Actividad 1. Cinemática: caso de aplicación de generación de trayectorias

Resultados de aprendizaje

1. Identifica, formula y resuelve problemas complejos de ingeniería con el uso de principios de ingeniería, ciencias y matemáticas.

2. Aplica el diseño de ingeniería para producir soluciones que satisfagan necesidades específicas teniendo en cuenta la salud pública, la seguridad y el bienestar, así como factores globales, culturales, sociales, ambientales y económicos.

Objetivo de la actividad

En esta actividad, relacionada con cinemática de la partícula, se requiere crear un programa con el que se pueda realizar, de forma sistemática, la planeación de trayectorias (posición, velocidad y aceleración) para el movimiento de un punto a otro. A partir de los resultados obtenidos con el programa se deben comparar dos alternativas de trayectoria y decidir cuál es la más adecuada para la aplicación.

Contexto del problema

El uso de sistemas aéreos no tripulados (UAS), sean autónomos o de operación remota, ha aumentado significativamente en los últimos años. Las aplicaciones son variadas, e incluyen, entre otras, agricultura de precisión [1], mapeo de ecosistemas [2], planeación urbana [3], mantenimiento de infraestructura [4] y espectáculos visuales [5]. Dentro de todas estas aplicaciones se requiere el uso de algoritmos y sistemas computacionales que permitan generar las trayectorias (historias de tiempo) de forma adecuada [6, 7] para las variables cinemáticas que debe tener el vehículo (especificaciones de movimiento deseado para el sistema de navegación, guía y control).

Requerimientos de la actividad

Se requiere diseñar dos posibles trayectorias para un UAS usado en fotogrametría, compararlas y decidir cuál es la más adecuada, de la siguiente manera:

a) Desarrolle una función computacional en la cual se implementen las ecuaciones para una trayectoria cúbica, las cuales están dadas por:

- Posición: $q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$
- Velocidad: $\dot{q}(t) = a_1 + 2a_2 t + 3a_3 t^2$
- Aceleración: $\ddot{q}(t) = 2a_2 + 6a_3 t$

Los coeficientes de los polinomios están dados por la solución del siguiente sistema lineal de ecuaciones que depende de las restricciones dadas para el movimiento: posición inicial q_0, posición final q_f, velocidad inicial $\dot{q}_0$, velocidad final $\dot{q}_f$, tiempo inicial t_0 y tiempo final t_f.

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_0 & 3t_0^2 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} q_0 \\ q_f \\ \dot{q}_0 \\ \dot{q}_f \end{bmatrix}$$

El ejemplo de la sintaxis para Matlab y GNU Octave es:

```
[q,qdot,qdotdot,t]=fn_trayectoria_cubica(q0,qf,qdot0,qdotf,t0,tf,tspace)
```

El ejemplo de la sintaxis para Python es:

```
def fn_trayectoria_cubica(q0, qf, qdot0, qdotf, t0, tf, tspace):
⋮
return q, qdot, qdotdot, t
```

La salida de la función está compuesta por los cuatro vectores que representan la posición `q`, la velocidad `qdot`, la aceleración `qdotdot` y el tiempo `t`. Este vector de tiempo se construye en Matlab y Octave con `t=t0:tspace:tf`, y en Python `t = np.arange(t0, tf, tspace)`. El valor de `tspace` corresponde a la mínima división del vector de tiempo, y puede asignarse para este caso `tspace=0.001` s.

b) Desarrolle una función computacional en la cual se implementen las ecuaciones de un generador de trayectoria que usa un tramo recto mezclado con funciones cuadráticas, las cuales están dadas por:

- Posición:

$$q(t) = \begin{cases} q_0 + \frac{1}{2}\ddot{q}_d t^2 & \text{for } 0 \le t \le t_b \\ \frac{q_h - q_b}{t_h - t_b}(t - t_b) + q_b & \text{for } t_b \le t \le t_f - t_b \\ q_f - \frac{1}{2}\ddot{q}_d (t_f - t)^2 & \text{for } t_f - t_b \le t \le t_f \end{cases}$$

