Introduzione

L'esercitazione sperimentale ha luogo nei locali del LADISPE, utilizzando un manipolatore planare a due bracci costruito dalla ditta IMI (USA).

L'esercitazione durerà circa 7 settimane. Ogni gruppo avrà a disposizione una coppia di ore settimanali, che potranno essere incrementate prenotando direttamente in Laboratorio altre ore, secondo le modalità concordate con il docente.

Il manipolatore oggetto dell'esercitazione è comandato da due motori brushless a presa diretta (senza motoriduttori) controllati da un DSP montato su un PC, che gestisce l'interazione uomo-macchina attraverso un programma sviluppato nel corso di una tesi di laurea.

È possibile realizzare algoritmi di controllo digitale con parametri di progetto e tempi di campionamento variabili da parte dell'utente, e scaricarli sul DSP, che si incarica dell'esecuzione degli algoritmi e della gestione dei convertitori A/D e D/A, sotto la supervisione del PC.

Gli studenti, divisi in gruppi, dovranno svolgere le esercitazioni secondo i contenuti specificati nel seguito, e possono presentare una relazione scritta di gruppo, che concorrerà al voto finale di esame, secondo le modalità seguenti: "*La prova orale vale un punteggio massimo di 10 punti; lo studente può presentare o meno la tesina di Laboratorio. Nel primo caso gli sarà rivolta una domanda di carattere teorico seguita da una domanda sulla tesina; nel secondo caso gli saranno rivolte due domande di carattere teorico. Ciascuna domanda vale al massimo 5 punti.*"

La relazione, per coloro che la presentano, dovrà essere scritta in formato *LaTex* oppure *Word* e rispondere ai requisiti fondamentali di una relazione scientifica, illustrati nel volumetto "Saper Comunicare", che si può richiedere al Servizio Studenti.

La relazione dovrà essere presentata al docente prima dell'esame orale a cui si intende partecipare.

Prerequisiti necessari per uno svolgimento proficuo dell'esercitazione sperimentale

- Conoscenza di nozioni di base su DOS e Windows.
- Capacità di lavorare in ambiente Matlab e SIMULINK.
- Capacità di scrivere semplici programmi in C o C++.

Strumenti

Il programma che gestisce il manipolatore si chiama **`IMIROBOT.EXE`**, risiede nella directory **`c:\robot\bin`** ed è presente come icona nel gruppo principale sotto Windows.

Svolgimento

L'esercitazione consta di due parti distinte, anche se collegate, come spiegato nel seguito.

1. Cinematica e pianificazione di traiettorie
2. Controllo del manipolatore

Parte 1
Analisi del Manipolatore Planare IMI
Cinematica - Pianificazione ed Esecuzione di Traiettorie

Scopo

Acquisire pratica dell'hardware e del software di gestione del manipolatore planare; eseguire una traiettoria cartesiana; analizzare e riportare su grafico le variabili relative a tale traiettoria; preparare i campioni del riferimento per far eseguire la traiettoria al manipolatore.

Fase 1 (1 settimana)

Si prende contatto con il programma di gestione del manipolatore.

- Accensione del manipolatore, verifica dello stato dell'hardware e avvio del software IMIROBOT.
- Procedura di "homing" del manipolatore.
- Analisi delle voci del menu di gestione e prove di funzionamento.
- Memorizzazione di traiettorie, secondo le modalità *movimento cartesiano, movimento giunti, movimento assoluto, movimento relativo.*
- Ripetizione delle traiettorie memorizzate, visualizzazione su video, variazione dei parametri dei grafici video.
- Raccolta dati, comprendente posizioni e velocità di riferimento e misurate, valore dei comandi agli attuatori e altri segnali significativi.
- Gestione dei dati così raccolti mediante programmi Matlab, analisi e trattamento dei dati, nel tempo ed eventualmente in frequenza, analisi degli errori; plot delle variabili interessanti nel tempo, nello spazio giunti e nello spazio cartesiano

Fase 2 (1 settimana)

Si utilizza Matlab per costruire i riferimenti per eseguire una traiettoria circolare (centro e raggio parametrici); i riferimenti verranno prima simulati sul modello Simulink del manipolatore, quindi eseguiti dal manipolatore (Fase 3).

Il manipolatore è presentato in Fig. 1; le lunghezze nominali dei bracci sono $L_1 = 0.3683$ m e $L_2 = 0.2413$ m

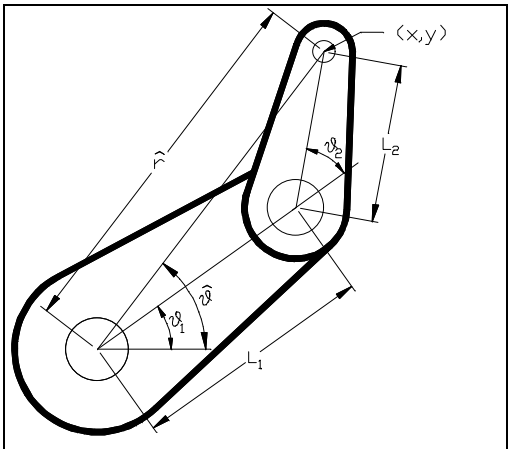

Figura 1
Schema del manipolatore

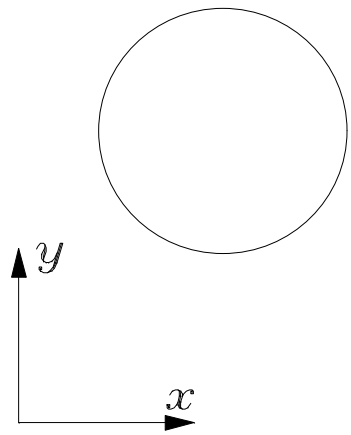

Figura 2
Traiettoria desiderata

Notare che il manipolatore a riposo ($q_1 = 0$, $q_2 = 0$) si colloca in una posizione "verticale" rispetto al tavolo di lavoro (vedi Fig. 3). I sistemi di riferimento sui bracci, il riferimento assoluto e le coordinate *x* e *y* devono essere "meditate con attenzione".

