Cinematica del subriflettore di SRT

F. Buffa

Osservatorio Astronomico di Cagliari
fbuffa@ca.astro.it

Cinematica di M2

Introduzione

La movimentazione cinematica del subriflettore (M2) del Sardinia Radio Telescope avviene attraverso la composizione dei movimenti di sei attuatori che conferiscono al sistema sei gradi di libertà. gli attuatori verranno guidati in retroazione attraverso il sistema di metrologia laser previsto (cfr. TM-1385-005 Laser Metrology). Nel presente studio si propone un semplice modello dei moti di M2 e quindi un'analisi della catena cinematica nella quale ogni attuatore può essere decomposto. Queste pagine costituiscono anche una guida essenziale all'utilizzo della libreria hexlib da utilizzarsi per il controllo della movimentazione del subriflettore. La fig.1 schematizza la geometria del sistema: nella notazione usata i sei attuatori risultano

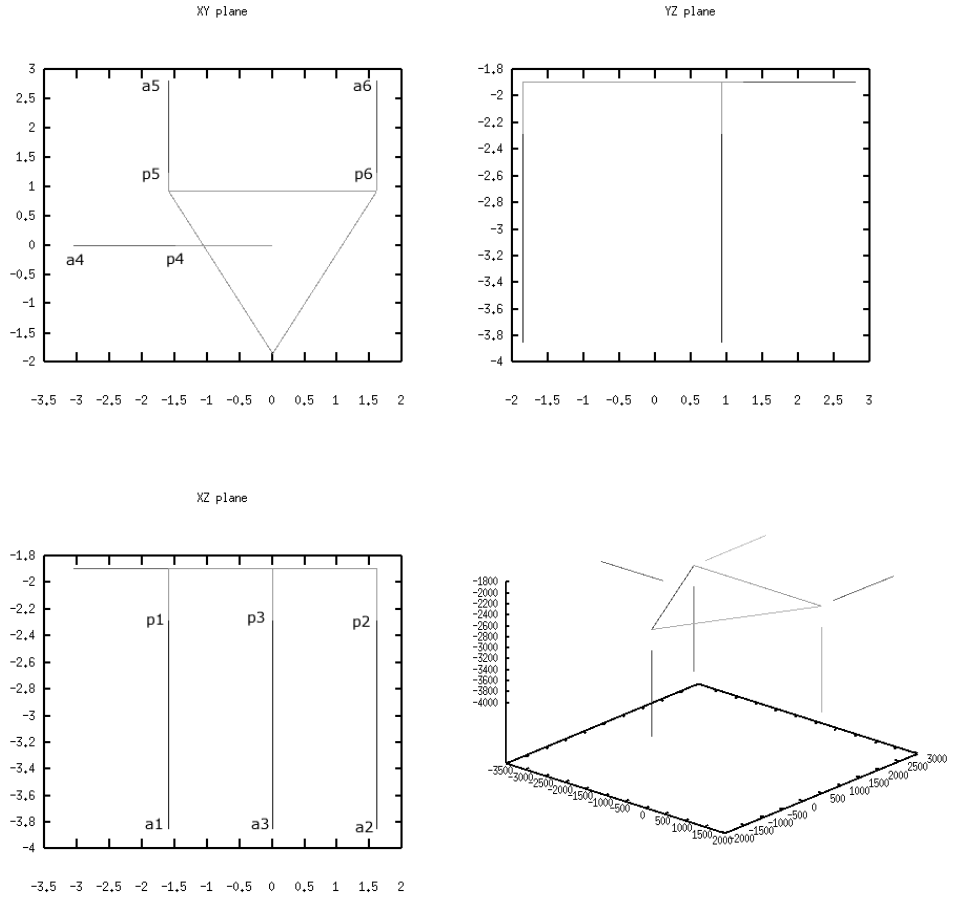

Figura 1: xxxx

identificati da coppie di punti riferiti all'origine convenzionale degli assi posta nel vertice di M2.

attuatore	tipo	x_p	y_p	z_p	x_a	y_a	z_a
a_1p_1	mov z	-1602.0	924.9	-2287.6	-1602.0	924.9	-3857.6
a_2p_2	mov z	1602.0	924.9	-2287.6	1602.0	924.9	-3857.6
a_3p_3	mov z	0.0	-1849.8	-2287.6	0.0	-1849.8	-3857.6
a_4p_4	mov x	-1474.0	0.0	-1893.6	-3044.0	0.0	-1893.6
a_5p_5	mov y	-1602.0	1230.9	-1893.6	-1602.0	2800.9	-1893.6
a_6p_6	mov y	1602.0	1230.9	-1893.6	1602.0	2800.9	-1893.6

