INSTITUTO POLITÉCNICO NACIONAL
UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y TECNOLOGÍAS AVANZADAS

TRABAJO TERMINAL

"DISEÑO Y SIMULACIÓN DE UN ROBOT MODULAR AUTOCONFIGURABLE"

Que para obtener el Título de

"Ingeniero en Mecatrónica"

Presenta
Becerra Pedraza Marco Antonio

Asesores:
Dr. Carlos Artemio Coello Coello
M. en C. Miguel Ángel Rodríguez Fuentes

upiita

. *Junio del 2008*

INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍAY TECNOLOGÍAS AVANZADAS

UPIITA

TRABAJO TERMINAL

"DISEÑO Y SIMULACIÓN DE UN ROBOT MODULAR AUTOCONFIGURABLE"

Que para obtener el Título de

"Ingeniero en Mecatrónica"

Presenta:

Becerra Pedraza Marco Antonio

Asesores:

Dr. Carlos Artemio Coello Coello	M. en C. Miguel Ángel Rodríguez Fuentes
Presidente del Jurado	***Secretario***
Dr. Emmanuel Carlos Dean León	Dr. José Alfredo Rosas Flores

Junio del 2008

A mamá y a papá

A mis hermanos Itzel y Daniel

A mi esposa e hijos *(… ya vendrán)*

A la memoria de mis abuelos

Agradecimientos

Gracias a mis padres, hermanos y seres queridos por todo el afecto y apoyo que he recibido siempre.

Gracias a mis asesores: Dr. Carlos A. Coello Coello y M.C. Miguel Ángel Rodríguez Fuentes por aconsejarme y apoyarme en todo momento, y por compartir valiosos conocimientos conmigo.

Gracias a todos los profesores que han hecho que el estudio valga la pena para mí.

Gracias a todos mis compañeros del poli (en la vocacional y en la superior), por haberme hecho querer a esta noble institución.

Índice General

Resumen

En este proyecto se ha desarrollado una metodología basada en un algoritmo genético para hacer la planeación de movimientos necesarios para el auto-ensamble de un robot manipulador modular auto-configurable. Debido a la gran cantidad ensambles posibles con estos sistemas, no es práctico obtener modelos matemáticos para cada uno; entonces, surge la necesidad de técnicas no convencionales para ser usadas en este tipo de sistemas.

Este trabajo se centra en la planeación de movimientos de un robot manipulador con un número cambiante de eslabones y grados de libertad, es decir, resolver el problema de la cinemática inversa. La metodología basada en el algoritmo genético es desarrollada, y se reportan los resultados obtenidos en las simulaciones.

Abstract

In this project, it has been developed a methodology based in a genetic algorithm for the motion planning needed in a self-assembling modular robotic manipulator. Because the great quantity of possible assembles, it's not practical to describe a mathematical model for each possible assemble; that's why there's a need of non conventional techniques to be used in this kind of systems.

This project is focused in the motion planning of a manipulator with a changing number of links and DOF, in other words, the inverse kinematics problem. The genetic approach is developed and the simulation results are reported.

Palabras Clave

Robot Manipulador, Robot Modular, Computación Evolutiva, Optimización, Algoritmo Genético, Planeación de Trayectorias, Auto-configurable, Simulación.

Capítulo 1

Introducción

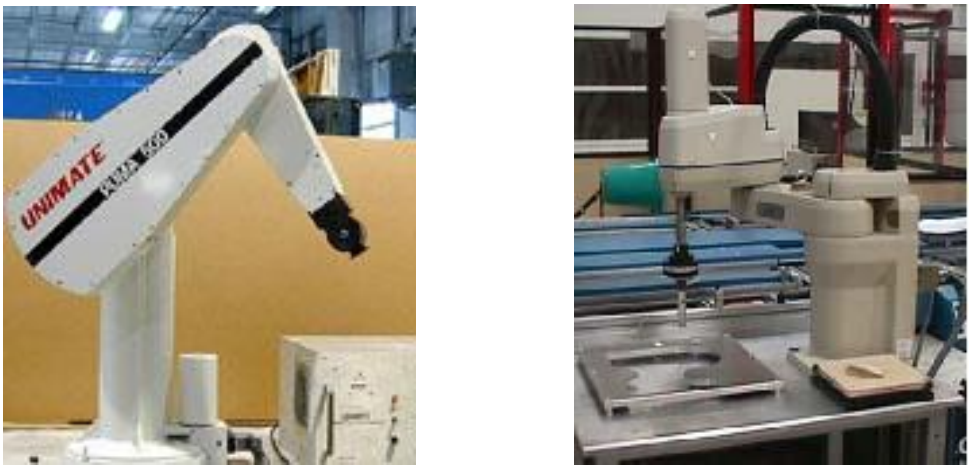
1.1 Introducción

Los robots con arquitectura fija constituyen la gran mayoría de los que se utilizan en todo el mundo. Esto es en parte porque se espera que el robot desarrolle una tarea simple y repetitiva dentro de un ambiente estructurado que no rebase las especificaciones para las que fue diseñado. Bajo estas condiciones, los robots se desenvuelven muy bien; sin embargo, el desempeño de un robot es muy bajo o nulo cuando el ambiente no cae en este rango, o cuando la tarea no puede ser realizada debido a restricciones mecánicas o físicas en el robot. En estos casos, surge la necesidad de sistemas robóticos que puedan adaptarse al ambiente en el que se desenvuelven y a la tarea que realizan. De esta manera, se han propuesto un tipo especial de robots capaces de modificar su forma para mejorar su desempeño.

Los Sistemas Robóticos Modulares (SRM) son aquellos que se componen de varios módulos conectados entre sí, con la característica de que pueden variar su configuración estructural de manera automática o semiautomática, para mejorar su desempeño. Los módulos pueden ser robots completos por sí mismos, capaces de realizar alguna función en particular; o pueden ser unidades que sólo son funcionales cuando se encuentran conectadas entre sí un número mínimo de ellas. Los módulos crean una estructura uniéndose a algún objeto fijo o entre ellos.

Se pueden obtener varias ventajas con este tipo de sistemas. Primero, el diseño del sistema es más simple, ya que solo implica el ensamble de piezas básicas (leggos robóticos). Segundo, dos módulos semejantes son intercambiables; esto añade al sistema una alta tolerancia a fallos, ya que si un módulo se descompone, puede ser sustituido por otro equivalente. Finalmente, estos sistemas son extensibles, se pueden agregar módulos extra al sistema para mejorar su desempeño, o para añadir redundancia. Todo esto trae como resultado sistemas robóticos más versátiles y confiables que sus homólogos de arquitectura fija.

En los robots manipuladores, la selección del robot depende en gran medida al tipo de tarea que se pretenda realizar. Al usar un tipo de robot en lugar de otro, se puede mejorar el desempeño del sistema al realizar la tarea. Por ejemplo, un robot tipo PUMA (fig. 1.1a) de 6-gdl (grados de libertad) tiene un alcance amplio en el espacio de operación, y es adecuado para tareas de pintado, soldado y manipulación de partes. Por otro lado, un robot SCARA (fig. 1.1b) de 3-gdl, es adecuado para ensamblajes de precisión sobre una superficie plana. Usar un manipulador PUMA para ensamblar partes electrónicas en una tablilla impresa sería mucho más complicado y menos eficiente que usar el robot SCARA. De esta manera, si sabemos de antemano la tarea que se va a realizar, podemos elegir el robot que más nos convenga. Sin embargo, en muchos ambientes no estructurados y menos predecibles, no es posible diseñar un sistema robótico que se adecúe a un amplio rango de tareas a realizar. En estos casos conviene un sistema que pueda adaptarse a las condiciones de su entorno para realizar la tarea, un SRM podría ser una alternativa.



(a) (b)

Fig. 1.1 Robots Manipuladores PUMA (a) y SCARA (b)

Hay muchos retos tecnológicos y teóricos que necesitan ser resueltos antes de que se pueda poner en marcha un SRM. El reto más elemental consiste en el diseño del módulo, que incluye el diseño mecánico y de la interfaz electrónica. Por otro lado se encuentra el diseño del sistema que se encargue de la planificación de las tareas y de cómo serán realizadas, y que regule la correcta interacción y funcionamiento de las partes.

Debido a que el sistema puede variar el número de eslabones que lo constituyen, y el cómo estos se conectan entre sí, no resulta práctico obtener el modelo dinámico ni las ecuaciones de cinemática inversa para cada uno de los casos posibles. Estas ecuaciones son necesarias para aplicar algún esquema de control entre los más comunes para robots manipuladores. Este proyecto se centra en el problema de la cinemática inversa de un SRM funcionando como

manipulador, en otras palabras, obtener los valores articulares necesarios para realizar alguna tarea.

1.2 Objetivos

Los objetivos principales de este Trabajo Terminal son:

- Desarrollar un método para resolver la cinemática inversa de un SRM, usando un algoritmo genético. Usarlo para generar trayectorias.

- Simular el auto-ensamble de un SRM en un ambiente virtual.

1.3 Estructura del Documento

El resto del documento está organizado de la siguiente manera. El Capítulo 2 realiza una semblanza histórica sobre la investigación en SRM, y en algunas ramas afines. El Capítulo 3 presenta algunos aspectos básicos para entender el resto del texto; análisis de robots manipuladores, optimización y una breve introducción a la computación evolutiva. En el Capítulo 4 se habla del planteamiento del problema en que se centra este trabajo. En el Capítulo 5 se desarrolla el algoritmo genético para resolver la cinemática inversa. En el Capítulo 6 se generan las trayectorias del auto-ensamble. Finalmente se presentan las conclusiones y el trabajo futuro.

Capítulo 2

Antecedentes y Estado del Arte

2.1 Antecedentes

2.1.1 Robótica

En el término robot confluyen las imágenes de máquinas para la realización de trabajos productivos así como de imitación de movimientos y comportamientos de seres vivos.

Desde la antigüedad, el hombre ha sentido fascinación por las máquinas que imitan la figura y los movimientos de seres animados, y existe una larga tradición de autómatas desde el mundo griego hasta nuestros días. El término 'robot' aparece por vez primera en la obra R.U.R. de Karel Capek en 1921.

De igual manera los avances logrados en la mecánica y en el control de máquinas nutren considerablemente el paso a dispositivos más sofisticados. Es a principios del siglo XX cuando se pasa de máquinas que tienen como objetivo exclusivo la amplificación de la potencia muscular del hombre, sustituyéndolo en su trabajo físico, a máquinas o instrumentos que son también capaces de procesar información, complementando, o incluso sustituyéndo al hombre en algunas actividades intelectuales. De esta manera, el siglo XX vio pasar el desarrollo de la robótica, pasando por los teleoperadores y las máquinas CNC hasta 1954 en que George Devol patentó el que se considera el primer robot industrial [13].

Los robots actuales son obras de ingeniería y como tales concebidas para producir bienes y servicios o explotar recursos naturales. Actualmente existe una gran variedad de robots, ya sean manipuladores, robots móviles, humanoides, zoomorfos, de expresión facial, etc., que también varían en el grado de autonomía con que cuentan. Sin embargo, la gran mayoría de los robots siguen siendo brazos manipuladores, debido a que son estos los que actualmente proveen mayores ventajas, y además son un bien presente y tangible.

2.1.2 Historia de los Sistemas Robóticos Modulares

El inicio del concepto de un robot modular auto-configurable puede ser puesto atrás con la aparición de los primeros efectores de 'cambio rápido' y los centros de maquinado CNC con cambio automático de herramienta en la década de 1970's. Aquí, módulos con conexiones comunes podían ser ensamblados automáticamente al final de un brazo robótico. Sin embargo, tomar el concepto de módulos con un mecanismo común, y aplicarlo a un robot en su totalidad fue idea de Toshio Fukuda [7] con su CEBOT (cellular robot) desarrollado al final de la década de 1980's.

A principios de la década de 1990's, el desarrollo en esta disciplina creció considerablemente, liderado por Greg Chirikjian, Mark Yim, Joseph Michael y Satoshi Murata. Chirikjian, Michael y Murata desarrollaron sistemas reconfigurables híbridos en red y en cadena, y Yim desarrolló un sistema reconfigurable en cadena (fig. 2.1). Mientras la investigación daba énfasis a construir los prototipos, y después desarrollar el código para ponerlos en marcha; los trabajos de Daniela Rus y Weimin Shen tuvieron un impacto más significativo en los aspectos de programación; ellos empezaron la tendencia en algoritmos distribuidos para manejar grandes cantidades de módulos.

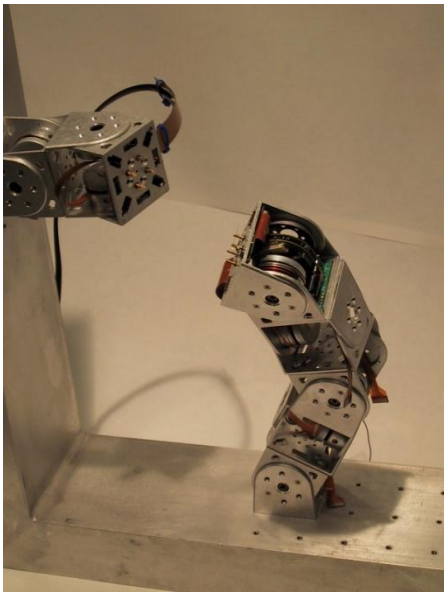


Fig. 2.1 Robot Modular Polybot G3, desarrollado en el PARC [b].

Recientemente se han tomado esfuerzos en auto-ensamble estocástico, esto es, el ensamble de módulos que se encuentran en desorden, incluso moviéndose en un medio (agua, aire, etc.). Cabe destacar el trabajo que se desarrolla en la

CMU, liderado por Seth Goldstein y Todd Mowry con miras a sistemas que cuenten con millones de módulos [a].

Muchas tareas han sido logradas, especialmente con configuraciones de SRM en cadena. Esto demuestra la versatilidad de estos sistemas; sin embargo, otros objetivos como robustez y bajo costo aún no han sido demostrados. En general, los sistemas desarrollados en los laboratorios han sido frágiles y muy caros, pero esto es de esperarse durante el inicio de cualquier rama de investigación tecnológica.

Hay un gran número de grupos de desarrollo involucrados en la investigación sobre SRM. A la fecha, existen cerca de 30 sistemas que han sido diseñados y construidos [d], algunos se muestran a continuación:

Sistema	Tipo, GDL	Autor	Año
CEBOT	movil	Fukuda et.al. (Tsukuba)	1988
Polypod	chain, 2, 3D	Yim (Stanford)	1993
Metamorphic	lattice, 6, 2D	Chirikjian (Caltech)	1993
Fracta	lattice, 3 2D	Murata (MEL)	1994
Fractal Robots	lattice, 3D	Michael(UK)	1995
Tetrobot	chain, 1 3D	Hamline et al (RPI)	1996
3D Fracta	lattice, 6 3D	Murata et al (MEL)	1998
Molecule	lattice, 4 3D	Kotay & Rus (Dartmouth)	1998
CONRO	chain, 2 3D	Will & Shen (USC/ISI)	1998
PolyBot	chain, 1 3D	Yim et.al (PARC)	1998
TeleCube	lattice, 6 3D	Suh et.al, (PARC)	1998
Vertical	lattice, 2D	Hosakawa et al, (Riken)	1998
Crystalline	lattice, 4 2D	Vona & Rus, (Dartmouth)	1999
I-Cube	lattice, 3D	Unsal, (CMU)	1999
M-TRAN I	hybrid, 2 3D	Murata et.al.(AIST)	1999
Pneumatic	lattice, 2D	Inou et.al., (TiTech)	2002
Uni Rover	mobile, 2 2D	Hirose et al., (TiTech)	2002
M-TRAN II	hybrid, 2 3D	Murata et al, (AIST)	2002
Atron	lattice, 1 3D	Stoy et.al., (U.S Denmark)	2003
S-bot	mobile, 3 2D	Mondada et.al, (EPFL)	2003
Stochastic	lattice, 0 3D	White, Kopanski, Lipson (Cornell)	2004
Superbot	hybrid, 3 3D	Shen et al, (USC/ISI)	2004
Y1 Modules	Chain, 1 3D	Gonzalez-Gomez et al, (UAM)	2004
M-TRAN III	hybrid, 2 3D	Kurokawa et.al., (AIST)	2005
Catom	lattice, 0 2D	Goldstein et al, (CMU)	2005
Stochastic-3D	lattice, 0 3D	White, Zykov, Lipson (Cornell)	2005
Molecubes	chain, 1 3D	Zykov, Mytilinaios, Lipson (Cornell)	2005
Prog. parts	lattice, 0 2D	Klavins, (U. Washington)	2005
Miche	lattice, 0 3D	Rus et.al, (MIT)	2006

2.1.3 Actualidad y Retos Futuros

Desde los primeros trabajos sobre SRM, el tamaño, robustez y desempeño de los sistemas ha sido continuamente mejorado. De la misma manera, se ha progresado en los algoritmos de planificación y control para manejar miles de módulos. Sin embargo, aún quedan algunos retos importantes que son necesarios para poder ver realizada la promesa de adaptabilidad, robustez y bajo costo. Estos retos pueden ser separados en el diseño del hardware, y en los algoritmos de control y planificación. Estos retos están relacionados entre sí. De acuerdo con [d], los retos más importantes que quedan pendientes son:

- **Diseño de Hardware**
  A la fecha se han demostrado sistemas de más de 50 piezas (56 piezas, Polybot [b]), sin embargo, este número se ha mantenido por casi una década. Existen algunas restricciones fundamentales que impiden aumentar este número.
    - Limitaciones en la fuerza y precisión de los mecanismos de ensamble.
    - Limitaciones en el par del motor, precisión de movimiento y eficiencia energética de los módulos.
    - Interacción Hardware-Software, ya que un buen diseño de hardware podría hacer más fácil la interacción con el software, y viceversa.

- **Planeación y control**
  A pesar de que se han desarrollado algoritmos para manejar miles de unidades en condiciones ideales, aún quedan retos pendientes para resolver problemas reales con pocos módulos y con una gran cantidad de ellos. Algunas mejoras potenciales serían:
    - Algoritmos para el movimiento paralelo entre agentes para manipulación y movimiento a gran escala.
    - Algoritmos para manejar robustamente una variedad de posibles fallas, como desalineaciones, módulos muertos (sin respuesta ni movimiento) y módulos con respuesta errónea.
    - Algoritmos que determinen la configuración óptima para alguna tarea determinada (síntesis de mecanismos).
    - Algoritmos para planear óptimamente (tiempo y energía) la reconfiguración.
    - Comunicación (asíncrona) eficiente y escalable entre módulos.

Capítulo 3

Conceptos Básicos

3.1 Análisis Robots Manipuladores[1]

Un robot manipulador es definido por el 'Robot Institute of America' de la siguiente manera [16]:

> *"Un manipulador reprogramable y multifuncional diseñado para mover material, partes y herramientas, o dispositivos especializados, mediante movimientos variables programados para la realización de una variedad de tareas"*

Esta definición provee una idea acerca de lo que puede ser considerado un robot manipulador, aún a pesar de que existan otras clases de robots, son los manipuladores los que han encontrado el campo de aplicación con mayor impacto para la robótica. Básicamente se puede entender por robot manipulador como un brazo mecánico operando bajo el mando de una computadora.

Un robot manipulador se compone de eslabones y articulaciones que los unen. Existen dos uniones posibles: prismática y revoluta.

[1] Para un análisis más detallado puede consultarse [18].

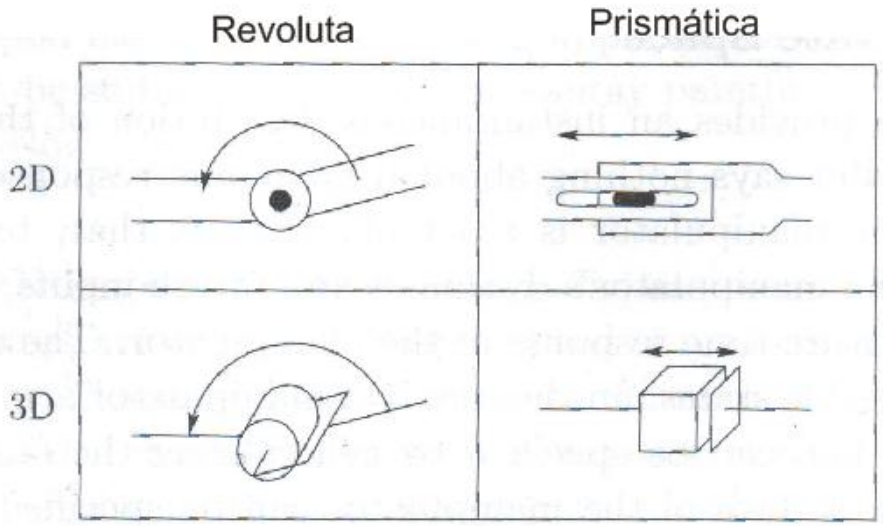


Fig. 3.1 Representación simbólica de las articulaciones de un robot manipulador.

A su vez un manipulador puede ser de cadena abierta, si está formado por una sucesión lineal de eslabones, o en el caso contrario de cadena cerrada.

(a) (b)

Fig. 3.2 (a) Manipulador de cadena abierta, y (b) un manipulador de cadena cerrada.

El análisis de un robot manipulador incluye la descripción del movimiento y de las fuerzas que intervienen en este, así mismo se busca poder predecir y controlar el comportamiento del sistema.

El estudio del movimiento puede dividirse en cinemática y dinámica. Por un lado, la cinemática atiende únicamente al movimiento (desplazamientos, velocidades y aceleraciones) entre los eslabones y en las articulaciones; a su vez, la dinámica toma en cuenta las fuerzas que intervienen en el movimiento.

Un robot manipulador cuenta con una base que en la mayoría de las aplicaciones se encuentra anclada, teniendo en cuenta esto, se puede asumir que se encuentra fija y ubicar ahí el sistema coordenado de referencia. La posición de todas las partes del sistema puede ser descrita en todo momento a partir de las variables articulares del sistema (q_i). Esto plantea un problema inicial, ya que

normalmente la tarea a realizar estará referida en coordenadas cartesianas del espacio de tarea, y no con respecto a las variables articulares del sistema.

El análisis cinemático se divide en cinemática directa e inversa. La cinemática directa se encarga de calcular la posición, orientación, velocidad y aceleración del efector en el espacio de tarea cuando los valores articulares son conocidos. La cinemática inversa se refiere al caso contrario, en el cual las variables articulares son calculadas a partir de los valores deseados del efector final en el espacio de tarea.

$$\text{Cinemática directa: } \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{pmatrix} \rightarrow \begin{pmatrix} x_e \\ y_e \\ z_e \\ o_e \end{pmatrix}$$

$$\text{Cinemática inversa: } \begin{pmatrix} x_e \\ y_e \\ z_e \\ o_e \end{pmatrix} \rightarrow \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{pmatrix}$$

Hay que mencionar que la cinemática inversa y directa proponen transformaciones entre dos espacios, el espacio articular y el espacio de tarea. El espacio de tarea a lo más 6 dimensiones (3 posiciones y 3 rotaciones para definir un cuerpo en el espacio). El espacio articular es n-dimensional, posee tantas dimensiones como articulaciones tenga el manipulador; de ahí que mientras más articulaciones posea un manipulador se incremente la complejidad de su análisis y la obtención de las ecuaciones de cinemática inversa.

Algunos aspectos del análisis cinemático incluyen el manejo de redundancia en el sistema (muchas posibilidades para efectuar el mismo movimiento), evasión de colisiones y evasión de singularidades.

Una vez que todas las posiciones, velocidades y aceleraciones han sido calculadas usando la cinemática, se usa la dinámica para estudiar el efecto de las fuerzas presentes en el sistema al ejecutar estos movimientos. La dinámica directa se refiere al cálculo de aceleraciones en el robot una vez que han aplicado fuerzas conocidas en las articulaciones. La dinámica inversa se refiere al cálculo de las fuerzas en los actuadores necesarias para llevar a cabo los movimientos deseados. Esta información es usada para implementar un esquema de control al robot y/o elegir actuadores.

3.1.1 Generación de Trayectorias

El problema de la generación (o planeación) de trayectorias puede ser planteado de la siguiente manera:

> *"A partir de los puntos, o de las configuraciones, inicial y final; encontrar una serie de movimientos articulares para alcanzar la posición final con una trayectoria válida y libre de colisiones."* [18]

Otro planteamiento [15] es el siguiente:

> *"Encontrar un camino para llegar de una configuración a otra en el espacio libre."*

Este problema puede ser planteado de una manera engañosamente simple, sin embargo la generación de trayectorias es uno de los problemas más complicados a tratar en ciencias de la computación. La complejidad computacional del mejor algoritmo completo para generación de trayectorias crece exponencialmente con el número de grados de libertad del robot; por eso, sólo se ocupan los algoritmos completos en robots pocos grados de libertad.

Cabe señalar que la planeación de trayectorias aporta información geométrica sobre el movimiento, no se especifica ningún aspecto dinámico consecuente al movimiento del robot.

3.1.1.1 Espacio de Configuración

En robótica, la especificación completa de la posición de cada punto del robot recibe el nombre de *configuración*, y el conjunto de todas las configuraciones posibles es llamado *espacio de configuración*. Normalmente se toma al vector q, compuesto por los valores articulares del robot, como una representación suficiente de la configuración. El espacio de configuración es denotado por $\mathcal{Q}$, y tiene dimensión igual al número de variables articulares con que se cuente.

Sin embargo, en la práctica no todo el espacio de configuración es útil, y esto es debido a dos razones: La primera es por las restricciones físicas que pueda presentar la articulación; por ejemplo, una revoluta idealmente se movería en el rango 0-360°, pero esto no siempre es posible, y puede ser que sólo esté permitido moverse en un rango, entonces su espacio configuración de delimitaría. En la fig. 3.3 se muestra un manipulador de 2-gdl con restricción de movimiento 0-180°, y su espacio de configuración utilizable (zona blanca).

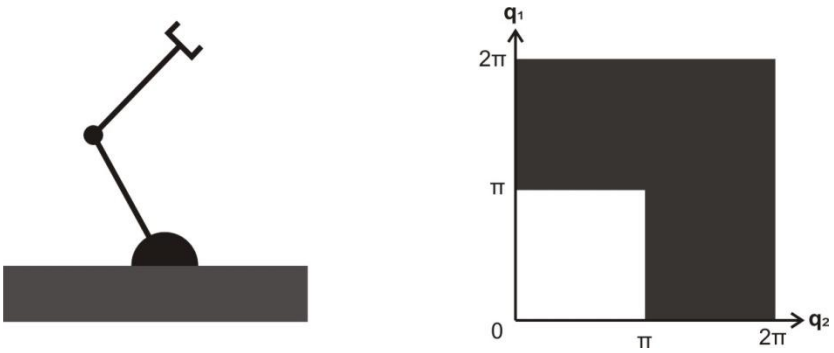
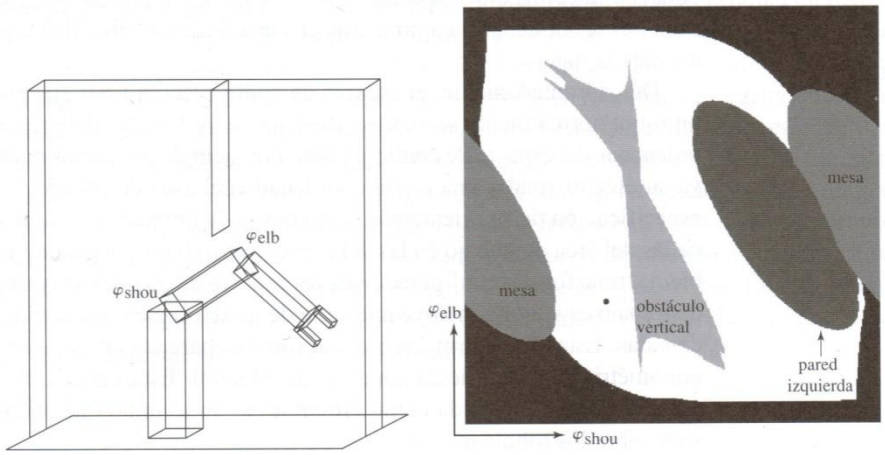


Fig. 3.3 Espacio de configuración utilizable (zona blanca) para un manipulador de 2-gdl con restricción de movimiento 0-180°.

El segundo factor que restringe el espacio de configuración utilizable son la presencia de obstáculos dentro del espacio operacional que puedan provocar colisiones con el manipulador. Para tener una trayectoria libre de colisión, es necesario que el robot nunca alcance una configuración que provoque contacto con obstáculos.



(a) (b)

Fig. 3.4 (a) Área de trabajo para un manipulador 2-gdl, compuesta por una caja y un obstáculo vertical plano. (b)El espacio de configuración del mismo robot. Sólo las regiones blancas son configuraciones libres de colisión. El punto corresponde a la configuración mostrada.

Resulta más fácil planificar con una representación en el espacio de configuración, ya que en vez de representar el estado del robot por las coordenadas cartesianas de sus elementos, su estado queda representado por la configuración de sus articulaciones.

Desafortunadamente, el espacio de configuración tiene sus problemas. La tarea de un robot normalmente se expresa mediante coordenadas operacionales (no articulares). Sin embargo, existen dos herramientas que permiten moverse de un espacio a otro: la cinemática directa y su inversa.

La cinemática directa se obtiene fácilmente con el método Denavit-Hartenberg, y a través de las matrices de transformación homogénea. Incluso puede obtenerse de manera automática a partir de los parámetros D-H y usando cálculo simbólico; la labor se limita a sustituir valores y multiplicar matrices.

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

Sin embargo, la obtención de las ecuaciones de cinemática inversa no es trivial, ya que se parte de las ecuaciones de cinemática directa y de ahí se despejan las variables articulares, labor nada sencilla de realizar.

3.2 Robots Modulares Auto-Configurables

Los robots modulares están compuestos por muchos bloques constructores de entre un limitado repertorio. Estos bloques cuentan con mecanismos de sujeción que les permiten acoplarse entre sí y transmitir fuerzas mecánicas y señales eléctricas.

Estos bloques constructores normalmente son unidades que aportan algún actuador, o unidades especializadas (un gripper, ruedas, cámaras, abastecimiento de energía, etc.). Existe la posibilidad de contar con un solo tipo de unidades (sistema homogéneo), o con varios tipos de ellas (sistema heterogéneo). Con un sistema homogéneo se obtiene simplicidad, pero con uno heterogéneo se hace más extenso el rango de actividades realizables, y se puede manejar especialización entre los módulos.

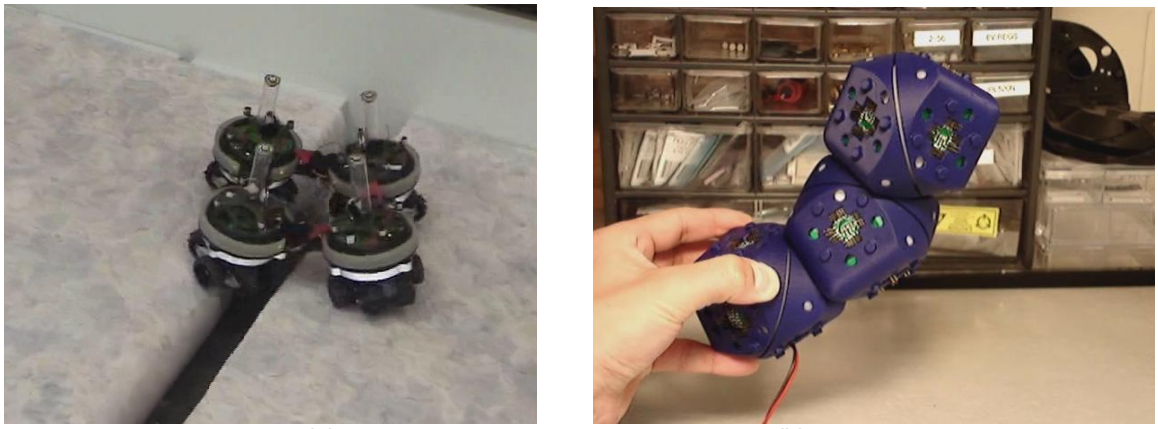
3.2.1 Taxonomía de las arquitecturas

Los robots modulares auto-configurables pueden ser agrupados en varios grupos atendiendo al tipo de ensamble geométrico que realizan. Algunos sistemas son híbridos entre dos o más grupos.

