Universidad Carlos III de Madrid
Escuela Politécnica Superior
Departamento de
Ingeniería de Sistemas y Automática

Ingeniería Técnica Industrial:
Electrónica Industrial

Proyecto Fin de Carrera

Evaluación de técnicas evolutivas en el cálculo de la cinemática inversa de robots manipuladores

AUTOR: ÁLVARO BRAVO SANCHO
TUTOR: CARLA GONZÁLEZ UZCÁTEGUI

LEGANÉS, JULIO DE 2012

Título: *EVALUACIÓN DE TÉCNICAS EVOLUTIVAS PARA EL CÁLCULO DE LA CINEMÁTICA INVERSA DE ROBOTS MANIPULADORES.*

Autor: *ÁLVARO BRAVO SANCHO*

Tutor: *CARLA GONZÁLEZ UZCÁTEGUI*

La defensa del presente Proyecto Fin de Carrera se realizó el día 19 de Julio de 2012; siendo calificada por el siguiente tribunal:

Presidente: *María Dolores Blanco Rojas*

Secretario: *Fernando Martín Monar*

Vocal: *Teresa Onorati*

Habiendo obtenido la siguiente calificación:

Calificación:

Presidente **Secretario** **Vocal**

Agradecimientos

Quiero agradecer a mi tutora Carla González por su interés y apoyo.

A mis padres que tanto me han apoyado.

Resumen

En este proyecto, se examina el rendimiento de un algoritmo de búsqueda directa a partir del problema de cinemática inversa de robots manipuladores con diferentes grados de libertad. Para ello se plantea el siguiente problema de optimización: conocida la posición y orientación deseada en el extremo final de un robot manipulador, y su función de cinemática directa, encontrar un posible vector de configuración articular óptimo dentro del espacio de trabajo del robot, de manera que, los errores de posición y orientación son mínimos. Un enfoque evolutivo, llamado *Differential Evolution*, es el utilizado para resolver el anterior problema de optimización con restricciones no lineales. *Differential Evolution* es un método de búsqueda directa estocástico basado en un proceso iterativo de mutación algebraica, cruce y selección sobre una población de soluciones candidatas. Basada en la experimentación empírica, este estudio pretende mostrar la influencia de los valores de los parámetros de control (tamaño de la población, factor de escala de mutación, etc.) en el comportamiento del algoritmo para el problema de inversión cinemática. Finalmente, se introduce un operador de búsqueda local para mejorar la velocidad de convergencia del algoritmo.

Abstract

In this dissertation, the performance of a direct search algorithm is examined for the inverse kinematics problem of robot manipulators with different degrees of freedom. The following optimization model has been considered: given a desired end-effector position and orientation, and the forward kinematics function of a robot manipulator, find a feasible optimal joint configuration vector within the robot's workspace, such that, the position and orientation errors are minimal. An evolutionary-based approach, called Differential Evolution, is used to solve the above nonlinear constrained optimization problem. Differential Evolution is a stochastic direct search method based on an iterative process of algebraic mutation, crossover, and selection over a population of candidate solutions. Based on empirical experimentation, the study aims to reveal the influence of control parameters values (population size, mutation scale factor, etc.) on the algorithm behaviour for the kinematic inversion problem. Finally, a local search operator is introduced to improve the convergence speed of the algorithm.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

INTRODUCCIÓN

1.1. Introducción

Un robot manipulador es una cadena cinemática compuesta por eslabones unidos entre si mediante articulaciones, con un extremo unido a la base y el otro extremo libre unido a una herramienta con la cual se realiza una tarea sobre el entorno, por ejemplo tareas de soldadura, pintura, ensamblaje de piezas, entre otros. El control de un robot manipulador involucra la planificación de la trayectoria que debe ejecutar el robot. Para ello es necesario modelar la dinámica y la cinemática del robot. Esto último es el punto de interés de este proyecto.

La cinemática se interesa por la descripción analítica del desplazamiento espacial del robot como una función del tiempo sin considerar las fuerzas que actúan sobre él. En particular, de las relaciones entre la posición de las variables de articulación y la posición y orientación del elemento final del brazo del robot. Hay dos problemas fundamentales en la cinemática del robot. El primero es conocido como el problema cinemático directo, en el cual se trata de determinar las coordenadas cartesianas de posición y orientación del elemento final a partir del conocimiento de las variables de cada articulación. El segundo corresponde con el problema cinemático inverso, el cual consiste en determinar las variables articulares a partir de una posición y orientación conocida del elemento final.

Resolver la cinemática inversa de forma exacta y rápida representa la piedra angular del control cinemático de los robots manipuladores. Puesto que en la mayoría de los casos no es posible obtener una solución analítica de la ecuación de cinemática inversa, se hace necesario utilizar técnicas numéricas que permitan obtener de forma aproximada el vector de coordenadas articulares

del robot a partir de la posición y orientación del elemento final en el espacio Cartesiano.

En general, los métodos numéricos utilizados para resolver la cinemática inversa de robots manipuladores se basan en la relación lineal que existe entre la velocidad del elemento final y la velocidad de las articulaciones, la cual queda establecida mediante la matriz Jacobiana del robot. De esta forma, la inversión cinemática se reduce a un proceso iterativo de interpolación lineal y de cálculo de la matriz Jacobiana inversa o pseudoinversa. Los métodos más utilizados son los basados en el método de Newton-Raphson [44], [9] [45] y [1], y el método de los mínimos cuadrados amortiguados [42]. Una primera desventaja de los algoritmos basados en la matriz Jacobiana es el alto coste computacional asociado a la inversión matricial en cada iteración. La segunda desventaja, aún más importante, es la inestabilidad numérica de estos algoritmos alrededor de las configuraciones singulares del robot manipulador. En estas configuraciones la matriz Jacobiana no posee rango completo, no es invertible, y por lo tanto el algoritmo no converge a una solución.

Un enfoque alternativo consiste en formular el problema de la cinemática inversa como un problema de optimización de alguna función objetivo adecuada de acuerdo a las tareas del robot. En general, consiste en calcular un vector de coordenadas articulares que minimice el error de posición y orientación del extremo respecto a un vector de coordenadas cartesianas deseadas. En la literatura, el problema anterior se ha abordado mediante técnicas clásicas de optimización como el método del gradiente [20] y el método *Cyclic Coordinate Descent* (CCD) [43]; y técnicas de inteligencia artificial como las redes neuronales [10], [17], [22], [24], [16], [14] y [7]; y los algoritmos evolutivos [26], [40], [21] y [13]. Entre los numerosos esquemas de optimización propuestos destacan los Algoritmos Evolutivos (AEs), en primer lugar porque no imponen restricciones de continuidad sobre la función objetivo, la búsqueda no se basa en la información local proporcionada por un solo punto, como en las técnicas basadas en el gradiente; y no requieren de una fase previa de entrenamiento como las redes neuronales. Todo esto, unido a un esquema de búsqueda aleatoria global basado en poblaciones de soluciones e intrínsecamente paralelo.

En este proyecto el problema cinemático se formula como un problema de optimización con restricciones y la solución óptima se obtiene mediante un algoritmo de optimización evolutivo conocido como *Differential Evolution* (DE). *Differential Evolution* es un método de optimización global de búsqueda directa cuya estructura responde al esquema general de los Algoritmos Evolutivos, y cuyo conductor principal consiste en un esquema algebraico de mutación. El algoritmo DE tiene numerosas ventajas: es robusto, flexible y fácil de implementar.

1.2. Objetivos

En este proyecto se propone resolver la cinemática inversa de robots manipuladores de diferentes grados de libertad. Para ello se plantea el siguiente problema de optimización con restricciones: conocida la posición y orientación deseada en el extremo del robot manipulador y la función de cinemática directa, calcular el vector de coordenadas articulares óptimo que minimice la función de error de posición y orientación del extremo, tal que la solución óptima se encuentre dentro del espacio de trabajo definido por los límites mecánicos de las articulaciones del robot. Teniendo como principales objetivos:

- Estudiar la cinemática inversa y directa de diversas plataformas robóticas.
- Estudiar las características del algoritmo evolutivo *Differential Evolution* como método de optimización.
- Utilizar el algoritmo *Differential Evolution* para resolver el problema de cinemática inversa de diversas plataformas robóticas.
- Evaluación de los parámetros de control del algoritmo *Differential Evolution* y su importancia en la convergencia del algoritmo.
- Implementar y evaluar soluciones para mejorar la rapidez de convergencia del algoritmo evolutivo.

1.3. Estructura del Documento

Este documento consta de cinco capítulos en los cuales se plantea y desarrolla la solución propuesta al problema de la cinemática inversa de robots manipuladores. En el capítulo 1 el problema se enmarca en el contexto del método propuesto y se enumeran los objetivos del trabajo. En el capítulo 2 se presenta la cinemática de robots manipuladores y se describe la resolución de sus dos problemas fundamentales: la cinemática directa y la cinemática inversa. En el capítulo 3 se describe el método *Differential Evolution* así como la mejora implementada denominada mecanismo *discarding*. En el capítulo 4, se muestran los resultados obtenidos de las pruebas empíricas realizadas para comprobar la influencia de los valores de los parámetros de control en el comportamiento del algoritmo para el problema de inversión cinemática, así como, los resultados obtenidos al introducirle un operador de búsqueda local para mejorar la velocidad de convergencia del algoritmo. Finalmente, en el capítulo 5 se enumeran algunas conclusiones y

se proponen las líneas de trabajo futuras. En el Apéndice A se presenta el presupuesto necesario para la realización de este documento.

Capítulo 2

CINEMÁTICA ROBOTS MANIPULADORES

2.1. Introducción

Un robot manipulador consiste en una cadena cinemática abierta de D grados de libertad (GDL) compuesta por cuerpos rígidos llamados eslabones unidos entre sí mediante articulaciones. Aquí se considera que cada articulación aporta un grado de libertad a la estructura cinemática del robot y que cada una posee un movimiento angular o lineal respecto a su eje según sea de tipo rotacional o prismática respectivamente [33].

La cinemática de los robots manipuladores estudia el movimiento del mismo con respecto a un sistema de referencia sin considerar las fuerzas o pares que actúan sobre él. Así, la cinemática se interesa por la descripción analítica del movimiento espacial del robot manipulador como una función del tiempo, y en particular por las relaciones entre la posición y la orientación del extremo final del robot con los valores que toman sus coordenadas articulares. Existen dos problemas fundamentales para resolver la cinemática de los robots manipuladores. El primero de ellos se conoce como problema cinemático directo, y consiste en determinar cual es la posición y orientación del extremo final del robot, respecto a un sistema de coordenadas que se toma como referencia, conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del robot manipulador. El segundo, denominado problema cinemático inverso resuelve la configuración articular que debe adoptar el robot para una posición y orientación del extremo conocidas.

2.2. Cinemática directa

El problema de la cinemática directa consiste en determinar la posición y orientación del extremo final del manipulador respecto a un sistema de referencia fijo conocida la configuración articular del robot $\vec{q} = [q_1, q_2, \ldots, q_i]$, donde $q_i \in \mathbb{R}$, $i = 1, \ldots, D$ es una variable articular del robot.

Una herramienta indispensable para describir la geometría espacial de un manipulador es la representación en coordenadas homogéneas. El concepto de una representación en coordenadas homogéneas en un espacio Euclídeo tridimensional es útil para desarrollar transformaciones matriciales que incluyan rotación, traslación, escalado y transformación de perspectiva.

En general, una matriz de transformación homogénea es una matriz de dimensión 4X4 que representa la transformación de un vector de coordenadas homogéneas de un sistema de coordenadas hasta otro sistema de coordenadas. En general se define como:

$$T = \begin{bmatrix} R_{3x3} & p_{3x1} \\ f_{1x3} & w_{1x1} \end{bmatrix} = \begin{bmatrix} \text{Matriz de rotación} & \text{Vector de posición} \\ \text{Transformación de perspectiva} & \text{Escalado} \end{bmatrix} \tag{2.1}$$

donde la submatriz R_{3x3} representa la matriz de rotación, la submatriz p_{3x1} representa el vector de traslación, la submatriz f_{1x3} representa la transformación de perspectiva, y la submatriz w_{1x1} es el factor de escala global. En aplicaciones de robótica generalmente sólo interesa conocer el valor de R_{3x3} y p_{3x1}, considerando una transformación de perspectiva nula y un factor de escala unitario, resultando en una matriz de transformación homogénea de la forma:

$$T = \begin{bmatrix} R_{3x3} & p_{3x1} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \text{Matriz de rotación} & \text{Vector de posición} \\ 0 & 1 \end{bmatrix} \tag{2.2}$$

Para analizar la cinemática de los robots manipuladores en este proyecto se utiliza la representación de Denavit-Hartenberg [6], la cual establece en forma sistemática un sistema de coordenadas para cada elemento de la cadena articulada. Dicha representación resulta en una matriz de transformación homogénea que representa cada uno de los sistemas de coordenadas de los elementos en la articulación con respecto al sistema de coordenadas del elemento previo. Para su aplicación, se tiene que establecer sistemas de coordenadas como se muestra en la figura 2.1.

A partir del sistema de coordenadas elegido y las dimensiones del brazo robot, se hallan los parámetros de la representación de Denavit-Hartenberg (θ_j, d_j, a_j, α_j) donde $j = 1, \ldots, D$.

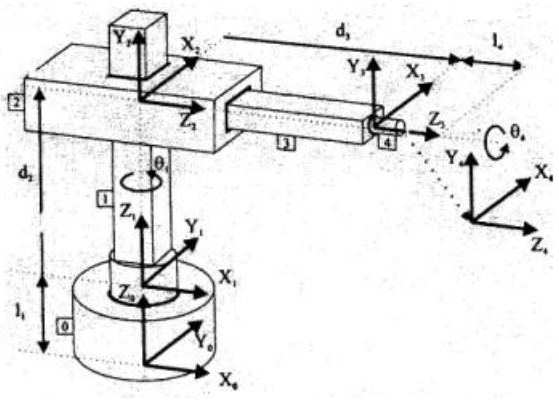


Figura 2.1: Robot Cilíndrico[2]

θ_j: Es el ángulo de la articulación del eje x_{j-1} al eje x_j respecto del eje z_{j-1}(utilizando la regla de la mano derecha).

d_j: Es la distancia desde el origen del sistema de coordenadas (j-1)-ésimo hasta la intersección del eje z_{j-1} con el eje x_j a lo largo del eje z_{j-1}.

a_j: Es la distancia de separación desde la intersección del eje z_{j-1} con el eje x_j hasta el origen del sistema j-ésimo a lo largo del eje x_j (ó la distancia más corta entre los ejes z_{j-1} y z_j).

α_j: Es el ángulo de separación del eje z_{j-1} al eje z_j respecto del eje x_j (utilizando la regla de la mano derecha).

Estos cuatro parámetros describen totalmente cualquier articulación prismática o rotacional. Son reemplazados en la matriz de transformación D-H en sistemas de coordenadas adyacentes j e j-1, notados en la expresión:

$$A_j^{j-1} = \begin{bmatrix} \cos\theta_j & -\cos\alpha_j \sin\theta_j & \sin\alpha_j \sin\theta_j & a_j \cos\theta_j \\ \sin\theta_j & \cos\alpha_i \cos\theta_j & -\sin\alpha_j \cos\theta_j & a_j \sin\theta_j \\ 0 & \sin\alpha_j & \cos\alpha_j & d_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

De esta forma el problema cinemático directo se reduce a calcular la matriz homogénea T_n^0, a partir de la matriz de transformación descrita en la expresión (2.3), la cual especifica la localización del sistema de coordenadas n-ésimo con respecto al sistema de coordenadas base,

$$f_k(q) = T_n^0 = \prod_{j=1}^{n} A_j^{j-1} = A_1^0 A_2^1 \dots A_j^{j-1} \dots A_n^{n-1} \tag{2.4}$$

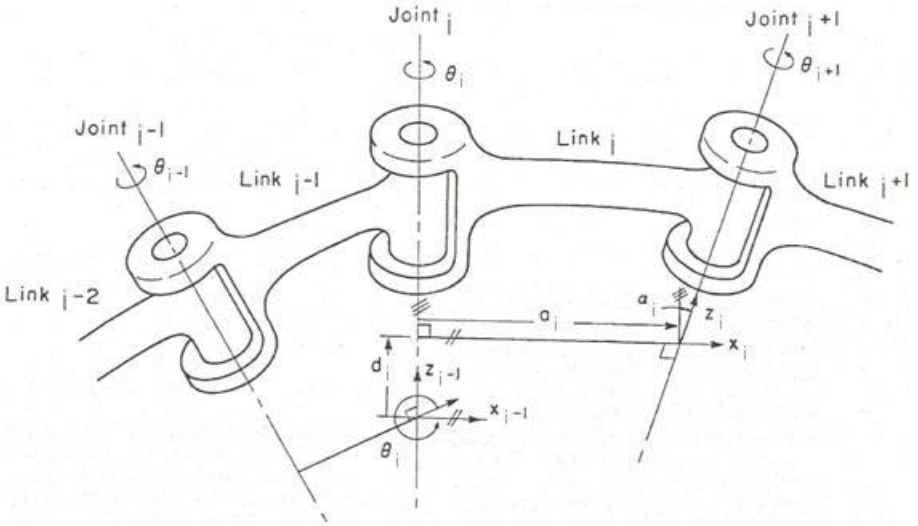


Figura 2.2: Asignación de sistemas de referencia fijos a cada articulación del robot y definición de los parámetros de Denavit-Hartenberg (a_j, d_j, α_j, θ_j)[27]

2.3. Cinemática inversa

El problema cinemático inverso consiste en encontrar los valores que deben adoptar las configuraciones articulares del robot $\vec{q}$ a partir de la orientación y posición del efector final,

$$\vec{q} = f_k^{-1}(p, \varphi) \tag{2.5}$$

donde p representa la posición y φ la orientación del extremo del robot.

Así como es posible abordar el problema cinemático directo de una manera sistemática a partir de la utilización de matrices de transformación homogéneas, e independientemente de la configuración del robot, no ocurre lo mismo con el problema cinemático inverso. En general, resolver la cinemática inversa de un mecanismo articulado general involucra resolver un sistema de ecuaciones fuertemente no lineales dependiente de la configuración articular del robot.

A la hora de resolver el problema cinemático inverso es adecuado encontrar una solución cerrada. Este tipo de solución presenta diversas ventajas. En muchas aplicaciones, el problema cinemático inverso ha de resolverse en tiempo real (por ejemplo. en el seguimiento de una determinada trayectoria), éste método garantiza tener la solución en el momento adecuado. Al contrario de lo que ocurría en el problema cinemático directo, con cierta frecuencia la solución del problema cinemático inverso no es única; existiendo múltiples configuraciones articulares $\vec{q}$

dada una posición y orientación deseada en el extremo. En estos casos una solución cerrada permite incluir determinadas reglas o restricciones que aseguren que la solución obtenida sea la más adecuada posible a la deseada.

No obstante, a pesar de las dificultades comentadas, la mayor parte de los robots poseen cinemáticas relativamente simples que facilitan en cierta medida la resolución de su problema cinemático inverso. Por ejemplo si se consideran solo los tres primeros grados de libertad de muchos robots, estos tienen una estructura planar, esto es, los tres primeros elementos quedan contenidos en un plano [19]. Esta circunstancia facilita la resolución del problema. Asimismo, en muchos robots se da la circunstancia de que los tres grados de libertad últimos, dedicados fundamentalmente a orientar el extremo del robot, correspondan a giros sobre los ejes que se cortan en un punto(por ejemplo: robot manipulador Puma 560) [28]. De nuevo esta situación facilita el calculo de las múltiples configuraciones articulares $\vec{q}$ correspondiente a la posición y orientación deseadas. Por lo tanto, para los casos citados, es posible establecer ciertas pautas generales que permitan plantear y resolver el problema cinemático inverso de una manera sistemática.

Los métodos geométricos son útiles para manipuladores simples con articulaciones rotacionales. Permiten tener normalmente los valores de las primeras variables articulares, que son las que consiguen posicionar el robot. Para ello utilizan relaciones trigonométricas y geométricas sobre los elementos del robot. Se suele recurrir a la resolución de triángulos formados por los elementos y articulaciones del robot. Como alternativa para resolver el mismo problema se puede recurrir a manipular directamente las ecuaciones correspondientes al problema cinemático directo.

Si se consideran robots con capacidad de posicionar y orientar su extremo en el espacio, esto es, robots con 6 grados de libertad, el método de desacoplamiento cinemático permite, para determinados tipos de robots, resolver los primeros grados de libertad, dedicados al posicionamiento, de una manera independiente a la resolución de los últimos grados de libertad, dedicados a la orientación. Cada uno de estos dos problemas simples podrá ser tratado y resuelto por cualquier procedimiento.

En contraste, problemas cinemáticos de estructuras robóticas más complejas requieren soluciones numéricas. Por ejemplo, utilizando una solución iterativa numérica basada en la matriz Jacobiana [11], [15]. Este método está basado en la relación que existe entre la velocidad del extremo y la velocidad en los actuadores de las articulaciones del robot mediante la matriz Jacobiana inversa. Además en el análisis y control del movimiento de un robot la matriz Jacobiana permite conocer el área de trabajo del robot y determinar sus singularidades. Esta solución pue-

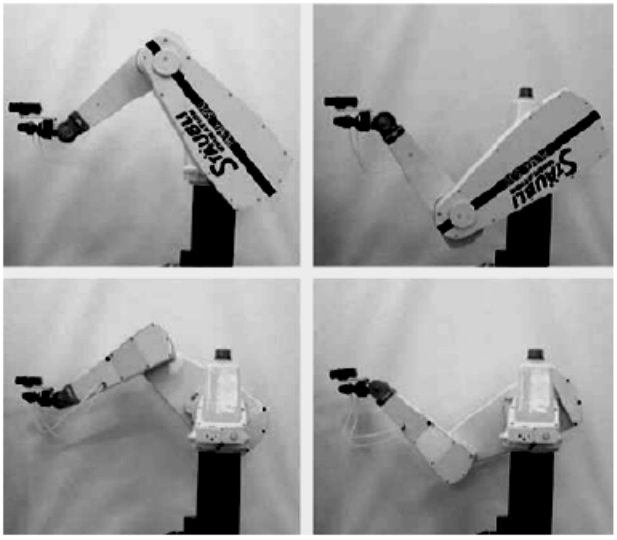


Figura 2.3: Múltiples configuraciones articulares que corresponden a una misma posición y orientación en el extremo de un robot manipulador de 6GDL.

de aplicarse para la mayoría de las configuraciones de manipulador comunes en la industria. Sin embargo, aunque es un método que converge rápidamente, se trata de un método local y no siempre garantiza producir todas las soluciones cinemáticas inversas posibles. Tiene problemas de convergencia en las singularidades del robot lo que implica un costo computacional significativo.

Una alternativa a la matriz Jacobiana consiste en plantear la cinemática inversa como un problema de optimización, esto es, calcular un vector de coordenadas articulares q que minimice una función de costo sujeto a las restricciones impuestas por la estructura o tarea del robot manipulador. Hasta la fecha se han propuesto numerosos esquemas que resuelven este problema de programación no lineal mediante técnicas clásicas de optimización como los métodos basados del gradiente en [20], métodos heurísticos de búsqueda directa (CCD) en [43], métodos utilizando redes neuronales en[4] y [25], y algoritmos evolutivos en [26] y [13], entre otros. La principal ventaja de estos métodos es su estabilidad numérica frente a los métodos basados en la matriz Jacobiana. Sin embargo estos métodos también tienen sus desventajas. Los métodos de optimización basados en el gradiente tienden a converger en puntos subóptimos y la función objetivo debe ser continua. El método CCD no emplea la matriz Jacobiana inversa y por lo tanto es estable alrededor de configuraciones singulares del robot, es de fácil implementación pero puede resultar de convergencia lenta dependiendo del número de grados de libertad del robot. Las redes neuronales tienen la capacidad de representar funciones no lineales, por lo que permiten representar

las relaciones cinemáticas inversas sin necesidad de especificar los parámetros cinemáticos del robot; sin embargo, el tiempo de entrenamiento es elevado y existen errores debido al proceso de aprendizaje.

Los algoritmos evolutivos destacan sobre los otros métodos anteriormente mencionados, con un esquema de búsqueda basado en la evolución de una población de soluciones mediante mecanismos estocásticos de reproducción, mutación y selección inspirados en procesos biológicos naturales [3]. Estos mecanismos permiten explorar el espacio de búsqueda sin necesidad de entrenamiento como las redes neuronales, siendo únicamente necesario definir los parámetros de cinemática directa del robot manipulador. Además no utilizan la matriz Jacobiana ni el gradiente de la función en el cálculo de la cinemática inversa de los robots manipuladores, por lo tanto el algoritmo no está condicionado por las configuraciones singulares del robot ni existen restricciones de continuidad sobre la función objetivo, lo que le aporta flexibilidad y robustez al algoritmo especialmente cuando la tarea asignada al robot demanda la conjunción de diversos objetivos.