Tabella 1: xxxxxxxxxxxxx

Studio cinematico

Si è studiata la mappatura delle elongazioni degli attuatori nella corrispondente roto-traslazione riferita ad un sistema cartesiano e conseguentemente si è studiata la trasformazione inversa che data una roto-traslazione permette di conoscere le elongazioni. Assumendo come sistema di riferimento convenzionale quello di fig.1, in virtù di una generica movimentazione degli attuatori i punti a_i risultano vincolati (tramite i cardani) alla struttura, mentre i punti p_i assumono nuove coordinate p'_i definite dalla trasformazione:

$$p'_i = [x'_i, y'_i, z'_i]^T = R_z R_y R_x [x_i, y_i, z_i]^T + [B_x, B_y, B_z]^T \quad \text{i} = 1,\ 2,\ ...6 \tag{1}$$

R_j è la matrice di rotazione rispetto all'asse $j (j = x, y, z)$ e B è il vettore di traslazione. Le misure delle elongazioni di ciascun attuatore, noto il vettore di roto-traslazione r, sono date dalle relazioni:

$$d_i = [(a_i p'_i)(a_i p'_i)^T]^{\frac{1}{2}} \tag{2}$$

Il problema inverso rispetto a quello descritto (determinazione degli angoli e della traslazione a partire dalle elongazioni degli attuatori) risulta complicato dalla non invertibilità in senso analitico del problema.
Si consideri il set di sei equazioni non lineari nelle incognite rappresentate dalle componenti di r, per una data configurazione degli attuatori d_i^*, il problema puo essere scritto:

$$F_i = d_i^* - [(a_i p'_i)(a_i p'_i)^T]^{\frac{1}{2}} = 0 \tag{3}$$

che in forma compatta diventa:

$$F(r) = 0 \tag{4}$$

$$r = [\theta_x, \theta_y, \theta_z, B_x, B_y, B_z]^T$$

dove F è un'applicazione non lineare in $\Re^6$ continua con la sua derivata $J = F'(r)$.
Gli algoritmi per risolvere la (4) sono di tipo ricorsivo: data una approssimazione iniziale r_0, si generano sucessive soluzioni r_k fino al raggiungimento di una buona approssimazione della soluzione cercata. Linearizzando la (4) si ottiene:

$$r_{k+1} = r_k - J^{-1} F(r_k) \tag{5}$$

o in forma equivalente:

$$Jdr = -F(r) \tag{6}$$

La (6) rappresenta il metodo di Newton che risulta applicabile se lo Jacobiano J non è singolare. Uno dei limiti del metodo di Newton è la relativa difficoltà a raggiungere la convergenza e questo effetto è tanto più marcato quanto r_0 si discosta dalla soluzione ottimale.
Il Powells Hybrid method è uno degli algoritmi definiti quasi-Newton methods che supera le limitazioni del metodo classico. Ad ogni iterazione l'algoritmo determina una soluzione alla Newton risolvendo il sistema (6), se la nuova soluzione è tale da migliorare la convergenza questa viene accettata, altrimenti l'algoritmo

utilizza una combinazione lineare del metodo di Newton con il metodo di ricerca gradientale di un minimo (steepest descending):

$$dr = \alpha J^{-1} F(r) - \beta \nabla \|F(r)\|^2 \tag{7}$$

La (7) è stata implementata in ambiente Octave e sempre in Octave si sono scritti i moduli di visualizzazione 3D delle parti meccaniche di interesse.
A titolo di verifica si è operata una mappatura della superficie (6D) degli errori di stima dei parametri generando 10^4 configurazioni casuali di parametri di roto-traslazione, calcolando le corrispondenti elongazioni e quindi invertendo il problema per ottenere un nuovo set di parametri da confrontare con quelli di partenza. In fig. 2 si riporta la proiezione della superficie degli errori sull'asse θ_y (gli errori sono espressi in metri e