- **Estructuras en red (Lattice architectures)**.- Las unidades se encuentran acomodadas bajo algún patrón tridimensional, como una red cúbica o hexagonal. El control y el movimiento es ejecutado en paralelo. Estas

estructuras en red ofrecen una representación computacional más simple y puede ser escalada más fácilmente a sistemas más complejos.

- **Estructuras en cadena y/o en árbol (Chain/Tree architectures)**.- Las unidades se conectan entre sí formando una cadena, varias cadenas pueden estar ensambladas entre sí a través de nodos para formar árboles. Una estructura en cadena puede alcanzar cualquier punto en el espacio, sin embargo son computacionalmente más difíciles de representar y de analizar.



(a) (b)

Fig. 3.5 (a) Swarm-bots [e] de la EPFL, Suiza, librando un obstáculo (abismo) con una estructura en red (*lattice*), y (b) tres módulos de Molecube [c] ensamblados en cadena.

Los robots modulares auto-configurables también pueden clasificarse atendiendo a la manera en que ocurre la reconfiguración.

- **Reconfiguración Determinística**.- Consiste en moverse hacia donde se encuentra el objetivo. Las posiciones y orientaciones de los módulos 'objetivo' son conocidas en todo momento. De esta manera los tiempos de reconfiguración pueden ser asegurados.

- **Reconfiguración Estocástica**.- Consiste en unidades que se mueven usando procesos estocásticos (como el movimiento Browniano). El ensamble de las unidades solo puede darse cuando se encuentran suficientemente cerca. Los tiempos de ensamble sólo pueden asegurarse estadísticamente.

3.3 Optimización

La optimización es una rama de las matemáticas con un amplio campo de aplicación. Normalmente la optimización atiende al problema de maximizar o minimizar el desempeño de un evento. En procesos industriales aparecen constantemente problemas de optimización, de igual manera en la vida diaria.

Un problema de optimización se describe por un conjunto de elementos, de los cuales los más importantes son:

- Las *variables de decisión*. Son los valores que se modifican para resolver un problema.
- Las *funciones objetivo*. Es necesaria al menos una. Se expresan en términos de las variables de decisión, y el resultado de su evaluación es el que se desea optimizar (maximizar o minimizar).
- Las *restricciones*. Desigualdades o ecuaciones que se tienen que cumplir para que la solución se considere factible. Puede ocurrir que el problema no presente restricciones, en cuyo caso todas las soluciones son válidas, y el proceso de búsqueda se enfoca entonces en optimizar las funciones objetivo.

La teoría de optimización desarrolla métodos para efectuar elecciones óptimas de las variables de decisión que maximicen o minimicen la función (o funciones) objetivo. En algunos casos, la elección de las variables de decisión no es completamente libre, y está sujeta a restricciones.

Cuando se tiene más de una función objetivo, el problema se convierte en optimización multi-objetivo. Esta vez, se desea encontrar valores de las variables de decisión (sujetos a las restricciones) que logren un desempeño aceptable en cada función objetivo; normalmente el desempeño entre las funciones entra en conflicto.

Algunos métodos clásicos para atacar los problemas de optimización son el método del gradiente y la programación lineal. Recientemente la optimización ha tomado una nueva dirección al hacer uso de herramientas computacionales, principalmente por las ventajas de cálculo que ofrecen. Con el uso de técnicas heurísticas, se ha podido hacer frente a problemas muy difíciles o intratables (ya que en muchos problemas de optimización no se cuenta con suficiente información) mediante métodos convencionales; entre las heurísticas más sobresalientes encontramos el recocido simulado, la búsqueda tabú y la computación evolutiva.

3.4 Computación Evolutiva[2]

La computación evolutiva es considerada un campo de la inteligencia artificial que implica problemas de optimización, búsqueda y toma de decisiones. El término engloba a una serie de técnicas inspiradas biológicamente (en los principios de la teoría Neo-Darwiniana de la evolución natural (fig. 3.6)). En términos generales, para simular el proceso evolutivo en una computadora se requiere:

- Codificar estructuras que se replicarán.
- Operaciones que afecten a los “individuos”.
- Una función de aptitud.
- Un mecanismo de selección.

La evolución se simula aplicando operadores (selección, cruza y/o mutación) a la población de individuos (soluciones potenciales) en cada generación (iteración) para obtener una nueva población. Mediante la repetición controlada de este proceso se espera que la población explore equitativamente el espacio de aptitud y se concentre en aquellas zonas con un mejor desempeño hasta que se encuentre una solución aceptable para el problema.

Debido a que la población puede no converger (por ejemplo, una mutación muy alta), o no explorar correctamente el espacio de aptitud (se le puede escapar el óptimo global), es necesario controlar correctamente el proceso evolutivo, mismo que depende de cómo se haya planteado el problema y elegido los parámetros que lo regulan.

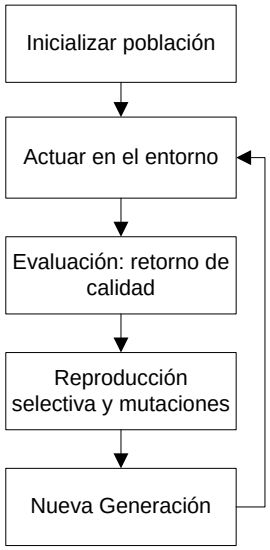


Fig. 3.6 Diagrama del proceso evolutivo.

[2] Para un mejor análisis puede verse también [8] y [4].

Se han desarrollado muchas técnicas que se basan en esta simulación evolutiva, sin embargo, se distinguen tres paradigmas principales:

- Programación Evolutiva
- Estrategias Evolutivas
- Algoritmos Genéticos
    - Programación Genética (es una variante de algoritmo genético).

Cada uno de estos paradigmas se originó de manera independiente y con motivaciones muy distintas [4].

3.4.1 Programación Evolutiva

Fue propuesta por Lawrence J. Fogel en los 1960's, el objetivo principal es obtener máquinas de estados finitos. Posteriormente David Fogel describió el algoritmo de programación evolutiva para optimización numérica.

En la programación evolutiva el operador de cruza no es importante, ya que se plantea como una abstracción de la evolución al nivel de las especies (especies diferentes no se cruzan entre sí). Asimismo, usa selección probabilística (comúnmente torneo estocástico).

3.4.2 Estrategia Evolutiva

A mediados de los 1960's, Peter Bienert, Ingo Rechenberg y Hans-Paul Shwefel desarrollaron un método de ajuste discreto aleatorio, inspirándose en el mecanismo de mutación presente en la naturaleza. Esta técnica fue llamada estrategia evolutiva y fue usada inicialmente para resolver problemas altamente complejos de hidrodinámica.

En la versión original se usa un solo padre que genera un solo hijo. Este hijo sobrevive para volverse el padre de la siguiente generación únicamente si tiene una mejor aptitud que el padre (mejor calificación); de lo contrario es eliminado.

En la estrategia evolutiva el operador principal es la mutación. El coeficiente de mutación se varía dependiendo la calidad de las soluciones obtenidas en la generación actual y la trayectoria de las generaciones pasadas. De esta manera se incrementa la mutación cuando la población se ha estancado (deja de explorar). El mejor individuo se mantiene a través de las generaciones.

3.4.3 Algoritmo Genético

Los algoritmos genéticos (AGs) fueron propuestos por John H. Holland [10] a principios de los 1960's. El objetivo principal de Holland era estudiar el mecanismo de adaptación natural para aplicarlo al aprendizaje de máquina. Actualmente los AGs son el tipo más popular de algoritmo evolutivo. El pseudocódigo de un algoritmo genético simple se muestra a continuación:

Algoritmo Genético Simple

1) Inicializar la población G con soluciones generadas aleatoriamente.
2) t = 0
3) **repetir**:
4) Evaluar G(t)
5) Elegir $G_1(t)$ de G(t) *[selección]*
6) Aplicar operadores de cruza y mutación a $G_1(t)$ para generar G(t+1)
7) t = t+1
8) **hasta** que la condición de paro se alcance.

El AG enfatiza la importancia de la cruza sexual (operador principal) sobre la mutación (operador secundario), y usa selección probabilística. La representación tradicional de los individuos es binaria; a esta cadena se le llama "cromosoma", y a cada posición en la cadena se le denomina "gene" y al valor dentro de esta posición se le llama "alelo".

3.4.4 Programación Genética

Nichal Lynn Cramer y después John R. Koza propusieron (de manera independiente) el uso de representaciones en árbol en las cuales del operador de cruza se aplicaba entre 'ramas' de las estructuras.

En la programación genética cada individuo es en realidad un programa codificado en un árbol (normalmente los programas se codifican en instrucciones de bajo nivel AND, OR, etc.). Se desea obtener un programa con algún comportamiento; entonces se evoluciona esta población hasta que la población codifica un programa adecuado. La cruza se realiza haciendo cortes en las ramas del árbol e intercambiándolas entre dos individuos. La mutación se realiza modificando la operación que realiza algún nodo.

La programación genética evoluciona programas para qué codifiquen algún comportamiento dado. Actualmente ha cobrado un gran impulso, y es usado entre otras, en aplicaciones de Robótica Autónoma, [14] y [3], y de diseño de circuitos lógicos [17].

Capítulo 4

Planteamiento del Problema

4.1 Análisis de los Sistemas Robóticos Modulares

El presente proyecto tiene como fin hacer un primer acercamiento al diseño de los Sistemas Robóticos Modulares (SRMs), así como de los mecanismos computacionales necesarios para que puedan realizar su tarea. Se ha hecho énfasis en un sistema modular auto-ensamblable y en el análisis cinemático necesario para el auto-ensamble.

4.1.1 Problemas iniciales

Debido a que para un sistema con estas características deseadas (número variable de eslabones y de grados de libertad) no resulta práctico realizar el análisis por los métodos convencionales, ya que se requiere un análisis individual cada vez que se tome un módulo y la complejidad computacional para hacer frente a algunas etapas del análisis crece exponencialmente (NP) con el número de grados de libertad en el sistema; por eso, es necesario proponer métodos alternos que faciliten el análisis de estos sistemas.

Existen muchos inconvenientes, el primero que salta a la vista es la obtención de un **modelo dinámico**. Debido a la naturaleza cambiante del sistema y a la posibilidad de contar con un número significativo de grados de libertad (10 ya pesarían en el análisis), obtener un modelo del sistema para cada caso posible se volvería una tarea muy complicada y nada factible de realizar.

Por otro lado, el **control** de un robot de esta naturaleza también sería un problema derivado principalmente por la carencia del modelo dinámico, esto porque la mayoría de los controles que se implementan en los manipuladores industriales hacen uso del modelo dinámico del mismo.

Otro problema sería también la **generación de trayectorias** para que estos robots puedan realizar una tarea. Este problema se puede descomponer en varias partes, ya que para que una trayectoria sea válida esta debería estar planteada para evitar colisiones y además, tomar en cuenta la dinámica del sistema (a lo mejor un robot realiza menos esfuerzos con un movimiento articular que con otro). Por otro

lado, también resulta complicado hacer planeación de trayectorias cuando el robot tiene un número cambiante de grados de libertad, , no resulta factible plantear ecuaciones de cinemática inversa para cada morfología posible del sistema. Por otro lado, es muy difícil calcular las ecuaciones de cinemática inversa para manipuladores con muchos grados de libertad debido al alto nivel de redundancia que poseen estos sistemas.

Finalmente existen otra serie de problemas que acarrean estos sistemas, algunos de los más importantes son: el **mecanismo de sujeción** (este debe ser fácil de acoplar y resistir los esfuerzos implicados), el **sistema de comunicación** (manejar una comunicación eficiente hacia los módulos y entre ellos), **síntesis de mecanismos** (obtener automáticamente un mecanismo apto para realizar una tarea dada y cómo ensamblarlo con los módulos disponibles) y **autonomía** (independencia y toma de decisiones por parte del sistema).

4.2 Caso de Estudio

Debido a la gran diversidad de posibles sistemas robóticos modulares auto-configurables, en este trabajo se ha propuesto un caso particular para su estudio: Un *manipulador modular planar auto-ensamblable.*

4.2.1 Descripción del experimento

Podemos separar en dos partes el funcionamiento de un robot modular auto-configurable: 1) El auto-ensamble del sistema para adquirir una estructura más adecuada para la tarea que deba realizar y 2) la realización de la tarea.

De esta manera se ha propuesto el experimento con las siguientes condiciones:

1. **El área de trabajo**
   El ambiente en el que se lleva la prueba es estático y libre de obstáculos. Tiene dos elementos principales, un punto de apoyo lateral y otro de abastecimiento de módulos. El primero es el punto de apoyo del manipulador, aquí se sujetará el primer eslabón. El segundo es la superficie sobre la cual se desplaza el módulo y en la cual se encuentran los módulos a ser ensamblados.

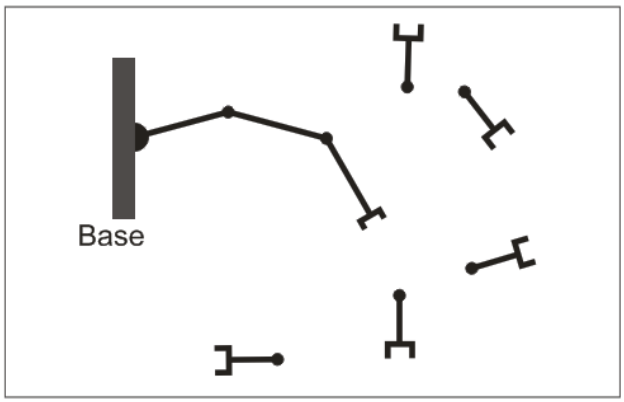


Fig. 4.1 Partes del área de trabajo: manipulador pre-ensamblado y eslabones sueltos.

2. **Inicio de la prueba**
   En un inicio ya hay módulos pre-ensamblados (al menos 3) y unidos a la base fija. De manera que, al comenzar la prueba ya se cuenta con un manipulador ‘pequeño’ listo para ser usado. Mientras tanto hay otros sobre la superficie de trabajo esperando ser ensamblados (fig. 4.1).

3. **Durante la prueba**
   En cada paso se genera la trayectoria necesaria para ir a tomar un nuevo eslabón, partiendo de una pose conocida en el manipulador. Una vez tomado el eslabón, el manipulador contará con un grado de libertad adicional.
   Este proceso se repite hasta que el manipulador cuente con el número esperado de módulos (fig. 4.2).

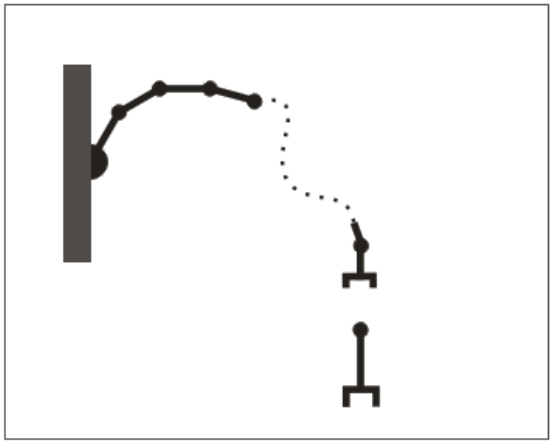


Fig. 4.2 Auto-ensamble del manipulador.

Es importante notar que el orden de ensamble de los módulos se define en el programa (el sistema no toma la desición).

4. **Conclusión**
   Una vez completado el auto-ensamble, el manipulador estará listo para desarrollar su tarea.

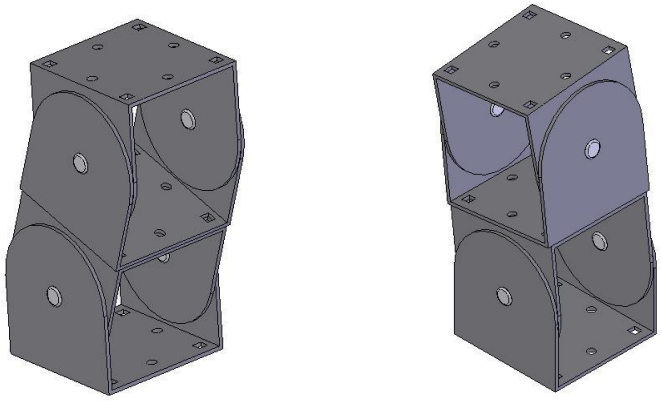
4.2.2 Sistema Usado

Se ha comentado el experimento en cuestión, sin embargo, es necesario proponer un módulo, debido a que las soluciones arrojadas deben respetar las restricciones de movimiento de los módulos, así que aquí se hace una presentación simplificada del módulo a usar, fig. 4.3

(a) (b)

Fig. 4.3 Tipo de módulo usado (a) y estructura ensamblada con 4 módulos (b).

Como se puede apreciar, cada módulo cuenta con 1-gdl y un sistema de sujeción automático. También, existen dos formas de conectar dos módulos entre sí: con sus ejes de movimiento paralelos (fig. 4.4a) o perpendiculares (fig. 4.4b). El sistema de sujeción debe admitir ambos ensambles y el módulo debe ser capaz de detectar la orientación del ensamble.



(a) (b)

Fig. 4.4 Conexiones entre módulos: (a) paralela y (b) perpendicular.

Es importante dejar en claro el conjunto de sistemas que se pueden obtener de estos módulos, debido a que el ensamble es en cadena se puede construir:

- **Manipuladores de revolutas.-** Formando una cadena de módulos, y fijando un extremo de la cadena, fig. 2.1.

- **Robots tipo Serpiente.-** Formando una cadena de módulos y dejando los extremos libres, fig. 4.5.

- **Orugas.-** Tomando una cadena de módulos suficientemente grande y uniendo los extremos, fig. 4.6.

Fig. 4.5 Robot tipo serpiente constituido de módulos.

Fig. 4.6 Robot modular ensamblado como oruga.

De entre los manipuladores de revolutas, los manipuladores planares son aquellos que, como su nombre lo indica, se mueven en el plano, esto sucede cuando los ejes de movimiento de los módulos son paralelos.

Fig. 4.7 Robot manipulador planar de N-gdl.

Finalmente, también hay que tomar en cuenta las restricciones físicas en el movimiento de las articulaciones. Debido al tipo de módulo, no es posible girar 360° con una articulación. Por motivos de estudio, se supondrá que cada articulación puede girar 180°, así como se muestra en la fig. 4.8. Se considerará el rango de movimiento de un módulo como [-90°, 90°].

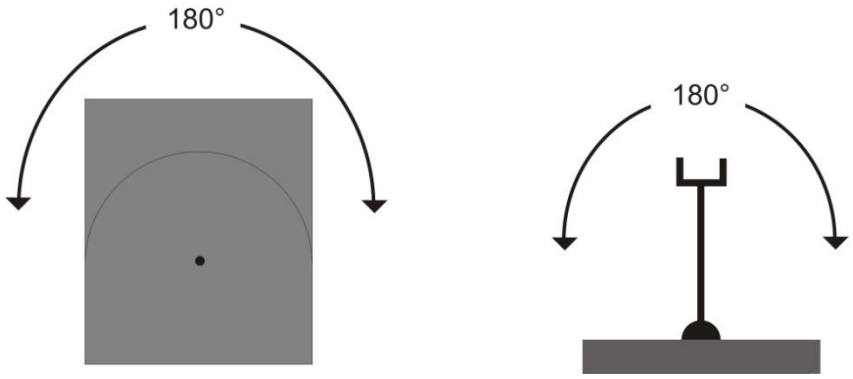
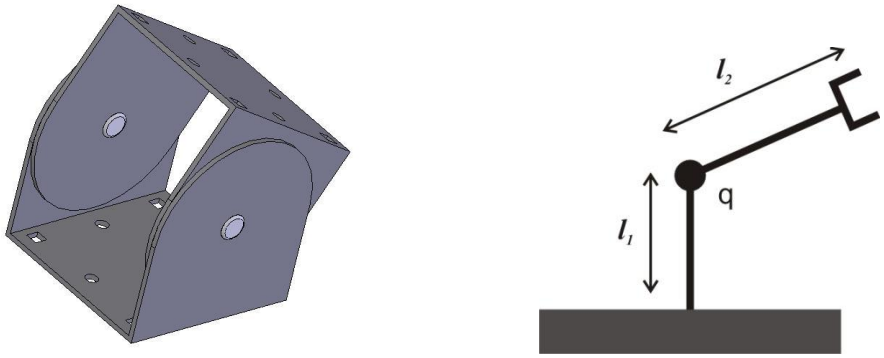


Fig. 4.8 Restricciones de movimiento físicas en el módulo propuesto.

4.2.3 Cinemática Directa del sistema propuesto

Aquí se presentan las ecuaciones resultantes para el manipulador planar que resulta de unir 'n' módulos. Estas ecuaciones serán ocupadas después por el algoritmo genético para calcular la cinemática inversa del manipulador.

En principio hay que revisar como está constituido el módulo. Este puede ser tratado como un robot de 1-gdl como el que se muestra en la fig. 4.9b.



(a) (b)

Fig. 4.9 (a) Módulo propuesto y (b) diagrama equivalente.

De este módulo, es necesario conocer la transformación que relaciona el las coordenadas del 'efector' con respecto al marco de referencia en la base. De la fig. 4.9b se puede observar que se realizan dos operaciones: una traslación constante l_1 (de la base al codo) y una rotación dependiente de la variable articular *q*.

Sin embargo, cuando conectamos un módulo adicional, la traslación constante del módulo nuevo solamente alarga la longitud del eslabón l_2 del módulo anterior; y así se sigue con cada módulo que es agregado. Por estas razones, y sin pérdida en el análisis del sistema, se puede simplificar el módulo como se muestra en la fig. 4.10, al que se le ha quitado la base.

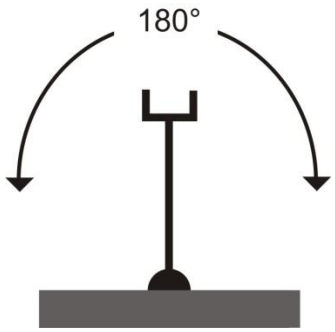


Fig. 4.10 Simplificación del módulo para su análisis. El movimiento del módulo sigue restringido en el rango [-90°, 90°].

Con este nuevo módulo, la matriz de transformación homogénea queda de la siguiente manera:

$$T = \begin{bmatrix} \cos q & -\sin q & 0 & l\cos q \\ \sin q & \cos q & 0 & l\sin q \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ahora, la cinemática directa de posición queda expresada por los primeros tres valores de la cuarta columna:

$$x = l * \cos(q)$$
$$y = l * \sin(q)$$
$$z = 0$$

Y la orientación del efector queda:

$$Oz = atan2(\sin(q), \cos(q))$$

Si se siguen agregando eslabones, sólo hay que multiplicar entre sí las matrices de transformación para cada eslabón. En el caso de un manipulador planar con n-gdl las ecuaciones de posición del efector final quedan:

$$X = \sum_{i=1}^{n} l_i * \cos\left(\sum_{j=1}^{i} q_j\right)$$

$$Y = \sum_{i=1}^{n} l_i * \sin\left(\sum_{j=1}^{i} q_j\right)$$

$$Z = 0$$

Y su orientación queda:

$$Oz = atan2(\sin\left(\sum_{i=1}^{n} q_i\right), \cos\left(\sum_{i=1}^{n} q_i\right))$$

Capítulo 5

Cinemática Inversa con un Algoritmo Genético

En el capítulo 3 se ha hecho una presentación breve a los algoritmos genéticos, ahora se explicará la manera en que se ha utilizado esta herramienta en el presente proyecto. Se muestran las tres versiones más significativas del AG usado para resolver la cinemática inversa del manipulador; se muestran simulaciones de ejemplo.

5.1 Análisis General

El problema de la cinemática inversa consiste en obtener un método para que a partir de la posición y orientación deseadas en el efector final de algún robot, obtener las variables articulares que realizan esa pose en el manipulador. Esta solución en la mayoría de los casos no es única, fig. 5.1.

La forma común en que se especifica la trayectoria de un robot es declarando una serie de puntos (posiciones y orientaciones) que se desea que recorra en orden el manipulador con su efector final. Estas trayectorias deberían tomar en cuenta el comportamiento dinámico y la evasión de obstáculos. En este trabajo se consideran las trayectorias de manera discreta y libres de obstáculos[3], y se han generado de manera independiente al comportamiento dinámico del manipulador.

Dado que las trayectorias se especifican cómo puntos, el problema inicial se puede plantear de la siguiente manera:

> *"Dada una configuración inicial, con un manipulador planar de n-gdl, obtener los movimientos articulares necesarios para llegar a un punto destino (dado en coordenadas cartesianas) con el efector final"*

Es decir, resolver la cinemática inversa para cada punto de la trayectoria, para esto se ha implementado un algoritmo genético.

[3] No se toma en cuenta la evasión de obstáculos, aunque la metodología desarrollada puede combinarse con otra que se encargue de ello.

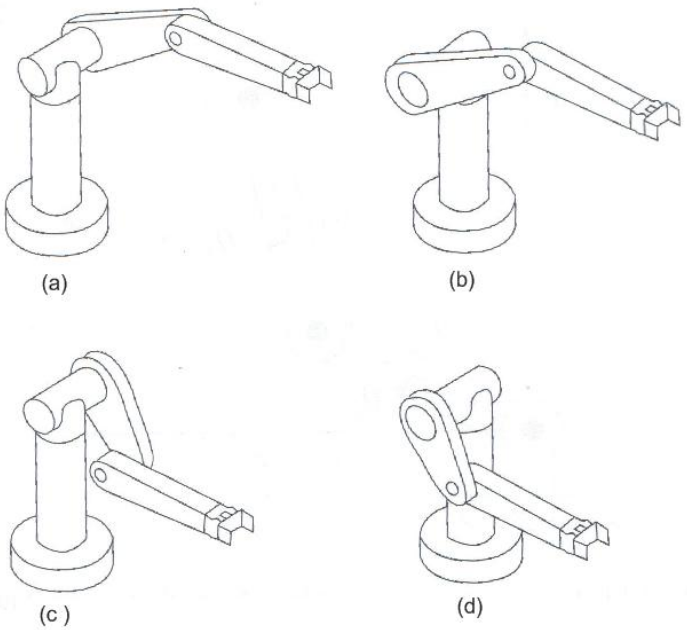


Fig. 5.1 Cuatro soluciones al problema de la cinemática inversa de posición para un robot PUMA.

Este problema debe ser entendido como un problema de **búsqueda** en el espacio de configuración. Dentro de este espacio se buscará un punto que satisfaga la cinemática inversa del manipulador (dentro de un rango de error). Esta solución deberá estar cercana a la configuración inicial para minimizar el movimiento articular en el manipulador fig. 5.2. (Nota: ya se ha descrito que un punto en el espacio de configuración representa una configuración particular del manipulador).

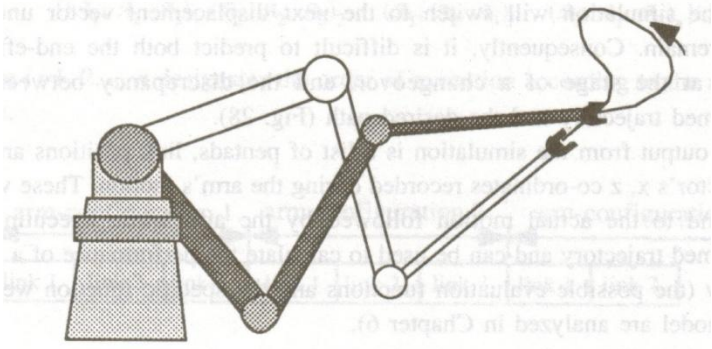
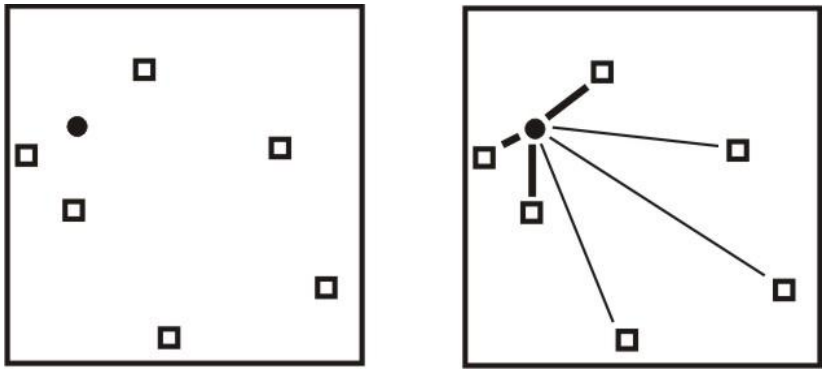


Fig. 5.2 A pesar de que en ambas configuraciones las posiciones de sus efectores están muy cercanas, el pasar de una a otra produce una trayectoria muy distorsionada; ésta es originada por los movimientos articulares vigorosos.

En la fig. 5.3 se presenta una aproximación al problema. Sólo por el momento, y para ilustrar la idea, supongamos un espacio de búsqueda bidimensional. En

este espacio se encuentra la configuración inicial (punto negro) y un conjunto de soluciones válidas al problema (cuadros). En el experimento planeado (ignorando los obstáculos presentes), es posible encontrar una trayectoria desde la configuración inicial a cualquiera de las soluciones. Sin embargo, aquellas que se encuentren más cercanas a la configuración representan trayectorias con menor movimiento articular (fig. 5.3b).



(a) (b)

Fig. 5.3 (a) Espacio de configuración hipotético; el punto representa la configuración inicial, y los cuadrados las posibles soluciones del problema. Aquellas soluciones más cercanas a la configuración inicial representarán trayectorias con menor movimiento articular. En (b) se muestran resaltadas las mejores trayectorias posibles.

Lo que interesa es encontrar un punto solución en el espacio de configuración y que este punto sea cercano a la configuración inicial para que no tenga un costo de movimiento muy alto.

Cabe señalar que para este experimento se consideraron las longitudes de todos los eslabones como unitarias; el error arrojado en las pruebas también está en esta misma proporción.

5.2 Planteamiento del AG

Los trabajos relacionados con la utilización de AGs para generar trayectorias de robots manipuladores se han enfocado en tres problemas, principalmente:

- La generación de trayectorias de manipuladores de varios grados de libertad. [5], [1] y [12].
- La generación de trayectorias en espacios de operación complejos y que cambian en el tiempo. Por ejemplo, para que dos manipuladores puedan compartir su espacio de operación sin chocar entre sí, e incluso compartir tareas. [2]
- Optimización de trayectorias. [5]

Los AGs poseen muchas características que resultan útiles en el problema de la generación de trayectorias:

- Debido a su eficiencia a la hora de explorar un espacio de aptitud en busca del óptimo.
- No es necesario conocer ni tener caracterizada la función a optimizar, tampoco se requiere información sobre esta (continuidad, derivable, etc.), sólo es necesario tener acceso a ella para poder evaluar a los individuos.
- De la función a optimizar, tampoco es necesario que el óptimo sea único.

De acuerdo con [4], para poder aplicar un AG a un problema se requiere de los 5 componentes básicos siguientes:

- Una representación de las soluciones potenciales del problema.
- Una forma de poder crear una población inicial de posibles soluciones (normalmente un proceso aleatorio).
- Una función de evaluación que juegue el papel de ambiente, clasificando las soluciones en términos de su “aptitud”.
- Operadores genéticos que alteren la composición de los hijos que se producirán en las siguientes generaciones.
- Valores para los diferentes parámetros que utiliza el AG (tamaño de la población, probabilidad de cruza, probabilidad de mutación, número máximo de generaciones, etc.).

Así que hay que plantear la manera en que se ha descrito cada uno de estos puntos.

5.2.1 Representación de los individuos

Debido a que se va a realizar una búsqueda en el espacio de configuración, es necesario que cada individuo del algoritmo genético represente un punto en este espacio. Cada individuo está constituido por un vector $q=(q_1, q_2, \ldots, q_n)$, cada q_i representa una variable articular del robot.