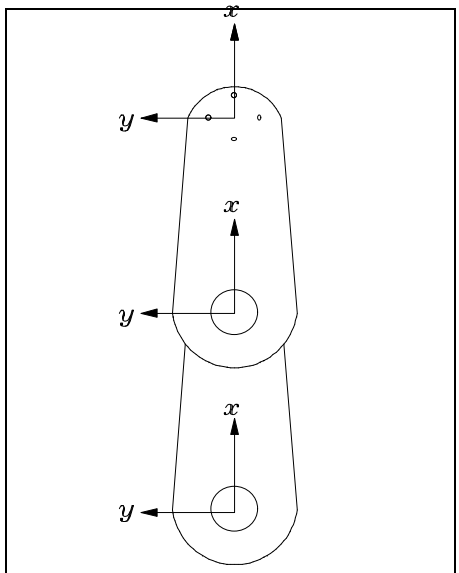

Fig. 3

La fase consiste nel:

1) Pianificare ed eseguire una traiettoria circolare, con centro e raggio parametrici. La pianificazione della traiettoria deve essere fatta predisponendo un programma di tracciamento di un cerchio sul piano *x-y* e quindi generando i campioni del riferimento con la routine **`s_prof.m`**, rispettando specifiche assegnate di velocità e accelerazione; occorre verificare fuori linea i risultati, predisponendo delle prove simulate con Simulink prima di fornire i dati al robot (vedi successivo punto 4). Allo scopo di fornire un esempio, viene fornito il file **`pcerchio.m`**, su cui gli studenti possono basarsi per scrivere le loro procedure.

2) Verificare che durante il movimento non si ottengano configurazioni singolari e che non si raggiungano i fine-corsa, che nel caso del robot reale valgono circa ± 135° per entrambi gli angoli.

3) Salvare su file i tempi di campionamento t_k e i riferimenti $q_i(t_k)$ per utilizzarli come ingressi di riferimento sia per il simulatore (vedi successivo punto 4), sia per il manipolatore reale (vedi successiva Fase 3).

4) Utilizzare il programma di simulazione del manipolatore in Simulink per riprodurre le stesse traiettorie (simulatore in **d:\rob_ind\simula\simula1.m**)

5) Analizzare e riportare su grafico, mediante Matlab, i dati relativi alle coordinate giunto e cartesiane (soprattutto velocità ed errori di traiettoria).

Riassumendo: è possibile procedere secondo i seguenti passi:

a) eseguire da Matlab il comando **`pcerchio`**,dopo aver eventualmente cambiato i parametri del cerchio e della pianificazione (**Attenzione**: verificare che non si vada fuori dallo spazio di lavoro e che il movimento duri più di 2.5 s);

b) eseguire da Matlab il comando **`par_imi`**, che fissa i valori numerici dei parametri del robot e dei guadagni dei controllori;

c) eseguire da Simulink i comandi **`cin_cer1`**, **`cin_cer2`** e **`cin_cer3`**, che permettono di visualizzare la cinematica del cerchio, ossia gli andamenti temporali delle grandezze

pianificate, delle loro velocità e accelerazioni (**Attenzione** a come modificate gli schemi a blocchi);

d) eseguire da Simulink i comandi **`simula1`** e **`simula2`**, che permettono di simulare il comportamento dinamico del robot con gli ingressi calcolati da pcerchio. Aprite pure i blocchi componenti, ma **attenzione** a come li modificate: potrebbe non funzionare più nulla! Nel blocco di dinamica troverete il modello **`coulomb1.m`** dell'attrito coulombiano (**`coulomb2.m`** è identico).

Fase 3 (1 settimana)

Si utilizzano i riferimenti creati nella Fase precedente per eseguire la traiettoria circolare sul manipolatore reale.

ATTENZIONE i riferimenti devono essere in RADIANTI

La fase consiste nel:

1) Fornire in ingresso il file Matlab creato nella fase precedente.
2) Raccogliere i dati utilizzando le funzionalità offerte dal programma **`IMIROBOT.EXE`** utilizzando l'algoritmo di controllo **`custpd`**.
3) Paragonare i dati reali con quelli ottenuti dal programma di simulazione.

Parte 2
Controllo del Manipolatore Planare IMI

Scopo

Simulare e realizzare un algoritmo di controllo digitale da utilizzare per il controllo del manipolatore. Si possono utilizzare ed eventualmente modificare gli algoritmi di controllo già presenti.

Occorre **prestare estrema attenzione** nello sperimentare l'algoritmo di controllo, per non danneggiare voi stessi e il manipolatore.

