POLITÉCNICA

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES
UNIVERSIDAD POLITÉCNICA DE MADRID

José Gutiérrez Abascal, 2. 28006 Madrid
Tel.: 91 336 3060
info.industriales@upm.es

www.industriales.upm.es

Jaime Bravo Algaba

05 TRABAJO FIN DE GRADO

INDUSTRIALES

TRABAJO FIN DE GRADO

OPTIMIZACIÓN DEL DISEÑO DE UN ROBOT HIPER - REDUNDANTE DE CABLES MEDIANTE ALGORITMO GENÉTICO

TRABAJO FIN DE GRADO PARA LA OBTENCIÓN DEL TÍTULO DE GRADUADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

SEPTIEMBRE **2021**

JAIME BRAVO ALGABA

DIRECTOR DEL TRABAJO FIN DE GRADO:
Antonio Barrientos Cruz

INDUSTRIALES
ETSII | UPM

POLITÉCNICA

“When you grow up and something touches you emotionally, it lives on forever.”

David Edward Goldberg

AGRADECIMIENTOS

A mis padres, por haberme dado medios para crecer, consejos para avanzar y fuerza para levantarme.

A mi abuela, por todo lo que habrá podido llegar a rezar para que aprobara Cálculo.

A mi tío, por haber apostado siempre por mí.

Gracias a los Gerardos, por tantos ratos en la multi y, sobre todo, por ser auténticos.

A mi compañera Elena, por haber escuchado mis audios con paciencia...

Agradezco también a mi tutor Antonio por sus consejos y por la implicación que demuestra día a día con los estudiantes.

Y a todos los que me habéis acompañado durante esta etapa de mi vida, con momentos buenos, momentos malos y momentos peores, pero que me ha ayudado a crecer como persona y de la que me llevo gente maravillosa.

RESUMEN

En los últimos años los robots Hiperredundantes están siendo aplicados en una gran variedad de ámbitos – *medicina, control de fabricación en la industria aeroespacial, etc.*-. Esto ha propiciado un aumento de la labor investigadora sobre esta tecnología en aspectos como la optimización mecánica, el perfeccionamiento del control cinemático o el uso de nuevos materiales. Tanto es así que empresas como Tesla o la NASA también han hecho aportaciones al respecto, como por ejemplo el famoso *"Tesla Snake Charger"*, que pretendía dar solución a la carga automática de sus vehículos usando un robot hiperredundante.

Figura 1: Tesla Snake Charger.
Fuente: https://www.cnet.com

Los robots hiperredundantes son aquellos que disponen de un elevado número de grados de libertad. Esto les dota de una agilidad y maniobrabilidad significativamente superiores a la de los robots manipuladores convencionales. Estos robots están pensados para trabajar en entornos angostos y de difícil acceso, pues son capaces de adoptar multitud de formas y de maniobrar con gran facilidad. Áreas como la cirugía invasiva, la inspección de piezas aeroespaciales o incluso labores de rescate son algunas de sus posibles aplicaciones. Su diseño puede compararse con el cuerpo de algunos seres vivos, estando típicamente inspirados por la forma de serpientes, tentáculos o incluso la trompa de los elefantes.

En la Escuela Técnica Superior de Ingenieros Industriales (ETSII – UPM) se lleva trabajando varios años con este tipo de robots, contribuyendo a la investigación y la formación de los estudiantes en esta materia. Durante la fase de diseño y construcción de un robot de estas características, surgen numerosas dudas sobre las decisiones a tomar: cuál es el número óptimo de articulaciones, su longitud, diámetro de los eslabones, número de actuadores, etc. Dichas decisiones se deben abordar desde la perspectiva de conseguir un diseño optimizado tanto en aspectos técnicos como económicos. Se trata de robots de construcción compleja, por lo que estas cuestiones no tienen en absoluto una respuesta trivial.

Las últimas actividades llevadas a cabo en la ETSII a este respecto vienen de la mano de Elena Muñoz con la construcción del "Pylori I". Este robot se encuadra dentro de la categoría "Cable Driven Hyper-Redundant" de tipo Serpentino. El término serpentino indica que los eslabones que lo componen son rígidos, pero debido a su elevado número se pueden estudiar como un sistema continuo. En el caso citado, dichos eslabones o vértebras constan de discos fabricados mediante Impresión 3D. Dichos discos han sido diseñados con su cara inferior plana y la superior acabada en cuña, permitiendo así un movimiento

pivotante entre ellos. El cuerpo del robot se compone de una cadena de discos superpuestos a lo largo de su eje vertical, de manera que el conjunto resultante se asemeja a una columna vertebral – *ver figura 2*-.

La unión entre los discos se realiza a base de cables tensores -*actuadores*-, que son los encargados del movimiento de los diferentes tramos del robot. El cuerpo del robot está dividido en diferentes tramos denominados *"secciones"*, cada una de las cuales tiene sus propios actuadores y se mueve de manera independiente a las demás. En la siguiente figura se representa un robot de este tipo, compuesto por cuatro secciones de distinta longitud.

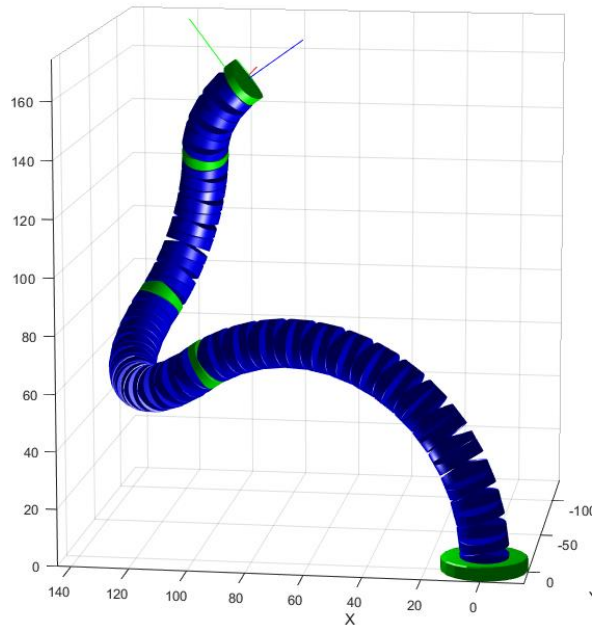


Figura 2: Simulación de Robot Hiperredundante de Discos Pivotantes.
Fuente: *elaboración propia.*

La complejidad del diseño de estos mecanismos tiene su origen en la multitud de parámetros a tener en cuenta para conseguir un desempeño óptimo. La construcción a base de discos pivotantes presenta una problemática particular derivada del elevado número de eslabones que componen el cuerpo del robot. Dicha problemática atiende a cuatro aspectos fundamentales para el diseño:

- Número total de discos que compondrán el cuerpo del robot.
- Número de tramos y su longitud.
- Ángulo máximo de pivote, así como de la dirección en la que se producirá.
- Número de actuadores.

Las múltiples combinaciones posibles en todos estos parámetros dificultan la obtención de expresiones analíticas que optimizar. La complejidad del modelo cinemático de estos robots, así como la confluencia de los efectos de múltiples variables de diseño entorpecen la formulación de funciones de coste.

Por otro lado, esta tecnología está siendo desarrollada en un ecosistema muy particular: el nacimiento de la Industria 4.0, también conocida como la Cuarta Revolución Industrial, que tiene como punta de lanza el uso de la **Inteligencia Artificial** como herramienta clave. Ámbitos como el Big Data, robótica industrial o el reconocimiento de imagen están renovando la industria actual, optimizando y agilizando los diferentes procesos productivos.

Todas estas tecnologías propician el caldo de cultivo perfecto para este trabajo, conectando el mundo de la Inteligencia Artificial con la investigación sobre robots hiperredundantes. Por todo lo anteriormente expuesto, **se pretende abordar la optimización del diseño de un robot de tipo CDHRR de Discos Pivotantes haciendo uso de un Algoritmo Genético de implementación propia**. Se requerirá, por tanto, de herramientas que faciliten esta tarea, permitiendo simular y evaluar las soluciones obtenidas. Las áreas de trabajo que componen el trabajo han sido:

- Desarrollo de un entorno de simulación, que permita la visualización del robot, así como la elaboración de gráficos con datos sobre su desempeño.
- Implementación del modelo cinemático del robot, facilitando su control y posicionamiento en diferentes puntos.
- Implementación de un algoritmo genético capaz de trabajar con los robots en cuestión, y que tenga en cuenta los diferentes aspectos y parámetros a optimizar.

El entorno de simulación ha sido desarrollado en Matlab, dada la gran cantidad de herramientas matemáticas que proporciona, así como la posibilidad del trabajo en entornos 3D. Tras hacer un estudio de las diferentes partes que componen un robot CDHRR de Discos Pivotantes, se ha implementado una serie de clases desde la perspectiva de la Programación Orientada a Objetos. Esto ha favorecido en gran medida la modularidad del programa y hace mucho más intuitivo su uso.

Las diferentes funcionalidades del programa atienden a las necesidades de la tarea a llevar a cabo: posicionamiento, visualización, cálculo del volumen de trabajo, cálculo de la matriz jacobiana o ejecución de trayectorias son algunas de estas funcionalidades. El resultado ha sido una herramienta muy útil a la hora de diseñar un robot de este tipo por la amplia variedad de aspectos que permite evaluar.

Ha sido el control cinemático el que ha requerido de un mayor esfuerzo desde el punto de vista matemático y de programación. Los métodos actuales para el cálculo de la cinemática de robots continuos se basan en los trabajos de Webster y Jones (1) sobre la Hipótesis de Curvatura Constante -*conocida por las siglas PCC*-. Esta hipótesis supone que la silueta del robot describe una curva formada por arcos tangentes entre sí. Esta metodología da buenos resultados en una amplia gama de robots, aunque empieza a ser reemplazada por el método conocido como *"Cosserat's Rods"* por su mayor precisión.

Ambas metodologías tienen como objetivo el cálculo de la silueta del robot, también llamada *"Backbone curve".* Sin embargo, en el presente trabajo se pretende simular la pose de todos y cada uno de los discos que componen al robot, para poder así evaluar las diferentes configuraciones de estos. Por ello, no resulta tan interesante el cálculo de la curva de backbone, si no de la pose de cada disco. Por este motivo, se ha desarrollado un método

numérico para el cálculo de la pose exacta de cada disco del robot, obteniendo simulaciones como la mostrada anteriormente en la Figura (2).

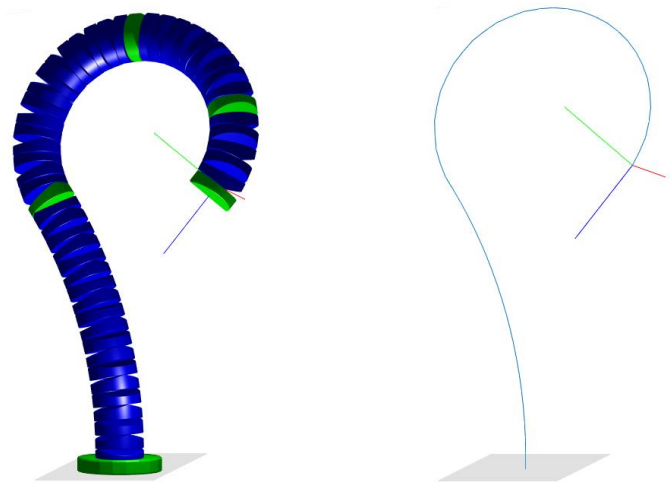


Figura 3: comparativa entre la simulación conseguida (izquierda) y la simulación de la curva de backbone (derecha).

Este método resuelve el problema cinemático inverso de las secciones del robot, obteniendo la pose exacta que deben adoptar los discos dada una posición objetivo para el extremo de la sección. Cabe destacar que se ha seguido la hipótesis de curvatura constante, de manera que la forma adoptada por las secciones se corresponde con arcos de circunferencia.

El control cinemático conseguido utiliza dos metodologías diferentes para el posicionamiento del robot:

- Posicionamiento mediante el Descenso del Gradiente.
- Posicionamiento mediante Matriz Jacobiana
- Aproximación por trayectorias.

Durante el trabajo se expondrán las ventajas e inconvenientes de cada una de ellas.

El algoritmo genético implementado utiliza todas estas funcionalidades para la obtención de una configuración óptima. El criterio de selección de esa configuración se basa en la evaluación de una función de coste – *la cual se deberá maximizar o minimizar en función del objetivo deseado*-. La función de coste debe evaluar una serie de parámetros que informan sobre el desempeño del robot en una determinada tarea. Estos son los llamados *"Índices de desempeño"* del robot. Se han tenido en cuenta tres de estos Índices: Manipulabilidad, Volumen de Trabajo y GMI -*Global Manipulability Index"-.*

Por último, los resultados arrojados por el algoritmo se han comparado con diferentes configuraciones elegidas según patrones prediseñados. Dichos patrones responden a distribuciones matemáticas de las que se prevé un comportamiento óptimo: Sucesión de Fibonacci, Relación Áurea, evolución exponencial, entre otras. Se realizará un análisis sobre cómo estas configuraciones presentes en gran cantidad de motivos geométricos de la naturaleza – *caracolas, plantas, flores, animales…*- también arrojan buenos resultados en el diseño de robots de tipo CDHRR.

Código del programa desarrollado

https://github.com/JaimeBravoAlgaba/Optimizaci-n-de-CDHRR-de-Discos-Pivotantes-Mediante-Algoritmo-Gen-tico.git

Palabras clave: robot hiperredundante, robot CDHRR, algoritmo genético, algoritmos evolutivos, optimización, simulación, robótica, automática.

Códigos Unesco

1203.26: Simulación

1203.04: Inteligencia Artificial

1206.08: Métodos Iterativo

3311.01: Tecnología de la Automatización

ÍNDICE

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN Y OBJETIVOS

El presente trabajo consiste en la simulación y optimización en Matlab de un Robot Hiperredundante, ayudándose para ello de un algoritmo genético de implementación propia. El adjetivo *"hiperredundante"* hace alusión al gran número de grados de libertad que poseen estos robots, dotándolos de una destreza y movilidad significativamente superiores a los manipuladores convencionales. En contraposición, se trata de robots realmente complejos desde los puntos de vista constructivo, cinemático y de control.

Pero ¿qué es un robot hiperredundante? Para responder a esta pregunta en primer lugar conviene definir qué es un **robot manipulador**. Un robot manipulador consiste en una cadena cinemática, generalmente en serie, en cuyo extremo posee una herramienta que le permite manipular objetos. Los grados de libertad de la cadena cinemática son los que determinan el espacio de trabajo y la agilidad del manipulador. Un ejemplo clásico puede ser un brazo robótico: en función del número de articulaciones del brazo su movilidad será mayor o menor.

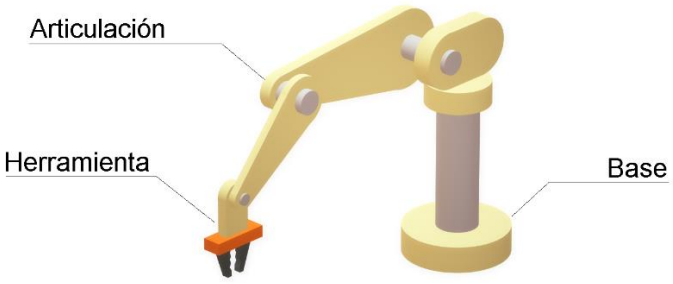


Figura 4: Robot manipulador. Fuente: *elaboración propia.*

El mínimo número de grados de libertad necesario para que el manipulador pueda trabajar en un espacio tridimensional es seis: tres para posicionamiento y tres para orientación (2). No obstante, este número puede hacerse mayor para conseguir un mejor desempeño: mayor movilidad, más poses posibles, mayor capacidad de esquivar obstáculos… En este caso, se habla de **robots redundantes**, que tienen más grados de libertad de los estrictamente necesarios para llevar a cabo una determinada tarea.

Los robots hiperredundantes llevan esto al extremo, superando con creces el mínimo número de GdL estrictamente necesario. Según (3), estos robots presentan una serie de ventajas frente a los manipuladores convencionales. Algunas de ellas, como ya se ha citado, tienen que ver con aspectos cinemáticos. Destreza, movilidad, evitar obstáculos, son algunas de ellas. Otras tienen que ver con su robustez, en el sentido de resistencia a fallos de algunos de sus componentes. Al disponer de un gran número de GdL, el fallo en, por ejemplo, una de sus articulaciones no tendría por qué suponer que el robot quede inutilizable.

Figura 5: "Series II X125" de OC Robotics. Fuente: https://www.wevolver.com/wevolver.staff/series.ii.x125.system

Debido a su compleja construcción, **resulta complicado realizar un diseño óptimo de este tipo de robots**. Un mayor número de GdL generalmente supondrá un mayor número de articulaciones y, en definitiva, **más parámetros de diseño**. La elección de estos parámetros no es aleatoria en ningún caso, si no que debe estar justificada respondiendo a una serie de especificaciones, como pueden ser el peso que debe poder manejar el robot, el espacio de trabajo en el que podrá moverse o su capacidad de introducirse por huecos o cavidades intrincadas. Los parámetros de diseño – *longitud del robot, distribución de articulaciones a lo largo del cuerpo, número de actuadores…* - deben establecerse de forma que el diseño final, no sólo sea capaz de suplir las necesidades descritas, sino que además lo haga de manera eficiente y con un diseño viable tanto económica como tecnológicamente.

Desde el punto de vista matemático, podrían obtenerse expresiones de forma analítica que permitieran la optimización de ciertas características del robot. Manipulabilidad, Volumen de Trabajo, Condicionamiento Dinámico, Elipsoide de Velocidad… son algunos de los conocidos como "Índices de Desempeño" del robot (4). No obstante, dada la naturaleza de este tipo de robots la matemática asociada será significativamente más compleja que la utilizada en manipuladores convencionales. Por otro lado, las técnicas de control tradicionales están todavía siendo adaptadas a las necesidades de estos robots en concreto, por lo que la optimización puede convertirse en una tarea arduo complicada.

Por la gran cantidad de soluciones posibles y combinaciones para estos robots, un posible enfoque de este problema tiene que ver con el uso de algoritmos de optimización e Inteligencia Artificial. Por la naturaleza de los robots hiperredundantes – *casi infinitas posibilidades de configuración, gran cantidad de articulaciones…*- se ha optado por la implementación en Matlab de un Algoritmo Genético.

Lo Algoritmos Genéticos consisten en la **optimización de una población de individuos cruzando y mutando sus genes generación tras generación, evaluándolos y priorizando los genes de aquellos que tengan un mejor desempeño**. En un ecosistema natural, los individuos mejor adaptados -*animales y seres vivos en general*- son los que sobrevivirán con mayor probabilidad y, en consecuencia, serán los más propensos a transmitir sus genes a la siguiente generación.

Basándose en esto, el algoritmo genético implementado genera una gran muestra de robots con configuraciones aleatorias. Los **genes de estos individuos serán los parámetros de diseño a optimizar**. Tras evaluar los robots mediante una función de coste previamente elegida, se cruzarán y combinarán los genes de aquellos que hayan tenido un mejor desempeño, generando así una nueva población. Tras una serie de iteraciones, los individuos **habrán heredado los genes de sus antecesores mejor adaptados**, optimizando así la información genética de la población en su conjunto. De tal modo, ha sido posible obtener una serie de configuraciones óptimas para determinadas tareas – *relacionadas con el posicionamiento del robot en un espacio de trabajo dado*-.

El presente trabajo se ha centrado en la optimización de un robot de tipo *"Cable Driven Hyper-Redundant Robot"* – *CDHRR*-, cuyas características serán descritas más adelante. Para hacer esto posible, ha sido necesario el desarrollo de una serie de funciones y clases en Matlab para el diseño, control y simulación de estos robots. La elaboración de este programa ha conllevado un gran esfuerzo desde el punto de vista matemático y de programación, pero sin duda ha resultado en una herramienta útil y robusta, que permite el control y visualización de estos robots. El entorno de desarrollo elegido ha sido Matlab, dada su simplicidad y versatilidad, así como por la cantidad de funciones y herramientas de cálculo matemático que ofrece - *operaciones con matrices, generación de gráficos tanto 2D como 3D, entre otras muchas-.*

2. ESTADO DEL ARTE

2.1 Clasificación de Robots Hiperredundantes

Como se ha mencionado, los robots hiperredundantes disponen de un número de GdL muy superior al estrictamente necesario. Este gran número de GdL puede lograrse de muchas maneras: utilizando materiales blandos, disponiendo un gran número de articulaciones, utilizando actuadores de diversos tipos… En definitiva, existe una gran variedad de robots de este tipo. A continuación, se expone una breve clasificación de los diferentes tipos de robots hiperredundantes que existen, según se expone en (2):

- ***Según funcionalidad:*** una primera clasificación puede hacerse atendiendo a la utilidad del robot en cuestión. Así, se diferenciarán dos grandes grupos:
  - *Manipuladores:* su función es trabajar o hacer determinadas tareas con objetos. Suelen tener una base fija sobre la que están montados.
  - *Móviles:* robots con capacidad de hacer desplazamientos por el terreno.
  - *Híbridos:* generalmente se trata de manipuladores cuya base es móvil, combinando ambas funcionalidades.
- ***Según el número de GdL:*** se puede diferenciar entre robots con un número de GdL menor o igual al necesario, o robots que tendrán más grados de libertad que los estrictamente necesarios. Así se diferencian dos grandes grupos: *no redundantes* y *redundantes*. Cuando el número de GdL es muy elevado, se habla de *hiperredundantes*. Siendo N el mínimo número de GdL necesarios, el criterio actual es el siguiente:
  - *No redundantes:* $GdL \leq N$
  - *Redundantes:* $N \leq GdL < 2N$
  - *Hiperredundantes:* $2N \leq GdL$

  El criterio diferenciador entre redundantes e hiperredundantes es difuso, dado que las clasificaciones que se han hecho hasta la fecha carecen de justificación matemática. No obstante, en (2) se justifica esta clasificación según lo siguiente: un robot redundante sólo será capaz de rodear y orientar un objeto si se cumple que $2N \leq GdL$.
- ***Según el número de articulaciones:*** pese a estar íntimamente relacionados, el concepto de *"articulación"* difiere del de *"grados de libertad"*. Una articulación es un mecanismo que permite una serie de movimientos, siendo posible que exista una articulación de varios grados de libertad. De acuerdo con este criterio, se diferencian dos tipos de robots:
  - *Discretos:* eslabones rígidos distinguibles entre sí.
  - *Continuos:* estructuras deformables de manera continua.

  Existe un grupo que, aun formando parte de los robots discretos, puede considerarse muy cercano a los continuos. Es el caso de los robots *"serpentinos"* , formados por un gran número de eslabones lo suficientemente pequeños como para poder considerarse una cadena continua.

- ***Según su actuación:*** atendiendo al tipo de actuadores se puede distinguir entre:
    - *Motores.*
    - *Hidráulicos.*
    - *Aleaciones con Memoria de Forma (SMA).*
    - *Piezoeléctricos.*
    - *Músculos Artificiales.*
    - *Neumáticos.*

- ***Según sus materiales:*** los materiales pueden ser de dos tipos: *rígidos* o *blandos.* Los materiales rígidos – *metales o plásticos*- confieren al robot de una resistencia mecánica y robustez altas. No obstante, materiales blandos como las siliconas o los cauchos permiten fabricar robots muy adaptables al entorno y muy seguros a la hora de interactuar con personas, seres vivos u objetos delicados.

2.2 Antecedentes

La aplicación más típica de este tipo de robots es la de **inspección** de lugares angostos y difícilmente accesibles. Es común equiparlos con cámaras y luces para facilitar la visión en lugares angostos. Otra posible aplicación tiene que ver con el mundo de la medicina, en campos como la **cirugía**, **endoscopias** y demás tareas que requieran de **tratamientos invasivos**. También se estudia en la actualidad su uso en **labores de rescate**, por ejemplo, en la búsqueda de personas sepultadas o atrapadas en terrenos angostos.

Son varios los estudios que se han llevado a cabo sobre robots hiperredundantes, favoreciendo así a la aparición de varios modelos comerciales en las diversas áreas de aplicación. Cabe destacar el caso del "Series II X125" de OC Robotics, el cual utiliza cables como actuadores de las diferentes secciones del robot. Su diseño está enfocado principalmente a inspección y limpieza. Una de las ventajas de este robot reside en la disposición de sus actuadores en la base del robot, consiguiendo un cuerpo muy ligero. La consecuencia de esto es que es posible construir un robot de gran longitud, aumentando con creces su espacio de trabajo.

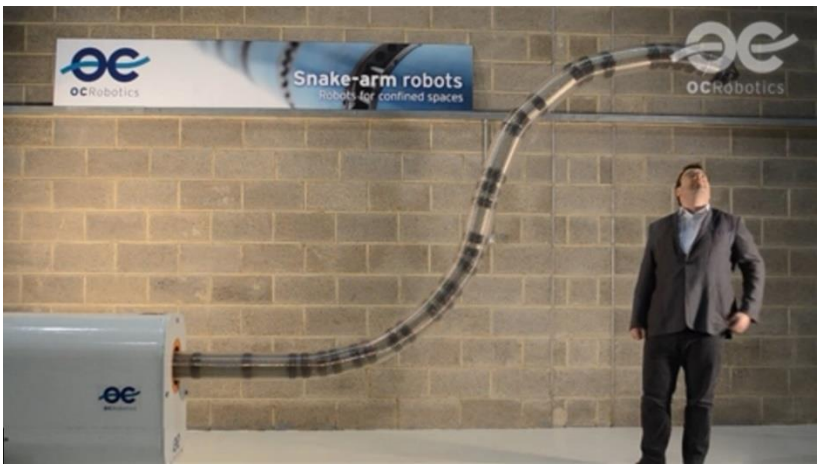


Figura 6: modelo Series II X125 en su máxima elongación.
Fuente: https://www.wevolver.com/wevolver.staff/series.ii.x125.system

Otra de las ventajas mencionadas tiene que ver con su robustez: al disponer de un gran número de actuadores, articulaciones y segmentos, el fallo de uno de los módulos del robot no tiene una influencia catastrófica en el desempeño global de este. Por ejemplo, en un robot con sólo tres actuadores, si uno de ellos fallase el impacto en el desempeño global será mayor que en un robot de veinte actuadores, ya que es posible adaptar muy fácilmente su pose para minimizar el impacto del fallo. Esto hace que los robots hiperredundantes sean especialmente interesantes en la **industria espacial**. La reparación de un robot en, por ejemplo, la Estación Espacial Internacional, es algo realmente costoso económicamente, por lo que un robot que no necesita reparaciones habitualmente resulta muy atractivo.

Figura 7: robot "Tendril". Fuente: (14)

Por este motivo, corporaciones como la NASA han llevado a cabo investigaciones con estos robots. Se trata del robot “Tendril”, cuyo cuerpo consiste en un muelle, accionado también mediante cables. Es un robot puramente continuo, dado que no está compuesto por componentes discretos. Su diseño está enfocado principlamente en la manipulación de pequeños objetos. Dado su pequeño tamaño, sería un robot especialmente útil en un laboratorio automatizado o controlado de forma remota, pues podría trabajar fácilmente con objetos de pequeño tamaño.

La Universidad Politécnica de Madrid también está contribuyendo en la actualidad a la investigación acerca de este tipo de robots. En concreto, el CAR ha llevado a cabo algunos prototipos de robots hiperredundantes. En el año 2017, Cecilia Martinez Martín desarrolló el llamado “Kyma” (5), de tipo blando y también actuado mediante cables. Su diseño estaba orientado a trabajar como **robot colaborativo**, dada su naturaleza blanda.

Figura 8: robot Kyma. Fuente: (5)

En la misma línea de investigación, Iván Rodríguez desarrolló con el Grupo de Robótica y Cibernética (UPM) el robot “Mach I” . Está inspirado en el miembro más ágil de los elefantes: su trompa.

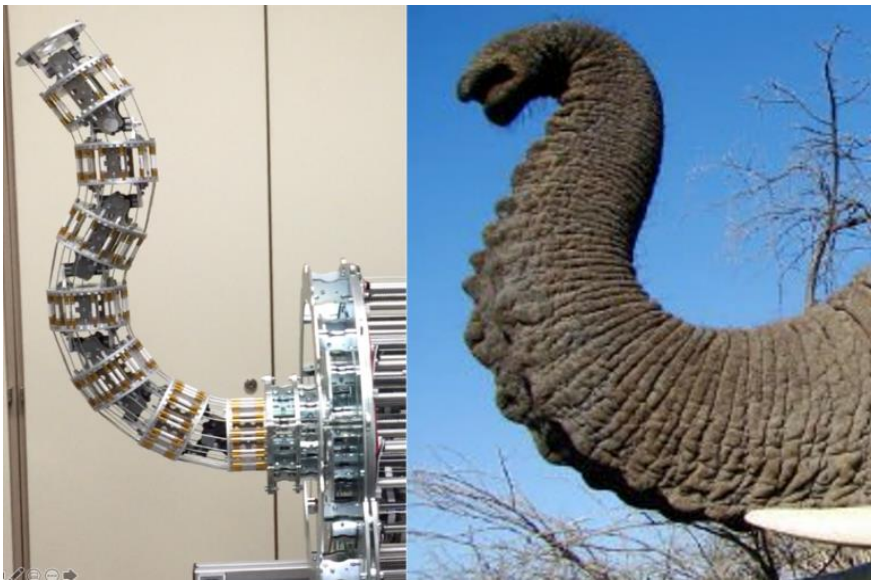


Figura 9: robot Mach I y trompa de elefante.
Fuente: *“El Blog de la Robótica en la UPM”*
https://blogs.upm.es/robcib/2020/09/04/mach-i-un-robot-como-una-trompa-de-elefante

El Mach I fue provisto de dispositivos ópticos – *cámaras de vídeo, visión infrarroja, iluminación*- para llevar a cabo inspecciones de lugares intrincados. Además, para controlar el robot, se desarrolló un sistema de realidad virtual que permitía al operario ver en primera persona las imágenes captadas por la cámara del robot.

