CONTROL DE UN BRAZO ROBÓTICO DE BAJO COSTE MEDIANTE DIFFERENTIAL EVOLUTION

Adrián Prados, Alicia Mora , Ramón Barber y Santiago Garrido
RoboticsLab, Universidad Carlos III de Madrid, Leganés, España
aprados@pa.uc3m.es, almorav@pa.uc3m.es, rbarber@ing.uc3m.es, sgarrido@ing.uc3m.es

Resumen

Los manipuladores robóticos son sistemas altamente no lineales, y es difícil obtener un modelo matemático preciso con técnicas convencionales. Aplicando técnicas de control clásico se puede resolver este tipo de problemas, pero tiene el inconveniente de que se tarda una gran cantidad de tiempo en obtener un resultado satisfactorio para manipuladores donde se tenga una gran cantidad de grados de libertad (GDL). Es por esto, por lo que se requiere una técnica eficiente para tratar con este tipo de sistemas complejos y dinámicos. El algoritmo de Differential Evolution (DE) es una técnica de optimización global muy potente la cual en la actualidad se ha hecho popular debido a su posible aplicación en una gran cantidad de campos dentro de la robótica. En el presente trabajo se muestra un aplicación directa de este método de optimización en un modelo simulado de un brazo de 6 GDL y una posterior aplicación sobre un modelo de bajo coste.

Palabras clave: Differential Evolution, Manipulación, Algoritmo Evolutivo

1 INTRODUCCIÓN

El método de Differential Evolution (DE) [1][2] es una técnica de búsqueda directa basado en principios estocásticos. La efectividad de este método ha sido probada en diferentes ámbitos dentro de la robótica [3]

La idea original del DE es la de minimización de funciones no lineales y que a la vez no sean diferenciables en el espacio continuo de configuraciones [4]. Dicho método ha sido comprobado que, frente a otros métodos conocidos [5] y los cuales eran muy utilizados hasta la fecha se obtenía un resultado mucho más óptimo y de una manera mucho más rápida.

En la actualidad, el DE sigue siendo uno de los métodos utilizados para la investigación dentro del ámbito de la robótica debido a sus grandes ventajas donde se pueden destacar fundamentalmente la mínima cantidad de parámetros los cuales han de ser ajustados para su correcto funcionamiento y su robustez, lo que le ha valido para ser altamente usado en diferentes aspectos de la ingeniería y que frente a la mayoría de los algoritmos evolutivos, el DE es el que mejores resultados ofrece en cuanto a términos de optimización. A continuación se muestran algunos de los ejemplos del método en diferentes ámbitos de la robótica

Algunos de las aplicaciones del DE dadas en los ultimos años se centran en el uso de dicho algoritmo para la resolución de problemas típicos de la robótica móvil como puede ser la localización y mapeado simultaneo de un entorno (SLAM)[6], el cual se basa en la búsqueda estocástica de soluciones haciendo uso del DE como elemento principal para la localización global.

Otras aplicaciones centradas en el ámbito de la planificación de trayectorias para brazos robóticos se pueden observar en [7], donde se propone el uso de DE como optimizador de energía en la generación de trayectorias de brazos robóticos dentro de entornos dinámicos como pueden ser en entornos industriales donde los brazos robóticos trabajen de manera colaborativa con los empleados de la planta. Algunos otros ejemplos se pueden ver en [8], donde se aplica el algoritmo de DE para el control de un robot con dos brazos el cual trabaja en ámbitos cooperativos con humanos, o [9] donde se centra en el uso del DE para, dentro del ámbito industrial, realizar la planificación de las trayectorias de los brazos de un robot con limitaciones de cargas de útiles.

Además de centrarse en la generación de trayectorias, hay trabajos que presentan ideas basadas en el DE para la manipulación y la generación del agarre, como se puede apreciar en [10], donde se centra el uso del algoritmo DE en la generación en simulación de los puntos de agarre para diferentes tipos de objetos y disposición de los mismos con la mano *Gifu III*