Fase 1 (1 settimana)

- Analizzare le architetture di controllo di coppia (*torque mode*) e di velocità (*velocity mode*) proprie dei motori NSK, facendo riferimento al manuale del manipolatore (in inglese), al manuale dei motori NSK (in inglese) ed alla tesi degli studenti Ameli & Berruti (in italiano) a disposizione presso il LADISPE.

  NOTA BENE: nell'esercitazione si farà esclusivamente uso del *torque mode*.

- Evidenziare le differenze di impostazione dei due algoritmi di controllo di coppia con e senza l'osservatore di velocità, analizzando i programmi in C contenuti nei file **`custpd`** (controllo di coppia senza osservatore) e **`osspd`** (controllo di coppia con osservatore). Limitarsi a considerare le parti di programma direttamente attinenti agli algoritmi.
- Notare i parametri numerici dei due algoritmi (costanti, guadagni negli algoritmi, guadagni sugli amplificatori, ecc.).
- Procedere ad un'analisi sperimentale, da correlarsi con la successiva simulazione, da articolare nei seguenti punti:

1. utilizzare come traiettoria "campione" uno dei cerchi tracciati nell'esercitazione precedente;
2. acquisire le seguenti grandezze: errore angolare, errore cartesiano, valore dei comandi, più altri segnali significativi a vostra scelta, nei vari sottocasi seguenti:
   - diversi algoritmi (**`custpd`**, **`osspd`**);
   - diverse costanti numeriche degli algoritmi (guadagni, ecc.);
   - diversi periodi di campionamento (attenzione a non fissare periodi di campionamento che rendano instabile il manipolatore);
   - diversi riferimenti, cioè traiettorie con diversa velocità massime.

Fase 2 (1 settimana)

- Utilizzare i programm in Simulink per simulare il comportamento del sistema a dati campionati controllato in coppia (**`simula1.m`**) e mediante osservatore degli stati (**`simula2.m`**), introducendo un attrito coulombiano e viscoso (**`coulomb1.m`** e **`coulomb2.m`**).
- Paragonare i risultati reali con quelli in simulazione. Il programma deve essere in grado di trattare gli stessi file di dati forniti in ingresso al manipolatore (tale richiesta è soddisfatta da **`simula1.m`** e **`simula2.m`**).

Fase 3 (opzionale, se resta tempo)

- Progettare un algoritmo di controllo in "velocity mode" oppure basato sulla dinamica inversa.

Organizzazione dei dati sul PC utilizzato

- Ogni squadra dispone di una directory

  `d:\rob_ind\squadra#`

  in cui salvare i propri files. Il simbolo # varia da 1 a 6 a seconda della squadra di appartenenza (la squadra 5, ad esempio, dovrà usare la directory **`d:\rob_ind\squadra5`**).

  Prima di entrare nell'ambiente del software di gestione del robot (dall'icona Windows o da Dos, con il comando **`c:\robot\imirobot.exe`**), è necessario ridefinire la configurazione della directory di lavoro copiando il proprio file **`robot.cfg`** (che ogni squadra trova già personalizzato nella propria directory) in **`c:\robot\bin`**, sostituendo così un eventuale file pre-esistente.

  Il file **`robot.cfg`** di ogni squadra ha la seguente struttura:

```
d:\rob_ind\squadra#\
d:\rob_ind\squadra#\
d:\rob_ind\squadra#\
d:\rob_ind\servo\
d:\rob_ind\squadra#\
```

- Nella directory **`d:\rob_ind`** si trovano file Matlab di possibile interesse, tra cui **`s_prof.m`** per la generazione di traiettorie di tipo 2-1-2 e **`imisave.m`** per la preparazione di file di campioni di riferimento ai giunti.
- I files a disposizione per il controllo del manipolatore (**`osspd.srv`** e **`custpd.srv`**) si trovano in **`d:\rob_ind\servo`**; entrambi realizzano un controllo ai giunti di tipo PD (Torque Mode): il primo utilizza un osservatore ad alto guadagno per la stima della velocità, il secondo ricostruisce la velocità dei giunti con un'approssimazione del primo ordine (vedere listati).

È VIETATO MODIFICARE I FILES CONTENUTI IN `d:\rob_ind\servo`

Per modificare le leggi di controllo è necessario creare una versione personalizzata dei files CUSTPD.SRV e osspd.srv nella propria directory (magari rinominandoli per evitare confusione!), ricompilarli e modificare concordemente il proprio file di configurazione (la 4ª riga diventa anch'essa **`d:\rob_ind\squadra#\`**), in modo che la propria directory di lavoro diventi la "sede" predefinita dei files di controllo.

