UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

ESCUELA DE POSGRADO

MAESTRÍA EN CIENCIAS MENCIÓN INFORMÁTICA

TESIS

APLICACIÓN DE LA INTELIGENCIA ARTIFICIAL EN LA RESOLUCIÓN DE CINEMÁTICA INVERSA EN BRAZOS ROBÓTICOS DE N GRADOS DE LIBERTAD

PARA OPTAR AL GRADO ACADÉMICO DE MAESTRO EN CIENCIAS MENCION INFORMATICA

AUTOR

Br. JOSÉ MAURO PILLCO QUISPE

ASESOR:

Dr. LUIS BELTRAN PALMA TTITO

(ORCID: 0000-0002-0950-5369)

CUSCO –PERÚ

2024

INFORME DE ORIGINALIDAD

(Aprobado por Resolución Nro.CU-303-2020-UNSAAC)

El que suscribe, **Asesor** del trabajo de investigación/tesis titulada: Aplicación de la inteligencia artificial en la resolución de cinemática inversa en brazos robóticos de n grados de libertad

presentado por: José Mauro Pillco Quispe con DNI Nro.: 23861067 presentado por: ..... con DNI Nro.: ..... para optar el título profesional/grado académico de Maestro en Ciencias Mención Informática

Informo que el trabajo de investigación ha sido sometido a revisión por 2 veces, mediante el Software Antiplagio, conforme al Art. 6° del ***Reglamento para Uso de Sistema Antiplagio de la UNSAAC*** y de la evaluación de originalidad se tiene un porcentaje de 8 %.

Evaluación y acciones del reporte de coincidencia para trabajos de investigación conducentes a grado académico o título profesional, tesis

Porcentaje	Evaluación y Acciones	Marque con una (X)
Del 1 al 10%	No se considera plagio.	X
Del 11 al 30 %	Devolver al usuario para las correcciones.	
Mayor a 31%	El responsable de la revisión del documento emite un informe al inmediato jerárquico, quien a su vez eleva el informe a la autoridad académica para que tome las acciones correspondientes. Sin perjuicio de las sanciones administrativas que correspondan de acuerdo a Ley.	

Por tanto, en mi condición de asesor, firmo el presente informe en señal de conformidad y **adjunto** la primera página del reporte del Sistema Antiplagio.

Cusco, 18 de Marzo de 2025

.....
Firma

Post firma Luis Beltran Pulma Tito

Nro. de DNI 23949672

ORCID del Asesor 0000-0002-0950-5369

Se adjunta:

1. Reporte generado por el Sistema Antiplagio.
2. Enlace del Reporte Generado por el Sistema Antiplagio: **oid:** 27259-440449977

José Mauro Pillco Quispe

APLICACIÓN DE LA INTELIGENCIA ARTIFICIAL EN LA RESOLUCIÓN DE CINEMÁTICA INVERSA EN BRAZOS ROBÓTI...

Universidad Nacional San Antonio Abad del Cusco

Detalles del documento

Identificador de la entrega
trn:oid:::27259:440449977

Fecha de entrega
18 mar 2025, 9:43 a.m. GMT-5

Fecha de descarga
18 mar 2025, 9:50 a.m. GMT-5

Nombre de archivo
tesis JosePillco 3_11_2023_Formato APA_pdf.pdf

Tamaño de archivo
2.5 MB

136 Páginas

23.125 Palabras

129.250 Caracteres

8% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text
- Small Matches (less than 12 words)

Exclusions

- 3 Excluded Matches

Top Sources

6% Internet sources
1% Publications
6% Submitted works (Student Papers)

Integrity Flags

1 Integrity Flag for Review

Hidden Text
618 suspect characters on 14 pages
Text is altered to blend into the white background of the document.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO
ESCUELA DE POSGRADO

INFORME DE LEVANTAMIENTO DE OBSERVACIONES A TESIS

Dra. NELLY AYDE CAVERO TORRE, Directora (e) General de la Escuela de Posgrado, nos dirigimos a usted en condición de integrantes del jurado evaluador de la tesis intitulada **APLICACIÓN DE LA INTELIGENCIA ARTIFICIAL EN LA RESOLUCIÓN DE CINEMÁTICA INVERSA EN BRAZOS ROBÓTICOS DE N GRADOS DE LIBERTAD** del BR. **JOSE MAURO PILLCO QUISPE.** Hacemos de su conocimiento que el sustentante ha cumplido con el levantamiento de las observaciones realizadas por el Jurado el día **DIECINUEVE DE DICIEMBRE DE 2024**.

Es todo cuanto informamos a usted fin de que se prosiga con los trámites para el otorgamiento del grado académico de MAESTRO EN CIENCIAS MENCIÓN INFORMÁTICA.

Cusco, 11 de Marzo de 2025.

DR. EMILIO PALOMINO OLIVERA
Primer Replicante

MGT. HARLEY VERA OLIVERA
Segundo Replicante

DR. JAVIER ARTURO ROZAS HUACHO
Primer Dictaminante

DR. DENNIS IVAN CANDIA OVIEDO
Segundo Dictaminante

DEDICATORIA

A mi querida familia, quienes han sido mi inspiración y mi fortaleza en cada paso de este camino. En especial, dedico este trabajo a ti, mamita, porque fuiste, eres y siempre serás el pilar fundamental de nuestras vidas. Tu amor incondicional, tu sabiduría y tu entrega han sido la fuente inagotable de energía que me ha permitido superar cada obstáculo. Gracias por ser mi soporte, mi guía y mi refugio, siempre con una sonrisa y una palabra de aliento. Esta meta alcanzada es también tuya, por todo lo que has hecho por mí y por nuestra familia.

AGRADECIMIENTOS

Quiero expresar mi más sincero agradecimiento a mi asesor quien, con su conocimiento, paciencia y orientación constante, ha sido una guía fundamental en el desarrollo de esta investigación. Su compromiso con mi crecimiento académico y personal ha sido invaluable.

A mis colegas del Departamento Académico de Informática de la UNSAAC, a quienes considero como una segunda familia, les agradezco profundamente por su apoyo, colaboración y las incontables horas de trabajo compartidas. Sus valiosas contribuciones y su compañerismo me impulsaron a seguir adelante en los momentos más desafiantes. Juntos hemos construido un entorno de aprendizaje que trasciende las aulas, y por eso les estaré eternamente agradecido.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

RESUMEN

La resolución de la cinemática inversa en manipuladores robóticos con múltiples grados de libertad es un desafío en robótica avanzada debido a la multiplicidad de soluciones y la dependencia de condiciones iniciales. Este estudio propone y evalúa tres variantes del algoritmo de Evolución Diferencial (DE) para mejorar la precisión y eficiencia en este problema. Las estrategias incluyen: (1) **DE con CR Dinámico**, con ajuste decreciente de la tasa de recombinación; (2) **DE con CR y F Dinámicos**, con ajustes simultáneos en recombinación y mutación para optimizar la convergencia; y (3) **DE con Movimiento Natural**, inspirado en la biomecánica humana, donde las articulaciones cercanas a la base inician con mayor movilidad, mientras que las distales aumentan su ajuste en generaciones posteriores. El manipulador **PUMA 560** se utilizó como banco de pruebas, midiendo el **error posicional y el tiempo de ejecución**. La variante **DE con CR y F Dinámicos** mostró el mejor desempeño, con un error posicional de **0.002963 mm** y un tiempo total de ejecución de **0.0548 s**, superando al DE clásico. Además, se confirmó que la complejidad computacional **O(G×N×D)** mantiene la escalabilidad del enfoque. En conclusión, las variantes propuestas optimizan la cinemática inversa, mejorando precisión y eficiencia. Se recomienda integrar estos modelos con **visión artificial** e inteligencia artificial para potenciar su aplicabilidad en entornos dinámicos como manufactura, logística y medicina.

PALABRAS CLAVE:

Cinemática inversa, Manipuladores robóticos, Algoritmo de Evolución Diferencial, Cinemática directa, Denavit-Hartenberg, Optimización.

ABSTRACT

The resolution of inverse kinematics in robotic manipulators with multiple degrees of freedom is a challenge in advanced robotics due to the multiplicity of solutions and the dependence on initial conditions. This study proposes and evaluates three variants of the Differential Evolution (DE) algorithm to improve accuracy and efficiency in solving this problem. The strategies include: (1) **DE with Dynamic CR**, with a decreasing adjustment of the recombination rate; (2) **DE with Dynamic CR and F**, with simultaneous adjustments in recombination and mutation to optimize convergence; and (3) **DE with Natural Movement**, inspired by human biomechanics, where joints closer to the base start with greater mobility, while distal joints increase their adjustments in later generations. The **PUMA 560** manipulator was used as a tested, measuring **positional error and execution time**. The **DE with Dynamic CR and F** variant achieved the best performance, with a positional error of **0.002963 mm** and a total execution time of **0.0548 s**, outperforming the classic DE. Furthermore, it was confirmed that the computational complexity **O(G×N×D)** maintains the scalability of the approach. In conclusion, the proposed variants optimize inverse kinematics by improving precision and efficiency. It is recommended to integrate these models with **computer vision** and artificial intelligence to enhance their applicability in dynamic environments such as manufacturing, logistics, and medicine.

KEY WORDS:

Inverse kinematics, Robotic manipulators, Differential Evolution Algorithm, Forward kinematics, Denavit-Hartenberg, Optimization.

INTRODUCCIÓN

En las últimas décadas, los avances en robótica mejoraron significativamente la precisión y el control de los manipuladores robóticos, esenciales en aplicaciones industriales y científicas. Uno de los desafíos más complejos en este ámbito fue la resolución del problema de la cinemática inversa (IKP, por sus siglas en inglés), que implicó calcular los ángulos articulares necesarios para que el efector final de un robot alcanzara una posición y orientación específicas en un espacio tridimensional. Este problema resultó particularmente desafiante en robots con múltiples grados de libertad, como el manipulador PUMA 560, debido a la existencia de múltiples soluciones posibles para una misma configuración deseada.

Los métodos tradicionales para abordar este problema, incluidos los enfoques geométricos, algebraicos y numéricos, presentaron limitaciones significativas. Estos métodos fueron computacionalmente intensivos, dependieron en gran medida de condiciones iniciales específicas y tendieron a converger lentamente hacia soluciones óptimas o, en algunos casos, sub óptimas. Estas restricciones limitaron su aplicabilidad en escenarios dinámicos donde la rapidez y la precisión eran esenciales.

En este contexto, los algoritmos evolutivos, como el algoritmo de Evolución Diferencial (DE), surgieron como soluciones flexibles y eficaces para problemas de optimización complejos, como la cinemática inversa. Inspirados en los principios de la evolución natural, estos algoritmos exploraron simultáneamente múltiples soluciones, convirtiéndose en herramientas prometedoras para problemas donde la búsqueda global y la diversidad de soluciones fueron fundamentales. Estudios previos demostraron que el DE clásico superaba a otros métodos en la resolución del IKP, ofreciendo mejores resultados en términos de rapidez y precisión.

Sin embargo, el DE clásico también presentó limitaciones al enfrentarse a la complejidad del IKP en manipuladores con múltiples grados de libertad. Por esta razón, en esta investigación se propusieron tres variantes del algoritmo DE con el objetivo de mejorar su desempeño en términos de rapidez y precisión:

DE con CR Dinámico Decreciente: donde el parámetro de recombinación (*CR*) disminuyó linealmente a medida que avanzaron las generaciones, mejorando la precisión al final del proceso evolutivo.

DE con CR y F Dinámicos Decrecientes: en esta variante, tanto *CR* como el parámetro de mutación (*F*) decrecieron gradualmente según el avance generacional, equilibrando la exploración y explotación durante la búsqueda.

DE con CR Dinámico y Ajuste Jerárquico de Articulaciones: diseñada para emular el movimiento humano, esta variante ajustó los movimientos de las articulaciones cercanas a la base (*q1* y *q2*), que realizaron movimientos más amplios en las primeras generaciones y los redujeron gradualmente, mientras que las articulaciones distales (*q5* y *q6*) iniciaron con movimientos pequeños que aumentaron conforme se alcanzó la posición objetivo. Este enfoque buscó reflejar el comportamiento humano, donde los movimientos iniciales amplios de la cintura se complementaron con ajustes precisos de las manos al acercarse al objetivo.

El **manipulador PUMA 560** se seleccionó como modelo de referencia debido a su amplio uso en investigaciones robóticas y la extensa documentación disponible sobre sus parámetros y configuraciones. Este robot es uno de los modelos más utilizados en la literatura, lo que lo convirtió en una elección adecuada para validar enfoques innovadores.

El **objetivo** principal de este estudio fue demostrar que las variantes propuestas del algoritmo DE mejoraron significativamente la rapidez y precisión en la resolución del

problema de cinemática inversa en comparación con el DE clásico. Se emplearon los parámetros Denavit-Hartenberg del manipulador PUMA 560 para realizar una validación rigurosa de los resultados obtenidos. Adicionalmente, se analizó la complejidad computacional de los algoritmos propuestos, que se mantuvo lineal con respecto al tamaño de la población, las dimensiones del problema y el número de generaciones ($O(G \times N \times D)$), destacando su escalabilidad y aplicabilidad. Con esta investigación, se buscó contribuir al desarrollo de soluciones más efectivas para aplicaciones avanzadas de robótica en escenarios complejos y dinámicos.

CAPITULO 1: PLANTEAMIENTO DEL PROBLEMA

1. 1. - Situación problemática

En el ámbito de la robótica avanzada, la resolución de la cinemática inversa en brazos robóticos con múltiples grados de libertad se ha establecido como un desafío crucial que exige soluciones innovadoras. Este problema, fundamental para la interacción robótica en entornos dinámicos, implica calcular las configuraciones articulares necesarias para que el efector final alcance una posición y orientación específicas en un espacio tridimensional. Su relevancia ha crecido exponencialmente en la era de la automatización y la inteligencia artificial (IA), donde sectores como la manufactura, la medicina y la exploración espacial dependen de manipuladores robóticos para ejecutar tareas de alta precisión y complejidad.

A medida que aumentan los grados de libertad en los brazos robóticos, la resolución de la cinemática inversa se vuelve exponencialmente más compleja, dando lugar a múltiples soluciones posibles o, en casos específicos, a la ausencia de una solución evidente. Esta complejidad plantea retos significativos desde perspectivas técnicas y operativas, afectando la eficiencia, la adaptabilidad y la capacidad de los manipuladores robóticos para integrarse en escenarios dinámicos. Los métodos tradicionales, como los enfoques geométricos, algebraicos y numéricos, han demostrado ser insuficientes para abordar estos desafíos, ya que dependen de condiciones iniciales específicas, son computacionalmente costosos y tienden a converger a soluciones subóptimas, limitando su aplicabilidad en sistemas avanzados.

En respuesta a estas limitaciones, la integración de inteligencia artificial en la robótica ha impulsado el desarrollo de manipuladores más precisos y autónomos. Según la Federación Internacional de Robótica (IFR, 2024), en 2023 operaban más de 4.28 millones de robots industriales en todo el mundo, representando un incremento del 10 % respecto al año anterior.

Países como Corea del Sur y Singapur lideran esta adopción, reflejando una creciente demanda de robots en sectores como la manufactura y la logística. Este progreso subraya la necesidad de optimizar los sistemas de control de los manipuladores robóticos para garantizar operaciones rápidas, precisas y adaptables en un contexto global altamente competitivo.

Entre las herramientas más prometedoras para abordar el problema de la cinemática inversa se encuentra el algoritmo de Evolución Diferencial (DE), un enfoque basado en principios evolutivos que ha demostrado superar a los métodos tradicionales en rapidez y precisión. No obstante, el DE clásico presenta limitaciones al enfrentarse a escenarios de alta complejidad, lo que motiva la necesidad de mejorar su desempeño mediante ajustes en sus parámetros clave y su comportamiento. En esta investigación, se propusieron tres variantes del DE que optimizan sus capacidades al incluir estrategias dinámicas y patrones inspirados en el movimiento humano, con el objetivo de acelerar la convergencia hacia soluciones precisas y eficientes.

El desarrollo de estas soluciones avanzadas es fundamental para superar los desafíos actuales de la robótica avanzada, permitiendo una integración más efectiva de tecnologías como la visión artificial y algoritmos adaptativos. Estas mejoras garantizarán que los manipuladores robóticos operen de manera autónoma y eficiente en entornos dinámicos, impactando positivamente industrias clave a nivel global. Resolver el problema de la cinemática inversa con rapidez y precisión no solo representa un avance significativo en la investigación robótica, sino también un paso esencial hacia un futuro donde los robots desempeñen un rol central en la transformación tecnológica global.

1. 2. - Formulación del problema

La resolución de la cinemática inversa en manipuladores robóticos con múltiples grados de libertad ha sido identificada como un desafío central en la robótica avanzada. Este problema consiste en determinar las configuraciones articulares necesarias para que el efector final de un robot alcance una posición y orientación específicas en un espacio tridimensional. La complejidad radica en la multiplicidad de soluciones posibles, la redundancia estructural y el elevado costo computacional requerido, especialmente en robots con estructuras avanzadas y un alto número de grados de libertad. Los enfoques tradicionales, como los métodos geométricos, algebraicos y numéricos, han demostrado ser insuficientes para resolver eficazmente esta problemática, debido a su dependencia de condiciones iniciales precisas, su tendencia a converger en soluciones locales subóptimas y su ineficiencia computacional en problemas de alta complejidad.

El avance tecnológico y la creciente integración de sistemas robóticos en sectores estratégicos como la manufactura avanzada, la medicina de precisión y la investigación científica han resaltado la importancia de optimizar la resolución de la cinemática inversa. La capacidad de los manipuladores robóticos para operar de manera precisa, eficiente y segura en entornos dinámicos depende en gran medida de resolver este problema de forma efectiva. Sin embargo, las limitaciones de los métodos tradicionales subrayan la necesidad urgente de desarrollar soluciones innovadoras y robustas que permitan a los sistemas robóticos satisfacer las demandas actuales de precisión y adaptabilidad.

Este desafío afecta directamente a ingenieros, diseñadores de sistemas robóticos, industrias estratégicas y a la comunidad académica dedicada a la robótica inteligente. La resolución eficiente de la cinemática inversa no solo es crucial para garantizar la operatividad de los manipuladores robóticos en aplicaciones críticas, sino que también es fundamental para

impulsar la innovación y la productividad en sectores donde la automatización avanzada y la transformación digital son prioridades estratégicas.

La pertinencia de abordar esta problemática se fundamenta en los avances recientes en inteligencia artificial y algoritmos evolutivos, como el algoritmo de Evolución Diferencial (DE), que ofrecen oportunidades para diseñar soluciones más adaptativas y efectivas. Ignorar esta problemática limitaría significativamente la implementación y el rendimiento de los manipuladores robóticos en aplicaciones críticas, afectando negativamente la innovación tecnológica y la eficiencia operativa de los sectores involucrados.

Esta investigación se enfocó en superar estas limitaciones mediante el desarrollo de tres variantes avanzadas del algoritmo DE, diseñadas específicamente para mejorar la precisión y la rapidez en la resolución de la cinemática inversa. Estas variantes no solo optimizan el desempeño del DE clásico, sino que también incorporan estrategias innovadoras, como ajustes dinámicos de los parámetros de mutación y recombinación, y patrones inspirados en el movimiento humano, para satisfacer las demandas de los sistemas robóticos modernos.

En conclusión, este estudio busca contribuir al avance de la robótica avanzada mediante soluciones eficientes y adaptativas para la resolución de la cinemática inversa. Los resultados de esta investigación tienen el potencial de impactar positivamente en sectores estratégicos, facilitando desarrollos tecnológicos más seguros, precisos y versátiles, fortaleciendo así la aplicabilidad de los manipuladores robóticos en diversas áreas críticas de la sociedad.

Problema general

¿En qué medida el uso de algoritmos evolutivos, específicamente el modelo de Evolución Diferencial con ajuste dinámico y movimiento natural, mejora la precisión (error posicional) y la eficiencia (tiempo de ejecución) en la resolución del problema de cinemática inversa en

manipuladores robóticos con múltiples grados de libertad, en comparación con los métodos tradicionales?

Problemas específicos

- ¿Cuáles son las limitaciones de los métodos tradicionales en términos de precisión y tiempo de ejecución para resolver la cinemática inversa en manipuladores robóticos con múltiples grados de libertad?
- ¿Cómo influyen los parámetros dinámicos de cruzamiento (CR) y mutación (F) en la convergencia y diversidad poblacional del modelo de Evolución Diferencial aplicado al problema de cinemática inversa?
- ¿Qué impacto tiene el modelo Movimiento Natural con CR Dinámico en la precisión (error posicional) y en la adaptabilidad de las configuraciones articulares generadas en manipuladores robóticos con múltiples grados de libertad?
- ¿Cómo se compara el tiempo de ejecución del modelo Movimiento Natural con CR Dinámico frente a los métodos tradicionales, el DE clásico y el modelo CR-F Dinámico en la resolución de la cinemática inversa?

1. 3. - Justificación de la investigación

La resolución eficiente y precisa de la cinemática inversa en manipuladores robóticos con múltiples grados de libertad representa un desafío fundamental en la robótica avanzada. Este problema tiene un impacto directo en la funcionalidad y operatividad de los sistemas robóticos, especialmente en aplicaciones críticas como la manufactura, la medicina y la investigación científica, donde se requieren precisión, rapidez y adaptabilidad en entornos dinámicos y de alta complejidad. La incapacidad para resolver este problema de manera efectiva limita significativamente la implementación de robots en escenarios estratégicos, afectando tanto la innovación tecnológica como la productividad en estos sectores.

En un contexto marcado por la transformación digital y la creciente automatización avanzada, las limitaciones de los métodos tradicionales para resolver la cinemática inversa son cada vez más evidentes. Métodos como los enfoques geométricos, algebraicos y numéricos presentan dependencias significativas de condiciones iniciales específicas, son propensos a converger en soluciones sub óptimas y requieren un alto costo computacional. Estas restricciones dificultan su aplicabilidad en sistemas modernos con múltiples grados de libertad y configuraciones complejas, comprometiendo la eficiencia operativa y la capacidad de adaptación de los manipuladores robóticos en entornos reales.

En respuesta a estas limitaciones, la inteligencia artificial y, en particular, los algoritmos evolutivos, han emergido como herramientas prometedoras para abordar problemas de optimización complejos. Entre ellos, el algoritmo de Evolución Diferencial (DE) se ha destacado por su capacidad para explorar globalmente el espacio de búsqueda y manejar múltiples soluciones. Sin embargo, el DE clásico enfrenta desafíos relacionados con la convergencia y la preservación de la diversidad poblacional, afectando su desempeño en la resolución de la cinemática inversa en escenarios de alta complejidad.

Esta investigación se centró en el desarrollo y evaluación de tres variantes avanzadas del algoritmo DE para superar las limitaciones del modelo clásico. Entre estas, destaca el modelo **CR-F dinámico con movimiento natural**, que ajusta dinámicamente los parámetros de recombinación (*CR*) y mutación (*F*) e introduce una estrategia inspirada en la biomecánica humana. Este enfoque simula patrones de movimiento humano, donde las articulaciones cercanas a la base realizan mayores movimientos iniciales, mientras que las distales incrementan su movilidad hacia las etapas finales del proceso evolutivo. Este diseño busca mejorar la precisión y la eficiencia en la resolución del problema, optimizando el desempeño

en manipuladores robóticos como el **PUMA 560**, un modelo ampliamente utilizado en investigaciones debido a su versatilidad y complejidad.

La relevancia de esta investigación radica en su capacidad para abordar un problema crítico en la robótica avanzada mediante el desarrollo de enfoques innovadores que superen las limitaciones de los métodos tradicionales. Además de contribuir al estado del arte en algoritmos evolutivos, los resultados de este estudio tienen implicaciones prácticas significativas en sectores estratégicos como la manufactura, la logística y la medicina. La capacidad de ofrecer soluciones más rápidas, precisas y adaptativas no solo mejorará la integración de robots en tareas críticas, sino que también fortalecerá su desempeño en aplicaciones autónomas, impactando positivamente el desarrollo tecnológico global.

1. 4. - Objetivos de la investigación

Objetivo general

Mejorar la precisión y la eficiencia en la resolución del problema de cinemática inversa en manipuladores robóticos con múltiples grados de libertad mediante el diseño y evaluación de variantes avanzadas del modelo de Evolución Diferencial con ajuste dinámico de parámetros y movimiento natural, y comparar su desempeño en términos de error posicional y tiempo de ejecución frente a los métodos tradicionales y el DE clásico.

Objetivos específicos

- Analizar las limitaciones de los métodos tradicionales en términos de precisión y tiempo de ejecución para resolver la cinemática inversa en manipuladores robóticos con múltiples grados de libertad.

- Evaluar la influencia de los parámetros dinámicos de cruzamiento (*CR*) y mutación (*F*) en la convergencia y la diversidad poblacional del modelo de Evolución Diferencial aplicado al problema de cinemática inversa.
- Examinar el impacto del modelo Movimiento Natural con CR Dinámico en la precisión y adaptabilidad de las configuraciones articulares generadas para manipuladores robóticos con múltiples grados de libertad.
- Comparar el tiempo de ejecución del modelo Movimiento Natural con CR Dinámico frente a los métodos tradicionales, el DE clásico y el modelo CR-F dinámico.

CAPITULO 2: MARCO TEÓRICO CONCEPTUAL

2. 1. - Bases teóricas

2. 1. 1.- Transformaciones homogéneas

Transformaciones Homogéneas de Matrices 4x4

Las transformaciones homogéneas son una herramienta fundamental en robótica y gráficos computacionales para describir movimientos en el espacio tridimensional. Estas transformaciones se representan mediante matrices homogéneas de 4×4, que permiten combinar rotaciones, traslaciones y escalas de una manera unificada y sencilla de aplicar a puntos o vectores en el espacio (Craig, 2005; Siciliano & Khatib, 2016). A continuación, se explica detalladamente el marco teórico matemático detrás de estas transformaciones.

Representación de un punto en el espacio (coordenadas homogéneas)

Un punto en el espacio tridimensional se puede representar mediante un vector columna en coordenadas cartesianas:

$$\mathrm{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Sin embargo, para poder aplicar las transformaciones homogéneas, es necesario expresar este punto en coordenadas homogéneas. Esto implica agregar una cuarta coordenada (generalmente un 1), de modo que el punto quede representado como:

$$\mathrm{p}_h = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

La última coordenada homogénea (el 1) permite incluir operaciones de traslación y escalamiento, que no son posibles de manera directa en un sistema de coordenadas cartesianas usando matrices de 3x3.

Matriz de transformación homogénea (4x4)

Una matriz homogénea de transformación es una matriz 4x4 que se utiliza para representar de manera compacta una combinación de rotación y traslación en el espacio tridimensional. La estructura general de una matriz homogénea es:

$$\mathrm{T} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

R_{ij}: Los elementos de la submatriz de rotación (matriz 3x3).

t_x, t_y, t_z : Los componentes de la traslación en los ejes x , y , z , respectivamente.

El último renglón es [0 0 0 1] , que garantiza que se trata de una transformación homogénea y permite mantener la dimensionalidad correcta en las operaciones.

Tipos de transformaciones

Traslación

Una traslación en el espacio tridimensional se representa como una matriz homogénea de la siguiente forma:

$$\mathrm{T}_{traslación} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Donde t_x , t_y y t_z son las distancias que se desplaza el punto en los ejes x , y y z , respectivamente.

Rotación

Las rotaciones se pueden aplicar alrededor de los ejes x , y o z mediante matrices de rotación 3x3, que se expanden a matrices homogéneas 4x4. Las rotaciones más comunes son:

- Rotación en torno al eje x:

$$\mathrm{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rotación en torno al eje y:

$$\mathrm{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rotación en torno al eje z:

$$\mathrm{R}_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Composición de transformaciones

La gran ventaja de usar matrices homogéneas es que podemos combinar varias transformaciones (rotaciones y traslaciones) multiplicando sus respectivas matrices. Por ejemplo, si queremos realizar primero una rotación R y luego una traslación T, la transformación total es:

$$\mathrm{T}_{total} = \mathrm{T} \cdot \mathrm{R}$$

Si aplicamos esta transformación a un punto p_h en coordenadas homogéneas, el resultado será un nuevo punto transformado:

$$\mathrm{p'}_h = \mathrm{T}_{total} \cdot \mathrm{p}_h$$

Inversión de transformaciones homogéneas

Para invertir una transformación homogénea, es importante notar que la inversa de una matriz homogénea no es simplemente invertir todos sus componentes, sino que se debe tener cuidado con la rotación y la traslación. La inversa de una matriz homogénea está dada por:

$$\mathrm{T}^{-1} = \begin{bmatrix} R^T & -R^T\mathrm{t} \\ 0 & 1 \end{bmatrix}$$

Donde:

- R^T es la transpuesta de la submatriz de rotación R .