Se sabe que cada q_i puede moverse en un rango $[q_{min}, q_{max}]$ debido a restricciones físicas en la articulación; para el sistema en cuestión ya se ha marcado que este rango es [-pi/2, pi/2].

Luego hay que binarizar este vector de la siguiente manera. Primero hay que saber cuántos bits se usarán para representar cada q_i, este número estará denotado por la letra l. De manera que convertir la variable articular a una cadena binaria se logra con la siguiente fórmula:

$$c_i = decimal\_a\_binario\left(\left\lfloor\left(\frac{q_i - q_{\min}}{q_{\max} - q_{\min}}\right) * 2^l\right\rfloor\right)$$

Para volver a obtener el valor de la variable articular (c_i → q_i), únicamente hay que aplicar la operación inversa:

$$q_i = \frac{binario\_a\_decimal(c_i)}{2^l} * (q_{\max} - q_{\min}) + q_{\min}$$

De esta manera ya se cuenta con la representación binaria de cada articulación. Se debe realizar esto para cada articulación de cada configuración (individuo). De manera que cada individuo tiene asociada una cadena (genotipo) que representa la configuración (fenotipo) de ese individuo particular. Cabe señalar entonces que esta cadena tendrá una longitud n*l, siendo n el número de grados de libertad del manipulador y l es el número de bits utilizados para representar cada variable articular.

Por ejemplo, para un manipulador de 3-gdl, y tomando una l=4, una cadena válida podría ser 001011101011, que en realidad codifica información ordenada en tres bloques [0010, 1110, 1011], y cada uno representa un valor articular. Esta representación se uso a lo largo de todo el proyecto sin modificaciones.

5.2.2 Generación de la población inicial

Usualmente la población inicial se genera de manera aleatoria, ya que así, se reparten los individuos sobre el espacio de aptitud de manera estadísticamente uniforme; esto es deseable, ya que de esta manera la población inicial representa una visión simplificada del espacio de aptitud, y le permite al algoritmo concentrar la búsqueda, en las siguientes generaciones, en aquellas regiones donde los individuos obtienen las mejores evaluaciones, mientras al mismo tiempo sigue explorando el espacio de aptitud.

Para calcular la cinemática inversa surgen algunos problemas:

- Lo primero sería **encontrar una configuración válida**, sin embargo, la solución casi nunca es única (fig. 5.1), y el número de soluciones se incrementa de manera exponencial con el número de articulaciones.
- Debido a que la configuración calculada es parte de una trayectoria es necesario **tomar en cuenta la pose anterior del manipulador** (fig. 5.2). Esto porque de entre todas las soluciones posibles, la mejor es aquella que implique el cambio articular menor[4].

[4] Esta solución no necesariamente es la mejor, debido a otros factores que no se han tomado en cuenta en este trabajo; por ejemplo, puede haber obstáculos que le impidan al manipulador alcanzar esa configuración, o los esfuerzos mecánicos implicados.

Los primeros experimentos de este proyecto se realizaron poniendo la población inicial de manera aleatoria:

$$población_{inicial} = rand$$

Sin embargo, como se aprecia en la (fig. 5.2), aunque se obtienen soluciones válidas, éstas no siempre se parecen[5] a la configuración inicial. Por estas razones en la segunda y tercera versiones del AG, se optó por **inicializar la población como la configuración anterior**, esto con el fin de explorar el espacio de configuración alrededor de la configuración inicial y encontrar aquellas soluciones que se encuentren más cercanas a esta, y por tanto que minimicen el cambio articular.

$$población_{inicial} = q_o$$

5.2.3 Función de evaluación

La función de evaluación (o de aptitud) es la encargada de decir que tan 'aptos' son los individuos de la población, esta función debe asignar mejores calificaciones a aquellos individuos que resuelvan mejor el problema.

Para el presente trabajo, se necesita que la solución que regrese el AG tome en cuenta tres factores:

1) Error de posición (e_{pos})

2) Error de Orientación (e_{or})

3) Cambio articular mínimo (e_{art})

Es necesario que el AG **minimice** estos parámetros hasta que se encuentren en un rango de error aceptable, principalmente los primeros 2 que componen la demanda principal del problema de la cinemática inversa (alcanzar la posición y la orientación deseadas).

5.2.4 Operadores Genéticos

Una vez que se cuenta con una población inicial, esta se evoluciona aplicando los *operadores genéticos* para generar la descendencia que pasará a la siguiente generación. Esta descendencia es (estadísticamente hablando) mejor que la anterior, y el objetivo del AG es hacerla tender al óptimo global de la función de aptitud.

[5] Minimizan el cambio articular.

Los operadores genéticos principales son tres (cruza, selección y mutación), y también se considerará aquí el mecanismo de elitismo.

5.2.4.1 Operador de Selección

La selección se ocupa de elegir de la población a los individuos que participarán en la elaboración de la descendencia. El principal criterio a tomar en cuenta para la selección es la aptitud de los individuos.

El método empleado en el presente trabajo es la selección por el método de la ruleta:

Selección por Ruleta

9) Sumar las aptitudes (A_i) de toda la población (q_i) en sum_apt
10) $P_1=A_1$/sum_apt
11) **for** i=2:n (tamaño de la población)
12) $P_i = (A_{i-1}+A_i)$ / sum_apt
13) **end for**

//selección de individuo:
14) Generar un número aleatorio $s \in [0,1]$ con distribución uniforme.
15) **for** i=1:n
16) **if** P_i>s **then**
17) q_i es el individuo seleccionado
18) salir del for
19) **end if**
20) **end for**

Con el método de la ruleta se escoge un individuo con una probabilidad proporcional al porcentaje que represente su aptitud en el total de la población, fig. 5.4. También se ha escogido esta debido a que es uno de los métodos más comunes de hacer selección y es el que se maneja en el *Algoritmo Genético Simple* [8].

Con el operador de selección se escogerá de entre la población los individuos padres de la siguiente generación.

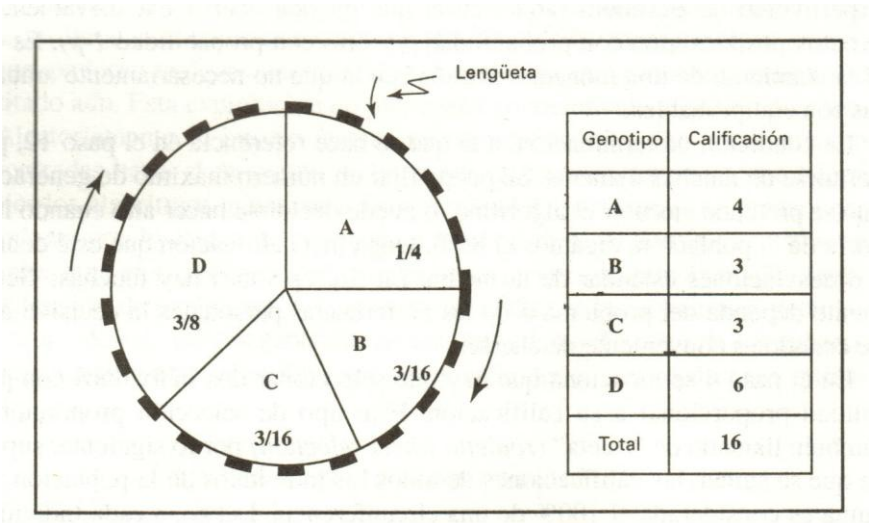


Fig. 5.4 Selección proporcional o por ruleta.

5.2.4.2 Operador de Cruza

La cruza es un operador que forma un nuevo cromosoma combinando partes de cada uno de los cromosomas padres.

Una vez que se han seleccionado dos individuos (padres), se procede a hacer la cruza de estos con probabilidad p_c, o el pase automático de los padres a la siguiente generación.

Si se hace cruza entre los padres, entonces se pueden establecer varios puntos de cruza. En este trabajo, la cruza que se realiza es simple y por un solo punto, como se muestra en la fig. 5.5.

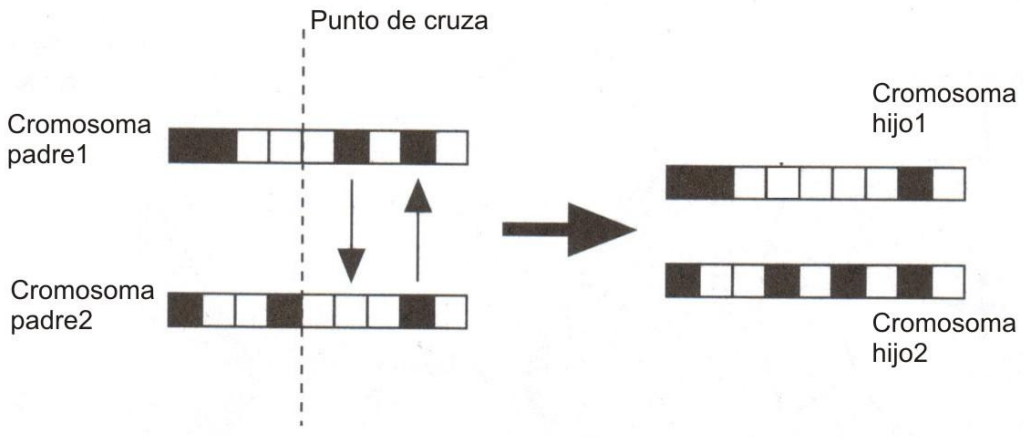


Fig. 5.5 Cruza de dos cromosomas

Después de la cruza se tienen dos individuos que pasan a la siguiente generación.

5.2.4.3 Operador de Mutación

Un operador de mutación consiste en modificar con probabilidad p_m los bits en los cromosomas de los hijos. Esto garantiza la exploración del espacio de aptitud, al mismo tiempo que la población converge en algunas regiones. La mutación es el operador secundario en un AG. Se recomienda usar valores pequeños del coeficiente de mutación (p_m) [4], valores en el rango de 0.001 - 0.01 se consideran aceptables.

En el presente trabajo se ha empleado dos tipos de mutación, de la manera convencional, y una mutación ponderada, en donde la probabilidad de mutación de los bits en un cromosoma esta ponderada a la posición que ocupa el bit. Ambas se describen más adelante.

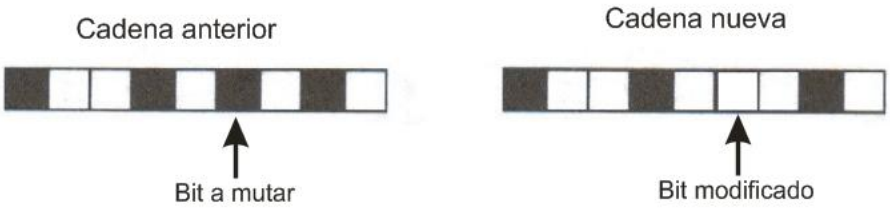


Fig. 5.6 Mutación simple.

5.2.4.4 Elitismo

El elitismo es el mecanismo usado en computación evolutiva para asegurar que los cromosomas de los miembros más aptos de una población se pasen a la siguiente generación sin ser alterados. Usar elitismo asegura que la aptitud máxima de la población nunca se reducirá de una generación a la siguiente. Sin embargo, no necesariamente mejora la posibilidad de localizar el óptimo global de una función[6].

En el proyecto en cuestión se ha usado elitismo. Una vez que se han calculado las aptitudes se la población, se busca el mejor individuo y se guarda para que pase a la siguiente generación. De esta manera este individuo puede participar en la elaboración de la demás descendencia, pero permanece una réplica de él para la siguiente generación.

[6] Podría condicionar la exploración del espacio de aptitud.

5.2.5 Parámetros del AG

Estos parámetros son los valores elegidos para el tamaño de la población, la probabilidad de cruza, la probabilidad de mutación, número máximo de generaciones, etc.

Desde los inicios de la computación evolutiva ha existido un debate sobre cómo elegir estos valores para que el AG tenga un comportamiento más eficiente (converja a la solución en un tiempo acotado y lo más pequeño posible). Algunos trabajos han estado enfocados a encontrar un rango de valores en el que el AG tenga un mejor desempeño; de entre otros, cabe destacar el trabajo doctoral de Kenneth De Jong [6] de principios de los 1970's.

En el trabajo de De Jong, se midió el desempeño de un AG al optimizar 5 funciones diferentes que resultaban difíciles de optimizar por el método de gradiente. De Jong puso especial cuidado para medir el desempeño del AG al variar los parámetros. Los valores a los que se llegó fueron:

- $p_c = 0.6$
- $p_m = 0.01$
- $Tamaño\ de\ la\ población = [50{,}100]$

En los 1980's Grefenstette [9] usó un AG para optimizar los parámetros de otro AG (un meta AG). Los valores a los que llegó en sus experimentos fueron:

- $Tamaño\ de\ la\ población = 30$
- $p_c = 0.95$
- $p_m = 0.01$
- $Selección\ elitista$

Sobre el tamaño de la población, en este trabajo se han empleado poblaciones de 100 y 51 individuos.

Tomando en cuenta la información recogida de la literatura [4], [8] y [11], se han propuesto los valores de los parámetros para las versiones de AG que se desarrollaron.

5.3 Primera Versión – AG básico

5.3.1 Descripción

En esta primera versión del AG se hizo énfasis en resolver la cinemática inversa de posición del robot. Las características más importantes a tomar en cuenta son:

- La representación de los individuos fue de 16 bits por variable articular, esto quiere decir que para manipuladores con *n* articulaciones, los cromosomas de los individuos fueron cadenas de *16*n* bits.

  El hecho de que se codifique una variable articular (revoluta) con 16 bits implica que se podrán codificar 2^{16} valores distintos, esto es 65536 valores posibles. Esto da que la resolución considerada en cada revoluta sea de $\frac{180°}{65536} = 0.0027°$, lo cual es bastante aceptable.

- La población inicial se generó aleatoriamente.

- La función de aptitud utilizada únicamente toma en cuenta el error de posición del robot, estando descrita de la siguiente manera:

  $$EVA_i = 2 * n * l_e - err_{pos}$$

  Donde *n* es el número de eslabones en el manipulador, y l_e es la longitud del eslabón. El primer término $(2 * n * l_e)$ representa la distancia máxima que puede existir entre las coordenadas de los efectores para dos configuraciones cualquiera, de manera que la aptitud asignada a los individuos siempre es positiva y tiene valor máximo cuando el error de posición (err_{pos}) es cero.

- Los operadores genéticos se manejaron de manera convencional:
  - La selección es por el método de la ruleta.
  - La cruza se efectuó por un punto, una vez obtenidos los hijos, se escogió al mejor individuo de entre los dos padres y los dos hijos para pasar a la siguiente generación.
  - La mutación se manejo en toda la cadena con la misma probabilidad $p_m = 0.001$
  - Se utilizó elitismo para el mejor individuo.

- Los parámetros del AG que se manejaron fueron:
  - $Tamaño\ de\ la\ población = 100$
  - $generaciones = 100$

Esta versión es la que más asemeja a la idea del Algoritmo Genético Simple (SGA) [8], con excepción de la implementación de la cruza. Sin embargo, debido a que en ningún momento se toma en cuenta la configuración anterior y que en la casi totalidad de los casos la solución a la cinemática inversa no es única, el AG regresará la primera solución que encuentre (entre todo el conjunto posible), y esta no es necesariamente la que mejor resuelve el problema, de hecho en la mayoría de los casos, la solución arrojada implicará un cambio articular significativo.

5.3.2 Experimentos

Con esta versión del AG se realizaron tres experimentos con un manipulador de 3, 6 y 15 articulaciones. Aquí se muestran los resultados arrojados en la ejecución del AG. Cada prueba se repitió tres veces, y el tiempo aproximado de la ejecución fue de 40 segundos, sin embargo, gran parte del tiempo de ejecución se usa en la graficación de la población actual.

El AG se implementó (Véase Apéndice A) y se ejecutó en MATLAB 2007 con un sistema operativo Windows Vista Premium; en una PC con un procesador Intel Core Duo, y con 2GB en RAM.

5.3.2.1 Prueba 1: Robot 3-gdl

En este experimento se pidió al programa generar la cinemática inversa para un manipulador de 3-gdl. El punto destino fue $P_{deseado}$=(1.5, 0).

Como se puede observar, el número de bits empleado para describir cada individuo (configuración) es de $16 * gdl$, por lo tanto la longitud de cada cromosoma será de $16 * 3 = 48 bits$.

Lo que se puede observar en las imágenes es la población de individuos (configuraciones de robots) en esa generación, y con un punto azul la posición meta.

- **Primera Ejecución**

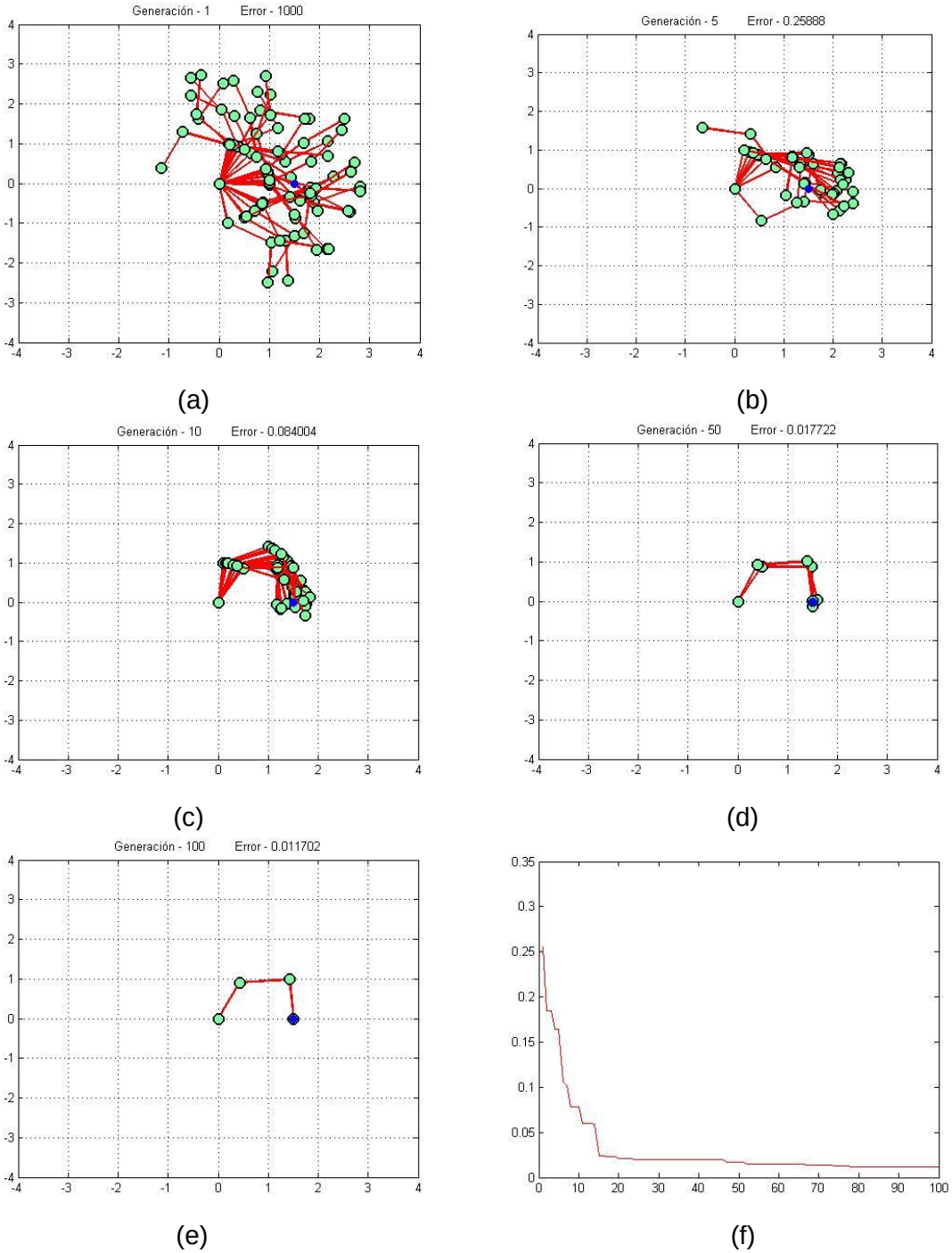


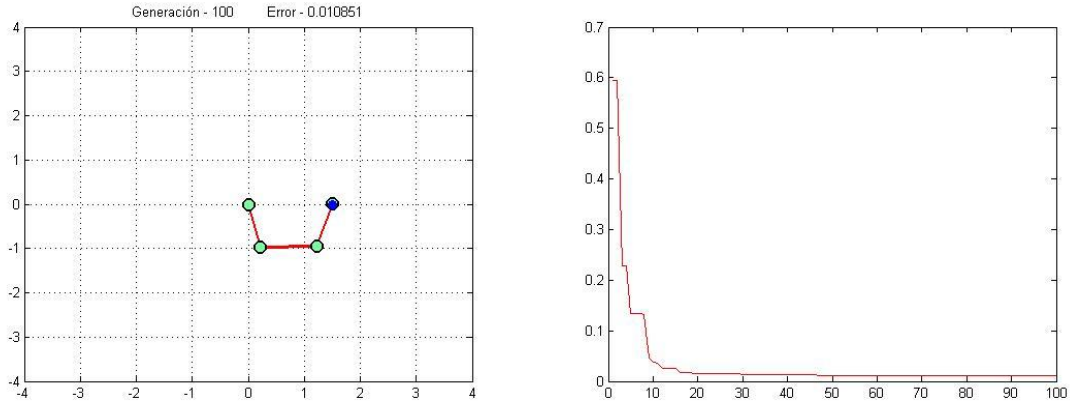
Fig. 5.7 Primera ejecución del AG para un robot 3-gdl. (a) Generación 1, error desconocido; (b) Generación 5, error=0.2588; (c) Generación 10, error=0.0840; (d) Generación 50, error=0.0177; (e) Generación 100, error=0.0117; (f) Gráfica del error menor en cada generación.

En la fig. 5.7 se puede ver el resultado arrojado por el algoritmo, así como la gráfica del error mínimo en cada generación. La mejor configuración que se obtuvo fue:

Q_{rad}=[1.1290 -1.0431 -1.5708]
Q_{grad}=[64.6848 -59.7656 -90.0000]

Con un error de posición: 0.0117

- **Segunda Ejecución**



(a) (b)

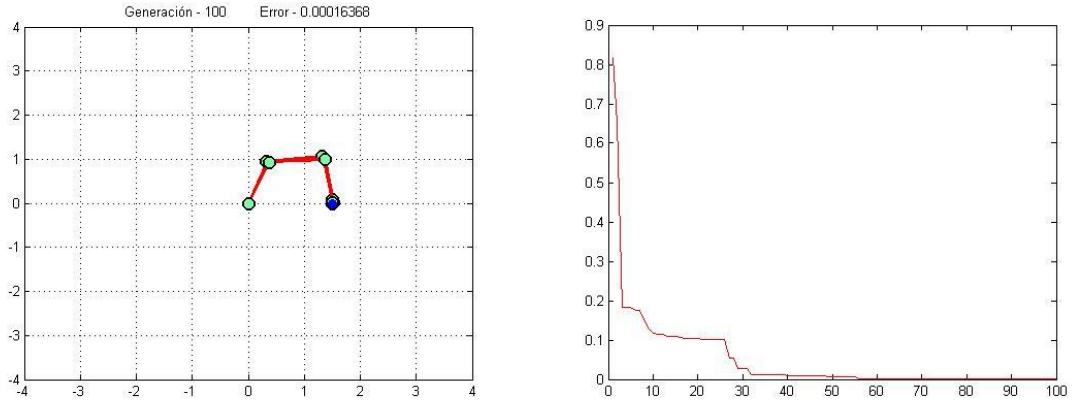
Fig. 5.8 Resultados de la segunda ejecución del AG para un robot 3-gdl. (a) Generación 100, error=0.0108; (b) Gráfica del error menor en cada generación.

La fig. 5.8 muestra los resultados obtenidos en esta prueba. La mejor configuración que se obtuvo fue:

Q_{rad}=[-1.3499 1.3744 1.2517]
Q_{grad}=[-77.3410 78.7445 71.7160]

Con un error de posición: 0.0108

- **Tercera Ejecución**



(a) (b)

Fig. 5.9 Resultados de la tercera ejecución del AG para un robot 3-gdl. (a) Generación 100, error=0.0001; (b) Gráfica del error menor en cada generación.

La fig. 5.9 muestra los resultados obtenidos en esta prueba. La mejor configuración que se obtuvo fue:

Q_{rad}=[1.1846 -1.1187 -1.5110]
Q_{grad}=[67.8708 -64.0970 -86.5750]

Con un error de posición: 0.0001

5.3.2.2 Prueba 2: Robot 6-gdl

En este experimento se pidió al programa generar la cinemática inversa para un manipulador de 6-gdl. El punto destino fue $P_{deseado}$=(3, 3).

Como se puede observar, el número de bits empleado para describir cada individuo (configuración) es de $16 * gdl$, por lo tanto la longitud de cada cromosoma será de $16 * 6 = 96bits$.

Lo que se puede observar en las imágenes es la población de individuos (configuraciones de robots) en esa generación, y con un punto azul la posición meta.

- **Primera Ejecución**

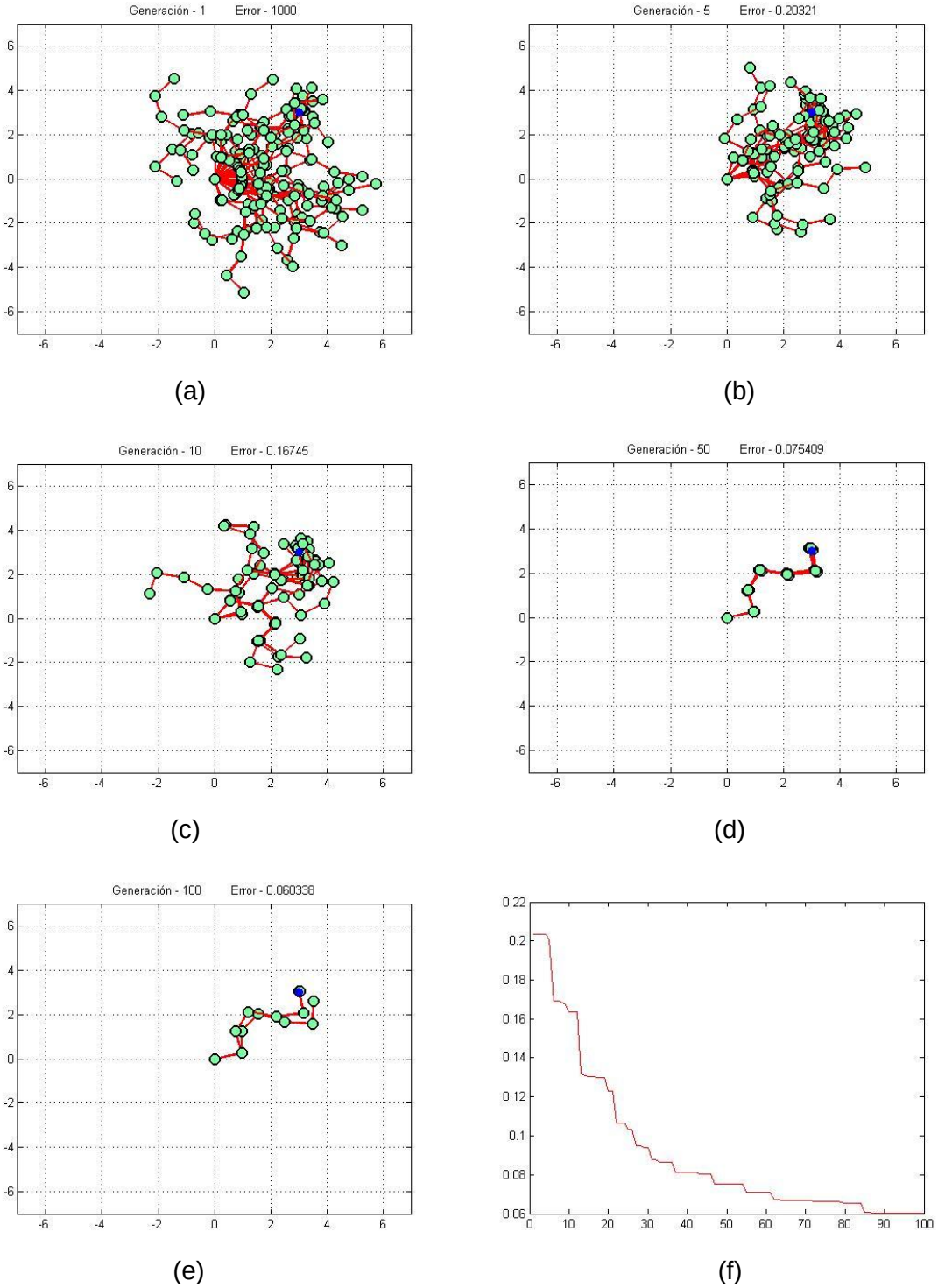


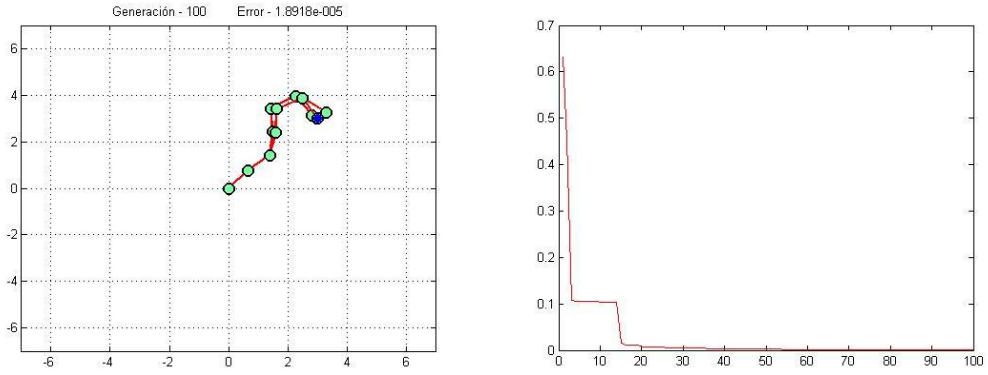
Fig. 5.10 Primera ejecución del AG para un robot 6-gdl. (a) Generación 1, error desconocido; (b) Generación 5, error=0.2032; (c) Generación 10, error=0.1674; (d) Generación 50, error=0.0754; (e) Generación 100, error=0.0603; (f) Gráfica del error menor en cada generación.

En la fig. 5.10 se puede ver el resultado arrojado por el algoritmo, así como la gráfica del error mínimo en cada generación. La mejor configuración que se obtuvo fue:

Q_{rad}=[0.2516 1.5159 -0.6581 -1.3253 0.3604 1.5707]
Q_{grad}=[14.4168 86.8552 -37.7051 -75.9348 20.6516 89.9918]

Con un error de posición: 0.0603

- **Segunda Ejecución**



(a) (b)

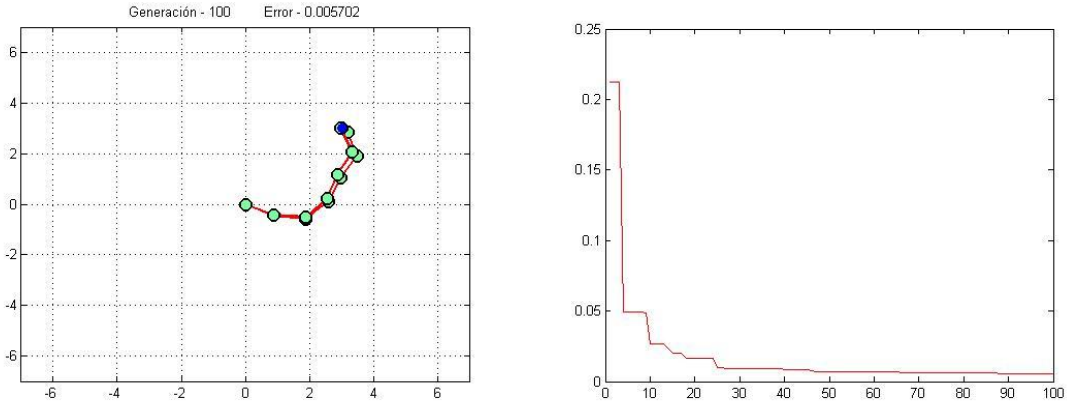
Fig. 5.11 Resultados de la segunda ejecución del AG para un robot 6-gdl. (a) Generación 100, error=1.8918e-5; (b) Gráfica del error menor en cada generación.