- Entrambi i files di controllo offrono la possibilità di visualizzare (e salvare) diverse variabili di interesse, oltre a quelle "standard" definite nel file di grafica **`baselbow.grp`** (che viene caricato automaticamente all'avvio); esse sono indicate come variabili User e sono date da:

  User 1 e 2: velocità dei giunti [rad/s] User 3 e 4: errori di velocità [rad/s]
  User 5 e 6: coppie applicate [Nm] User 7 e 8: errori di posizione [rad]

Materiale di supporto

- Manuale d'uso del software IMIROBOT (in versione cartacea e sotto WWW)
- Manuale originale del manipolatore IMI in inglese (solo versione cartacea) presso Ladispe
- Manuale dei motori NSK (solo versione cartacea) presso Ladispe
- Manuali vari DSP (solo versione cartacea) presso Ladispe
- Tesi Ameli & Berruti (solo versione cartacea) presso Ladispe
- Programmi in C++ di controllo digitale contenuti nei file custpd.srv (controllo di coppia senza osservatore) e osspd.srv (controllo di coppia con osservatore), nella directory **`d:\rob_ind\servo`**
- Software di simulazione Simulink contenuto in **`d:\rob_ind\simula\simula1.m`** e **`d:\rob_ind\simula\simula2.m`**

Norme di sicurezza

Per evitare danni alle persone, lo studente non deve avvicinarsi né lasciare avvicinare altre persone al manipolatore quando il programma **`Imirobot.exe`** è attivo. La prudenza non è mai troppa.
Per evitare danni al manipolatore, si invita lo studente a provare le traiettorie, i nuovi parametri del controllo o nuovi algoritmi in ambiente simulato, prima di applicarli al manipolatore reale. È comunque buona norma preparasi ad attivare il tasto rosso per lo stop di emergenza ogniqualvolta si esegue una nuova operazione sul manipolatore reale.
È opportuno ricordare che l'acquisto del manipolatore è avvenuto con contributi di origine studentesca e quindi un danneggiamento dell'apparato si traduce in uno spreco del vostro denaro.

Listato file `s_prof.m` per generare traiettorie trapezoidali

```
function [s,v,a,k1,k2,n] = s_prof(vmax,amaxp,amaxm,alfa,deltat)

% Uso: [s,v,a,k1,k2,n] = s_prof(vmax,amaxp,amaxm,alfa,deltat)
%
% Calcola i valori discreti del profilo s(k) di tipo 2-1-2
% la velocita` v(k) e l'accelerazione a(k)
% con i vincoli di velocita` massima = vmax
% accelerazione massima = amaxp
% decelerazione massima = amaxm
%
% alfa = percentuale di velocita` massima effettivamente applicata
% deltat = periodo di campionamento
% k1,k2 = valori dei campioni di commutazione
% n = numero totale di campioni s(k), k=1:n

vmax = alfa*vmax;
k = 1;
s(k)=0.0;
v(k)=0.0;
a(k)=amaxp;
iflag = 1;  % primo tratto accelerazione costante
while s(k) < 1.0,
      if iflag == 1
            sr = 1-s(k);
            vp = v(k)+amaxp*deltat;
            if(sr > (v(k)*deltat+0.5*amaxp*deltat^2+0.5*vp^2/amaxm))
                  % ho ancora spazio per un altro passo a velocita` crescente
                  v(k+1) = vp;
                  if(v(k+1) > vmax)
                        % passo a velocita` costante
                        v(k+1) = vmax;
                        a(k+1) = (vmax-v(k))/deltat;
                        s(k+1) = s(k) + v(k)*deltat + 0.5*(vmax-v(k))*deltat;
                        iflag = 2;
                  else
                        % passo a accelerazione costante
                        a(k+1) = amaxp;
                        s(k+1) = s(k) + v(k)*deltat + 0.5*amaxp*deltat^2;
                  end
                  k1 = k;
                  k = k+1;
            else
                  % inizio a decelerare
                  c2=deltat^2;
                  c1=-(amaxm*deltat^2+2*v(k)*deltat);
                  c0=v(k)^2-2*amaxm+2*amaxm*s(k)+2*amaxm*v(k)*deltat;
                  r=roots([c2 c1 c0]);
                  ad=min(r);
                  a(k+1) = -ad;
                  v(k+1) = v(k) - ad*deltat;
                  s(k+1) = s(k) + v(k)*deltat - 0.5*ad*deltat^2;
                  iflag = 3;
                  k2 = k;
                  k = k+1;
            end
      elseif iflag == 2 % secondo tratto velocita costante
            sr = 1-s(k);
            if(sr > (vmax*deltat+0.5*v(k)^2/amaxm))
                  % ho ancora spazio per un altro passo a max velocita`
                  a(k+1) = 0;
                  v(k+1) = vmax;
                  s(k+1) = s(k) + v(k)*deltat;
            else
```

```
                % inizio a decelerare
                c2=deltat^2;
                c1=-(amaxm*deltat^2+2*v(k)*deltat);
                c0=v(k)^2-2*amaxm+2*amaxm*s(k)+2*amaxm*v(k)*deltat;
                r=roots([c2 c1 c0]);
                ad=min(r);
                a(k+1) = -ad;
                v(k+1) = v(k) - ad*deltat;
                s(k+1) = s(k) + v(k)*deltat - 0.5*ad*deltat^2;
                iflag = 3;
          end
          k2 = k;
          k = k+1;
    elseif iflag == 3 % terzo tratto decelerazione costante
          vk2 = v(k) - 2*amaxm*deltat; % calcolo vel due passi in avanti
          if(vk2 > 0) % mi sta bene e continuo a decelerare
                a(k+1) = -amaxm;
                v(k+1) = v(k) - amaxm*deltat;
                s(k+1) = s(k) + v(k)*deltat - 0.5*amaxm*deltat^2;
                k=k+1;
          else  % devo evitare di andare a vel negativa
                a1f = (s(k)+1.5*v(k)*deltat-1)/deltat^2;
                a2f = (1-s(k)-0.5*v(k)*deltat)/deltat^2;
                if((a1f > amaxm)|(a2f > amaxm))
                      a(k+1) = -amaxm;
                      a(k+2) = -amaxm;
                      v(k+1) = v(k) - a1f*deltat;
                      v(k+2) = v(k+1) - a2f*deltat;
                      s(k+1) = s(k) + v(k)*deltat - 0.5*a1f*deltat^2;
                      s(k+2) = s(k+1) + v(k+1)*deltat - 0.5*a2f*deltat^2;
                      k = k+2;
                else
                      a(k+1) = -a1f;
                      a(k+2) = -a2f;
                      v(k+1) = v(k) - a1f*deltat;
                      v(k+2) = v(k+1) - a2f*deltat;
                      s(k+1) = s(k) + v(k)*deltat - 0.5*a1f*deltat^2;
                      s(k+2)= s(k+1) + v(k+1)*deltat - 0.5*a2f*deltat^2;
                      k = k+2;
                end
          end
    end
n = k;
end
```

