INTRODUZIONE A

ROBOTICS TOOLBOX

a cura di

Adriano Fagiolini

---

24 ottobre 2005

TRASFORMAZIONI GEOMETRICHE ELEMENTARI E MATRICI DI TRASFORMAZIONE OMOGENEE

1. Traslazioni

Un'operazione di traslazione in Robotics Toolbox (RT) si ottiene utilizzando il comando *transl*. Tale comando prende in ingresso i valori di traslazione lungo gli assi *x*, *y* e *z*. Questi valori possono essere specificati come tre scalari oppure come un unico vettore *v* di tre componenti. In uscita, tale comando restituisce la matrice di trasformazione omogenea associata alla traslazione specificata dai parametri di ingresso.

```
T = transl(x,y,z);

v = [x y z];
T = transl(v);
```

2. Rotazioni

In RT, le operazioni di rotazione attorno agli assi *x*, *y* e *z* si ottengono usando rispettivamente i comandi *rotx*, *roty* e *rotz*. Tutti e tre i comandi accettano in ingresso un valore che rappresenta in radianti l'angolo *theta* di rotazione. In uscita, tali funzioni restituiscono la matrice di trasformazione omogenea che realizza la rotazione specificata.

```
T = rotx(theta);
T = roty(theta);
T = rotz(theta);
```

Si puo' ottenere anche una rotazione attorno ad un asse generico usando il comando *rotvec*, che accetta due parametri in ingresso: il vettore *v* che individua l'asse di rotazione e l'angolo *theta* di rotazione in radianti. In uscita, *rotvec* restituisce la matrice di trasformazione omogenea associata.

```
T = rotvec(v, theta);
```

Esempio – Passaggio da una terna di riferimento ad un'altra

Si consideri un sistema di riferimento cartesiano e una terna 0 centrata nell'origine e con assi coincidenti con quelli del sistema. Si consideri inoltre una terna 1 ottenuta dalla terna 0 eseguendo le seguenti operazioni (espresse in assi correnti):

1. traslazione dell'origine di un vettore *p = [3 2 4]*;
2. rotazione attorno all'asse *x* di un angolo pari a *–pi/2* radianti;
3. rotazione attorno all'asse *z* di un angolo pari a *pi/2* radianti;

Scrivere un listato Matlab che calcola la matrice di trasformazione omogenea *T* in assi correnti che consente di passare dal sistema di riferimento 0 al sistema di riferimento 1.

Soluzione:
La matrice di trasformazione omogenea è ottenuta dalla composizione di queste operazioni. Pertanto si ha:

```
p = [3 2 4];
T = transl(p) * rotx(-pi/2) * rotz(pi/2);
```

In questo modo si ottiene:

```
T =
```

```
    0.0000   -1.0000         0    3.0000
    0.0000    0.0000    1.0000    2.0000
   -1.0000   -0.0000    0.0000    4.0000
         0         0         0    1.0000
```

3. Estrarre le informazioni di traslazione e rotazione da una matrice omogenea T

Per ricavare esplicitamente le informazioni relative alla traslazione *p* e alla rotazione *R* contenute in una matrice di trasformazione omogenea *T* si utilizzano rispettivamente il comando overloaded (cfr. *Nota1*) *transl* e il comando *tr2rot*.

Nota1: il comando transl, che abbiamo già incontrato, si comporta in modo diverso se riceve in ingresso una matrice invece che un vettore. In questo caso, infatti, estrae da tale matrice le componenti relative alla sola traslazione.

```
p = transl(T);
R = tr2rot(T);
```

In questo modo si ottiene:

```
p =
     3
     2
     4
R =
    0.0000    1.0000         0
   -0.0000    0.0000    1.0000
    1.0000   -0.0000    0.0000
```

Si possono estrarre da una matrice di trasformazione omogenea *T* direttamente i valori degli angoli di rotazione nelle due rappresentazioni minime (angoli di Eulero e Roll/Pitch/Yaw), usando i comandi *tr2eul* e *tr2rpy*. Per quanto riguarda la rappresentazione in angoli di Eulero, si scrive:

```
eul_ang = tr2eul(T);
```

```
eul_ang = tr2eul(T);
a = eul_ang(1);
b = eul_ang(2);
c = eul_ang(3);
```