Figura 10: robot Mach I realizando una inspección.
Fuente: *“El Blog de la Robótica en la UPM”*
https://blogs.upm.es/robcib/2020/09/04/mach-i-un-robot-como-una-trompa-de-elefante

Inspirado en el Mach I, nace el robot de tipo CDHRR de Discos Pivotantes *"Pylori I"*, diseñado también por Iván Rodríguez y construido por Elena Muñoz. Consiste en un robot compuesto por discos construidos mediante impresión 3D. Cada uno de los discos pivota con el anterior, realizando un movimiento similar al de una columna vertebral. También está accionado por cables, y su tamaño es significativamente menor que sus dos predecesores.

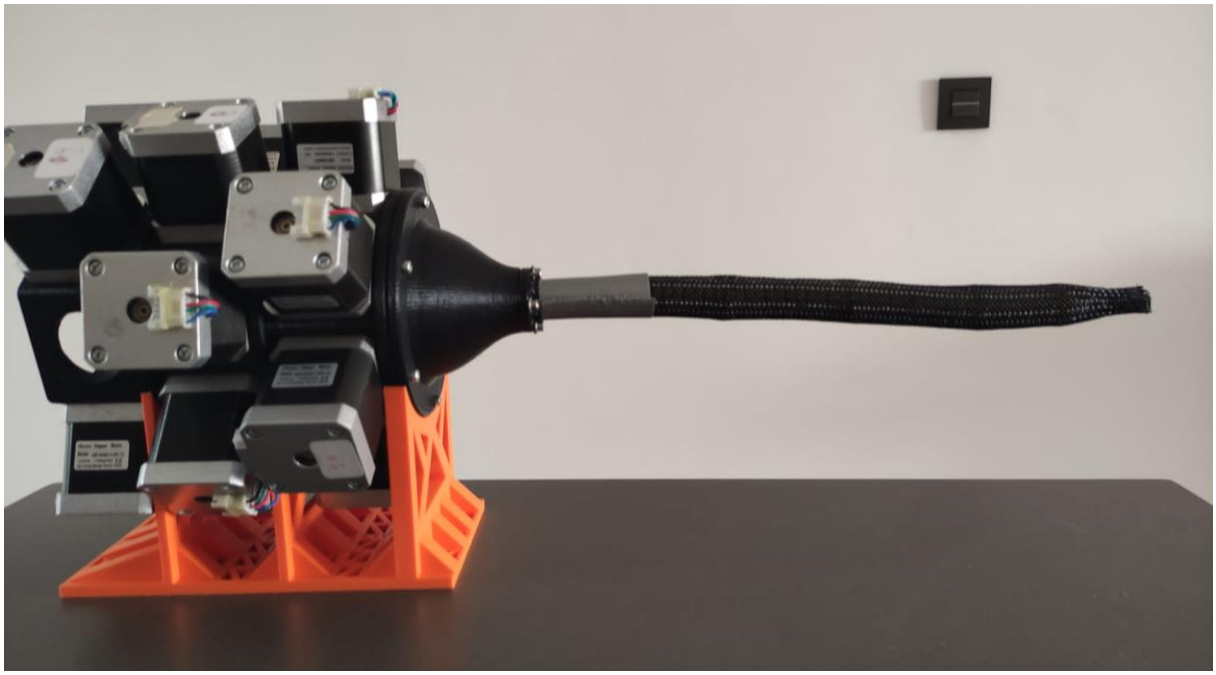


Figura 11: imagen del Pylori I. Cortesía de Elena Muñoz.

El Pylori I ha introducido una serie de cuestiones sobre la optimización de su diseño, reiterando la necesidad ya existente de buscar métodos alternativos para la optimización del diseño de estos robots. Se pretende arrojar luz sobre estos aspectos, estudiando las diferentes configuraciones posibles a la hora de diseñar un robot de este tipo mediante el uso de inteligencia artificial, en concreto, de algoritmos genéticos. Para abordar esta tarea, primero es necesario hacer una descripción detallada de la construcción y funcionamiento de estos robots, para poder así comprender en profundidad la problemática que se plantea.

2.3 Cable Driven H. Redundant Manipulators de Discos Pivotantes

Los robots de tipo CDHRR en los que se enfoca este trabajo están compuestos por una **cadena de discos que pivotan entre sí** en torno a un eje. La cadena de discos se divide en grupos denominados *"secciones"*. Estos grupos de discos son actuados por cables, que van anclados al último disco de cada sección, distribuyendo la tensión por todos los discos de esta. El cuerpo del robot se construye concatenando secciones entre sí.

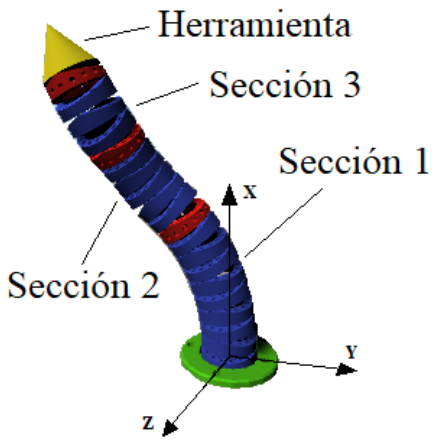


Figura 12: Esquema de Secciones. Fuente: elaboración propia.

Atendiendo a la clasificación antes expuesta, son robots **Manipuladores, Hiperredundantes, Serpentinos, Rígidos, actuados mediante Cables**. Cabe mencionar que los cables que distribuyen la fuerza a lo largo de las secciones están anclados a una serie de motores a través de poleas, por lo que realmente el modo de actuación es mediante **motores eléctricos**. Dado que se trata de robots manipuladores, disponen de una herramienta en su extremo que les permite llevar a cabo las labores necesarias.

En la siguiente figura, se muestra con detalle cómo los cabes actuadores atraviesan los discos, distribuyendo la tensión por todos ellos. Poniendo tensión en unos cables u otros, la sección se inclinará formando un determinado ángulo. Debido a su construcción, **cada una de las secciones dispone de dos grados de libertad**. Esto implica que el movimiento de una única sección tiene forma de superficie, mientras que concatenando dos o más secciones será posible describir un volumen.

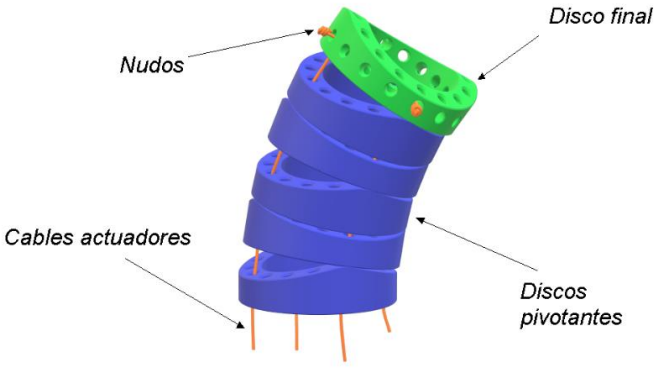


Figura 13: Detalle de cables actuadores. Fuente : elaboración propia.

Debido al gran número de discos del que disponen, se puede considerar estos robots como de tipo continuo. Son uno de los posibles ejemplos de robots serpentinos: tienen un gran número de eslabones rígidos, pero al ser estos muy pequeños en comparación con la longitud total, pueden considerarse como una cadena continua de material blando. Este aspecto **permite que puedan ser estudiados como robos continuos o discretos** en función de las necesidades de la tarea que se esté llevando a cabo. Por ejemplo, a la hora de simular el cuerpo del robot de forma aproximada, quizá no sea interesante considerar todos y cada uno de los eslabones, sino que es más eficiente considerarlo como un sistema continuo. Por el contrario, de ser necesario un estudio de, por ejemplo, la posición de un disco en concreto, es posible hacerlo fácilmente considerando el robot como un sistema discreto.

Como se ha mencionado en el caso del Series II X125, una de las claras ventajas de los robots actuados mediante cables tiene que ver con la localización de sus actuadores. Estos suelen situarse en la base del robot, de manera que el cuerpo de este no soporta el peso de dichos actuadores. El resultado es un cuerpo más ligero que permite manipular objetos más pesados. Esto no ocurre en otros manipuladores, por ejemplo, brazos robóticos en los que los actuadores están situados en las propias articulaciones del robot. Las implicaciones de esto tienen también que ver con cuestiones de seguridad. Al tratarse de robots con un cuerpo más liviano, cualquier tipo de colisión con el entorno o con una persona resultaría menos violenta.

La problemática del diseño de estos robots tiene que ver con tres aspectos clave. El primero es el ***número total de discos*** del robot. Lógicamente, la longitud es un aspecto determinante el desempeño del robot, e influye de forma tanto positiva como negativa: un robot más largo permitirá llegar a puntos más lejanos y tendrá un espacio de trabajo mayor. Por el contrario, requerirá de actuadores más grandes que soporten su peso y que permitan mover su cuerpo ágilmente. También será un aspecto negativo al moverse en espacios pequeños, donde maniobrar resulta más complicado de tratarse de un robot demasiado grande.

El segundo aspecto es el ***número de discos de los que se compone cada sección***. Este es, sin duda, uno de los aspectos que más condicionan la movilidad del robot. Resulta complicado justificar por qué se elige una distribución de discos u otra. A priori, resulta intuitivo pensar que las secciones deberán ir disminuyendo en longitud conforme se aproximan al extremo del robot por motivos de distribución de peso y precisión en la movilidad. No obstante, obtener una distribución óptima de forma analítica no es tarea sencilla.

Por último, el tercer aspecto de diseño a considerar es el ***ángulo de montaje entre discos***. Recordar que cada disco pivota con el anterior en torno a un eje. Ese eje puede establecerse en la dirección que se desee, siendo prácticamente infinitas las combinaciones que pueden hacerse en cada disco. Sin embargo, debería haber combinaciones que optimicen aspectos como la movilidad, agilidad, volumen de trabajo… Esta es la labor que se lleva a cabo en el presente trabajo, pretendiendo obtener una serie de soluciones óptimas que permitan optimizar el diseño y montaje del actual robot CDHRR de Discos Pivotantes.

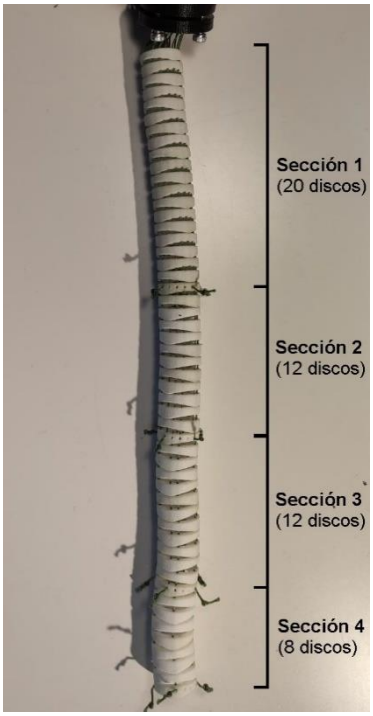


Figura 14: detalle del cuerpo del Pylori I.
Cortesía de Elena Muñoz.

2.4 Algoritmos Evolutivos

La inteligencia artificial es un campo en pleno apogeo en los tiempos que corren. Tanto es así, que esta disciplina es considerada como la punta de lanza de la Cuarta Revolución Industrial o *"Industria 4.0"*. Automatización de procesos, técnicas de big-data, uso de robótica industrial, IIOT -*Industrial Internet of Things*- o Inteligencia Artificial son algunas de las principales características del desarrollo tecnológico actual. En este contexto, el uso de algoritmos de Inteligencia Artificial está encontrando aplicación en una gran variedad de campos en los que se requiere de soluciones sencillas a problemas complejos, de naturaleza abstracta y difícilmente abordables desde un punto de vista puramente teórico.

Los primeros algoritmos Evolutivos se desarrollaron en los años 60 de la mano del profesor John H. Holland, en la Universidad de Michigan, tratándose de versiones muy primitivas de los algoritmos actuales. Durante esta década se fueron sentando las bases de lo que más adelante serían los Algoritmos Evolutivos, gracias al excelente trabajo de Holland y sus alumnos, a los que también pedía participación. Los primeros trabajos sobre esta materia propiciaron el ecosistema adecuado para que a finales de los 70 David Goldberg - *alumno de Holland*- desarrollara el primer algoritmo genético propiamente dicho.

David Goldberg tenía formación de ingeniero industrial, y estaba muy interesado en aplicar estos algoritmos a la optimización de procesos industriales y resolución de problemas propios de esta área. Resulta inspirador cómo ya en los 70 Goldberg se adelantó a su época, adentrándose en un campo que 50 años después ha desencadenado una auténtica revolución tecnológica. Su intención era la aplicación de estos algoritmos para optimizar la distribución del mapa de tuberías y cableado de una planta industrial -*también conocido como "pipeline"*-.

El origen de los algoritmos genéticos fue promovido por la necesidad de resolver problemas complejos de la industria, partiendo del completo desconocimiento de la solución adecuada. Conforme la tecnología ha ido avanzando, el rango de aplicación de estos algoritmos se ha visto ampliado con creces. Disciplinas como el procesamiento de imágenes o el reconocimiento de voz hacen un uso intensivo de estas herramientas, dado el nivel de abstracción que requieren. Otro campo quizá más cercano al mundo científico es la predicción y simulación de sistemas astronómicos. Comportamiento de nebulosas, estimaciones del número de cuerpos que componen un sistema astronómico o modelado del comportamiento de agujeros negros son algunas de las áreas de aplicación de los algoritmos de inteligencia artificial en general.

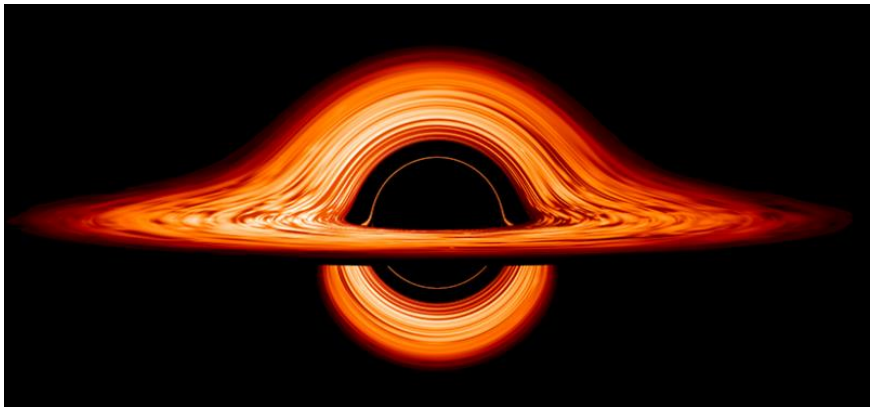


Figura 15: Simulación del disco de acreción de un agujero negro, realizada por la NASA mediante técnicas de IA y procesamiento de imagen.
Fuente: NASA GSFC/J. Schnittman

Como se ha mencionado, todas estas labores de cálculo tienen como punto en común el desconocimiento de una expresión analítica a resolver, o bien que, pese a conocerla, la dificultad de obtener una solución es muy elevada. En esta misma línea se encuentra el presente trabajo, que pretende abordar una labor compleja de optimización utilizando como herramienta un algoritmo genético.

En el mundo de la robótica, lo algoritmos genéticos llevan varios años teniendo un lugar muy relevante. Uno de los primeros artículos publicados sobre esta materia fue (6), en el que sus autores se enfrentaban a la labor de controlar un robot hexápodo de 12 grados de libertad. Su trabajo consistió en la construcción de una red neuronal encargada de calcular la cinemática del robot. Esta construcción es, al fin y al cabo, una optimización de los parámetros de la red neuronal, y fue llevada a cabo haciendo uso de un algoritmo genético. El algoritmo genético propone una *"población"* de redes neuronales con configuraciones aleatorias. Tras evaluar el desempeño de cada una, selecciona las mejores, combina sus configuraciones y genera una nueva población, la cual hereda los *"genes"* de sus antecesores mejor adaptados. Iteración tras iteración, el algoritmo consigue optimizar la configuración de la red neuronal.

Son todos estos antecedentes los que motivan este trabajo. Aplicar los algoritmos genéticos a un tipo de robots cuyo diseño y perfeccionamiento son todavía áreas de intenso estudio, supone sin duda una tarea apasionante. Para enfrentarse a ello, el aprendizaje previo ha sido intenso, como no podía ser de otra manera, y ha requerido de conocimientos nuevos y entusiasmantes.

3. FUNDAMENTOS

En este apartado se expondrán las bases teóricas que se han utilizado durante el trabajo. Pretende ser un apartado explicativo, dotado de ejemplos básicos y sencillos que faciliten la comprensión de los diferentes conceptos explicados. Se partirá, en el punto 3.1, de conocimientos básicos, conocidos por todo ingeniero, pero que servirán de justificación y de base teórica cuando se pase a comentar aspectos más concretos sobre el código de Matlab y de los algoritmos implementados.

En el apartado (3.2) se explicará meticulosamente los algoritmos que se han utilizado, también con ejemplos básicos, aplicados a algunas funciones típicas. De estos ejemplos, aunque puedan resultar simples en un primer momento, se podrán extraer conclusiones y similitudes que ayudarán en los siguientes apartados.

3.1 Conceptos Matemáticos

3.1.1 Sucesiones de Números: convergencia.

Una sucesión $(\mathrm{a_n})_{\mathrm{n}\in\mathbb{N}}$ es una aplicación de los números naturales $\mathbb{N}$ en los reales $\mathbb{R}$ tal que:

$$\mathrm{n} \in \mathbb{N} \rightarrow \mathrm{a(n)} = \mathrm{a_n} \in \mathbb{R} \qquad (3.1)$$

Es decir, se trata de un conjunto infinito de números ordenados según una aplicación de $\mathbb{N}$. A esta aplicación se le denomina *término general*, y es la que determina el valor de un elemento de la sucesión en función del valor de su índice (n). Así, un ejemplo de sucesión podría ser $a_n = \frac{1}{n}$ „ $n = 1, 2, 3$ ...Sus valores son $\{\mathrm{a_n}\} = \{1/1, 1/2, 1/3, 1/4 \ldots\}$.

En función del rango de valores que puede tomar una sucesión, se hablará de:

- *Sucesión acotada inferiormente:* $\exists n_0 \in \mathbb{R}$ „ $\mathrm{a_n} \geq n_0 \quad \forall \mathrm{n} \in \mathbb{N}$
- *Sucesión acotada superiormente:* $\exists n_0 \in \mathbb{R}$ „ $\mathrm{a_n} \leq n_0 \quad \forall \mathrm{n} \in \mathbb{N}$
- *Sucesión acotada:* $\exists n_0 \in \mathbb{R}$ „ $|\mathrm{a_n}| \leq n_0 \quad \forall \mathrm{n} \in \mathbb{N}$

Cuando la sucesión tiende a un valor determinado, se habla de *límite*. Este límite se corresponde con el valor al que se aproxima la sucesión conforme aumenta el valor de n. Según (7), El concepto de límite se puede definir formalmente como sigue:

> *"Una sucesión* $(\mathrm{a_n})_{\mathrm{n}\in\mathbb{N}}$ *es convergente con límite* $l \in \mathbb{R}$ *cuando, dado* $\varepsilon > 0$ *existe* $n_0 \in \mathbb{N}$ *tal que* $\forall n \geq n_0$ *se cumple que* $|a_n - l| < \varepsilon$*."* (3.2)

En la siguiente imagen queda ilustrado el ejemplo anterior. Se trata de una sucesión acotada inferiormente, cuyo límite es cero. Existe un valor n_0 que cumple que, para un épsilon dado, los valores posteriores de la sucesión siempre son menores que dicho épsilon.

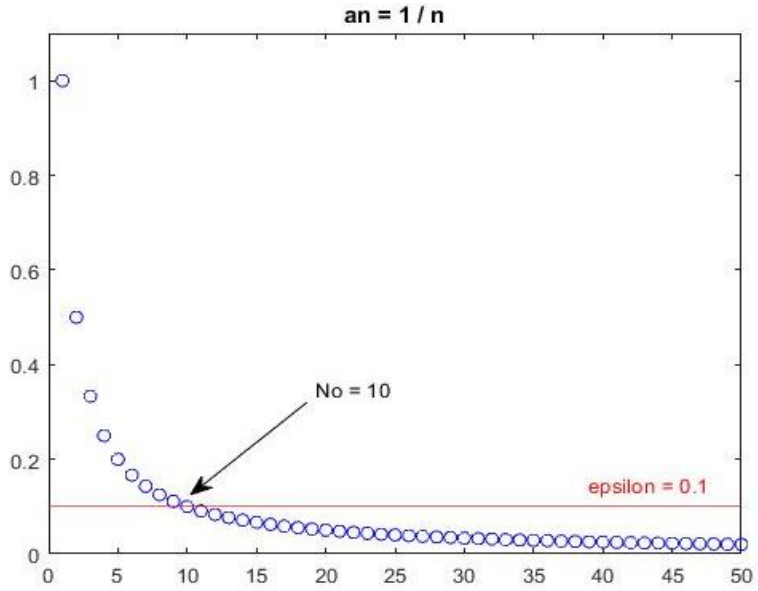


Figura 16: Sucesión acotada inferiormente.
Fuente: *elaboración propia*.

Como se expondrá más adelante, estos conceptos se han utilizado como **criterio de terminación para algoritmos iterativos**. Por ejemplo, sea un algoritmo encargado de posicionar el extremo del robot en un punto determinado. Este algoritmo irá calculando poses del robot iterativamente, acercándose poco a poco al punto deseado.

El objetivo es que el error de posición se vaya acercando progresivamente a cero. Sin embargo, es posible que el algoritmo no esté convergiendo hacia una solución correcta, y que se haya vuelto inestable. Parámetros mal configurados, paso por puntos singulares o falta de precisión numérica en los cálculos pueden ser posibles causas.

Una forma simple de comprobar esto es establecer que **a partir de la iteración n_0 se compruebe si el error de posición es menor que un valor épsilon**. De ser así, se asume que el algoritmo está convergiendo y que tarde o temprano alcanzará una solución correcta. De lo contrario, se detendrá el algoritmo, pues o bien se ha vuelto inestable o bien no está consiguiendo llegar a una solución adecuada.

3.1.2 Cálculo Numérico de Derivadas

A lo largo del trabajo ha sido necesario calcular derivadas por computador en varias ocasiones. Esta tarea puede abordarse aplicando métodos de derivación discreta, que arrojan soluciones aproximadas al cálculo de la derivada de funciones. En primer lugar, es necesario introducir el concepto de derivada en un punto. Según (7):

> *"Se dice que una función f definida en un entorno de a, f : (a-δ, a+δ) → ℝ , es derivable en a, si su cociente incremental tiene límite finito cuando x → a. En ese caso su derivada $f'(a)$ en el punto a se define como:*
>
> $$f'(a) = \lim_{x \to a} \frac{f(x)-f(a)}{x-a} \quad (3.3)$$
>
> *En muchas ocasiones se usa para definir $f'(a)$ la variable h = x – a en lugar de la propia variable x. Entonces se escribe:"*
>
> $$\boxed{f'(a) = \lim_{h \to 0} \frac{f(a+h)-f(a)}{h}} \quad (3.4)$$

La expresión (3.4) permite la obtención numérica de derivadas de manera muy sencilla. Los métodos utilizados se denominan *"Métodos de Diferencias Finitas"*. Consisten en aproximar la derivada de una función en un punto calculando la pendiente que se forma con otros puntos cercanos. De entre las muchas formas que hay de hacer esto, las más simples y las que se han utilizado en el presente trabajo son:

- *Diferencias hacia Adelante:* $f'(x) \approx \frac{f(a+h)-f(a)}{h}$ (3.5)
- *Diferencias hacia Atrás:* $f'(x) \approx \frac{f(a-h)-f(a)}{h}$ (3.6)
- *Diferencias Centradas:* $f'(x) \approx \frac{f(a+h)-f(a-h)}{2h}$ (3.7)

Conocida la función f(x), resulta muy sencillo aplicar las fórmulas anteriores. Dando un valor pequeño a h – *por ejemplo 0.001* – y evaluando la función en los dos puntos necesarios, puede obtenerse un valor aproximado para la derivada en el punto de manera muy rápida. A modo de ejemplo, se procede a calcular la derivada de la función f(x) = sen(x) en el punto x = 0.453 aplicando la ecuación 3.5:

$$f(0{,}453 + 0{,}001) = sen(0{,}454) = 0{,}43856$$

$$f(0{,}453) = sen(0{,}453) = 0{,}43766$$

$$f'(0{,}453) \approx \frac{0{,}43856 - 0{,}43766}{0{,}001} = \mathbf{0,9}$$

Para comprobarlo, es sabido que la derivada de la función seno es $f'(x) = \cos(x)$. Evaluando el coseno en el punto 0.453 se obtiene que cos(0.453) = **0.89914** , valor muy cercano al obtenido con la aproximación.

3.1.3 Matriz Jacobiana

La matriz Jacobiana es una matriz que se construye con las derivadas parciales de una aplicación **F**(**x**) = (f_1(**x**), f_2(**x**), …, f_n(**x**)). A continuación, se introduce su expresión general. Notar que el operador "D" se trata del operador derivada. Siendo **x**=(x_1, x_2, …, x_n) el vector con las variables del problema, el operador D_i será la derivada respecto a x_i, resultando:

Sea $\boldsymbol{F}: A \subset \mathbb{R}^n \to \mathbb{R}^m$ *una aplicación diferenciable en un punto* $a \in A$ *dada por* ***F(x)*** *= (f1(****x****); f2(****x****); …, fm(****x****)). Se llama Matriz Jacobiana a la matriz asociada a la aplicación lineal* $D\boldsymbol{F}(a)$*, y es de la forma:*

$$\boldsymbol{J} = \boldsymbol{F}'(\boldsymbol{a}) = \begin{bmatrix} D_1 f_1(\boldsymbol{a}) & D_2 f_1(\boldsymbol{a}) \dots & D_n f_1(\boldsymbol{a}) \\ D_1 f_2(\boldsymbol{a}) & D_2 f_2(\boldsymbol{a}) \dots & D_n f_2(\boldsymbol{a}) \\ D_1 f_m(\boldsymbol{a}) & D_2 f_m(\boldsymbol{a}) \dots & D_n f_m(\boldsymbol{a}) \end{bmatrix} \qquad (3.8)$$

Observar que, por filas, la matriz Jacobiana se construye con los gradientes de cada una de las funciones que componen la aplicación **F**(**x**), particularizados para el punto a:

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{\nabla} f_1(\boldsymbol{a}) \\ \boldsymbol{\nabla} f_2(\boldsymbol{a}) \\ \boldsymbol{\nabla} f_m(\boldsymbol{a}) \end{bmatrix} \qquad (3.9)$$

Estas matrices son especialmente útiles para la mecánica clásica en el cálculo de velocidades en el espacio. Sea $\boldsymbol{F}: V \subset \mathbb{R}^n \to \mathbb{R}^3$ la aplicación lineal que describe el movimiento de un punto en un espacio tridimensional, donde V es el volumen en el que puede moverse dicho punto. Las columnas de la matriz jacobiana asociada a esta aplicación lineal ($\boldsymbol{J_F}$) representan la evolución de cada una de las componentes de **F** con respecto a las variables del problema.

Sea $F: V \subset \mathbb{R}^n \to \mathbb{R}^m$ *una aplicación diferenciable en un punto* $a \in V$ *dada por* ***F*** *(x, y, z, φ, θ, ψ), donde los tres primeros términos representan la posición de un sistema de referencia móvil S en el espacio V, y los tres últimos su orientación. La Matriz Jacobiana* ***J*** *asociada a esta aplicación se puede dividir en dos matrices jacobianas, una asociada a la traslación y otra a la rotación de este sistema* (4)*:*

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J}_v \\ - \\ \boldsymbol{J}_w \end{bmatrix} \to \begin{bmatrix} \boldsymbol{v} \\ - \\ \boldsymbol{w} \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_v \\ - \\ \boldsymbol{J}_w \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ \varphi \\ \theta \\ \psi \end{bmatrix} \qquad (3.10)$$

EJEMPLO: *Sea un punto que se mueve en un espacio de tres dimensiones a lo largo de una trayectoria definida por la aplicación F(x, y, z) = (cos(z), sen(z), z). Se pretende dibujar la dirección de avance por la curva en los puntos P y Q, situados en z = 10 y z = 25 respectivamente.*

En primer lugar, se calcula la Matriz Jacobiana asociada aplicando la ecuación (3.8). Dado que se trata de un punto, hablar de orientación no tiene sentido alguno, por lo que se calculará la jacobiana asociada a traslación:

$$F(x,y,z) = (cos(z), sen(z), z) \rightarrow$$

$$\rightarrow f_1(x,y,z) = cos(z), f_2(x,y,z) = sen(z), f_3(x,y,z) = z$$

$$\boldsymbol{J} = \boldsymbol{F}'(x,y,z) = \begin{bmatrix} D_1f_1 & D_2f_1 & D_3f_1 \\ D_1f_2 & D_2f_2 & D_3f_2 \\ D_1f_3 & D_2f_3 & D_3f_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -sen(z) \\ 0 & 0 & \cos(z) \\ 0 & 0 & 1 \end{bmatrix}$$

Las direcciones en los puntos P y Q se obtienen a partir de la ecuación 3.10 particularizando los respectivos valores de z:

$$\boldsymbol{d}_P = \boldsymbol{J}_{z=10} \cdot \begin{bmatrix} f_1(10) \\ f_2(10) \\ f_3(10) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0.544 \\ 0 & 0 & -0.839 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -0.839 \\ -0.544 \\ 10 \end{bmatrix} = \begin{bmatrix} 5.44 \\ -8.39 \\ 10 \end{bmatrix}$$

$$\boldsymbol{d}_Q = \boldsymbol{J}_{z=25} \cdot \begin{bmatrix} f_1(25) \\ f_2(25) \\ f_3(25) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0.132 \\ 0 & 0 & 0.991 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.991 \\ -0.132 \\ 25 \end{bmatrix} = \begin{bmatrix} 3.3 \\ 24.775 \\ 25 \end{bmatrix}$$

Tras hacer los vectores unitarios, es posible conocer la dirección en la que avanzará el punto al moverse por la trayectoria. En la siguiente imagen se ilustra el ejemplo tratado.

