Proyecto Fin de Carrera
Ingeniería Industrial

Análisis de optimización de las configuraciones de un robot SCARA vertical en aplicaciones aeronáuticas

Autor: Carolina García Padilla

Tutor: Ángel Rodríguez Castaño

Dep. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

Proyecto Fin de Carrera
Ingeniería Industrial

Análisis de optimización de las configuraciones de un robot SCARA vertical en aplicaciones aeronáuticas

Autor:

Carolina García Padilla

Tutor:

Ángel Rodríguez Castaño

Profesor titular

Dep. de Ingeniería de Sistemas y Automática

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2017

Proyecto Fin de Carrera: Análisis de optimización de las configuraciones de un robot SCARA vertical en aplicaciones aeronáuticas

Autor: Carolina García Padilla

Tutor: Ángel Rodríguez Castaño

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2017

El Secretario del Tribunal

Agradecimientos

Quisiera aprovechar para agradecer a mis padres por su apoyo y por creer en mí cuando yo no lo hacía; siento lo insoportable que me ponía cuando estaba de exámenes. A mí hermana, por ser la mejor cura contra el aburrimiento, ¿qué hice los nueve primeros años de mi vida sin ti? A mis eternas "niñas", por estar siempre ahí y hacerme llorar de la risa. A Ángel, por su amabilidad y la claridad de sus explicaciones, volvería a pedirle que fuera mi tutor sin dudarlo. Y por último, quisiera agradecer a todos los profesores que he tenido a lo largo de estos años, especialmente a aquellos que se preocuparon de que aprendiese.

Carolina García Padilla

Sevilla, 2017

Resumen/Abstract

Este proyecto trata sobre un robot manipulador tipo SCARA vertical usado en aplicaciones aeronáuticas. En primer lugar, se explicarán los motivos por los que se usa este robot y por los que se descartan otras alternativas. Posteriormente, se realizará un estudio más profundo del robot y de su entorno de trabajo. Por último, se analizará el efecto de usar diversos criterios de optimización del movimiento del robot. Para ello, se simulará el seguimiento de varias trayectorias con la ayuda de MATLAB.

This project is about a vertical SCARA robot for aeronautical applications. Firstly, we will explain the reasons why this robot is used instead of other alternatives. Afterwards, we will make a deeper study of the robot and its work environment. Finally, we will analyse the effect of following different robot motion optimization techniques. To this end, we will simulate in MATLAB how the robot follows several trajectories.

Índice

Agradecimientos **vii**

Resumen/Abstract **ix**

Índice **xi**

Índice de Figuras **xiii**

Índice de abreviaturas y acrónimos **xv**

1 Introducción **1**
1.1 Objetivos *1*
1.2 Antecedentes *1*
1.2.1 Descripción de los C/Bs 1
1.2.2 Ubicación de los C/Bs 2
1.2.3 Operación actual de los C/Bs 4
1.3 Alternativas al control de C/Bs *5*
1.3.1 Sustitución de los C/Bs existentes 6
1.3.2 Robot individual para cada C/B 6
1.3.3 Sistema robótico global 7
1.3.3.1 Robots comerciales 7
1.3.3.2 Robots a medida 12

2 Descripción del robot y el espacio de trabajo **15**
2.1 Introducción *15*
2.2 Espacio de trabajo *16*
2.3 Configuración del robot *16*
2.3.1 Geometría y modelo cinemático directo 16
2.3.2 Modelo cinemático inverso 17

3 Análisis de configuraciones **25**
3.1 Mapa de accesibilidad *25*
3.2 Criterios de optimización *29*
3.2.1 Criterio I - Máxima aproximación 29
3.2.2 Criterio II – Configuraciones preferidas 29
3.2.3 Criterio III – Mínimo tiempo invertido 30
3.2.3.1 Motores de movimiento simultáneo 30
3.2.3.2 Motores de movimiento secuencial 30

4 Resultados y Simulaciones **31**
4.1 Visualización *31*
4.1.1 Criterio I - Máxima aproximación 31
4.1.1.1 Simulación 1 31
4.1.1.2 Simulación 2 32
4.1.1.3 Simulación 3 32
4.1.1.4 Simulación 4 33
4.1.2 Criterio II – Configuraciones preferidas 33

4.1.2.1 Simulación 5 33
4.1.2.2 Simulación 6 34
4.1.2.3 Simulación 7 34
4.1.2.4 Simulación 8 35
4.1.3 Criterio III a. – Mínimo tiempo invertido (Motores en paralelo) 35
4.1.3.1 Simulación 9 35
4.1.3.2 Simulación 10 36
4.1.3.3 Simulación 11 37
4.1.3.4 Simulación 12 38
4.1.4 Criterio III b. – Mínimo tiempo invertido (Motores en serie) 38
4.1.4.1 Simulación 13 38
4.1.4.2 Simulación 14 39
4.1.4.3 Simulación 15 39
4.1.4.4 Simulación 16 40
4.2 Conclusiones y consideraciones futuras *40*

5 Códigos **41**
5.1 cb_coord.m *41*
5.2 inv_kin_geometry.m *42*
5.3 n_sol.m *47*
5.4 robot_pic.m *48*
5.5 q123_pic.m *50*
5.6 robot_movie.m *50*
5.7 t_horizontal.m *52*
5.8 t_vertical.m *53*

Referencias **55**

ÍNDICE DE FIGURAS

Fig. 1-1: Ejemplo de *circuit breaker* (C/B) 1

Fig. 1-2: Esquema de distribución eléctrica a través de C/Bs 2

Fig. 1-3: Estados abierto y cerrado de un C/B 2

Fig. 1-4: Puertas del EEPDC en el avión A400M 3

Fig. 1-5: Localización del EEPDC en un avión A400M 3

Fig. 1-6: Entorno del EEPDC en un avión A400M 4

Fig. 1-7: Esquema de la solución actual para el control de C/Bs 4

Fig. 1-8: Conexión del CBS-AIM al EEPDC mediante mazos de cables 5

Fig. 1-9: *Solid State Power Controller* (SSPC) y *Remote Control Circuit Breaker* (RCCB) 6

Fig. 1-10: Soporte metálico para los robots individuales de una fila de C/Bs 7

Fig. 1-11: Detalle de los dispositivos robóticos individuales 7

Fig. 1-12: Robot KR Agilus sixx de la firma alemana KUKA 8

Fig. 1-13: Espacio de trabajo de un robot manipulador articulado RRR 8

Fig. 1-14: Controlador KR C4 compact, 271 x 483 x 460mm (H x W x D) 9

Fig. 1-15: Robot cartesiano y correspondiente espacio de trabajo 9

Fig. 1-16: Robot cartesiano tipo *gantry* 9

Fig. 1-17: Módulos lineales serie Phyton de Adept 10

Fig. 1-18: SCARA horizontal de la marca japonesa EPSON 10

Fig. 1-19: Espacio de trabajo de un robot horizontal tipo SCARA 11

Fig. 1-20: Robot Jaco de la marca Kinova 11

Fig. 1-21: Robot cartesiano a medida 12

Fig. 1-22: Robot manipulador de 6 grados de libertad a medida 12

Fig. 1-23: Robot SCARA a medida 13

Fig. 1-24: Robot redundante tipo SCARA 13

Fig. 1-25: Solución con varios robots cartesianos 14

Fig. 1-26: Solución con varios robots SCARA 14

Fig. 2-1: Implementación física de un SCARA vertical de 3 GDL 15

Fig. 2-2: Relación entre cinemática directa e inversa 15

Fig. 2-3: Esquema del panel de C/Bs (unidades: m) 16

Fig. 2-4: Asignación de coordenadas articulares 17

Fig. 2-5: Resolución del problema cinemático inverso (I) 18

Fig. 2-6: Resolución del problema cinemático inverso (II) 18

Fig. 2-7: Resolución del problema cinemático inverso (III) 19
Fig. 2-8: Solución del problema cinemático inverso (configuración "a") 20
Fig. 2-9: Solución del problema cinemático inverso (configuración "b") 20
Fig. 2-10: Montaje "alternado" (arriba) frente a montaje "escalonado" (abajo) 21
Fig. 2-11: Colisión del efector y el brazo (I) 22
Fig. 2-12: Colisión del efector y el brazo (II) 22
Fig. 3-1: Mapa accesibilidad (*resol_ang* = 1º; *umbral_error* = 0.001 m; sin restr. art.) 25
Fig. 3-2: Mapa accesibilidad (*resol_ang* = 1º; *umbral_error* = 0.1/0.05/0.01/0.001 m; sin rest. art.) 26
Fig. 3-3: Diferencia en el nº de sol. usando *umbral_error* = 0.01 y *umbral_error* = 0.001 27
Fig. 3-4: Mapa accesibilidad (*resol_ang* = 1º; *umbral_error* = 0.001 m; con restr. art.) 28
Fig. 3-4: Mapas de accesibilidad sin restricciones y con restricciones 28
Fig. 4-1: Simulación 1 (Criterio I – Máxima aproximación) 31
Fig. 4-2: Simulación 2 (Criterio I – Máxima aproximación) 32
Fig. 4-3: Simulación 3 (Criterio I – Máxima aproximación) 32
Fig. 4-4: Simulación 4 (Criterio I – Máxima aproximación) 33
Fig. 4-5: Simulación 5 (Criterio II – Configuraciones preferidas) 33
Fig. 4-6: Simulación 6 (Criterio II – Configuraciones preferidas) 34
Fig. 4-7: Simulación 7 (Criterio II – Configuraciones preferidas) 34
Fig. 4-8: Simulación 8 (Criterio II – Configuraciones preferidas) 35
Fig. 4-9: Configuración de partida $q\_ant = \pi/2\ \ \pi/2\ \ \pi/2$ 35
Fig. 4-10: Simulación 9 (Criterio III – Mín. tiempo invertido, motores en paralelo) 36
Fig. 4-11: Simulación 10 (Criterio III – Mín. tiempo invertido, motores en paralelo) 36
Fig. 4-12: Configuración de partida $q\_ant = \pi/2\ \ -\pi/2\ \ -\pi/2$ 37
Fig. 4-13: Simulación 11 (Criterio III – Mín. tiempo invertido, motores en paralelo) 37
Fig. 4-14: Simulación 12 (Criterio III – Mín. tiempo invertido, motores en paralelo) 38
Fig. 4-15: Simulación 13 (Criterio III – Mín. tiempo invertido, motores en serie) 38
Fig. 4-16: Simulación 14 (Criterio III – Mín. tiempo invertido, motores en serie) 39
Fig. 4-17: Simulación 15 (Criterio III – Mín. tiempo invertido, motores en serie) 39
Fig. 4-18: Simulación 16 (Criterio III – Mín. tiempo invertido, motores en serie) 40

ÍNDICE DE ABREVIATURAS Y ACRÓNIMOS

2D: Espacio bidimensional

3D: Espacio tridimensional

C/B: Circuit Breaker

CAN: Controller Area Network

CATS: Computer Aided Test System

CBS-AIM: Circuit Breaker System – Aircraft Interface Module

EEPDC: Emergency Electrical Power Distribution Center

GDL: Grados de libertal

PC: Personal Computer

RCCB: Remote Control Circuit Breaker

RS232: Recommended Standard 232

SCARA: Selective Compliant Assembly Robot Arm

SSPC: Solid State Power Controller

USB: Universal Serial Bus

1 INTRODUCCIÓN

1.1 Objetivos

Durante la construcción de un avión, es fundamental la realización de un proceso de pruebas que permita verificar el funcionamiento de los subsistemas de dicho avión. En el presente proyecto, nos centraremos, concretamente, en el accionamiento eficiente, durante dicho proceso de pruebas, de unos dispositivos *circuit breakers* (C/Bs) ubicados en el avión A400M.

Para llevar a cabo el accionamiento de los C/Bs, se analizarán diversas alternativas al control actual de los C/Bs; siendo finalmente el dispositivo de control elegido un robot SCARA vertical de tres grados de libertad. Este robot será el objeto de estudio de la mayor parte del documento, realizándose un análisis de su espacio de trabajo, disposición geométrica y cinemática directa e inversa.

Finalmente, mediante la implementación en MATLAB del análisis anterior, se realizarán diversas simulaciones atendiendo a ciertas funciones de optimización indicadas más adelante. Esto nos permitirá optimizar las configuraciones adoptadas por el robot en el seguimiento de trayectorias verticales y horizontales; o lo que es lo mismo, en el accionamiento de columnas y filas de C/Bs durante la realización de las pruebas funcionales.

1.2 Antecedentes

1.2.1 Descripción de los C/Bs

Los *circuit breakers* (ver Fig. 1-1) son dispositivos de protección del cableado eléctrico ante sobrecalentamiento o sobreintensidades.

Fig. 1-1: Ejemplo de *circuit breaker* (C/B)

Los C/Bs se sitúan en los cables de alimentación de cada equipo, entre las barras de distribución de corriente eléctrica y dichos equipos, tal y como se representa en el esquema de la Fig. 1-2. El sistema de alimentación consta de barras de distribución de dos tipos: corriente alterna (115 VAC, 400 MHz) y corriente continua (28 VDC).

Un C/B tiene dos estados o posiciones posibles (Fig. 1-3): cerrado y abierto. Si se encuentra cerrado, se permite el paso de corriente eléctrica y, por tanto, el equipo correspondiente estará alimentado. Por el contrario, si el C/B se encuentra abierto, no se permite el paso de corriente eléctrica y, por consiguiente, el equipo no estará alimentado. El estado normal de un C/B es cerrado, de forma que si se produce, por ejemplo, un cortocircuito, la protección "saltará", y el C/B pasará a estar abierto.

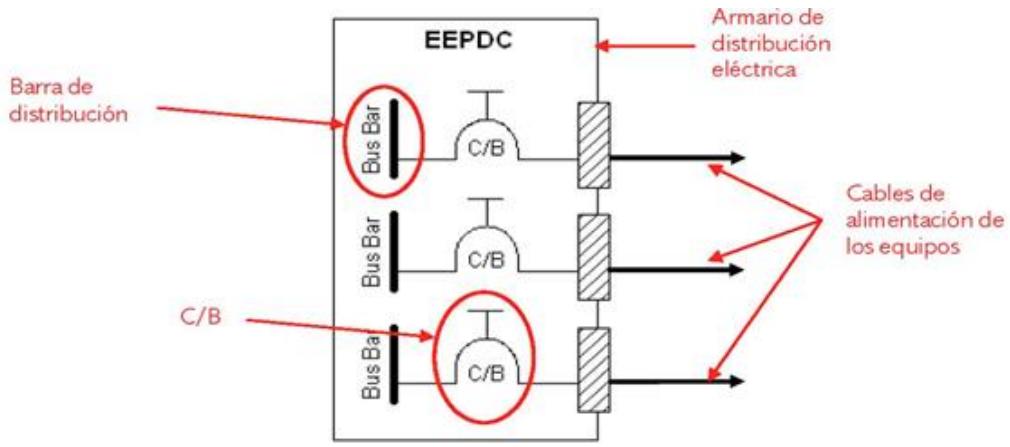

Fig. 1-2: Esquema de distribución eléctrica a través de C/Bs

El C/Bs mostrado en la Fig. 1-1 y esquematizado en la Fig. 1-3 tiene un recorrido entre las posiciones de abierto y cerrado de 4 mm y un pulsador de plástico de 10.2 mm de diámetro exterior, para el cual será necesario aplicar 2.5 kg de fuerza para cerrarlo y 2.66 kg para extraerlo.