In questo modo si ottiene:

```
a =
    1.5708
b =
    1.5708
c =
    -3.1416
```

L’operazione opposta, che consiste nel determinare la matrice di trasformazione omogenea *T* la cui rotazione è specificata da angoli di Eulero noti, puo’ essere ottenuta come:

```
T = rotz(a) * roty(b) * rotz(c);
```

oppure in un unico passo usando il comando *eul2tr*:

```
T = eul2tr([a b c]);
```

In questo modo si ottiene:

```
T =
    0.0000    1.0000    0.0000         0
   -0.0000   -0.0000    1.0000         0
    1.0000   -0.0000    0.0000         0
         0         0         0    1.0000
```

Nota2: Si tenga presente che le matrici di rotazione non godono della proprietà commutativa; ciò implica che, se desideriamo ad esempio ricostruire la matrice di rotazione di una terna a partire dai valori degli angoli espressi in una qualunque rappresentazione minima, si deve prestare attenzione all'ordine di composizione delle singole matrici.

In modo analogo si estraggono da una matrice di trasformazione omogenea *T* le informazioni relative agli angoli Roll/Pitch/Yaw usando il comando *tr2rpy*.

```
rpy_ang = tr2rpy(T);
```

```
rpy_ang = tr2rpy(T);
a = rpy_ang(1);
b = rpy_ang(2);
c = rpy_ang(3);
```

In questo modo si ottiene:

```
a =
     0
b =
   -1.5708
c =
   -1.5708
```

L'operazione opposta si puo' ottenere combinando le tre rotazioni

```
T = rotz(a) * roty(b) * rotx(c);
```

oppure in un unico passo usando il comando *rpy2tr*:

```
T = rpy2tr([a b c]);
```

Esempio

Si scriva uno script in Matlab che estrae:

1. la posizione *p* dell'origine della terna 1 riferita alla terna 0;
2. la matrice di rotazione *R*;
3. gli angoli di Eulero;
4. gli angoli di Roll/Pitch/Yaw;

Successivamente, si ricostruisca la matrice di trasformazione *T* a partire dalle informazioni trovate nei punti 1-4.

Soluzione:

La prima parte dell'esercizio si risolve scrivendo il seguente script:

```
p = transl(T);
R = tr2rot(T);
eul_ang = tr2eul(T);
a1 = eul_ang(1); b1 = eul_ang(2); c1 = eul_ang(3);
rpy_ang = tr2rpy(T);
a2 = rpy_ang(1); b2 = rpy_ang(2); c2 = rpy_ang(3);
```

Infine, la matrice *T* si puo' ricostruire nei seguenti modi equivalenti:

```
T = transl(p) * rotz(a1) * roty(b1) * rotz(c1);
T = transl(p) * rotz(a2) * roty(b2) * rotx(c2);
T = transl(p) * eul2tr([a1 b1 c2]);
T = transl(p) * rpy2tr([a2 b2 c2]);
```

DEFINIZIONE DI UN ROBOT

In RT, tutte le informazioni relative ad un manipolatore vengono passate ad unico comando, il comando *robot*, che provvede alla costruzione di un oggetto Matlab che le contiene interamente (classe). Tra queste informazioni troviamo anzitutto quelle che servono per descrivere la catena cinematica del robot e che vengono usate, ad esempio, per il calcolo della cinematica diretta.

Procedura per la definizione di un oggetto della classe robot:
Una volta fissato il modello cinematico del manipolatore a catena aperta, si procede a riempire la tabella contenente i parametri di Denavit-Hartenberg. Come noto, ogni riga della tabella contiene i parametri delle quattro operazioni elementari che permettono di passare da un terna alla successiva.

1. Oggetto *link*

In RT, per definire un robot, è necessario definire i singoli corpi rigidi (bracci) che lo compongono. Questo si ottiene utilizzando il comando *link* che ha il seguente prototipo:

```
L = link([alpha A theta d sigma], convention);
```

1. i parametri *alpha*, *A*, *theta*, *d* sono i parametri nella tabella DH relativi alla riga del link che stiamo considerando;
2. s*igma* specifica se il giunto è rotoidale, 0, oppure prismatico, 1;
3. *convention* indica se si sta utilizzando la convezione di DH (‘standard’) oppure quella modificata (‘modified’).