- Velocidad:

$$\dot{q}(t) = \begin{cases} \ddot{q}_d t & \text{for } 0 \le t \le t_b \\ \frac{q_h - q_b}{t_h - t_b} & \text{for } t_b \le t \le t_f - t_b \\ \ddot{q}_d (t_f - t) & \text{for } t_f - t_b \le t \le t_f \end{cases}$$

- Aceleración:

$$\ddot{q}(t) = \begin{cases} \ddot{q}_d & \text{for } 0 \le t \le t_b \\ 0 & \text{for } t_b \le t \le t_f - t_b \\ -\ddot{q}_d & \text{for } t_f - t_b \le t \le t_f \end{cases}$$

Los coeficientes de las funciones por tramos se calculan con las restricciones dadas para el movimiento: posición inicial q_0, posición final q_f, tiempo final t_f y aceleración deseada para los tramos cuadráticos $\ddot{q}_d$. Las ecuaciones dadas son válidas para movimientos que inician y terminan con velocidad cero. Para estas ecuaciones se tiene que:

$$q_h = \frac{q_f - q_0}{2} + q_0, \quad t_h = \frac{t_f}{2},$$

$$t_b = \frac{tf}{2} - \frac{\sqrt{\ddot{q}_d^2 t_f^2 - 4\ddot{q}_d (q_f - q_0)}}{2\ddot{q}_d} \quad \text{con} \quad \ddot{q}_d \ge \frac{4(q_f - q_0)}{t_f^2},$$

$$q_b = q_0 + \frac{1}{2}\ddot{q}_d t_b^2;$$

El ejemplo de la sintaxis para Matlab y GNU Octave es:

`[q,qdot,qdotdot,t]=fn_trayectoria_recta_cuadratica(q0,qf,qdotdot_d,tf,tspace)`

El ejemplo de la sintaxis para Python es:

```
def fn_trayectoria_recta_cuadratica(q0, qf, qdotdot_d, tf, tspace):
⋮
```

`return q, qdot, qdotdot, t`

La salida de la función está compuesta por los cuatro vectores que representan la posición `q`, la velocidad `qdot`, la aceleración `qdotdot` y el tiempo `t`. Este vector de tiempo se construye en Matlab y Octave con `t=t0:tspace:tf`, y en Python `t = np.arange(t0, tf, tspace)`. El valor de `tspace` corresponde a la mínima división del vector de tiempo, y puede asignarse para este caso `tspace=0.001` s.

Para lo anterior, desarrolle un programa principal (`main_generacion_trayectoria_dron`) que utilice y sirva para comparar el resultado obtenido con las dos funciones implementadas. La comparación se debe realizar para las siguientes condiciones de movimiento: $q_0 = 0$, $\dot{q}_0 = 0$ y $\dot{q}_f = 0$. Los otros valores deben ser seleccionados por el equipo de trabajo y ajustados, según el análisis que se quiera realizar. Por ejemplo, para un trayecto de toma de imágenes dentro de una región cuadrada de una hectárea, $q_f \approx$ 100 m y $t_f \approx \frac{q_f}{[1.0-2.0]}$ s (usando una velocidad usual para conservar la calidad de las imágenes). El valor de $\ddot{q}$ se debe seleccionar por usted como diseñador de la trayectoria para esta aplicación. La Figura 1 muestra un ejemplo de cómo se debe presentar la comparación.

