Laborator 12: Controlul robotului Puma cu ajutorul Jacobianului, structuri cu numar mai mic/mare de grade de libertate

Obiective:

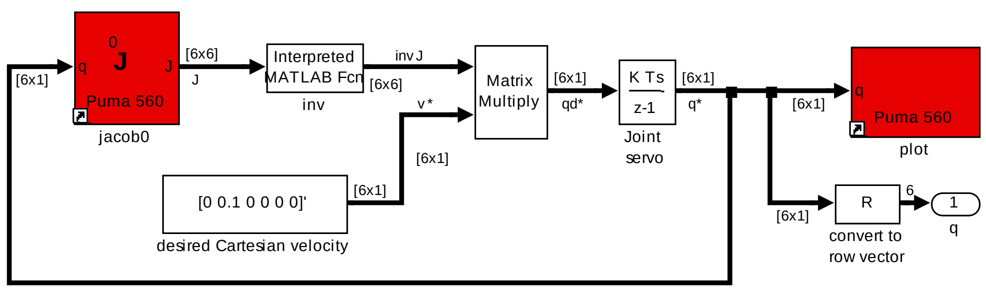
Utilizarea cunostintelor referitoare la calculul Jacobianului. Controlul robotlui Puma 560 cu ajutorul Jacobianului. Rezolvarea situatiilor cu nu numar de grade de libertate (DOF) diferit de 6

Elemente teoretice:

Studiul de caz 1: Jacobianul utilizat la controlul robotului (bucla deschisa)

(Peter Corke toolbox) Controlul pe baza Jacobianului

```
%Construirea robotului
mdl_puma560
```



. Schema utilizata la simularea comenzii Robotului Puma 560 [PC]

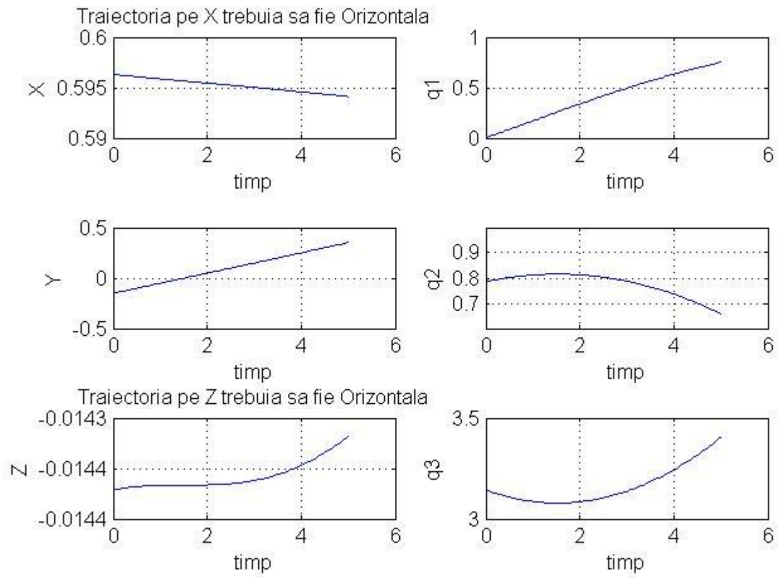
Simularea comenzii utilizeaza obiecte si programe deja implementate in toolbox pcorke. Jacobianul Jacob0 apeleaza o functie matlab care calculeaza matricea Jacobian; Puma 560 este un program sfunction care realizeaza animarea structurii in timpul simularii. Se observa ca viteza dorita are 6 componente, singura componenta nenula este cea pe directia Y.

Afisarea rezultatelor

```
t=r.find('tout'); %extragerea datelor din simulare
q=r.find('yout');
T=p560.fkine(q);% calculul posturilor pozitiile generalizate (din cuple)
xyz=transl(T); % extragerea celor 3 translatii din matricele posturilor

figure; % reprezentare grafica a rezultatelor obtinute
```

```
subplot(3,2,1); plot(t,xyz(:,1));grid; xlabel 'timp'; ylabel 'X'
 title 'Traiectoria pe X trebuia sa fie Orizontala'
subplot(3,2,3); plot(t,xyz(:,2));grid; xlabel 'timp'; ylabel 'Y'
subplot(3,2,5); plot(t,xyz(:,3));grid; xlabel 'timp'; ylabel 'Z'
subplot(3,2,2); plot(t,q(:,1));grid; xlabel 'timp'; ylabel 'q1'
subplot(3,2,4); plot(t,q(:,2));grid; xlabel 'timp'; ylabel 'q2'
subplot(3,2,6); plot(t,q(:,3));grid; xlabel 'timp'; ylabel 'q3'
```