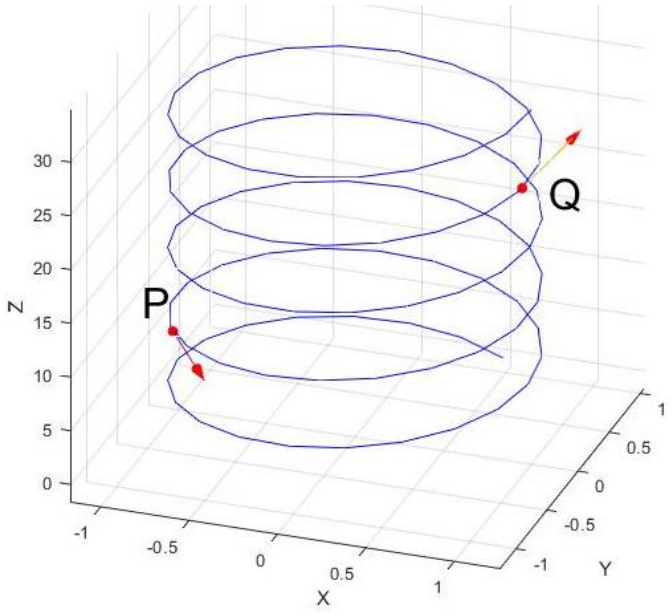


Figura 17: Aplicación de la Matriz Jacobiana para el cálculo de direcciones.
Fuente*: elaboración propia*.

Las matrices Jacobianas son ampliamente utilizadas en el mundo de la robótica, siendo una herramienta básica y fundamental para el cálculo de los modelos tanto cinemático como dinámico del robot. En el presente trabajo, se ha hecho uso de estas matrices para el posicionamiento del manipulador. Obteniendo la matriz Jacobiana asociada a la posición del extremo del robot, es posible posicionarlo de forma análoga al ejemplo anterior.

La complejidad de la cinemática de los robots CDHRR impide disponer de una expresión analítica de la posición del efector final de forma sencilla. Esto complica en gran medida la obtención de una Jacobiana y, por ende, el control del manipulador. Existen algoritmos que permiten la obtención de una Jacobiana Geométrica a partir de las matrices de Denavit-Hartenberg de la cadena cinemática (8). Sin embargo, dada la naturaleza de estos robots, **no resulta trivial el uso de matrices de transformación para los robots de este tipo**. Recordar que Denavit-Hartemberg asume articulaciones prismáticas y/o rotativas, cuando en el caso de robots CDHRR **las articulaciones describen un movimiento de varios grados de libertad**.

Algunos autores proponen un modelo para este tipo de robots, basado en cuatro articulaciones rotativas perpendiculares entre sí más una articulación prismática (1).

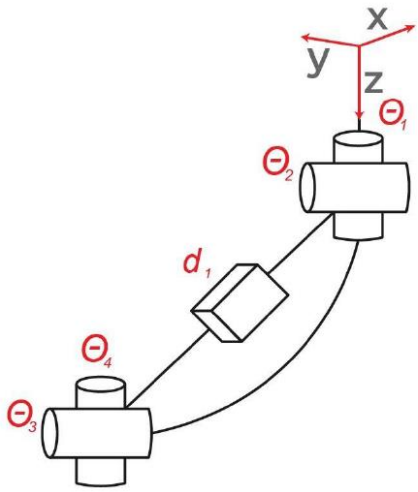


Figura 18: Modelo de Webster y Jones para robots continuos.
Fuente: (9).

Sin embargo, en este trabajo se necesita simular la pose exacta de todos y cada uno de los discos, siendo insuficiente la obtención de la matriz de transformación de la sección. Por ello, se ha desarrollado un **método numérico para calcular la cinemática de cada una de las secciones del robot que contempla la posición de cada disco**. Este método, descrito en el apartado 6.1 permite obtener de forma numérica la matriz de transformación hasta el extremo de una sección, solventando así el problema que se presentaba al no conocer dicha matriz.

Por otro lado, se ha utilizado el concepto de **derivadas discretas para calcular las componentes de la matriz jacobiana** según la ecuación 3.8. Si bien es cierto que no se conoce la aplicación lineal **F** (**q**) - *siendo q el vector de variables de control del robot* - , sí que es posible obtener esta posición partiendo de la matriz de transformación obtenida con el método del apartado 6.1. Por lo tanto, **será posible evaluar las componentes de la Jacobiana introduciendo pequeños incrementos en las variables de control del robot**, obteniendo así una solución análoga a la obtenida en el ejemplo del apartado 3.1.2 con la función seno. Todo esto se desarrollará con detalle en los próximos apartados.

3.2 Algoritmos

3.2.1 Descenso del Gradiente

El algoritmo del Descenso del Gradiente es un algoritmo iterativo que permite optimizar una función de coste dada. La elección de la función de coste es un aspecto crítico, pues de ello dependerá que la solución obtenida por el algoritmo realmente sea la buscada. Por ejemplo, si el objetivo es posicionar un determinado objeto en el espacio, la función de coste a optimizar será el error de posición del objeto con respecto a la posición requerida. Optimizar la posición de este objeto consistirá en **minimizar su error de posición** con respecto al punto objetivo. En consonancia con lo anterior, optimizar la pose de un robot manipulador para conseguir posicionar su herramienta, consistirá en minimizar el error de posición de su extremo eligiendo la configuración adecuada en las articulaciones del robot.

Como es sabido, el mínimo local de una función se encuentra situado en un valle. En este punto se cumple que la derivada de la función se anula. El Descenso del Gradiente aprovecha este concepto para aproximarse al mínimo local por la dirección de descenso de la derivada de la función. Cuando el gradiente sea cero, se habrá llegado a un mínimo local. Hay que tener en cuenta que en una función puede haber varios mínimos locales, por lo que la elección del punto de partida condicionará la solución obtenida.

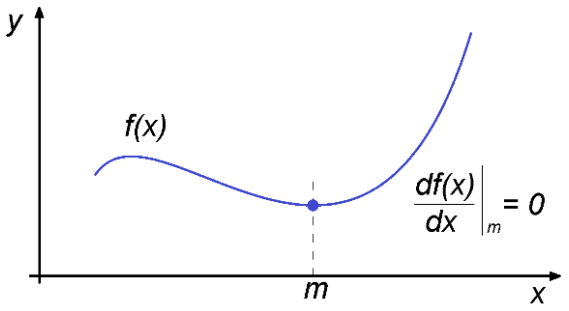


Figura 19: Mínimo Local de una función.
Fuente: *elaboración propia.*

El procedimiento para alcanzar el mínimo local consiste en lo siguiente:

1. **Establecer un punto de partida.** *La elección de este punto es arbitraria, pero condiciona la solución obtenida, pudiendo variar el mínimo local hallado. Es conveniente partir de soluciones iniciales cercanas a la real para asegurar llegar al mínimo local adecuado.*
2. **Calcular la derivada en el punto.**
3. **Corregir el punto actual**, *restándole la derivada calculada en el punto anterior ponderada por una constante de corrección.*
4. **Iterar desde el paso 2** *hasta que la derivada se haga nula o cercana a cero .*

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - k \cdot \boldsymbol{\nabla} f(\boldsymbol{x}) \qquad (3.11)$$

Tener en cuenta que esto no solo es aplicable a funciones de una variable, sino que también es aplicable a funciones de varias variables, obteniendo vectores como solución. A continuación, se expone un ejemplo ilustrativo.

EJEMPLO: *Sea la función de coste* $F(x,y) = \sqrt{(x-0{,}5)^2 + (y-0{,}75)^2} = z$ *cuya forma es la de un paraboloide. Se va a buscar su vértice -mínimo local- mediante el algoritmo del descenso del gradiente, partiendo del punto x = 1,2 y = 0,1. De la fórmula se puede deducir fácilmente que la solución será el punto x = 0,5 y = 0,75. Se ha planteado un problema sencillo para que resulte más ilustrativo.*

En primer lugar, se va a simplificar la función eliminando la raíz, pues el punto mínimo es el mismo en ambas funciones. Esta operación es típica al trabajar con funciones con radicales o potencias. Un ejemplo clásico es el Error Cuadrático, del cual puede eliminarse su radical para simplificar el cálculo. Así, se pasará a minimizar la función $G(x,y) = (x-0{,}5)^2 + (y-0{,}75)^2$. *Seguidamente, se calculará el gradiente de G(x, y). Su expresión es* $\nabla G(x,y) = (\,2x-1\;,\;\; 2y-1.5)$.

A continuación, se plantea una tabla en la que se ha implementado el algoritmo descrito en la ecuación 3.11. El valor de k utilizado ha sido k = 0.3. La determinación de esta constante afecta enormemente a la velocidad de convergencia del algoritmo, así como a su precisión.

ITERACIÓN	X	Y	$D_1G(x, y)$	$D_2G(x, y)$	| DG(x, y) |
1	1,2	0,1	1,4	-1,3	1,910497317
2	0,78	0,49	0,56	-0,52	0,764198927
3	0,612	0,646	0,224	-0,208	0,305679571
4	0,5448	0,7084	0,0896	-0,0832	0,122271828
5	0,51792	0,73336	0,03584	-0,03328	0,048908731
6	0,507168	0,743344	0,014336	-0,013312	0,019563493
7	0,5028672	0,7473376	0,0057344	-0,0053248	0,007825397

Se puede ver fácilmente cómo el módulo del gradiente se aproxima a cero conforme los valores de x e y se acercan a 0.5 y 0.75 respectivamente. Como se mencionó en el apartado anterior, se han utilizado los **criterios de convergencia de las sucesiones numéricas como criterio de terminación para el algoritmo**. A continuación, se muestra la evolución del módulo del gradiente del ejemplo, para ilustrar el paralelismo con el ejemplo del apartado (3.1.1):

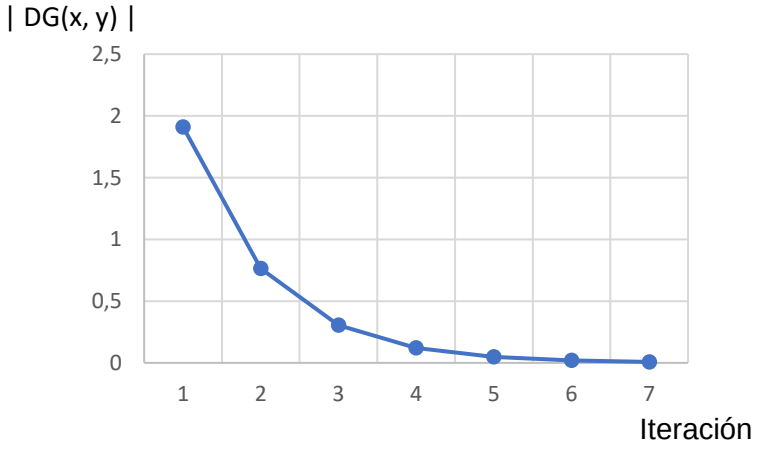


Figura 20: Evolución del gradiente.

3.2.2. Método de Montecarlo

El Método de Montecarlo es un algoritmo estadístico numérico ampliamente utilizado para la resolución de funciones matemáticas de gran complejidad. Consiste en la evaluación de la función en puntos aleatorios dentro de un rango determinado de valores, y extraer una solución basada en la probabilística de los resultados obtenidos. Los primeros datos sobre este método datan de 1944. El nacimiento de este método vino motivado por las labores de investigación sobre la bomba atómica. Pese a su origen, a lo largo de los años ha encontrado aplicación en gran cantidad de áreas de las matemáticas.

Un ejemplo clásico es el cálculo del valor de π. Sea un círculo de radio R inscrito en un cuadrado de lado 2R. Es sabido que el cociente entre el área del círculo y el área del cuadrado es igual a π/4. Para obtener una estimación del número π, el método de Montecarlo genera un número N de puntos aleatorios dentro del cuadrado. Así, una fracción de los puntos generados estará contenida dentro del círculo, y otra fuera del círculo. La relación de áreas será aproximadamente igual al cociente entre el número de puntos que han caído dentro del círculo y el número total de puntos generados N. Así, es posible obtener una estimación de π sin necesidad de conocer las áreas de las dos formas geométricas. El error cometido por el algoritmo decrece con $1/\sqrt{N}$.

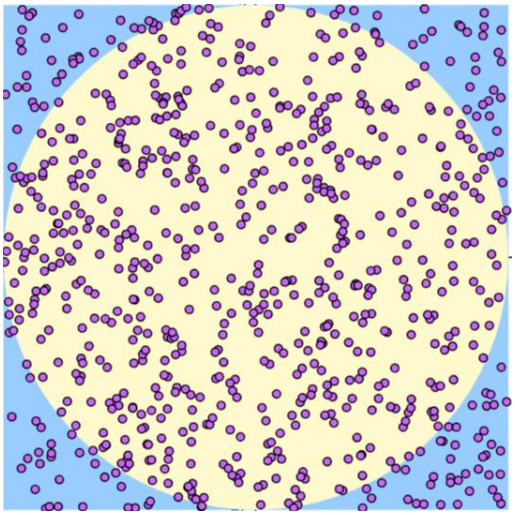


Figura 21: Estimación del número PI por el Método de Montecarlo.
Fuente: https://www.geogebra.org/m/cF7RwK3H

Para este trabajo, el Método de Montecarlo se ha utilizado en el **cálculo del volumen de trabajo del manipulador**. El procedimiento es similar al del cálculo del número π: se define un espacio de trabajo -*de forma cúbica*-, cuyo volumen es conocido. Se evalúa la posición del robot en puntos aleatorios de este volumen. Al finalizar, habrá una fracción de los puntos a los que el robot habrá podido llegar, y otra fracción de puntos que no haya sido posible alcanzar. La estimación del volumen de trabajo se extrae de la relación de puntos alcanzados entre el total de puntos, multiplicada por el volumen del espacio total. Esto se desarrollará con detalle en el apartado 4.3.4.

3.2.3 Algoritmos Genéticos

Como ya se mencionó en el capítulo de Estado del Arte, los Algoritmos Genéticos tienen como utilidad el cálculo de soluciones de problemas complejos sin necesidad de resolver expresiones analíticas. Por el contrario, el algoritmo genera una población de N soluciones posibles al problema a tratar, las evalúa, y comienza una serie de operaciones de selección y cruce para generar otra nueva población que herede los mejores genes de la anterior.

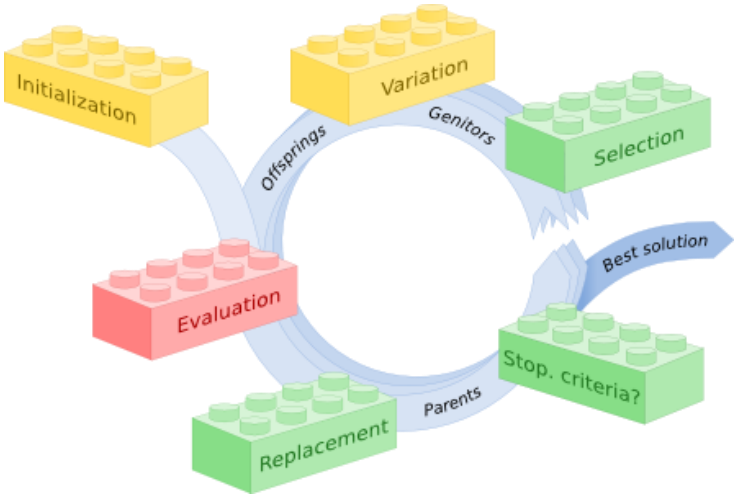


Figura 22: estructura básica de un algoritmo genético.
Fuente: *(15)* https://arxiv.org/abs/2105.00420

La figura ilustra la estructura básica de un algoritmo genético canónico. A continuación, se pasa a describir con detalle cada una de las etapas:

- ***Inicialización:*** consiste en la generación de una población inicial, optando generalmente por una primera solución aleatoria. No obstante, es posible condicionar el algoritmo partiendo de una primera posible solución aproximada. Es decir, que la primera solución no sea completamente aleatoria, sino que sus individuos se asemejen todos a una posible solución. Esto es especialmente útil a la hora de ejecutar varias veces el algoritmo, pues es posible introducir como población inicial la solución obtenida durante otra ejecución anterior.

  Es importante que en la población inicial exista una cierta variabilidad en los genes de los individuos. De no ser así, el algoritmo no explorará diferentes soluciones, estancándose en soluciones no óptimas. La variabilidad genética tiene también implicaciones biológicas, siendo un aspecto muy importante para el desarrollo exitoso de una población. Un ejemplo ampliamente conocido son los problemas que surgen cuando los padres de un individuo comparten una parte importante de su información genética. Históricamente, esto ha sido una problemática típica de las casas reales medievales de todo el mundo, siendo muy común el matrimonio entre miembros de las mismas familias generación tras generación. Resulta llamativo cómo ocurre exactamente lo mismo en el caso de estos algoritmos, los cuales necesitan un cierto grado de aleatoriedad en la información genética para llegar a soluciones exitosas.

- ***Evaluación:*** sin duda se trata de un aspecto clave en la implementación de un algoritmo genético, pues condicionará directamente la solución obtenida. Se trata de la función que pretende optimizarse *-bien maximizándola o minimizándola-*, por lo que su elección debe ser especialmente cuidadosa. Cada uno de los individuos son evaluados por una función de coste, que dictará qué individuos han tenido un buen desempeño y cuáles no.

  Existen muchas posibilidades a la hora de elegir una función de coste en función del número de parámetros que se pretenda optimizar. En este trabajo, tan solo se ha evaluado un único parámetro en cada ejecución – *manipulabilidad, volumen de trabajo, error de posición…*- por lo que la salida de esta función de coste será directamente el valor del parámetro elegido. Sin embargo, sería posible evaluar varios parámetros en la misma ejecución. La obtención de una puntación para el robot sería la combinación de los valores obtenidos para cada uno de los parámetros. Una posibilidad es ponderar cada uno de los parámetros mediante combinación lineal, dando a cada uno de ellos un nivel de importancia diferente. Fruto de esa combinación lineal se obtiene la puntuación del robot.

  a) Maximizando un parámetro: $p = f(x)$
  b) Minimizando un parámetro: $p = \frac{1}{f(x)}$
  c) Maximizando varios parámetros: $p = k_1 f_1(x) + k_2 f_2(x) + \cdots + k_m f_m(x)$
  d) Minimizando varios parámetros: $p = k_1 \frac{1}{f_1(x)} + k_2 \frac{1}{f_2(x)} + \cdots + k_m \frac{1}{f_m(x)}$

- ***Selección:*** consiste en, una vez evaluados los individuos de la población, seleccionar cuáles de ellos sobreviven y cuáles no. Esta elección debe ser también enfocada desde la probabilística: los individuos que hayan obtenido mejor puntuación durante la evaluación tendrán más probabilidades de supervivencia. Por tanto, que un individuo haya tenido una mala puntuación no implica que no vaya a sobrevivir, sino que tendrá menos probabilidades de hacerlo.

  En un principio, podría parecer que esto resulta negativo para la optimización de la información genética global, pues introducir individuos peor adaptados a la población empeora la puntuación media de esta. Sin embargo, ocurre precisamente todo lo contrario, pues al permitir la presencia de individuos con diferentes niveles de adaptación, se favorece la variabilidad genética y por tanto se aumentan las probabilidades de encontrar mejores soluciones.

  A modo de ejemplo, podría darse el caso de dos individuos A y B miembros de la misma especie. El individuo A tiene una puntuación mejor que el individuo B. Sin embargo, el individuo B, aunque pudiera pensarse que por su mala puntuación no contiene información genética de valor, contiene un gen concreto muy valioso y mucho mejor que el de su compañero A. Si no se permitiera a B transmitir su información genética a la siguiente generación, ese gen tan valioso se perdería y la especie no evolucionaría en ese aspecto concreto. De ahí la importancia de la variabilidad genética para la exploración de nuevas soluciones óptimas.

  Sobre este aspecto, hay otro asunto clave a la hora de seleccionar los individuos mejor adaptados. Dado que la selección tiene una naturaleza probabilística, es posible que en una de las etapas de selección, el individuo mejor

adaptado de la población no sobreviva, perdiéndose así su información genética. Sería conveniente que para evitar esto, se promoviera la supervivencia de los mejores individuos. A esto se le denomina *elitismo del algoritmo genético.* Según (10) los algoritmos genéticos necesitan de cierto grado de elitismo para conseguir soluciones óptimas.

- ***Cruce:*** se trata de la mezcla que se produce entre los individuos seleccionados. Al fin y al cabo, se trata de la recombinación de los individuos seleccionados para obtener una nueva población. Esta recombinación es conveniente hacerla de manera aleatoria por los motivos anteriormente citados.

  La forma de cruce depende directamente del tipo de genes elegidos. En caso de tratarse de números binarios, podrá hablarse de *cruce en un punto, cruce en N puntos* o *cruce uniforme.* De tratarse de números reales, podría realizarse una simple media geométrica entre los datos – *solución escogida en este trabajo*-.

Figura 23: diferentes tipos de cruce para genes binarios.
Fuente: (11)

- ***Mutación:*** se trata de otro de los aspectos clave de los algoritmos genéticos, siendo el responsable principal del éxito de estos algoritmos. Consiste en la introducción aleatoria de variaciones en la información genética de los individuos de una nueva población. La importancia de esto reside precisamente en la exploración de nuevas soluciones óptimas, introduciendo un cierto grado de variabilidad en los genes de cada individuo. La probabilidad de que un gen mute suele ser del orden de 1/N, siendo N el número de individuos de la población.

4. ENTORNO DE DESARROLLO

4.1 Justificación

Como ya se ha mencionado, el objetivo de este trabajo es la optimización del diseño de un CDHRR, para lo cual se han elegido los algoritmos genéticos como herramienta. Independientemente del desarrollo matemático que ha habido detrás, ha sido muy relevante el trabajo de programación necesario para conseguir un algoritmo genético funcional. Para llevar a cabo cualquier tipo de evaluación, **es necesario disponer de herramientas que permitan manejar y visualizar adecuadamente el robot**. Control cinemático, simulación gráfica, definición de variables constructivas, evaluación de posiciones, comprobación de colisiones... son algunas de las tareas que habrá que llevar a cabo para poder afrontar una tarea de esta naturaleza.

Para facilitar este proceso, dado el nivel de abstracción y complejidad que requerirán las labores a realizar, se ha hecho uso de POO –*"Programación Orientada a Objetos"-.* La POO favorece la modularidad del programa, simplificando el debugging y reduciendo considerablemente las líneas de código necesarias. La trazabilidad y la reutilización de código son también dos aspectos que se ven favorecidos gracias a la POO, reduciendo de forma directa el tiempo de desarrollo y el número de iteraciones necesarias para conseguir un código funcional.

Matlab ofrece un entorno de desarrollo muy sencillo e intuitivo para afrontar esta labor, facilitando en gran medida los cálculos a realizar: operaciones con matrices, resolución de sistemas de ecuaciones y operaciones matemáticas en general. Por otro lado, también dispone de herramientas para la elaboración de gráficos y animaciones, que resultarán muy interesantes en este trabajo.

Cabe destacar el uso de las diferentes Toolboxes de Matlab que se han utilizado. Estas han simplificado el trabajo en gran medida, permitiendo centrar esfuerzos en otras áreas más concretas y de mayor relevancia. Se han utilizado las siguientes Toolboxes:

-*"Robotics System Toolbox":* esta toolbox ha facilitado la definición de objetos básicos del robot, tales como sus articulaciones y partes móviles. Además, permite manejar cadenas cinemáticas de forma muy sencilla, gestionando las relaciones entre los diferentes eslabones que forman el robot.

-*"Parallel Computing Toolbox":* el algoritmo genético supone una gran carga computacional para realizar sus cálculos. Esta toolbox permite paralelizar algunos de los bucles del programa que más consumen, acelerando así el proceso de cálculo.

4.2 Descripción de Clases

Por todo lo anterior, **se ha elaborado un sistema de clases en Matlab** que permite la construcción, control, simulación y evaluación de un robot de tipo *Cable Driven Hyper Redundant Robot -CDHRR-.* Estas clases contienen los métodos y atributos necesarios para trabajar con estos robots de manera rápida e intuitiva, con el objetivo de favorecer la ampliación y optimización del código en trabajos futuros, así como el apoyo a las labores de control y simulación del robot CDHRR de Discos Pivotantes desarrollado por el CAR (ETSII UPM).

A continuación, se ofrece una breve descripción de las diferentes clases, sus métodos y atributos. Mencionar que, de cara al usuario, tan sólo resultan interesantes las clases HRRTree y HRRobot, en especial esta última. El resto están planteadas como clases auxiliares para desarrollo. En el Anexo 1 se ofrece un listado más detallado de los atributos y métodos de todas ellas.

- ***HRRObject***: se trata de la clase básica para los objetos *"disco"* y *"base"* del robot. Hereda de la clase *rigidBody* perteneciente a la Robotics System Toolbox de Matlab.

- ***HRRLink*** *y* ***HRRBase***: son clases hijas de *HRRObject*. Incluyen atributos relacionados con la geometría de las piezas del robot, así como métodos para la gestión de los elementos gráficos.

- ***HRRSection***: se trata de una clase auxiliar, utilizada para la generación de una sección del robot. Hereda de la clase *rigidBodyTree* de la Robotics System Toolbox. Contiene atributos relacionados con aspectos constructivos de las secciones – *número de discos, desfase alfa entre discos…*- así como métodos que facilitan la generación de una sección y la gestión de relaciones entre discos. No es una clase enfocada al usuario, sino a la ayuda durante el desarrollo.

- ***HRRTree***: esta clase hereda de *rigidBodyTree*, y contiene todos los elementos que componen a un robot CDHRR, siendo posible el control cinemático, su construcción y la visualización de su curva de *"backbone"*. Sin embargo, no tiene implementadas las funciones para visualizar y graficar el robot, pues estas se incluyen en la clase *HRRobot.*

- ***HRRobot***: hereda todos los métodos y atributos de *HRRTree*, pero además permite graficar y simular el robot. Mientras que *HRRTree* contiene sólo las funcionalidades relacionadas con aspectos constructivos y control, *HRRobot* incluye la parte gráfica y visual.

  Así, *HRRTree* se puede utilizar en tareas que requieran del manejo de varios objetos robot, ahorrando tiempo de procesamiento. Al no contener la parte gráfica, *HRRTree* tiene un menor tamaño, siendo menos costoso desde el punto de vista computacional trabajar con instancias de esta clase. De ser necesario visualizar y simular el robot, la clase *HRRobot* proporciona todos los métodos necesarios.

4.3 Funcionalidades

4.3.1 Creación de un Robot

La funcionalidad básica e imprescindible es la creación de un robot especificando los parámetros de diseño deseados. En concreto, el programa permite definir tres parámetros: *Número de secciones, Número de discos por sección* y *Ángulo alfa de las secciones*. Por ejemplo, si se desea construir un robot que tenga 4 secciones, con un número de discos de 30, 22, 15 y 7. Los ángulos alfa de las secciones se establecerán a 30, 60, 90 y 120° respectivamente. Las instrucciones necesarias para crear y visualizar el robot serían:

```
nLinks = [30, 22, 15, 7];
alfas = [pi/6, pi/3, pi/2, 2*pi/3];
robot1 = HRRobot(nLinks, alfas);
robot1.plot2('BodyColor', 'blue', 'EndLinkColor', 'green');
```

Es decir, que tan solo es necesario inicializar los vectores con el número de discos y la distribución alfa deseada, y llamar al constructor de la clase HRRobot. Para visualizarlo, el método **`plot2`** permite establecer parámetros gráficos. La figura generada es la siguiente:

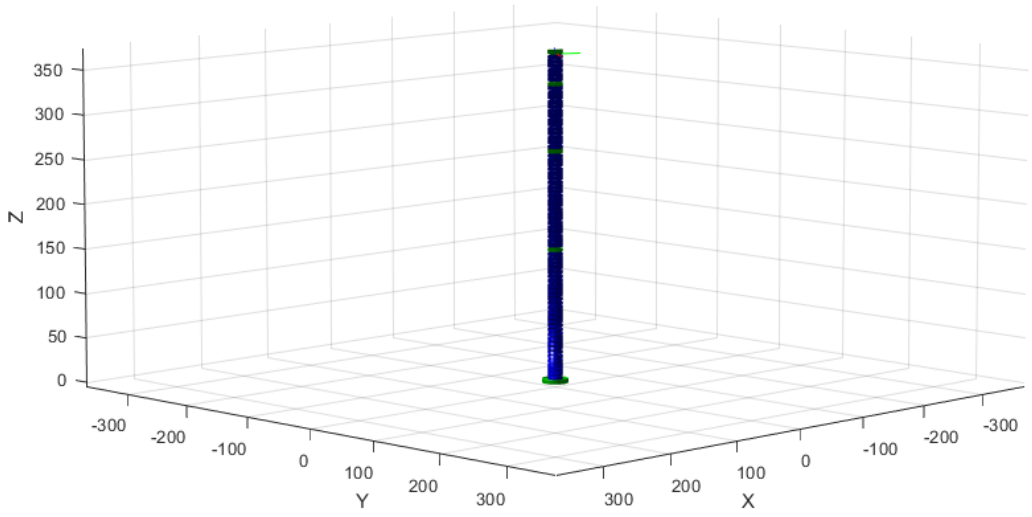


Figura 24: creación de CDHRR de [30, 22, 15, 7] discos.