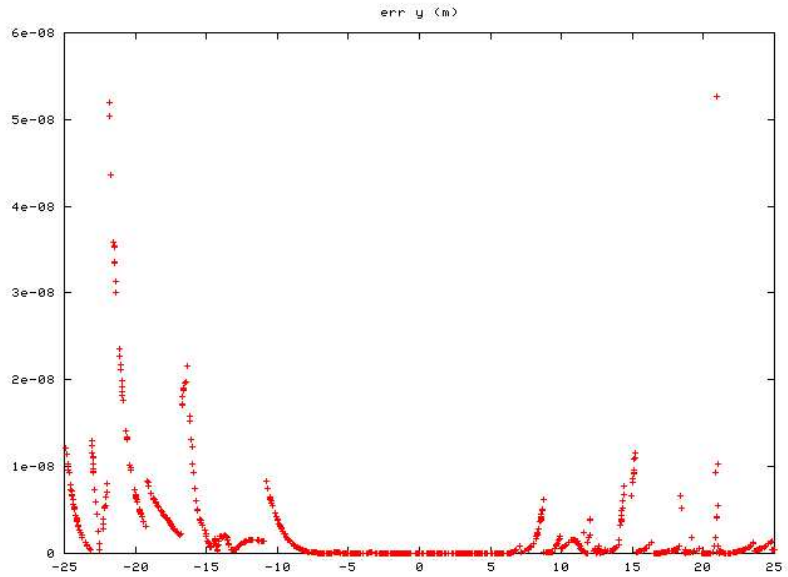

Figura 2: xxxx

gli angoli in gradi). Il range angolare utilizzato risulta molto più ampio di quello effettivamente concesso al sistema reale al fine di testare l'efficienza dell'algoritmo. Ulteriori verifiche sono state effettuate in ambiente CAD al fine di controllare la coerenza dei calcoli rispetto a un sistema simulato.

Catena cinematica

Nell'analisi sin qui svolta ciascun attuatore è stato trattato come un sistema formato da una barra estensibile con due giunti sferici alle estremità. Questo approccio permette di prevedere il comportamento del sistema attuatori-M2 attraverso un modello relativamente semplice sia nel caso diretto che in quello inverso.
Gli attuatori che effettivamente verranno utilizzati differiscono in realtà dal semplice modello utilizzato in quanto i giunti sferici sono sostituiti da un più complesso sistema di giunti cardanici (fig. 3). Se nella trattazione l'aver sostituito i giunti con delle sfere non ha nessun effetto sul modello cinematico di M2, questa approssimazione non è più valida se si vogliono studiare i moti dei singoli giunti cardanici allo scopo, ad esempio, di effettuare una mappatura dei loro range angolari di utilizzo e quindi se si vogliono simulare particolari configurazioni estreme che possano determinare rotture o stress meccanici del sistema.
Per lo studio e la caratterizzazione dei moti cardanici è utile ricorrere alla decomposizione dei moti della catena cinematica che definisce ciascun attuatore. L'obiettivo è quello di riuscire a determinare gli angoli di torsione di ciascun cardano per una data posizione del subriflettore nello spazio descritta dal vettore r definito nella (4) o in maniera equivalente dalle sei elongazioni definite nella (2).

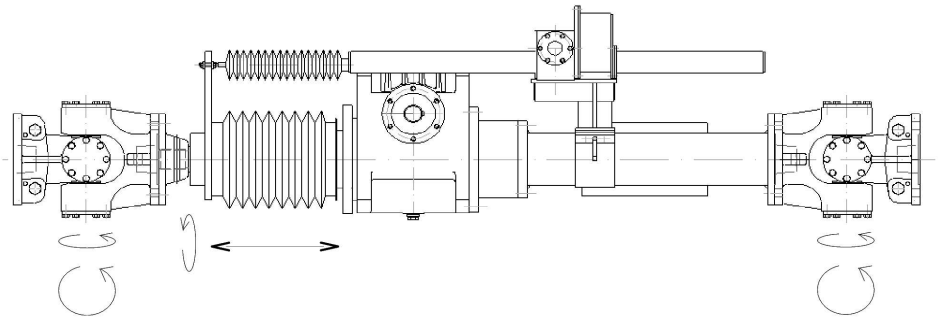

Figura 3: xxxx

Con riferimento alla figura 4, dove viene mostrato uno dei sei attuatori del subriflettore di SRT in una rappresentazione che permette di scomporre l'attuatore nei suoi moti elementari, si individuano le parti mobili (*giunti*) distinguendo tra quelle a cui corrisponde un moto circolare, *giunto di rivoluzione* (rappresentato da un cilindro) e quelle a cui corrisponde una traslazione, *giunto prismatico* (rappresentato in figura da un cubo). I *giunti* sono tra loro collegati da *link* che sono le parti non mobili della catena cinematica. Nel caso

