IMPLEMENTACIÓN CON LABVIEW Y ARDUINO DEL ALGORITMO DE CAMINATA ALEATORIA PARA LA OBTENCIÓN DE LA CINEMÁTICA INVERSA DE UN ROBOT PLANAR DE TRES GRADOS DE LIBERTAD

M. en C. Raúl López Muñoz
rlopezm1209@alumno.ipn.mx

Es importante emplear los métodos determinísticos para no dejar dudas sobre la respuesta que se obtiene al resolver un problema, pero en la práctica es necesario dar resultados y a veces no se dispone del tiempo para abordar todos los problemas con esos métodos, por lo que es necesario recurrir a otras técnicas. Un ejemplo de ello surge en técnicas heurísticas cuya premisa parte de ideas lógicas o con algún sentido que después se matematizan. En este trabajo se emplea una de esas técnicas para implementar la obtención de la cinemática inversa de un robot de tres grados de libertad.

Introducción

En el presente trabajo se expone la implementación del algoritmo de caminata aleatoria en el entorno de LabVIEW con comunicación a Arduino para el control de tres servomotores los cuales representarían las articulaciones del robot.

El problema

En distintas áreas como la robótica es necesario llevar la parte final de una cadena cinemática (efector final) a una posición deseada, esto se puede lograr mediante la manipulación directa de los valores asociados a las posiciones de los motores que influyen en el movimiento del robot, pero suele ser deseable disponer de algún medio para enviar las coordenadas deseadas dentro de su espacio de trabajo y que el software controlador determine el movimiento de los motores para lograr esta tarea. Está documentado que existen diversas formas para poder lograr este objetivo y algunas de ellas emplean métodos no determinísticos debido a la complejidad de obtener modelos matemáticos para dar solución a esta tarea (Parent, 2012).

Algoritmo de caminata aleatoria

Como se mencionó previamente, una forma de posicionar el efector final del robot sería incrementar los valores asociados a los efectores del robot, una posibilidad sería incrementar poco a poco los valores y de acuerdo con el criterio de un usuario considerar que la solución mejora o no, si mejora, continúa variando los valores en busca de una configuración aún mejor, de lo contrario regresa al valor anterior y prueba otra variación.

Esto representa una descripción informal del algoritmo de caminata aleatoria (Norving y Rusell, 1995) y el cual solo requiere sustituir al usuario por un indicador comprensible en términos computacionales, por lo que en lugar de dejarlo al criterio de una persona que podría indicar a partir de una inspección visual si el robot se acerca o no al objetivo, se debe implementar una función que cuantifique que tan cerca o lejos se encuentra el efector del objetivo.

Esta función será entonces y para efectos del trabajo la distancia entre la posición deseada y la del efector final que se obtiene mediante la cinemática del robot que se muestra en la Figura 1

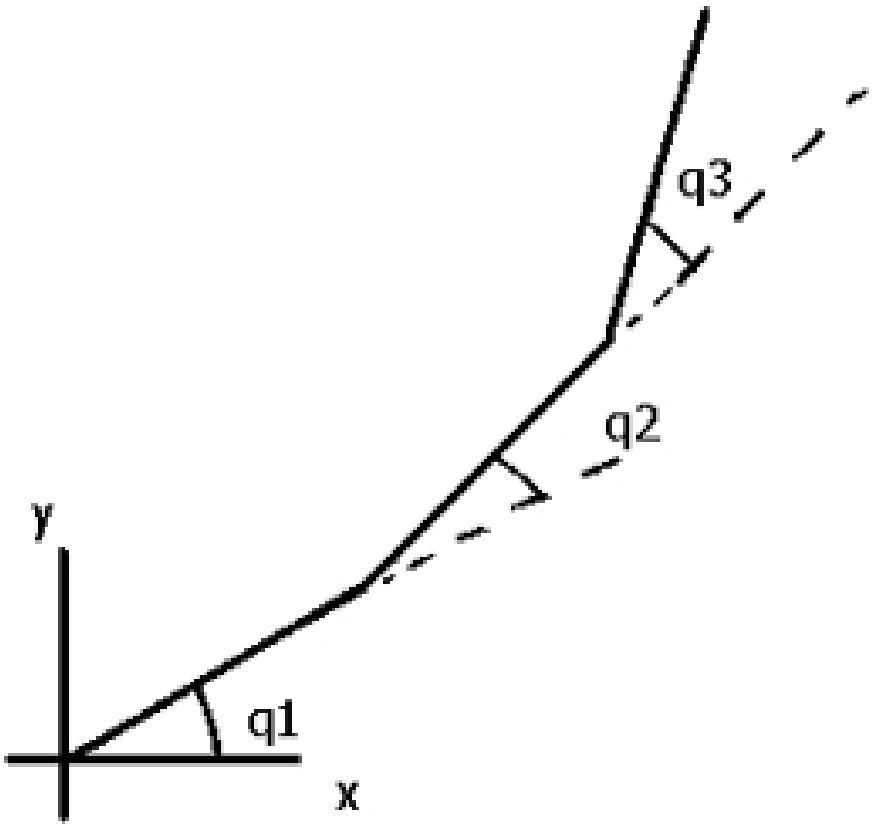

Figura 1. Diagrama de referencia del robot.