Los algoritmos evolutivos, en particular los algoritmos genéticos, han sido utilizados para resolver la cinemática inversa de robots manipuladores para puntos individuales. En [26], una población de coordenadas articulares del robot son codificadas como cadenas de bits y los algoritmos genéticos binarios son utilizados para obtener la configuración articular óptima que minimiza el error de posición del extremo y el máximo desplazamiento angular de las articulaciones del robot. En [8] se describe un método para la cinemática inversa utilizando la programación genética, siendo capaz de automatizar el control de movimiento de bajo nivel, siendo la principal ventaja de este método su simplicidad y su velocidad cuando se trata con especificaciones complicadas. En [12] utilizan una función multi-objetivo de optimización satisfaciendo posición y orientación del extremo del robot. Introducen una tercera función objetivo para minimizar las velocidades discretas conjuntas, trabajando eficientemente con singularidades y diseños redundantes.

Una desventaja de los algoritmos genéticos binarios son los errores de exactitud asociados a la codificación binaria de parámetros continuos, razón por la cual se han introducido nuevos algoritmos adaptados a la exploración de espacios de búsqueda continuos, uno de estos algoritmos es el *Differential Evolution* (DE).

2.3.1. Cinemática inversa como problema de optimización

En este proyecto para resolver la cinemática inversa de robots manipuladores se plantea el siguiente problema de optimización: conocida la posición y orientación deseada en el extremo del robot manipulador y la función de cinemática directa, se calcula el vector de coordenadas articulares óptimo que minimice la función de error absoluto de posición y orientación del extremo, tal que la solución óptima se encuentre dentro del espacio de trabajo definido por los límites mecánicos de las articulaciones del robot.

Para ello se considera una población de NP vectores de coordenadas articulares de un robot manipulador de N grados de libertad, $\{\vec{q}_1, \ldots, \vec{q}_i, \ldots, \vec{q}_{NP}\}$, donde cada vector $\vec{q}_i = (q_{1,i}, \ldots, q_{j,i}, \ldots, q_{N,i})$, $i = 1, \ldots, NP$, cumple la ecuación de cinemática directa,

$$(\vec{p}_i, \varphi_i) = f_k(\vec{q}_i), \text{ sujeto a } (\vec{p}_i, \varphi_i) \in C \tag{2.6}$$

donde $\vec{p}_i$ es la posición y φ_i la orientación del extremo del robot, f_k es la función de cinemática directa que se obtiene a partir de las relaciones geométricas entre las articulaciones del manipulador y el extremo expresado respecto a un sistema de referencia fijo, y C es el espacio de trabajo definido por todos los puntos del espacio Cartesiano que el robot manipulador puede alcanzar. En el espacio de configuraciones el espacio de trabajo está definido por los límites de desplazamiento articular, esto es, cada parámetro $q_{j,i}$ del vector $\vec{q}_i$ debe cumplir, $q_j^- \leq q_{j,i} \leq q_j^+$, donde q_j^- y q_j^+ son respectivamente los límites inferior y superior de la articulación $j = 1, \ldots, N$.

Dadas la posición y orientación $(\vec{p}_d, \varphi_d)$ deseadas del extremo del robot, a cada vector de la población se le asigna un valor de aptitud (*fitness*) que describe que tan cercana es la solución de cinemática inversa de cada individuo q_i al valor deseado. La selección de la función de aptitud es fundamental en el proceso de selección de los mejores individuos y por lo tanto lo es para la convergencia del proceso de optimización. Una función de aptitud adecuada y ampliamente utilizada es el error absoluto de posición y orientación definidos como,

$$E_p = \mid p_d - p_i \mid \text{ y } E_o = \mid \varphi_d - \varphi_i \mid \tag{2.7}$$

donde la posición y orientación $(\vec{p}_i, \varphi_i)$ se calcula a partir de la ecuación de cinemática directa del robot descrita en la ecuación (2.6).

Ambos criterios se combinan en una función de error total definido como la suma ponderada del error de posición y el error de orientación,

$$f(q) = w_p E_p + w_o E_o \tag{2.8}$$

donde w_p y w_o son constantes positivas de normalización.

Finalmente el problema de optimización de cinemática inversa consiste en hallar un vector $\vec{q}$ de la población que minimice el error de posición y orientación, esto es, *mín f(q)* sujeto a $q_j^- \leq q_j \leq q_j^+$.

El método utilizado para resolver este problema de optimización es el *Differential Evolution* que permite combinar diferentes criterios de optimización sin perjudicar la robustez, precisión y velocidad de convergencia del algoritmo y además de mantener una codificación sencilla de las soluciones.

Capítulo 3

DIFFERENTIAL EVOLUTION

3.1. Introducción

El método *Differential Evolution* (DE) es una herramienta potente, rápida, fiable y fácil de usar para hacer frente a problemas difíciles de optimización global. Desde su creación en 1995, la reputación del DE como un algoritmo de optimización efectivo ha aumentado y diferentes variantes del DE han sido propuestas [18].

Los orígenes del DE se encuentran en la modificación de un método de optimización combinatorio basado en poblaciones llamado *Genetic Annealing* (Recocido Genético) propuesto por Kenneth Price [29]. En un primer momento DE fue desarrollado entorno a la idea de encontrar un método que resolviera el problema de ajuste del polinomio de Tchebychev en pocas iteraciones. La modificación del algoritmo *Genetic Annealing* utilizando la codificación de punto flotante y las operaciones aritméticas, así como la introducción del operador de mutación diferencial dio lugar a la primera variante de DE.

DE fue presentado por Rainer Storn y Kenneth Price en 1995 [35] seguido de [36][34][37] y fue bien recibido en el *First International Contest on Evolutionary Optimization*, donde obtuvo el tercer puesto como mejor método de optimización. Más aclaraciones sobre el funcionamiento interno del DE se hacía en 2002 [46] con un artículo sobre el análisis de los valores críticos de los parámetros de control del DE, donde se trata el problema de convergencia prematura común en las estrategias de evolución, y la selección de parámetros de control que impidan esta convergencia prematura o el estancamiento (*stagnation*).

3.2. Differential Evolution (DE)

DE es un algoritmo evolutivo de búsqueda directa heurística de optimización global, basado en poblaciones de soluciones candidatas, capaz de manejar funciones objetivo no diferenciables, no lineales, y multimodales [31].

La optimización es un concepto abstracto que describe el proceso de búsqueda de la mejor solución, que puede ser un máximo o un mínimo, a un problema entre un conjunto de alternativas. Desde un punto de vista matemático el problema de minimización puede ser formulado de la siguiente manera:

$$\min f(\vec{x}), \text{ sujeto a } \vec{x} \in C \tag{3.1}$$

donde f se conoce como la función objetivo y describe la característica a optimizar del proceso, $\vec{x}$ es un vector de parámetros de entrada conocido como vector objetivo, y C es el subconjunto del espacio de búsqueda factible definido por el conjunto de restricciones del problema. Por lo tanto, el problema de optimización consiste en encontrar un vector de prueba $\vec{x}$ para el cual la función objetivo f es óptima.

El proceso de optimización del algoritmo DE se realiza por medio de tres operaciones principales: mutación, cruce y selección. La población del algoritmo se inicializa aleatoriamente dentro de los límites del espacio de búsqueda del problema. En cada generación, cada individuo de la población actual se convierte en un vector objetivo. Para cada vector objetivo la operación de mutación produce un vector mutante, como resultado de agregar la diferencia ponderada entre dos vectores a un tercer vector. Seguidamente la operación de cruce genera un nuevo vector, llamado vector de prueba, mediante la mezcla de los parámetros del vector mutante con los de el vector objetivo correspondiente. Sabiendo que el valor de aptitud (*fitness*) es el valor que se asigna a cada individuo y que indica qué tan bueno es éste con respecto a los demás para la solución del problema, tenemos que, si el vector de prueba obtiene un valor de aptitud (*fitness*) menor que el vector objetivo, entonces el vector de prueba sustituye al vector objetivo en la próxima generación. En general, la version original del DE puede definirse en base a los conceptos que se describen a continuación.

3.2.1. La Población

Sea $P(\vec{x}, g)$ una población de NP soluciones candidatas en la generación g,

$$P(\vec{x}, g) = (\vec{x}_{1,g}, \ldots, \vec{x}_{i,g}, \ldots, \vec{x}_{NP,g}) \tag{3.2}$$

donde $\vec{x}_{i,g}$ es un vector de D parámetros de entrada definido como,

$$\vec{x}_{i,g} = (x_{1,i}, \ldots, x_{j,i}, \ldots, x_{D,i}) \tag{3.3}$$

El tamaño de la población permanece fijo durante el proceso evolutivo de optimización. Este parámetro es definido por el usuario, y está directamente relacionado a la dimensión del problema.

3.2.2. Inicialización de la Población

Como primer paso en el algoritmo, se generan aleatoriamente NP individuos que conforman la población inicial $P(\vec{x}, 0) = (\vec{x}_{1,0}, \ldots, \vec{x}_{i,0}, \ldots, \vec{x}_{NP,0})$. En la mayoría de los casos los individuos están uniformemente distribuidos en el espacio de búsqueda dentro de los límites definidos por el problema, como se muestra en la siguiente ecuación,

$$\vec{x}_{i,0} = \{x_{j,i}\} = x_j^- + rand_j(0,1)(x_j^+ - x_j^-), j = 1, \ldots, D \tag{3.4}$$

donde $rand_j(0,1)$ es un número aleatorio generado para cada parámetro del vector en el intervalo $[0,1]$ y los superíndices - y $+$ indican los límites inferior y superior del rango de valores de cada parámetro $x_{j,i}$ del vector $\vec{x}_{i,0}$, $i = 1,\ldots,$ NP. Si está disponible una solución aproximada al problema, entonces se pueden generar los individuos iniciales perturbando a la solución dada mediante una función de distribución normal-gaussiana.

3.2.3. Mutación

Conceptualmente, el objetivo de la mutación es generar nueva descendencia. Consiste en formar tantos individuos nuevos como existan en la población actual, esto es, por cada vector de la población se genera un vector mutante. Para generar cada uno de estos individuos se deben seleccionar tres vectores aleatoriamente: $\vec{x}_{r_1}$, $\vec{x}_{r_2}$ y $\vec{x}_{r_3}$, de tal forma que los índices r_1, r_2 y r_3 sean distintos entre sí y distinto al índice i del vector objetivo de la mutación $\vec{v}_{i,g}$. Estos tres vectores servirán para generar el nuevo descendiente,

$$\vec{v}_{i,g} = \vec{x}_{r_1,g} + F(\vec{x}_{r_2,g} - \vec{x}_{r_3,g}) \tag{3.5}$$

donde F es un factor de escala constante y positivo, con valores entre $[0, 1+]$, definido por el usuario. Este factor se encarga de controlar la amplificación del vector diferencia $\vec{x}_{r2,g} - \vec{x}_{r3,g}$ y su rol es fundamental en la convergencia del algoritmo. El vector base x_{r1} puede ser determinado de diferentes formas las cuales se mencionarán mas adelante.

3.2.4. Recombinación Discreta

La mutación es el principal responsable de explorar el espacio de búsqueda. La recombinación discreta o cruce, en cambio, es un proceso complementario que permite el intercambio de información de los vectores a sus descendientes. Se crea un vector de prueba $\vec{u}_{i,g}$ a través de un método de recombinación entre el padre o vector objetivo y el vector de mutación. Inicialmente se propusieron dos procedimientos de recombinación discreta o cruce: binomial y exponencial.

3.2.4.1. Recombinación Binomial

El objetivo de la recombinación binomial consiste en generar una población intermedia, $P(\vec{u}, g)$, donde los vectores $\vec{u}_{i,g}$ son el resultado de la combinación aleatoria de los vectores $\vec{v}_{i,g}$ y $\vec{x}_{i,g}$ según cierta probabilidad CR. Siguiendo el siguiente criterio:

$$\vec{u}_{i,g} = u_{j,i} = \begin{cases} v_{j,i}, & \text{si } rand_j(0,1) \leq CR \text{ ó } j = j_{rand} \\ x_{j,i}, & \text{en otro caso} \end{cases} \tag{3.6}$$

donde CR $\in [0, 1]$ es la probabilidad de cruce. Este valor determina el número de parámetros del vector objetivo y vectores mutantes que se utilizan en la creación de los vectores de prueba. La ecuación (3.6) establece que el cruce binomial consiste en generar un número aleatorio entre cero y uno ($rand_j(0, 1)$) para cada parámetro $u_{j,i}$, si este valor es menor o igual a CR el parámetro se copia del vector $\vec{v}_{i,g}$, en caso contrario el parámetro se hereda del vector $\vec{x}_{i,g}$. La condición $j = j_{rand}$ establece que por lo menos un parámetro de $\vec{u}_{i,g}$ (seleccionado aleatoriamente) se hereda del vector $\vec{v}_{i,g}$, de esta forma se evita que el vector resultante sea una réplica exacta del vector $\vec{x}_{i,g}$.

3.2.4.2. Recombinación Exponencial

Para la recombinación exponencial, el parámetro se hereda del vector mutación $\vec{v}_{i,g}$ hasta que el número aleatorio generado entre cero y uno ($rand_j(0, 1)$) supera el valor de CR, en este momento los parámetros restantes se toman del padre original $\overline{x}_{i,g}$.

$$\vec{u}_{i,g} = u_{j,i} = \begin{cases} v_{j,i}, & \text{mientras } rand_j(0,1) \leq CR \\ x_{j,i}, & \text{desde } rand_j(0,1) > CR \end{cases} \tag{3.7}$$

3.2.5. Selección

El mecanismo de selección DE es el encargado de determinar cuales vectores de la población $P(\vec{x}, g)$ son reemplazados en la siguiente generación por los vectores de la población intermedia $P(\vec{u}, g)$. La selección consiste en la comparación uno a uno del valor de aptitud (*fitness*) para cada uno de los vectores $\vec{x}_{i,g}$ y $\vec{u}_{i,g}$, para i = 1,2,. . . , NP, siguiendo el siguiente criterio:

$$\vec{x}_{i,g+1} = x_{j,i} = \begin{cases} \vec{u}_{i,g}, & \text{si } f(\vec{u}_{i,g}) \leq f(\vec{x}_{i,g}) \\ \vec{x}_{i,g}, & \text{en otro caso} \end{cases} \tag{3.8}$$

Una vez completa la selección de la población $P(\vec{x}, g + 1)$ el algorimto repite el ciclo de mutación, recombinación y selección hasta que se alcanza algún criterio de terminación adecuado al problema, por ejemplo un número de iteraciones máximo, un umbral mínimo de optimización, etc.

En el Algoritmo 1 se resume la estructura en pseudocódigo del algoritmo DE.

Algorithm 1 Algoritmo Differential Evolution

Require: $D, g_{max}, NP, F \in (0, 1+), CR \in [1, 0], x_j^+ y x_j^-$.

1: **for** $i \leq NP \wedge j \leq D$ **do**
2: $x_{j,i} = x_j^- + rand_j(0,1)(x_j^+ - x_j^-)$
3: **end for**
4: **while** $g \leq g_{max}$ **do**
5: **for** $i \leq NP$ **do**
6: $r_1, r_2, r_3 = round(rand(1, NP))$
7: $j_{rand} = round(rand(1, D))$
8: **for** $j \leq D$ **do**
9: **if** $rand(0,1) \leq CR \vee j = j_rand$ **then**
10: $u_{j,i} = x_{j,r_1} + F(x_{j,r_2} - x_{j,r_3})$
11: **else**
12: $u_{j,i} = x_{j,i}$
13: **end if**
14: **end for**
15: **if** $f(\vec{u}_{i,g}) \leq f(\vec{x}_{i,g})$ **then**
16: $\vec{x}_{i,g+1} = \vec{u}_{i,g}$
17: **else**
18: $\vec{x}_{i,g+1} = \vec{x}_{i,g}$
19: **end if**
20: $g = g + 1$
21: **end for**
22: **end while**

3.3. Parámetros de Control DE

El tamaño de la población NP, el factor de escala F y la probabilidad de cruce CR, son los parámetros de control del algoritmo DE. Estos parámetros son definidos por el usuario y permanecen constantes durante la ejecución del algoritmo. A continuación se describen algunos criterios para fijar estos valores correctamente.

3.3.1. Tamaño de la población (NP)

En general, el tamaño de la población depende de la complejidad y del número de parámetros D del vector solución. Un valor pequeño de NP favorece a la velocidad de convergencia del algoritmo; sin embargo, puede ocasionar convergencia prematura no-óptima ó estancamiento en la evolución de la población (*stagnation*). Valores más altos de NP evitan la convergencia prematura pero comprometen el tiempo de cálculo empleado por el algoritmo.

El valor de NP puede variar entre 2D y 100D, aunque NP = 20D resulta un valor adecuado en la mayoría de los casos según [30]. En el estudio [35] se considera apropiados valores de NP: 3D≤NP≤10D, y en [5] de 5D≤NP≤10D.

En su versión original, DE emplea una población de tamaño fijo. Aunque existe otra propuesta donde el tamaño de la población se codifica como un parámetro de diseño, es decir, que la población aumenta o disminuye dinámicamente [39].

3.3.2. Factor de escala (F)

La selección del factor de escala F ha sido objeto de estudio en [46] donde se recomienda el uso de un valor de F dentro de un umbral establecido por unos límites inferior y superior. Este estudio proporciona un criterio para calcular de manera efectiva el límite inferior F_{crit},

$$F_{crit} = \sqrt{\frac{1 - \frac{CR}{2}}{NP}} \tag{3.9}$$

aunque los resultados obtenidos en [46] indican que valores inferiores a 0,3 hacen converger prematuramente al algoritmo. Por otro lado, F = 1 no da buenos resultados, puesto que la variación sobre el vector base no es distinguible. En [34] se indica que es recomendable escoger valores de F entre 0,5 y 0,95.

3.3.3. Probabilidad de cruce (CR)

La probabilidad de cruce CR∈ $[0, 1]$ determina la diversidad de la población en las generaciones sucesivas. Un valor pequeño de CR significa que en proporción la población intermedia $P(\vec{u}, g)$ hereda un número menor de parámetros provenientes de la mutación. Por otro lado, valores de CR cercanos a uno transforman al algoritmo en un esquema de sólo mutación. Finalmente, debe tomarse en cuenta el tipo de función objetivo a optimizar, por ejemplo: en funciones separables

un valor de $0 \leq CR \leq 0{,}2$ resulta adecuado y para funciones no separables y con múltiples puntos óptimos es conveniente fijar el valor entre $0{,}9 \leq CR \leq 1$ [31].

3.4. Estrategias DE

La nomenclatura utilizada para identificar las diferentes estrategias que pueden adoptarse en el algoritmo del *Differential Evolution* es la siguiente: DE/Vector a Perturbar/Número de Vectores/Tipo de Recombinación.

• Vector a Perturbar: representa el mecanismo de selección del vector base ($\vec{x}_{(r1,g)}$). Por lo tanto, *rand*: indica la selección aleatoria del vector base para ser perturbado. *best*: utliza como vector base al mejor individuo de la población actual, $\vec{x}_{(best,g)}$. Fianlamente *rand-to-best*: calcula la diferencia entre el miembro del i-ésimo y el mejor hasta ahora miembro de la población actual.

• Número de Vectores: representa el número de pares de vectores que forman la diferencia.

• Recombinación: indica el método de recombinación discreta utilizado: binomial (bin) ó exponencial (exp).

La estrategia de mutación no depende de la estrategia de cruce, por tanto para cada estrategia de mutación puede darse el cruce exponecial como el cruce binomial. A continuación se muestran diferentes estrategias de mutación que fueron sugeridas por Storn y Price:

• DE/rand/1: $\vec{v}_{i,g} = \vec{x}_{r1,g} + F(\vec{x}_{r2,g} - \vec{x}_{r3,g})$

• DE/best/1: $\vec{v}_{i,g} = \vec{x}_{best,g} + F(\vec{x}_{r2,g} - \vec{x}_{r3,g})$

• DE/best/2: $\vec{v}_{i,g} = \vec{x}_{best,g} + F(\vec{x}_{r2,g} + \vec{x}_{r3,g} - \vec{x}_{r4,g} - \vec{x}_{r5,g})$

• DE/rand/2: $\vec{v}_{i,g} = \vec{x}_{r1,g} + F(\vec{x}_{r2,g} + \vec{x}_{r3,g} - \vec{x}_{r4,g} - \vec{x}_{r5,g})$

• DE/rand-to-best/1: $\vec{v}_{i,g} = \vec{x}_{i,g} + F(\vec{x}_{best,g} - \vec{x}_{i,g}) + F(\vec{x}_{r2,g} - \vec{x}_{r3,g})$

De tal forma, una estrategia que funcione mejor para resolver un problema puede no ser adecuada cuando se aplica para otro problema distinto. Así también, los parámetros que se adoptan para un problema tienen que ser determinados mediante experimentación a partir de los criterios mencionados anteriormente. Sin embargo, la estrategia DE/rand/1/bin, denominada DE clásico, es la más comúnmente utilizada.

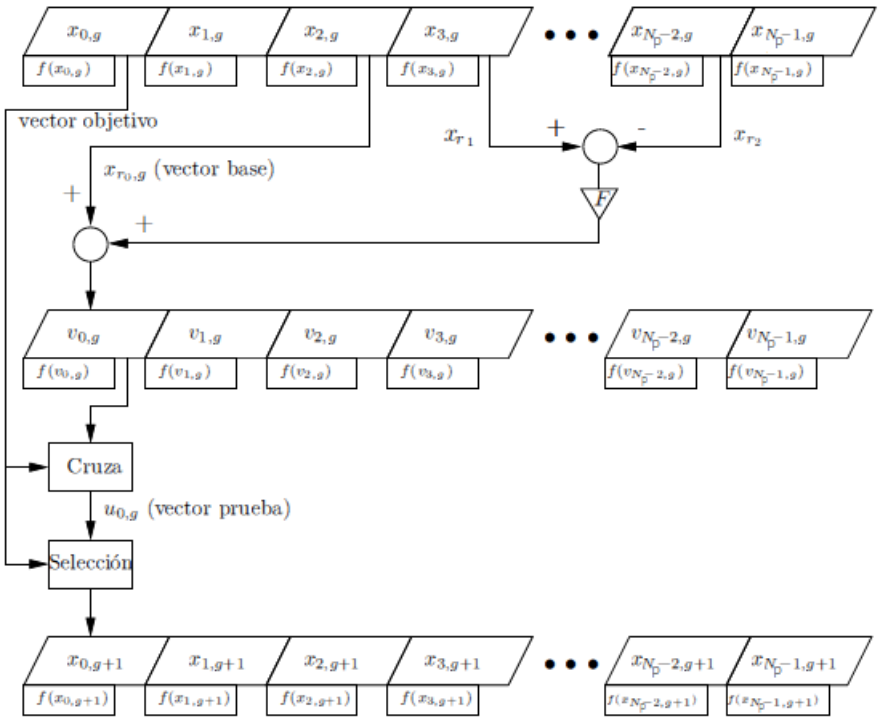


Figura 3.1: Diagrama de flujo del *Differential Evolution* [41].

3.5. Mecanismo *discarding*

El mecanismo *discarding* (dDE) es implementado en el *Differential Evolution* con el objetivo de acelerar la convergencia del algoritmo para obtener un resultado de forma más rápida [23]. De este modo el proceso de optimización del algoritmo DE se realiza por medio de cinco operaciones principales: mutación, cruce, selección, *discarding* y selección. El *discarding* está basado en la idea de reemplazar un pequeño porcentaje de los peores elementos de la población actual por unos nuevos más cercanos a las mejores soluciones obtenidas. Para evitar la convergencia prematura, cada individuo descartado es remplazado por una solución candidata seleccionada aleatoriamente entre los individuos más aptos de la población actual mas un valor de perturbación aleatorio.

Se considera una población $P(\vec{x}, g)$ de NP soluciones de candidatos ordenadas de acuerdo con su valor de aptitud en orden creciente, $\overline{P} = (\vec{x}_1, \ldots, \vec{x}_i, \ldots, \vec{x}_{NP})$, tal que, $f(\vec{x}_i) \leq f(\vec{x}_{i+1})$.