En este trabajo se propone una aplicación directa del método de Differential Evolution para el control de un brazo tanto en un entorno simulado como en un entorno real. Para ello, se ha decidido realizar un control a través de un modelo

 https://doi.org/10.17979/spudc.9788497498418.0812

PID típico, donde los parámetros que componen los controladores de cada uno de los PID aplicados a cada motor son estimados a través del uso de DE. Para la parte de simulación se ha modelado un brazo de 6 GDL haciendo uso de la herramienta de SimScape que proporciona MATLAB. En cuanto el modelo del brazo real ha sido desarrollado por investigadores de la Universidad Carlos III, dentro del laboratorio de Robótica Móvil como plataforma de enseñanza para la aplicación de algoritmos y controladores centrados en la manipulación así como para la creación de un manipulador móvil propio mediante la combinación de dicho brazo con una base móvil. Tanto el modelo del brazo real como el simulado se pueden apreciar en Fig. 1

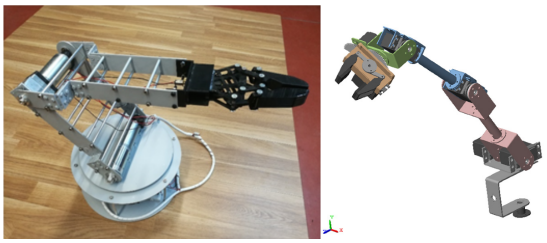

Figura 1: (Izquierda) Modelo del robot real, conformado por 5 GDL, (Derecha) Modelo del brazo en simulación conformado por 6 GDL.

El trabajo queda organizado de la siguiente manera. En la Sección 2 se explica brevemente el desarrollo del método de Differential Evolution, mostrando como funciona el algoritmo así como cuales son los factores relevantes que se modifican y como se aplica dicho método en la búsqeda de los factores de los controladores PID para cada uno de los motores que conforman el brazo. En la Sección 3 se muestra como se ha desarrollado tanto el modelo simulado como el modelo real, explicando los elementos de hardware que lo componen en el caso del brazo real y el proceso llevado a cabo para la obtención de los valores para los controladores en simulación. En la Sección 4 se mostrarán los resultados obtenidos tanto en simulación como llevados a un entorno real. Finalmente, en la Sección 5 se especificarán las conclusiones obtenidas derivadas de las pruebas realizadas así como posibles trabajos futuros.

2 DIFFERENTIAL EVOLUTION: DESCRIPCIÓN Y APLICACIÓN

2.1 DESCRIPCIÓN DEL ALGORITMO

De igual manera que otro tipo de algoritmos genéticos, el algoritmo de Differential Evolution [11][12] se basa en mantener una población de candidatos que puedan ser la solución al problema de optimización los cuales sufrirán procesos de cruces y mutaciones que irán afectando a dichos candidatos buscando siempre mejorar los resultados obtenidos. Para la elección de dichos resultados es necesario hacer uso de una función de salud, la cual permite al método poder estimar si los valores estimados cumplen los requisitos para ser tenidos en cuenta como posibles soluciones. Lo que caracteriza a dicho método frente a otros algoritmos genéticos es el uso de N vectores de prueba de parámetros de dimensión D

$$x_{i,G}, i = 1, 2, ..., N \tag{1}$$

los cuales compiten con los individuos pertenecientes a la población actual para cada una de las iteraciones G. El valor inicial del vector de las poblaciones es escogido de manera aleatoria y tendrá que cubrir el espacio de parámetros de manera completa. Por lo tanto, dicho algoritmo estima que todas las variables a optimizar son un vector de números reales. El Differential Evolution se basa en 4 pasos básicos que a través de su combinación permite obtener los resultados del problema: **Inicialización, Mutación, Cruce y Selección**.

2.1.1 Inicialización

El primer paso del algoritmo es la generacion de N individuos de manera aleatoria, los cuales conformarán a la población inicial Po:

$$P_o = \{1, 2, ..., N\} \tag{2}$$

Normalmente, los individuos generados en las poblaciones están uniformemente distribuidos en el espacio de búsqueda del problema, dentro de los límites previamente definidos por los valores máximos y mínimos, como se puede apreciar en la ecuación 3.

$$x_{ij} = x_j^{min} + U(0,1)(x_j^{max} - x_j^{min}) \quad \forall i \in \{1, ..., N\}, \forall j \in \{1, ..., N\} \tag{3}$$

donde i y j representan cada uno el índice que tiene el individuo dentro de una población (i,j = 1,...,N), U(0,1) representa una variable aleatoria la cual se encuentra uniformemente distribuida dentro de un rango de [0,1] y x_j^{max}, x_j^{min} representan los límites tanto superior como inferior, respectivamente, con

los cuales el algoritmo restringe el dominio de las variables.