La fig. 5.11 muestra los resultados obtenidos en esta prueba. La mejor configuración que se obtuvo fue:

Q_{rad}=[0.8657 -0.1339 0.6280 0.1881 -1.0650 -1.5417]
Q_{grad}=[49.6005 -7.6712 35.9830 10.7776 -61.0181 -88.3301]

Con un error de posición: 1.8918e-5

- **Tercera Ejecución**



(a) (b)

Fig. 5.12 Resultados de la tercera ejecución del AG para un robot 6-gdl. (a) Generación 100, error=0.0057; (b) Gráfica del error menor en cada generación.

La fig. 5.12 muestra los resultados obtenidos en esta prueba. La mejor configuración que se obtuvo fue:

Q_{rad}=[-0.4512 0.3581 0.9326 0.4042 -0.1437 0.8165]
Q_{grad}=[-25.8536 20.5197 53.4348 23.1564 -8.2343 46.7798]

Con un error: 0.0057

5.3.2.3 Prueba 3: Robot 15-gdl

En este experimento se pidió al programa generar la cinemática inversa para un manipulador de 15-gdl. El punto destino fue $P_{deseado}$=(3, -7).

Como se puede observar, el número de bits empleado para describir cada individuo (configuración) es de $16 * gdl$, por lo tanto la longitud de cada cromosoma será de $16 * 15 = 240 bits$.

Lo que se puede observar en las imágenes es la población de individuos (configuraciones de robots) en esa generación, y con un punto azul la posición meta.

- **Primera Ejecución**

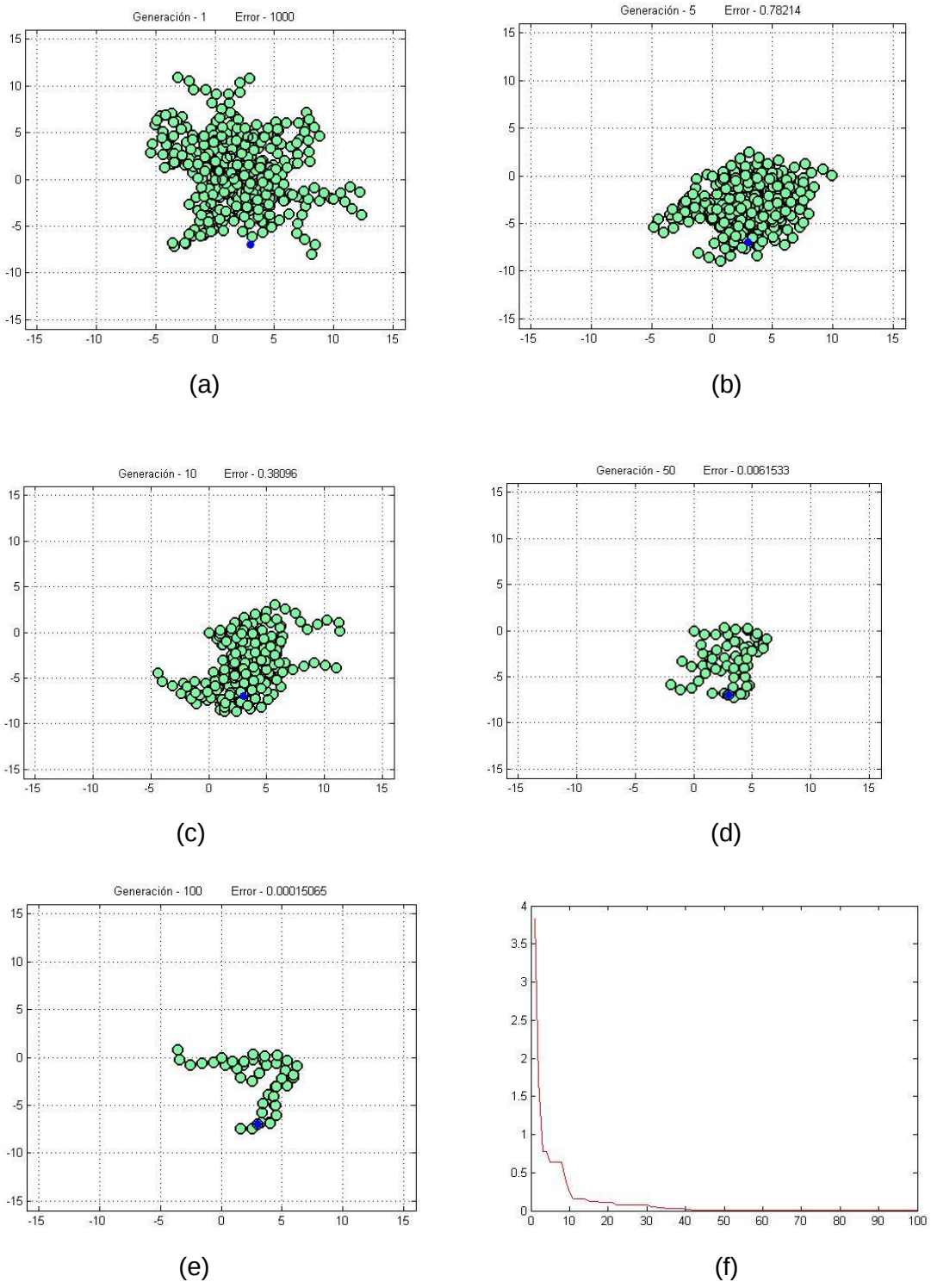


Fig. 5.13 Primera ejecución del AG para un robot 15-gdl. (a) Generación 1, error desconocido; (b) Generación 5, error=0.7821; (c) Generación 10, error=0.3809; (d) Generación 50, error=0.0061; (e) Generación 100, error=0.0001; (f) Gráfica del error menor en cada generación.

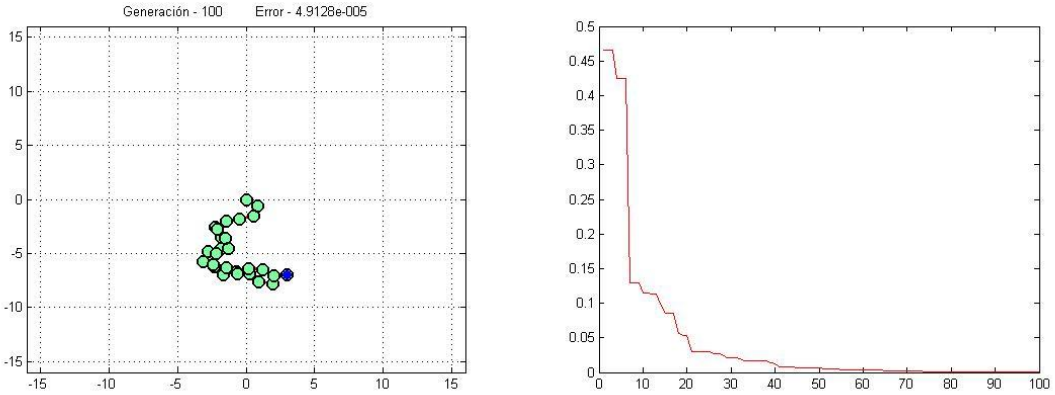
En la fig. 5.13 se puede ver el resultado arrojado por el algoritmo, así como la gráfica del error mínimo en cada generación. La mejor configuración que se obtuvo fue:

Q_{rad}=[-0.4173 0.4088 0.7957 -0.9089 0.2055 -0.7022 -0.0227 -1.2479 -0.9613 0.8718 0.1290 0.4208 0.0001 -0.7608 -0.7834]

Q_{grad}=[-23.9117 23.4229 45.5905 -52.0779 11.7746 -40.2319 -1.3019 -71.4990 -55.0800 49.9493 7.3911 24.1095 0.0055 -43.5910 -44.8846]

Con un error de posición: 0.0001

- **Segunda Ejecución**



(a) (b)

Fig. 5.14 Resultados de la segunda ejecución del AG para un robot 15-gdl. (a) Generación 100, error=4.9128e-5; (b) Gráfica del error menor en cada generación.

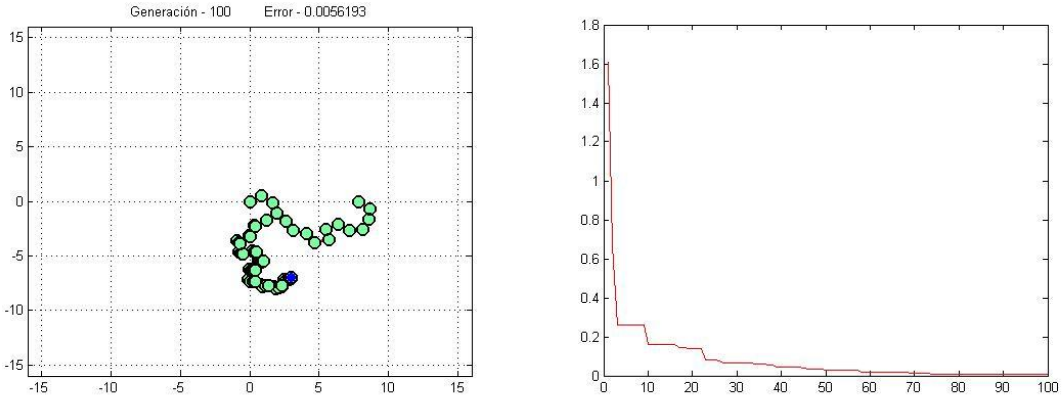
La fig. 5.14 muestra los resultados obtenidos en esta prueba. La mejor configuración que se obtuvo fue:

Q_{rad}=[-0.6173 -1.2565 -1.0109 -0.0501 0.6464 1.3344 -0.3975 -1.2766 0.8177 1.4993 -0.3475 1.1843 -0.6005 -0.5706 0.7394]

Q_{grad}=[-35.3705 -71.9907 -57.9227 -2.8702 37.0349 76.4539 -22.7774 -73.1442 46.8512 85.9021 -19.9127 67.8571 -34.4064 -32.6926 42.3633]

Con un error de posición: 4.9128e-5

- **Tercera Ejecución**



(a) (b)

Fig. 5.15 Resultados de la tercera ejecución del AG para un robot 15-gdl. (a) Generación 100, error=0.0056; (b) Gráfica del error menor en cada generación.

La fig. 5.15 muestra los resultados obtenidos en esta prueba. La mejor configuración que se obtuvo fue:

Q_{rad}=[0.4972 -1.1259 -0.6745 -1.0630 -0.2389 0.7216 -0.6166 1.1767 1.5213 -1.2824 -1.1091 0.6683 1.1412 0.3066 0.9320]

Q_{grad}=[28.4875 -64.5117 -38.6472 -60.9055 -13.6890 41.3470 -35.3293 67.4203 87.1655 -73.4766 -63.5477 38.2901 65.3879 17.5644 53.3990]

Con un error de posición: 0.0056

5.4 Segunda Versión – AG modificado I

5.4.1 Descripción

Ya con la experiencia de la primera versión de AG, se pudo comprobar su efectividad para resolver la cinemática inversa de posición. La preocupación ahora sería la de minimizar el cambio articular encontrando soluciones que se 'parezcan' a la configuración inicial. Esto es deseable cuando estos puntos contiguos forman parte de una trayectoria, es importante que el movimiento entre un punto y otro sea lo menos brusco posible.

De esta manera se propuso una segunda versión de AG para atacar estos problemas, se realizaron dos modificaciones significativas con respecto al anterior AG:

- Se toma en cuenta la pose inicial del manipulador para definir la población inicial.

- Mutación ponderada.

Los parámetros usados fueron:

- $tamaño\ del\ cromosoma: 16 * gdl$
- $tamaño\ de\ la\ población: 100$
- $generaciones = 100$
- $cruza\ no\ convencional$
- $p_m = 0.0625$ → mutación ponderada

5.4.1.1 Definición de la Población Inicial

En el AG simple [6] se proponía inicializar la población aleatoriamente, esto para hacer una exploración uniforme del espacio de búsqueda.

Sin embargo, a la hora de buscar una configuración que resuelva la cinemática inversa, en la mayoría de los casos esta solución no es única. Entonces, si se pudiera elegir cuál configuración es la que más conviene, esta sería aquella que se encuentre más cerca de la configuración inicial. Esto porque la trayectoria (línea recta) entre ambas será la que se realice con el menor movimiento articular.

Entonces, antes de realizar un exploración aleatoria por todo el espacio de configuración (población inicial aleatoria), sería deseable explorar en las cercanías de la configuración inicial, por si existe una solución de cinemática inversa cercana a la configuración inicial.

Esto se ha logrado redefiniendo la población inicial, y en lugar de generarla aleatoriamente, se ha igualado con la configuración inicial (a la que se quiere sea similar la solución). Inicialmente todos los individuos son el mismo (clones), y este individuo es la configuración inicial.

¿Qué consecuencias tiene esto? Ya no se está explorando el espacio de aptitud uniformemente, y la cruza pierde efectividad; dos individuos con cromosomas semejantes o idénticos no producirán un hijo muy diferente.

En esta ocasión será la mutación la que se encargará de producir individuos nuevos y diferentes, en estas circunstancias iniciales es necesario elevar el coeficiente de mutación. Como la mutación casi no modifica los bits del cromosoma, este permanece con alteraciones muy pequeñas. De esta manera, con estos pequeños cambios a la configuración inicial, causadas por la mutación, se lograría explorar los alrededores de esta y encontrar una configuración cercana que resuelva la cinemática inversa (fig. 5.16).

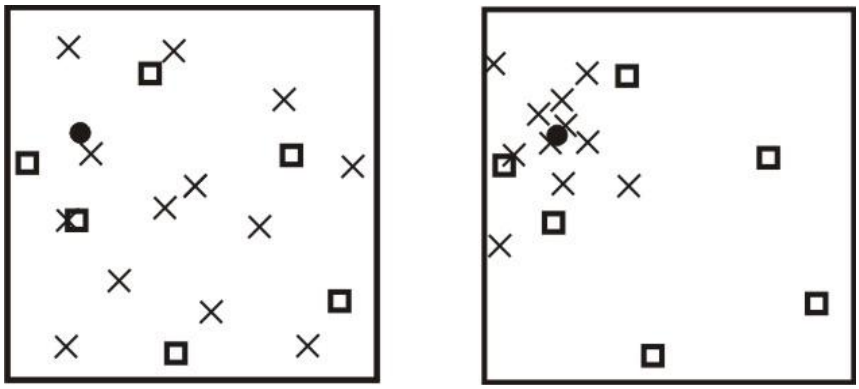


Fig. 5.16 Espacio de configuración hipotético (a) Uniformemente explorado y (b) explorado alrededor de la configuración inicial.

No obstante la idea funciona, tiene un problema: La probabilidad para cambiar un bit menos significativo es la misma que para cambiar un bit más significativo. El cromosoma está compuesto por un vector binario, con cada entrada representando una variable articular. El problema de mutar indistintamente los bits más significativos es que pueden generarse movimientos muy bruscos en algunas articulaciones. Lo que sería más adecuado es mutar con mayor probabilidad los bits menos significativos que los bits más significativos del cromosoma.

5.4.1.2 Mutación Ponderada

Ahora lo que se busca es una mutación ponderada dentro del cromosoma. Sabemos que éste está compuesto por un vector binario, como q=[0010, 1001, 1011]. Dentro de este vector, cada casilla representa una variable articular. Es en cada una de estas casillas donde se ponderará la mutación.

Supóngase que se tiene una cadena binaria con n bits que representa una variable articular (una celda del vector cromosoma), de la siguiente manera:

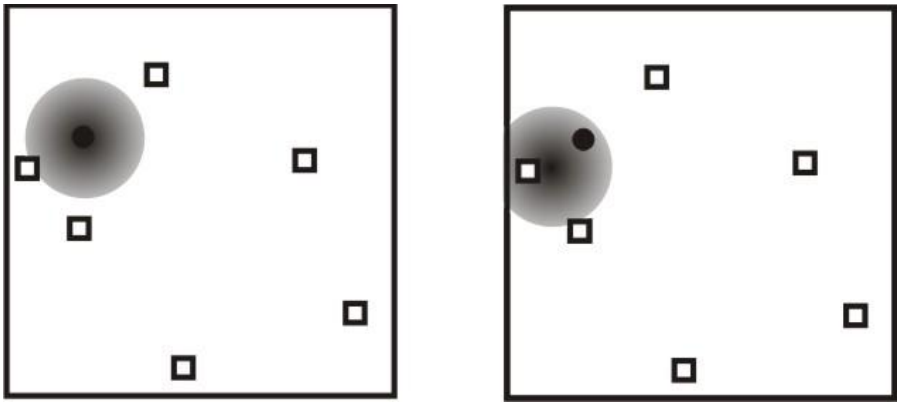
$$Cfg=[b_1\ b_2\ b_3\ \ldots b_i \ldots\ b_{n-1}\ b_n]$$

También se tiene que p_m es la probabilidad de mutar cada bit.Lo que se hace es asignar una nueva probabilidad en función a su posición, de la siguiente manera:

$$p\_m_i = p\_m * \frac{2^i}{2^n}$$

Siendo p_m_i la probabilidad de mutar el bit i. Hay que mencionar que es necesario aumentar el coeficiente de mutación para que se produzca la exploración del espacio de búsqueda dentro de un rango aceptable.

Algo que se logra con esto es que la población se sitúa alrededor de su mejor individuo. Esto a su vez produce que se sigan explorando los alrededores para seguir mejorando la población. Ver fig. 5.17



(a) (b)

Fig. 5.17 Espacio de configuración hipotético (a) La población inicial muta alrededor de la configuración inicial, con cambios pequeños más frecuentes; y (b) la población se sitúa alrededor de su mejor individuo, y va moviéndose hacia la primera configuración óptima (resuelve cinemática inversa) que encuentre en los alrededores, esta no debe ser necesariamente la más cercana.

5.4.2 Experimentos

Con esta segunda versión del AG se realizaron los mismos experimentos que para la versión anterior, sólo que ahora hubo que definir una configuración inicial del manipulador, esta se generó aleatoriamente, y se utilizo en las tres ejecuciones del AG para manipuladores con 3, 6 y 15 articulaciones. Aquí se muestran los resultados arrojados en la ejecución del AG.

5.4.2.1 Prueba 1: Robot 3-gdl

En este experimento se pidió al programa generar la cinemática inversa para un manipulador de 3-gdl. El punto destino fue $P_{deseado}$=(0.5, -2). La longitud de cada cromosoma será de $16 * 3 = 48 bits$.

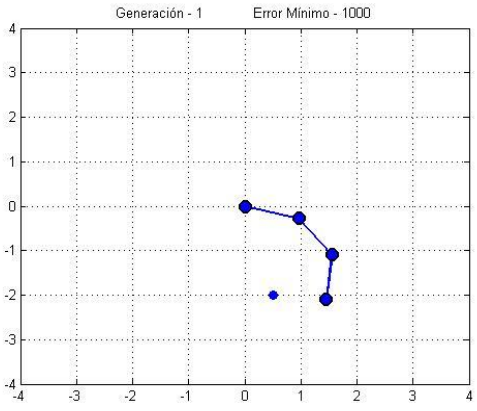
La configuración inicial que se usó fue:

Q_0_{rad}=[-0.2844 -0.6568 -0.7295]

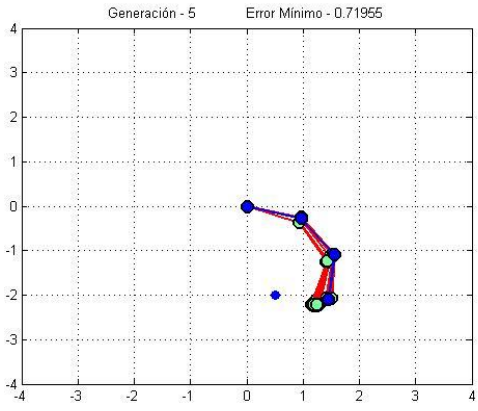
Q_0_{grad}=[-16.2960 -37.6292 -41.7951]

Lo que se puede observar en las imágenes es la población de individuos (configuraciones de robots) en esa generación, y con un punto azul la posición meta.

- **Primera Ejecución**



(a)



(b)

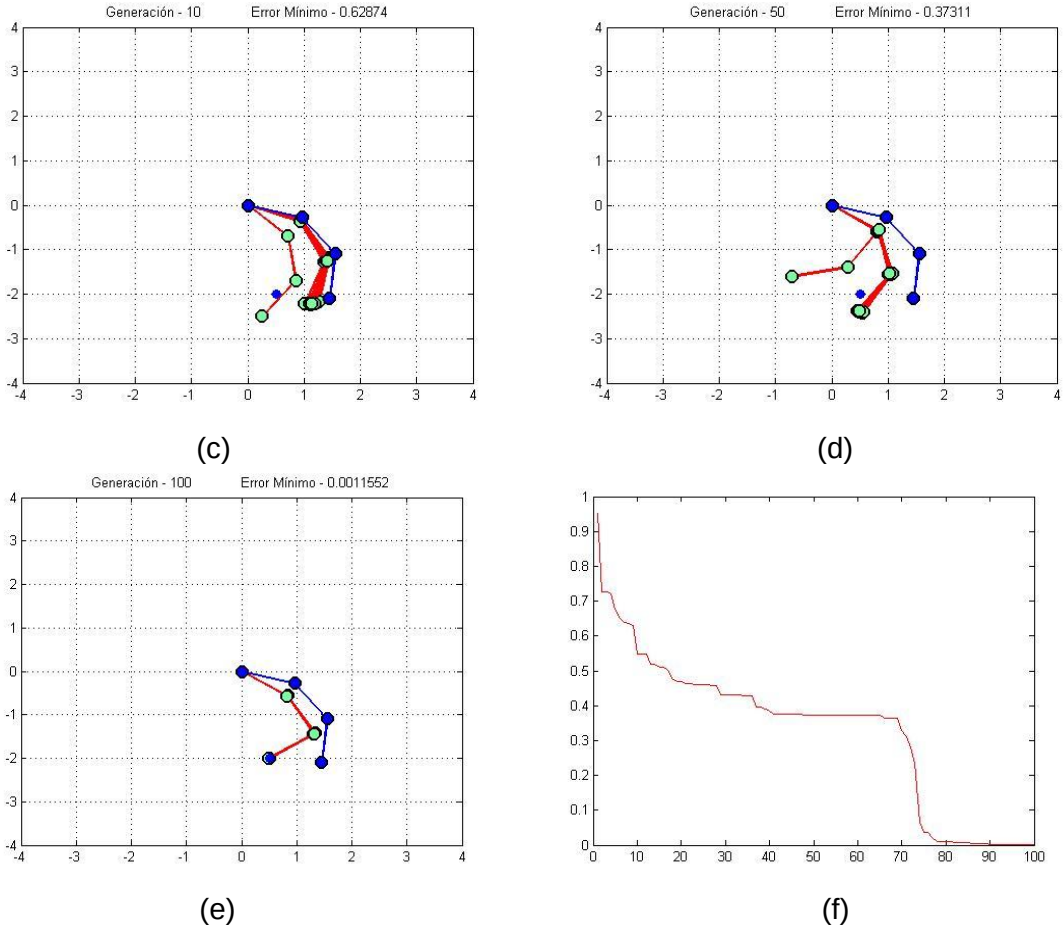


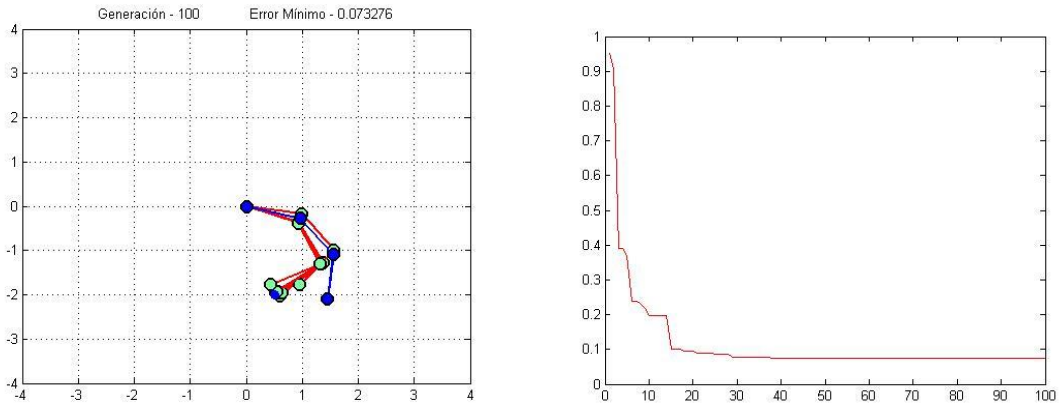
Fig. 5.18 Primera ejecución del AG modificado para un robot 3-gdl. (a) Generación 1, error desconocido; (b) Generación 5, error=0.7195; (c) Generación 10, error=0.6287; (d) Generación 50, error=0.3731; (e) Generación 100, error=0.0011; (f) Gráfica del error menor en cada generación.

En la fig. 5.18 se puede ver el resultado arrojado por el algoritmo, así como la gráfica del error mínimo en cada generación. La mejor configuración que se obtuvo fue:

Q_{rad}=[-0.5923 -0.4636 -1.4784]
Q_{grad}=[-33.9368 -26.5622 -84.7046]

Con un error de posición: 0.0011

- **Segunda Ejecución**



(a) (b)

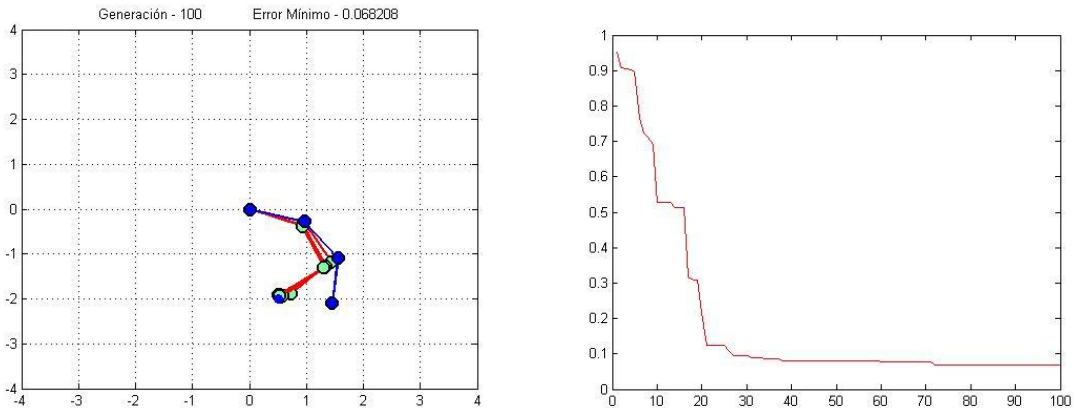
Fig. 5.19 Resultados de la segunda ejecución del AG modificado para un robot 3-gdl. (a) Generación 100, error=0.0732; (b) Gráfica del error menor en cada generación.

La fig. 5.19 muestra los resultados obtenidos en esta prueba. La mejor configuración que se obtuvo fue:

Q_{rad}=[-0.3896 -0.7839 -1.2763]
Q_{grad}=[-22.3242 -44.9121 -73.1277]

Con un error de posición: 0.0732

- **Tercera Ejecución**



(a) (b)

Fig. 5.20 Resultados de la tercera ejecución del AG modificado para un robot 3-gdl. (a) Generación 100, error=0.0682; (b) Gráfica del error menor en cada generación.

La fig. 5.20 muestra los resultados obtenidos en esta prueba. La mejor configuración que se obtuvo fue:

Q_{rad}=[-0.3927 -0.7854 -1.2771]
Q_{grad}=[-22.5000 -45.0000 -73.1717]

Con un error de posición: 0.0682

5.4.2.2 Prueba 2: Robot 6-gdl

En este experimento se pidió al programa generar la cinemática inversa para un manipulador de 6-gdl. El punto destino fue $P_{deseado}$=(3, 3). La longitud de cada cromosoma será de $16 * 6 = 96\ bits$.

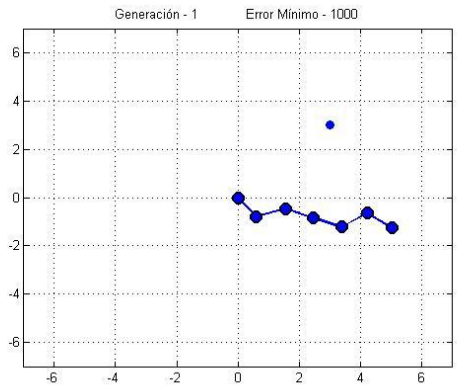
La configuración inicial que se usó fue:

Q_0_{rad}=[-0.9160 1.2499 -0.7353 0.0410 0.9533 -1.2420]

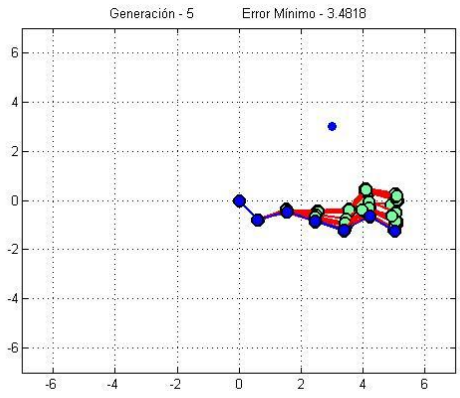
Q_0_{grad}=[-52.4853 71.6132 -42.1321 2.3512 54.6196 -71.1601]

Lo que se puede observar en las imágenes es la población de individuos (configuraciones de robots) en esa generación, y con un punto azul la posición meta.

