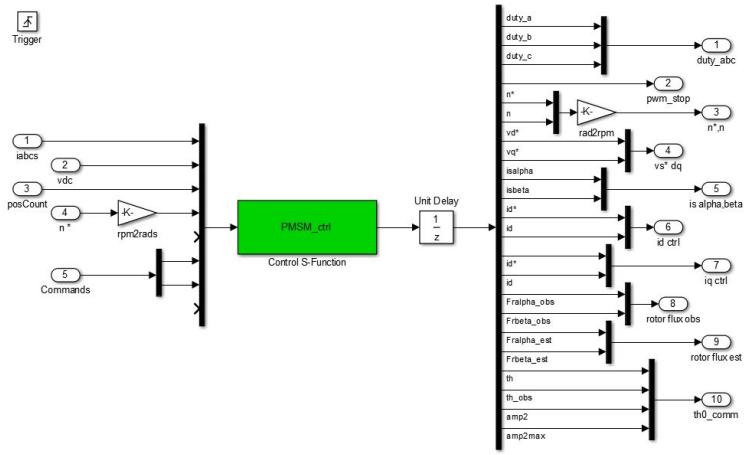
I blocchi "Reset", "Go" e "PWM Interrupt" di PLECS sono esattamente speculare ai loro corrispettivi Simulink e non richiedono approfondimento data la semplicità realizzativa. Il blocco "Digital Control" è stato strutturato in modo analogo a quello di Simulink usando il blocco "C-Script" duale alla "S-Function" (Figura 2.4c).



(a) Digital Control Simulink

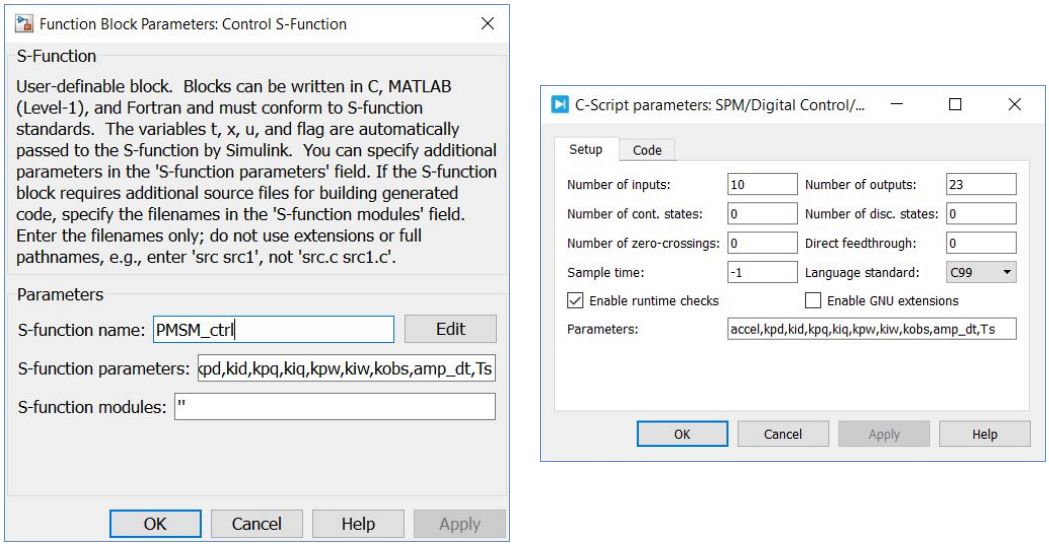


(b) Digital Control PLECS



(c) Digital Control S-Function

Figura 2.4: Digital Control



(a) Maschera Digital Control Simulink (b) Maschera Digital Control PLECS

Figura 2.5: Parametri Script

La pagina di impostazione del blocco C-Script è simile a quella della S-Function, viene inoltre richiesto il numero di input/output e un sample time. Quest'ultimo è impostato a -1 e corrisponde ad un sample time "inherited" ossia il cui valore viene "ereditato" dai blocchi a cui è connesso il C-Script. I parametri presenti nelle due pagine di impostazione corrispondo alle variabili il cui valore numerico può essere impostato direttamente dalla maschera senza necessità di intervenire sul codice.

La differenza tra i due programmi è netta, almeno a livello strutturale, nel modo di implementare il codice C. In Simulink la S-Function richiama un file .c (in questo esempio il file: *"PMSM_ ctrl.c"*) che racchiude al suo interno la definizione di input/output, la lettura delle variabili passate tramite maschera e ovviamente il richiamo della routine di controllo motore. In PLECS queste funzioni devono essere scritte nell'apposita function selezionabile tramite il pannello "Code" del blocco C-Script (Figura 2.5b). Le varie functions sono descritte all'interno dell'help di PLECS [6] e sono:

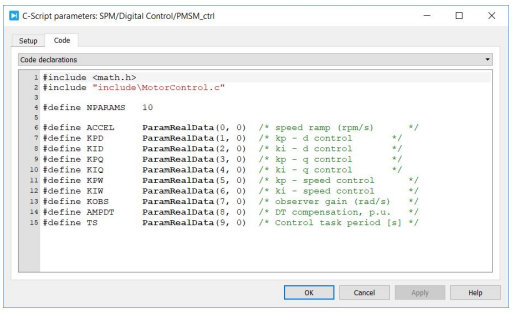
- Code declaration;
- Start function code;
- Output function code;
- Update function code;
- Derivative function code;
- Terminate function code;

Il codice contenuto nella S-function del modello benchmark è stato quindi suddiviso come segue:

- **Code declaration** In questa sezione vengono inclusi i file e le librerie necessarie al funzionamento del codice e le definizioni dei parametri passati da maschera tramite l'apposita macro *ParamRealData(int i, j)*.

```
#include "simstruc.h"
#include <math.h>

#include "include\MotorControl.c"

#define NINPUTS     10
#define NOUTPUTS    23

#define ACCEL       (mxGetPr(ssGetSFcnParam(S, 0))[0])  /*  speed ramp (rpm/s)      */
#define KPD         (mxGetPr(ssGetSFcnParam(S, 1))[0])  /*  kp - d control          */
#define KID         (mxGetPr(ssGetSFcnParam(S, 2))[0])  /*  ki - d control          */
#define KPQ         (mxGetPr(ssGetSFcnParam(S, 3))[0])  /*  kp - q control          */
#define KIQ         (mxGetPr(ssGetSFcnParam(S, 4))[0])  /*  ki - q control          */
#define KPW         (mxGetPr(ssGetSFcnParam(S, 5))[0])  /*  kp - speed control      */
#define KIW         (mxGetPr(ssGetSFcnParam(S, 6))[0])  /*  ki - speed control      */
#define KOBS        (mxGetPr(ssGetSFcnParam(S, 7))[0])  /*  observer gain (rad/s)   */
#define AMPDT       (mxGetPr(ssGetSFcnParam(S, 8))[0])  /*  DT compensation, p.u.   */
#define TS          (mxGetPr(ssGetSFcnParam(S, 9))[0])  /*  Control task period [s] */

#define NPARAMS     10
```

(a) Definizione parametri Simulink



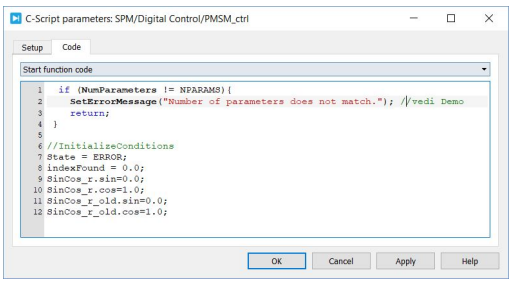
(b) Definizione parametri PLECS

Figura 2.6: Code Declaration

- **Start Function Code** Nella start function vengono riportati i parametri da inizializzare all'avvio della simulazione ossia quelli che in Simulink vengono impostati nelle function *mdlInitializeSizes* e *mdlInitializeConditions*.

```
static void mdlInitializeSizes(SimStruct *S) {
    ssSetNumSFcnParams(S, NPARAMS);  /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return; /* Parameter mismatch will be reported by Simulink */
    }

static void mdlInitializeConditions(SimStruct *S)
/*this routine is called at the start of simulation */
{

    State = ERROR;
    indexFound = 0.;
    SinCos_r.sin=0.0;
    SinCos_r.cos=1.0;
    SinCos_r_old.sin=0.0;
    SinCos_r_old.cos=1.0;

}
```

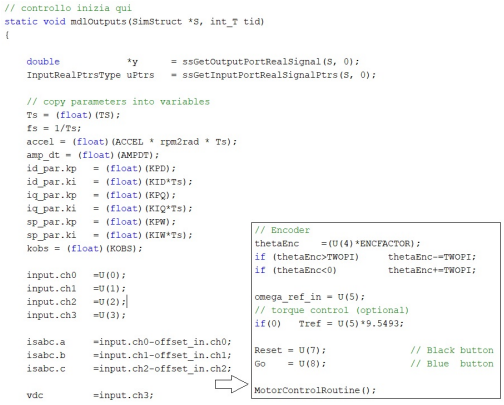
(a) Inizializzazione parametri Simulink



(b) Inizializzazione parametri PLECS

Figura 2.7: Start Function Code

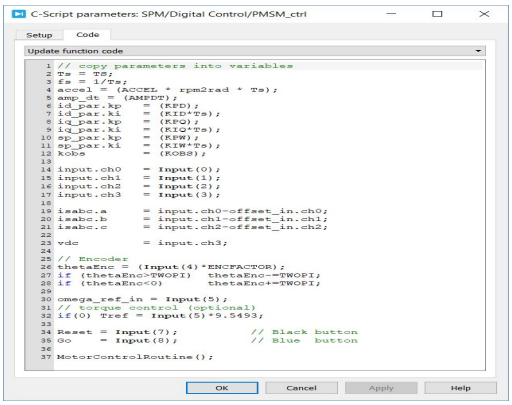
- **Update Function Code** Il codice riportato in questa function serve per scrivere i valori dei parametri passati tramite maschera e gli input provenienti dagli altri blocchi (macro *Input()*) in apposite variabili, viene infine richiamata la *motor control routine* ossia la parte di codice contenente l'algoritmo di controllo del motore.
- **Output Function Code** Infine vengono definiti i valori di output tramite la macro Output():



```
// controllo inizia qui
static void mdlOutputs(SimStruct *S, int_T tid)
{

    double            *y      = ssGetOutputPortRealSignal(S, 0);
    InputRealPtrsType uPtrs   = ssGetInputPortRealSignalPtrs(S, 0);

    // copy parameters into variables
    Ts = (float)(TS);
    fs = 1/Ts;
    accel = (float)(ACCEL * rpm2rad * Ts);
    amp_dt = (float)(AMPDT);
    id_par.kp    = (float)(KPD);
    id_par.ki    = (float)(KID*Ts);
    iq_par.kp    = (float)(KPQ);
    iq_par.ki    = (float)(KIQ*Ts);
    sp_par.kp    = (float)(KPW);
    sp_par.ki    = (float)(KIW*Ts);
    kobs = (float)(KOBS);

    input.ch0   =U(0);
    input.ch1   =U(1);
    input.ch2   =U(2);
    input.ch3   =U(3);

    isabc.a     =input.ch0-offset_in.ch0;
    isabc.b     =input.ch1-offset_in.ch1;
    isabc.c     =input.ch2-offset_in.ch2;

    vdc         =input.ch3;
```

```
// Encoder
thetaEnc    =(U(4)*ENCFACTOR);
if (thetaEnc>TWOPI)     thetaEnc-=TWOPI;
if (thetaEnc<0)         thetaEnc+=TWOPI;

omega_ref_in = U(5);
// torque control (optional)
if(0    Tref = U(5)*9.5493;

Reset = U(7);               // Black button
Go    = U(8);               // Blue  button

MotorControlRoutine();
```

(a) Update parametri Simulink

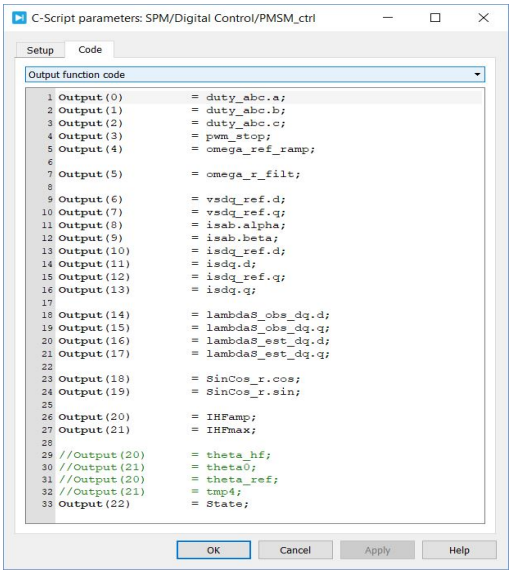


(b) Update parametri PLECS

Figura 2.8: Update Function

```
y[0]  = duty_abc.a;
y[1]  = duty_abc.b;
y[2]  = duty_abc.c;
y[3]  = pwm_stop;
y[4]  = omega_ref_ramp;

y[5]  = omega_r_filt;

y[6]  = vsdq_ref.d;
y[7]  = vsdq_ref.q;
y[8]  = isab.alpha;
y[9]  = isab.beta;
y[10] = isdq_ref.d;
y[11] = isdq.d;
y[12] = isdq_ref.q;
y[13] = isdq.q;

y[14] = lambdaS_obs_dq.d;
y[15] = lambdaS_obs_dq.q;
y[16] = lambdaS_est_dq.d;
y[17] = lambdaS_est_dq.q;

y[18] = SinCos_r.cos;
y[19] = SinCos_r.sin;

y[20] = IHFamp;
y[21] = IHFmax;

//y[20] = theta_hf;
//y[21] = theta0;
//y[20] = theta_ref;
//y[21] = tmp4;
y[22] = State;
```

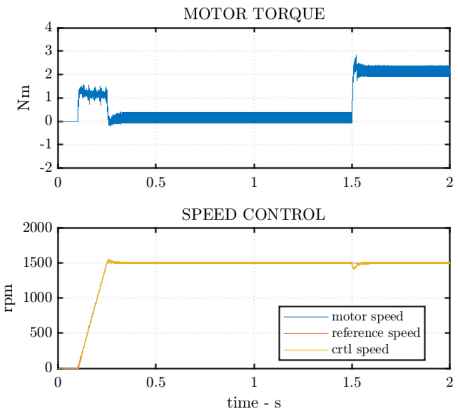
(a) Output Simulink



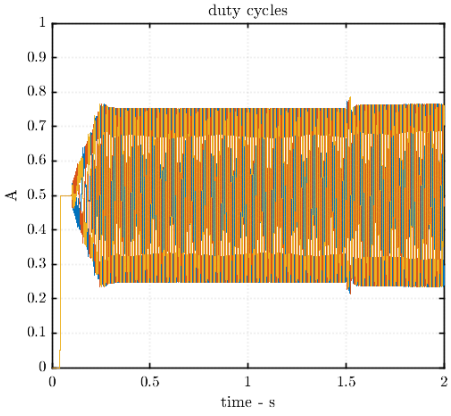
(b) Output PLECS

Figura 2.9: Code Declaration

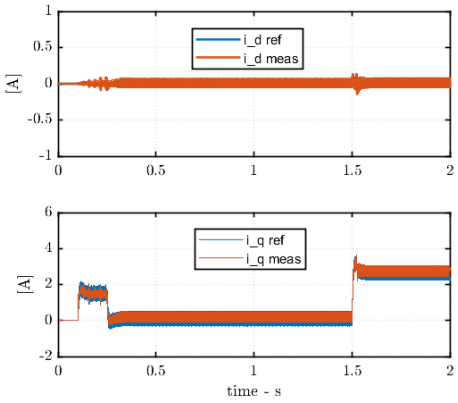
Risultati Simulazione



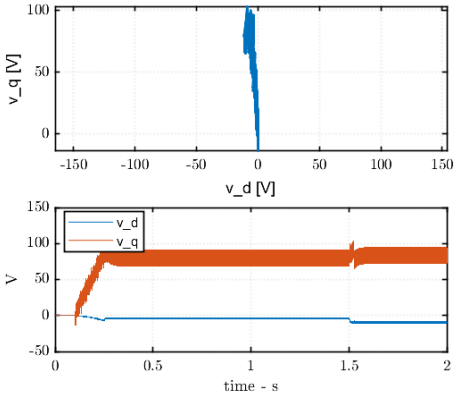
(a) Velocità e coppia



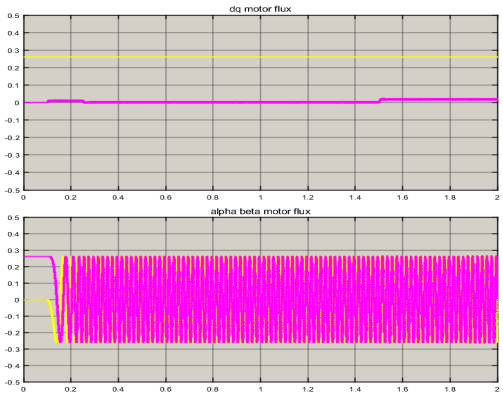
(b) Duty Cicle



(c) Correnti assi dq



(d) Tensioni assi dq



(e) Flussi assi dq e $\alpha\beta$

Figura 2.10: Modello Benchmark SPM

2.1.2 Modello Benchmark IM

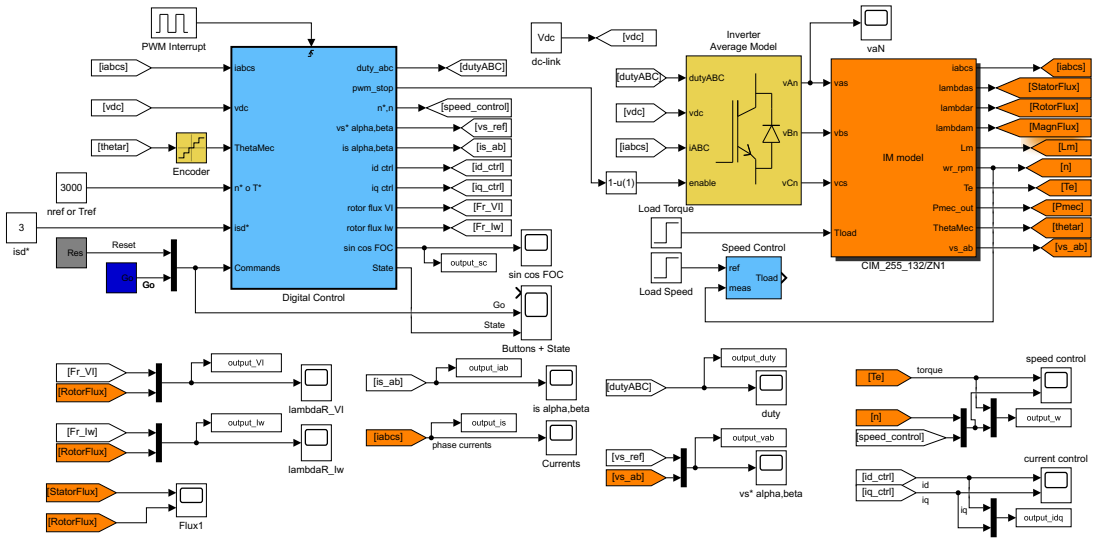


Figura 2.11: Modello benchmark IM

Il modello benchmark del motore asincrono si basa sul motore *C.e.SET CIM 2/55 132/ZN* , su questo motore sono state effettuate delle prove a vuoto e a rotore bloccato, i dati ottenuti da questi test (come ad esempio le induttanze di statore e rotore) sono stati raccolti in un file .mat denominato *Ceset700W_ TestData.mat* e caricato dal software quando viene lanciato il file di inizializzazione il cui codice è riportato al 2.4 [14]. L'inverter ed il blocco di controllo sono gli stessi presentati per il motore SPM.

Per il motore asincrono è stato implementato il controllo scalare V/Hz riportato in Figura 2.12 [3, p. 339].

```
% PWM
Tsw  = 500e-6;          % control task [s] = PWM task
%dc link voltage
Vdc = 310;              % dc link voltage

% MOTOR DATA
% Motor name:   CESET CIM2/55-132/ZN
% Ratings:      700W, 220 V line @ 83.3 Hz, 3A, 300 Hz max
%               1.8 Nm, 4820 rpm, sn = 3.6%
%          No-load test run at 100Hz, Locked rotor test run at 20 Hz
```

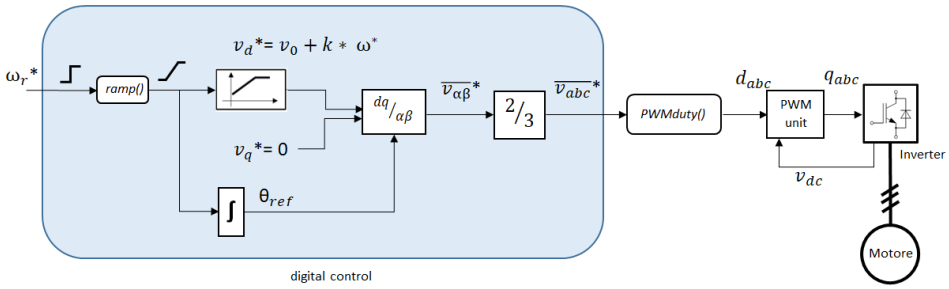
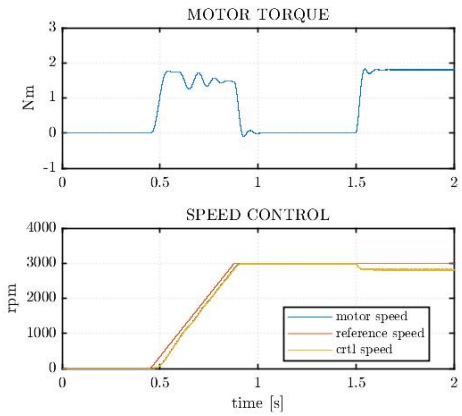


Figura 2.12: Controllo scalare V/Hz

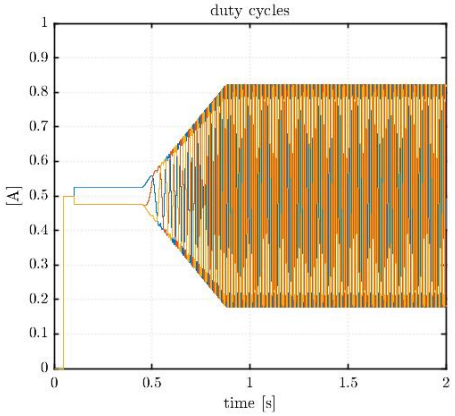
```
load Ceset700W_TestData     % no load and short circuit test data

Vnom = 220;     % (V rms - line)
Inom = 3;       % (A rms)
fnom = 83.3;    % (Hz)

Fnom = Vnom*sqrt(2/3)/(2*pi*fnom);   % k factor (Vs)

% other data
Rs = 3.25;      % stator resistance (Ohm)
Rr = 1.64;      % rotor resistance (Ohm)
Rfe = 1000;     % Iron loss resistance (Ohm)
J = 0.002;      % total inertia (kgm^2)
flag_sat = 1;   % magnetic saturation [if flag_sat == 0 Sat.= OFF];
flag_pfe = 0;   % iron loss [if flag_pfe == 0 Rfe = inf];

% Mechanical loss Nm = Tmec_a0 + Tmec_a2 * w^2
P0 = 10;        % ventilation loss at nmax (W)
nmax = 6000;    % maximum operating speed (rpm)
Bm = 0;         % damping constant (Nm/(rad/sec))
Tc = eps;       % friction loss (Nm) [eps=2.2204e-16]
Kv = P0/(nmax*pi/30)^3; % ventilation loss coeff. (W/(rad/s)^3 = Nm/(rad/s)^2)
```

Codice 2.4: Inizializzazione Motore IM. File: "init_sim.m"

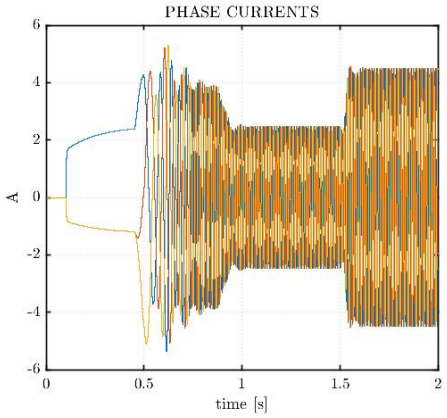
Risultati simulazione benchmark



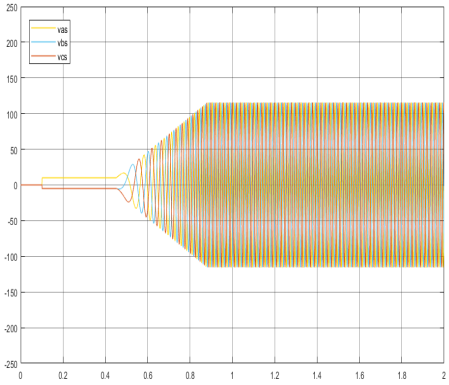
(a) Velocità e coppia



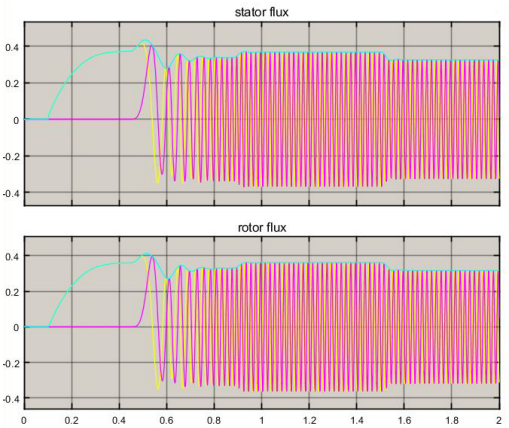
(b) Duty Cicle



(c) Correnti assi abc



(d) Tensioni assi $\alpha\beta$



(e) Flussi assi dq e $\alpha\beta$

Figura 2.13: Modello Benchmark IM

2.2 Modelli circuitali

I modelli che sono presentati in questa sezione sono basati su componenti circuitali di inverter e motore provenienti dalle librerie Simulink e PLECS. I blocchi fisici dei motori sono presentati nelle pagine seguenti mentre per quanto riguarda l'inverter sono presentati tre diversi modelli:

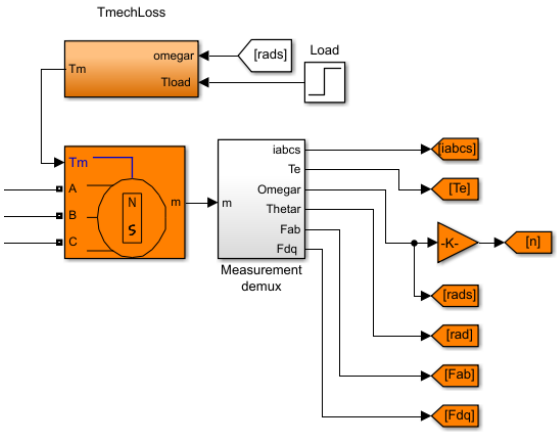
- Inverter Average
- Inverter Logico
- Inverter Fisico

2.2.1 Modello fisico SPM

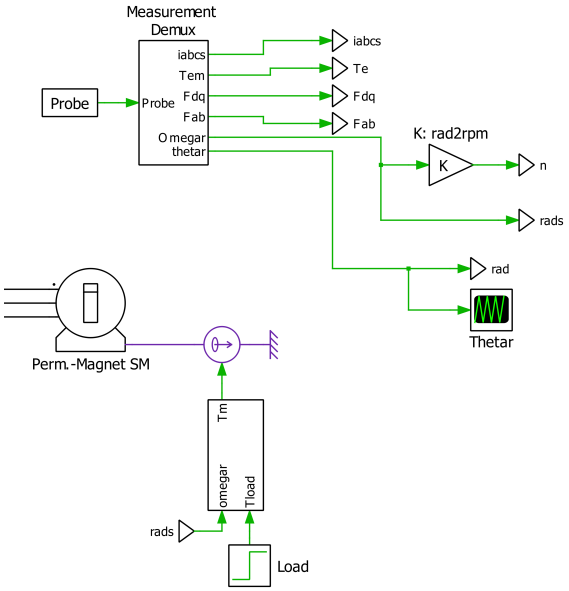
In Figura 2.14 sono riportati i blocchi utilizzati per il motore SPM, entrambi gli ambienti forniscono un modello di motore sincrono a magneti permanenti denominato *"Permanet-Magnet Synchronous Machine"*. In Figura 2.15 vengono riportate le pagine di configurazione dei motori nei due programmi. I dati utilizzati per il motore provengono dal file *init_SPM.m* del modello benchmark riportati nel paragrafo dedicato al SPM nella sezione 2.1. Il controllo del motore SPM richiede l'utilizzo di un encoder assoluto, i modelli Simulink utilizzano lo stesso encoder utilizzato nel modello benchmark, in PLECS viene replicata la stessa struttura. In Figura 2.16 vengono riportati i due schemi, l'unica differenza è data dal C-Script nel modello PLECS (fig. 2.16b), quest'ultimo serve per saturare tra zero e 2π il valore riportato nei grafici e utilizza il codice 2.5 cosa che in Simulink viene effettuata nel momento in cui vengono rappresentati i risultati.

```
posMot = InputSignal(0,0);
posEnc = InputSignal(0,1);
if (posMot<0) posMot+=(2*pi);
if (posMot>(2*pi)) posMot-=(2*pi);
if (posEnc<0) posEnc+=(2*pi);
if (posEnc>(2*pi)) posEnc-=(2*pi);
```

Codice 2.5: C-Script Encoder

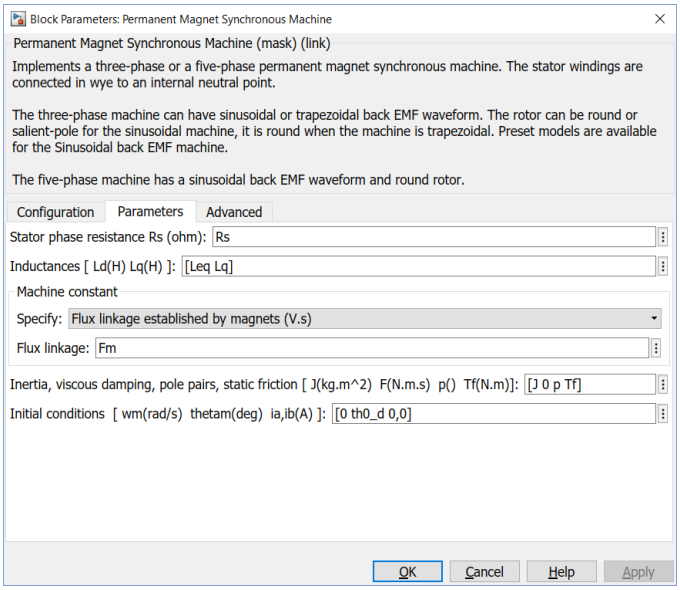


(a) SPM - Simulink

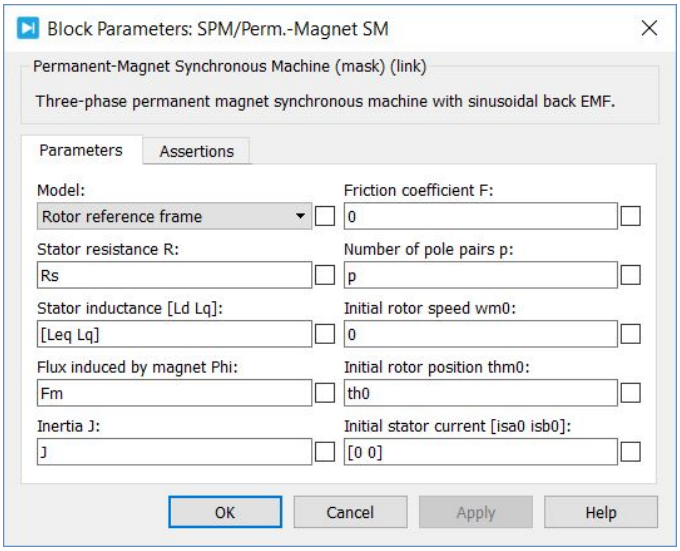


(b) SPM - Plecs

Figura 2.14: Modello motore SPM

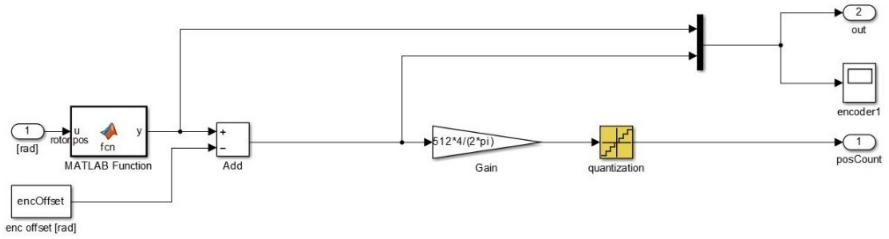


(a) Impostazioni Motore - Simulink

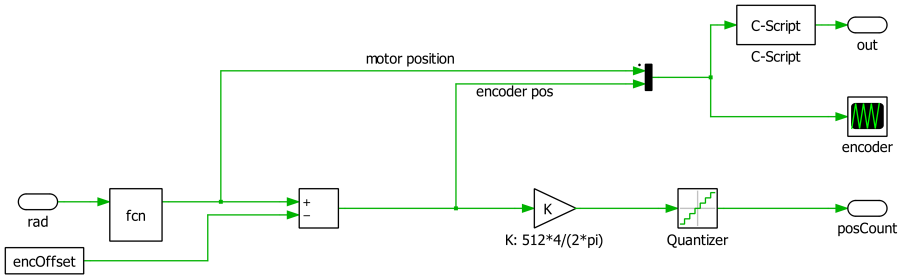


(b) Impostazioni Motore - Plecs

Figura 2.15: Impostazioni SPM



(a) Encoder Assoluto Simulink



(b) Encoder Assoluto PLECS

Figura 2.16: Schema encoder assoluto

Perdite Meccaniche

Per entrambi i motori la simulazione benchmark prevede delle perdite meccaniche dovute alla presenza di attrito. La coppia di attrito è tipicamente caratterizzata da tre componenti: una coppia dovuta all'attrito statico, una dovuta a quello viscoso e proporzionale alla velocità ed una definita di ventilazione funzione della velocità al quadrato. Nel modello benchmark l'effetto dell'attrito viene considerato all'interno delle equazioni meccaniche del motore e non tiene conto della componente viscosa, la coppia d'attrito per questi modelli è quindi descrivibile con la seguente equazione:

$$T_{loss} = T_c + K_v \cdot \omega_r^2 \qquad [Nm] \tag{2.1}$$

- T_c = coppia attrito statico $[Nm]$
- K_v = coefficiente perdite ventilazione $[Nm/(rad/s^2)]$

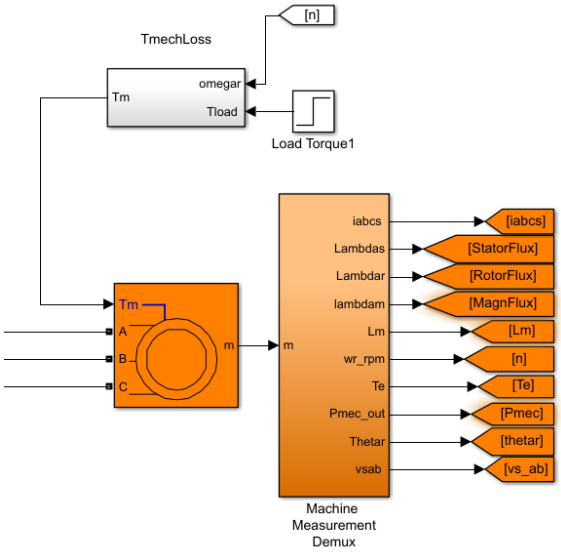
I modelli di motore presentati nelle sottosezioni precedenti non prevedono invece questo tipo di perdita meccanica. Nei modelli qui presentati ho quindi introdotto il subsystem denominato *"TmechLoss"*, sia in ambiente MATLAB che PLECS, che riceve in ingresso i valori di ω e della coppia di carico (T_{load}) e restituisce in uscita la coppia meccanica all'albero T_m secondo l'equazione 2.2.

$$T_m = T_{load} + (T_c + K_v \cdot \omega_r^2) \cdot sign(\omega_r) \qquad [Nm] \tag{2.2}$$

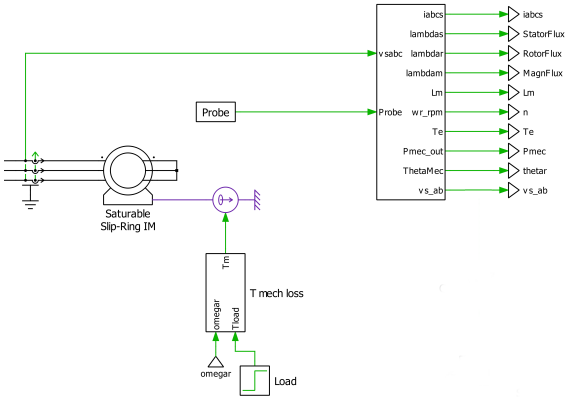
Nel caso del motore SPM in ambiente MATLAB viene impostato a zero il valore di T_c all'interno del subsystem in quanto l'attrito statico è già presente nel modello di motore (vedi Figura 2.15a).

La T_m viene fornita come input meccanico per il motore e considerata come coppia resistente, in Simulink può essere inviata direttamente alla blocco motore mentre in PLECS è necessario introdurre la sorgente di coppia *"Torque (controlled)"* della libreria meccanica rotazionale.

2.2.2 Modello fisico IM



(a) IM - Simulink



(b) IM - Plecs

Figura 2.17: Modello motore asincrono

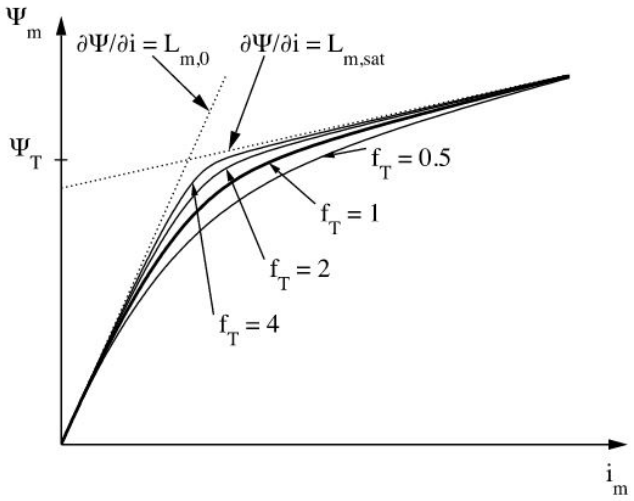
In Figura 2.17 sono riportati i blocchi utilizzati nei due ambienti per l'IM; in Simulink è stato selezionato il motore *Asynchronous Machine SI Unit* mentre in PLECS il componente *Induction Machine (Slip Ring, Saturable)*. In Figura 2.19 vengono riportate le configurazione dei blocchi motore nei due programmi; in entrambi i casi è prevista la saturazione del motore, il modo in cui questa viene modellata è però nettamente diverso e richiede quindi un approfondimento.

Nel modello Simulink la saturazione è modellata a partire dai valori di corrente e tensione ottenuti durante la prova a vuoto del motore, come già detto in sezione 2.1 sono stati forniti i valori delle prove a vuoto[14], i valori di corrente e tensione sono stati quindi inseriti nell'apposita casella come riportato in Figura 2.19b. Per quanto riguarda PLECS la pagina di impostazione del motore è riportata in Figura 2.19c. Rispetto alle impostazioni richieste da Simulink troviamo alcuni parametri aggiuntivi: *Lm0*, *Lmsat*, *PsiT* e *fT*. Questi ultimi sono necessari per introdurre l'effetto della saturazione che viene in questo caso modellata a partire dalla curva ψ_m - i_m riportata in Figura 2.18a.

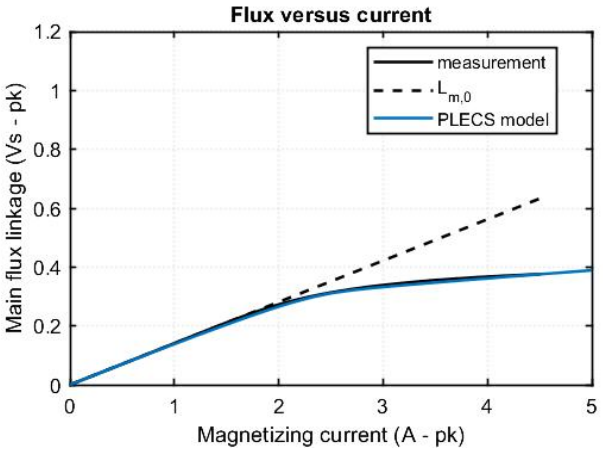
I parametri corretti da inserire nel modello PLECS sono stati ottenuti confrontando tramite MATLAB la curva ψ_m - i_m ottenuta a partire dai dati raccolti durante le prove sul *C.e.SET CIM 2/55 132/ZN* con la curva ottenuta con le equazioni utilizzate da PLECS per modellare la saturazione e contenute nello script interno al modello di motore [4], stimando dei valori di partenza dai dati forniti dalle prove ho corretto iterativamente i valori delle quattro variabili citate fino a sovrapporre esattamente la curva del modello PLECS a quella reale come riportato in Figura 2.18b. I valori finali sono riassunti in tabella 2.1.

Tabella 2.1: Parametri modello IM

Parametro	*Valore*	[*Unità di misura*]
Lm0	0.1483	[H]
Lm_{sat}	0.026	[H]
PsiT	0.32	[Vs]
fT	5	-

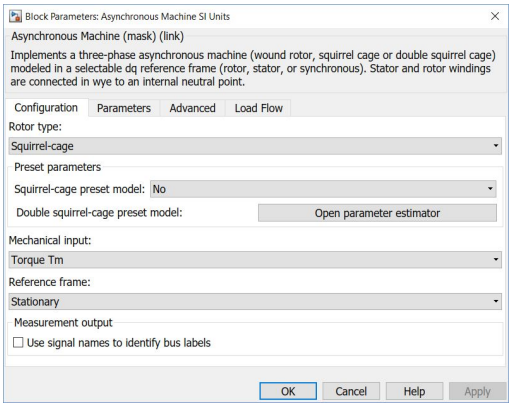


(a) Curva ψ_m - i_m.
Immagine tratta da [6]

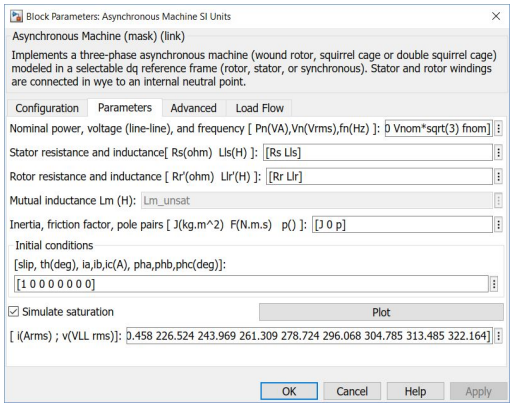


(b) Confronto curve ψ_m - i_m

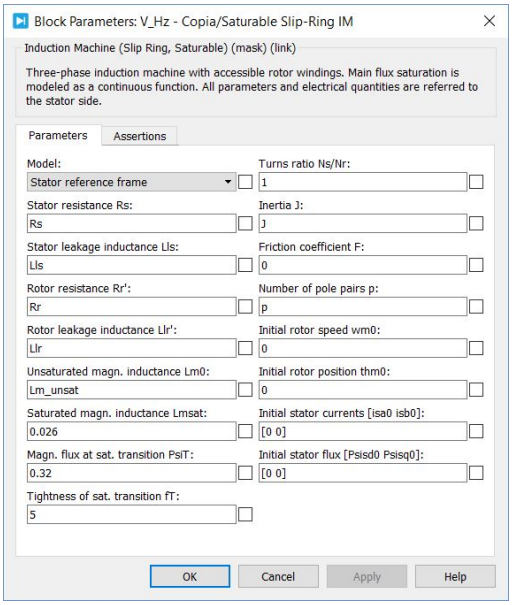
Figura 2.18: Curve ψ_m - i_m



(a) Impostazioni Motore - Simulink. Configurazione



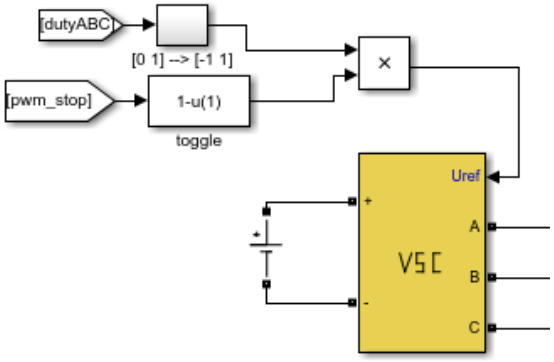
(b) Impostazioni Motore - Simulink. Parametri e saturazione



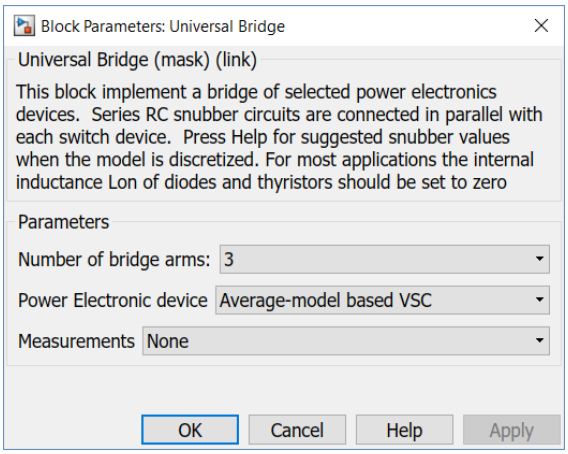
(c) Impostazioni Motore - Plecs

Figura 2.19: Impostazioni IM

2.2.3 Modello Average Inverter



(a) Connessioni inverter AVG



(b) Impostazioni Universal Bridge

Figura 2.20: Inverter Average Simulink

Questa sotto sezione presenta il modello di inverter Average (AVG), questo tipo di modello utilizza un valore mediato del segnale di switch calcolato o su un "ciclo di switch" dell'inverter o sovra-campionando il segnale a intervalli regolari. Citando un esempio degli autori di [1] se durante uno step di simulazione non avvengono eventi di switching il segnale inviato all'inverter rimane 1 o 0 mentre se per esempio avviene un evento a metà step il valore mediato sarà 0.5. Questo concetto viene ripreso più avanti e nello specifico per il modello utilizzato in PLECS.

Per quanto riguarda il modello Simulink è stato utilizzato il blocco *Universal Bridge* presente in libreria, le impostazioni possibili, riportate in Figura 2.20b sono limitate per il modello AVG (numero di gambe ed eventuali valori da riportare fuori maschera per la misura). Il modello utilizza il segnale di riferimento per generare un valore di tensione medio ai terminali ABC del convertitore.

La libreria PLECS non fornisce un blocco inverter direttamente utilizzabile in modalità average ma quest'ultimo può essere realizzato usando 3 blocchi *IGBT Half Bridge*. I quali presentano due modalità di utilizzo:

- *Switched*: I semiconduttori sono modellati con degli interruttori ideali. Il singolo IGBT è controllato con i segnali logici di gate.
- *Sub-cycle average*: In questa modalità il modulo è realizzato usando delle sorgenti di tensione e corrente funzione dei comandi di switch. Gli input di controllo devono essere forniti sotto forma di valori compresi tra 0 ed 1 e possono essere o i comandi logici istantanei oppure i duty cycles dei singolo IGBT [6].

Se un segnale di switch viene campionato ad intervalli regolari, corrispondenti al time step della simulazione, il duty del segnale campionato è tipicamente affetto da un errore quando la transizione del segnale avviene tra due time steps. La modalità Sub-cycle average" (SCA) consente di ridurre questo errore tramite un sovra-campionamento del segnale di switch, come evidente in Figura 2.21, con intervalli di tempo nettamente minori allo step size della simulazione. Inoltre, la SCA non necessita, contrariamente alla modalità *Switched*, di matrici di stato per le varie combinazioni di switch che devono essere calcolate a priori e la cui lettura tende a rallentare la simulazione [1].

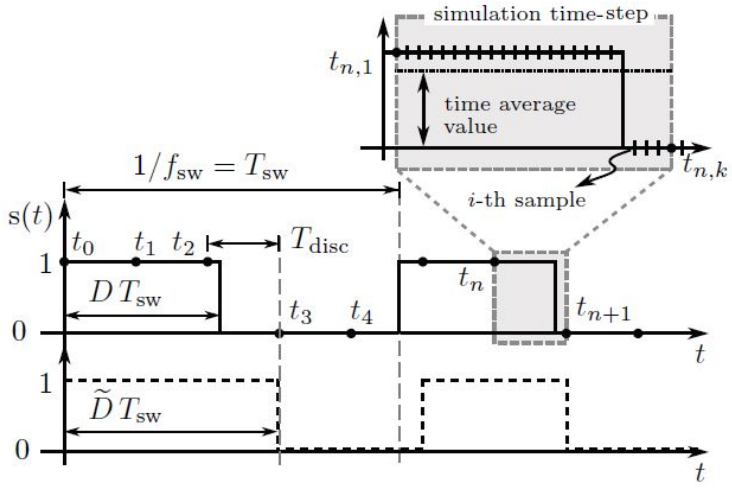
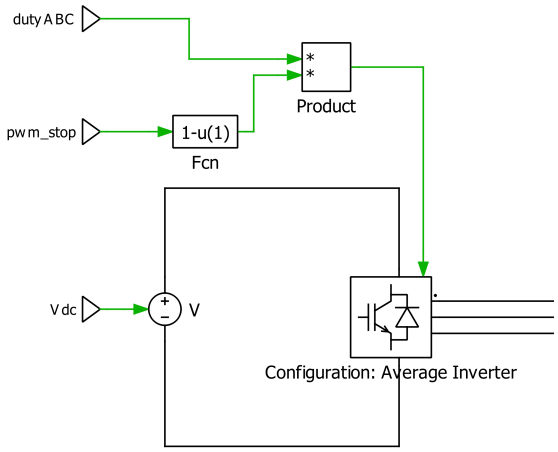


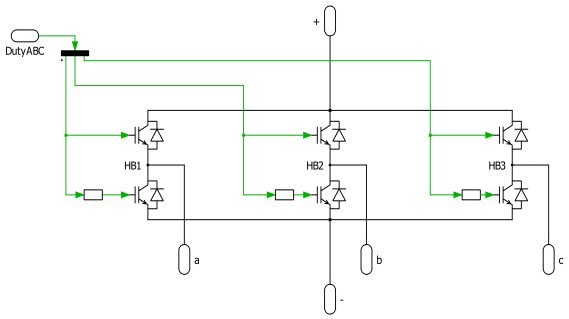
Fig. 1: Comparison between original switching signal and sampled switching signal. The points t_i denote a simulation step. An expanded view is provided for one simulation time-step including over-sampled data points $t_{n,i}$.

h]

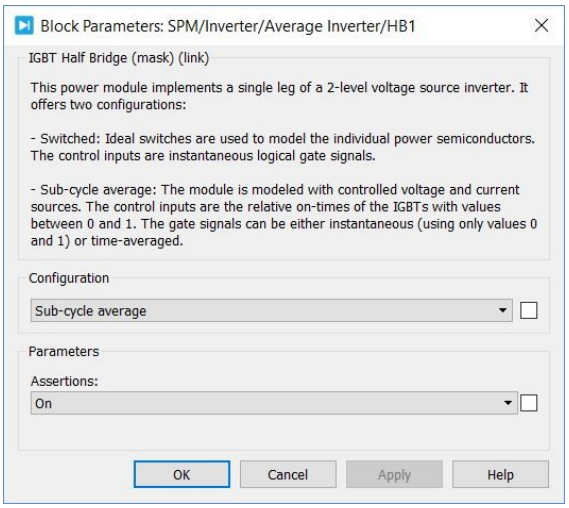
Figura 2.21: Confronto tra segnale originale e campionato. Figura tratta da [1]



(a) Average Inverter



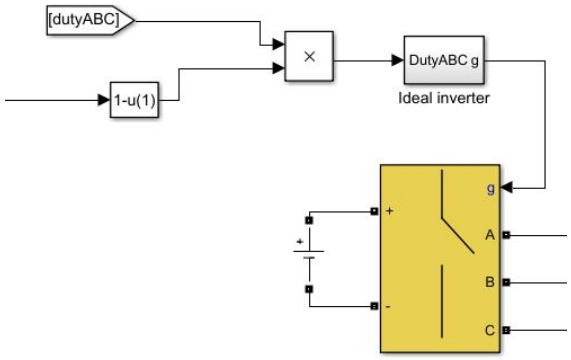
(b) Average Inverter schema



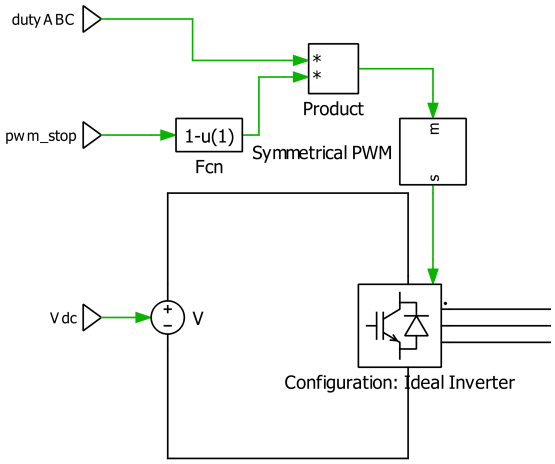
(c) Impostazioni Half Bridge

Figura 2.22: Inverter Average PLECS

2.2.4 Modello Logico Inverter



(a) Inverter Simulink



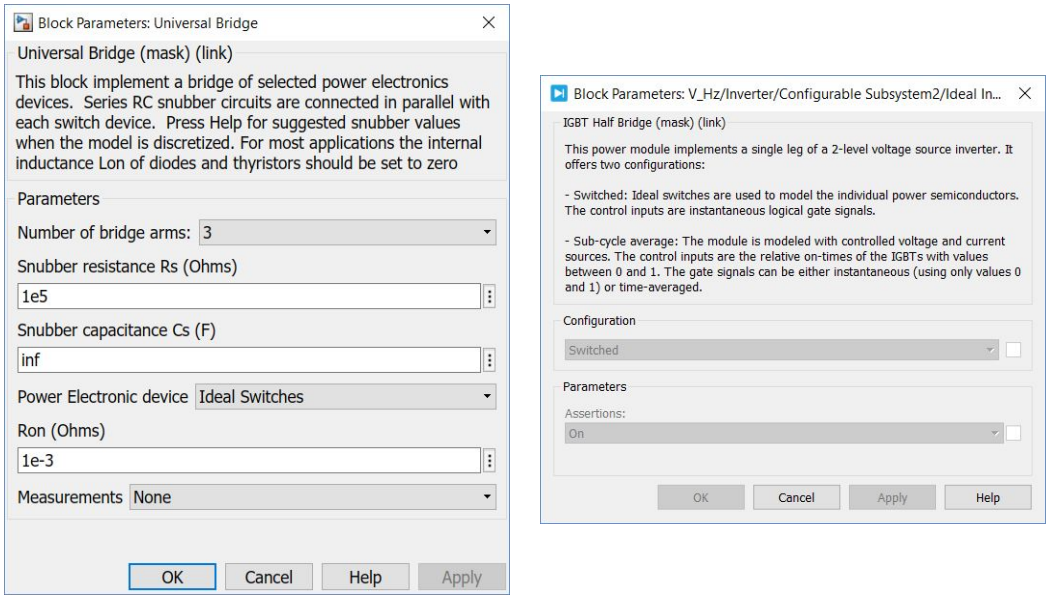
(b) Inverter PLECS

Figura 2.23: Inverter ideale

I modelli di tipo logico utilizzano un inverter basato su interruttori ideali. Sia il blocco inverter di Simulink che quello di PLECS usati nei modelli average sono impostabili in modalità "ideale" anche se con alcune differenze. Come è evidente dalla Figura 2.24a l'inverter di Simulink presenta uno snubber resistivo e una resistenza di on il cui valore deve essere non nullo per il corretto funzionamento della blocco. I valori utilizzati (e riportati in Figura) sono quelli suggeriti nell'Help.