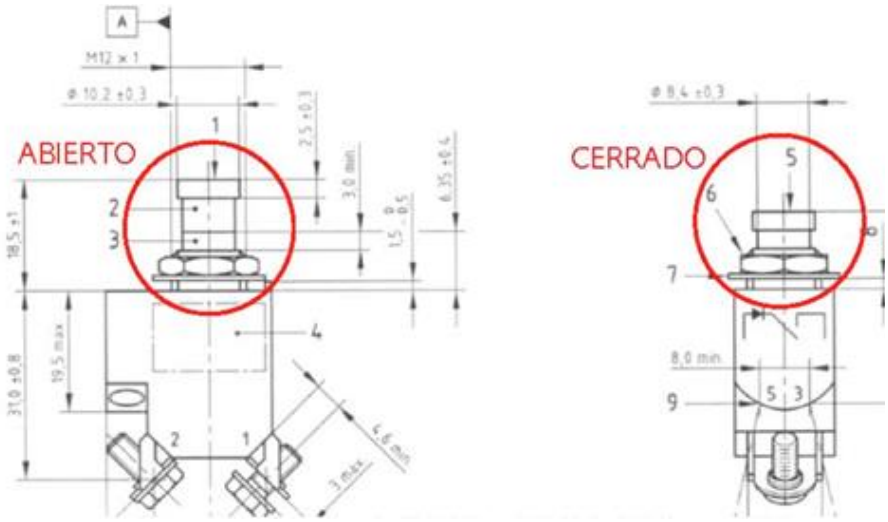

Fig. 1-3: Estados abierto y cerrado de un C/B

1.2.2 Ubicación de los C/Bs

La forma de controlar y monitorizar el estado de los C/Bs no se realiza de forma homogénea, sino que depende del tipo de avión. Por ejemplo, en los aviones C295 y CN235, los paneles con los C/Bs se hallan principalmente en el techo de la cabina, donde se accionan manualmente por un trabajador, según la documentación técnica de las instrucciones de pruebas. Esto es posible porque el número de C/Bs no es muy elevado.

En el avión A380, los C/Bs se controlan mediante un robot neumático que se fija a la puerta del armario de C/Bs. Este robot posiciona una pinza en el lugar indicado y abre o cierra el C/B extrayendo o empujando el accionamiento mecánico. El robot se controla desde una maleta que se sitúa cerca del armario de C/Bs. Este dispositivo requiere de un espacio que puede no estar disponible en otros aviones.

En el modelo A400M, los C/Bs se encuentran en las puertas o paneles exteriores del armario de distribución eléctrica del avión (EEPDC), tal y como se muestra en la Fig. 1-4. El EEPDC se sitúa en la parte delantera del avión, como se muestra en la Fig. 1-5.

Fig. 1-4: Puertas del EEPDC en el avión A400M

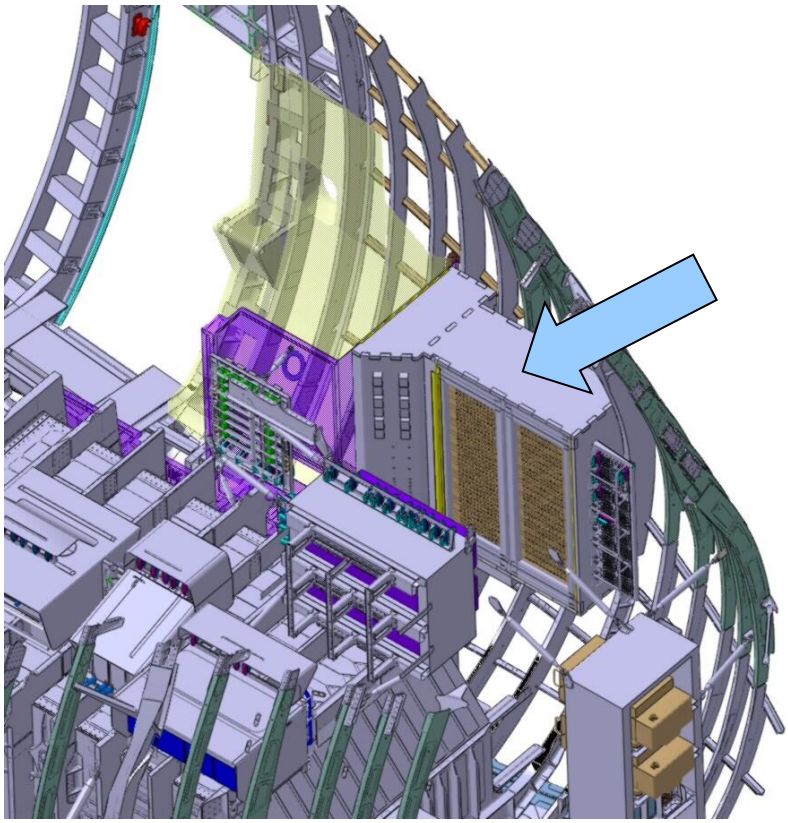

Fig. 1-5: Localización del EEPDC en un avión A400M

Una de las principales limitaciones en este caso es la escasez de espacio disponible en el lugar donde está ubicado el EEPDC. En la Fig. 1-6, se puede ver el espacio disponible con las puertas del EEPDC cerradas; se dispone de un área de poco más de 400 mm por 1000 mm, con un acceso de tan solo 300 mm, dificultado por una barra de la estructura del avión. Además, dicho espacio debe permanecer accesible a un trabajador para poder realizar tareas de *troubleshooting*.

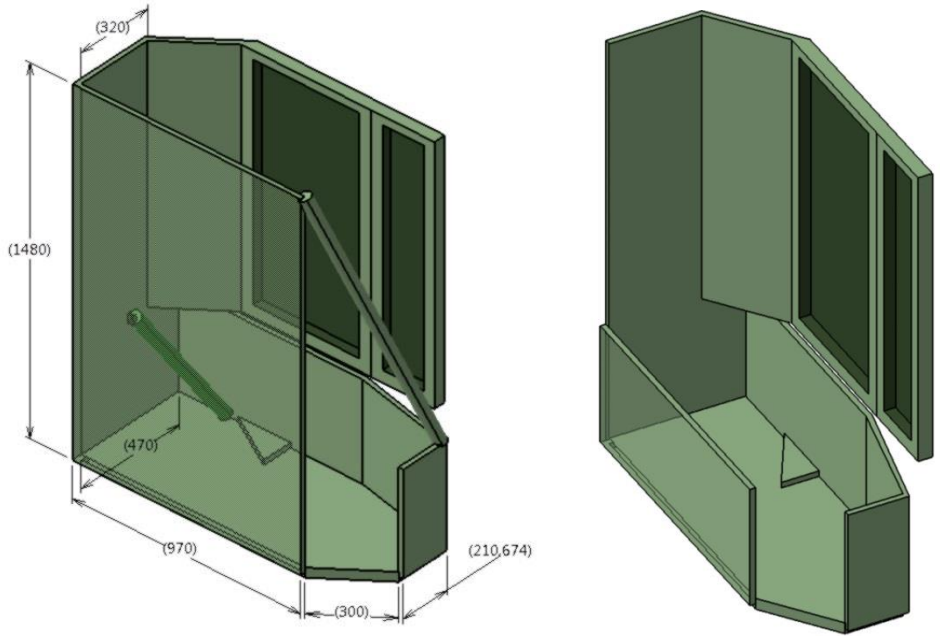

Fig. 1-6: Entorno del EEPDC en un avión A400M

1.2.3 Operación actual de los C/Bs

En el A400M, los C/Bs son controlados desde el sistema de pruebas funcionales CATS de Airbus Military. Este sistema permite la realización de las pruebas funcionales de los subsistemas de las aeronaves conforme se van ensamblando.

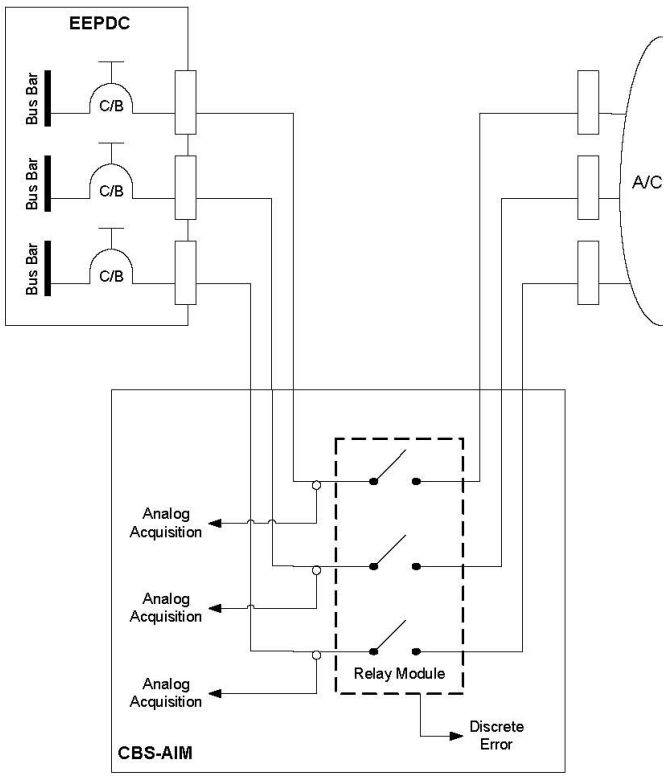

Fig. 1-7: Esquema de la solución actual para el control de C/Bs

Como puede observarse en el esquema de la Fig. 1-7, los C/Bs no se accionan físicamente, sino que se incorpora un relé en serie a cada uno de ellos. Estos relés son los que se controlan de forma automática desde CATS, y simulan la apertura y cierre de los C/Bs. Para que este esquema funcione, los C/Bs deben estar siempre cerrados.

Todos los relés están alojados en un armario fuera del avión, denominado CBS-AIM. Mediante mazos de cableado, la salida del armario de distribución eléctrica es conducida al CBS-AIM, y desde la salida de este, es llevada de nuevo a los mazos de avión que originalmente iban conectados al propio armario eléctrico.

Aunque la solución anterior es bastante robusta, tiene la desventaja de que si el número de C/Bs situados en el armario de distribución eléctrica es elevado, el número y las dimensiones de los mazos serán considerables. Esta situación puede observarse en la Fig. 1-8. En la actualidad, el montaje y desmontaje del CBS-AIM requiere un tiempo se considera excesivo para una futura producción.

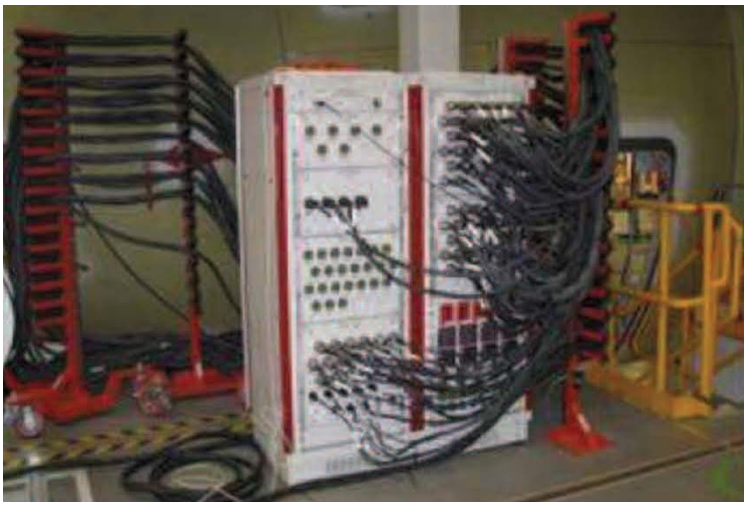

Fig. 1-8: Conexión del CBS-AIM al EEPDC mediante mazos de cables

Además, para poder operar, el sistema actual necesita tener disponible la red de datos interna de Airbus Military, lo que provoca la indisponibilidad puntual del sistema si la red no se encuentra disponible. Por lo tanto, se considera importante que haya una alternativa para el accionamiento de C/Bs, que pueda manejarse de forma independiente, sin necesidad de conectarse a la red de datos de Airbus Military.

1.3 Alternativas al control de C/Bs

Con el objetivo de mejorar las prestaciones existentes, se busca una alternativa al método de control de los C/Bs descrito en el apartado 1.2.3. Para ello, se desea diseñar un sistema que permita controlar de forma automática la apertura y cierre de los C/Bs, durante la ejecución de pruebas funcionales en el sistema CATS, y que implique las siguientes mejoras:

- Reducir los tiempos de instalación y ejecución.
- Reducir los costes de los equipos.
- Evitar la manipulación del cableado del avión.
- Reducir las necesidades de mantenimiento.

Se analizarán tres soluciones alternativas:

- Sustituir los C/Bs existentes.
- Disponer de un robot individual para cada C/B.
- Disponer de un sistema robótico global.

1.3.1 Sustitución de los C/Bs existentes

Esta solución consistiría en sustituir los C/Bs actuales por otros controlables de forma remota. Ya existen C/Bs de este tipo en el avión, empleados en los armarios secundarios, alimentando equipos con funciones no críticas; son los RCCBs y los SSPCs, similares a los que se muestran en la Fig. 1-9. Estos dispositivos tienen una parte mecánica y una parte electrónica que no pueden controlarse de manera independiente. Tienen una doble función: la protección eléctrica de los equipos y la conmutación de su alimentación. Esto permite controlar la alimentación de los equipos en función de unas señales de control. Estos C/Bs ya están perfectamente integrados en el sistema de pruebas CATS de Airbus Military.

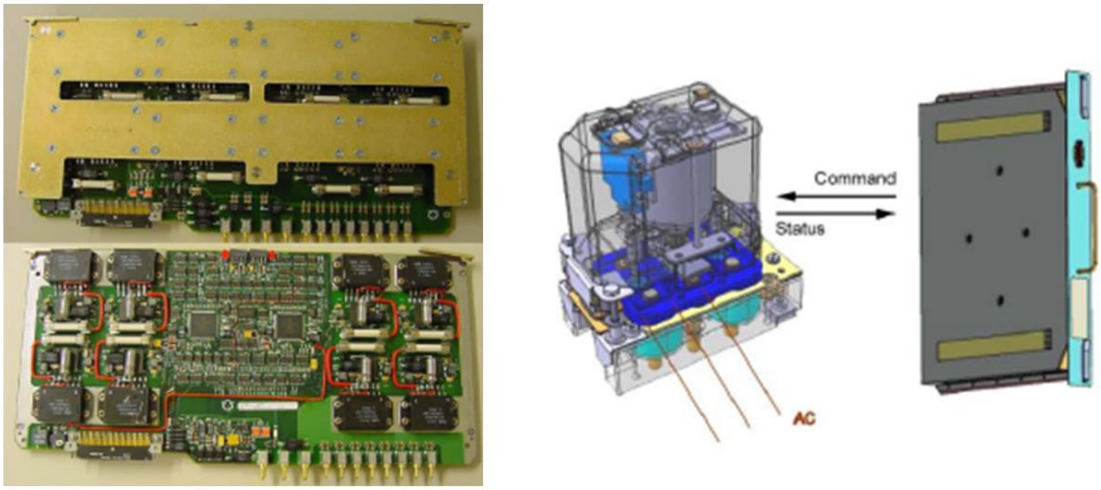

Fig. 1-9: *Solid State Power Controller* (SSPC) y *Remote Control Circuit Breaker* (RCCB)

Estos dispositivos tienen dos inconvenientes. El primero es su tamaño, mucho mayor que los C/Bs mecánicos, lo que impide que quepan en el espacio previsto en las aeronaves actualmente en producción; sería necesario modificar el diseño de dichas aeronaves. El segundo son las condiciones de operación; estos dispositivos necesitan energía eléctrica para funcionar, y podrían quedar inutilizados ante un fallo eléctrico, pudiendo comprometer la seguridad del avión.