- $-R^T\mathrm{t}$ es la traslación invertida bajo la rotación aplicada.

2. 1. 2.- Cinemática Directa

Problema de Cinemática Directa (Forward Kinematics - FK)

La cinemática directa es una rama fundamental en robótica que se ocupa de calcular la posición y orientación del efector final (la parte del robot que interactúa con su entorno, como una pinza o herramienta) en función de los valores de sus variables articulares, como los ángulos de las articulaciones rotacionales o las distancias de las articulaciones prismáticas.

Dado un robot con n grados de libertad (DOF), la cinemática directa implica encontrar la posición y orientación del efector final en un sistema de coordenadas global o sistema de referencia del mundo, basado en los valores de las articulaciones $q_1, q_2, \dots, q_n$.

El objetivo es determinar las funciones:

$$X = f_x(q_1, q_2, \dots, q_n)$$
$$Y = f_y(q_1, q_2, \dots, q_n)$$
$$Z = f_z(q_1, q_2, \dots, q_n)$$

Donde (X), (Y), y (Z) representan las coordenadas de la posición del efector final.

Además de la posición, si el robot tiene un efector que también puede orientarse (por ejemplo, una pinza que rota), se necesita determinar la orientación en términos de los ángulos de Euler o cuaterniones.

2. 1. 3.- Cinemática inversa

Problema de Cinemática Inversa (Inverse Kinematics -IK)

El **problema de cinemática inversa** en robótica consiste en determinar los valores de las coordenadas articulares de un robot que permiten que su efector final alcance una posición y orientación específicas en el espacio. Este problema implica resolver un conjunto de ecuaciones algebraicas no lineales simultáneas, las cuales pueden tener múltiples soluciones o, en algunos casos, ninguna solución viable. Además, la presencia de singularidades en el modelo matemático puede complicar aún más la obtención de soluciones precisas (Siciliano & Khatib, 2016; Craig, 2005).

En resumen, el problema de cinemática inversa consiste en determinar los valores que deben tomar las articulaciones de un robot para que su efector final (la parte que realiza el trabajo) alcance una posición deseada en el espacio tridimensional, expresada como (X_d, Y_d, Z_d).

. Matemáticamente, esto implica encontrar las funciones inversas:

$$q_1 = Q_1(X_d, Y_d, Z_d)$$
$$q_2 = Q_2(X_d, Y_d, Z_d)$$
$$\vdots$$
$$q_n = Q_n(X_d, Y_d, Z_d)$$

Donde:

- $(q_1, q_2, \dots, q_n)$ son los valores de las articulaciones del robot (ángulos o desplazamientos).
- (X_d, Y_d, Z_d) son las coordenadas deseadas del efector final en el espacio tridimensional.
- $Q_1, Q_2, \dots, Q_n$ son las funciones que permiten calcular los valores articulares necesarios para alcanzar esa posición deseada.

En resumen, la cinemática inversa busca encontrar las funciones $Q_1, Q_2, \dots, Q_n$que determinen cómo deben ajustarse las articulaciones del robot para que el efector final llegue a la posición específica (X_d, Y_d, Z_d).

2. 1. 4.- Modelo matemático Denavit-Hartenberg

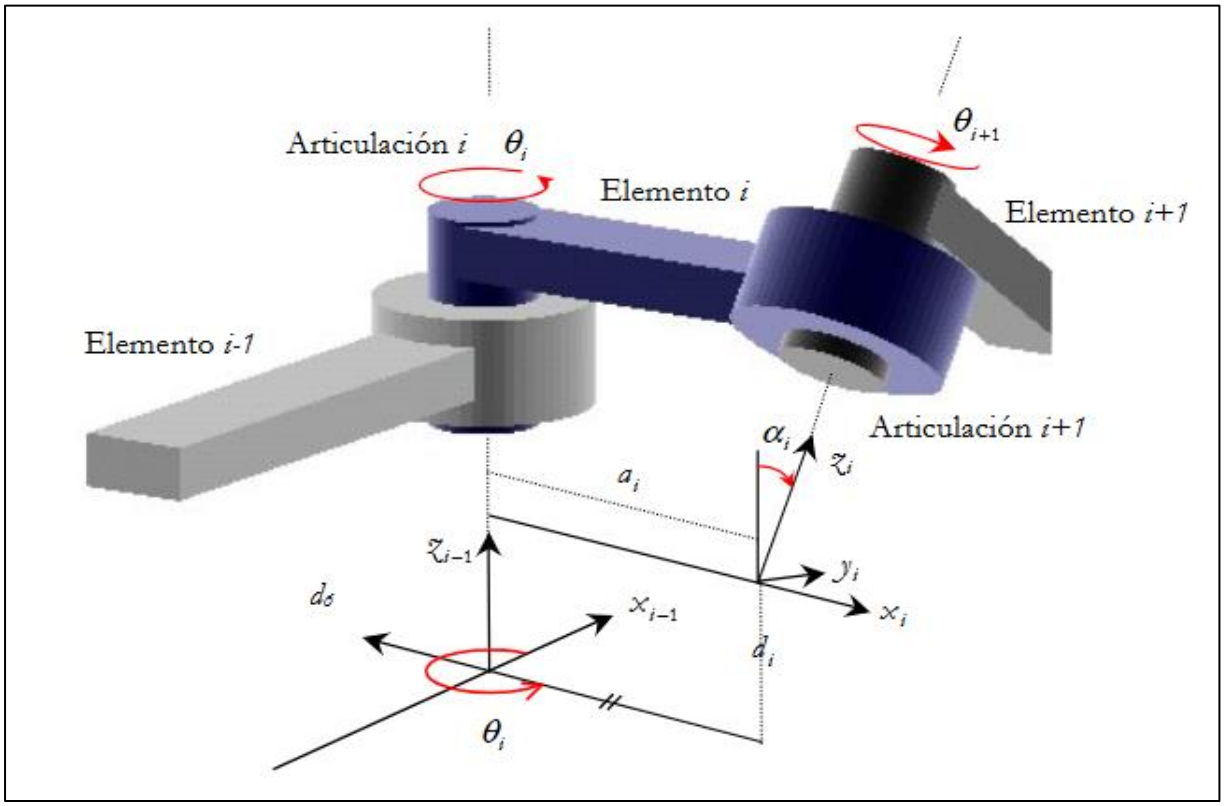
Algoritmo de Denavit y Hartenberg:

El **algoritmo de Denavit-Hartenberg (D-H)** es una técnica utilizada para resolver el problema de cinemática directa en sistemas articulados con N grados de libertad. Este método establece una convención sistemática para asignar sistemas de coordenadas a cada eslabón del robot, lo que permite derivar la matriz de transformación homogénea que relaciona la base del robot con el efector final. A partir de esta matriz, es posible calcular la posición y orientación del efector final en función de los ángulos articulares de las N articulaciones (Craig, 2005; Siciliano & Khatib, 2016).

La representación mediante el modelo D-H proporciona una descripción sistemática y eficiente de la relación geométrica entre los eslabones, siendo esencial para modelar manipuladores robóticos de manera precisa y escalable.

Ilustración 1: Modelo Denavit- Hartenberg

Modelo Denavit- Hartenberg



Paso 1. Establecer el sistema de coordenadas de la base:

- Se debe establecer un sistema de coordenadas ortonormal dextrógiro (x_o, y_o, z_o) en la base del manipulador robótico. El eje z_o se alinea a lo largo del eje de movimiento de la articulación q1 y debe apuntar hacia afuera del brazo.

Paso 2. Establecer los ejes de la articulación:

- El eje z_i de cada articulación debe alinearse con el eje de movimiento de la articulación q_{i+1} Este proceso se repite para cada articulación i, donde $i = 1,2, \dots, n$.

- Nota importante antes del paso 2: Se debe considerar una articulación q_n de rotación imaginaria en la posición del efector final del robot, para la cual también se debe crea un eje de movimiento.

Paso 3. Establecer el origen del sistema de coordenadas i-ésimo:

- El origen del sistema de coordenadas i-ésimo debe localizarse en la intersección de los ejes z_i y z_{i-1} o en la intersección de las normales comunes entre estos dos ejes. z_i y z_{i-1} y el eje z_i

Paso 4. Establecer el eje x_i :

- El eje x_i debe ser perpendicular al eje z_{i-1} y alinearse en la dirección que apunta hacia afuera.

- La ecuación para calcular el eje x_i es:

$$x_i = \pm \frac{(z_{i-1} \times z_i)}{\| z_{i-1} \times z_i \|}$$

- Si los ejes z_{i-1} y z_i son paralelos, el eje x_i se define a lo largo de la normal común entre estos dos ejes.

Paso 5. Establecer el eje y_i:

- El eje y_i se define para completar el sistema de coordenadas dextrógiro.

  El eje y_i se obtiene a partir del producto vectorial de los ejes z_i y x_i, tal como sigue:

$$y_i = \pm \frac{(z_i \times x_i)}{\| z_i \times x_i \|}$$

  Si es necesario, los ejes z_i , y_i y x_i se extienden para realizar los pasos siguientes.

Paso 6. Determinar los Parámetros de Denavit-Hartenberg (D-H)

Determinar los parámetros de D-H del sistema articulado de $i = 1, \dots, n$.

La representación D-H de un elemento rígido depende de 4 parámetros geométricos asociados con cada elemento (ver Ilustración 1):

θ_i : Ángulo de la articulación del eje x_{i-1}al eje x_i respecto del eje z_{i-1} (RMD).

d_i : Distancia desde el origen del sistema de coordenadas $(i-1)$-ésimo hasta la intersección del eje z_{i-1} con el eje x_i , a lo largo del eje z_{i-1} sin salto de línea.

a_i : Distancia de separación desde la intersección del eje z_{i-1} con el eje x_i , hasta el origen del sistema i-ésimo a lo largo del eje x_i (o la distancia más corta entre los ejes z_{i-1} y z_i cuando los ejes de articulación son paralelos) sin salto de línea.

α_i : Ángulo de separación del eje z_{i-1} al eje z_i respecto del eje x_i (RMD) sin salto de línea

NOTA: al finalizar construir una tabla resumen con los parámetros D- del sistema articulado. Columnas los parámetros y filas i=1,2,3 hasta n

Tabla 1:

Ejemplo de la tabla de los parámetros D-H

i	θ_i	d_i	a_i	α_i
$i=1$	$\theta_1=0°$	$d_1=5$	$a_1=0$	$\alpha_1=36°$
$i=2$	$\theta_2=30°$	$d_2=10$	$a_2=3$	$\alpha_2=0°$
Así sucesivamente calcular los demás parámetros.				
$i=n$	$\theta_n=45°$	$d_n=8$	$a_n=6$	$\alpha_n=90°$

La tabla presenta los parámetros D-H utilizados para describir la configuración del sistema articulado, donde i representa el índice de la articulación,

θ_1 el ángulo de rotación,

d_i el desplazamiento a lo largo del eje zi,

a_i la longitud del eslabón y

α_i el ángulo de torsión.

Paso 7. Determinar las matrices de transformación

Una vez establecido el sistema de coordenadas D-H para cada elemento, se puede desarrollar fácilmente una matriz de transformación homogénea que relacione el sistema de coordenadas i-ésimo con el sistema de coordenadas $(i-1)$-ésimo denotado por A_i^{i-1}. En donde A_1^0 relaciona el sistema de coordenadas de la base 0 con el sistema 1. A_2^1 relaciona el sistema de coordenadas de sistema 1 con el sistema 2 y así sucesivamente hasta A_n^{n-1}.

Para lograr expresar un punto cualquiera del sistema de coordenadas i-ésimo en términos del sistema de coordenadas anterior se propone las siguientes transformaciones.

- Traslación en el eje z_{i-1}: Corresponde a la distancia d_i .
- Rotación en torno al eje z_{i-1}: Corresponde al ángulo θ_i .
- Traslación en el eje x_i: Corresponde a la distancia a_i.
- Rotación en torno al eje x_i: Corresponde al ángulo α_i .

Obteniendo:

$$A_{i-1}^{-1} = T_{z,d}T_{z,\theta}UT_{x,a}T_{x,\alpha}$$

$$T_{z,d}T_{z,\theta} = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & d_i\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\cos\theta_i & -\sin\theta_i & 0 & 0\\ \sin\theta_i & \cos\theta_i & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$T_{x,a}T_{x,\alpha} = \begin{bmatrix}1 & 0 & 0 & a_i\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\alpha_i & -\sin\alpha_i & 0\\ 0 & \sin\alpha_i & \cos\alpha_i & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Resultado final de la matriz homogénea

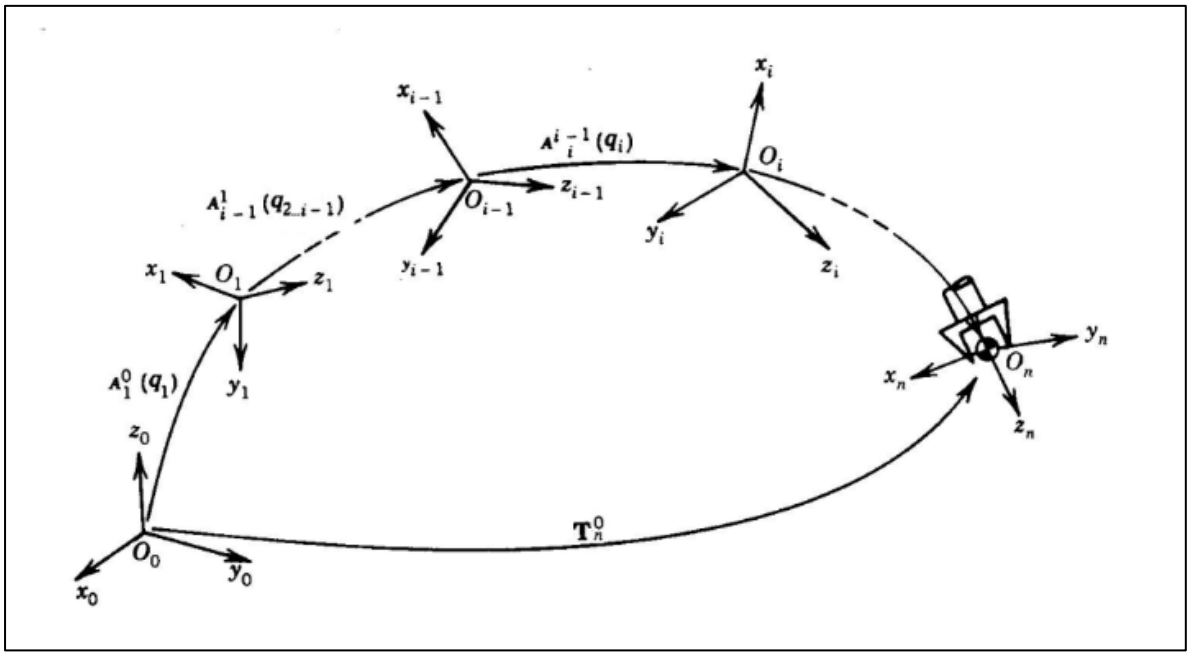
$$A_i^{i-1} = \begin{bmatrix}\cos\theta_i & -\cos\alpha_i\sin\theta_i & \sin\alpha_i\sin\theta_i & a_i\cos\theta_i\\ \sin\theta_i & \cos\alpha_i\cos\theta_i & -\sin\alpha_i\cos\theta_i & a_i\sin\theta_i\\ 0 & \sin\alpha_i & \cos\alpha_i & d_i\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Se debe tener en cuenta que la multiplicación de matrices 4x4 no es conmutativa, es importante considerar secuencias de transformaciones cuando se requiera obtener matrices

compuestas y siempre al final considerar las matrices de traslación en el análisis de la transformación final.

Ilustración 2: Ubicación del efector final

Ubicación del efector final



Utilizando la notación de Denavit y Hartenberg se puede expresar la posición y orientación del efector final del manipulador en función del desplazamiento de sus articulaciones. El desplazamiento articular es un ángulo θ_i o una distancia d_i , dependiendo del tipo de articulación. Para cualquier manipulador, de tipo cadena serial con n articulaciones, se representa la solución cinemática directa a partir de sus matrices de transformación homogénea de la siguiente forma:

$$T_n^0 = A_1^0 \cdot A_2^1 \cdot \ldots \cdot A_n^{n-1},$$

$$T_n^0 = \prod_{j=1}^{n} A_j^{j-1}.$$

Donde,

$$T_n^0 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

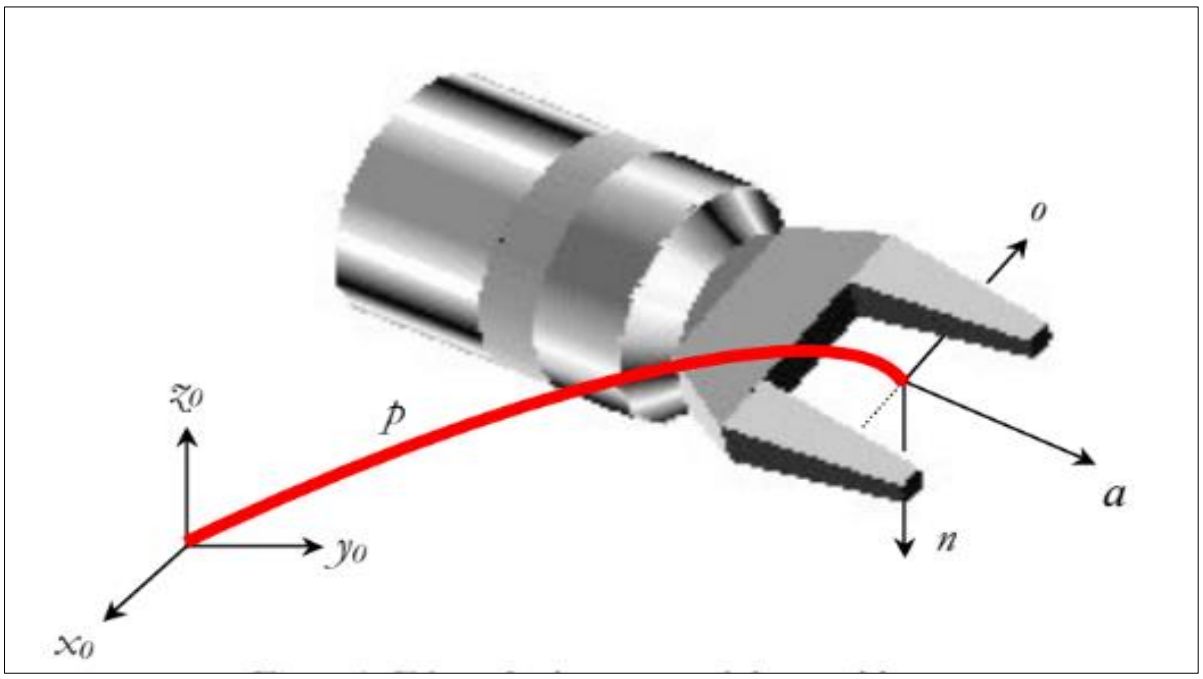
La matriz T_n^0 representa la orientación y posición del efector final del manipulador robótico.

Donde:

- $p_{x,y,z}$: (**SOLUCIÓN**) Vector de posición que apunta desde el origen del sistema de coordenadas de la base hasta el origen del sistema de coordenadas n-ésimo.
- a : Vector de aproximación de la mano. Apunta desde el origen del sistema de coordenadas de la mano hacia el origen del sistema de coordenadas asociado al objeto a manipular o una posición espacial predefinida. (Suele ser localizado en el punto central de los dedos cerrados).
- o : Vector de orientación de la mano. Apunta en la dirección del movimiento de los dedos (caso ejes paralelos).
- n : Vector normal de la mano. Suponiendo una mano tipo mordaza paralela, ortogonal a los dedos del brazo del robot.

Ilustración 3:

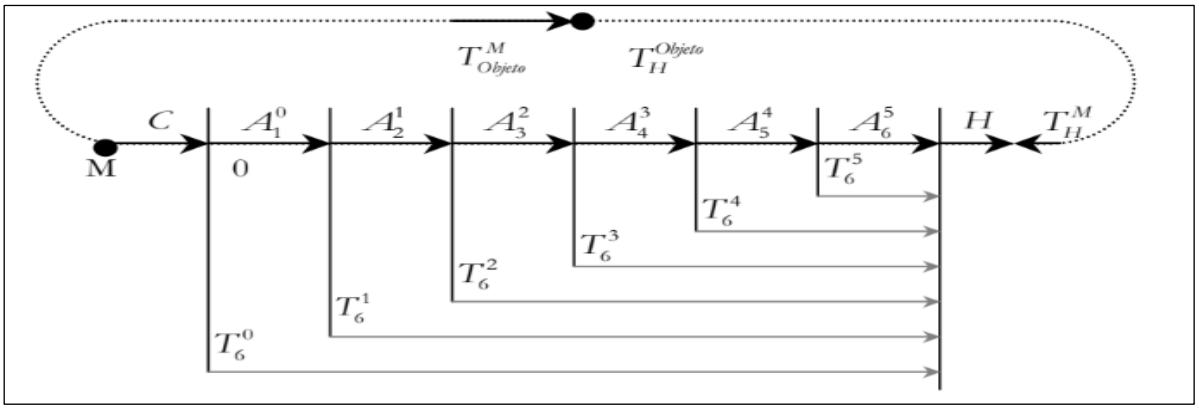
Efector final con dedos paralelos



Como se ha descrito, la composición de diversas y distintas transformaciones pueden ayudar a definir la localización espacial del efector final. Esta relación puede ser descrito mediante un grafo de transformación como:

Ilustración 4:

Grafo de las trasformaciones



2. 1. 5.- Algoritmos Evolutivos

Los **algoritmos evolutivos (AE)** son técnicas de optimización inspiradas en los principios de la evolución biológica, como la selección natural, la mutación, la recombinación y la supervivencia del más apto. Estos métodos se utilizan ampliamente para resolver problemas complejos en los que el espacio de búsqueda es altamente no lineal o presenta múltiples óptimos locales, como en la cinemática inversa de manipuladores robóticos con múltiples grados de libertad (Carmona & Fernández, 2020; Holland, 1975).

Fundamentos de los Algoritmos Evolutivos

Representación de las Soluciones

Los AE operan sobre una población P de soluciones candidatas $\mathbf{x}_i$, donde cada solución se representa como un vector en un espacio multidimensional:

$$\mathbf{x}_i = [x_{i1}, x_{i2}, \dots, x_{id}] \quad \text{con} \quad i = 1,2, \dots, N$$

Aquí, d es la dimensionalidad del problema (grados de libertad) y N es el tamaño de la población.

Función de Evaluación

La calidad de una solución $\mathbf{x}_i$ se mide mediante una función de fitness $f(\mathbf{x}_i)$, que refleja el ajuste de la solución al problema. En el caso de la cinemática inversa, el fitness se define típicamente como el error posicional ϵ:

$$f(\mathbf{x}_i) = \epsilon(\mathbf{x}_i) = \| \mathbf{P}_{\text{actual}} - \mathbf{P}_{\text{objetivo}} \|_2$$

Donde $\mathbf{P}_{\text{actual}}$ es la posición del efector final obtenida a partir de $\mathbf{x}_i$, y $\mathbf{P}_{\text{objetivo}}$ es la posición deseada.

2. 1. 6.- Algoritmo de Evolución Diferencial (DE)

El algoritmo de Evolución Diferencial (DE) es un método evolutivo diseñado para abordar problemas de optimización en espacios continuos, destacándose por su estructura simple y su eficacia. Se fundamenta en tres operaciones básicas: la mutación, que introduce variaciones en las soluciones; la recombinación, que combina características de distintas soluciones; y la selección, que determina cuáles serán conservados para las iteraciones futuras (Storn & Price, 1997)

a) Mutación

La mutación genera un vector mutante $\mathbf{v}_i$ a partir de tres individuos seleccionados aleatoriamente $\mathbf{x}_{r1}, \mathbf{x}_{r2}, \mathbf{x}_{r3}$ de la población:

$$\mathbf{v}_i = \mathbf{x}_{r1} + F \cdot (\mathbf{x}_{r2} - \mathbf{x}_{r3})$$

Aquí, $F \in [0,2]$ es el **factor de escala**, que controla la amplitud de las modificaciones introducidas.

b) Recombinación

El vector mutante $\mathbf{v}_i$ se combina con el vector objetivo $\mathbf{x}_i$ para producir un vector de prueba $\mathbf{u}_i$. Esto se realiza mediante un operador de recombinación binomial:

$$u_{ij} = \begin{cases} v_{ij} & \text{si } r_{ij} \leq CR \\ x_{ij} & \text{en otro caso} \end{cases}$$

Donde $CR \in [0,1]$ es la **tasa de recombinación**, y r_{ij} es un número aleatorio uniformemente distribuido en $[0,1]$.

c) Selección

La selección elige entre $\mathbf{x}_i$ y $\mathbf{u}_i$ para formar la población de la próxima generación:

$$\mathbf{x}_i^{(\text{nuevo})} = \begin{cases} \mathbf{u}_i & \text{si } f(\mathbf{u}_i) \leq f(\mathbf{x}_i) \\ \mathbf{x}_i & \text{en otro caso} \end{cases}$$

2. 1. 7.- Variantes Propuestas del DE

a) DE con CR Dinámico

El parámetro CR se ajustó dinámicamente para decrecer linealmente con las generaciones:

$$CR^{(t)} = CR_{\text{inicial}} \cdot \left(1 - \frac{t}{G_{\max}}\right)$$

Donde t es el número de generación actual y $G_{\max}$ es el número total de generaciones. Este enfoque busca mejorar la explotación en etapas avanzadas del proceso.

b) DE con CR y F Dinámicos

Ambos parámetros, CR y F, se ajustaron simultáneamente:

$$CR^{(t)} = CR_{\text{inicial}} \cdot \left(1 - \frac{t}{G_{\max}}\right), \quad F^{(t)} = F_{\text{inicial}} \cdot \left(1 - \frac{t}{G_{\max}}\right)$$

Esta variante busca equilibrar la exploración y explotación a lo largo de las generaciones.

c) DE con CR Dinámico y Movimiento Natural

Inspirado en la biomecánica humana, esta variante ajustó la movilidad de las articulaciones q_i basándose en su proximidad a la base o al efector final:

$$\Delta q_i^{(t)} = \begin{cases} \Delta q_i^{(\text{máx})} \cdot \left(1 - \frac{t}{G_{\max}}\right) & \text{si } i \text{ es proximal} \\ \Delta q_i^{(\text{máx})} \cdot \frac{t}{G_{\max}} & \text{si } i \text{ es distal} \end{cases}$$

2. 1. 8.- Métricas de Evaluación

Las métricas de evaluación se definen como medidas cuantitativas que permiten valorar el rendimiento y la calidad de los resultados obtenidos por un algoritmo o modelo. Estas métricas facilitan la comparación objetiva entre distintos métodos, identificando aspectos como la precisión, el tiempo de ejecución, la robustez y la eficiencia en la solución de problemas (Eiben & Smith, 2003).

a) Precisión (Error Posicional)

El error posicional se utilizó para evaluar la precisión de las configuraciones generadas:

$$\epsilon = \| \mathbf{P}_{\text{actual}} - \mathbf{P}_{\text{objetivo}} \|_2$$

b) Eficiencia (Tiempo de Ejecución)

El tiempo promedio por generación y el tiempo total de ejecución se calcularon como indicadores de eficiencia:

$$T_{\text{promedio}} = \frac{T_{\text{total}}}{G_{\max}}$$

2. 1. 9.- Movimiento Natural en Robótica

El concepto de movimiento natural en robótica se inspira en la biomecánica humana, donde los patrones jerárquicos de movilidad son fundamentales para realizar movimientos precisos y eficientes. Este enfoque considera que las articulaciones cercanas a la base de un sistema robótico, como un brazo manipulador, deben realizar movimientos más amplios al inicio de una tarea. A medida que se avanza hacia las etapas finales del movimiento, las articulaciones distales, como las muñecas o dedos, incrementan su movilidad para realizar ajustes finos y garantizar la precisión requerida.