Il blocco inverter di PLECS invece, impostato in modalità *"Switched"*, non ha valori impostabili ed utilizza degli interruttori ideali. Nel momento in cui si passa da un inverter average a quello ideale non è più possibile utilizzare direttamente i duty cycle come segnali di comando ma bisogna ricavare degli opportuni valori PWM.

Questo risulta molto semplice in PLECS in cui viene fornito il blocco *"Symmetrical PWM"* riportato in Figura 2.24b. Quest'ultimo si comporta come un modulatore PWM confrontando il segnale in ingresso (Duty trifase) con una portante triangolare simmetrica della quale possiamo selezionare tipo, frequenza ed eventuale offset, inoltre è possibile decidere il range dei valori di input e output (fig. 2.25). In Simulink è invece necessario modellare un subsystem appropriato il cui schema è riportato in Figura 2.26b.



(a) Impostazioni inverter Simulink (b) Impostazioni inverter PLECS

Figura 2.24: Impostazioni inverter ideale

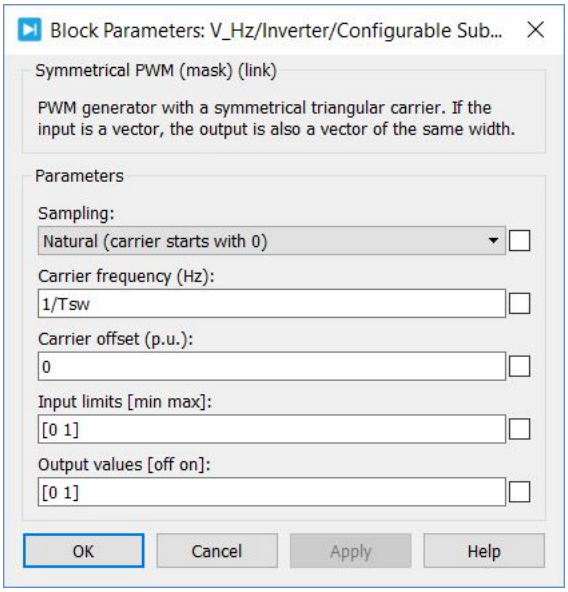
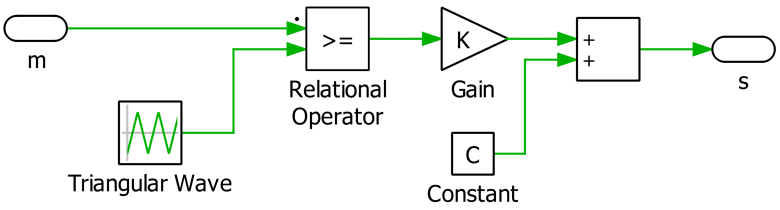
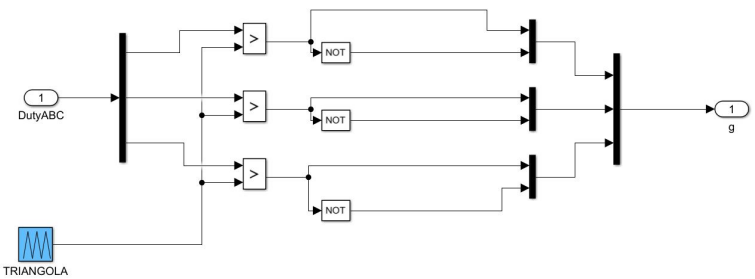


Figura 2.25: Impostazioni symmetrical PWM



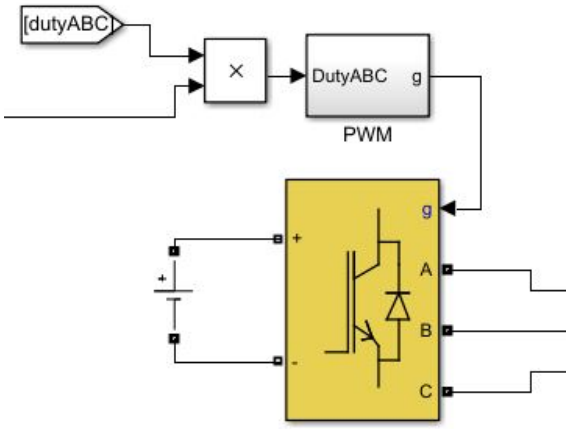
(a) Symmetrical PWM



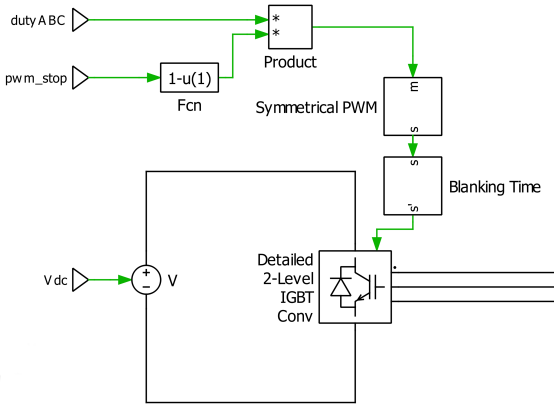
(b) Comandi di gamba Simulink

Figura 2.26: Generazione comandi di gamba

2.2.5 Modello Fisico Inverter



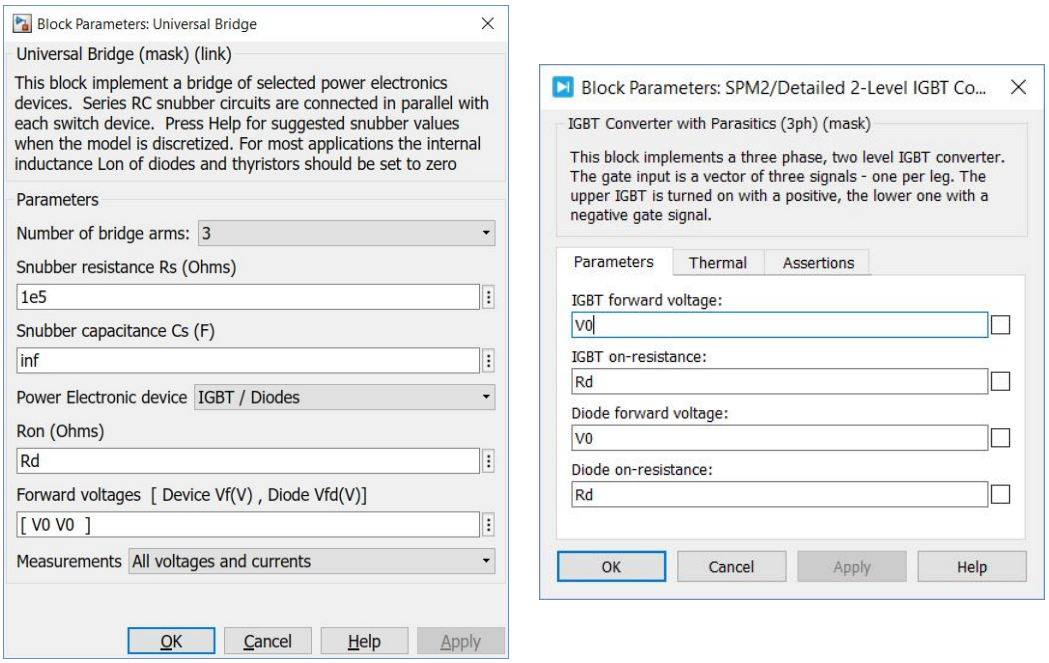
(a) Inverter modello fisico Simulik



(b) Inverter modello fisico PLECS

Figura 2.27: Modelli inverter fisico

L'ultimo modello presentato viene chiamato *"Fisico"*, a differenza dei modelli precedenti include anche il deadtime e utilizza un inverter con parametri "reali". Il modello di benchmark è sempre il modello didattico ma, contrariamente a quanto fatto nel caso dei modelli *Average* e *Logico*, sono stati attivati deadtime e cadute di tensione delle componenti elettroniche. Inoltre è stata riattivata la compensazione del deadtime nel codice di controllo. Il blocco inverter in Simulink è lo stesso



(a) Impostazioni inverter fisico Simulink (b) Impostazioni inverter fisico PLECS

Figura 2.28: Impostazioni inverter

utilizzato nei precedenti modelli; tra le varie impostazioni possibili per questo blocco è stata selezionata la modalità *"IGBT/Diodes"*. I valori impostabili sono la resistenza di On (Ron) dei componenti elettronici e le tensioni di forward (Vf). I valori di Ron e Vf che compaiono in Figura 2.28a sono ricavati ancora una volta dal file di inizializzazione del modello benchmark [vedi sezione 2.1.1]. In PLECS ho invece utilizzato il blocco *"IGBT Converter with Parasitics (3ph)"* ossia un modello di inverter trifase a IGBT presente in libreria con valori di resistenza di on e forward voltage impostabili da maschera (fig. 2.28b).

Come già detto in questo modello è stato introdotto il deadtime. Per quanto riguarda Simulink, la libreria fornisce il componente *"On/Off delay"*, quest'ultimo applica un ritardo al segnale di input quando l'input cambia da FALSE (0) a TRUE (1) o viceversa. In questo caso è stato impostato in modalità "On delay" in modo tale che quando l'input diventa TRUE l'output diventa a sua volta TRUE dopo un intervallo specifico di tempo (1 μs), viceversa se l'input passa al valore FALSE l'output diventa immediatamente FALSE. Il delay è stato aggiunto per ogni comando di gamba come riportato in Figura 2.29. In PLECS invece è stato introdotto il blocco *"Blanking Time"* a valle del modulatore (vedi Figura 2.27b); quest'ultimo provvede ad inserire opportunamente il ritardo di on su ogni comando di gamba ottenendo quindi lo stesso risultato ottenuto in Simulink.

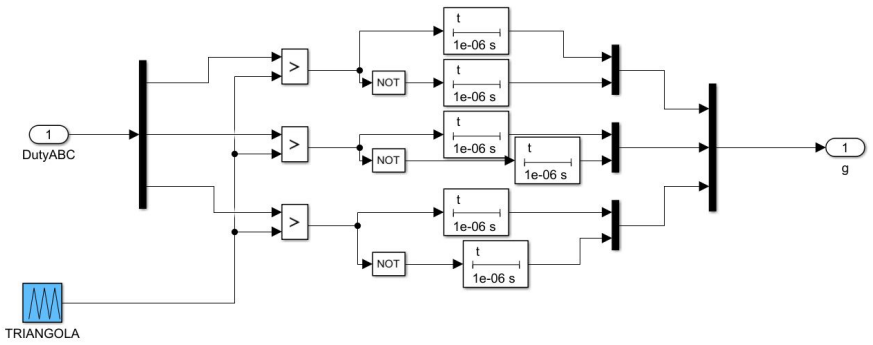


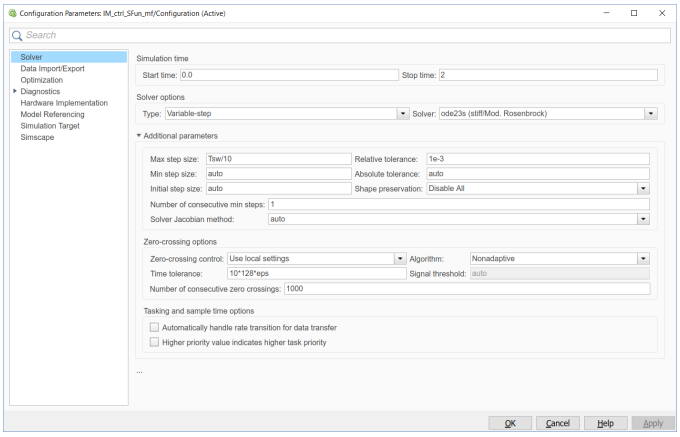
Figura 2.29: On Delay Simulink

2.3 Confronto risultati simulazione

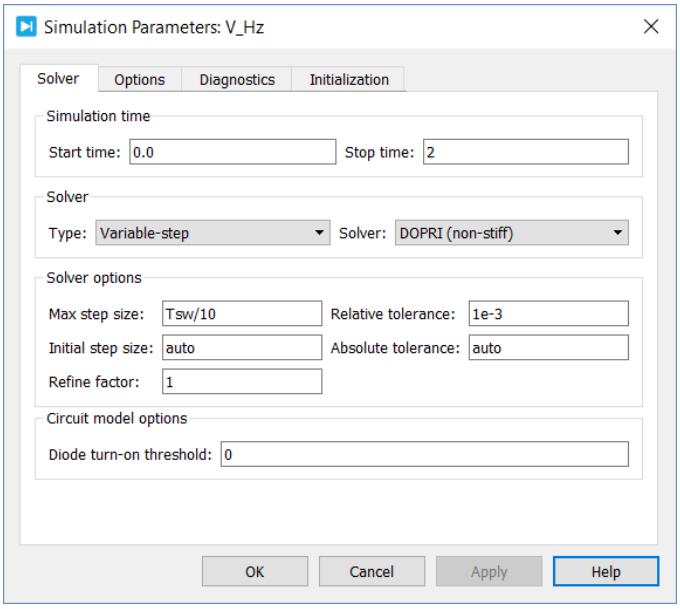
In questa sezione vengono presentati i risultati ottenuti dalle simulazioni dei modelli. Per entrambi i motori troviamo a sinistra i valori ottenuti con Simulink mentre a destra quelli del modello analogo realizzato in PLECS. Tutti i modelli presentati sono stati simulati sia con un solver Fixed-Step che con uno di tipo Variable-Step. I modelli del motore SPM però risultano incompatibili con un solver a step variabile, in entrambi gli ambienti infatti la simulazione si blocca dopo pochi secondi senza però segnalare errori.

Simulink presenta in generale una maggiore varietà di solver mentre PLECS propone solamente il solver "Discrete" per il caso fixed e "RADAU (stiff)" o "DOPRI (Non stiff)" per il variable. Nel caso delle simulazioni fixed step è stato usato il solver denominato "Discrete", presente anche in Simulink, mentre per il caso variable ho utilizzato un solver di tipo "stiff"; in generale possiamo definire un modello "rigido" (*stiff* in inglese) quando sono presenti componenti che variano con costanti di tempo estremamente diverse [15], nei modelli presi in considerazione, per esempio, viene simulato un inverter ideale con switching istantaneo ma anche delle componenti meccaniche molto più lente. É stato quindi selezionato il solver RADAU per PLECS e Ode23s per Simulink.

Nelle prossime pagine vengono presentarti i risultati ottenuti dalle simulazioni, il passo in fixed step è stato impostato a 5 μs per i modelli AVG e Logico mentre per il modello con inverter Fisico in cui viene modellato un dead time di 1μs è necessario ridurre il passo di simulazione. Usando un passo di simulazione maggiore del dead time quest'ultimo ovviamente non sarebbe simulato correttamente di conseguenza il passo è stato impostato esattamente a 1μ. Non è stato possibile ridurre il passo a valori inferiori al microsecondo, la RAM richiesta dai software infatti avrebbe superato quella disponibile sul computer utilizzato.

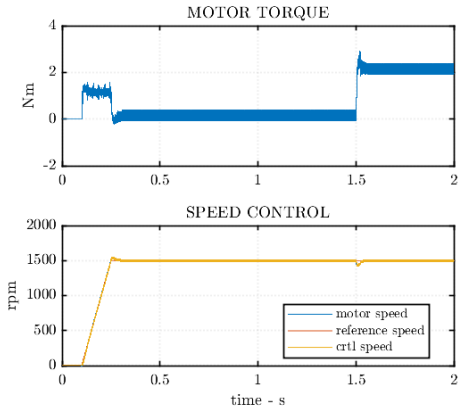


(a) Impostazioni solver Ode23s

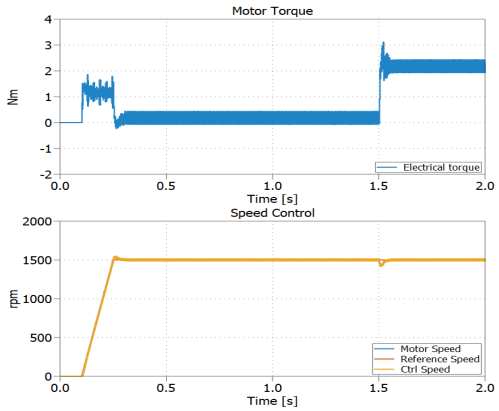


(b) Impostazioni solver RADAU

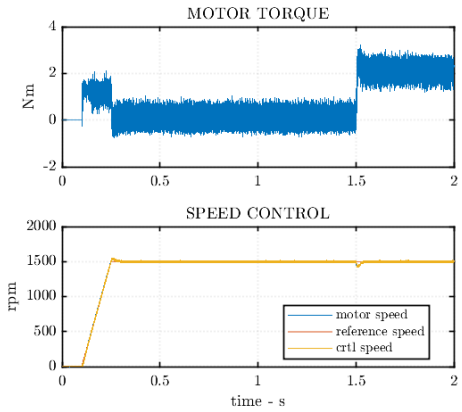
Figura 2.30: Impostazioni solver variable step



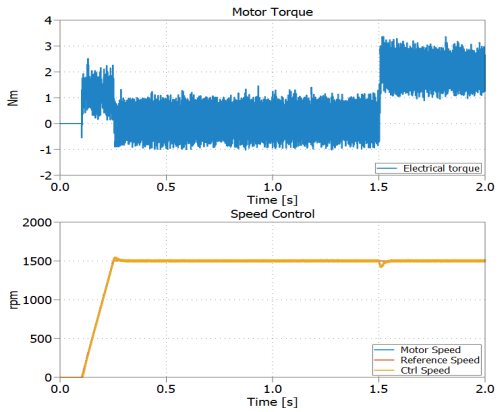
(a) Modello Average Simulink



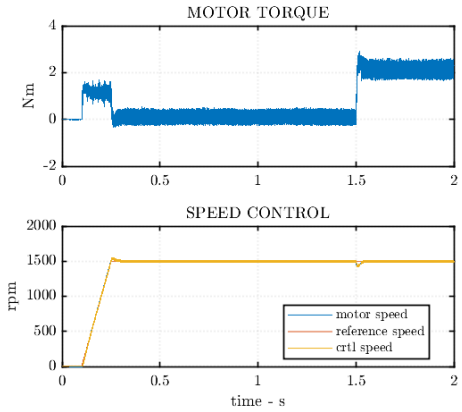
(b) Modello Average PLECS



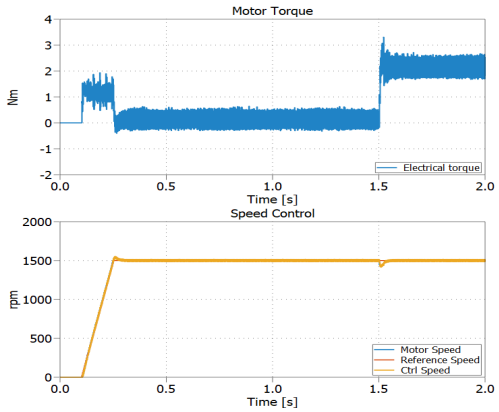
(c) Modello Logico Simulink



(d) Modello Logico PLECS

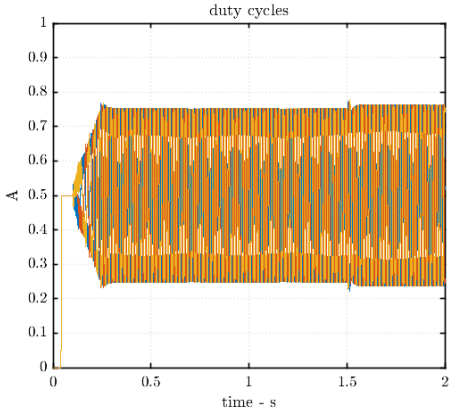


(e) Modello Fisico Simulink

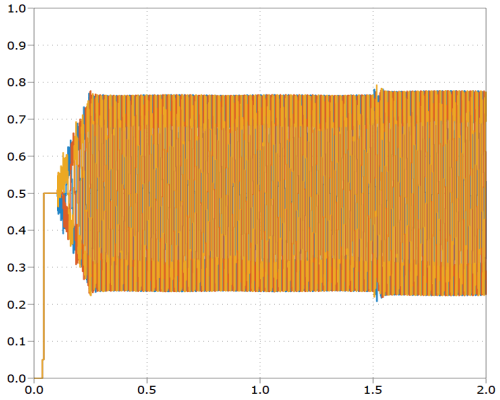


(f) Modello Fisico PLECS

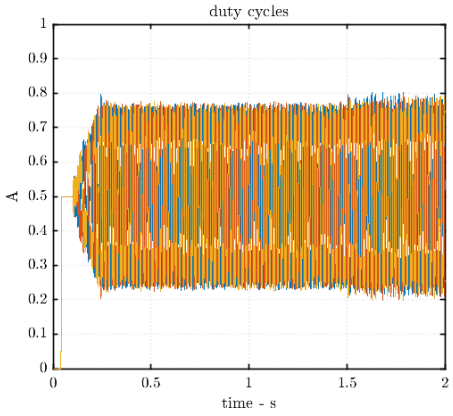
Figura 2.31: Velocità e coppia modelli SPM