Tras haber realizado el proceso de inicialización, la población se somete a proceso de manera iterativa de mutación, cruce y selección, las cuales se irán repitiendo hasta que se alcance el criterio de parada (las cuales pueden depender de diversos factores como número de iteraciones, disminución del error menor a un cierto porcentaje...)

2.1.2 Mutación

Tras haber realizado la inicialización, el DE realiza un proceso de mutación y recombinación de la población para obtener a raíz de esta una población de N vectores de prueba N. Para esto,el algoritmo genera una mutación diferencial haciendo uso del *vector de diferencia* (el cual se ha obtenido de manera aleatoria) que, realizando un escalado, se añade a un tercer vector para crear el vector mutado $v_{j,g}$:

$$v_{j,g} = x_{r_0,g} + F(x_{r_1,g} - x_{r_2,g}) \tag{4}$$

El factor de escala, $F \in (0, 1+)$, es un número real y positivo que permite al algoritmo controlar la tasa de mutación de la población. Dicho valor no tiene un límite superior, pero suele establecerse como 1. Además de esto, el índice del vector base, r_0, puede ser determinado de diversas formas, pero normalmente es aleatoriamente elegido y cumple que es diferente del índice del vector objetivo, i. Los índices usados en el vector de diferencia son distintos entre sí, del vector base y del vector objetivo, y son elegidos aleatoriamente, r_1 y r_2, para cada vector mutado. Una representacion gráfica de este proceso se puede observar en la Fig. 2

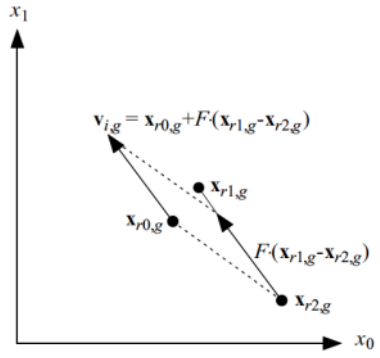

Figura 2: Vector de mutación en espacio bidimensional

2.1.3 Cruce

Para completar la estrategia de mutación es eficiente también el uso de una estrategia de cruce uniforme. El cruce se centra en construir los vectores de prueba haciendo uso de los parámetros tomados de dos vectores distintos, en este caso dichos vectores son la población inicial y los vectores mutados. Esto se puede ver en la la siguiente ecuación:

$$u_{i,g} = u_{j,i,g} = \begin{cases} v_{j,i,g} & \text{si } (rand_j(0,1) \leq C_r) \text{ o } (j < j_{rand}) \\ x_{j,i,g} & \text{otro caso} \end{cases} \tag{5}$$

La probabilidad de cruce $C_r \in [0, 1]$, es un valor el cual el usuario tendrá que definir, y con el cual el algoritmo controla la cantidad de valores copiados del vector mutado. Para determinar que elemento contribuye a un parámetro dado, el cruce uniforme se encarga de comparar C_r con la salida de un generador de números aleatorios, $rand_j(0, 1)$. Si dicho valor aleatorio es menor a C_r, se probará con un parámetro heredado de la mutación, en caso contrario, se probará con un valor de la población inicial. En la Fig. 3 se aprecia de manera gráfica dicho proceso.

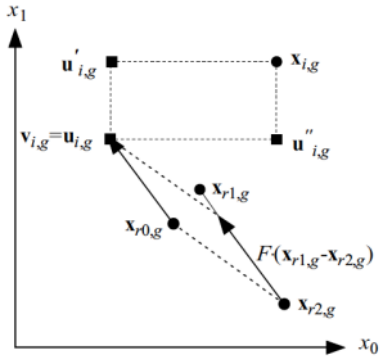

Figura 3: Posibles vectores de prueba a partir del cruce del vector mutado y la población inicial.

2.1.4 Selección

Tras el proceso de mutación y cruce, es necesario determinar cuales son los elementos que formarán parte de la nueva generación de valores, es por ello que es necesario realizar un proceso de selección. Para realizar dicho proceso, compararemos la salud del vector resultante de la mutacón y el cruce $u_{i,g}$ con la salud del vector inicial $x_{i,g}$, a través de la función de salud. De esta manera, el vector de la generación siguiente será el que tenga el mejor valor de ambos. Un esquema del proceso total puede observarse en la Fig. 4.

https://doi.org/10.17979/spudc.9788497498418.0812

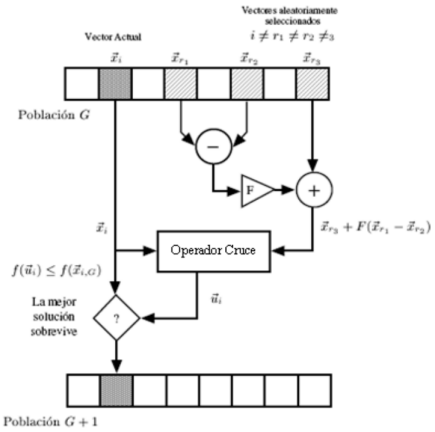

Figura 4: Esquema proceso completo.