Se crea una subpoblación $W \subset P$ que contiene un tamaño δ de los peores elementos de la población actual:

$$\overline{W} = (x_1, \ldots, x_i, \ldots, x_\delta), \tag{3.10}$$

y también se crea una subpoblación $S \subset \overline{P}$ con un número β (Tamaño de la población de reemplazo) de los mejores elementos de la población actual:

$$\overline{S} = (x_1, \ldots, x_i, \ldots, x_\beta), \tag{3.11}$$

tal que $\beta \gg \delta$ y donde $\delta = \dfrac{NP * DRATE}{100}$, siendo el DRATE un valor entre 0 y 100, elegido por el usuario. Este valor representa el porcentaje de la población que será sustituido por el mecanismo *discarding*.

Por tanto se tiene una población de prueba de tamaño δ, $\overline{Q}$, que se construye con las soluciones de los candidatos elegidos al azar entre los elementos de $\overline{S}$. Puesto que la diversidad debe ser preservada con el fin de explorar de forma eficiente el espacio de búsqueda, una perturbación de ruido aleatorio α, siendo $\alpha = \sigma(\lambda)$, se le añade a las soluciones de los nuevos candidatos de la siguiente manera:

$$\overline{Q} = (x_1, \ldots, x_i, \ldots, x_\delta), \text{ siendo } x_i = x_s + \alpha \tag{3.12}$$

donde λ es un valor de distribución Gaussiana y el índice s es un número aleatorio uniforme designado dentro de $[1, \beta]$. Finalmente, los elementos en $\overline{W}$ son descartados y reemplazados en la población si, en comparación, el valor de costo del elemento correspondiente en $\overline{Q}$ es inferior.

Dos condiciones deben cumplirse para que el mecanismo *discarding* funcione correctamente. En primer lugar, el tamaño de la población tiene que ser lo suficientemente grande como para asegurar la diversidad pero no demasiado grande para aumentar el coste computacional. En segundo lugar, el valor δ tiene que establecerse en un pequeño porcentaje de la población; de lo contrario, se corre el riesgo de quedar atrapado en un falso valor óptimo, especialmente en entornos ruidosos. En el Algoritmo 2 se resume la estructura en pseudocódigo del mecanismo *discarding*.

Algorithm 2 Mecanismo *discarding*

Require: $DRATE, \alpha = \sigma(\lambda), \beta$.

1: $P = \{\vec{x}_1, \ldots, \vec{x}_i, \ldots, \vec{x}_{N_P}\}$, de manera que $f(\vec{x}_i) \leq f(\vec{x}_{i+1})$
2: **for all** $i \in [N_P - \delta + 1, N_P]$ **do**
3: $s = rand[1, \beta]$
4: $\vec{Q} = \vec{x}_s + \alpha$
5: **if** $f(\vec{Q}) < f(\vec{W})$ **then**
6: $\vec{x}_i = \vec{Q}$
7: **end if**
8: **end for**

3.5.1. Mecanismo *discarding* adaptativo

En este proyecto se presenta una idea de modificación del Mecanismo *discarding*, denominado Mecanismo *discarding* adaptativo (dADE). Del mismo modo que el mecanismo *discarding*, el dADE es implementado en el algoritmo DE con el objetivo de acelerar la convergencia del algoritmo para obtener un resultado de forma más rápida, pero en este caso, de manera más controlada.

El funcionamiento del mecanismo dADE es el siguiente: Inicialmente se ejecuta el mecanismo dDE, de modo que, las primeras generaciones convergen de forma más rápida. En el momento que el mejor valor de aptitud (*best fitness*) alcanza un valor mínimo determinado por el usuario, entonces, las generaciones restantes convergerán bajo el algoritmo DE clásico.

Capítulo 4

RESULTADOS EXPERIMENTALES

4.1. Introducción

En este proyecto se resuelve la cinemática inversa de robots manipuladores de diferentes grados de libertad. Para ello se plantea la cinemática inversa como un problema de optimización con restricciones, de manera que, conocida la posición y orientación deseada en el extremo del robot manipulador y su función de cinemática directa, se calcula un vector de coordenadas articulares óptimo que minimice la función de error absoluto de posición y orientación del extremo. Además, la solución óptima debe encontrarse dentro de los límites articulares del robot. El método utilizado para resolver este problema de optimización es el *Differential Evolution* que permite combinar diferentes criterios de optimización sin perjudicar la robustez, precisión y velocidad de convergencia del algoritmo y además de mantener una codificación sencilla de las soluciones.

Basándose en la experimentación empírica, este estudio pretende mostrar la influencia de los valores de los parámetros de control del algoritmo DE(tamaño de la población, factor de escala de mutación y probabilidad de cruce) en el comportamiento del algoritmo para el problema de inversión cinemática. También, examina la introducción de un operador de búsqueda local para mejorar la velocidad de convergencia del algoritmo. Los aspectos que se han tenido en consideración en la evaluación del algoritmo han sido los errores de posición y orientación finales del extremo del robot con respecto a los valores deseados.

4.2. Plataformas Experimentales

A continuación se describen las plataformas robóticas consideradas para la ejecución de las pruebas. Se han seleccionado dos robots con 3 y 6 grados de libertad (GDL) respectivamente. El movimiento del robot está restringido al espacio Cartesiano bidimensional, en el caso del robot 3 GDL, y tridimensional en el caso del robot de 6 GDL.

4.2.1. Robot manipulador de 3 GDL

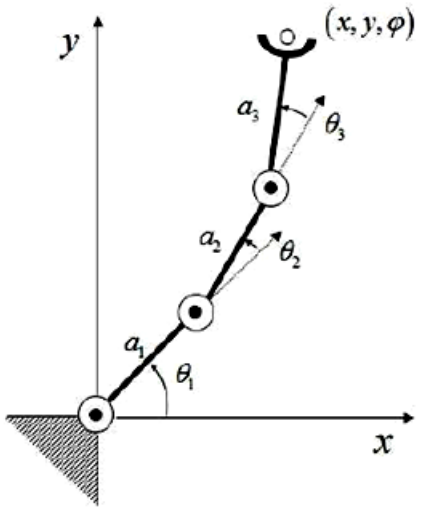


Figura 4.1: Robot manipulador plano de 3 GDL [40].

En la Figura 4.1 se muestra un brazo plano de tres grados de libertad (D=3). Como las tres articulaciones son rotacionales, a este manipulador se le conoce como mecanismo RRR ó 3R. El movimiento del extremo está restringido al espacio Cartesiano bidimensional, descrito por las coordenadas de posición *(x,y)*, y la orientación φ respecto al eje horizontal. En la Tabla 4.1 se muestran sus parámetros de Denavit-Hartenberg y sus límites articulares,

Tabla 4.1: Parámetros D-H de un robot manipulador plano de 3 GDL.

Articulación	a_j	α_j	d_j	θ_j	Limites articulares
1	a_1	0	0	θ_1	-90 - +90
2	a_2	0	0	θ_2	-90 - +90
3	a_3	0	0	θ_3	-90 - +90

Los valores de a_1, a_2 y a_3 son iguales a 0,2 m, 0,15 m y 0,1 m, respectivamente [40].

A partir de los parámetros cinemáticos D-H se puede obtener la posición y orientación del elemento final a partir de las ecuaciones de cinemática directa del robot manipulador,

$x = a_1 cos(\theta_1) + a_2 cos(\theta_1 + \theta_2) + a_3 cos(\theta_1 + \theta_2 + \theta_3)$

$y = a_1 sin(\theta_1) + a_2 sin(\theta_1 + \theta_2) + a_3 sin(\theta_1 + \theta_2 + \theta_3)$

$\varphi = \theta_1 + \theta_2 + \theta_3$

Estas ecuaciones son utilizadas en el algoritmo de inversión cinemática propuesto.

4.2.2. Robot manipulador de 6 GDL

Figura 4.2: Robot manipulador Puma560 de 6GDL.

El robot Puma 560 es un brazo articulado con 6 articulaciones rotatorias que le proporcionan 6 grados de libertad (D=6) y le permiten posicionar y orientar su herramienta final en el espacio Cartesiano tridimensional. Las 3 primeras articulaciones (hombro-codo-muñeca) posicionan el objeto en el espacio y el grupo formado por las tres últimas orientan el elemento final del robot. En el espacio Cartesiano esto corresponde a tres coordenadas de posición (x,y,z) y tres coordenadas de orientación en el extremo, representadas, en este caso, por una terna de ángulos de Euler ZYZ (ϕ, θ, ψ).

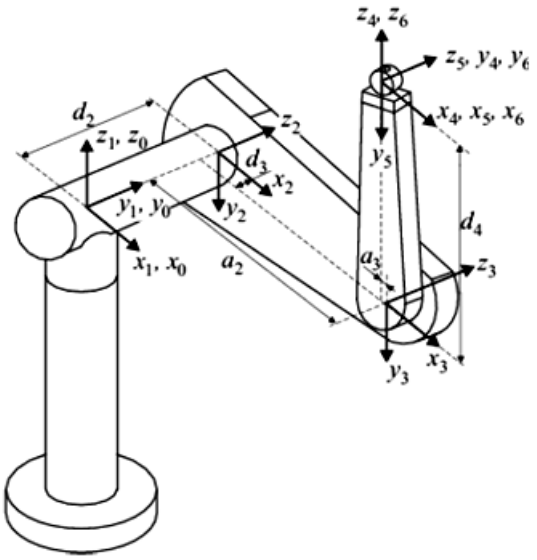


Figura 4.3: Asignación de los sistemas de referencia y parámetros DH del robot Puma 560 [40].

En la Figura 4.3 se muestra la asignación de los sistemas de referencia y los parámetros de Denavit-Hartenberg (DH) del robot manipulador Puma 560. Un desarrollo más completo del procedimiento y notación DH se encuentra en [33]. En la Tabla 4.2 se muestran sus parámetros de Denavit-Hartenberg y sus límites articulares,

Tabla 4.2: Parámetros cinemáticos D-H de un robot manipulador plano de 6GDL.

Articulación	a_j	α_j	d_j	θ_j	Límites articulares
1	0	90°	0	θ_1	-160° - +160°
2	a_2	0°	0	θ_2	-225° - +45°
3	a_3	-90°	d_3	θ_3	-45° - +225°
4	0	90°	d_4	θ_4	-110° - +170°
5	0,	-90°	0	θ_5	-100° - +100°
6	0	0°	0	θ_6	-266° - +266°

Los valores de a_2 y a_3 son iguales a 0,4318 m y 0,0191 m, y los valores de d_3 y d_4 son iguales a 0,1254 m y 0,4318 m, respectivamente [40].

La cinemática directa del robot Puma 560 se muestra en [27], donde la posición y orientación del manipulador son dados por el producto de las matrices de transformación homogénea $T_j^{(j-1)}$

de cada eslabón.

4.3. Descripción de los experimentos

A continuación se describe detalladamente el diseño experimental implementado para evaluar al algoritmo DE como método de inversión cinemática. La finalidad de estos experimentos es la de evaluar la robustez, precisión, y velocidad de convergencia del algoritmo DE en términos del error de posición y orientación del extremo del robot, tasa de acierto, y número de iteraciones. Los experimentos han sido implementados en el entorno de simulación de Matlab® .

- Criterios de Parada.
  - CP1: Número de iteraciones. La ejecución del algoritmo concluye cuando se alcanzan un número máximo de iteraciones.

Tabla 4.3: Número de iteraciones para cada robot.

Robot	3 GDL	6 GDL
Nº Iteraciones	100	200

  - CP2: Umbral de error. La ejecución del algoritmo concluye cuando se alcanza un umbral de error de posición y orientación predefinido.
- Población inicial.

  Para cada conjunto de pruebas se ha considerado la misma población inicial. De este modo cada configuración del algoritmo es evaluado bajo las mismas condiciones iniciales.
- Umbrales de Error.

  Se han establecido un conjunto de umbrales de error de posición y orientación con la finalidad de evaluar las distintas etapas del proceso evolutivo de la población. Los valores de umbral utilizados se muestran en la Tabla 4.4.

Tabla 4.4: Umbrales de error de posición y de orientación.

Umbral	E1	E2	E3
Ep (mm)	0,002	0,02	0,2
Eo (º)	0,0005	0,005	0,05

- Parámetros de control DE.

  Se seleccionan unos parámetros de control base, que serán los que se utilizarán en caso de no estar evaluando dicho parámetro. Se han escogido los considerados óptimos según el sección 3.2 de este proyecto.

Tabla 4.5: Parámetros de control DE.

Parámetro	D	NP	F	CR
3 GDL	3	30	0,5	0,9
6 GDL	6	60	0,5	0,9

- Constantes de normalización.

  Las constantes de normalización utilizadas dependerán de los grados de libertad del robot. Para robots de 3 GDL las constantes de normalización utilizadas son las propuestas por [32], en cambio, para robots manipuladores de 6 GDL las constantes de normalización utilizadas son las propuestas en [38]. Las constantes de normalización utilizadas se muestran en la Tabla 4.6.

Tabla 4.6: Parámetros cinemáticos D-H de un robot manipulador plano de 6GDL.

Constantes	w_p	w_o
3 GDL	$\frac{1}{\Sigma a_i}$	$\frac{1}{2\pi}$
6 GDL	1	$\frac{\sqrt{\lambda_{KC} \parallel Pd \parallel}}{2\pi}$

El valor Σa_i corresponde al alcance máximo del robot manipulador, λ_{KC} es un coeficiente que depende de las dimensiones del robot manipulador y $\| Pd \|$ es la distancia del punto de trabajo desde la base del robot.

- Coordenadas Cartesianas deseadas.

  Las pruebas son realizadas para un único punto predefinido en el espacio de trabajo de cada robot manipulador. Las coordenadas Cartesianas utilizadas se muestran en la Tabla 4.7 para un robot de 3 GDL y en la Tabla 4.8 para un robot de 6 GDL.

Tabla 4.7: Coordenadas para un robot de 3 GDL.

Coordenadas	x (m)	y (m)	φ (º)
3 GDL	0,083	0,422	100

x e y corresponden a las coordenadas de posición y φ es la coordenada de orientación respecto al eje horizontal

Tabla 4.8: Coordenadas para un robot de 6 GDL.

Coordenadas	x (m)	y (m)	z (m)	ϕ (º)	θ (º)	ψ (º)
6 GDL	0,112	-0,058	0,241	-48,67	76,41	-63,83

x (m), y (m) y z (m) corresponden a tres coordenadas de posición y $\phi, \theta y \psi$ a tres coordenadas de orientación en el extremo, representadas por una terna de ángulos de Euler ZYZ.

- Criterios de Evaluación.

  A continuación se describen los criterios que se han tenido en cuenta a la hora de evaluar el algoritmo.

  - Tasa de Acierto (%): porcentaje de acierto del algoritmo DE para 100 pruebas. La tasa de acierto se trata de la probabilidad que tiene el algoritmo de conseguir que los errores de posición y orientación finales del extremo del robot, respecto a unas

coordenadas Cartesianas deseadas, sean menores que los umbrales de error definidos (E1, E2 y E3), considerando así la solución obtenida como adecuada. Se definen tres tasas de acierto distintas (Tasa 1, Tasa 2 y Tasa 3), uno correspondiente a cada umbral de error (E1, E2 y E3).

- Errores ($M_e \pm STD_e$): se calcula el error medio de posición, de orientación, y error total. M corresponde al error medio final, para 100 pruebas, del extremo del robot con respecto a unas coordenadas Cartesianas deseadas. STD corresponde a la desviación estándar del error para las 100 pruebas.
- Nº de iteraciones ($M_i \pm STD_i$). Número de iteraciones necesarias para que el algoritmo converga, de modo que, se obtengan unos errores de posición y orientación finales del extremo del robot menores a unos umbrales de error definidos (E1, E2 y E3). Se definen tres resultados de Nº de iteraciones distintos (Nº de iteraciones 1, Nº de iteraciones 2 y Nº de iteraciones 3), uno correspondiente a cada umbral de error (E1, E2 y E3).

4.4. Parámetros de control de *Differential Evolution*

Differential Evolution posee pocos parámetros de control que permanecen constantes durante la ejecución del algoritmo, estos son: el tamaño de la población (NP), el factor de mutación (F) y la probabilidad de cruce (CR).

En este apartado se muestran los resultados obtenidos en las pruebas realizadas para conocer la influencia de los parámetros de control en la convergencia del algoritmo para la resolución del problema cinemático inverso de robots manipuladores de 3 GDL y de 6 GDL.

4.4.1. Tamaño de la Población (NP)

En general, el tamaño de la población depende de la complejidad y del número de parámetros D del vector solución. Un valor pequeño de NP favorece a la velocidad de convergencia del algoritmo; sin embargo, puede ocasionar convergencia prematura no-óptima ó estancamiento en la evolución de la población (*stagnation*). Valores más altos de NP evitan la convergencia prematura pero comprometen el tiempo de cálculo empleado por el algoritmo. A continuación se muestran los resultados obtenidos de las pruebas realizadas para determinar la influencia del NP en el comportamiento del algoritmo DE.

4.4.1.1. Robot de 3 GDL

En la tabla 4.9 se muestra las tasas de acierto (%) obtenidas, así como los errores medios finales de posición (Ep), orientación (Eo) y total (Et), con sus respectivas desviaciones estándar (error ± desviación estándar) calculados a partir de 100 pruebas del algoritmo. El tiempo mostrado hace referencia al tiempo medio de ejecución empleado por el algoritmo en cada repetición.

Tabla 4.9: Errores finales y tasas de acierto de un robot de 3 GDL para distintos valores de NP.

NP	15	30	60	120
Tasa 1 (%)	90	69	51	40
Tasa 2 (%)	94	97	95	95
Tasa 3 (%)	98	100	100	100
E_P (mm)	1,9E-2 ± 1,2E-1	2,1E-3 ± 4E-3	2,9E-3 ± 3,7E-7	2,9E-3 ± 3,6E-3
E_O (º)	1,4E-1 ± 1,2	7,3E-4 ± 1,6E-4	1,1E-3 ± 1,9E-3	9,6E-4 ± 1,3E-3
E_T	4,3E-4 ± 3,5E-3	1E-5 ± 1E-5	1E-5 ± 1E-5	1E-5 ± 1E-5
Tiempo (s)	4,5E-2	7,9E-2	14,8E-2	28,7E-2

En la tabla 4.10 se muestra el número medio de iteraciones que necesita el algoritmo para alcanzar los umbrales de error fijados con sus respectivas desviaciones estándar (nº iteraciones ± desviación estándar) calculadas a partir de 100 pruebas del algoritmo.

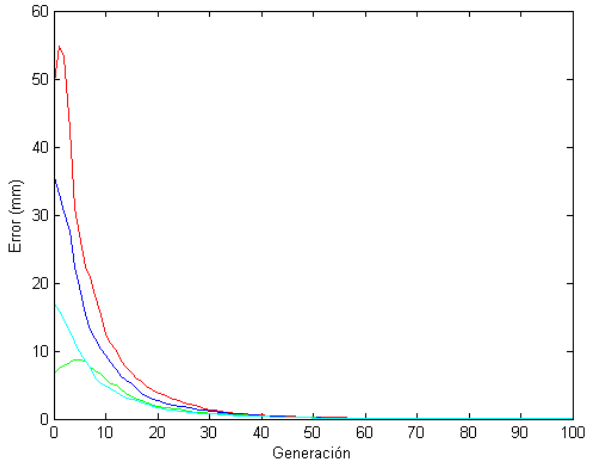
Tabla 4.10: Número de iteraciones de un robot de 3 GDL para distintos valores de NP.

NP	15	30	60	120
Nº Iteraciones 1	88,3 ± 20,3	92 ± 13,8	99,8 ± 14,1	106,4 ± 13,7
Nº Iteraciones 2	74 ± 32,7	71,3 ± 12,2	78,6 ± 11,2	81,8 ± 10,5
Nº Iteraciones 3	56,8 ± 28,5	52,9 ± 8,9	54,7 ± 7,6	55,1 ± 7,7

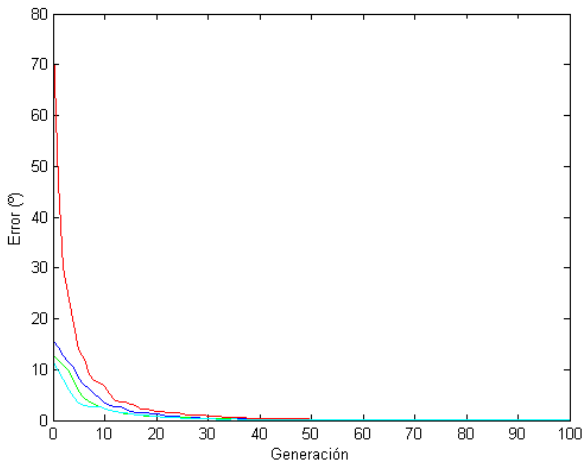
De los resultados de la tabla 4.9 se deduce que el valor más óptimo de NP es 10*D, aunque NP=15 tenga mejores resultados en la tasa de acierto también tiene unos errores medios y unas desviaciones estándar bastante mayores al resto, lo que implica que le genera al algoritmo una menor estabilidad para encontrar la solución óptima. Para NP=60 y NP=120 se obtienen

mayores errores de posición y orientación que para NP=30 además de que su tiempo de ejecución del algoritmo es mayor a medida que aumentamos NP.

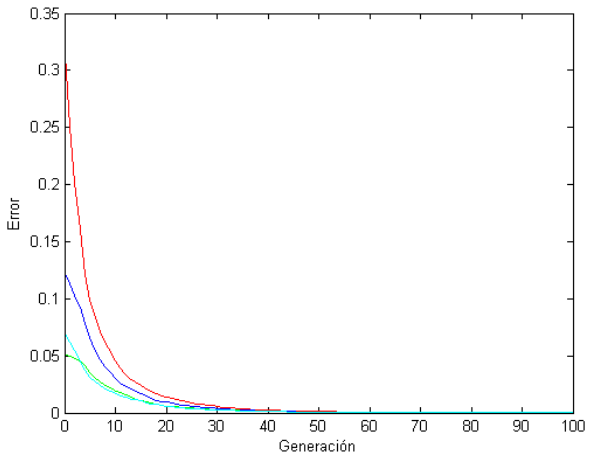
De los resultados de la tabla 4.10 se determina nuevamente que el mejor valor para NP es igual a 10*D. Aquí también se demuestra la poca estabilidad del algoritmo para NP menores de 10*D donde el rango de número de iteraciones en el que se ejecuta el algoritmo para llegar al umbral es muy amplio. A mayor NP menor desviación estándar en las iteraciones, por tanto mayor probabilidad de que el algoritmo converga correctamente. En la figura 4.4 se muestra la evolución de los errores medios durante la ejecución del algoritmo.



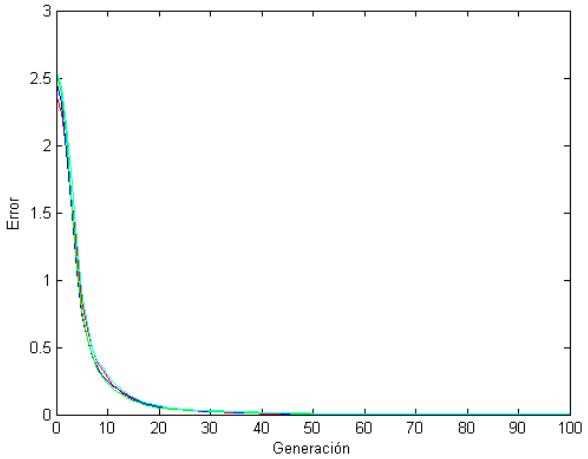
(a) Mejor error medio de posición



(b) Mejor error medio de orientación



(c) Mejor error medio total



(d) Peor error medio total

Figura 4.4: Evolución del error de un robot de 3 GDL para distintos valores de NP: NP=15(rojo), NP=30(azul), NP=60(verde) y NP=120(azul claro).

En la figura 4.4 se observa al algoritmo converger correctamente para los 4 valores de NP.

4.4.1.2. Robot de 6 GDL

En la tabla 4.11 se muestra las tasas de acierto (%) obtenidas, así como los errores medios finales de posición (Ep), orientación (Eo) y total (Et), con sus respectivas desviaciones estándar (error ± desviación estándar) calculados a partir de 100 pruebas del algoritmo. El tiempo mostrado hace referencia al tiempo medio de ejecución empleado por el algoritmo en cada repetición.

Tabla 4.11: Errores finales y tasas de acierto de un robot de 6 GDL para distintos valores de NP.

