Pontificia Universidad JAVERIANA Bogotá

HERRAMIENTAS DE ROBÓTICA
Proyecto de Semestre - Parte 2 - 3
Diseño y modelado de un robot manipulador de 2GdL

- María Paula Carreño Fernández – carreno.mariap@javeriana.edu.co
- Sergio Lavao Osorio – s.lavaoo@javeriana.edu.co

PARTE 2: MODELADO CINEMÁTICO ROBOT

Para implementar nuestro robot manipulador de 2GdL primero se tiene en cuenta los parámetros geométricos de la matriz DH definidos en la primera entrega, siendo los siguientes:

Elemento	α	a	θ	d
1	0	0.1m	$\theta_{1=\pi/2}$	0.007m
2	0	0.1m	$\theta_{2=0}$	0

Tabla 1. Parámetros DH

Con base a la tabla 1, se define entonces en matlab la estructura del robot para poder implementar los algoritmos de la siguiente forma:

```
function [PLUMA, PARAM] = pluma_param()

L1=Link([  0     0     0.1        0       0  ],'standard');
L2=Link([  0     -0.007     0.1   0       0  ],'standard');

PLUMA = SerialLink([L1 L2]);
PLUMA.name='PLUMA';
```

Ilustración 2. Estructura del robot PLUMA

Además, se agregó nuestro modelo CAD del robot para las validaciones importando individualmente las partes del robot (base, brazo 1 y brazo 2) en formato STL y agregándolas a la carpeta model como link0, link1 y link2 respectivamente.

Luego se hizo uso de la propiedad de plot3d **'path'** y especificando la ruta, como resultado se obtiene nuestro modelo en Matlab:

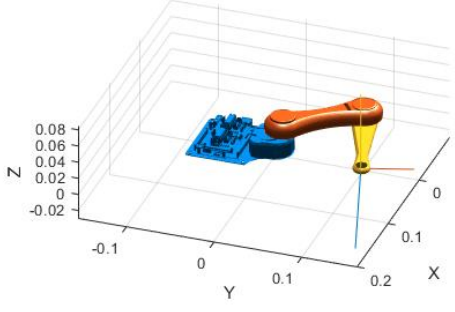

Ilustración 1. Modelo Pluma.

1. Resuelva el problema cinemático directo utilizando matrices de transformación homogénea. Implementar algoritmo en Matlab y validar con Toolbox de Robótica.

Implementando la función ForwardKinematics se tienen como parámetros de entrada la estructura del robot (ilustración 1) y las posiciones articulares q, a lo largo del tiempo de la trayectoria ya definida, para este caso se implementaron las q de los dos primeros grados de libertad del robot puma y se obtuvo lo siguiente:

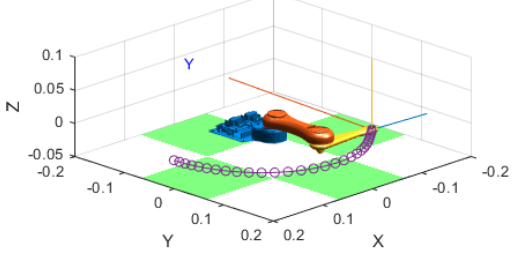

Ilustración 3. Validación ForwardKinematics pluma

*Como se emplearon las **q** definidas para el puma para el cual se tienen rotaciones en las articulaciones 2 y 3, en nuestro caso solo*

tendríamos movimiento en el brazo 2, por lo que se entiende que la trayectoria calculada final correspondería a un arco.

Se puede encontrar el código en el repositorio como ***plumaFKTest,*** *así como el gif del movimiento en results.*

2. Resuelva el problema cinemático inverso (método Jacobiano). Asegúrese que la solución permita encontrar todas las posibles configuraciones (poses) según los requerimientos de la tarea y las restricciones cinemáticas del brazo. Implementar algoritmo en Matlab y validar con Toolbox de Robótica.