2.2 APLICACIÓN DEL ALGORITMO

El objetivo de la aplicación creada es la de realizar el control del brazo robótico desarrollado por el laboratorio de Robótica Móvil de la Universidad Carlos III. Para ello, en lugar de aplicar el DE para la generación de trayectorias, se ha decidido hacer un controlador a un nivel más bajo, realizando directamente el control sobre los motores de cada una de las juntas del brazo. La idea es la de hacer uso de controladores PID pero, en lugar de realizar la elección de las características de cada motor aplicando técncias clásicas, se ha decidido usar el DE para dicho propósito [13].

Los controladores PID están conformados por 3 términos: proporcional(K_p), integral(K_i) y el derivativo(K_d). La combinación de estos tres parámetros permite mejorar la respuesta dinámica del sistema, reducir las sobre oscilaciones, eliminar el error estacionario e incrementar la estabilidad del sistema [14]. La función de transferencia típica de los controladores PID viene dada por:

$$C(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d s \qquad (6)$$

Por lo tanto, el objetivo del DE es la de obtener los valores de K_p, K_i, K_d que se han de aplicar a cada uno de los PID de los motores del sistema. Para la obtención de dichos parámetros, únicamente es necesario establecer los valores que se quieran en los parámetros de control, siendo esa una de las grande ventajas de este modelo frente a los modelos clásico u otros algoritmos genéticos. En primer lugar es necesario establecer cual va a ser nuestra función de coste o salud, la cual será la parte más compleja del proceso. Esta se explicará en la Sección 3. Tras tener una función de coste establecida, únicamente se tendrá que variar 4 factores.

- Tamaño de la población (N): controlará la cantidad de individuos que conforman una población. Se suele estimar como un buen tamaño una elección de $10 * D$, donde D nos indica el número de cromosomas, o dimensiones del problema (en este caso al trabajar con un PID, tendremos 3 cromosomas, K_p, K_i y K_d)
- Constante de cruce (C_r): cuyos valores se encuentran en [0,1]. Este valor, cuanto más próximo a 0, menos afectará el cruce al algoritmo, y cuanto más próximo a 1 mayor efecto tendrá.
- Constante de mutación (F): indica el factor de amplitud de diferencias del DE, es decir, como de dispersos se encuentran los datos. Una buena aproximación para dicho factor es trabajar con valores en un rango entre 0.5 y 1. Cuanto mayor sea la población N menor deberá escogerse el valor de F.
- Factor de parada: indicará cuando para de ejecutarse el algoritmo.

Con la modificación de estos elementos, se consigue obtener un resultado con el cual se puede controlar de manera correcta el brazo.

3 ENTORNOS DE PRUEBAS

3.1 MODELO SIMULADO

Para el desarrollo del modelo simulado así como para el control y la simulación del mismo se ha hecho uso de las herramientas proporcionadas por MATLAB. El modelo del brazo robótico se ha generado haciendo uso de SimScape, herramienta que permite generar modelos basados en conexiones físicas los cuales nos permiten modelar tantos las dinámicas como las cinemáticas de elementos reales. El modelo Fig.5, está conformado por dos partes claramente diferenciadas, el bloque controlador y el bloque del modelo.

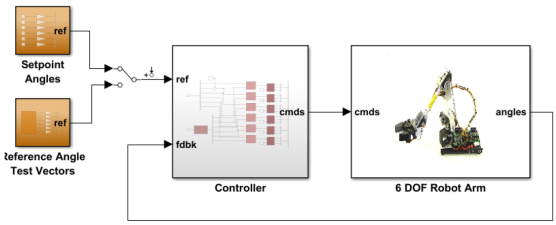

Figura 5: Modelo completo simulación.