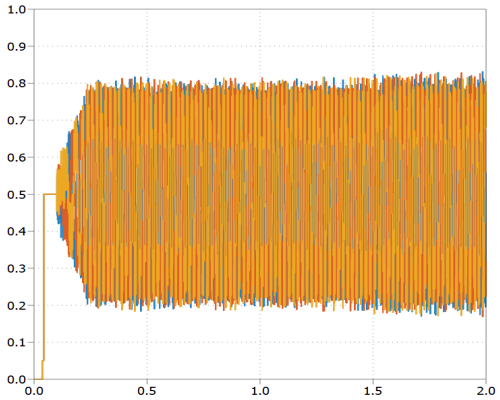
(a) Modello Average Simulink



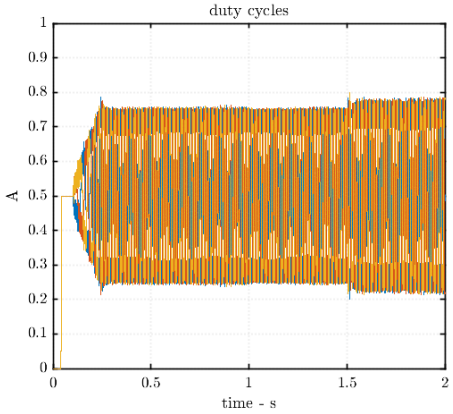
(b) Modello Average PLECS



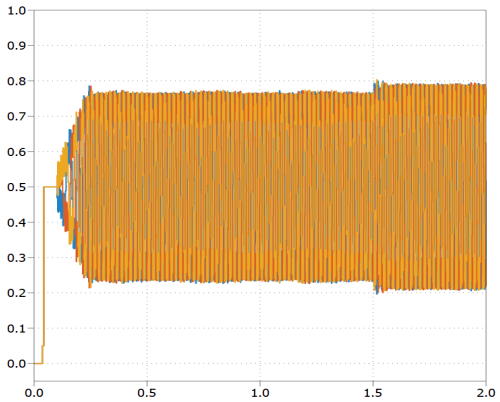
(c) Modello Logico Simulink



(d) Modello Logico PLECS

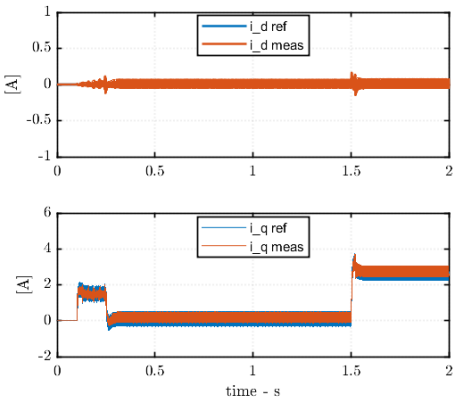


(e) Modello Fisico Simulink

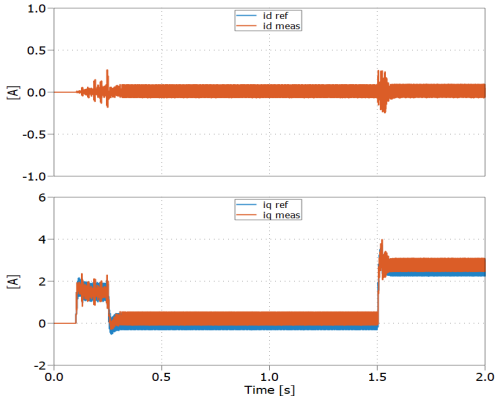


(f) Modello Fisico PLECS

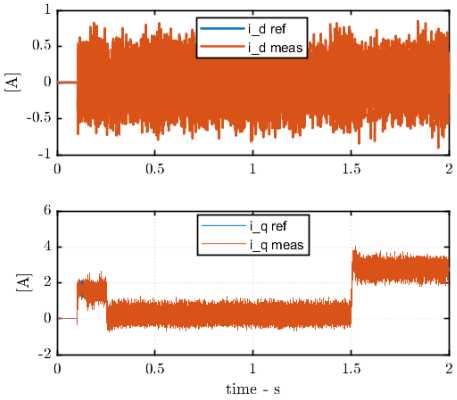
Figura 2.32: Duty Cylce modelli SPM



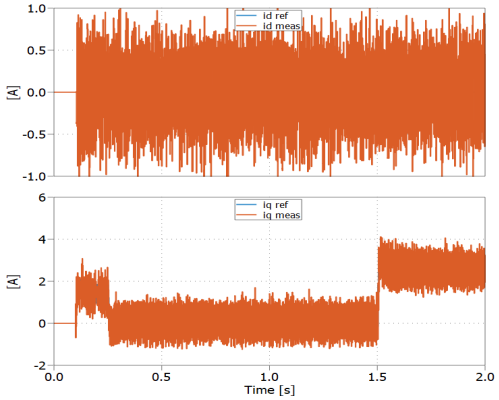
(a) Modello Average Simulink



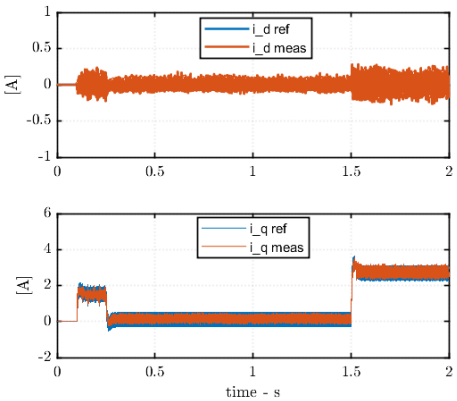
(b) Modello Average PLECS



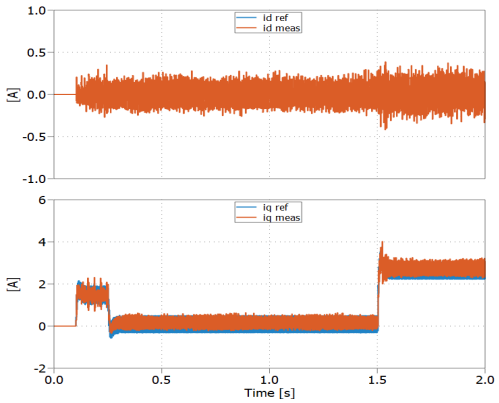
(c) Modello Logico Simulink



(d) Modello Logico PLECS

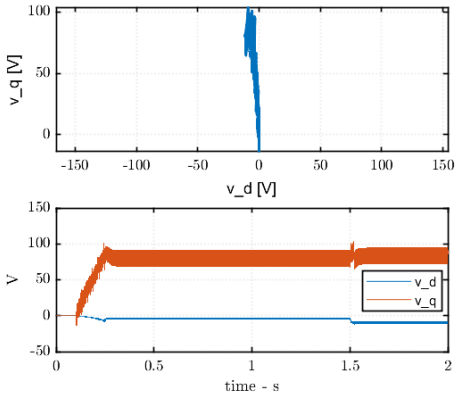


(e) Modello Fisico Simulink

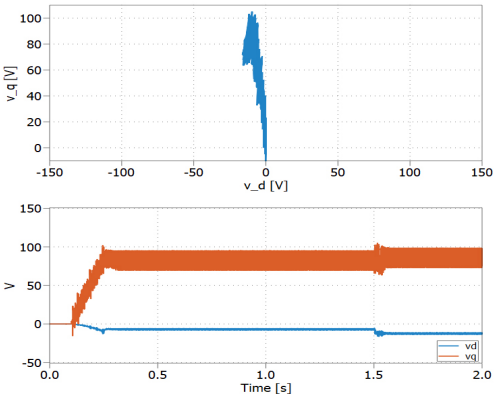


(f) Modello Fisico PLECS

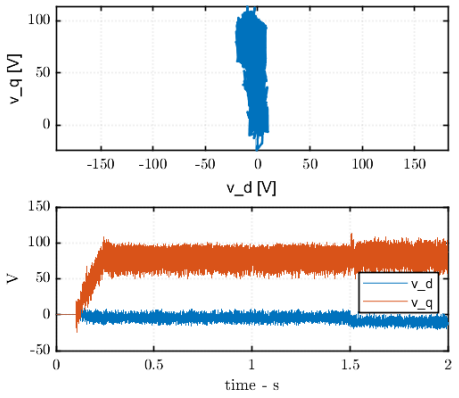
Figura 2.33: Correnti assi dq modelli SPM



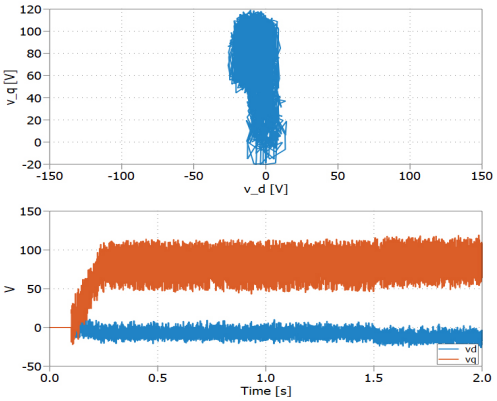
(a) Modello Average Simulink



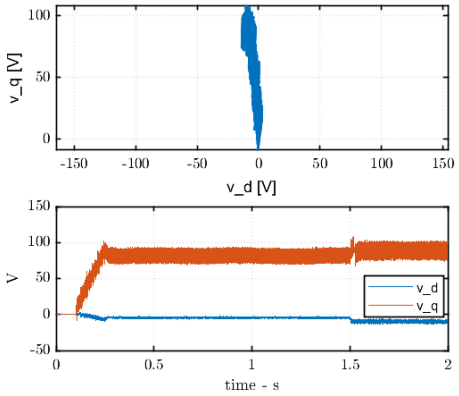
(b) Modello Average PLECS



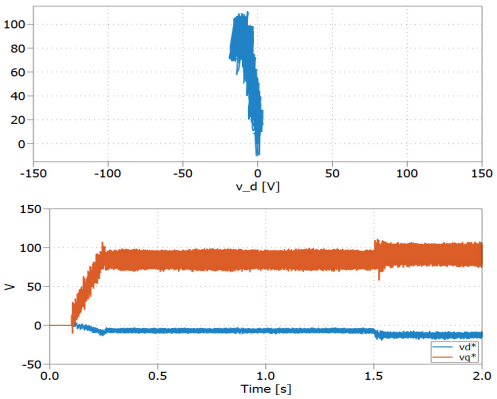
(c) Modello Logico Simulink



(d) Modello Logico PLECS

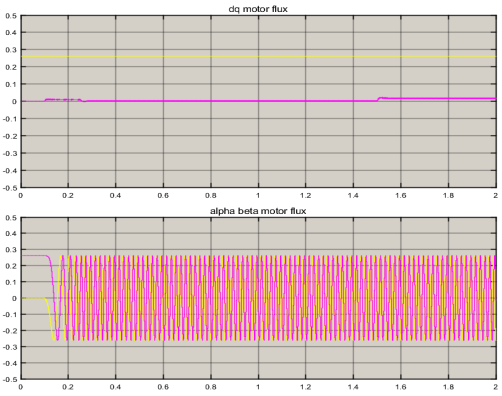


(e) Modello Fisico Simulink

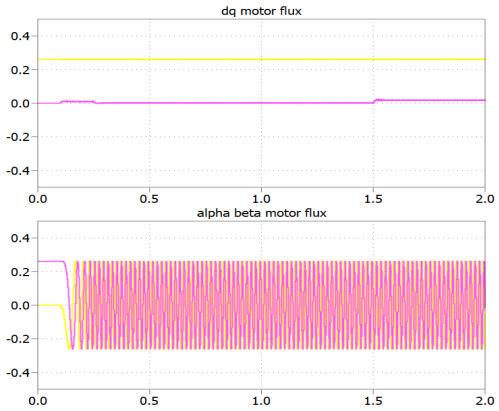


(f) Modello Fisico PLECS

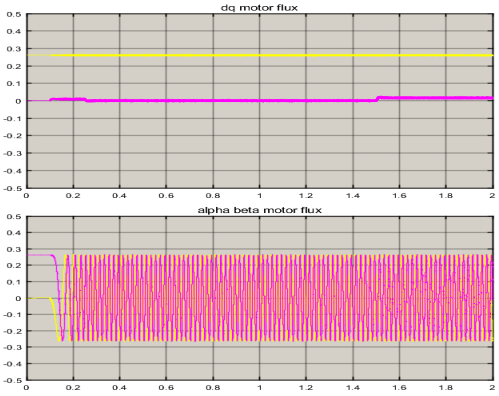
Figura 2.34: Tensioni assi dq modelli SPM



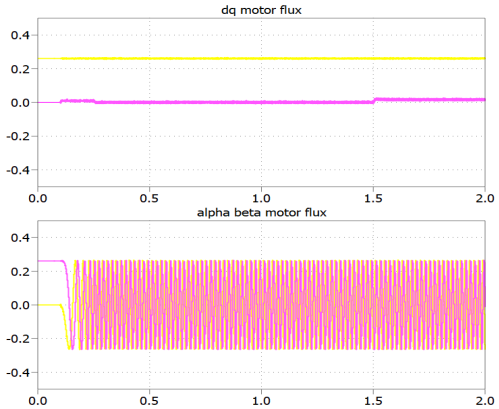
(a) Modello Average Simulink



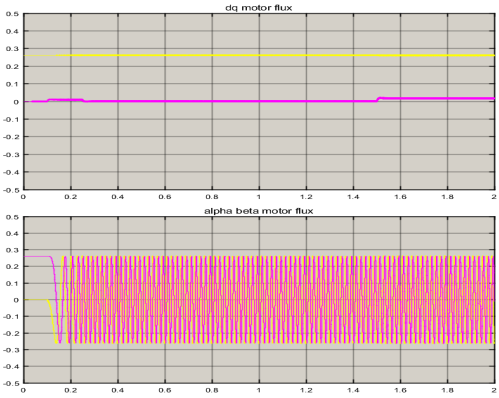
(b) Modello Average PLECS



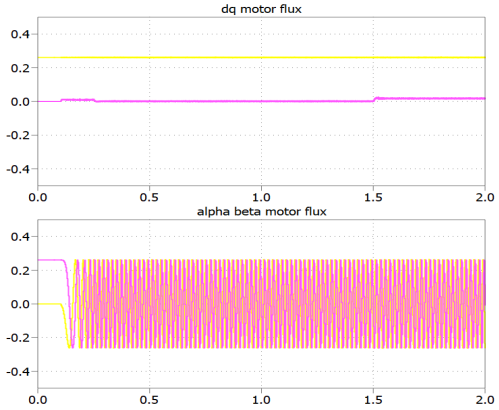
(c) Modello Logico Simulink



(d) Modello Logico PLECS

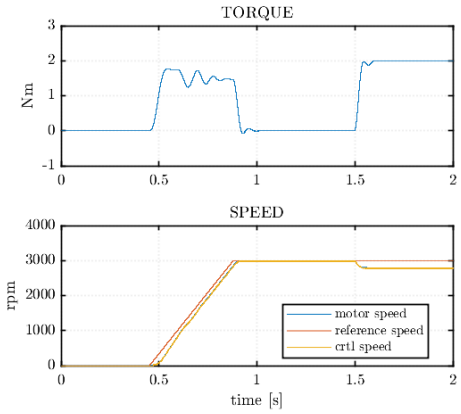


(e) Modello Fisico Simulink

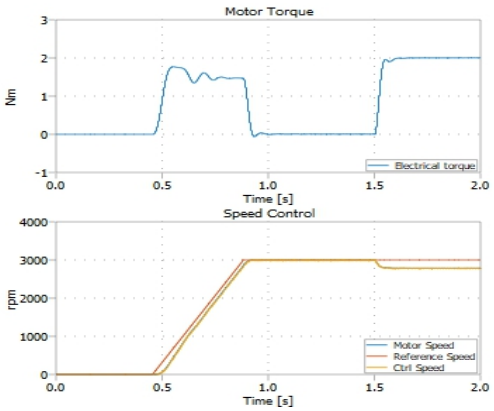


(f) Modello Fisico PLECS

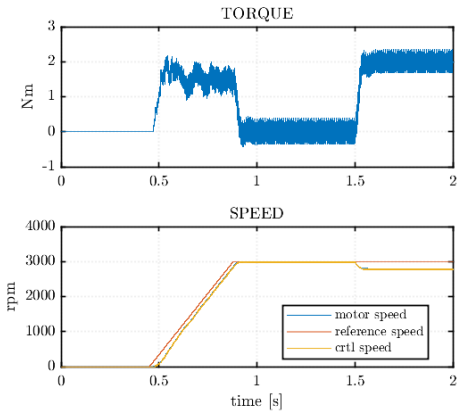
Figura 2.35: Flussi modelli SPM



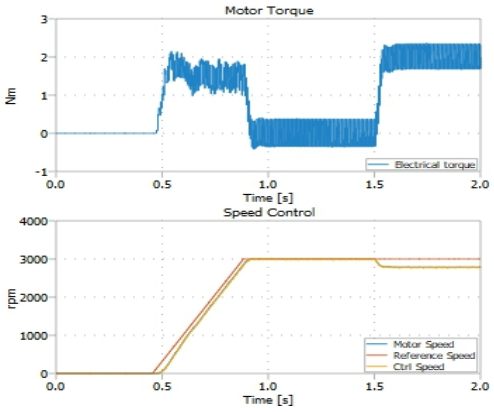
(a) Modello Average Simulink



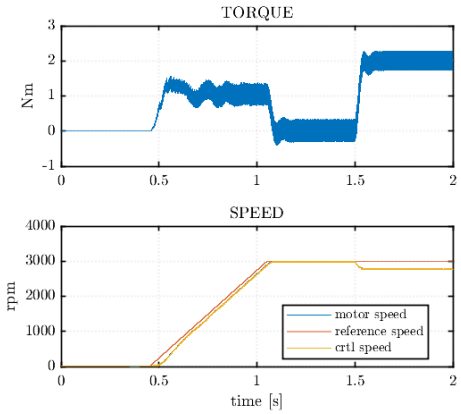
(b) Modello Average PLECS



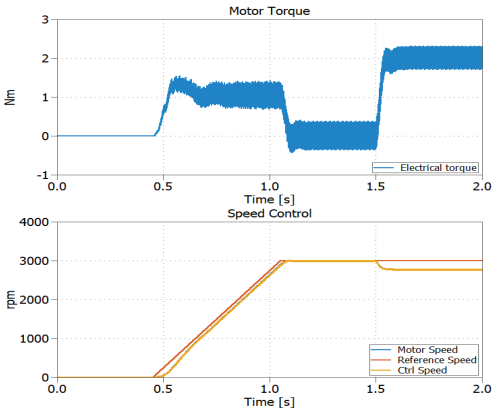
(c) Modello Logico Simulink



(d) Modello Logico PLECS

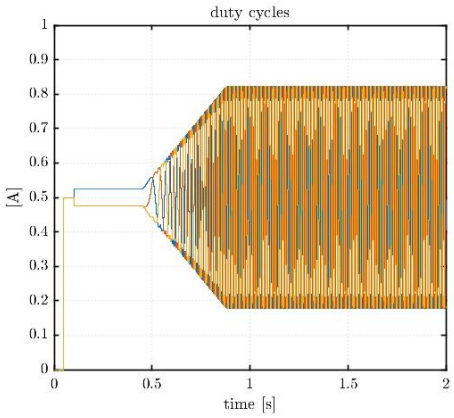


(e) Modello Fisico Simulink

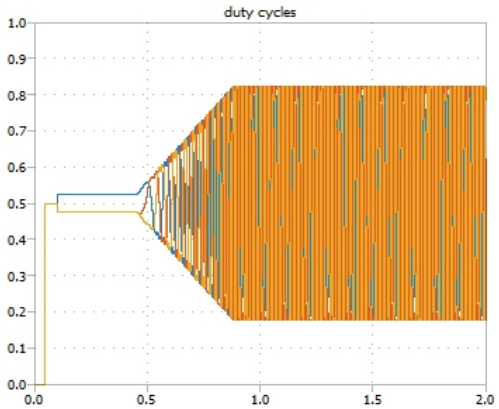


(f) Modello Fisico PLECS

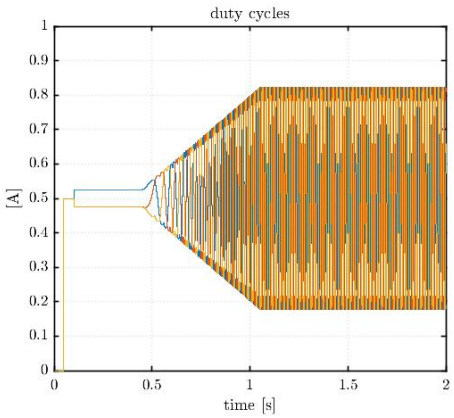
Figura 2.36: Velocità e coppia modelli IM



