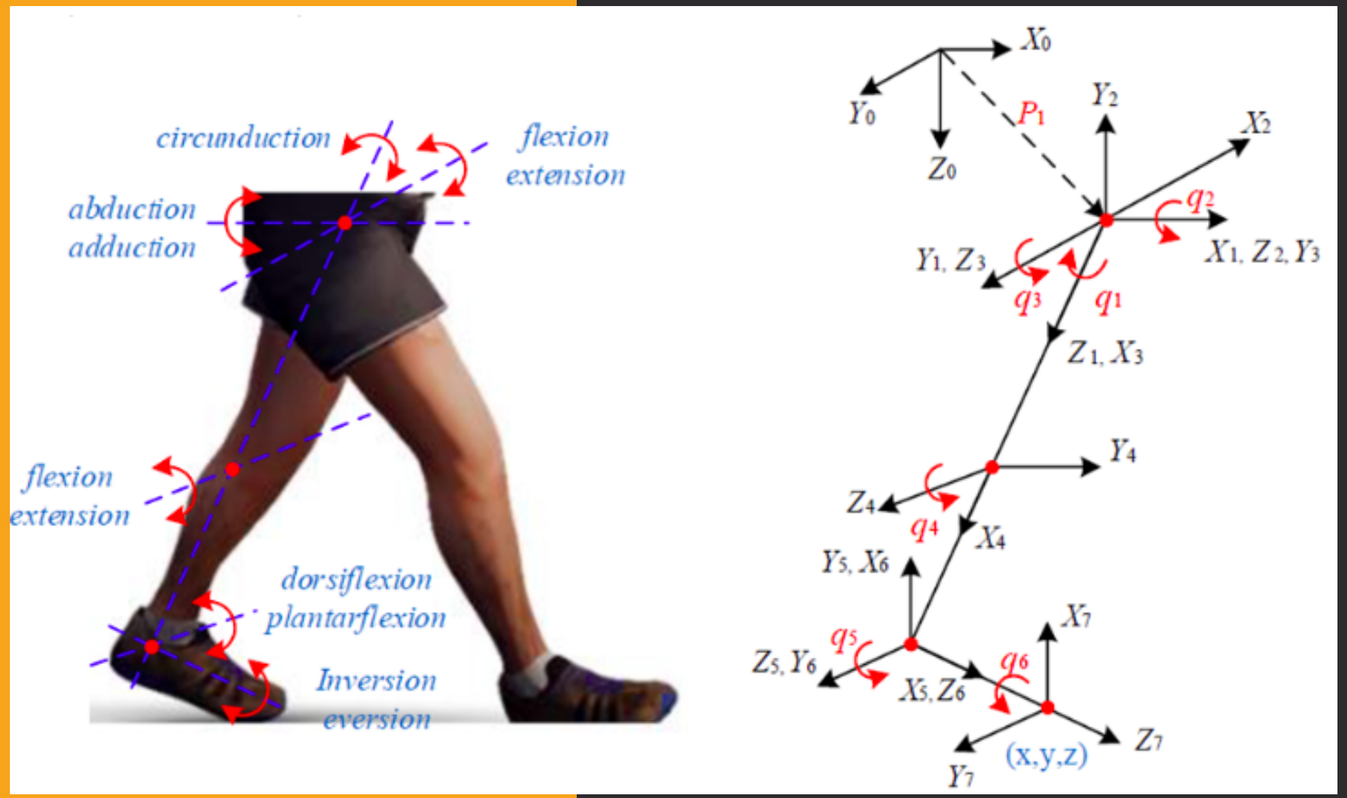
PRESENTACION FINAL

CINEMATICA DIFERENCIAL DE PIERNAS

José Ángel Ramírez Ramírez | A01735529

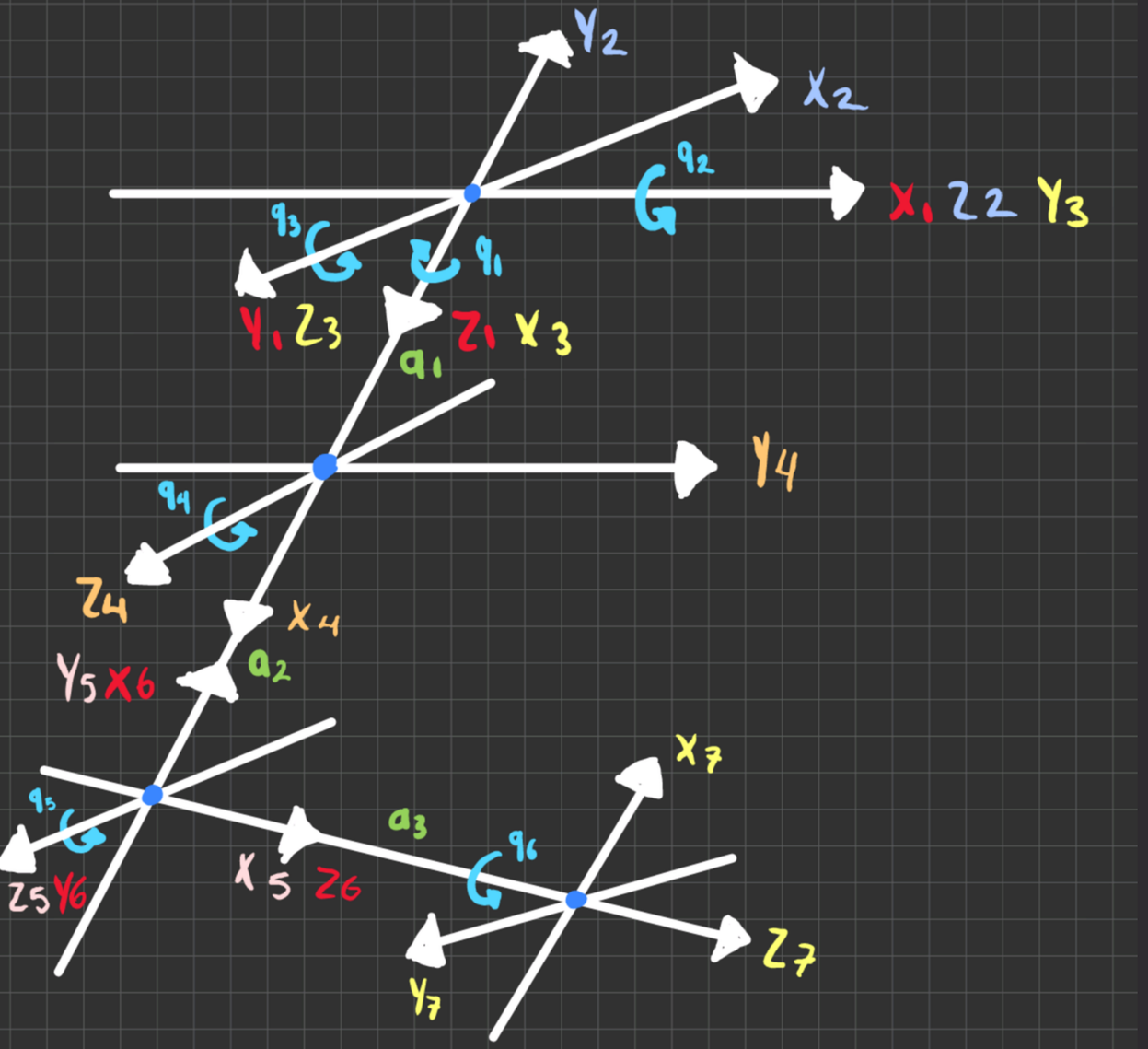
01

SISTEMA


circumduction
flexion
extension
abduction
adduction
flexion
extension
dorsiflexion
plantarflexion
Inversion
eversion

X0
Y0
Z0
P1
Y2
X2
q2
Y1, Z3
X1, Z2, Y3
q3
q1
Z1, X3
Y4
Z4
q4
X4
Y5, X6
X7
q5
Z5, Y6
q6
X5, Z6
Z7
(x,y,z)
Y7