- **Primera Ejecución**



(a)



(b)

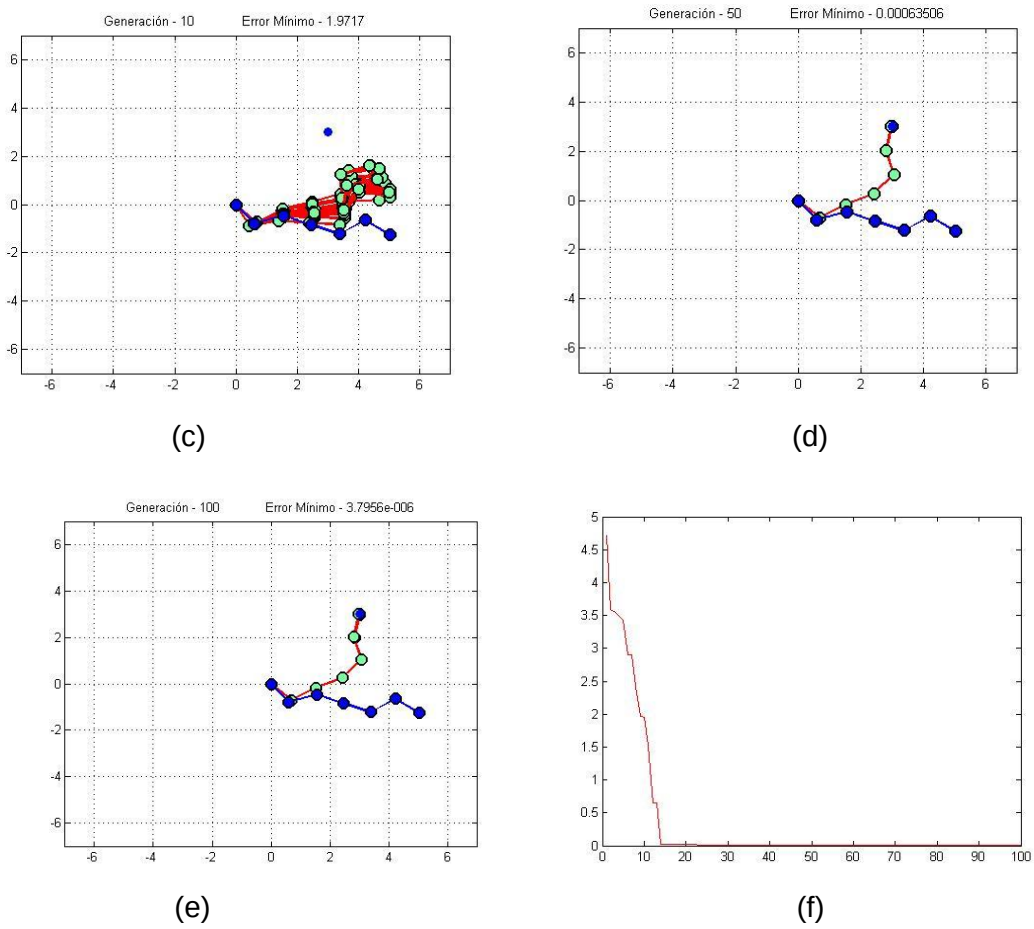


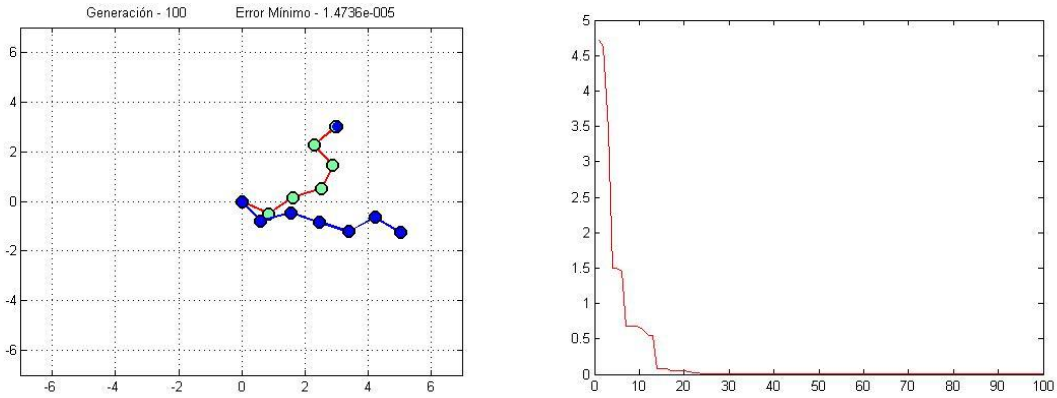
Fig. 5.21 Primera ejecución del AG modificado para un robot 6-gdl. (a) Generación 1, error desconocido; (b) Generación 5, error=3.4818; (c) Generación 10, error=1.9717; (d) Generación 50, error=0.0006; (e) Generación 100, error=3.7956e-6; (f) Gráfica del error menor en cada generación.

En la fig. 5.21 se puede ver el resultado arrojado por el algoritmo, así como la gráfica del error mínimo en cada generación. La mejor configuración que se obtuvo fue:

Q_{rad}=[-0.7933 1.3726 -0.1227 0.4216 0.9406 -0.4199]
Q_{grad}=[-45.4504 78.6429 -7.0285 24.1534 53.8934 -24.0601]

Con un error de posición: 3.7956e-6

- **Segunda Ejecución**



(a) (b)

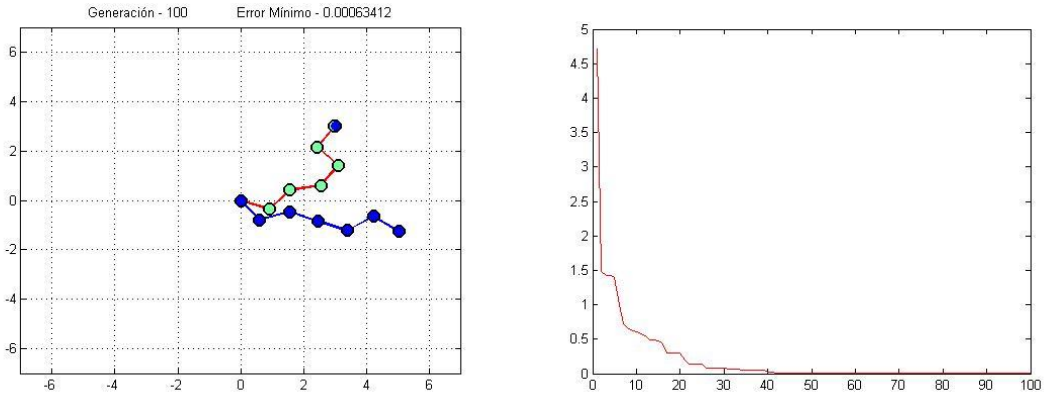
Fig. 5.22 Resultados de la segunda ejecución del AG modificado para un robot 6-gdl. (a) Generación 100, error=1.4736e-5; (b) Gráfica del error menor en cada generación.

La fig. 5.22 muestra los resultados obtenidos en esta prueba. La mejor configuración que se obtuvo fue:

Q_{rad}=[-0.5367 1.2498 -0.3305 0.8267 0.9778 -1.3652]
Q_{grad}=[-30.7507 71.6089 -18.9377 47.3676 56.0220 -78.2227]

Con un error de posición: 1.4736e-5

- **Tercera Ejecución**



(a) (b)

Fig. 5.23 Resultados de la tercera ejecución del AG modificado para un robot 6-gdl. (a) Generación 100, error=0.0006; (b) Gráfica del error menor en cada generación.

La fig. 5.23 muestra los resultados obtenidos en esta prueba. La mejor configuración que se obtuvo fue:

Q_{rad}=[-0.3636 1.2501 -0.6983 0.7762 1.3454 -1.3264]
Q_{grad}=[-20.8328 71.6226 -40.0122 44.4727 77.0856 -75.9979]

Con un error de posición: 0.0006

5.4.2.3 Prueba 3: Robo15-gdl

Finalmente se pidió al programa generar la cinemática inversa para un manipulador de 15-gdl. El punto destino fue $P_{deseado}$=(3, -7). La longitud de cada cromosoma será de $16 * 3 = 48 bits$.

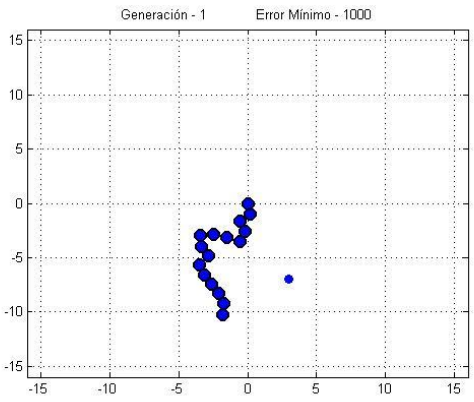
La configuración inicial que se usó fue:

Q_0_{rad}=[-1.3577 -1.0689 1.2273 -0.7344 -1.5425 0.0327 0.4127 1.5354 0.4484 -1.1896 1.0389 0.1148 0.0451 -0.1019 -0.5058]

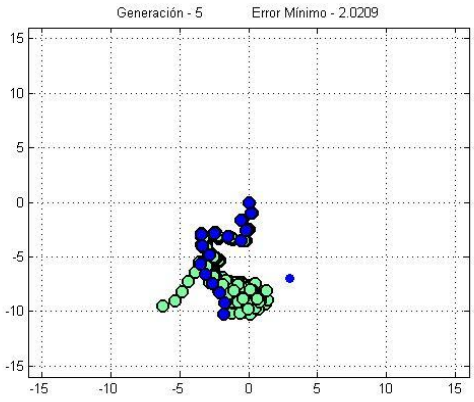
Q_0_{grad}=[-77.7926 -61.2432 70.3195 -42.0777 -88.3772 1.8719 23.6485 87.9722 25.6898 -68.1597 59.5263 6.5789 2.5850 -5.8370 -28.9776]

Lo que se puede observar en las imágenes es la población de individuos (configuraciones de robots) en esa generación, y con un punto azul la posición meta.

- **Primera Ejecución**



(a)



(b)

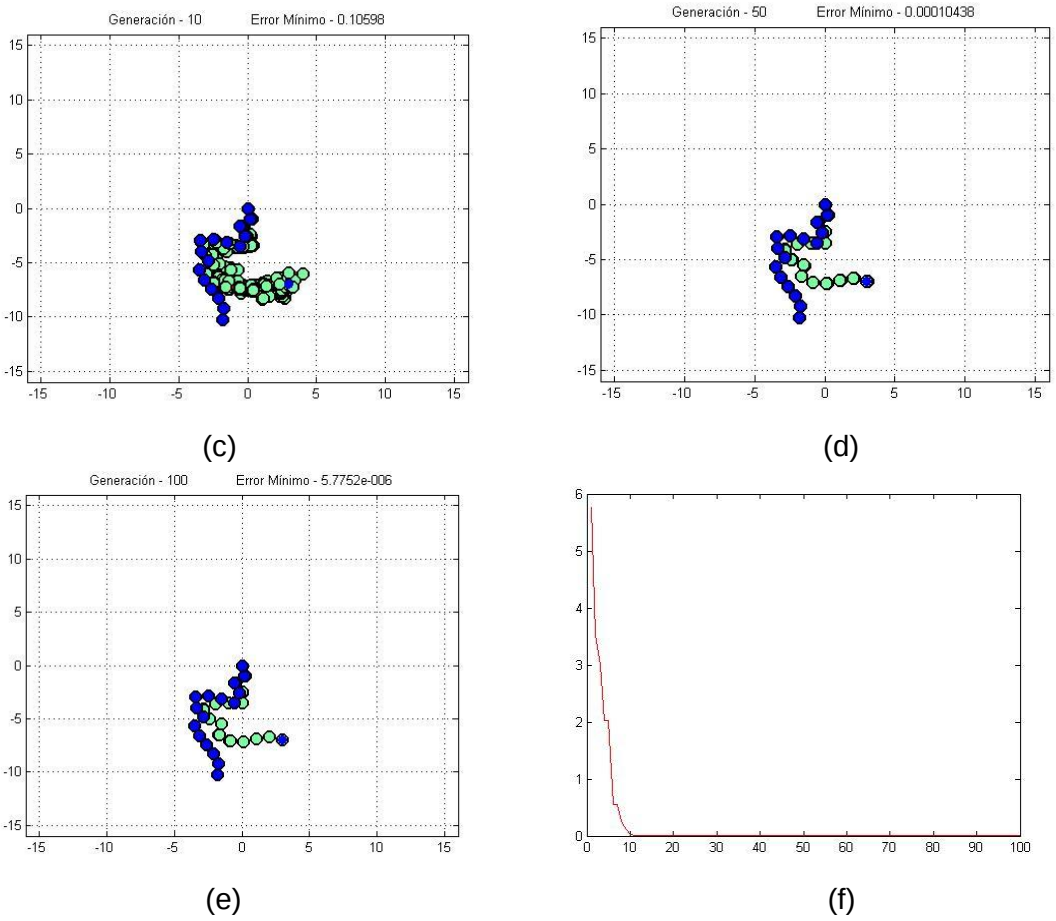


Fig. 5.24 Primera ejecución del AG modificado para un robot 15-gdl. (a) Generación 1, error desconocido; (b) Generación 5, error=2.0209; (c) Generación 10, error=0.1059; (d) Generación 50, error=0.0001; (e) Generación 100, error=5.7752e-6; (f) Gráfica del error menor en cada generación.

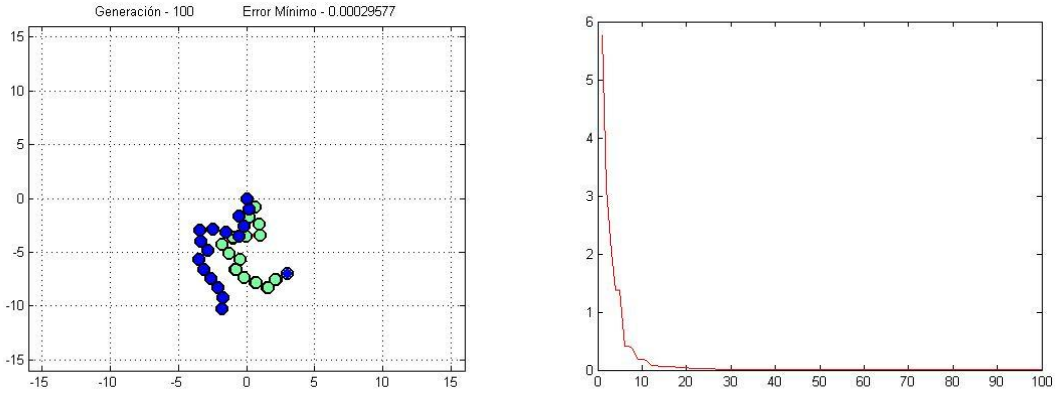
En la fig. 5.24 se puede ver el resultado arrojado por el algoritmo, así como la gráfica del error mínimo en cada generación. La mejor configuración que se obtuvo fue:

Q_{rad}=[-1.3088 -1.0689 1.3254 -0.5136 -1.5677 0.0826 0.4252 1.5355 0.5591 -1.2267 1.1371 0.4952 0.4378 -0.1139 -0.5057]

Q_{grad}=[-74.9872 -61.2460 75.9375 -29.4269 -89.8242 4.7324 24.3594 87.9758 32.0334 -70.2850 65.1517 28.3722 25.0845 -6.5286 -28.9764]

Con un error de posición: 5.7752e-6

- **Segunda Ejecución**



(a) (b)

Fig. 5.25 Resultados de la segunda ejecución del AG modificado para un robot 15-gdl. (a) Generación 100, error=0.0002; (b) Gráfica del error menor en cada generación.

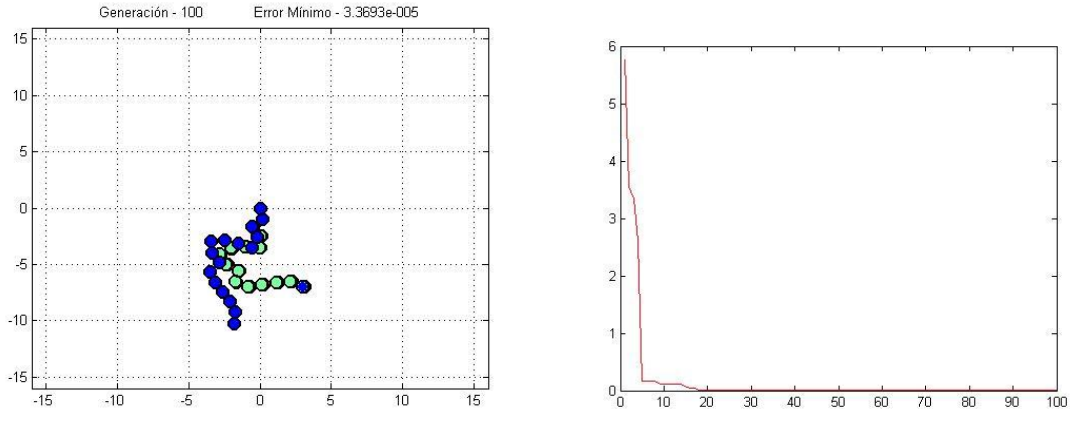
La fig. 5.25 muestra los resultados obtenidos en esta prueba. La mejor configuración que se obtuvo fue:

Q_{rad}=[-0.9399 -1.0508 1.2524 -0.7713 -1.5430 0.0803 0.4383 1.5354 0.3990 -1.3375 1.0376 0.3602 0.0571 1.4320 -0.4199]

Q_{grad}=[-53.8495 -60.2051 71.7572 -44.1925 -88.4097 4.6033 25.1120 87.9703 22.8625 -76.6324 59.4525 20.6378 3.2739 82.0459 -24.0601]

Con un error de posición: 0.0002

- **Tercera Ejecución**



(a) (b)

Fig. 5.26 Resultados de la tercera ejecución del AG modificado para un robot 15-gdl. (a) Generación 100, error=3.3693e-5; (b) Gráfica del error menor en cada generación.

La fig. 5.26 muestra los resultados obtenidos en esta prueba. La mejor configuración que se obtuvo fue:

Q_{rad}=[-1.3689 -1.0075 1.4235 -0.7222 -1.5663 0.2660 0.4375 1.4739 0.4609
-1.1900 1.3831 0.5685 0.0204 -0.1016 -0.5795]

Q_{grad}=[-78.4314 -57.7277 81.5625 -41.3800 -89.7446 15.2380 25.0653 84.4464 26.4056 -68.1839 79.2471 32.5745 1.1700 -5.8200 -33.2034]

Con un error de posición: 3.3693e-5

5.5 Tercera Versión – AG modificado II

5.5.1 Descripción

Finalmente se ha elaborado esta versión del AG con el objetivo de obtener la cinemática completa (posición y orientación) del robot, y al mismo tratar de minimizar el cambio articular.

Los puntos más importantes a tomar en cuenta son:

- La población se inicializa con el estado inicial del robot.
- En la función de aptitud se toman en cuenta los errores de posición, orientación y diferencia articular.
- La cruza es simple, con $p_c = 0.8$
- La mutación vuelve a ser simple ($p_m = 0.01$), con el fin de no sesgar la exploración del espacio.

Los parámetros usados son:

- $tamaño\ del\ cromosoma: 16 * gdl$
- $tamaño\ de\ la\ población: 51$
- $generaciones: 100$
- $p_c = 0.8$
- $p_m = 0.01$

5.5.1.1 Función de Aptitud

Esta vez se buscó resolver el problema de la manera más íntegra posible, tomando en cuenta tres términos del problema que nos interesa minimizar:

1) **Error de posición (e_{pos})**

   Calculado con ecuación de la distancia entre dos puntos (fórmula pitagórica):

$$e_{pos} = \sqrt{(x_{efector} - x_{deseada})^2 + (y_{efector} - y_{deseada})^2}$$

2) **Error de Orientación (e_{or})**

   Calculado como el valor absoluto de la diferencia entre la orientación deseada y la del efector en el individuo.

$$e_{or} = abs(o_{efector} - o_{deseada})$$

3) **Cambio articular mínimo (e_{art})**

   Calculado como la distancia 'pitagórica' entre la configuración inicial y la final.

$$e_{art} = \sqrt{\sum_{i=1}^{gdl} (q_{i0} - q_i)^2}$$

Al minimizar estos errores aproximamos a la solución óptima, que se da cuando estos los dos primeros errores (posición y orientación) son cero, y cuando error articular es lo más pequeño posible. Es importante señalar que los valores de estas tres variables siempre son positivos.

En esta versión se replanteó la función de aptitud debido la otra permitía añadir fácilmente los otros términos a minimizar. De esta manera la función quedó planteada de la siguiente manera:

$$EVA_i = \mathrm{k}_1 \exp(-c_1 * e_{pos}) + k_2 \exp(-c_2 * e_{or}) + k_3 \exp(-c_3 * e_{art})$$

Esta función es la suma de tres exponenciales muy similares. Los coeficientes c_1, c_2 y c_3 números positivos mayores que uno y regulan la influencia (en aptitud) que tiene cada error; los valores posibles de esta función están acotados en $[0, k_1 + k_2 + k_3]$. De esta manera, esta función incrementa su valor cuando los tres errores se van haciendo más pequeños, y alcanza su máximo $(k_1 + k_2 + k_3)$ cuando son cero, sin embargo esto no sucede porque err_{art} nunca es cero[7].

La analogía fue la siguiente, primero hay que ver cómo se comporta una exponencial de la forma $f_x = e^{-x}$

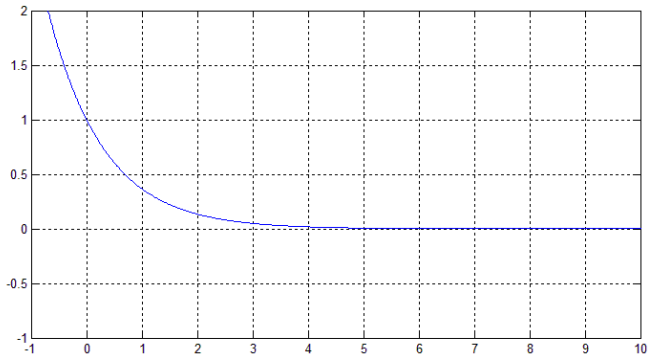
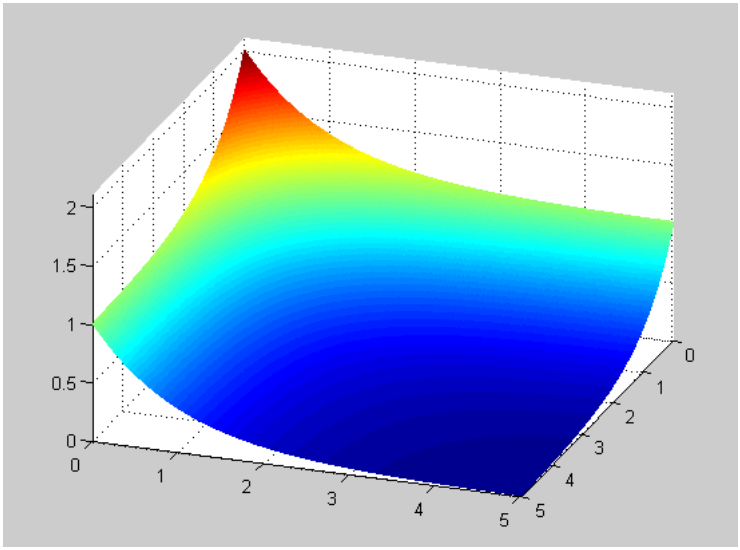


Fig. 5.27 Gráfica de $f_x = e^{-x}$ en el intervalo x=[-1,10].

Esta función crece cuando *x→0*, para valores positivos (los errores calculados nunca son negativos). Siguiendo la analogía, si tenemos $f_{xy} = e^{-x} + e^{-y}$, esta función crece cuando *(x, y) → (0, 0),* fig. 5.28.



5.28 Gráfica de $f_x = e^{-x} + e^{-y}$ en el intervalo x,y=[0,5].

[7] Solo es cero cuando la solución obtenida por el AG y la inicial son iguales (no hay cambio articular).

Con la función de aptitud propuesta de esta manera se logra acotar los valores de aptitud posibles y maximizar este valor cuando los errores se minimicen.

Hay que señalar en la realidad la función de aptitud no es tan suave, y puede presentar máximos y mínimos locales. Sin embargo, el AG es una buena herramienta para encontrar una solución satisfactoria dentro de un rango de error aceptable; tal vez no la mejor solución posible, pero si una buena aproximación.

5.5.1.2 Operadores Genéticos Simples (Cruza y Mutación)

Se han utilizado formas simples de los operadores de cruza y mutación.

La cruza se realiza escogiendo dos padres, y cruzándolos con probabilidad p_c, es decir, pueden cruzarse y generar hijos, o pasar a la siguiente generación. El coeficiente de cruza que se ha elegido es $p_c = 0.8$.

La mutación se realiza de la manera convencional (sin ponderación), ya que por un lado de esta manera se garantiza una buena exploración en el espacio de aptitud, y también se agiliza el AG. Con la mutación ponderada se sesga la exploración y el AG tarda más en converger a una solución aceptable, si no es que no llega a converger (si los parámetros no están bien equilibrados).

5.5.1.3 Minimización del cambio articular

Un objetivo importante también es minimizar el cambio articular. Lo primordial es resolver la cinemática inversa, sin embargo, es deseable que la pose final del manipulador se parezca a la anterior. Esto se ha manejado de dos maneras:

1) Función de Aptitud.- El cambio articular se contempla (con menor impacto que los errores de orientación y posición) en la función de aptitud. De esta manera, cuando la población se desplaza sobre el espacio de aptitud, o algún individuo alcanza una solución, se obtienen un ‘premio’ en la aptitud si también se minimiza el cambio articular.

2) Población inicial.- La población inicial se define idénticamente sobre la pose inicial del manipulador, esto asegura que en las primeras generaciones se exploren los alrededores con mayor eficacia, y conforme avanza la población se exploren igual otras regiones más alejadas. De esta manera no se sesga el carácter exploratorio del operador de mutación (la mutación ponderada sí lo hacía).

5.5.2 Experimentos

Con esta versión final del AG se lograron resultados satisfactorios, dentro de un margen de error aceptable. Se ha realizado la misma serie de experimentos (manipuladores de 3, 6 y 15 articulaciones), con la diferencia de que ahora también hay que definir una orientación deseada.

5.5.2.1 Prueba 1: Robot 3-gdl

En este experimento se pidió al programa generar la cinemática inversa para un manipulador de 3-gdl. La longitud de cada cromosoma será de $16 * 3 = 48bits$. El punto deseado fue:

- $posición\ deseada$: $(\ 2\ ,1.5\ )$
- $orientación\ deseada$: $45°$

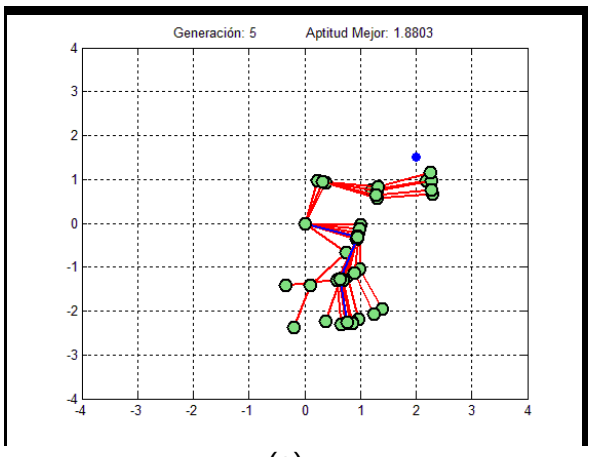
La configuración inicial que se usó fue:

Q_0_{rad}=[-0.3306 -1.5499 0.4383]

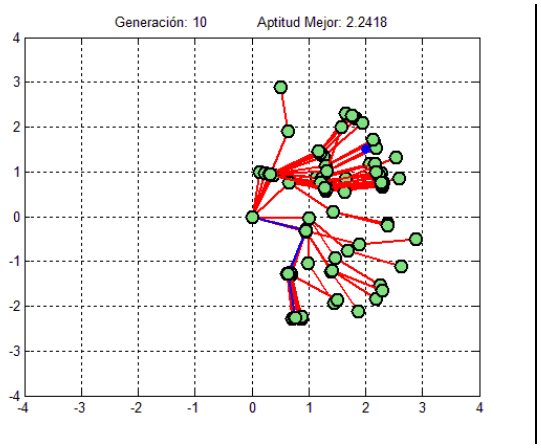
Q_0_{grad}=[-18.9432 -88.8042 25.1132]

Lo que se puede observar en las imágenes es la población de individuos (configuraciones de robots) en esa generación, y con un punto azul la posición meta.

- **Primera Ejecución**



(a)



(b)

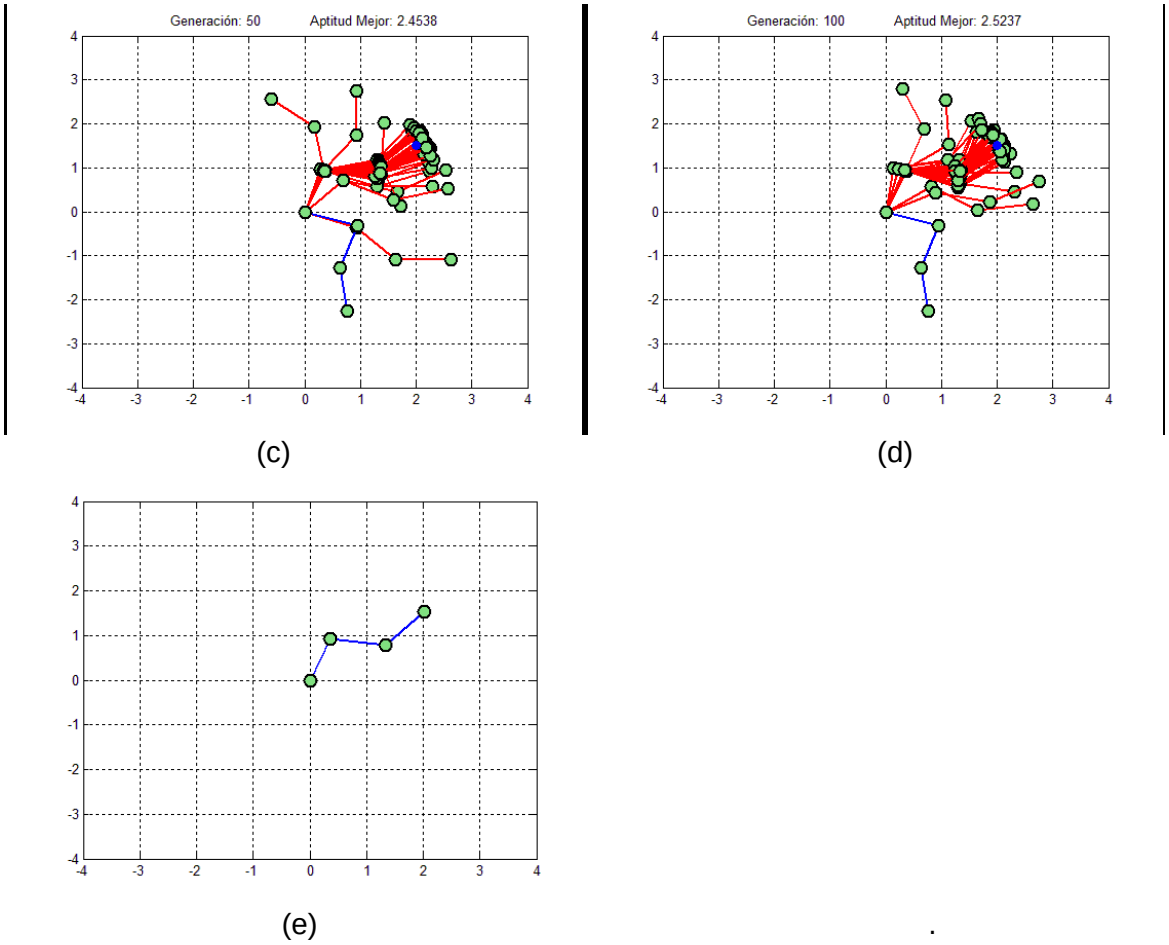


Fig. 5.29 Primera ejecución del AG (3v) para un robot 3-gdl. (a) Generación 5, mejor_aptitud=1.8803; (b) Generación 10, mejor_aptitud=2.2418; (c) Generación 50, mejor aptitud=2.4538; (d) Generación 100, mejor_aptitud=2.5237; (e) Mejor individuo obtenido

El mejor individuo obtenido fue:

Q_{rad}=[1.2100 -1.3606 0.9778]
Q_{grad}=[69.3265 -77.9590 56.0248]

La evolución del AG se muestra en la fig. 5.30:

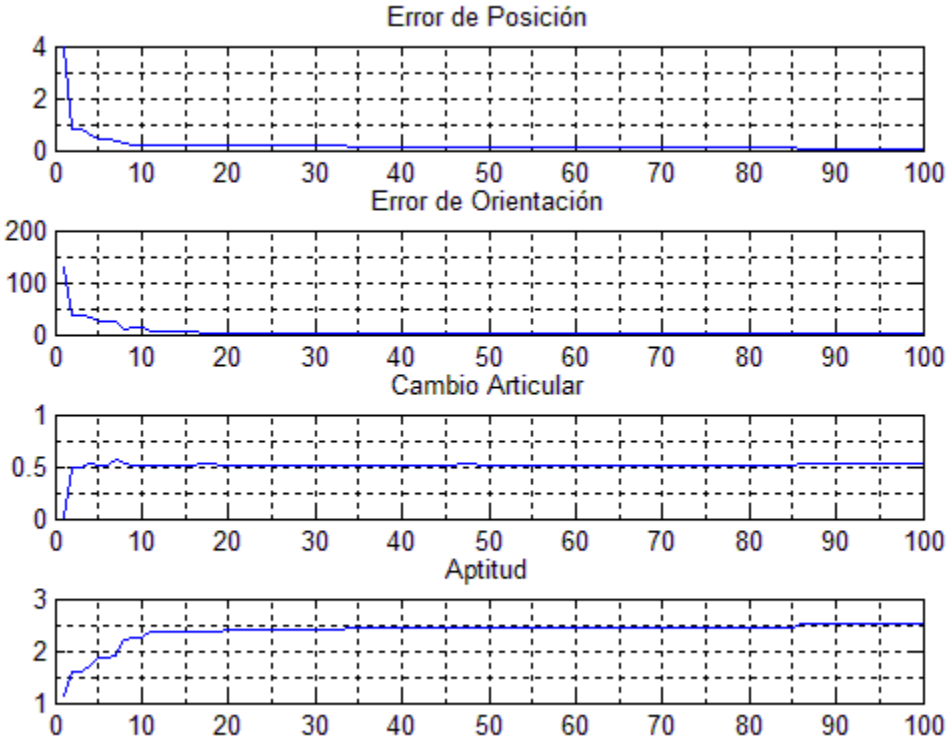


Fig. 5.30 Estado de los errores de posición, orientación, cambio articular y aptitud a través de las 100 generaciones en que se ejecutó el AG.

