ARTE

A ROBOTICS TOOLBOX FOR EDUCATION

SESIÓN PRÁCTICA
PLANIFICACIÓN DE TRAYECTORIAS

Arturo Gil Aparicio

arturo.gil@umh.es

OBJETIVOS

Después de la realización de la práctica, el estudiante debería ser capaz de:

- Comprender las diferentes estrategias para planificar trayectorias, tanto en el espacio de la tarea como en el espacio articular.
- Implementar en Matlab diferentes estrategias para planificar el movimiento de las articulaciones de un robot.
- Implementar en Matlab diferentes estrategias para posicionar el extremo del robot en una posición/orientación determinadas.
- Considerar la existencia de diferentes soluciones de la cinemática inversa y proponer un algoritmo que pueda planificar una trayectoria continua en el espacio de la tarea.

1 Introducción

Considere que el robot se encuentra en una posición articular determinada $\mathbf{q}_0$. Considere, a continuación, que desea ubicar al robot en otra posición articular $\mathbf{q}_1$. El robot se deberá mover, por tanto, desde unos valores iniciales de los ángulos de las articulaciones hasta unos valores finales.

Indique, desde su punto de vista qué **datos de partida** necesita Ud. para realizar la planificación. Seguidamente, evalúe Ud. qué datos de salida serían interesante analizar para evaluar el desempeño de cada uno de los planificadores… p.e. el tiempo total en realizar el movimiento… las velocidades alcanzadas… etc.

Considere que el robot se encuentra en una posición articular determinada $\mathbf{q_0}$. Considere, a continuación, que desea ubicar al robot en otra posición articular $\mathbf{q}_1$. El robot se deberá mover entre $\mathbf{q}_0$ y $\mathbf{q}_1$ esto os debería abrir muchas incógnitas... pues el movimiento de las articulaciones debería ser suave… pero… ¿qué significa suave?. Igualmente, debería ser rápido y preciso. Se espera, también, que el robot evite obstáculos en el entorno. Finalmente, deberíamos contar con alguna función que permitiera que el robot realizara una trayectoria recta en el espacio de la tarea y también cambios de orientación.

Anote aquí qué otras características considera deseables para un módulo de planificación de trayectorias en un robot de tipo serie.

Por otra parte, en un lenguaje de programación (RAPID de ABB o KRL de KUKA) son típicas dos tipos de funciones:

- funciones que permiten llevar al robot entre dos posiciones articulares.

- funciones que permiten realizar una trayectoria recta en el espacio de la tarea interpolando linealmente la posición y orientación del extremo entre dos puntos.

- Otro tipo de trayectorias: p.e. trayectorias circulares.

1.1 Definiciones

Debemos distinguir entre:

- **tipos de trayectoria**: Indica cuál es la función que mejor aproxima la trayectoria. Esta trayectoria puede referirse a una trayectoria articular o bien a una trayectoria en el espacio de la tarea. P.e. podemos hablar de una trayectoria de primer orden de la forma $q_1(t) = k_1 + k_2 \cdot t$, donde las constantes k1 y k2 definen el movimiento de q1 en un periodo de tiempo t. Hablaremos de trayectorias de primer orden, de segundo orden... etc.

- **tipos de planificadores:** plantea cómo se calculan las trayectorias anteriores para obtener diferentes resultados con el robot. Hablaremos, en este caso, de planificadores eje a eje, sincronizados... etc.

1.2 Tipos de trayectoria

Trayectorias de primer orden. Responden a la ecuación:

```
q(t) = k1 + k2*t
```

El planificador deberá ser capaz de calcular las constantes para que el movimiento se realice entre dos instantes de tiempo t1 y t2. Se observa, claramente que tendremos una velocidad constante entre ambos puntos articulares.

Trayectorias de segundo orden: En este caso la trayectoria se calcula como:

```
q(t) = k1 + k2*t + k3*t^2
```

Vemos que la velocidad es lineal entre ambos puntos y la aceleración es constante a lo largo del movimiento.

Podemos plantear otros planificadores de mayor orden.

1.3 Tipos de planificadores

Conviene en este momento que analicemos soluciones posibles al problema del path planning. Algunas de estas soluciones corresponden a robots fabricados en el pasado y que podríamos considerar poco avanzados tecnológicamente.

1.3.1 Movimiento eje a eje

Sencillo. Solamente una articulación se mueve en cada instante de tiempo. La trayectoria del extremo del robot no puede seguir una línea recta. El tiempo total que dura la trayectoria corresponde a la suma de los movimientos de todas las articulaciones que se realizan de forma consecutiva.

