UNIVERSIDAD TECNOLÓGICA NACIONAL – FRBA

INGENIERÍA ELECTRÓNICA

Asignatura: Robótica
Curso: R6051

Código: 95-0482
Año: 2011

Trabajo Práctico n° 2:

Análisis dinámico de la plataforma Stewart

Alumnos:

Cignoli, Rodolfo
Pietrasanta, Agustín

Profesor: Mas. Ing. GIANNETA, Hernán

JTP: Ing. GRANZELLA, Damián

Objetivos:

- Obtención del modelo dinámico inverso de una plataforma Stewart genérica.
- Implementación en Matlab.
- Resultados de la simulación.

Introducción:

La dinámica es la parte de la física que describe la evolución en el tiempo de un sistema físico en relación a las causas que provocan los cambios de estado físico y/o estado de movimiento.
El objetivo de la dinámica es describir los factores capaces de producir alteraciones de un sistema físico, cuantificarlos y plantear ecuaciones de movimiento o ecuaciones de evolución para dicho sistema.

Por lo tanto, el modelo dinámico de un robot tiene por objetivo conocer la relación entre el movimiento del robot y las fuerzas implicadas en el mismo.
Esta relación se obtiene mediante el denominado modelo dinámico, que relaciona matemáticamente:
La localización del robot definida por sus variables articulares o por las coordenadas de localización de su extremo, y sus derivadas: velocidad y aceleración.
Las fuerzas y pares aplicados en las articulaciones (o en el extremo del robot).
Los parámetros dimensionales del robot, como longitud, masas e inercias de sus elementos.

Dinámica de la plataforma Stewart:

Las ecuaciones clásicas de movimiento pueden obtenerse de la ecuación de Lagrange.

Donde “q” representa coordenadas, y “” representa fuerzas, es la energía cinética y es la energía potencial. Esta misma expresión puede reformularse de la siguiente manera:

Donde y son la Matriz de Inercia y la Matriz de Coriolis respectivamente. F representa las fuerzas de cada actuador o “pierna”.
Para obtener la energía cinética y potencial, el estudio de la plataforma Stewart puede dividirse en dos partes, la plataforma superior y las seis piernas. Sin embargo en este trabajo, se despreciará la masa de los actuadores frente a la parte superior, como así también las fuerzas debidas al efecto Coriolis, por lo que la matriz C(q) no tendrá efecto en nuestra simulación.
Del estudio analítico que puede verse en la referencia [1] se obtiene la matriz M(q).

$$M(q)=\begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_X C_\gamma^2 + I_Y S_\gamma^2 & (I_X - I_Y) C_\alpha C_\gamma S_\gamma & 0 \\ 0 & 0 & 0 & (I_X - I_Y) C_\alpha C_\gamma S_\gamma & C_\alpha^2 (I_X S_\gamma^2 + I_Y C_\gamma^2) + I_Z S_\alpha^2 & -I_Z S_\alpha \\ 0 & 0 & 0 & 0 & -I_Z S_\alpha & I_Z \end{bmatrix}$$

Donde “m” es la masa de la plataforma superior y los términos Ix, Iy e Iz son componentes del tensor de inercias I = diag{Ix Iy Iz}.

Cálculo del Tensor de Inercia

El tensor de inercia es un tensor simétrico de segundo orden que caracteriza la inercia rotacional de un sólido rígido.

Expresado en una base ortonormal viene dado por una matriz simétrica, dicho tensor se forma a partir de los momentos de inercia según tres ejes perpendiculares y tres productos de inercia.

El tensor de inercia sólido rígido se define como un tensor simétrico de segundo orden tal que la forma cuadrática construida a partir del tensor y la velocidad angular Ω da la energía cinética de rotación, es decir:

$$E_{rot} = \frac{1}{2} \begin{pmatrix} \Omega_x & \Omega_y & \Omega_z \end{pmatrix} \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix} \begin{pmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{pmatrix} = \frac{1}{2} \sum_j \sum_k I_{jk} \Omega_j \Omega_k$$

Donde las componentes de este tensor de inercia en una base ortonormal XYZ pueden calcularse a partir de los tres momentos de inercia según esos tres ejes perpendiculares:

$$\begin{cases} I_{xx} = \int_M d_x^2 dm = \int_V \rho(y^2 + z^2) dx dy dz \\ I_{yy} = \int_M d_y^2 dm = \int_V \rho(z^2 + x^2) dx dy dz \\ I_{zz} = \int_M d_z^2 dm = \int_V \rho(x^2 + y^2) dx dy dz \end{cases}$$

Donde es la densidad del material.