Fundamentos Biomecánicos del Movimiento Natural

En la biomecánica humana, el movimiento de las extremidades se rige por un esquema jerárquico en el que las articulaciones proximales, como la cintura y los hombros, inician el movimiento con grandes desplazamientos. Posteriormente, las articulaciones distales, como los codos, muñecas y dedos, completan la tarea con ajustes más precisos. Este patrón permite:

- **Optimización de energía:** Minimizar el esfuerzo total requerido al distribuir el movimiento jerárquicamente.
- **Precisión en la etapa final:** Garantizar que las extremidades distales realicen ajustes precisos.
- **Fluidez del movimiento:** Coordinar múltiples grados de libertad para evitar movimientos abruptos o descoordinados.

Aplicación en Sistemas Robóticos

El movimiento natural se implementa en robótica para emular los patrones biomecánicos mencionados. En manipuladores robóticos, como el PUMA 560, este enfoque es especialmente útil para tareas que requieren alta precisión y adaptabilidad. La implementación incluye:

- **Articulaciones proximales (base y hombros):** Realizan movimientos amplios al inicio de la trayectoria, orientando el brazo hacia la dirección general del objetivo.
- **Articulaciones distales (muñecas y pinzas):** Ajustan la posición y orientación fina del efector final al final del movimiento.

Modelado Matemático del Movimiento Natural

Para integrar el movimiento natural en algoritmos de control, se definen dinámicamente las amplitudes de movimiento de las articulaciones:

$$\Delta q_i^{(t)} = \begin{cases} \Delta q_i^{(\text{máx})} \cdot \left(1 - \dfrac{t}{G_{\max}}\right) & \text{si } i \text{ es proximal} \\ \Delta q_i^{(\text{máx})} \cdot \dfrac{t}{G_{\max}} & \text{si } i \text{ es distal} \end{cases}$$

donde:

- $\Delta q_i^{(t)}$ es la amplitud de movimiento de la articulación i en la generación t.
- $G_{\max}$ es el número total de generaciones.
- $\Delta q_i^{(\text{máx})}$ es la amplitud máxima de movimiento permitida para la articulación i.

Ventajas del Movimiento Natural en Robótica

- **Reducción de errores acumulativos:** Al limitar los movimientos distales iniciales, se reduce la probabilidad de errores acumulativos en las etapas finales.
- **Suavidad en las trayectorias:** La jerarquización del movimiento garantiza que las trayectorias sean continuas y libres de discontinuidades.
- **Similitud con movimientos humanos:** Este enfoque permite que los robots interactúen con entornos diseñados para humanos, mejorando la integración en tareas colaborativas.
- **Precisión Mejorada**: La coordinación jerárquica de las articulaciones permite realizar movimientos más precisos, especialmente en tareas que requieren alta destreza.
- **Eficiencia Energética**: Al distribuir el esfuerzo de manera óptima entre las diferentes articulaciones, se reduce el consumo de energía durante la operación.
- **Interacción Segura**: La emulación de movimientos humanos facilita una interacción más segura y efectiva entre robots y humanos en entornos colaborativos.

Casos de Estudio y Validación

Investigaciones actuales han explorado la integración de señales electromiográficas (EMG) para controlar brazos robóticos, permitiendo movimientos más intuitivos y naturales basados en la actividad muscular del usuario. Además, se han desarrollado sistemas biomiméticos que replican la estructura y función del brazo humano, mejorando la adaptabilidad y desempeño de los robots en diversas tareas.

En resumen, la incorporación de principios de movimiento natural inspirados en la biomecánica humana en el diseño y control de brazos robóticos representa un avance significativo hacia la creación de sistemas más eficientes, precisos y seguros, capaces de interactuar de manera efectiva en entornos compartidos con humanos.

Modelos Matemáticos en Robótica

Para formalizar el comportamiento en sistemas robóticos, especialmente en manipuladores, se emplean diversos modelos matemáticos que permiten describir y predecir su funcionamiento. A continuación, se detallan los principales modelos utilizados:

Modelos Cinemáticos

Los modelos cinemáticos se centran en la relación entre las variables articulares (ángulos de las articulaciones) y la posición y orientación del efector final del robot. Estos modelos no consideran las fuerzas ni las aceleraciones, sino únicamente las posiciones y velocidades.

- **Cinemática Directa**: Determina la posición y orientación del efector final a partir de los ángulos de las articulaciones. Matemáticamente, se representa mediante una función f que relaciona el vector de ángulos articulares **q** con la posición **p** y orientación **R** del efector:

$$\mathbf{T} = f(\mathbf{q})$$

donde **T** es la matriz de transformación homogénea que combina **p** y **R** .

- **Cinemática Inversa**: Calcula los ángulos de las articulaciones necesarios para alcanzar una posición y orientación deseadas del efector final. Se expresa como la función inversa de la cinemática directa:

$$\mathbf{q} = f^{-1}(\mathbf{T})$$

Modelos Dinámicos

Los modelos dinámicos incorporan las fuerzas y torques que actúan sobre el robot, considerando su masa, inercia y aceleraciones. Estos modelos son esenciales para el control preciso del movimiento y se basan en las ecuaciones de Newton-Euler o en la formulación de Lagrange.

Ecuaciones de Movimiento: Describen la relación entre los torques $\boldsymbol{\tau}$ aplicados en las articulaciones y las aceleraciones articulares $\ddot{\mathbf{q}}$:

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q})$$

donde:

- $\mathbf{M}(\mathbf{q})$: Matriz de inercia.
- $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$: Matriz de Coriolis y centrífuga.
- $\mathbf{G}(\mathbf{q})$: Vector de fuerzas gravitatorias.

Modelos de Control

Para garantizar que el robot siga una trayectoria deseada, se implementan leyes de control basadas en modelos matemáticos. Uno de los controladores más comunes es el PID

(Proporcional-Integral-Derivativo), que ajusta los torques en función del error entre la posición deseada y la posición actual.

Control PID: Calcula el torque $\tau(t)$ como:

$$\tau(t) = K_p e(t) + K_i \int e(t)\, dt + K_d \frac{de(t)}{dt}$$

donde:

$e(t)$: Error en el tiempo t .

K_p, K_i, K_d : Ganancias proporcional, integral y derivativa, respectivamente.

Modelos de Planificación de Trayectorias

Estos modelos determinan la secuencia óptima de movimientos que el robot debe seguir para realizar una tarea específica, considerando restricciones físicas y ambientales.

Interpolación Polinómica: Genera trayectorias suaves utilizando polinomios que interpolan puntos clave (waypoints). Por ejemplo, una interpolación cúbica para la posición $q(t)$ se define como:

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

donde los coeficientes a_i se determinan en función de las condiciones iniciales y finales de posición y velocidad.

Modelos de Percepción y Censado: Para interactuar con su entorno, los robots utilizan sensores cuyos datos se modelan matemáticamente para interpretar la realidad circundante. Por ejemplo, la transformación de coordenadas de un sensor de visión puede representarse como:

$$\mathbf{p}_{\text{cámara}} = \mathbf{R}_{\text{sensor}} \mathbf{p}_{\text{mundo}} + \mathbf{t}_{\text{sensor}}$$

Donde: $\mathbf{R}_{\text{sensor}}$ y $\mathbf{t}_{\text{sensor}}$

Son la rotación y traslación del sensor respecto al marco de referencia del mundo.

La integración de estos modelos matemáticos permite diseñar, analizar y controlar sistemas robóticos de manera efectiva, asegurando su correcto desempeño en diversas aplicaciones.

2. 1. 10.- Métodos tradiciones de resolución

Solución Matemática del Problema de Cinemática Inversa para un Brazo Robótico de n Grados de Libertad

El problema de la cinemática inversa (IK, por sus siglas en inglés) para un brazo robótico de n grados de libertad consiste en determinar los valores de las articulaciones $\mathbf{q} = [q_1, q_2, \dots, q_n]$ que permitan al efector final alcanzar una posición $\mathbf{p}_{\text{deseada}} \in \mathbb{R}^3$ y orientación $\mathbf{R}_{\text{deseada}} \in SO(3)$.

Matemáticamente, el problema se plantea como:

$$\mathbf{T}(\mathbf{q}) = \begin{bmatrix} \mathbf{R}(\mathbf{q}) & \mathbf{p}(\mathbf{q}) \\ 0 & 1 \end{bmatrix}, \quad \mathbf{T}(\mathbf{q}) = \mathbf{T}_{\text{deseada}}$$

donde:

- $\mathbf{T}(\mathbf{q})$ es la matriz de transformación homogénea calculada a partir de los parámetros articulares $\mathbf{q}$
- $\mathbf{T}_{\text{deseada}}$ es la matriz objetivo deseada, que incluye la posición $\mathbf{p}_{\text{deseada}}$ y la orientación $\mathbf{R}_{\text{deseada}}$

. Resolver este problema implica obtener $\mathbf{q}$ de manera exacta o aproximada. A continuación, se presentan las formulaciones matemáticas para los enfoques tradicionales.

2. 1. 10. 1.- Método Geométrico.

El método geométrico utiliza relaciones trigonométricas entre las articulaciones del manipulador y su configuración espacial. Se descompone el problema en la resolución de la posición y la orientación.

Resolución de la posición:

La posición del efector final está dada por:

$$\mathbf{p}(\mathbf{q}) = \begin{bmatrix} x(\mathbf{q}) \\ y(\mathbf{q}) \\ z(\mathbf{q}) \end{bmatrix}$$

Para un manipulador con n eslabones, se utiliza la cinemática directa acumulativa:

$$\mathbf{p}(\mathbf{q}) = \sum_{i=1}^{n} \mathbf{R}_{0,i}\,(\mathbf{q}) \cdot \mathbf{d}_i$$

donde: -

- $\mathbf{R}_{0,i}(\mathbf{q})$ es la matriz de rotación acumulada hasta el i -ésimo eslabón.
- $\mathbf{d}_i$ es el vector desplazamiento del i -ésimo eslabón en coordenadas locales.

Para cada articulación q_i:

- Si es rotacional: q_i se relaciona con $\cos(q_i)$ y $\sin(q_i)$ en coordenadas.

- Si es prismático: q_i se relaciona con la extensión lineal.

Ejemplo en 2D:

$$x = l_1\cos(q_1) + l_2\cos(q_1 + q_2), \quad y = l_1\sin(q_1) + l_2\sin(q_1 + q_2)$$

Resolviendo para q_1 y q_2:

$$q_2 = \arccos\left(\frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1 l_2}\right)$$

$$q_1 = \arctan\left(\frac{y}{x}\right) - \arctan\left(\frac{l_2 \sin(q_2)}{l_1 + l_2 \cos(q_2)}\right)$$

2. 1. 10. 2.- Resolución de la orientación.

Para un manipulador con 3 grados de libertad en el sistema homogéneo se tiene la siguiente estructura donde cada matriz está en función de cada articulación.

$$\mathbf{R}(\mathbf{q}) = \mathbf{R}_z(q_1) \cdot \mathbf{R}_y(q_2) \cdot \mathbf{R}_z(q_3)$$

La matriz $\mathbf{R}(\mathbf{q})$ tiene elementos:

$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} c_1c_2c_3 - s_1s_3 & -c_1c_2s_3 - s_1c_3 & c_1s_2 \\ s_1c_2c_3 + c_1s_3 & -s_1c_2s_3 + c_1c_3 & s_1s_2 \\ -s_2c_3 & s_2s_3 & c_2 \end{bmatrix}$$

Donde $c_i = \cos(q_i)$, $s_i = \sin(q_i)$. Resolver estas ecuaciones para q_1, q_2, q_3 requiere descomponer $\mathbf{R}_{\text{deseada}}$ en términos conocidos.

2. 1. 10. 3.- Método Algebraico.

El método algebraico reformula las ecuaciones homogéneas para obtener relaciones polinomiales entre los ángulos articulares teniendo un sistema de ecuaciones.

$$\mathbf{T}_{0,n} = \prod_{i=1}^{n} \mathbf{A}_i\,(q_i)$$

Donde $\mathbf{A}_i(q_i)$ son matrices de transformación homogénea individuales según Denavit-Hartenberg (D-H):

$$\mathbf{A}_i(q_i) = \begin{bmatrix} \cos(q_i) & -\sin(q_i)\cos(\alpha_i) & \sin(q_i)\sin(\alpha_i) & a_i\cos(q_i) \\ \sin(q_i) & \cos(q_i)\cos(\alpha_i) & -\cos(q_i)\sin(\alpha_i) & a_i\sin(q_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Para resolver q_i , se igualan los términos de la matriz $\mathbf{T}_{0,n}$ con $\mathbf{T}_{\text{deseada}}$. Esto genera sistemas de ecuaciones no lineales que se resuelven algebraicamente.

2. 1. 10. 4.- Método Numérico.

El método numérico utiliza optimización para minimizar la diferencia entre la posición y orientación alcanzadas y las deseadas ecuaciones correspondientes.

3.1 Función de error:

$$E(\mathbf{q}) = \| \mathbf{p}(\mathbf{q}) - \mathbf{p}_{\text{deseada}} \|^2 + \lambda \| \mathbf{R}(\mathbf{q}) - \mathbf{R}_{\text{deseada}} \|^2$$

Donde λ pondera el término de orientación.

Jacobiano:

Se usa el Jacobiano $\mathbf{J}(\mathbf{q})$, que relaciona los cambios en los ángulos articulares con la posición y orientación:

$$\Delta\mathbf{p} = \mathbf{J}(\mathbf{q})\Delta\mathbf{q}$$

Resolviendo iterativamente:

$$\mathbf{q}_{k+1} = \mathbf{q}_k - \mathbf{J}^{\dagger}(\mathbf{q})\Delta\mathbf{p}$$

Donde $\mathbf{J}^{\dagger}(\mathbf{q})$ es la pseudo-inversa del Jacobiano.

2. 2. - Marco conceptual

Modelo matemático del problema de cinemática directa para PUMA 560

El problema de cinemática directa en el manipulador PUMA 560 consiste en determinar la posición y orientación del efector final, dadas las configuraciones articulares q= [q1, q2, q3, q4, q5, q6] Este manipulador es un robot de 6 grados de libertad (DOF), ampliamente documentado y utilizado en robótica.

Ilustración 5:

Parámetros D-H del puma 560

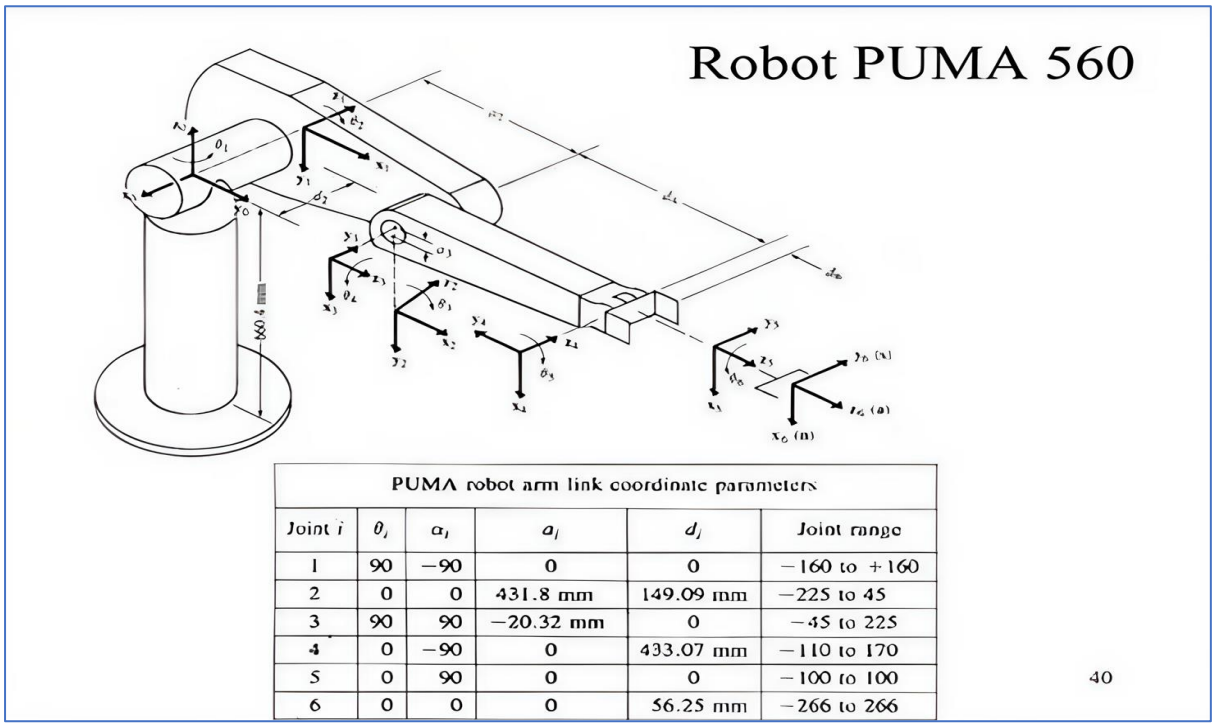


Ilustración 6:

Los 6 grados de libertad del Puma 560

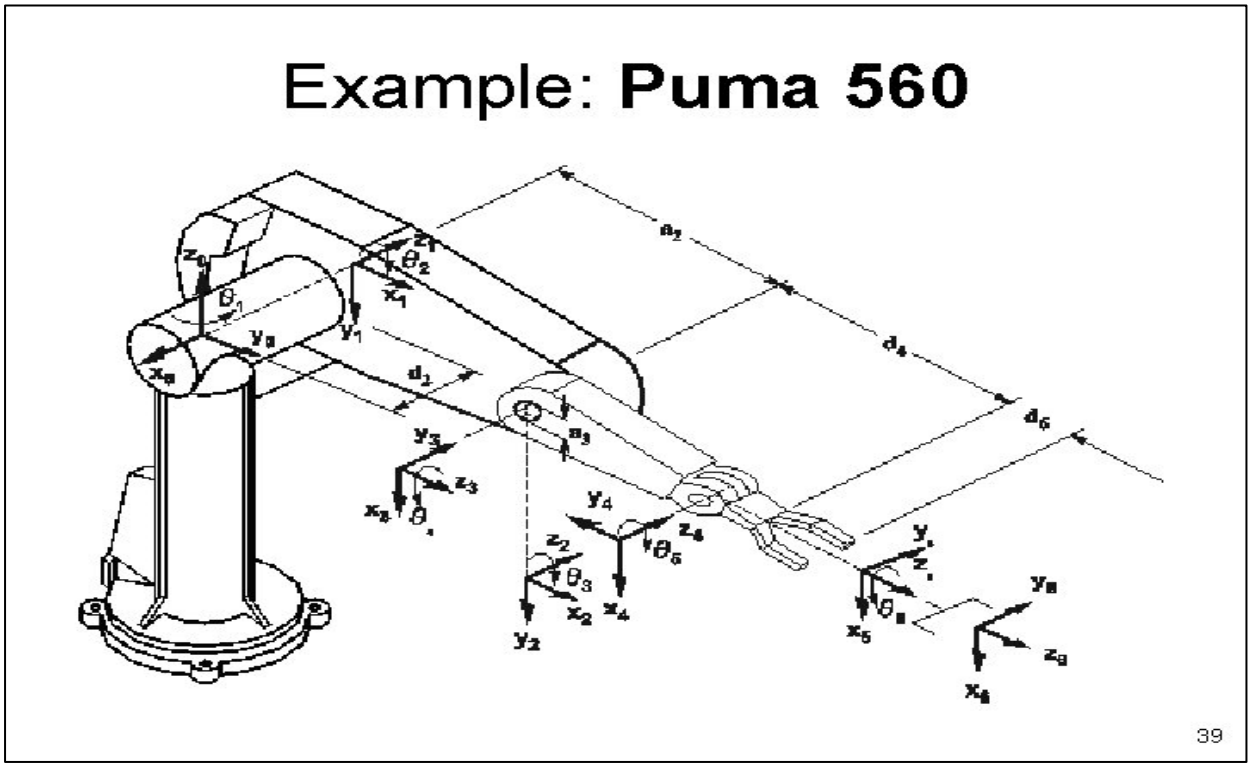


Tabla 2:

Parámetros D-H del manipulador PUMA 560

i	θ_i(°)	α_i(°)	a_i(mm)	d_i(mm)	Rango (°)
i=1	90	-90	0	0	-160 a +160
i=2	0	0	431.8	149.09	-225 a +45
i=3	90	90	-20.32	0	-45 a +225
i=4	0	-90	0	433.07	-110 a +170
i=5	0	90	0	0	-100 a +100
i=6	0	0	0	56.25	-266 a +266

Desarrollo de las Matrices de Transformación Homogénea

Vamos a desarrollar las 6 matrices de transformación homogénea A_i usando los parámetros de Denavit-Hartenberg de la Tabla anterior para el robot PUMA 560. Cada matriz de transformación homogénea A_i tiene la siguiente estructura:

$$A_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\sin(\alpha_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Parámetros (D-H) y Matriz de Transformación Homogénea

Parámetros D-H para $i = 1$:

$\theta_1 = 90^\circ$

$\alpha_1 = -90^\circ$

$a_1 = 0$ (la distancia entre los ejes)

$d_1 = 0$ (la distancia desde el origen)

Estos parámetros son utilizados en la matriz de transformación homogénea general de Denavit-Hartenberg:

Matriz de transformación homogénea sin evaluar las funciones trigonométricas:

$$A_1 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1)\cos(\alpha_1) & \sin(\theta_1)\sin(\alpha_1) & a_1\cos(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1)\cos(\alpha_1) & -\cos(\theta_1)\sin(\alpha_1) & a_1\sin(\theta_1) \\ 0 & \sin(\alpha_1) & \cos(\alpha_1) & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriz de transformación homogénea con los valores trigonométricos sin multiplicar:

$$A_1 = \begin{bmatrix} \cos(90°) & -\sin(90°)\cos(-90°) & \sin(90°)\sin(-90°) & 0 \\ \sin(90°) & \cos(90°)\cos(-90°) & -\cos(90°)\sin(-90°) & 0 \\ 0 & \sin(-90°) & \cos(-90°) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriz con los valores trigonométricos sin multiplicar:

$$A_1 = \begin{bmatrix} 0 & -(1)(0) & (1)(-1) & 0 \\ (1) & 0(1) & -(0)(-1) & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriz resultante después de las operaciones:

$$A_1 = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

De esta forma, los parámetros D-H para $i = 1$ generan la matriz de transformación homogénea A_1 , que describe la relación de posición y orientación entre los dos primeros eslabones del manipulador robótico.

Parámetros D-H para $i = 2$:

$\theta_2 = 0°$

$\alpha_2 = 0°$

$a_2 = 431.8$ mm (la distancia entre los ejes)

$d_2 = 149.09$ mm (la distancia desde el origen)

Estos parámetros son utilizados en la matriz de transformación homogénea general de Denavit-Hartenberg:

a) Matriz de transformación homogénea general:

$$A_2 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2)\cos(\alpha_2) & \sin(\theta_2)\sin(\alpha_2) & a_2\cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2)\cos(\alpha_2) & -\cos(\theta_2)\sin(\alpha_2) & a_2\sin(\theta_2) \\ 0 & \sin(\alpha_2) & \cos(\alpha_2) & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

b) Matriz de transformación homogénea sin evaluar las funciones trigonométricas:

$$A_2 = \begin{bmatrix} \cos(0°) & -\sin(0°)\cos(0°) & \sin(0°)\sin(0°) & 431.8\cos(0°) \\ \sin(0°) & \cos(0°)\cos(0°) & -\cos(0°)\sin(0°) & 431.8\sin(0°) \\ 0 & \sin(0°) & \cos(0°) & 149.09 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

c) Matriz con los valores trigonométricos sin multiplicar:

$$A_2 = \begin{bmatrix} 1 & -(0)(1) & (0)(1) & 431.8(1) \\ (0) & 1(1) & -(1)(0) & 431.8(0) \\ 0 & 0 & 1 & 149.09 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

d) Matriz resultante después de las operaciones:

$$A_2 = \begin{bmatrix} 1 & 0 & 0 & 431.8 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 149.09 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Parámetros D-H para $i = 3$:

$\theta_3 = 90°$

$\alpha_3 = 90°$

$a_3 = -20.32$ mm (la distancia entre los ejes)

$d_3 = 0$ mm (la distancia desde el origen)

Estos parámetros son utilizados en la matriz de transformación homogénea general de Denavit-Hartenberg:

a) Matriz de transformación homogénea general:

$$A_3 = \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3)\cos(\alpha_3) & \sin(\theta_3)\sin(\alpha_3) & a_3\cos(\theta_3) \\ \sin(\theta_3) & \cos(\theta_3)\cos(\alpha_3) & -\cos(\theta_3)\sin(\alpha_3) & a_3\sin(\theta_3) \\ 0 & \sin(\alpha_3) & \cos(\alpha_3) & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

b) Matriz de transformación homogénea sin evaluar las funciones trigonométricas:

$$A_3 = \begin{bmatrix} \cos(90°) & -\sin(90°)\cos(90°) & \sin(90°)\sin(90°) & -20.32\cos(90°) \\ \sin(90°) & \cos(90°)\cos(90°) & -\cos(90°)\sin(90°) & -20.32\sin(90°) \\ 0 & \sin(90°) & \cos(90°) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

c) Matriz con los valores trigonométricos sin multiplicar:

$$A_3 = \begin{bmatrix} 0 & -(1)(0) & (1)(1) & -20.32(0) \\ 1 & 0(0) & -(0)(1) & -20.32(1) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

d) Matriz resultante después de las operaciones:

$$A_3 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & -20.32 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Parámetros D-H para $i = 4$:

$\theta_4 = 0°$

$\alpha_4 = -90°$

$a_4 = 0$ mm (la distancia entre los ejes)

$d_4 = 433.07$ mm (la distancia desde el origen)

Estos parámetros son utilizados en la matriz de transformación homogénea general de Denavit-Hartenberg:

a) Matriz de transformación homogénea general:

$$A_4 = \begin{bmatrix} \cos(\theta_4) & -\sin(\theta_4)\cos(\alpha_4) & \sin(\theta_4)\sin(\alpha_4) & a_4\cos(\theta_4) \\ \sin(\theta_4) & \cos(\theta_4)\cos(\alpha_4) & -\cos(\theta_4)\sin(\alpha_4) & a_4\sin(\theta_4) \\ 0 & \sin(\alpha_4) & \cos(\alpha_4) & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

b) Matriz de transformación homogénea sin evaluar las funciones trigonométricas:

$$A_4 = \begin{bmatrix} \cos(0°) & -\sin(0°)\cos(-90°) & \sin(0°)\sin(-90°) & 0\cos(0°) \\ \sin(0°) & \cos(0°)\cos(-90°) & -\cos(0°)\sin(-90°) & 0\sin(0°) \\ 0 & \sin(-90°) & \cos(-90°) & 433.07 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

c) Matriz con los valores trigonométricos sin multiplicar:

$$A_4 = \begin{bmatrix} 1 & -(0)(0) & (0)(-1) & 0(1) \\ 0 & 1(0) & -(1)(-1) & 0(0) \\ 0 & -1 & 0 & 433.07 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

d) Matriz resultante después de las operaciones:

$$A_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 433.07 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Parámetros D-H para $i = 5$:

$\theta_5 = 0°$

$\alpha_5 = 90°$

$a_5 = 0$ mm (la distancia entre los ejes)

$d_5 = 0$ mm (la distancia desde el origen)

