Fundamentación de robótica (Gpo 101) Martes 28 de Febrero del 2023
Antonio Silva Martínez
A01173663
José Jezarel Sánchez Mijares
A01735226
Frida Lizett Zavala Pérez
A01275226

Actividad 2 (Modelado de Energía Cinética)

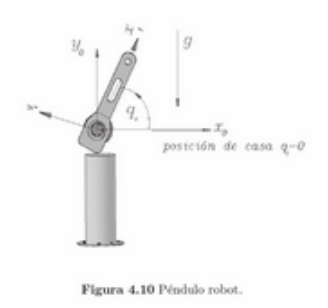
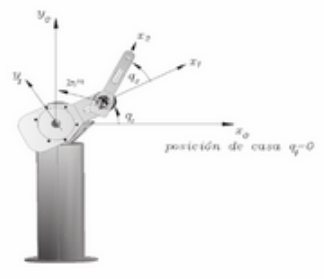
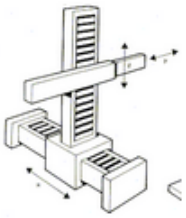


Figura 4.10 Péndulo robot.

Robot Péndulo (1gdl)



Robot Rotacional (2gdl)



Robot Cartesiano (3gdl)

El modelado de energía cinética se refiere al proceso de crear modelos matemáticos que describen cómo cambia la energía cinética de un objeto en movimiento. La energía cinética se refiere a la energía que un objeto posee debido a su movimiento, y se puede calcular como la mitad de la masa del objeto multiplicada por su velocidad al cuadrado.
El modelado de energía cinética se utiliza en una amplia variedad de aplicaciones, desde la física y la ingeniería hasta la animación por computadora y los videojuegos. Por ejemplo, en física e ingeniería, se pueden usar modelos de energía cinética para entender cómo un objeto en movimiento interactúa con su entorno o para diseñar sistemas de frenado o amortiguación para reducir la velocidad de un objeto en movimiento.
En la animación por computadora y los videojuegos, los modelos de energía cinética se pueden utilizar para crear animaciones realistas de objetos en movimiento, como coches que chocan o pelotas que rebotan. También se pueden utilizar para simular cómo un personaje en un videojuego se mueve y responde a diferentes estímulos, como saltar o correr.

Análisis de los códigos:

El funcionamiento de los códigos que se mostrarán a continuación es similar, esto se debe a que el análisis que se ha hecho a estos es el mismo, siendo que únicamente lo que cambia es el tipo de robot que se está analizando y las cantidades de articulaciones con las que este cuenta.

En esta primera parte del código lo que se realiza es la declaración de variables, la inicialización de vectores y el guardado de datos dentro de los vectores que utilizaremos.

```
%Limpieza de pantalla
clear all
```

```
close all
clc
tic
%Declaración de variables simbólicas
syms th1(t)  t  %Angulos de cada articulación
syms m1 Ixx1 Iyy1 Izz1  %Masas y matrices de Inercia
syms t1 %Tiempos
syms l1 lc1  %l=longitud de eslabones y lc=distancia al centro de masa
de cada eslabón
syms pi g
%Creamos el vector de coordenadas articulares
 Q= [th1];
%disp('Coordenadas generalizadas');
%pretty (Q);
%Creamos el vector de velocidades articulares
 Qp= diff(Q, t);
%disp('Velocidades generalizadas');
%pretty (Qp);
%Creamos el vector de aceleraciones articulares
 Qpp= diff(Qp, t);
%disp('Aceleraciones generalizadas');
%pretty (Qpp);
%Configuración del robot, 0 para junta rotacional, 1 para junta
prismática
RP=[0];
%Número de grado de libertad del robot
GDL= size(RP,2);
GDL_str= num2str(GDL);
%Articulación 1
%Posición de la articulación 1 respecto a 0
```

Se utilizan las matrices de rotación para poder posteriormente guardar los valores dentro de las matrices.

```
P(:,:,1)= [0;0;l1];
%Matriz de rotación de la junta 1 respecto a 0....
R(:,:,1)= [cos(th1) -sin(th1)  0;
          sin(th1)  cos(th1)  0;
          0         0         1];
%Creamos un vector de ceros
Vector_Zeros= zeros(1, 3);
%Inicializamos las matrices de transformación Homogénea locales
A(:,:,GDL)=simplify([R(:,:,GDL) P(:,:,GDL); Vector_Zeros 1]);
%Inicializamos las matrices de transformación Homogénea globales
T(:,:,GDL)=simplify([R(:,:,GDL) P(:,:,GDL); Vector_Zeros 1]);
%Inicializamos las posiciones vistas desde el marco de referencia
```

```
inercial
PO(:,:,GDL)= P(:,:,GDL);
%Inicializamos las matrices de rotación vistas desde el marco de
referencia inercial
RO(:,:,GDL)= R(:,:,GDL);
```