NP	30	60	120	240
Tasa 1 (%)	20	15	0	1
Tasa 2 (%)	63	74	33	37
Tasa 3 (%)	70	97	85	70
E_P (mm)	2,5 ± 6,8	1,2E-2 ± 2,4E-2	3,6E-2 ± 6,4E-2	1,3E-1 ± 3,1E-1
E_O (º)	7,9E-1 ± 1,9	6,3E-3 ± 1,3E-2	2,1E-2 ± 2,8E-2	5,7E-2 ± 1,3E-1
E_T	3,9E-3 ± 7,4E-3	2E-5 ± 4E-5	7E-5 ± 1E-4	2,4E-4 ± 5,1E-4
Tiempo (s)	0,8	1,6	3,3	6,9

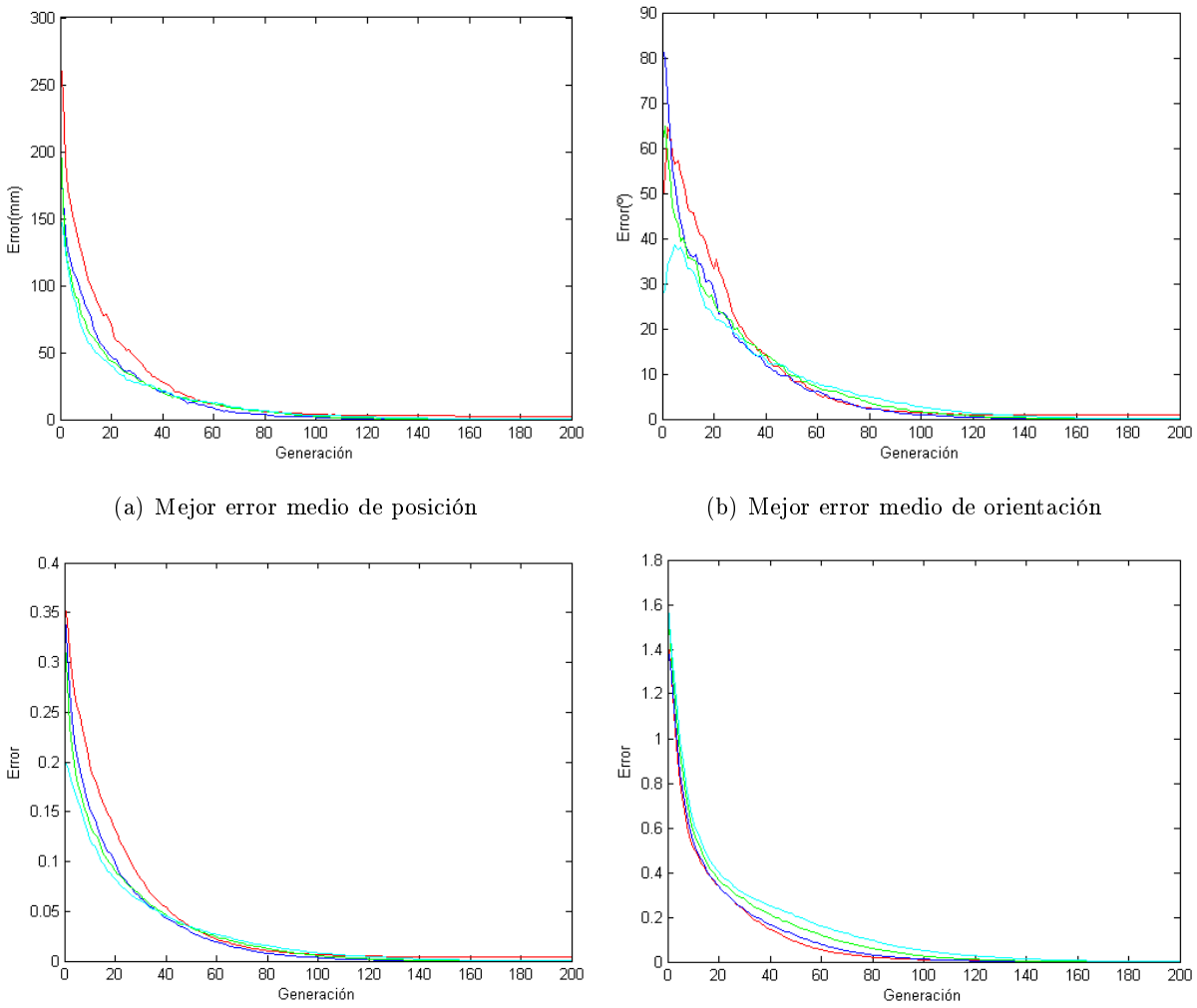
En la tabla 4.12 se muestra el número medio de iteraciones que necesita el algoritmo para alcanzar los umbrales de error fijados con sus respectivas desviaciones estándar (nº iteraciones ± desviación estándar) calculadas a partir de 100 pruebas del algoritmo.

Tabla 4.12: Número de iteraciones de un robot de 6 GDL para distintos valores de NP.

NP	30	60	120	240
Nº Iteración 1	280,2 ± 120,6	232,4 ± 37,9	267,6 ± 43,6	280,6 ± 31,8
Nº Iteración 2	266,3 ± 150,2	209,8 ± 70,9	228,1 ± 55,6	237,3 ± 30,1
Nº Iteración 3	253,9 ± 175,1	159,6 ± 55,5	188,7 ± 76,8	193,3 ± 57,2

Los resultados de la tablas 4.11 y 4.12 muestran como mejor opción 10*D ya que se obtienen menores errores medios de posición y orientación, y se necesitan menos iteraciones para obtener

el resultado deseado, teniendo además menor desviación estándar, lo cual hace al algoritmo más robusto. En la figura 4.5 se muestra la evolución de los errores medios durante la ejecución del algoritmo.



(a) Mejor error medio de posición

(b) Mejor error medio de orientación

(c) Mejor error medio total

(d) Peor error medio total

Figura 4.5: Evolución del error de un robot de 6 GDL para distintos valores de NP: NP=30(rojo), NP=60(azul); NP=120(verde), NP=240(azul claro).

Para todos los valores de NP seleccionados se puede observar en la figura 4.5 que el algoritmo converge correctamente, aunque para NP=30 se aprecia como sobresale sobre las demás obtenido mayores errores en la última iteración, presentado una mayor inestabilidad en los resultados.

4.4.2. Factor de Escala F

El factor de escala ha sido objeto de estudio en [46] y los resultados obtenidos indican que valores inferiores a 0,3 hacen converger prematuramente al algoritmo. Por otro lado, $F = 1$ no da buenos resultados, puesto que la variación sobre el vector base no es distinguible. En [34] se indica que es recomendable escoger valores de F entre 0,5 y 0,95. A continuación se muestran los resultados obtenidos de las pruebas realizadas para determinar la influencia de F en el comportamiento del algoritmo DE.

4.4.2.1. Robot de 3 GDL

En la tabla 4.13 se muestra las tasas de acierto (%) obtenidas, así como los errores medios finales de posición (Ep), orientación (Eo) y total (Et), con sus respectivas desviaciones estándar (error ± desviación estándar) calculados a partir de 100 pruebas del algoritmo. El tiempo mostrado hace referencia al tiempo medio de ejecución empleado por el algoritmo en cada repetición.

Tabla 4.13: Errores finales y tasas de acierto de un robot de 3 GDL para distintos valores de F.

F	0,25	0,5	0,75	1
Tasa 1 (%)	56	71	0	0
Tasa 2 (%)	71	96	21	0
Tasa 3 (%)	84	100	89	17
E_P (mm)	1,1E-1 ± 4,3E-1	2,5E-3 ± 6,5E-3	4,7E-2 ± 4,8E-2	2,5E-1 ± 2E-1
E_O (^{o})	8,9E-2 ± 4,6E-1	7,2E-4 ± 1,5E-3	2E-2 ± 2,4E-2	9,2E-2 ± 8,5E-2
E_T	4,9E-4 ± 1,7E-3	1E-5 ± 2E-5	1,6E-4 ± 1,4E-4	8E-4 ± 4,8E-4
Tiempo (s)	8,1E-2	7,9E-2	7,7E-2	7,7E-2

En la tabla 4.14 se muestra el número medio de iteraciones que necesita el algoritmo para alcanzar los umbrales de error fijados con sus respectivas desviaciones estándar (n^o iteraciones ± desviación estándar) calculadas a partir de 100 pruebas del algoritmo.

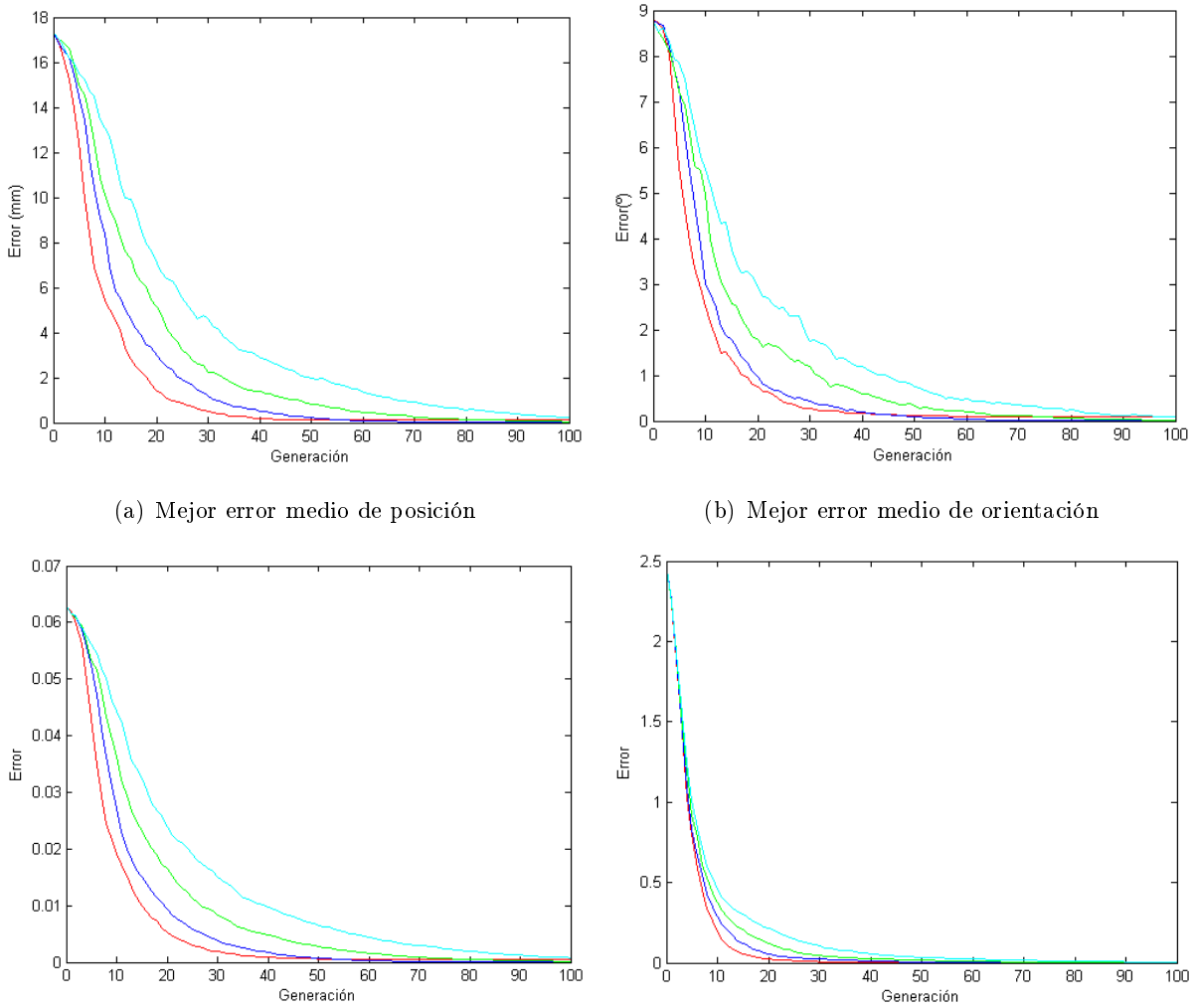
Tabla 4.14: Número de iteraciones de un robot de 3 GDL para distintos valores de F.

F	0,25	0,5	0,75	1
Nº Iteración 1	142,9 ± 92	91,5 ± 12,9	146,7 ± 25,2	223,3 ± 19,4
Nº Iteración 2	106,7 ± 90,3	72,3 ± 11,5	118,7 ± 19	166,9 ± 18,1
Nº Iteración 3	64 ± 72,6	53,1 ± 9,8	83,1 ± 12,9	111,3 ± 14,5

De la tabla 4.13 se obtienen unas conclusiones bastante claras. Para F=0,25 aún teniendo una tasa de acierto aceptable, se observa a partir de las altas desviaciones de error de posición y orientación que el algoritmo en alguna de las 100 repeticiones converge prematuramente y ha aumentado la media de los errores de posición y orientación. Para F=0,75 y F=1 los resultados reflejan que a medida que el valor de F se acerca a 1 no se obtienen buenos resultados. Por tanto, y viendo los bajos valores de error, podemos afirmar que F=0,5 es el valor óptimo para este tipo de problemas de optimización.

La tabla 4.14 contrasta con las gráficas mostradas en la figura 4.6 y los resultados de la tabla 4.13 al necesitar el algoritmo, para F=1, más de 100 iteraciones para poder llegar a los umbrales de error establecidos. De nuevo se obtiene como mejor opción un factor de escala igual a 0,5.

En la figura 4.6 se muestra la evolución de los errores medios durante la ejecución del algoritmo.



(a) Mejor error medio de posición

(b) Mejor error medio de orientación

(c) Mejor error medio total

(d) Peor error medio total

Figura 4.6: Evolución del error de un robot de 3 GDL para distintos valores de F: F=0,25(rojo), F=0,5(azul), F=0,75(verde) y F=1(azul claro).

En la figura 4.6 se aprecia que para F=0,25 en alguna de las repeticiones el algoritmo converge prematuramente y que para F=1 el algoritmo no llega a converger.

4.4.2.2. Robot de 6 GDL

En la tabla 4.15 se muestra las tasas de acierto (%) obtenidas, así como los errores medios finales de posición (Ep), orientación (Eo) y total (Et), con sus respectivas desviaciones estándar (error ± desviación estándar) calculados a partir de 100 pruebas del algoritmo. El tiempo mostrado hace referencia al tiempo medio de ejecución empleado por el algoritmo en cada repetición.

Tabla 4.15: Errores finales y tasas de acierto de un robot de 6 GDL para distintos valores de F.

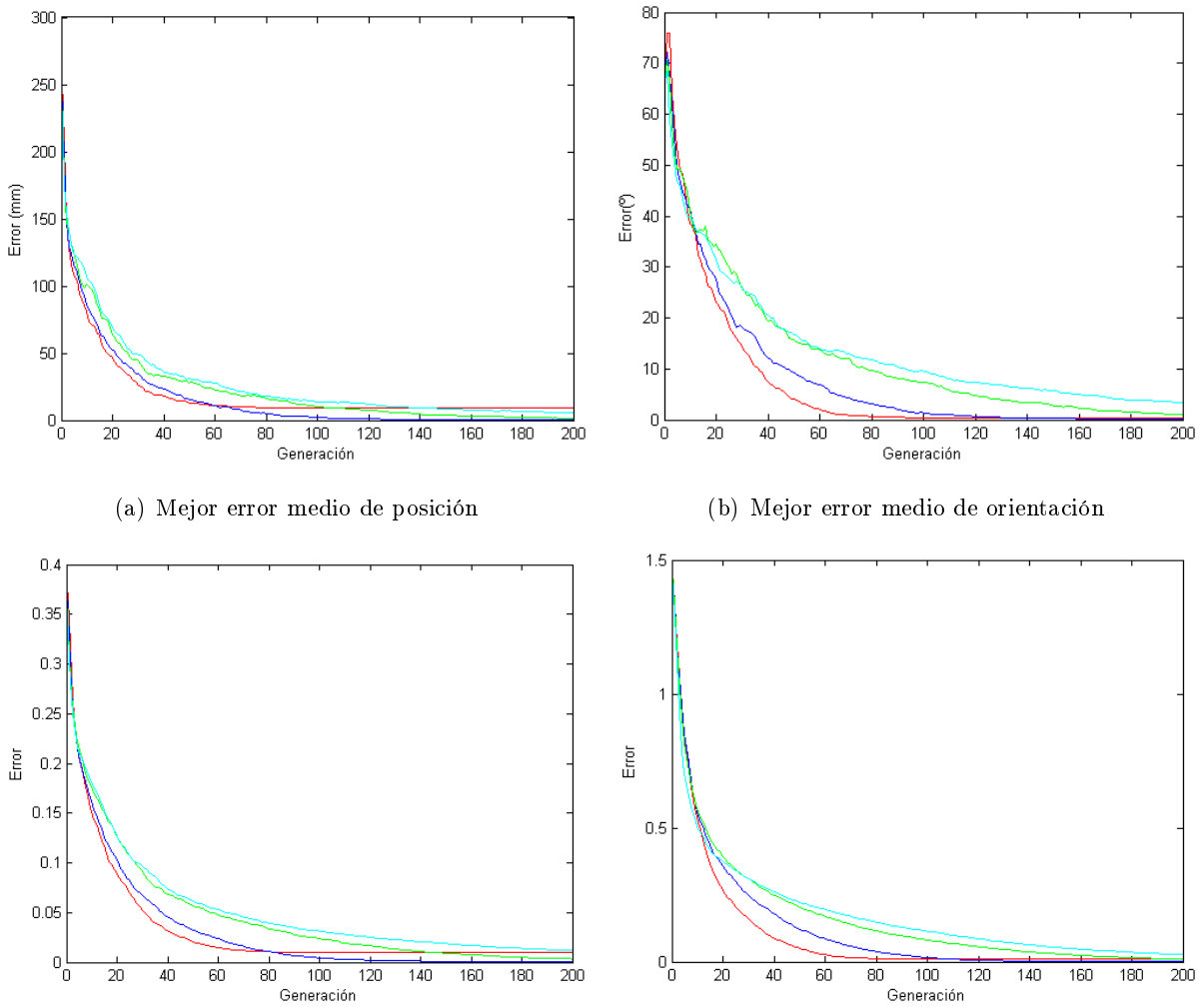
F	0,25	0,5	0,75	1
Tasa 1 (%)	12	6	0	0
Tasa 2 (%)	14	57	0	0
Tasa 3 (%)	25	93	3	0
E_P (mm)	9 ± 10	2,9E-1 ± 2,4	1,5 ± 3,5	5,9 ± 16,1
E_O (º)	3E-1 ± 9,1E-1	1,8E-1 ± 9,5E-1	1,1 ± 1,7	3,3 ± 2,1
E_T	9,6E-3 ± 1E-2	6,1E-4 ± 3E-3	3,5E-3 ± 4,9E-3	1,2E-2 ± 1,6E-2
Tiempo (s)	1,6	1,6	1,7	1,7

En la tabla 4.10 se muestra el número medio de iteraciones que necesita el algoritmo para alcanzar los umbrales de error fijados con sus respectivas desviaciones estándar (nº iteraciones ± desviación estándar) calculadas a partir de 100 pruebas del algoritmo.

Tabla 4.16: Número de iteraciones de un robot de 6 GDL para distintos valores de F.

F	0,25	0,5	0,75	1
Nº Iteración 1	447,5 ± 125,7	237,4 ± 52,8	451 ± 51	500 ± 0
Nº Iteración 2	411,2 ± 163,4	200,8 ± 74,3	395,6 ± 65,9	498,2 ± 12,6
Nº Iteración 3	378,9 ± 186	168,5 ± 81,1	293,6 ± 68,4	481 ± 45,3

Los resultados de la tabla 4.15 y 4.16 reflejan como mejor opción un valor de F igual a 0,5. Para valores de F igual a 0,25 se deduce por las altas desviaciones de error de posición y orientación que el algoritmo converge prematuramente en la mayoría de las repeticiones. Para F=0,75 y F=1 los resultados reflejan que a medida que el valor de F se acerca a 1 no se obtienen buenos resultados, como reflejan las el porcentaje de acierto de ambos valores. En la figura 4.7 se muestra la evolución de los errores medios durante la ejecución del algoritmo.



(a) Mejor error medio de posición

(b) Mejor error medio de orientación

(c) Mejor error medio total

(d) Peor error medio total

Figura 4.7: Evolución del error de un robot de 6 GDL para distintos valores de F: F=0,25(rojo), F=0,5(azul), F=0,75(verde) y F=1(azul claro).

En la figura 4.7 se observa como únicamente converge correctamente el algoritmo para un valor de F=0,5. En F=0,25 el algoritmo converge prematuramente, por lo que sus errores finales son elevados. Y para valores de F igual a 0,75 y 1, el algoritmo no llega a converger para 200 iteraciones.

4.4.3. Probabilidad de Cruce CR

La probabilidad de cruce CR$\in [0,1]$ determina la diversidad de la población en las generaciones sucesivas. Un valor pequeño de CR significa que en proporción la población intermedia

$P(\vec{u}, g)$ hereda un número menor de parámetros provenientes de la mutación. Por otro lado, valores de CR cercanos a uno transforman al algoritmo en un esquema de sólo mutación. Finalmente, debe tomarse en cuenta el tipo de función objetivo a optimizar, por ejemplo: en funciones separables un valor de $0 \leq CR \leq 0{,}2$ resulta adecuado y para funciones no separables y con múltiples puntos óptimos es conveniente fijar el valor entre $0{,}9 \leq CR \leq 1$ [31]. A continuación se muestran los resultados obtenidos de las pruebas realizadas para determinar la influencia de CR en el comportamiento del algoritmo DE.

4.4.3.1. Robot de 3 GDL

En la tabla 4.17 se muestra las tasas de acierto (%) obtenidas, así como los errores medios finales de posición (Ep), orientación (Eo) y total (Et), con sus respectivas desviaciones estándar (error ± desviación estándar) calculados a partir de 100 pruebas del algoritmo. El tiempo mostrado hace referencia al tiempo medio de ejecución empleado por el algoritmo en cada repetición.

Tabla 4.17: Errores finales y tasas de acierto de un robot de 3 GDL para distintos valores de CR.

CR	0,3	0,6	0,9	1
Tasa 1 (%)	0	0	74	98
Tasa 2 (%)	0	0	97	100
Tasa 3 (%)	0	17	100	100
E_P (mm)	3,2± 1,7	3,3E-1 ± 2,8E-1	1,6E-3 ± 3,3E-3	1E-4 ± 4,3E-4
E_O (º)	1,3 ± 1,2	1,E-1 ± 1,8E-1	5,7E-4 ± 1,5E-3	1E-5 ± 4E-5
E_T	1,1E-2 ± 4,6E-3	1,2E-3 ± 8,9E-4	1E-5 ± 1E-5	0 ± 0
Tiempo (s)	8,6E-2	8E-2	8,6E-2	8,9E-2

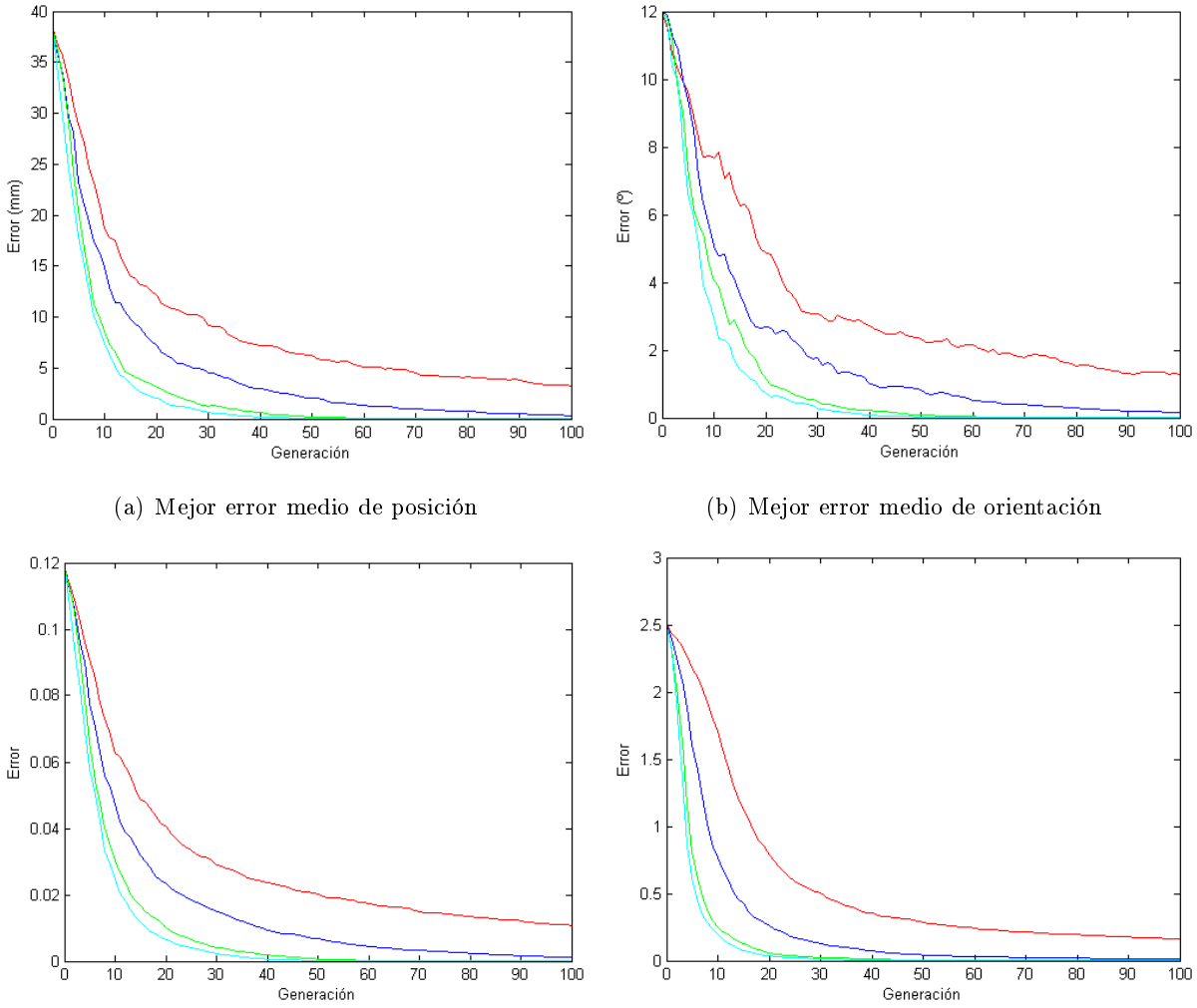
En la tabla 4.18 se muestra el número medio de iteraciones que necesita el algoritmo para alcanzar los umbrales de error fijados con sus respectivas desviaciones estándar (nº iteraciones ± desviación estándar) calculadas a partir de 100 pruebas del algoritmo.