1.3.2 Movimiento simultáneo de ejes

Sencillo. Varias articulaciones inician su movimiento al mismo tiempo. Todas las articulaciones se mueven a su velocidad máxima. El movimiento de las articulaciones no finaliza en el mismo instante de tiempo.

1.3.3 Movimiento coordinado isócrono de ejes

Las articulaciones se mueven al tiempo. El movimiento de las articulaciones comienza y finaliza en el mismo instante de tiempo. Generalmente, alguna de las articulaciones funcionará en su velocidad máxima, según el número de radianes que deba moverse.

1.3.4 Movimiento en el espacio de la tarea

Plantea un nivel de abstracción mayor a los comentados anteriormente. Primero se define una trayectoria del extremo del robot. Esta puede consistir, por ejemplo, en:

- Una trayectoria recta a velocidad constante y orientación constante.
- Una trayectoria recta con un perfil trapezoidal de velocidades.
- Una reorientación del extremo a posición constante.
- Una trayectoria circular entre 3 puntos.

- Una trayectoria circular con reorientación.

2 Ejercicios

Realice los ejercicios que se plantean en arte/exercises/path_planning/interpolators.m

Ejercicio 1: Programe un planificador de primer orden. Planifique una trayectoria articular desde dos valores con un planificador de primer orden. El movimiento debe realizarse a una velocidad constante de 2 rad/s. Muévase de q(0)=0 a q(T)=1.8 rad. a) Complete la función first_order que devuelva los datos necesarios. b) Dibuje la trayectoria en posición con variaciones de 0.001 segundos. Plotee también la velocidad y la aceleración en cada tiempo. c) Presente sus conclusiones. Desde su punto de vista, ¿qué aspectos deberían mejorarse? Observe que la aceleración toma valores infinitos en el inicio y final del movimiento y que esto podría dar lugar a la aplicación de pares innecesariamente altos en las articulaciones.

El ejercicio 1 se anota aquí comentado su solución.

`**1**` --> dada la velocidad máxima se calcula el vector de tiempo en base a la trayectoria a realizar.

`**2**` --> en este punto, se calculan los coeficientes del de la ecuación de la trayectoria. Tenga Ud. en cuenta, que debe clumplirse.
q(t=0) = k1 = 0 rad
q(t=T2) = k1 + k2*T2 = 1.8 rad
La solución de este problema se obtiene como k = inv(A)*b';

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Exercise 1:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function exercise1()
% Go from q1=0 to q2=1.8 rad
q=[0 1.8];
% At a constant speed.
qd=2; %rad/s
% DEFINE t = [T1 T2]
T1 = 0;
T2 = (q(2)-q(1))/qd;

% (**1**)


t=[T1 T2];
%the minimum difference in seconds between times to compute q(t)
functions
delta_t=0.001;

% FIRST ORDER PLANNER FROM q1 to q2
```

```
figure, xlabel('t (s)'), ylabel('q (rad), q_d (rad/s)'), title('FIRST
ORDER PLANNER. EXERCISE 1'), hold on
[q_t, qd_t, time, k]=first_order([q(1) q(2)], [t(1) t(2)], delta_t);
plot(time, q_t, 'r'), plot(time, qd_t, 'g')

legend('Position (rad)','Speed (rad/s)')
%WRITE EQUATION
disp('The equation computed can be written as:')
fprintf('\n q(t)=k1+k2*t=%.3f+%.3f*t', k(1), k(2))



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% First order interpolator.
%   Given the start and end joint positions and times compute a first %
order polinomial of the form:
%   q(t) = k(1) + k(2)*t
%
%   Inputs:
%       q: a joint vector of two inputs q = [q1 q2]
%       t: a time vector with t = [t1 t2]
%       delta_t: the sample time between computed q(t)'s
%   Returns:
%
%       q: the values of q(t) as a function of the time vector used.
%       qd: the speed qd(t) as a function of the time vector.
%       t: the time vector used.
%       k: the polynomial coefficients k that allow to compute q(t) as:
%               q(t) = k(1) + k(2)*t
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%function [q_t, qd_t, time, k]=first_order(q, t, delta_t)
% define the time matrix

% (**2**)

A = [1 t(1);
     1 t(2)];
b = [q(1) q(2)];
% compute the coefficients of k for a first order planner
k = inv(A)*b';