File `pcerchio.m` per provare il generatore di traiettorie `s_prof.m`

```
% PIANIFICAZIONE DEL CERCHIO;
% GENERAZIONE DEI RIFERIMENTI AI GIUNTI q1 E q2

clear

%  Caratteristiche geometriche del robot (in m)

L1=0.3683;  % lunghezza braccio 1
L2=0.2413;  % lunghezza braccio 2
L=L1+L2;

% costruzione cerchio massimo ammissibile (spazio di lavoro)

m=49;
for i=0:m
   p1_max(i+1)=L*cos(2*pi*i/m);
   p2_max(i+1)=L*sin(2*pi*i/m);
end

% limiti fisici degli angoli giunto (in rad)
```

```
q1_max=135; q1_min=-135; q2_max=135; q2_min=-135;

%%%%% Lettura dati %%%%%

% parametri del cerchio

par_c=[0.12 0.3 0.3]; R=par_c(1); Xc=par_c(2); Yc=par_c(3);

% parametri della pianificazione

par_s=[0.001  0.35  1  1  1.0];  deltat=par_s(1);  vmax=par_s(2);  amaxp=par_s(3);
amaxm=par_s(4); alfa=par_s(5);

%  Pianificazione di s

[s,v,a,k1,k2,npassi] = s_prof(vmax,amaxp,amaxm,alfa,deltat);

% creazione vettore tempi

for i=1:npassi;
      tempo(i)=i*deltat;
end
t_fin=tempo(npassi);

% pianificazione del cerchio;

theta=2*pi*s;
p(1,:)=Xc+R*cos(theta);
p(2,:)=Yc+R*sin(theta);

% cinematica inversa e ricalcolo traiettoria cartesiana (per essere sicuri)

[q] = cinvimi(p,L1,L2,0);
[pnew,teta] = cdirimi(q,L1,L2);

% salvataggio dati su file

[ir,ic]=size(q(1,:));

for i=1:ic,
      qrad(1,i)=grad2rad(q(1,i));
      qrad(2,i)=grad2rad(q(2,i));
end

imisave(qrad(1,:),qrad(2,:),'cerchio.txt');

% plot di varie figure

figure(1);  % angolo q1
plot(tempo,q(1,:));
hold on;
xlabel('tempo'), ylabel('gradi')
title('angolo q1')
grid on
plot(tempo,q1_max*ones(1,npassi),'r');
plot(tempo,-q1_max*ones(1,npassi),'r');
hold off

figure(2);  % angolo q2
plot(tempo,q(2,:));
hold on;
xlabel('tempo'), ylabel('gradi')
title('angolo q2')
grid on
plot(tempo,q2_max*ones(1,npassi),'r');
plot(tempo,-q2_max*ones(1,npassi),'r');
```

```
hold off

figure(3);  % traiettoria ricalcolata e spazio di lavoro
plot(pnew(1,:),pnew(2,:),'b')
hold on;

title('traiettoria x-y')
grid on
xlabel('asse x'), ylabel('asse y')

plot(p1_max,p2_max,'r')      % spazio di lavoro

% calcolo di alcuni parametri utili a definire lo spazio di lavoro cartesiano

e1c_max=L1*cos(grad2rad(q1_max));
e2c_max=L2*cos(grad2rad(q1_max+q2_max));
e1s_max=L1*sin(grad2rad(q1_max));
e2s_max=L2*sin(grad2rad(q1_max+q2_max));

e1c_min=L1*cos(grad2rad(q1_min));
e2c_min=L2*cos(grad2rad(q1_min+q2_min));
e1s_min=L1*sin(grad2rad(q1_min));
e2s_min=L2*sin(grad2rad(q1_min+q2_min));

e2c_ai=L2*cos(grad2rad(q1_max+q2_min));
e2s_ai=L2*sin(grad2rad(q1_max+q2_min));
e2c_ia=L2*cos(grad2rad(q1_min+q2_max));
e2s_ia=L2*sin(grad2rad(q1_min+q2_max));

gamma1=atan2(e1s_max+e2s_max,e1c_max+e2c_max);
gamma2=atan2(e1s_min+e2s_min,e1c_min+e2c_min);

m=50;
rv=sqrt((e1s_max+e2s_max)^2+(e1c_max+e2c_max)^2);
dgamma=(gamma1-gamma2)/(m-1);

for i=1:m
   c1_max(i)=rv*cos(gamma1-dgamma*(i-1));
   c2_max(i)=rv*sin(gamma1-dgamma*(i-1));
end

% hl=line([0 e1c_max e1c_max+e2c_max],[0 e1s_max e1s_max+e2s_max]);
% set(hl,'Color','red')
% hl=line([0 e1c_min e1c_min+e2c_min],[0 e1s_min e1s_min+e2s_min]);
% set(hl,'Color','red')

plot(c1_max,c2_max,'r')      % vincoli x-y

hl=line([0.1 0 0],[0 0 0.1]); % assi x-y
set(hl,'Color','blue')

axis('equal')

% plot delle variabili curvilinee  s, v, a

figure(4);
plot(tempo,s,'b')

hold on;
plot(tempo,v,'g')
plot(tempo,a,'r')
grid on
title('parametri pianificazione')
xlabel('tempi'), ylabel('s, v, a')
hold off
```