In uscita, restituisce una variabile strutturata *L* atta a contenere tutte le informazioni relative a quel braccio.

I valori di questi parametri possono essere modificati anche accedendo direttamente ai campi della variabile *L*. Si puo’ ad esempio scrivere:

```
L.alpha = 0;
L.A = 0;
L.theta = pi/2;
L.d = 0.15;
L.sigma = ‘R’;
L.mdh = 0;
```

Si devono poi specificare altri parametri tra cui:

1. valori limite della variabile di giunto (posizione dei fine corsa) scrivendo
   ```
   L.qLim = [0 0.25];
   ```
2. parametri inerziali (massa e inerzia) inizializzando i campi *L.m* e *L.I*;
3. massa e inerzia del motore di attuazione del giunto, rapporto di riduzione e attrito viscoso.

Per visualizzare le informazioni relativi al link, si puo’ usare il comando *showlink*:

```
showlink(link);
```

Esempio

```
L = link([-pi/2 0.02 0 0.15]);
L.m = 0.5;
L.I = 0.015;
showlink(L)
```

In questa modo si ottiene:

```
alpha  = -1.5708
A      = 0.02
theta  = 0
D      = 0.15
sigma  = 0
mdh    = 0
offset = 0
m      = 0.5
r      =
I      =
Jm     =
G      =
B      = 0
Tc     = 0  0
qlim   =
```

2. Oggetto *robot*

Per completare l'operazione di definizione di un robot, una volta che sono state inserite tutte le informazioni relative ai bracci che lo compongono, si chiama il comando *robot* che accetta in ingresso la lista dei link appena definiti, più altre informazioni come il nome del robot, il costruttore e una stringa di commento.

```
L = {L1, L2, …, Ln};
myRobot = robot(L, 'myRobotName', 'UNIPI', '');
```

Esempio

Si consideri il file *puma560.m* che definisce la catena cinematica di un robot *puma*. Nel seguente estratto è possibile riconoscere i passaggi sopra descritti:

```
L{1} = link([ pi/2 0    0     0     0], 'standard');
L{2} = link([ 0   .4318 0     0     0], 'standard');
L{3} = link([-pi/2 .0203     0     .15005     0], 'standard');
L{4} = link([pi/2 0     0     .4318 0], 'standard');
L{5} = link([-pi/2 0    0     0     0], 'standard');
L{6} = link([0    0     0     0     0], 'standard');

L{1}.m = 0;
L{2}.m = 17.4;
L{3}.m = 4.8;
L{4}.m = 0.82;
L{5}.m = 0.34;
L{6}.m = .09;

…
```

```
p560 = robot(L, 'Puma 560', 'Unimation', 'params of 8/95');
clear L
p560.name = 'Puma 560';
p560.manuf = 'Unimation';
```

Il comando puma560 carica nel workspace di Matlab la descrizione cinematica di un robot *Puma* e la memorizza nella variabile *p560*.

Esempio

Si consideri il modello cinematico di un robot planare a due link con giunti rotoidali. Definire in Matlab l'oggetto *twolink_robot* usando i comandi di RT.

Soluzione:

```
L{1} = link([0 1 0 0 0], 'modified');
L{2} = link([0 1 0 0 0], 'modified');
twolink_robot = robot(L, 'TwoLink-Robot', 'UNIPI')
```

In questo modo si ottiene

```
twolink_robot =

TwoLink-Robot (2 axis, RR) [UNIPI]
            grav = [0.00 0.00 9.81]       modified D&H parameters

  alpha         A         theta         D        R/P
0.000000    1.000000    0.000000    0.000000    R     (mod)
0.000000    1.000000    0.000000    0.000000    R     (mod)
```

Infine, si puo' digitare:

```
>> showlink(twolink_robot)
Link 1------------------------
alpha  = 0
A      = 1
theta  = 0
D      = 0
sigma  = 0
mdh    = 1
offset = 0
m      =
r      =
I      =
Jm     =
G      =
B      = 0
Tc     = 0  0
qlim   =
Link 2------------------------
alpha  = 0
A      = 1
theta  = 0
D      = 0
sigma  = 0
mdh    = 1
offset = 0
m      =
r      =
I      =
Jm     =
```

```
G      =
B      = 0
Tc     = 0   0
qlim   =
```

CINEMATICA DIRETTA E INVERSA E GENERAZIONE DI TRAIETTORIE

1. Cinematica diretta

La cinematica diretta di un manipolatore è quella relazione che permette di esprimere la posizione e l'orientamento dell'end-effector in funzione delle posizioni delle variabili di giunto.