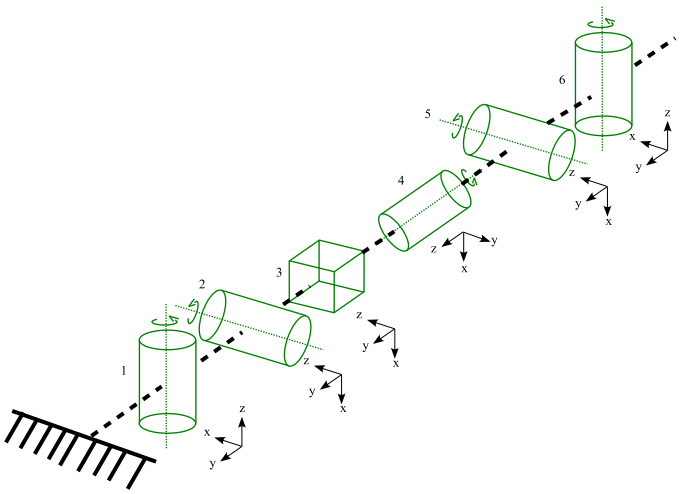

Figura 4: xxxx

mostrato in figura 4 sono rappresentati sei giunti (cinque di rivoluzione ed uno prismatico) corrispondenti ai gradi di mobilità dell'attuatore. Si può notare che ad ogni giunto viene associato un sistema di assi cartesiani (numerati dal #1 corrispondente al primo cardano vincolato alla struttura, fino al #6 corrispondente all'ultimo cardano solidale a M2). Sia allora A_i l'operatore di roto-traslazione che permette il passaggio dal sistema di assi con indice i a quello con indice $i-1$, si avrà che l'operatore T_1, espresso dalla relazione:

$$T_1 = \prod_{i=2}^{6} A_i \tag{8}$$

rappresenta l'operatore di passaggio dal sistema #6 al sistema #1. Si consideri il punto materiale q_6 nello spazio #6 e quindi a questo solidale (non essendoci altri moti *a valle* di quello considerato). In virtù di una generica movimentazione del subriflettore, le coordinate del punto rimangono invariate rispetto al sistema

#6, ma variano rispetto a tutti gli altri sistemi e in particolare rispetto al sistema #1. Il sistema #1 è solidale (ma non coincidente) con il sistema di riferimento di M2 definito nel precedente paragrafo (che verrà indicato come il sistema di assi #0) quindi per una qualunque movimentazione del subriflettore le nuove coordinate q_1 del punto materiale espresse nel sistema #1 sono ottenibili a partire dalle trasformazioni definite nel paragrafo precedente (a meno di un'ulteriore roto-traslazione che trasferisce il sistema #1 sul sistema #0. Queste considerazioni sono valide per ciascun attuatore anche se ovviamente la roto-traslazione che trasferisce il sistema #1 di ciascun attuatore sul sistema #0 di M2 sarà definita da parametri differenti, ma noti a priori.
Applicando la (8) a q_6 si ottiene dunque $q_1 = T_1 q_6$: qui tanto q_1 quanto q_6 sono noti a priori, mentre incogniti sono i valori dei parametri associati alla trasformazione T_1 (ovvero le posizioni angolari cercate dei giunti di rivoluzione). La tabella 2 riporta i parametri associati alla catena cinematica tramite la trasformazione

joint	α	β	γ	d_x	d_y	d_z
revol. #1	0	0	γ_1	0	0	0
revol. #2	0	$\frac{\pi}{2}$	γ_2	0	0	0
prism. #3	0	0	0	0	d_3	0
revol. #4	$-\frac{\pi}{2}$	0	γ_4	0	0	0
revol. #5	$\frac{\pi}{2}$	0	γ_5	0	0	0
revol. #6	0	0	γ_6	0	0	0

Tabella 2: Parametri cinematici, α, β e γ sono le rotazione attorno agli assi x, y e z di ciascun giunto e d_x, d_y e d_z le relative traslazioni