File `par_imi.m` per assegnare i parametri del manipolatore

```
% PARAMETRI MANIPOLATORE IMI
% Far eseguire prima di iniziare le simulazioni in Simulink del manipolatore IMI
d2v=0.005859;     % digits to volt
v2d=204.8;        % volt to digits
r2c=24446.19926;  % radiants to counts
c2r=0.00004091;   % counts to radiants

Kb=19.3;          % guadagno base
Ke=3.12;          % guadagno elbow
K1=1.5;
VG1=2;            % Guadagni amplificatori
VG2=3;

I1=.267;        % Inerzia rotore 1
I2=0.334;       % Momento centrale di inerzia braccio 1
I3=0.0075;      % Inerzia rotore 2
I3c=0.04;       % Inerzia statore 2
I4=0.063;       % Momento centrale di inerzia braccio 2
Ip=0;           % Momento di inerzia del carico
M1=73;          % Massa motore 1
M2=9.78;        % Massa braccio 1
M3=14;          % Massa motore 2
M4=4.45;        % Massa braccio 2
Mp=0;           % Massa carico
L1=0.3683;      % Lunghezza braccio 1
L2=0.2413;      % Lunghezza braccio 2
L3=0.136;       % Distanza dall' asse di rotazione del CG braccio 1
L4=0.102;       % Distanza dall' asse di rotazione del CG braccio 2
f1=5.3;         % axis1 friction coefficient
f2=1.1;         % axis2 friction coefficient

% par matrice dinamica

p1=I1+I2+I3c+I3+I4+Ip+(M3+M4+Mp)*L1^2+M2*L3^2+M4*L4^2+Mp*L2^2;
p2=I3+I4+Ip+M4*L4^2+Mp*L2^2;
p3=M4*L1*L4+Mp*L1*L2;

Kp1=0.04;   % guadagno posizione giunto 1 controllore PD
Kp2=0.017;  % guadagno posizione giunto 2 controllore PD
Kd1=2;      % guadagno velocita giunto 1 controllore PD
Kd2=1;      % guadagno velocita giunto 2 controllore PD
```

File `coulomb.m` per modellare l'attrito coulombiano

```
function [sys,x0]=coulomb(t,x,u,flag,p1,p2,p3,f1,f2);
% u(1) = q1
% u(2) = qpunto1 ovvero omega1
% u(3) = tau 1
% u(4) = q2
% u(5) = qpunto2 ovvero omega2
% u(6) = tau 2

if abs(flag)==3,

  soglia=0;

  if u(2)>soglia ,
     attrito1=f1;
  elseif u(2)<-soglia,
     attrito1=-f1;
  else
     if u(3)>f1,attrito1=f1;
        elseif u(3)<-f1,attrito1=-f1;
        else attrito1=u(3);
     end
```

```
  end

  if u(5)>soglia ,
     attrito2=f2;
  elseif u(5)<-soglia,
     attrito2=-f2;
  else
     if u(6)>f2,attrito2=f2;
        elseif u(6)<-f2,attrito2=-f2;
        else attrito2=u(6);
    end
  end

  sys=(p2*(u(3)+u(5)*(2*u(2)+u(5))*p3*sin(u(4))-attrito1)...
    -(p2+p3*cos(u(4)))*(u(6)-u(2)^2*p3*sin(u(4))-attrito2))...
    /((p1+2*p3*cos(u(4)))*p2-(p2+p3*cos(u(4)))^2);

elseif abs(flag)==0,

  sys=[0;0;1;6;0;0];

else

  sys=[];

end
```

Come fornire da file una traiettoria di riferimento ai giunti

Per assegnare in ingresso al manipolatore reale una particolare traiettoria di riferimento ai giunti, è necessario preparare un file di tipo testo (con estensione `.txt`, ad esempio **`cerchio.txt`** nel file **`pcerchio.m`**), contenente:

- sulla prima colonna il contatore dei campioni (1, 2, ecc.) in formato *intero*;
- sulla seconda e sulla terza colonna i valori di riferimento delle posizioni dei giunti 1 e 2 rispettivamente, salvati in *floating point* ogni millisecondo.
- Esempio:

```
1   0.000000   0.000000
2   0.001473  -0.000243
3   0.002946  -0.000487
4   0.004419  -0.000730
5   0.005891  -0.000974
6   0.007364  -0.001217
7   0.008837  -0.001461
8   0.010310  -0.001704
9   0.011782  -0.001947
10  0.013255  -0.002191
```

Una volta calcolati in Matlab i vettori dei valori di riferimento dei giunti (ad esempio in **`pcerchio.m`**), è possibile generare il file nel formato richiesto, utilizzando il comando **`imisave`** a disposizione.