Dentro de dicho controlador se encuentra los elementos específicos para poder realizar el control pertinente, como se puede apreciar en la Fig. 6.

https://doi.org/10.17979/spudc.9788497498418.0812

El subsistema *Controlador* consta de seis controladores PID digitales (uno por articulación). Cada controlador PID se implementa utilizando el bloque *2-DOF PID Controller* de la librería Simulink. El tiempo de muestreo del control es Ts=0,1 (10 Hz). Además de esto, se ha añadido unos bloques *To Workspace* con los cuales se enviará al código creado para controlar tanto la referencia como el valor del ángulo de cada una de las juntas a estudiar.

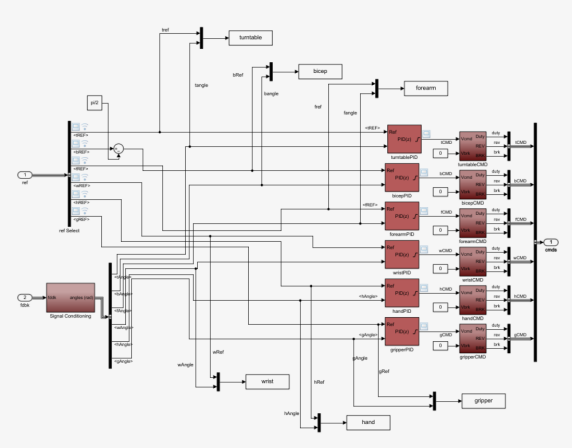

Figura 6: Modelo interno del controlador.

En la parte del modelo real encontramos el esquema mostrado en la Fig.7, donde se muestra el modelo mecánico del brazo, así como los actuadores correspondientes. Dichos actuadores son representados como servomotores en el modelo, pero se han configurado de igual manera que tengan las mismas caracteristícas que los motores del brazo real.

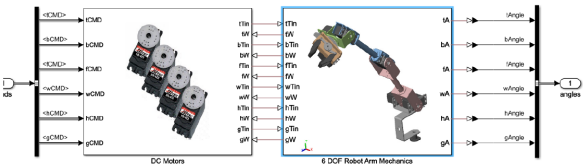

Figura 7: Modelo electromecánico.

El funcionamiento en simulación esta controlado por dos *scripts* en Matlab, donde en el primero de ellos se ha generado el algoritmo del DE y se realiza la programación de lo previamente descrito, y el segundo de ellos donde se evalúa, haciendo uso de la función de salud, los valores para K_p, K_i y K_d. Para dicho proceso, lo que se hace es enviar los datos al modelo descrito en la Fig.7, y se ejecuta el movimiento a través de un un pulso de entrada. En función de las posiciones finales se estimará el buen desempeño de dichos valores, y se les asignará un determinado peso de error. Puesto que lo que se quiere es optimizar, se buscará obtener el menor error posible. Por lo tanto, la función de salud se basa en una diferencia entre la señal de entrada y la de salida a la que le añadimos un eliminador de oscilaciones (picos), para evitar que los motores sufran vibraciones. De esta manera, aquellas oscilaciones que sean superiores a 1^o (rango mínimo de los motores reales) se entenderán como vibraciones dañinas para los motores, siendo así descartadas.

3.2 MODELO REAL

El modelo real desarrollado en el laboratorio es un prototipo utilizado como banco de pruebas de algoritmos de planificación y de control para manipuladores móviles. El brazo consta de cuatro GDL más un quinto siendo la pinza y se basan en rotaciones simples sobre los ejes . La primera de ellas permite que los tres eslabones que lo componen, que forman el brazo giren sobre la base. El rango de este movimiento está limitado por la longitud del cableado necesario para el movimiento de los eslabones, inicialmente se permiten 360^o. El modelo se puede apreciar en la Fig.8

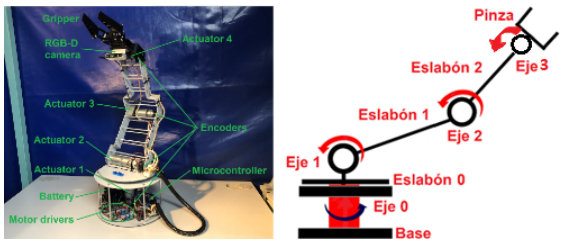

Figura 8: (Izquierda) Modelo brazo real,(Derecha) Esquema de eslabones.