Tabla 4.18: Número de iteraciones de un robot de 3 GDL para distintos valores de CR.

CR	0,3	0,6	0,9	1
Nº Iteración 1	250 ± 0	209,4 ± 29,7	92,4 ± 16	74,1 ± 11,7
Nº Iteración 2	250 ± 0	174,1 ± 34,6	75,9 ± 12,8	59,8 ± 9,6
Nº Iteración 3	249,4 ± 6,2	124,3 ± 27,2	52,1 ± 9,2	43 ± 7,7

En la tabla 4.17 se demuestra que la función objetivo a optimizar en el problema de la cinemática inversa de robots manipuladores de 3 GDL no es separable puesto que el algoritmo no llega a converger para valores de CR igual a 0,3 y 0,6. En cambio para valores de CR igual a 0,9 y 1 el algoritmo converge de forma exitosa reafirmando que se trata de una función objetivo no separable y con múltiples puntos óptimos. Fijándonos en los porcentajes de tasas de acierto y los errores de posición y orientación el mejor valor para asignar a CR es 1, pero teniendo en cuenta que esto supondría que el algoritmo se convirtiera en un esquema solo de mutación, es más correcto escoger como valor de CR a 0,9.

En la tabla 4.18 se confirma lo mencionado anteriormente. Como el número máximo de iteraciones es 250, para CR=0,3 en los dos primeros umbrales no llega a converger en ninguna de las 100 repeticiones como demuestran las desviaciones estándar de las iteraciones. De nuevo, el mejor valor es CR=1 ya que llega en menos iteraciones a los umbrales fijados y sus desviaciones estándar son menores. En la figura 4.8 se muestra la evolución de los errores medios durante la ejecución del algoritmo.



(a) Mejor error medio de posición

(b) Mejor error medio de orientación

(c) Mejor error medio total

(d) Peor error medio total

Figura 4.8: Evolución del error de un robot de 3 GDL para distintos valores de CR: CR=0,3(rojo), CR=0,6(azul), CR=0,9(verde) y CR=1(azul claro).

La conclusión que se puede obtener a partir de las gráficas de la figura 4.8 es que para la solución al problema de la cinemática inversa de robots manipuladores es mas óptimo a medida que los valores de CR se acercan a 1.

4.4.3.2. Robot de 6 GDL

En la tabla 4.19 se muestra las tasas de acierto (%) obtenidas, así como los errores medios finales de posición (Ep), orientación (Eo) y total (Et), con sus respectivas desviaciones estándar (error ± desviación estándar) calculados a partir de 100 pruebas del algoritmo. El tiempo mos-

trado hace referencia al tiempo medio de ejecución empleado por el algoritmo en cada repetición.

Tabla 4.19: Errores finales y tasas de acierto de un robot de 6 GDL para distintos valores de CR.

CR	0,3	0,6	0,9	1
Tasa 1 (%)	0	0	5	92
Tasa 2 (%)	0	0	65	95
Tasa 3 (%)	0	3	96	95
E_P (mm)	9,4 ± 6,1	2,1 ± 2,6	5E-1 ± 3,4	7,6E-1 ± 4,1
E_O (^{o})	6,4 ± 3,5	1,3 ± 1,9	6,3E-2 ± 5,6E-1	6,6E-2 ± 5,7E-1
E_T	2,1E-2 ± 8,1E-3	4,5E-3 ± 5,4E-3	6,2E-4 ± 3,5E-3	8,8E-4 ± 4,2E-3
Tiempo (s)	1,5	1,6	1,6	1,6

En la tabla 4.20 se muestra el número medio de iteraciones que necesita el algoritmo para alcanzar los umbrales de error fijados con sus respectivas desviaciones estándar (n^o iteraciones ± desviación estándar) calculadas a partir de 100 pruebas del algoritmo.

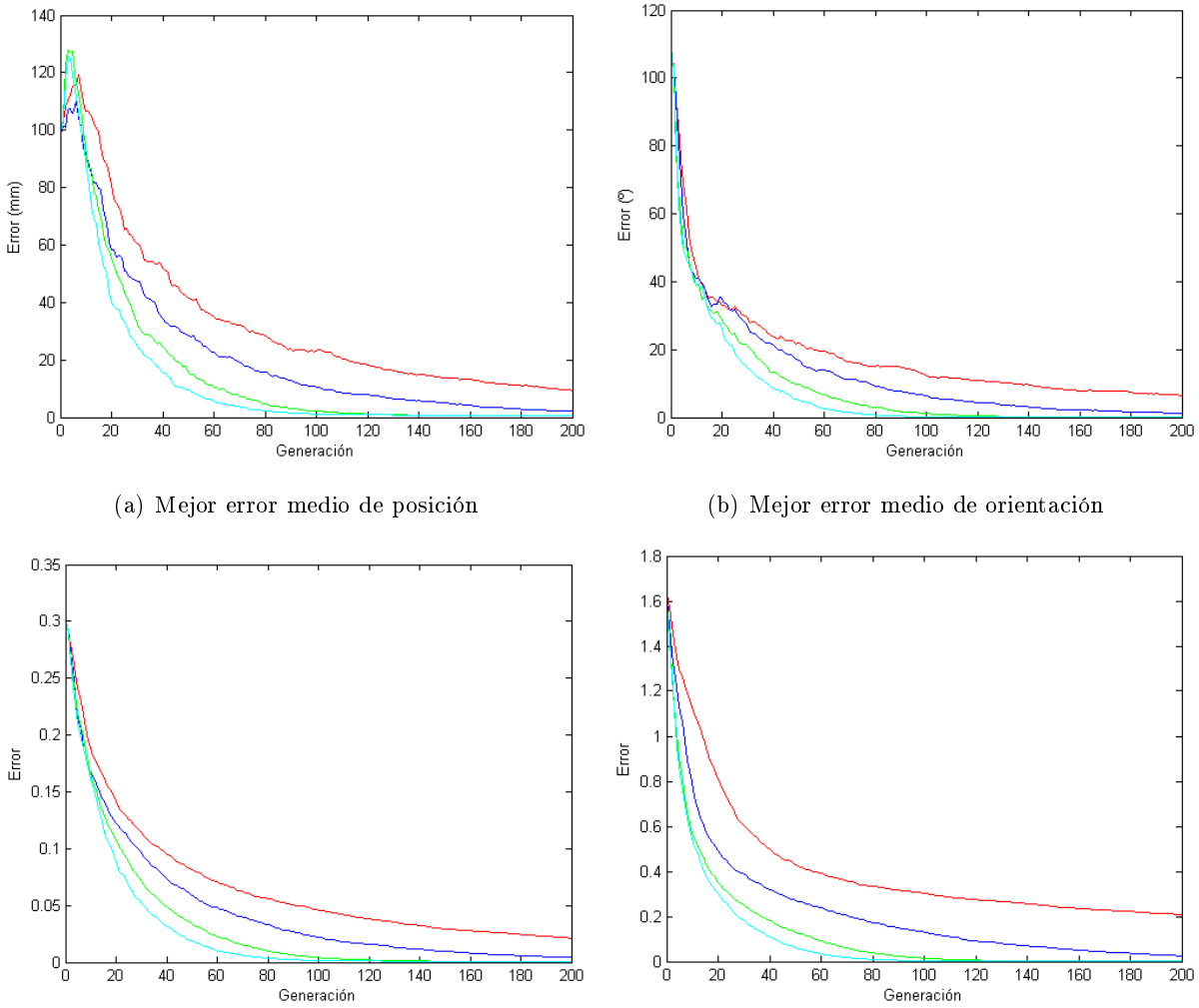
Tabla 4.20: Número de iteraciones de un robot de 6 GDL para distintos valores de CR.

CR	0,3	0,6	0,9	1
N^o Iteración 1	500 ± 0	491,1 ± 19,9	236,9 ± 69,5	168,5 ± 37,8
N^o Iteración 2	500 ± 0	442,4 ± 63,5	191,4 ± 68,6	153,7 ± 72,8
N^o Iteración 3	500 ± 0	329,5 ± 94,4	143,2 ± 44,6	109,5 ± 43

En la tabla 4.19 se demuestra que la función objetivo a optimizar en el problema de la cinemática inversa de robots manipuladores de 6 GDL no es separable puesto que el algoritmo no llega a converger para valores de CR igual a 0,3 y 0,6. En cambio para valores de CR igual a 0,9 y 1 el algoritmo converge de forma exitosa reafirmando que se trata de una función objetivo no separable y con múltiples puntos óptimos. Fijándonos en los porcentajes de tasas de acierto y los errores de posición y orientación el mejor valor para asignar a CR es 1, pero teniendo en cuenta que esto supondría que el algoritmo se convirtiera en un esquema solo de mutación, es más correcto escoger como valor de CR a 0,9.

En la tabla 4.20 respalda los resultados obtenidos en la tabla 4.19. Para CR=0,3 el algoritmo no llega a converger en 500 iteraciones a los umbrales de error fijados . A su vez, el valor de CR con el que mejor resultados se obtiene es 1.

En la figura 4.9 se muestra la evolución de los errores medios durante la ejecución del algoritmo.



(a) Mejor error medio de posición

(b) Mejor error medio de orientación

(c) Mejor error medio total

(d) Peor error medio total

Figura 4.9: Evolución del error de un robot de 6 GDL para distintos valores de CR: CR=0,3(rojo), CR=0,6(azul), CR=0,9(verde) y CR=1(azul caro).

En la figura 4.9 se observa que para la solución al problema de la cinemática inversa de robots manipuladores de 6 GDL se obtienen mejores resultados a medida que los valores de CR se acercan a 1.

4.5. Parámetros *discarding*

El mecanismo *discarding*, al igual que el *Differential Evolution*, posee pocos parámetros de control que permanecen constantes durante la ejecución del algoritmo DE, estos son: $DRATE$, α y β.

En este apartado se muestran los resultados obtenidos en las pruebas realizadas para conocer la influencia de estos parámetros de control del mecanismo *discarding* en la convergencia del algoritmo para le resolución del problema cinemático inverso de un robot manipulador de 3 GDL y 6 GDL. Para ello se escogen unos parámetros de control base, en este caso los que consideramos óptimos, que serán los mismos para ambos robots manipuladores: $DRATE = 2.5$; $\alpha = 2^{o}$; $\beta =$ NP/2. Estos valores se mantienen constantes durante la ejecución del mecanismo *discarding*. Los parámetros de control del algoritmo DE para un robot de 3 GDL (D=3) y otro de 6 GDL(D=6) son los siguientes: NP=10*D, F=0,5 y CR=0,9.

4.5.1. Tamaño de descarte (DRATE)

El DRATE representa el porcentaje de la población que será sustituido por el mecanismo *discarding*. Puede tener un valor entre 0 y 100, que es elegido por el usuario. A continuación se muestran los resultados obtenidos de las pruebas realizadas para determinar la influencia del DRATE en el comportamiento del algoritmo DE.

4.5.1.1. Robot de 3 GDL

En la tabla 4.21 se muestra las tasas de acierto (%) obtenidas, así como los errores medios finales de posición (Ep), orientación (Eo) y total (Et), con sus respectivas desviaciones estándar (error $\pm$ desviación estándar) calculados a partir de 100 pruebas del algoritmo. El tiempo mostrado hace referencia al tiempo medio de ejecución empleado por el algoritmo en cada repetición.

Tabla 4.21: Errores finales y tasas de acierto de un robot de 3 GDL para distintos valores de DRATE.

Drate	2,5	5	10	20
Tasa 1 (%)	80	85	94	94
Tasa 2 (%)	99	98	99	100
Tasa 3 (%)	100	100	100	100
E_P (mm)	1,4E-3 ± 3,3E-3	6,6E-4 ± 2E-3	4,6E-4 ± 2E-3	2,4E-4 ± 1,1E-3
E_O (º)	5,3E-4 ± 1,4E-3	5E-4 ± 1,9E-3	1,3E-4 ± 6,6E-4	7E-5 ± 2,1E-4
E_T	0 ± 1E-5	0 ± 1E-5	0 ± 0	0 ± 1E-5
Tiempo (s)	1,2E-1	1,1E-1	1,2E-1	1,5E-1

En la tabla 4.22 se muestra el número medio de iteraciones que necesita el algoritmo para alcanzar los umbrales de error fijados con sus respectivas desviaciones estándar (nº iteraciones ± desviación estándar) calculadas a partir de 100 pruebas del algoritmo.

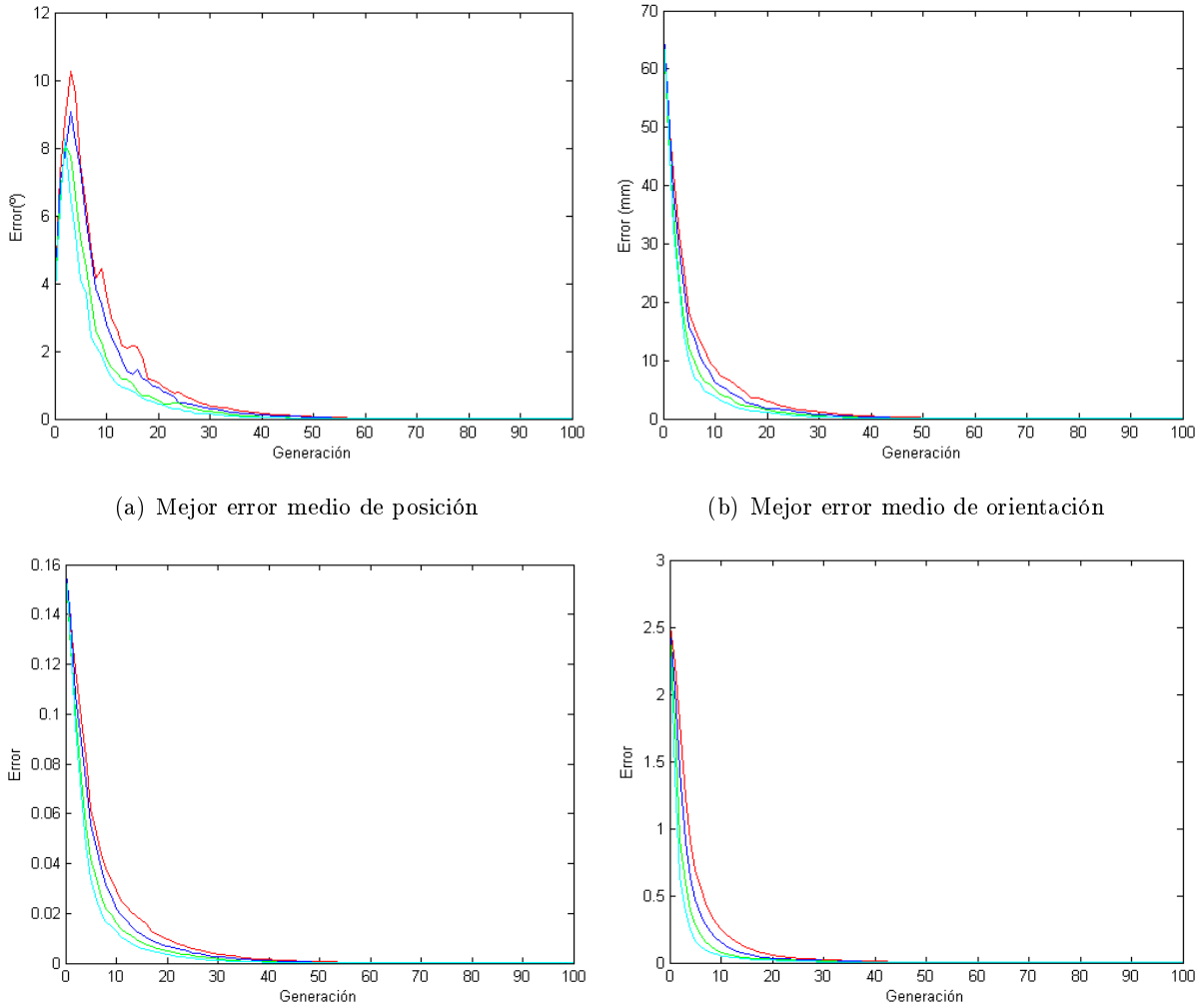
Tabla 4.22: Número de iteraciones de un robot de 3 GDL para distintos valores de DRATE.

Drate	2,5	5	10	20
Nº Iteración 1	92,6 ± 15,4	87,3 ± 13,6	83,3 ± 12,8	78,5 ± 13,3
Nº Iteración 2	70,9 ± 10,7	69,5 ± 12,8	62,3 ± 11	60 ± 11,6
Nº Iteración 3	52,8 ± 9,7	48,4 ± 8,1	42,2 ± 8	38,9 ± 10,3

En la tabla 4.21 los resultados muestran como mejores valores de DRATE a 10 y 20, ya que presentan menores errores de posición y orientación con desviaciones estándar también menores. Entre ambos valores la diferencia es mínima para poder elegir entre uno de ellos como el más adecuado al problema, en todo caso sería un DRATE igual a 20.

En la tabla 4.22 muestra resultados que concuerdan con las gráficas anteriores, para estos valores, cuanto mayor es el DRATE menos iteraciones se necesitan para llegar al umbral de error fijado. Por tanto sería un valor de DRATE = 20 el más adecuado en este caso.

En la figura 4.10 se muestra la evolución de los errores medios durante la ejecución del algoritmo.



(a) Mejor error medio de posición

(b) Mejor error medio de orientación

(c) Mejor error medio total

(d) Peor error medio total

Figura 4.10: Evolución del error de un robot de 3 GDL para distintos valores de DRATE: DRATE=2,5(rojo), DRATE=5(azul), DRATE=10(verde) y DRATE=20(azul claro).

En las gráficas que se muestran en la figura 4.10 se ve que para todos los valores de DRATE el algoritmo converge correctamente, pero a medida que el valor del DRATE aumenta la convergencia es más rápida.

4.5.1.2. Robot de 6 GDL

En la tabla 4.23 se muestra las tasas de acierto (%) obtenidas, así como los errores medios finales de posición (Ep), orientación (Eo) y total (Et), con sus respectivas desviaciones estándar (error ± desviación estándar) calculados a partir de 100 pruebas del algoritmo. El tiempo mos-

trado hace referencia al tiempo medio de ejecución empleado por el algoritmo en cada repetición.

Tabla 4.23: Errores finales y tasas de acierto de un robot de 6 GDL para distintos valores de DRATE.

Drate	2,5	5	10	20
Tasa 1 (%)	19	45	62	68
Tasa 2 (%)	84	91	89	86
Tasa 3 (%)	100	96	94	91
E_P (mm)	5E-3 ± 8,3E-3	7,3E-1 ± 4,1	9,8E-1 ± 4,9	1,7 ± 7
E_O (º)	2,4E-3 ± 3,1E-3	5,7E-2 ± 5,6E-1	1,1E-1 ± 7,8E-1	1,7E-1 ± 9,6E-1
E_T	1E-5 ± 1E-5	8,3E-4 ± 4,2E-3	1,2E-3 ± 5E-3	2E-3 ± 7,2E-3
Tiempo (s)	1,9	1,8	1,9	1,9

En la tabla 4.24 se muestra el número medio de iteraciones que necesita el algoritmo para alcanzar los umbrales de error fijados con sus respectivas desviaciones estándar (nº iteraciones ± desviación estándar) calculadas a partir de 100 pruebas del algoritmo.

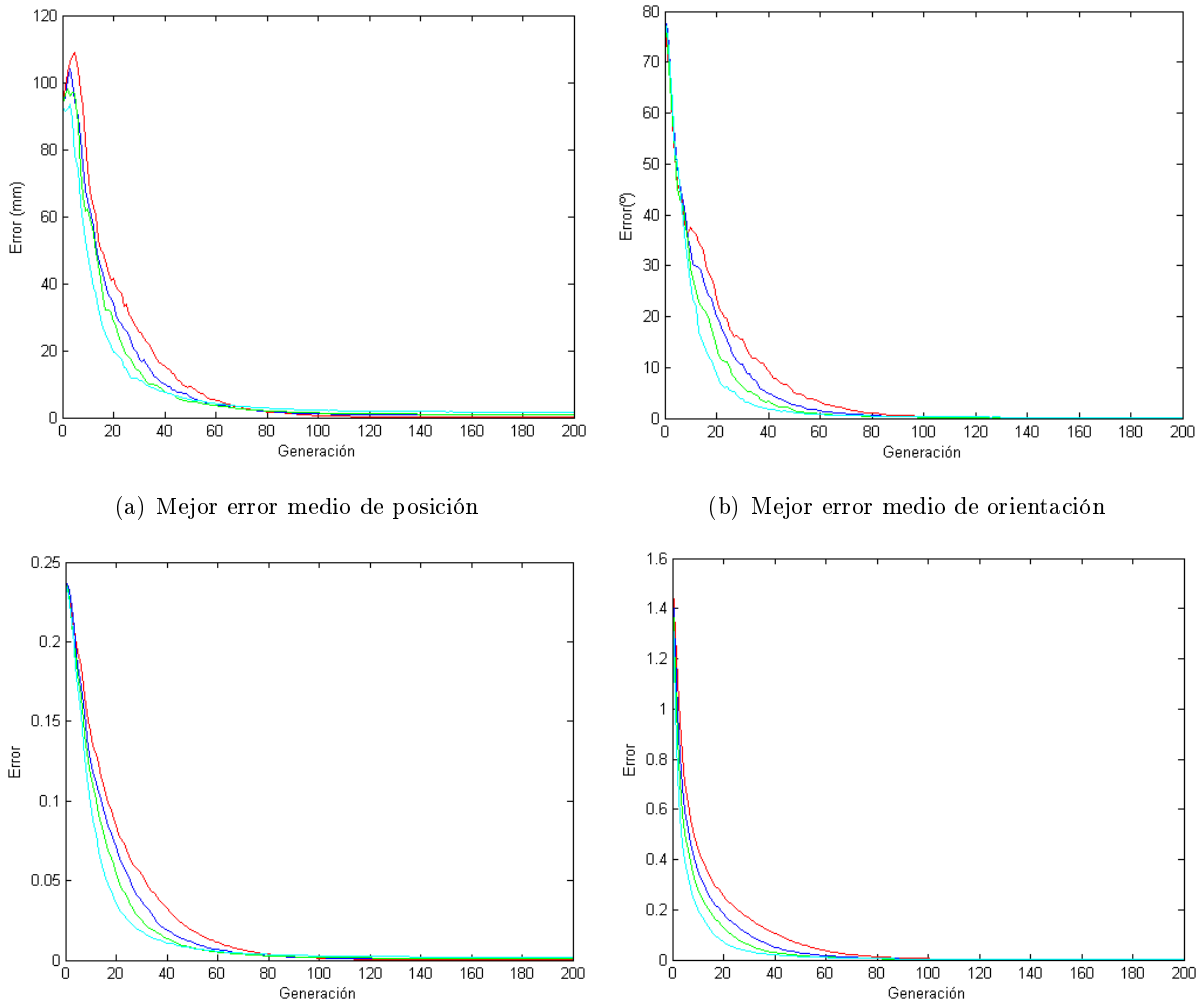
Tabla 4.24: Número de iteraciones de un robot de 6 GDL para distintos valores de DRATE.