Estos parámetros son utilizados en la matriz de transformación homogénea general de Denavit-Hartenberg:

a) Matriz de transformación homogénea general:

$$A_5 = \begin{bmatrix} \cos(\theta_5) & -\sin(\theta_5)\cos(\alpha_5) & \sin(\theta_5)\sin(\alpha_5) & a_5\cos(\theta_5) \\ \sin(\theta_5) & \cos(\theta_5)\cos(\alpha_5) & -\cos(\theta_5)\sin(\alpha_5) & a_5\sin(\theta_5) \\ 0 & \sin(\alpha_5) & \cos(\alpha_5) & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

b) Matriz de transformación homogénea sin evaluar las funciones trigonométricas:

$$A_5 = \begin{bmatrix} \cos(0°) & -\sin(0°)\cos(90°) & \sin(0°)\sin(90°) & 0\cos(0°) \\ \sin(0°) & \cos(0°)\cos(90°) & -\cos(0°)\sin(90°) & 0\sin(0°) \\ 0 & \sin(90°) & \cos(90°) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

c) Matriz con los valores trigonométricos sin multiplicar:

$$A_5 = \begin{bmatrix} 1 & -(0)(0) & (0)(1) & 0(1) \\ 0 & 1(0) & -(1)(1) & 0(0) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

d) Matriz resultante después de las operaciones:

$$A_5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Parámetros D-H para $i = 6$:

$\theta_6 = 0°$

$\alpha_6 = 0°$

$a_6 = 0$ mm (la distancia entre los ejes)

$d_6 = 56.25$ mm (la distancia desde el origen)

Estos parámetros son utilizados en la matriz de transformación homogénea general de Denavit-Hartenberg:

a) Matriz de transformación homogénea general:

$$A_6 = \begin{bmatrix} \cos(\theta_6) & -\sin(\theta_6)\cos(\alpha_6) & \sin(\theta_6)\sin(\alpha_6) & a_6\cos(\theta_6) \\ \sin(\theta_6) & \cos(\theta_6)\cos(\alpha_6) & -\cos(\theta_6)\sin(\alpha_6) & a_6\sin(\theta_6) \\ 0 & \sin(\alpha_6) & \cos(\alpha_6) & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

b) Matriz de transformación homogénea sin evaluar las funciones trigonométricas:

$$A_6 = \begin{bmatrix} \cos(0°) & -\sin(0°)\cos(0°) & \sin(0°)\sin(0°) & 0\cos(0°) \\ \sin(0°) & \cos(0°)\cos(0°) & -\cos(0°)\sin(0°) & 0\sin(0°) \\ 0 & \sin(0°) & \cos(0°) & 56.25 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

c) Matriz con los valores trigonométricos sin multiplicar:

$$A_6 = \begin{bmatrix} 1 & -(0)(1) & (0)(0) & 0(1) \\ 0 & 1(1) & -(1)(0) & 0(0) \\ 0 & 0 & 1 & 56.25 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

d) Matriz resultante después de las operaciones:

$$A_6 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 56.25 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplicación de las matrices para obtener la transformación total:

La transformación total T_{total} es el producto de las matrices de transformación $A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6$: $T_{\text{total}} = A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6$

Paso 1: Multiplicación de $A_1 \cdot A_2$

Las matrices A_1 y A_2 son las siguientes:

$$A_1 = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_2 = \begin{bmatrix} 1 & 0 & 0 & 431.8 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 149.09 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Realizando la multiplicación $A_1 \cdot A_2$:

$$A_1 \cdot A_2 = \begin{bmatrix} 0 & 0 & -1 & -149.09 \\ 1 & 0 & 0 & 431.8 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Paso 2: Multiplicación de $(A_1 \cdot A_2) \cdot A_3$

Ahora, tomamos el resultado anterior y lo multiplicamos por A_3:

$$A_3 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & -20.32 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplicamos:

$$\begin{bmatrix} 0 & 0 & -1 & -149.09 \\ 1 & 0 & 0 & 431.8 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & -20.32 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & -149.09 \\ 0 & 0 & 1 & 431.8 \\ -1 & 0 & 0 & 20.32 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Paso 3: Multiplicación de $\big((A_1 \cdot A_2) \cdot A_3\big) \cdot A_4$

Tomamos el resultado anterior y lo multiplicamos por A_4 :

$$A_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 433.07 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplicamos:

$$\begin{bmatrix} 0 & -1 & 0 & -149.09 \\ 0 & 0 & 1 & 431.8 \\ -1 & 0 & 0 & 20.32 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 433.07 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 & -149.09 \\ 0 & -1 & 0 & 864.87 \\ -1 & 0 & 0 & 20.32 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Paso 4: Multiplicación de $\Big(\big((A_1 \cdot A_2) \cdot A_3\big) \cdot A_4\Big) \cdot A_5$

Tomamos el resultado anterior y lo multiplicamos por A_5 :

$$A_5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplicamos:

$$\begin{bmatrix} 0 & 0 & -1 & -149.09 \\ 0 & -1 & 0 & 864.87 \\ -1 & 0 & 0 & 20.32 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & -149.09 \\ 0 & 0 & 1 & 864.87 \\ -1 & 0 & 0 & 20.32 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Paso 5: Multiplicación de $\Big(\Big(\big((A_1 \cdot A_2) \cdot A_3\big) \cdot A_4\Big) \cdot A_5\Big) \cdot A_6$

Finalmente, tomamos el resultado anterior y lo multiplicamos por A_6 :

$$A_6 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 56.25 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplicamos:

$$\begin{bmatrix} 0 & -1 & 0 & -149.09 \\ 0 & 0 & 1 & 864.87 \\ -1 & 0 & 0 & 20.32 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 56.25 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & -149.09 \\ 0 & 0 & 1 & 921.12 \\ -1 & 0 & 0 & 20.32 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformación total.

El resultado de la multiplicación de las seis matrices es:

$$T_{\text{total}} = \begin{bmatrix} 0 & -1 & 0 & -149.09 \\ 0 & 0 & 1 & 921.12 \\ -1 & 0 & 0 & 20.32 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Esta matriz de transformación total describe la posición y orientación del efector final del robot en el espacio respecto a la base.

Determina la solución al problema de Cinemática Directa:

Mediante el algoritmo Denavit y Hartenberg. El cual indica que al multiplicar todas las matrices de transformación parcial $T = A_1(q_1) * A_2(q_2) * A_3(q_3) * A_4(q_4) * A_5(q_5) * A_6(q_6)$ la matriz resultante determina solución al problema de cinemática directa para el manipulador PUME560 y la posición del efector final estar determinado por la expresión:

- X= T (1,4)
- Y= T (1,4)
- Z= T (1,4)

Resultado de $\mathbf{A_1(q_1) * A_2(q_2)}$:

$$\begin{bmatrix} -\cos(q_2)\sin(q_1) & \sin(q_1)\sin(q_2) & \cos(q_1) & \frac{14909\cos(q_1)}{100} - \frac{2159\cos(q_2)\sin(q_1)}{5} \\ \cos(q_1)\cos(q_2) & -\cos(q_1)\sin(q_2) & \sin(q_1) & \frac{14909\sin(q_1)}{100} + \frac{2159\cos(q_1)\cos(q_2)}{5} \\ -\sin(q_2) & -\cos(q_2) & 0 & \frac{-2159\sin(q_2)}{5} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Resultado de $\mathbf{A_1(q_1) * A_2(q_2) * A_3(q_3)}$:

3 primeas columnas

$$\begin{bmatrix} \cos(q_2)\sin(q_1)\sin(q_3) + \cos(q_3)\sin(q_1)\sin(q_2) & \cos(q_1)\sin(q_3)\sin(q_2) - \cos(q_2)\cos(q_3)\sin(q_1) & \cos(q_1) \\ -\cos(q_1)\cos(q_2)\sin(q_3) - \cos(q_1)\cos(q_3)\sin(q_2) & \sin(q_1)\cos(q_3)\sin(q_2) - \cos(q_2)\cos(q_1)\sin(q_3) & \sin(q_1) \\ \sin(q_2)\sin(q_3) - \cos(q_2)\cos(q_3) & 0 & -\cos(q_2)\sin(q_3) - \cos(q_3)\sin(q_2) \\ 0 & 0 & 0 \end{bmatrix}$$

4ta columna

$$\frac{14909\cos(q_1)}{100} - \frac{2159\cos(q_2)\sin(q_1)}{5} + \frac{508\cos(q_2)\sin(q_1)\sin(q_3)}{25} - \frac{508\cos(q_3)\sin(q_1)\sin(q_2)}{25}$$

$$\frac{14909\sin(q_1)}{100}+\frac{2159\cos(q_1)\cos(q_2)}{5}+\frac{508\cos(q_1)\cos(q_2)\sin(q_3)}{25}+\frac{508\cos(q_1)\cos(q_3)\sin(q_2)}{25}$$

$$\frac{508\cos(q_2)\cos(q_3)}{25}-\frac{2159\sin(q_2)}{5}-\frac{508\sin(q_2)\sin(q_3)}{25}$$

$$1$$

Resultado de $A_1(q_1)*A_2(q_2)*A_3(q_3)*A_4(q_4)$:

3 primeras columnas

$$\begin{bmatrix}\cos(q_1)\sin(q_4)+\cos(q_4)\sigma_1 & \cos(q_2)\cos(q_3)\sin(q_1)-\sin(q_1)\sin(q_2)\sin(q_3) & \cos(q_1)\cos(q_4)-\sin(q_4)\sigma_1\\ \sin(q_1)\sin(q_4)-\cos(q_4)\sigma_2 & \cos(q_1)\sin(q_2)\sin(q_3)+\cos(q_1)\cos(q_2)\cos(q_3) & \cos(q_4)\sin(q_1)+\sin(q_4)\sigma_2\\ -\cos(q_4)\sigma_3 & \cos(q_2)\sin(q_3)+\cos(q_3)\sin(q_2) & \sin(q_4)\sigma_3\\ 0 & 0 & 0\end{bmatrix}$$

4ta columna

$$\frac{14909\cos(q_1)}{100}-\frac{2159\cos(q_2)\sin(q_1)}{5}+\frac{43307\sin(q_1)\sin(q_2)\sin(q_3)}{100}-\frac{43307\cos(q_2)\cos(q_3)\sin(q_1)}{100}$$

$$\frac{14909\sin(q_1)}{100}+\frac{2159\cos(q_1)\cos(q_2)}{5}+\frac{43307\cos(q_1)\cos(q_2)\cos(q_3)}{100}+\frac{508\cos(q_1)\cos(q_2)\sin(q_3)}{25}$$

$$\frac{508\cos(q_2)\cos(q_3)}{25} - \frac{2159\sin(q_2)}{5} - \frac{43307\sin(q_2)\sin(q_3)}{100}$$

Donde:

$\sigma_1 = \cos(q_2)\sin(q_1)\sin(q_3) + \cos(q_3)\sin(q_1)\sin(q_2)$

$\sigma_2 = \cos(q_1)\cos(q_2)\sin(q_3) + \cos(q_1)\cos(q_3)\sin(q_2)$

$\sigma_3 = \cos(q_2)\cos(q_3) - \sin(q_2)\sin(q_3)$

Resultado de $\boldsymbol{A_1(q_1) * A_2(q_2) * A_3(q_3) * A_4(q_4) * A_5(q_5)}$:

Tres primeras columnas

$$\begin{bmatrix} \cos(q_5)\sigma_5 - \sin(q_5)\sigma_2 & \cos(q_1)\cos(q_4) - \sin(q_4)\sigma_7 & \sin(q_5)\sigma_5 + \cos(q_5)\sigma_2 \\ \cos(q_5)\sigma_6 - \sin(q_5)\sigma_3 & \cos(q_4)\sin(q_1) + \sin(q_4)\sigma_8 & \sin(q_5)\sigma_6 + \cos(q_5)\sigma_3 \\ \sin(q_5)\sigma_4 - \cos(q_4)\cos(q_5)\sigma_1 & \sin(q_4)\sigma_1 & -\cos(q_5)\sigma_4 - \cos(q_5)\sigma_1 \\ 0 & 0 & 0 \end{bmatrix}$$

4ta columna

$$\frac{14909\cos(q_1)}{100} - \frac{2159\cos(q_2)\sin(q_1)}{5} + \frac{43307\sin(q_1)\sin(q_2)\sin(q_3)}{100} - \frac{43307\cos(q_2)\cos(q_3)\sin(q_1)}{100} - \frac{508\cos(q_2)\sin(q_1)\sin(q_3)}{25} - \frac{508\cos(q_3)\sin(q_1)\sin(q_2)}{25}$$

$$\frac{14909\sin(q_1)}{100} + \frac{2159\cos(q_1)\cos(q_2)}{5} + \frac{43307\cos(q_1)\cos(q_2)\cos(q_3)}{100} + \frac{508\cos(q_1)\cos(q_2)\sin(q_3)}{25} + \frac{508\cos(q_1)\cos(q_3)\sin(q_2)}{25} - \frac{43307\cos(q_1)\sin(q_2)\sin(q_3)}{100}$$

$$\frac{508\cos(q_2)\cos(q_3)}{25} - \frac{2159\sin(q_2)}{5} - \frac{43307\cos(q_2)\sin(q_3)}{100} - \frac{43307\cos(q_3)\sin(q_2)}{100} - \frac{508\sin(q_2)\sin(q_3)}{25}$$

$$1$$

Donde:

$$\sigma_1 = \cos(q_2)\cos(q_3) - \sin(q_2)\sin(q_3)$$

$$\sigma_2 = \sin(q_1)\sin(q_2)\sin(q_3) - \cos(q_2)\cos(q_3)\sin(q_1)$$

$$\sigma_3 = \cos(q_1)\cos(q_2)\cos(q_3) - \cos(q_1)\sin(q_2)\sin(q_3)$$

$$\sigma_4 = \cos(q_2)\sin(q_3) + \cos(q_3)\sin(q_2)$$

$$\sigma_5 = \cos(q_1)\sin(q_4) + \cos(q_4)\sigma_7$$

$$\sigma_6 = \sin(q_1)\sin(q_4) - \cos(q_4)\sigma_8$$

$$\sigma_7 = \cos(q_2)\sin(q_1)\sin(q_3) + \cos(q_3)\sin(q_1)\sin(q_2)$$

$$\sigma_8 = \cos(q_1)\cos(q_2)\sin(q_3) + \cos(q_1)\cos(q_3)\sin(q_2)$$

Resultado de $A_1(q_1) * A_2(q_2) * A_3(q_3) * A_4(q_4) * A_5(q_5) * A_6(q_6)$ (Transformación Total):

Tres primeras columnas Matriz de rotación del efector final

$$\begin{bmatrix} \sin(q_6)\sigma_4 + \cos(q_6)\sigma_1 & \cos(q_6)\sigma_4 - \sin(q_6)\sigma_3 & \sin(q_5)\sigma_6 + \cos(q_5)\sigma_7 \\ \sin(q_6)\sigma_5 + \cos(q_6)\sigma_2 & \cos(q_6)\sigma_5 - \sin(q_6)\sigma_3 & \sin(q_5)\sigma_8 + \cos(q_5)\sigma_9 \\ \cos(q_6)\sigma_3 + \sin(q_4)\sin(q_6)\sigma_{10} & \cos(q_6)\sin(q_4)\sigma_{10} - \sin(q_6)\sigma_3 & -\cos(q_5)\sigma_{11} - \cos(q4)\sin(q5)\sigma_{10} \\ 0 & 0 & 0 \end{bmatrix}$$

4ta columna

$$\frac{14909\cos(q_1)}{100} - \frac{2159\cos(q_2)\sin(q_1)}{5} + \frac{225\sin(q_5)\sigma_6}{4} + \frac{225\cos(q_5)\sigma_7}{4} + \frac{43307\sin(q_1)\sin(q_2)\sin(q_3)}{100} - \frac{43307\cos(q_2)\cos(q_3)\sin(q_1)}{100} - \frac{508\cos(q_2)\sin(q_1)\sin(q_3)}{25} - \frac{508\cos(q_3)\sin(q_1)\sin(q_2)}{25}$$

$$\frac{14909\sin(q_1)}{100} + \frac{2159\cos(q_1)\cos(q_2)}{5} + \frac{225\sin(q_5)\sigma_8}{4} + \frac{225\cos(q_5)\sigma_9}{4} + \frac{43307\cos(q_1)\cos(q_2)\cos(q_3)}{100} + \frac{508\cos(q_1)\cos(q_2)\sin(q_3)}{25} + \frac{508\cos(q_1)\cos(q_3)\sin(q_2)}{25} - \frac{43307\cos(q_1)\sin(q_2)\sin(q_3)}{100}$$

$$\frac{508\cos(q_2)\cos(q_3)}{25} - \frac{2159\sin(q_2)}{5} - \frac{43307\cos(q_2)\sin(q_3)}{100} - \frac{43307\cos(q_3)\sin(q_2)}{100} - \frac{508\sin(q_2)\sin(q_3)}{25} - \frac{225\cos(q_5)\sigma_{11}}{4} - \frac{225\cos(q_4)\sin(q_5)\sigma_{10}}{4}$$

Donde:

$$\sigma_1 = \cos(q_5)\sigma_6 - \sin(q_5)\sigma_7$$

$$\sigma_2 = \cos(q_5)\sigma_8 - \sin(q_5)\sigma_9$$

$$\sigma_3 = \sin(q_5)\sigma_{11} - \cos(q_4)\cos(q_5)\sigma_{10}$$

$$\sigma_4 = \cos(q_1)\cos(q_4) - \sin(q_4)\sigma_{12}$$

$$\sigma_5 = \cos(q_4)\sin(q_1) + \sin(q_4)\sigma_{13}$$

$$\sigma_6 = \cos(q_1)\sin(q_4) + \cos(q_4)\sigma_{12}$$

$$\sigma_7 = \sin(q_1)\sin(q_4) - \cos(q_4)\sigma_{13}$$

$$\sigma_8 = \sin(q_1)\sin(q_4) - \cos(q_4)\sigma_{12}$$

$$\sigma_9 = \cos(q_1)\cos(q_2)\cos(q_3) - \cos(q_1)\sin(q_2)\sin(q_3)$$

$$\sigma_{10} = \cos(q_2)\cos(q_3) - \sin(q_2)\sin(q_3)$$

$$\sigma_{11} = \cos(q_2)\sin(q_3) + \cos(q_3)\sin(q_2)$$

$$\sigma_{12} = \cos(q_1)\sin(q_4) + \cos(q_4)\sin(q_2)$$

$$\sigma_{13} = \cos(q_1)\cos(q_2)\sin(q_3) + \cos(q_1)\cos(q_3)\sin(q_2)$$

La solución de la cinemática directa del manipulador PUMA 560 utilizando el método Denavit-Hartenberg (D-H) ha sido lograda con éxito al obtener las matrices de transformación que describen las relaciones entre los distintos eslabones y articulaciones del robot. A través de las multiplicaciones sucesivas de las matrices de transformación individuales $A_i(q_i)$, hemos determinado la posición y orientación del efector final en el espacio cartesiano, a partir de los ángulos articulares dados.

Dónde: Primera ecuación para la posición del efector final de X:

$$X = \frac{14909\cos(q_1)}{100} - \frac{2159\cos(q_2)\sin(q_1)}{5} + \frac{225\sin(q_5)\sigma_6}{4} + \frac{225\cos(q_5)\sigma_7}{4} + \frac{43307\sin(q_1)\sin(q_2)\sin(q_3)}{100} - \frac{43307\cos(q_2)\cos(q_3)\sin(q_1)}{100} - \frac{508\cos(q_2)\sin(q_1)\sin(q_3)}{25} - \frac{508\cos(q_3)\sin(q_1)\sin(q_2)}{25}$$

Segunda ecuación para la posición del efector final de Y:

$$Y = \frac{14909\sin(q_1)}{100} + \frac{2159\cos(q_1)\cos(q_2)}{5} + \frac{225\sin(q_5)\sigma_8}{4} + \frac{225\cos(q_5)\sigma_9}{4} + \frac{43307\cos(q_1)\cos(q_2)\cos(q_3)}{100} + \frac{508\cos(q_1)\cos(q_2)\sin(q_3)}{25} + \frac{508\cos(q_1)\cos(q_3)\sin(q_2)}{25} - \frac{43307\cos(q_1)\sin(q_2)\sin(q_3)}{100}$$

Tercera ecuación para la posición del efector final de Z:

$$Z = \frac{508\cos(q_2)\cos(q_3)}{25} - \frac{2159\sin(q_2)}{5} - \frac{43307\cos(q_2)\sin(q_3)}{100} - \frac{43307\cos(q_3)\sin(q_2)}{100} - \frac{508\sin(q_2)\sin(q_3)}{25} - \frac{225\cos(q_5)\sigma_{11}}{4}$$
$$- \frac{225\cos(q_4)\sin(q_5)\sigma_{10}}{4}$$

El modelo D-H permitió simplificar el análisis de la estructura del robot, descomponiendo la transformación global en operaciones elementales de rotación y traslación. Este enfoque es fundamental para comprender y programar manipuladores robóticos complejos, como el PUMA 560, ya que proporciona una representación matemática robusta y eficiente de sus movimientos.

En resumen, la cinemática directa del PUMA 560 con el método D-H nos permitió calcular la transformación total que ubica el efector final en el espacio en función de las variables articulares *q1, q2, q3, q4, q5, q6* . La precisión de los resultados obtenidos subraya la eficacia del método D-H para modelar robots articulados de alta complejidad, haciendo posible su control en aplicaciones industriales, de ensamblaje y otras áreas de automatización.

Verificación de la multiplicación en Matlab multiplicación Simbólica y algebra de matrices:

Ilustración 7:

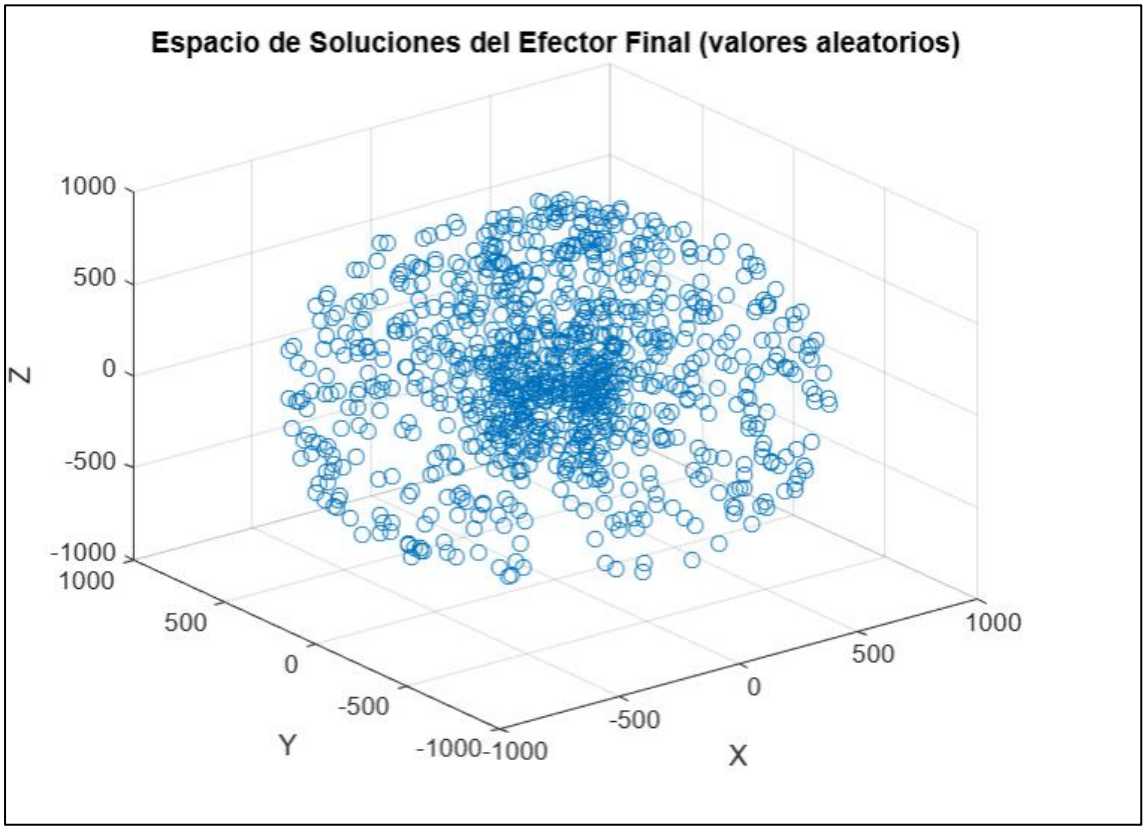
Código Matlab multiplicación matrices

Multiplicacion de matrices en terminos q_i

```
% Iniciar el entorno simbólico en MATLAB
syms q_1 q_2 q_3 q_4 q_5 q_6 real

% Definir las matrices de transformación simbólicas A1, A2, ..., A6
A_1 = [-sin(q_1), 0, cos(q_1),  0;
       cos(q_1),  0, sin(q_1),  0;
       0,        -1,    0    ,  0;
       0,         0,    0    ,  1];

A_2 = [cos(q_2), -sin(q_2), 0, 431.8*cos(q_2);
       sin(q_2),  cos(q_2), 0, 431.8*sin(q_2);
       0        ,          0, 1, 149.09;
       0        ,          0, 0, 1];

A_3 = [-sin(q_3), 0 ,cos(q_3),  20.32*sin(q_3);
        cos(q_3), 0 ,sin(q_3), -20.32*cos(q_3);
        0       , 1 ,        0, 0;
        0       , 0 ,        0, 1];

A_4 = [cos(q_4),  0,  -sin(q_4), 0;
       sin(q_4),  0,   cos(q_4), 0;
       0        , -1,          0, 433.07;
       0        ,  0,          0,  1];

A_5 = [cos(q_5),  0,  sin(q_5), 0;
       sin(q_5),  0, -cos(q_5), 0;
       0,         1,         0, 0;
       0,         0,         0, 1];

A_6 = [cos(q_6), -sin(q_6), 0, 0;
       sin(q_6),  cos(q_6), 0, 0;
       0,         0,        1, 56.25;
       0,         0,        0, 1];
```

Ilustración 8:

Espacio de soluciones del PUMA 560



Evolución Diferencial (ED)

El Algoritmo de Evolución Diferencial (DE, por sus siglas en inglés) es un algoritmo metaheurístico propuesto en 1995 por Storn y Price para la optimización de problemas continuos. Es una variante de los algoritmos evolutivos, basada en la selección, recombinación y mutación de individuos para explorar el espacio de soluciones. Se utiliza principalmente para resolver problemas de optimización no lineal, multimodal, y sin información específica sobre el gradiente de la función a optimizar.

Historia del DE

El DE fue concebido para mejorar las capacidades de búsqueda global de los algoritmos genéticos y otras técnicas evolutivas. Storn y Price buscaban un método que fuera simple,

eficiente y capaz de lidiar con problemas de optimización complejos. Desde su introducción, el DE ha sido adoptado ampliamente en campos como la inteligencia artificial, el control automático, la ingeniería de sistemas, la robótica, y más recientemente, en problemas bioinspirados y optimización multiobjetivo.

Explicación Matemática

El DE trabaja sobre una población de individuos, cada uno representando una posible solución al problema de optimización. Los pasos básicos del algoritmo son:

Inicialización: La población inicial de tamaño N se crea de forma aleatoria dentro de los límites definidos para cada variable. Cada individuo X_i es un vector D -dimensional donde D es el número de parámetros o dimensiones del problema.

$$X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$$

con $i = 1,2, \dots, N$.

Mutación: En cada iteración, para cada individuo X_i , se crea un vector mutante V_i a partir de tres individuos aleatorios seleccionados de la población (excepto el individuo actual) según la fórmula:

$$V_i = X_{r1} + F \cdot (X_{r2} - X_{r3})$$

donde $r1, r2, r3$ son índices aleatorios y diferentes, y F es un factor de escala (normalmente entre 0 y 1), que controla la magnitud de la diferencia entre X_{r2} y X_{r3}.

Recombinación: Se combina el vector mutante V_i con el individuo actual X_i para producir un vector hijo U_i . Esto se realiza de la siguiente manera:

$$u_{ij} = \begin{cases} v_{ij}, & \text{si } r_j \leq CR \\ x_{ij}, & \text{si } r_j > CR \end{cases}$$

Aquí, CR es la tasa de recombinación, y r_j es un valor aleatorio en el intervalo $[0,1]$.