Dentro de este ciclo for lo que realizaremos será guardar los datos obtenidos de la matriz en una matriz local y una matriz global para poder usarla posteriormente

```
for i = 1:GDL
   i_str= num2str(i);
  %disp(strcat('Matriz de Transformación local A', i_str));
   A(:,:,i)=simplify([R(:,:,i) P(:,:,i); Vector_Zeros 1]);
  %pretty (A(:,:,i));
  %Globales
   try
      T(:,:,i)= T(:,:,i-1)*A(:,:,i);
   catch
      T(:,:,i)= A(:,:,i);
   end
%     disp(strcat('Matriz de Transformación global T', i_str));
   T(:,:,i)= simplify(T(:,:,i));
%     pretty(T(:,:,i))
   RO(:,:,i)= T(1:3,1:3,i);
   PO(:,:,i)= T(1:3,4,i);
   %pretty(RO(:,:,i));
   %pretty(PO(:,:,i));
end
```

Calculamos los jacobianos a través de una derivación respecto a las thetas que utilizamos en el sistema y posteriormente se crea una matriz de jacobianos para poder calcular el jacobiano lineal

```
%Calculamos el jacobiano lineal de forma diferencial
%disp('Jacobiano lineal obtenido de forma diferencial');
%Derivadas parciales de x respecto a th1 y th2
Jv11= functionalDerivative(PO(1,1,GDL), th1);
%Derivadas parciales de y respecto a th1 y th2
Jv21= functionalDerivative(PO(2,1,GDL), th1);
%Derivadas parciales de z respecto a th1 y th2
Jv31= functionalDerivative(PO(3,1,GDL), th1);
%Creamos la matríz del Jacobiano lineal
jv_d=simplify([Jv11;
               Jv21;
               Jv31]);
```

```
%pretty(jv_d);
%Calculamos el jacobiano lineal de forma analítica
Jv_a(:,GDL)=PO(:,:,GDL);
Jw_a(:,GDL)=PO(:,:,GDL);
```

Utilizamos los jacobianos antes obtenidos para poder encontrar las submatrices de los jacobianos y así obtener los jacobianos lineales y angulares

```
for k= 1:GDL
   if RP(k)==0
      %Para las juntas de revolución
       try
           Jv_a(:,k)= cross(RO(:,3,k-1), PO(:,:,GDL)-PO(:,:,k-1));
           Jw_a(:,k)= RO(:,3,k-1);
       catch
           Jv_a(:,k)= cross([0,0,1], PO(:,:,GDL));%Matriz de rotación de
0 con respecto a 0 es la Matriz Identidad, la posición previa tambien
será 0
           Jw_a(:,k)=[0,0,1];%Si no hay matriz de rotación previa se
obtiene la Matriz identidad
        end
    else
%         %Para las juntas prismáticas
       try
           Jv_a(:,k)= RO(:,3,k-1);
       catch
           Jv_a(:,k)=[0,0,1];
       end
           Jw_a(:,k)=[0,0,0];
    end
end

%Obtenemos SubMatrices de Jacobianos
Jv_a= simplify (Jv_a);
Jw_a= simplify (Jw_a);
%disp('Jacobiano lineal obtenido de forma analítica');
%pretty (Jv_a);
%disp('Jacobiano ángular obtenido de forma analítica');
%pretty (Jw_a);
%Matriz de Jacobiano Completa
%disp('Matriz de Jacobiano');
Jac= [Jv_a;
     Jw_a];
Jacobiano= simplify(Jac);
```

```
% pretty(Jacobiano);
%Obtenemos vectores de Velocidades Lineales y Angulares
% disp('Velocidad lineal obtenida mediante el Jacobiano lineal');
V=simplify (Jv_a*Qp);
% pretty(V);
% disp('Velocidad angular obtenida mediante el Jacobiano angular');
W=simplify (Jw_a*Qp);
%     pretty(W);
```

En la siguiente parte del código obtenemos la Energía Cinética a través de la distancia de los eslabones a través de los vectores y la matriz de inercia de cada eslabón, es por esto que al solo contar con una articulación únicamente contaremos con una matriz de inercia para el eslabón, posteriormente extraemos la velocidad lineal de cada eje y obtenemos las velocidades angulares de euler.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%Energía Cinética
%Distancia del origen del eslabón a su centro de masa
%Vectores de posición respecto al centro de masa
P01=subs(P(:,:,1)/2, l1, lc1);
%Creamos matrices de inercia para cada eslabón
I1=[Ixx1 0 0;
   0 Iyy1 0;
   0 0 Izz1];
%Función de energía cinética
%Extraemos las velocidades lineales en cada eje
V=V(t);
Vx= V(1,1);
Vy= V(2,1);
Vz= V(3,1);
%Extraemos las velocidades angular en cada ángulo de Euler
W=W(t);
W_pitch= W(1,1);
W_roll= W(2,1);
W_yaw= W(3,1);
```

Calculamos las velocidades del cada eslabón y construimos la matriz jacobiana para cada articulación posteriormente obtenemos los vectores de velocidad angular y lineal para finalmente se suman todas las velocidades obtenidas para poder obtener una velocidad total