Drate	2,5	5	10	20
Nº Iteración 1	212,6 ± 18,2	203,6 ± 20,8	190,6 ± 19,3	186,4 ± 27,7
Nº Iteración 2	179,7 ± 28,2	166,5 ± 34,4	157,3 ± 35,1	153,7 ± 42,8
Nº Iteración 3	140,6 ± 37,6	126,1 ± 38,8	117,3 ± 41,3	112,4 ± 47,1

En la tabla 4.23 se observa como a medida que el valor del DRATE aumenta el porcentaje de acierto de la Tasa 1 es mayor, sin embargo, la Tasa 3 disminuye. Esto es debido a que, un valor elevado de DRATE acelera la velocidad de convergencia del algoritmo, por lo tanto para 200 iteraciones, hay mayor probabilidad de llegar a umbrales de error muy pequeños. Sin embargo también existe la posibilidad de que el algorimto no converga correctamente y se desvíe de forma considerable de la solución deseada, es por eso, por lo que se obtiene menores porcentajes de acierto en la Tasa 3, y a su vez los errores medios obtenidos son más elevados.

Los resultados de la tabla 4.24 muestran la influencia del valor DRATE en la velocidad de convergencia del algoritmo, al necesitar menos iteraciones para alcanzar un umbral de error fijado utilizando un valor de DRATE elevado.

En la figura 4.11 se muestra la evolución de los errores medios durante la ejecución del algoritmo.



(a) Mejor error medio de posición

(b) Mejor error medio de orientación

(c) Mejor error medio total

(d) Peor error medio total

Figura 4.11: Evolución del error de un robot de 6 GDL para distintos valores de DRATE: DRATE=2,5(rojo), DRATE=5(azul), DRATE=10(verde) y DRATE=20(azul claro).

Como se puede observar en la figura 4.11, al tener un valor de DRATE elevado se alcanzan unos valores de umbral de error más rápidamente, pero el algoritmo no termina de converger y se desvía de la solución deseada.

4.5.2. Tamaño del ruido(ALPHA)

ALPHA ($\alpha = \sigma(\lambda)$) es una perturbación de ruido aleatorio que se le añade a las soluciones de los nuevos candidatos con el fin de explorar de forma eficiente todo el espacio de búsqueda. A continuación se muestran los resultados obtenidos de las pruebas realizadas para determinar la influencia del α en el comportamiento del algoritmo DE.

4.5.2.1. Robot de 3 GDL

En la tabla 4.25 se muestra las tasas de acierto (%) obtenidas, así como los errores medios finales de posición (Ep), orientación (Eo) y total (Et), con sus respectivas desviaciones estándar (error ± desviación estándar) calculados a partir de 100 pruebas del algoritmo. El tiempo mostrado hace referencia al tiempo medio de ejecución empleado por el algoritmo en cada repetición.

Tabla 4.25: Errores finales y tasas de acierto de un robot de 3 GDL para distintos valores de α.

Alpha(º)	1	2	4	8
Tasa 1 (%)	88	87	81	80
Tasa 2 (%)	100	99	98	97
Tasa 3 (%)	100	100	100	100
E_P (mm)	4,7E-4 ± 1,2E-3	7,2E-4 ± 1,7E-3	9,8E-4 ± 2,5E-3	9,8E-4 ± 2,2E-3
E_O (º)	2E-4 ± 4,6E-4	3,1E-4 ± 1E-3	4,1E-4 ± 1,1E-3	4,7E-4 ± 1,3E-3
E_T	0 ± 0	0 ± 1E-5	0 ± 1E-5	0 ± 1E-5
Tiempo (s)	8,3E-2	8,6E-2	8,3E-2	8,4E-2

En la tabla 4.26 se muestra el número medio de iteraciones que necesita el algoritmo para alcanzar los umbrales de error fijados con sus respectivas desviaciones estándar (nº iteraciones ± desviación estándar) calculadas a partir de 100 pruebas del algoritmo.

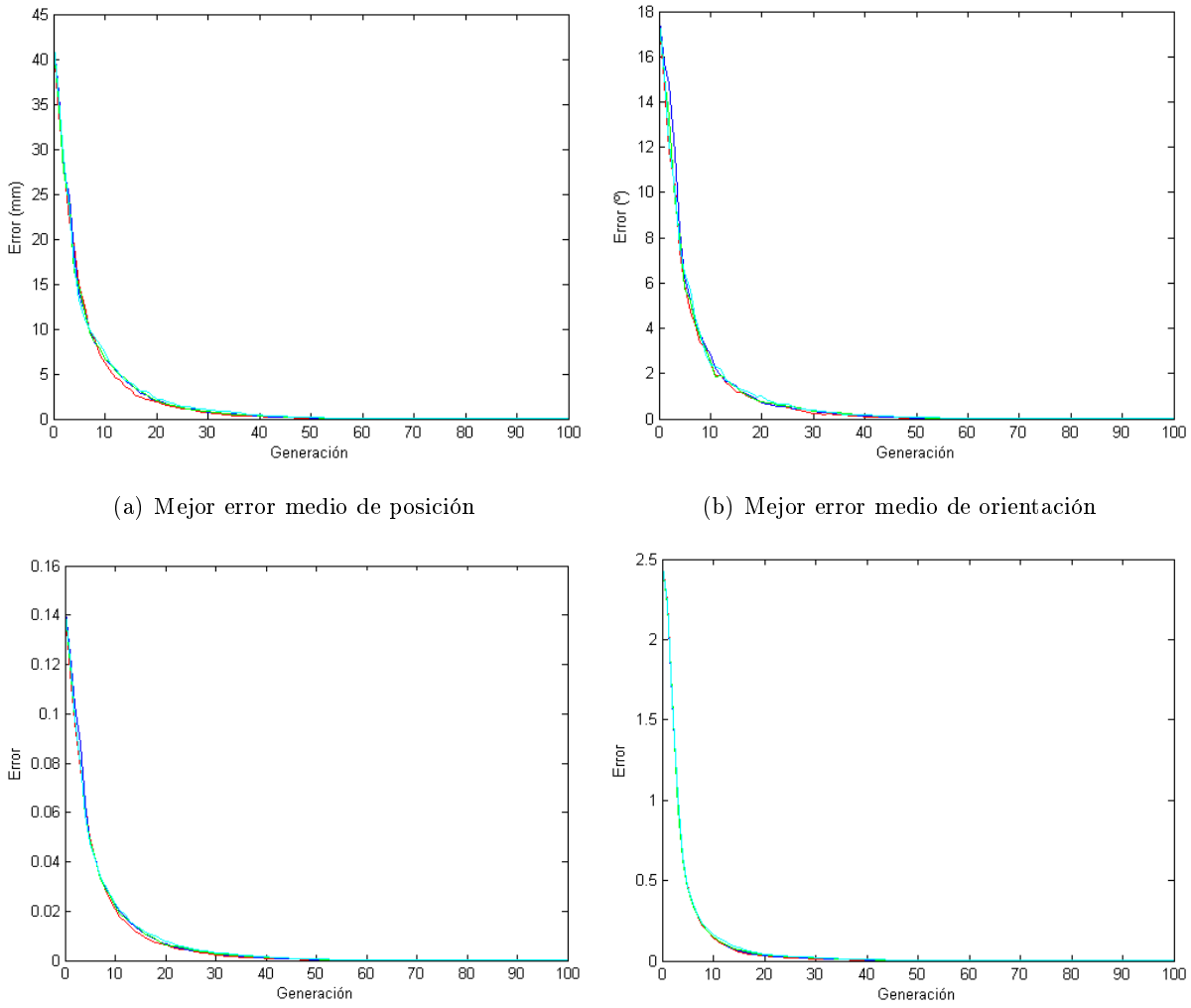
Tabla 4.26: Número de iteraciones de un robot de 3 GDL para distintos valores de α.

Alpha(^{o})	1	2	4	8
Nº Iteración 1	85,3 ± 10,8	88,1 ± 13,4	88,7 ± 14	89,3 ± 14,6
Nº Iteración 2	66,2 ± 12,1	69,9 ± 13,4	71,5 ± 11,7	72,2 ± 12,2
Nº Iteración 3	47,7 ± 8,7	48,4 ± 8,4	50,5 ± 8,7	52,4 ± 9,7

La influencia de α sobre el mecanismo *discarding* se empieza a notar cuando los umbrales de error son cada vez menores, puesto que para los umbrales 2 y 3 las tasas de acierto son prácticamente iguales. No obstante viendo los errores de posición y orientación mostrados en la tabla 4.25 se deduce que a menores valores de α se obtienen mejores resultados.

De nuevo se vuelve a observar en la tabla 4.26 la gran igualdad en os resultados para los diferentes valores de α, siendo de nuevo α=1º el que mejor responde al necesitar menos iteraciones para alcanzar los umbrales fijados.

En la figura 4.12 se muestra la evolución de los errores medios durante la ejecución del algoritmo.



(a) Mejor error medio de posición

(b) Mejor error medio de orientación

(c) Mejor error medio total

(d) Peor error medio total

Figura 4.12: Evolución del error de un robot de 3 GDL para distintos valores de α: α=1º(rojo), α=2º(azul), α=4º(verde) y α=8º(azul claro).

La diferencia en el resultado obtenido es tan mínima entre unos valores unos valores u otros de α que en las gráficas de la figura 4.25 prácticamente se superponen todos sobre la misma línea de convergencia.

4.5.2.2. Robot de 6 GDL

En la tabla 4.27 se muestra las tasas de acierto (%) obtenidas, así como los errores medios finales de posición (Ep), orientación (Eo) y total (Et), con sus respectivas desviaciones estándar (error $\pm$ desviación estándar) calculados a partir de 100 pruebas del algoritmo. El tiempo mos-

trado hace referencia al tiempo medio de ejecución empleado por el algoritmo en cada repetición.

Tabla 4.27: Errores finales y tasas de acierto de un robot de 6 GDL para distintos valores de α.

Alpha(º)	1	2	4	8
Tasa 1 (%)	53	50	38	37
Tasa 2 (%)	87	94	91	83
Tasa 3 (%)	90	95	94	89
E_P (mm)	7,8E-1 ± 4,1	7,4E-1 ± 4,1	9,8E-1 ± 4,7	8,3E-1 ± 4,1
E_O (º)	3,4E-1 ± 1,3	1,1E-1 ± 7,8E-1	1,1E-1 ± 7,8E-1	3,4E-1 ± 1,3
E_T	1,4E-3 ± 4,8E-3	9,4E-4 ± 4,3E-3	1,2E-3 ± 5E-3	1,5E-3 ± 4,8E-3
Tiempo (s)	1,6	1,7	1,6	1,6

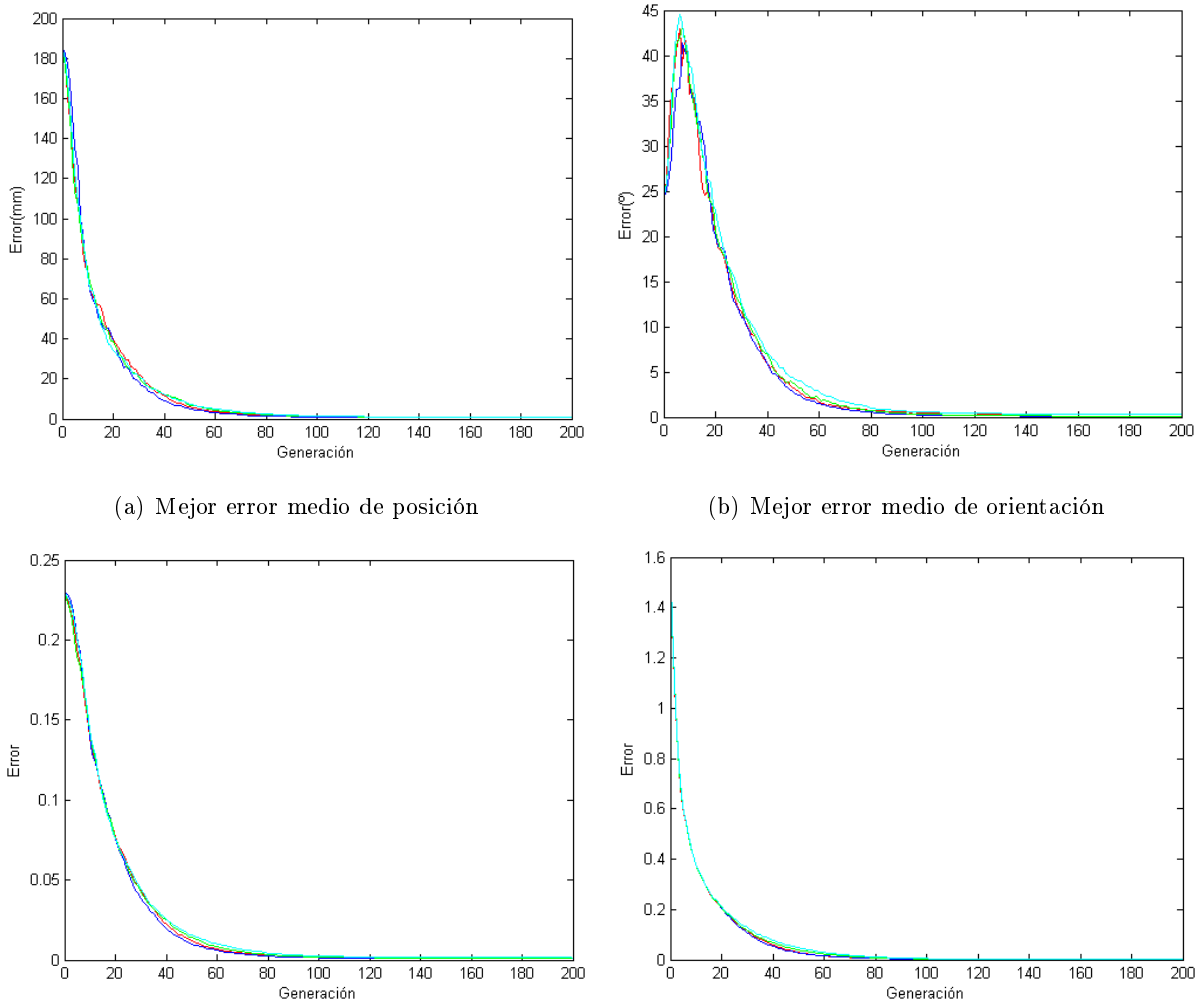
En la tabla 4.28 se muestra el número medio de iteraciones que necesita el algoritmo para alcanzar los umbrales de error fijados con sus respectivas desviaciones estándar (nº iteraciones ± desviación estándar) calculadas a partir de 100 pruebas del algoritmo.

Tabla 4.28: Número de iteraciones de un robot de 6 GDL para distintos valores de α.

Alpha(º)	1	2	4	8
Nº Iteración 1	207,1 ± 24,9	205,3 ± 19,7	210,6 ± 24,9	209,5 ± 23,4
Nº Iteración 2	175,1 ± 37,2	176,9 ± 37	167,8 ± 31,1	174,4 ± 29,3
Nº Iteración 3	133,2 ± 47,3	130,5 ± 43,9	140,9 ± 49,5	139,8 ± 39

La selección del valor de α en el mecanismo *discarding* para la cinemática inversa de robots manipuladores de 6 GDL no es determinante en la obtención de buenos resultados. No obstante como se puede observar en las tablas 4.27 y 4.28 es recomendable escoger valores igual o menor a 2º.

En la figura 4.13 se muestra la evolución de los errores medios durante la ejecución del algoritmo.



(a) Mejor error medio de posición

(b) Mejor error medio de orientación

(c) Mejor error medio total

(d) Peor error medio total

Figura 4.13: Evolución del error de un robot de 6 GDL para distintos valores de α: α=1º(rojo), α=2º(azul), α=4º(verde) y α=8º(azul claro).

En la figura 4.13 se observa como los valores de los errores medios obtenidos evolucionan de forma similar durante la ejecución del algoritmo, por lo que no se puede destacar ningún valor de α como óptimo a partir de las gráficas.

4.5.3. Tamaño de la Población de reemplazo (BETA)

β se trata de un número de población escogida entre los mejores elementos de la población actual. A continuación se muestran los resultados obtenidos de las pruebas realizadas para determinar la influencia del β en el comportamiento del algoritmo DE.

4.5.3.1. Robot de 3 GDL

En la tabla 4.29 se muestra las tasas de acierto (%) obtenidas, así como los errores medios finales de posición (Ep), orientación (Eo) y total (Et), con sus respectivas desviaciones estándar (error ± desviación estándar) calculados a partir de 100 pruebas del algoritmo. El tiempo mostrado hace referencia al tiempo medio de ejecución empleado por el algoritmo en cada repetición.

Tabla 4.29: Errores finales y tasas de acierto de un robot de 3 GDL para distintos valores de β.

Beta	NP/8	NP/4	NP/2	NP
Tasa 1 (%)	85	75	80	83
Tasa 2 (%)	99	96	98	96
Tasa 3 (%)	100	100	100	100
E_P (mm)	7,9E-4 ± 2,3E-3	1,4E-3 ± 3,3E-3	1,2E-3 ± 2,8E-3	1,4E-3 ± 4,6E-3
E_O (º)	2,2E-4 ± 6,6E-4	6,2E-4 ± 1,6E-3	5,5E-4 ± 1,4E-3	4,4E-4 ± 1,5E-3
E_T	0 ± 1E-5	0 ± 1E-5	0 ± 1E-5	0 ± 1E-5
Tiempo (s)	8,3E-2	8,2E-2	8,2E-2	8,2E-2

En la tabla **??** se muestra el número medio de iteraciones que necesita el algoritmo para alcanzar los umbrales de error fijados con sus respectivas desviaciones estándar (nº iteraciones ± desviación estándar) calculadas a partir de 100 pruebas del algoritmo.

Tabla 4.30: Número de iteraciones de un robot de 3 GDL para distintos valores de β.

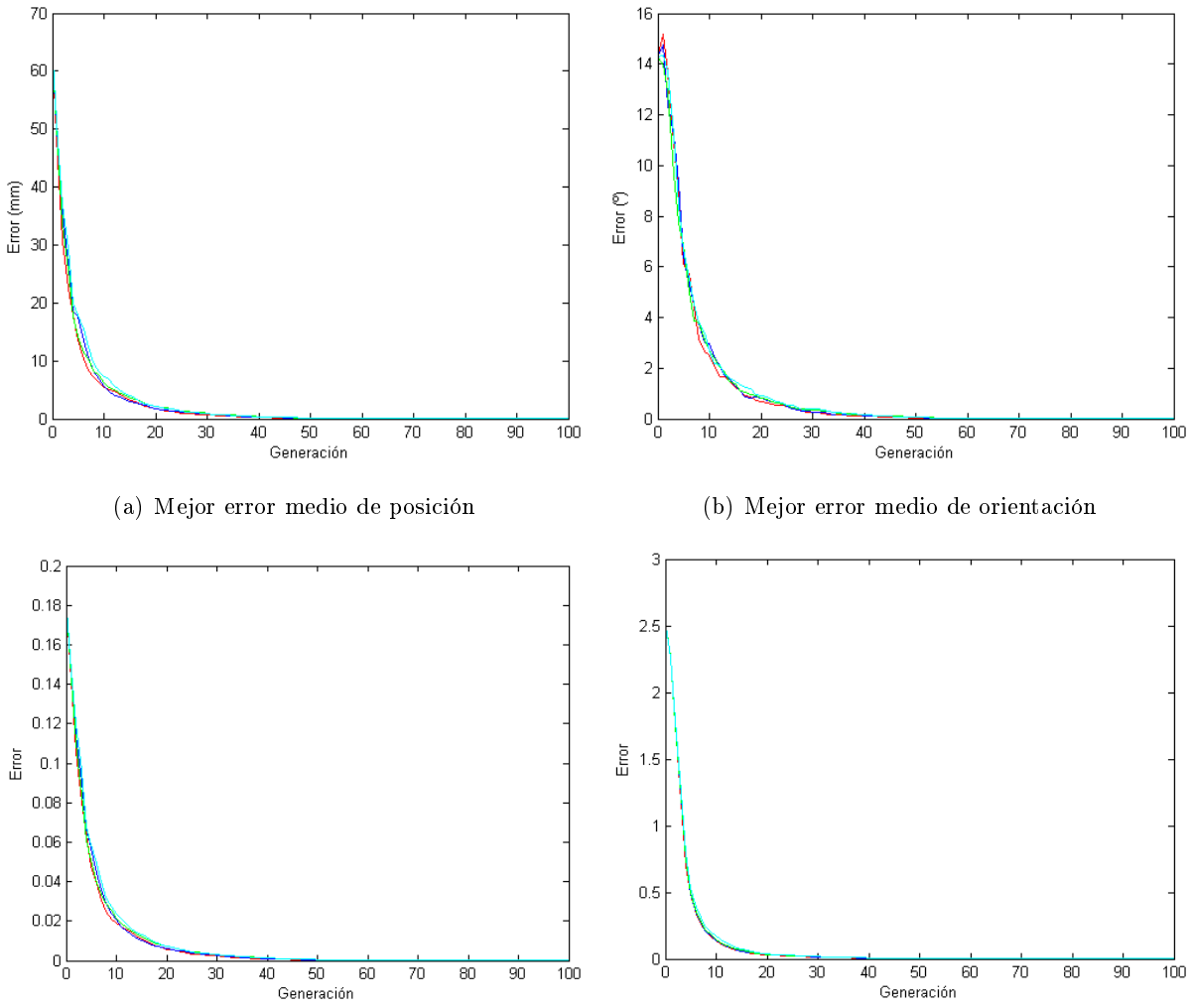
Beta	NP/8	NP/4	NP/2	NP
Nº Iteración 1	80 ± 12,4	85,5 ± 16,5	85 ± 14,5	86 ± 14,6
Nº Iteración 2	65,3 ± 12,6	65,6 ± 11,2	67,3 ± 12,4	66,7 ± 10,4
Nº Iteración 3	45,8 ± 8,8	45,6 ± 9,8	47,4 ± 9,3	48,1 ± 10

Las tasas de acierto mostrados en la tabla 4.29 son muy parecidas para los cuatro valores de β seleccionados. Si nos guiamos por los errores medios finales de posición y orientación habría que escoger un valor de Población pequeño para obtener la solución más adecuada.

De la tabla 4.30 se puede deducir que se obtienen mejores resultados con valores de β de

NP/8 ó NP/4.

En la figura 4.14 se muestra la evolución de los errores medios durante la ejecución del algoritmo.



(a) Mejor error medio de posición

(b) Mejor error medio de orientación

(c) Mejor error medio total

(d) Peor error medio total

Figura 4.14: Evolución del error de un robot de 3 GDL para distintos valores de β. β=NP/8(rojo), β=NP/4(azul), β=NP/2(verde) y β=NP-1(azul claro).

Las gráficas de la figura 4.14 no aclaran nada sobre el mejor valor de β.

4.5.3.2. Robot de 6 GDL

En la tabla 4.31 se muestra las tasas de acierto (%) obtenidas, así como los errores medios finales de posición (Ep), orientación (Eo) y total (Et), con sus respectivas desviaciones estándar

(error ± desviación estándar) calculados a partir de 100 pruebas del algoritmo. El tiempo mostrado hace referencia al tiempo medio de ejecución empleado por el algoritmo en cada repetición.

Tabla 4.31: Errores finales y tasas de acierto de un robot de 6 GDL para distintos valores de β.