Cálculo Matriz Potencial

$$P = mgZ = \begin{bmatrix} 0 \\ 0 \\ mg \\ 0 \\ 0 \\ 0 \end{bmatrix}^T \begin{bmatrix} X \\ Y \\ Z \\ \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} 0 & 0 & mg & 0 & 0 & 0 \end{bmatrix} q$$

Entonces

$$G(q) = \frac{\partial P(q)}{\partial q} = \begin{bmatrix} 0 & 0 & mg & 0 & 0 & 0 \end{bmatrix}$$

Cálculo Matriz Jacobiana

Las ecuaciones de L1…L6 (ver referencia [2]), dan explícitamente las longitudes de los actuadores en función de las coordenadas de la plataforma superior (XT1, YT1, ZT1), y luego esta se encuentran en función del centro de la misma q = (X,Y,Z,alpha,beta,gamma). Por lo tanto sus velocidades serán:

Resolución de dinámica inversa

El método para resolver la dinámica inversa, es que "q" es identificado como una entrada y a la vez es función del tiempo. Por ello puede obtenerse su derivada primera y segunda, logrando así la velocidad y aceleración. Usando esto como dato, en la ecuación de Lagrange anteriormente planteada, se resuelve el sistema de ecuaciones para obtener las "F", que determinan una trayectoria.

Resolución numérica e Implementación en Matlab

Consideramos las dimensiones físicas de la plataforma como en la referencia [1].

a = 0.2 (m)
b = 0.4 (m)
d = 0.1 (m)
m = 1.36 (kg)
g = 9.8066 (m/s 2)

$$q = \begin{bmatrix} -0.1 \\ 0.1\,\mathrm{Sin}(\omega t) \\ 0.7 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \omega = 2\,(\mathrm{rad/s})$$

Script GraficoStewart.m

```
syms alpha beta gamma ; %alpha: rot en x
                        %beta: rot en y
                        %gamma: rot en z

syms px py pz;  %Distancia al origen del TOP respecto de BASE

%Parametros BASE stewart
b=0.400;    %m Lado largo de la base
d=0.100;   %m Lado corto de la base

%Parametro de TOP
%Parametros para el tensor de inercia
dens = 7580; %kg/m3
espesor_chapa = 0.01036; %m
masa = 1.36; %masa kg
a=0.200;  %m Lado TOP
g=9.81; %gravedad m/seg2

%Calculo cinematica inversa. Determina las ecuaciones de L1...L6
Stewart;

%Calcula el tensor de inercia.
%Determina la los elementos de tensor de inercia. Ixx, Iyy, Izz.
i_tensor;

%Calcula expresion de matriz jacobiana para obtener velocidades de L y la
%deja cargada en Jsymbolic.
Jacobiana;

%Inercias para energia cinetica. Queda cargada en Msymbolic
MatrizInercia;

%Energia potencial. Queda cargada en Gsymbolic.
MatrizPotencial;

%Generar trayectorias. Y obtiene Fuerzas para los actuadores.
%Grafica.
Generador;
```