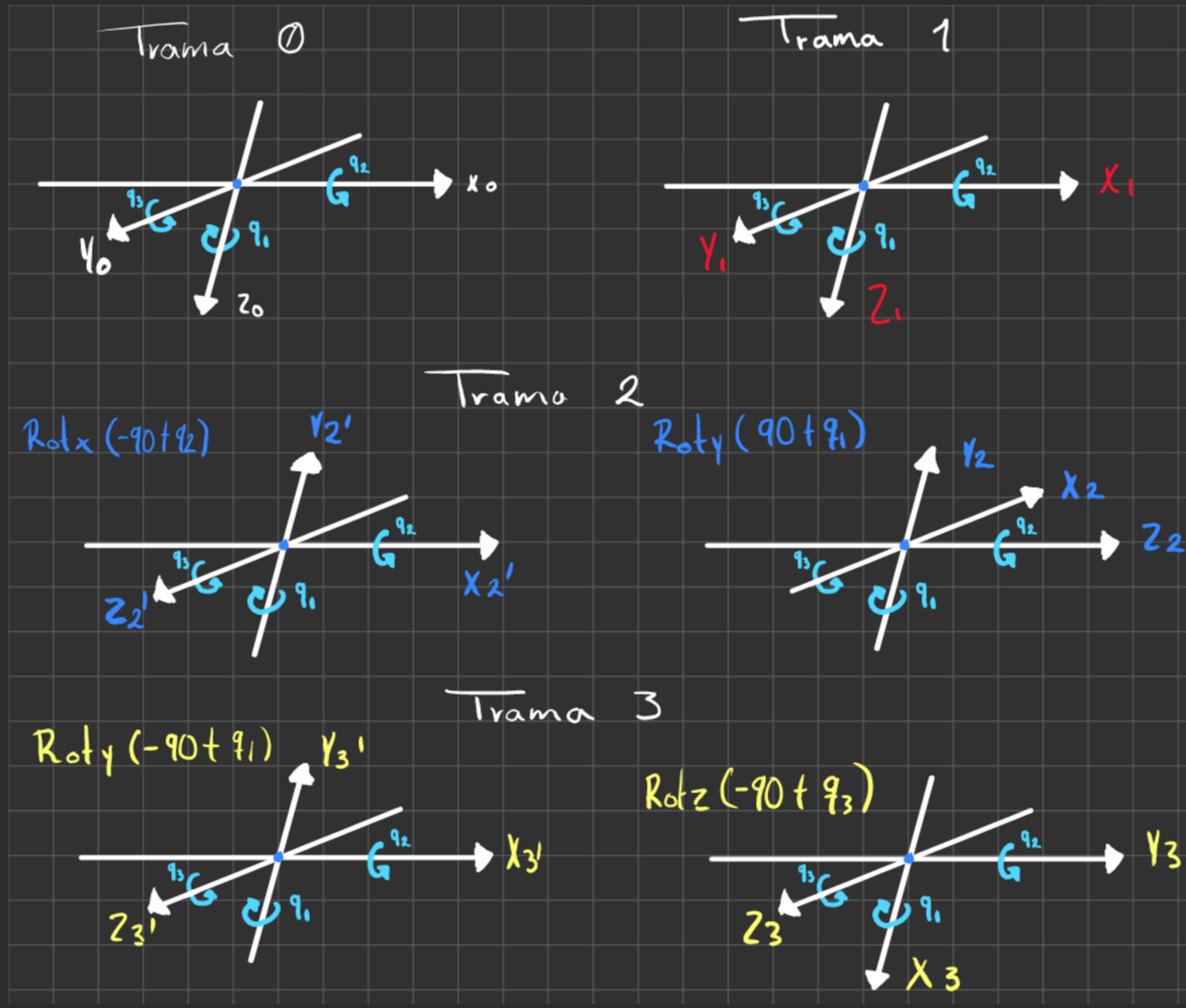
02


Y2
X2
q2
X1 Z2 Y3
q3
q1
Y1 Z3
Z1 X3
a1
Y4
q4
Z4
X4
Y5 X6
a2
X7
q5
a3
q6
Z5 Y6
X5 Z6
Z7
Y7