Beta	NP/8	NP/4	NP/2	NP
Tasa 1 (%)	72	68	60	42
Tasa 2 (%)	87	88	96	87
Tasa 3 (%)	89	90	98	96
E_P (mm)	1,5 ± 5,7	1 ± 4,7	2,5E-1 ± 2,4	7,3E-1 ± 4,1
E_O (^{o})	1,7E-1 ± 9,5E-1	2,8E-1 ± 1,2	5,6E-2 ± 5,6E-1	5,7E-2 ± 5,6E-1
E_T	1,8E-3 ± 5,9E-3	1,5E-3 ± 5,2E-3	3,5E-4 ± 2,6E-3	8,3E-4 ± 4,2E-3
Tiempo (s)	1,6	1,6	1,6	1,6

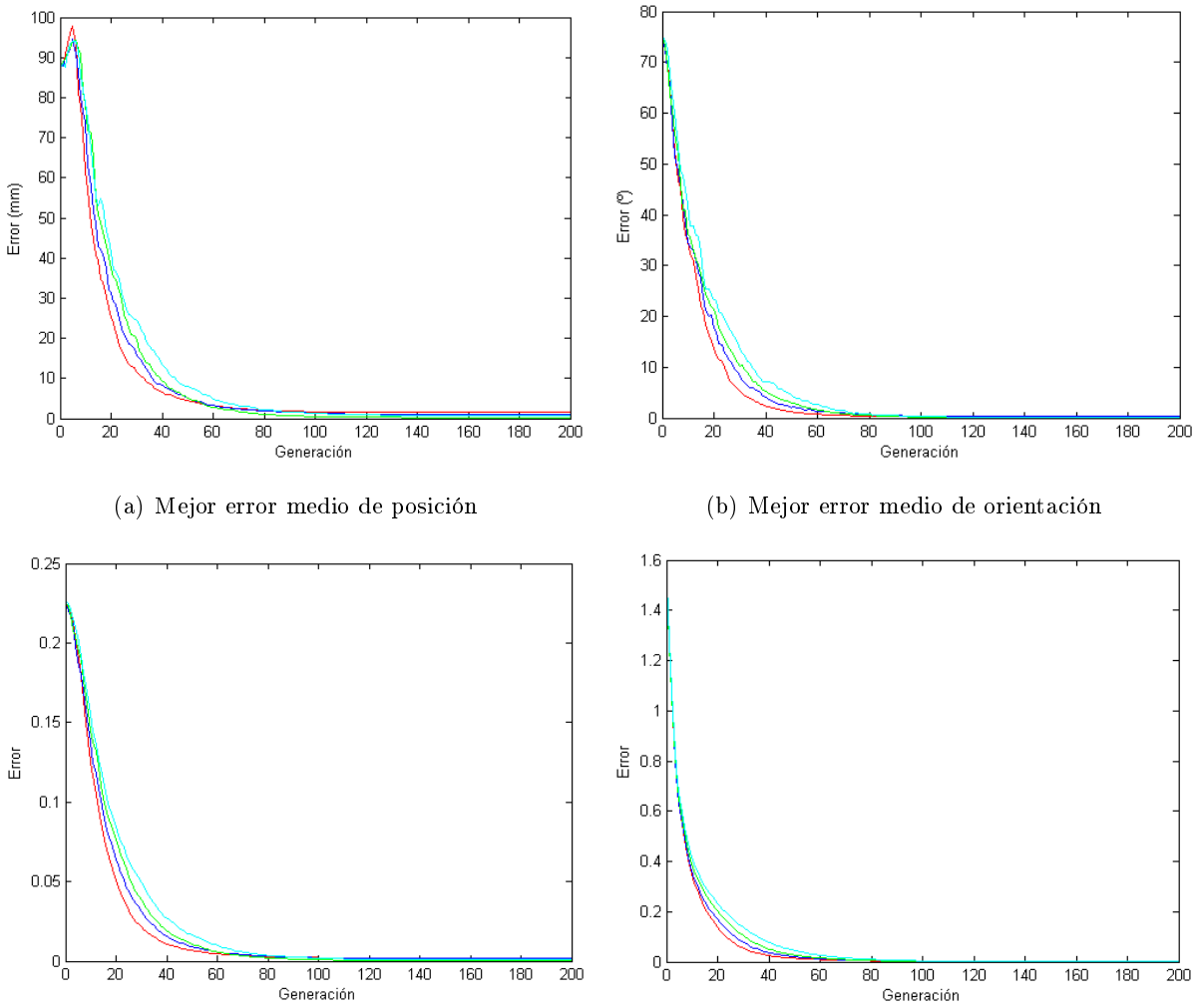
En la tabla 4.32 se muestra el número medio de iteraciones que necesita el algoritmo para alcanzar los umbrales de error fijados con sus respectivas desviaciones estándar (n^o iteraciones ± desviación estándar) calculadas a partir de 100 pruebas del algoritmo.

Tabla 4.32: Número de iteraciones de un robot de 6 GDL para distintos valores de β.

Beta	NP/8	NP/4	NP/2	NP
N^o Iteración 1	192,7 ± 23,1	189,8 ± 19,5	199,5 ± 20,8	207,1 ± 20,4
N^o Iteración 2	151,6 ± 34,5	157,8 ± 29,6	159 ± 22,7	167,6 ± 25,8
N^o Iteración 3	112,1 ± 40,7	119 ± 40,7	116,6 ± 32	127 ± 30,8

Ciñéndonos a los resultados obtenidos en las tabla 4.31 el mejor valor de βes igual a la mitad del tamaño de la población, es decir NP/2. De la tabla 4.10 no se obtiene una conclusión concreta al ser los resultados bastantes similares para los cuatro valores de β.

En la figura 4.15 se muestra la evolución de los errores medios durante la ejecución del algoritmo.



(a) Mejor error medio de posición

(b) Mejor error medio de orientación

(c) Mejor error medio total

(d) Peor error medio total

Figura 4.15: Evolución del error de un robot de 6 GDL para distintos valores de β: β=NP/8(rojo), β=NP/4(azul), β=NP/2(verde) y β=NP-1(azul claro).

Por las gráficas mostradas en la figura 4.15 se puede deducir que un valor pequeño de β aumenta la velocidad de convergencia del algoritmo, aunque el algoritmo no llega a converger correctamente.

4.6. *Differential Evolution* vs DE + *discarding*

En este apartado se muestran los resultados obtenidos en las pruebas realizadas comparando el mecanismo *discarding* frente al algoritmo DE clásico en la resolución del problema cinemático

inverso para robots manipuladores de 3 GDL y 6 GDL.

4.6.1. Robot de 3 GDL

En la tabla 4.33 se muestra las tasas de acierto (%) obtenidas, así como los errores medios finales de posición (Ep), orientación (Eo) y total (Et), con sus respectivas desviaciones estándar (error ± desviación estándar) calculados a partir de 100 pruebas del algoritmo. El tiempo mostrado hace referencia al tiempo medio de ejecución empleado por el algoritmo en cada repetición.

Tabla 4.33: Errores finales y tasas de acierto de un robot de 3 GDL utilizando DE y *discarding*.

	DE	DE + *discarding*
Tasa 1 (%)	79	84
Tasa 2 (%)	97	100
Tasa 3 (%)	100	100
E_P (mm)	1,1E-3 ± 2,7E-3	6,4E-4 ± 1,4E-3
E_O (º)	4,7E-4 ± 1,4E-3	2E-4 ± 4,3E-4
E_T	0 ± 1E-5	0 ± 0
Tiempo (s)	8,4E-2	8,5E-2

En la tabla 4.34 se muestra el número medio de iteraciones que necesita el algoritmo para alcanzar los umbrales de error fijados con sus respectivas desviaciones estándar (nº iteraciones ± desviación estándar) calculadas a partir de 100 pruebas del algoritmo.

Tabla 4.34: Número de iteraciones de un robot de 3 GDL utilizando DE y *discarding*.

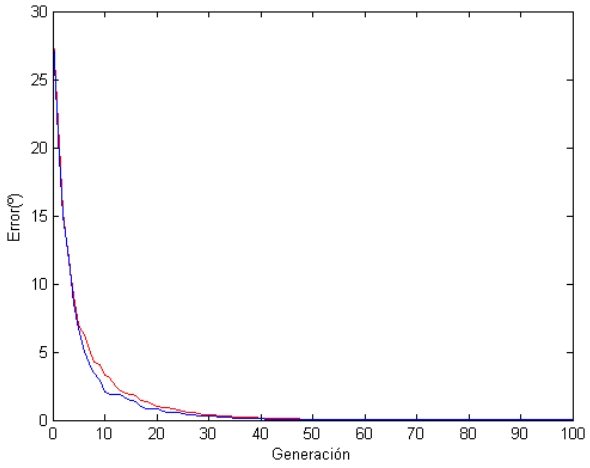
	DE	DE + *discarding*
Nº Iteración 1	91 ± 14,1	87,6 ± 15,3
Nº Iteración 2	72 ± 13	69,6 ± 13,1
Nº Iteración 3	53,2 ± 10,1	47,7 ± 7,7

Los resultados de la tabla 4.33 muestran unos mejores resultados utilizando el mecanismo *discarding* para un robot manipulador de 3 GDL. Se obtienen mejores tasas de aciertos para

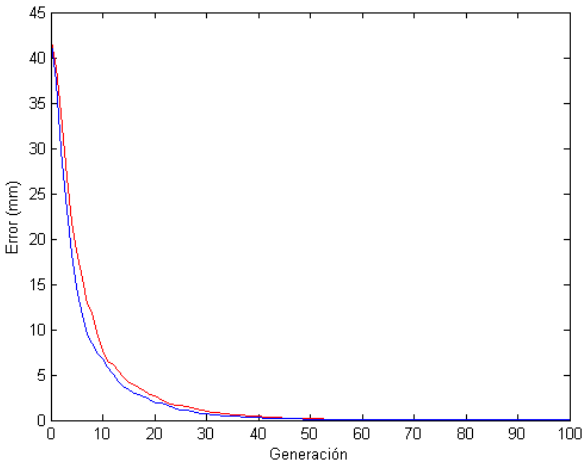
las 100 repeticiones del algoritmo, menores errores de posición y orientación, y presenta mayor estabilidad al tener valores de las desviaciones estándar menores.

La tabla 4.34 reafirma todo lo comentado anteriormente, con el mecanismo *discarding* se necesitan menos iteraciones para alcanzar los umbrales de error establecidos. Se obtiene un porcentaje de mejora en la velocidad de convergencia del algoritmo del 3,5 % para los umbrales de error E1 y E2, y un 10,3 % de mejora para el umbral E3.

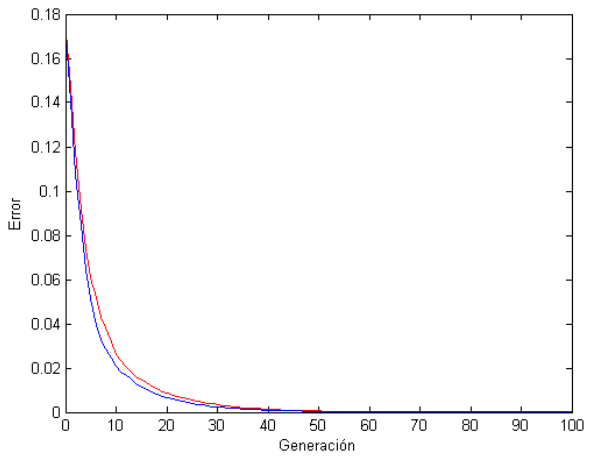
En la figura 4.16 se muestra la evolución de los errores medios durante la ejecución del algoritmo



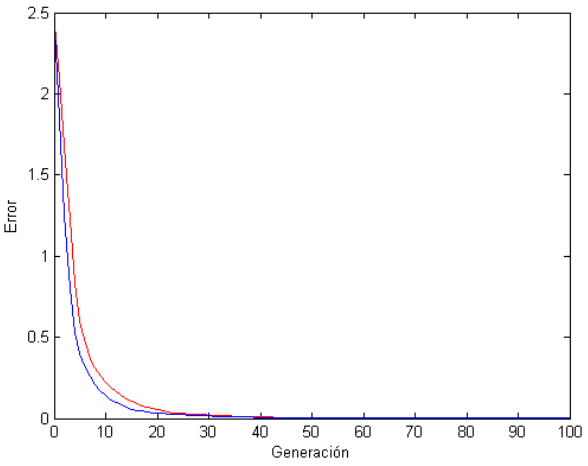
(a) Mejor error medio de posición



(b) Mejor error medio de orientación



(c) Mejor error medio total



(d) Peor error medio total

Figura 4.16: Evolución del error de un robot de 3 GDL utilizando DE y *discarding*: DE(rojo) y *discarding*(azul).

En las gráficas de la figura 4.16 se aprecia como el algorimto DE con el mecanismo *discarding* implementado converge antes que el algoritmo DE clásico.

4.6.2. Robot de 6 GDL

En la tabla 4.35 se muestra las tasas de acierto (%) obtenidas, así como los errores medios finales de posición (Ep), orientación (Eo) y total (Et), con sus respectivas desviaciones estándar (error ± desviación estándar) calculados a partir de 100 pruebas del algoritmo. El tiempo mostrado hace referencia al tiempo medio de ejecución empleado por el algoritmo en cada repetición.

Tabla 4.35: Errores finales y tasas de acierto de un robot de 6 GDL utilizando DE y *discarding*.

	DE	DE + *discarding*
Tasa 1 (%)	11	46
Tasa 2 (%)	71	90
Tasa 3 (%)	94	95
E_P (mm)	7,4E-1 ± 4,1	9,6E-1 ± 4,7
E_O (º)	6,2E-2 ± 5,6E-1	5,7E-2 ± 5,6E-1
E_T	8,5E-4 ± 4,2E-3	1,1E-3 ± 4,8E-3
Tiempo (s)	1,5	1,6

En la tabla 4.36 se muestra el número medio de iteraciones que necesita el algoritmo para alcanzar los umbrales de error fijados con sus respectivas desviaciones estándar (nº iteraciones ± desviación estándar) calculadas a partir de 100 pruebas del algoritmo.

Tabla 4.36: Número de iteraciones de un robot de 6 GDL utilizando DE y *discarding*.

	DE	DE + *discarding*
Nº Iteración 1	228,2 ± 19,6	196,8 ± 18,4
Nº Iteración 2	187,6 ± 28,5	156,8 ± 24,9
Nº Iteración 3	146,8 ± 25,9	118,4 ± 34,2

De la tabla 4.35 se obtiene la conclusión que utilizando el mecanismo *discarding* aumenta el

porcentaje de acierto de que la solución obtenida sea la deseada en la resolución de la cinemática inversa de un robot manipulador de 6 GDL. Sin embargo, se obtienen errores medios mayores, lo que refleja una cierta inestabilidad en el mecanismo *discarding*.

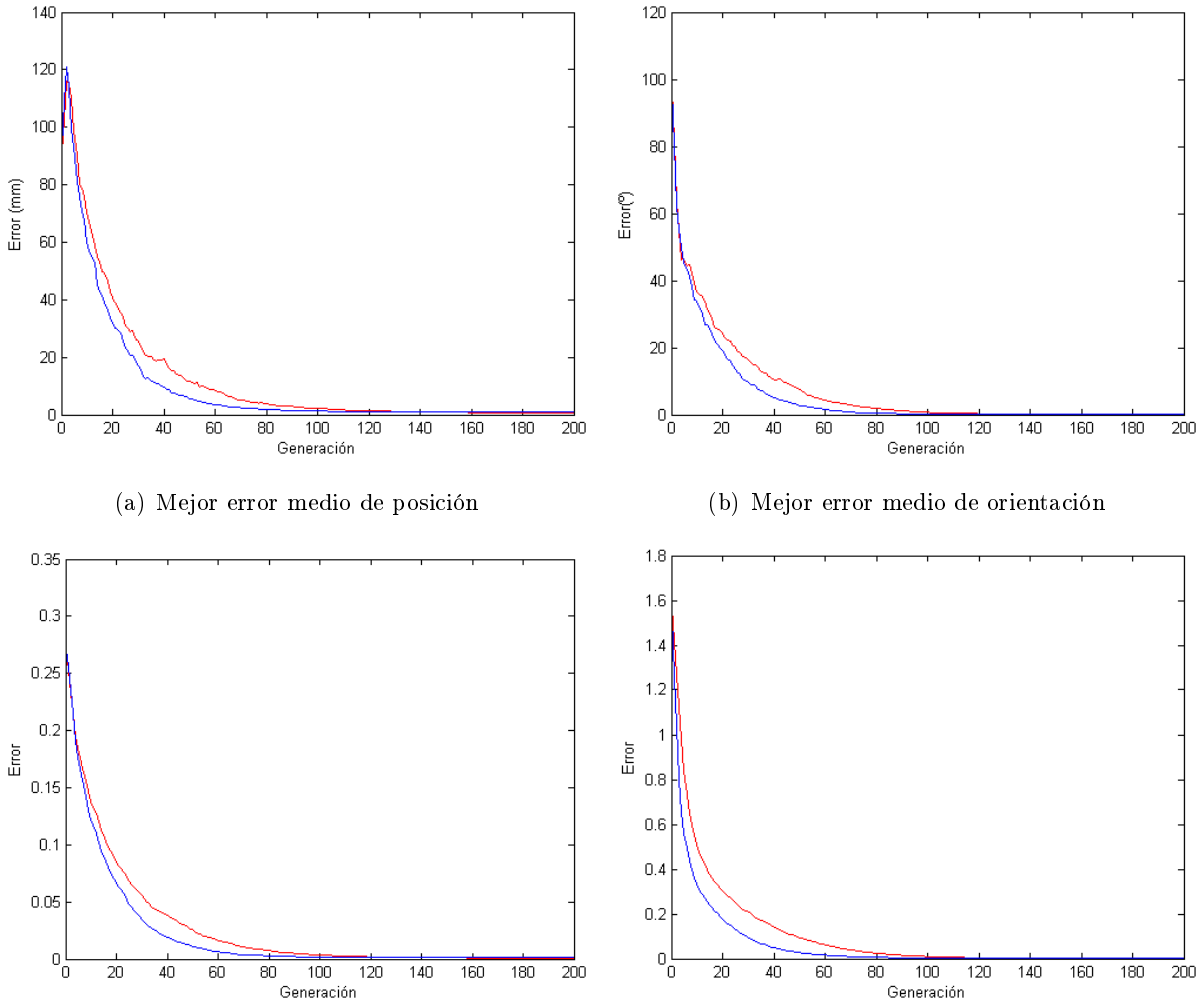
En la tabla 4.36 se observa como con el mecanismo *discarding* implementado se necesitan un menor número de iteraciones para alcanzar un cierto umbral de error fijado. En la tabla 4.43 se muestran los porcentajes de mejora obtenidios con el mecanismo *discarding* implementado.

Tabla 4.37: Porcentaje de mejora de un robot de 6 GDL utilizando DE + *discarding*.

Umbral Error	E1	E2	E3
Mejora (%)	13,8	16,4	19,4

Se observa que el porcentaje de mejora es mayor con umbrales de error menos restrictivos, es decir, para E3. Esta situación es debida a que la aceleración de convergencia del algoritmo que el mecanismo *discarding* se produce principalmente en la primera mitad de iteraciones.

En la figura 4.17 se muestra la evolución de los errores medios durante la ejecución del algoritmo. Las gráficas muestran como inicialmente los errores medios obtenidos mediante el mecanismo *discarding* adquieren valores mínimos más rápidamente, sin embargo, en las generaciones finales se desvían ligeramente de la solución obteniendo finalmente un error final mayor que utilizando el algoritmo DE.



(a) Mejor error medio de posición

(b) Mejor error medio de orientación

(c) Mejor error medio total

(d) Peor error medio total

Figura 4.17: Evolución del error de un robot de 6 GDL utilizando DE y *discarding*: DE(rojo) y *discarding*(azul).

4.7. Espacio de trabajo

Hasta ahora se ha realizado un trabajo de evaluación del algoritmo para un solo punto. Por tanto, para explorar el espacio de trabajo de los robots manipuladores y probar que los resultados son consistentes se han seleccionado aleatoriamente cinco coordenadas en el espacio de trabajo de los robots manipuladores de 3 GDL y de 6 GDL. Para demostrar la consistencia del algoritmo DE para todo el espacio de trabajo de los 2 robots manipuladores utilizados se realiza únicamente la Prueba 1, pero en este caso el número de repeticiones que haremos ejecutar el algoritmo para

el robot de 6 GDL será de 50.

4.7.1. Robot de 3 GDL

En la tabla 4.38 se muestran los puntos escogidos dentro del espacio de trabajo del robot de 3 GDL.

Tabla 4.38: Coordenadas de 5 puntos en el espacio de trabajo de un robot de 3 GDL.

Punto	x (m)	y (m)	φ (^{o})
1	2,1E-1	4E-1	70
2	2,5E-1	3,5E-1	80
3	3,2E-1	-1,7E-1	-40
4	-1,5E-1	3,2E-1	165
5	-2,5E-2	-4,4E-1	-110

La tabla 4.39 muestra los buenos resultados obtenidos para la resolución del problema cinemático inverso de un robot manipulador de 3 GDL usando el algoritmo DE para todo su espacio de trabajo. La posición 1 corresponde a una configuración singular del robot. Los puntos 2 y 3 tienen menores tasas de acierto, esto es debido a las dos configuraciones articulares que puede adquirir el robot para un mismo punto en el espacio cartesiano, denominadas codo arriba y codo abajo. Para los puntos 4 y 5, el robot también puede adquirir dos configuraciones articulares diferentes, pero el algoritmo únicamente encuentra una de las soluciones.

Tabla 4.39: Errores finales y tasas de acierto de un robot de 3 GDL para 5 coordenadas.

Punto	1	2	3	4	5
Tasa 1(%)	100	81	75	100	100
Tasa 2(%)	100	97	94	100	100
Tasa 3(%)	100	100	100	100	100
x (m)	2,1E-1 ± 0	2,5E-1 ± 0	3,2E-1 ± 0	-1,5E-1 ± 0	-2,5E-2 ± 0
y (m)	4E-1 ± 0	3,5E-1 ± 0	-1,7E-1 ± 0	3,2E-1 ± 0	-4,4E-1 ± 0
φ (º)	70 ± 1E-5	80 ± 2,6E-3	-40 ± 1,9E-3	165 ± 4E-5	-110 ± 5E-5
E_P(mm)	1E-5 ± 2E-5	1,3E-3 ± 2,9E-3	1,8E-3 ± 4,8E-3	7E-5 ± 6E-5	9E-5 ± 1E-4
E_O(º)	1E-5 ± 1E-5	6,8E-4 ± 2,5E-3	7,5E-4 ± 1,7E-3	2E-5 ± 3E-5	3E-5 ± 4E-5
E_T	0 ± 0	0 ± 1E-5	1E-5 ± 1E-5	0 ± 0	0 ± 0

4.7.2. Robot de 6 GDL

En la tabla 4.40 se muestran los puntos escogidos dentro del espacio de trabajo del robot de 6 GDL.

Tabla 4.40: Coordenadas de 5 puntos en el espacio de trabajo de un robot de 6 GDL.

Punto	x (m)	y (m)	z (m)	ϕ (º)	θ (º)	ψ (º)
1	1,1E-1	-5,8E-2	2,4E-1	-48,7	76,4	-63,8
2	-1,2E-1	-2,8E-2	2,4E-1	168,7	93,5	46,3
3	1,7E-1	-8,3E-1	-9,1E-2	-128,7	93,5	133,7
4	4E-1	7,4E-1	-9,1E-2	11,3	93,5	-176,3
5	-6,2E-1	3,6E-1	-3,6E-1	-99,5	99,5	-46,2

La tabla 4.41 muestra los buenos resultados obtenidos para la resolución del problema cinemático inverso de un robot manipulador de 6 GDL usando el algoritmo DE para todo su espacio de trabajo. Aun teniendo un porcentaje de acierto de las dos primeras tasas muy bajo, que se solucionaría aumentando el número de iteraciones de ejecución del algoritmo. Aún así, los um-

brales de error determinados por ambas tasas son muy pequeños, mas pequeños incluso que la tolerancia de desplazamiento del robot Puma 560.

Tabla 4.41: Errores finales y tasas de acierto de un robot de 6 GDL para 5 coordenadas.