T_1. Con riferimento alla figura 4, se si considera il giunto #1 a questo compete una rotazione libera γ_1 attorno all'asse z del suo sistema di riferimento, al giunto #2 compete una rotazione *libera* dello stesso tipo, ma preceduta da una rotazione *fissata* di $\frac{\pi}{2}$ attorno all'asse y, con considerazioni analoghe si può passare da un sistema all'atro sino al #6 utilizzando i parametri riportati in tabella. Gli unici movimenti liberi corrispondono dunque alle rotazioni attorno all'asse z (γ_i) dei corrispondenti cardani e alla traslazione lungo l'asse y del giunto prismatico #3 (il cui valore è comunque noto perché deducibile dalla trattazione cinematica del precedente capitolo); le rotazioni attorno agli assi x e y (α e β) sono solo di tipo *fissato*. Si noti ancora che i sistemi di riferimento dei primi due cardani e degli ultimi due sono uniti da link di misura nulla, e ciò per tener conto che nell'attuatore *reale* di figura 3 gli assi dei cardani si intersecano.
Per testare la catena cinematica descritta è dunque necessario risolvere l'eq. (8) per i valori consentiti al vettore di roto-traslazione r definito nella (4). In questo studio i range nominali applicati alle componenti di r sono $\theta_x = \pm 0.25$ (deg), $\theta_y = \pm 0.25$ (deg), $\theta_z = \pm 0.25$ (deg), $B_x = \pm 0.05$ (m), $B_y = \pm 0.11$ (m) e $B_z = \pm 0.05$ (m). I risultati di seguito esposti sono riferiti all'attuatore $a_6 p_6$, ma naturalmete l'analisi potrà essere facilmente estesa anche agli altri cinque attuatori.
Nelle figure successive sono presentate tre serie ciascuna delle quali è composta da sei grafici corrispondenti ai moti dei sei giunti dell'attuatore $a_6 p_6$. I moti sono espressi in funzione degli angoli di rotazione del sistema θ_x, θ_y e θ_z (indicati nei grafici come ROT(x), ROT(y) e ROT(z)). Nella simulazione le tre componenti di traslazione del vettore r erano poste pari a zero.
I grafici mostrano i range di lavoro dei singoli cardani, ma anche le correlazioni esistenti tra il moto di ciascun giunto e le componenti angolari del vettore r. Se si considera ad esempio il giunto di rivoluzione #1 si può notare che questo risulta non correlato a θ_x e correlato a θ_y e θ_z, nel caso del giunto #4 questo risulta strettamente correlato soltanto a θ_y.

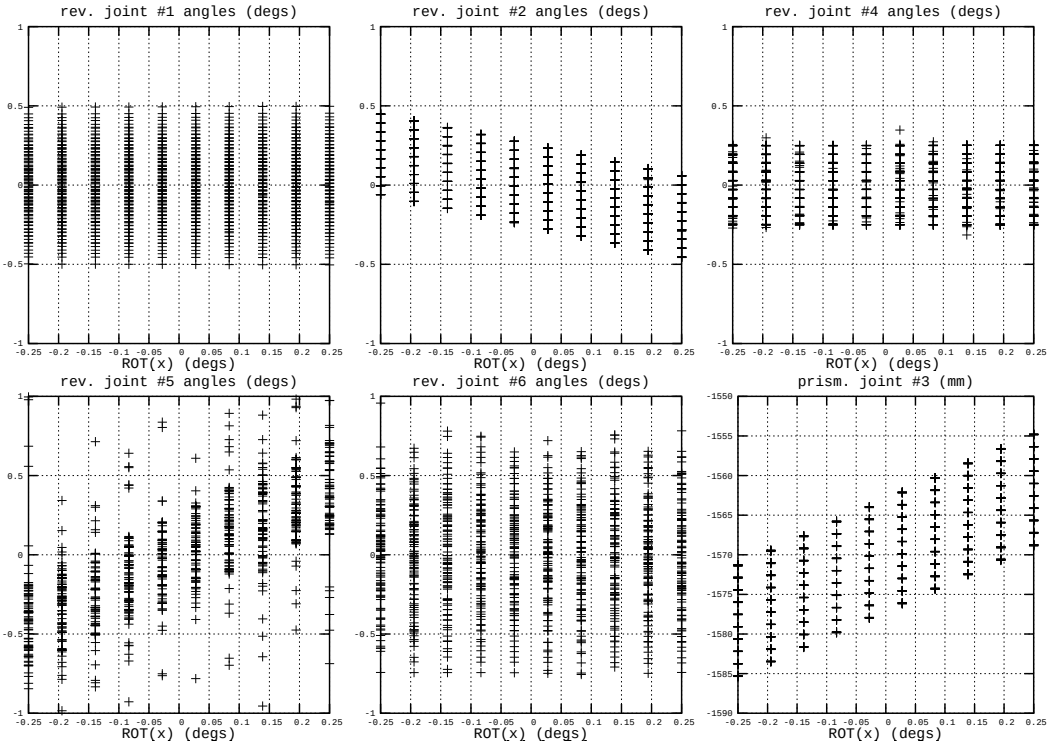

Figura 5: Valori dei giunti prismatici e di rivoluzione in funzione di θ_x (ROT(x) nel grafico).