1.3.2 Robot individual para cada C/B

La segunda opción sería disponer de un elemento robótico por cada C/B, de manera que pudieran accionarse individual o simultáneamente. La principal ventaja de este enfoque es la velocidad de actuación, gracias a la posibilidad de accionamiento simultáneo; y la robustez ante fallos, ya que si un dispositivo robótico fallara solo afectaría al C/B correspondiente, pero los demás seguirían estando operativos. Además, al tratarse de dispositivos idénticos, podrían sustituirse de manera inmediata por cualquier otro que no fuera necesario.

Sin embargo, dada la escasa separación entre los C/Bs de una misma fila, resulta poco factible diseñar e implementar un robot que pueda anclarse al armario para cada C/B. Una posible solución sería emplear un soporte metálico que vaya anclado a los laterales del armario, de forma que contenga los robots de una fila completa (ver Fig. 1-10 y Fig. 1-11).

Estos dispositivos individuales constarían de una pequeña pinza montada sobre un actuador lineal, de forma que el actuador lineal acerque la pinza al C/B, empujándolo para cerrarlo o sujetándolo y extrayéndolo para abrirlo. Esta solución dificultaría el acceso a los pulsadores, pero podría ser sencillo de desmontar. El principal inconveniente de esta solución sería la cantidad y la complejidad del cableado necesario, así como el elevado coste económico que supondría.

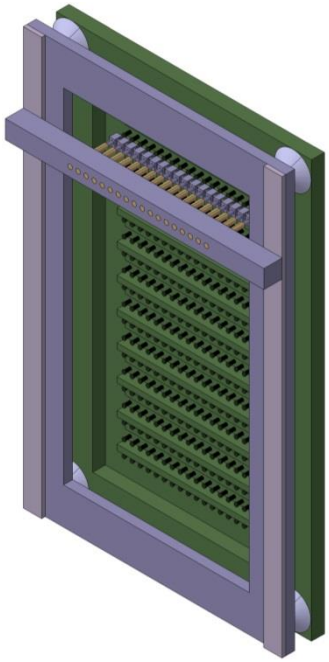

Fig. 1-10: Soporte metálico para los robots individuales de una fila de C/Bs

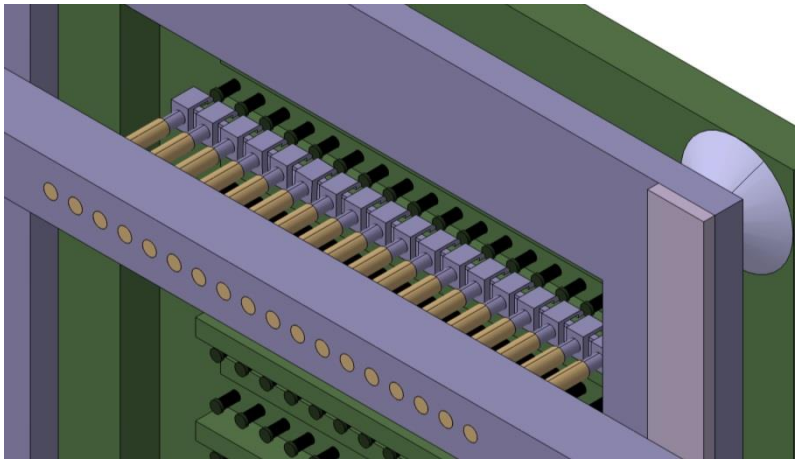

Fig. 1-11: Detalle de los dispositivos robóticos individuales

1.3.3 Sistema robótico global

La última alternativa consistiría en el uso de un robot manipulador que se encargue de accionar el C/B correspondiente, según se le indique. El manipulador posicionaría su efector final en un C/B y dicho efector final se encargaría del accionamiento.

1.3.3.1 Robots comerciales

A continuación, se analizarán varias soluciones basadas en robots comerciales. Estas incluyen tanto tecnologías sobradamente probadas, como son los manipuladores industriales, como nuevas tecnologías aplicadas en investigación, rehabilitación o interacción hombre-máquina.

Robot angular de 6 grados de libertad

Actualmente, la mayoría de los robots manipuladores industriales tienen como máximo seis grados de libertad. Estos manipuladores suelen clasificarse cinemáticamente de acuerdo con el carácter rotacional (R) o prismático (P) de sus tres primeras articulaciones.

Un robot angular RRR de 6 grados de libertad, como el que se muestra en la Fig. 1-12, presenta un espacio de trabajo como el de la Fig. 1-13. Para poder acceder a los extremos de los paneles se necesitarían unos eslabones de gran longitud. Esto, combinado con el hecho de que el robot no podría ubicarse muy cerca del panel para que el extremo accediese a los C/Bs más cercanos, supondría que el espacio ocupado sería demasiado grande.

Fig. 1-12: Robot KR Agilus sixx de la firma alemana KUKA

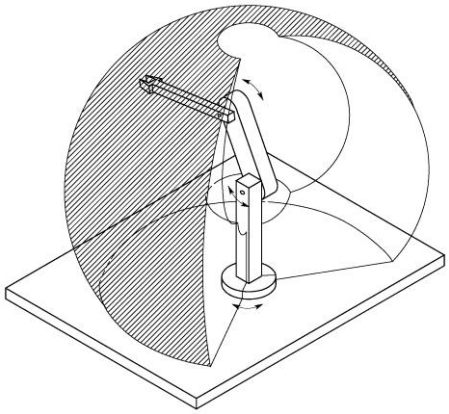

Fig. 1-13: Espacio de trabajo de un robot manipulador articulado RRR

Por otro lado, fijar el robot al panel de C/Bs presentaría dos inconvenientes. En primer lugar, el elevado peso de este tipo de robots; en segundo, la dificultad, debido a su morfología, de acceder a las zonas más cercanas a la base.

Además, estos manipuladores necesitan de un armario externo para albergar sus sistemas de control. Algunos de estos armarios presentan la ventaja de que pueden ubicar en su interior el sistema de control de varios manipuladores; pero dado el peso y tamaño que tienen, su ubicación quedaría restringida a un lugar fijo y exterior al avión. También es habitual que los fabricantes dispongan de una versión compacta del sistema de control (ver Fig. 1-14), en cuyo caso solo tienen capacidad para controlar un robot.

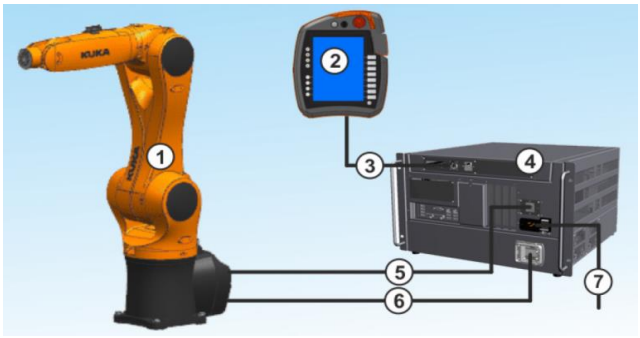

Fig. 1-14: Controlador KR C4 compact, 271 x 483 x 460mm (H x W x D)

Robot cartesiano

Las articulaciones prismáticas de este tipo de robots permiten el movimiento lineal a lo largo de los ejes cartesianos. Por definición, los robots cartesianos generan un espacio de trabajo con forma de prisma rectangular, tal y como se muestra en la Fig. 1-15.

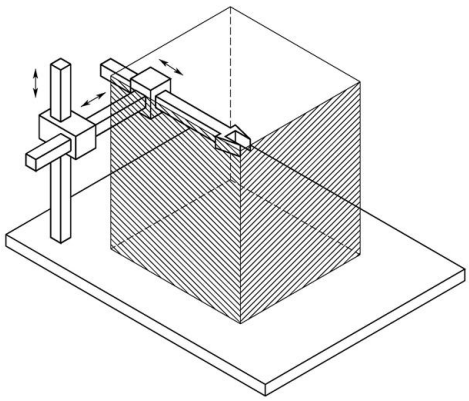

Fig. 1-15: Robot cartesiano y correspondiente espacio de trabajo

Una configuración de robot cartesiano especialmente adecuada para nuestro propósito de anclar el robot a un panel sería la del tipo pórtico o *gantry*, representado en la Fig. 1-16. Este robot se ubicaría en paralelo al panel de C/Bs, bien sea verticalmente (como en el caso del A400M) u horizontalmente en el techo (como en el CN235 y C295).

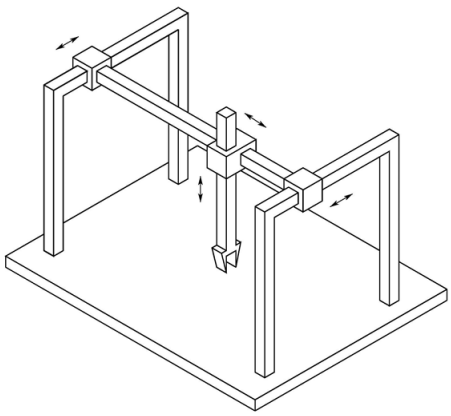

Fig. 1-16: Robot cartesiano tipo *gantry*

Al colocarse muy cerca y en paralelo a los paneles, el espacio que ocuparía y que necesitaría para operar sería relativamente pequeño. Además, dada la forma de su espacio de trabajo, se podría dividir el área total de operación en áreas independientes, de forma que podrían colocarse varios robots trabajando simultáneamente. Por ejemplo, se podría disponer de un robot por cada varias filas de C/Bs (ver el panel de la Fig. 1-4). Esto hace que la solución basada en robots cartesianos sea muy escalable.

Una desventaja es que no existen robots genéricos (*off-the-shelf*) de este tipo, sino que se desarrollan específicamente para una aplicación concreta a base de diversos módulos. Con estos módulos puede construirse un robot cartesiano de 3 grados de libertad como el mostrado en la Fig. 1-17 empleando la serie Phyton de Adept. Además, cada uno de estos módulos lineales incorpora un motor y su soporte, por lo que pueden llegar a ser bastante pesados. También son más lentos que los robots tipo SCARA que se describen en el siguiente apartado.

Fig. 1-17: Módulos lineales serie Phyton de Adept

El sistema de control de los robots cartesianos depende del fabricante. En algunos casos, consiste en un chasis o armario similar a los presentados en el apartado anterior; pero en otros, la electrónica de control está integrada en el propio actuador, como en el caso del fabricante Schunk, que también dispone de módulos lineales dentro de su serie PLS. En estos casos, el robot se conecta mediante una interfaz (bus CAN en el modelo de Schunk) a un PC externo, el cual hace las funciones de sistema de control.

Robots horizontales tipo SCARA

Como se puede observar en la Fig. 1-18, este manipulador posee dos articulaciones de rotación de ejes paralelos y una articulación prismática, lo que determina un espacio de trabajo como el representado en la Fig. 1-19. Como en el caso del robot cartesiano, se instalaría paralelo al panel de C/Bs.

Fig. 1-18: SCARA horizontal de la marca japonesa EPSON

Debido a la forma de su espacio de trabajo, presenta una escalabilidad complicada frente al robot cartesiano. Sería necesario coordinar los diversos robots, ya que habría zonas de trabajo comunes. También presenta el inconveniente de que los equipos de catálogo de los fabricantes son muy pesados, por lo que habría que recurrir a un desarrollo específico.

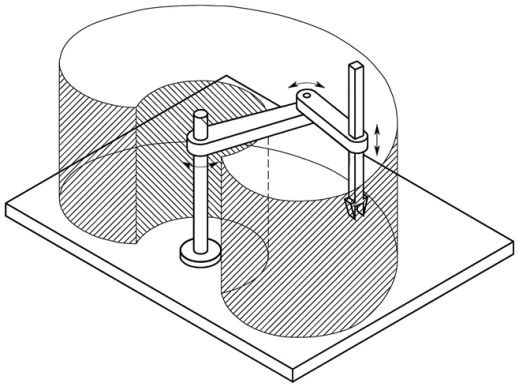

Fig. 1-19: Espacio de trabajo de un robot horizontal tipo SCARA

Robots especiales

Existen diversos robots manipuladores que, si bien no han sido probados en aplicaciones industriales, están teniendo un gran auge en el desarrollo de nuevas técnicas en cirugía, rehabilitación e investigación.

La mayoría de estos manipuladores tienen como principal diferencia respecto a los industriales que no necesitan de un armario externo para el sistema de control. En cada articulación del manipulador, se integra el actuador, la electrónica de potencia y el control del actuador. El software del sistema de control del robot se ejecuta en un ordenador convencional (en entorno Windows o Linux) que se comunica con el brazo manipulador por diversas interfaces (USB, Ethernet, RS232, bus CAN,…).

Un ejemplo de este tipo de robots es el Jaco de Kinova que se muestra en la Fig. 1-20. Tiene un efector terminal de 3 dedos cuyo movimiento se puede controlar de manera independiente, es de 6 grados de libertad y el software de control se ejecuta en un PC bajo Ubuntu o Linux, que se comunica con el manipulador mediante una interfaz USB 2.0 o bus CAN.

Fig. 1-20: Robot Jaco de la marca Kinova

Este tipo de robots resulta especialmente amigable cuando trabaja en un entorno humano; sin embargo, esta cualidad no es especialmente útil para nuestro propósito. Además, presenta otra serie de desventajas como su falta de robustez y velocidad, así como sus limitaciones en términos de repetibilidad y *payload*.

1.3.3.2 Robots a medida

Dado que no existen soluciones comerciales que cumplan de manera completamente óptima con las funcionalidades y prestaciones requeridas, existe la posibilidad de diseñar y fabricar un brazo robótico específico para esta aplicación.

Dentro de esta opción, es posible diseñar un robot cartesiano como se muestra en la Fig. 1-21, fabricándolo con motores lineales de precisión y con materiales rígidos pero ligeros para que el peso sea lo más reducido posible.

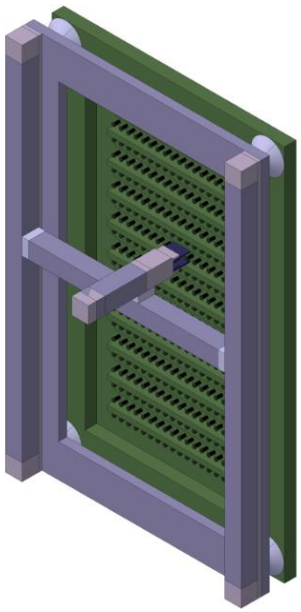

Fig. 1-21: Robot cartesiano a medida

Una opción menos interesante, debido al mayor volumen necesario para el espacio de trabajo del manipulador, es el diseño de un robot angular específico de 6 grados de libertad (ver Fig. 1-22). Su ventaja respecto al cartesiano es su mayor velocidad.