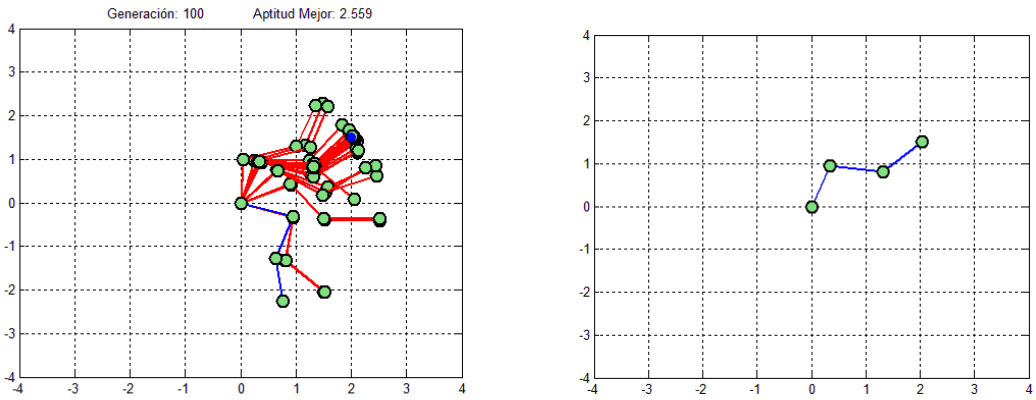
Puede observarse como a lo largo de las 100 generaciones a que se sometió la prueba el AG :

- Minimiza el error de posición (*error_final= 0.0285*).
- Minimiza el error de orientación (*error_final= 2.3923°*).
- Disminución del cambio articular.

La mejor solución obtuvo los siguientes resultados:

- $Punto\ alcanzado: (x = 2.0187, y = 1.5215)$
- $Orientación\ final: 47.3923°$

- **Segunda Ejecución**



(a) (b)

Fig. 5.31 Resultados de la segunda ejecución del AG (3v). (a) Generación 100, mejor_aptitud=2.559; (b) Mejor individuo obtenido.

El mejor individuo obtenido fue:

Q_{rad}=[1.2307 -1.3683 0.9228]
Q_{grad}=[70.5130 -78.3984 52.8717]

La evolución del AG se muestra en la fig. 5.32:

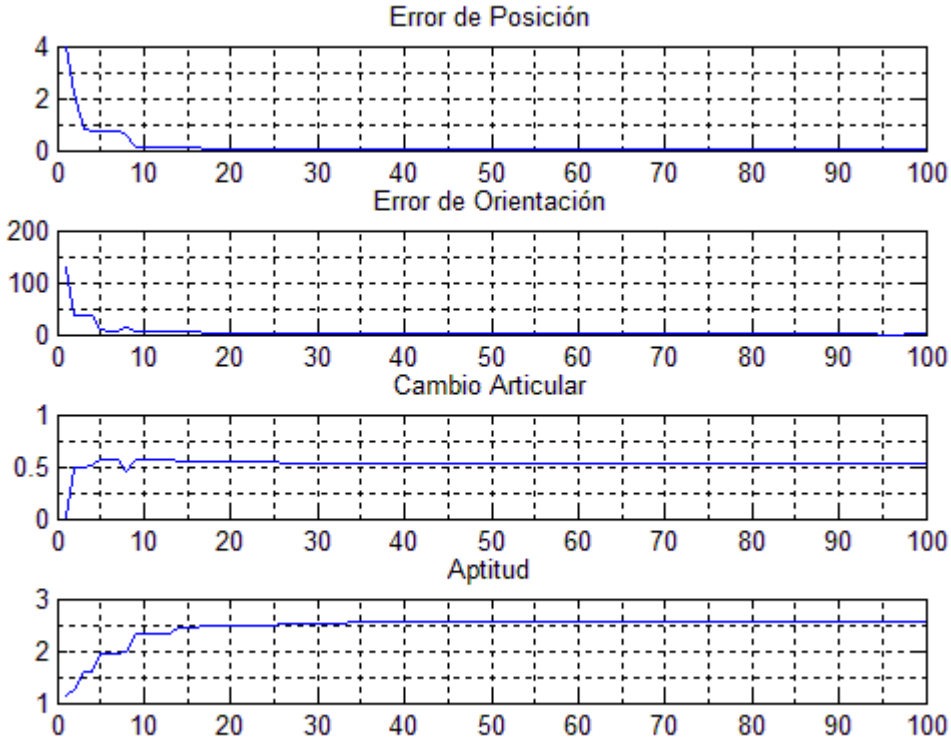


Fig. 5.32 Estado de los errores de posición, orientación, cambio articular y aptitud a través de las 100 generaciones en que se ejecutó el AG.

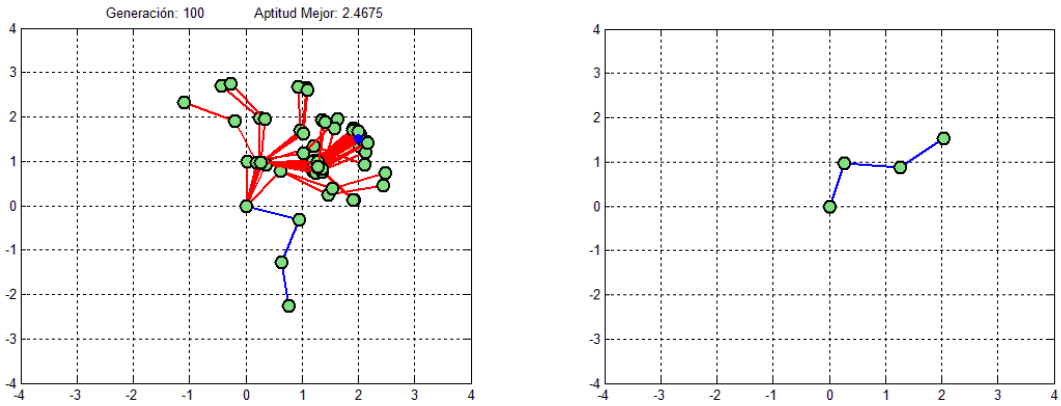
Puede observarse como a lo largo de las 100 generaciones a que se sometió la prueba el AG :

- Minimiza el error de posición (*error_final= 0.0338*).
- Minimiza el error de orientación (*error_final= 0.0137°*).
- Disminución del cambio articular.

La mejor solución obtuvo los siguientes resultados:

- $Punto\ alcanzado$: $(x = 2.0314, y = 1.5125)$
- $Orientación\ final$: $44.9863°$

- **Tercera Ejecución**



(a) (b)

Fig. 5.33 Resultados de la segunda ejecución del AG (3v). (a) Generación 100, mejor_aptitud=2.4675; (b) Mejor individuo obtenido.

El mejor individuo obtenido fue:

Q_{rad}=[1.2934 -1.3668 0.7853]
Q_{grad}=[74.1055 -78.3105 44.9945]

La evolución del AG se muestra en la 5.34:

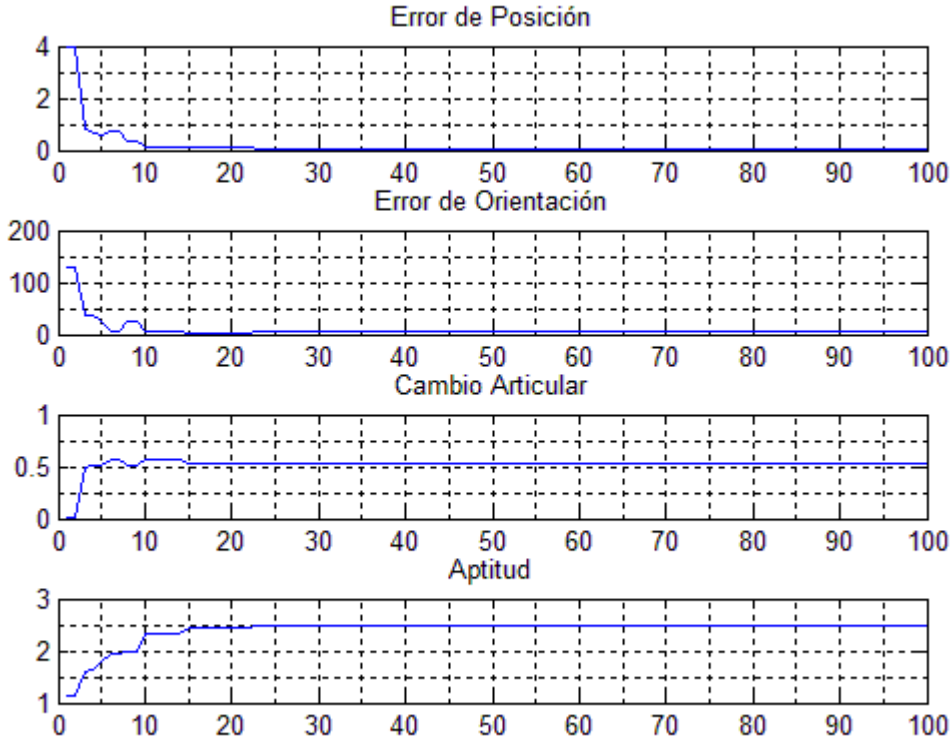


Fig. 5.34 Estado de los errores de posición, orientación, cambio articular y aptitud a través de las 100 generaciones en que se ejecutó el AG.

Puede observarse como a lo largo de las 100 generaciones a que se sometió la prueba el AG :

- Minimiza el error de posición (*error_final= 0.0504*).
- Minimiza el error de orientación (*error_final= 4.2105°*).
- Disminución del cambio articular.

La mejor solución obtuvo los siguientes resultados:

- $Punto\ alcanzado: (x = 2.0283, y = 1.5417)$
- $Orientación\ final: 40.7895°$

5.5.2.2 Prueba 2: Robot 6-gdl

En este experimento se pidió al programa generar la cinemática inversa para un manipulador de 6-gdl. La longitud de cada cromosoma será de $16 * 6 = 96bits$. El punto deseado fue:

- $posición\ deseada$: $(2, 4)$
- $orientación\ deseada$: $135°$

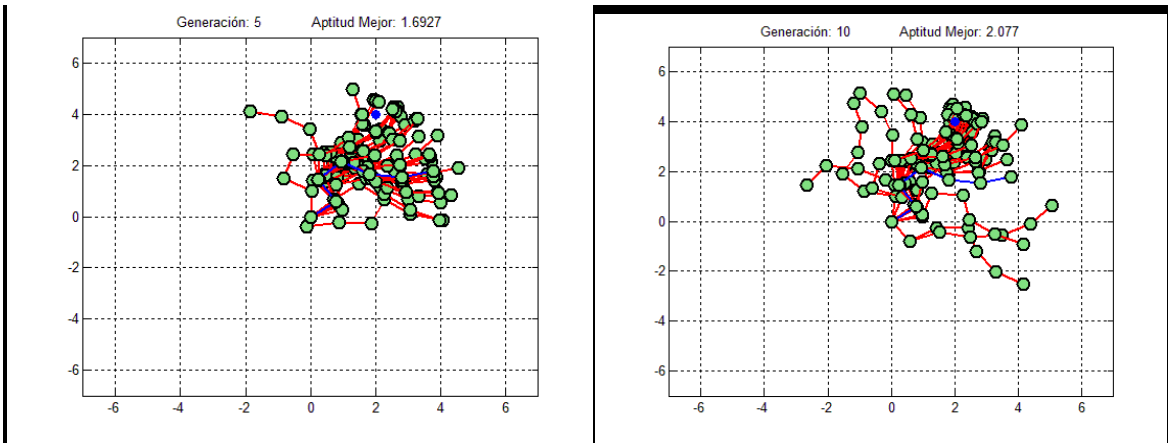
La configuración inicial que se usó fue:

Q_0_{rad}=[0.6450 1.5083 -1.3792 -1.2871 0.3911 0.3751]

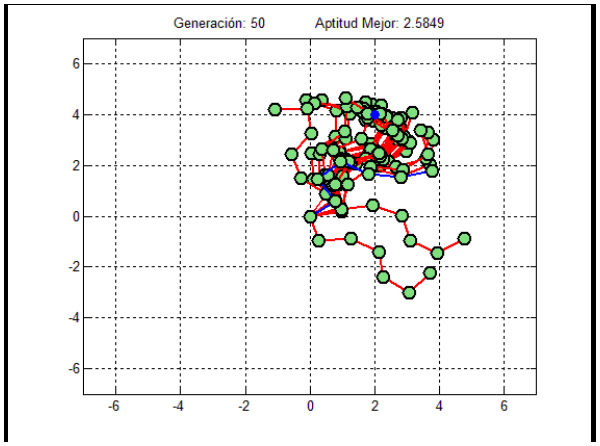
Q_0_{grad}=[36.9540 86.4180 -79.0200 -73.7460 22.4100 21.4920]

Lo que se puede observar en las imágenes es la población de individuos (configuraciones de robots) en esa generación, y con un punto azul la posición meta.

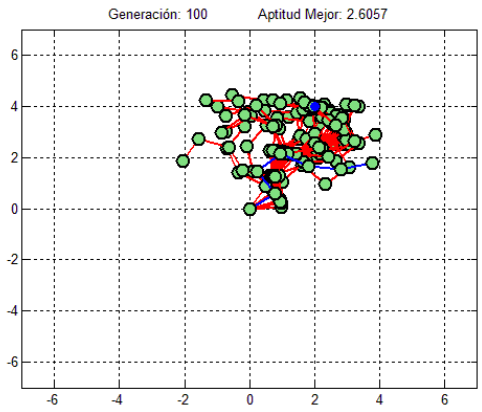
- **Primera Ejecución**



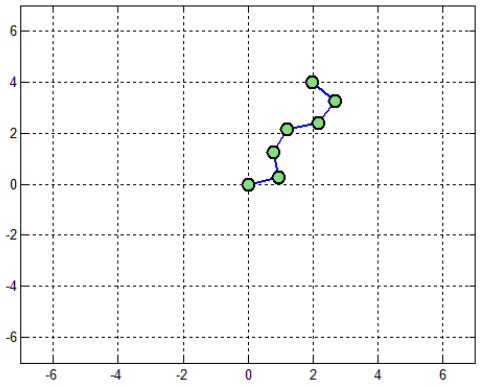
(a) (b)



(c)



(d)



(e)

.

Fig. 5.35 Primera ejecución del AG (3v) para un robot 6-gdl. (a) Generación 5, mejor_aptitud=1.6927; (b) Generación 10, mejor_aptitud=2.077; (c) Generación 50, mejor aptitud=2.5849; (d) Generación 100, mejor_aptitud=2.6057; (e) Mejor individuo obtenido.

El mejor individuo obtenido fue:

Q_{rad}=[00.2747 1.4727 -0.5968 -0.9092 0.7837 1.3035]
Q_{grad}=[15.7407 84.3805 -34.1922 -52.0944 44.9011 74.6851]

La evolución del AG se muestra en la fig. 5.36:

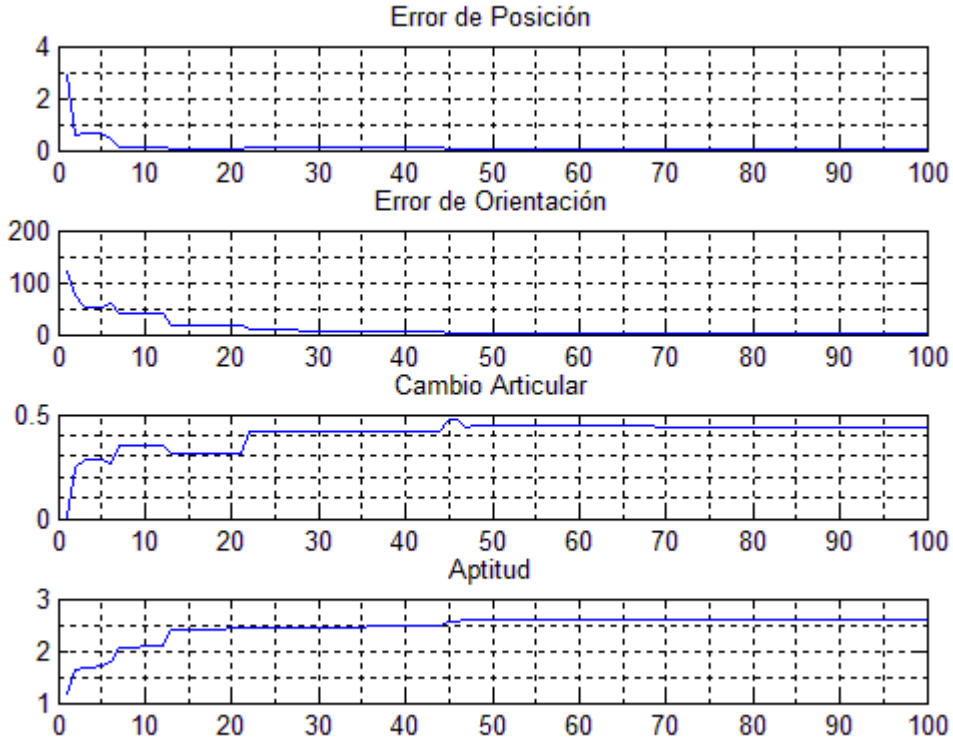


Fig. 5.36 Estado de los errores de posición, orientación, cambio articular y aptitud a través de las 100 generaciones en que se ejecutó el AG.

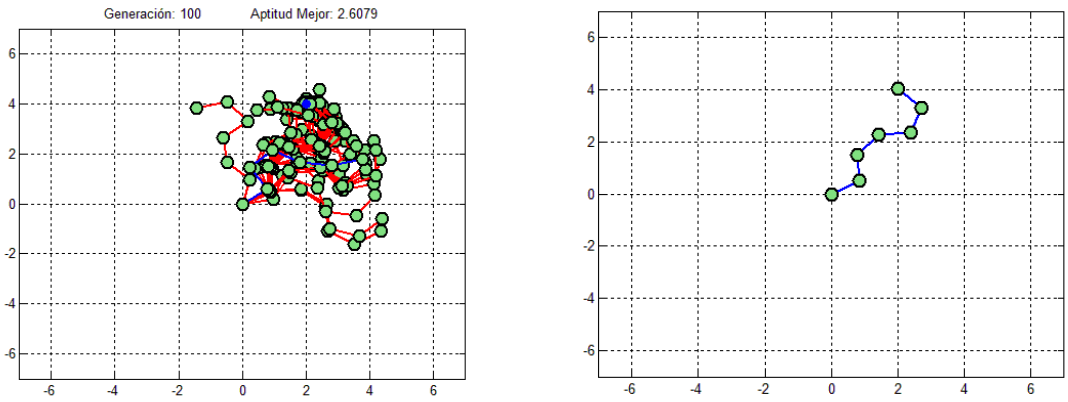
Puede observarse como a lo largo de las 100 generaciones a que se sometió la prueba el AG :

- Minimiza el error de posición (*error_final= 0.0113*).
- Minimiza el error de orientación (*error_final= 1.5793°*).
- Disminución del cambio articular

La mejor solución obtuvo los siguientes resultados:

- $Punto\ alcanzado$: $(x = 1.9973, y = 3.9890)$
- $Orientación\ final$: $133.4207°$

- **Segunda Ejecución**



(a) (b)

Fig. 5.37 Resultados de la segunda ejecución del AG (3v). (a) Generación 100, mejor_aptitud=2.6079; (b) Mejor individuo obtenido.

El mejor individuo obtenido fue:

Q_{rad}=[00.5191 1.1157 -0.7296 -0.8222 1.1672 1.1052]
Q_{grad}=[29.7427 63.9267 -41.8030 -47.1066 66.8738 63.3252]

La evolución del AG se muestra en la fig. 5.38:

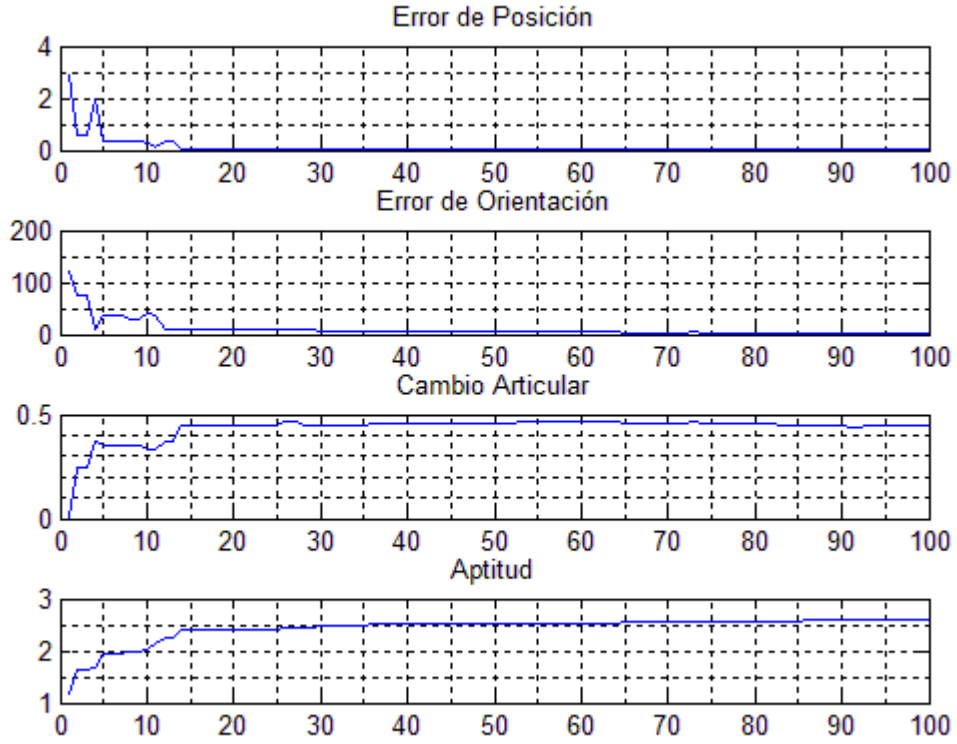


Fig. 5.38 Estado de los errores de posición, orientación, cambio articular y aptitud a través de las 100 generaciones en que se ejecutó el AG.

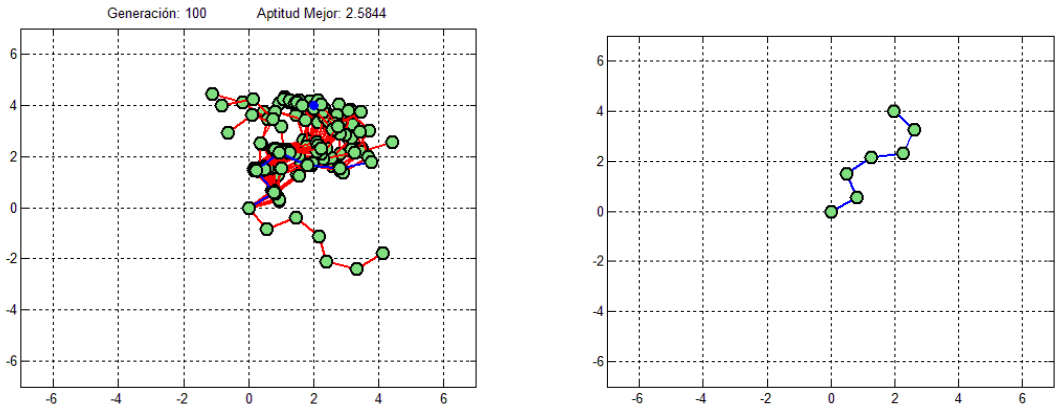
Puede observarse como a lo largo de las 100 generaciones a que se sometió la prueba el AG :

- Minimiza el error de posición (*error_final= 0.0336*).
- Minimiza el error de orientación (*error_final= 0.0412°*).
- Disminución del cambio articular.

La mejor solución obtuvo los siguientes resultados:

- $Punto\ alcanzado$: $(x = 2.0268, y = 4.0203)$
- $Orientación\ final$: $134.9588°$

- **Tercera Ejecución**



(a) (b)

Fig. 5.39 Resultados de la segunda ejecución del AG (3v). (a) Generación 100, mejor_aptitud=2.5844; (b) Mejor individuo obtenido.

El mejor individuo obtenido fue:

Q_{rad}=[0.5952 1.3011 -1.2196 -0.4995 1.0158 1.0796]
Q_{grad}=[34.1016 74.5477 -69.8785 -28.6194 58.2028 61.8558]

La evolución del AG se muestra en la fig. 5.40:

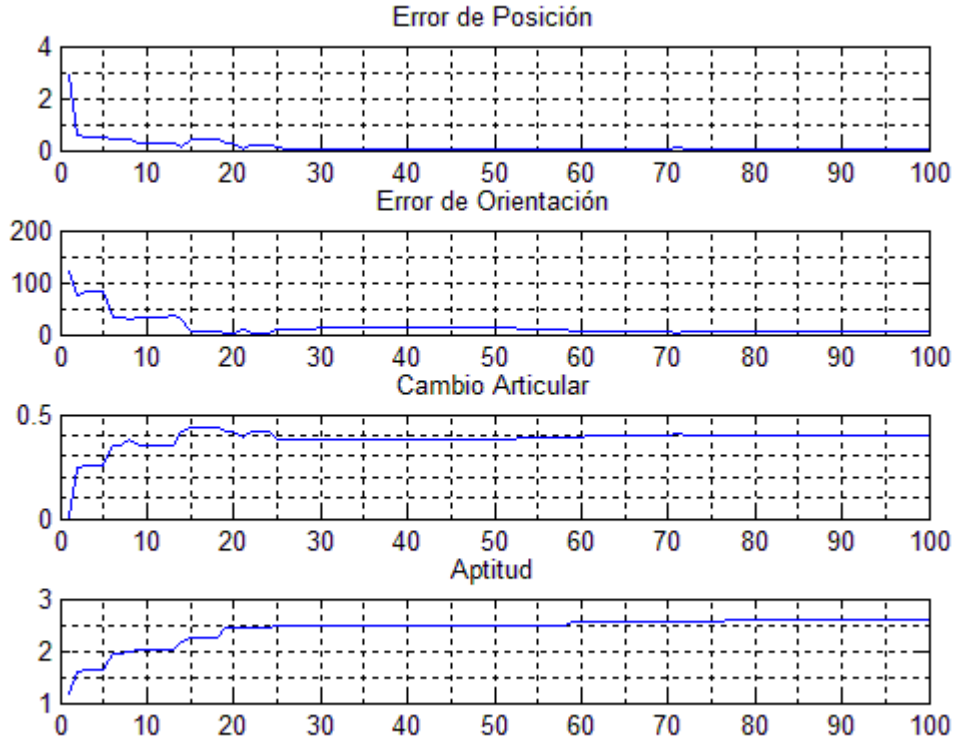


Fig. 5.40 Estado de los errores de posición, orientación, cambio articular y aptitud a través de las 100 generaciones en que se ejecutó el AG.

Puede observarse como a lo largo de las 100 generaciones a que se sometió la prueba el AG :

- Minimiza el error de posición (*error_final= 0.0058*).
- Minimiza el error de orientación (*error_final= 4.7900°*).
- Disminución del cambio articular.

La mejor solución obtuvo los siguientes resultados:

- $Punto\ alcanzado$: $(x = 1.9956, y = 4.0038)$
- $Orientación\ final$: $130.2100°$

5.5.2.3 Prueba 3: Robot 15-gdl

En este experimento se pidió al programa generar la cinemática inversa para un manipulador de 15-gdl. La longitud de cada cromosoma será de $16 * 15 = 240 bits$. El punto deseado fue:

- $posición\ deseada$: $(\ 10\ ,04\ )$
- $orientación\ deseada$: $0°$

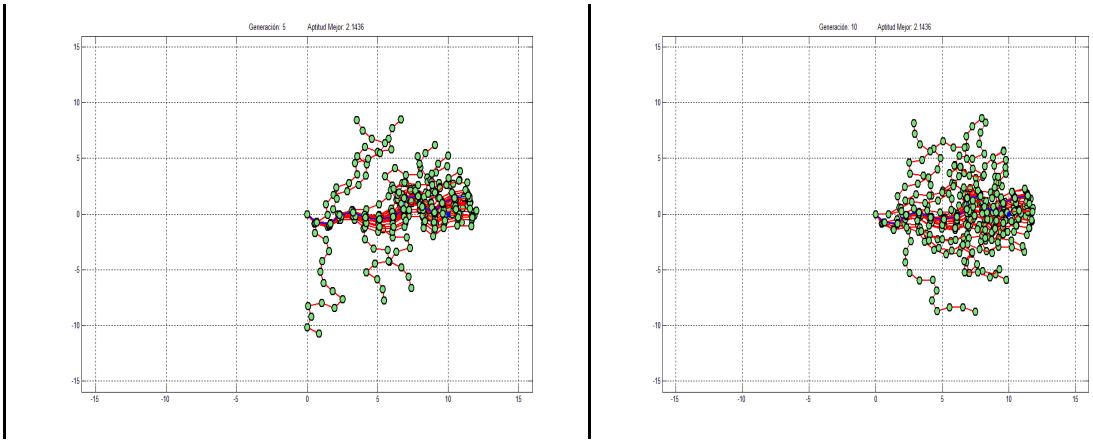
La configuración inicial que se usó fue:

Q_0_{rad}=[-0.8259, 0.7185, 1.0022, -0.6783, -0.7037, 0.3355, 0.3704, 0.5168, 0.8020, -1.3534, -1.4382, 0.6996, 1.5491, -0.4954, -0.3019]

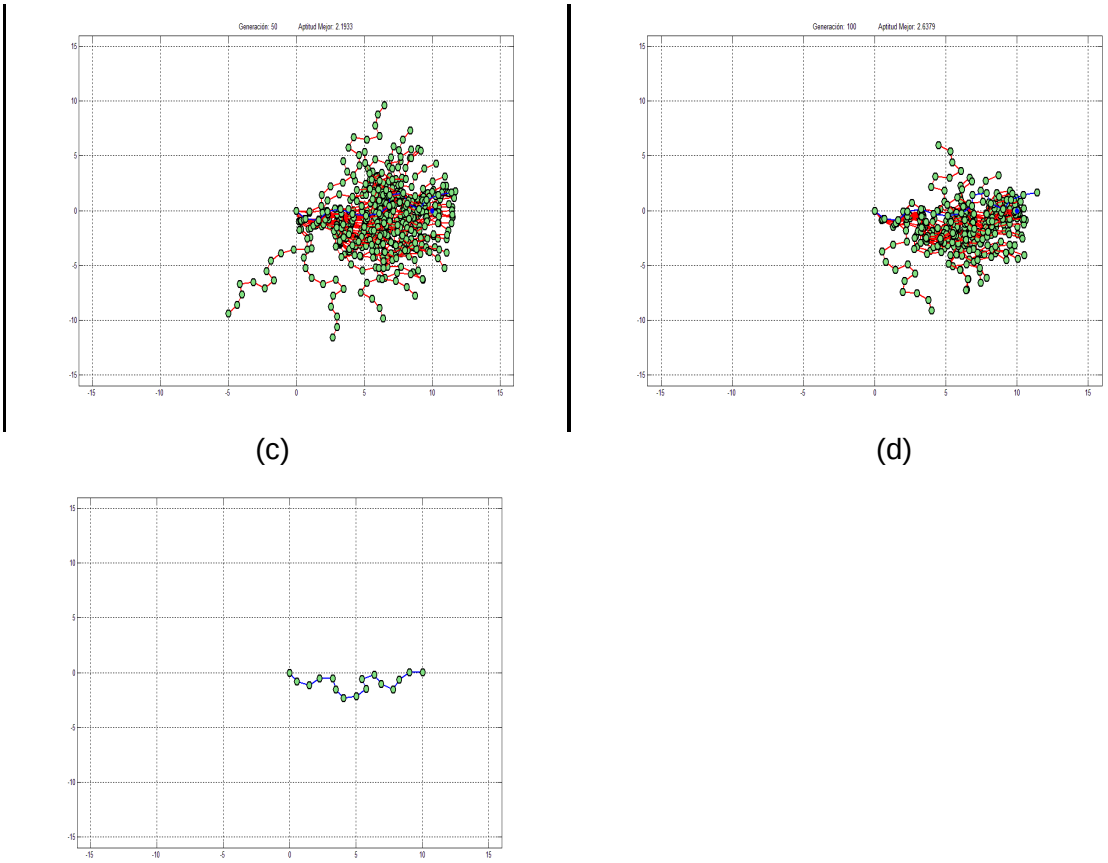
Q_0_{grad}=[-47.3220, 41.1660, 57.4200, -38.8620, -40.3200, 19.2240, 21.2220, 29.6100, 45.9540, -77.5440, -82.4040, 40.0860, 88.7580, -28.3860, -17.2980]

Lo que se puede observar en las imágenes es la población de individuos (configuraciones de robots) en esa generación, y con un punto azul la posición meta.