SISTEMA

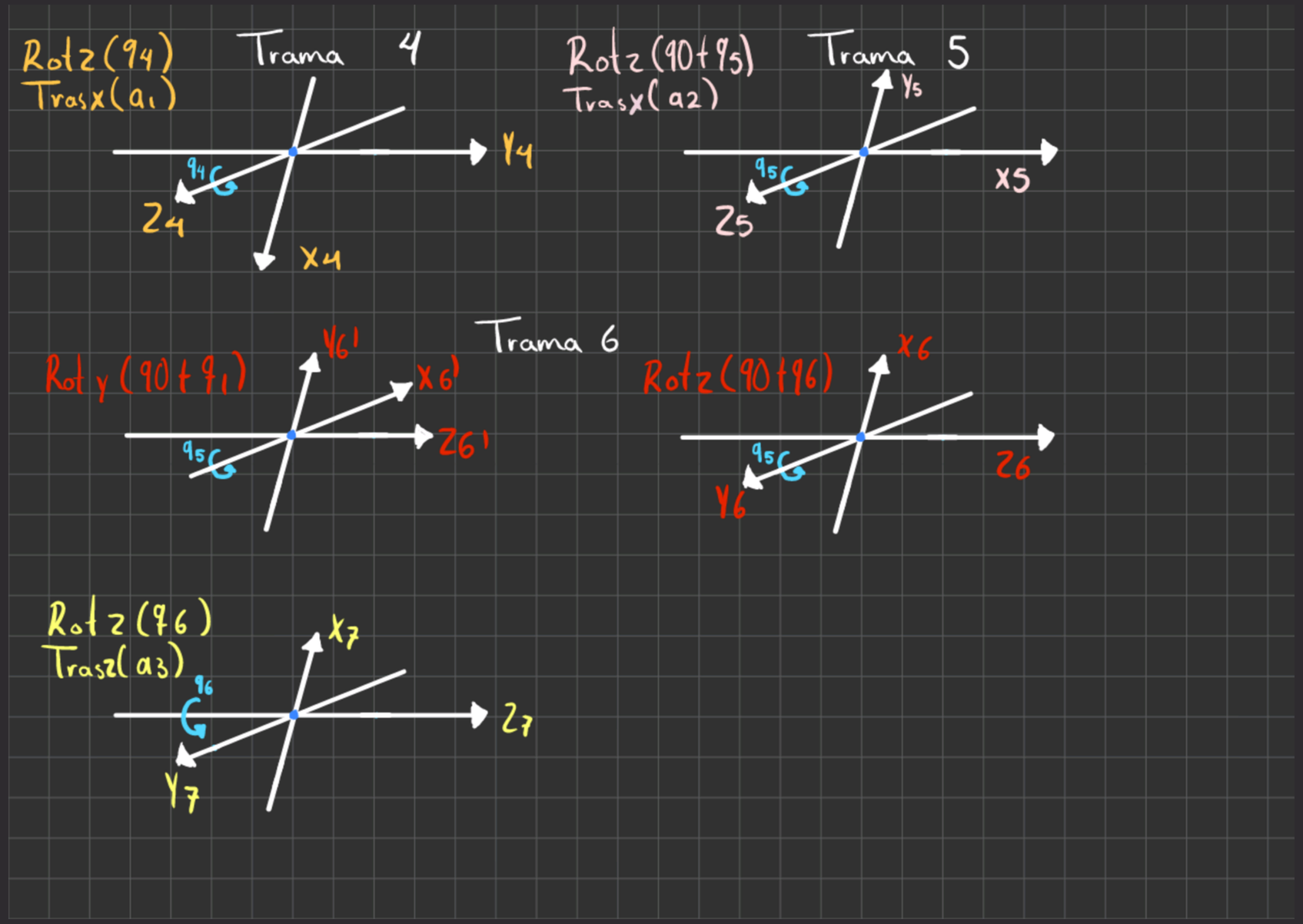
03

TRAMAS


Trama 0
x0
y0
z0
q1
q2
q3
Trama 1
X1
Y1
Z1
Trama 2
Rotx (-90 + q2)
Y2'
X2'
Z2'
Roty (90 + q1)
Y2
X2
Z2
Trama 3
Roty (-90 + q1)
Y3'
X3'
Z3'
Rotz (-90 + q3)
Y3
Z3
X3

04

TRAMAS


Rotz(q4)
Trasx(a1)
Trama 4
Y4
q4
Z4
X4
Rotz(90+q5)
Trasx(a2)
Trama 5
Y5
q5
X5
Z5
Trama 6
Roty(90+q1)
Y6'
X6'
q5
Z6'
Rotz(90+q6)
X6
q5
Z6
Y6
Rotz(q6)
Trasz(a3)
X7
q6
Z7
Y7