% define the time vector using delta_time
time=t(1):delta_t:t(2);
% the first order equation
q_t=k(1) + k(2)*time;
% return a constant speed
qd_t=k(2)*ones(1,length(time));
```

Dado que conoce q_t, use la función animate con un robot y observe su movimiento.

Ejercicio 2: Derive un planificador de segundo orden.

Escriba un script de Matlab para el cálculo de una trayectoria de la forma:
q(t) = k1 + k2*t + k3*t^2
Para ello, escriba una función que tenga la forma siguiente:

[q_t, qd_t, qdd_t, t, k]=second_order(q, qd, t, delta_t)

Con entradas:

q: un vector con los valores iniciales y finales de la trayectoria para una articulación. q = [q1 q2]
qd: la velocidad inicial en t1.
t: el vector del tiempo inicial y final con t = [t1 t2]
delta_t: tiempo entre muestras para construir q(t)

Salidas:

q: valores de q(t) como función del vector temporal
qd: velocidad como función del vector de tiempo usado.
t: el vector de tiempo usado.
k: el vector de coeficientes que permite calcular la trayectoria como:
q(t) = k(1) + k(2)*t + k(3)*t^2

Finalmente presente sus conclusiones. ¿Qué aspectos se deberían mejorar en este planificador, desde su punto de vista?

Ejercicio 3: Utilice varios interpoladores lineales en la trayectoria.

Realice las siguientes cuestiones:
a) Use un planificador de primer orden para interpolar una trayectoria en una sola articulación a lo largo del tiempo. Considere las siguientes coordenadas en el espacio de la articulación y el tiempo.
q1 = 0.2 rad. t1 = 0 s
q2 = 0.5 rad. t2 = 6 s
q3 = 0.8 rad. t3 = 8 s
q4 = 1.5 rad. t4 = 10 s
b) Dibuje en Matlab la trayectoria total en velocidad y posición. c) Indique sus conclusiones principales, desde su punto de vista.

Ejercicio 4: Cree un interpolador de orden 5.

Escriba un script de Matlab para el cálculo de una trayectoria de la forma:

$q(t) = k1 + k2*t + k3*t^2 + k4*t^3 + k5*t^4 + k6*t^5$

Para ello, escriba una función que tenga la forma siguiente:

[q_t, qd_t, qdd_t, t, k]=fifth_order(q, qd, qdd, t, delta_t)

Con entradas:

q: un vector con los valores iniciales y finales de la trayectoria para una articulación. q = [q1 q2]
qd: la velocidades iniciales en t1 y t2.