Como puede verse, el robot se imprime es su posición de reposo. Haciendo zoom en los discos de las diferentes secciones, se puede apreciar el desfase alfa entre ellos:

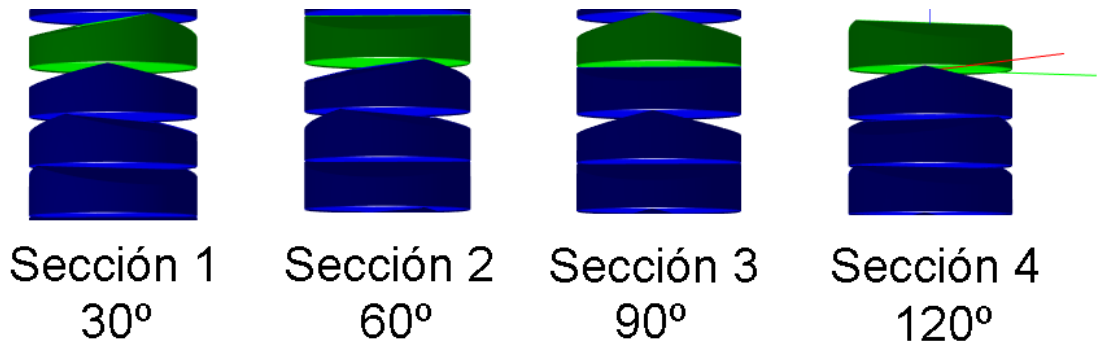


Figura 25: detalle del ángulo alfa de las secciones.

4.3.2 Posicionamiento

Una vez creado el robot, habrá que posicionarlo en un punto. La versión actual del programa no permite el posicionamiento del extremo con orientación, si no que tan solo es posible hacerlo en traslación – *x, y, z*-. Esto es así porque por cuestiones de tiempo se ha priorizado esta funcionalidad, aunque se planea implementarlo en un futuro.

La clase HRRobot contiene varios métodos que hacen posible el posicionamiento del robot indicando un punto en el espacio. Estos métodos calculan la cinemática inversa utilizando diferentes metodologías. En concreto, se han implementado tres cinemáticas diferentes, las cuales serán desarrolladas en detalle en el apartado 5. Estas son: *C.I. por el Descenso del Gradiente*, *C.I. por la Jacobiana de los Discos* y *C.I. por la Jacobiana de las Secciones*.

Para simplificar el trabajo al usuario, se ha desarrollado un método que permite elegir entre las diferentes cinemáticas fácilmente. Este es el método **`move`**, el cual calcula la cinemática según el procedimiento seleccionado y mueve el robot según la pose calculada. Por ejemplo, partiendo del programa anterior, es posible llevar el extremo del robot al punto (135, 80.34, 62.11) e imprimir su posición.

```
[newConfig, error, iter] = robot1.move(135, 80.34, 62.11, 'Default');
robot1.plot2();
```

Es posible consultar el error cometido, el número de iteraciones que ha llevado calcular la pose, así como la posición actual del efector final del robot:

```
disp(error);
disp(iter);
disp(robot1.currentPos());
```

La salida por pantalla es la siguiente:

```
>> {[0.0021]}

>> {[34]}

>> 134.9997   80.3380   62.1098
```

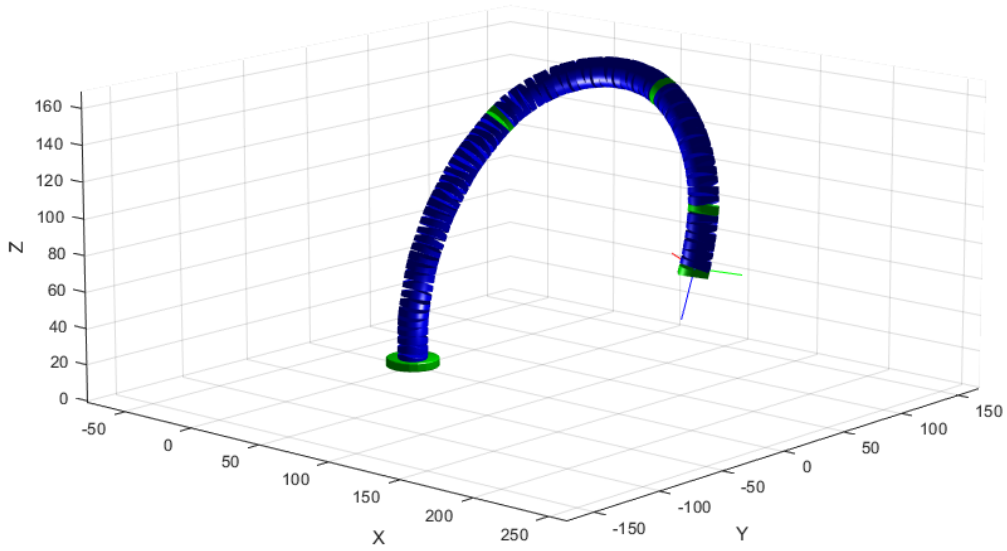


Figura 26: figura generada. Punto [135, 80.34, 62.11].]

Como parámetros, el método **move** recibe las coordenadas X, Y, Z del punto al que se desea llegar y el método de cálculo de la cinemática inversa, entre otros opcionales. El método por defecto – `'Default'`- utiliza la Jacobiana calculada por Discos para realizar una primera aproximación de la pose, y la corrige iterativamente utilizando la Jacobiana calculada por Parámetros de Sección. En próximos apartados se hará una comparativa entre la precisión y carga computacional de los diferentes métodos implementados. Observar que el error obtenido en este ejemplo es de 0.0021 unidades en 34 iteraciones.

El parámetro de error es completamente configurable, siendo posible pasar su valor por parámetro en función de las necesidades de precisión requeridas. Dado que el robot tiene una longitud de 5 unidades por cada disco -*atributo configurado por defecto en la clase HRRLink*-, el error cometido es ínfimo. Es posible consultar la posición actual del robot con el método `currentPos`. La salida obtenida es [134.9997 80.3380 62.1098], muy cercano al punto deseado.

4.3.3 Trayectorias

Una de las funcionalidades que se requiere de cualquier manipulador en la industria es describir trayectorias dentro de un espacio de trabajo. Realizar cordones de soldadura, mover pieza esquivando obstáculos, aplicar cordones de pegamento son algunas de las casi infinitas tareas que podría pedírsele a un robot manipulador. Todas estas tienen en común un aspecto clave: requieren que el robot se vaya moviendo entre varios puntos. Por ello, una funcionalidad obligada de este programa debe ser el trabajo con trayectorias.

Para ello, se ha dotado al método **move** de una característica esencial: acepta vectores de puntos como parámetros, de manera que es posible enviarle los puntos por los cuales se desea que se mueva el robot, y el método **move** irá calculando las poses necesarias para alcanzarlos. Además, como parámetros devolverá tres arrays: un array con las configuraciones en cada punto, otro con el error cometido en cada uno de ellos y otro con el número de iteraciones que ha necesitado para calcular la cinemática de cada punto.

Partiendo del robot del ejemplo anterior, partiendo del punto [135, 80.34, 62.11] en el que se le había situado, sería posible definir una trayectoria de puntos para que el robot la fuera siguiendo progresivamente. Por ejemplo, se definirá una trayectoria helicoidal de 6 puntos:

```
z = [135:40:335];

x = 100*cos(z/335*2*pi);

y = 100*sin(z/335*2*pi);
```

Ya con el conjunto de puntos de la trayectoria, se reseteará la pose del robot para partir desde el reposo. Antes de comenzar a recorrer la trayectoria, **conviene realizar una primera aproximación al primer punto, dado que se parte de otro muy alejado**. Para realizar este primer posicionamiento lo más recomendable es utilizar el Descenso del Gradiente, pues, al ser un método iterativo, las aproximaciones a las derivadas iteración tras iteración serán más precisas. Los métodos de la Jacobiana, por el contrario, cometerán un gran error en los puntos más alejados, haciendo que el algoritmo no converja. En este caso se ha establecido el error de posición máximo a 1 unidad para agilizar los cálculos.

```
robot1.resetConfig();

robot1.move(x(1), y(1), z(1), 'GradientDescent', 'Error', 1);
```

Una vez situado el robot en un punto cercano al inicial, tan sólo resta calcular las poses en los puntos de la trayectoria mediante el método **`move`**. Es posible pasar como parámetros los vectores **`x, y, z`**. El método devolverá como salidas los arrays con las configuraciones, errores cometidos e iteraciones realizadas para calcular cada punto. En este caso, ya sí que es más conveniente utilizar las cinemáticas de la Jacobiana, pues los puntos son más cercanos y la pose que se obtendrá será más estilizada que con el Descenso del Gradiente:

```
[newConfig, error, iter] = robot1.move(x, y, z, 'Default', 'Error', 1);
```

Es posible imprimir las poses del robot en cada punto, así como la trayectoria descrita. Para los puntos intermedios, se imprimirá el robot como una línea curva – *conocida como "backbone curve"-.*

```
for i=1:5
    robot1.setConfig(newConfig{i});
    robot1.plot2('Visuals','off');
    hold on;
end
robot1.setConfig(newConfig{6});
robot1.plot2('Visuals','on');
hold on;
scatter3(x,y,z,'ok','filled');
hold on;
plot3(x,y,z,'k');
```

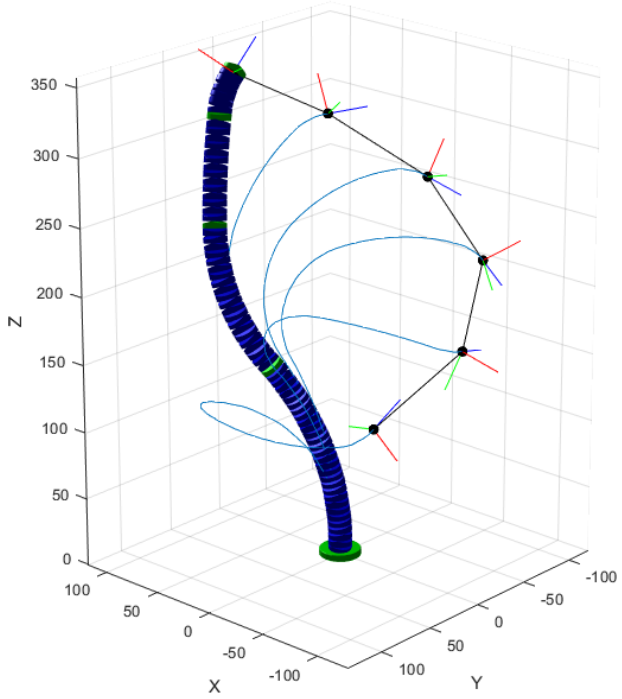


Figura 27: Trayectoria obtenida.

Si se imprime el array **`error`**, se puede ver como el robot consigue posicionarse con éxito en todos los puntos de la trayectoria, cometiendo un error inferior al que se le ha solicitado:

```
>> disp(error)
    {[0.0151]}  {[0.2632]}  {[0.0076]}  {[0.0092]}  {[0.0094]}  {[0.1240]}
```

4.3.4 Volumen de Trabajo

Otra de las funcionalidades que se han incluido es el cálculo del volumen de trabajo del manipulador. Para ello se ha implementado un script de ejemplo –**"workVolume"-**, en el que se calcula este parámetro mediante el Método de Montecarlo. El volumen de trabajo es uno de los parámetros más importantes a la hora de evaluar un robot, pues permite deducir la movilidad de este en las diferentes zonas del espacio. Que un robot tenga un volumen de trabajo mayor que otro no implica que tenga una mayor movilidad, pero puede ser una primera comparación cualitativa del desempeño que va a tener.

Como se mencionó en el apartado 3, el método de Montecarlo obtiene una estimación estadística de la función implicada. Consiste en la evaluación de la función en una serie de puntos aleatorios, y obtener una estimación de la solución en función de los resultados obtenidos. En el caso del cálculo del volumen de trabajo, se ha llevado a cabo como sigue:

1. Se define un cubo que encierre el volumen de trabajo en el que se podrá mover el robot. El volumen de ese cubo será conocido.
2. Se genera un punto aleatorio contenido en el cubo, y se evalúa si el robot es capaz de alcanzarlo. Esto se hace un número N de veces.
3. El volumen de trabajo será aproximadamente igual al volumen del cubo multiplicado por la relación entre el número de puntos alcanzados N_a y el número total de puntos generados N:

$$V_w \approx v \cdot \frac{N_a}{N}$$

Al tratarse de un método iterativo en el que además habrá que calcular la cinemática inversa del robot en cada punto, se trata de un algoritmo de gran carga computacional. Cuanto menor sea el error que se desee obtener, mayor será el número de iteraciones necesarias y, por tanto, mayor el tiempo de cálculo. Como se dijo en el apartado 3, el error cometido por el Método de Montecarlo decrece como $1/\sqrt{N}$. Esto permitirá establecer un número de iteraciones aproximado en función del error que se haya establecido. Por ejemplo, el número de puntos a estudiar para obtener un error del 10% será del orden de 100.

A la hora de ejecutar este algoritmo son muchos los puntos en los que evaluar la cinemática, por lo que es conveniente buscar estrategias para reducir el tiempo de cálculo. Una de estas estrategias es la **búsqueda de simetrías en el volumen de trabajo**. En este aspecto, sólo influye la distribución de los ángulos α de los discos, pues son los que marcan las direcciones preferentes de giro del robot. En la siguiente imagen se muestra cómo es posible favorecer unas direcciones u otras en función de las diferentes distribuciones posibles de este ángulo:

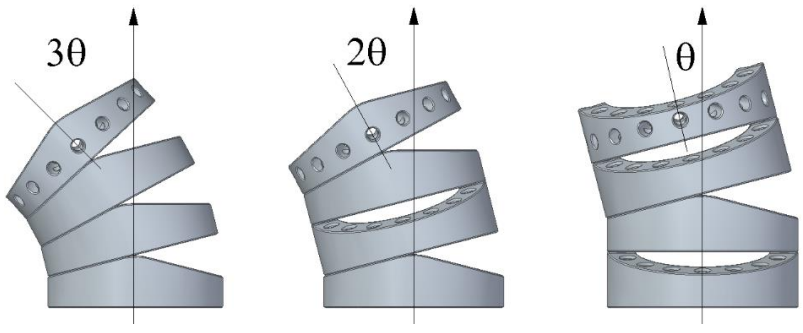


Figura 28: influencia del ángulo α en la dirección de máxima movilidad.
Fuente: elaboración propia.

En el caso de la izquierda, α = 0º, se favorece la movilidad en la dirección perpendicular al eje de las articulaciones de los discos, mientras que se inhibe completamente el movimiento en otras direcciones. Por el contrario, en las otras configuraciones con α = 90º, es posible el movimiento en ambas direcciones. Sin embargo, dado que el disco base es fijo, el número de discos móviles en estos ejemplos es de tres. Al estar los discos montados a 90º, habrá dos discos con α = 90 y sólo uno con α = 0º. El resultado de esto es una distribución asimétrica de los grados de libertad, favoreciendo unas direcciones de giro y perjudicando otras – *en la dirección preferente, la sección del ejemplo puede girar el doble que en la dirección perjudicada, al haber dos discos a 90º frente a uno a 0º*-.

Esto en principio no tiene por qué ser un inconveniente, pudiendo ser algo intencional en el diseño en caso de no requerir de un espacio de trabajo simétrico. De este modo es posible dar movilidad extra en unas direcciones y restringir el movimiento en otras en las que no sea necesario. No obstante, para el caso que ocupa este apartado, cuanto más simétrico sea el espacio de trabajo, más sencillo será el cálculo de su volumen, pues las posibilidades de simplificar su forma serán mayores.

Por este motivo, se ha decidido reducir el estudio del volumen de trabajo a casos más simples, para poder realizar pruebas más rápidamente. En concreto, se plantea a modo de ejemplo una estimación de la forma del espacio de trabajo de un robot con los siguientes parámetros de montaje:

```
nLinks = [20 12 8 5]; alfas = [pi/2 pi/2 pi/2 pi/2]
```

Haciendo uso del script anteriormente citado, se ha calculado la forma que tendría la sección central del volumen de trabajo de este robot evaluando puntos aleatorios en un área rectangular. Los puntos exitosos se han impreso de color azul, mientras que los rojos son puntos a los que el robot no ha logrado llegar.

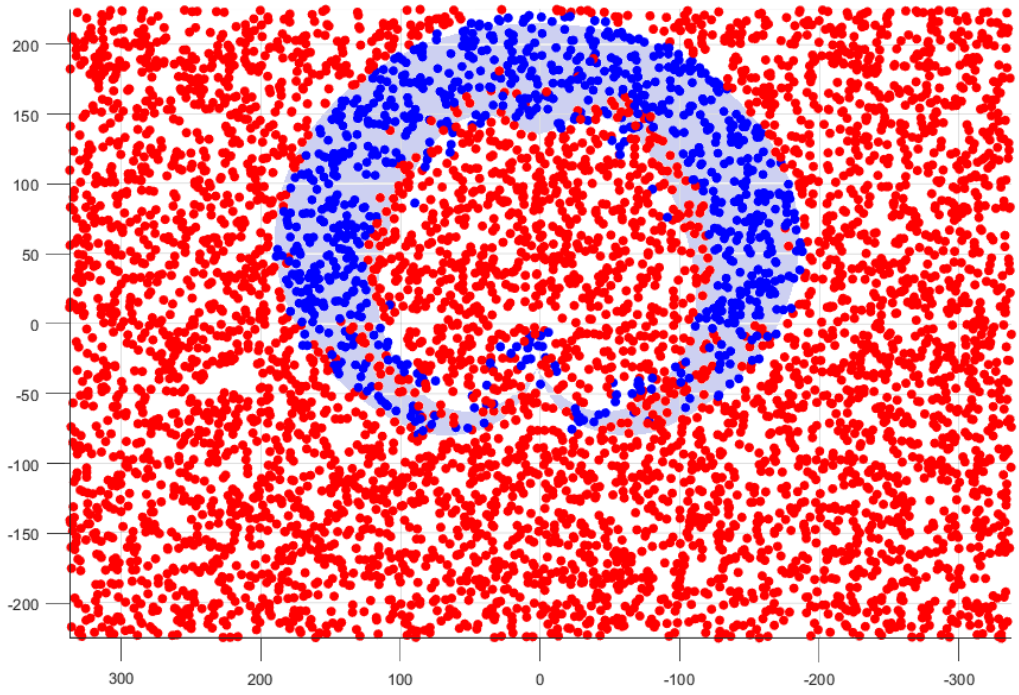


Figura 29: sección central de un robot CDHRR de discos pivotantes.
Configuración: nLinks = [20 12 8 5]; alfas = [pi/2 pi/2 pi/2 pi/2], N = 5000 puntos.
Porcentaje de puntos exitosos: 14.46%
Fuente: elaboración propia.

Para la elaboración de la imagen anterior se ha seguido la siguiente estrategia: en primer lugar, se genera un punto aleatorio dentro del rango establecido. En segundo lugar, se posiciona el robot en un punto de partida cercano en función del cuadrante en el que se encuentre el punto objetivo. Por último, se calcula la cinemática según el método descrito en el apartado 5.5 Control Cinemático por el Método de la Jacobiana. Se pueden observar algunos puntos conflictivos en la zona inferior del volumen de trabajo. Son puntos muy cercanos a los puntos singulares del robot, lo que provoca que el error cometido en el posicionamiento sea mayor. Tener en cuenta que, para reducir el tiempo de cálculo de la imagen, se ha reducido la precisión del algoritmo, siendo esta otra de las posibles causas de estos puntos atipo.

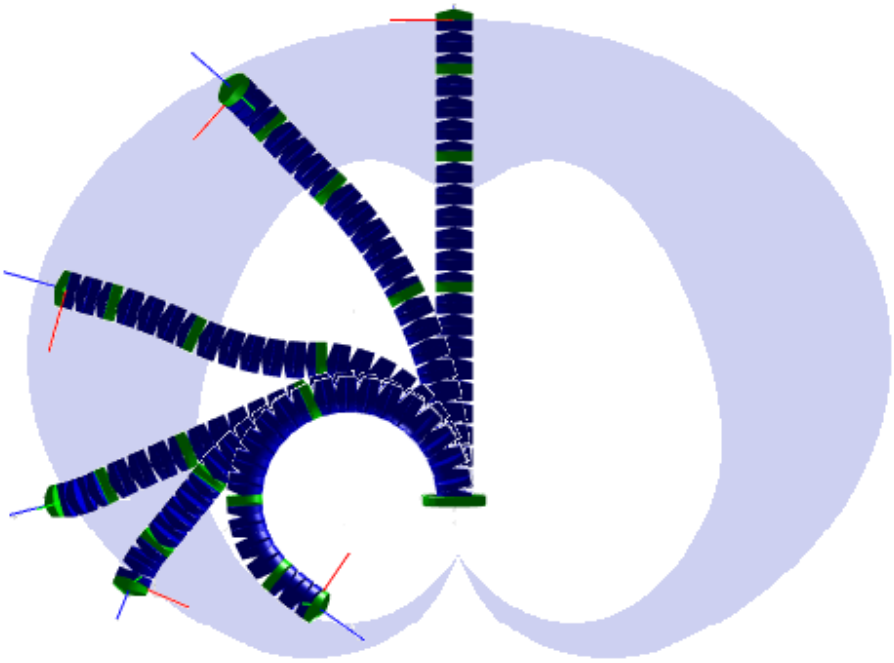


Figura 30: aproximación de la sección central del volumen de trabajo.
Fuente: elaboración propia.

El volumen de trabajo real del robot será la revolución del área sombreada. La parte interna de esta área – *con forma de manzana invertida*- es una parte restringida para el robot, suponiendo una gran cantidad de volumen de trabajo perdido. El tamaño de esta región está directamente influenciado por el ángulo máximo β de giro de los discos – *descripción de los parámetros de diseño en el apartado 5.1 Nomenclatura y Sistemas de Coordenadas* -. Cuanto mayor sea el ángulo que puede girar cada disco con respecto al anterior, mayor capacidad tendrá el robot de *"enrollarse"* sobre sí mismo. Por tanto, **la mejor estrategia para reducir esta zona interna restringida sería aumentar la capacidad de pivote de los discos**.

La zona mejor condicionada para el movimiento del robot es la parte media del área sombreada, es decir, con el robot ligeramente inclinado. Se puede ver fácilmente cómo la zona más ancha se encuentra aproximadamente a la altura de la mitad del robot. La forma obtenida variará, como es lógico, en función de la elección que se haya hecho a la hora de establecer los parámetros de diseño del robot. No obstante, resulta muy ilustrativo ver la forma aproximada obtenida, pues tendrá una forma similar para otros casos.

La imagen obtenida también permite hacer una predicción sobre cuál será la posición óptima a la hora de instalar un robot colaborativo de este tipo como manipulador. Por ejemplo, si se utilizase este robot para manejar objetos en una mesa horizontal, lo óptimo sería montarlo de manera que su zona de máxima maniobrabilidad coincidiera con las necesidades de la tarea a desempeñar:

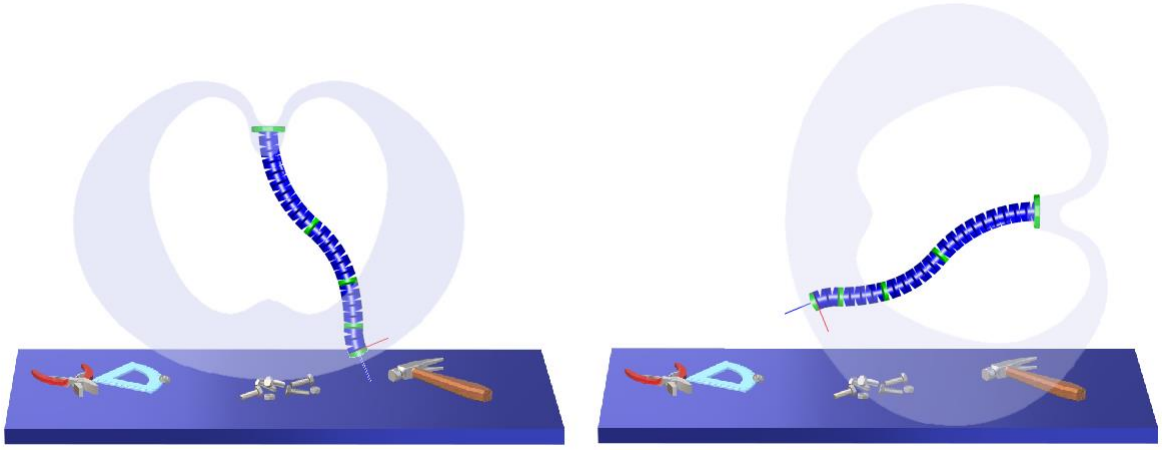


Figura 31: ejemplos de aplicación como robot colaborativo (imagen artística).
Fuente: *elaboración propia.*

En la figura superior se ha querido ilustrar desde un punto de vista cualitativo la importancia y utilidad del cálculo de la zona de trabajo del robot. Diferentes montajes tendrán un buen desempeño en unas tareas, y malo en otras. Por ejemplo, en la imagen de la izquierda el robot estaría mejor condicionado para el trabajo en dirección horizontal, llegando a zonas más lejanas de la mesa. Este montaje sería ideal para labores *pick and place*. Sin embargo, en la imagen de la derecha el robot estaría mejor condicionado para el trabajo en altura, llegando a zonas menos alejadas, pero teniendo mejor movilidad en una zona más reducida. Esta configuración podría ser la adecuada en el caso de ser usado entareas de atornillado, montaje o como robot auxiliar.

5. CONTROL CINEMÁTICO

El control cinemático ha resultado ser uno de los puntos más relevantes de este trabajo. Se trata de un aspecto básico, pues sin poder posicionar el robot en poses o puntos a voluntad no tiene sentido siquiera intentar abordar cualquier tipo de problema de optimización. Es por esto por lo que se ha puesto un **especial esfuerzo en conseguir una cinemática fiable**, capaz de posicionar el robot con un error razonablemente pequeño, sin que el tiempo de cálculo sea excesivo.

Como se expuso en el apartado 2, los robots CDHRR de Discos Pivotantes están compuestos por grupos de vértebras a los que se ha denominado *"secciones"*, que son actuadas mediante cables tensores. **Para elaborar un modelo cinemático del robot será necesario elaborar uno previo para el control de estas secciones**, dividiendo así el problema cinemático en dos: *Control de Secciones* y *Control Total del Robot*.

5.1 Nomenclatura y Sistemas de Coordenadas

Se ha establecido una nomenclatura para referirse a cada una de las variables constructivas de la sección para llevar a cabo esta tarea de forma coherente. Como consideraciones previas:

- Los discos de una sección de N discos **se numeran desde el 0**, correspondiente al primer disco o *"disco base"*, **hasta el N-1**, correspondiente al último disco o *"disco extremo".*
- **Se ha aproximado el movimiento de pivote entre discos a una rotación pura**. Así, se considera que cada disco contiene una articulación rotativa en su extremo, alrededor de la cual rota el disco siguiente.
- El sistema de referencia de un disco estará definido según el criterio de Denavit-Hartenberg. El **eje X coincidirá con el eje axial de los discos** y el **Z coincide con el eje de la articulación** del disco. El origen del sistema está contenido en la planta del disco.
- Se ha definido un sistema global para una sección al que se ha llamado *"Sistema Base"*. Este **coincide con el sistema de su disco base {S_0}**, y será útil a la hora de definir la posición general de la sección, o posiciones de los discos con relación a la base de la sección.
- Se ha supuesto que la tensión de los cables de una sección se distribuye uniformemente por todos sus discos. Esto supone asumir que **las secciones describen arcos de circunferencia**, ajustándose así a la *Hipótesis de Curvatura Constante (PCC)*.