Punto	1	2	3	4	5
Tasa 1 (%)	4	2	2	0	66
Tasa 2 (%)	72	56	44	40	76
Tasa 3 (%)	94	86	88	82	90
x (m)	1,1E-1 ± 3E-5	-1,2E-1 ± 1,7E-3	1,7E-1 ± 3,1E-4	4E-1 ± 1,2E-4	-6,2E-1 ± 0
y (m)	-5,8E-2 ± 4E-5	-2,8E-2 ± 3,9E-4	-8,3E-1 ± 6,7E-4	7,4E-1 ± 7E-5	3,6E-1 ± 0
z (m)	2,4E-1 ± 4E-5	2,4E-1 ± 5E-5	-9,1E-2 ± 2,7E-4	-9,1E-2 ± 1,1E-4	-3,6E-1 ± 0
ϕ (º)	-48,7 ± 1,1E-2	168,7 ± 1,4E-1	-128,5 ± 1,4	11,3 ± 2,8E-2	-99,7 ± 7,2E-1
θ (º)	76,4 ± 7,2E-3	93,5 ± 5E-3	93,7 ± 1,7	93,5 ± 4,5E-2	99,4 ± 2,2E-1
ψ (º)	-63,8 ± 1,3E-2	46,7 ± 2,4	133,4 ± 1,5	-176,3 ± 2,9E-2	-45,5 ± 2,7
E_P (mm)	2,2E-2 ± 5,7E-2	5,7E-1 ± 1,7	1,9E-1 ± 7,7E-1	9,5E-2 ± 1,5E-1	1,5E-3 ± 3,4E-3
E_O (º)	8E-3 ± 1,6E-2	3,4E-1 ± 2,4	4E-1 ± 2,7	2,6E-2 ± 5,5E-2	8,2E-1 ± 2,8
E_T	4E-5 ± 9E-5	1,2E-3 ± 5,8E-3	2,1E-3 ± 1,3E-2	2,2E-4 ± 3,9E-4	3,6E-3 ± 1,2E-2

4.8. Mecanismo *discarding* adaptativo

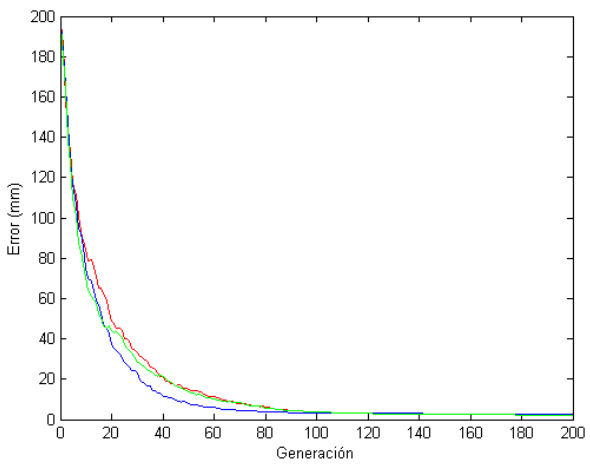
En los resultados obtenidos en el capítulo 4.5.2. los errores medios finales son mayores para dDE aún teniendo mejores tasas de acierto que el algoritmo DE. Para solventar este problema se propone implementar una posible idea de mejora en el algoritmo evolutivo dDE. Esta implementación consiste en: ejecutando el algoritmo evolutivo con el mecanismo *discarding* implementado y un DRATE igual a 5, a partir de un cierto umbral, en este caso la cuando el mejor error total es menor que 0.15, el DRATE se reduce a 0, por tanto para el número de iteraciones restantes el algoritmo converge según el DE clásico. Los resultados obtenidos se muestran a continuación:

Tabla 4.42: Errores finales y tasas de acierto de un robot de 6 GDL utilizando DE, *discarding* y *discarding* adaptativo.

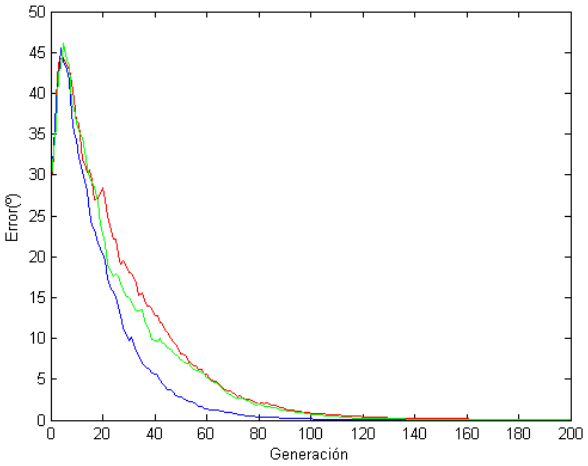
Drate	0.0	5	5 a 0
Tasa 1 (%)	7	45	12
Tasa 2 (%)	64	87	69
Tasa 3 (%)	87	90	88
E_P (mm)	2,4 ± 7,2	2,9 ± 7,8	2,3 ± 6,9
E_O (º)	1,2E-1 ± 7,8E-1	1,7E-3 ± 9,8E-3	8,6E-3 ± 2,9E-2
E_T	2,6E-3 ± 7,3E-3	2,9E-3 ± 7,8E-3	2,3E-3 ± 6,9E-3
Tiempo (s)	1,5	1,6	1,6

Debido a la modificación realizada, podemos observar en la tabla 4.42 que los errores con el mecanismo *discarding* adaptativo son son levemente menores con unas tasas de acierto mejores que el algoritmo DE clásico.

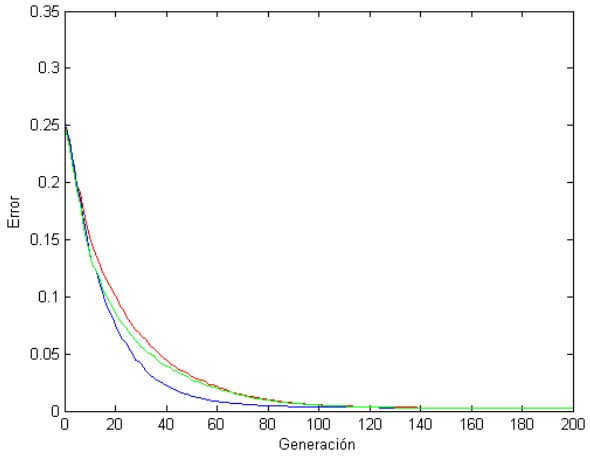
En la figura 4.18 se aprecia como inicialmente el mecanismo *discarding* adaptativo converge del mismo modo que el mecanismo *discarding* pero a medida que avanzan las iteraciones acaba convergiendo del mismo modo que el algoritmo DE clásico.



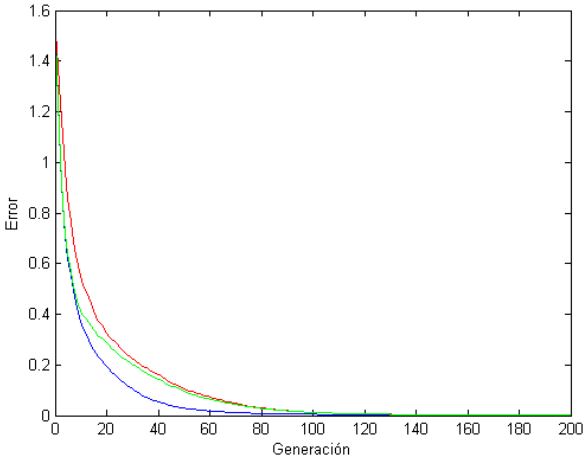
(a) Mejor error medio de posición



(b) Mejor error medio de orientación



(c) Mejor error medio total



(d) Peor error medio total

Figura 4.18: Evolución del error de un robot de 6 GDL utilizando DE, *discarding* y *discarding* adaptativo: DE(rojo), *discarding*(azul) y *discarding* adaptativo(verde).

4.9. Análisis de los resultados

A continuación se muestra un análisis de los resultados obtenidos en las pruebas empíricas descritas anteriormente.

4.9.1. Parámetros de control DE

En los resultados experimentales se observa que los parámetro de control DE influyen del mismo modo en la convergencia del algoritmo tanto para un robot de 3 GDL (D=3) como

uno de 6 GDL (D=6). Por tanto para ambos robots manipuladores, los valores óptimos de los parámetros DE son los siguientes:

- Tamaño de la población (NP) = 10*D.

  Para un valor de NP=5*D, la velocidad de convergencia del algoritmo aumenta, aunque presenta mayor inestabilidad al poder ocasionar una convergencia prematura no-óptima o *stagnation*, como demuestran las altas desviaciones estándar de las tablas 4.9 y 4.11.

  Para valores de NP mayores, NP=60 y NP=120, se obtienen buenos resultados, pero no mejoran los obtenidos para NP=10*D y el tiempo de ejecución del algoritmo es mayor.

  Por tanto NP=10*D es el valor óptimo obtenido, puesto que presenta mayor robustez con tiempo de ejecución del algoritmo pequeño.

- Factor de mutación (F) = 0,5.

  Para F=0,25 se obtienen unas altas desviaciones estándar como se observa en las tablas 4.13 y 4.15, por tanto en algunas de las 100 pruebas el algoritmo ha convergido prematuramente.

  Por otro lado para valores de F próximos a 1, no se obtienen buenos resultados puesto que la variación sobre el vector base no es distinguible. En las figuras 4.6 y 4.7 se observa que el algoritmo no llega a conveger para estos valores.

  Por tanto F=0,5 es el valor óptimo obtenido, puesto que presenta mejores resultados que el resto de valores probados.

- Probabilidad de cruce (CR) = 0,9.

  Para valores de CR igual a 0,3 y 0,6 el algoritmo no llega a converger como se observa en las figuras 4.8 y 4.9.

  En cambio para CR=0,9 y CR=1, la convergencia del algoritmo es exitosa, siendo el valor CR=1 el que mejores resultados obtiene como muestran las tablas 4.17 y 4.19. Aunque en este caso, es más correcto considerar el valor CR=0,9 como óptimo, puesto que para CR=1 el algorimto se convierte en un esquema solo de mutación. Por tanto, se demuestra que la función objetivo del problema de optimización planteado es no separable y con múltiples puntos óptimos.

4.9.2. Parámetros de control *discarding*

En los resultados experimentales se demuestra que el único parámetro que afecta a la velocidad de convergencia del algoritmo, para ambos robots manipuladores, es el DRATE. Ya que los resultados de las pruebas realizadas para diferentes valores de α y β no son concluyentes para poder afirmar que influyen en la velocidad convergencia del algoritmo.

- Para un robot de 3 GDL el valor de DRATE es directamente proporcional a la velocidad de convergencia del algoritmo, como se puede observar en los resultados de la tabla 4.22. Además se obtienen menores errores finales de posición y orientación. Basándonos en los valores escogidos para las pruebas empíricas el valor de DRATE óptimo sería igual a 20.
- Para un robot de 6 GDL un valor de DRATE elevado también aumenta la velocidad de convergencia del algoritmo, pero en este caso, el algoritmo no llega a converger correctamente como se puede observar en la tabla 4.23. Por tanto a mayor DRATE, mayores errores finales de posición y orientación.

4.9.3. *Differential Evolution* vs. DE + *discarding*

El mecanismo *discarding* aumenta la velocidad de convergencia del algoritmo, para ambos robots manipuladores, como se puede observar en las figuras 4.16 y 4.17. A continuación se muestran los porcentajes de mejora de la velocidad de convergencia del algoritmo:

Tabla 4.43: Porcentaje de mejora de un robot de 6 GDL utilizando DE + *discarding*.

Umbral Error	E1	E2	E3
3 GDL	3,5 %	3,5 %	10,3 %
6 GDL	13,8 %	16,4 %	19,4 %

Debido a que la aceleración del algoritmo se produce principalmente en las primeras iteraciones, los porcentajes de mejora son mayores para el umbral de error menos restrictivo.

4.9.4. Mecanismo *discarding* adaptativo

El mecanismo *discarding* adaptativo no es del todo efectivo, pero es un posible acercamiento para hallar la solución al problemas de convergencia final del algoritmo DE utilizando *discarding*

para un robot de 6 GDL.

4.9.5. Espacio de trabajo

Para ambos robots manipuladores el algoritmo DE explora de manera satisfactoria todo el espacio de trabajo.

Capítulo 5

CONCLUSIONES Y TRABAJOS FUTUROS

5.1. Conclusiones

En este proyecto el problema de la cinemática inversa de robots manipuladores se ha planteado como un problema de optimización, más concretamente de minimización, del error de posición y del error de orientación del extremo del robot respecto a unas coordenadas de posición y orientación deseadas dentro del espacio de trabajo del robot. Este problema de optimización se ha resuelto mediante la técnica evolutiva *Differential Evolution*. Esta técnica es de fácil implementación y rápida convergencia frente a otros métodos antes propuestos que son complejos de desarrollar.

La importancia de los parámetros de control del *Differential Evolution* para la correcta convergencia del algoritmo ha sido demostrada en este proyecto. Para la resolución de la cinemática inversa para robots de 3 y 6 GDL, los parámetros de control escogidos deberán ser: NP = 10*D; F = 0,5; CR = 0,9.

Se ha demostrado la eficacia y la eficiencia del algoritmo DE para resolver la cinemática inversa para cualquier punto del espacio de trabajo de los robots manipuladores propuestos.

En este proyecto también se ha incorporado al algoritmo DE una técnica de aceleración de la convergencia, denominada mecanismo *discarding*, y se han evaluado sus parámetros de control, obteniendo como conclusión que el parámetro determinante del *discarding* es el DRATE, el cual acelera la convergencia del algoritmo DE cuanto mayor es su valor. A partir de los resultados

obtenidos, para un robot manipulador de 3 GDL, se ha demostrado la eficacia del mecanismo tanto en la mayor velocidad de convergencia como en los menores errores finales obtenidos. Sin embargo, para un robot manipulador de 6 GDL este mecanismo genera cierta dudas al presentar mayores errores en la solución. Por ello, en este proyecto se ha presentado una idea de un método adaptativo del mecanismo *discarding*, el cual puede llegar a solucionar el problema anteriormente comentado.

5.2. Trabajos futuros

A continuación se enumeran los posibles trabajos futuros a realizar.

- Comparar los resultados obtenidos con otros métodos de inversión cinemática, como por ejemplo, técnicas de inversión cinemática basadas en la matriz Jacobiana.
- Proponer e implementar mejoras al Algoritmo Evolutivo para el cálculo de la cinemática inversa de robots manipuladores redundantes.
- Utilizar el algoritmo DE para resolver la cinemática inversa de robots manipuladores móviles con un mayor número de grados de libertad.
- Proponer un esquema modificado del algoritmo DE basado en técnicas conocidas de *niching* y *clustering* para obtener las múltiples soluciones del problema de cinemática inversa de un robot manipulador.
- Implementar la solución propuesta en los robots manipuladores móviles y comprobar los resultados obtenidos en las simulaciones.
- Mejorar el método adaptativo *discarding*, siendo una posible idea disminuir el valor del DRATE progresivamente.

APÉNDICES

APÉNDICE A

PRESUPUESTO DEL PROYECTO

En este apéndice se presentan justificados los costes globales de la realización del Proyecto Fin de Carrera. Tales costes se imputan a gastos de personal y de material. La realización del proyecto e puede dividir en varias fases:

1. Familiarización y primer contacto con el problema. Lectura de la documentación de Differential Evolution y Data Sheet de los robots, comprensión del funcionamiento del algoritmo.

2. Desarrollo del banco de pruebas. Programación en Matlab.

3. Experimentación. Pruebas realizadas para el análisis de los resultados.

4. Redacción de la memoria. La memoria se desarrolla utilizando un editor de Latex.

Dentro de los gastos personales hay tener en cuenta también el tiempo invertido por el tutor en el proyecto.

En la tabla A.1 se muestran las diferentes fases del proyecto y las horas dedicadas en cada una de ellas, así como las horas dedicadas por el tutor.

El total de horas dedicadas al proyecto son 890 horas, considerando 60€/hora para las horas dedicadas por el tutor, y 30€/hora las horas dedicadas por el alumno como ingeniero en prácticas. Se obtiene un total de 29.700€en gastos personales.

En la tabla A.2 se muestran los gastos materiales, que incluyen un portátil, acceso a Internet durante 1 año y 6 meses, y los gastos de encuadernación y otros gastos.

El presupuesto final del proyecto se muestra en la tabla A.3.

Tabla A.1: *Horas dedicadas.*

Fase	Horas	Precio	Importe
Documentación	100	30€	3.000€
Desarrollo	350	30€	10.500€
Experimentación	90	30€	2.700€
Memoria	250	30€	7.500€
Tutor	100	60€	6.000€
Total			29.700€

Tabla A.2: *Gastos materiales.*

Concepto	Importe
Portátil	800€
Acceso Internet	936€
Otros Gastos	400€
Total	2.136€

Tabla A.3: *Presupuesto.*

Concepto	Importe
Gastos Personales	29.700€
Gastos Materiales	2.136€
Base Imponible	31.836€
I.V.A. (18 %)	5.730,5€
Total	37.566,48€

Bibliografía

[1] J. Baillieul. Kinematic programming alternatives for redundant manipulators. *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, 2:722–728, Marzo 1985.

[2] A. Barrientos. *Fundamentos de Robótica*. McGraw-Hill, 1997.

[3] T. Bäck, U. Hammel, and H.-P. Schwefel. Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1:3–17, Abril 1997.

[4] L. Carvalho and E. Gaspar. The solution of the inverse kinematic problem of robot arms with neural networks. *XI Brazilian Congress on Mechanical Engineering COBEM - Brasil*, 1991.

[5] S. Das and P. N. Suganthan. Differential evolution: A survey of the state-of-the-art. *Ieee Transactions On Evolutionary Computation*, 15(1), Febrero 2011.

[6] J. Denavit and R. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *SME Journal of Applied Mechanics*, 25:215–221, June 1955.

[7] A. e. a. Hasan. An adaptive-learning algorithm to solve the inverse kinematics problem of a 6 d.o.f serial robot manipulator. *Advances in Engineering Software*, page 37, 2006.

[8] J. Gibbs. Easy inverse kinematics using genetic programming. *Proceeding GECCO '96 Proceedings of the First Annual Conference on Genetic Programming*, page 422, Julio 1996.

[9] A. Goldenberg and D. Lawrence. A generalized solution to the inverse kinematics of robotics manipulators. *ASME J. Dynamic Systems, Measurement and Control*, 107:103–106, Marzo 1985.

[10] A. Guez and Z. Ahmad. Solution to the inverse kinematics problem in robotics by neural networks. *Neural Networks, 1988., IEEE International Conference on*, 2:617–624, 1988.

[11] W. Khalil and E. Dombre. *Modelling, Identification and Control of Robots*. Hermes Penton Ltd., 2002.

[12] A. Khawaja, M. Rahman, and M. Wagner. Inverse kinematics of arbitrary robotic manipulators using genetic algorithms. In J. Lenarcic and M. L. Husty, editors, *Advances in Robot Kinematics: Analysis and Control*. Kluwer Academic Publishers, 1998.

[13] S. Kim and J.-H. Kim. Optimal path generation of a redundant manipu- lator with evolutionary programming. *Industrial Electronics, Control, and Instrumentation, 1996., Proceedings of the 1996 IEEE IECON 22nd International Conference on, 3*, pages 1909–1914, 1996.

[14] K. Kinoshita, H. M. M. Izumida, and K. Murakami. Estimation of inverse kinematics model by forward-propagation rule with a high-order term. *Control, Automation, Robotics and Vision, 2006. ICARCV ?06. 9th International Conference on*, pages 1–6, 2006.

[15] A. Koivo. *Fundamentals for Control of Robotic Manipulators*. John Wiley and Sons, 1989.

[16] R. Koker. Reliability-based approach to the inverse kinematics solution of robots using elman?s networks. *Engineering Applications of Artificial Intelligence*, 18(6):685–693, 2005.

[17] Y. Kuroe, Y. Nakai, and T. Mori. A new neural network approach to the inverse kinematics problem in robotics. *Motion Control Proceedings, 1993., Asia-Pacific Workshop on Advances in*, pages 112–117, 1993.

[18] J. Lampinen. A bibliography of differential evolution algorithm. *Technical Report, Lappeenranta University of Technology, Department of Information Technology, Laboratory of Information Processing*, 1999.

[19] C. Lee and M. Ziegler. Geometric approach in solving inverse kinematics of puma robots. *IEEE Transactions on Aerospace and Electronic Systems*, 1984.

[20] J. Lenarcic. An efficient numerical approach for calculating the inverse kinematics for robot manipulators. *Robotica*, 3(1):21–26, 1985.

[21] E. D. Luis Felipe Giraldo and G. Castellanos. Cinemática inversa de un brazo robot utilizando algoritmos genéticos. *Avances en Sistemas e Informática*, 3(1):29–34, Junio 2006.

[22] Z. Mao and T. C. Hsia. Obstacle avoidance inverse kinematics solution of redundant robots by neural networks. *Robotica*, 15(1):3–10, 1997.

[23] F. Martín, L. Moreno, S. Garrido, and D. Blanco. High-accuracy global localization filter for three-dimensional environments. *Robotica*, pages 1–16, 2011.

[24] P. Martín and J. d. R. Millán. Robot arm reaching through neural inversions and reinforcement learning. *Robotics and Autonomous Systems*, 31(4):227–246, 2000.

[25] E. Oyama, N. Chong, A. Agah, T. Maeda, and S. Tachi. Inverse kinematics learning by modular architectore neural networks with performance perdiction networks. *IEEE International Conference on Robotics and Automation*, page 1006, 2001.

[26] J. Parker, A. Khoogar, and D. Goldberg. Inverse kinematics of redundant robots using genetic algorithms. *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, 1:271–276, Mayo 1989.

[27] R. P. Paul and H. Zhang. Comnputionally efficient kinematics for manipulators with spherical wrists based on the homogeneous transformation representation. *The International Journal of Robotics Research*, 1986.

[28] D. Pieper. *The Kinematics of Manipulators Under Computer Control.* PhD thesis, Stanford University, 1968.

[29] K. Price. Genetic annealing, a versatile approach to global optimization that's based on a combination of genetic algorithms and simulated-annealing techniques. *Dr. Dobb's Journal*, Octubre 1994.

[30] K. Price. New ideas on optimization. *D. Corne, M. Dorigo, and F. Glover (Eds.)*, pages 79–106, 1999.

[31] K. Price, R. Storn, and J. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization.* Springer, 2005.

[32] C. C. W. M. S. Tabandeh. A genetic algorithm approach to solve for multiple solutions of inverse kinematics using adaptive niching and clustering. *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 1815–1822, 2006.

[33] L. Sciavico and B. Siciliano. *Modeling and control of robot manipulators (J. M. M. Lynn B.Cox, Ed.)*. McGraw-Hill, 1996.

[34] R. Storn. On the usage of differential evolution for function optimization. *Smith, M.H., Lee, M.A., Keller, J., Yen, J. (eds.) Proceedings of the 1996 biennial conference of the North American fuzzy information processing society*, pages 519–523, Junio 1996.

[35] R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, Marzo 1995.

[36] R. Storn and K. Price. Minimizing the real functions of the ICEC contest by differential evolution. *Proceedings of the 1996 IEEE international conference on evolutionary computation, Nagoya, Japan*, pages 842–844, 1996.

[37] R. Storn and K. Price. Differential evolution: a simple evolution strategy for fast optimization. *Dr. Dobbes Journal 22*, pages 18–22, Abril 1997.

[38] S. Tabandeh, W. W. Melek, and M. C. Clark. An adaptive niching genetic algorithm approach for generating multiple solutions of serial manipulator inverse kinematics with applications to modular robots. *Robotica*, 28(4):493–507, 2010.

[39] J. Teo. Differential evolution with self-adaptative populations. *Proceedings of the 9th International Conference on Knowledge-Based Intelligent Information and Engineering Systems*, pages 1284–1290, September 2005.

[40] C. G. Uzcátegui. Generación de caminos óptimos para robots manipuladores usando differential evolution. Master's thesis, Universidad Carlos III de Madrid, Septiembre 2008.

[41] J. C. Villela. *Mecanismo de selección y control de una hiperheurística basada en Evolución Diferencial para optimización en espacios restringidos.* PhD thesis, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, Octubre 2010.

[42] C. W. Wampler. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *Systems, Man and Cybernetics, IEEE Transactions on*, 16(1):93–101, Enero 1986.

[43] L. Wang and C. Chen. A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *Robotics and Automation, IEEE Transactions on*, 7(4):489–499, Agosto 1991.

[44] D. Whitney. Resolved motion rate control of manipulators and human prosthesis. *Man-Machine Systems, IEEE Transactions on*, 10(2):47–53, Junio 1969.

[45] W. A. Wolovich and H. Elliott. A computational technique for inverse kinematics. *Decision and Control, 1984. The 23rd IEEE Conference on*, 23:1359–1363, Diciembre 1984.

[46] D. Zaharie. Critical values for the control parameters of differential evolution algorithms. *Matouek, R., Omera, P. (eds.) Proceedings of MENDEL 2002, 8th international conference on soft computing, Brno, Czech Republic. Brno University of Technology, Faculty of Mechanical Engineering*, pages 62–67, Junio 2002.