Para este caso se definió una línea recta implementando la función Traj_Planner, que estuviera dentro del campo de trabajo del robot pluma, se hace uso de InverseKinematics y se grafican ambas trayectorias, con el propósito de comparar su comportamiento de la siguiente forma:

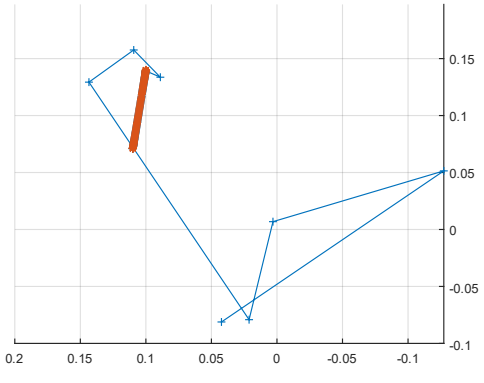

Ilustración 5. Trayectoria usando InverseKinematics

Como se observa existe un error al inicio de la trayectoria, que es debido a que la posición en reposo del robot no se encuentra en un punto cercano a la trayectoria que debe seguir, y como nuestro algoritmo no utiliza un lazo de control en el que se seleccione la trayectoria cuando exista un mínimo error, se tiene como consecuencia un comportamiento no deseado mientras se estabiliza.

Se tiene luego la validación con el robot pluma:

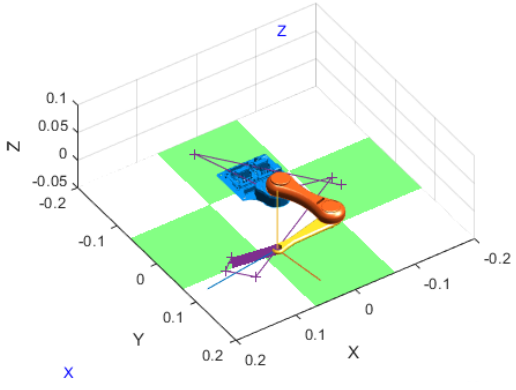

Ilustración 4. Validación InverseKinematics

3. Defina las soluciones de trayectorias cartesianas (interpoladas) requeridas para ejecutar las tareas propias del proceso (posiciones, velocidades y aceleraciones). Exprese las trayectorias mediante polinomios (resueltos). Grafique las trayectorias articulares correspondientes usado la solución cinemática inversa. Implementar algoritmo en Matlab y validar con Toolbox de Robótica.

Se encontraron los puntos de trayectoria importando el esquema del dibujo en blender, en donde se definen los puntos iniciales y finales de cada uno de los trazos que constituyen el dibujo a realizar, de cual se obtienen un total de 12 lineas, de las cuales se obtiene su punto inicial y final, se realiza una función que halle la trayectoria de cada linea y las concatene para obtener una unica matriz que describa toda la trayectoria a realizar. Se puede encontrar el desarrollo de la función como ***DrawTrajPlanner****, como resultado se obtiene:*

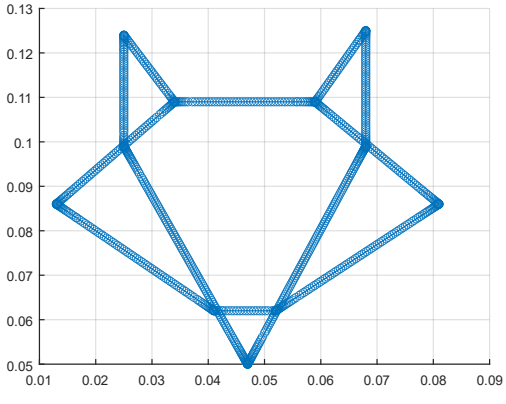

Ilustración 6. Puntos de trayectoria del dibujo

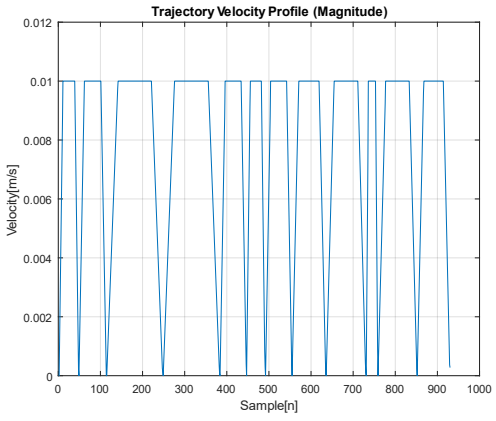

Ilustración 7. Perfil de velocidad, trayectoria total

Se compara la trayectoria deseada (ilustración 6) con la generada implementando Inverse Kinematics obteniendo como resultado:

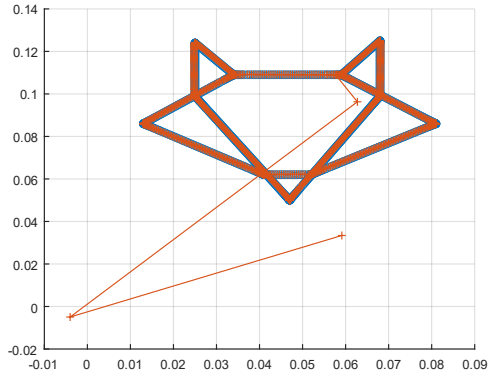

Ilustración 8. Comparación de trayectorias

Al implementar InverseKinematics se obtiene un error al inicio de la trayectoria como se explicó anteriormente. Cómo se resultado se tiene:

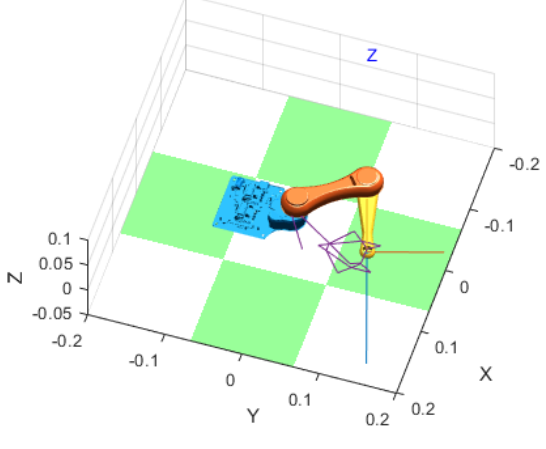

Ilustración 9. Validación trayectoria robot pluma

PARTE 3: MODELADO DINÁMICO ROBOT

4. Calcule el modelo dinámico inverso del brazo utilizando la formulación de Euler-Lagrange de acuerdo con su diseño cinemático y utilizando elementos estructurales simples (cuerpos simples).

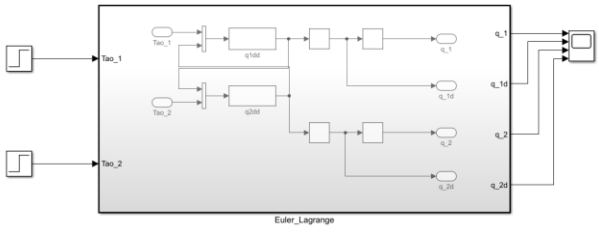

Ilustración 9. Modelo dinámico Euler Lagrange, Simulink.

Una vez se tiene el modelo en Simulink, se asignan (distancias y masas) en un archivo .mat.

Name	Value
lc_1	0.1000
lc_2	0.1000
m_1	0.0100
m_2	0.0100

Ilustración 10. Parámetros del robot.

Donde distancia lc_1[m], lc_2[m], m_1[kg] y m_2[kg].

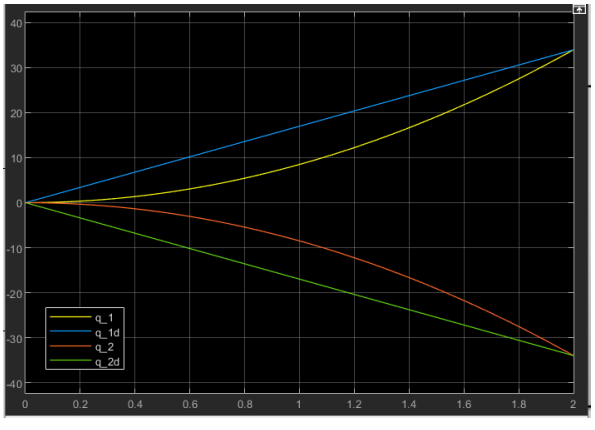

Ilustración 11. Parámetros del robot.

Es posible verificar el modelo dinamico analizando el comportamiento de las salidas en funcion de los torques de entrada, Tao_1 positivo y Tao_2 negativo, donde el resultado en el espacio articular para q_1 y q_2 debe ser exponencial(Posición), q_1d y q_2d lineal (velocidad).

Los archivos del modelo 3d y códigos de Matlab se pueden encontrar en el siguiente repositorio:
https://github.com/SergioLavao/PlumaBot