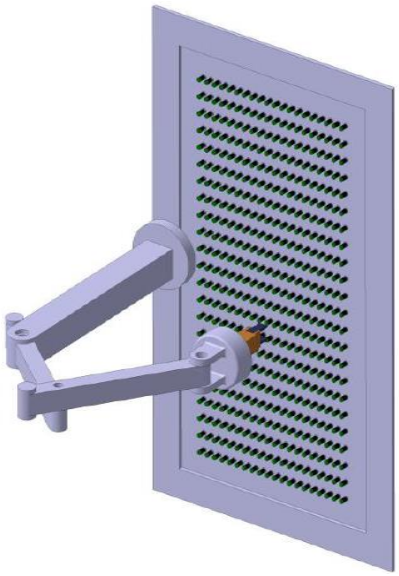

Fig. 1-22: Robot manipulador de 6 grados de libertad a medida

Una opción muy interesante en cuanto a velocidad y ocupación del espacio de trabajo en el caso del A400M es el diseño de un robot SCARA para trabajar en el plano vertical. Este robot consta de dos eslabones articulados por una unión de rotación, de forma que es posible cubrir todos los puntos de un plano. En el extremo del eslabón secundario se incorpora un actuador que accionaría los C/Bs y cuyo movimiento es lineal y perpendicular al plano del movimiento del robot. En la Fig. 1-23, se observa una representación de este robot montado sobre el panel de C/Bs.

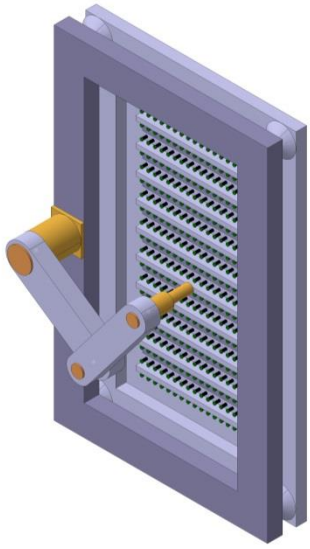

Fig. 1-23: Robot SCARA a medida

El principal inconveniente de esta configuración de SCARA vertical frente al SCARA horizontal tradicional es que la gravedad actúa en el mismo plano que el plano de movimiento de los motores, mientras que en la configuración horizontal es perpendicular; es decir, en el caso del SCARA vertical, los motores tienen que generar el par suficiente para mover el robot y además contrarrestar el efecto de la gravedad. Esto hace que se necesiten motores más potentes (y por tanto, más pesados y voluminosos) que en un SCARA horizontal.

Por último, es posible diseñar un robot redundante con morfología tipo SCARA como el que se muestra en la Fig. 1-24; un SCARA multieje que permita un movimiento más versátil y facilite que ningún elemento del robot salga del espacio de trabajo definido por el panel de C/Bs. Su principal inconveniente es el mismo que el señalado en el párrafo anterior; el tamaño y potencia de los motores para contrarrestar la gravedad, acrecentado en este caso por el hecho de tener más de 2 motores.

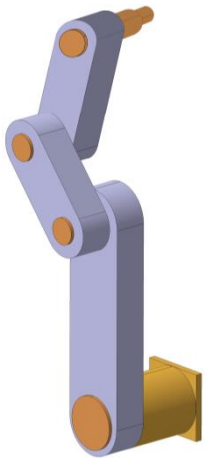

Fig. 1-24: Robot redundante tipo SCARA

Existe la posibilidad de instalar más de un brazo robótico para actuar simultáneamente sobre los C/Bs de un mismo panel. De esta forma, la velocidad de actuación sobre los C/Bs aumentaría considerablemente. Este tipo de configuración se consigue segregando el espacio de trabajo en zonas independientes para cada robot.

La manera más sencilla de segregar el espacio se consigue con robots cartesianos, tal y como se muestra en la Fig. 1-25. En este caso, incluso se podría llegar a disponer de un actuador por cada fila de C/Bs; de manera que, en cada robot, el eje vertical quedaría fijado y sólo se producirían desplazamientos en el horizontal. Esto haría que el sistema fuera más robusto ante fallos, ya que cada fila funcionaría de manera independiente; además, se eliminaría la necesidad de un segundo actuador para el movimiento vertical.

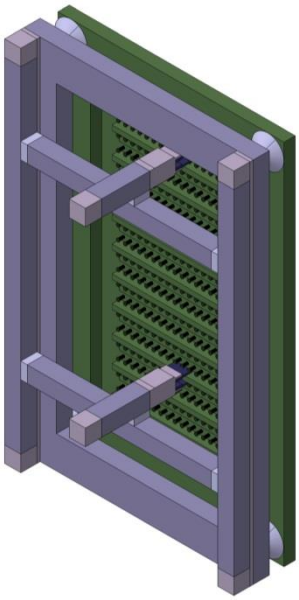

Fig. 1-25: Solución con varios robots cartesianos

Si se quisieran usar varios robots SCARA, como se muestra en la Fig. 1-26, dado que las zonas de trabajo no son complementarias de manera directa, sería necesario coordinar los robots o introducir restricciones en el espacio de trabajo para evitar que chocasen.

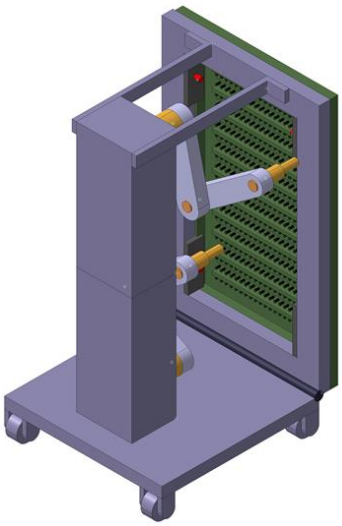

Fig. 1-26: Solución con varios robots SCARA

2 DESCRIPCIÓN DEL ROBOT Y EL ESPACIO DE TRABAJO

2.1 Introducción

Según lo visto en el capítulo anterior, la solución más adecuada para llevar a cabo el accionamiento de los C/Bs sería la de fabricar a medida un robot cartesiano o uno tipo SCARA vertical. A partir de ahora, nos centraremos exclusivamente en el estudio del robot SCARA, concretamente en uno de tres grados de libertad, como el representado anteriormente en la Fig. 1-24.

La decisión de construir el robot con tres eslabones se debe a que, de esta manera, se puede alcanzar cualquier posición del panel, siempre que la base del robot se coloque en el puntos medio del lado mayor del panel y que la suma de los eslabones sea como mínimo la necesaria para llegar a los extremos. En la Fig. 2-1, puede verse esta estructura ya implementada.

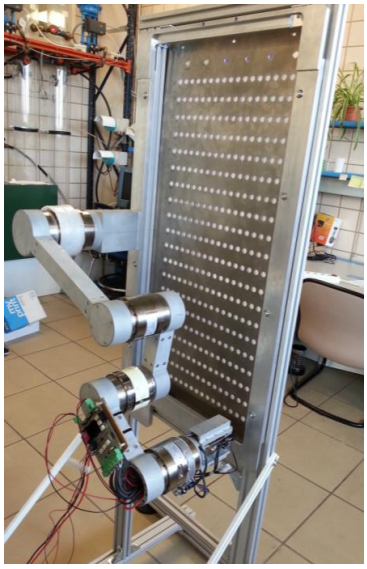

Fig. 2-1: Implementación física de un SCARA vertical de 3 GDL

A continuación, se analizarán las dimensiones teóricas tanto del espacio de trabajo, entendiéndose como tal el panel de C/Bs, como del manipulador. Además, para este último, se resolverán los problemas cinemático directo e inverso (ver Fig. 2-2), con el objetivo de implementarlos en MATLAB.

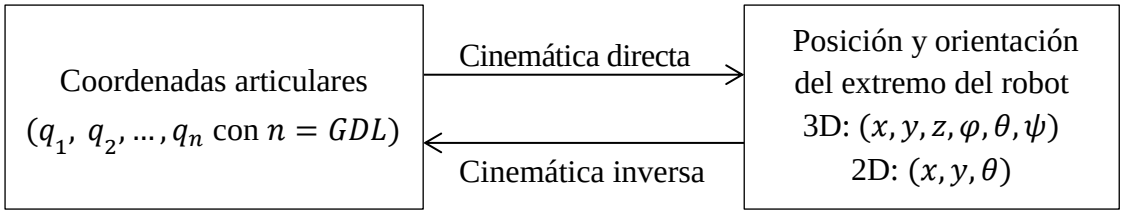

Fig. 2-2: Relación entre cinemática directa e inversa

2.2 Espacio de trabajo

El espacio de trabajo será un panel de 880 mm de alto por 470 mm de ancho. La base del robot se encuentra en el punto medio del extremo izquierdo del panel. El panel contiene 21 filas y 18 columnas de C/Bs, con una separación de 40 mm y 20 mm respectivamente. La primera columna de C/Bs se encuentra a 100 mm del límite izquierdo; y la última, a 30 mm del límite derecho. La primera y la última fila tienen una separación de 40 mm con los límites superior e inferior respectivamente.

Ejecutando el fichero de MATLAB **cb_coord.m**, se obtiene una representación esquemática del panel descrito, considerándose como origen de coordenadas el C/B inferior izquierdo (ver Fig. 2-3).

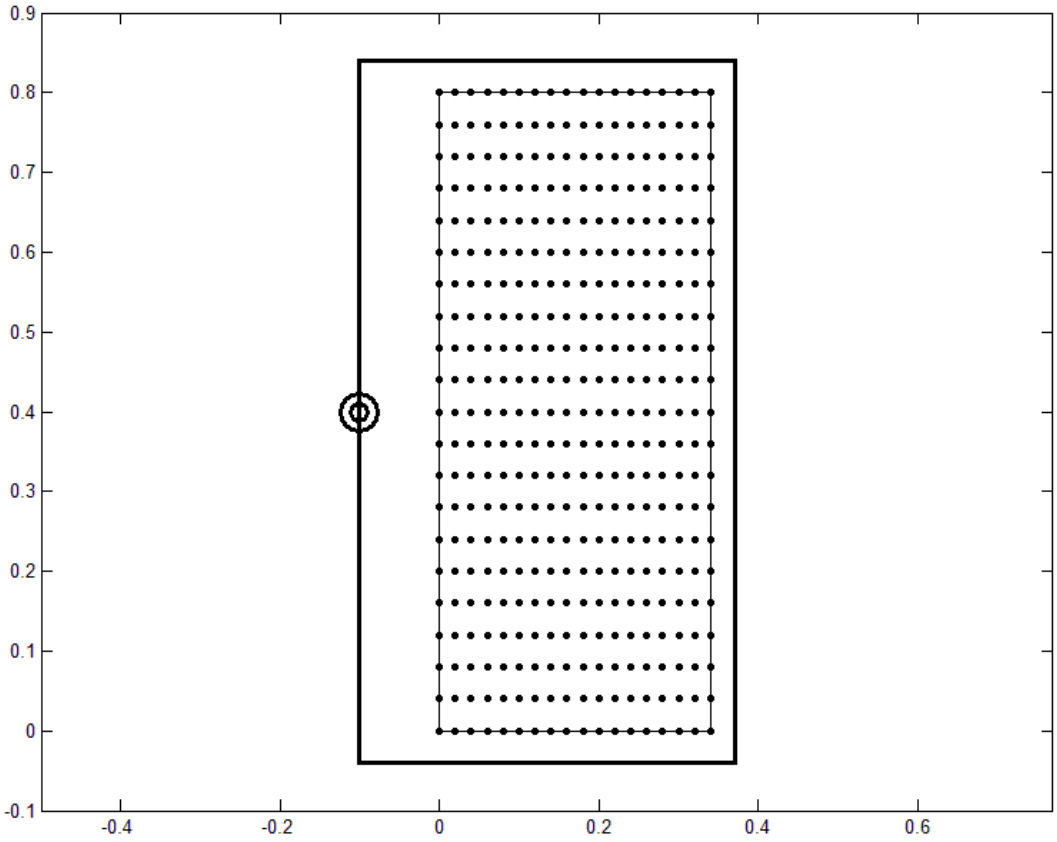

Fig. 2-3: Esquema del panel de C/Bs (unidades: m)

2.3 Configuración del robot

2.3.1 Geometría y modelo cinemático directo

Como ya hemos mencionado, la base del robot, la cual coincide con su primera articulación, se ancla en el punto medio del borde izquierdo del panel. El primer eslabón (brazo) tiene una longitud de 380 mm. El segundo eslabón (antebrazo) y el tercero (muñeca) miden lo mismo, 186 mm. Con estas proporciones, se facilita que el brazo robótico quede contenido dentro del panel.

La resolución del problema cinemático directo consiste en determinar la posición y la orientación que adopta el extremo del robot a partir de las variables que fijan la posición u orientación de sus articulaciones, es decir, de sus coordenadas articulares.

En la Fig. 2-4, se definen las coordenadas articulares de nuestro robot. La coordenada q_1 se define como el ángulo que forma el eje $y < 0$ con el brazo; la coordenada q_2, como el ángulo que forman la extensión del eje del brazo con el antebrazo; y la coordenada q_3, como el ángulo que forman la extensión del eje del antebrazo con la muñeca. Todos ellas se consideraran positivas en sentido antihorario y negativas en sentido horario. Por consiguiente, en el ejemplo de la Fig. 2-4, q_1 es positiva, mientras que q_2 y q_3 son negativas.

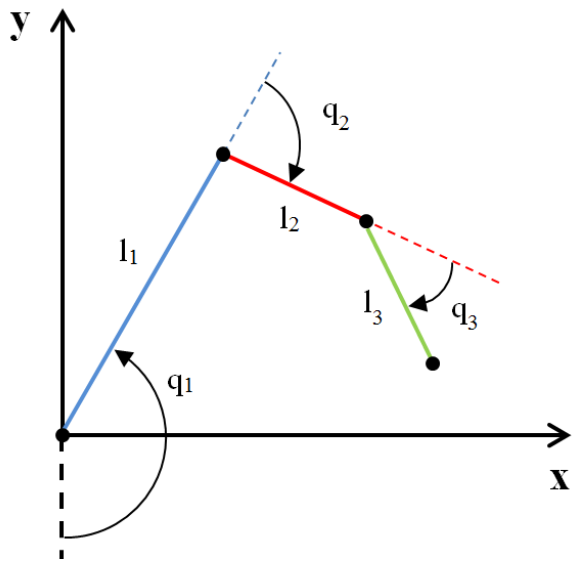

Fig. 2-4: Asignación de coordenadas articulares

La obtención del modelo cinemático directo puede ser abordado mediante métodos geométricos o mediante métodos basados en cambios de sistemas de referencia. Como nuestro robot posee una estructura virtualmente plana de 3 GDL, resulta sencillo encontrar las siguientes relaciones mediante simples consideraciones geométricas (ver Fig. 2-4):

$$x = l_1 \cdot sen(q_1) + l_2 \cdot sen(q_1 + q_2) + l_3 \cdot sen(q_1 + q_2 + q_3)$$
$$y = -l_1 \cdot cos(q_1) - l_2 \cdot cos(q_1 + q_2) - l_3 \cdot cos(q_1 + q_2 + q_3) \quad [2\text{-}1]$$

Las expresiones [2-9] permiten conocer la localización del extremo del robot referidas al sistema de la base en función de las coordenadas articulares del robot q_1, q_2 y q_3, correspondiendo, por tanto, a la solución del problema cinemático directo. La obtención del ángulo (θ) que indique la orientación con la que se posicionara el efector final encima del correspondiente C/B resulta trivial y no se estudiará con mayor profundidad.

2.3.2 Modelo cinemático inverso

El objetivo del problema cinemático inverso consiste en encontrar los valores que deben adoptar las coordenadas articulares del robot para que su extremo se posicione y oriente según una determinada localización espacial. El problema cinemático inverso se resuelve con el fichero **inv_kin_geometry.m**.

En primer lugar, se comprueba que las coordenadas (x, y) en las que se quiere posicionar el extremo del manipulador se encuentran en el área que contiene los C/Bs, esto se lleva a cabo con la función "check_pos".