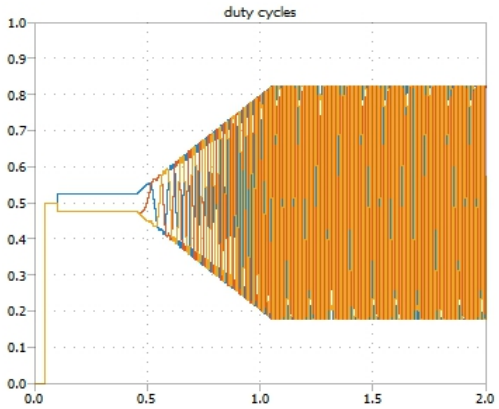
(a) Modello Average Simulink



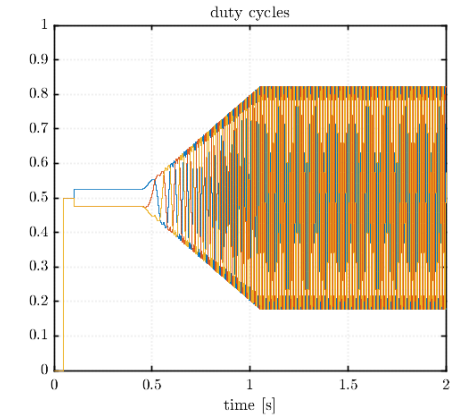
(b) Modello Average PLECS



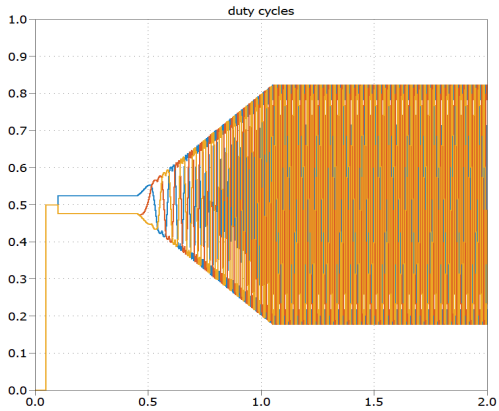
(c) Modello Logico Simulink



(d) Modello Logico PLECS

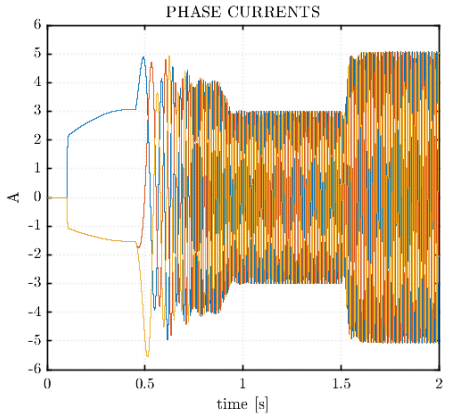


(e) Modello Fisico Simulink

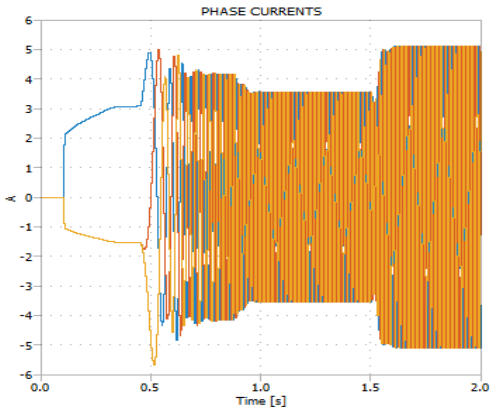


(f) Modello Fisico PLECS

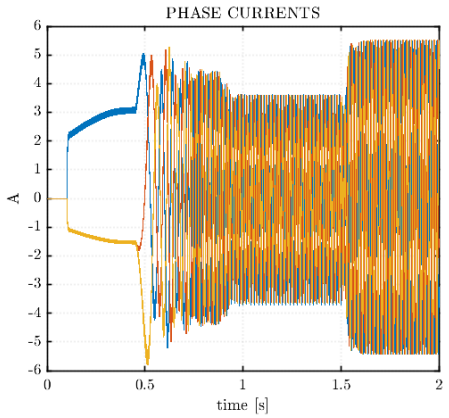
Figura 2.37: Duty Cylce modelli IM



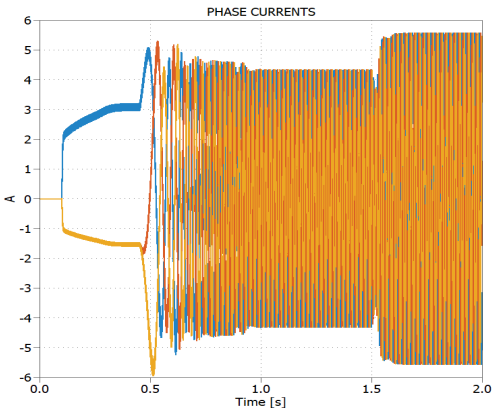
(a) Modello Average Simulink



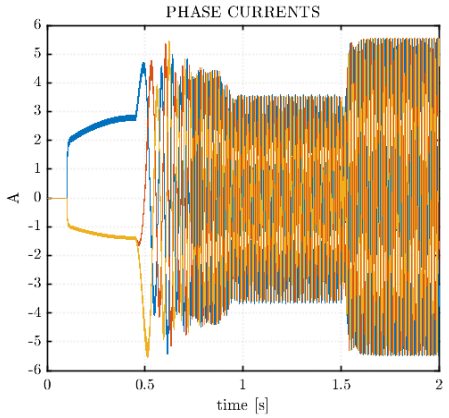
(b) Modello Average PLECS



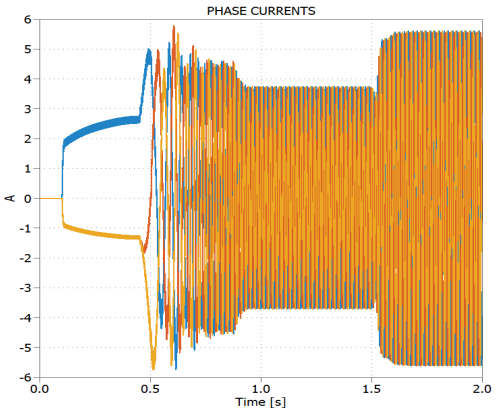
(c) Modello Logico Simulink



(d) Modello Logico PLECS

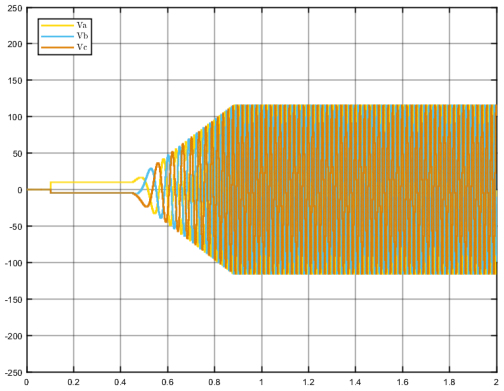


(e) Modello Fisico Simulink

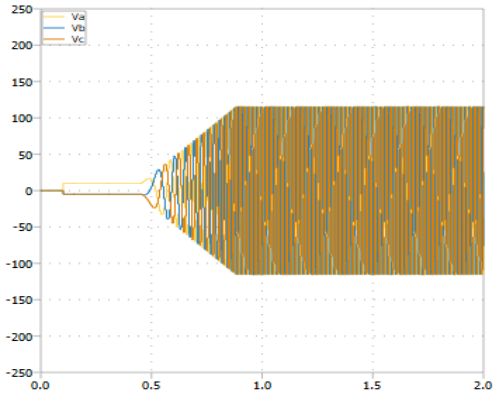


(f) Modello Fisico PLECS

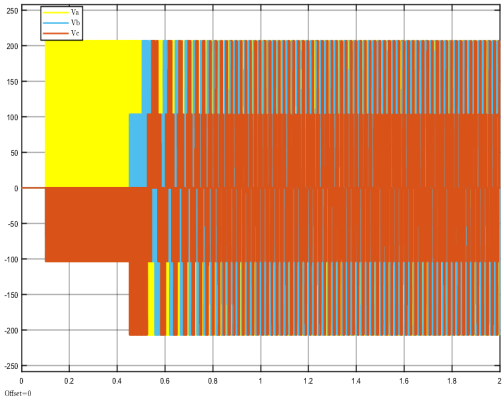
Figura 2.38: Correnti modelli IM



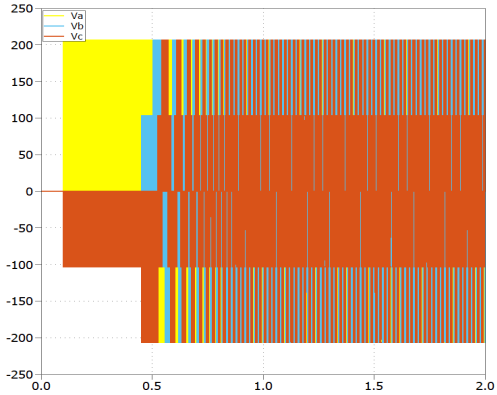
(a) Modello Average Simulink



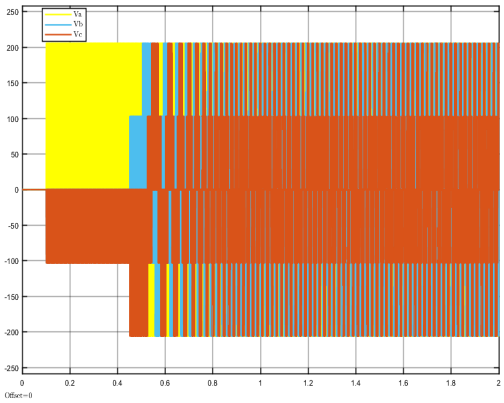
(b) Modello Average PLECS



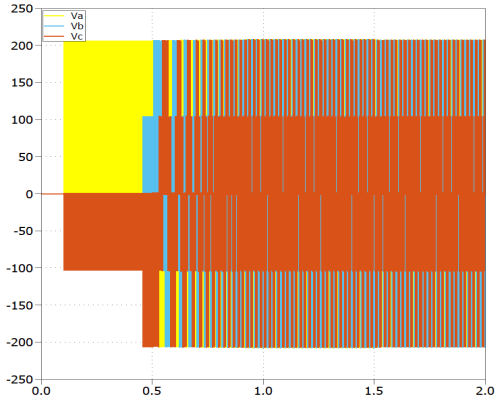
(c) Modello Logico Simulink



(d) Modello Logico PLECS

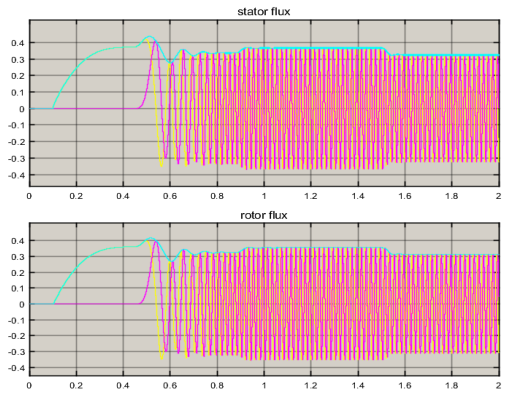


(e) Modello Fisico Simulink

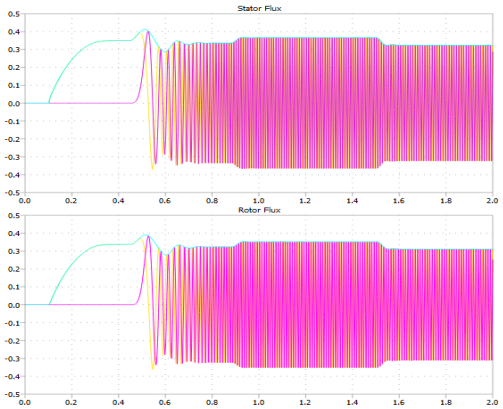


(f) Modello Fisico PLECS

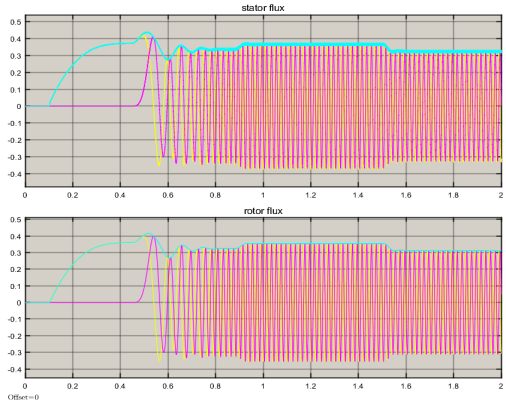
Figura 2.39: Tensioni trifase modulate modelli IM



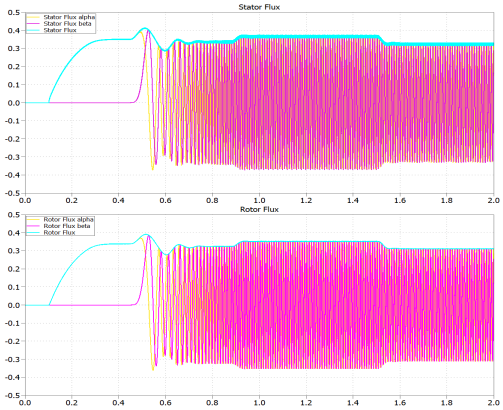
(a) Modello Average Simulink



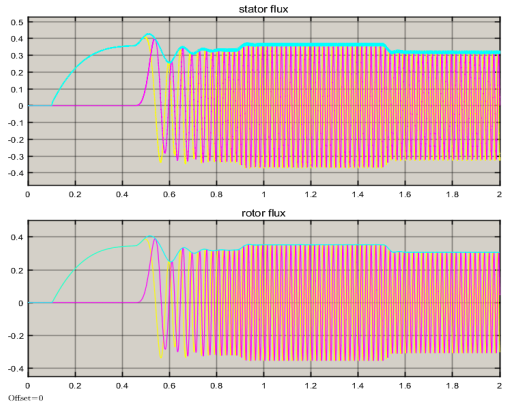
(b) Modello Average PLECS



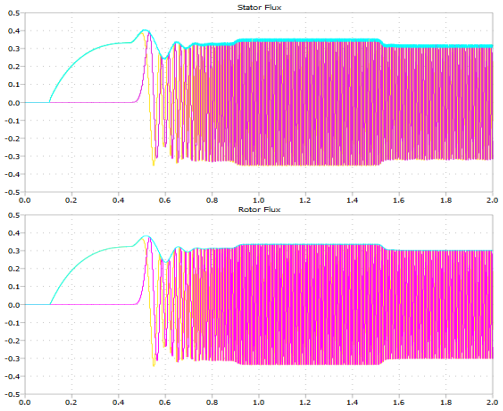
(c) Modello Logico Simulink



(d) Modello Logico PLECS



(e) Modello Fisico Simulink



(f) Modello Fisico PLECS

Figura 2.40: Flussi modelli IM

Osservando i risultati riportati nelle pagine precedenti risulta evidente, a parità di solver e di modello, una maggiore rumorosità nei modelli PLECS. La differenza è evidente per esempio per le correnti in assi dq del modello logico del motore SPM (vedi Figura 2.33d). Il passo di calcolo è ovviamente lo stesso e le impostazioni dei modelli nei due ambienti sono speculari, questa differenza sembra essere imputabile quindi al modello dell'inverter logico usato in PLECS che, per esempio, non presenta resistenza di On contrariamente a quello di Simulink, nel quale come già detto in sottosezione 2.2.4 è presente, seppur piccola, una Ron. Questa differenza rimane, seppur in maniera minore grazie anche ad un passo di simulazione più fitto nel modello Fisico. Per quanto riguarda il modello con inverter Average non si notano invece differenze evidenti tra i due programmi se non in termini di tempi di calcolo come illustrato nelle prossime pagine.

2.3.1 Tempi di calcolo

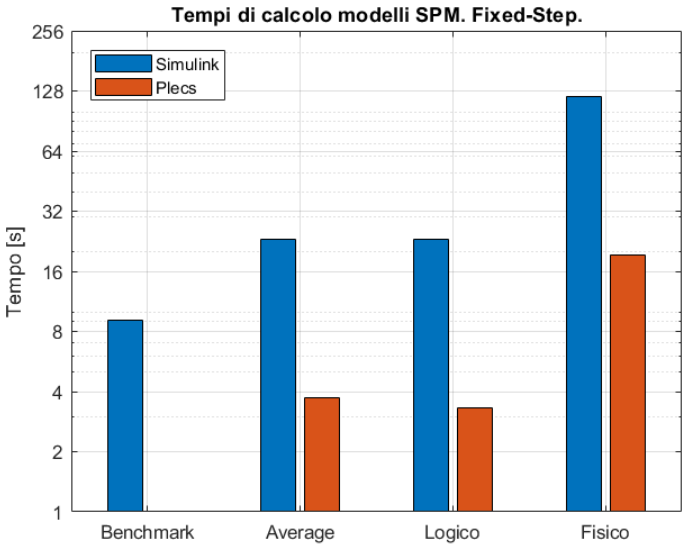
Viene infine presentato un confronto dei tempi di calcolo necessari ai due programmi per simulare i modelli precedentemente presentati. Vengono riportati sia i tempi in Fixed-Step che in Variable-Step. I valori riportati alle figure 2.41 e 2.42 sono ottenuti dalla media dei tempi di 5 simulazioni per ogni modello e per ogni solver. L'asse dei tempi, data la notevole disparità tra le simulazioni è presentato in scala logaritmica. I tempi di calcolo di Simulink sono ottenuti usando le funzioni tic-toc richiamate automaticamente all'avvio/stop della simulazione. Lo stesso risultato si può ottenere con PLECS usando lo script riportato a fondo pagina (codice 2.6).

É evidente la disparità di prestazioni tra Simulink e PLECS, il secondo software risulta in generale più veloce, in particolare la differenza è netta nel caso fixed step; per esempio, simulando il modello fisico a 1μs i tempi necessari a PLECS sono paragonabili a quelli che Simulink richiede per simulare il modello benchmark, in generale più rapido rispetto agli altri modelli in ambiente MATLAB, con un passo di simulazione 5 volte maggiore. L'andamento viene confermato anche nel caso del solver a passo variabile, anche se il confronto è possibile soltanto nel caso del motore asincrono dato che i modelli SPM non risultano compatibili come già detto, anche in questo caso infatti il software PLECS risulta nettamente più veloce.

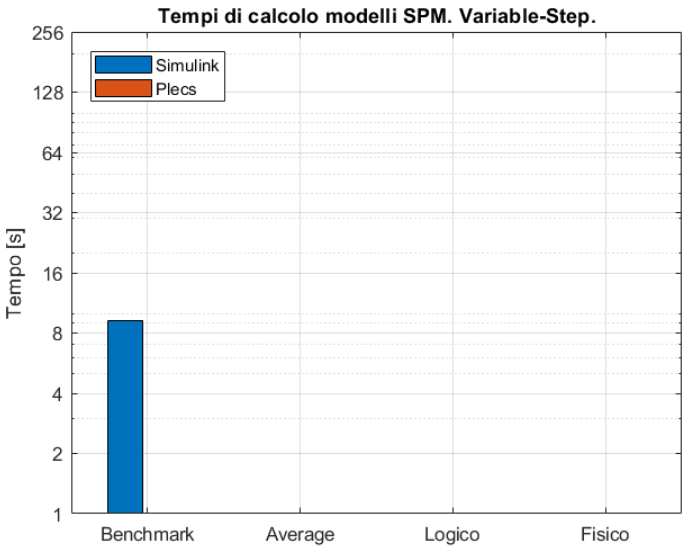
Per quanto riguarda invece il confronto tra i due tipi di solver, fixed e variable step, se dal punto di vista dei risultati non si notano differenze tali da essere evidenziate, possiamo notare alcune differenze nei tempi di calcolo. Per quanto riguarda i modelli *Average* il solver a passo variabile richiede tempi di calcolo minori, con differenze evidenti per l'ambiente MATLAB; il passo di calcolo utilizzato dal solver variable risulta probabilmente maggiore rispetto ai 5 μs scelti per il caso fixed consentendo quindi di ottenere un risultato con un numero di passi minore e quindi un tempo di calcolo minore; un comportamento analogo viene evidenziato per il modello *Fisico*. Al contrario nel caso *Logico*, il tempo di calcolo aumenta in entrambi gli ambienti ed in particolare in PLECS, questo è dovuto all'utilizzo di un passo di calcolo minore rispetto a quello impostato in modalità fixed per effetto probabilmente delle commutazioni degli interruttori ideali che costituiscono l'inverter logico.

```
tic
plecs('simulate')
toc
```

Codice 2.6: Script PLECS tic-toc

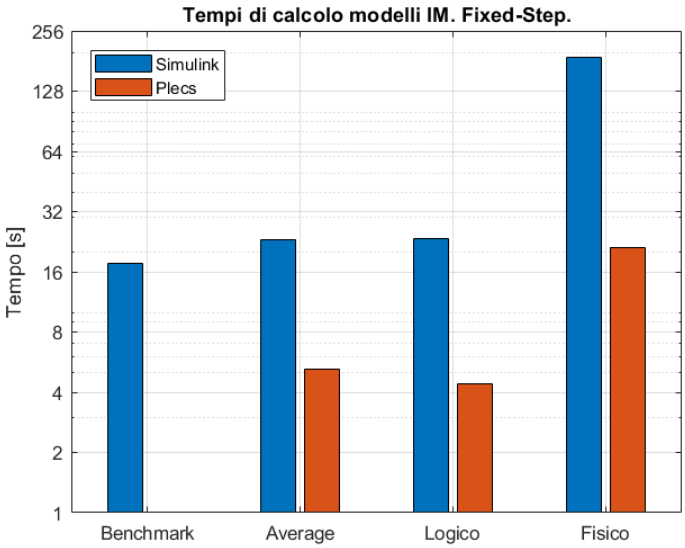


(a) Fixed-Step

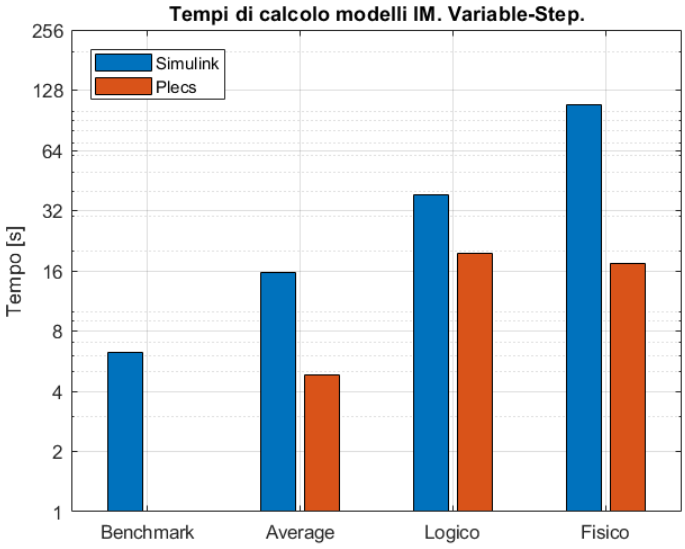


(b) Variable-Step

Figura 2.41: Tempi di calcolo SPM



(a) Fixed-Step



(b) Variable-Step

Figura 2.42: Tempi di calcolo IM

Avendo osservato i risultati e i tempi di calcolo si può affermare che, sicuramente, PLECS risulta essere una valida alternativa all'ambiente MATLAB, risultando in generale più rapido, leggero ed intuitivo rispetto all'alternativa di proprietà MathWorks. Per quanto riguarda i modelli presentatati però, pensando per esempio allo scopo didattico per cui sono nati, ritengo che l'unico modello che potrebbe sostituire i modelli *Benchmark* stessi sarebbe un modello *Average* in ambiente PLECS in cui possono essere implementati in aggiunta la modulazione PWM ed il deadtime. Il problema principale di introdurre il deadtime è dato dal fatto che, per essere competitivo in termini di tempi di calcolo, il modello dovrebbe essere simulato con un solver a passo variabile che, come sottolineato in precedenza, non risulta però compatibile con ogni modello; usando un solver fixed dovremmo utilizzare anche in questo caso un passo estremamente ridotto (almeno pari al deadtime) riconducendo la simulazione a tempi di calcolo prossimi a quelli del modello *Fisico* senza però avere le cadute di on e senza che vengano evidenziati nei risultati gli effetti della commutazione presenti invece nel caso con inverter fisico.

Capitolo 3

Hardware-in-the-loop

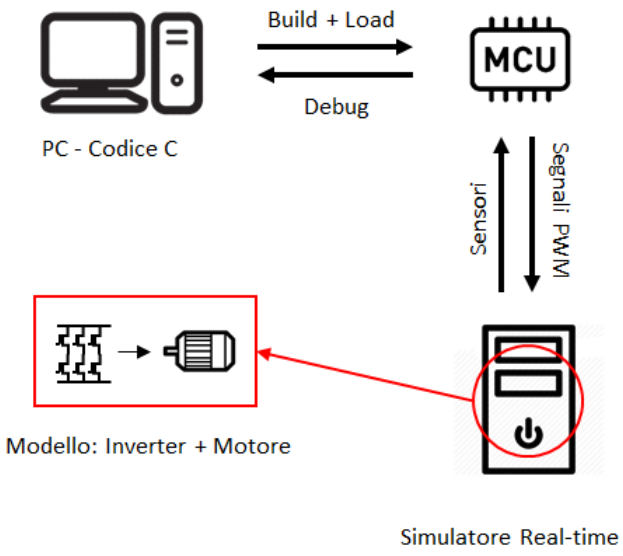


Figura 3.1: Sistema Hardware-in-the-loop di un azionamento elettrico

Obiettivo di questa seconda parte dell'elaborato è quello di realizzare un modello del motore sincrono a magneti permanenti Microphase S140-2B353, utilizzato per introdurre gli studenti alla programmazione dei microcontrollori durante il corso [1] già citato. Il modello viene sviluppato in ambiente PLECS, sfruttando quindi i modelli e le abilità acquisite durante la realizzazione della prima parte dell'elaborato,

[1] *"Controllo digitale di convertitori e azionamenti"* tenuto dal Professor. Pellegrino per il corso di laurea magistrale in Ingegneria Elettrica del Politecnico di Torino

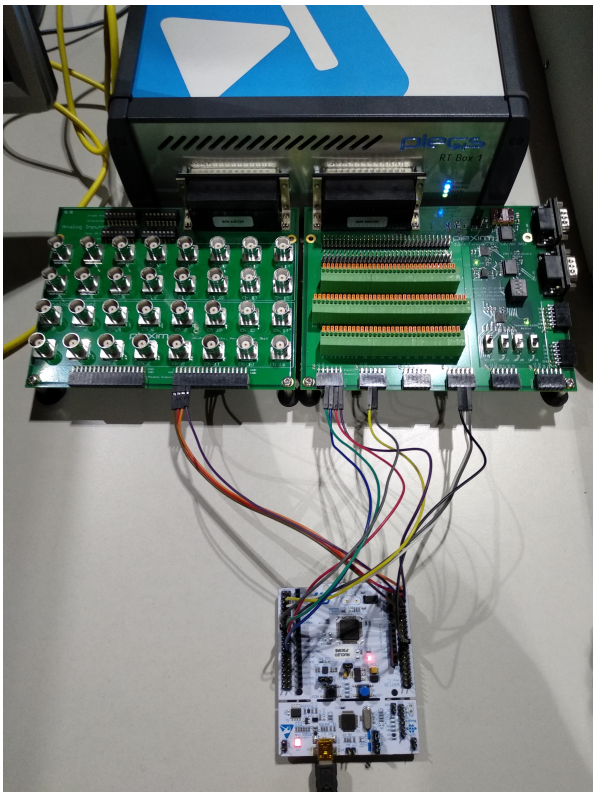


Figura 3.2: Hardware-in-the-loop. RT Box e scheda Nucleo

e è utilizzato in un sistema Hardware-in-the-loop (HiL) grazie al simulatore real time "RT Box" prodotta da Plexim. Innanzitutto viene implementato un controllo su microcontrollore connesso al motore tramite un inverter, analizzato il comportamento del motore e a partire dal datasheet e da alcune misure viene quindi sviluppato il modello da simulare in real time. Prima di procedere con questo caso specifico viene presentata una breve introduzione sul concetto di HiL.

3.1 Cos'è l'Hardware-in-the-loop?

La simulazione con Hardware-in-the-loop è una tecnica che consente di sviluppare e testare sistemi di controllo usati per applicazioni real-time. In un sistema reale la macchina o le componenti fisiche sono connessi ad un controllore tramite l'uso di sensori. In un HiL le componenti fisiche vengono sostituite da una simulazione che, se progettata in modo corretto, replica esattamente il comportamento dell'impianto anche in termini di sensori ed attuatori ed è in grado di ricevere ed interpretare i segnali fisici provenienti dal sistema di controllo; il tutto con dei tempi di simulazione compatibili con i tempi del sistema simulato, in altre parole gli eventi simulati devono avvenire ed avere una durata pari a quella degli eventi "originali".

Molto spesso il sistema di controllo e l'impianto "fisico" vengono progettati contemporaneamente consentendo di testare il controllo solamente durante la fase di commissioning con il rischio di ritardare la consegna o danneggiare il prodotto in caso di errori. La soluzione HiL consente di anticipare il testing che può a questo punto essere fatto anche in assenza del sistema finito ed in completa sicurezza. Inoltre, essendo la componente fisica simulata, diventa possibile testare condizioni nelle quali l'hardware verrebbe danneggiato irrimediabilmente in modo da verificare, per esempio, il corretto intervento di protezioni o algoritmi di sicurezza. Tutto ciò con un vantaggio anche dal punto di vista economico andando ad eliminare la necessità di utilizzare prototipi e precauzioni di sicurezza che possono rivelarsi costose o, più semplicemente, la necessità di recarsi nel luogo di installazione del sistema per il test.

La soluzione HiL più semplice consiste nell'utilizzare un computer sul quale viene installato un software di modellistica e simulazione per simulare il comportamento dell'impianto o dell'azionamento. Bisogna però tener conto del fatto che i sistemi operativi comunemente installati su un PC (Windows per esempio) non lavorano in real-time; per un'accurata simulazione diventa quindi necessario equipaggiarsi di un simulatore HiL basato su un'architettura real-time come la *RT Box* presentata più avanti [10, 9].

3.1.1 Simulatore Real-Time: RT Box

La RT Box (RTB) è un simulatore real-time che può essere utilizzato sia in funzione HiL sia come strumento per "rapid control prototyping". Come già spiegato in precedenza usando la RTB in termini di HiL andremo a simulare in real-time gli stadi di potenza (elettronica di potenza e motore) sfruttando gli input/output analogici e digitali della RTB per comunicare bidirezionalmente con un microcontrollore (MCU). In termini di rapid prototyping invece la RTB può essere usata in sostituzione del MCU con il vantaggio di avere un maggior numero di canali ed una CPU più veloce della maggior parte dei microcontrollori [16].

A livello hardware la RTB è fornita di una CPU dual core da 1 Ghz, 32 canali analogici (16 input + 16 output) e 64 digitali (32 input + 32 output) [vedi 8, p. 21]. La connessione degli input/output della RTB con il MCU è possibile grazie all'utilizzo di due schede di interfaccia, una per i canali analogici e l'altra per quelli digitali. Entrambe le schede forniscono input ed output sotto forma di pin headers facilmente compatibili con qualunque microcontrollore. L'interfaccia analogica è inoltre dotata di connessioni BNC mentre quella digitale presenta anche una morsettiera. In Figura 3.3b sono visibili le due schede connesse alla RTB.

(a) RT Box

(b) Schede di interfaccia RT Box

Figura 3.3: RT Box Hardware. Immagini tratte da [16]

3.2 Configurazione Hardware

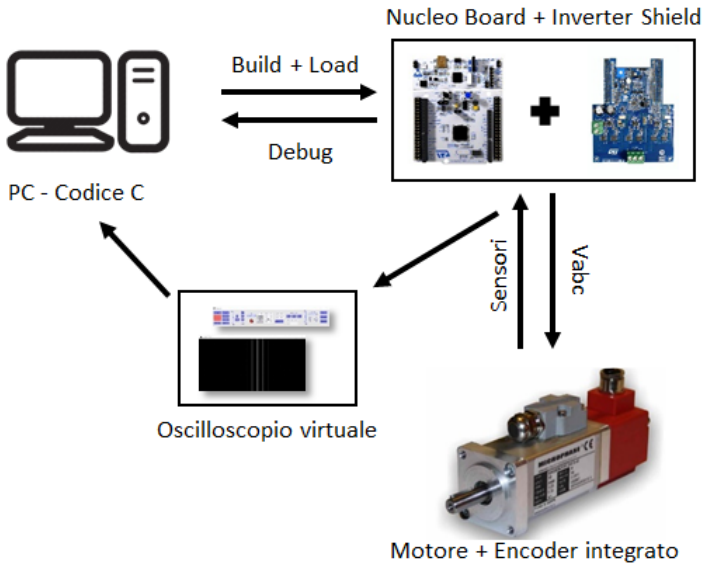


Figura 3.4: Schema configurazione hardware

In Figura 3.4 è riportato uno schema della configurazione hardware che è stata simulata. La scheda inverter X-NUCLEO-IHM08M1 viene montata su una scheda Nucleo STM32 e connessa al motore Microphase S140 ed all'encoder integrato. Nei prossimi paragrafi sono presentati singolarmente i componenti presenti nello schema appena illustrato.

Microcontrollore L'algoritmo di controllo è implementato su un microcontrollore (MCU) della famiglia STM a 32 bit. In particolare è stato utilizzato il STM32F303RE basato su processore ARM Cortex (72MHz clock, DMIPS[2] 90) e montato sulla scheda Nucleo STM32 (Figura 3.5) ossia un "Evaluation Kit" che consente di testare rapidamente il controllo ed il MCU integrando direttamente sulla scheda le connessioni necessarie per interfacciarsi alle componenti di potenza e per consentire la programmazione ed il debug tramite PC.