Pasan a describirse a continuación los diferentes términos que se han utilizado:

- **Ángulo β:** es el ángulo que forman el eje X de un disco con el del anterior. Se designa con el subíndice del disco al que está referido, que es el disco móvil. Así, β_1 es el ángulo que forma X_1 con X_0, siendo 1 el disco móvil. Es un parámetro de control, pues es el ángulo que gira cada disco al mover la sección. Su valor máximo es otro parámetro importante de diseño, siendo de 15º para el caso del Pylori I.
- **Ángulo α:** se trata del ángulo que forman los ejes Z de dos discos cuando la sección está en posición extendida – *ángulos* β *iguales a cero* –. Así, el ángulo $\alpha_{2,1}$ es el ángulo que forman los ejes Z_2 y Z_1. No obstante, en el cálculo de la cinemática es más relevante conocer el ángulo que forma cada disco con respecto al sistema base,

es decir, conocer los $\alpha_{i,\,0}$. Para simplificar la notación, en este caso se hablará simplemente de α_i, que será el ángulo de montaje del disco i-ésimo con respecto al sistema base. Este ángulo es uno de los parámetros de diseño a optimizar.

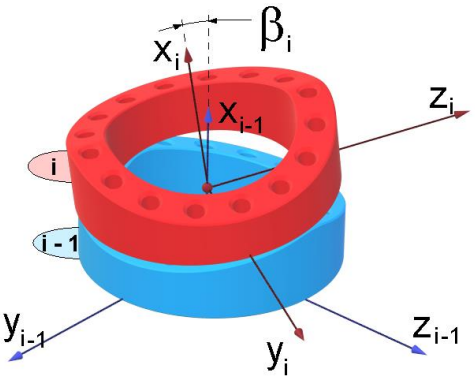


Figura 32: Representación del ángulo Beta.
Fuente: *elaboración propia.*

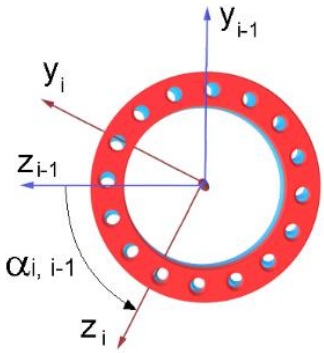


Figura 33: ángulo alfa.
Fuente: *elaboración propia.*

- **Altura h:** es la distancia desde el plano de la cara base del disco hasta el eje de pivote.
- **Radio medio Rm:** es el radio medio de la circunferencia sobre la que se encuentran los orificios pasantes para los cables.
- **Radio externo Re:** es el radio de la circunferencia total del disco.
- **Radio interno Ri:** radio del orificio interno del disco.
- **Ángulo θ:** es el ángulo que forma el eje X del sistema del extremo con el eje X del sistema base de la sección. El sistema base de la sección coincide con el del primer disco de esta.

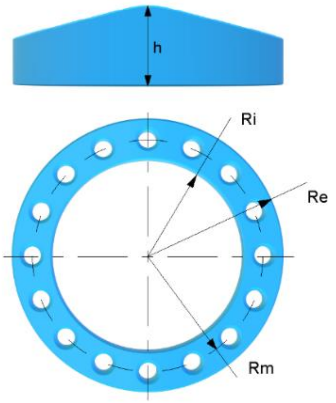


Figura 34: dimensiones del disco
Fuente: *elaboración propia.*

- **Ángulo φ:** es el ángulo que forma la proyección del eje X del extremo en el plano Z_0Y_0 del sistema base, con el eje Z_0 del sistema base. Observar que es equivalente a la coordenada φ del origen del sistema del extremo, en esféricas.

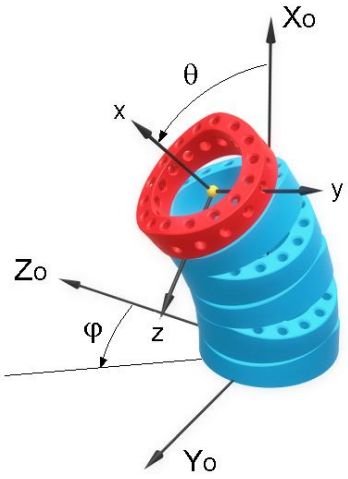


Figura 35: representación de los ángulos theta y phi de la sección.
Fuente: *elaboración propia*.

5.2 Cinemática de Sección

Para obtener el modelo cinemático de una sección se va a comenzar obteniendo la matriz de transformación homogénea entre dos discos, desde el disco de índice (i-1) hasta el de índice i. Según la nomenclatura seguida, la matriz de transformación del disco i-ésimo tiene la siguiente expresión:

$$A_{(i-1),i} = \begin{bmatrix} \cos(\beta i) & -sen(\beta i)\cdot\cos(\alpha_{(i-1),i}) & sen(\beta i)\cdot sen(\alpha_{(i-1),i}) & h \\ sen(\beta i) & \cos(\beta i)\cdot\cos(\alpha_{(i-1),i}) & -\cos(\beta i)\cdot sen(\alpha_{(i-1),i}) & 0 \\ 0 & sen(\alpha_{(i-1),i}) & \cos(\alpha_{(i-1),i}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.1)$$

Esta matriz se ha calculado como composición de dos giros y una traslación: **rotación de ángulo β en torno al eje Z**, **rotación de ángulo α en torno al eje X** y, por último, **traslación de longitud h a lo largo de X**.

Para obtener la matriz de transformación desde la base de la sección hasta el extremo habría que multiplicar las matrices asociadas a cada disco desde el primero hasta el último. Las matrices $A_{(k-1),k}$ siguen la expresión 4.1.

$$A_i = \Pi_{k=0}^{i} A_{(k-1),k} = A_{0,1}\cdot A_{1,2}\cdot A_{2,3}\cdots A_{(i-1),i}$$

Esto a priori no supone un problema: el ángulo α y la altura h son parámetros de diseño, y el ángulo β de los discos puede ser conocido en caso de saber la pose del robot. Por ello, **la Cinemática Directa no plantea ningún problema y puede resolverse fácilmente** utilizando estas matrices. Recordar que la matriz de transformación homogénea contiene información sobre la orientación y la posición del disco al que está referida. Es decir, consiguiendo esta matriz, es posible conocer exactamente la posición y orientación del disco en cuestión.

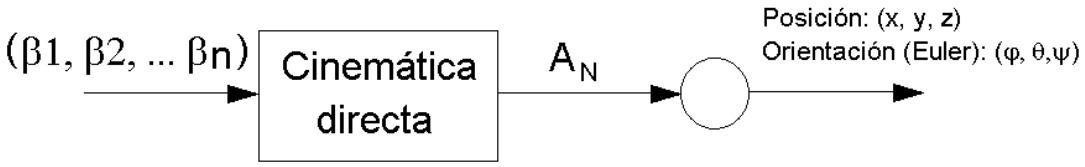


Figura 36: esquema de la Cinemática Directa. Es posible obtener la posición y la orientación de un disco a partir de su matriz de transformación.

El problema reside en el cálculo de la cinemática inversa. Es decir: **conocidos los ángulos θ y φ que gira la sección, se busca averiguar cuál es el ángulo β que gira cada disco**:

Figura 37: esquema Cinemática Inversa

El ángulo que giran los discos no es el mismo, sino que cada uno gira un ángulo diferente en función del α con el que han sido diseñados. La matriz de transformación del extremo ($A_{extremo}$) será función de los ángulos β_i de los discos anteriores, siendo incógnitas dentro de la matriz de transformación del extremo, según la expresión 4.1. Despejar estas incógnitas no es tarea sencilla, dado que **se encuentran dentro de funciones trigonométricas** – *senos y cosenos*-.

Puesto que no es sencillo resolver el sistema planteado, se han buscado formas de simplificar la expresión de la matriz de transformación del extremo. El inconveniente de su resolución reside en la presencia de los ángulos β como incógnitas dentro de funciones trigonométricas. Recordar que son precisamente esos ángulos lo que se pretende calcular. Si las matrices de transformación pudieran simplificarse, de modo que no apareciera ningún ángulo β en ellas, el sistema a resolver resultaría más sencillo.

Debido a esto último, se ha hecho lo siguiente: **se han eliminado los ángulos β de la expresión de la matriz de transformación** para conseguir un sistema de ecuaciones más sencillo. Esto hará que el sistema que se plantee sea una mera aproximación, al no contemplar el movimiento real de los discos. El objetivo de esto no es plantear como válida dicha aproximación, sino utilizar este sistema aproximado para **obtener una solución de forma iterativa**, corrigiendo el término independiente iteración tras iteración hasta que la solución obtenida se ajuste a la real.

La transformación entre discos al eliminar las incógnitas β resulta como sigue:

$$A_{(i-1),i} = \begin{bmatrix} 1 & 0 & 0 & h \\ 0 & \cos(\alpha_{(i-1),i}) & -sen(\alpha_{(i-1),i}) & 0 \\ 0 & sen(\alpha_{(i-1),i}) & \cos(\alpha_{(i-1),i}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.2)$$

Es decir, que se simplifica la transformación entre discos a una simple rotación en torno al eje X, y una traslación. Esta aproximación será tanto mejor cuanto mayor sea el número de discos, pues menor será el ángulo β de los discos. Al haberse simplificado la transformación a un giro en torno al eje X, es equivalente a que la sección estuviera completamente recta, dado que los ángulos β son nulos. Recordar que esta situación es completamente ficticia, con el único propósito de plantear un sistema más sencillo para utilizarlo en el método iterativo.

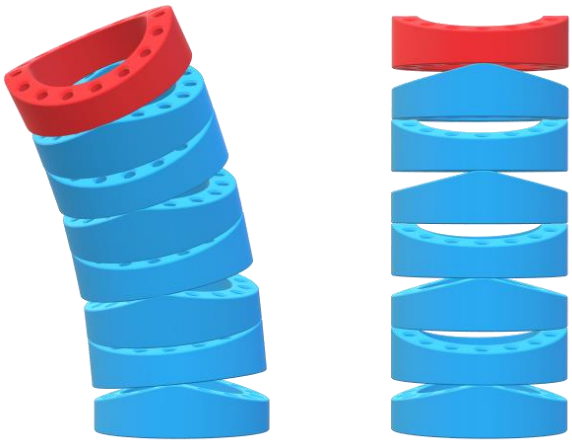


Figura 38: sección desplazada (izquierda), sección tras la aproximación (derecha).
Fuente: *elaboración propia*.

La aproximación realizada al no tener en cuenta los β en las transformaciones **simplifica la posición en la que estarían los discos**, quedando la sección en una posición recta. Dado que todos los discos quedarían uno encima del otro, sería posible proyectar los sistemas de los discos al sistema base, quedando todos ellos en el mismo plano:

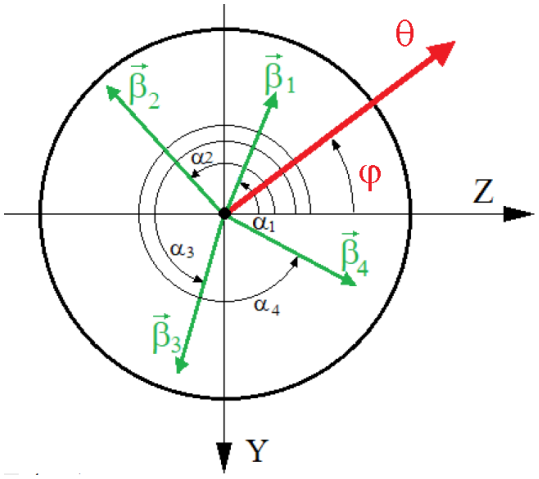


Figura 39: proyección de los vectores rotación de las articulaciones.
Fuente: *elaboración propia*.

Los vectores $\vec{\boldsymbol{\beta}}$ de la figura tienen como módulo el ángulo girado por cada articulación, y sus direcciones vienen dadas por los ángulos α. En el caso del vector $\vec{\boldsymbol{\theta}}$, su módulo representa el ángulo θ que gira la sección, y su dirección viene dada por el ángulo φ de esta. Puesto que los vectores son coplanares, la suma de los vectores rotación de las articulaciones debe ser igual al vector rotación de la sección. Esta combinación lineal se puede expresar de forma matricial de la siguiente manera:

$$\theta \cdot \begin{bmatrix} cos(\varphi) \\ sen(\varphi) \end{bmatrix} = \begin{bmatrix} cos(\alpha_1) & cos(\alpha_2) & \cdots & cos(\alpha_n) \\ sen(\alpha_1) & sen(\alpha)_2 & \cdots & sen(\alpha_n) \end{bmatrix} \cdot \begin{bmatrix} \beta_1 \\ \beta_2 \\ \dots \\ \beta_n \end{bmatrix} \qquad (4.3)$$

Se trata de un sistema compatible indeterminado, lo cual complica su resolución. Sin embargo, es posible calcular la *Solución de Mínima Norma* muy fácilmente:

$$\boldsymbol{M} = \begin{bmatrix} cos(\alpha_1) & cos(\alpha_2) & \cdots & cos(\alpha_n) \\ sen(\alpha_1) & sen(\alpha)_2 & \cdots & sen(\alpha_n) \end{bmatrix}, \qquad \boldsymbol{b} = \theta \cdot \begin{bmatrix} cos(\varphi) \\ sen(\varphi) \end{bmatrix}$$

$$\boldsymbol{\beta}^* = \boldsymbol{M}^T \cdot \boldsymbol{c}_{,,} \;\; \boldsymbol{M}\boldsymbol{M}^T \cdot \boldsymbol{c} = \boldsymbol{b} \qquad (4.4)$$

El vector solución $\boldsymbol{\beta}^*$ contiene los **ángulos β_i ficticios** de cada disco. Es decir, al no haber tenido en cuenta los ángulos β_i en la matriz de transformación, ha sido posible plantear la situación en la que los discos son coplanares. En esa situación imaginaria, se puede plantear la ecuación (4.3), y obtener una **solución ficticia** $\boldsymbol{\beta}^*$.

Con esta solución ficticia, al ejecutar su Cinemática Directa, se obtendrá una posición diferente a la deseada. En función del error cometido, se **corregirá el vector b de la ecuación 4.4**. Resolviendo el sistema con el nuevo vector corregido, se volverá a obtener un vector solución $\boldsymbol{\beta}^*$que esta vez **se acercará un poco más a la solución real**. Haciendo esto de forma **iterativa**, llegará un momento en que al ejecutar la cinemática directa del vector $\boldsymbol{\beta}^*$no se cometa un error significativo, momento en el cual se habrá logrado la solución real a la cinemática inversa.

El procedimiento es el siguiente:

1. Dados los ángulos θ y φ deseados en la sección, obtener una primera solución β_0 mediante la expresión 4.4.
2. Obtener los ángulos θ_r y φ_r que consigue β_0 haciendo uso de la cinemática directa *-multiplicación de las matrices de transformación de los discos según la expresión 4.1-.*
3. Calcular los errores cometidos como $e_\theta = |\theta_r - \theta|$, $e_\varphi = |\varphi_r - \varphi|$.
4. Volver al paso 1, modificar el vector b corrigiendo los θ y φ de entrada ponderando el error cometido por una constante de corrección K.

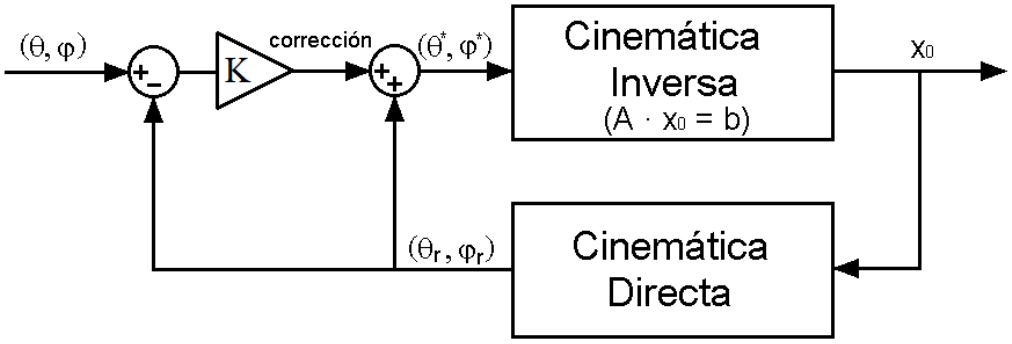


Figura 40: método iterativo para el cálculo de la cinemática de sección.
Fuente: *elaboración propia*.

5.3 Comentario sobre la elección del ángulo α entre discos

Uno de los aspectos más interesantes de este método es el hecho de que **la solución obtenida para la pose de la sección tiene forma de arco de circunferencia cuando los ángulos α de los discos siguen un patrón repetitivo**. Esto es un resultado trivial pero relevante, que se deduce de la propia construcción de la matriz M. Como se ha comentado, si se establece un patrón repetitivo en los ángulos α de los discos, existirán subgrupos de columnas de la matriz M que serán iguales entre sí. De este modo, cuando resuelva 4.4, el vector solución de mínima norma β_0 sufrirá el mismo fenómeno que la matriz M: sus elementos se repetirán según el mismo patrón.

A modo de ejemplo, si se elige un patrón repetitivo en el que todos los discos estén montados a 90º, la matriz M resulta:

$$\begin{bmatrix} cos(\alpha_1) & cos(\alpha_2) & \cdots & cos(\alpha_n) \\ sen(\alpha_1) & sen(\alpha)_2 & \cdots & sen(\alpha_n) \end{bmatrix} =$$

$$= \begin{bmatrix} cos(0) & cos(90) & \cos(0) \ldots & cos(90) \\ sen(0) & sen(90) & sen(0) \ldots & sen(90) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \ldots & 0 \\ 0 & 1 & 0 \ldots & 1 \end{bmatrix}$$

Por su parte, el vector **c** tendrá dos componentes c_1 y c_2, las cuales no tienen por qué ser iguales entre sí. Al realizar el producto $\boldsymbol{\beta_0} = \boldsymbol{M}^T \cdot \boldsymbol{c}$, es trivial percatarse de que las componentes de β_0 seguirán el mismo patrón repetitivo que la matriz M^T.

Al elegir una distribución de ángulos α que siga un patrón repetitivo, **se generan subgrupos de discos dentro de una misma sección que siguen la misma geometría**, lo que resulta en un comportamiento y movimiento iguales en todos ellos. La ventaja que tiene esto frente a un montaje aleatorio, es que como todos los subgrupos se comportan de la misma manera, se minimiza la diferencia en el ángulo que gira cada subgrupo con respecto al ángulo que giran los otros subgrupos, haciendo que la distribución de tensión entre los discos sea más parecida – *nunca será igual debido al rozamiento entre las diferentes partes mecánicas que componen la sección-.*

Haciendo que el comportamiento de la sección se aproxime a la *Hipótesis de Curvatura Constante* se logra **que la pose obtenida en la sección se aproxime más a un arco de circunferencia** y, en consecuencia, a que la curva de *backbone* del robot tenga una **expresión analítica conocida**. Esto es especialmente interesante a la hora de desarrollar métodos más eficientes para el cálculo de la cinemática de la sección. Mientras que el presente método depende de iteraciones para conseguir la pose de cada uno de los discos de la sección, resulta mucho más ágil obtener los parámetros del arco que describe la sección, ignorando el ángulo que gira cada disco.

Estos métodos no son nuevos, sino que son la base de toda la cinemática desarrollada hasta la fecha sobre este tipo de robots. La *Hipótesis de Curvatura Constante* fue propuesta por Hannan y Walker en (12), donde abordaban el diseño de un robot trompa de elefante. La ventaja de este método reside en la simplificación de un elemento complejo como es una sección compuesta por varios discos, a la forma de un simple arco de circunferencia cuya expresión analítica es fácilmente calculable a partir de datos conocidos – *radio del arco, longitud de la sección y ángulo girado-*. Es precisamente de esta aproximación de la que se deriva el modelo mencionado en el apartado 3.1.3 , el cual consiste en la obtención de una matriz jacobiana para la sección basándose en la suposición de Curvatura Constante.

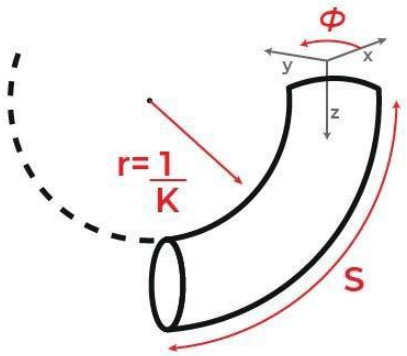


Figura 41: modelo de Hannan y Walker. Fuente: (9)

Dado que en el presente trabajo se pretende estudiar y simular la pose real del robot, no se aprovecharán las ventajas computacionales del modelado de las secciones como un simple cilindro curvo. Por el contrario, se pretende estudiar de forma individualizada la pose de cada uno de los discos, motivo por el cual se seguirá utilizando el método anteriormente descrito. No obstante, se obtiene ya una primera para la elección del ángulo de montaje de los discos, condensado a continuación:

> *Es conveniente que el ángulo α de los discos de una sección siga un patrón repetitivo, de modo que se obtengan subconjuntos de discos con la misma distribución dentro de una sección. Con esto se logra que el ángulo girado por cada subconjunto sea igual o similar al que giran los otros subconjuntos, acercándose así la pose adoptada por la sección a un arco de circunferencia. De este modo, se logra que la geometría de la sección cumpla con mayor fidelidad la Hipótesis de Curvatura Constante (PCC), que permite cálculos de cinemática más rápidos y eficientes.*

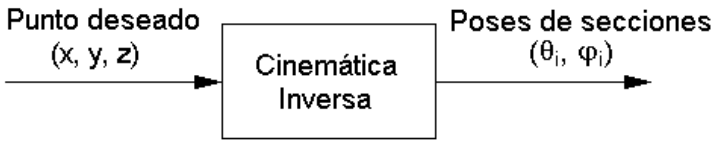
No olvidar que esto sólo pretende la simplificación del cálculo de la cinemática del robot, por lo que el ángulo de montaje óptimo de los discos obtenido por el genético no tiene por qué respetar lo aquí expuesto.

5.4 Control Cinemático mediante Descenso del Gradiente

Una vez se dispone de un método de cálculo de la cinemática de la sección, es posible abordar el control cinemático del conjunto del robot. Sin el método descrito en el apartado anterior no sería posible calcular la pose adoptada por la sección, siendo imposible determinar la pose del robot completo. En este apartado se describirá uno de los métodos de cálculo que se ha empleado: el algoritmo del Descenso del Gradiente.

Como se describe en el apartado 3.2.1, el Descenso del Gradiente consiste en minimizar una determinada función de coste aproximándose a un mínimo local por la dirección que marca el gradiente de esta. En este caso, lo que se busca es un método que consiga posicionar el extremo del robot en un determinado punto con el mínimo error de posición posible. Es decir, que **la función de coste a minimizar es el error de posición del extremo del robot con respecto al punto deseado**.

En este caso, **las variables de control que definen el problema son los parámetros θ y φ de las secciones**, pues son los que describen la posición de estas. El objetivo será calcular estos parámetros, de manera que la pose final posicione el extremo del robot lo más cerca posible del punto objetivo.



1. *Se parte de una pose del robot, dada por los parámetros (θ_i, φ_i) de sus secciones, y se pretende llegar a un punto (x, y, z). Calcular la cinemática inversa de cada sección para conseguir los ángulos β_i iniciales de los discos.*
2. *Con los ángulos β_i, calcular la Matriz de Transformación Homogénea del extremo del robot mediante la Cinemática Directa. Convertir la M.T.H. obtenida a coordenadas cartesianas para conseguir el punto de partida (x_0, y_0, z_0).*
3. *Calcular el error de posición del punto anterior.*
4. *Introducir un pequeño incremento "h" en cada una de las variables θ y φ de las secciones. Calcular los puntos (x_i, y_i, z_i) en los que se sitúa el extremo al introducir esos pequeños incrementos y calcular el error de posición de cada uno de ellos.*

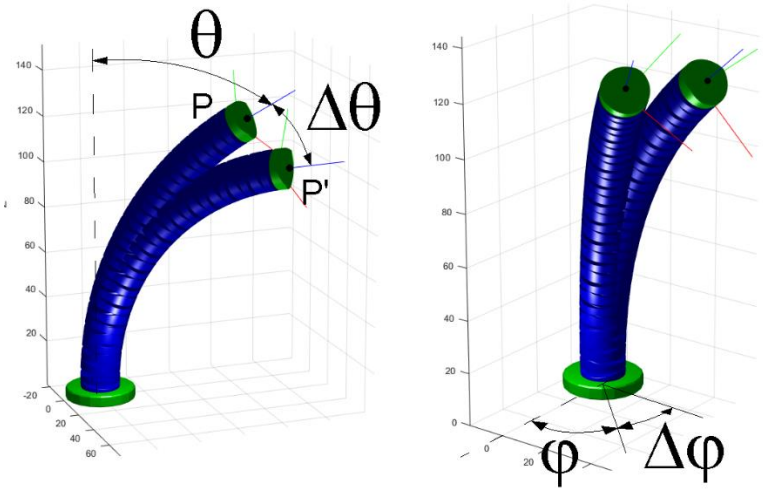


Figura 42: descripción del punto 4 para el caso de una sección. Estas operaciones habría que realizarlas para cada sección del robot, calculando el incremento del error de posición que se produce en cada una. Fuente: *elaboración propia*.

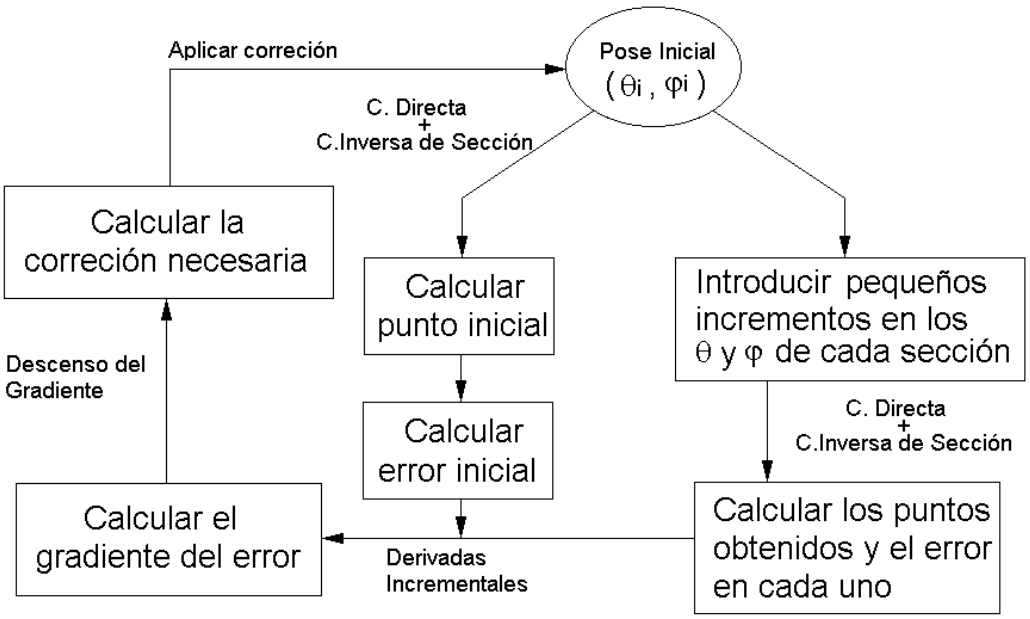
5. *Calcular el gradiente del error de posición mediante Derivadas Finitas según la ecuación 3.5. Observar que la derivada del error tendría la siguiente expresión, en función de la variable que se incremente:*

$$D_\theta e = \frac{de}{d\theta}\Big|_{P(x,yz)} \approx \frac{e_{P'} - e_P}{\Delta\theta}, \quad D_\varphi e = \frac{de}{d\varphi}\Big|_{P(x,yz)} \approx \frac{e_{P''} - e_P}{\Delta\varphi}, \quad \boldsymbol{\nabla e} = \begin{bmatrix} D_{\theta 1} e \\ \vdots \\ D_{\theta N} e \\ D_{\varphi 1} e \\ \vdots \\ D_{\varphi N} e \end{bmatrix} \quad (4.5)$$

6. *Corregir los ángulos (θ, φ) según la ecuación 3.11.*

$$\begin{bmatrix} \theta_1 \\ \vdots \\ \theta_N \\ \varphi_1 \\ \vdots \\ \varphi_N \end{bmatrix}_{i+1} = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_N \\ \varphi_1 \\ \vdots \\ \varphi_N \end{bmatrix}_{i} - k \cdot \boldsymbol{\nabla} e \quad (4.6)$$

7. *Iterar hasta que la derivada se acerque a ser nula o bien hasta conseguir un determinado valor de error admisible.*



5.5 Control Cinemático por el Método de la Jacobiana

Una segunda forma de posicionar el robot es utilizar el método de la Matriz Jacobiana, que se basa en el cálculo de dicha matriz para resolver la Cinemática Inversa. Como ya se mencionó en el apartado 3.1.3 la jacobiana se construye con las derivadas parciales de la función a la que está asociada. En este caso, la función sería la posición del extremo del robot. El método de cálculo de las componentes de la jacobiana consiste en introducir pequeños incrementos en las variables de control y calcular la derivada incremental de la posición. El objetivo será: **dado el vector de *"incremento de posición y orientación deseado"*, obtener el vector *"incremento de variables de control necesario"* resolviendo un sistema de ecuaciones con la jacobiana.**

Existen dos posibilidades de obtención de una jacobiana en función de las variables de control elegidas:

1) *Utilizar como variables los ángulos β de los n discos del robot:*

$$\begin{bmatrix}\boldsymbol{\Delta p}\\ \boldsymbol{\Delta r}\end{bmatrix} = \begin{bmatrix}\Delta x\\ \Delta y\\ \Delta z\\ \Delta\varphi\\ \Delta\theta\\ \Delta\psi\end{bmatrix} = \begin{bmatrix}\frac{dx}{d\beta_1} & \frac{dx}{d\beta_2} & \cdots & \frac{dx}{d\beta_n}\\ \frac{dy}{d\beta_1} & \frac{dy}{d\beta_2} & \cdots & \frac{dy}{d\beta_n}\\ \vdots & \vdots & \ddots & \vdots\\ \frac{d\psi}{d\beta_1} & \frac{d\psi}{d\beta_2} & \cdots & \frac{d\psi}{d\beta_n}\end{bmatrix}_{P=(Xo,Yo,Zo)} \cdot \begin{bmatrix}\Delta\beta_1\\ \Delta\beta_2\\ \cdots\\ \Delta\beta_N\end{bmatrix} = \begin{bmatrix}\boldsymbol{J_p}\\ \boldsymbol{J_w}\end{bmatrix} \cdot \boldsymbol{\Delta q} \qquad (4.6)$$

2) *Utilizar como variables los parámetros θ y φ de las N secciones del robot:*

$$\begin{bmatrix}\boldsymbol{\Delta p}\\ \boldsymbol{\Delta w}\end{bmatrix} = \begin{bmatrix}\Delta x\\ \Delta y\\ \Delta z\\ \Delta\varphi\\ \Delta\theta\\ \Delta\psi\end{bmatrix} = \begin{bmatrix}\frac{dx}{d\theta_1} & \cdots & \frac{dx}{d\theta_N} & \frac{dx}{d\varphi_1} & \cdots & \frac{dx}{d\varphi_N}\\ \frac{dy}{d\theta_1} & \cdots & \frac{dy}{d\theta_N} & \frac{dy}{d\varphi_1} & \cdots & \frac{dy}{d\varphi_N}\\ \vdots & \cdots & \vdots & \vdots & \ddots & \vdots\\ \frac{d\psi}{d\theta_1} & \cdots & \frac{d\psi}{d\theta_N} & \frac{d\psi}{d\varphi_1} & \cdots & \frac{d\psi}{d\varphi_N}\end{bmatrix}_{P=(Xo,Yo,Zo)} \cdot \begin{bmatrix}\Delta\theta_1\\ \vdots\\ \Delta\theta_N\\ \Delta\varphi_1\\ \vdots\\ \Delta\varphi_N\end{bmatrix} = \begin{bmatrix}\boldsymbol{J_p}\\ \boldsymbol{J_w}\end{bmatrix} \cdot \boldsymbol{\Delta q} \qquad (4.7)$$

El vector **Δp** es fácil de obtener, siendo igual a la resta entre las coordenadas del punto objetivo y las del punto de partida. Lo mismo para **Δw**.

$$\Delta\boldsymbol{p} = \begin{bmatrix}x_f - x_0\\ y_f - y_0\\ z_f - z_0\end{bmatrix} = \begin{bmatrix}\Delta x\\ \Delta y\\ \Delta z\end{bmatrix}, \Delta\boldsymbol{w} = \begin{bmatrix}\varphi_f - \varphi_0\\ \theta_f - \theta_0\\ \psi_f - \psi_0\end{bmatrix} = \begin{bmatrix}\Delta\varphi\\ \Delta\theta\\ \Delta\psi\end{bmatrix} \rightarrow \begin{bmatrix}\boldsymbol{\Delta p}\\ \boldsymbol{\Delta w}\end{bmatrix} = \begin{bmatrix}\Delta x\\ \Delta y\\ \Delta z\\ \Delta\varphi\\ \Delta\theta\\ \Delta\psi\end{bmatrix} \qquad (4.8)$$

En ambos casos la solución que se halle alcanzará el punto deseado, cometiendo un error tanto menor cuanto más cercanos sean los puntos inicial y objetivo. Este error se debe a que la matriz jacobiana está particularizada en el punto de partida, y va cambiando a lo largo de la trayectoria de aproximación hacia el punto objetivo. La solución a este problema es realizar el cálculo de la cinemática inversa entre dos puntos dividiendo la trayectoria entre ellos en más puntos cercanos. De este modo, el cálculo se hace entre puntos muy cercanos entre sí, desde el primero hasta el último. Esto minimiza en gran medida el error cometido por el método.