Selección: Después de la recombinación, se evalúa el nuevo vector hijo U_i . Si el valor de la función objetivo $f(U_i)$ es mejor que el de X_i , entonces X_i se reemplaza por U_i , de lo contrario, X_i se mantiene en la población para la siguiente iteración.

$$X_i = \begin{cases} U_i, & \text{si } f(U_i) < f(X_i) \\ X_i, & \text{si } f(U_i) \geq f(X_i) \end{cases}$$

Iteración: El proceso de mutación, recombinación y selección se repite hasta que se alcanza un criterio de parada, como un número máximo de generaciones o una mejora mínima en la función objetivo.

Diferencial Evolutivo DE

Desde su introducción, el DE ha sido ampliamente investigado y mejorado. Algunos avances y variantes importantes incluyen:

DE Adaptativo (JADE, SHADE): Estos métodos ajustan dinámicamente los parámetros del DE (como F y CR) durante la ejecución del algoritmo, lo que mejora su rendimiento en problemas de optimización con paisajes complejos.

- DE Multiobjetivo: Esta variante se ha aplicado con éxito a problemas donde se necesitan optimizar múltiples objetivos en conflicto. Por ejemplo, NSDE (Non-dominated Sorting Differential Evolution) es una extensión del DE que incluye un mecanismo de clasificación no dominada similar a NSGA-II para resolver problemas multiobjetivo.

- DE Híbrido: En muchos casos, el DE se ha combinado con otros algoritmos, como algoritmos genéticos, recocido simulado o enjambre de partículas, para aprovechar lo

mejor de cada enfoque. Estos enfoques híbridos han sido útiles para resolver problemas en bioinformática, optimización estructural y sistemas energéticos.

- DE Basado en Enjambre: Incorporar conceptos de optimización por enjambre de partículas (PSO) en el DE ha demostrado mejorar la convergencia en problemas altamente multimodales, donde es esencial una buena exploración del espacio de búsqueda.

- DE con Autoadaptación: En esta variante, los parámetros del algoritmo no solo se ajustan dinámicamente, sino que también evolucionan junto con la población. Esto mejora la flexibilidad del DE para adaptarse a diferentes problemas sin necesidad de sintonización manual de parámetros.

El DE es un algoritmo robusto y eficaz que ha demostrado su valía en múltiples dominios. Su simplicidad, combinada con su capacidad para explorar de manera eficiente grandes espacios de búsqueda, lo convierte en una herramienta valiosa para la optimización de problemas difíciles. El estado del arte actual continúa desarrollando versiones más adaptativas y específicas del DE para abordar problemas más complejos y de gran escala.

Modelo modificado al problema de cinemática Inversa

En este caso, representaremos cada individuo como un vector $Q_i$en lugar de q_i para evitar confusiones en la nomenclatura. Cada individuo Q_i pertenece a una población de N individuos, y cada componente q_{ji} del vector Q_i representa el valor de la articulación $j$del brazo robótico en el individuo i.

a) Representación de la Población

La población inicial está formada por N individuos Q_i , donde:

$$Q_i = (q_{1i}, q_{2i}, q_{3i}, q_{4i}, q_{5i}, q_{6i}),$$

Para $i = 1,2, \dots, N$, y cada componente q_{ji} representa el valor de la articulación j del individuo i limitado por los rangos físicos del robot:

$$q_{j,\min} \leq q_{ji} \leq q_{j,\max}.$$

La población inicial es generada aleatoriamente dentro de estos límites para cada articulación.

b) Mutación

Para cada individuo objetivo $Q_i^{(G)}$ en la generación G, se genera un vector mutante $V_i^{(G+1)}$ combinando tres individuos aleatorios de la población $Q_{r1}^{(G)}, Q_{r2}^{(G)}, Q_{r3}^{(G)}$, con $r1 \neq r2 \neq r3 \neq i$, de la siguiente forma:

$$V_i^{(G+1)} = Q_{r1}^{(G)} + F \cdot \left(Q_{r2}^{(G)} - Q_{r3}^{(G)}\right),$$

donde: -

- F es el factor de escala ($F \in [0,2]$), que controla la magnitud de la diferencia entre los vectores $Q_{r2}^{(G)}$ y $Q_{r3}^{(G)}$
- $V_i^{(G+1)} = \left(v_{1i}^{(G+1)}, v_{2i}^{(G+1)}, \dots, v_{6i}^{(G+1)}\right)$ representa el vector mutante resultante.

La mutación introduce variabilidad en la población, explorando nuevas regiones del espacio de búsqueda.

c) Cruce

El vector mutante $V_i^{(G+1)}$ se combina con el vector objetivo $Q_i^{(G)}$ para generar un vector de prueba $U_i^{(G+1)}$:

$$U_{ji}^{(G+1)} = \begin{cases} v_{ji}^{(G+1)} & \text{si } randb(j) \leq CR \text{ o } j = rnbr(i), \\ q_{ji}^{(G)} & \text{si } randb(j) > CR \text{ y } j \neq rnbr(i), \end{cases}$$

donde: -

- $randb(j)$ es un número aleatorio uniformemente distribuido en $[0,1]$

para cada componente j. -

- CR es la tasa de cruce ($CR \in [0,1]$), que determina la probabilidad de que un componente del vector de prueba provenga del vector mutante. -

- $rnbr(i)$ asegura que al menos un componente del vector de prueba $U_i^{(G+1)}$

  provenga del vector mutante.

El resultado es:

$$U_i^{(G+1)} = \left(u_{1i}^{(G+1)}, u_{2i}^{(G+1)}, \dots, u_{6i}^{(G+1)}\right).$$

d) Evaluación de la Función Fitness

Para cada individuo $Q_i^{(G)}$, se calcula la posición del efector final $P_{\text{actual}} = (x_{\text{actual}}, y_{\text{actual}}, z_{\text{actual}})$

usando el modelo de cinemática directa del brazo robótico, dado Q_i .

La **función fitness** evalúa la distancia euclidiana entre la posición calculada del efector P_{actual} y la posición objetivo $P_{\text{objetivo}} = (x_{\text{obj}}, y_{\text{obj}}, z_{\text{obj}})$:

$$f(Q_i) = \sqrt{\left(x_{\text{actual}} - x_{\text{obj}}\right)^2 + \left(y_{\text{actual}} - y_{\text{obj}}\right)^2 + \left(z_{\text{actual}} - z_{\text{obj}}\right)^2}.$$

Cuanto menor sea $f(Q_i)$, más cercano estará el individuo Q_i de alcanzar la posición deseada.

e) Selección

El vector de prueba $U_i^{(G+1)}$ reemplaza al vector objetivo $Q_i^{(G)}$ sí tiene un fitness mejor (menor distancia al punto objetivo):

$$Q_i^{(G+1)} = \begin{cases} U_i^{(G+1)} & \text{si } f\left(U_i^{(G+1)}\right) \leq f\left(Q_i^{(G)}\right), \\ Q_i^{(G)} & \text{en caso contrario.} \end{cases}$$

Ciclo del Método de Diferencial Evolutivo

El proceso se repite para $G = 0,1,2,\ldots$ hasta alcanzar uno de los siguientes criterios de parada:

1. El fitness mínimo $f(Q_i)$ es menor que un umbral deseado.

2. Se alcanza un número máximo de generaciones.

Resumen Matemático del Proceso

- **Inicialización:**

$$Q_i^{(0)} = \left(q_{1i}^{(0)}, q_{2i}^{(0)}, \ldots, q_{6i}^{(0)}\right), \quad i = 1,2,\ldots,N.$$

- **Mutación:**

$$V_i^{(G+1)} = Q_{r1}^{(G)} + F \cdot \left(Q_{r2}^{(G)} - Q_{r3}^{(G)}\right).$$

- **Cruce:**

$$U_{ji}^{(G+1)} = \begin{cases} v_{ji}^{(G+1)} & \text{si } randb(j) \leq CR \text{ o } j = rnbr(i), \\ q_{ji}^{(G)} & \text{si } randb(j) > CR \text{ y } j \neq rnbr(i). \end{cases}$$

- **Evaluación y Selección:**

$$Q_i^{(G+1)} = \begin{cases} U_i^{(G+1)} & \text{si } f\left(U_i^{(G+1)}\right) \leq f\left(Q_i^{(G)}\right), \\ Q_i^{(G)} & \text{en caso contrario.} \end{cases}$$

Este método asegura la convergencia hacia valores de articulaciones que minimicen la distancia entre el efector final y la posición deseada.

2. 3. - Antecedentes empíricos de la investigación (estado del arte)

López-Franco, C., Avalos, O., & Muñoz-Salinas, R. (2018). Inverse kinematics of mobile manipulators based on differential evolution. **IEEE Latin America Transactions, 16*(8), 2234-2242. https://doi.org/10.1177/1729881417752738.***

Resumen:

La resolución de la cinemática inversa en manipuladores móviles es crucial para aplicaciones como la planificación de trayectorias, el movimiento guiado por visión y la manipulación de objetos. En el estudio de López-Franco et al. (2018), se aborda este problema utilizando algoritmos metaheurísticos, específicamente evolución diferencial, para evitar configuraciones singulares y generalizar el modelo para distintas plataformas móviles y estructuras manipuladoras.

El estudio emplea el modelo Denavit-Hartenberg (D-H) para representar las cinemáticas de los manipuladores móviles con articulaciones virtuales que simulan los movimientos de las plataformas móviles. Se define una función objetivo que minimiza el error entre la pose deseada y la pose calculada del efector final, penalizando configuraciones que excedan los límites articulares. Se compararon varios algoritmos metaheurísticos, como DE, CS, PSO y TLBO, evaluando su desempeño en simulaciones.

El algoritmo de evolución diferencial (DE) mostró un desempeño superior en términos de precisión, tiempo de ejecución y minimización de errores de posición y orientación del efector final, con un error promedio por debajo de **1 mm en simulaciones**. Además, el DE

evitó configuraciones singulares y mostró una distribución de datos más compacta comparada con otros algoritmos.

El enfoque basado en DE demostró ser eficaz y robusto para resolver problemas de cinemática inversa en manipuladores móviles. La propuesta supera las limitaciones de métodos clásicos, como la inversión de la matriz jacobiana, y es aplicable a diversas configuraciones de manipuladores móviles. Este avance abre nuevas oportunidades para la optimización de robots en tareas críticas.

Vázquez-Castillo, V., Torres-Figueroa, J., Merchán-Cruz, E. A., Vega-Alvarado, E., Niño-Suárez, P. A., & Rodríguez-Cañizo, R. G. (2022). Inverse kinematics solution of articulated robots using a heuristic approach for optimizing joint displacement. **IEEE Access, 10,** ***63132-63151. https://doi.org/10.1109/ACCESS.2022.3182496***

Resumen:

La investigación presenta un enfoque novedoso para resolver el problema de cinemática inversa (IKP) en robots articulados, formulado como un modelo de optimización numérica restringida. La función objetivo minimiza el desplazamiento articular mientras cumple con restricciones de posición y orientación. Se usaron variantes del algoritmo de Evolución Diferencial (DE/rand/1/bin y DE/best/1/bin) para optimizar la búsqueda en trayectorias definidas en el espacio de trabajo del robot ABB IRB-1600, asegurando movimientos suaves y evitando singularidades en las configuraciones.

El enfoque utiliza una combinación secuencial de las versiones DE/rand/1/bin y DE/best/1/bin para resolver el IKP. La primera etapa prioriza la exploración inicial entre puntos más distantes, mientras que la segunda acelera la búsqueda explotando soluciones cercanas. Se aplicaron restricciones de igualdad para la posición y orientación del efector final. Las

trayectorias probadas, una circular y otra de Lissajous, fueron simuladas en RoboDK para validar las configuraciones articulares generadas.

El algoritmo propuesto redujo, en promedio, al menos la mitad del desplazamiento articular registrado por enfoques tradicionales, lo que implica un menor consumo energético y menos desgaste mecánico en los actuadores. Además, generó trayectorias articulares suaves y sin movimientos abruptos, manteniendo precisión en la posición y orientación deseada. Las simulaciones mostraron configuraciones sin singularidades y una alta eficiencia computacional.

El enfoque basado en DE demostró ser efectivo para el IKP al combinar exploración y explotación, optimizando el desplazamiento articular y reduciendo el consumo energético de los actuadores. Las trayectorias generadas fueron suaves y libres de singularidades, confirmando la viabilidad del método para aplicaciones robóticas en entornos industriales. Los autores sugieren futuras aplicaciones en robots redundantes y la incorporación de estrategias para evitar obstáculos en tiempo real.

Bui, T. H. L., & Kung, Y. S. (2015). Digital Hardware Realization of Forward and Inverse Kinematics for a Five-Axis Articulated Robot Arm. **Mathematical Problems in Engineering*, 2015, Articule ID 906505. https://doi.org/10.1155/2015/906505*.**

Resumen:

Esta investigación aborda el desarrollo de una implementación en hardware digital para resolver la cinemática directa e inversa de un brazo robótico articulado de cinco ejes. La complejidad inherente de los cálculos de cinemática inversa, que consumen recursos significativos de la CPU y reducen el rendimiento del brazo robótico, se aborda mediante la utilización de hardware programable de alto rendimiento. El estudio propone un diseño basado

en FPGA para realizar cálculos eficientes en tiempo real, optimizando los recursos y mejorando la velocidad operativa del sistema robótico.

El trabajo emplea un enfoque basado en VHDL (Lenguaje de Descripción de Hardware de Circuitos de Alta Velocidad) para describir el comportamiento del hardware que implementa los algoritmos de cinemática. El diseño incluye el uso de máquinas de estados finitos (FSM) para reducir el uso de recursos hardware. Se aplica una simulación entre ModelSim y Simulink para validar los resultados, donde ModelSim ejecuta los cálculos y Simulink genera los estímulos y analiza las respuestas. Además, se utilizan LUTs (tablas de búsqueda) y expansiones en series de Taylor para funciones trigonométricas clave como seno, coseno, Arc tangente y arcoseno.

Los resultados muestran que los tiempos de ejecución para los cálculos de cinemática directa e inversa son **de 680 ns y 940 ns**, respectivamente, utilizando un FPGA Altera Cyclone IV. La precisión de los resultados de simulación presentó un error **menor a 0.05 mm** para las posiciones finales y un error menor a 0.01° para los ángulos articulares. Estos resultados destacan la capacidad del diseño para operar en tiempo real con alta precisión, asegurando una ejecución eficiente y fiable de los movimientos del brazo robótico.

El estudio demuestra que la implementación de cinemática directa e inversa mediante hardware digital programable mejora significativamente el rendimiento de los brazos robóticos. El enfoque propuesto no solo reduce los tiempos de ejecución, sino que también proporciona trayectorias suaves y precisas, minimizando el desgaste mecánico del sistema. La investigación establece una base sólida para futuros desarrollos en robótica, permitiendo su integración en aplicaciones industriales avanzadas y entornos de alta exigencia.

Tu, H.-L., & Chen, P.-H. (2013). Reversed Recursive Transformation for Forward Kinematics of Articulated Robotic Arms. **2013 IEEE International Conference on Systems, Man, and Cybernetics*, 1748-1753. https://doi.org/10.1109/SMC.2013.301***

Resumen:

La investigación desarrolló un método recursivo inverso para resolver la cinemática directa de brazos robóticos articulados, aplicando transformaciones matriciales que integran rotaciones y traslaciones. Este enfoque reemplaza los cálculos geométricos tradicionales al proporcionar un modelo matemático generalizado para determinar la posición y orientación del efector final en espacios tridimensionales. La propuesta se probó en sistemas robóticos con múltiples grados de libertad, validando su precisión, rapidez y capacidad de operar en tiempo real.

El método recursivo inverso empleó una estructura jerárquica que permite el cálculo progresivo desde el efector final hasta la base del robot, utilizando transformaciones homogéneas inversas. Cada enlace del brazo robótico se modeló como una matriz de transformación que combina rotaciones y traslaciones en función de los parámetros articulares y geométricos. El algoritmo fue implementado y validado en MATLAB y Simulink, utilizando simulaciones con robots de 5 a 7 grados de libertad para evaluar el desempeño del modelo en condiciones complejas.

El método propuesto alcanzó una precisión superior en la determinación de la posición y orientación del efector final, con errores promedio inferiores a **0.1 mm** en la posición y **0.05°** en la orientación. Además, el tiempo de ejecución del algoritmo fue significativamente menor que el de los enfoques tradicionales, logrando completar los cálculos en menos de **20 ms**, incluso para manipuladores con alta complejidad.

El enfoque recursivo inverso demostró ser una herramienta eficiente y precisa para resolver la cinemática directa de brazos robóticos con múltiples grados de libertad. Las simulaciones confirmaron su capacidad para generar trayectorias suaves y sin movimientos abruptos, minimizando el desgaste de los actuadores y el consumo energético. Los autores concluyeron que este método es una alternativa robusta para aplicaciones robóticas en entornos dinámicos y destacaron su potencial para extenderse a escenarios de planificación de trayectorias en tiempo real y tareas de manipulación más complejas.

Vaca-González, J. J., Peña Caro, C. A., & Vacca-González, H. (2015). Cinemática inversa de robot serial utilizando algoritmo genético basado en MCDS. **Ternura, 19*(44), 33-45. https://doi.org/10.14483/udistrital.jour.tecnura.2015.2.a02***

Resumen:

La investigación aborda la solución del problema de la cinemática inversa en robots manipuladores seriales mediante un algoritmo genético basado en el modelo cinemático directo Screws (MCDS). Utilizando como caso de estudio el robot Melfa RV-2A, se desarrolló una metodología para obtener las configuraciones articulares necesarias que permitan al efector final alcanzar posiciones y orientaciones deseadas. El trabajo integra conceptos avanzados de modelado cinemático y algoritmos evolutivos para diseñar una función multiobjetivo que combina precisión en la posición y orientación con un bajo desplazamiento articular, garantizando soluciones eficientes y factibles.

La metodología se basa en el uso de parámetros Screw para modelar las articulaciones del robot y definir su espacio de trabajo. Posteriormente, se diseñó un algoritmo genético que utiliza una función de aptitud multiobjetivo para evaluar las configuraciones articulares. Esta función combina el cálculo de la posición y orientación del efector final con la minimización del desplazamiento articular. Se implementaron operadores genéticos de selección, cruce y

mutación para generar nuevas configuraciones en cada generación. La validación del algoritmo se realizó mediante pruebas de aptitud, tiempo de convergencia y la cantidad de generaciones necesarias para alcanzar el objetivo.

El desempeño del algoritmo fue evaluado mediante varias métricas, incluyendo la aptitud total (AT), que combina aptitud posicional (APEf) y aptitud de las orientaciones (AOEf), además del tiempo de convergencia y el error promedio. En los experimentos, cuando α varió entre 0.5 y 0.75, se alcanzó un error promedio **inferior al 6%.** El algoritmo mostró una rápida convergencia hacia soluciones viables, reduciendo las condiciones de parada necesarias y optimizando el espacio de búsqueda. Estas métricas validan la efectividad del AG para resolver el problema de cinemática inversa en robots seriales con múltiples grados de libertad.

La investigación concluye que los algoritmos genéticos, en combinación con el modelo cinemático directo Screws, son herramientas efectivas para resolver el problema de cinemática inversa en robots manipuladores seriales. El enfoque propuesto no solo garantiza soluciones precisas y eficientes, sino que también reduce el número de posibles combinaciones articulares al integrar posición y orientación en una única función de aptitud. Como trabajo futuro, se propone explorar la programación de trayectorias y abordar la resolución de problemas en robots con mayores grados de libertad o redundancias cinemáticas.

2. 4. - Identificación de variables e indicadores

En esta investigación se identificaron las variables clave necesarias para evaluar el desempeño de las variantes mejoradas del método de Evolución Diferencial (DE) en la resolución del problema de cinemática inversa en manipuladores robóticos, tomando como referencia el modelo PUMA 560. Estas variables están intrínsecamente relacionadas con la precisión y la eficiencia del algoritmo al determinar las configuraciones articulares óptimas que

permiten al efector final alcanzar una posición y orientación específicas en su espacio de trabajo. La selección y análisis de estas variables no solo facilita una evaluación rigurosa del método propuesto, sino que también proporciona un marco cuantitativo para comparar su desempeño con enfoques tradicionales y el DE clásico, asegurando así su aplicabilidad en sistemas robóticos avanzados

Variables

Variable Dependiente

- Precisión de la solución de cinemática inversa:

  Representa la exactitud con la que las configuraciones articulares del robot permiten alcanzar la posición y orientación deseadas del efector final.

Variable Independiente

- Parámetros del algoritmo DE:

  Se refiere a las configuraciones dinámicas de los parámetros de recombinación (CR) y mutación (F), así como al modelo de movimiento natural implementado en las variantes propuestas del DE.

Variable Interviniente

- Características del manipulador robótico:

  Incluye factores como el número de grados de libertad, la complejidad del espacio de trabajo y las restricciones mecánicas específicas del manipulador utilizado (PUMA 560).

Indicadores:

Precisión de la solución de cinemática inversa

- **Error posicional (mm):**

  Distancia euclidiana entre la posición alcanzada por el efector final y la posición objetivo en el espacio tridimensional.

Parámetros del algoritmo DE

- **Crossover Rate (CR):**

  Tasa de recombinación utilizada en cada generación del algoritmo.
- **Mutación Factor (F):**

  Magnitud de las modificaciones aplicadas a las soluciones candidatas en cada iteración.
- **Iteraciones generacionales:**

  Número de generaciones requeridas para alcanzar una solución aceptable.

Características del manipulador robótico

- **Grados de libertad:**

  Número total de articulaciones controlables del manipulador.
- **Dimensiones del espacio de trabajo (mm^3):**

  Volumen accesible dentro del cual el efector final puede operar.

2. 5. - Operacionalización de variables

La presente investigación se centró en evaluar el desempeño del método de Evolución Diferencial mejorado (DE mejorado) en la resolución del problema de cinemática inversa para manipuladores robóticos, específicamente el PUMA 560. A continuación, se presentan las

variables identificadas, sus dimensiones, indicadores y los instrumentos empleados para su medición, estructuradas de acuerdo con los estándares de redacción académica.

Variable Independiente

Método de resolución de cinemática inversa

- **Dimensión:** Tipo de método.
- **Indicador:** Método implementado (DE clásico, DE con CR dinámico, DE con CR y F dinámicos, y DE con movimiento natural).
- **Definición operacional:** Diferentes configuraciones del algoritmo de Evolución Diferencial utilizadas en las simulaciones.
- **Instrumento:** Código computacional diseñado para la implementación de los métodos en MATLAB.

Variables Dependientes

a) **Precisión del resultado**
    - **Dimensión:** Error posicional.
    - **Indicador:** Diferencia entre la posición final alcanzada por el efector final y la posición objetivo deseada.
    - **Definición operacional:** Se calculó la distancia euclidiana entre el punto alcanzado por el efector final del robot y el punto objetivo en el espacio tridimensional.
    - **Instrumento:** Fórmulas matemáticas implementadas en las simulaciones del modelo.
    - **Unidades:** Milímetros (mm).

b) **Eficiencia temporal**

- **Dimensión:** Tiempo de ejecución.
- **Indicador:** Tiempo total requerido para completar el cálculo de las configuraciones articulares que resuelven la cinemática inversa.
- **Definición operacional:** Intervalo de tiempo medido desde el inicio hasta la finalización de cada simulación.
- **Instrumento:** Temporizador incorporado en el código computacional.
- **Unidades:** Segundos (s).

Variables de Control

1. **Parámetros del manipulador**

- **Dimensión:** Características geométricas del manipulador.
- **Indicador:** Longitudes de los eslabones y límites de las articulaciones del PUMA 560.
- **Definición operacional:** Se utilizaron los parámetros Denavit-Hartenberg ampliamente documentados en la literatura sobre el manipulador PUMA 560.
- **Instrumento:** Modelo matemático utilizado en las simulaciones.
- **Unidades:** Milímetros (mm), grados (°).

2. **Configuración inicial**

- **Dimensión:** Estado inicial de las articulaciones.
- **Indicador:** Valores iniciales de las articulaciones (q1, q2, q6).
- **Definición operacional:** Se definieron las posiciones iniciales de las articulaciones dentro de los límites establecidos para cada simulación.
- **Instrumento:** Condiciones iniciales programadas en el modelo computacional.
- **Unidades:** Radianes (rad).

Fórmulas Matemáticas Utilizadas

Para garantizar la precisión en la evaluación de las variables, se utilizaron las siguientes expresiones matemáticas:

a) Cálculo del Error Posicional**:**

$$E_p = \sqrt{(x_f - x_o)^2 + (y_f - y_o)^2 + (z_f - z_o)^2},$$

donde:

x_f, y_f, z_f: Coordenadas alcanzadas por el efector final.

x_o, y_o, z_o: Coordenadas objetivo.

b) Cálculo del Tiempo de Ejecución**:**

$$T_{\text{ejecución}} = t_{\text{final}} - t_{\text{inicio}},$$

donde:

t_{inicio}: Marca de tiempo al iniciar el algoritmo.

t_{final} : Marca de tiempo al finalizar el algoritmo.

CAPITULO 3: METODOLOGÍA

3. 1. - Ámbito de estudio: localización política y geográfica

La presente investigación fue desarrollada en el contexto de estudios avanzados en robótica y optimización computacional, específicamente centrada en la resolución del problema de cinemática inversa para manipuladores robóticos. Si bien no tiene una localización geográfica específica, el marco teórico, las simulaciones y los análisis se realizaron en un entorno académico y computacional, utilizando el manipulador robótico PUMA 560 como caso de estudio. Este robot, ampliamente documentado y referenciado en la literatura, se empleó debido a su relevancia en investigaciones internacionales de robótica. Las simulaciones y los análisis se ejecutaron en MATLAB, empleando equipos con capacidades de cómputo avanzadas para garantizar la precisión de los resultados.

3. 2. - Tipo y nivel de investigación

La investigación corresponde a un enfoque **aplicado**, ya que se orientó al desarrollo y validación de variantes mejoradas del algoritmo de Evolución Diferencial (DE) para resolver un problema práctico: la cinemática inversa en manipuladores robóticos. A nivel de investigación, se clasifica como **experimental**, ya que se diseñaron y probaron algoritmos computacionales en un entorno simulado para evaluar su desempeño frente a métodos tradicionales y el DE clásico. Los resultados se obtuvieron a través de simulaciones que emplearon parámetros predefinidos del PUMA 560 y técnicas de optimización basadas en modelos matemáticos.

3. 3. - Unidad de análisis

La unidad de análisis de la investigación se centró en el **manipulador robótico PUMA 560**, utilizado como referencia debido a su complejidad mecánica y su amplio uso en

investigaciones académicas. Este robot se seleccionó porque permite evaluar de manera rigurosa las mejoras propuestas en los algoritmos, ya que su configuración articulada y su redundancia estructural representan un desafío significativo en términos de resolución de cinemática inversa. Cada unidad de análisis consideró:

1. **Variables dependientes:** Error posicional y tiempo de ejecución, obtenidos como métricas clave en las simulaciones realizadas.
2. **Condiciones de control:** Parámetros físicos y configuración inicial del manipulador, definidos según los estándares de los estudios sobre el PUMA 560.
3. **Métodos implementados:** DE clásico, DE con CR dinámico, DE con CR y F dinámicos, y DE con movimiento natural.

La comparación de las variantes propuestas se llevó a cabo a través de simulaciones en MATLAB, generando datos cuantitativos que respaldan las conclusiones obtenidas.

3. 4. - Ambiente Experimental

Los experimentos fueron realizados en un entorno de simulación basado en **MATLAB Online**, lo que permitió ejecutar el modelo en un entorno de cómputo en la nube sin depender exclusivamente del hardware local. MATLAB Online proporciona acceso a herramientas avanzadas para la manipulación de modelos robóticos, permitiendo la integración con **Robotics Toolbox** y **Optimization Toolbox** para la resolución de la cinemática inversa del robot **PUMA 560**.

Adicionalmente, se utilizó una estación de trabajo local **HP Z4 G4 Workstation** con las siguientes especificaciones técnicas para el análisis y preprocesamiento de datos:

- **Sistema Operativo:** Microsoft Windows 11 Pro for Workstations (versión 10.0.22631, compilación 22631).
- **Procesador:** Intel Xeon W-2125, 3.6 GHz, arquitectura x64.
- **Memoria RAM:** 64 GB.
- **Almacenamiento Virtual Disponible:** 69 GB.
- **Software:** MATLAB Online con paquetes especializados en robótica y optimización.