[2]Dhrystone MIPS, benchmark di misura del numero di istruzioni eseguite al ciclo. MIPS è acronimo di million instructions per second

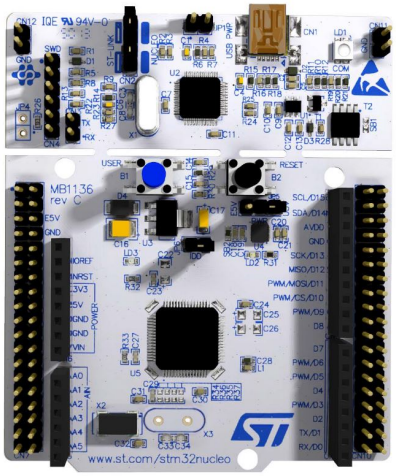
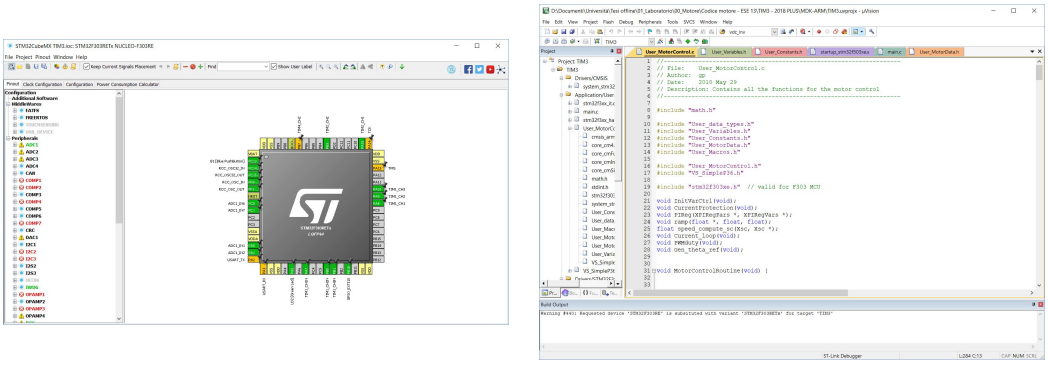


Figura 3.5: Scheda Nucleo STM32. Immagine tratta da [18]

Per programmare la scheda e fare debug e build del codice sono stati utilizzati i seguenti software:

- *STM32CubeMX*[3] prodotto dalla STM consente di attivare ed impostare tramite interfaccia grafica le periferiche presenti sulla scheda Nucleo
- *KEIL µVision5*[4] prodotto dalla KEIL è un ambiente di sviluppo che combina editing, compilazione e debug real-time del codice.



(a) CubeMX (b) µVision5

Figura 3.6: Software

[3]www.st.com/en/development-tools/stm32cubemx.html

[4]www.keil.com/download/product/

Inverter La scheda X-NUCLEO-IHM08M1 (Figura 3.7) è una expansion board prodotta dalla STM e pensata per essere compatibile con le schede Nucleo STM32. La scheda contiene tutto il necessario per controllare un motore brushless trifase. Sulla scheda troviamo un'inverter realizzato con Power MOSFET STripFET™ F7 STL220N6F7 (R_{DS}=0.0014 Ω)[vedi 19] insieme ovviamente ai driver necessari per il loro controllo. Sono presenti inoltre 3 shunt di corrente e un sensore per la misura della tensione di DC-link.

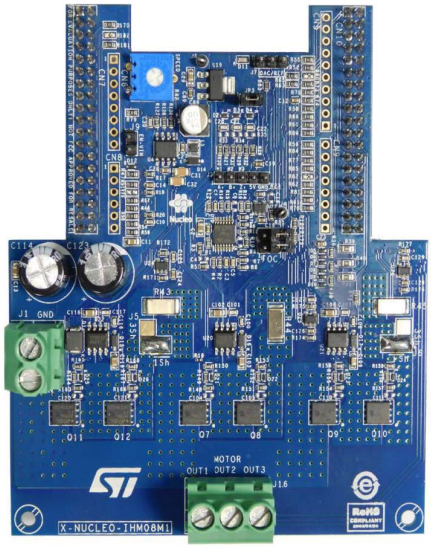


Figura 3.7: Scheda X-NUCLEO-IHM08M1. Immagine tratta da [20]

Microphase S140-2B353 Il motore S140-2B353 , prodotto dalla Microphase, è un servomotore brushless sinusoidale a magneti permanenti (NdFeB) a 4 coppie polari dotato di encoder incrementale a 2048 divisioni. Il motore S140 non viene fatto girare a vuoto ma viene usato per trascinare un secondo motore, in particolare viene utilizzato il Microphase S160-1B303. I parametri principali indicati nel datasheet [11] sono i seguenti:

Tabella 3.1: Principali parametri Motore Microphase S140-2B353

Parametro	*Valore*	[*Unità di misura*]
V_N	17	[V_{AC}]
I_N	6.6	[A]
P_N	100	[W]
N_N	3000	[rpm]
K_E	3.15	[V_{RMS}/krpm]
J	0.06	[kg cm^2]
$R_{U\text{-}V}$	0.5	[Ω]
$L_{U\text{-}V}$	0.53	[mH]

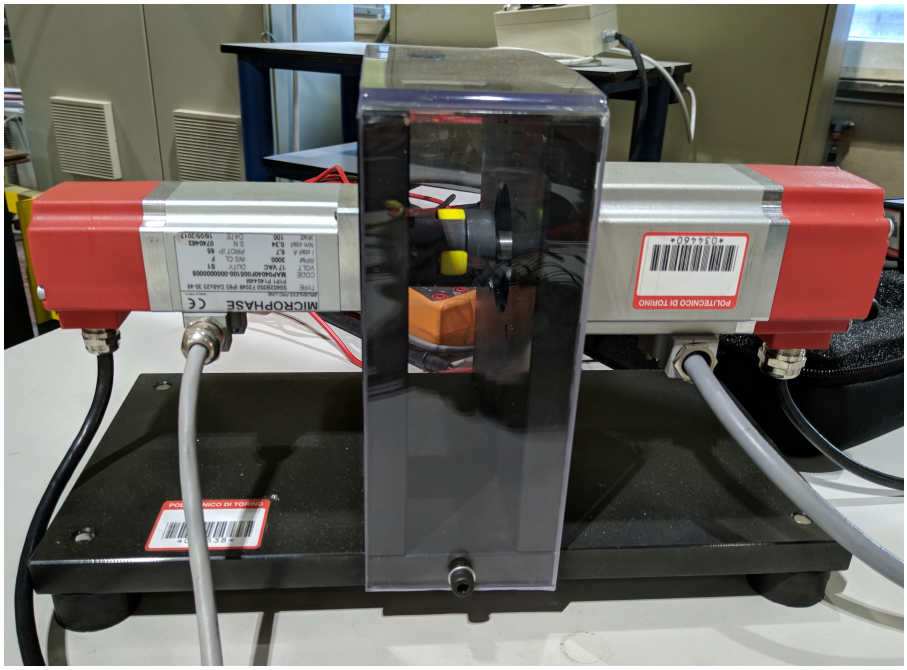


Figura 3.8: Microphase S140 e S160

3.3 Codice di controllo

Il codice di controllo implementato è suddiviso in cinque stati operativi:

- Error;
- Wake Up;
- Commissioning;
- Ready;
- Start;

Lo stato di *Error* è lo stato in cui si trova l'azionamento all'accensione, è essenzialmente una fase di reset in cui viene spenta la modulazione PWM e vengono azzerate alcune variabili. Alla pressone del tasto "Go" sul MCU si passa allo stato di *Wake Up* utilizzato per calcolare gli offset presenti sui canali di acquisizione delle correnti di fase e per eseguire il bootstrap. Dallo stato di Wake Up il controllo passa automaticamente al *Commissioning*, questo stato ha lo scopo di di individuare la posizione iniziale del motore e compensare l'offset dell'encoder; nella prima parte del commissioning viene fatto ruotare il motore per almeno un giro meccanico con l'obiettivo di far scattare la tacca di zero dell'encoder incrementale in modo da rendere assoluto il valore della posizione angolare inviata dall'encoder stesso. A questo punto viene generato un vettore di corrente continua allineato all'asse *alpha* del sistema di riferimento fisso a statore (α,β) in modo da allineare l'asse d di rotore con l'asse α stesso; le correnti sono infine resettate e, prima di passare allo stato di Ready, viene salvata come offset la lettura dell'encoder (*thetaEnc*) con il motore allineato. Questa fase è stata impostata in modo da essere effettuata in automaticamente ad ogni avvio del MCU sebbene sarebbe sufficiente utilizzarla solamente al primo avvio e memorizzare l'offset in una costante. Ripetendo però il commissioning ad ogni avvio viene migliorata la precisione del controllo nel caso in cui dovesse cambiare l'offset angolare dell'encoder nel corso della vita del motore. Inoltre, essendo il Microphase caratterizzato da quattro coppie polari associate ad altrettanti assi d/q automatizzando la procedura di allineamento si rende indipendente il controllo dalla posizione iniziale del motore. La fase di commissioning non è necessaria ai fini del controllo quando, come in questo caso, viene implementato un controllo in anello aperto di velocità ma diventa essenziale qualora si decida di chiudere l'anello per avere una corretta lettura della posizione del motore e quindi della velocità.

Terminato il commissioning si passa automaticamente allo stato di *Ready* in cui vengono inizializzate le variabili necessarie al controllo e vengono impostati a 0.5 i duty cycles di riferimento in modo da modulare a zero le tensioni di fase del motore.

Dallo stato di Ready in seguito alla nuova pressione del tasto "Go" si passa quindi allo stato di *Start* in cui viene introdotto il controllo vero e proprio.

Lo stato di Start implementa un controllo I-Hz [2] il cui schema a blocchi è riportato in Figura 3.9; le componenti riportate in rosso sono superflue per il controllo in se ma vengono comunque utilizzate, in questo caso, per verificare il corretto funzionamento del commissioning e del controllo ed anche per confrontare il comportamento del motore "fisico" e del suo modello HiL.

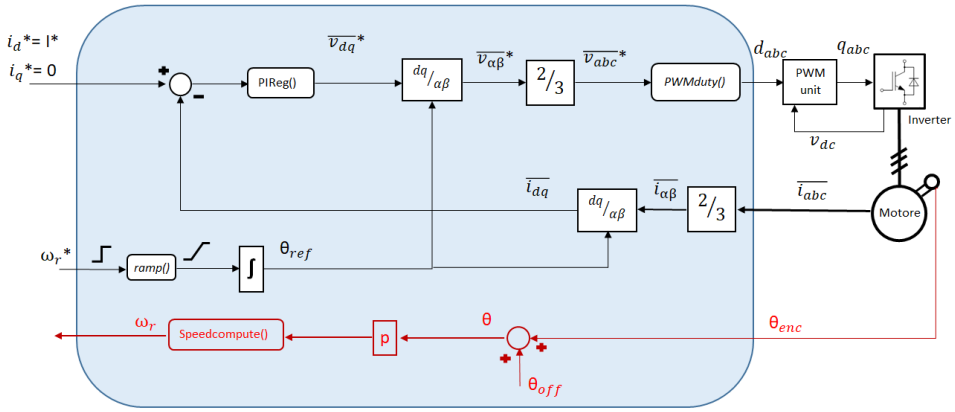


Figura 3.9: Controllo vettoriale I-Hz

Tabella 3.2: Parametri e riferimenti controllo motore

I-Hz
$I_d^* = 3$ A
$I_q^* = 0$ A
$V_{max} = Vdc/\sqrt{3} = 24/\sqrt{3}V$
$Kp_{i\ d/q} = 0.4$
$Ki_{i\ d/q} = 80$*Tsw $= 0.02$

3.4 Implementazione passo passo modello RT Box

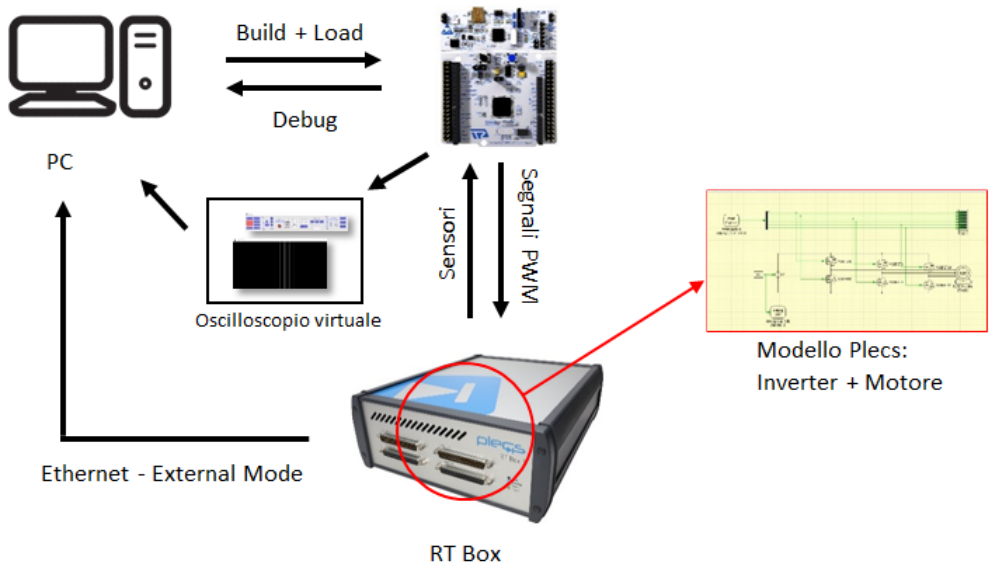
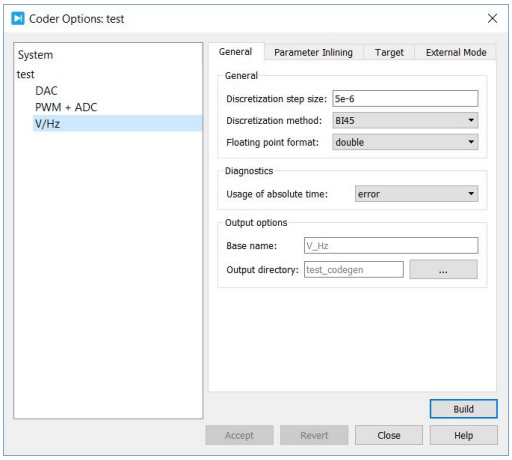


Figura 3.10: Schema HiL - RT Box

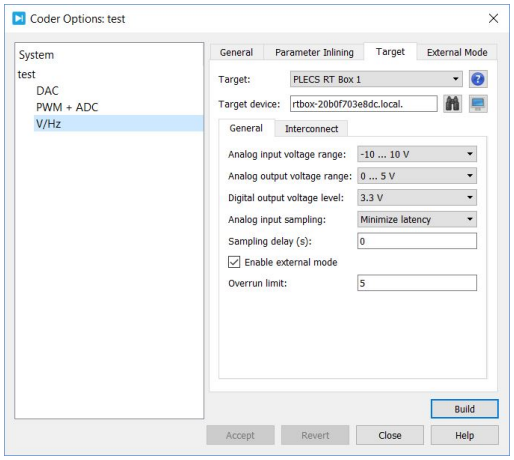
In Figura 3.10 è riportato lo schema dell'HiL con RT Box. Quest'ultima come già detto alla sotto sezione 3.1.1 comunica con un host PC dal quale è possibile caricare il modello e monitorare alcuni valori di simulazione. La comunicazione avviene tramite standard Ethernet ma per poter utilizzare la RTB è necessario che sul computer sia installato, oltre ovviamente il software PLECS (Standalone o Blockset) ed il suo Coder, il protocollo *Zero Configuration Networing*. I passaggi sono esaustivamente spiegati nel manuale della RT Box [8, p. 3]. Ultimate queste impostazioni è finalmente possibile utilizzare il simulatore RTB.

Selezionato il progetto, dalla finestra *Coder* selezioniamo innanzitutto lo step size; l'ampiezza di quest'ultimo è funzione delle dimensioni e della complessità del modello, in una simulazione real time infatti il tempo impiegato per risolvere le equazioni e le funzioni del sistema modellato deve sempre essere minore del passo di simulazione. Nel caso in cui il passo selezionato sia troppo piccolo la RTB va in over run e produce un segnale d'errore.

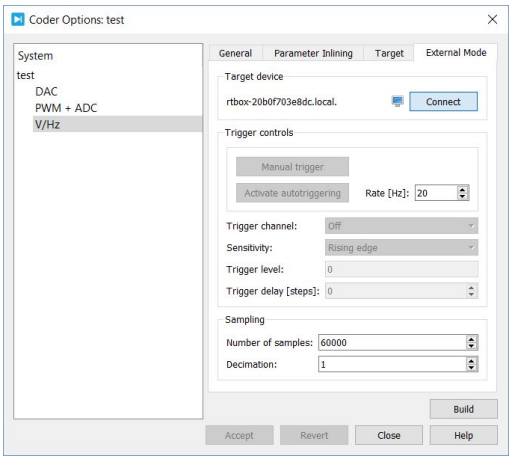
Sempre nella finestra Coder, passiamo alla scheda *Target* e spostiamo il selettore da "Generic" a "PLECS RT Box" selezionando come target la Box da noi utilizzata verificando l'hostname riportato sulla targa [vedi 3.11b]. Premendo infine il tasto "Build" viene generato, compilato e caricato sulla Box il codice del modello. Infine, per connettere il modello real-time con dei segnali di I/O reali è sufficiente utilizzare gli appositi blocchi dalla libreria *PLECS RT Box* che verranno approfonditi in seguito. A questo punto la simulazione è avviata, è possibile monitorare i risultati



(a) General



(b) Selezione Target



(c) External Mode

Figura 3.11: Coder Options

sugli oscilloscopi inseriti nel nostro modello selezionando "Connect" ed avviando l'auto trigger dalla scheda "External Mode" riportata in Figura 3.11c.

Per familiarizzare con la RTB e per analizzare e presentare i blocchi specifici per la simulazione real time ho deciso di procedere con un implementazione passo passo del modello completo di motore. Ho quindi creato una serie di modelli più semplici con cui testare i blocchi della libreria RT Box. Partendo dai risultati di questi test, nei prossimi paragrafi sono illustrati i vari componenti software di input/output necessari alla simulazione HiL.

DAC In un primo test ho utilizzato la RTB come un "Digital to Analog Converter" generando una tensione sinusoidale tramite software ed inviandola all'esterno della box con il blocco "Analog Out". Come è possibile notare in Figura 3.12 nel modello troviamo un generatore di tensione sinusoidale a 50 Hz di ampiezza "vac" il cui valore numerico (24 V) è impostato tramite la pagina di inizializzazione. L'oscilloscopio interno di PLECS, attivando l'external mode come spiegato in precedenza, consente di visualizzare la forma d'onda originale riportata in Figura 3.14a.

L'output analogico è ottenuto grazie ad un fattore di scala ed un offset impostabili tramite il blocco *"Analog Out"* ed è calcolabile come segue:

$$output = input \cdot scale + offset \tag{3.1}$$

Impostando opportunamente la scala e l'offset come riportato in Figura 3.13 ho ottenuto sull'uscita analogica 0 la forma d'onda originale (fig. 3.14a) scalata tra 0V e 3.3V in modo da essere compatibile con la scheda Nucleo. Il valore numerico del voltaggio massimo di output (Adc_max) è come sempre impostato da inizializzazione ed è uguale a 3.3 V. Il corretto funzionamento è stato verificato con l'oscilloscopio digitale ed è riportato in Figura 3.14b.

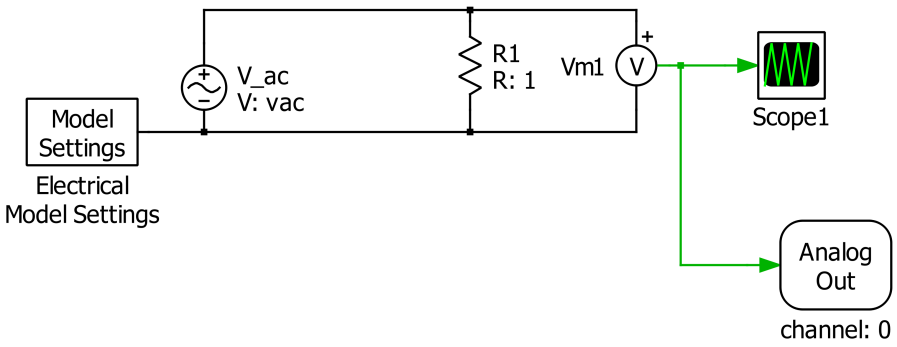


Figura 3.12: Schema DAC

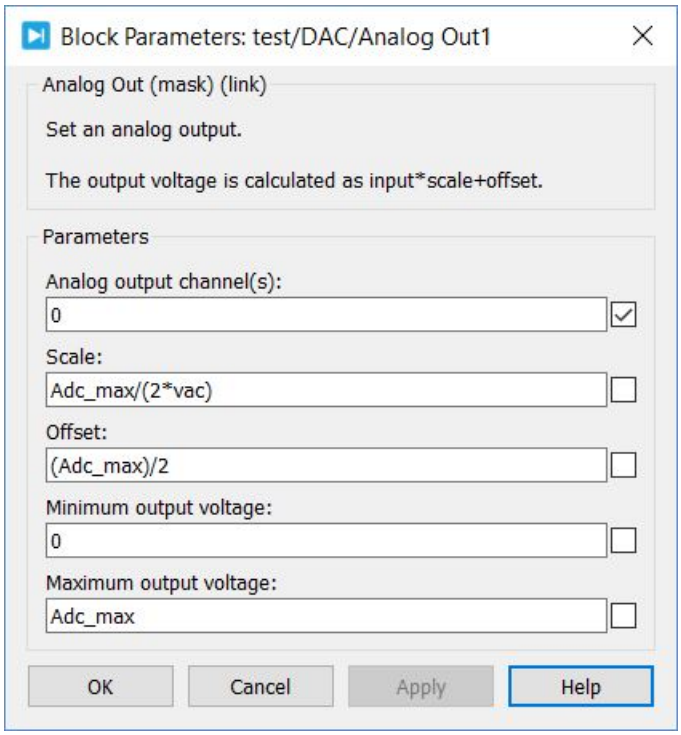
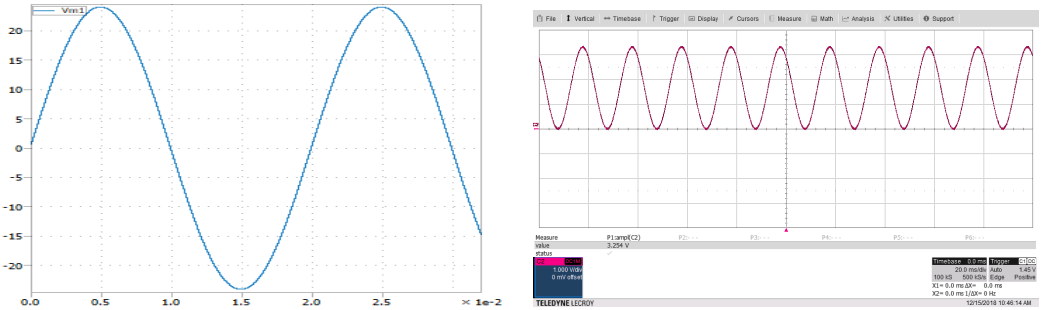


Figura 3.13: Impostazioni Analog Out



(a) Oscilloscopio interno PLECS (b) Oscilloscopio Wave Surfer

Figura 3.14: Forma d'onda tensione

ADC e PWM Capture Ho a questo punto testato la comunicazione tra Nucleo e RT Box inviando prima un segnale analogico (le correnti iabc del motore Microphase) alla Box per visualizzarlo su uno oscilloscopio interno a PLECS tramite il blocco *"Analog In"* e poi un segnale digitale, o meglio i segnali PWM generati dalla Nucleo, con il blocco "PWM Capture". Partendo dalla configurazione presentata in sezione 3.2 ho connesso i pin della scheda inverter IHM08M1 associati alle correnti trifase come indicato sull'user manual [20] ai canali di input analogico da 0 a 2 della RTB. Analogamente ho connesso il blocco *PWM Capture* andando a campionare i tre canali (più i rispettivi "negati") della scheda Nucleo addetti alla generazione del segnale PWM. Le connessioni sono riportate in Figura 3.16.

Ho quindi portato il motore a regime a 500 rpm tramite il controllo I-Hz (vedi sezione 3.3) e visualizzato le correnti in PLECS, ottenendo il risultato di Figura 3.18a. I valori rappresentati sono quelli ottenuti tramite gli shunt della scheda X-NUCLEO-IHM08M1 scalati per essere compatibili con la Nucleo. Come riportato sul manuale [20, pp. 17–18] i valori sono riscalati da una scala di ±30A a una di 0-3.3V tramite uno shunt da 10mΩ. Il valore in Volt misurato ai capi dello shunt è proporzionale alla corrente tramite l'equazione 3.2:

$$\begin{aligned} curr_{fbk} &= \frac{6.8k}{(6.8k+680)} \cdot (1+\frac{4.7k}{1k}) \cdot Vshunt + \frac{680}{6.8k+680} \cdot (1+\frac{4.7k}{1k}) \cdot 3.3V \\ curr_{fbk} &= 5.18 \cdot Vshunt + 1.71V \\ curr_{fbk} &= (0.0518 \cdot i + 1.71)V \end{aligned} \tag{3.2}$$

Invertendo l'equazione 3.2 possiamo quindi risalire al valore di scala e offset da inserire in PLECS per visualizzare la corrente in scala originale. Impostando il blocco *Analog In* come riportato in Figura 3.17 sullo scope interno otteniamo i valori di Figura 3.18b.