Los eslabones uno y dos giran de forma conjunta sobre el eslabón cero por medio del segundo eje de rotación (eje 1). El rango mecánico de este movimiento es de 190^o. Por último, el eslabón dos gira respecto al eslabón uno por medio del tercer eje de rotación (eje 2). El rango de este movimiento es de 240^o. Los actuadores que permiten generar dichos movimientos están basados en motores de corriente continua de escobillas, acoplados a reductoras planetarias y de engranajes planos. La potencia de los mismos permite manejar al prototipo una carga de pago máxima de 350 g. La pinza tiene la capacidad de abrirse y cerrarse, contemplándose (al igual que en modelo simulado) como un GDL añadido. El movimiento de apertura y cierre se basa en un eje lineal accionado por medio de un tornillo sin fin que se acopla a un pequeño motor reductor de corriente continua de escobillas. La potencia

https://doi.org/10.17979/spudc.9788497498418.0812

mecánica es transmitida desde el tornillo sin fin a un mecanismo basado en un paralelepípedo articulado acoplado a las garras de la pinza.

4 RESULTADOS EXPERIMENTALES

4.1 SIMULACIÓN

Para las pruebas de funcionamiento se ha realizado un proceso de prueba y error hasta obtener los resultados que más se ajusten al proceso requerido. Para ello, lo que se ha hecho es dividir el proceso de entrenamiento en los 6 GDL. De esta manera, se fomentaba que el algoritmo convergiese más rápido ya que únicamente tiene que estimar un PID cada vez. Los experimentos en simulación se han basado en realizar modificaciones en los aprámetros descritos en la Sección. 2.2. Para la estimación de los valores de K_p, K_i y K_d se ha hecho uso de una entrada escalón, ya que de esta manera el sistema podía ajustarse de una manera más sencilla, y además con la idea de hacer uso de otro tipo de señales más complejas para la comprobación del correcto funcionamiento de los parámetros de los controladores. Un ejemplo de esto se puede apreciar en la Fig.9.

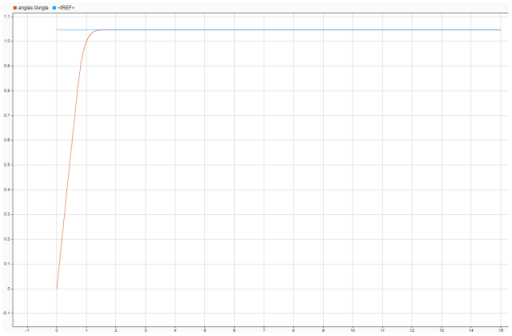

Figura 9: Ejemplo entrenamiento base frente entrada escalón.

Los resultados obtenidos, tras la realización de los entrenamientos para los 6 GDL se pueden apreciar en la Tabla 1.

Tabla 1: Resultados finales de los controladores.

Eslabón	K_p	K_i	K_d	**Error(rad)**
Base	12.2640	0	0	$3.5457e^{-6}$
Bicep	25.8466	0	5.1646	$1.6995e^{-2}$
Forearm	26.1168	0	2.9342	$1.259e^{-2}$
Wrist	33.8111	0	2.3999	$9.54e^{-4}$
Hand	15.4470	0	0	$3.457e^{-6}$
Gripper	14.489	0	0	$2.456e^{-6}$

Como se puede apreciar, en ningún caso sobrepasa el límite máxim establecido de 1^o, y prácticamente en todos los eslabones del brazo, se ha obtenido un error cercano a 0^o. Se puede observar que los que tienen un mayor error son los correspondientes al *Bicep* y al *Forearm*, ya que estos son los que más afectados se ven por el efecto del péndulo invertido al cual se somete el brazo, ya que la parte de la pinza al ser más pesada que el resto de los elementos, desplaza el centro de masas hacia el extremo, fomentando así la generación de dicho efecto. Esto se puede apreciar en la Fig.10

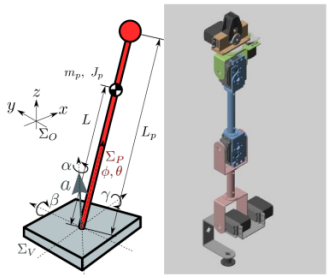

Figura 10: Efecto péndulo invertido.

En la Fig.11 se muestra un ejemplo de los resultados de los controladores PID con los valores de la Tabla 1 frente a diversas entradas.