A continuación, para considerar las diferentes orientaciones con las que la muñeca puede alcanzar el punto objetivo, se realiza un barrido del ángulo de cabeceo, definido tal y como se muestra en la Fig. 1-1Fig. 2-5.

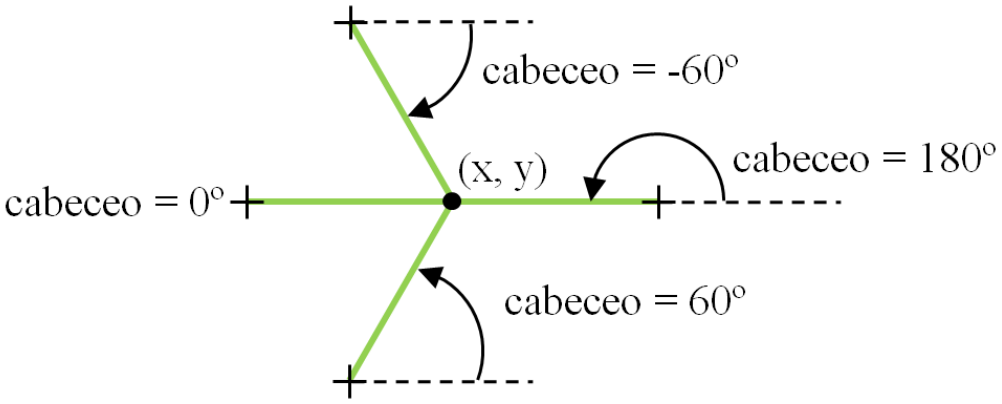

Fig. 2-5: Resolución del problema cinemático inverso (I)

De esta forma, para cada valor del ángulo de cabeceo, se calculan las coordenadas de la base de la muñeca (x_{b3}, y_{b3}) , tal y como se muestra en la Fig. 2-6:

$$\Delta x = l_3 \cdot \cos(cabeceo)$$

$$\Delta y = l_3 \cdot \text{sen}(cabeceo)$$

$$x_{b3} = x - \Delta x$$

$$y_{b3} = y - \Delta y$$

[2-2]

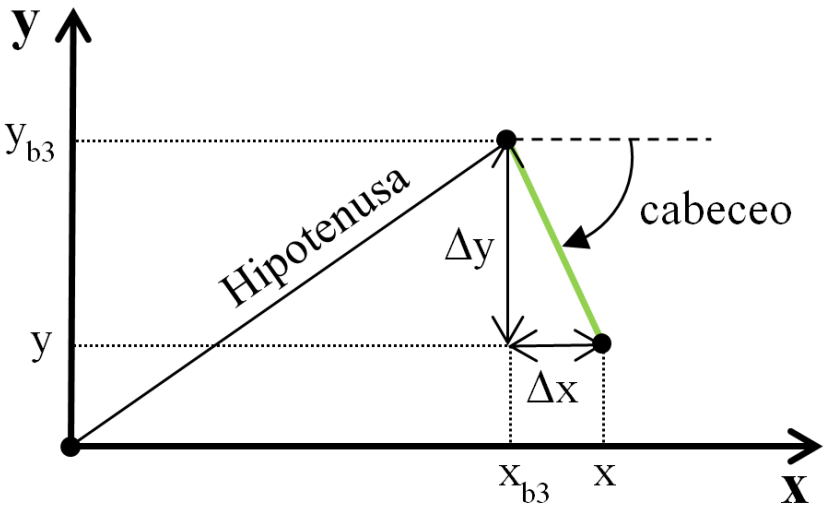

Fig. 2-6: Resolución del problema cinemático inverso (II)

A continuación, se comprueba que la longitud total del brazo y el antebrazo es suficiente para alcanzar la posición (x_{b3}, y_{b3}). Para ello, se calcula la siguiente distancia:

$$Hipotenusa = \sqrt{(x_{b3})^2 + (y_{b3})^2} \qquad [2\text{-}3]$$

De forma que si se cumple que $l_1 + l_2 \geq Hipotenusa$, continuamos resolviendo el problema.

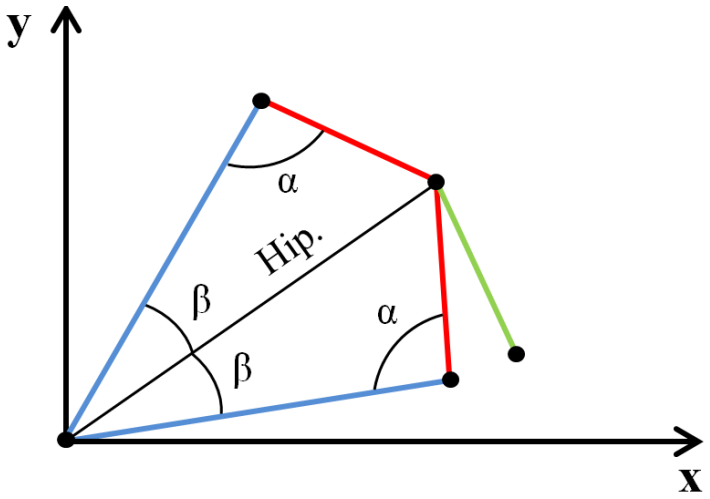

Fig. 2-7: Resolución del problema cinemático inverso (III)

Haciendo uso del teorema del coseno, se calcula el ángulo α que forman el brazo y el antebrazo:

$$Hipotenusa^2 = l_1^2 + l_2^2 - 2 \cdot l_1 \cdot l_2 \cdot \cos(\alpha)$$

$$\alpha = arcos\left(\frac{l_1^2 + l_2^2 - Hipotenusa^2}{2 \cdot l_1 \cdot l_2}\right) \quad [2\text{-}4]$$

Aplicando de nuevo el teorema del coseno, se obtiene el ángulo β que forman el brazo con la línea que une la base del robot con la base de la articulación de la muñeca ($Hipotenusa$):

$$l_2^2 = Hipotenusa^2 + l_1^2 + -2 \cdot Hipotenusa \cdot l_1 \cdot \cos(\beta)$$

$$\beta = arcos\left(\frac{Hipotenusa^2 + l_1^2 - l_2^2}{2 \cdot Hipotenusa \cdot l_1}\right) \quad [2\text{-}5]$$

Tal y como se representa en la Fig. 2-7, con estos ángulos se pueden formar dos configuraciones distintas. Para continuar resolviendo el problema, se calcula el ángulo b_3 que forma la $Hipotenusa$ y el eje $x > 0$.

$$b_3 = \arctan\left(\frac{y_{b3}}{x_{b3}}\right) \quad [2\text{-}6]$$

En el caso el caso de la configuración "a", representada en la Fig. 2-8, las coordenadas articulares se calculan como:

$$q_{1a} = \frac{\pi}{2} + b_3 + \beta$$

$$q_{2a} = -\pi + \alpha \quad [2\text{-}7]$$

$$q_{3a} = cabeceo - q_{1a} - q_{2a} + \frac{\pi}{2}$$

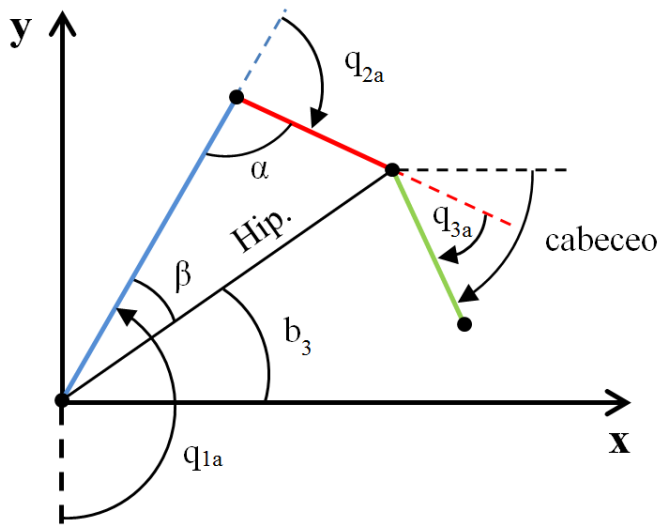

Fig. 2-8: Solución del problema cinemático inverso (configuración "a")

Análogamente, para la configuración "b" (Fig. 2-9):

$$q_{1a} = \frac{\pi}{2} + b_3 - \beta$$

$$q_{2a} = \pi - \alpha \qquad [2\text{-}8]$$

$$q_{3a} = cabeceo - q_{1a} - q_{2a} + \frac{\pi}{2}$$

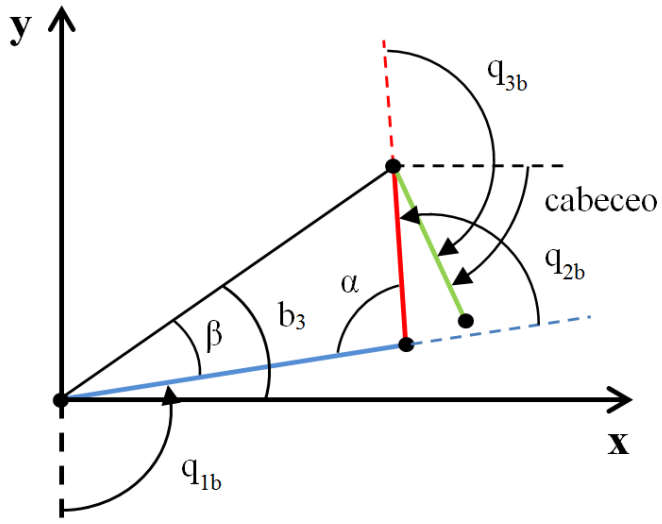

Fig. 2-9: Solución del problema cinemático inverso (configuración "b")

Con el objetivo de comprobar que no se hayan ido acumulando errores de cálculo, se resuelve el problema cinemático directo con las coordenadas articulares $q_a = [q_{1a}\ q_{2a}\ q_{3a}]$ y $q_b = [q_{1b}\ q_{2b}\ q_{3b}]$. El resultado obtenido se compara con las coordenadas de partida (x, y) del problema cinemático inverso; descartando aquellas configuraciones que según el modelo cinemático directo posicionen el efector a una distancia mayor de un cierto valor definido por la variable *umbral_error*. Por ejemplo, para la configuración "a", el procedimiento sería el siguiente (siendo análogo para la configuración "b"):

- Con la función "din_kin" y usando las coordenadas articulares $q_a = [q_{1a}\ q_{2a}\ q_{3a}]$, se resuelve la cinemática directa, obteniéndose las coordenadas articulares (x_a, y_a):

$$x_a = l_1 \cdot sen(q_{1a}) + l_2 \cdot sen(q_{1a} + q_{2a}) + l_3 \cdot sen(q_{1a} + q_{2a} + q_{3a})$$
$$y_a = -l_1 \cdot cos(q_{1a}) - l_2 \cdot cos(q_{1a} + q_{2a}) - l_3 \cdot cos(q_{1a} + q_{2a} + q_{3a}) \quad [2\text{-}9]$$

- Con la función "dist_cart", se calcula la distancia entre (x_a, y_a) y el punto objetivo (x, y):

$$d_a = \sqrt{(x_a - x)^2 + (y_a - y)^2} \quad [2\text{-}10]$$

- Si $d_a < umbral\_error$, la configuración articular $q_a = [q_{1a}\ q_{2a}\ q_{3a}]$ se considera válida. En caso contrario, se descarta dicha configuración.

Aunque, idealmente, pueda considerarse todo el rango de amplitud de las coordenadas articulares, en la práctica es habitual que existan ciertas restricciones. Estas restricciones articulares pueden deberse a cuestiones estructurales (motores, cableado, grosor y montaje de los eslabones, etc.) o de limitación del espacio. Por ejemplo, si la unión de los eslabones es "alternada" en lugar de "escalonada" (ver Fig. 2-10), se obtiene una estructura más compacta pero con menos capacidad para plegarse.

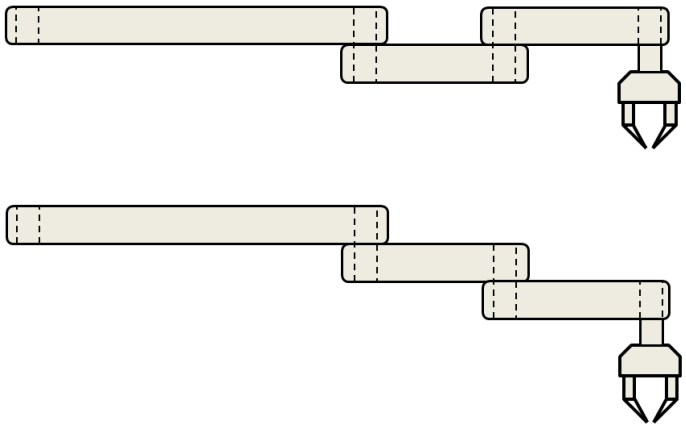

Fig. 2-10: Montaje "alternado" (arriba) frente a montaje "escalonado" (abajo)

La función "check_angles" comprueba si se está pidiendo que se consideren las restricciones articulares y, de ser así, descarta aquellas configuraciones que las incumplan. Se debe cumplir que:

✓ El brazo o primer eslabón no gire de forma que sobresalga del marco izquierdo del panel, es decir:

$$180° \leq q_1 \leq 0° \quad [2\text{-}11]$$

✓ El antebrazo no se pliegue completamente sobre el brazo, dejándose como mínimo un margen de 25º:

$$180° - 25 \leq q_2 \leq -180° + 25° \Rightarrow 155° \leq q_2 \leq -155° \quad [2\text{-}12]$$

✓ La muñeca no se pliegue completamente sobre el brazo, dejándose como mínimo un margen de 35º:

$$180° - 35 \leq q_3 \leq -180° + 35° \Rightarrow 145° \leq q_3 \leq -145° \quad [2\text{-}13]$$

✓ El extremo del robot no colisione con el brazo. Para encontrar la condición necesaria para asegurar que no se produzca la colisión, estudiamos el caso de la Fig. 2-11:

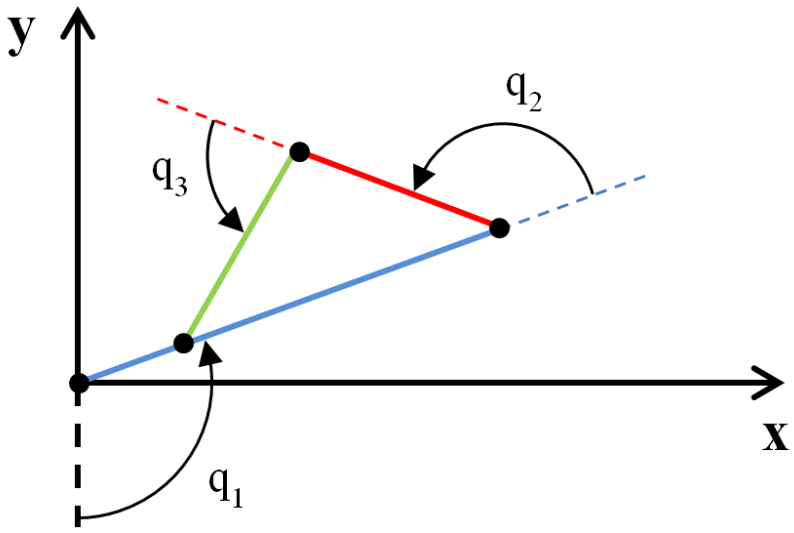

Fig. 2-11: Colisión del efector y el brazo (I)