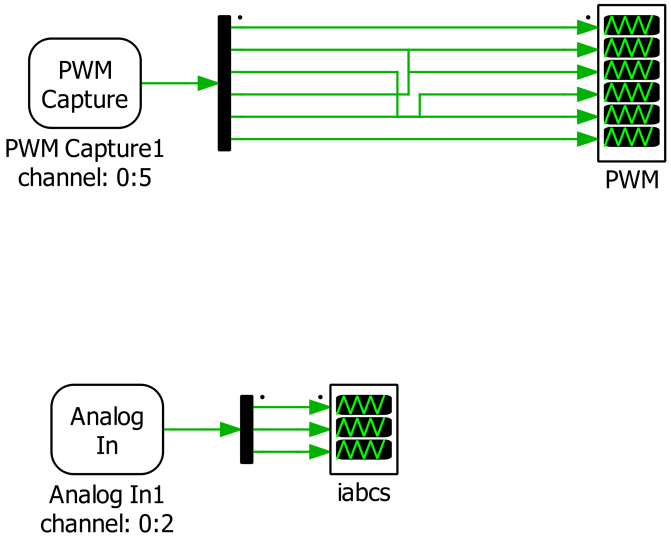


Figura 3.15: Schema PLECS ADC e PWM Capture

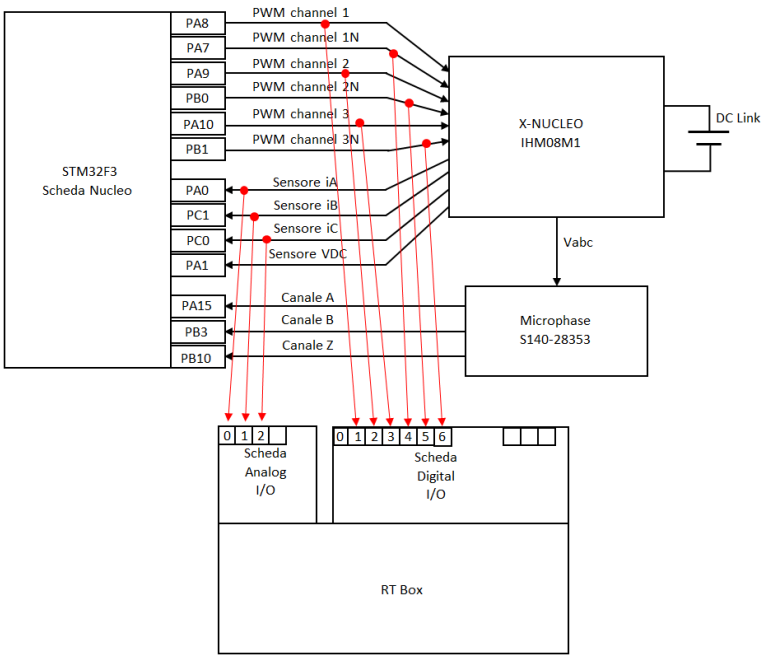


Figura 3.16: Configurazione connessioni

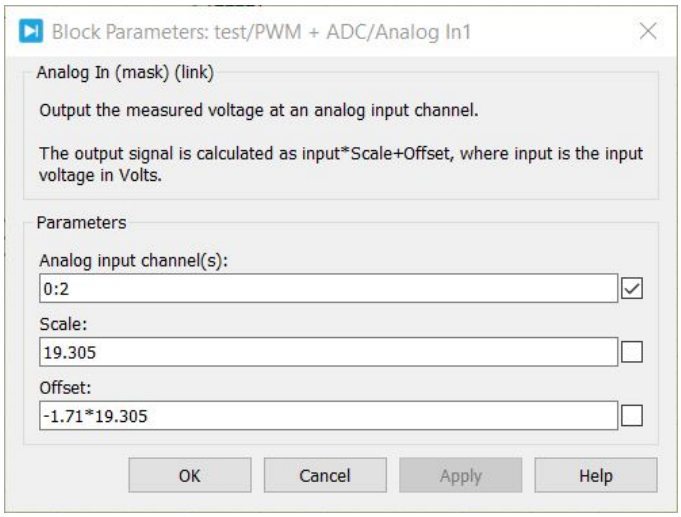
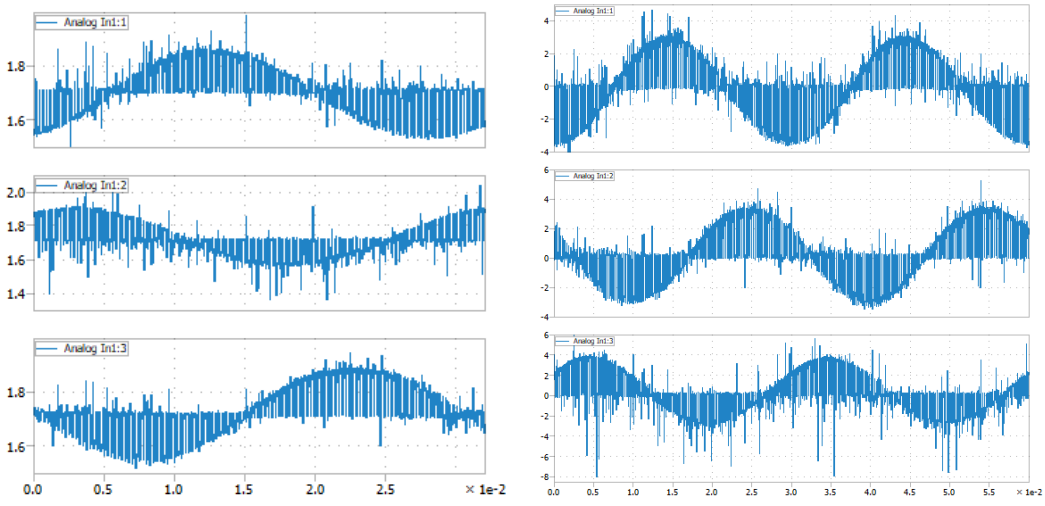


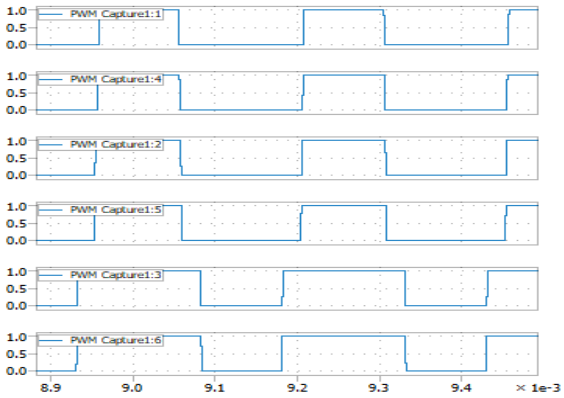
Figura 3.17: Impostazioni Analog Input



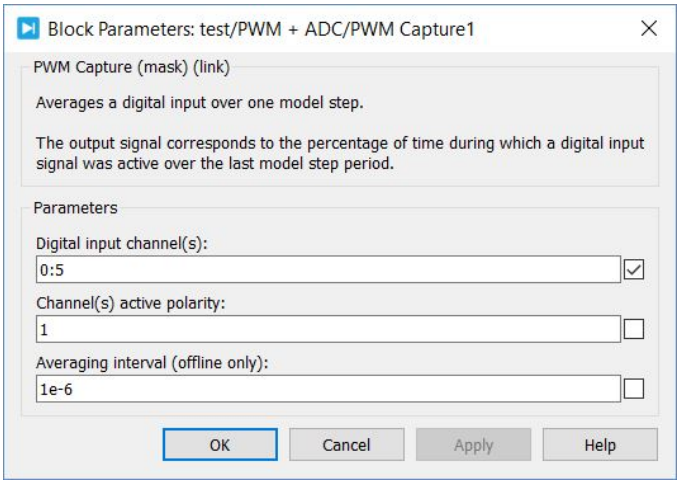
(a) Correnti non scalate (b) Correnti scalate

Figura 3.18: Correnti abc

Per quanto riguarda i segnali digitali invece, come è possibile notare in Figura 3.19a i 6 canali (1:3 normale, 4:6 negato) vengono campionati correttamente. Dal canale negato ci si aspetterebbe un comportamento complementare al canale principale (se ch1 è 1 ch1N è 0) il fatto che non sia così è dovuto alle impostazioni della polarità della gamba della scheda inverter in cui il segnale negato ha polarità inversa (1=OFF, 0=ON). É possibile però impostare la polarità del singolo canale direttamente all'interno del blocco PWM Capture in modo da ricevere sulla RT Box i segnali con la polarità richiesta dall'inverter simulato (Figura 3.19b).



(a) Canali PWM



(b) Impostazioni PWM Capture

Figura 3.19: PWM Capture

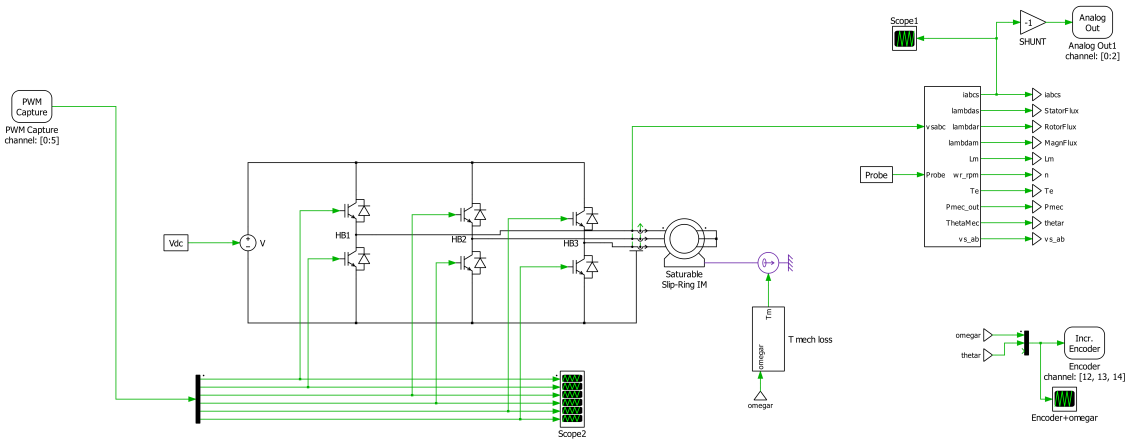


Figura 3.20: Modello RT Box Motore Asincrono

Open Loop Prima di passare al modello del hardware presentato in sezione 3.2 ho deciso di testare un controllo completamente open loop. Ho utilizzato il modello di macchina asincrona presentata nella prima parte di questo elaborato (sottosezione 2.2.2) comandata con un controllo V/Hz (vedi Figura 2.12). Il codice di controllo è stato implementato su scheda Nucleo mentre l'inverter ed il motore sono stati simulati in real time nella Box. I blocchi di interfaccia utilizzati sono ovviamente un *"PWM Capture"*, un *"Analog Out"* utilizzato per monitorare le correnti anche se non necessario per il controllo ed infine il blocco *"Incremental Encoder"*. Come si può notare in Figura 3.20 prima del blocco "Analog Out" di output delle correnti è stato inserito un guadagno pari a "-1" denominato *shunt* atto a simulare il comportamento dalla scheda inverter X-NUCLEO-IHM08M1 che misura le correnti "-iabc", il segno viene corretto nel codice di controllo (codice 3.1).

Il blocco *Incremental Encoder*, a partire dalla pulsazione e dall'angolo meccanico del motore simulato, fornisce su tre output digitali i segnali tipici di un encoder incrementale: canale A, B e Index. Il codice di controllo è stato scritto per l'encoder integrato nel motore microphase caratterizzato da 2048 divisioni. Tra le impostazioni possibili, riportate in Figura 3.21, troviamo anche il numero di divisioni che è stato impostato di conseguenza.

Ho quindi connesso la scheda nucleo direttamente alla RT Box: per quanto riguarda i blocchi *Analog Out* e *PWM Capture* ho utilizzato gli stessi pin del paragrafo 3.4 mentre ho connesso le uscite digitali del blocco *Incremental Encoder* ai pin della Nucleo precedentemente connessi all'encoder del Microphase. Le connessioni sono riportate in Figura 3.22.

Dopo aver ultimato i collegamenti ho quindi caricato il modello sulla RTB ed avviato il controllo sulla scheda nucleo. La simulazione è stata effettuata con uno step-size di 5μs. I risultati coincidono con quelli ottenuti dalle simulazioni riportate nella prima parte dell'elaborato e sono monitorabili tramite gli oscilloscopi interni a PLECS ma anche tramite oscilloscopio virtuale VS_Simple che comunica

direttamente con la scheda Nucleo. Le figure 3.23 e 3.24 consento di confrontare alcuni dati visibili sugli scope di PLECS con il valore letto dalla Nucleo.

```
input.ch0 = (ADC1->JDR1);
input.ch1 = (ADC1->JDR2);
input.ch2 = (ADC1->JDR3);

isabc.a = -(input.ch0-offset_current_a)*scala_current;
isabc.b = -(input.ch1-offset_current_b)*scala_current;
isabc.c = -(input.ch2-offset_current_c)*scala_current;
```

Codice 3.1: Input correnti

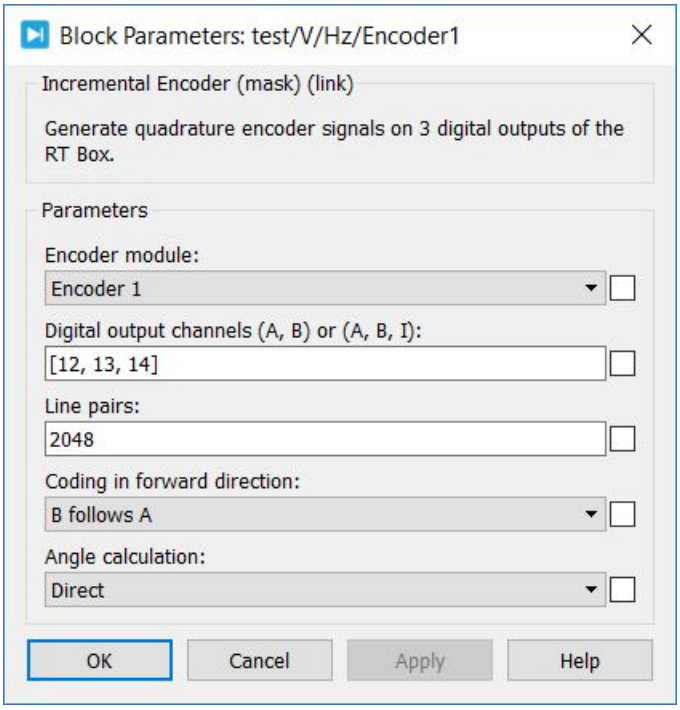


Figura 3.21: Impostazioni Incremental Encoder

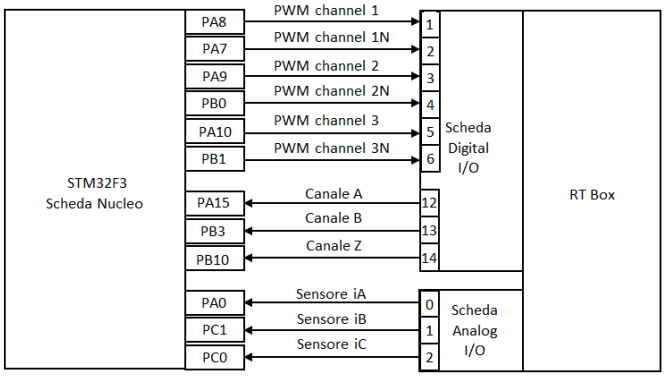
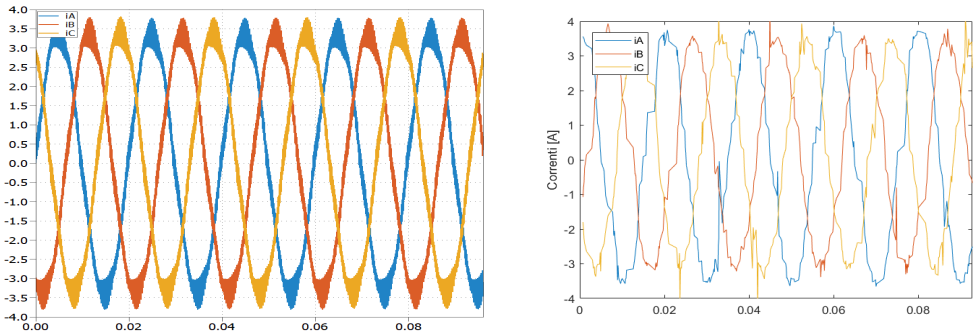
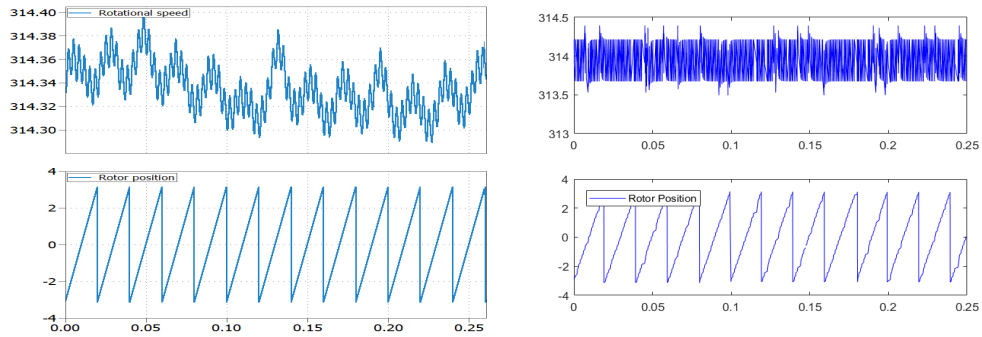


Figura 3.22: Connessione Nucleo e RT Box. Controllo V/Hz



(a) Scope PLECS (b) Oscilloscopio virtuale

Figura 3.23: Correnti iabcs



(a) Scope PLECS (b) Oscilloscopio virtuale

Figura 3.24: Pulsazione ed angolo meccanici

3.5 Modello HiL Motore Microphase

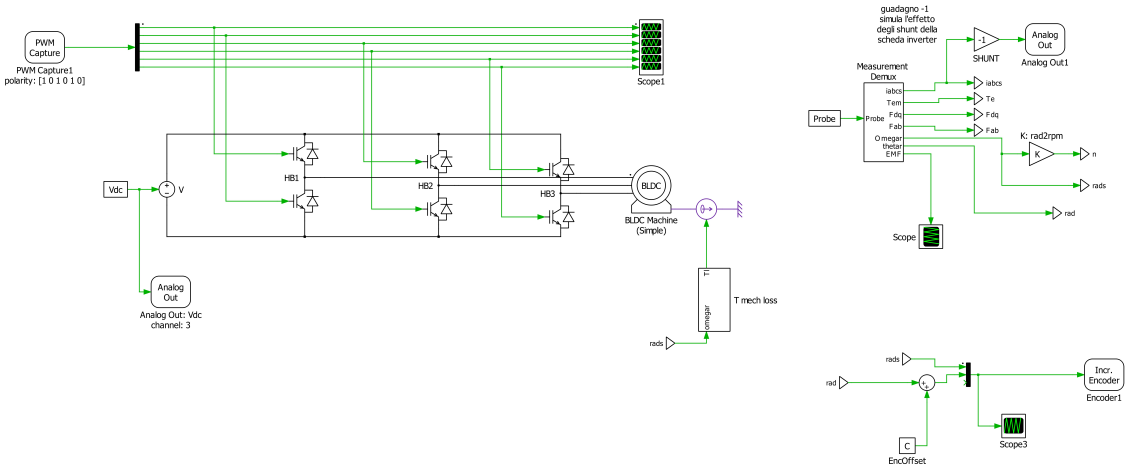


Figura 3.25: Modello per RTB Microphase S140-2B353

Arriviamo finalmente all'obiettivo di questa seconda parte di tesi, ossia la creazione di un modello della configurazione hardware presentata alla sezione 3.2 da usare in un sistema HiL. Vediamo come è stato realizzato.

Partiamo analizzando le impostazioni dei blocchi specifici per la RT Box: i componenti *Analog Out* ed *Encoder* sono invariati, anche in termini di connessioni fisiche, rispetto a quanto presentato nella sezione 3.4. L'unica differenza in termini di connessioni rispetto alla configurazione di Figura 3.22 è l'aggiunta di una connessione analogica introdotta per acquisire sulla scheda nucleo la tensione del DC link (Vdc) di 24 volt. Da evidenziare è il comportamento del blocco *PWM Capture*, come già spiegato al paragrafo 3.4 è infatti possibile impostare la polarità dei singoli canali. La scheda inverter X-NUCLEO-IHM08M1 lavora con i comandi negati con polarità negativa ed essendo il codice di controllo lo stesso usato per la configurazione MCU-Inverter-Motore è necessario impostare il valore attivo dei canali negati a 0. Lo schema delle connessioni analogiche e digitali è riportato in Figura 3.26.

Lo step size è stato impostato a 5μs. In questo modo risulta quindi multiplo della frequenza di commutazione (Tsw=250 μs) ma non è una condizione necessaria per il corretto funzionamento; durante la costruzione del modello sono state fatte alcune simulazioni anche a 3 μs ma l'aggiunta progressiva di componenti ha richiesto di aumentare il passo per evitare di incorrere in un over run, in altre parole aggiungendo elementi alla simulazione i tempi di calcolo aumentano e di conseguenza deve aumentare il passo di simulazione che deve sempre essere maggiore del tempo necessario ad aggiornare le variabili.

L'inverter utilizzato è un inverter ai valori medi. Usando un solver con passo fisso infatti l'effetto di un evento di switch, ma in generale di un qualunque evento,

viene simulato solamente al primo step successivo all'evento stesso; questo porta inevitabilmente ad un errore che in una simulazione offline può essere ridotto diminuendo il passo scelto, come fatto per esempio nel caso dei modelli con inverter fisico (vedi sottosezione 2.2.5), in un simulatore real-time questo non è però possibile dato che lo step size dipende dal tempo di calcolo del simulatore stesso. La soluzione migliore per aumentare la precisione è quindi quella di utilizzare un modello average di inverter come già descritto in sezione 2.2.3 [1].

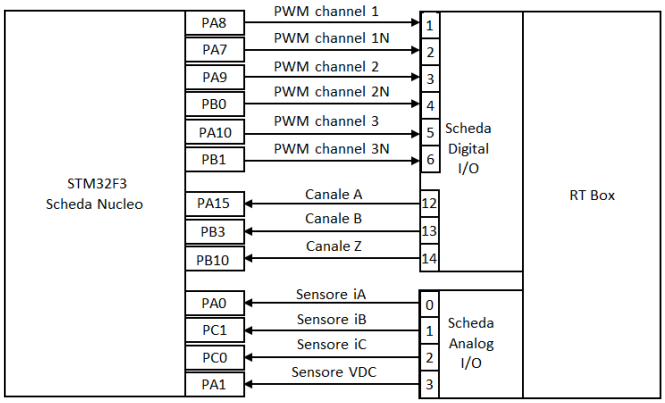


Figura 3.26: Schema connessione HiL

3.5.1 Parametri motore

Nel modello viene usato un motore *BLDC* e non il componente *SPM*, usato nella prima parte dell'elaborato, perché se l'induttanza di statore è indipendente dall'angolo di rotore, ossia se $L_d = L_q$, come in questo caso, risulta computazionalmente più efficiente usare il modello di macchina *"Simplified Brushless DC Machine"* con forza contro-elettromotrice (EMF) sinusoidale e *"Combined stator and mutual inductance"* pari all'induttanza $L_d = L_q$ [6] .

I parametri da inserire nel modello del motore provengono dal datasheet [vedi [11]] e da alcune misure; i valori numerici corrispondenti alle variabili di Figura 3.27 sono riportati nel file di inizializzazione della simulazione (codice 3.2). I valori della resistenza e dell'induttanza presentati sul datasheet sono i valori fase-fase che devono quindi essere divisi per due. La determinazione della costante di Back EMF (*KE*) e dell'inerzia richiedono un'analisi più accurata presentata nelle prossime pagine.

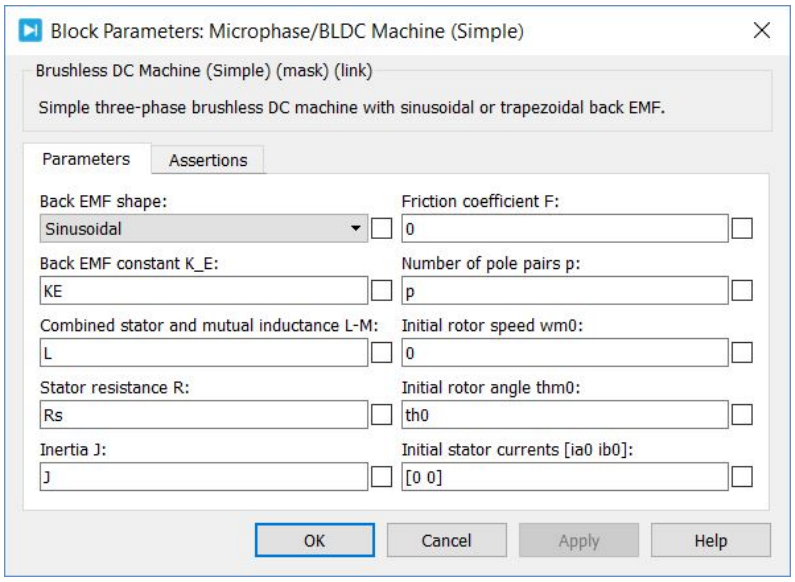


Figura 3.27: Configurazione Motore BLDC HiL

Identificazione KE

Il valore di KE riportato dal datasheet [11] è pari a 3.15 $Vrms/krpm$, il modello di motore BLDC di PLECS richiede però un valore espresso in $volt/(rad/s)$ ricavabile tramite l'equazione 3.3:

$$KE = 3.15\left[\frac{Vrms}{krpm}\right] = \frac{3.15 \cdot \sqrt{2}}{1000} \cdot \frac{30}{\pi} = 0.0426\left[\frac{V}{rad/s}\right] \tag{3.3}$$

Utilizzando però il valore di KE ottenuto con l'equazione 3.3 i duty cycle nel primo test del modello HiL non corrispondevano a quelli ottenuti con lo stesso algoritmo di controllo I-Hz sul motore (vedi figure 3.28a e 3.28b). Ho quindi effettuato una misura delle EMF trascinando il motore a vuoto e misurandone le tensioni concatenate. Le EMF sono risultate, come indicato correttamente nel datasheet, sinusoidali ma i valori rilevati consento di risalire ad un valore di KE di circa 0.0569 $volt/(rad/s)$. Inserendo il valore così ottenuto nel modello HiL il comportamento risulta analogo a quello del motore reale come dimostra la Figura 3.28c.

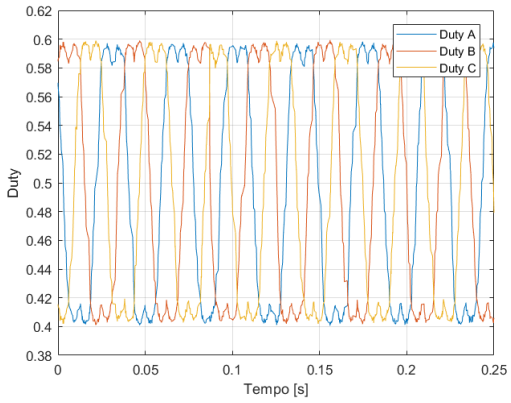
La differenza tra il KE misurato e quello indicato dal datasheet può essere giustificata dalla differenza di temperatura alla quale viene effettuata la misura. Supponendo quindi che il KE indicato sia misurato alla massima temperatura di servizio ammessa dai magneti a NdFeB (150°C) e supponendo una perdita di efficienza pari al $-0.10\%/°C$ possiamo stimare la perdita di efficienza dei magneti

come riportato in equazione 3.4 [17].