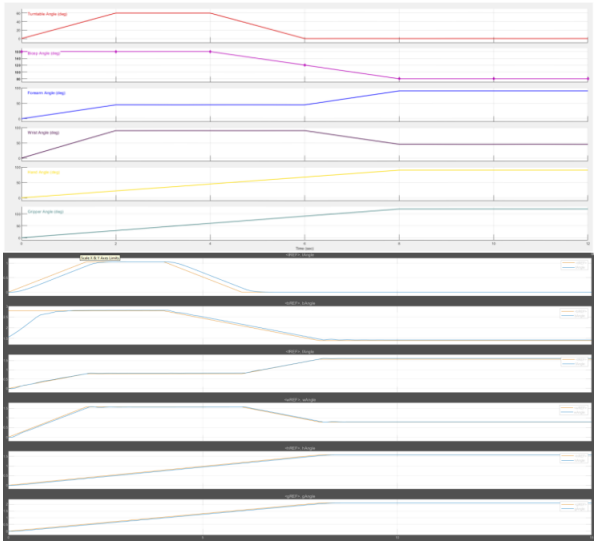

Figura 11: (Arriba) Señal de entrada propuesta, (Abajo)Respuesta del brazo frente a entrada propuesta.

4.2 BRAZO REAL

Para la realización de las pruebas con el brazo real se han realizado una serie de trayectorias combinando los posibles movimientos del brazo. Para ello, se ajustaron los PID de los driver de cada uno de los motores del brazo con los resultados

obtenidos en simulación y se comprobó que las trayectorias realizadas eran las mismas que las obtenidas en el entorno simulado. Esto se ha podido reaalizar de manera directa, ya que en la simulación se han establecido los parámetros dinámicos y cinemáticos exactamente iguales a los del modelo real, es decir, se ha hecho un gemelo digital. La única diferencia es la cantidad de grados de libertad entre el modelo simulado y el real. Puesto que el real tiene 1 GDL menos que el simulado, y dichos GDL corresponden con movimiento del efector final (*Hand*), no afectan de manera directa al movimiento del modelo real, por lo tanto, se desprecian dichos valores obtenidos para ese GDL y se trabajará de igual manera. Algunos ejemplos de posiciones alcanzadas con el brazo real pueden observarse en la Fig.12.

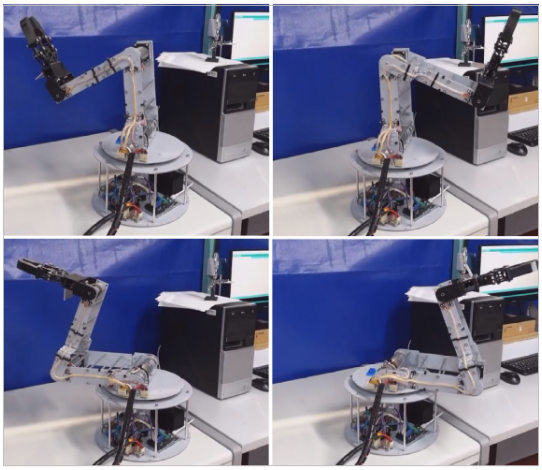

Figura 12: Secuencia de movimientos del brazo real.

Los resultados de las pruebas comparando el modelo real y el modelo simulado se pueden ver en el siguiente vídeo: `https://youtu.be/L5_X5BUt0rU`

5 CONCLUSIONES Y TRABAJOS FUTUROS

Con este trabajo se ha intentado realizar un control de un modelo simulado y real en una plataforma propia generada como base experimental de prueba de controladores. Los resultados obtenidos tanto en el modelo real como en el modelo simulado arrojan que los valores del controlador consiguen optimizar de manera correcta los diferentes PID del sistema en cuestión. Además de esto, en este trabajo se ha demostrado los beneficios del uso de DE en este tipo de proceso, ya que la obtención de los valores de los controladores está sujeto únicamente a la modificación de los valores iniciales del algoritmo (población, constante de cruce...), por lo tanto, frente a algoritmos clásicos de teoría de control, donde es necesario realizar una serie de cálculos previos para la obtención de la planta del sistema, este proceso ahorra una gran cantidad de tiempo la cual, en el ,ámbito de la investigación permite poder seguir desarrollando el trabajo de manera paralela. La función por lo tanto del ingeniero es la de establecer la función de salud o coste y la de corroborar que los resultados obtenidos cumplen los requisitos del sistema.

Como trabajos futuros se propone la aplicación del método para el control de diversos brazos robóticos (como podría ser los UR3, UR5...), centrandose no solo en la generación y control de las trayectorias como ya se ha hecho, sino en la aplicación de dicho método para el control a un menor nivel. Otro posible trabajo futuro podría ser el de generar una comparativa entre el modelo generado con DE y otros tipos de algoritmos genéticos, los cuales proporcionarían una base datos comparativa entre los diversos métodos.