Il calcolo della cinematica diretta si ottiene in RT usando il comando *fkine* (forward kinematics). Questo comando prende in ingresso due parametri:

1. un oggetto *robot* che contiene il modello cinematico del manipolatore;
2. un vettore *q* che viene interpretato come la posizione delle coordinate generalizzate dei giunti.

Il comando restituisce la matrice di trasformazione omogenea relativa all'ultimo link del robot che contiene le informazioni di posizione e orientamento della terna utensile.

```
T = fkine(robot, q);
```

Esempio

Si desidera calcolare le informazioni di posizione e orientamento della terna utensile di un robot *puma* nelle posizioni di riposo *q_rest* = [0 0 0 0 0 0] e di robot-pronto *q_ready* = [0 pi/2 -pi/2 0 0 0].

```
q_rest = [0 0 0 0 0 0];
T_rest = fkine(p560, q_rest);
p_rest = transl(T_rest);
R_rest = tr2rot(T_rest);

q_ready=[0, pi/2, -pi/2, 0, 0, 0];
T_ready = fkine(p560, q_ready);
p_ready = transl(T_ready);
R_ready = tr2rot(T_ready);
```

In questo modo si ottiene:

```
p_rest =
    0.4521
   -0.1500
    0.4318

R_rest
     1     0     0
     0     1     0
     0     0     1

p_ready =
    0.0203
   -0.1500
    0.8636

R_ready =
     1     0     0
     0     1     0
     0     0     1
```

Passando al comando *fkine* una matrice *Q* al posto di un vettore *q*, il comando interpreta ogni riga della matrice *Q* come una diversa posizione del robot per la quale si deve calcolare la cinematica diretta. Pertanto, restituisce in uscita una matrice che ha dimensione 4x4xm (dove *m* è il numero di righe di *Q*) la cui ultima componente specifica l'indice della posizione calcolata. Dunque, per estrarre ad esempio, la matrice di trasformazione omogenea associata al i-esimo punto si deve scrivere:

```
T_step_i = T(:, :, i);
```

2. Cinematica inversa

La cinematica inversa è quella relazione che permette di risalire ai valori delle variabili di giunto a partire dalla conoscenza della posizione e dell'orientamento dell'end-effector.

Il calcolo di quest'ultima è ottenuto in RT usando il comando *ikine* (inverse kinematics). Tale comando riceve in ingresso:

1. un oggetto *robot*;
2. la matrice di trasformazione omogenea *T* nella quale sono contenute le informazioni relative alla posizione e all'orientamento dell'end-effector.

In uscita, restituisce il vettore *q* delle corrispondenti valori per le variabili di giunto.

```
q = ikine(robot, T);
```

Esempio:

Generiamo la matrice di trasformazione omogenea per un robot *puma* relativa alla posizione di giunto q = [0 -pi/4 -pi/4 0 pi/8 0]:

```
q1 = [0 -pi/4 -pi/4 0 pi/8 0];
T1 = fkine(p560, q1);
```

e calcoliamoci la soluzione della cinematica inversa

```
q1_bis = ikine(p560, T1);
```

Il risultato è dato da:

```
q1_bis =
   -0.0000   -0.7854   -0.7854   -0.0000    0.3927    0.0000       -0.0000
```

Consideriamo l'effetto della *ridondanza* nella catena cinematica di un robot sulla soluzione della cinematica inversa. Il comando *ikine* è in grado di trovare una delle soluzioni possibili.