Rezultatele simularii

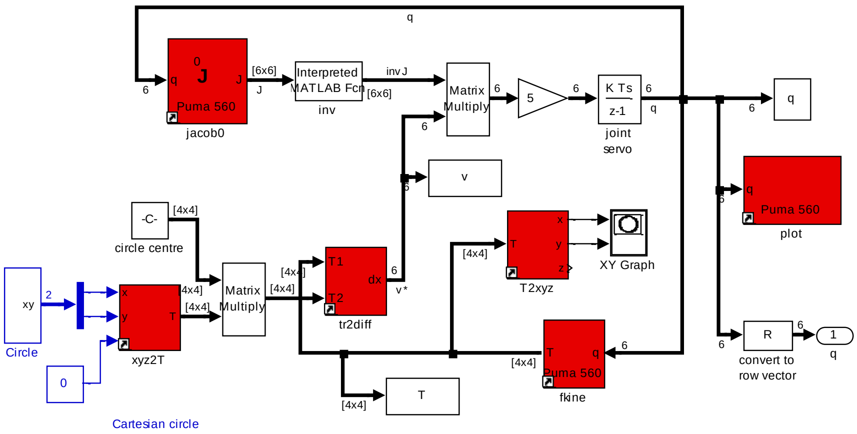
Studiul de caz2: Jacobianul utilizat la controlul robotului (bucla inchisa)

(Peter Corke toolbox) Controlul pe baza Jacobianului

```
%Construirea robotului
mdl_puma560
```

Apelarea modelului; Simularea

```
r=sim('sl_rrmc3')
```

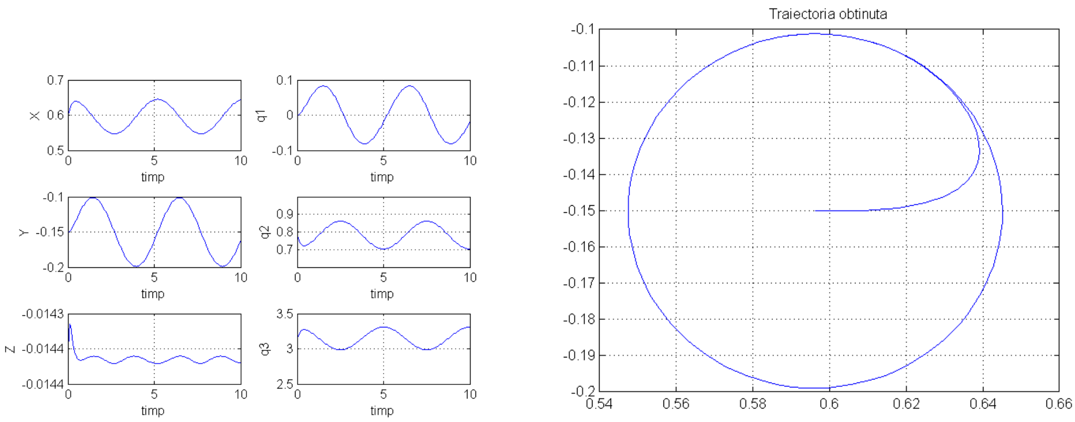


Schema de simulare [PC]

Afisarea rezultatelor

```
t=0:10/(length(q(:,1))-1):10;
 T=p560.fkine(q);% calculul posturilor pozitiile generalizate (din cuple)
 xyz=transl(T); % extragerea celor 3 translatii din matricele posturilor

 figure; % reprezentare grafica a rezultatelor obtinute
 subplot(3,2,1); plot(t,xyz(:,1));grid; xlabel 'timp'; ylabel 'X'
 subplot(3,2,3); plot(t,xyz(:,2));grid; xlabel 'timp'; ylabel 'Y'
 subplot(3,2,5); plot(t,xyz(:,3));grid; xlabel 'timp'; ylabel 'Z'
 subplot(3,2,2); plot(t,q(:,1));grid; xlabel 'timp'; ylabel 'q1'
 subplot(3,2,4); plot(t,q(:,2));grid; xlabel 'timp'; ylabel 'q2'
 subplot(3,2,6); plot(t,q(:,3));grid; xlabel 'timp'; ylabel 'q3'
 figure;
 plot(xyz(:,1),xyz(:,2)); title 'Traiectoria obtinuta'; grid
```



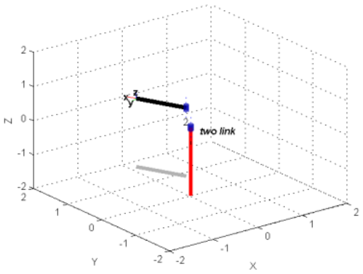
Rezultatele simularii a) coordonatele sculei; b) traiectoria sculei in plan

Studiul de caz 3: Sisteme cu un numar mai mic de grade de libertate

(Peter Corke toolbox) Operatii cu Jacobianul pentru sisteme cu mai putine, grade de libertate.