Script Stewart.m

Es el mismo script utilizado en el TP1, dado que utiliza las mismas ecuaciones de cinemática inversa.

```
Xb1 = (sqrt(3)/6)*(2*b+d);
Yb1 = (1/2)*d;
Zb1 = 0;
```

```
Xb2 = -(sqrt(3)/6)*(b-d);
Yb2 = (1/2)*(b+d);
Zb2 = 0;

Xb3 = -(sqrt(3)/6)*(b+2*d);
Yb3 = (1/2)*b;
Zb3 = 0;

Xb4 = -(sqrt(3)/6)*(b+2*d);
Yb4 = -(1/2)*b;
Zb4 = 0;

Xb5 = -(sqrt(3)/6)*(b-d);
Yb5 = -(1/2)*(b+d);
Zb5 = 0;

Xb6 = (sqrt(3)/6)*(2*b+d);
Yb6 = -(1/2)*d;
Zb6 = 0;

%Con subindice en minuscula es respecto a TOP
%Con mAyuscula respecto a la base
xt1 = (sqrt(3)/6)*a;
yt1 = (1/2)*a;
zt1 = 0;

xt2 = -(sqrt(3)/3)*a;
yt2 = 0;
zt2 = 0;


xt3 = (sqrt(3)/6)*a;
yt3 = -(1/2)*a;
zt3 = 0;



TTB = [cos(beta)*cos(gamma)+sin(alpha)*sin(beta)*sin(gamma) ...
        -cos(beta)*sin(gamma)+sin(alpha)*sin(beta)*cos(gamma) ...
        cos(alpha)*sin(beta) px;...

        cos(alpha)*sin(gamma)...
        cos(alpha)*cos(gamma)...
        -sin(alpha) py;...

        -sin(beta)*cos(gamma)+sin(alpha)*cos(beta)*sin(gamma) ...
         sin(beta)*sin(gamma)+sin(alpha)*cos(beta)*cos(gamma) ...
         cos(alpha)*cos(beta) pz;...

         0 0 0 1;...
        ];

XYZt1 = TTB * [xt1 yt1 zt1 1]';
XYZt2 = TTB * [xt2 yt2 zt2 1]';
XYZt3 = TTB * [xt3 yt3 zt3 1]';
```

```
Xt1=XYZt1(1);
Yt1=XYZt1(2);
Zt1=XYZt1(3);

Xt2=XYZt2(1);
Yt2=XYZt2(2);
Zt2=XYZt2(3);

Xt3=XYZt3(1);
Yt3=XYZt3(2);
Zt3=XYZt3(3);

L1 = sqrt((Xt1 - d/(2*sqrt(3)) - b/sqrt(3))^2 + (Yt1 - d/2)^2 + Zt1^2 );
L2 = sqrt((Xt1 - d/(2*sqrt(3)) + b/(2*sqrt(3)))^2 + (Yt1 - d/2 - b/2)^2 +
Zt1^2 );
L3 = sqrt((Xt2 + d/(sqrt(3)) + b/(2*sqrt(3)))^2 + (Yt2 - b/2)^2 + Zt2^2 );
L4 = sqrt((Xt2 + d/(sqrt(3)) + b/(2*sqrt(3)))^2 + (Yt2 + b/2)^2 + Zt2^2 );
L5 = sqrt((Xt3 - d/(2*sqrt(3)) + b/(2*sqrt(3)))^2 + (Yt3 + b/2 + d/2)^2 +
Zt3^2 );
L6 = sqrt((Xt3 - d/(2*sqrt(3)) - b/sqrt(3))^2 + (Yt3 + d/2)^2 + Zt3^2 );
```

Script i_tensor.m

```
syms x y z

Ixx = dens * int(int(int((y^2+z^2),z,0,espesor_chapa),y, (x-xt2)*((yt3-yt2)/
(xt3-xt2)), (x-xt2)*((yt1-yt2)/(xt1-xt2))),x,xt2,xt1);
Iyy = dens * int(int(int((x^2+z^2),z,0,espesor_chapa),y, (x-xt2)*((yt3-yt2)/
(xt3-xt2)), (x-xt2)*((yt1-yt2)/(xt1-xt2))),x,xt2,xt1);
Izz = dens * int(int(int((y^2+x^2),z,0,espesor_chapa),y, (x-xt2)*((yt3-yt2)/
(xt3-xt2)), (x-xt2)*((yt1-yt2)/(xt1-xt2))),x,xt2,xt1);
```

Script Jacobiana.m

```
%Calcula la matriz jacobiana total para la obtencion de las velocidades de
%L1...L6
f1  = [L1; L2; L3; L4; L5; L6];
v1 =  [px, py, pz, alpha, beta, gamma];
Jsymbolic = jacobian(f1,v1);
```