qdd: las aceleraciones iniciales en t1 y t2.
t: el vector del tiempo inicial y final con t = [t1 t2]. Puede considerarse t = [0 1] por ejemplo.
delta_t: tiempo entre muestras para construir q(t)

Salidas:
q: valores de q(t) como función del vector temporal
qd: velocidad como función del vector de tiempo usado.
Qdd: la aceleración como función del tiempo.
t: el vector de tiempo usado.
k: el vector de coeficientes que permite calcular la trayectoria como:
$q(t) = k(1) + k(2)*t + k(3)*t^2 + k(4)*t^3 + k(5)* t^4 + k(6)*t^5$

Finalmente presente un gráfico con la posición q(t), velocidad qd(t) y aceleración qdd(t).

2 Movimiento isócrono

Una planificación en el espacio de las articulaciones considera, generalmente que todas las articulaciones se mueven de la siguiente manera:

- El movimiento de todas las articulaciones se inicia y se finaliza al mismo tiempo.
- Una de las articulaciones se moverá a su velocidad máxima.
- El resto de articulaciones se moverá dependiendo de la cantidad de grados/radianes que deba recorrer.

Hay diferentes formas de resolver el anterior problema. Se expone, a continuación, una posible solución:

1) Asumir un tipo de interpolación para el movimiento: primer orden, segundo orden, … etc.

2) Hallar el tiempo t_i de cada articulación en realizar el movimiento si se supone que cada articulación se mueve a su velocidad y aceleraciones máximas.

3) Hallar la articulación "más lenta", es decir, aquella que necesita un tiempo T mayor para realizar el movimiento.

4) Calcular la velocidad del resto de articulaciones para que se muevan en un tiempo T.

Ejercicio 5: Planificación isócrona de articulaciones

Ahora considere que desea mover un robot de 6 GDL desde una posición

inicial q1 = [pi/8 -pi/8 pi/2 pi/3 pi/4 pi/2] hasta una posición final q2=[-pi/8 pi/8 -pi/2 -pi/3 -pi/4 -pi/2]. Puede utilizar cualquier robot de la librería.
La planificación de trayectorias debe ser de primer orden para cada articulación. Visualice la planificación de trayectorias y utilice la función animate para ver al robot moverse.

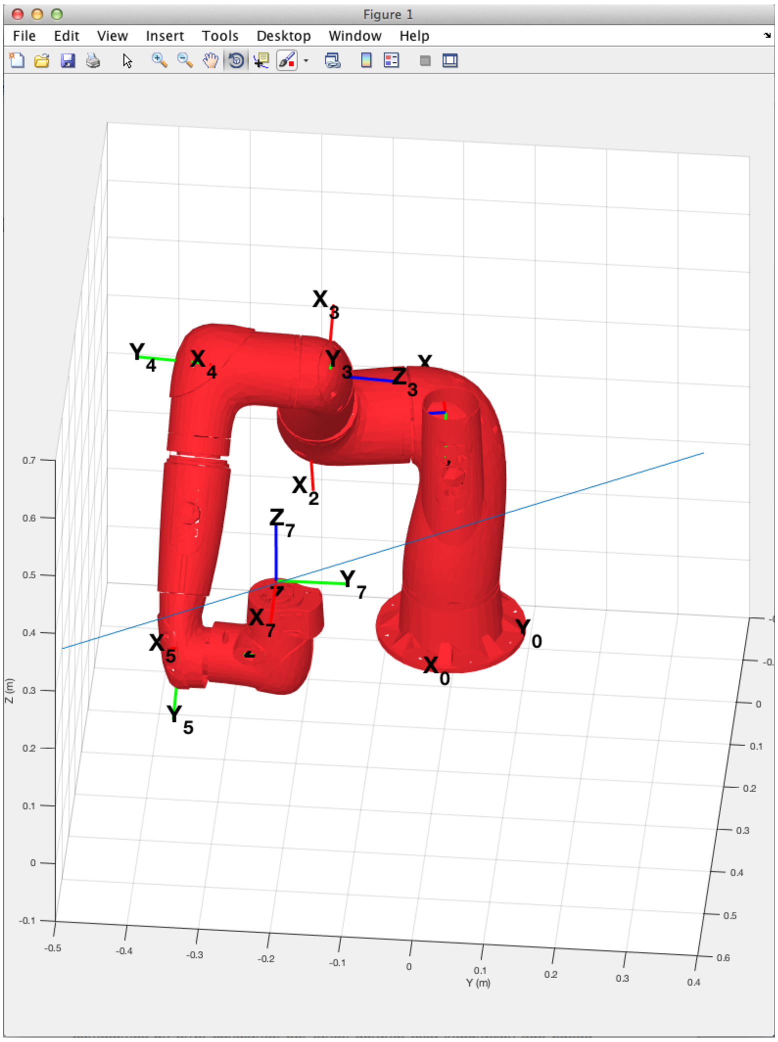
3 Seguimiento de trayectorias rectas en el espacio de trabajo