Agradecimientos

Este proyecto ha sido apoyado por el proyector HEROITEA: Heterogeneous Intelligent Multi-Robot Team for Assistance of Elderly People (RTI2018-095599-B-C21), financiado por el Ministerio de Economía y Competitividad español, y el proyecto RoboCity2030 DIH-CM (S2018/NMT-4331, RoboCity2030 Madrid Robotics Digital Innovation Hub)

English summary

CONTROL OF A LOW-COST ROBOTIC ARM BY DIFFERENTIAL EVOLUTION

Abstract

Robotic manipulators are highly non-linear systems, and it is difficult to obtain an accurate mathematical model with conventional techniques. Applying classical control techniques can solve this type of problem, but it has the disadvantage that it takes a large amount of time to obtain a satisfactory result for manipulators with a large number of degrees of freedom (GDL). Therefore, an efficient technique is required to deal with such complex and dynamic systems. The Differential Evolution (DE) algorithm is a very powerful global optimisation technique which has nowa-

 https://doi.org/10.17979/spudc.9788497498418.0812

days become popular due to its possible application in a large number of fields within robotics. This paper shows a direct application of this optimisation method on a simulated model of a 6 GDL arm and a subsequent application on a low-cost model.

Keywords: Differential Evolution, Manipulation, Genetic Algorithms.

Referencias

[1] Corne, D., Dorigo, M., Glover, F., Dasgupta, D., Moscato, P., Poli, R., & Price, K. V. (Eds.). (1999). New ideas in optimization. McGraw-Hill Ltd., UK.

[2] Opara, K. R., & Arabas, J. (2019). Differential Evolution: A survey of theoretical analyses. *Swarm and evolutionary computation*, 44, 546-558.

[3] Ahmad, M. F., Isa, N. A. M., Lim, W. H., & Ang, K. M. (2021). Differential evolution: A recent review based on state-of-the-art works. *Alexandria Engineering Journal*.

[4] Storn, R., & Price, K. (1997). Differential evolution a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4), 341-359.

[5] Borase, R. P., Maghade, D. K., Sondkar, S. Y., & Pawar, S. N. (2021). A review of PID control, tuning methods and applications. *International Journal of Dynamics and Control*, 9(2), 818-827.

[6] Moreno, L., Garrido, S., Blanco, D., & Munoz, M. L. (2009). Differential evolution solution to the SLAM problem. *Robotics and Autonomous Systems*, 57(4), 441-450.

[7] Das, S. D., Bain, V., & Rakshit, P. (2018, June). Energy optimized robot arm path planning using differential evolution in dynamic environment. In *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)* (pp. 1267-1272). IEEE.

[8] Hernández-Barragán, J., López-Franco, C., Alanis, A. Y., Arana-Daniel, N., & López-Franco, M. (2019). Dual-arm cooperative manipulation based on differential evolution. *International Journal of Advanced Robotic Systems*, 16(1), 1729881418825188.

[9] Saravanan, R., Ramabalan, S., & Balamurugan, C. (2008). Evolutionary optimal trajectory planning for industrial robot with payload constraints. *The International Journal of Advanced Manufacturing Technology*, 38(11), 1213-1226.

[10] Muñoz, J., López, B., Quevedo, F., Barber, R., Garrido, S., & Moreno, L. (2022). Geometrically constrained path planning for robotic grasping with Differential Evolution and Fast Marching Square. *Robotica*, 1-19.

[11] Price, K. V. (2013). Differential evolution. In *Handbook of optimization* (pp. 187-214). Springer, Berlin, Heidelberg.

[12] Pant, M., Zaheer, H., Garcia-Hernandez, L., & Abraham, A. (2020). Differential Evolution: A review of more than two decades of research. *Engineering Applications of Artificial Intelligence*, 90, 103479.

[13] Saad, M. S., Jamaluddin, H., & Darus, I. Z. M. (2012). Implementation of PID controller tuning using differential evolution and genetic algorithms. *International Journal of Innovative Computing, Information and Control*, 8(11), 7761-7779.

[14] Wu, H., Su, W., & Liu, Z. (2014, June). PID controllers: Design and tuning methods. In 2014 *9th IEEE Conference on industrial electronics and applications* (pp. 808-813). IEEE.