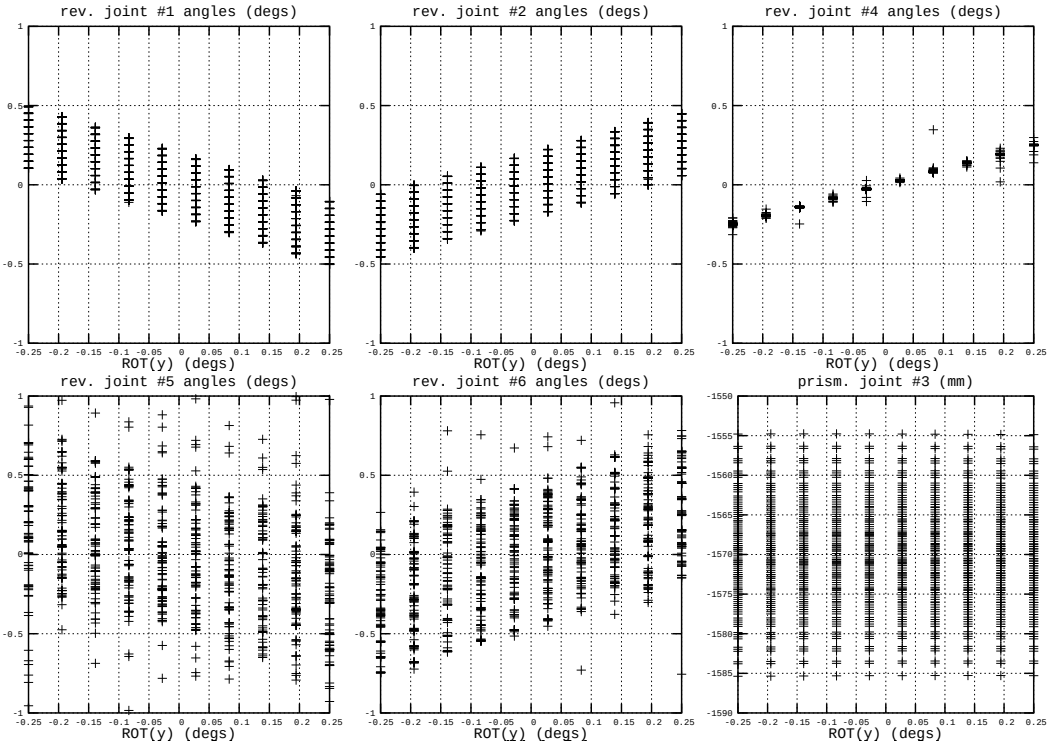

Figura 6: Valori dei giunti prismatici e di rivoluzione in funzione di θ_y (ROT(y) nel grafico).