Esempio:

```
q_ready = [0 pi/2 -pi/2 0 0 0];
T = fkine(p560, q_ready);
q_inverse = ikine(p560, T);
```

In questo modo si ottiene:

```
q_inverse =
   -0.0000    1.5238   -1.4768    0.0000   -0.0470    0.0000
```

La posizione relativa al vettore *q_inverse* è diversa da quella specificata attraverso il vettore *q_ready*. Ciò nonostante, la cinematica diretta per entrambe per le posizioni restituisce la medesima configurazione dell'end-effector :

```
error = fkine(p560, q_ready) – fkine(p560, q_inverse)
error =
  1.0e-016 *
         0    0.0000    0.0000    0.3123
   -0.0000         0   -0.0000         0
    0.0000    0.0000         0         0
         0         0         0         0
```

---

3. Generazione di traiettorie e animazione

La generazione di traiettorie è un argomento che viene ampiamente trattato nell'ambito della pianificazione del moto di un sistema. Nel caso più semplice, un problema di pianificazione consiste nel determinare un percorso che parta da un punto iniziale ed arrivi in un punto finale dati. In aggiunta a questo, si puo' richiedere che il percorso trovato sia compatibile con i vincoli cinematici e dinamici del sistema in esame. Più in generale, si puo' richiedere che il percorso trovato sia ottimo, cioè tale da minimizzare un indice di costo opportuno (tempo, spazio, potenza dissipata). Infine, la pianificazione di una traiettoria puo' essere svolta in presenza di ostacoli, eventualmente mobili.

RT è in grado generare traiettorie in assenza di ostacoli e tenendo conto solo dei vincoli cinematici del robot. Esistono due comandi, *jtraj* e *ctraj* che generano traiettorie rispettivamente nello spazio dei giunti e in quello operativo.

Il comando *jtraj* (joint trajectory) prende in ingresso:

1. un vettore *q0* che rappresenta la posizione iniziale delle variabili di giunto;
2. un vettore *qn* che rappresenta la posizione finale delle variabili di giunto;
3. un vettore *t* che contiene gli istanti di tempo in cui si desidera calcolare la posizione dei giunti.

In uscita, restituisce una matrice le cui righe rappresentano il vettore di posizione nello spazio dei giunti calcolato negli istanti indicati.

```
q = jtraj(q0, qn, t);
```

Esempio:

Per il robot *puma*, si desidera generare una traiettoria nello spazio dei giunti che parta dalla posizione di riposo *q_rest* alla posizione di robot-pronto *q_ready*. Si richiede inoltre che il moto venga eseguito complessivamente in 2 secondi e che sia calcolata una nuova posizione di giunto ogni 0.20.

Soluzione:

```
t = [0:0.20:2];
q_rest = [0 0 0 0 0 0];
q_ready = [0 pi/2 -pi/2 0 0 0];
Q = jtraj(q_rest, q_ready, t);
```

In questo modo si ottiene:

```
Q =
         0         0         0         0         0         0
         0    0.0134   -0.0134         0         0         0
         0    0.0910   -0.0910         0         0         0
         0    0.2562   -0.2562         0         0         0
         0    0.4986   -0.4986         0         0         0
         0    0.7854   -0.7854         0         0         0
         0    1.0722   -1.0722         0         0         0
         0    1.3146   -1.3146         0         0         0
         0    1.4798   -1.4798         0         0         0
         0    1.5574   -1.5574         0         0         0
         0    1.5708   -1.5708         0         0         0
```

Il comando *jtraj* puo' restituire anche i valori di velocità e accelerazione durante l'intervallo specificato:

```
[q dq ddq] = jtraj(q0, qn, t);
```

Nota: i valori di velocità e accelerazione vengono calcolati utilizzando un polinomio del settimo grado con condizioni al contorno nulle.

La traiettoria ottenuta puo' essere visualizzata mediante un'animazione usando il comando *plot* che prende in ingresso un oggetto *robot* e una matrice *Q* che contiene la sequenza dei valori assunti dalle variabili di giunto q.

```
plot(robot, Q)
```

Si desidera visualizzare un'animazione della traiettoria ottenuta nell'esempio precedente per il robot *puma*.

```
plot(p560, Q);
```

Esempio:

Si desidera far muovere il robot planare a due link dalla posizione di riposo q0 = [0 0] alla posizione qn = [pi/2, -pi/2]. Si richiede inoltre che il percorso sia eseguito complessivamente in 5 secondi. Infine si vuole che venga generato un punto della traiettoria ogni 0.3 secondi.