La diferencia fundamental entre elegir unas variables u otras reside en que en el caso 1 se está asumiendo que se tiene control sobre cada articulación de forma independiente a las demás. Esto sería así de tratarse de un manipulador convencional, pero al tratarse de un manipulador actuado mediante cables, **no se tiene control sobre cada disco sino sobre cada sección**. Por este motivo, la pose que devuelva el método 1 no será la pose real del robot, mientras que la que adopte el método 2 sí.

Así, el método 2 obtendrá una solución cercana a la realidad, respetando el movimiento real de las secciones. Sin embargo, por la naturaleza del presente trabajo, a la hora de realizar las simulaciones es conveniente tener en cuenta la pose de cada disco, pues pretende realizarse un estudio detallado de las posiciones de estos para así poder optimizar su montaje. Esto implica que, tras calcular la cinemática inversa mediante el método 2, además habría que calcular la cinemática inversa de cada sección mediante el método del apartado 4.2 Cinemática de Sección para saber así cuál es la orientación exacta de todos y cada uno de los discos del robot. En otras aplicaciones esto no sería realmente necesario, dado que solo resultaría interesante conocer la forma del arco descrito por cada sección -*la cual podría aproximarse a un cilindro curvo, ignorando que está compuesta por discos* -. Esto reduciría el coste computacional del segundo método en estos casos en los que no es necesario el nivel de detalle que aquí se requiere.

Una forma de optimizar el proceso es usar ambos métodos para el cálculo de la pose del robot: utilizar el primero para obtener una primera aproximación de la pose – *dado que es más rápido*- y finalmente corregir esa pose con el método 2 para que se corresponda con la pose real.

6. EVALUACIÓN DE LA CINEMÁTICA

A continuación, se realizará una serie de pruebas de desempeño del control cinemático implementado., con el objetivo de validar los diferentes métodos y algoritmos implementados. A la hora de evaluar el desempeño del control cinemático:

- Sólo se ha tenido en cuenta posicionamiento en traslación y no en orientación.
- No se han evaluado cuestiones de forma o colisión entre las diferentes partes del robot.
- Se hará un estudio del error de posición cometido en el posicionamiento en distintos puntos.

6.1 Cinemática de la Sección

En primer lugar, se comprobará el funcionamiento de la cinemática de sección utilizando un robot compuesto por una única sección. Se evaluará el error de posición cometido al introducir un punto alcanzable y uno no alcanzable. Recordar que una sección sólo dispone de dos GdL, por lo que el espacio en el que se encuentran contenidos los puntos alcanzables será una superficie. Se ha encontrado -mediante prueba y error- el punto (0.0953 , 52.8570 , 133.6243), el cual forma parte del espacio de trabajo. Las siguientes instrucciones permiten definir el robot, así como evaluarlo en el citado punto.

```
% Creación del robot.
    robot = HRRobot([30], [pi/2]);

% Primer posicionamiento para salir del punto singular:
    [newConfig, error, iter] = robot.move(0, 52.86, 133.62,
    'GradientDescent', 'Error', 0.1);

% Posicionamiento definitivo:
    [newConfig, error, iter] = robot.move(0, 52.86, 133.62,
    'GradientDescent', 'Error', 0.1);

% Simulación:
    figure; robot.plot2('Visuals', 'off'); hold on;
    scatter3(0, 52.86, 133.62, 'or', 'filled');
    figure; robot.plot2();
```

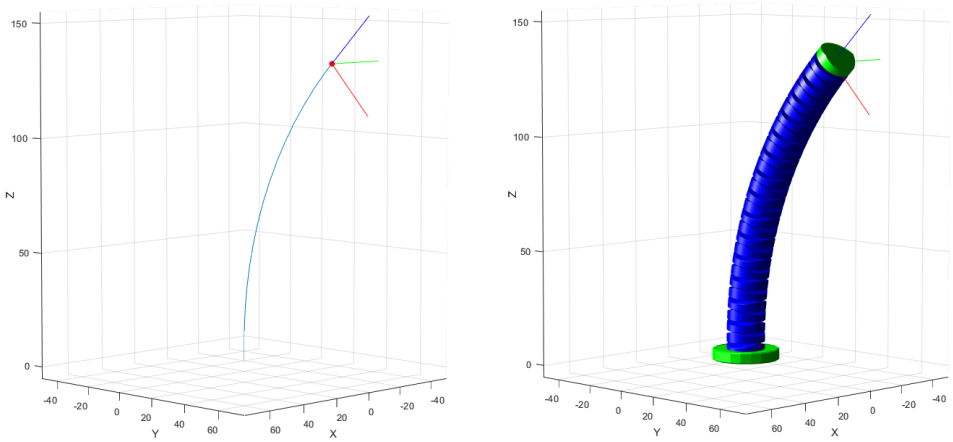


Figura 43: validación de la cinemática de sección. Punto alcanzable.

Ejecutando la instrucción, `robot.currentPos()` es posible visualizar la posición actual del extremo del robot:

```
>> robot.currentPos()

      ans = 0.0953   52.8570  133.6243

>> disp(error)

    {[0.0955]}
```

Como puede observarse, el punto actual es el mismo que el solicitado, con un error inferior a 0.1 unidades, tal y como se estableció en la llamada al método `move`. Tal y como se indicó en el apartado 5.2 Cinemática de Sección, el método obtenido obedece a la hipótesis de curvatura constante, de manera que los discos forman un arco de circunferencia. Esto puede comprobarse tanto en las gráficas obtenidas como visualizando los valores de los ángulos girados por los discos – *ver Figura 33-.*

Estos valores están almacenados en la variable newConfig, la cual guarda el identificador de los discos del robot, así como el ángulo β girado por este. Observar que los ángulos siguen un mismo patrón repetitivo, lo que resulta en un arco de circunferencia.

newConfig | newConfig{1, 1}

newConfig{1, 1}

Fields	JointName	JointPosition
1	"L1.1_Joint"	0
2	"L1.2_Joint"	9.6121e-05
3	"L1.3_Joint"	0.0569
4	"L1.4_Joint"	-9.6121e-05
5	"L1.5_Joint"	-0.0569
6	"L1.6_Joint"	9.6121e-05
7	"L1.7_Joint"	0.0569
8	"L1.8_Joint"	-9.6121e-05
9	"L1.9_Joint"	-0.0569

Figura 44: recorte de los valores de la variable newConfig.

En caso de no ser alcanzable el punto deseado, el robot adoptará la pose que minimice el error de posición entre el extremo y el punto. Por ejemplo, para el punto (100, 100, 13) el robot adopta la siguiente pose:

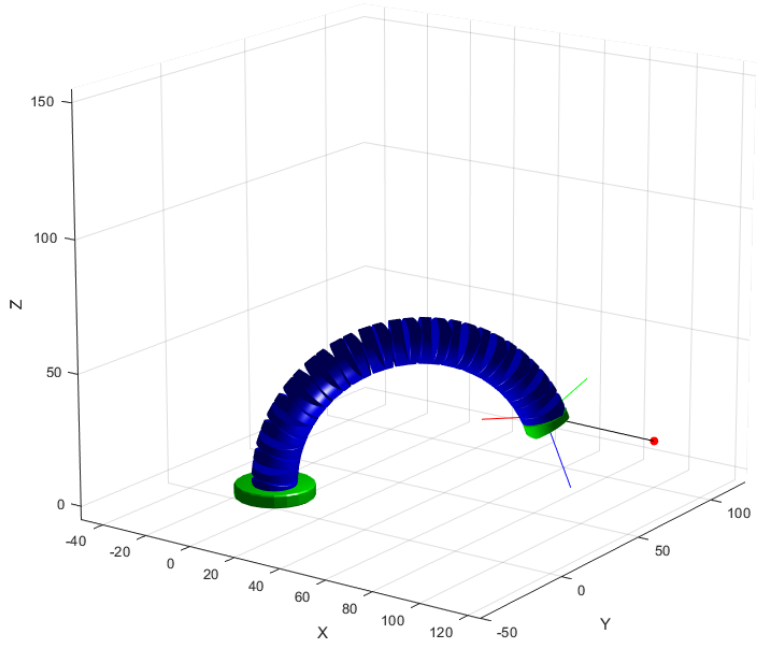


Figura 45: Validación de la cinemática de sección. Punto no alcanzable.

6.2 Cinemática Total: comparativa

Una vez validado el funcionamiento de la sección, se definirá un robot de cuatro secciones y se comparará el posicionamiento conseguido utilizando los diferentes métodos descritos en el apartado 5. Se comenzará definiendo un robot con la siguiente configuración:

```
robot = HRRobot([30 18 12 7], [2*pi/3 pi/2 pi/2 pi/4]);

[~, error, iter] = robot.move(-25, 70, 150, 'GradientDescent',
'Error', 0.1);

robot.plot2();
```

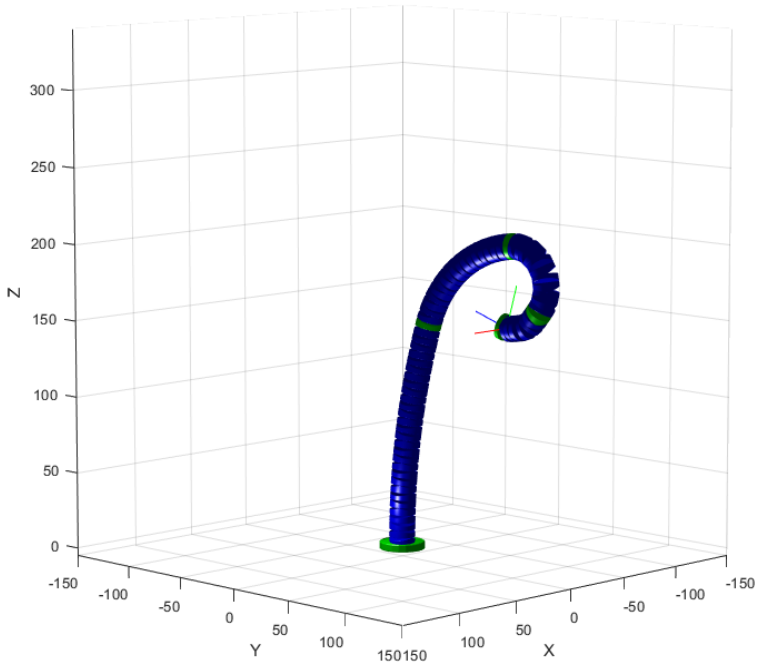


Figura 46: posicionamiento por el Descenso del Gradiente.

Al comprobar la posición, error cometido e iteraciones realizadas, se obtiene lo siguiente:

```
>> disp(error);

    {[0.0978]}

>> disp(iter);

    {[21]}

>> robot.currentPos()

ans = -24.9841   69.9289  149.9348
```

Recordar que el punto deseado es el (-25, 70, 150), resultando en el correcto posicionamiento del robot, consiguiendo un error inferior al deseado. Para aproximarse a este punto, el robot parte desde la posición de máxima extensión, lejos del punto objetivo. Dado que el Descenso del Gradiente se aproxima progresivamente hacia el punto deseado, conseguirá posicionar el robot con éxito sea cual sea la distancia entre el punto de partida y el punto objetivo.

Esto no ocurre con los métodos de la jacobiana, que cometen un gran error al calcular la jacobiana desde el punto de máxima extensión. Este punto es uno de los puntos singulares, y el cálculo de derivadas en él no es preciso. De ahí que estos métodos no arrojen buenos resultados. La pose obtenida al ejecutar el mismo código, pero utilizando el método de la jacobiana es la siguiente:

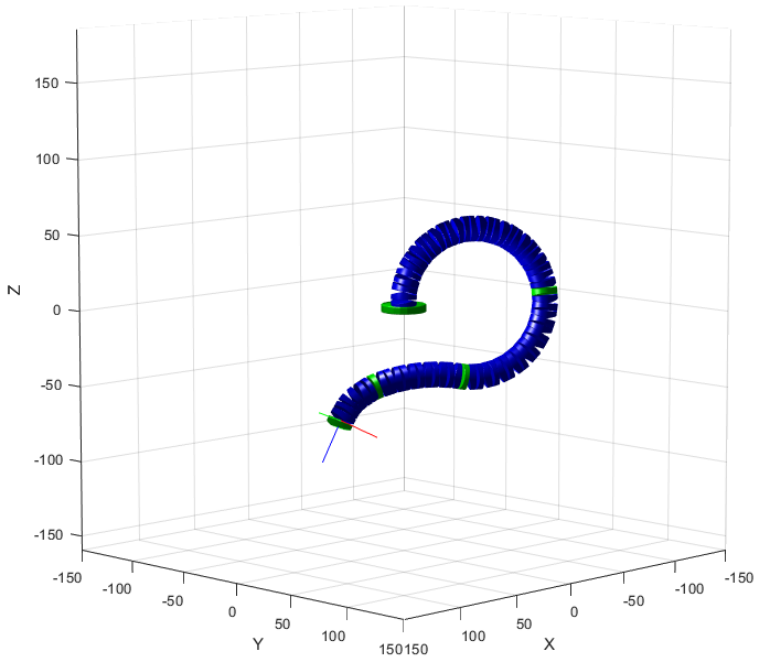


Figura 47:pose fallida.

Evidentemente, el punto no ha sido alcanzado correctamente. Como se ha mencionado, esto se debe al error cometido al calcular la jacobiana desde una singularidad. Existen dos posibles soluciones a esto:

- Utilizar primero el método del gradiente y corregir la pose obtenida mediante el método de la jacobiana.
- Aproximarse desde un punto de partida distinto.

La primera solución es sencilla de comprobar partiendo de la primera posición calculada. En la siguiente figura37 se muestra la pose de partida *-izquierda-* y la pose conseguida al ejecutar el método de la jacobiana *-derecha-*. Observar que la pose resultante del método de la jacobiana es más uniforme, describiendo una curva más continua. Las instrucciones ejecutadas han sido las siguientes:

```
robot.resetConfig()

[newConfig, error, iter] = robot.move(-25, 70, 150,
'GradientDescent', 'Error', 0.1);

figure; robot.plot2('BodyColor', 'r');

robot.resetConfig()

[newConfig, error, iter] = robot.move(-25, 70, 150, 'Default',
'Error', 0.1);

figure; robot.plot2('BodyColor', 'b');
```

El error final cometido es de 0.0719, ligeramente inferior al anterior.

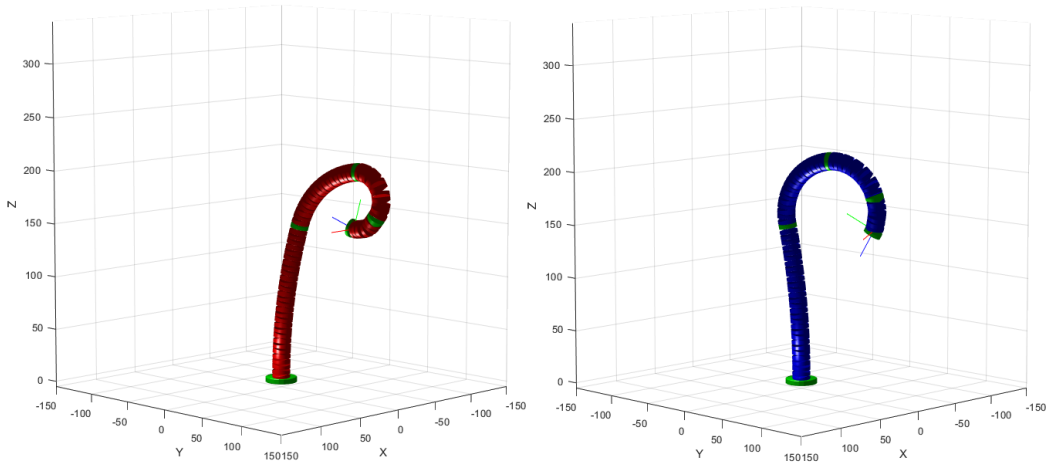


Figura 48: pose obtenida con el Descenso del Gradiente *-izquierda-* y tras la corrección por la jacobiana *-derecha-.*

Esto es una estrategia típica para el control cinemático, obteniendo una pose inicial utilizando métodos más rápidos y/o robustos, y corrigiendo la pose obtenida mediante métodos de corrección de forma o que arrojen resultados mejores según criterios cinemáticos o geométricos. Así, pueden corregirse formas demasiado intrincadas, posibles colisiones entre partes del robot o poses con actuadores cercanos a sus límites de movimiento.

La segunda estrategia tiene que ver con la aproximación hacia el punto desde otro con una mejor disposición. Así, se evita el error provocado por el punto singular. No hay un consenso sobre la elección del punto de partida para este propósito. Según (4) algunos autores siguen criterios como la maximización de la manipulabilidad en el punto de partida. En definitiva, se trata de elegir un punto que minimice el error cometido en los cálculos.

Por último, cabe mencionar la aproximación a puntos lejanos a través de trayectorias. Al desplazar el robot entre puntos lejanos entre sí, la precisión de los cálculos disminuye. La solución a este problema es el cálculo de una trayectoria de aproximación dividida en puntos cercanos entre sí. Esto hace que el error cometido durante el desplazamiento sea menor, logrando alcanzar puntos más lejanos. La trayectoria puede ser recta, curva, o incluso obedecer a criterios como la maximización de la manipulabilidad a lo largo del recorrido. En la siguiente imagen se muestra un ejemplo esto.

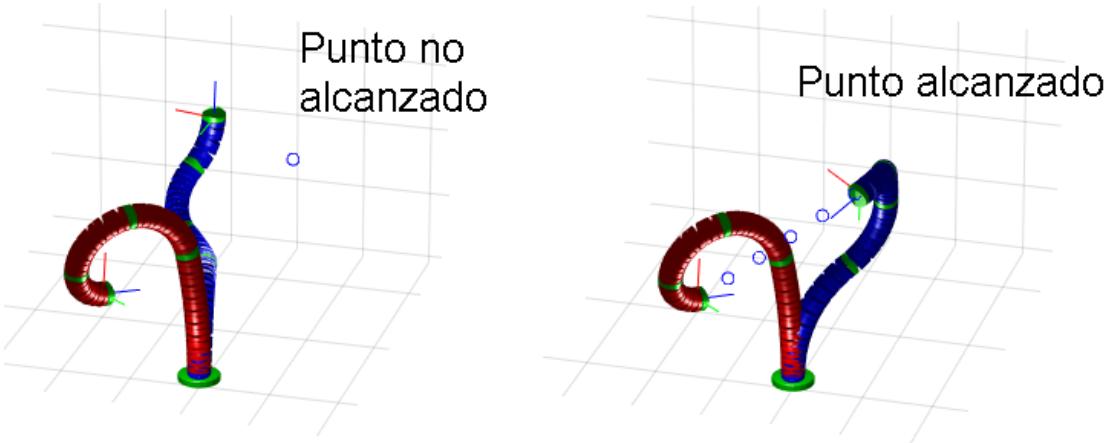


Figura 49: el punto no se alcanza al desplazarse hacia él directamente *-izquierda-*. Al dividir la trayectoria en 6 puntos, se consigue posicionar el robot con éxito *-derecha-.*

7. OPTIMIZACIÓN DE CDHRM MEDIANTE ALGORITMO GENÉTICO

Todo lo anteriormente expuesto sienta las bases necesarias para abordar la optimización de un robot de tipo CDHRR de Discos Pivotantes. Como se mencionó en los primeros capítulos, esta labor viene en parte motivada por la construcción del Pylori I, con la intención de resolver algunas de las incógnitas de diseño que han surgido. Pese a esto, el trabajo realizado en el presente TFG tiene carácter general, pretendiendo estudiar en profundidad estos robots en conjunto y no sólo el citado caso particular.

El trabajo llevado a cabo ha supuesto un desarrollo previo intenso hasta llegar a este punto. Por este motivo, se han centrado esfuerzos en optimizar uno de los parámetros más importantes: la longitud de las secciones. En definitiva, se pretende obtener soluciones óptimas a la hora de distribuir la longitud de estas secciones dependiendo de la función de coste elegida. Las funciones de coste que se han tenido en cuenta en el presente trabajo tienen que ver con aspectos de desempeño cinemático. Basándose en (4), los índices de desempeño elegidos han sido: *Manipulabilidad de Yoshikawa* (13)*, Índice Global de Manipulabilidad* y *Volumen de Trabajo.*

El Índice de Manipulabilidad de Yoshikawa aporta información sobre el desempeño dinámico que tiene el robot en un determinado punto. En líneas generales, este índice estima cuán ágil será el robot en función del número de condición de su matriz jacobiana, particularizada en dicho punto. Así, cuanto mayor sea el parámetro de Manipulabilidad, mayor será la destreza del robot en el punto dado.

$$w = \sqrt{\det(\boldsymbol{J} \cdot \boldsymbol{J}^{\boldsymbol{T}})} \tag{6.1}$$

Sin duda este parámetro es uno de los más relevantes para el objetivo de este trabajo, dada la rapidez del cálculo de su valor y la relevancia de la información que aporta. Resulta muy sencillo evaluar esta característica del robot en varios puntos, pudiendo obtener una noción de la capacidad de movimiento del robot en una zona determinada del espacio de trabajo. Nace así el concepto de Índice Global de Manipulabilidad, que es , en definitiva, la media de los diferentes valores de Manipulabilidad obtenida en varios puntos:

$$GMI = \frac{\sum_1^N w_i}{N} \tag{6.2}$$

Por último, se ha utilizado el Volumen de Trabajo del manipulador en el espacio de trabajo. Sobre este aspecto ya se ha hecho una serie de consideraciones previas en el apartado 4.3.4 Volumen de Trabajo. En resumen, este parámetro permite calcular el tamaño del espacio útil del robot, sin duda otro de los aspectos claves a la hora de diseñar cualquier tipo de robot manipulador.

7.1 Configuraciones Previas

Los índices de desempeño son las fuentes de información disponibles acerca del comportamiento del robot en sus diferentes configuraciones. Pretende encontrarse una o varias configuraciones que maximicen estos parámetros, consiguiendo así un comportamiento óptimo del robot. Como se ha mencionado en el presente capítulo, este estudio se ha centrado en la distribución de los discos de las secciones. Resulta complejo decidir cuántos discos ha de tener cada sección, pues en función de la longitud de cada una de ellas el espacio de trabajo obtenido será completamente diferente.

El diseño de los robots hiperredundantes suele estar inspirado en la anatomía de algunos animales. Ya se han mencionado en el capítulo 2.2 Antecedentes varios ejemplos, como el Mach I, inspirado en una trompa de elefante, u otros diseños que pretenden asemejarse al cuerpo de reptiles como las serpientes. Esta inspiración en el mundo natural tiene un motivo que enlaza con los algoritmos genéticos, pues la naturaleza, en definitiva, es el más complejo de los algoritmos evolutivos. No resulta descabellado, por tanto, buscar esas configuraciones que la naturaleza ha encontrado para tareas similares a la que pretende abordarse.

Es común encontrar en el mundo animal motivos geométricos que obedecen a patrones tales como la Sucesión de Fibonacci o la Proporción Áurea. Por este motivo, se realizará un estudio previo de algunas de estas posibles soluciones, analizando sus índices de desempeño para posteriormente compararlos con los resultados arrojados por el genético.

Como consideraciones previas:

- Se estudiarán robots de 100 discos todos ellos, de modo que todos ellos tengan la misma longitud.

- Todos los robots estarán divididos en 5 secciones. Así, se podrá observar claramente si existe algún tipo de patrón en la distribución de sus longitudes. No se ha elegido un número mayor para evitar aumentar el tiempo de cálculo.

- Se han establecido ángulos $\alpha = 90^{o}$ en todas las secciones para tener volúmenes de trabajo simétricos.

- Dado que los volúmenes de trabajo son simétricos, sólo se estudiarán los puntos contenidos en la mitad de la sección central de dicho volumen. El volumen de trabajo real resultaría de la revolución de la superficie obtenida en torno al eje Z *-vertical-*. Se reutiliza la figura 28 a modo de ejemplo. La superficie que se va a estudiar es la zona derecha en color:

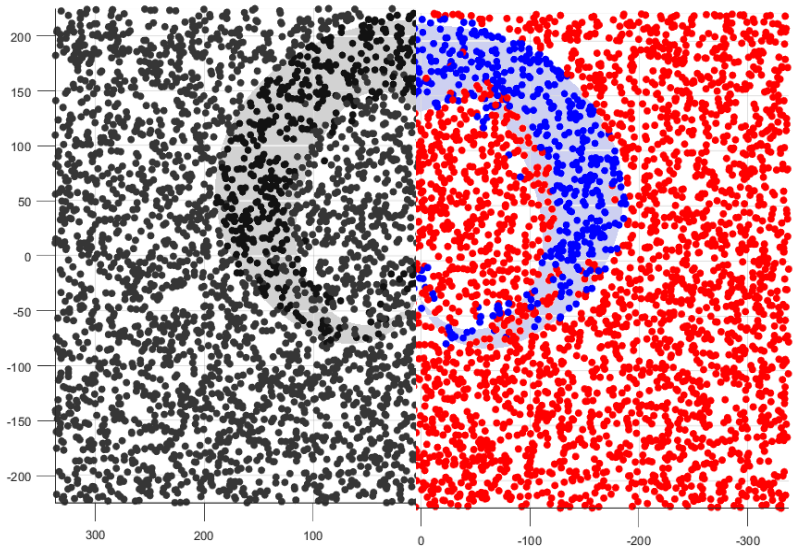


Figura 50: se calcularán los puntos pertenecientes a la mitad derecha de la sección central del volumen de trabajo.