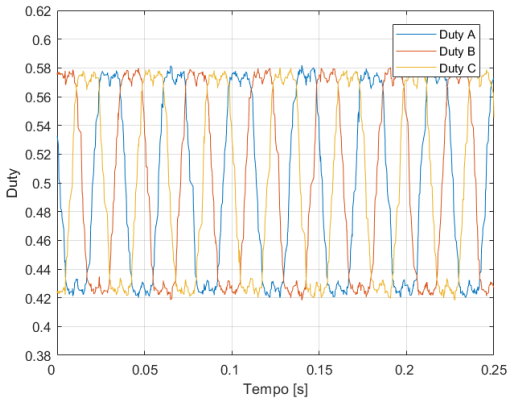
$$\begin{aligned} \Delta T &= T_h - T_{amb} = 150°\text{C} - 25°\text{C} = 125°\text{C} \\ \Delta KE_{\%} &= 125°\text{C} \cdot (-0.10\%/°\text{C}) = -12.5\% \end{aligned} \tag{3.4}$$

- T_h = Temperatura di esercizio a caldo
- T_{amb} = Temperatura di esercizio a freddo

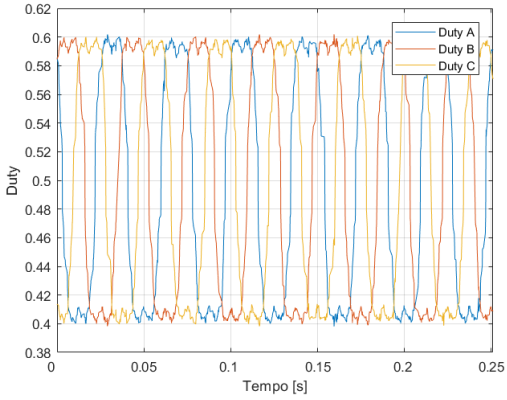
La differenza tra il KE da me misurato e quello indicato sul datasheet corrisponderebbe però a una perdita del 25% circa, doppia quindi a quella stimata. Si può quindi ragionevolmente pensare che ci sia un errore o una scorretta interpretazione nel dato indicato sul datasheet.



(a) S140-2B353



(b) HiL - KE Datasheet



(c) HiL - KE Misurato

Figura 3.28: Duty Cycle a regime, 400 rpm

Identificazione inerzia e componenti d'attrito

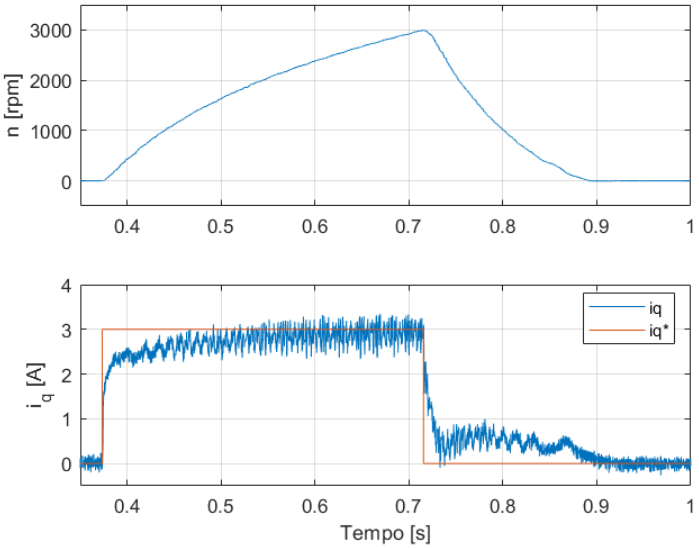


Figura 3.29: Gradino di corrente, S140-2B353

Per quanto riguarda l'inerzia del motore non è possibile utilizzare direttamente il valore indicato sul datasheet (0.06 $kg\,cm^2$) in quanto, come già detto, il controllo del motore è stato implementato in condizione di carico, viene infatti trascinato il motore Microphase S160-1B303 la cui inerzia, secondo il datasheet, è di 0.19 $kg\,cm^2$ [12].

Inoltre bisogna anche considerare la presenza di attriti, per questo motivo ho effettuato una semplice caratterizzazione dell'inerzia del motore iniettando una corrente di riferimento in asse q (i_q^*) ad onda quadra e osservando la risposta in termini di velocità. In particolare sono stati effettuati due test:

1. Viene imposta una corrente $i_q^* = +3$ A fino al raggiungimento di 3000 rpm di velocità per poi annullarla, lasciando quindi rallentare il motore liberamente (fig. 3.29).

2. Viene imposta una corrente di modulo $|i_q^*| = 2.5$ A fino al raggiungimento di 1500 rpm di velocità per poi invertirne il segno, imponendo quindi una coppia di segno opposto a quella precedente (fig. 3.31).

Supponendo, a partire dai valori forniti nei datasheet dei due motori, un'inerzia totale $J = 0.06 + 0.19 = 0.25 \cdot 10^{-4}\ kg\,m^2$ è possibile ricavare una stime della coppia d'attrito totale a partire dall'equazione 3.5.

$$T_{em} - T_{loss} = J \cdot \dot{\omega}_r \tag{3.5}$$

$$T_{em} = \frac{3}{2} KE \cdot i_q = KT \cdot i_q \tag{3.6}$$

- T_{em} = coppia elettromotrice
- T_{loss} = coppia di attrito totale
- $\dot{\omega}_r$ = accelerazione

Selezionato un opportuno intervallo di velocità dal test 1 ho ricavato il valore della coppia elettromotrice media (equazione 3.6) e dell'accelerazione $\dot{\omega}_r = \Delta\omega_r/\Delta t$. Inserendo i valori ottenuti in equazione 3.5 ho stimato una coppia d'attrito complessiva $T_{loss} = 0.1237\,Nm$.

Come già introdotto nella sotto sezione 2.2.1 l'effetto dell'attrito può essere suddiviso in tre componenti: statico, viscoso e di ventilazione. Per stimare i valori di queste tre componenti è stato utilizzato un metodo iterativo che ricostruisce i profili di accelerazione e decelerazione del motore in funzione dei parametri inerziali e li confronta con i valori ottenuti con il test riportato in Figura 3.29. In particolare, ad ogni iterazione viene stimata l'accelerazione in funzione dei valori di inerzia, coefficienti d'attrito e coppia elettromotrice impostati come riferimento e quindi ricavato l'andamento della velocità. L'accelerazione è calcolata come segue:

$$\dot{\omega}_r = \frac{T_{em} - T_{loss}}{J} \tag{3.7}$$

In cui la coppia $\mathrm{T_{loss}}$ è definita come:

$$T_{loss} = T_c + B_m \cdot \omega_r + K_v \cdot \omega_r^2 \qquad [Nm] \tag{3.8}$$

- T_c = coppia attrito statico $[Nm]$
- B_m = coefficiente attrito viscoso $[Nm/(rad/s)]$
- K_v = coefficiente perdite per ventilazione $[Nm/(rad/s^2)]$

Partendo dalla stima iniziale di $\mathrm{T_{loss}}$ vengono variati i parametri J, T_c, B_m e K_v fino ad ottenere una curva di accelerazione e decelerazione stimata il più possibile coincidente con quella ottenuta dai dati del test inerziale. I parametri meccanici finali sono riportati in tabella 3.3 e sono quindi inseriti nel modello (vedi codice 3.2).

Tabella 3.3: Parametri meccanici stimati

Parametro	*Valore*	[*Unità di misura*]
J	$3.1 \cdot 10^{-5}$	$[kgm^2]$
T_c	$8 \cdot 10^{-2}$	$[Nm]$
B_m	$1.5 \cdot 10^{-6}$	$[Nm/(rad/s)]$
K_v	$1.99 \cdot 10^{-6}$	$[Nm/(rad/s^2)]$

La coppia d'attrito totale viene inviata sull'asse del motore come coppia resistente tramite il subsystem *"Tmechloss* il cui schema, simile a quello presentato nella sotto sezione 2.2.1, è riportato in Figura 3.30. A questo punto, ho eseguito il test con onda quadra di corrente sia per il motore che per il modello HiL, i risultati sono riportati in Figura 3.31. La differenza tra l'andamento del motore reale e di quello simulato è minima, i valori ottenuti tramite stima risultano quindi essere adeguati. Considerando l'intervallo di tempo necessario perché il motore vada da 0 a -1500 risulta una differenza complessiva di circa 0.02 s a vantaggio del motore reale.

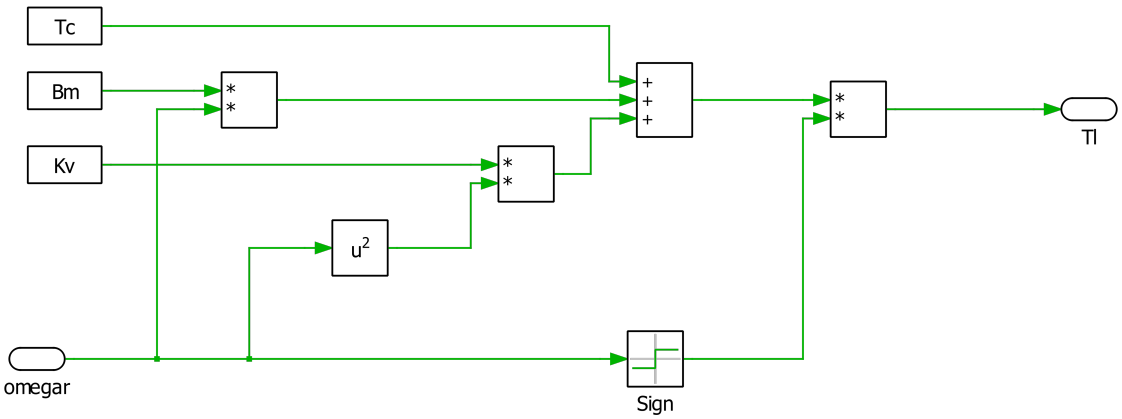
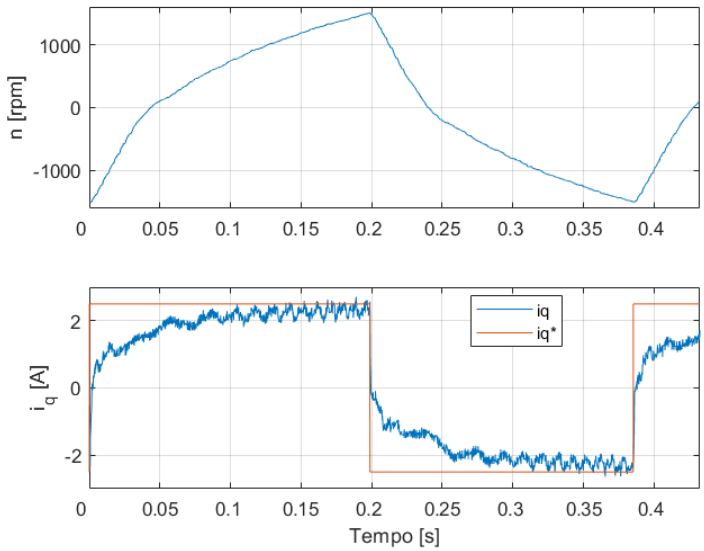
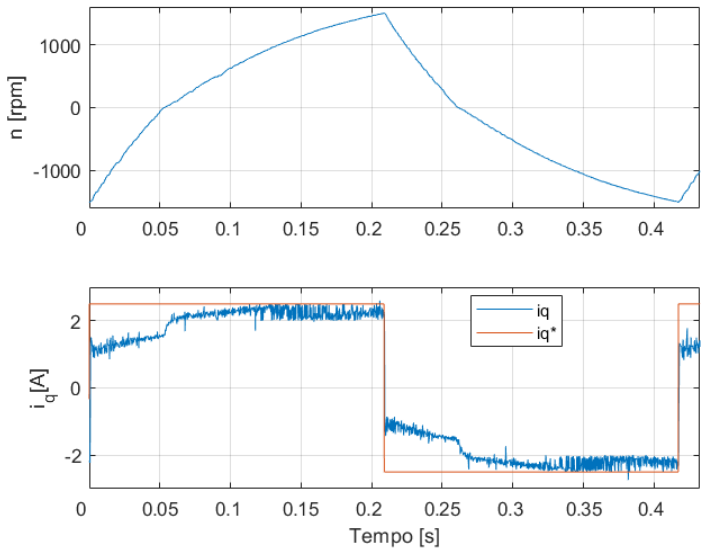


Figura 3.30: Subsystem perdite meccaniche



(a) Microphase



(b) HiL

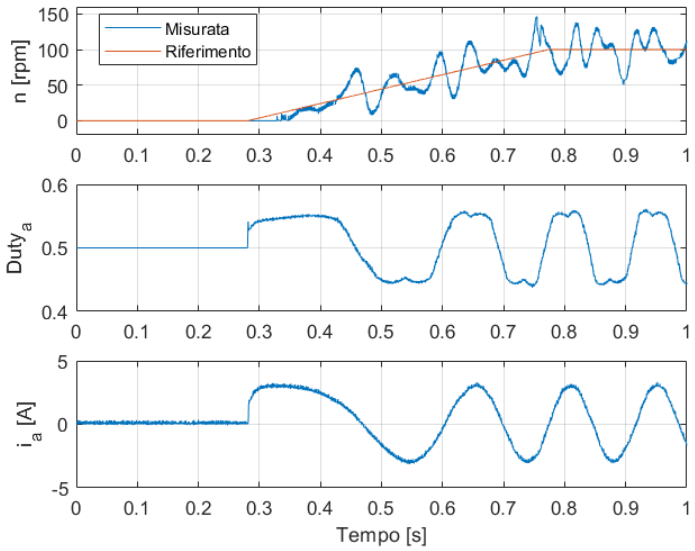
Figura 3.31: Test inerzia, onda quadra di corrente

```
% dc link voltage
Vdc = 24;

% mechanical quantities
encOffset = (rand*2*pi)-pi
th0       = (rand*2*pi)-pi;

J    = 3.1e-5;
Tc   = 8e-2;
Bm   = 1.5e-6;
Kv   = 1.99e-6;

% motor data
Rs   = 0.5/2;
p    = 4;
KE   = 0.0569;
L    = (0.53e-3)/2;
```

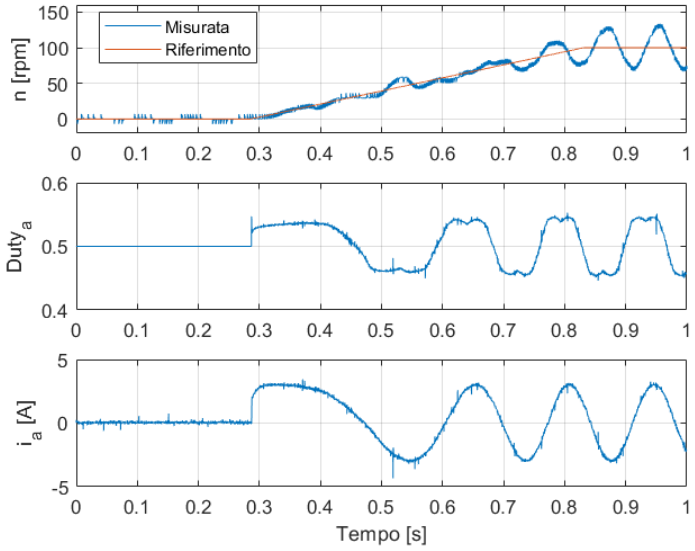
Codice 3.2: Parametri Microphase S140-2B353

3.5.2 Risultati

Nelle prossime pagine vengono infine presentati gli andamenti di alcuni valori significativi della simulazione HiL confrontati con il comportamento del motore fisico, i dati in entrambi i casi sono raccolti tramite oscilloscopio virtuale ed elaborati in Matlab. Come si può osservare il modello HiL presenta un comportamento adeguatamente simile al motore reale anche se con un andamento più ideale per quanto riguarda la risposta in velocità come evidente per esempio sulla rampa di velocità da 0 a 100 rpm in Figura 3.32. La maggior idealità del modello HiL è da ricercarsi nel componente BLDC utilizzato, il quale è a tutti gli effetti un motore ideale, perfettamente sinusoidale e privo, per esempio, di eccentricità o altri difetti costruttivi. Per quanto riguarda le grandezze elettriche, come i duty cycle (fig.3.33) e le correnti (fig.3.34) il comportamento del Hil risulta estremamente simile a quello del motore reale. Si può ottenere quindi un modello complessivamente soddisfacente usando strutture e componenti relativamente semplici; non è da escludersi quindi la possibilità di usare la RT Box per simulare eventualmente sistemi più complessi.

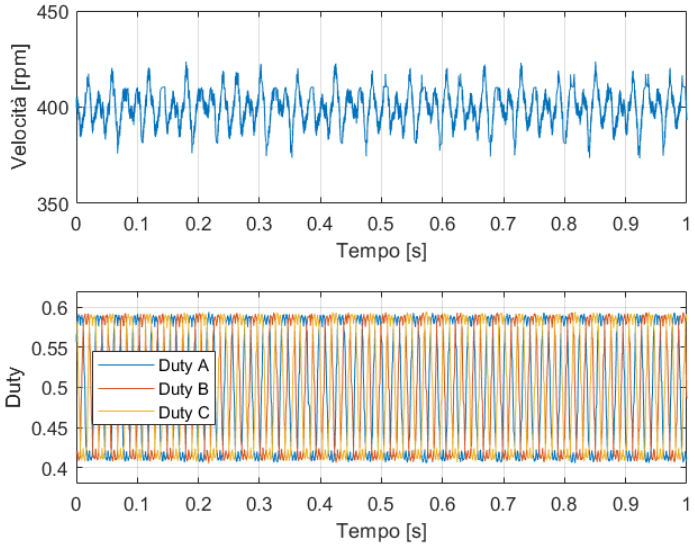


(a) S140-2B353

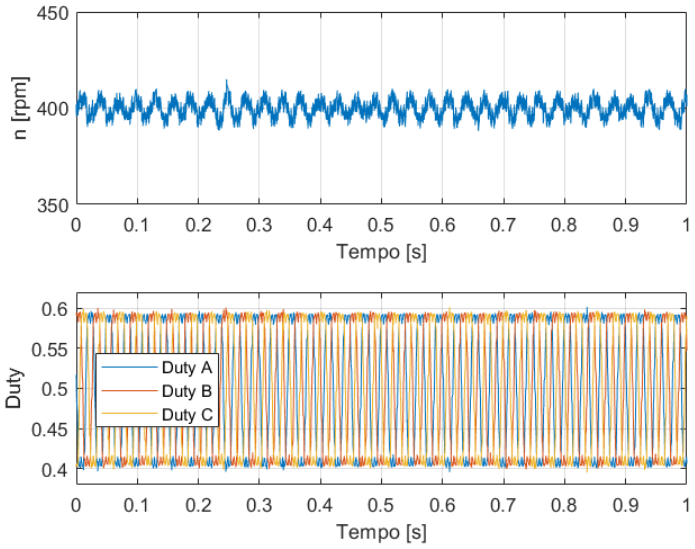


(b) HiL

Figura 3.32: Rampa di velocità da 0 a 100 rpm

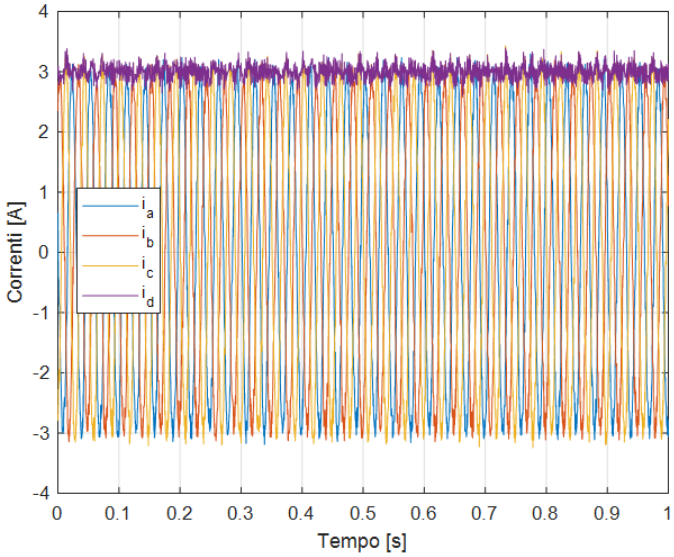


(a) S140-2B353

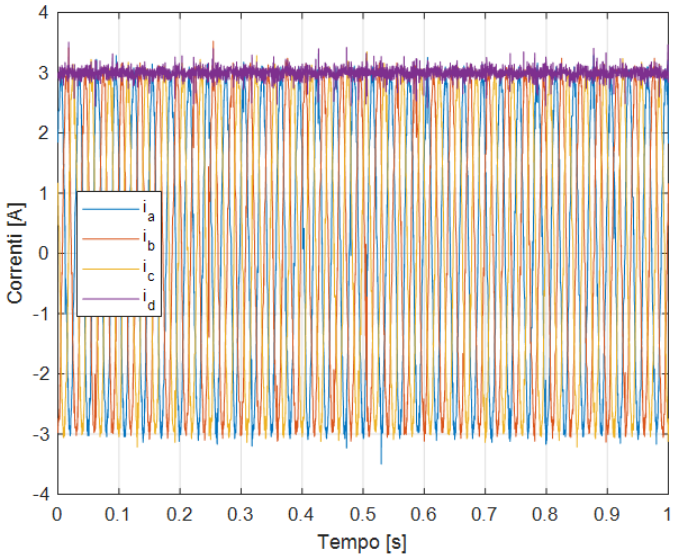


(b) HiL

Figura 3.33: Duty Cylce a regime 400 rpm



(a) S140-2B353



(b) HiL

Figura 3.34: Correnti a regime 400 rpm

Capitolo 4

Conclusioni

In conclusione, valutate le prestazioni sia nella modellistica offline che in quella real-time, posso affermare che PLECS risulta a tutti gli effetti una valida alternativa a Simulink laddove si desideri realizzare un modello usando delle componenti circuitali per convertitore e motore. A parità di risultati infatti l'ambiente PLECS risulta nettamente più veloce in termini di tempi di calcolo.

Pensando allo scopo didattico del corso citato nell'introduzione, ossia quello di introdurre gli studenti allo sviluppo di un controllo digitale, l'unica alternativa valida ai modelli benchmark risulta essere il modello Average in ambiente PLECS, introducendo, qualora sia necessario, anche gli effetti del deadtime con la necessità però di simulare con un passo molto fitto o di utilizzare un solver a passo variabile. Ovviamente, un modello del genere non evidenzia gli effetti delle commutazioni ma consente comunque allo studente di comprendere gli effetti del controllo implementato interfacciandosi con un modello circuitale e sufficientemente simile alla realtà.

Ritengo infine che il grande vantaggio di PLECS sia quello di poter convertire in modo rapido ed intuitivo un qualunque modello nella sua versione HiL. Grazie all'utilizzo dei blocchi RT Box da inserire nel circuito, il passaggio da un modello "offline" a quello real-time può essere fatto in modo graduale consentendo di comprendere le varie componenti del microcontrollore con un introduzione passo passo degli input/output. Inoltre, eliminando il rischio di danneggiamento dell'azionamento a causa di errori nel codice di controllo lo studente può interfacciarsi con diversi tipi di controllo e di motori.

Bibliografia

[1] J. Allmeling e N. Felderer. «Sub-cycle average models with integrated diodes for real-time simulation of power converters». In: *2017 IEEE Southern Power Electronics Conference (SPEC)*. Dic. 2017, pp. 1–6. DOI: 10.1109/SPEC.2017.8333566.

[2] C. L. Baratieri e H. Pinheiro. «An I-F starting method for smooth and fast transition to sensorless control of BLDC motors». In: *2013 Brazilian Power Electronics Conference*. Ott. 2013, pp. 836–843. DOI: 10.1109/COBEP.2013.6785212.

[3] B.K. Bose. *Power electronics and ac drives*. Prentice-Hall, 1986. ISBN: 9780136868828.

[4] K. A. Corzine et al. «An improved method for incorporating magnetic saturation in the q-d synchronous machine model». In: *IEEE Transactions on Energy Conversion* 13.3 (set. 1998), pp. 270–275. ISSN: 0885-8969. DOI: 10.1109/60.707607.

[5] *Materiale PLECS Workshop: Advanced Modeling and Simulation of Power Electronic Systems*. Power Electronics Innovation Center, Politecnico di Torino. Lug. 2017.

[6] Plexim GmbH. *Plecs Help*. Ver. 4.2. 2019.

[7] Plexim GmbH. *PLECS User Manual Version 4.2*. Ago. 2018. URL: www.plexim.com/download/documentation.

[8] Plexim GmbH. *RT Box User Manual*. Apr. 2018. URL: www.plexim.com/download/documentation.

[9] *What Is Hardware-in-the-Loop?* 2017. URL: www.ni.com/white-paper/53958/en/.

[10] C. Kleijn. *Introduction to Hardware-in-the-Loop Simulation*. Controllab Products B.V. URL: www.hil-simulation.com.

[11] Microphase. *S140 series, brushless servomotors - Datasheet*. URL: www.microphase.eu/prodotto/servomotori-brushless-serie-s.

[12] Microphase. *S160 series, brushless servomotors - Datasheet*. URL: www.microphase.eu/prodotto/servomotori-brushless-serie-s.

[13] G. Pellegrino, R. I. Bojoi e P. Guglielmi. «Unified Direct-Flux Vector Control for AC Motor Drives». In: *IEEE Transactions on Industry Applications* 47.5 (set. 2011), pp. 2093–2102. ISSN: 0093-9994. DOI: 10.1109/TIA.2011.2161532.

[14] Gianmario Luigi Pellegrino. *Controllo digitale di convertitori ed azionamenti*. Appunti, dispense ed esercitazioni del corso. Laurea Magistrale in Ingegneria Elettrica, Politecnico di Torino, 2018.

[15] *Supporto Web Plexim*. 2018. URL: https://www.plexim.com/support/solutions/158.

[16] *Sito Web Plexim*. 2018. URL: www.plexim.com.

[17] Juha Pyrhönen, Tapani Jokinen e Valéria Hrabovcová. *Design of Rotating Electrical Machines*. 2009. ISBN: 978-0-470-69516-6.

[18] STMicroelectronics. *UM1724 User manual - STM32 Nucleo-64 boards*. Nov. 2015. URL: www.st.com/en/evaluation-tools/nucleo-f303re.html.

[19] STMicroelectronics. *STL220N6F7 - STripFET F7 Power MOSFET*. Mag. 2017. URL: www.st.com/en/power-transistors/stripfet-f7-series.html.

[20] STMicroelectronics. *UM1996 User manual - Scheda Inverter X-NUCLEO-IHM08M1*. Mag. 2016. URL: www.st.com/en/ecosystems/x-nucleo-ihm08m1.html.