05

CODIGO

```
%Limpieza de pantalla
clear all
close all
clc

%Declaración de variables simbólicas
syms th1(t) th2(t) th3(t) th4(t) th5(t) th6(t) t a0 a1 a2 a3

n = 3; % Tamaño de la matriz identidad
matriz_identidad = sym(eye(n));


%Configuración del robot, 0 para junta rotacional, 1 para junta prismática
RP=[0 0 0 0 0 0];

%Creamos el vector de coordenadas articularesS
Q= [th1, th2, th3, th4, th5, th6];
%disp('Coordenadas generalizadas');
%pretty (Q);

%Creamos el vector de velocidades generalizadas
Qp= diff(Q, t);
%disp('Velocidades generalizadas');
%pretty (Qp);
%NÚmero de grado de libertad del robot
GDL= size(RP,2);
GDL_str= num2str(GDL);
```

```

%Trama 2
%Posición de la articulación 2 respecto a 1
P(:,:,1)= [a0;a0;a0];
%Matriz de rotación de la junta 2 respecto a 1....
R(:,:,1)= rotacion('x',-90+th2);
R(:,:,1)= R(:,:,1)*rotacion('y',90+th1);

%Trama 3
%Posición de la articulación 3 respecto a 2
P(:,:,2)= [0;0;0];
%Matriz de rotación de la junta 3 respecto a 2
R(:,:,2)= rotacion('y',-90+th1);
R(:,:,2)= R(:,:,2)*rotacion('z', -90+th3);

%Trama 4
%Posición de la articulación 4 respecto a 3
P(:,:,3)= [a1;0;0];
%Matriz de rotación de la junta 4 respecto a 3
R(:,:,3)= R(:,:,2);

%Trama 5
%Posición de la articulación 5 respecto a 4
P(:,:,4)= [a2;0;0];
%Matriz de rotación de la junta 5 respecto a 4
R(:,:,4)= rotacion('z', 90+th5);
```

CODIGO

```
56      %Trama 6
57      %Posición de la articulación 6 respecto a 5
58 -    P(:,:,5)= [0;0;0];
59      %Matriz de rotación de la junta 6 respecto a 5
60 -    R(:,:,5)= rotacion('y', 90+th1);
61 -    R(:,:,5)= R(:,:,5) * rotacion('z', 90+th6);
62
63      %Trama 7
64      %Posición de la articulación 7 respecto a 6
65 -    P(:,:,6)= [0;0;a3];
66      %Matriz de rotación de la junta 7 respecto a 6
67 -    R(:,:,6)= R(:,:,5);
68
69
70      %Creamos un vector de ceros
71 -    Vector_Zeros= zeros(1, 3);
72
73      %Inicializamos las matrices de transformación Homogénea locales
74 -    A(:,:,GDL)=simplify([R(:,:,GDL) P(:,:,GDL); Vector_Zeros 1]);
75      %Inicializamos las matrices de transformación Homogénea globales
76 -    T(:,:,GDL)=simplify([R(:,:,GDL) P(:,:,GDL); Vector_Zeros 1]);
77      %Inicializamos las posiciones vistas desde el marco de referencia inercial
78 -    PO(:,:,GDL)= P(:,:,GDL);
79      %Inicializamos las matrices de rotación vistas desde el marco de referencia inerci
80 -    RO(:,:,GDL)= R(:,:,GDL);
```

```
83 -  for i = 1:GDL
84 -      i_str= num2str(i);
85       %disp(strcat('Matriz de Transformación local A', i_str));
86 -     A(:,:,i)=simplify([R(:,:,i) P(:,:,i); Vector_Zeros 1]);
87       %pretty (A(:,:,i));
88
89       %Globales
90 -     try
91 -        T(:,:,i)= T(:,:,i-1)*A(:,:,i);
92 -     catch
93 -        T(:,:,i)= A(:,:,i);
94 -     end
95 -     disp(strcat('Matriz de Transformación global T', i_str));
96 -     T(:,:,i)= simplify(T(:,:,i));
97 -     pretty(T(:,:,i))
98
99 -     RO(:,:,i)= T(1:3,1:3,i);
100 -    PO(:,:,i)= T(1:3,4,i);
101      %pretty(RO(:,:,i));
102      %pretty(PO(:,:,i));
103 - end
```