- En caso de que el patrón elegido no permita la obtención de 100 discos exactos en total, se harán los redondeos pertinentes de manera que el patrón resultante se ajuste lo mejor posible al teórico.

7.1.1 Secciones Iguales

Se estudiará el caso en el que las secciones tengan la misma longitud. En este caso, se ha escogido una distribución de 100 / 5 = 20 discos /sección. Se ha simulado un total de 500 puntos, que se corresponden con la **mitad derecha de la sección central del espacio de trabajo**. El volumen de trabajo resultaría de la revolución de las superficies obtenidas, a lo largo del eje vertical izquierdo. A la izquierda, se muestran en azul los puntos a los que el robot ha llegado con éxito, y en rojo, los puntos que no ha sido capaz de alcanzar. A la derecha, se muestran los puntos exitosos, coloreados en función del valor de manipulabilidad del robot – *los más oscuros tienen menor manipulabilidad*- :

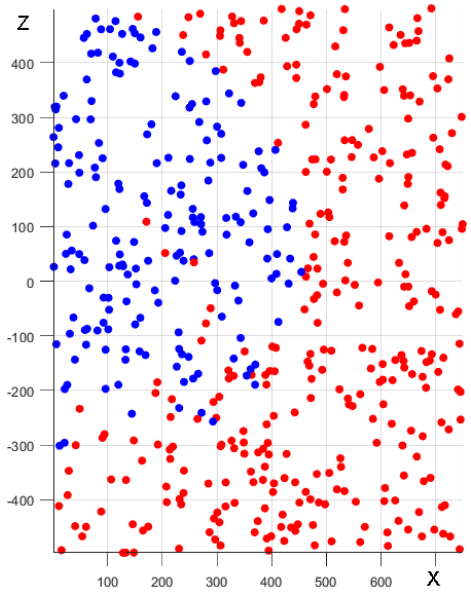


Figura 52: mitad derecha de la sección central del espacio de trabajo.
Caso: secciones de igual longitud.

Azul: puntos exitosos (**181**)
Rojo: puntos fallidos (**319**)

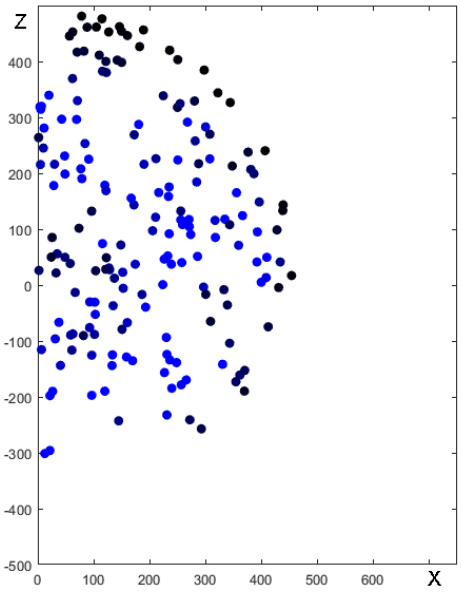


Figura 51: manipulabilidad en los puntos exitosos.
Caso: secciones de igual longitud.

Manipulabilidad máxima: **1,56E+08**
Manipulabilidad mínima: **4,82E+04**
Manipulabilidad media: **1,79E+07**

En la imagen izquierda, se aprecia que esta configuración tiene un desempeño razonablemente bueno en valores negativos de Z: gran número de puntos negativos exitosos, todos ellos con manipulabilidades altas. Observar, en la imagen derecha, cómo los puntos más alejados del robot están más oscurecidos al ser más baja su manipulabilidad. Como comentario adicional, se ha reducido la precisión en las cinemáticas utilizadas para que los tiempos de cálculo no fueran excesivos. Esto ha producido la aparición de puntos atipo en la zona central que deberían ser puntos exitosos pero que el programa no ha sido capaz de calcular correctamente.

7.1.2 Sucesión de Fibonacci

Otro patrón muy común en la naturaleza es la sucesión de Fibonacci, presente en algunas caracolas de crustáceos o en la distribución de pétalos de algunas flores. Para la construcción de un robot que siga esta regla, se ha elaborado una sucesión que sigue la misma regla de cálculo que la sucesión de Fibonacci, pero con valores que consigan un total de 100 discos. Esto sería una distribución de [40 24 16 8 8] discos por sección, siendo 40 los discos de la primera sección y 8 los de la última. Dado que estas cifras no suman 100 discos, se ha modificado la sucesión a [42 25 17 8 8]. Los resultados obtenidos son los siguientes:

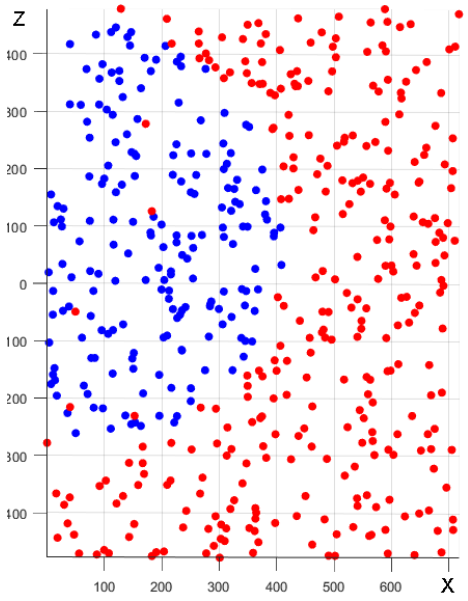


Figura 53: mitad derecha de la sección central del espacio de trabajo.
Caso: Fibonacci (forward).

Azul: puntos exitosos (**192**)
Rojo: puntos fallidos (**308**)

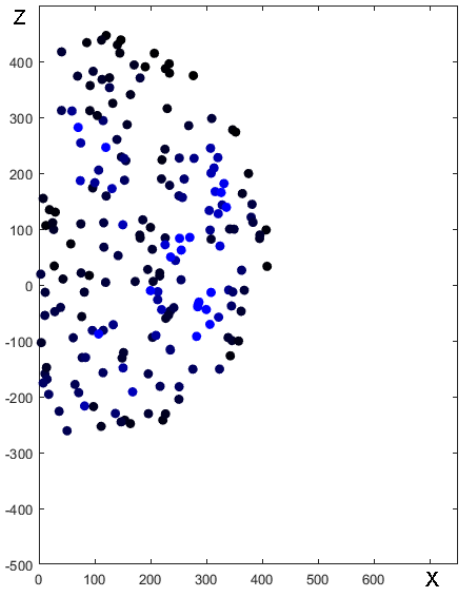


Figura 54: manipulabilidad en los puntos exitosos.
Caso: Fibonacci (forward).

Manipulabilidad máxima: **1,22E+08**
Manipulabilidad mínima: **1,39E+04**
Manipulabilidad media: **8,40E+06**

La superficie obtenida con esta configuración es muy cercana a un semicírculo, lo que hace suponer que este robot se comportará de una forma muy similar en todo el volumen de trabajo. De la imagen izquierda se deduce que también tendrá un buen desempeño para puntos negativos – por debajo de la base del robot- siendo estos en torno al 20% de los puntos exitosos calculados. Además, la forma obtenida es muy regular, lo que reduce el número de puntos singulares y uniformiza la manipulabilidad.

No obstante, la media de la manipulabilidad es menor a la obtenida con la configuración de secciones iguales. Como puede verse en la imagen derecha, los puntos de mayor destreza *-de color azul más intenso-* se sitúan en la zona intermedia entre el eje Z y el borde de la superficie. Esto también ocurría en el caso de secciones iguales, aunque aquí se ve de manera más pronunciada.

Se estudia ahora el caso de la sucesión de Fibonacci, pero a la inversa. Es decir, se ha analizado la distribución [42 25 17 8 8] -*Fibonacci "Forward"*- y se pasa a estudiar la inversa [8 8 17 25 42] – *Fibonacci "Backwards"*-.

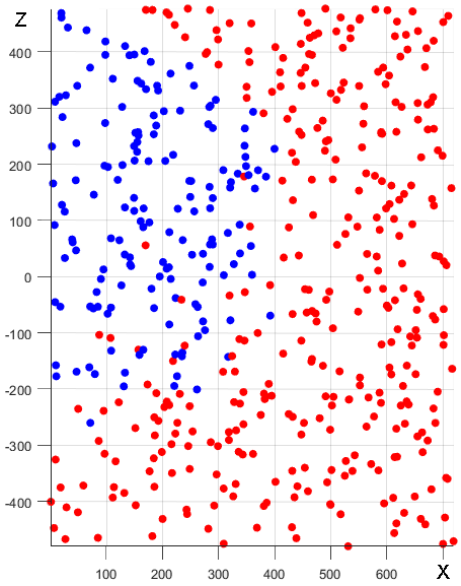


Figura 56: mitad derecha de la sección central del espacio de trabajo.
Caso: Fibonacci (backwards).

Azul: puntos exitosos (**160**)
Rojo: puntos fallidos (**340**)

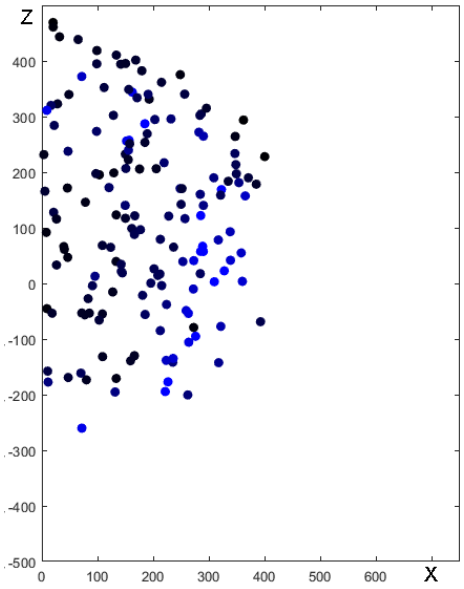


Figura 55: manipulabilidad en los puntos exitosos.
Caso: Fibonacci (backwards).

Manipulabilidad máxima: **6,44E+07**
Manipulabilidad mínima: **3,07E+05**
Manipulabilidad media: **8,33E+06**

Como se puede observar en la imagen izquierda, la forma obtenida resulta un tanto irregular: una zona superior bastante recta que evoluciona a una zona central ligeramente abombada hacia dentro. La parte inferior tiene una gran cantidad de puntos no exitosos infiltrados entre puntos exitosos. El motivo de esto puede deberse a que, dado que las secciones de la base son mucho más cortas que las siguientes, un pequeño incremento en ellas supone un gran cambio de posición en el extremo. Esto provoca que en las zonas cercanas a puntos singulares la precisión en el cálculo se vea reducida.

De la imagen derecha se puede concluir que los puntos de mayor movilidad se encuentran mucho más desplazados hacia el exterior que en los casos anteriores. El núcleo del volumen de trabajo -*zona más cercana al eje Z*- tiene una zona de mala manipulabilidad de mayor tamaño que en los otros casos. Pese a que el valor máximo de manipulabilidad es menor que en el caso de Fibonacci forward, la media se mantiene muy similar. No obstante, Dada la irregularidad de la forma obtenida y el desplazamiento de la zona de mayor destreza hacia el exterior, se puede concluir que la configuración de Fibonacci Forward es mejor que la Fibonacci Backwards en lo que a comportamiento cinemático se refiere.

7.1.3 División por dos

En este caso, se han establecido longitudes de secciones de manera que el número de discos de una sección es igual a la mitad del número de discos que la sección anterior. Al igual que en los casos anteriores, al no ser posible obtener 100 discos en total siguiendo esta norma, se han modificado ligeramente los valores para conseguir el citado número. Es decir, se ha construido un robot cuya distribución es de [50 26 13 7 4].

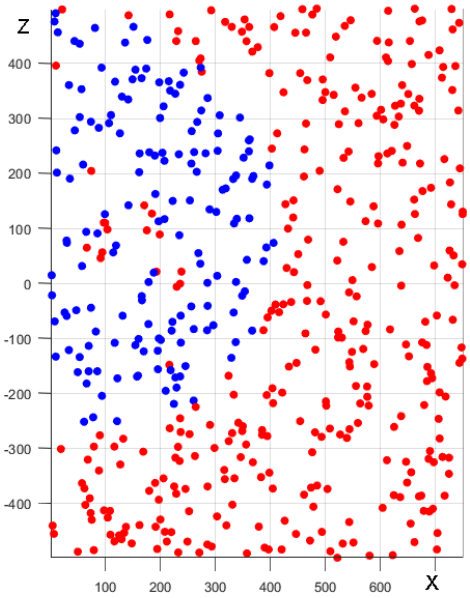


Figura 57: mitad derecha de la sección central del espacio de trabajo.
Caso: División por dos.

Azul: puntos exitosos (**159**)
Rojo: puntos fallidos (**341**)

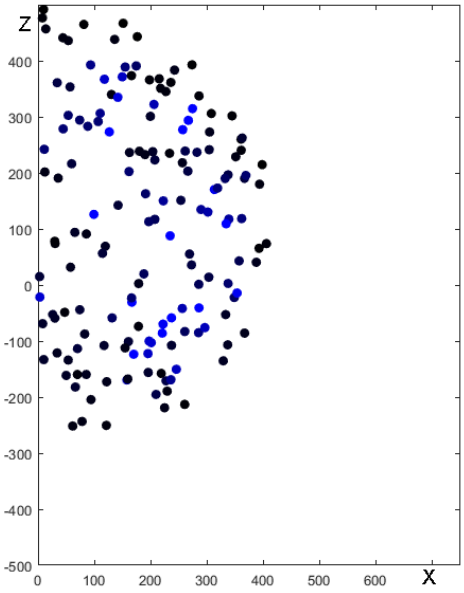


Figura 58: manipulabilidad en los puntos exitosos.
Caso: División por dos.

Manipulabilidad máxima: **2,74E+08**
Manipulabilidad mínima: **5,26E+04**
Manipulabilidad media: **8,52E+06**

La imagen de la izquierda deja entrever cómo en la zona central el robot tiene problemas de movilidad, existiendo una pequeña nube de puntos fallidos. La causa de esto puede ser que la primera sección es demasiado grande en comparación con las demás, lo que impide que el robot alcance fácilmente los puntos más cercanos a la base del robot. Esto no ocurre en el caso de la sucesión d Fibonacci “Forward”, con 42 discos en la primera sección.

Por el gran tamaño de la primera sección, los valores de manipulabilidad más altos se verán desplazados hacia el exterior del volumen de trabajo. Pese a ello, los valores de manipulabilidad son mejores que para el caso de Fibonacci “Forward”, tanto en valor medio como en valor máximo y mínimo. No ocurre lo mismo con la configuración de Secciones Iguales, la cual tiene unos muy buenos valores de manipulabilidad. No obstante, la “zona muerta” en la parte central de la sección pueden suponer un problema a la hora de trazar trayectorias o de trabajar en zonas cercanas a la base del robot.

7.1.4 Relación Áurea

Se ha establecido una distribución de discos que sigue la relación áurea. El número áureo, de valor φ = 1,618… está presente en infinidad de elementos de la naturaleza: la distribución de hojas en las ramas, la distancia entre los nervios de las hojas, diferentes espirales de caracolas de animales o incluso las proporciones del cuerpo humano responden, en mayor o menor medida, a esta relación. Se ha establecido una configuración de discos de [44 26 16 10 6] siguiendo, de manera aproximada, la mencionada relación.

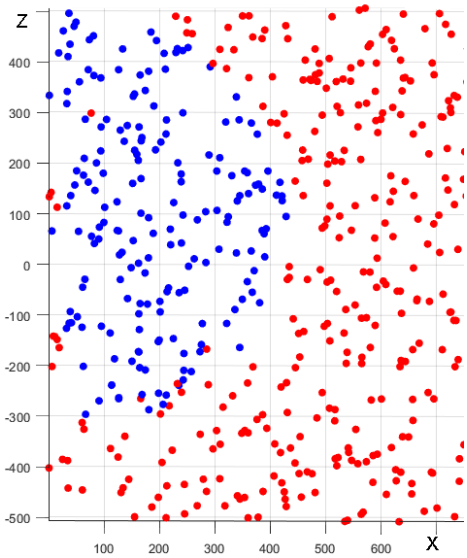


Figura 59: mitad derecha de la sección central del espacio de trabajo.
Caso: Proporción Áurea.

Azul: puntos exitosos (**190**)
Rojo: puntos fallidos (**310**)

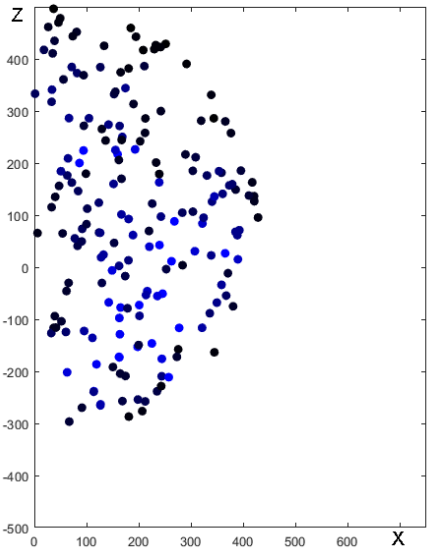


Figura 60: manipulabilidad en los puntos exitosos.
Caso: Proporción Áurea

Manipulabilidad máxima: **1,26E+08**
Manipulabilidad mínima: **3,26E+04**
Manipulabilidad media: **9,54E+06**

De la imagen izquierda, se deduce un buen desempeño del robot en todas las zonas. En los puntos por debajo de la base del robot, es el que mejor se desenvuelve, atendiendo al número de puntos y a los valores altos de manipulabilidad en ellos. Con respecto a la zona superior, hay un volumen de puntos importante, siendo la configuración que mayor volumen de trabajo ha conseguido junto a la de Secciones Iguales. Parece haber algunos puntos conflictivos junto al eje Z, aunque pueden deberse a errores en el cálculo puesto que no son numerosos. Habría que hacer pruebas más exhaustivas para corroborarlo.

Según la imagen de la derecha, esta configuración dota al robot de una buena manipulabilidad en general, con valores muy altos y uniformes en toda el área calculada. La manipulabilidad media es mucho mejor que la de la configuración de Fibonacci “Forward”, lo que, sumado al mayor volumen de trabajo, sentencia que se trata de una configuración mejor que la de la sucesión de Fibonacci.

7.2 Resultados del Algoritmo Genético

Las ejecuciones del algoritmo genético han sido secuenciales, incrementando progresivamente la dificultad de las mismas para poder identificar fácilmente posibles fallos de funcionamiento. Tras algunas pruebas preliminares, los parámetros de diseño que dieron mejores resultados fueron los siguientes:

- Se sigue un método de selección proporcional, teniendo en cuenta el valor de puntuación del robot en relación con la máxima puntuación de la población.
- El mejor individuo siempre pasa a la siguiente generación.
- La probabilidad de producirse una mutación en la sección de un individuo es de un 10%.
- La amplitud de las mutaciones es de ±2 discos por sección.

Las primeras pruebas se realizaron con una función objetivo sencilla: maximizar la manipulabilidad en un determinado punto. A la vista de los espacios de trabajo obtenidos, **las primeras ejecuciones del algoritmo buscaron optimizar la manipulabilidad en la zona central del área de trabajo**, suponiendo que esta sería la zona por la que el robot se movería más frecuentemente para cualquier tarea.

Se eligió un robot de 3 secciones, con un total de 50 discos. Estos números fueron escogidos para simplificar el trabajo del algoritmo y reducir los tiempos de ejecución. El tamaño muestral fue de 50 individuos. El punto objetivo en el que maximizar la manipulabilidad fue el punto (0, 125, 100), escogido según el criterio anteriormente citado.

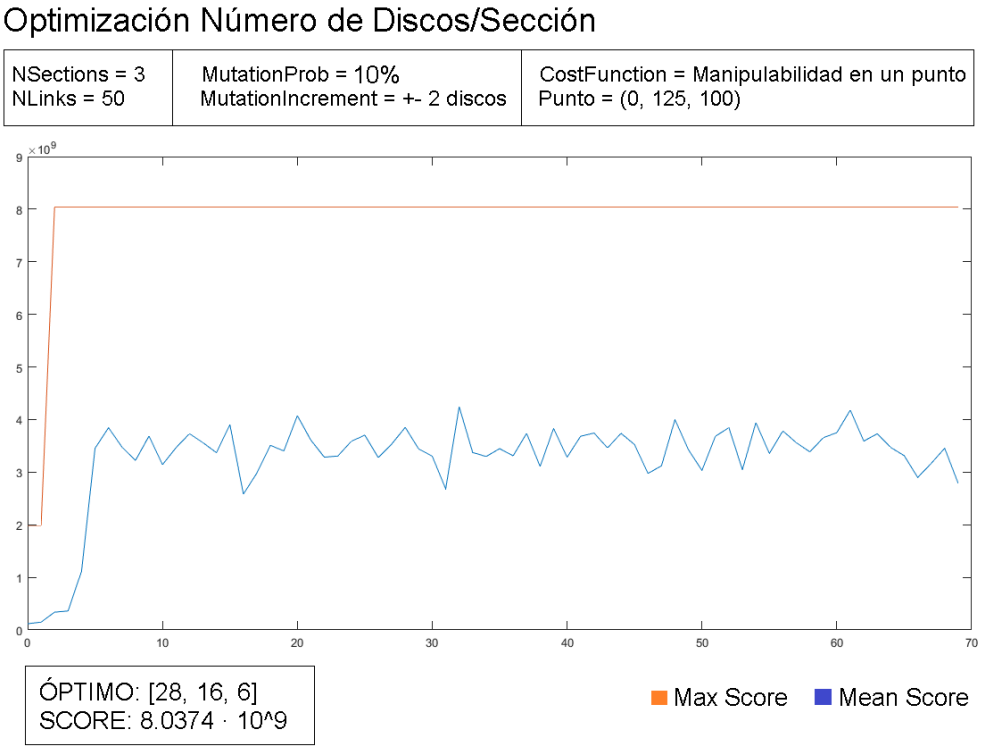


Figura 61: resultados algoritmo genético (3 secciones, punto 0, 125, 100).

De estos resultados pueden sacarse varias conclusiones. La primera tiene que ver con la convergencia del algoritmo. Ya en las primeras ejecuciones se encuentra la solución que será la que mayor manipulabilidad tiene a lo largo de todas las iteraciones. Esto hace que la puntuación media de la población aumente exponencialmente durante las primeras ejecuciones, hasta alcanzar el punto de equilibrio en torno a la décima iteración. Esto puede deberse a un fenómeno denominado *"convergencia prematura"*, que consiste en la convergencia temprana hacia una solución, impidiendo encontrar otras soluciones mejores.

La causa de esto podría ser un **elitismo excesivo** en el algoritmo, que satura la población con individuos de elevada puntuación, forzando al algoritmo a converger hacia esa solución. Sin embargo, dado el elevado número de ejecuciones realizadas, el algoritmo podría haber encontrado otras soluciones fácilmente *-el ruido presente en la media poblacional indica que sigue habiendo variabilidad genética-*, por lo que se ha dado el resultado como válido.

Para corroborar la generalidad de este resultado, se efectuaron pruebas posteriores. Dado que lo que se pretende estudiar no es tanto el número de discos sino la distribución de estos se permitió esta vez que el número total de discos pudiera variar para tener así un criterio de selección menos restrictivo. Los resultados de estas ejecuciones se ponen a disposición del lector en el Anexo 2.

Todas estas pruebas parecen seguir un mismo patrón, siendo claro el **acortamiento progresivo de las secciones desde la base hacia el extremo**. Por la repetición de los resultados para las diferentes ejecuciones, se concluye que la configuración de discos [28, 16, 6] es la que maximiza la manipulabilidad en el punto (0, 125, 100), para el caso de 50 discos. Esto confirma las pruebas realizadas anteriormente, que preveían un buen desempeño de las configuraciones cuya evolución era progresiva – *áurea y Fibonacci-* en torno a la zona central y superior del volumen de trabajo

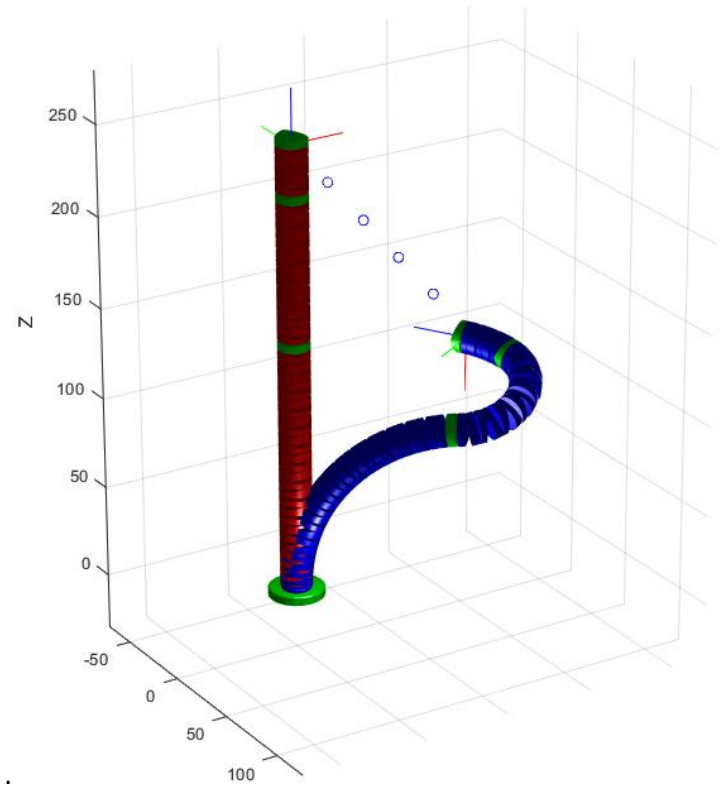


Figura 62: posicionamiento del caso [28, 16, 6] en el punto (0, 125, 100).

Observar que el robot alcanza perfectamente el punto, siendo posible adoptar una gran diversidad de poses para alcanzarlo. Más abajo, se muestra la imagen de la sección del volumen de trabajo para esta configuración. Se ha seguido la misma metodología que en el apartado anterior, calculando sólo la sección que forma uno de los cuartos del volumen de trabajo. Al igual que ocurría en el ejemplo de la Figura 28, los puntos cercanos al eje z no son alcanzables. Existen tres posibles estrategias para atajar esto:

- Aumentar el número total de discos.
- Aumentar el ángulo máximo que pueden pivotar los discos entre sí.
- Reducir las longitudes de las primeras secciones.

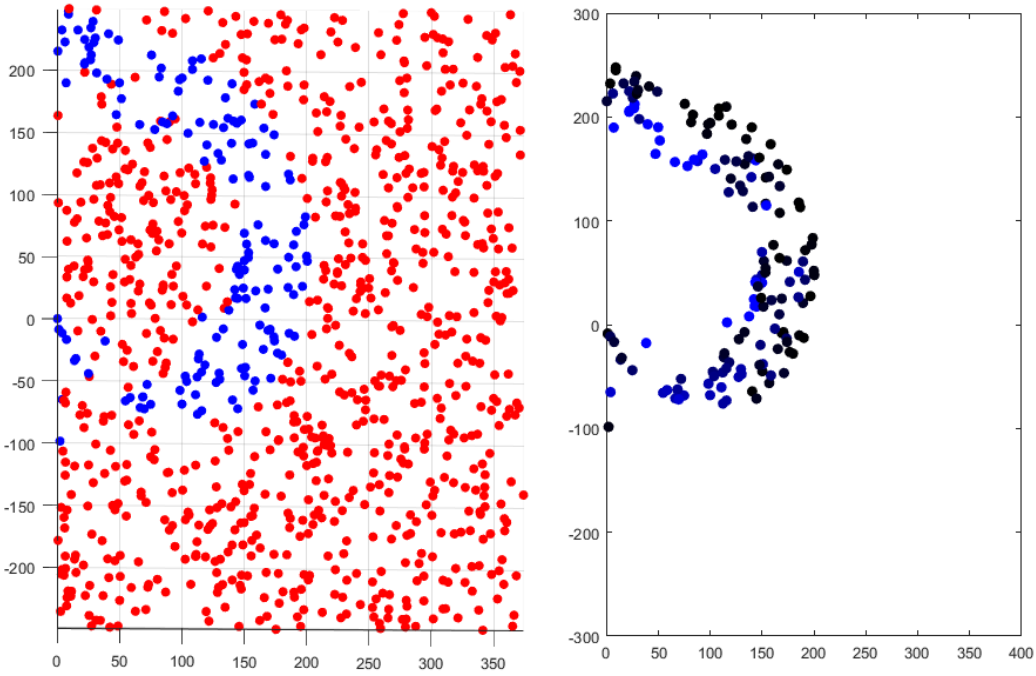


Figura 63: sección del volumen de trabajo para la distribución [28, 16, 6]

Observar que el algoritmo ha llevado a cabo su tarea con éxito, maximizando la manipulabilidad en la zona central y superior. No obstante, como se ha mencionado en el párrafo anterior, existe una gran zona central a la que el robot no es capaz de llegar. Con ánimo de comprobar si este problema se soluciona añadiendo más secciones, se realizó otra ejecución con 4 secciones. La evolución del algoritmo se muestra en la página siguiente.

En la iteración número 16 el algoritmo encuentra una solución óptima cuya distribución de discos es **[19, 12, 10, 9]**. En este caso, la evolución de las longitudes parece seguir una progresión parabólica, reduciéndose progresivamente la diferencia de longitud entre secciones.