```
%Calculamos las velocidades para cada eslabón
%Matriz de Jacobiano Completa
```

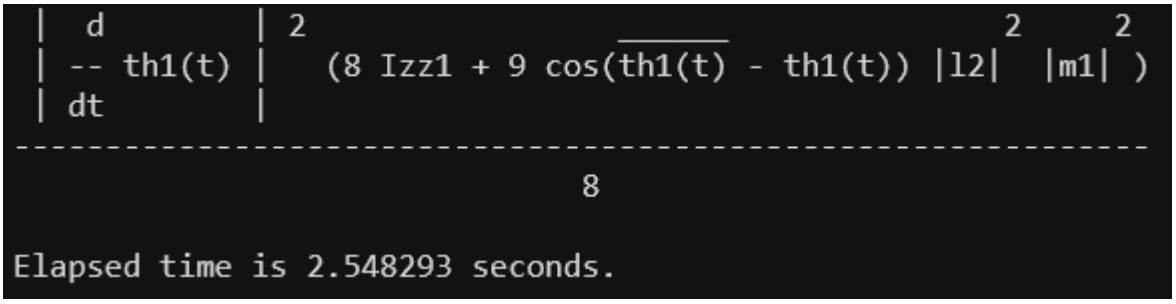
```
%disp('Matriz de Jacobiano');
Jac1= [Jv_a;
     Jw_a];
Jacobiano1= simplify(Jac1);
% pretty(Jacobiano);
%Obtenemos vectores de Velocidades Lineales y Angulares
%disp('Velocidad lineal obtenida mediante el Jacobiano lineal del
Eslabón 1');
Qp=Qp(t);
V1=simplify (Jv_a*Qp(1));
%pretty(V1);
% disp('Velocidad angular obtenida mediante el Jacobiano angular del
Eslabón 1');
W1=simplify (Jw_a*Qp(1));
% pretty(W1);
%Eslabón 1
V1_Total= V1+cross(W1,P01);
K1= (1/2*m1*(V1_Total))'*(1/2*m1*(V1_Total)) + (1/2*W1)'*(I1*W1);
%disp('Energía Cinética en el Eslabón 1');
K1= simplify (K1);
%pretty (K1);
K_Total= simplify (K1);
pretty(K_Total)
toc
```

Resultado obtenido

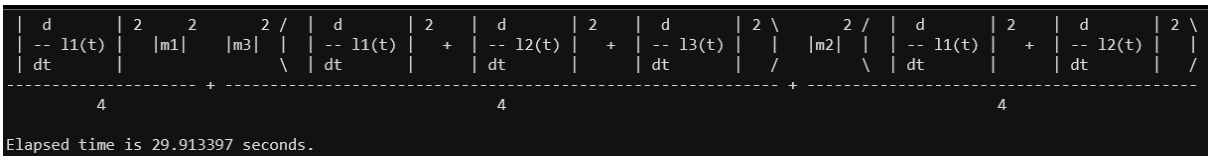
```
      |  d          | 2
Izz1  |  -- th1(t)  |
      |  dt         |
---------------------
          2

Elapsed time is 1.866187 seconds.
>>
```

En los códigos de los otros dos robots, el funcionamiento es similar, siendo que las únicas diferencias que encontraremos son que se agregan más variables para poder encontrar la velocidad en dichas articulaciones o eslabones, como es el caso del robot de 2 GDL y el de 3 GDL, por otra parte el análisis que haremos para poder manipular el robot de 2 y 3 GDL será diferente siendo que en el de 2 tendremos que tomar en cuenta el marco de referencia, mientras que en el de 3 gdl se afectará en el desplazamiento en los ejes.
Otro de los factores que cambian con respecto al primer modelo se encuentra con respecto a la cantidad de vectores con respecto al centro de masa, siendo que en este caso debemos de tomar en cuenta más elementos, también se tendrán que calcular una mayor cantidad de ciclos for para poder calcular los jacobianos del modelo que estamos analizando.

Finalmente estos son los resultados obtenidos del análisis de los robots de 2GDL y de 3GDL



2GDL



3GDL

Conclusiones:
El modelado de energía cinética es una consideración importante en el modelado de robots porque ayuda a comprender cómo el robot se moverá en diferentes situaciones y entornos. Al entender la energía cinética del robot, se pueden hacer predicciones precisas sobre cómo el robot se moverá, cómo responderá a diferentes fuerzas externas y cómo se puede optimizar su diseño y programación para realizar tareas específicas.
Además, el modelado de energía cinética es crucial para la seguridad del robot y de las personas que lo rodean. Si un robot está diseñado para moverse a alta velocidad o levantar objetos pesados, es importante que su energía cinética se modele y se comprenda completamente para evitar situaciones peligrosas. Por ejemplo, si un robot está levantando un objeto pesado y pierde el control debido a la energía cinética del objeto, podría causar daños a la propiedad o lesiones a las personas cercanas.
En resumen, el modelado de energía cinética es una herramienta esencial para el modelado de robots, ya que ayuda a predecir el comportamiento del robot, optimizar su diseño y programación, y garantizar su seguridad y la de las personas cercanas.