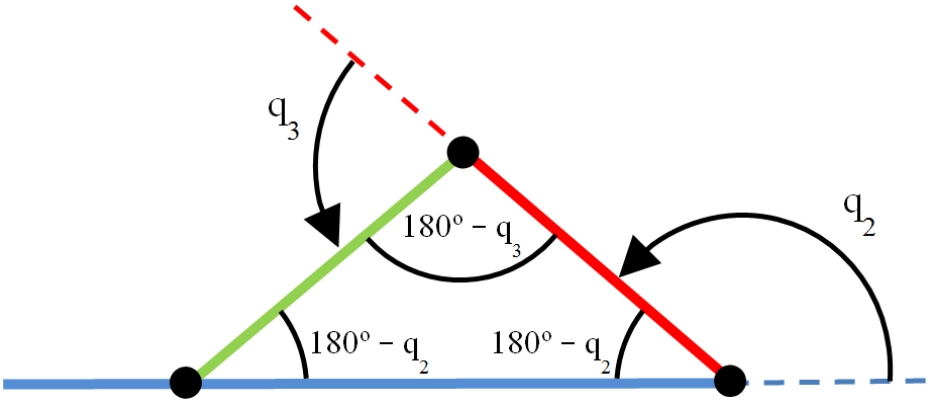

Fig. 2-12: Colisión del efector y el brazo (II)

Como $l_2 = l_3$, los eslabones chocarán cuando formen un triángulo isósceles como el de la Fig. 2-12, es decir, cuando se cumpla que:

$$2 \cdot (180° - q_2) + (180° - q_3) = 180° \Rightarrow 2 \cdot q_2 + q_3 = 360° \quad [2\text{-}14]$$

Por lo tanto, no se produce colisión cuando:

$$2 \cdot (180° - q_2) + (180° - q_3) > 180° \Rightarrow 2 \cdot q_2 + q_3 < 360° \quad [2\text{-}15]$$

Sin embargo, para dejar cierto margen de seguridad, la expresión [2-15] se restringe a:

$$2 \cdot q_2 + q_3 < 345° \quad [2\text{-}16]$$

También existe la posibilidad de que el robot choque con el antebrazo y la muñeca hacia abajo, o lo que es lo mismo, cuando $q_2 < 0$ y $q_3 < 0$. En este caso, la colisión no se producirá cuando:

$$2 \cdot (180° + q_2) + (180° + q_3) > 180° \Rightarrow 2 \cdot q_2 + q_3 < 360° \quad [2\text{-}17]$$

Y restringiendo de nuevo:

$$2 \cdot q_2 + q_3 > -345° \quad [2\text{-}18]$$

Finalmente, se realiza un proceso de selección para elegir la configuración más óptima de entre todas las obtenidas durante la resolución del problema cinemático inverso. Este proceso se detallará en el capítulo siguiente.

3 ANÁLISIS DE CONFIGURACIONES

3.1 Mapa de accesibilidad

El número de soluciones obtenidas durante el proceso de resolución del problema cinemático inverso depende de varios factores. El primero es la resolución angular con la que se realiza el barrido del ángulo de cabeceo y que se define con la variable *resol_ang* en el fichero **inv_kin_geometry.m**.

Con el valor de la resolución angular, podemos calcular el número de soluciones máximas que idealmente pueden encontrarse para cada localización espacial del extremo del robot. Por ejemplo, para una resolución angular de 1º, se tendrán 360 valores del ángulo de cabeceo; y teniendo en cuenta que para cada valor del ángulo de cabeceo se obtenían dos posibles configuraciones (configuración "a" y configuración "b"), se obtendrá un máximo de 720 soluciones. Ahora bien, podría ocurrir que la longitud de los eslabones no fuera suficiente para alcanzar el punto objetivo con los 360 valores del ángulo de cabeceo; en cuyo caso, se obtendría un número menor de soluciones. Usaremos siempre una resolución angular de 1º, puesto que con 10º solo se obtienen 72 soluciones como máximo, y el tiempo de ejecución de usar 0.1º es demasiado alto.

Otros factores que influyen en el número de soluciones son la variable *umbral_error* y la restricción de las coordenadas articulares; ya mencionados en el punto 2.3.2.

Con el fichero **n_sol.m**, se obtiene un mapa de accesibilidad del panel, es decir, una representación del número de soluciones obtenidas para cada una de las posiciones de los C/Bs que se denotarán como (columna, fila). De esta forma, podemos estudiar el efecto que tienen los factores anteriores. Por ejemplo, en la Fig. 3-1, se representa el mapa de accesibilidad para una resolución angular de 1°, un *umbral_error* de 1 mm y sin que se consideren las restricciones articulares. Como cabía esperar, las zonas de más difícil acceso (y, por tanto, con menos soluciones) son las esquinas superior e inferior derechas y las proximidades a la base del robot.

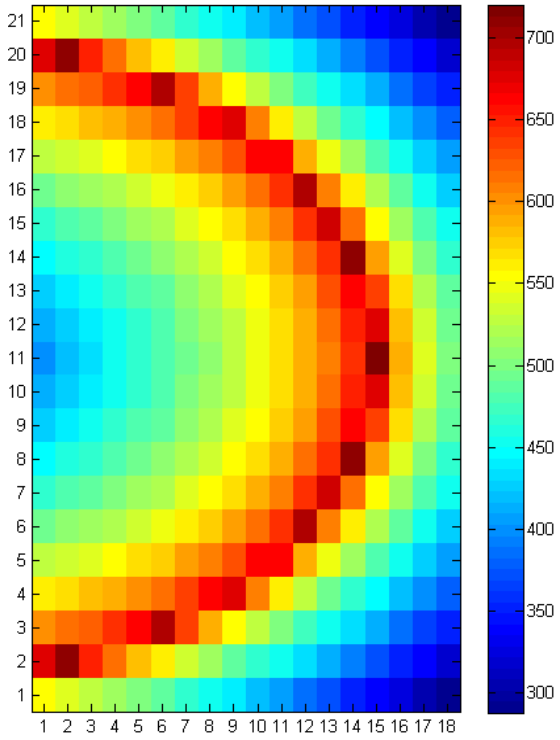

Fig. 3-1: Mapa accesibilidad (*resol_ang* = 1º; *umbral_error* = 0.001 m; sin restr. art.)

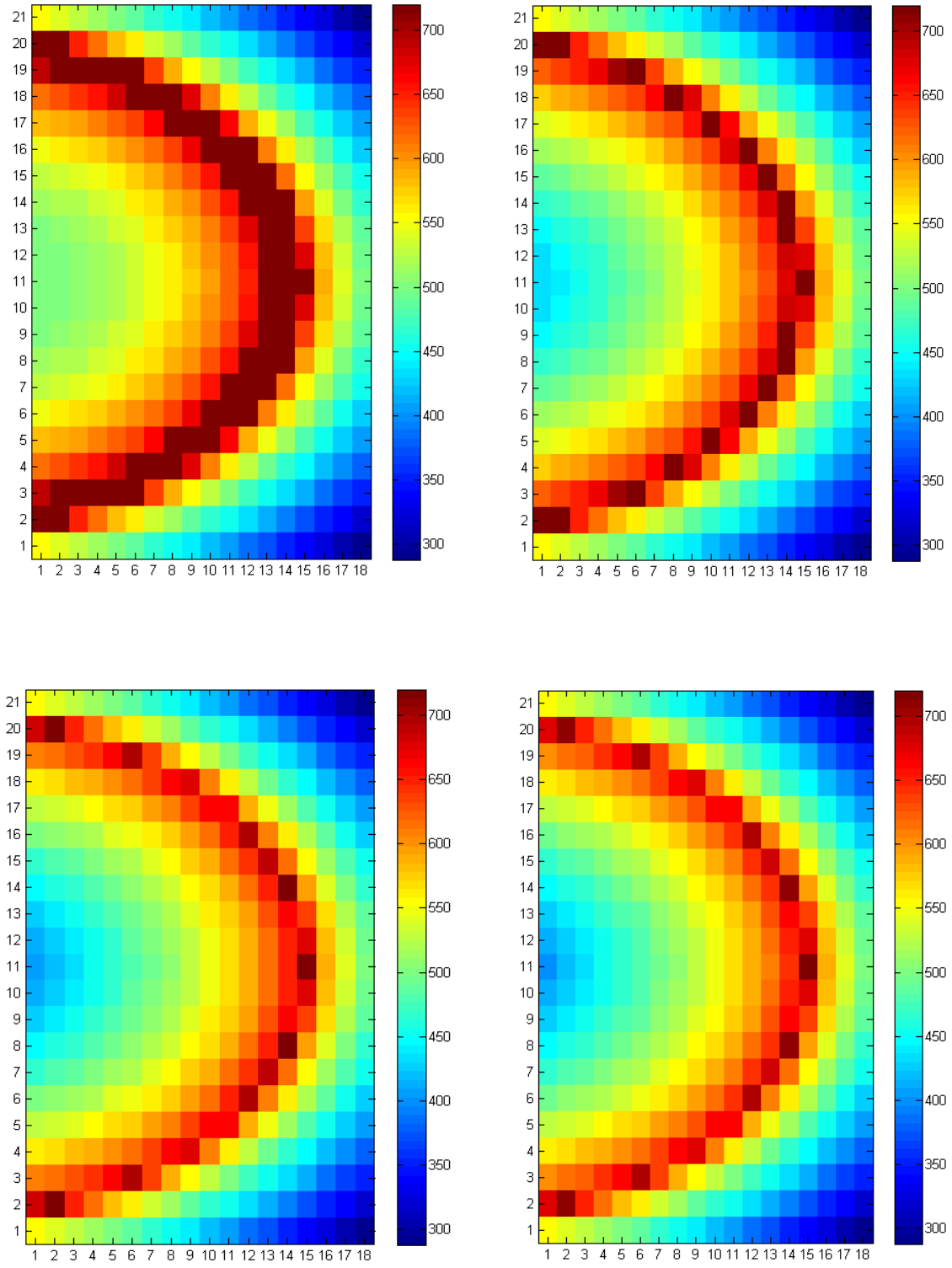

Fig. 3-2: Mapa accesibilidad (*resol_ang* = 1º; *umbral_error* = 0.1/0.05/0.01/0.001 m; sin rest. art.)

En la Fig. 3-2, se representa el mismo caso de la Fig. 3-1 para distintos valores de *umbral_error*. A simple vista, parece que no hay diferencia entre usar 10 mm y 1 mm, pero si restamos el número de soluciones de ambas (Fig. 3-3), se comprueba que con 1 mm se siguen restringiendo soluciones. Como la diferencia entre un caso y otro en términos de tiempo computacional es imperceptible, usaremos el valor de 1 mm. Además, puede demostrarse que usar un valor de 0.1 mm no restringe más el número de soluciones.

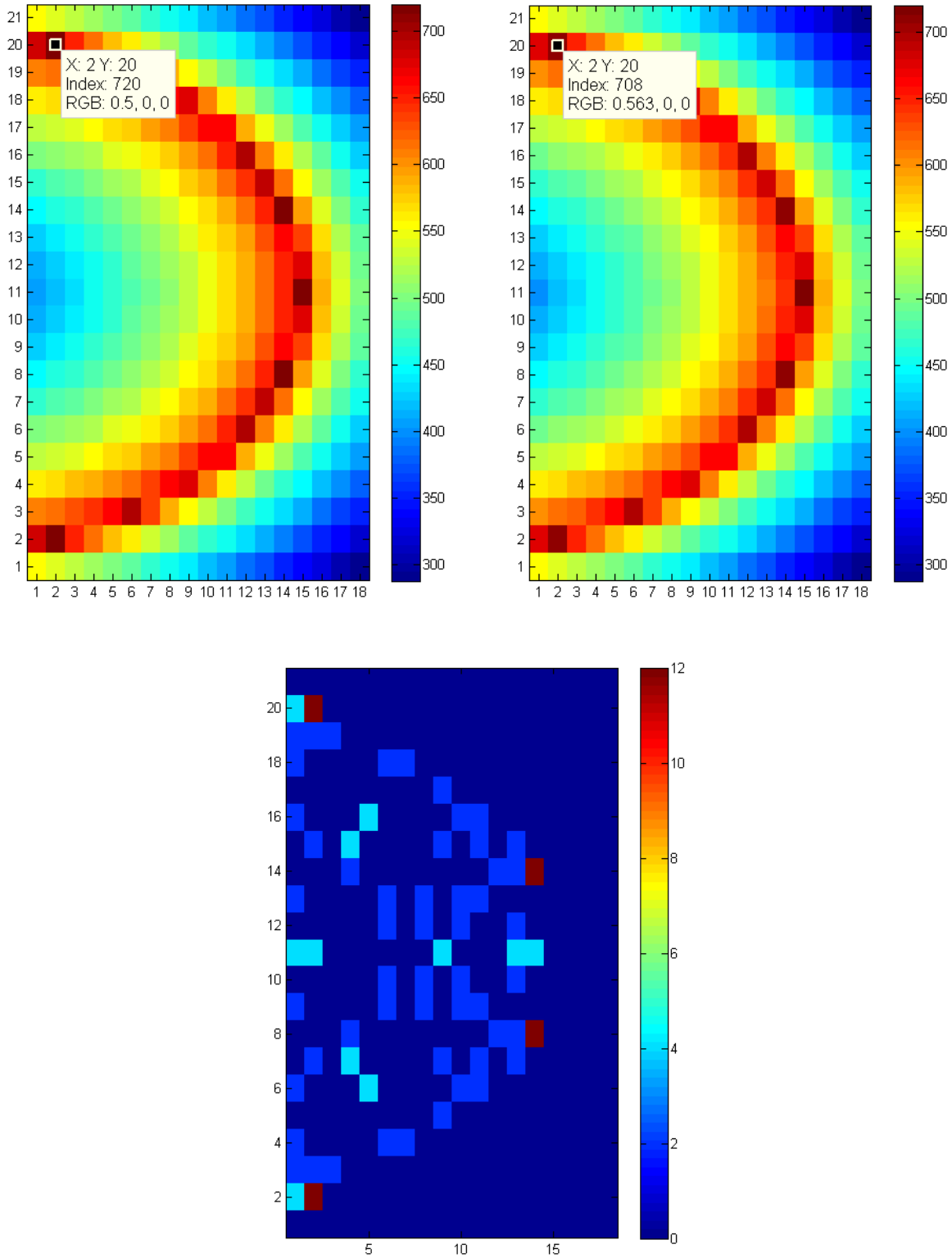

Fig. 3-3: Diferencia en el nº de sol. usando *umbral_error* = 0.01 y *umbral_error* = 0.001

En la Fig. 3-4, se muestra el mapa de accesibilidad del panel cuando se aplican las restricciones articulares. Como cabía esperar el número de soluciones de cada C/B disminuye, y debido a las restricciones que impiden que el brazo se pliegue sobre sí mismo, ahora las zonas menos accesibles son los cuadrantes superior e inferior izquierdo.