La posición de su efector final se describe mediante:

$$f_y(q) = L(\sin(q_1) + \sin(q_1 + q_2) + \sin(q_1 + q_2 + q_3))$$
$$f_x(q) = L(\cos(q_1) + \cos(q_1 + q_2) + \cos(q_1 + q_2 + q_3))$$

Donde *L*, es la longitud de los eslabones que es la misma para todos y q_i son los valores angulares de las articulaciones.

Por lo que la función del algoritmo sería

$$d(q,p) = |f_y(q) - p_y| + |f_x(q) - p_x|$$

La cual, es la distancia Manhattan (Krause, 1987) entre el punto deseado *p* y el mapeo de *q* al espacio de coordenadas de *p*.

Por lo que el algoritmo quedaría descrito como:

1. Evaluar *d(q,p)*
2. Aplicar un incremento aleatorio a los valores de $q, q_{prueba} = q + k(rand(-1,1))$
   Donde $rand(-1,1)$es una función que genera un vector de tres números aleatorios en el rango de -1 a 1 y *k* es un factor de escalado.
3. Comparar $-1 \text{ a } 1$
4. Si $\text{Comparar } d(q,p) \text{ con } d(q_{prueba},p)$

Implementación en LabVIEW con interfaz a Arduino

Se desarrollo un proyecto en LabVIEW el cual cuenta con el instrumento virtual principal que consiste en una interfaz simple con controles para asignar el valor de p, la posición inicial de los servos, el puerto en el cual se encuentra conectada la tarjeta de Arduino, los canales dónde se enviará la señal de los servomotores e indicadores con el valor actual tanto en grados como el pulso en microsegundos de las señales de cada servomotor.

En la Figura 2, se muestra el diagrama a bloques del instrumento, en el cual se destaca que cuenta con otros instrumentos virtuales, de los cuales los referentes a la conexión con Arduino forman parte de la paleta de MakerHub, que es un toolkit que ofrece National Instruments para comunicarse con Arduino, mientras que BusquedaAleatoria.vi es el algoritmo de búsqueda.

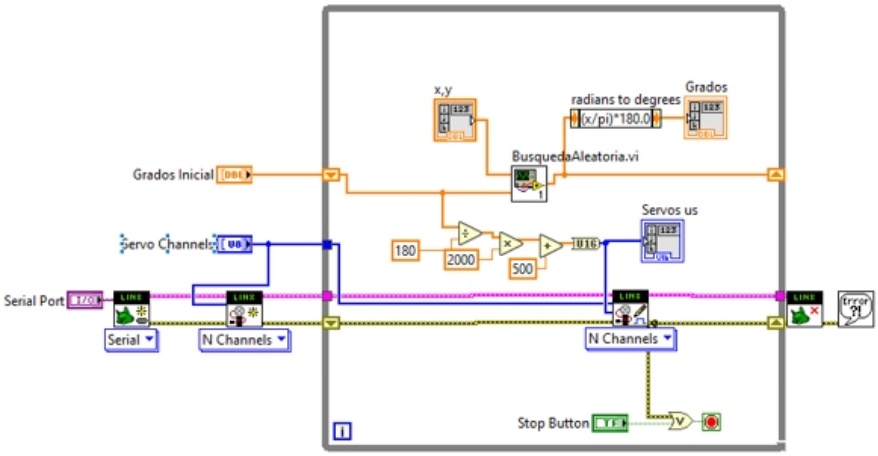

Figura 2. Diagrama a bloques del instrumento virtual para el control de servomotores.

Es importante mencionar que el algoritmo busca posiciones angulares pero los servomotores se mueven con una señal que modula un ancho de pulso para determinar la posición del servomotor, este pulso se encuentra en un rango típico de 500 a 2500 microsegundos por lo que es necesario realizar una conversión del rango de búsqueda que es de 0 a 180. Esta conversión se expresa mediante la siguiente función:

$$f(q_i) = 2000\frac{q_i}{180} + 500$$

Donde q_i es el valor angular del motor *i* .