Script MatrizInercia.m

```
%Matriz inercia M(q)

Msymbolic = [masa,      0,      0,      0,              0,      0;...
                0,      masa,   0,      0,              0,      0;...
                0,      0,      masa,   0,              0,      0;...
                0,      0,      0,      Ixx*cos(gamma)^2+Iyy*sin(gamma)^2,
(Ixx-Iyy)*cos(alpha)*cos(gamma)*sin(gamma), 0;...
                0,      0,      0,      (Ixx-
Iyy)*cos(alpha)*cos(gamma)*sin(gamma),
(cos(alpha)^2)*(Ixx*(sin(gamma)^2)+Iyy*(cos(gamma)^2))+Izz*(sin(alpha)^2),
-Izz*sin(alpha);...
                0,      0,      0,      0,              -Izz*sin(alpha),
Izz];
```

Script MatrizPotencial.m

```
%Matriz de Energia potencial
Gsymbolic = [0; 0; masa*g; 0; 0; 0];
```

Script Generador.m

```
%Genreamos vector de estado. Con trayectoria.

w=2;%2 r/seg

figure(1);
xlim([0 2]);
ylim([-2 6]);

f_ant1=0;
t_ant1=0;
f_ant2=0;
t_ant2=0;
f_ant3=0;
t_ant3=0;
f_ant4=0;
t_ant4=0;
f_ant5=0;
t_ant5=0;
f_ant6=0;
t_ant6=0;
for t=0:0.1:4

px = -0.1;
py = 0.1*sin(w*t);
pz = 0.7;
alpha = 0;
beta = 0;
gamma = 0;

q   = [px; py; pz; alpha; beta; gamma];

qdd = [0; -0.4*sin(w*t); 0; 0; 0; 0];

B = eval(Msymbolic*qdd + Gsymbolic);
A = eval(Jsymbolic');

F = linsolve(A,B);

%Grafico las 6 curvas de fuerzas para cada pierna.
Faux1=[ f_ant1 F(1)];
taux1=[ t_ant1 t];
Faux2=[ f_ant2 F(2)];
taux2=[ t_ant2 t];
Faux3=[ f_ant3 F(3)];
taux3=[ t_ant3 t];
Faux4=[ f_ant4 F(4)];
taux4=[ t_ant4 t];
Faux5=[ f_ant5 F(5)];
taux5=[ t_ant5 t];
Faux6=[ f_ant6 F(6)];
taux6=[ t_ant6 t];
```