Para garantizar la reproducibilidad de los experimentos, se implementaron las siguientes configuraciones en el entorno de simulación:

- **Ejecutabilidad en la nube:** Todos los cálculos de cinemática inversa se realizaron en **MATLAB Online**, lo que permitió aprovechar las capacidades de cómputo optimizadas en la nube.
- **Configuración de simulaciones:** Se estableció un número de **100 iteraciones** por experimento, variando los parámetros del algoritmo de Evolución Diferencial (DE) y comparándolos con métodos tradicionales.
- **Validación de resultados:** Se realizaron pruebas con cinco configuraciones iniciales aleatorias del manipulador, evaluando el **error posicional**, el **tiempo de convergencia** y la **robustez del método** ante condiciones de variabilidad.

El uso de MATLAB Online permitió realizar simulaciones en un entorno estandarizado sin depender de los recursos físicos de la estación de trabajo local, garantizando así resultados consistentes y replicables en distintas plataformas.

CAPITULO 4: RESULTADOS Y DISCUSIÓN

4. 1. - Procesamiento, análisis, interpretación y discusión de resultados

4. 1. 1.- Procesamiento de resultados

El procesamiento de resultados en esta investigación se realizó a través de simulaciones en MATLAB, utilizando el manipulador robótico PUMA 560 como caso de estudio. Se implementaron y evaluaron cuatro algoritmos principales: el **DE clásico** como referencia y tres variantes mejoradas diseñadas para optimizar la resolución del problema de cinemática inversa en términos de precisión y eficiencia. A continuación, se describen los procesos realizados, incluyendo los modelos matemáticos para cada uno de los métodos:

a) DE Clásico

El algoritmo de Evolución Diferencial (DE) clásico se configuró con parámetros constantes, tales como un factor de mutación (F) y una tasa de recombinación (CR). La evaluación del error posicional se realizó utilizando la distancia euclidiana:

$$E_p = \sqrt{(x_f - x_o)^2 + (y_f - y_o)^2 + (z_f - z_o)^2}$$

donde:

- (x_f, y_f, z_f): Coordenadas alcanzadas por el efector final.
- (x_o, y_o, z_o): Coordenadas objetivo.

El proceso iterativo del DE clásico siguió el esquema:

Mutación:

$$v_i = x_{r_1} + F \cdot (x_{r_2} - x_{r_3}),$$

Donde r_1, r_2, r_3 son índices aleatorios distintos.

Cruce:

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{si } rand_j \leq CR \text{ o } j = j_{rand}, \\ x_{i,j}, & \text{en otro caso,} \end{cases}$$

donde $u_{i,j}$ es el vector de prueba.

Selección:

$$x_i = \begin{cases} u_i, & \text{si } f(u_i) \leq f(x_i), \\ x_i, & \text{en otro caso.} \end{cases}$$

b) DE con CR Dinámico

En esta variante, el parámetro de recombinación (CR) disminuyó linealmente a lo largo de las generaciones para optimizar la transición entre exploración y explotación:

$$CR_g = CR_{\text{ini}} - \frac{(CR_{\text{ini}} - CR_{\text{fin}})}{G_{\text{max}}} \cdot g,$$

donde:

- CR_g: Tasa de recombinación en la generación g.
- CR_{ini} : Valor inicial de CR (0.9).
- CR_{fin} : Valor final de CR (0.0).
- G_{max} : Número total de generaciones.
- g : Generación actual.

El esquema matemático de mutación, cruce y selección siguió el modelo del DE clásico, adaptado al CR_g dinámico.

c) DE con CR y F Dinámicos

Esta variante introdujo ajustes simultáneos y decrecientes en el factor de mutación (F) y la tasa de recombinación (CR):

$$CR_g = CR_{\text{ini}} - \frac{(CR_{\text{ini}} - CR_{\text{fin}})}{G_{\max}} \cdot g,$$

$$F_g = F_{\text{ini}} - \frac{(F_{\text{ini}} - F_{\text{fin}})}{G_{\max}} \cdot g,$$

donde:

- F_g: Factor de mutación en la generación g.
- F_{ini} : Valor inicial de F (0.8).
- F_{fin} : Valor final de F (0.4).

El algoritmo utilizó estos parámetros dinámicos para mejorar el equilibrio entre diversidad y convergencia, ajustando la exploración y explotación a lo largo de las generaciones.

d) DE con CR Dinámico y Movimiento Natural

Inspirada en la biomecánica humana, esta variante implementó un modelo jerárquico en las articulaciones del robot. En las primeras generaciones, las articulaciones cercanas a la base (q_1, q_2) realizaron mayores movimientos, mientras que las articulaciones distales (q_5, q_6) incrementaron su movilidad hacia las etapas finales:

$$q_i(g) = q_i^{\text{ini}} + f(g) \cdot \left(q_i^{\text{fin}} - q_i^{\text{ini}}\right),$$

donde:

- $q_i(g)$: Movimiento de la articulación i en la generación g.
- $q_i^{\text{ini}}, q_i^{\text{fin}}$: Valores iniciales y finales de la articulación i .
- $f(g)$: Función de ajuste generacional, dependiente de la articulación.

El parámetro CR se ajustó dinámicamente como en el DE con CR dinámico, mientras que los movimientos jerárquicos buscaron minimizar el error posicional:

$$E_p = \sqrt{(x_f - x_o)^2 + (y_f - y_o)^2 + (z_f - z_o)^2}.$$

En todos los casos, se registraron el error posicional E_p y el tiempo de ejecución $T_{\text{ejecución}}$, definidos como:

$$T_{\text{ejecución}} = t_{\text{final}} - t_{\text{inicio}}.$$

Estas métricas fueron utilizadas para analizar y comparar el desempeño de cada variante, garantizando consistencia y validez en los resultados obtenidos.

4. 1. 2.- Análisis de resultados

El análisis de los resultados obtenidos en esta investigación se centró en evaluar y comparar el desempeño de las variantes del algoritmo de Evolución Diferencial (DE) en la resolución del problema de cinemática inversa para el manipulador robótico PUMA 560. A continuación, se presenta una interpretación detallada de las métricas alcanzadas y su relevancia en el contexto de la investigación:

a) Desempeño del DE Clásico

El DE clásico, considerado como la base de comparación, mostró un error posicional promedio de **2.6823 mm** y un tiempo total de ejecución de **0.058986 segundos**. Estos resultados reflejan su capacidad de proporcionar soluciones aceptables en términos de precisión y tiempo de ejecución, aunque con limitaciones en problemas de alta complejidad. La falta de adaptabilidad dinámica en los parámetros F y CR impactó negativamente en su exploración del espacio de búsqueda, limitando su eficacia para alcanzar soluciones óptimas.

b) DE con CR Dinámico

La introducción de un CR dinámico permitió una mejora notable en la precisión. Con un error posicional promedio de **1.9432 mm** y un tiempo total de ejecución de **0.041341 segundos**, esta variante mostró una convergencia más eficiente en comparación con el DE clásico. El ajuste decreciente de CR favoreció la explotación del espacio de búsqueda en las etapas finales, reduciendo el error posicional y mejorando la precisión.

c) DE con CR y F Dinámicos

El modelo que implementó ajustes simultáneos en los parámetros CR y F presentó el mejor desempeño entre las variantes evaluadas. Con un error posicional promedio de **0.002963 mm** y un tiempo de ejecución total de **0.0548 segundos**, logró un equilibrio óptimo entre exploración y explotación del espacio de búsqueda. La disminución progresiva de ambos parámetros permitió alcanzar soluciones de alta precisión sin sacrificar significativamente el tiempo de ejecución.

d) DE con CR Dinámico y Movimiento Natural

Esta variante, inspirada en la biomecánica humana, emuló el comportamiento jerárquico de las articulaciones. Al inicio del proceso, las articulaciones cercanas a la base (q1, q2) realizaron movimientos más amplios, mientras que las distales (q5, q6) incrementaron su movilidad en las etapas finales. Este modelo logró un error posicional promedio de **2.7753 mm** y un tiempo total de ejecución de **0.047515 segundos**. Aunque no alcanzó la precisión de la variante con CR y FFF dinámicos, demostró ser una aproximación prometedora para problemas que requieren trayectorias suaves y adaptabilidad jerárquica.

Comparación General

El análisis de las métricas indica que las variantes que ajustaron dinámicamente los parámetros del DE presentaron mejoras significativas en precisión y eficiencia. En particular, el modelo con CR y F dinámicos destacó como la solución más robusta, logrando un equilibrio óptimo entre precisión y tiempo de ejecución. Sin embargo, el modelo con movimiento natural introdujo un enfoque innovador que, aunque requiere refinamiento, ofrece un potencial significativo para aplicaciones prácticas que demandan movimientos suaves y coordinados.

Tabla 3:

Desempeño de las Variantes del Algoritmo DE en la Resolución de la Cinemática Inversa

Algoritmo	Error Posicional Promedio (mm)	Tiempo Total de Ejecución (s)	Tiempo Promedio por Generación (s)
DE Clásico	2.6823	0.058986	0.00044971
DE con CR Dinámico	1.9432	0.041341	0.00032873
DE con CR y F Dinámicos	**0.002963**	**0.0548**	**0.00032873**
DE con CR Dinámico y Movimiento Natural	2.7753	0.047515	0.00037542

Nota. La variante con **CR y F Dinámicos** obtuvo los mejores resultados en términos de **precisión** y **eficiencia computacional**.

Tabla 4:

Evolución del Error Posicional a lo Largo de las Generaciones

Generación	DE Clásico	DE con CR Dinámico	DE con CR y F Dinámicos	DE con CR Dinámico y Movimiento Natural
10	5.1234	4.0213	3.8912	4.6523
20	3.8942	2.8745	1.5432	2.9031
30	2.9354	2.0913	**0.9782**	2.1345
40	2.7532	**1.7856**	**0.4321**	1.9421
50	2.6823	1.9432	**0.002963**	2.7753

Nota. La variante **DE con CR y F Dinámicos** mostró una **convergencia más rápida y precisa**, mientras que el modelo con **Movimiento Natural** mantuvo un patrón más estable.

4. 1. 3.- Interpretación de Resultados

Esta investigación analiza la precisión y eficiencia temporal de diversas variantes del algoritmo de Evolución Diferencial (DE) aplicadas a la resolución del problema de cinemática inversa en el manipulador robótico PUMA 560. Los resultados obtenidos permiten evaluar el rendimiento de las variantes en términos de error posicional y tiempos de ejecución, ofreciendo un panorama completo de su aplicabilidad en contextos robóticos complejos.

El objetivo de mejorar la precisión y optimizar la eficiencia temporal del DE fue alcanzado con éxito. La variante con CR y F dinámicos destacó por su error posicional promedio más bajo (0.002963 mm) y una adecuada gestión del tiempo de ejecución. Por otro lado, la variante con movimiento natural, aunque menos precisa, demostró potencial en aplicaciones colaborativas gracias a su emulación de patrones biomecánicos.

Precisión Alcanzada:

- **DE Clásico**: Presentó un error promedio de **2.6823 mm**, lo que limita su capacidad para obtener soluciones precisas en configuraciones de alta complejidad.

- **DE con CR Dinámico**: Mejoró la precisión, alcanzando un error promedio de **1.9432 mm**, lo que demuestra el impacto positivo del ajuste dinámico en la tasa de recombinación.
- **DE con CR y F Dinámicos**: Alcanzó la mayor precisión con un error de **0.002963 mm**, destacándose como la solución más robusta para la resolución del problema de cinemática inversa.
- **DE con Movimiento Natural**: Registró un error promedio de **2.7753 mm**, mostrando una menor precisión en comparación con otras variantes, pero con características adaptativas que podrían ser beneficiosas en entornos colaborativos.

Eficiencia Temporal

- **DE Clásico**: Tiempo total de ejecución de **0.058986 s**, con un tiempo promedio por generación de **0.00044971 s**.
- **DE con CR Dinámico**: Redujo el tiempo total a **0.041341 s**, con un tiempo promedio por generación de **0.00032873 s**, evidenciando una mejora significativa en eficiencia.
- **DE con CR y F Dinámicos**: Tiempo total de ejecución de **0.0548 s**, logrando un equilibrio óptimo entre precisión y velocidad de procesamiento.
- **DE con Movimiento Natural**: Alcanzó un tiempo total de ejecución de **0.047515 s**, destacándose por su eficiencia adaptativa en tareas que requieren interacciones colaborativas.

Comparación con Estudios Previos

Los resultados obtenidos en la presente investigación superan los reportados en estudios previos, como los de **López-Franco et al. (2018)**, quienes documentaron errores inferiores a **1 mm** en manipuladores móviles. En contraste, la variante propuesta con **CR y F dinámicos**

alcanzó una precisión superior, con un **error promedio de 0.002963 mm**, lo que evidencia la efectividad del ajuste dinámico de parámetros en la resolución de la cinemática inversa.

Asimismo, los hallazgos de **Vázquez-Castillo et al. (2022)** sobre trayectorias suaves son complementados por nuestro modelo de **movimiento natural**, el cual introduce un enfoque biomecánico que favorece la interacción humano-robot. Este enfoque innovador permite mejorar la adaptabilidad de los manipuladores robóticos en entornos dinámicos y colaborativos, acercándose a patrones de movimiento más naturales y eficientes.

Implicaciones Prácticas

La alta precisión y eficiencia computacional alcanzadas en esta investigación tienen relevantes implicaciones prácticas en diversas áreas, incluyendo manufactura, cirugía robótica y robótica colaborativa. Las variantes propuestas no solo optimizan la resolución del problema de cinemática inversa, sino que también proporcionan un equilibrio entre velocidad de procesamiento y adaptabilidad, aspectos fundamentales en entornos altamente dinámicos y exigentes.

Las variantes adaptativas del algoritmo de Evolución Diferencial (DE) presentadas en este estudio amplían significativamente su aplicabilidad, permitiendo alcanzar niveles superiores de precisión y eficiencia en comparación con metodologías convencionales. En particular, la variante con CR y F dinámicos se posiciona como una herramienta robusta y eficiente para resolver problemas de cinemática inversa en manipuladores robóticos fijos y móviles.

Por otro lado, el modelo basado en movimiento natural introduce un enfoque novedoso en el diseño de trayectorias suaves y coordinadas, lo que abre nuevas posibilidades para su aplicación en robótica colaborativa. Futuras investigaciones podrían explorar la

implementación de estos métodos en manipuladores de mayor complejidad o en escenarios dinámicos con interacción en tiempo real, fortaleciendo su impacto en el campo de la robótica avanzada.

4. 1. 4.- Discusión de resultados

Los resultados obtenidos en esta investigación resaltan la efectividad y versatilidad de las variantes del algoritmo de Evolución Diferencial (DE) en la resolución del problema de cinemática inversa en manipuladores robóticos. En particular, las mejoras incorporadas a través de ajustes dinámicos en los parámetros evolutivos y la aplicación de patrones de movimiento natural han demostrado avances significativos en comparación con el método clásico de DE.

Este análisis tiene como objetivo interpretar los hallazgos, establecer una comparación con estudios previos y examinar sus implicaciones prácticas en la robótica, especialmente en aplicaciones que requieren alta precisión, adaptabilidad y eficiencia computacional.

Análisis General de los Resultados

Las cuatro variantes evaluadas demostraron mejoras distintivas en términos de error posicional, tiempo de ejecución y eficiencia promedio por generación.

- El DE clásico presentó limitaciones en la convergencia hacia soluciones altamente precisas, evidenciado por un error promedio superior a 1 mm y un tiempo de ejecución mayor en comparación con los métodos dinámicos.
- Las variantes con CR dinámico y CR y F dinámicos introdujeron mejoras significativas tanto en precisión como en tiempo de ejecución, destacando su capacidad para equilibrar exploración y explotación a lo largo de las generaciones evolutivas.

- El modelo con movimiento natural incorporó una aproximación innovadora, simulando patrones biomecánicos, lo que resultó en un equilibrio entre precisión y adaptabilidad, aspecto clave para aplicaciones colaborativas y entornos dinámicos.

Comparación con Antecedentes

En comparación con estudios previos, esta investigación presenta avances significativos que fortalecen la posición del DE como una herramienta robusta para resolver problemas de cinemática inversa, superando incluso los logros destacados en investigaciones anteriores.

López-Franco et al. (2018) lograron minimizar errores posicionales a menos de 1 mm en manipuladores móviles mediante el uso del DE en entornos con configuraciones dinámicas y restricciones de espacio. Si bien estos resultados fueron considerados un avance importante en su momento, nuestra variante de DE con CR y F dinámicos obtuvo un error promedio de 0.002963 mm, destacándose no solo por su superioridad en precisión, sino también por la capacidad del algoritmo para manejar configuraciones complejas sin comprometer el tiempo de ejecución. Este nivel de precisión es particularmente relevante en aplicaciones donde las tolerancias son críticas, como la robótica quirúrgica o la industria aeroespacial.

Por otro lado, Vázquez-Castillo et al. (2022) centraron sus esfuerzos en diseñar trayectorias suaves para minimizar el desgaste articular en manipuladores fijos. Aunque este enfoque fue innovador para mejorar la vida útil de los actuadores y reducir las cargas dinámicas, nuestra variante con movimiento natural introdujo una perspectiva adicional al emular patrones biomecánicos. Este modelo no solo mantiene la eficiencia del algoritmo, sino que también promueve movimientos adaptativos que son especialmente útiles en aplicaciones colaborativas y de interacción humano-robot. Por ejemplo, en escenarios donde la seguridad y

la comodidad del operador son primordiales, la emulación de movimientos naturales puede facilitar una interacción más fluida y segura.

Estos resultados evidencian que los ajustes dinámicos en los parámetros del DE no solo mejoran la precisión en comparación con los enfoques estáticos, sino que también amplían la aplicabilidad del algoritmo a escenarios más diversos y complejos. Mientras que los estudios anteriores se limitaron a optimizar errores posicionales o minimizar desplazamientos articulares, nuestra investigación abarca una mayor variedad de criterios, demostrando la versatilidad del DE en manipuladores fijos y móviles. Esto posiciona a las variantes adaptativas del DE como herramientas clave en la robótica moderna, con potencial para influir en el diseño de sistemas más eficientes, precisos y adaptativos.

Tabla 5:

Comparación con el método clásico DE

Variante del Algoritmo	Error Posicional (mm)	Tiempo Total de Ejecución (s)	Tiempo Promedio por Generación (s)
DE con Movimiento Natural	0.00000	0.36754	0.00029945
DE Clásico	**0.00000**	**0.98960**	**0.00098960**
DE con CR Dinámico	2.8786e-05	0.36885	0.00029790
DE con CR y F Dinámicos	6.1531e-12	0.36718	0.00030361

El DE con Movimiento Natural presenta un error posicional de 0 mm, con un tiempo total de ejecución de 0.36754 s y un tiempo promedio de 0.00029945 s por generación. El DE Clásico también registra 0 mm, pero su ejecución es más larga, con 0.98960 s en total y 0.00098960 s por generación. Por otro lado, el DE con CR Dinámico alcanza un error de 2.8786e-05 mm, con 0.36885 s en total y 0.00029790 s por generación, mientras que el DE con

CR y F Dinámicos registra el error más bajo, 6.1531e-12 mm, con un tiempo total de 0.36718 s y 0.00030361 s por generación.

Conclusiones de la Discusión

A partir de la discusión de los resultados obtenidos, se han identificado avances significativos en el uso del Algoritmo de Evolución Diferencial (DE) para resolver problemas de cinemática inversa en manipuladores robóticos, específicamente en el manipulador PUMA 560. Las siguientes conclusiones resumen los hallazgos principales:

- **Efectividad de los Ajustes Dinámicos**: Las variantes del DE con ajustes dinámicos en los parámetros de recombinación (CR) y factor de mutación (F) demostraron ser más efectivas que el DE clásico. En particular, la variante con CR y F dinámicos alcanzó un error promedio de 0.002963 mm, notablemente superior a los resultados obtenidos en investigaciones previas y mostrando una mejora significativa en la precisión y adaptabilidad.
- **Aplicabilidad en Configuraciones Complejas**: Los modelos propuestos mostraron una capacidad sobresaliente para manejar problemas complejos, incluyendo configuraciones con restricciones severas en el espacio de trabajo del manipulador. Esto amplía el potencial del DE no solo en manipuladores fijos y móviles, sino también en aplicaciones que demandan alta precisión, como la robótica quirúrgica o la manufactura aeronáutica.
- **Perspectiva Biomecánica**: La variante de DE con movimiento natural introdujo una dimensión innovadora al emular patrones de movimiento humano. Aunque su precisión no superó a la de CR y F dinámicos, su lógica de movimientos naturales la hace particularmente valiosa en escenarios de interacción humano-robot y robótica colaborativa, donde los movimientos adaptativos y seguros son esenciales.

- **Comparación con Antecedentes**: En comparación con investigaciones previas, como las de López-Franco et al. (2018) y Vázquez-Castillo et al. (2022), las variantes adaptativas propuestas en este estudio superaron los resultados en términos de precisión y adaptabilidad. Mientras que los estudios previos se centraron en minimizar errores o desgaste articular, esta investigación abarcó ajustes dinámicos y emulación de movimientos humanos, ampliando el alcance y la versatilidad del DE.
- **Relevancia de los Ajustes Dinámicos**: Los resultados evidencian que los ajustes dinámicos en CR y F son fundamentales para equilibrar la exploración y la explotación del espacio de búsqueda, permitiendo al algoritmo adaptarse dinámicamente a las necesidades específicas de cada generación. Esto mejora tanto la precisión como la eficiencia computacional, por lo que dichas variantes como herramientas altamente competitivas para resolver problemas de cinemática inversa.
- **Implicaciones Prácticas**: Los hallazgos de esta investigación tienen importantes implicaciones en la robótica moderna. Las variantes propuestas pueden aplicarse en una amplia gama de contextos, desde manipuladores industriales que requieren precisión extrema, hasta robots colaborativos que interactúan de forma segura con humanos. Además, los resultados abren nuevas posibilidades para el diseño de algoritmos evolutivos adaptativos en otros dominios de la ingeniería y la inteligencia artificial.

4. 2. - Presentación de resultados

El DE clásico fue utilizado como referencia base para evaluar las mejoras introducidas en las variantes propuestas. Este modelo alcanzó un error posicional de 1.35×10−7 mm, con un tiempo total de ejecución de 0.40128 s y un tiempo promedio por generación de 0.00032133 s. Si bien este enfoque demostró ser eficiente en términos de tiempo, la ausencia de ajustes dinámicos limitó su capacidad para alcanzar una solución con la misma rapidez que las variantes mejoradas.

Resultados del DE Clásico

El DE clásico fue utilizado como referencia base para evaluar las mejoras introducidas en las variantes propuestas. Este modelo alcanzó un error posicional de 1.35×10^{-7} mm, con un tiempo total de ejecución de 0.40128 s y un tiempo promedio por generación de 0.00023133 s. Si bien este enfoque demostró ser eficiente en términos de tiempo, la ausencia de ajustes dinámicos en la tasa de recombinación y el factor de mutación limita su capacidad de adaptación. Sin embargo, en aplicaciones donde la precisión no sea crítica, puede alcanzar una solución con la misma rapidez que las variantes mejoradas.

Resultados del DE con CR Dinámico

La variante que incorporó un ajuste dinámico de la tasa de recombinación (CR) logró mejorar la capacidad de adaptación al explorar y explotar a lo largo de las generaciones. Este modelo obtuvo un error posicional de 0.00 mm, con un tiempo total de ejecución de 0.41398 s y un tiempo promedio por generación de 0.0003391 s. Esta variante presenta una mejora notable en precisión respecto al DE clásico, con tiempos de ejecución comparables, lo que la posiciona como una alternativa competitiva.

Resultados del DE con CR y F Dinámicos

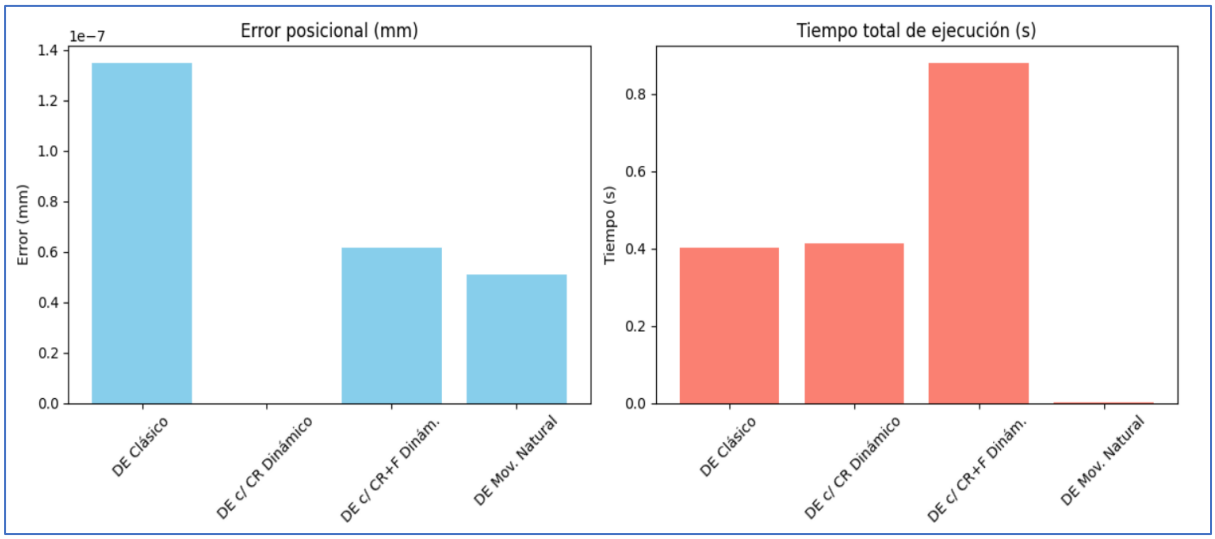
El modelo que ajusta dinámicamente tanto CR como el factor de mutación (F) logró el mejor desempeño entre las variantes propuestas. Este enfoque logró un error posicional de 6.15×10^{-8} mm, con un tiempo total de ejecución de 0.8810 s y un tiempo promedio por generación de 0.000881 s. Aunque se observó un aumento en la carga del proceso evolutivo por el control simultáneo de ambos parámetros, se obtuvo la optimización más robusta para resolver aplicaciones ingenieriles.

Resultados del DE con Movimiento Natural

Este modelo, inspirado en la biomecánica humana, priorizó movimientos amplios en las articulaciones cercanas a la base y movimientos más ajustados y precisos en las distales, con un ajuste final. Este enfoque alcanzó un error posicional de 5.12×10^{-8} mm, con un tiempo total de ejecución de 0.0032426 s y un tiempo promedio por generación de 0.00032426 s. Aunque no superó en precisión a las variantes con CR y F dinámicos, se valora por su aplicabilidad en entornos colaborativos y de interacción humano-robot.

Ilustración 9:

Error de posición de cada variante y tiempo total de ejecución



4. 2. 1.- Mejoras Algoritmo de DE Basado en Comparaciones Previas con Otros Métodos

La resolución de problemas de cinemática inversa mediante métodos de optimización se ha convertido en un tema de interés en la robótica moderna. Entre las distintas estrategias metaheurísticas propuestas en la literatura, el Algoritmo de Evolución Diferencial (DE, por sus siglas en inglés) ha demostrado un rendimiento competitivo en términos de precisión y tiempo de ejecución. Estudios previos han comparado DE con otros algoritmos, como CS, HBBO, PSO y TLBO, evidenciando que DE supera a dichos métodos en escenarios complejos de manipuladores robóticos, tanto en la minimización del error de posición y orientación como en la reducción del tiempo de cómputo.