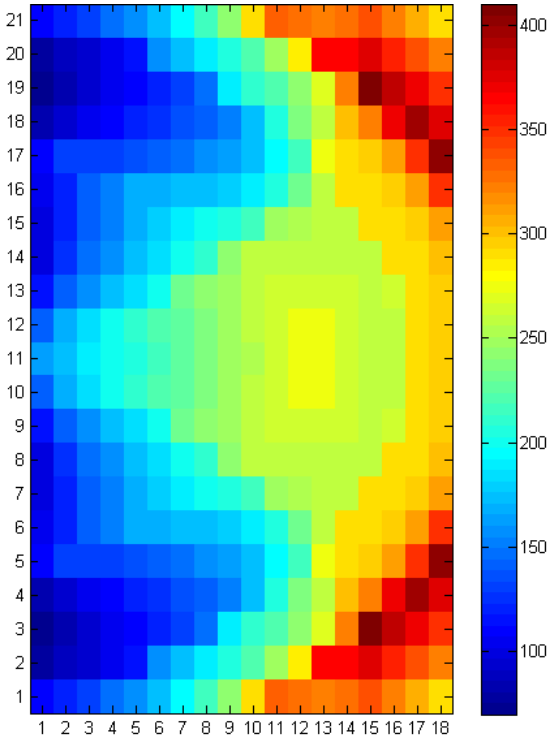

Fig. 3-4: Mapa accesibilidad (*resol_ang* = 1º; *umbral_error* = 0.001 m; con restr. art.)

En el caso sin restricciones, el C/B que más soluciones tiene es el (15, 11); y los que menos, los de las esquinas (18, 1) y (18, 21). Por el contrario, en el caso con restricciones, lo que más soluciones tienen son el (15, 3) y el (15, 19); y los que menos, el (1, 3) y el (1, 19). Estos puntos se representan en la Fig. 3-5.

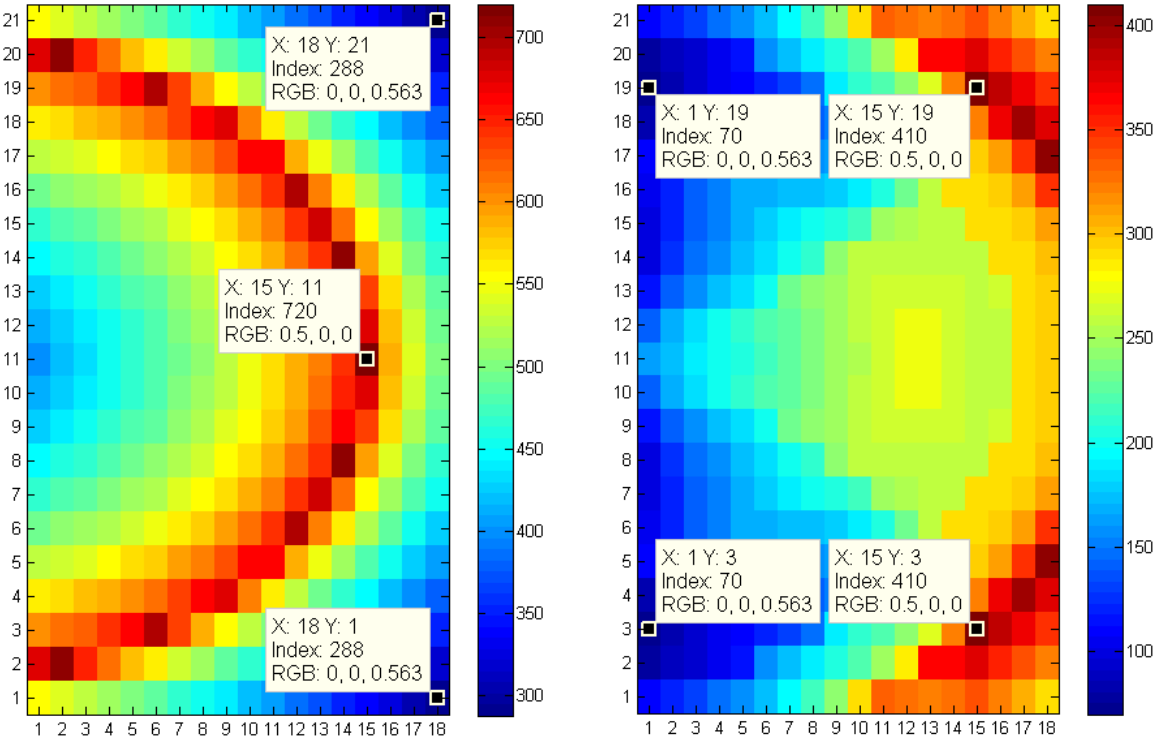

Fig. 3-5: Mapas de accesibilidad sin restricciones y con restricciones

3.2 Criterios de optimización

Aunque para para alcanzar cada C/B tengamos múltiples configuraciones posibles, en la práctica, necesitamos elegir solamente una. El proceso de selección de la configuración más adecuada se realiza minimizando el valor de una expresión en función las coordenadas articulares. Esta expresión dependerá del criterio de optimización seleccionado, es decir, de las características que queramos que cumpla la configuración elegida.

Este procedimiento se implementa en MATLAB de la siguiente forma:

1. El criterio de optimización se elige al llamar a la función "inv_kin_geometry" con el argumento de entrada *crit_opt*. Otros parámetros de entrada son el punto objetivo en el que se quiere posicionar el efector (*p_obj*), la configuración inicial del brazo (*q_ant*) y la indicación de si se quiere o no usar restricciones articulares (*limite*).

2. Se inicializa la variable *Jmin* con un valor elevado cualquiera, y se iguala la variable *qmin* a *q_ant*.

3. Cada vez que se calcula una nueva configuración, se llama a la función "func_opt", la cual devuelve la variable *J* con el valor de la función del criterio seleccionado.

4. Se compara la variable *J* con *Jmin.* Si $J < Jmin$, actualizamos *Jmin* con el valor de *J* y *qmin* con las coordenadas de la nueva configuración.

5. Seguimos este procedimiento hasta haber calculado todas las soluciones; de forma que, al final, *qmin* contiene las coordenadas de la configuración óptima.

A continuación, se describen los criterios que se pueden usar para elegir al llamar a la función "inv_kin_geometry".

3.2.1 Criterio I - Máxima aproximación

Como ya vimos en el apartado 2.3.2, para verificar que una solución del problema cinemático inverso era válida, se seguía el siguiente procedimiento:

$$(x, y) \rightarrow \text{Cinemática inversa} \rightarrow q = [q_1\ q_2\ q_3]$$

$$q = [q_1\ q_2\ q_3] \rightarrow \text{Cinemática directa} \rightarrow (\mathrm{x}', \mathrm{y}')$$

$$d = \sqrt{(x' - x)^2 + (y' - y)^2} \begin{cases} < umbral\_error \Rightarrow \text{configuración válida} \\ > umbral\_error \Rightarrow \text{configuración no válida} \end{cases} \quad [3\text{-}1]$$

El criterio de máxima aproximación, busca reducir al máximo la distancia entre (x, y) y $(\mathrm{x}', \mathrm{y}')$, por lo que la función de optimización queda definida como:

$$J = d = \sqrt{(x' - x)^2 + (y' - y)^2} \quad [3\text{-}2]$$

Para elegir este criterio, debemos usar: *crit_opt* = 1.

3.2.2 Criterio II – Configuraciones preferidas

A la hora de elegir una configuración sobre otra, podría interesar favorecer aquella que presente alguna característica particular que pueda considerarse una ventaja. Por ejemplo, dado que el motor de la base del robot es el que más peso soporta y, por lo tanto, el que normalmente debe generar un mayor par, podríamos intentar minimizar el momento generado por el peso del primer eslabón. Esto se consigue eligiendo aquellas

configuraciones en las que el primer eslabón se posiciona verticalmente o lo más verticalmente posible. Para ello, la función de optimización de este criterio se elige de forma que q_1 tienda a ser 0º o 180º, o lo que es lo mismo de forma que se minimice el valor absoluto del $sen(q_1)$.

$$J = |\, sen(q_1) \,| \qquad [3\text{-}3]$$

Este criterio se elige con *crit_opt* = 2.

3.2.3 Criterio III – Mínimo tiempo invertido

Otro criterio de optimización lógico podría ser aquel que busque que el cambio de una configuración a otra se realice en el menor tiempo posible. Este criterio se aplicará de forma diferente dependiendo de si los motores de las articulaciones se mueven de forma simultánea (en paralelo) o secuencial (en serie). Sin embargo, en ambos casos, habrá que tener en cuenta la configuración previa o de partida q_ant para calcular la siguiente más óptima.

Para simplificar el problema, se supondrá que todos los motores se mueven con la misma velocidad y de forma contante. Por lo que el tiempo invertido en mover una articulación será proporcional al incremento de su coordenada articular, Δq_i.

$$q\_ant = [q\_ant_1 \;\; q\_ant_2 \;\; q\_ant_3]$$
$$|\, \Delta q_i| = |\, q\_ant_i - q_i| \qquad [3\text{-}4]$$
$$i = 1, 2, 3$$

3.2.3.1 Motores de movimiento simultáneo

Si el movimiento de los motores se realiza de forma simultánea, será necesario minimizar el incremento máximo, es decir, el de la articulación con mayor recorrido.

$$J = máx(|\Delta q_1|, |\Delta q_2|, |\Delta q_3|) \qquad [3\text{-}5]$$

Para seleccionar este criterio, se elige *crit_opt* = 31.

3.2.3.2 Motores de movimiento secuencial

Cuando el movimiento de los motores se realiza de forma secuencial, el tiempo que invierte cada motor en moverse suma en el cómputo total de tiempo. Por lo tanto, la función de optimización se define como el sumatorio de los incrementos articulares.

$$J = \sum_{1=1}^{3} |\, \Delta q_i| \qquad [3\text{-}6]$$

Este criterio se elige con *crit_opt* = 32.

Las funciones [3-5] y [3-6] tienen la ventaja añadida de que tienden conservar la apariencia de la configuración previa, lo cual podría tomarse como otro criterio de optimización.

4 RESULTADOS Y SIMULACIONES

4.1 Visualización

En este capítulo, se realizarán diversas simulaciones aplicando los criterios de optimización vistos en el apartado 3.2. En cada simulación, el efector seguirá una trayectoria horizontal (fila de C/Bs) o vertical (columna de C/Bs). Estas trayectorias se implementan en los ficheros **t_horitontal.m** y **t_vertical.m**, que al ejecutarse generan tres figuras. En la primera figura, se representa la configuración inicial o de reposo. En la segunda, se muestra la superposición de todas las configuraciones, una por cada C/B de la trayectoria, marcándose en un color más intenso las configuraciones del primer y último C/B y en un color más suave las de los C/Bs intermedios. Por último, se representa la evolución de las coordenadas articulares durante el seguimiento de la trayectoria. También se puede generar una animación de la simulación con la función "robot_movie".

Los criterios de optimización I y II son independientes de la configuración inicial, por lo que no se tendrá en cuenta en sus simulaciones. El efecto de aplicar o no las restricciones articulares vistas en al final del apartado 2.3.2 se comprobará con las simulaciones del criterio I; en el resto de los casos, se usarán siempre restricciones articulares por ser necesarias en la práctica.

Para definir las trayectorias numerar las filas y las columnas de C/Bs. Como el origen de coordenadas se sitúa en el C/B de la esquina inferior izquierda, resulta lógico numerar las filas de abajo a arriba y las columnas de izquierda a derecha.

4.1.1 Criterio I - Máxima aproximación

4.1.1.1 Simulación 1

Condiciones: trayectoria horizontal, fila: 18, columna inicial: 17, columna final: 4, sin restricciones articulares.

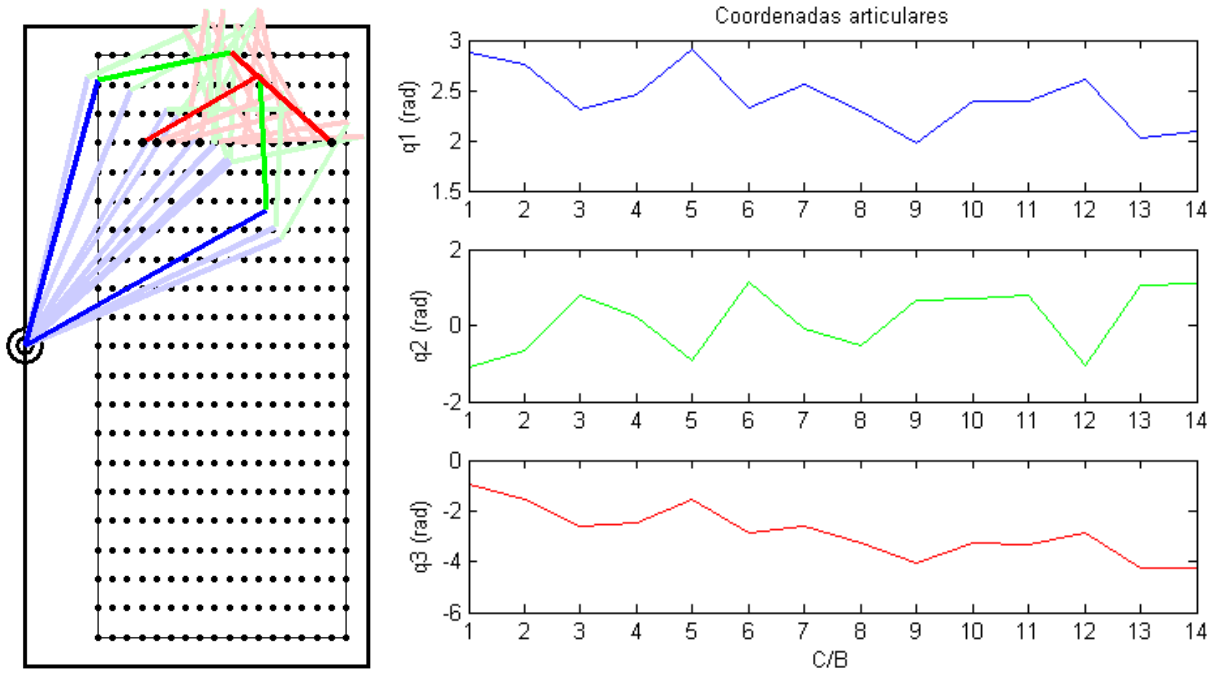

Fig. 4-1: Simulación 1 (Criterio I – Máxima aproximación)

4.1.1.2 Simulación 2

Condiciones: tray. horizontal, fila: 18, columna inicial: 17, columna final: 4, con restricciones articulares.

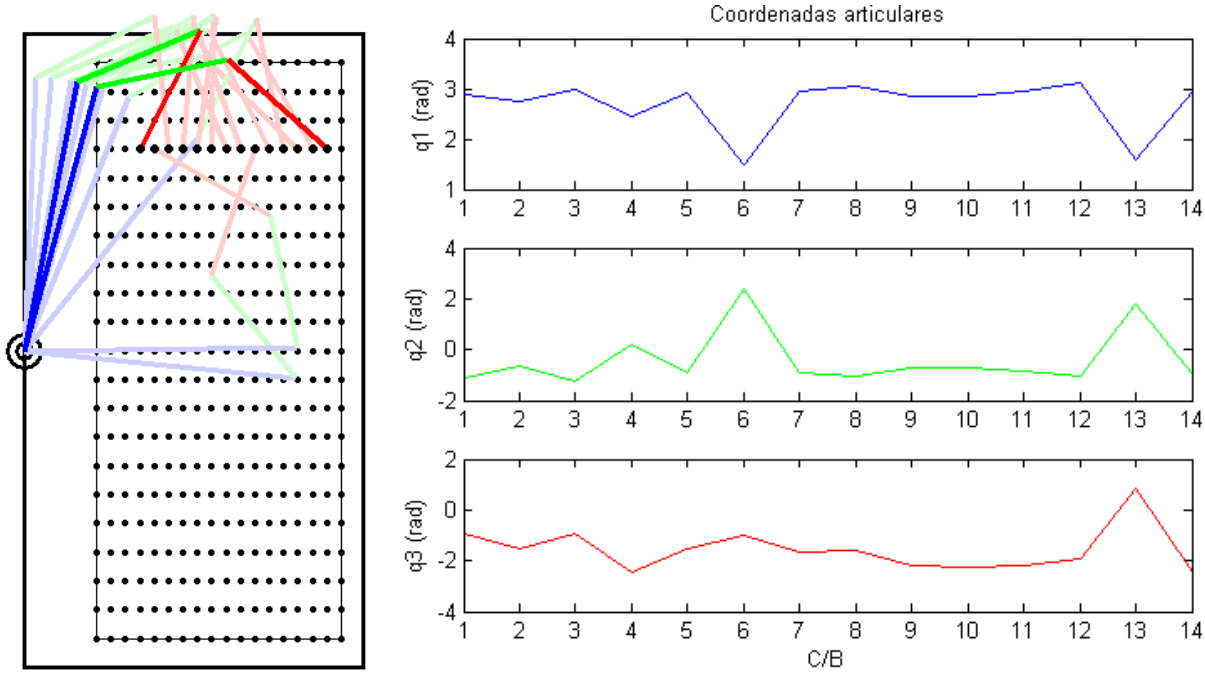

Fig. 4-2: Simulación 2 (Criterio I – Máxima aproximación)

4.1.1.3 Simulación 3

Condiciones: trayectoria vertical, columna: 8, fila inicial: 18, fila final: 4, sin restricciones articulares.