Cazul Robotului RR

```
mdl_twolink % apelarea modelului
qn=[1,1]; %pozitia nominala
twolink.plot(qn)
```



Robotul RR

Calculul Jacobianului

```
J=jacob0(twolink,qn)
```

```
J =
   -1.7508   -0.9093
    0.1242   -0.4161
         0         0
         0         0
         0         0
    1.0000    1.0000
```

Impunerea unei miscari pe axa X,calculul vitezelor unghiulare

```
dq=pinv(J)*[0.1,0,0,0,0,0]' % pseudo inversa inmultita cu vectorul vitezelor
% daca in cuple se impune viteza dq atunci scula robotului va avea
% vitezele:
dx=J*dq
% se observa componenta pe directia Y (pe care nu o doream
```

dq =
-0.0698
0.0431
dx =
0.0829
-0.0266
0
0
0
-0.0266

Solutia utilizarii minorului corespunzator mobilitatii robotului

```
Jxy=J(1:2,:) % se extrage doar partea superioara corespunzatoare mobilitatii pe X,Y
dq=inv(Jxy)*[0.1,0]' % utilizand vitezele sculei se determina vitezele din cuple
% din nou, daca in cuple avem vitezele calculate, atunci viteza sculei
% este:
dx=J*dq
% componenta pe Y a disparut, apare insa si aici componenta vitezei
% unghiulare pe Z (impotriva careia nu putem face nimic )
```

```
Jxy =
  -1.7508   -0.9093
   0.1242   -0.4161
dq =
  -0.0495
  -0.0148
dx =
   0.1000
   0.0000
        0
        0
        0
  -0.0642
```

: Sisteme cu un numar mai mare de grade de libertate Cazul Robotului

(Peter Corke toolbox) Operatii cu Jacobianul pentru sisteme cu mai multe, grade de libertate.

Roboti redundanti

In exemplu de caz se construieste un astfel de robot dintr-o masa X,Y peste care se aseaza robotul Puma 560

```
mdl_puma560
% Definirea mesei: sunt doua cuple de translatie X,Y (reciproc perpendiculare)
masa=SerialLink([0,0,0,-pi/2,1;-pi/2,0,0,pi/2,1],'base',troty(pi/2),'name','masa');
% Rotatia cu pi/2 in jurul lui Y este necesara ptc formalismul D-H impune,
% translatiile pe Z; masa insa este in planul XY
% Pregatirea robotului Puma: fata de configuratia implicita se doreste ca
% robotul Puma sa se inalte (pe un piedestal) d=0.3 m
p560.links(1).d =0.1;
%Asamblarea celor doi roboti
roby=SerialLink([masa, p560],'name','roby')

% impunerea unei posturi: s-a utilizat vectorul qn al robotului Puma560
% la care s-au adaugat cele 2 componente [0,0] ale mesei
qn2=[0,0,qn];
roby.plot(qn2)
% calculul Jacobianului
J=jacob0(roby,qn2);
```

```
roby (8 axis, PPRRRRRR, stdDH)

+---+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |
+---+-----------+-----------+-----------+-----------+
```

```
|   1|         0|        q1|         0|    -1.571|
|   2|    -1.571|        q2|         0|     1.571|
|   3|        q3|       0.1|         0|     1.571|
|   4|        q4|         0|    0.4318|         0|
|   5|        q5|      0.15|    0.0203|    -1.571|
|   6|        q6|    0.4318|         0|     1.571|
|   7|        q7|         0|         0|    -1.571|
|   8|        q8|         0|         0|         0|
+---+-----------+-----------+-----------+-----------+
```

Impunerea unei miscari dorite

se impune o translatie pe cele 3 componente X,Y,Z

```
dx=[0.2,0.2,0.2,0,0,0]';
dq=pinv(J)*dx; % se utilizeaza pseudoinversa
dx'
% utilizand viteza calculata se re calculeaza viteza sculei
dx=J*dq;
dx'
```

```
dx = 0.2000    0.2000    0.2000         0         0         0
dx = 0.2000    0.2000    0.2000    0.0000   -0.0000   -0.0000
```

Extragerea coloanelor liniar dependente

```
rank(J) % rangul matricei este 6
N=null(J) % extragerea celor 2 (8-6=2) coloane
```

```
ans =
   6
N =
   0.2543  -0.0320
   0.1086   0.2635
  -0.1821  -0.4419
   0.3543  -0.1534
  -0.7260   0.3144
  -0.2576  -0.6250
   0.3718  -0.1610
   0.1821   0.4419
```

Impunerea de miscari suplimentare care nu au efect asupra sculei

```
dq_ns=[0,0,0,0,-0.1,0,0,0]'; % se impune o miscare de rotatie in cupla 5 a robotului
% transformarea miscarii cu ajutorul coloanelor liniar dependente
dq_ns_tr=N*pinv(N)*dq_ns;
dq_ns_tr=dq_ns_tr/dq_ns_tr(5)*dq_ns(5);
dq_ns_tr'
% transformata a modificat scara asa ca e necesara o rescalare
dq_r=dq+dq_ns_tr;% rescrierea vitezei din cuple cu adaugarea vitezelor din spatiu zero
dq_r'
xd=J*dq;
xd'
```

```
dq_ns_tr' = 0.0311   -0.0006    0.0011    0.0488   -0.1000    0.0015    0.0512 -0.0011
dq_r' =     0.2112    0.1793    0.0347    0.3685   -0.0678    0.0490   -0.3007 -0.034

xd' =  0.2000    0.2000    0.2000    0.0000   -0.0000   -0.0000
```