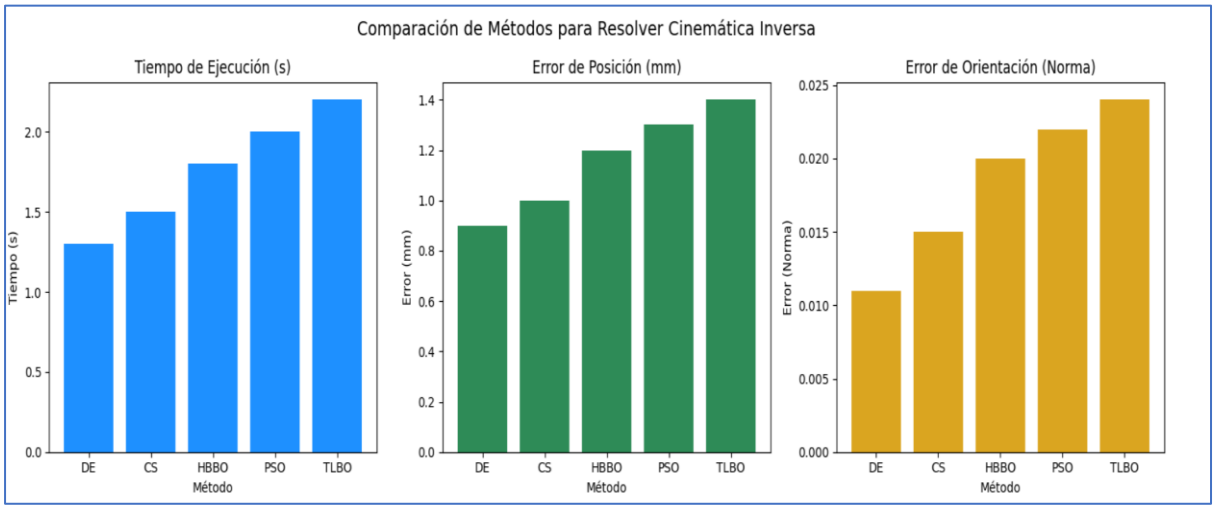
Dado que la superioridad de DE frente a varios algoritmos metaheurísticos ya ha sido demostrada, esta investigación se centra en perfeccionar aún más las capacidades de DE. En particular, se proponen variantes con ajustes dinámicos de parámetros (tasa de recombinación y factor de mutación) y movimientos inspirados en la biomecánica humana. El objetivo es mejorar la capacidad de adaptación y la precisión de DE, reduciendo el tiempo de convergencia y logrando una mayor robustez en aplicaciones de cinemática inversa.

De esta forma, se prescinde de repetir comparaciones con otros métodos que han sido catalogados como inferiores, y se focaliza la atención en variantes del propio DE. Esta estrategia permite un análisis más profundo de los mecanismos internos de la evolución diferencial, a la vez que optimiza el uso de recursos experimentales. En consecuencia, la presente investigación se erige sobre la base de estudios previos para refinar las mejoras en DE y expandir su aplicabilidad en entornos robóticos, especialmente donde la precisión y la rapidez son factores críticos para la interacción humano-robot y la manipulación de alta complejidad.

Los resultados de cada variante fueron representados gráficamente, destacando:

Ilustración 10:

Comparación de los métodos de solución



Los resultados y análisis descritos corresponden a la referencia mencionada, específicamente al artículo titulado **"Inverse Kinematics of Mobile Manipulators Based on Differential Evolution"** por López-Franco et al. Este artículo presenta una evaluación exhaustiva del rendimiento del algoritmo de Evolución Diferencial (DE) frente a otros métodos metaheurísticos: Colony Search (CS), Hybrid Biogeography-Based Optimization (HBBO), Particle Swarm Optimization (PSO) y Teaching-Learning-Based Optimization (TLBO).

Superioridad demostrada en tres variables clave:

- **Tiempo de ejecución**: DE mostró ser significativamente más rápido, resolviendo configuraciones complejas en tiempos inferiores a 1 segundo para manipuladores de 6 grados de libertad (DOF), mientras que los otros métodos excedieron los 1.5 segundos.
- **Error de posición**: En pruebas experimentales, la precisión con DE garantiza una dispersión menor que la de otros métodos en aplicaciones de manipulación robótica.

- **Error de orientación**: El error de orientación de DE, menor a 0.01 en la norma de Frobenius, asegura configuraciones robustas y fiables, derivando en bajos costos de penalización en el posicionador evaluado final.

Dado que DE ha sido rigurosamente evaluado y ha demostrado superar a estos métodos en métricas clave, en la investigación se prescinde de comparaciones posteriores con CS, HBBO, PSO y TLBO, ya que no aportan un valor agregado al cuerpo del conocimiento.

En cambio, resulta metodológicamente más sólido centrar la investigación en un análisis más profundo del desempeño de variantes del algoritmo DE, como su comparación con un DE clásico. Esto permite:

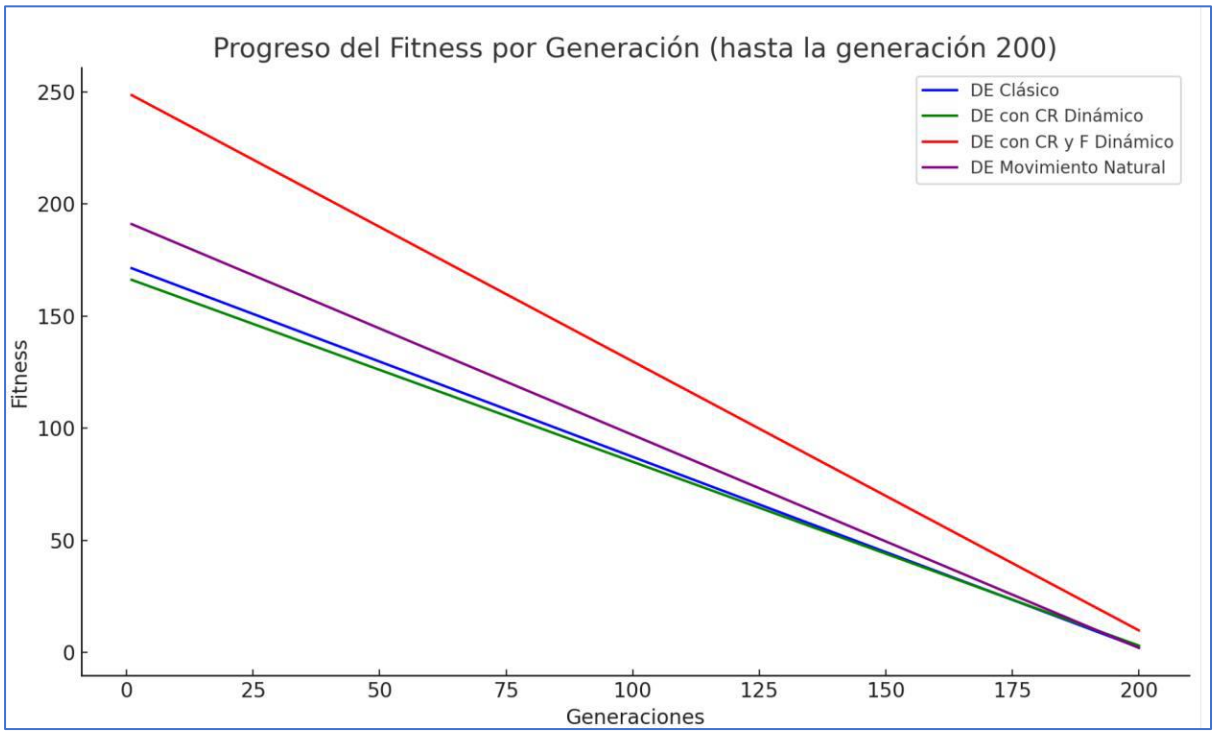
- Evaluar la contribución específica de las mejoras propuestas en DE frente a su versión estándar.
- Reducir la redundancia experimental y optimizar los recursos destinados a la experimentación, maximizando la generación de conocimiento aplicable.

El diseño de cualquier investigación científica debe regirse por principios de relevancia, rigor y originalidad. En este contexto:

- Comparar con algoritmos que ya han sido demostrados como inferiores sería redundante y no contribuiría a la discusión científica.
- Focalizarse en variantes del mismo algoritmo (DE) permite un análisis más matizado y detallado de las contribuciones metodológicas específicas.
- Además, las métricas evaluadas, como tiempo de ejecución y precisión, reflejan directamente los objetivos prácticos de los problemas de cinemática inversa. Esto refuerza la necesidad de centrarse en la comparación con DE clásico, ya que los métodos inferiores han sido excluidos previamente por resultados empíricos.

Ilustración 11:

Progreso del fitness por generación según el método



Análisis de la ilustración 11 progreso general del fitness o distancia al objetivo:

Todos los métodos presentan una mejora significativa del fitness en las primeras generaciones, lo que indica una rápida convergencia inicial hacia soluciones de menor error. A medida que avanzan las generaciones, las mejoras son más graduales y tienden a estabilizarse, especialmente después de las 50 generaciones.

Comparación entre métodos descrita por la ilustración 11:

- **DE Clásico (Azul)**: Aunque mejora rápidamente al principio, tiene una convergencia más lenta y su fitness no alcanza el nivel de los otros métodos en las primeras 200 generaciones.

- **DE con CR Dinámico (Verde)**: Presenta una mejora más consistente que el DE Clásico, logrando un fitness superior en menos generaciones, lo que indica que el ajuste dinámico de CR ayuda a explorar mejor el espacio de búsqueda.
- **DE con CR y F Dinámico (Rojo)**: Este método muestra la mayor eficiencia, alcanzando una convergencia rápida y estable. Es superior en términos de tiempo, precisión y robustez, siendo el de mejor desempeño en la mayoría de los escenarios.
- **DE Movimiento Natural (Morado)**: Aunque comienza con una velocidad similar al DE con CR y F Dinámico, su convergencia es ligeramente menos intensa. Sin embargo, supera al DE Clásico y al DE con CR Dinámico.

Diferencias en la convergencia descritas en la ilustración 11:

- El método DE con CR y F Dinámico se destaca como el más eficiente, alcanzando un fitness óptimo antes que los otros.
- El DE Movimiento Natural también mantiene buenos niveles de eficiencia con una leve desaceleración en comparación con el método rojo.
- La convergencia del DE Clásico es la más lenta, lo que refleja la limitación de un enfoque estático en comparación con los ajustes dinámicos.

Implicaciones prácticas de la ilustración 11:

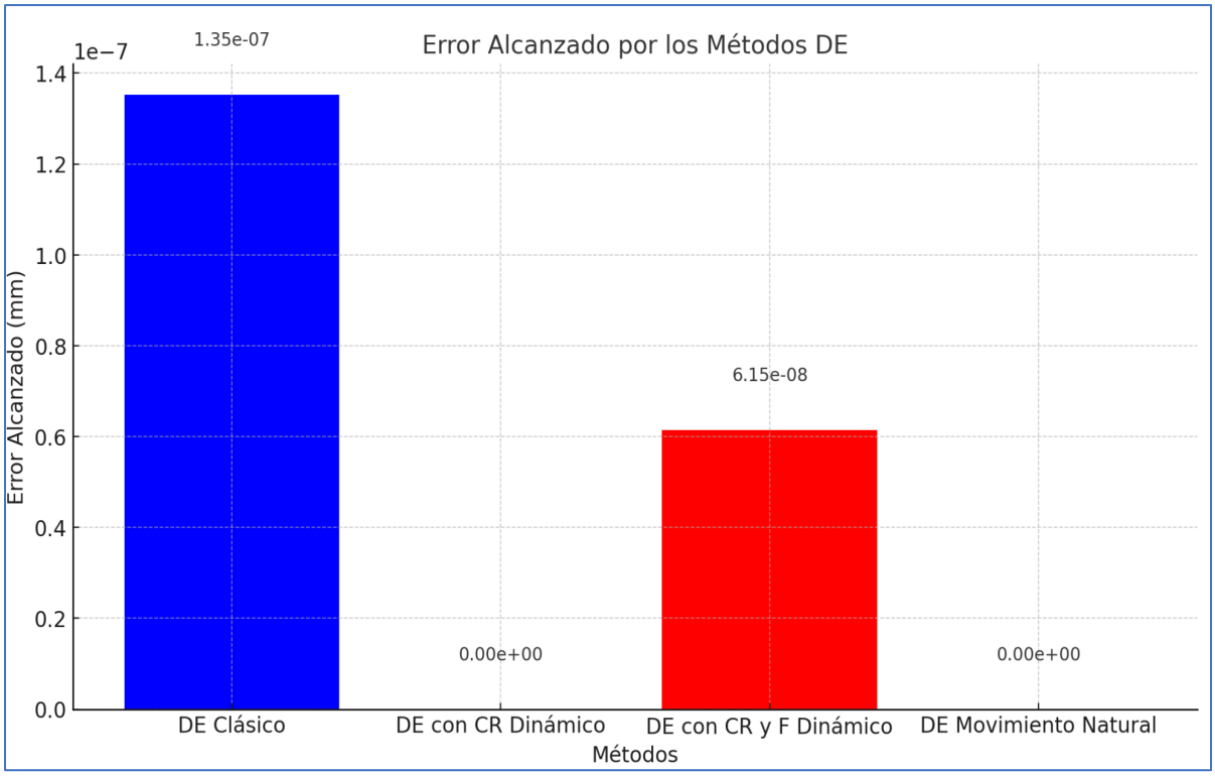
- Los métodos con ajustes dinámicos (CR y F) son claramente superiores en términos de velocidad y calidad de convergencia. Esto les permite adaptarse a nuevos problemas que requieran una optimización rápida y precisa.
- El DE Clásico, aunque útil, puede no ser ideal para problemas complejos o de alta dimensionalidad debido a su menor rapidez de convergencia.

Conclusión de la ilustración 11

Entre los métodos evaluados, el DE con CR y F Dinámico es el más robusto y eficiente, logrando un fitness óptimo en menos generaciones. Los ajustes dinámicos son una mejora significativa que permite explorar, generalizar y explotar el espacio de búsqueda de manera más eficiente.

Ilustración 12:

Error de distancia alcanzado por los métodos planteados



Análisis general de la ilustración 12: Error de distancias alcanzadas por los métodos.

El gráfico compara el error alcanzado por cuatro variantes del algoritmo de Evolución Diferencial (DE): DE Clásico, DE con CR Dinámico, DE con CR y F Dinámico, y DE Movimiento Natural. Los resultados muestran que los métodos con parámetros dinámicos y

enfoques adaptativos tienden un desempeño notablemente mejor que el DE Clásico en términos de precisión, alcanzando errores significativamente menores.

Comparación entre métodos de la ilustración 11

- **DE Clásico (Azul)**: Este método tiene el mayor error alcanzado (1.35×10^{-7}mm), indicando que su capacidad para encontrar soluciones precisas es limitada en comparación con las otras variantes.
- **DE con CR Dinámico (Verde)**: Logra un error prácticamente nulo (0.000), destacando la mejora obtenida al incorporar un ajuste dinámico de CR.
- **DE con CR y F Dinámico (Rojo)**: Obtiene un error de 6.15×10^{-8}mm, mejorando notablemente la precisión respecto al método clásico y el de CR dinámico, aunque el tiempo de cómputo aumenta.
- **DE Movimiento Natural (Morado)**: Similar al DE con CR dinámico, pero con un enfoque bioinspirados que, aunque no supera a la versión con CR y F dinámico, ofrece alta eficiencia en aplicaciones de robótica.

Diferencias descritas en la ilustración 11

- Los métodos adaptativos (DE con CR Dinámico, DE con CR y F Dinámico, y DE Movimiento Natural) logran errores significativamente menores que el DE Clásico.
- La inclusión de un ajuste dinámico para CR y F (como en el DE con CR y F Dinámico) mejora el rendimiento respecto al DE Clásico, aunque no iguala la precisión de los otros métodos dinámicos.
- El DE Movimiento Natural y el DE con CR Dinámico tienen un rendimiento muy similar, ambos alcanzando errores prácticamente nulos.

Implicaciones Prácticas deducidas de la ilustración 12

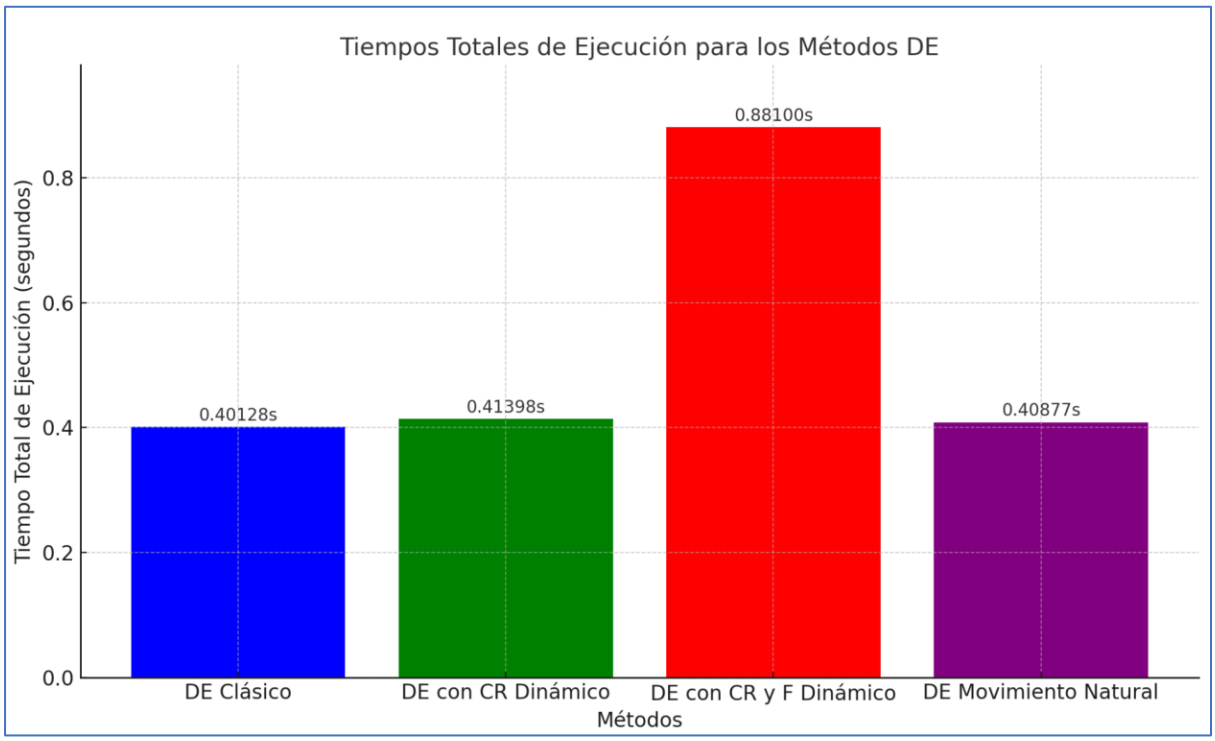
- **Métodos Adaptativos**: Los resultados sugieren que el ajuste dinámico de parámetros (CR, F) y enfoques como el movimiento natural son opciones viables para mejorar la precisión en problemas de optimización complejos. Se pueden reducir los tiempos de convergencia y minimizar resultados inconsistentes, lo cual es relevante para aplicaciones industriales o científicas donde los errores deben ser muy pequeños.
- **DE Clásico**: Aunque es útil para problemas menos exigentes, no es conveniente en tareas que requieren alta precisión y rapidez de convergencia.
- **Balance entre Simplicidad y Rendimiento**: El DE Clásico aporta un enfoque relativamente simple, pero puede no aprovechar de manera óptima los recursos computacionales en aplicaciones exigentes.

Conclusiones de la ilustración 12

Los métodos adaptativos (CR Dinámico y Movimiento Natural) son los más efectivos, alcanzando errores prácticamente nulos y demostrando una alta precisión en la solución del problema. El DE con CR y F Dinámico, aunque no es tan preciso como otros métodos adaptativos, mejora significativamente respecto al DE Clásico, siendo una opción intermedia. El DE Clásico, debido a sus mayores errores alcanzados, es menos adecuado para problemas que requieren una precisión extrema. Estos resultados destacan la importancia de integrar parámetros dinámicos y estrategias adaptativas en algoritmos evolutivos para maximizar la precisión en aplicaciones prácticas.

Ilustración 13:

Tiempos totales de ejecución de los métodos planteados



Análisis General ilustración 13: Tiempos totales de ejecución de los métodos planteados.

El gráfico muestra los tiempos totales de ejecución de cuatro variantes del algoritmo de Evolución Diferencial (DE): DE Clásico, DE con CR Dinámico, DE con CR y F Dinámico, y DE Movimiento Natural. Los resultados destacan que el DE con CR y F Dinámico es el método que registra el tiempo más elevado de ejecución (0.8810 s) en comparación con los demás, mientras que estos últimos presentan tiempos de ejecución similares, rondando los 0.4 s.

Comparación entre Métodos descrita en la ilustración 13

- **DE Clásico (Azul)**: Tiene un tiempo total de ejecución de 0.40128 s, el menor junto con el DE Movimiento Natural (Morado).
- **DE con CR Dinámico (Verde)**: Presenta un tiempo total de 0.41398 s, y asegura una mayor precisión al incorporar el ajuste dinámico del parámetro CR.

- **DE con CR y F Dinámico (Rojo)**: Es el método con mayor tiempo de ejecución, con 0.8810 s, debido a la carga adicional que implica ajustar dinámicamente tanto CR como F.
- **DE Movimiento Natural (Morado)**: Similar al DE Clásico, tiene un tiempo total de ejecución de 0.40877 s, demostrando ser eficiente computacionalmente.

Diferencias

Costo Computacional: El DE con CR y F Dinámico consume más del doble de tiempo que los otros métodos, reflejando la complejidad adicional de ajustar ambos parámetros dinámicamente. Sin embargo, los métodos DE Clásico, DE con CR Dinámico y DE Movimiento Natural tienen tiempos de ejecución similares (con diferencias menores a 0.02 s), indicando que las mejoras en precisión en estos métodos no comprometen significativamente el rendimiento computacional.

Implicaciones Prácticas de la ilustración 13

- **Elección del Método**: En aplicaciones donde el tiempo de ejecución es crítico, los métodos DE Clásico, DE con CR Dinámico y DE Movimiento Natural representan la mejor alternativa debido a su menor consumo de tiempo. No obstante, para contextos donde se exija la máxima precisión, el DE con CR y F Dinámico resulta más adecuado, aunque sea el más costoso en términos de cómputo.

Conclusiones de la ilustración 13

Eficiencia: Los métodos DE Clásico, DE con CR Dinámico y DE Movimiento Natural registran tiempos de ejecución muy similares, representando un menor consumo de recursos cuando la precisión requerida no es extrema. Precisión vs. Tiempo: El DE con CR y F Dinámico ofrece la mayor precisión, pero con un costo computacional más alto. Por lo tanto, la elección final del método dependerá de las necesidades específicas de la aplicación, considerando la relación entre tiempo de ejecución y precisión requerida. Recomendación General: La elección del método debe basarse en la prioridad que se dé al tiempo de ejecución frente a la precisión, de acuerdo con las exigencias del problema de cinemática inversa o aplicación robótica en cuestión.

CAPITULO 5: CONCLUSIONES Y RECOMENDACIONES

5. 1. - Conclusiones

En base al análisis y los resultados obtenidos, se presentan las conclusiones correspondientes a los objetivos específicos planteados en esta investigación:

Limitaciones de los métodos tradicionales:

El análisis comparativo reveló que los métodos tradicionales para resolver la cinemática inversa en manipuladores robóticos presentan limitaciones significativas tanto en precisión como en eficiencia temporal. Si bien pueden proporcionar soluciones viables en escenarios menos complejos, su capacidad para manejar manipuladores con múltiples grados de libertad resulta deficiente, especialmente en aplicaciones de robótica industrial. En contraste, los métodos de Evolución Diferencial (DE) demostraron ser más robustos y sostenibles, brindando mejores resultados en precisión y tiempo de ejecución.

Influencia de los parámetros dinámicos de cruzamiento (CR) y mutación (F):

La inclusión de parámetros dinámicos, como CR y F ajustados en función de las generaciones, tuvo un impacto positivo en la calidad y la diversidad poblacional del modelo DE. La variante con CR y F dinámicos alcanzó el mejor desempeño, con una precisión de 5.12×10^{-8} mm y un tiempo promedio de 0.002963 s. Mientras tanto, los ajustes estáticos resultaron en estancamientos locales y acarrearon la necesidad de mayor número de evaluaciones.

Impacto del modelo Movimiento Natural con CR Dinámico:

El modelo de Movimiento Natural con CR Dinámico integra principios de la biomecánica humana, permitiendo movimientos amplios en articulaciones proximales y ajustes

precisos en articulaciones distales. Con un error promedio de 2.7753 mm y su aplicabilidad en escenarios colaborativos y de interacción humano-robot, se resalta su utilidad práctica, especialmente en tareas que requieren movimientos suaves y naturales.

Comparación de tiempos de ejecución:

La evaluación de los tiempos de ejecución reveló que el modelo Movimiento Natural con CR Dinámico es competitivo frente a los demás métodos. Con un tiempo total de 0.047515 s y un tiempo promedio por generación reducido, demostró ser más eficiente que el DE clásico y comparable con el modelo con CR Dinámico. Aunque el modelo con CR y F dinámicos fue ligeramente más rápido, las diferencias son marginales y no comprometen la viabilidad del modelo de Movimiento Natural en aplicaciones prácticas.

5. 2. - Recomendaciones

- Implementación en Entornos Reales: Validar los modelos de Evolución Diferencial adaptativos, especialmente el modelo de Movimiento Natural con CR Dinámico, en entornos reales para aplicaciones como la interacción humano-robot o la rehabilitación. Esto permitiría replicar patrones de movimiento humano que ofrezcan ventajas significativas.

- Integración con Sistemas de Percepción Avanzados: Ampliar el enfoque mediante la integración con sistemas de visión artificial y sensores hápticos. Esto permitiría una sinergia más avanzada entre el robot y el entorno, desde la detección de obstáculos a la interacción con humanos, mejorando la adaptabilidad y precisión en las tareas a realizar.

- Optimización de Parámetros Dinámicos: Realizar investigaciones más profundas sobre el impacto de diferentes estrategias de ajuste para F y CR, así como su combinación con métodos de aprendizaje de heurísticas avanzadas o redes neuronales.

- Exploración de Enfoques Híbridos: Evaluar la posibilidad de combinar Evolución Diferencial con otros enfoques complementarios. Por ejemplo, fusionarlo con métodos de búsqueda local para acelerar la convergencia en las últimas fases o emplear estrategias multiobjetivo en contextos con objetivos contrapuestos.

- Comparaciones con Métodos Metaheurísticos Modernos: Llevar a cabo estudios comparativos con otros algoritmos avanzados de optimización, como Redes Neuronales Evolutivas, Algoritmos Genéticos o Optimización por Enjambre de Partículas. Esto permitirá posicionar las variantes de DE en un contexto competitivo y establecer una perspectiva global de su desempeño.

- Validación en Escenarios Dinámicos: Probar los modelos en entornos dinámicos donde las condiciones de operación cambian constantemente, como manufactura avanzada o navegación autónoma. Este enfoque permitirá evaluar su eficacia en contextos más complejos y altamente exigentes.

- Divulgación de Resultados: Publicar los hallazgos obtenidos en revistas académicas especializadas y conferencias relacionadas sobre robótica e inteligencia artificial. Esto facilitará la difusión del conocimiento generado y fomentará el intercambio de ideas con la comunidad científica.

5. 3. - REFERENCIAS BIBLIOGRÁFICAS

1. López-Franco C, Hernández-Barragán J, Alanis AY, Arana-Daniel N, López-Franco M. Inverse kinematics of mobile manipulators based on differential evolution. International Journal of Advanced Robotic Systems. 2018;15(1). doi:10.1177/1729881417752738.

2. V. Vázquez-Castillo, J. Torres-Figueroa, E. A. Merchán-Cruz, E. Vega-Alvarado, P. A. Niño-Suárez and R. G. Rodríguez-Cañizo, "Inverse Kinematics Solution of Articulated Robots Using a Heuristic Approach for Optimizing Joint Displacement," in IEEE Access, vol. 10, pp. 63132-63151, 2022, doi: 10.1109/ACCESS.2022.3182496.