07

CODIGO

```
107      %Calculamos el jacobiano lineal de forma analítica
108 -    Jv_a(:,GDL)=PO(:,:,GDL);
109 -    Jw_a(:,GDL)=PO(:,:,GDL);
110
111 -  for k= 1:GDL
112 -      if RP(k)==0
113           %Para las juntas de revolución
114 -         try
115 -             Jv_a(:,k)= cross(RO(:,3,k-1), PO(:,:,GDL)-PO(:,:,k-1));
116 -             Jw_a(:,k)= RO(:,3,k-1);
117 -         catch
118 -             Jv_a(:,k)= cross([0,0,1], PO(:,:,GDL));%Matriz de rotación de 0 co
119 -             Jw_a(:,k)=[0,0,1];%Si no hay matriz de rotación previa se obtiene
120 -          end
121 -      else
122      %         %Para las juntas prismáticas
123 -         try
124 -             Jv_a(:,k)= RO(:,3,k-1);
125 -         catch
126 -             Jv_a(:,k)=[0,0,1];
127 -         end
128 -             Jw_a(:,k)=[0,0,0];
129 -      end
130 -  end
```

```
132 -    Jv_a= simplify (Jv_a);
133 -    Jw_a= simplify (Jw_a);
134      %disp('Jacobiano lineal obtenido de forma analítica');
135      %pretty (Jv_a);
136      %disp('Jacobiano ángular obtenido de forma analítica');
137      %pretty (Jw_a);
138
139
140 -    disp('Velocidad lineal obtenida mediante el Jacobiano lineal');
141 -    V=simplify (Jv_a*Qp');
142 -    pretty(V);
143 -    disp('Velocidad angular obtenida mediante el Jacobiano angular');
144 -    W=simplify (Jw_a*Qp');
145 -    pretty(W);
```

CODIGO

```
147     %% Funcion para hacer rotacion en cualquiera de los 3 ejes
148
149     function matriz_rotacion = rotacion(eje, theta)
150
151         % Validar el eje de rotación y construir la matriz de rotación correspondiente
152 -       switch eje
153 -           case 'x'
154 -               matriz_rotacion = [1       0             0;
155                                    0  cos(theta) -sin(theta);
156                                    0  sin(theta)  cos(theta)];
157 -           case 'y'
158 -               matriz_rotacion = [ cos(theta)  0  sin(theta);
159                                           0       1       0    ;
160                                    -sin(theta)  0  cos(theta)];
161 -           case 'z'
162 -               matriz_rotacion = [cos(theta)  -sin(theta)  0;
163                                    sin(theta)   cos(theta)  0;
164                                        0             0      1];
165 -           otherwise
166 -               error('Eje de rotación no válido. Los valores válidos son "x", "y" o "z"');
167 -       end
168 -   end
```

CODIGO

```
Velocidad angular obtenida mediante el Jacobiano angular
--                                                    2                                2
|  [#4 #12 + #3 #12 + #5 #16 + #6 #24 - #2 (#24 (#7 (#15 #18  + #17 #16 #19 - #15 #20 #17 )
--

   + #18 #1 (#15 #17 - #16 #19 + #15 #20 #17)) - #23 #12)],

  [#2 (#23 #8 + #24 (#7 (#20 #17 #13 - #18 #10 + #15 #17 #19 #21) - #1 (#17 #10 + #20 #18 #13 + #15 #19 #21 #18)))

   + #4 #8 + #3 #8 - #5 #15 #21 - #6 #23 #21],

  --      /  d        \
  |  conj| -- th1(t) | - #2 (#24 (#7 (#20 #17 #14 - #18 #11 + #15 #22 #17 #19) - #1 (#17 #11 + #20 #18 #14 + #15 #22 #19 #18))
  --      \ dt        /

                                                        -- --
   + #23 #9) - #4 #9 - #3 #9 + #5 #15 #22 + #6 #23 #22  |  |
                                                        -- --
```

THANK YOU
FOR YOUR ATTENTION