Seguir una trayectoria recta en el espacio es una tarea común y frecuente de un robot industrial en cualquier aplicación. Un operador humano encuentra, generalmente, fácil, la realización de un movimiento por parte de un robot que siga una trayectoria recta entre dos puntos en un espacio cartesiano que conecte dos posiciones/orientaciones del robot. En general, seguir una trayectoria recta en el espacio de la tarea no será óptimo en términos de velocidad o eficiencia pero, aún así, es posible encontrar una gran cantidad de aplicaciones en las que se precisa este tipo de trayectorias, por ejemplo:

a) Soldadura TIG de dos planchas de metal a lo largo de una línea.

b) Soldadura por puntos a lo largo de una línea. Point welding es típico en la industria de fabricación de vehículos automóviles.

Por tanto, se propone al estudiante seguir una trayectoria recta en el espacio de trabajo de la tarea. Complete el código en `arte/exercises/path_planning/path_planning_line_simple.m`



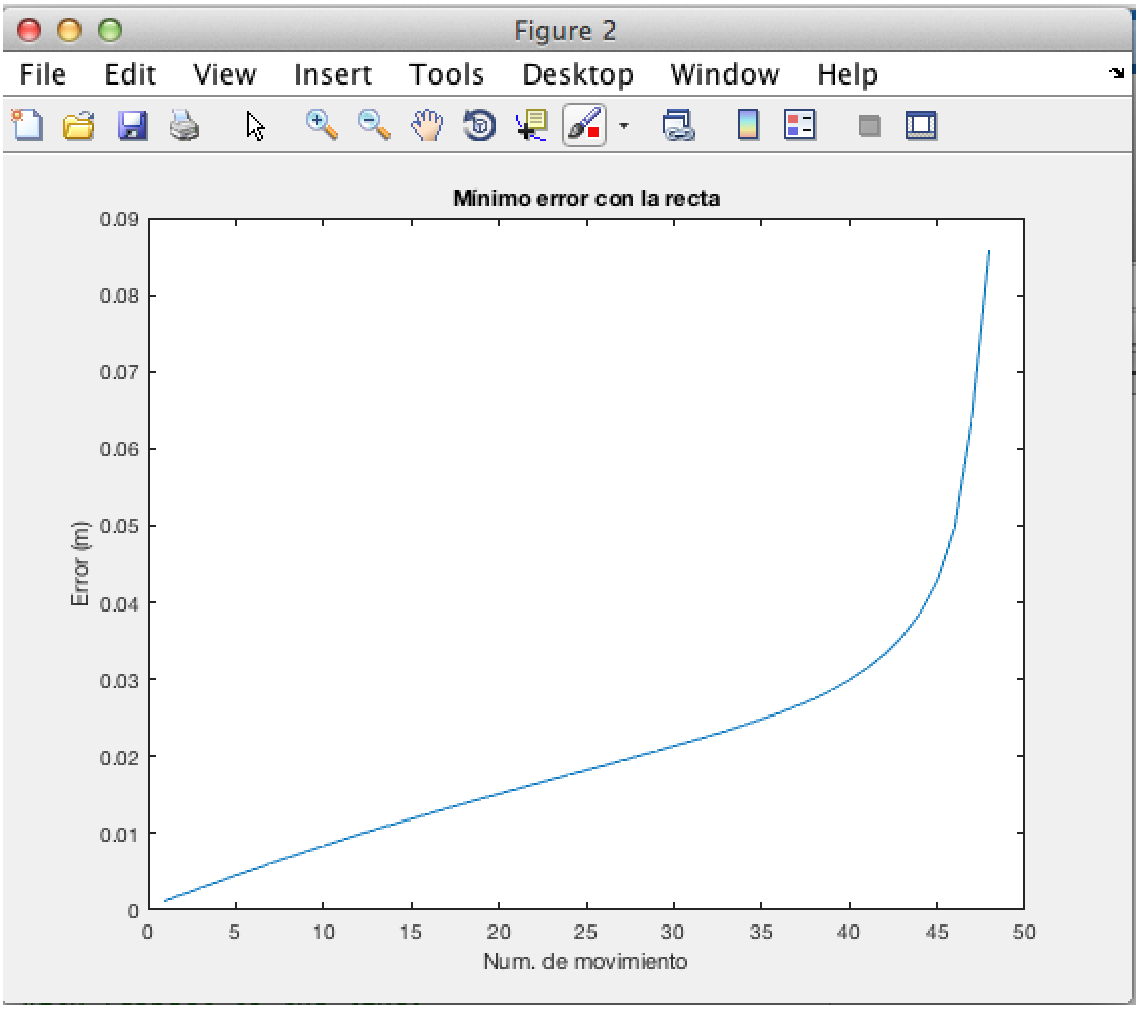
Cualquiera de las acciones de control planteadas hasta ahora sufre de un pequeño problema, por ejemplo:

$$\dot{q} = J^{\dagger} \cdot \hat{V}$$

La velocidad articular así calculada es una aproximación lineal (pues se calcula usando la Jacobiana) válida únicamente en el instante t. Al integrar y hacer

$$q = q + \Delta t \cdot \dot{q}$$

cometemos un error constante, por tanto, durante esta integración que puede llegar a ser problemático en algún caso. Se presenta a continuación el error con respecto a la recta a seguir.



Proponga una solución a este problema. En el script de seguimiento de línea cuenta ya con la función:

```
[delta_end, error_line, error_line_vector] = find_errors(start_point,
end_point, p);
```

Dicha función proporciona un vector unitario en dirección con el mínimo error del extremo y la recta. Plantee, también, formas de corregir el error de orientación.

Se propone al estudiante que complete el código en `arte/exercises/path_planning/path_planning_line_simple.m` siguiendo los comentarios que hay en él y lo haga funcionar para completar la aplicación deseada.