La sintassi di tale comando è spiegata all'inizio del file di funzione **`imisave.m`**, di cui si riporta il listato:

File `imisave.m`

```
function s=imisave(q1,q2,dati)

% Il comando
%
%                  imisave(q1,q2,'dati.txt');
%
% salva i vettori delle variabili giunto q1 e q2 nel vettore DATI.TXT
```

```
% nel formato adatto per l'utilizzo del comando FROM FILE del menu MOVE.
% ATTENZIONE: q1 e q2 devono essere vettori RIGA.

[r1,c1]=size(q1);
[r2,c2]=size(q2);

if c1==c2,
      t=1:c1;
      y=[t;q1;q2];
      fp=fopen(dati,'w');
      fprintf(fp,'%d %f %f\n',y);
      fclose(fp);
      s=1;
else
disp('WARNING: I vettori devono avere la stessa lunghezza!')
s=0;
end
```

Controllo di coppia PD con osservatore ad alto guadagno

- Modello dell'osservatore utilizzato (realizzato a tempo discreto in **`osspd.srv`**):

$$\dot{\hat{\boldsymbol{x}}}_1 = \hat{\boldsymbol{x}}_2 + \frac{1}{\mathrm{e}}\boldsymbol{H}_{\mathrm{p}}(\boldsymbol{y} - \hat{\boldsymbol{x}}_1)$$

$$\dot{\hat{\boldsymbol{x}}}_2 = \frac{1}{\mathrm{e}^2}\boldsymbol{H}_{\mathrm{v}}(\boldsymbol{y} - \hat{\boldsymbol{x}}_1)$$

ove $\boldsymbol{y}$ rappresenta la misura delle posizioni angolari dei giunti acquisita dal DSP, $\hat{\boldsymbol{x}}_1$ la stima delle posizioni angolari e $\hat{\boldsymbol{x}}_2$ la stima delle velocità angolari dei giunti.

Listato C di `osspd.srv`

```
/* OSSERVATORE 1 con posizione misurata*/

/* Parametri controllo Kp1=2,Kd1=0.04,Kp2=1,Kd2=0.017 */
/* Parametri osservatore Hp1=7,Hv1=2,Hp2=6.2,Hv2=4 */

#include <cntrl.h>
#include <math.h>

float ts=0.001;
float c2r=0.00004091,r2c=24446.19926 ;
float v2c=204.8 ;

float VF_kp1=2, VF_kp2=1;
float VF_kd1=0.04, VF_kd2=0.017;
float VF_hv1=6, VF_hv2=4;
float VF_hp1=7, VF_hp2=6.2;
float VF_eps=0.002;

int VI_Azzeramento=0;
float TF_velerr1[5000];
float TF_velerr2[5000];
float TF_avel1[5000];
float TF_avel2[5000];
float TF_err1[5000];
float TF_err2[5000];
float TF_u1[5000];
float TF_u2[5000];
float TF_pos1[5000];
float TF_pos2[5000];
float x11, x12;
float old1x11, old1x12, old2x11, old2x12;
```

```
float old1pos1, old1pos2;
float old2pos1, old2pos2;
float x21, old1x21, old2x21, old3x21;
float x22, old1x22, old2x22, old3x22;
float oldref1, oldref2;
float errpos1, errpos2;
float errvel1, errvel2 ;
float vel1, vel2;
float Da_to_Volts;
int i = -1;
int k = -1;

init_control()
   {

   vel1=0.0;
   vel2=0.0;
   errvel1=0.0;
   errvel2=0.0;
   errpos1=0.0;
   errpos2=0.0;
   old1pos1=0.0;
   old1pos2=0.0;
   old2pos1=0.0;
   old2pos2=0.0;
   x21=0.0;
   x22=0.0;
   old1x21=0.0;
   old1x22=0.0;
   old2x21=0.0;
   old2x22=0.0;
   oldref1=0.0;
   oldref2=0.0;
   Da_to_Volts=10/2048;
   }

control()
   {

   if (VI_Azzeramento==1)
       {
       for (i=0;i<5000;i++)
         {
            TF_velerr1[i]=0.0;
            TF_velerr2[i]=0.0;
            TF_avel1[i]=0.0;
            TF_avel2[i]=0.0;
            TF_err1[i]=0.0;
            TF_err2[i]=0.0;
            TF_u1[i]=0.0;
            TF_u2[i]=0.0;
            TF_pos1[i]=0.0;
            TF_pos2[i]=0.0;
         }
       VI_Azzeramento=0;
       i=-1;
       }

   x21=0.8964*old1x21-0.0302*old2x21+129.0501*old1pos1-129.0501*old2pos1;
   x22=0.7586*old1x22-0.045*old2x22+265.4585*old1pos2-265.4585*old2pos2;

   errpos1=(float)(ref1-pos1);
   errpos2=(float)(ref2-pos2);
   errvel1=(float)(((ref1-oldref1)/ts)-x21);
   errvel2=(float)(((ref2-oldref2)/ts)-x22);

   vel1=x21;
```

```
   vel2=x22;

   u1=(int)(VF_kp1*errpos1+VF_kd1*errvel1);
   u2=(int)(VF_kp2*errpos2+VF_kd2*errvel2);

/* grandezze user disponibili all'utente */

   User1=(float)vel1*c2r;
   User2=(float)vel2*c2r;
   User3=(float)errvel1*c2r;
   User4=(float)errvel2*c2r;
   User5=(float)(u1*Da_to_Volts*19.3);
   User6=(float)(u2*Da_to_Volts*3.12);
   User7=(float)errpos1*c2r;
   User8=(float)errpos2*c2r;

   if (i<5000)
   {
   i++;

   TF_u1[i]=0.0;
   TF_u2[i]=0.0;
   TF_pos1[i]=0.0;
   TF_pos2[i]=0.0;
   TF_avel1[i]=vel1*c2r;
   TF_avel2[i]=vel2*c2r;
   TF_err1[i]=errpos1*c2r;
   TF_err2[i]=errpos2*c2r;
   TF_u1[i]=(float)(u1*Da_to_Volts*19.3);
   TF_u2[i]=(float)(u2*Da_to_Volts*3.12);
   TF_velerr1[i]=errvel1*c2r;
   TF_velerr2[i]=errvel2*c2r;
   TF_pos1[i]=pos1;
   TF_pos2[i]=pos2;
   }

oldref1=ref1;
oldref2=ref2;

old2pos1=old1pos1;
old2pos2=old1pos2;
old1pos1=pos1;
old1pos2=pos2;


old2x11=old1x11;
old2x12=old1x12;
old1x11=x11;
old1x12=x12;

old2x21=old1x21;
old2x22=old1x22;
old1x21=x21;
old1x22=x22;

   }
```