Soluzione:

```
t = [0 : 0.3 : 5];
q0 = [0 0];
qn = [pi/2 –pi/2];
Q = jtraj(q0, qn, t);
plot(twolink_robot, Q);
```

Il comando *ctraj* (Cartesian trajectory) permette di generare una traiettoria nello spazio operativo. Tale comando prende in ingresso:

1. la matrice di trasformazione omogenea *T0* relativa alla postura iniziale dell'end-effector;

2. la matrice di trasformazione omogenea *Tn* relativa alla postura finale dell'end-effector;
3. un vettore *r* che specifica le distanze tra i punti lungo il cammino (contiene valori compresi tra 0 e 1).

In uscita, restituisce una matrice 4x4xm, dove *m* è il numero di campioni.

```
TC = ctraj(T0, Tn, r);
```

Il commando *ctraj* puo' essere usato insieme al commando *ikine* per visualizzare il movimento di un robot lungo una traiettoria che porta l'end-effector dalla configurazione indicata in *T0* alla configurazione specificata in *Tn*:

Esempio:

Generare una traiettoria nello spazio operativo che parta dalla posizione [0.6 -0.5 0] e arrivi nella posizione [0.4 0.5 0.2].

```
t = [0 : 0.028 : 1];
T0 = transl([0.6 -0.5 0]);
Tn = transl([0.4 0.5 0.2]);
TC = ctraj(T0, Tn, t);
```

Supponendo che la traiettoria ottenuta sia relativa all'end-effector di un robot *puma*, si visualizzi un'animazione del movimento eseguito dal robot.

```
Q = ikine(p560, TC);
plot(p560, Q);
```

Esempio:

Generare una traiettoria nello spazio operativo per il robot planare a due link che parta dalla posizione [1 1 0] e arrivi in [2 0 0]. Visualizzare il risultato attraverso un'animazione.

```
t = [0 : 0.1 : 1];
T0 = transl([1 1 0]);
Tn = transl([2 0 0]);
TC = ctraj(T0, Tn, t);
Q = ikine(twolink_robot, TC);
plot(twolink_robot, Q);
```

CINEMATICA DIFFERENZIALE E JACOBIANI

1. Matrici di trasformazione omogenee associate a moti differenziali

Un moto differenziale *dx* = (*v_x*, *v_y*, *v_z*, *omega_x*, *omega_y*, *omega_z*) puo' essere rappresentato attraverso una matrice di trasformazione omogenea *T_diff* ottenuta dalla composizione di opportune matrici omogenee differenziali di traslazione e rotazione. Usando i comandi di RT, tale matrice si puo' ottenere come:

```
T_diff = transl([v_x v_y v_z]) * rotx(omega_x) * roty(omega_y) * rotz(omega_z);
```

In RT, è possibile ottenere in un solo passo la matrice omogenea associata ad un dato moto differenziale usando il comando *diff2tr*:

```
T_diff = diff2tr([v_x v_y v_z omega_x omega_y omega_z]);
```

Esempio

Calcolare la matrice di trasformazione omogenea associata al moto differenziale descritto dal vettore dx = [0.1 -0.2 0.1 0.5 1 -1].

Soluzione:

```
dx = [0.1 -0.2 0.1 0.5 1 -1];
T_diff = diff2tr(dx);
```

In questo modo si ottiene:

```
T_diff =
         0    1.0000    1.0000    0.1000
   -1.0000         0   -0.5000   -0.2000
   -1.0000    0.5000         0    0.1000
         0         0         0         0
```

Inoltre, data una matrice di trasformazione omogenea, è possibile estrarre l'informazione sul moto differenziale ad essa associato usando il comando *tr2diff*:

Applicando questo comando alla matrice omogenea dell'esempio precedente, si ottiene naturalmente:

```
dx = tr2diff(T_diff);

dx =
    0.1000
   -0.2000
    0.1000
    0.5000
    1.0000
   -1.0000
```