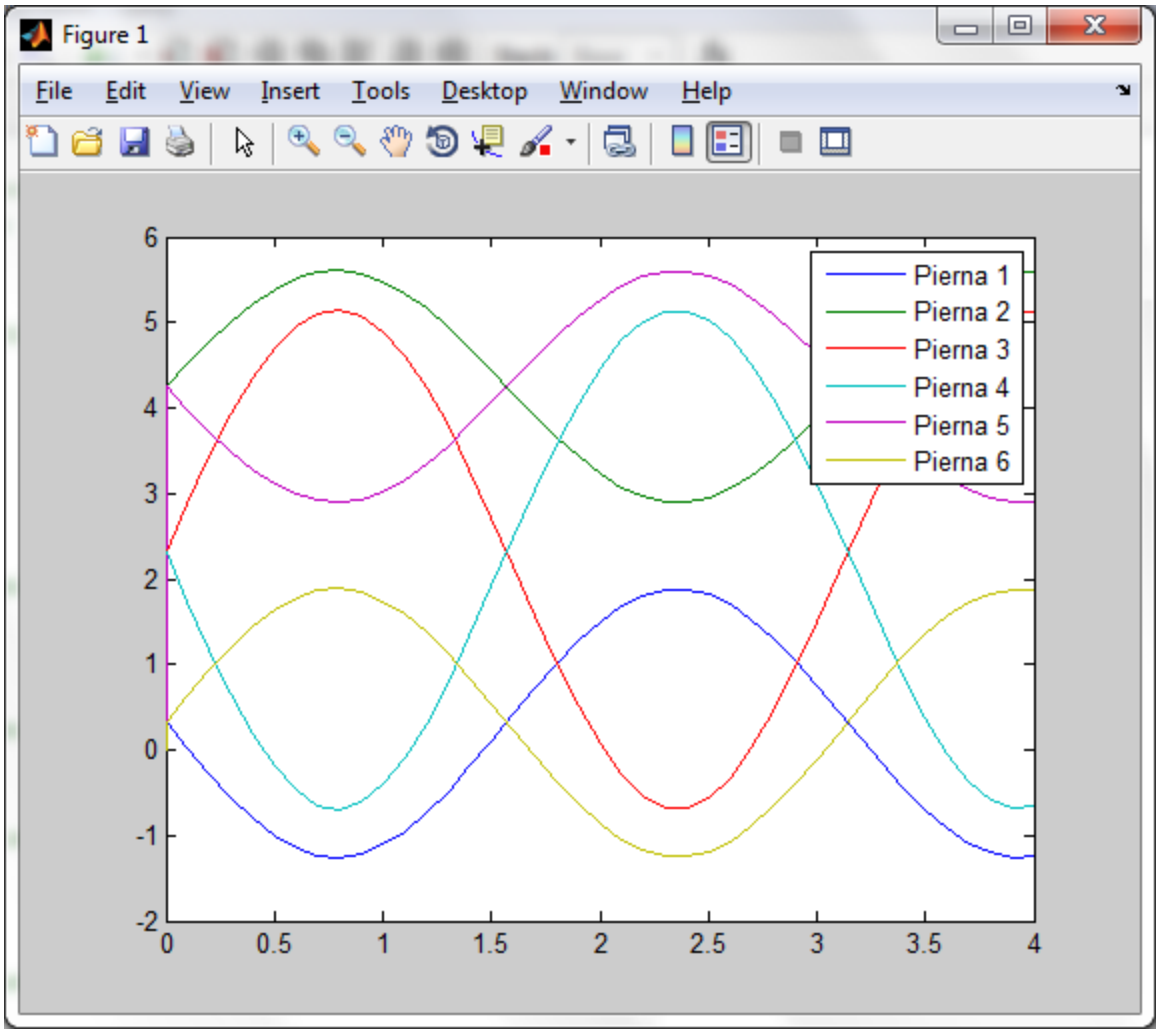
```
plot(taux1, Faux1, taux2, Faux2,taux3, Faux3,taux4, Faux4,taux5, Faux5,taux6,
Faux6);
hleg = legend('Pierna 1','Pierna 2','Pierna 3', 'Pierna 4', 'Pierna 5',
'Pierna 6');

t_ant1=t;
f_ant1=F(1);
t_ant2=t;
f_ant2=F(2);
t_ant3=t;
f_ant3=F(3);
t_ant4=t;
f_ant4=F(4);
t_ant5=t;
f_ant5=F(5);
t_ant6=t;
f_ant6=F(6);

hold on;
drawnow;

end
hold off;
```

Resultados Simulación



Conclusiones

El análisis de la dinámica inversa en la plataforma Stewart, es un mapeo desde la posición y orientación de la plataforma superior en función del tiempo (“q”) hacia la fuerza aplicada por cada uno de los actuadores. Pudimos simular este sistema en matlab, y obtuvimos resultados idénticos a los publicados en el paper de la referencia [1], habiendo solo nosotros despreciado el efecto de la matriz de Coriolis, confirmando que hicimos bien en hacerlo.
Por otra parte, vimos que es sencillo expresar matemáticamente el método Lagrangiano utilizado, pero, no obstante, se vuelve un tanto lento porque para cada iteración se debe recalcular la matriz Jacobiana.

Referencias

[1] Dynamics Analysis and Simulation of Parallel Robot Stewart Platform. Hamidreza Hajimirzaalian. Hasan Moosavi. Mehdi Massah.
[2] Trabajo práctico Nº1, Análisis cinematic de la plataforma Stewart. Rodolfo Cignoli, Agustín Pietrasanta.