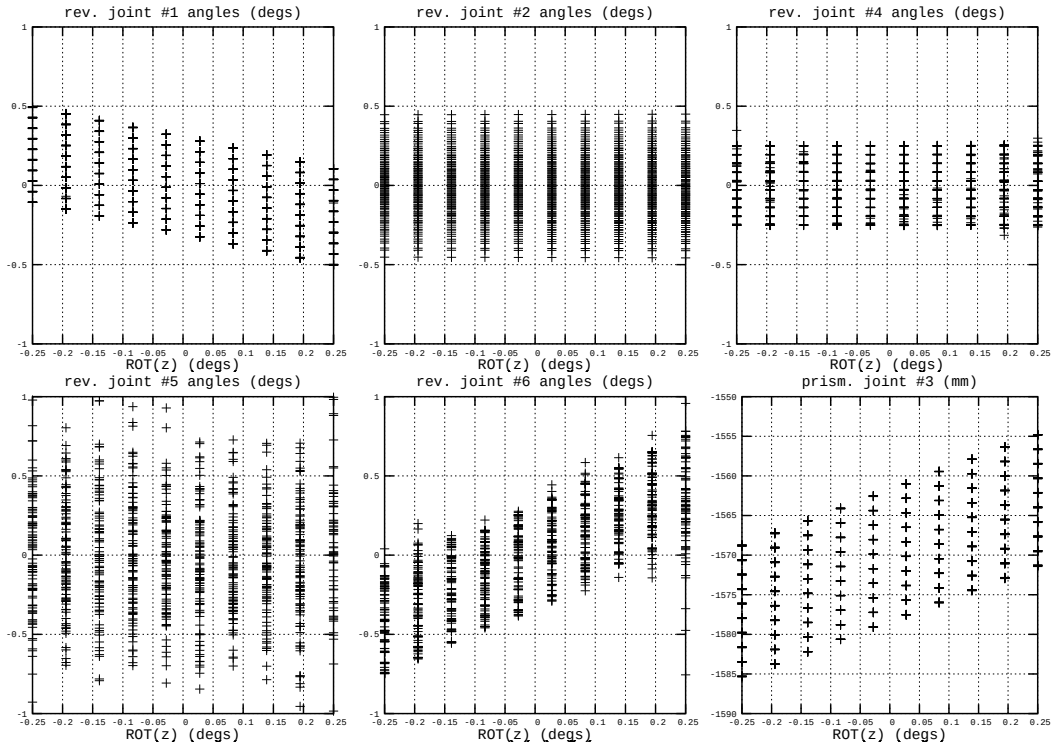

Figura 7: Valori dei giunti prismatici e di rivoluzione in funzione di θ_z (ROT(z) nel grafico).

Libreria hexlib

Funzioni di libreria

La libreria hexlib consente di trattare il problema cinematico ed è quindi utilizzabile per il controllo di M2. Le due funzioni di libreria più importanti **dir** e **inv** permettono di risolvere il problema diretto ed inverso, le altre funzioni svolgono il compito di inizializzazione dei parametri e di visualizzazione dei risultati del calcolo. La libreria non è, per il momento, utilizzabile per la decomposizione dei moti cardanici.
Per il porting in C della libreria si è utilizzata la GNU Scientific Library (GSL), questa libreria dovrà ovviamente essere installata sulla macchina nella quale si intende utilizzare la hexlib. La GSL è disponibile all'indirizzo: *http://www.gnu.org/software/gsl/*, sia per Linux che per Winxx.

init_p

```
#include "hexlib.h"
int init_p(p)
struct rparams *p;
```

Inizializza la struttura dei parametri p con i valori di default. La struttura dei parametri contiene le coordinate di punti che definiscono la culla di M2 più altri parametri utilizzati nel calcolo. I valori di default sono calcolati a partire da alcune costanti (lunghezza degli attuatori, degli snodi ecc.) definite in hexlib.h. L'inizializzazione deve essere necessariamente eseguita all'inzio del calcolo (in alternativa si può utilizzare load_p).

load_p

```
#include "hexlib.h"
int load_p(p, fname)
struct rparams *p;
char *fname;
```

Inizializza la struttura dei parametri p con i valori contenuti nel file fname. La struttura p contiene le coordinate di punti che definiscono la culla di M2. L'inizializzazione deve necessariamente avvenire all'inizio del calcolo. Il file fname è un file di testo costituito dalle coordinate dei punti ai e pi espresse in mm. Le prime 6 righe contengono le coordinate dei punti p1, p2,..., p6, le successive 6 quelle dei punti p1, p2,..., p6. Le coordinate sono separate da tabulazioni. L'utente deve predisporre questo file utilizzando coordinate note. Ad esempio si può modificare il file generato con il comando print_p. La funzione restituisce il valore 0 se il file è stato aperto con successo 1 altrimenti.

print_p

```
#include "hexlib.h"
int print_p(p)
struct rparams *p;
```

Stampa a schermo le coordinate dei punti a_i e p_i. I punti sono nello stesso ordine utilizzato nella funzione load_p.
Esempio:

```
#include "hexlib.h"
int main ()
{
  struct rparams p;
  init_p(&p);
  print_p(&p);
  return 0;
}
```

dir

```
#include "hexlib.h"
void dir(p)
struct rparams *p;
```

Calcola le elongazioni degli attuatori a partire da una roto-traslazione data (problema diretto). I valori corrispondenti alla roto-traslazione (espressi in radianti e in millimetri) vanno caricati in *p.x*.
Esempio:
vedi il programma test01.c.

inv

```
#include "hexlib.h"
void inv(p)
struct rparams *p;
```

Calcola una roto-traslazione a partire dalle elongazioni espresse in millimetri (problema inverso). I valori alle elongazioni vanno caricati in *p.d*.
Esempio:
vedi i programmi test02.c e test03.c.

print_state

```
#include "hexlib.h"
int print_state (p)
struct rparams *p;
```

Restituisce alcune informazioni relative alla soluzione del problema non lineare, quali in numero di iterazioni, gli errori ecc.
Esempio:
vedi il programma test03.c.