---

2. Relazione tra moti differenziali visti da sistemi di riferimento diversi

In molti casi è importante conoscere come un moto differenziale che ha luogo in un sistema di riferimento viene *visto* in un altro sistema di riferimento. Per fare ciò, si puo' utilizzare il comando *tr2jac*. Questo comando prende in ingresso la descrizione cinematica diretta che porta il primo sistema di riferimento a coincidere con il secondo e restituisce lo jacobiano che lega i vettori velocità nei due sistemi di riferimento.

```
jac_a2b = tr2jac(T_a2b);
```

Esempio

Si consideri il moto differenziale descritto, nel sistema di riferimento, *a* dal vettore *dx_a* = [0.1 0.2 0.3 -1 -2 -3]. Si prenda poi un secondo sistema di riferimento, *b*, ottenuto dal primo attraverso una traslazione dell'origine di una quantità pari a [1 2 -1] e una rotazione attorno all'asse *x* di un angolo pari a *pi/2* radianti. Si desidera trovare il vettore *dx_b* che descrive il moto differenziale considerato nel sistema di riferimento *b*.

Soluzione:

```
dx_a = [0.1 0.2 0.3 -1 -2 -3]';
T_a2b = transl([1 2 -1]) * rotx(pi/2);
jac_a2b = tr2jac(T_a2b);
dx_b = jac_a2b * dx_a;
```

In questo modo si ottiene:

```
dx_b =
    8.1000
    0.3000
    3.8000
   -1.0000
   -3.0000
    2.0000
```

---

3. Jacobiano di un robot

Lo jacobiano di un robot è utilizzato per calcolare le velocità di traslazione e rotazione della terna utensile, una volta assegnate le velocità dei giunti. Il suo valore in un dato istante dipende dal valore della variabili di giunto in quell'istante.

Lo jacobiano di un robot costituito da una catena cinematica aperta puo' essere calcolato usando il comando *jacobn*. Tale comando prende in ingresso:

1. un oggetto *robot* che contiene la descrizione cinematica del manipolatore;
2. un vettore *q* che rappresenta la posizione istantanea delle variabili di giunto.

In uscita, restituisce una matrice 6x6 che rappresenta lo jacobiano del *robot* calcolato nel punto indicato attraverso *q*.

```
jacobn(robot, q);
```

Esempio

Si desidera calcolare lo jacobiano del robot *puma* nella sua posizione di riposo che è descritta dal vettore *q_rest* = [0 0 0 0 0 0].

Soluzione:

```
q_rest = [0 0 0 0 0 0];
J = jacobn(p560, q_rest);
```

In questo modo si ottiene:

```
J =
    0.1500   -0.4318   -0.4318         0         0         0
    0.4521    0.0000    0.0000         0         0         0
         0    0.4521    0.0203         0         0         0
         0         0         0         0         0         0
         0   -1.0000   -1.0000         0   -1.0000         0
    1.0000    0.0000    0.0000    1.0000    0.0000    1.0000
```

Nota: dal valore numerico ottenuto per lo jacobiano del robot puma in posizione di riposo, si puo' osservare che i tre giunti del polso non contribuiscono alla velocità di traslazione dell'end-effector.

---

4. Inversa dello jacobiano

La conoscenza della matrice inversa dello jacobiano di un robot è necessaria in molti schemi di controllo. Ciò è vero, ad esempio, quando le specifiche di moto sono date nello spazio operativo. In questi casi, è necessario ricavare le corrispondenti specifiche nello spazio dei giunti. Tale operazione puo' essere eseguita facendo uso di algoritmi che invertono la cinematica diretta del robot in modo numerico (iterativo) e che si basano proprio sull'informazione dell'inversa dello jacobiano.

Per ricavare questa informazione, si puo' usare il comando *inv* di Matlab che inverte la matrice data come parametro di ingresso:

```
J_inv = inv(J);
```

Si noti inoltre che, in corrispondenza di ogni configurazione singolare, lo jacobiano perde rango massimo. Questo succede, ad esempio, per il robot *puma* nella configurazione di riposo. Infatti si ha:

```
rank(jacobn(p560, q_rest))
ans = 5
```

In questi casi, è interessante calcolare i valori singolari con il commando *svd*:

```
svd_values = svd(jacobn(p560, q_rest))
svd_values =
    1.8307
    1.7526
    0.4443
    0.3620
    0.2310
    0.0000
```