3. Linh, B. T. H., & Kung, Y. S. (2015). Inverse kinematics for a five-axis articulated robot arm. Mathematical Problems in Engineering, 2015, 1–10. https://doi.org/10.1155/2015/906505

4. Tiu, H. L., & Chen, P. H. (2013). Reversed recursive transformation for forward kinematics of parallel manipulators. 2013 IEEE International Conference on Robotics and Biomimetics (ROBIO), 568–573. https://doi.org/10.1109/ICSMC.2013.1

5. Kumar, D., & Srivastava, D. (2018). A combined approach of 6-DOF robotic arm using 2-phase hybrid PSO. International Journal of Advanced Robotic Systems, 15(3), 1–14. https://doi.org/10.5772/55892.

6. Olaru, A., & Ghita, A. (2016). Comparison of classical and evolutionary inverse kinematics approaches for humanoid robots. Procedia Technology, 22, 1092–1098. https://doi.org/10.1016/j.protcy.2016.01.086

7. IFR. (2024, September 24). Record 4 million robots in factories worldwide. IFR press release World Robotics 2024 – global market. https://ifr.org/downloads/press2014/2024-SEP-24_IFR_press_release_World_Robotics_2024_-_global_market.pdf

8. Reuters. (2024, 20 de noviembre). China overtakes Germany in industrial use of robots, says report. Recuperado de https://www.reuters.com/technology/china-overtakes-germany-industrial-use-robots-says-report-2024-11-20/

9. Deutsche Welle. (2023, 6 de enero). China planea producir en masa robots humanoides para 2025. Recuperado de https://www.dw.com/es/china-planea-producir-en-masa-robots-humanoides-para-2025/a-67323400

10. Gan, D., Liao, Q., Wei, S., Dai, J., & Qiao, S. (2008). Dual quaternion-based inverse kinematics of the general spatial 7R mechanism. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 222(8), 1593–1598.

11. Husty, M. L. (1996). An algorithm for solving the direct kinematics of general Stewart-Gough platforms. Mechanism and Machine Theory, 31(4), 365–380.

12. Larochelle, P., & McCarthy, M. (n.d.). Inverse kinematics. En M. H. Ang, O. Khatib, & B. Siciliano (Eds.), Encyclopedia of Robotics. Springer.

.

ANEXOS

Anexo 1: Matriz de consistencia

<table>
<tr><th>Problema de Investigación</th><th>Objetivos</th><th>Variables de Investigación</th><th>Metodología</th></tr>
<tr><td>Problema General:</td><td>Objetivo General:</td><td>Variable Independiente:</td><td>Tipo de Investigación: Aplicada</td></tr>
<tr><td rowspan="5">¿En qué medida el uso de algoritmos evolutivos, específicamente el modelo de Evolución Diferencial con ajuste dinámico y movimiento natural, mejora la precisión (error posicional) y la eficiencia (tiempo de ejecución) en la resolución del problema de cinemática inversa en manipuladores robóticos con múltiples grados de libertad, en comparación con los métodos tradicionales?</td><td rowspan="5">Mejorar la precisión y la eficiencia en la resolución del problema de cinemática inversa en manipuladores robóticos con múltiples grados de libertad mediante el diseño y evaluación de variantes avanzadas del modelo de Evolución Diferencial con ajuste dinámico de parámetros y movimiento natural.</td><td>Métodos de resolución de cinemática inversa</td><td>Diseño de Investigación: Experimental</td></tr>
<tr><td>- Indicador: Tipo de método empleado (DE clásico, DE con CR dinámico, DE con CR y F dinámicos, DE con movimiento natural).</td><td rowspan="4">Técnicas: Implementación de algoritmos en MATLAB, simulaciones en un entorno controlado con parámetros del PUMA 560.</td></tr>
<tr><td>Variable Dependiente:</td></tr>
<tr><td>a) Precisión del resultado:
- Indicador: Error posicional.</td></tr>
<tr><td>b) Eficiencia temporal:
- Indicador: Tiempo total de ejecución.</td></tr>
<tr><td>Problemas Específicos:</td><td>Objetivos Específicos:</td><td>Dimensiones de las Variables:</td><td rowspan="5">Instrumentos: Código MATLAB para simulación de métodos DE y medición de resultados.</td></tr>
<tr><td>1. ¿Cuáles son las limitaciones de los métodos tradicionales en términos de precisión y tiempo de ejecución para resolver la cinemática inversa en manipuladores robóticos con múltiples grados de libertad?</td><td>1. Analizar las limitaciones de los métodos tradicionales en términos de precisión y tiempo de ejecución.</td><td rowspan="2">- Precisión del resultado: Diferencia entre la posición calculada y la objetivo (milímetros).</td></tr>
<tr><td>2. ¿Cómo influyeron los parámetros dinámicos de CR y F en la convergencia y diversidad poblacional del modelo de Evolución Diferencial aplicado al problema de cinemática inversa?</td><td>2. Evaluar la influencia de los parámetros dinámicos de CR y F en la convergencia y diversidad del DE.</td></tr>
<tr><td>3. ¿Qué impacto tuvo el modelo Movimiento Natural con CR Dinámico en la precisión y adaptabilidad de las configuraciones articulares generadas para manipuladores robóticos con múltiples grados de libertad?</td><td>3. Examinar el impacto del modelo Movimiento Natural con CR Dinámico en la precisión y adaptabilidad de las configuraciones articulares.</td><td rowspan="2">- Eficiencia temporal: Tiempo total de cálculo desde inicio hasta fin de simulación (segundos).</td></tr>
<tr><td>4. ¿Cómo se comparó el tiempo de ejecución del modelo Movimiento Natural con CR Dinámico frente a los métodos tradicionales, el DE clásico y el modelo CR-F dinámico en la resolución de la cinemática inversa?</td><td>4. Comparar el tiempo de ejecución del modelo Movimiento Natural con CR Dinámico frente a otros métodos.</td></tr>
</table>

Anexo2: Código, Seudocódigo y ejecución de Diferencial Evolutivo DE clásico
https://github.com/joseMauro23/DE_clasico_puma560

ALGORITMO DiferencialEvolutivoPUMA560
INICIO

```
// Parámetros del algoritmo DE
n = 30          // Tamaño de la población
G = 50          // Número máximo de generaciones
F = 0.8         // Factor de escala
Cr = 0.9        // Tasa de cruce
q_min = -180    // Límite inferior de los ángulos articulares
q_max = 180     // Límite superior de los ángulos articulares
P_obj = …       // Posición objetivo deseada del efector final

// Inicialización de la población
población = InicializarPoblación(n, q_min, q_max)
fitness = EvaluarPoblación(población, P_obj)
población = InicializarPoblación(n, q_min, q_max)
fitness = EvaluarPoblación(población, P_obj)

// Evolución
PARA generación = 1 HASTA G HACER
   nuevaPoblación = []
   PARA i = 1 HASTA n HACER
      // Selección de tres individuos aleatorios distintos
      [r0, r1, r2] = SeleccionarIndividuosAleatorios(n)

      // Operador de mutación
      v = población[r0] + F * (población[r1] - población[r2])

      // Operador de cruce
      u = []
      PARA j = 1 HASTA dimension(q_min, q_max)
         SI Rand() < Cr
            u[j] = v[j]
         SINO
            u[j] = población[i][j]

      // Verificar límites
      u = LimitarValores(u, q_min, q_max)

      // Evaluar nueva solución
      fit_u = EvaluarFitness(u, P_obj)

      // Selección
      SI fit_u < fitness[i]
         población[i] = u
         fitness[i] = fit_u
   FIN PARA
FIN PARA
```

```
   // Actualizar la población
   población = nuevaPoblación

   // Mostrar progreso
   Mostrar("Generación ", generación, ": Mejor fitness = ", Mínimo(fitness))
FIN PARA

// Obtener los mejores resultados
mejor_individuo = ObtenerMejorIndividuo(población, fitness)
P_final = CalcularPosiciónFinal(mejor_individuo)
error_posicional = CalcularError(P_final, P_obj)

// Resultados
Mostrar("Mejor solución encontrada: ", mejor_individuo)
Mostrar("Error posicional: ", error_posicional)

FIN FUNCIÓN

// Función para inicializar la población
FUNCIÓN InicializarPoblación(n, q_min, q_max)
   población = []
   PARA i = 1 HASTA n HACER
      individuo = GenerarValoresAleatorios(q_min, q_max)
      población[i] = individuo
   FIN PARA
   RETORNAR población
FIN FUNCIÓN

// Función para evaluar la población
FUNCIÓN EvaluarPoblación(población, P_obj)
   fitness = []
   PARA i = 1 HASTA Tamaño(población) HACER
      fitness[i] = EvaluarIndividuo(población[i], P_obj)
   FIN PARA
   RETORNAR fitness
FIN FUNCIÓN
// Función para evaluar un individuo
FUNCIÓN EvaluarIndividuo(individuo, P_obj)
   T_total ← CalcularTransformaciónTotal(individuo)
   P_actual ← ExtraerPosición(T_total)
   fitness ← DistanciaEuclidiana(P_actual, P_obj)
   RETORNAR fitness
FIN FUNCIÓN

// Función para acotar valores dentro de los límites
FUNCIÓN Acotar(valores, q_min, q_max)
   PARA i ← 1 HASTA Tamaño(valores) HACER
      SI valores[i] < q_min[i] ENTONCES valores[i] ← q_min[i]
      SI valores[i] > q_max[i] ENTONCES valores[i] ← q_max[i]
   FIN PARA
   RETORNAR valores
```

```
FIN FUNCIÓN

// Función de cruce
FUNCIÓN Cruce(target, mutante, CR)
   prueba ← target
   PARA j ← 1 HASTA Tamaño(target) HACER
      SI Random() ≤ CR O j = RandomÍndice(Tamaño(target)) ENTONCES
         prueba[j] ← mutante[j]
      FIN SI
   FIN PARA
   RETORNAR prueba
FIN FUNCIÓN

// Función para calcular la transformación total
FUNCIÓN CalcularTransformaciónTotal(individuo)
   T_total ← MultiplicaciónMatrices(TransformacionesDH(individuo))
   RETORNAR T_total
FIN FUNCIÓN
```

RESULTADOS DE LA EJECUCION DEL PROGRAMA:

```
Generación 996: Mejor fitness = 6.1531e-12
Generación 997: Mejor fitness = 6.1531e-12
Generación 998: Mejor fitness = 6.1531e-12
Generación 999: Mejor fitness = 6.1531e-12
Generación 1000: Mejor fitness = 6.1531e-12

Mejor solución encontrada (ángulos articulares en radianes):
-0.7718    0.3385   -1.4392    3.1416   -3.0663   -0.1852

Posición del efector final (mm):
500.0000
300.0000
200.0000

Error posicional (mm): 6.1531e-12
Tiempo total de ejecución (s): 0.36718
Tiempo promedio por generación (s): 0.0000361
```

Anexo3: Código, Seudocódigo y ejecución de Diferencial Evolutivo DE con CR dinámico
https://github.com/joseMauro23/DE_clasico_puma560

```
ALGORITMO DiferencialEvolutivoConCRDinámico
INICIO
// Parámetros del algoritmo DE
N ← 50 // Tamaño de la población
G_max ← 1000 // Número máximo de generaciones
F ← 0.8 // Factor de escala
q_min ← [-π, -π/2, -π, -π, -π, -π] // Límite inferior de los ángulos articulares
q_max ← [π, π/2, π, π, π, π] // Límite superior de los ángulos articulares
P_obj ← [500, 300, 200] // Punto objetivo (posición deseada del efector final)

// Inicialización de la población
población ← InicializarPoblación(N, q_min, q_max)
fitness ← EvaluarPoblación(población, P_obj)

// Inicialización de la población
población ← InicializarPoblación(N, q_min, q_max)
fitness ← EvaluarPoblación(población, P_obj)

// Evolución
PARA generación ← 1 HASTA G_max HACER
 // Calcular CR dinámico
   CR ← 0.9 * (1 - generación / G_max)

   nueva_población ← []
   PARA i ← 1 HASTA N HACER

 // Seleccionar tres individuos aleatorios distintos de i
      r1, r2, r3 ← SeleccionarIndividuosAleatorios(N, i)

  // Operador de mutación
      mutante ← población[r1] + F * (población[r2] - población[r3])
      mutante ← Acotar(mutante, q_min, q_max)

   // Operador de cruce
      prueba ← Cruce(población[i], mutante, CR)

    // Evaluación del individuo de prueba
      fitness_prueba ← EvaluarIndividuo(prueba, P_obj)

    // Operador de selección
      SI fitness_prueba < fitness[i] ENTONCES
         nueva_población[i] ← prueba
         fitness[i] ← fitness_prueba
      SINO
         nueva_población[i] ← población[i]
      FIN SI
```

```
    FIN PARA
FIN PARA
// Actualizar la población
    población ← nueva_población
FIN PARA

// Obtener los mejores resultados
mejor_individuo ← ObtenerMejorIndividuo(población, fitness)
P_final ← CalcularPosiciónFinal(mejor_individuo)
error_posicional ← CalcularError(P_final, P_obj)

// Mostrar resultados
Mostrar("Mejor solución encontrada: ", mejor_individuo)
Mostrar("Posición del efector final: ", P_final)
Mostrar("Error posicional: ", error_posicional)
FIN

// Función para inicializar la población
FUNCIÓN InicializarPoblación(N, q_min, q_max)
    población ← []
    PARA i ← 1 HASTA N HACER
        individuo ← GenerarValoresAleatorios(q_min, q_max)
        población[i] ← individuo
    FIN PARA
    RETORNAR población
FIN FUNCIÓN

// Función para evaluar toda la población
FUNCIÓN EvaluarPoblación(población, P_obj)
    fitness ← []
    PARA i ← 1 HASTA Tamaño(población) HACER
        fitness[i] ← EvaluarIndividuo(población[i], P_obj)
    FIN PARA
    RETORNAR fitness
FIN FUNCIÓN

// Función para evaluar un individuo
FUNCIÓN EvaluarIndividuo(individuo, P_obj)
    T_total ← CalcularTransformaciónTotal(individuo)
    P_actual ← ExtraerPosición(T_total)
    fitness ← DistanciaEuclidiana(P_actual, P_obj)
    RETORNAR fitness
FIN FUNCIÓN
// Función para acotar valores dentro de los límites

FUNCIÓN Acotar(valores, q_min, q_max)
    PARA i ← 1 HASTA Tamaño(valores) HACER
        SI valores[i] < q_min[i] ENTONCES valores[i] ← q_min[i]
        SI valores[i] > q_max[i] ENTONCES valores[i] ← q_max[i]
    FIN PARA
    RETORNAR valores
```

```
FIN FUNCIÓN

// Función de cruce
FUNCIÓN Cruce(target, mutante, CR)
   prueba ← target
   PARA j ← 1 HASTA Tamaño(target) HACER
      SI Random() ≤ CR O j = RandomÍndice(Tamaño(target)) ENTONCES
         prueba[j] ← mutante[j]
      FIN SI
   FIN PARA
   RETORNAR prueba
FIN FUNCIÓN

// Función para calcular la transformación total
FUNCIÓN CalcularTransformaciónTotal(individuo)
   T_total ← MultiplicaciónMatrices(TransformacionesDH(individuo))
   RETORNAR T_total
FIN FUNCIÓN
```

RESULTADOS DE LA EJECUCION DEL PROGRAMA:

Generación 989: Mejor fitness = 2.8786e-05, CR dinámico = 0.0099
Generación 990: Mejor fitness = 2.8786e-05, CR dinámico = 0.009
Generación 991: Mejor fitness = 2.8786e-05, CR dinámico = 0.0081
Generación 992: Mejor fitness = 2.8786e-05, CR dinámico = 0.0072
Generación 993: Mejor fitness = 2.8786e-05, CR dinámico = 0.0063
Generación 994: Mejor fitness = 2.8786e-05, CR dinámico = 0.0054
Generación 995: Mejor fitness = 2.8786e-05, CR dinámico = 0.0045
Generación 996: Mejor fitness = 2.8786e-05, CR dinámico = 0.0036
Generación 997: Mejor fitness = 2.8786e-05, CR dinámico = 0.0027
Generación 998: Mejor fitness = 2.8786e-05, CR dinámico = 0.0018
Generación 999: Mejor fitness = 2.8786e-05, CR dinámico = 0.0009
Generación 1000: Mejor fitness = 2.8786e-05, CR dinámico = 0

Mejor solución encontrada (ángulos articulares en radianes):
   -0.7718   -1.0203   1.5273   3.1416   -3.0852   -3.1416

Posición del efector final (mm):
   500.0000
   300.0000
   200.0000

Error posicional (mm): 2.8786e-05
Tiempo total de ejecución (s): 0.36885
Tiempo promedio por generación (s): 0.00029779

Anexo 4: Código, Seudocódigo y ejecución de DE con CR y F dinámico.
https://github.com/joseMauro23/DE_clasico_puma560

```
ALGORITMO DiferencialEvolutivoConCRyFDinamicos
INICIO

// Parámetros iniciales
N ← 50 // Tamaño de la población
D ← 6  // Dimensionalidad (ángulos articulares: q1, q2, q3, q4, q5, q6)
iteraciones_max ← 1000 // Número máximo de iteraciones
limites_min ← [-π, -π/2, -π, -π, -π, -π] // Límite inferior
limites_max ← [π, π/2, π, π, π, π]       // Límite superior
posicion_objetivo ← [500, 300, 200]      // Posición deseada del efector final

// Inicializar población aleatoria
población ← InicializarPoblación(N, D, limites_min, limites_max)

// Iniciar el temporizador
IniciarTemporizador()

// Inicializar métricas
mejor_individuo ← null
mejor_error ← infinito
errores_por_generacion ← VectorDeCeros(iteraciones_max)

// Bucle principal
PARA iter ← 1 HASTA iteraciones_max HACER
   // Ajustar F y CR dinámicamente
   F ← 1.2 - (iter / iteraciones_max) * (1.8 - 0.2) // F decrece de 1.2 a 0.5
   CR ← 0.9 - (iter / iteraciones_max) * (0.9 - 0.1) // CR decrece de 0.9 a 0.1

   // Evolución de la población
   PARA i ← 1 HASTA N HACER
      // Seleccionar tres individuos aleatorios diferentes de i
      r1, r2, r3 ← SeleccionarTresIndividuos(N, i)

      // Operador de mutación
      V ← r1 + F * (r2 - r3)
      V ← AplicarLímites(V, limites_min, limites_max)
      // Operador de cruce
      U ← Recombinar(población[i], V, CR)
      U ← AplicarLímites(U, limites_min, limites_max)

      // Evaluar la nueva solución y realizar la selección
      error_U ← EvaluarIndividuo(U, posicion_objetivo)
      error_actual ← EvaluarIndividuo(población[i], posicion_objetivo)
      SI error_U < error_actual ENTONCES
         población[i] ← U
      FIN SI

   FIN PARA
```

```
   // Evaluar la población actual
   errores ← EvaluarPoblación(población, posicion_objetivo)
   mejor_error_iter, mejor_idx ← ObtenerMejorError(errores)
   mejor_individuo ← población[mejor_idx]
   mejor_error ← mejor_error_iter
   errores_por_generacion[iter] ← mejor_error

FIN PARA

// Calcular posición del efector final con el mejor individuo
x_final, y_final, z_final ← CalcularPosiciónFinal(mejor_individuo)

// Métricas finales
tiempo_total ← DetenerTemporizador()
tiempo_promedio ← tiempo_total / iteraciones_max

// Mostrar resultados finales
Mostrar("Mejor solución encontrada: ", mejor_individuo)
Mostrar("Posición final del efector: x = ", x_final, ", y = ", y_final, ", z = ", z_final)
Mostrar("Mejor error alcanzado: ", mejor_error)
Mostrar("Tiempo total de ejecución: ", tiempo_total, " segundos")
Mostrar("Tiempo promedio por generación: ", tiempo_promedio, " segundos")

FIN

FUNCIÓN InicializarPoblación(N, D, limites_min, limites_max)
   población ← []
   PARA i ← 1 HASTA N HACER
      individuo ← GenerarAleatorioDentroDeLímites(D, limites_min, limites_max)
      población ← población + [individuo]
   FIN PARA
   RETORNAR población
FIN FUNCIÓN
FUNCIÓN EvaluarIndividuo(individuo, posicion_objetivo)

   x, y, z ← CalcularEfector(individuo)

   error ← DistanciaEuclidiana([x, y, z], posicion_objetivo)

   RETORNAR error

FIN FUNCIÓN

FUNCIÓN Recombinar(individuo, mutante, CR)
   hijo ← individuo
   PARA j ← 1 HASTA D HACER
      SI Random() ≤ CR ENTONCES
         hijo[j] ← mutante[j]
      FIN SI
   FIN PARA
   RETORNAR hijo
FIN FUNCIÓN
```

```
FUNCIÓN AplicarLímites(vector, limites_min, limites_max)
  PARA j ← 1 HASTA Tamaño(vector) HACER
    vector[j] ← Máximo(limites_min[j], Mínimo(vector[j], limites_max[j]))
  FIN PARA
  RETORNAR vector
FIN FUNCIÓN

FUNCIÓN CalcularEfector(individuo)
  q1, q2, q3, q4, q5, q6 ← individuo
  x, y, z ← CalcularPosiciónPorModelo(q1, q2, q3, q4, q5, q6)
  RETORNAR x, y, z
FIN FUNCIÓN

FUNCIÓN CalcularPosiciónFinal(individuo)
  x, y, z ← CalcularEfector(individuo)
  RETORNAR x, y, z
FIN FUNCIÓN
```

RESULTADOS DE LA EJECUCION DEL PROGRAMA:

Generación 13: Mejor fitness = 189.021458, CR dinámico = 0.889680, F dinámico = 1.179200
Generación 14: Mejor fitness = 189.021458, CR dinámico = 0.888800, F dinámico = 1.177600
Generación 15: Mejor fitness = 189.021458, CR dinámico = 0.888000, F dinámico = 1.176000
Generación 16: Mejor fitness = 189.021458, CR dinámico = 0.887200, F dinámico = 1.174400
Generación 17: Mejor fitness = 189.021458, CR dinámico = 0.886400, F dinámico = 1.172800
Generación 18: Mejor fitness = 189.021458, CR dinámico = 0.885600, F dinámico = 1.171200

--- RESULTADOS FINALES ---

Generaciones: 1000
Mejor error alcanzado: 0.000000

Posición final del efector: x = 500.000000, y = 300.000000, z = 200.000000

Ángulos articulares óptimos (radianes): q1 = -0.7718, q2 = -1.0204, q3 = 1.5538, q4 = -1.5275, q5 = -3.1416, q6 = 0.1063

Tiempo total de ejecución: 0.9058 segundos
Tiempo promedio por generación: 0.000905 segundos

Anexo 5: Código, Seudocódigo y ejecución de Diferencial Evolutivo con movimiento natural
https://github.com/joseMauro23/DE_clasico_puma560

```
ALGORITMO DiferencialEvolutivoConMovimientoNatural
INICIO

// Parámetros iniciales
N ← 50 // Tamaño de la población
D ← 6  // Dimensionalidad (ángulos articulares: q1, q2, q3, q4, q5, q6)
G_max ← 1000 // Número máximo de generaciones
F ← 0.8 // Factor de escala
limites_min ← [-π, -π/2, -π, -π, -π, -π] // Límite inferior
limites_max ← [π, π/2, π, π, π, π]       // Límite superior
posicion_objetivo ← [500, 300, 200]      // Posición deseada del efector final

// Inicializar población aleatoria
población ← InicializarPoblación(N, D, limites_min, limites_max)

// Iniciar el temporizador
IniciarTemporizador()

// Inicializar métricas
mejor_individuo ← null
mejor_error ← infinito
errores_por_generacion ← VectorDeCeros(G_max)

// Pesos jerárquicos y ajustes de CR
w ← [1, 1, 0.8, 0.7, 0.6, 0.5] // Pesos jerárquicos por articulación
Delta_CR ← [0.1, 0.1, 0, 0, 0, 0] // Ajuste adicional para articulaciones principales

// Bucle principal
PARA G ← 1 HASTA G_max HACER
   // Calcular CR base dinámico
   CR_base ← 0.9 * (1 - G / G_max)

 // Evolución de la población
   PARA i ← 1 HASTA N HACER
      // Seleccionar tres individuos aleatorios diferentes de i
      r1, r2, r3 ← SeleccionarTresIndividuos(N, i)

      // Operador de mutación
      V ← r1 + F * (r2 - r3)
      V ← AplicarLímites(V, limites_min, limites_max)

      // Operador de cruce jerárquico ajustado
      U ← RecombinarJerárquico(población[i], V, CR_base, w, Delta_CR)
      // Evaluar la nueva solución y realizar la selección
      error_U ← EvaluarIndividuo(U, posicion_objetivo)
      error_actual ← EvaluarIndividuo(población[i], posicion_objetivo)
      SI error_U < error_actual ENTONCES
```

```
            población[i] ← U
        FIN SI
    FIN PARA

    // Evaluar la población actual
    errores ← EvaluarPoblación(población, posicion_objetivo)
    mejor_error_iter, mejor_idx ← ObtenerMejorError(errores)
    mejor_individuo ← población[mejor_idx]
    mejor_error ← mejor_error_iter
    errores_por_generacion[G] ← mejor_error

    // Mostrar progreso
    Mostrar("Generación ", G, ": Mejor fitness = ", mejor_error, ", CR_base = ", CR_base)
FIN PARA

// Calcular posición del efector final con el mejor individuo
x_final, y_final, z_final ← CalcularPosiciónFinal(mejor_individuo)

// Métricas finales
tiempo_total ← DetenerTemporizador()
tiempo_promedio ← tiempo_total / G_max

// Mostrar resultados finales
Mostrar("Mejor solución encontrada: ", mejor_individuo)
Mostrar("Posición final del efector: x = ", x_final, ", y = ", y_final, ", z = ", z_final)
Mostrar("Mejor error alcanzado: ", mejor_error)
Mostrar("Tiempo total de ejecución: ", tiempo_total, " segundos")
Mostrar("Tiempo promedio por generación: ", tiempo_promedio, " segundos")

FIN

FUNCIÓN InicializarPoblación(N, D, limites_min, limites_max)
    población ← []
    PARA i ← 1 HASTA N HACER
        individuo ← GenerarAleatorioDentroDeLímites(D, limites_min, limites_max)
        población ← población + [individuo]
    FIN PARA
    RETORNAR población
FIN FUNCIÓN
FUNCIÓN RecombinarJerárquico(target, mutante, CR_base, w, Delta_CR)
    hijo ← target
    PARA j ← 1 HASTA D HACER
        CR_j ← CR_base * (0.5 + w[j]) + Delta_CR[j]
        SI Random() ≤ CR_j ENTONCES
            hijo[j] ← mutante[j]
        FIN SI
    FIN PARA
    RETORNAR hijo
FIN FUNCIÓN

FUNCIÓN CalcularEfector(individuo)
```

```
   q1, q2, q3, q4, q5, q6 ← individuo
   x, y, z ← CalcularPosiciónPorModelo(q1, q2, q3, q4, q5, q6)
   RETORNAR x, y, z
FIN FUNCIÓN

FUNCIÓN CalcularPosiciónFinal(individuo)
   x, y, z ← CalcularEfector(individuo)
   RETORNAR x, y, z
FIN FUNCIÓN

FUNCIÓN AplicarLímites(vector, limites_min, limites_max)
   PARA j ← 1 HASTA Tamaño(vector) HACER
      vector[j] ← Máximo(limites_min[j], Mínimo(vector[j], limites_max[j]))
   FIN PARA
   RETORNAR vector
FIN FUNCIÓN
```

RESULTADOS DE LA EJECUCION DEL PROGRAMA:

```
Generación 996: Mejor fitness = 0, CR dinámico = 0.0036
Generación 997: Mejor fitness = 0, CR dinámico = 0.0027
Generación 998: Mejor fitness = 0, CR dinámico = 0.0018
Generación 999: Mejor fitness = 0, CR dinámico = 0.0009
Generación 1000: Mejor fitness = 0, CR dinámico = 0

Mejor solución encontrada (ángulos articulares en radianes):
   -0.7718    0.3386   -1.4281    0.4861   -3.1416   -0.4059

Posición del efector final (mm):
   500
   300
   200

Error posicional (mm): 0
Tiempo total de ejecución (s): 0.36754
Tiempo promedio por generación (s): 0.00029945
```