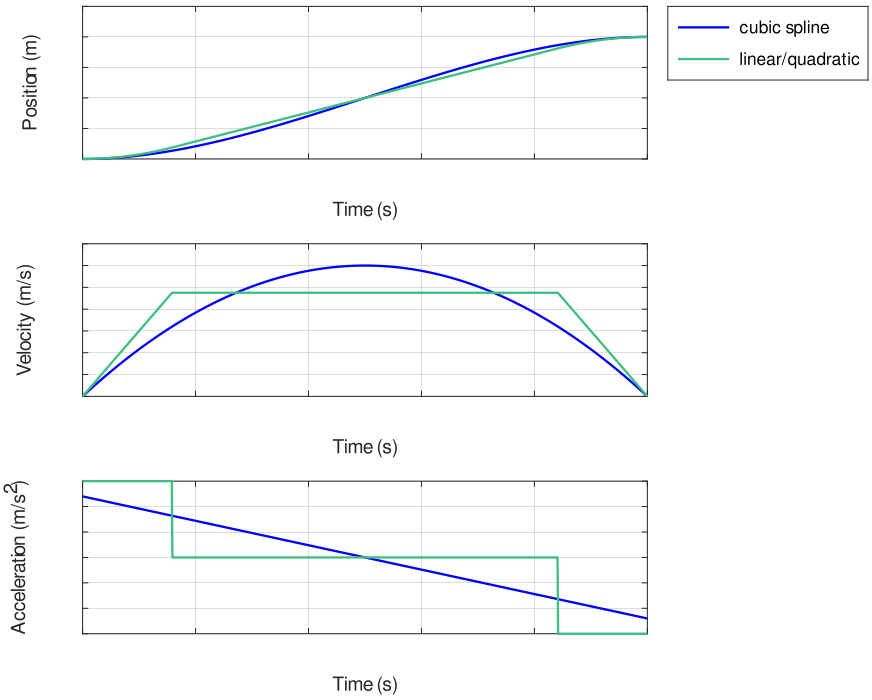

Figura 1: Comparación de trayectorias. Posición, velocidad y aceleración.

Entrega

Para dar cumplimiento a la actividad debe entregar un solo archivo con extensión `.zip` que contenga:

- Un archivo con extensión `.pdf` de máximo dos páginas que contenga los nombres de los autores, enunciar el objetivo del cálculo, plantear las ecuaciones cinemáticas aplicadas en el problema, enumerar los datos conocidos y supuestos, desarrollar las ecuaciones, mostrar la solución del problema y la figura de comparación para los dos generadores de trayectoria y un análisis de lo presentado en esta, con la decisión de trayectoria de acuerdo con el análisis.
- Para Matlab/Octave: el programa principal `main_generacion_trayectoria_dron`. Este programa debe llamar las dos funciones donde se genera la trayectoria, `fn_trayectoria_recta_cuadratica` y la función `fn_trayectoria_cubica` por separado.

- Para Python: el programa principal `main_generacion_trayectoria_dron`. Este programa debe llamar las dos funciones donde se genera la trayectoria. Las funciones `fn_trayectoria_recta_cuadratica` y `fn_trayectoria_cubica` deben estar definidas dentro del programa.

Referencias

[1] S. S. Chouhan, R. K. Patel, U. P. Singh, and G. G. Tejani, "Integrating drone in agriculture: Addressing technology, challenges, solutions, and applications to drive economic growth," *Remote Sensing Applications: Society and Environment*, vol. 38, p. 101576, Apr. 2025, 10.1016/j.rsase.2025.101576.

[2] P. A. Zapata-Ramírez, H. Hernández-Hamón, C. Fitzsimmons, M. Cano, J. García, C. A. Zuluaga, and R. E. Vásquez, "Development of a google earth engine-based application for the management of shallow coral reefs using drone imagery," *Remote Sensing*, vol. 15, no. 14, p. 3504, Jul. 2023, 10.3390/rs15143504.

[3] D. Gu, N. Zhang, Q. Shuai, Z. Xu, and Y. Xu, "Drone photogrammetry-based wind field simulation for climate adaptation in urban environments," *Sustainable Cities and Society*, vol. 117, p. 105989, Dec. 2024, 10.1016/j.scs.2024.105989.

[4] P. Kim and J. Youn, "Drone path planning for bridge substructure inspection considering gnss signal shadowing," *Drones*, vol. 9, no. 2, p. 124, Feb. 2025, 10.3390/drones9020124.

[5] K.-C. Weng, S.-T. Lin, C.-C. Hu, R.-T. Soong, and M.-T. Chi, "Multi-view approach for drone light show," *The Visual Computer*, vol. 39, no. 11, pp. 5797–5808, Nov. 2022, 10.1007/s00371-022-02696-8.

[6] G. Gugan and A. Haque, "Path planning for autonomous drones: Challenges and future directions," *Drones*, vol. 7, no. 3, p. 169, Feb. 2023, 10.3390/drones7030169.

[7] M. Sushma, B. Mashhoodi, W. Tan, K. Liujiang, and Q. Xu, "Spatial drone path planning: A systematic review of parameters and algorithms," *Journal of Transport Geography*, vol. 125, p. 104209, May 2025, 10.1016/j.jtrangeo.2025.104209.