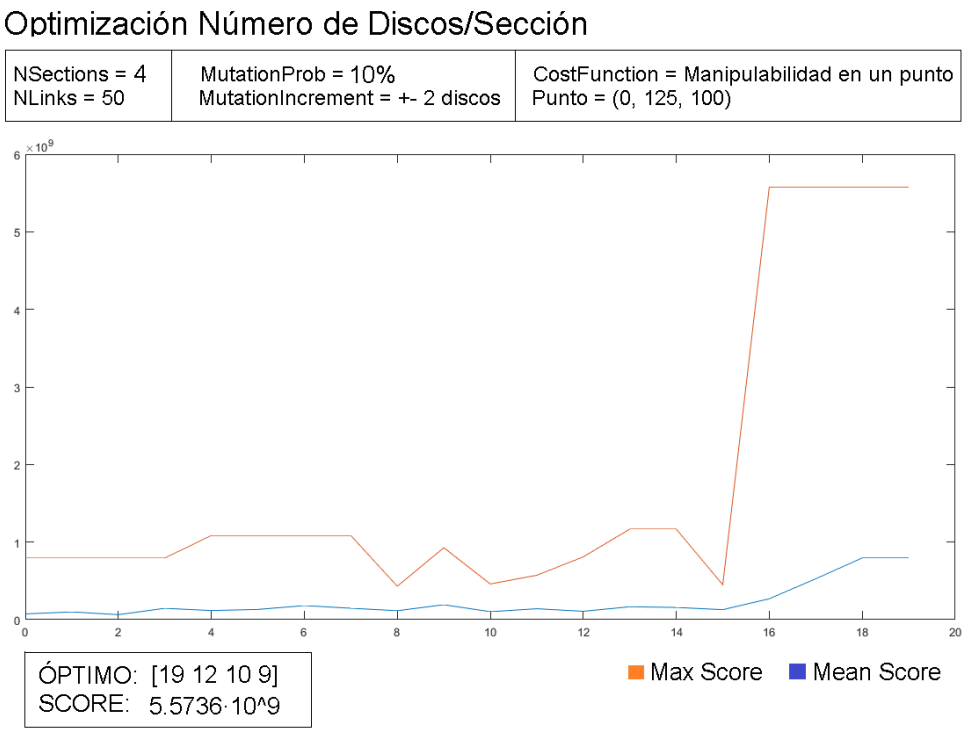


Figura 64: resultados algoritmo genético (4 secciones, punto 0, 125, 100).

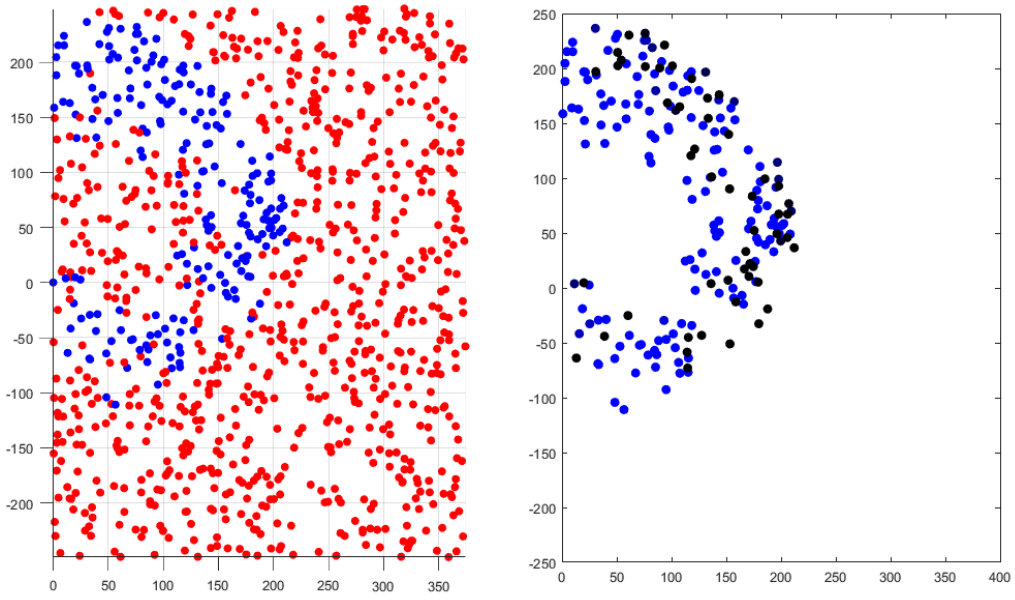


Figura 65: sección del volumen de trabajo para la distribución [19, 12, 10, 9]

De la figura 57 se concluye que ambos objetivos han sido logrados: aumentar el volumen de trabajo y mejorar la manipulabilidad global del robot. Puede concluirse que al aumentar el número de secciones se tienen dos claras ventajas:

- Se aumenta el número de puntos alcanzables, incrementando el volumen de trabajo.
- Se tiene un mayor control sobre la pose del robot, mejorando su manipulabilidad.

8. CONCLUSIONES Y LÍNEAS FUTURAS

De las pruebas realizadas, puede concluirse que las configuraciones que maximizan tanto la manipulabilidad como el volumen de trabajo son aquellas en las que la longitud de las secciones se va acortando desde la base hasta el extremo. Esto queda demostrado en configuraciones como la de Fibonacci o la Progresión Áurea, las cuales consiguen espacios de trabajo muy uniformes y con valores de manipulabilidad razonablemente buenos. Bien es cierto que la configuración de Secciones iguales también ha dado unos muy buenos resultados, pero su desempeño en los puntos singulares del robot *-por ejemplo, en la zona cercana a la base del robot-* no es tan bueno como en los casos anteriores.

Las ejecuciones del algoritmo genético corroboran estos datos. En las pruebas realizadas, se ha tenido como objetivo la maximización de la manipulabilidad en los puntos intermedios. Para ello, se ha elegido un único punto lo suficientemente representativo como para que la manipulabilidad se maximice en toda la zona central y superior del volumen de trabajo. Los resultados obtenidos concluyen que la configuración que consigue este propósito sigue una evolución tal que la **longitud de las secciones decrece al acercarse al extremo** del robot. No obstante, la progresión seguida no parece aproximarse a ninguna de las progresiones estudiadas en el apartado 6.1, sino que **se aproxima a una evolución logarítmica**.

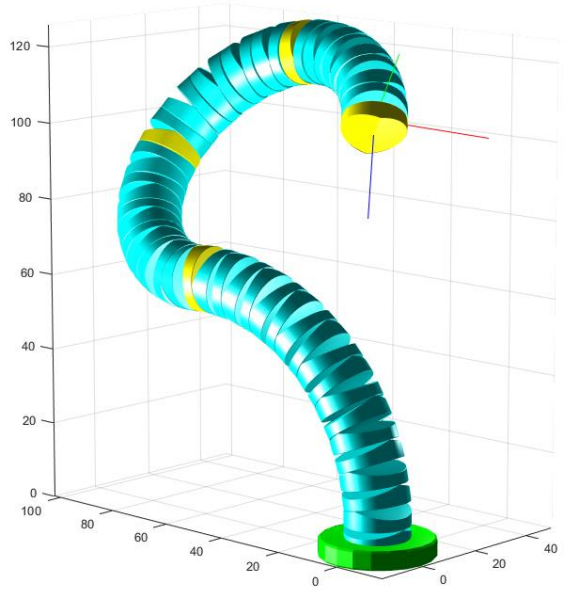


Figura 66: imagen del robot de configuración [19 12 10 9] en una pose intrincada.

También ha podido comprobarse que **un mayor número de secciones favorece la obtención de un volumen de trabajo mayor, así como una manipulabilidad más alta**. Esto se ha visto constatado en el apartado anterior, pues manteniendo el mismo número de discos y aumentando el número de secciones se ha conseguido el citado efecto.

El efecto del aumento del número de secciones va en paralelo con el efecto de la longitud de las primeras secciones. En los primeros casos estudiados, **los robots con una gran longitud en las secciones cercanas a la base tenían más dificultades para alcanzar los puntos más cercanos a la coordenada (0,0,0)**. Todo esto también se ha

visto constatado en las ejecuciones realizadas por el genético, obteniendo un mayor volumen de trabajo en los casos de menor longitud en las secciones cercanas a la base.

Las ejecuciones del algoritmo genético han tenido un gran coste temporal, pues han llevado del orden de las 10 horas de ejecución. Uno de los motivos es el tamaño muestral utilizado – *del orden de 50 individuos*- , pues se ha escogido de manera que el algoritmo obtenga resultados fiables. Sin embargo, la causa principal es el largo tiempo que conlleva el cálculo de la cinemática inversa de la sección. Como se ha explicado en los apartados relativos a este tema, se ha calculado la cinemática teniendo en cuenta la pose de todos y cada uno de los discos del robot. Hubiera sido más interesante implementar el método de Webster y Jones para agilizar estos cálculos, reduciendo los tiempos de cálculo considerablemente.

Agilizando el cálculo de la cinemática, sería posible afrontar ejecuciones del algoritmo genético que tuvieran en cuenta un mayor número de parámetros. Se proponen varios experimentos que arrojarían resultados muy interesantes:

- **Definir como función de coste el volumen de trabajo** . Esto permitiría obtener aquellas configuraciones que consiguen alcanzar un mayor de puntos. Como restricción, habría que limitar el número de discos a un valor determinado – *un mayor número discos, lógicamente, implicará un mayor volumen de trabajo*-.
- **Definir varios puntos de evaluación de la manipulabilidad**, estableciendo como función de coste la media de los valores obtenidos. De este modo, se optimizaría la manipulabilidad no ya en un punto, sino en varios, consiguiendo así resultados más fiables.
- **Combinar ambas funciones de coste**, optimizando así los dos aspectos a la vez.

Por último, se plantea una reflexión sobre las configuraciones previamente analizadas. El uso de un algoritmo genético permite la obtención de configuraciones óptimas en función de distintos aspectos a la vez, siendo posible optimizar en gran medida ciertos aspectos concretos del robot para tareas específicas. No obstante, ha quedado constatado cómo aquellas configuraciones que siguen patrones similares a los presentes en la naturaleza – *Fibonacci y relación Áurea*- dan también muy buenos resultados en general. El uso de estas configuraciones como solución de partida para el algoritmo genético puede ser una muy buena opción, facilitando así la convergencia de este.

Pero más allá de utilizarlos como una mera solución inicial, pueden ser escogidos a la hora de diseñar un robot de estas características. El coste temporal de implementación y ejecución de un algoritmo de inteligencia artificial puede no ser viable en muchos casos, siendo quizá más eficiente escoger patrones como los citados para agilizar la fase de diseño. Pese a ello, de ser necesario un estudio pormenorizado del desempeño del robot en distintos aspectos, siempre estará justificado el uso de inteligencia artificial por el gran valor de los resultados que ofrece.

9. IMPACTO SOCIAL

Como se menciona en el apartado del Estado del Arte, los robots hiperredundantes están siendo utilizados en áreas como la medicina, labores de rescate o inspección. En definitiva, suponen un gran avance para el mundo de la robótica, renovando la visión de la sociedad sobre los robots convencionales. Esa imagen de los robots como una máquina rígida, con movimientos antinaturales y artificiales queda anticuada con el desarrollo de robots como el estudiado en este trabajo, cuyos movimientos y poses son todo lo contrario.

En el mundo de la medicina, estos robots tendrán un fuerte impacto, mejorando intervenciones quirúrgicas invasivas al conseguir un acceso más ágil y preciso a las zonas dañadas. Esto reduce en gran medida los tiempos de recuperación del paciente, lo cual es vital en muchos casos. Por otro lado, al reducir el impacto de la intervención, la tasa de éxito se verá aumentada significativamente.

Con respecto a las labores de rescate, son muchas las regiones del mundo en las que las catástrofes naturales hacen estragos cada año. Las pérdidas de vidas humanas pueden llegar, en muchos casos, a alcanzar los miles de personas. Las labores de los equipos de rescate en estas situaciones están condicionadas por la orografía del terreno, así como por la gravedad de los desperfectos causados. Estos robots pueden ayudar en gran medida al reconocimiento de zonas de difícil acceso, así como a la inspección visual de zonas derrumbadas en busca de personas sepultadas. Facilitar el trabajo de los equipos de rescate es, sin duda alguna, un punto muy a tener en cuenta, ya sea aportando visión en zonas inaccesibles, llevando víveres a personas atrapadas o incluso permitiendo mover y esquivar obstáculos.

Por último, se han mencionado ventajas tales como disponer de un cuerpo más liviano que el de los robots convencionales. Al tener un menor peso y una estructura más blanda, hacen que su uso como robots manipuladores colaborativos sea muy interesante. En muchas áreas industriales, la zona de trabajo de los robots está completamente separada de la zona de trabajo de las personas, siendo habitual restringir el tránsito de los trabajadores por motivos de seguridad. Los robots convencionales disponen de partes móviles muy pesadas, y carcasas metálicas que podrían dañar gravemente a una persona. No ocurre lo mismo con los robots hiperredundantes, que suelen disponer de partes mecánicas más blandas y menos peligrosas. El impacto de esto en la sociedad es claro, pues favorece a acortar esa distancia entre robots y humanos, facilitando así la integración de estas máquinas en el trabajo diario y en la vida cotidiana de las personas.

10. REFERENCIAS

1. **Webster, R. J. y Jones, B. A.** *Design and Kinematic Modeling of a Constant Curvature Continuum Robots: a Review.* 2010.

2. **Martín, A., y otros.** *Robots Hiper-Redundantes: Clasificación, Estado del Arte y Problemática.*

3. **Chirikjian, G. S. y Burdick, J. W.** *A modal approach to hyper-redundant manipulator kinematics.* 1994.

4. **Moreno , Héctor A., y otros.** *índices de Desempeño de Robots Manipuladores: una revisión del Estado del Arte.*

5. **Martín, Cecilia Martinez.** *Desarrollo de un robot manipulador blando e híperredundante.* 2017.

6. **Lewis, M. Anthony, Fagg, Andrew H. y Solidum, Alan.** *Genetic programming approach to the construction of a neural network for control of a walking robot.* 1992.

7. **Universidad Politécnica de Madrid.** *Cálculo I: Introducción al Cálculo de una Variable.* Madrid, . .

8. **Barrientos, A., y otros.** *Fundamentos de Robótica.*

9. **López, Andrea.** *Modelado y Control De Un Robot Blando Actuado Con SMA. Trabajo de Fin de Máster.*

10. **Rudolph, Günter.** *Convergence Analysis of Canonical Genetic Algorithms.* 1994.

11. **Berzal, Fernando.** *Genetic Algorithms.*

12. **Hannan, M. W. y Walker, I. D.** *Novel Kinematics for Continuum Robots.* 2000.

13. **Yoshikawa, Tsuneo.** *Manipulability of Robotic Mechanisms.* 1985.

14. **Cowan, Lara.** *Analysis and Experiments for Tendril-Type Robots.* 2008.

15. **Dreo, J., y otros.** *Paradiseo: From a Modular Framework for Evolutionary Computation to the Automated Design of Metaheuristics.* 2021.

16. **Holland, John H.** *Adaptation in Natural and Artificial Systems.* 1975.

17. **Pamanes, J. Alfonso y Zeghloul, Said.** *Optimal Placement of Robotic Manipulators Using Multiple Kinematic Criteria.* 1991.

18. **Jones, Bryan A.** *A New Approach to Jacobian Formulation for a Class of Multi-Section.*

19. **Ahmad Abu Alqumsan, Suiyang Khoo y Michael Norton.** *Robust control of continuum robots using Cosserat rod theory.*

ANEXO 1: Referencia de Clases

HRRObject

Clase Padre: *"rigidBody"*

Atributos y Métodos:

Públicos

HRRObject(name, varargin): constructor.

HRRBase

Clase Padre: *"HRRObject"*

Atributos y Métodos:

Públicos

H: distancia desde la base hasta la planta del primer disco.

HRRBase(name,varargin): constructor.

HRRLink

Clase Padre: *"HRRObject"*

Atributos y Métodos:

Públicos

H: altura desde la base hasta el punto de pivote.

R: radio hasta la línea media de los orificios para cables.

HRRLink(name, varargin): constructor

HRRSection

Clase Padre: *"rigidBodyTree"*

Atributos y Métodos:

Públicos

Index: índice dentro de un robot.

Nlinks: número de discos.

EndLink: nombre identificador del último disco.

Assembly: configuración del montaje.

Alpha: ángulo alpha de los discos.

A: matriz A para el método de Cinemática de Sección.

HRRSection(Index, NLinks, Alpha, varargin): constructor.

Privados

calculateA(): calcula la matriz A.

nameLink(linkIndex): devuelve el nombre identificador de un disco.

addLink(linkIndex, destination, zDistance): añade un disco a la sección.

createLinks(): genera los discos de la sección.

HRRTree

Clase Padre: *"rigidBodyTree"*

Atributos y Métodos:

Públicos

NSections: número de secciones

Nlinks: vector de número de discos en cada sección.

TotalLinks: número total de discos

Assembly: vector de configuraciones de las seciones.

Alpha: vector de ángulos alfa en cada sección.

EndLink: nombre identificador del disco del extremo.

A: array de matrices A de cada sección.

Config: vector de configuración ("nombre", "beta") de los discos.

HRRTree(NLinks, Alpha): constructor.

sectionIKine(index, coords, type): calcula la cinemática inversa de una sección.

fKine(config): calcula la cinemática directa del robot.

currentPos(): devuelve la posición actual del robot en cartesianas.

iKine(tform, varargin): cinemática inversa del robot usando la Jacobiana General y la Jacobiana HRR.

iKineGradientDescent(tform, varargin): cinemática inversa mediante Descenso del Gradiente.

iKineJacobi(tform, varargin): cinemática inversa mediante la Jacobiana General.

iKineHRR(tform, varargin): cinemática inversa mediante la Jacobiana HRR.

jacobian(varargin): calcula la Jacobiana General (e.f. de las articulaciones de los discos).

jacobianHRR(varargin): calcula la Jacobiana HRR (e.f. de los parámetros de sección).

changeSectionConfig(index, sectionJoints, varargin): cambia los ángulos de la configuración de una sección del robot.

setSectionConfig(index, sectionConfig): establece la configuración de una sección del robot.

setConfig(config): establece la configuración del robot.

resetConfig(): establece la configuración del robot a la de reposo.

move(x, y, z, type, varargin): caclula la cinemática inversa y establece la configuración del robot a lo calculado.

moveTrayectory(x, y, z, type, N, traj, varargin): calcula la trayectoria y mueve el robot.

maxManipConfig(varargin): devuelve la configuración de máxima manipulabilidad.

angulosSecciones(positions): devuelve los parámetros de las secciones dadas un vector de ángulos beta.

Privados

createBase()

createSections()

HRRobot

Clase Padre: *“HRRTree”*

Atributos y Métodos:

Públicos

BaseSTL_File: nombre del fichero con el STL de la base.

LinkSTL_File: nombre del fichero con el STL de los discos.

BaseSTL: mallado 3D de la base.

LinkSTL: mallado 3D de los discos.

HRRobot(NLinks, Alpha): constructor.

plot(varargin): imprime el robot usando el método *"show".*

plot2(varargin): imprime el robot con color y ejes de referencia del extremo.

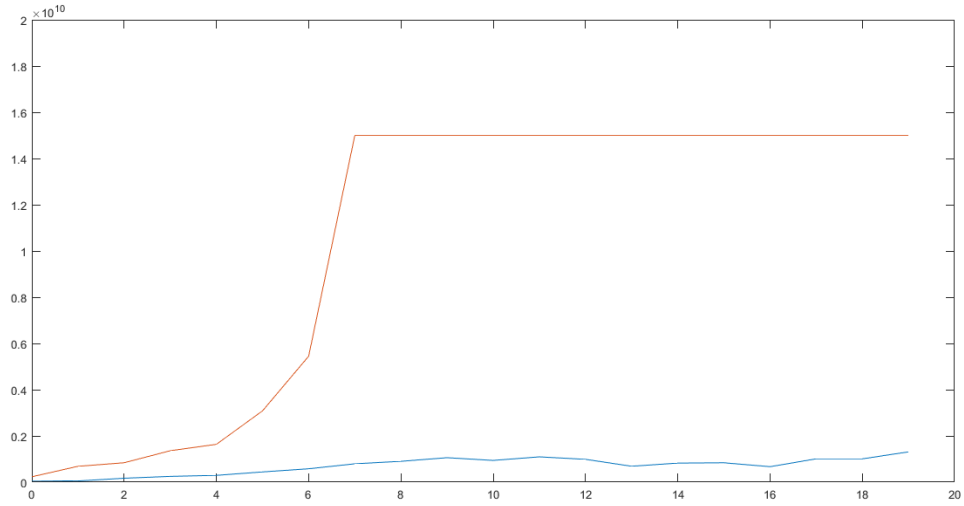
plotEllipsoid(scale, J): imprime el elipsoide de velocidad del extremo (versión en desarrollo).

addVisuals(varargin): establece los STL de los modelos 3D.

ANEXO 2: Resultados adicionales del Algoritmo Genético para Tres Secciones

Optimización Número de Discos/Sección

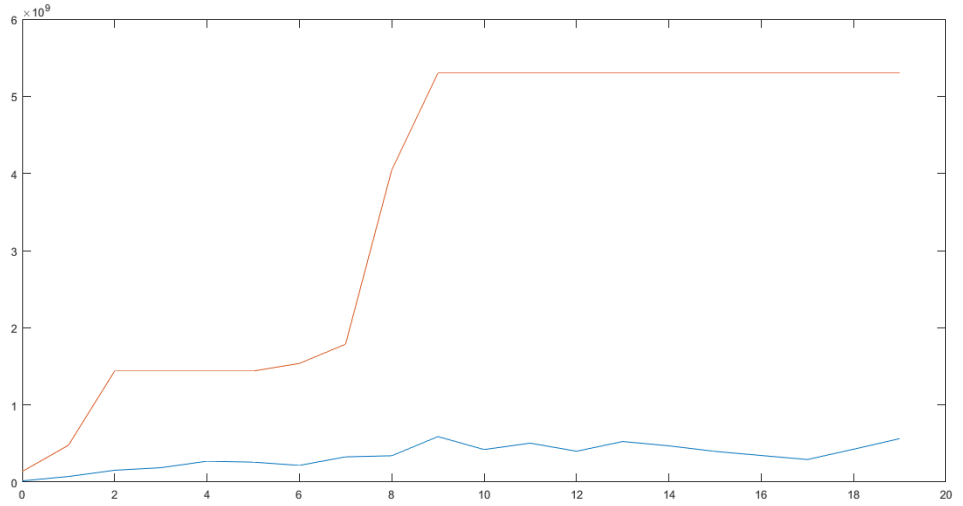
NSections = 3 NLinks = 70	MutationProb = 10% MutationIncrement = +- 2 discos	CostFunction = Manipulabilidad en un punto Punto = (0, 125, 100)



ÓPTIMO: [36 22 12]
SCORE: 1.4995 · 10^10

Optimización Número de Discos/Sección

NSections = 3 NLinks = 59	MutationProb = 10% MutationIncrement = +- 2 discos	CostFunction = Manipulabilidad en un punto Punto = (0, 125, 100)



ÓPTIMO: [32 18 9]
SCORE: 5.3010 · 10^9

ANEXO 3: Resultados del Algoritmo Genético para cinco secciones.

NUMBER OF LINKS	SCORE
[11;33;25;11;20]	2,58E+06
[11;30;23;18;18]	1,61E+08
[10;31;22;17;20]	2,13E+07
[11;30;23;18;18]	1,61E+08
[10;13;33;28;15]	9,75E+06
[11;30;23;18;18]	1,61E+08
[8;32;25;19;15]	3,67E+06
[9;26;21;15;29]	2,74E+06
[12;32;25;17;14]	1,63E+07
[10;30;22;20;17]	1,16E+07
[13;28;25;15;19]	3,26E+06
[11;29;23;19;18]	1,28E+07
[11;33;24;17;15]	2,52E+05
[24;27;20;15;15]	1,66E+07
[14;29;22;17;17]	9,44E+05
[11;30;23;17;17]	5,51E+06
[10;30;24;18;18]	2,72E+06
[12;28;26;17;17]	3,05E+07
[12;30;23;18;18]	2,68E+05
[11;30;24;17;17]	8,72E+05
[10;33;23;17;17]	3,64E+06
[11;30;23;18;18]	1,61E+08
[13;30;23;18;18]	1,83E+07
[11;31;23;18;17]	3,96E+06
[8;31;23;19;19]	2,45E+07
[12;33;24;20;10]	6,15E+06
[11;32;24;15;19]	3,26E+06
[10;29;28;17;17]	3,46E+06
[7;32;24;18;19]	4,41E+07
[11;32;21;18;18]	8,32E+04
[12;31;24;16;17]	1,49E+07
[11;28;24;18;18]	6,15E+06
[11;30;22;18;18]	1,33E+07
[11;30;23;18;18]	1,61E+08
[11;31;24;19;14]	3,75E+06
[11;30;23;18;18]	1,61E+08
[11;31;23;18;18]	1,27E+07
[11;30;22;18;18]	1,33E+07
[11;30;23;18;18]	1,61E+08
[12;29;23;18;18]	1,46E+08
[10;28;27;17;17]	3,23E+06
[12;31;24;13;19]	2,06E+07
[17;28;21;17;17]	3,66E+06
[12;26;24;19;19]	8,87E+05
[13;29;24;16;19]	3,13E+06
[10;28;20;17;25]	6,23E+06
[11;30;23;18;18]	1,61E+08
[11;30;23;19;17]	5,62E+06
[11;39;15;18;18]	1,70E+07
[11;30;23;18;18]	3,35E+07

ANEXO 4: Planificación Temporal y Presupuesto

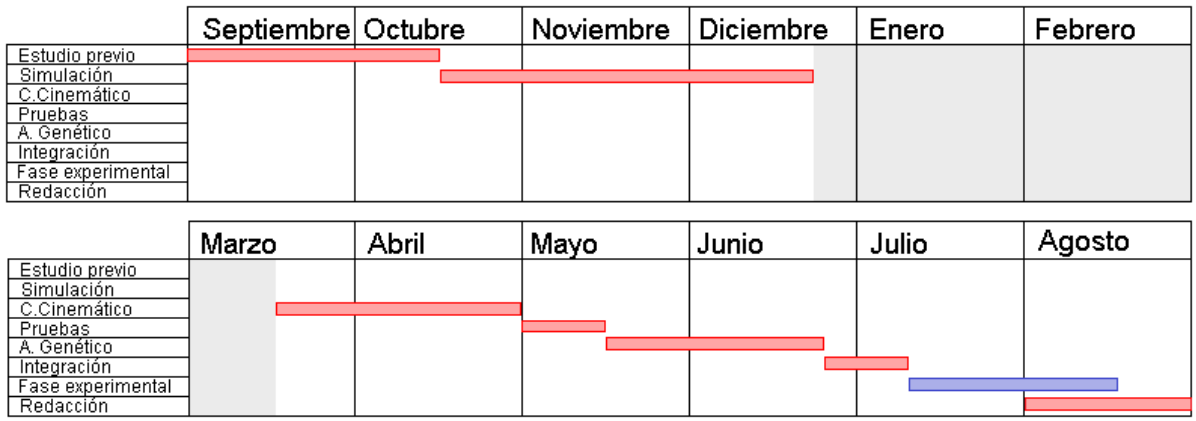


Figura 67: planificación temporal, diagrama de Gannt.

La planificación temporal ha tenido en cuenta ocho bloques bien diferenciados:

- **Estudio Previo**: se trata de la fase inicial de todo proyecto, que consiste en la recopilación de información y adquisición del *"know how"*.
- **Desarrollo del Entorno de Simulación**: ya en la fase de desarrollo, consiste en la implementación de las bases del entorno de simulación y sus herramientas más básicas – *definición de sistema de clases, creación y visualización de un robot básico, etc-*.
- **Desarrollo e Implementación del Control Cinemático**: consistente en la elaboración e implementación de las herramientas matemáticas y algoritmos para el posicionamiento del robot.
- **Fase de Pruebas**: compone el final de la primera fase de desarrollo, comprobando el correcto funcionamiento y la fiabilidad de lo desarrollado en los puntos anteriores.
- **Implementación del Algoritmo Genético**: supone el comienzo de la segunda fase de desarrollo, compuesta por la implementación del algoritmo genético.
- **Fase de Integración**: adaptación y optimización de todo el código, resolviendo posibles bugs e incompatibilidades. Supone el fin de la fase de desarrollo.
- **Fase Experimental**: una vez finalizado el programa, es posible comenzar los experimentos y la adquisición de datos.
- **Redacción**: se correspondería con la puesta en marcha del proyecto, recopilando toda la información sobre el trabajo realizado, así como las diferentes conclusiones.

Figura 68: planificación temporal, fases del proyecto.

Sobre el presupuesto requerido para el proyecto, dada la naturaleza del mismo, tan sólo se tendrán en cuenta las inversiones en recursos humanos, material de oficina y dispositivos informáticos. En base a esto, se ha hecho la siguiente estimación:

Material de Oficina		
	Básicos	20,00 €
	Calculadora científica	30,00 €
Material Informático		
	PC	500,00 €
Recursos Humanos		
	Salarios	1350,00 €
	Becario Grado (112,50€/25h) 12 Créditos ECTS (1ECTS = 25h)	
	TOTAL	**1.900,00 €**