Esempi di utilizzo

test01.c

Il codice test01.c contiene un esempio di trasformazione diretta da una roto-traslazione alla lunghezza delle elongazioni degli attuatori.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "hexlib.h"
```

```
int main (int argc, char *argv[])
{
  int i;
  const size_t n = _n;
  struct rparams p;

  init_p(&p);

  if(argc<7)
  {
     printf("\n\nCalcolo delle elongazioni assolute (in mm) degli\n");
     printf("attuatori della movimentazione cinematica di M2.\n");
     printf("uso:\n%s tx ty tz rx ry rz -v\n\n",argv[0]);
     printf("l'opzione -v fa il dump della struttura dei parametri\n");
     printf("tx=traslazione x\n");
     printf("ty=traslazione y\n");
     printf("tz=traslazione z\n");
     printf("x=rotazione x\n");
     printf("y=rotazione y\n");
     printf("z=rotazione z\n");
     exit(0);
  }
  for(i=0;i<n;i++) p.x[i]=atof(argv[i+1]);
  dir(&p);
  for (i=0;i<n;i++) printf("d[%d]=%f\n",i,p.d[i]);
  if(argc>7) if (!strcmp(argv[7],"-v")) print_p(&p);

  return 0;
}
```

test02.c

Il programma test02.c è un esempio di trasformazione inversa dalla lunghezza delle elongazioni degli attuatori alla corrispondente una roto-traslazione.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "hexlib.h"

int main (int argc, char *argv[])
{
  int j;
  const size_t n = _n;
  struct rparams p;

  init_p(&p);
  if(argc<7)
  {
     printf("\n\nCalcola le traslazioni (mm) e gli angoli di rotazione (deg) date\n");
     printf("le elongazioni assolute degli attuatori della movimentazione\n");
     printf("cinematica di M2. \n");
     printf("uso:\n%s d1 d2 d3 d4 d5 d6\n\n",argv[0]);
     printf("d1=attuatore Z_1\n");
```

```
    printf("d2=attuatore Z_2\n");
    printf("d3=attuatore Z_3\n");
    printf("d4=attuatore X_1\n");
    printf("d5=attuatore Y_1\n");
    printf("d6=attuatore Y_2\n");
    exit(0);
  }
  for(j=0;j<n;j++) p.d[j]=atof(argv[j+1]);
  inv(&p);
  print_state (&p);

  return 0;
```

test03.c

Il codice test03.c verifica la precisione dell'algoritmo calcolando le elongazioni a partire da un input dato da linea di programma e quindi invertendo il problema per ottenere un nuovo set di parametri da confrontare con quelli di partenza. I parametri vengono forniti dall'utente attraverso il file data.txt.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "hexlib.h"

int main (int argc, char *argv[])
{
  int i;
  const size_t n = _n;
  double x0[n];
  struct rparams p;

  if(load_p(&p,"data.txt"))
  {
    printf("non trovo il file attuatori\n");
    exit(0);
  }

  if(argc<7)
  {
    printf("\n\nCalcolo delle elongazioni assolute (in mm) degli\n");
    printf("attuatori della movimentazione cinematica di M2.\n");
    printf("uso:\n%s tx ty tz rx ry rz -v\n\n",argv[0]);
    printf("l'opzione -v fa il dump della struttura dei parametri\n");
    printf("tx=traslazione x\n");
    printf("ty=traslazione y\n");
    printf("tz=traslazione z\n");
    printf("x=rotazione x\n");
    printf("y=rotazione y\n");
    printf("z=rotazione z\n");
    exit(0);
  }
  for(i=0;i<n;i++) p.x[i]=atof(argv[i+1]);
  dir(&p);
  printf("Calcolo diretto, rototrasl.->elongazioni\n");
  for (i=0;i<n;i++) printf("d[%d]=%f\n",i,p.d[i]);
```

```
  printf("Calcolo inverso, elongazioni->rototrasl.\n");
  inv(&p);
  print_state (&p);

  if(argc>7) if (!strcmp(argv[7],"-v")) print_p(&p);

  return 0;
}
```