Una última aclaración sobre el algoritmo de búsqueda aleatoria es que es necesario implementar una condición de paro y se deben realizar ajustes para que no explore soluciones no alcanzables. En la Figura 3 se observa que después de la adición del vector aleatorio se aplicó la función de absoluto esto con la finalidad de evitar soluciones con valores negativos, por otra parte, la búsqueda finaliza cuando se evalúan 2000 posibilidades o bien la función tiene un error menor a 0.1. El número de evaluaciones se seleccionó a partir de las pruebas realizadas, en las que, se observó que en promedio requería 1200 evaluaciones para encontrar una solución cuyo error fuera menor al propuesto.

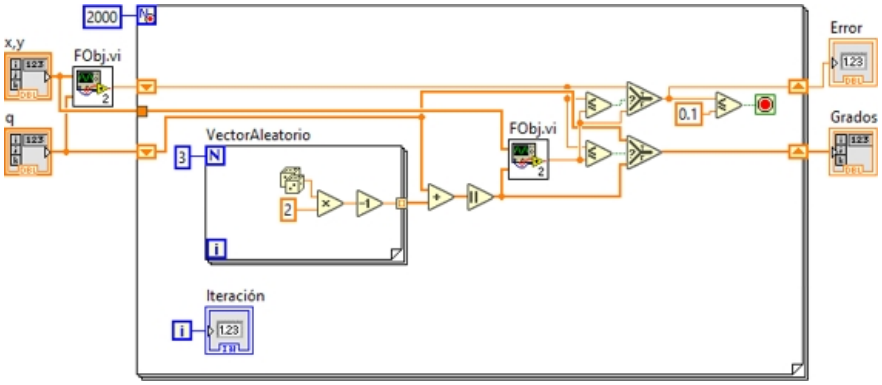

Figura 3. Diagrama a bloques del algoritmo de búsqueda aleatoria.

Finalmente, en cuanto a la implementación de la función de error, hay que considerar que el nodo de fórmula de LabVIEW trabaja con radianes por lo que para evaluar correctamente las expresiones es necesario transformar los grados a radianes, para el cual LabVIEW cuenta con un nodo de expresión mismo que se empleó como se muestra en la Figura 4.

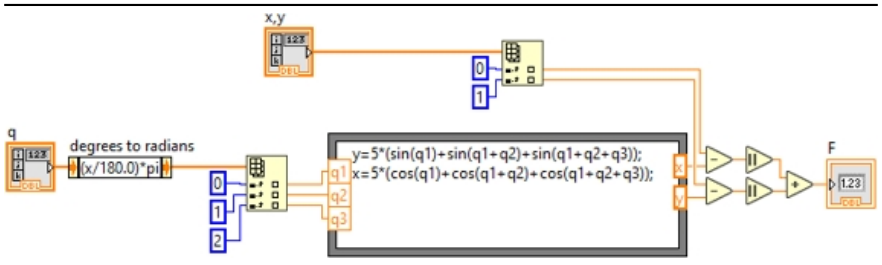

Figura 4. Diagrama a bloques de la función de error (FObj.vi).

Aplicaciones y comentarios finales

Se resalta el hecho de que este algoritmo es fácil de implementar para obtener la cinemática inversa de robots en tanto se disponga de su modelo de cinemática directa, en el caso de estudio, al ser un robot con pocos grados de libertad el modelado es sencillo y las evaluaciones que emplea el algoritmo ocupan tiempos menores a 0.5 segundos. Esto en una computadora con sistema operativo Windows 10, procesador Intel i7 y 16 GB de RAM.

Por otra parte, y de manera general es importante mencionar que el algoritmo se puede emplear de manera general para resolver problemas de minimización, pero no garantiza el mínimo global de la función. Además de que es susceptible a quedar atrapado en mínimos locales.

En el caso de la aplicación abordada en este trabajo y como consecuencia de la presencia de variables aleatorias en el algoritmo es posible encontrar distintas soluciones para un mismo punto deseado y las trayectorias pueden diferir considerablemente unas de otras, una sugerencia para minimizar este efecto es que, si se desea llegar a un punto en específico en línea recta, por ejemplo, entonces se debe describir el trayecto como una sucesión de puntos.

El algoritmo completo se puede integrar en Arduino y sustituir algunos valores de la interfaz por lecturas de sensores como la posición inicial del robot, pero la ventaja del vínculo con LabVIEW es que permite implementar algoritmos más robustos los cuales podrían exceder la memoria del microcontrolador de Arduino.

Referencias

1. Krause, E. (**1987**). *Taxicab Geometry*Dover.

2. Norving, P., y Rusell, S. (**1995**). *). Artificial Intelligence – A Modern Approach*Prentice Hall.

3. Parent, R. (**2012**). *Computer animation: algorithms and techniques*Newnes.