- **Primera Ejecución**



(a) (b)



(c) (d)

(e) .

Fig. 5.41 Primera ejecución del AG (3v) para un robot 15-gdl. (a) Generación 5, mejor_aptitud=2.1436; (b) Generación 10, mejor_aptitud=2.1436; (c) Generación 50, mejor aptitud=2.1933; (d) Generación 100, mejor_aptitud=2.6379; (e) Mejor individuo obtenido.

El mejor individuo obtenido fue:

Q_{rad}=[-0.9707 0.6208 1.0263 -0.6806 -1.3429 0.3750 1.1557 0.5365 1.1963 -1.5475 -1.3885 0.5401 1.5674 -0.3935 -0.6970]

Q_{grad}=[-55.6155 35.5682 58.8016 -38.9960 -76.9427 21.4865 66.2146 30.7397 68.5410 -88.6652 -79.5547 30.9457 89.8077 -22.5439 -39.9326]

La evolución del AG se muestra en la 5.42:

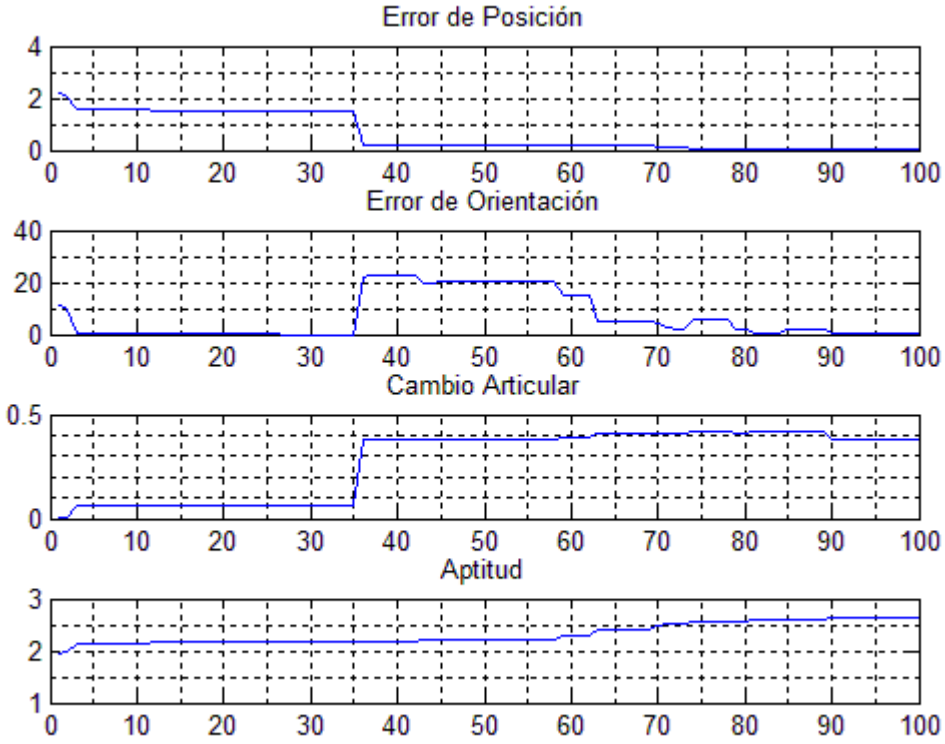


Fig. 5.42 Estado de los errores de posición, orientación, cambio articular y aptitud a través de las 100 generaciones en que se ejecutó el AG.

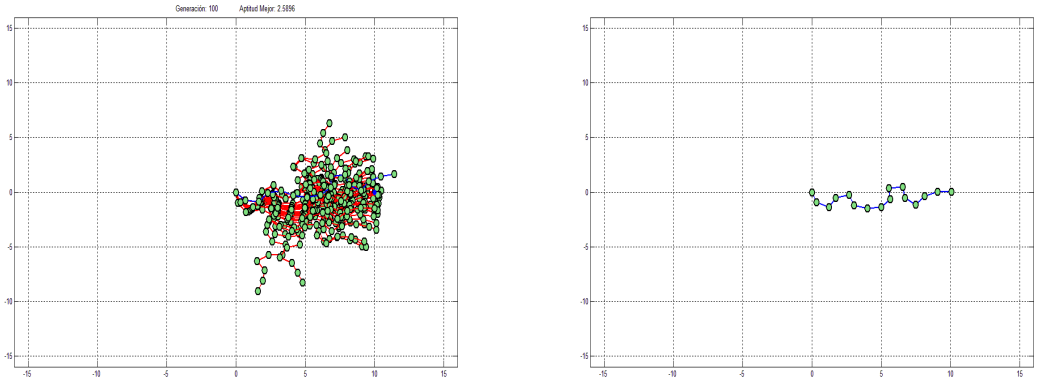
Puede observarse como a lo largo de las 100 generaciones a que se sometió la prueba el AG :

- Minimiza el error de posición (*error_final= 0.0424*).
- Minimiza el error de orientación (*error_final= 0.1456°*).
- Disminución del cambio articular.

La mejor solución obtuvo los siguientes resultados:

- $Punto\ alcanzado: (x = 10.0420, y = 0.0056)$
- $Orientación\ final: -0.1456°$

- **Segunda Ejecución**



(a) (b)

Fig. 5.43 Resultados de la segunda ejecución del AG (3v). (a) Generación 100, mejor_aptitud=2.5896; (b) Mejor individuo obtenido.

El mejor individuo obtenido fue:

Q_{rad}=[1.2187 0.7553 1.5177 -0.7826 -1.4862 0.9306 0.3612 0.7699 0.8231 -1.5498 -1.5380 0.7517 1.5491 -0.5200 -0.3438]

Q_{grad}=[36-69.8236 43.2779 86.9595 -44.8407 -85.1523 53.3194 20.6927 44.1129 47.1588 -88.7943 -88.1186 43.0692 88.7558 -29.7949 -19.6957]

La evolución del AG se muestra en la fig. 5.44:

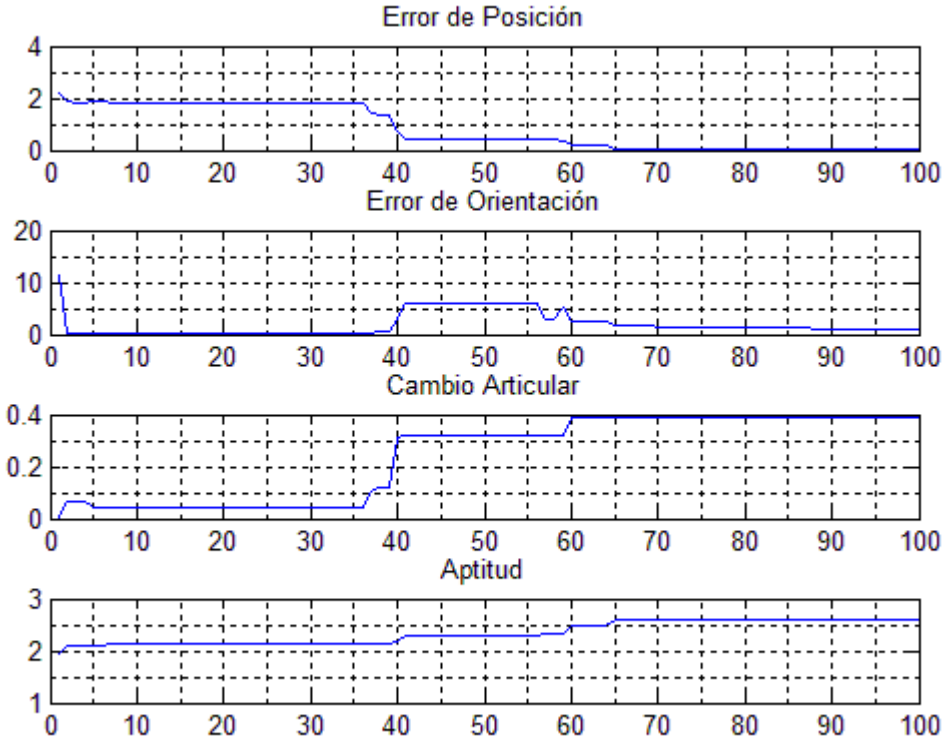


Fig. 5.44 Estado de los errores de posición, orientación, cambio articular y aptitud a través de las 100 generaciones en que se ejecutó el AG.

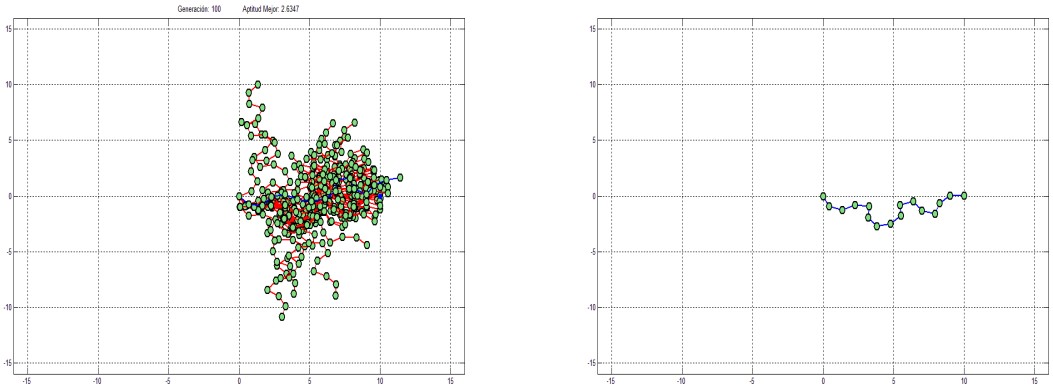
Puede observarse como a lo largo de las 100 generaciones a que se sometió la prueba el AG :

- Minimiza el error de posición (*error_final= 0.0690*).
- Minimiza el error de orientación (*error_final= 1.1261°*).
- Disminución del cambio articular.

La mejor solución obtuvo los siguientes resultados:

- $Punto\ alcanzado$: $(x = 10.0661, y = 0.0197)$
- $Orientación\ final$: $1.1261°$

- **Tercera Ejecución**



(a) (b)

Fig. 5.45 Resultados de la segunda ejecución del AG (3v). (a) Generación 100, mejor_aptitud=2.6347; (b) Mejor individuo obtenido.

El mejor individuo obtenido fue:

Q_{rad}=[-1.1193 0.7676 0.8066 -0.5713 -1.5129 0.7294 1.1311 0.5247 0.8889 -1.3542 -1.2266 0.6289 1.5600 -0.5139 -0.7429]

Q_{grad}=[-64.1299 43.9810 46.2167 -32.7338 -86.6821 41.7892 64.8083 30.0641 50.9299 -77.5909 -70.2768 36.0352 89.3820 -29.4434 -42.5665]

La evolución del AG se muestra en la fig. 5.46:

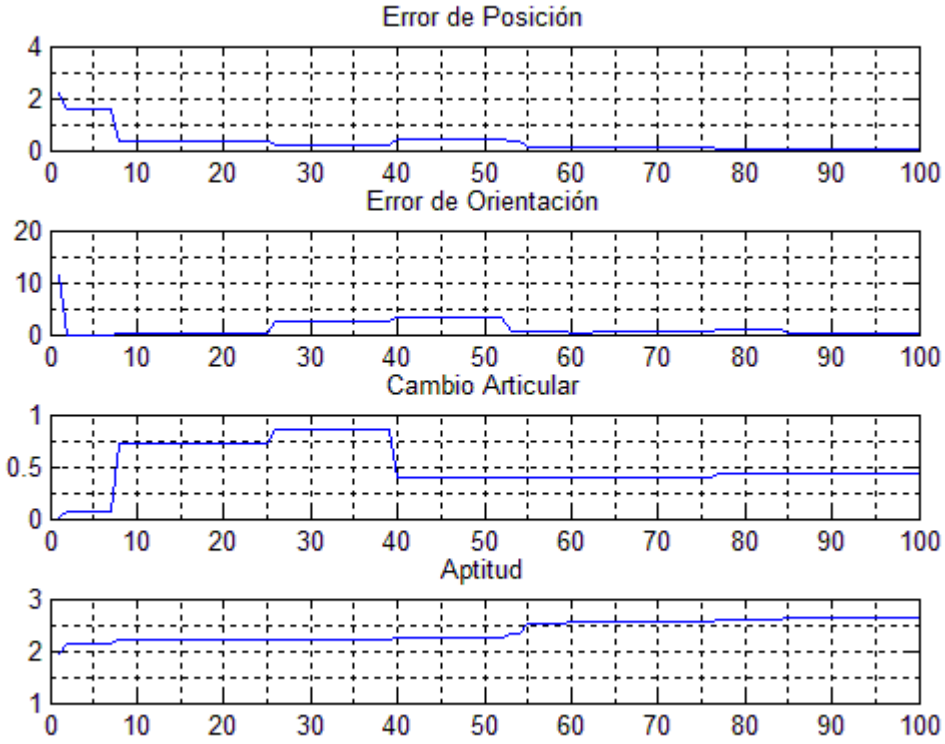


Fig. 5.46 Estado de los errores de posición, orientación, cambio articular y aptitud a través de las 100 generaciones en que se ejecutó el AG.

Puede observarse como a lo largo de las 100 generaciones a que se sometió la prueba el AG :

- Minimiza el error de posición (*error_final= 0.0157*).
- Minimiza el error de orientación (*error_final= 0.2170°*).
- Disminución del cambio articular.

La mejor solución obtuvo los siguientes resultados:

- $Punto\ alcanzado: (x = 10.0128, y = 0.0090)$
- $Orientación\ final: -0.2170°$

5.6 Comentario final del capítulo

Hasta aquí se ha desarrollado con éxito un método para generar la cinemática inversa completa (posición y orientación) de un robot manipulador con n articulaciones. Esta metodología será utilizada más adelante en la generación de trayectorias para el auto-ensamble del robot modular planar.

Cabe señalar que los parámetros (p_c, p_m, *tamaño de la población, representación, número de generaciones, etc.*) reportados en este trabajo fueron propuestos basándose en la bibliografía y en las pruebas realizadas durante este trabajo. Aún es un problema fundamental de la computación evolutiva la elección correcta de estos parámetros.

Finalmente, hay que mencionar que esta metodología también es aplicable a cualquier otro tipo de manipulador de cadena abierta. Lo único con lo que hay que contar son las ecuaciones de cinemática directa del manipulador, esto para poder construir la función de aptitud (se necesita poder calcular los errores de posición y de orientación para cada individuo

Capítulo 6

Generación de Trayectorias

En el capítulo 5 se presentó una metodología para aproximar (con un rango de error aceptable) la cinemática inversa completa (posición y orientación) de un robot manipulador dado, ahora, queda aplicar esta metodología a la generación de trayectorias para el auto-ensamble de un robot planar.

6.1 Análisis General

Ya se ha descrito el problema a resolver, en un inicio, hay un manipulador pre-ensamblado con unos cuantos módulos (al menos 3), y hay varios módulos esparcidos sobre la mesa, el problema ahora estriba en ir por cada uno de los módulos.

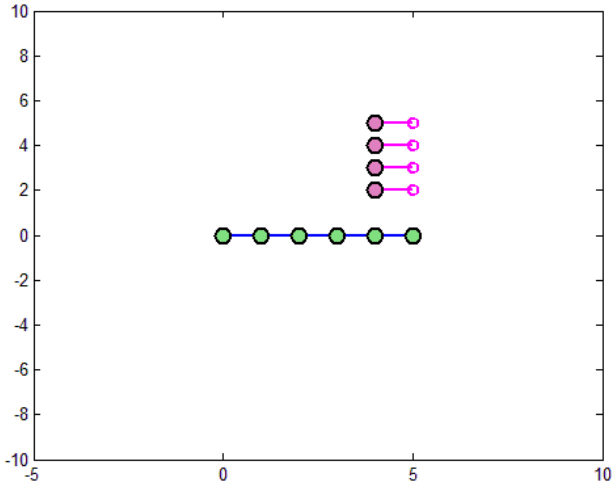


Fig. 6.1 Caso particular de una condición inicial de de ensamble; el manipulador (en azul) con base en (0,0) debe ensamblar a sí mismo los otros módulos (rosa).

Un caso particular se muestra en la fig. 6.1. Se puede observar un manipulador (en azul) de 5 articulaciones, y en el área de trabajo hay 4 eslabones sueltos (en rosa). Estos módulos tienen dos extremos diferentes, la idea es que el ensamble ocurra por el lado con el círculo negro.

Para que el manipulador pueda ensamblar estos módulos así mismo debe llegar al punto de ensamble con la orientación adecuada, y en esta dirección. Esto es suponiendo que el sistema de sujeción de los módulos requiera que ambos módulos sean colineales al momento de efectuar el ensamble, fig. 6.2.

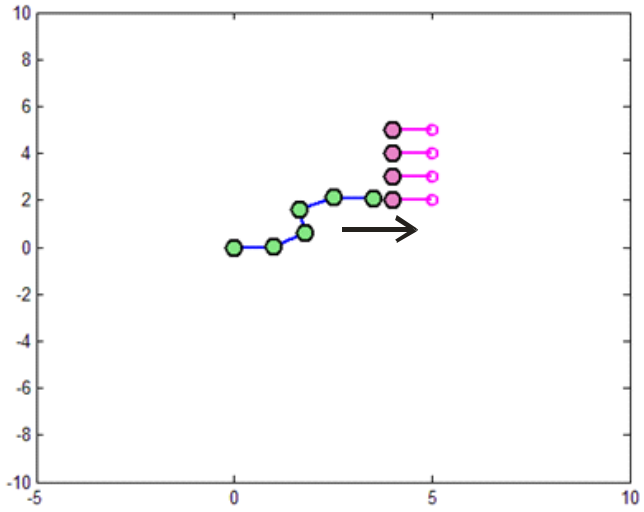


Fig. 6.2 Acercamiento del manipulador para ensamblar un eslabón. El manipulador debe acercarse con la orientación adecuada en la dirección del eje del eslabón.

Entonces, para conseguir el ensamble con el módulo, el manipulador debería seguir una trayectoria similar a la de la fig. 6.3.

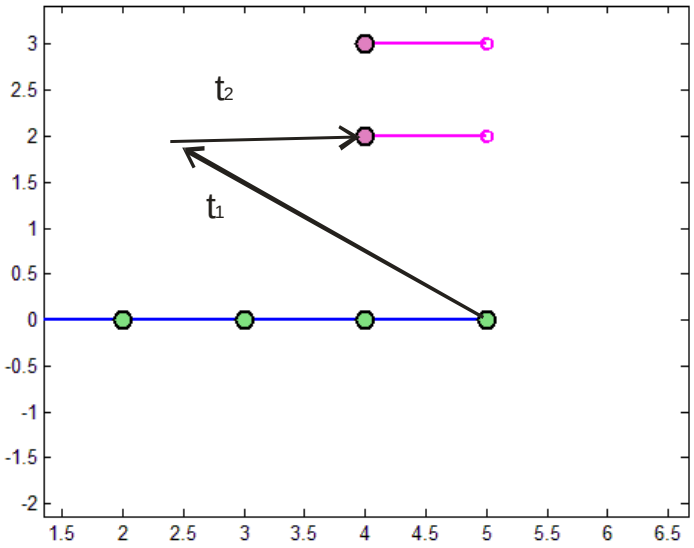


Fig. 6.3 Trayectoria de ensamble en dos pasos.

6.2 Generación de Trayectorias

Como se comentó en el capítulo 3, la generación de una trayectoria debería considerar la evasión de obstáculos y el comportamiento dinámico del sistema (minimizar esfuerzos). Sin embargo, en el presente trabajo se han omitido estos análisis para simplificar el problema. De esta manera, la trayectoria necesaria para realizar el ensamble de un módulo se puede completar en dos pasos, fig. 6.3:

1) Llegar a un punto anticipado, que sea colineal al módulo destino, con la orientación adecuada (t_1).

2) Acercarse al módulo destino de manera colineal y sin variar la orientación hasta el punto de ensamble de este (t_2).

Después que se han cumplido estos tiempos, se puede decir que el ensamble ha ocurrido con éxito. Un ensamble exitoso tiene un doble impacto en el manipulador; por un lado, el manipulador ahora tiene una articulación más (otro grado de libertad), y además su punto de referencia (efector final) se ha recorrido al extremo del nuevo eslabón ensamblado.

Ahora para cubrir estas trayectorias (t_1 y t_2), se deben generar puntos a lo largo de estas líneas para que el manipulador los siga. Estos puntos se obtienen con curvas tipo spline, fig. 6.3.

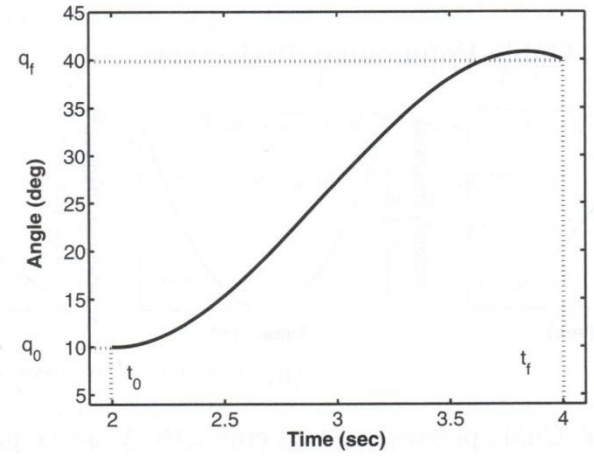


Fig. 6.4 Ejemplo de curva spline para el movimiento de una articulación

Se trabaja con splines debido a que implican un movimiento suavizado, y le da tiempo al manipulador para acelerar y/o frenar durante el movimiento. Para generar una curva spline se debe conocer el tiempo inicial (t_0) y el tiempo final (t_1) en que se realiza el movimiento, las coordenada inicial ($x(t_0)$) y final ($x(t_1)$) y tantas derivadas de las coordenadas evaluadas en estos instantes como se requieran para calcular la spline del grado deseado.

En este proyecto se han ocupado splines de 3° grado, y las velocidades inicial y final en cada instante se han supuesto en 0.

De esta manera, haciendo splines para el ensamble de cada módulo, se puede calcular la trayectoria para el auto-ensamble de los 4 eslabones presentados en el ejemplo, fig. 6.5:

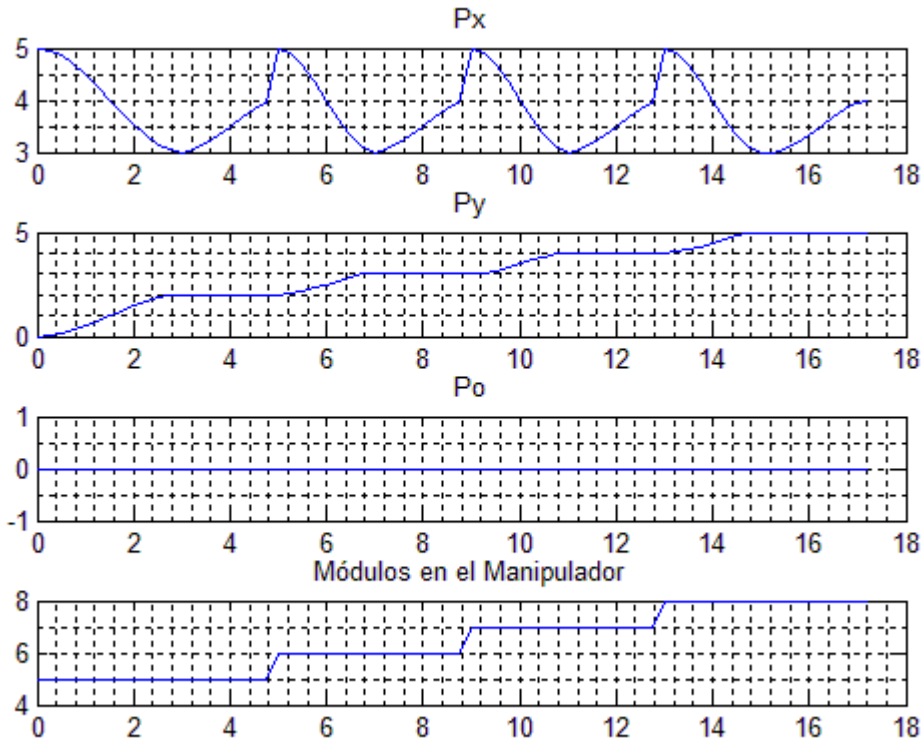


Fig. 6.5 Trayectorias en x (Px), en y (Py) y en la orientación (Po) con respecto al tiempo; hasta abajo se muestran las articulaciones del manipulador en ese instante (aumenta con cada ensamble).

Obsérvense que la trayectoria Px del ejemplo no es suave, esto es debido a que cada que se toma un nuevo eslabón el punto de referencia cambia y se recorre, y esto se ve en la trayectoria como un 'salto' repentino de un valor a otro. Sin embargo, la trayectoria es en realidad suave.

Finalmente si se grafica esta trayectoria en el espacio de tarea, quedaría de la siguiente manera, fig. 6.6:

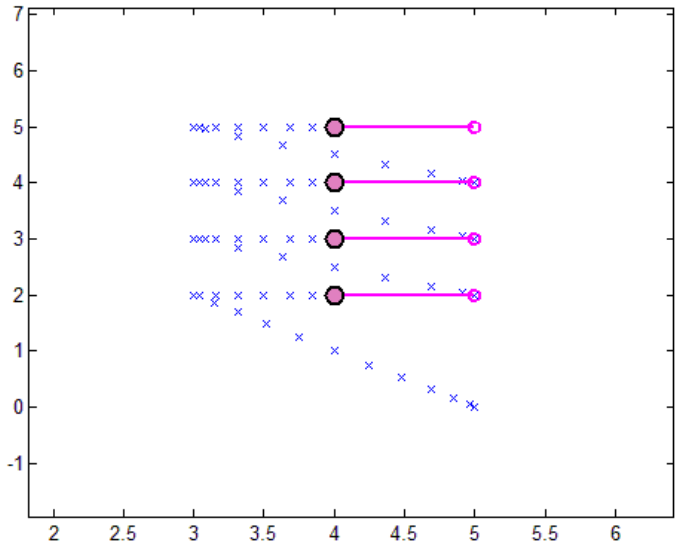


Fig. 6.6 Mismas trayectorias de la fig. 6.6 graficadas en el espacio de tarea.

Hasta aquí, ya se cuenta con la trayectoria en el espacio cartesiano, sin embargo, para que el robot pueda ejecutarla es necesario pasarla al espacio articular, para ello es necesaria la cinemática inversa. En este proyecto, la ausencia de las ecuaciones de cinemática inversa está cubierta con el AG desarrollado en el capítulo 5.

Al AG hay que darle la posición inicial del manipulador, el punto (posición y orientación) al que se desea llegar, y una manera de calcular la cinemática directa de manipulador. Así, para resolver toda la trayectoria articular del problema es necesario ejecutar un AG por cada punto de la trayectoria.

6.3 Simulaciones de Auto-Ensamble

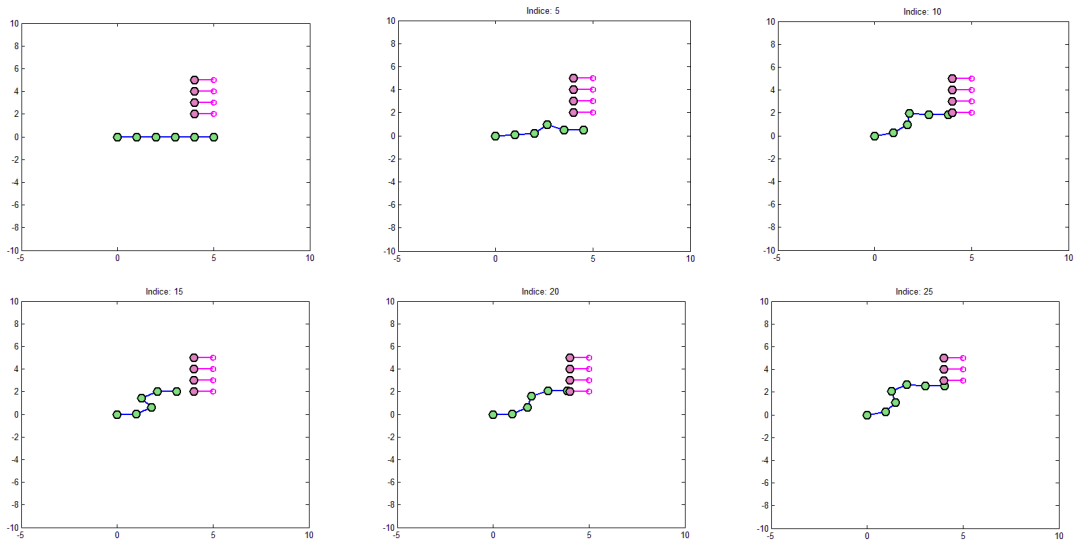
Se presentan tres simulaciones de auto-ensamble de un robot planar. En todas ellas de inició con un manipulador de 5 articulaciones en su posición cero ($q_i = 0$), y con varios eslabones repartidos en el área de trabajo. El orden de ensamble de los módulos fue declarado en el programa.

6.3.1 Simulación 1 – Caso de ejemplo

En esta simulación se repartieron 4 módulos en las siguientes posiciones:

Posición x	Posición y	Orientación
4	2	0
4	3	0
4	4	0
4	5	0

La posición inicial se muestra en la fig. 6.1, y las trayectorias del espacio de tarea se muestran en las figs. 6.5 y 6.6. La trayectoria completa está formada por 70 puntos, es decir, se requiere calcular el AG 70 veces para obtener la trayectoria articular completa. En la fig. 6.7 se muestra el resultado arrojado por el AG; el programa requirió 20 minutos para calcular esta trayectoria articular.