Controllo di coppia PD con approssimazione della velocità del primo ordine

Listato C di `custpd.srv`

```
#include <cntrl.h>

float ts=0.0035;
float c2r=0.00004091,r2c=24446.19926 ;
float v2c=204.8 ;

float VF_kp1=2, VF_kp2=1;
float VF_kd1=0.04, VF_kd2=0.017;

int VI_Azzeramento=0;
float olderr1, olderr2, T1;
float TF_velerr1[5000];
float TF_velerr2[5000];
float TF_avel1[5000];
float TF_avel2[5000];
float TF_err1[5000];
float TF_err2[5000];
float TF_u1[5000];
float TF_u2[5000];
float TF_pos1[5000];
float TF_pos2[5000];
float oldp1, oldp2;
float oldref;
float p1,p2;
float derr1, derr2 ;
float vel1, vel2;
float Da_to_Volts;
int i = -1;
int k = -1;

init_control()
   {

   p1=0.0;
   p2=0.0;
   vel1=0.0;
   vel2=0.0;
   derr1=0.0;
   derr2=0.0;
   olderr1=0.0;
   olderr2=0.0;
   oldp1=0;
   oldp2=0;
   oldref=0.0;
   Da_to_Volts=10.0/2048.0;
   }

control()
   {

   if (VI_Azzeramento==1)
       {
       for (i=0;i<5000;i++)
         {
            TF_velerr1[i]=0.0;
            TF_velerr2[i]=0.0;
            TF_avel1[i]=0.0;
            TF_avel2[i]=0.0;
            TF_err1[i]=0.0;
```

```
            TF_err2[i]=0.0;
            TF_u1[i]=0.0;
            TF_u2[i]=0.0;
            TF_pos1[i]=0.0;
            TF_pos2[i]=0.0;
          }
        VI_Azzeramento=0;
        i=-1;
        }

   olderr1=err1 ;
   olderr2=err2 ;
   err1=(float)(ref1 - pos1);
   err2=(float)(ref2 - pos2);
   derr1=(err1 - olderr1)/ts ;
   derr2=(err2 - olderr2)/ts ;

   vel1=(pos1-oldp1)/ts;
   vel2=(pos2-oldp2)/ts;

   u1=(int)(VF_kp1*err1+VF_kd1*derr1);
   u2=(int)(VF_kp2*err2+VF_kd2*derr2);

/* grandezze user disponibili all'utente */

   User1=(float)(vel1*c2r);
   User2=(float)(vel2*c2r);
   User3=(float)(derr1*c2r);  /* errore di velocita' braccio 1 */
   User4=(float)(derr2*c2r);  /* errore di velocita' braccio 2 */
   User5=(float)(u1*Da_to_Volts*19.3);
   User6=(float)(u2*Da_to_Volts*3.12);
   User7=(float)(err1*c2r);
   User8=(float)(err2*c2r);

   if (i<5000)
   {
   i++;
/*
   TF_rifpos[i]=ref1*c2r;
   TF_rifvel[i]=c2r*(ref1-oldref)/ts;
*/
   TF_u1[i]=0.0;
   TF_u2[i]=0.0;
   TF_pos1[i]=0.0;
   TF_pos2[i]=0.0;
   TF_avel1[i]=vel1*c2r;
   TF_avel2[i]=vel2*c2r;
   TF_err1[i]=err1*c2r;
   TF_err2[i]=err2*c2r;
   TF_u1[i]=(float)(u1*Da_to_Volts*19.3);
   TF_u2[i]=(float)(u2*Da_to_Volts*3.12);
   TF_velerr1[i]=derr1*c2r;
   TF_velerr2[i]=derr2*c2r;
   TF_pos1[i]=pos1;
   TF_pos2[i]=pos2;
   }

oldp1=pos1;
oldp2=pos2;
oldref=ref1;
   }
```