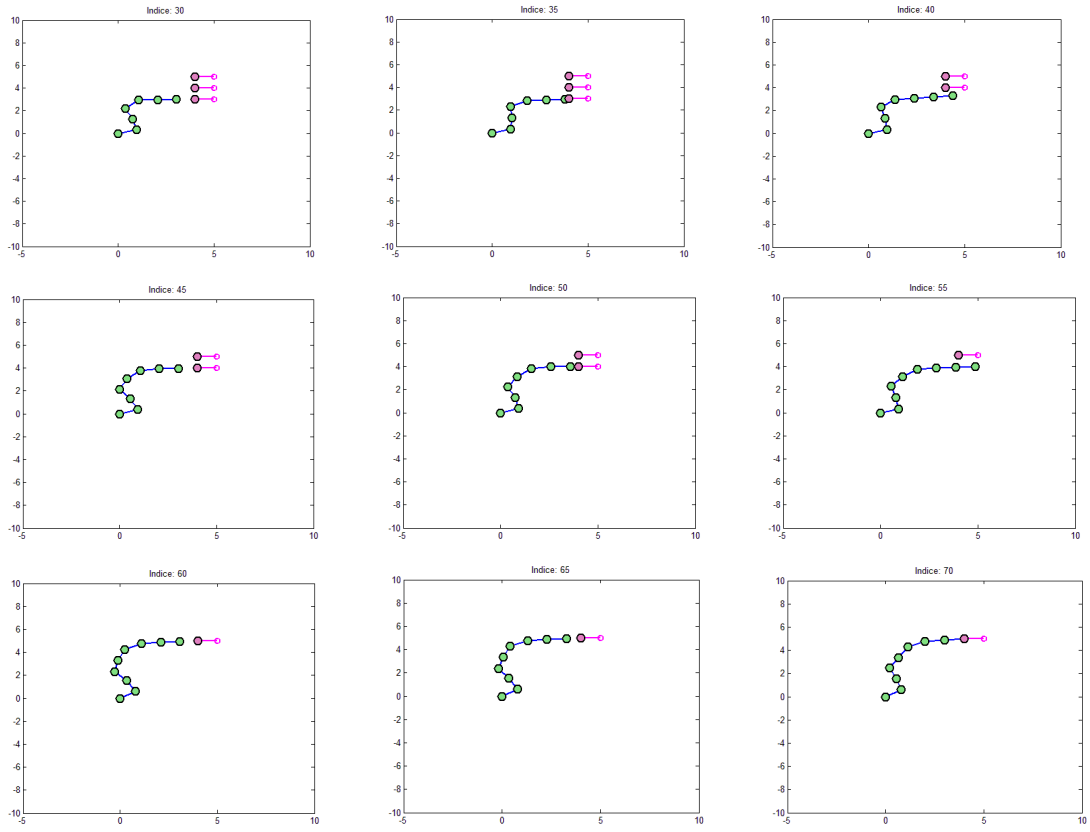


Fig. 6.7 Trayectoria articular del auto-ensamble propuesto construida por el AG.

6.3.2 Simulación 2

En esta simulación se repartieron 5 módulos en las siguientes posiciones:

Posición x	Posición y	Orientación
4	-2	-45
3	2	90
5	2	-45
6	-1	45
6	-3	0

La posición inicial se muestra en la fig. 6.8:

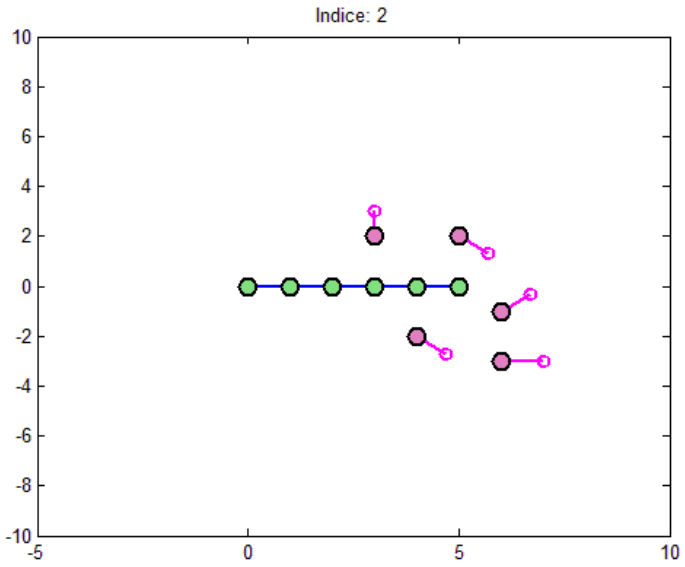


Fig. 6.8 Condiciones iniciales en la segunda simulación.

Las trayectorias del espacio de tarea se muestran en las figs. 6.9 y 6.10. La trayectoria completa está formada por 85 puntos.

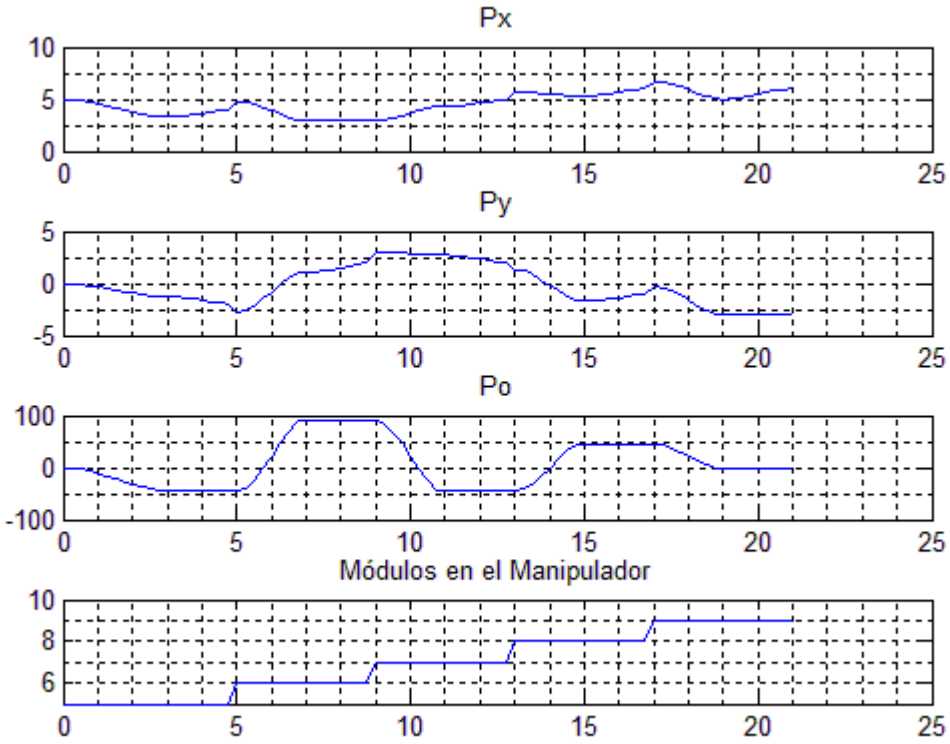


Fig. 6.9 Trayectorias con respecto al tiempo para el auto-ensamble de la simulación2.

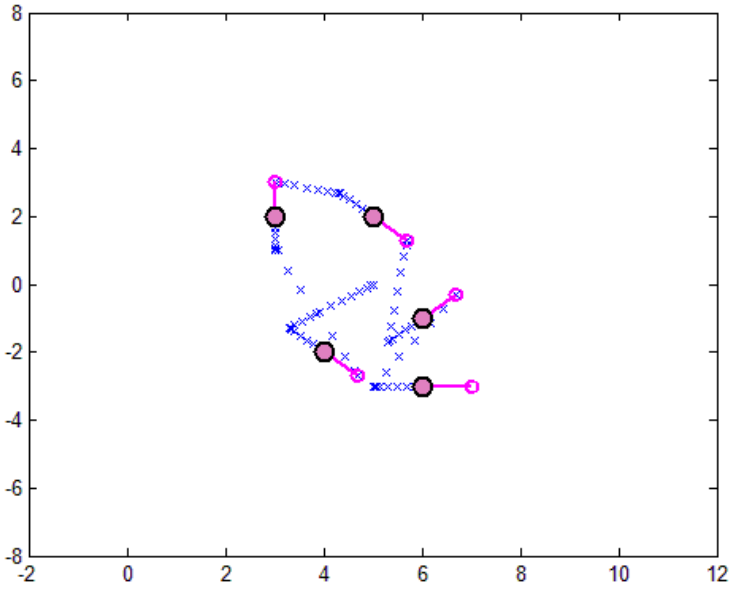
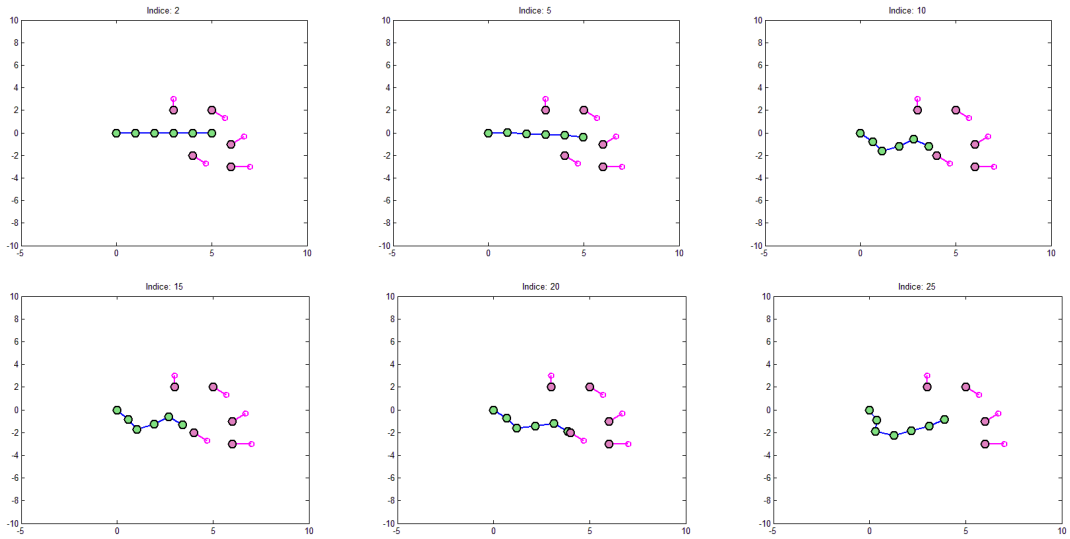


Fig. 6.10 Mismas trayectorias de la fig. 6.10 en el espacio de tarea.

Finalmente, en la fig. 6.11 se muestra el resultado arrojado por el AG; El programa requirió 30 minutos para calcular la trayectoria articular.



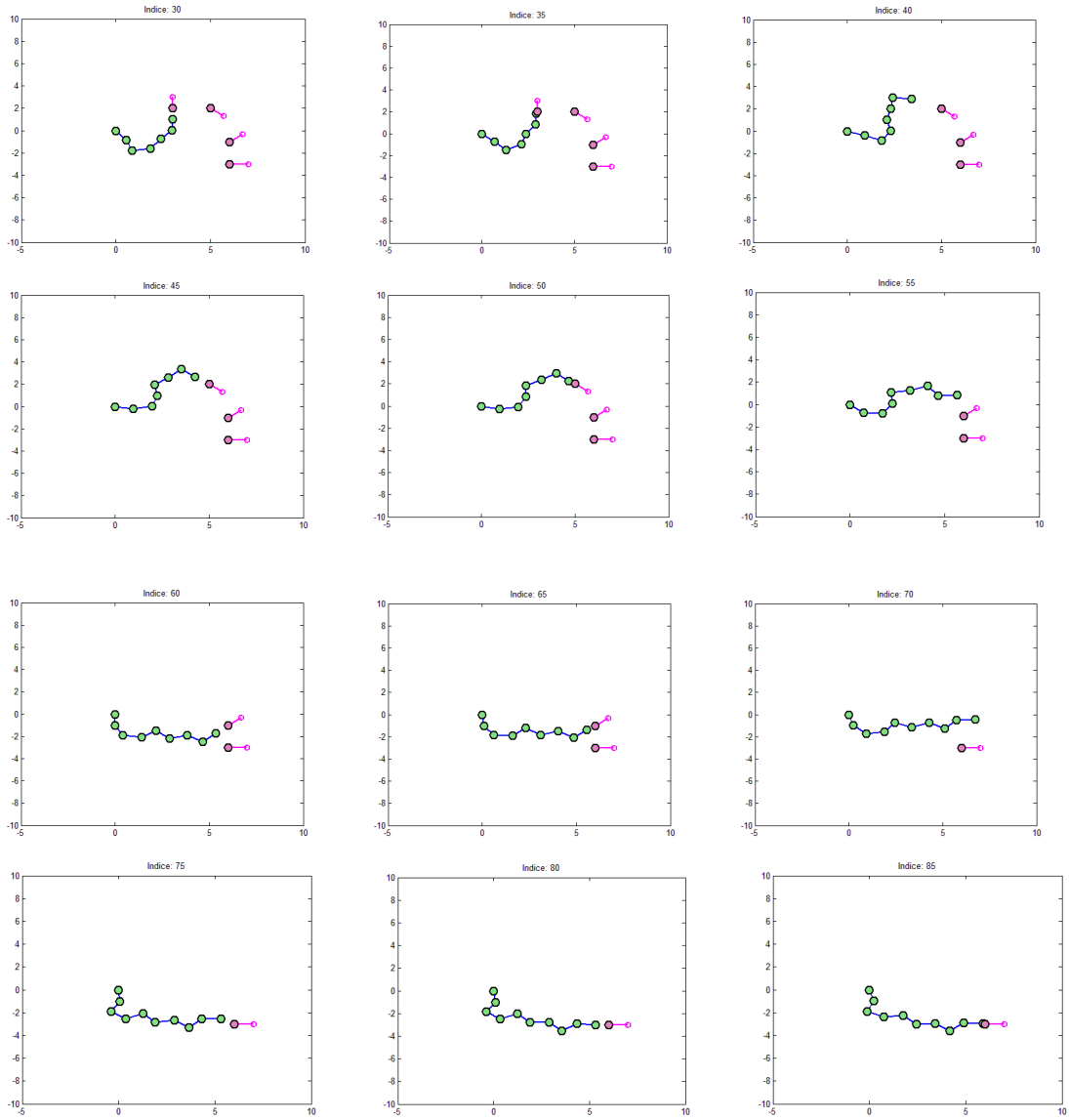


Fig. 6.11 Trayectoria articular del auto-ensamble propuesto construida por el AG.

6.3.2 Simulación 3

En esta simulación se repartieron 17 módulos en las siguientes posiciones:

Posición x	Posición y	Orientación
4	-2	-45
3	2	90
5	2	-45
6	-1	45
6	-3	-45
8	-5	0
8	-3	0
8	-1	0
8	1	0
8	3	0
6	4	180
8	5	90
12	4	45
12	2	45
12	0	45
12	-2	45
12	-4	45

La posición inicial se muestra en la fig. 6.12:

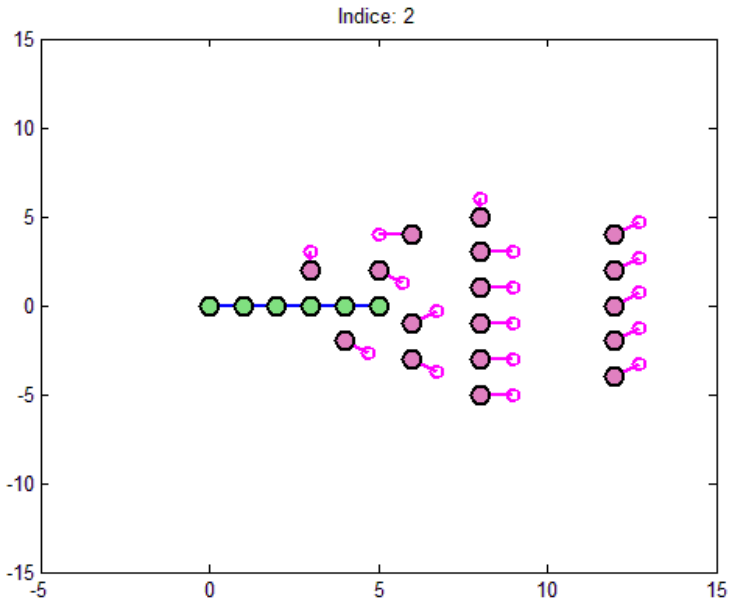


Fig. 6.12 Condiciones iniciales en la segunda simulación.

Las trayectorias del espacio de tarea se muestran en las figs. 6.13 y 6.14. La trayectoria completa está formada por 277 puntos.

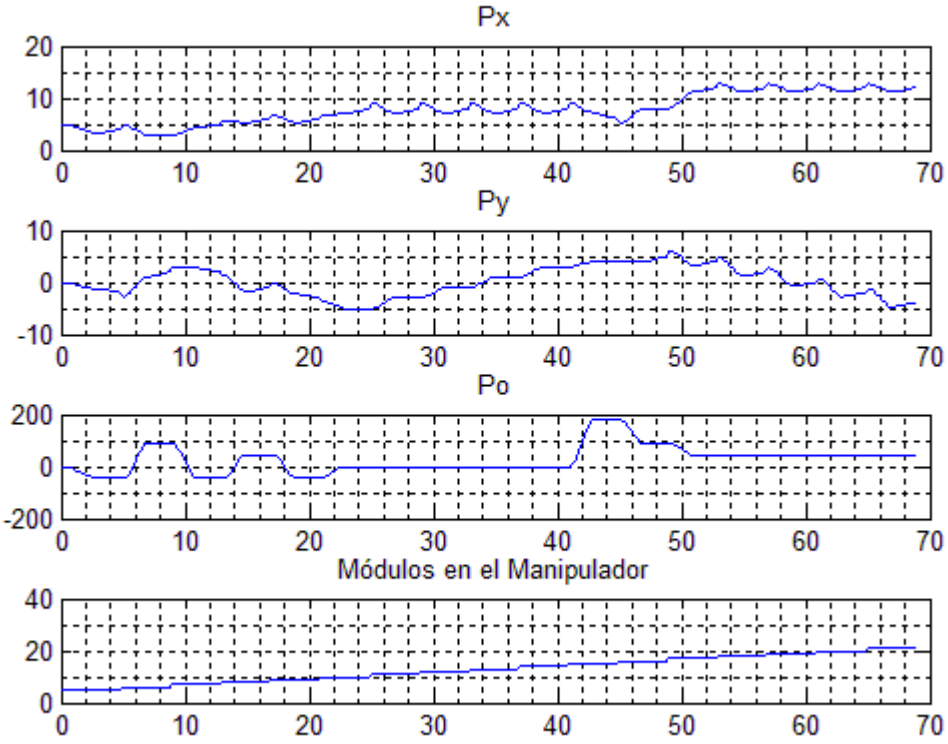


Fig. 6.13 Trayectorias con respecto al tiempo para el auto-ensamble de la simulación2.

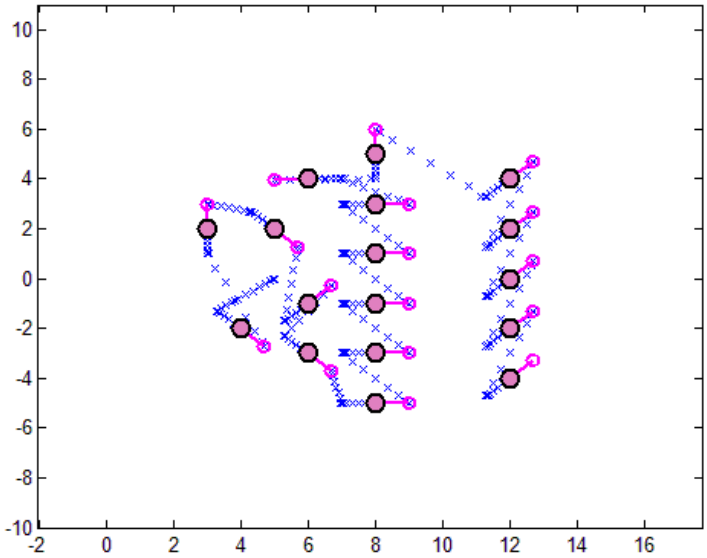
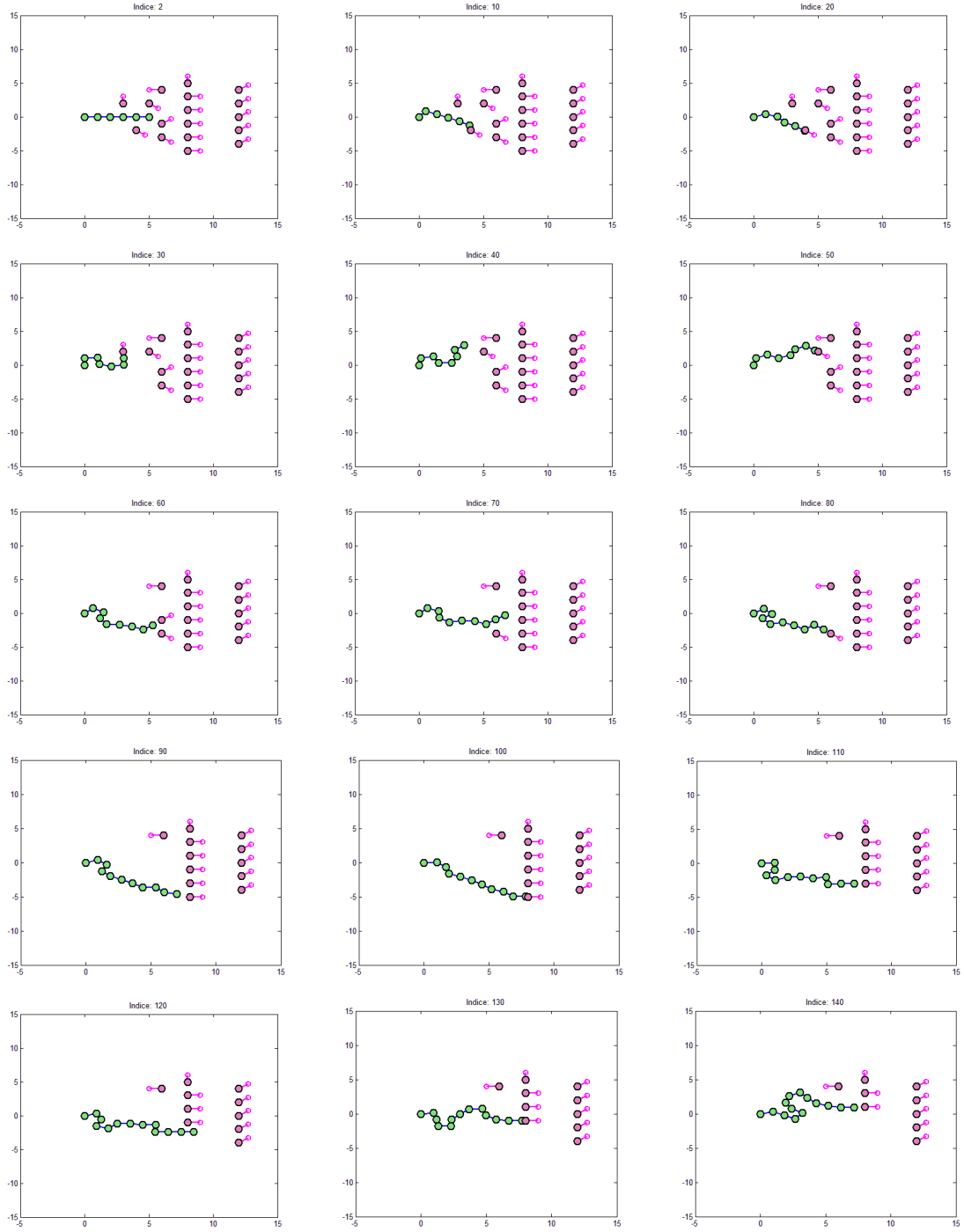


Fig. 6.14 Mismas trayectorias de la fig. 6.10 en el espacio de tarea.

Finalmente, en la fig. 6.15 se muestra el resultado arrojado por el AG; el programa requirió 5 horas y 30 minutos para calcular esta trayectoria articular.



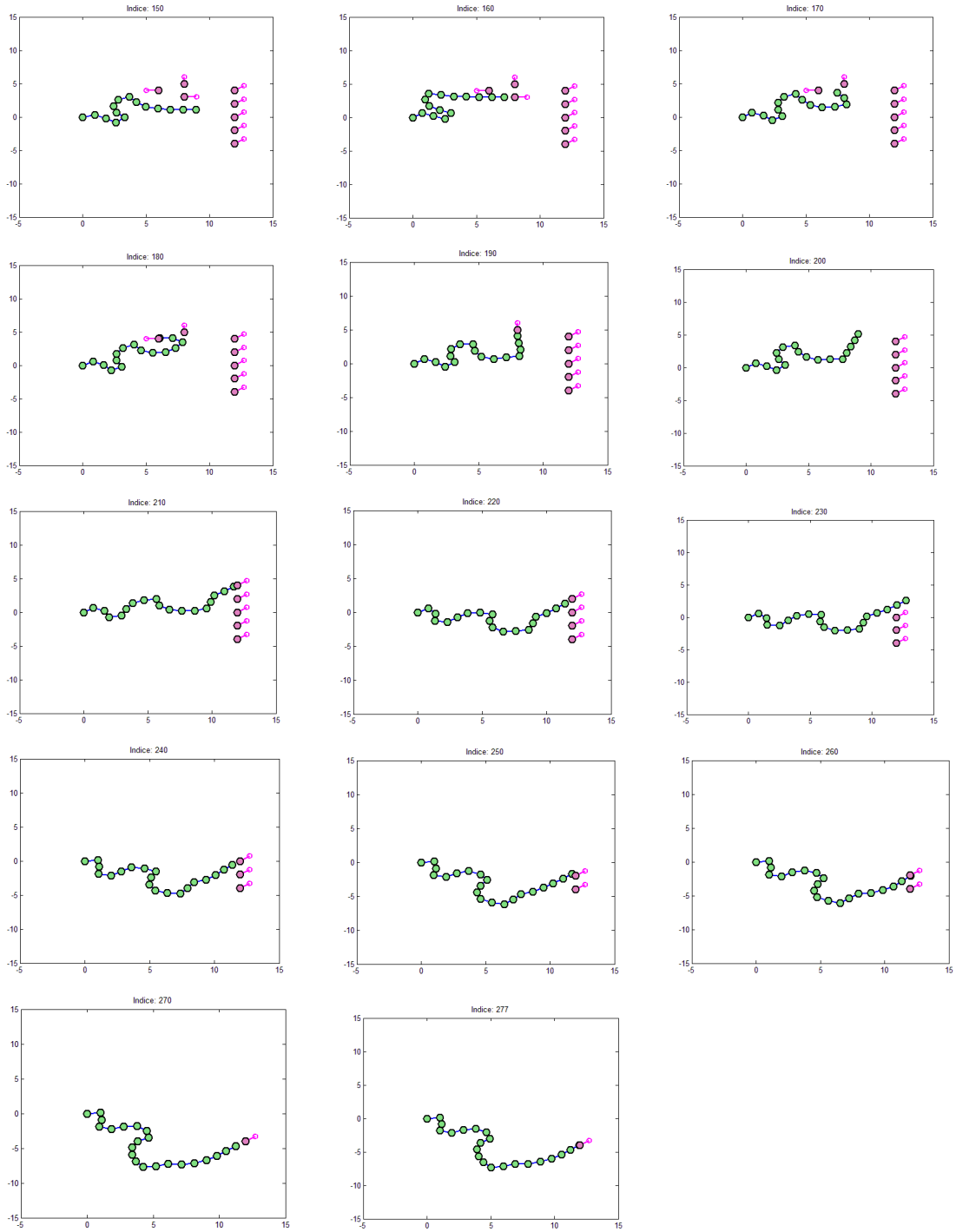


Fig. 6.15 Trayectoria articular del auto-ensamble propuesto construida con el AG.

.

6.4 Comentario final del capítulo

En este capítulo se ha mostrado como se construyeron las trayectorias para el auto-ensamble del robot modular; también se puso a prueba el desempeño del AG a la hora de generar la cinemática inversa de trayectorias completas.

Cabe mencionar que mientras va creciendo el manipulador, el AG se vuelve más tardado. Esto es normal, ya que la cadena binaria de los individuos crece 16 bits cada que se agrega un nuevo eslabón, este incremento en la cadena hace más tardados los cálculos que se realizan dentro del AG.

Capítulo 7

Conclusiones y Trabajo Futuro

Un SRM ofrece muchos retos que deben ser resueltos de forma eficiente. En este trabajo se ha atacado el problema del auto-ensamble (en condiciones ideales) y se ha verificado la efectividad del AG para generar soluciones adecuadas a la cinemática inversa.

El funcionamiento del generador de trayectorias ha sido aceptable y satisfactorio bajo las condiciones ideales que se permiten en una simulación, sin embargo, aún queda pendiente como trabajo futuro, su utilización en un sistema físico, ya sea un robot modular, o un manipulador convencional.

Otra colaboración importante sería la de incluir parámetros dinámicos en la generación de trayectorias, ya que el análisis realizado ha sido cinemático, pero no se garantiza en ningún punto que un robot dado con las características mencionadas pueda ejecutar el mismo movimiento.

Otro punto importante sería el ahondar en una manera de generar secuencias de movimiento para robots modulares móviles. Esto es, generar estrategias de movimiento usando paradigmas evolutivos.

Apéndice A

Web del proyecto y Material disponible

Se ha realizado un sitio web para ese proyecto en la siguiente URL:

http://mx.geocities.com/robot_modular

El propósito de este sitio es la difusión del presente trabajo y del seguimiento que se le dé a futuro. También se incluyen enlaces de interés sobre el proyecto.

Descargas

Dentro del sitio web se podrá descargar esta tesis en formato pdf, así como todos los códigos matlab de las simulaciones utilizadas en el texto. Cualquier trabajo futuro del proyecto también estará disponible.

Contacto

Cualquier comentario, crítica, duda, sugerencia o colaboración es bienvenida. Mi correo electrónico es:

likus_kikus@hotmail.com

Siéntase en confianza de escribir.

BIBLIOGRAFÍA

1. J. M. Ahuactzin, E. Mazer, P. Bessiere, E.G. Talbi. *Using Genetic Algorithms for Robot Motion Planning.* Lecture Notes in Computer Science 708, Geometric Reasoning for perception and Action, pp. 84-93 Springer-Verlag, 1993.

2. J. M. Ahuactzin. *Le Fil d'Ariadne*. PhD thesis, Institut National Politechnique de Grenoble, 1994.

3. Ronald C. Arkin, *Behavior-Based Robotics,* MIT Press, 1998.

4. Carlos A. Coello Coello. *Introducción a la Computación Evolutiva (Notas de Curso).* CINVESTAV México D.F., Abril 2006.

5. Yuval Davidor. *Genetic Algorithms and Robotics: A Heuristic Strategy for Optimization,* vol. 1. World Scientific Publishing Co., 1990.

6. Kenneth A. De Jong. *An Analysis of Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.

7. T. Fukuda y T. Ueyama. *Cellular Robotics and Micro Robotic Systems,* vol. 10. World Scientific Publishing Co., 1994.

8. David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Publishing Co. Reading, Massachusetts, 1989.

9. J. Grefenstette. Optimization of Control Parameters of Genetic Algorithms*. IEEETransactions on Systems, Man and Cybernetics, SMC-16(1):122-128,* 1986*.*

10. John H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, Michigan, 1975.

11. Ángel Kuri y J. Galavis, *Algoritmos genéticos.* IPN, UNAM y FCE. Primera Edición, México D.F., 2002.

12. T. Nishimura, K. Sugawara, I. Yoshihara y K. Abe. *A Motion Planning Method for a Hyper Multi-joint Manipulator using a Genetic Algorithm.* 1998.

13. Anibal Ollero Baturone . *Robótica: manipuladores y robots* móviles. Ed. Alfaomega/Marcombo, 2001.

14. S. Nolfi y D. Floreano. *Evolutionary Robotics.* MIT Press, 2000.

15. S. Russell y P. Norvig, *Inteligencia Artificial: Un enfoque moderno.* Pearson Education., Segunda Edición, 2004.

16. José Santos y Richard J. Duro, *Evolución Artificial y Robótica Autónoma,* Ed. Alfaomega/Ra-Ma, 2005.

17. Eduardo Serna, *Diseño de Circuitos Lógicos Combinatorios usando Programación Genética,* Tesis de Maestría, Universidad Veracruzana, México, 2001.

18. Mark W. Spong, Seth Hutchinson y M. Vidyasagar, *Robot Modeling and Control.* John Wiley & Sons Inc., 2006.

LINKS

a. Claytronics Project, *Carnegie Mellon University*
http://www.cs.cmu.edu/~claytronics

b. Modular Robotics, *Palo Alto Research Center*
http://www2.parc.com/spl/projects/modrobots

c. Molecubes, *University of Cornell, New York*
http://www.molecubes.org

d. Self-Reconfiguring Modular Robotics, *Wikipedia*
http://en.wikipedia.org/wiki/Self-Reconfiguring_Modular_Robotics

e. Swarm-bots Project, *EPFL en Lausanne, Suiza*
http://www.swarm-bots.org