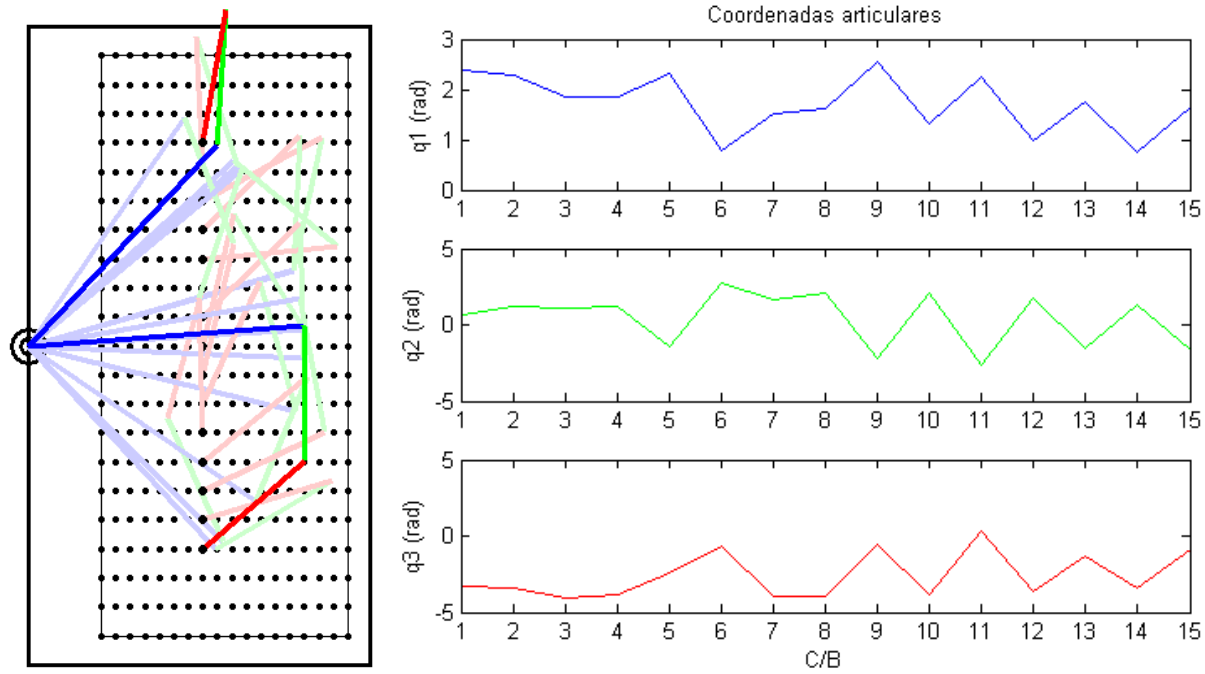

Fig. 4-3: Simulación 3 (Criterio I – Máxima aproximación)

4.1.1.4 Simulación 4

Condiciones: trayectoria vertical, columna: 8, fila inicial: 18, fila final: 4, con restricciones articulares.

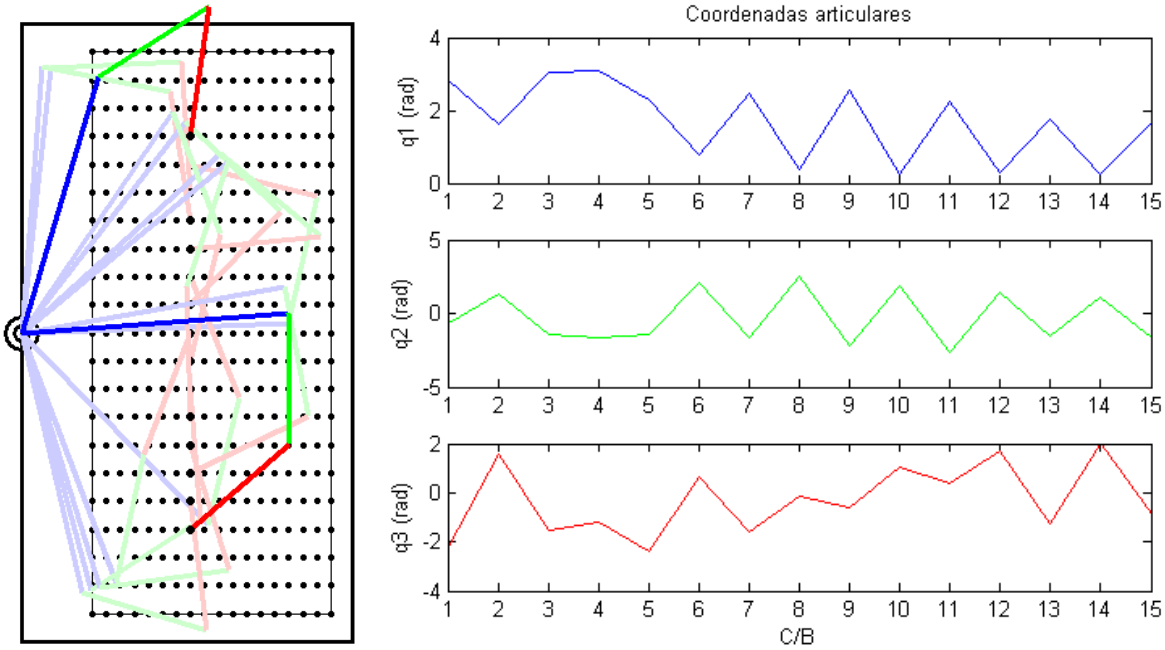

Fig. 4-4: Simulación 4 (Criterio I – Máxima aproximación)

4.1.2 Criterio II – Configuraciones preferidas

4.1.2.1 Simulación 5

Condiciones: tray. horizontal, fila: 18, columna inicial: 17, columna final: 4, con restricciones articulares.

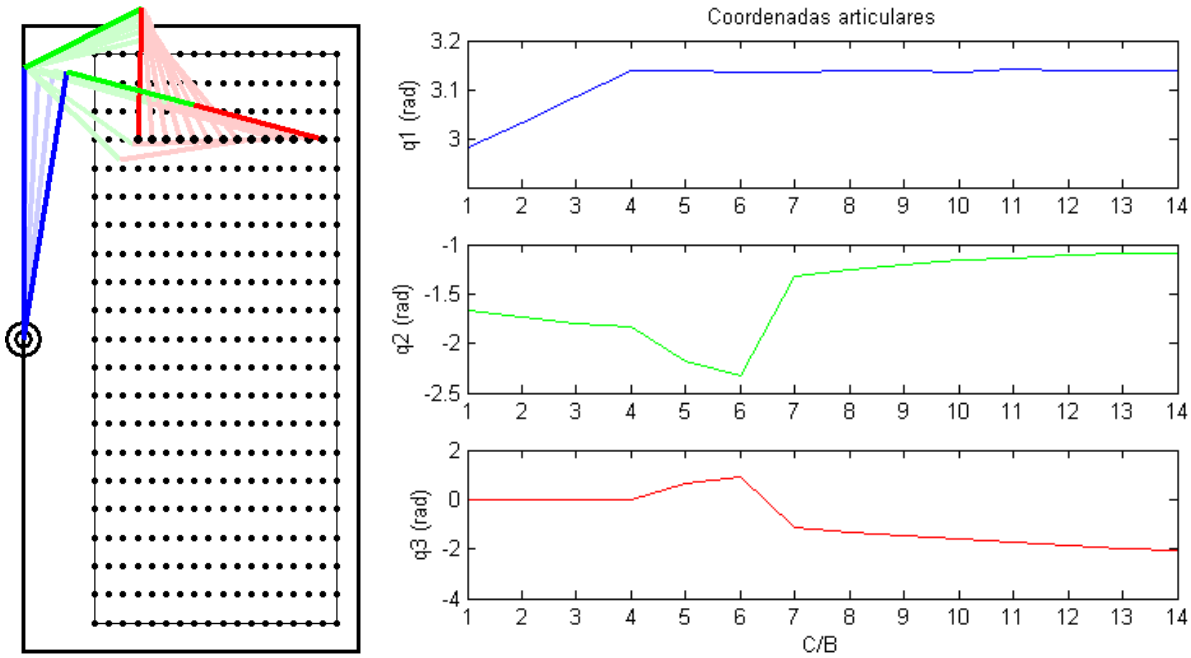

Fig. 4-5: Simulación 5 (Criterio II – Configuraciones preferidas)

4.1.2.2 Simulación 6

Condiciones: tray. horizontal, fila: 5, columna inicial: 5, columna final: 16, con restricciones articulares.

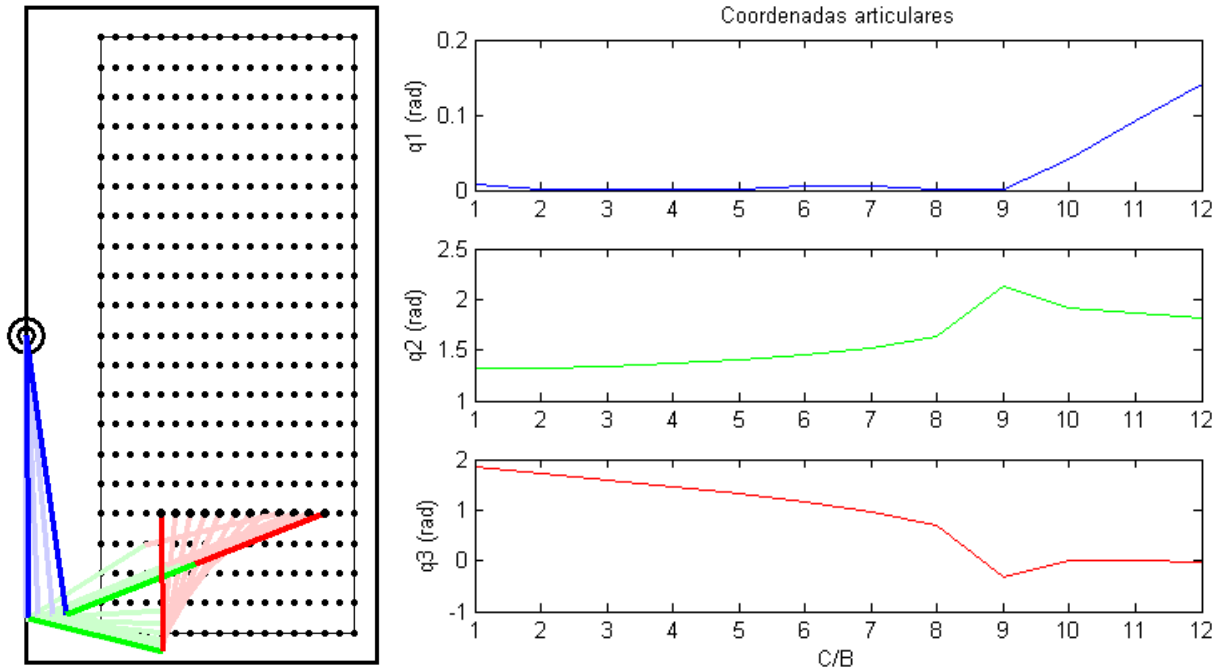

Fig. 4-6: Simulación 6 (Criterio II – Configuraciones preferidas)

4.1.2.3 Simulación 7

Condiciones: tray. horizontal, fila: 11, columna inicial: 3, columna final: 14, con restricciones articulares.

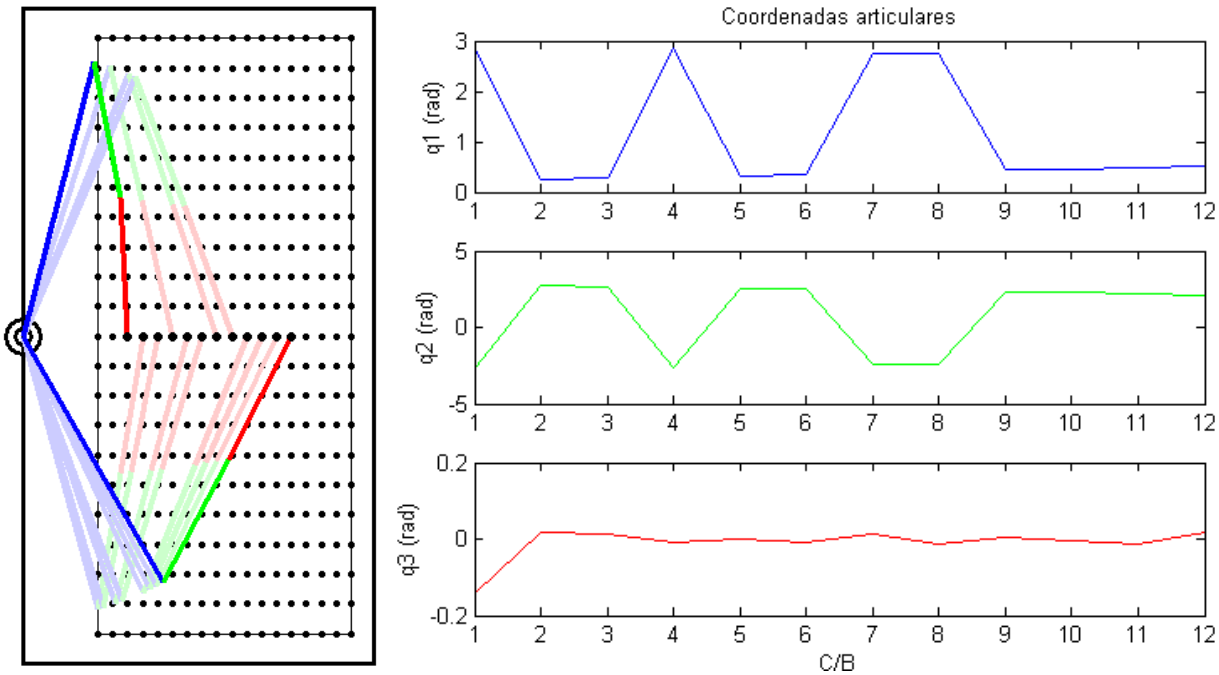

Fig. 4-7: Simulación 7 (Criterio II – Configuraciones preferidas)

4.1.2.4 Simulación 8

Condiciones: trayectoria vertical, columna: 8, fila inicial: 18, fila final: 4, con restricciones articulares.

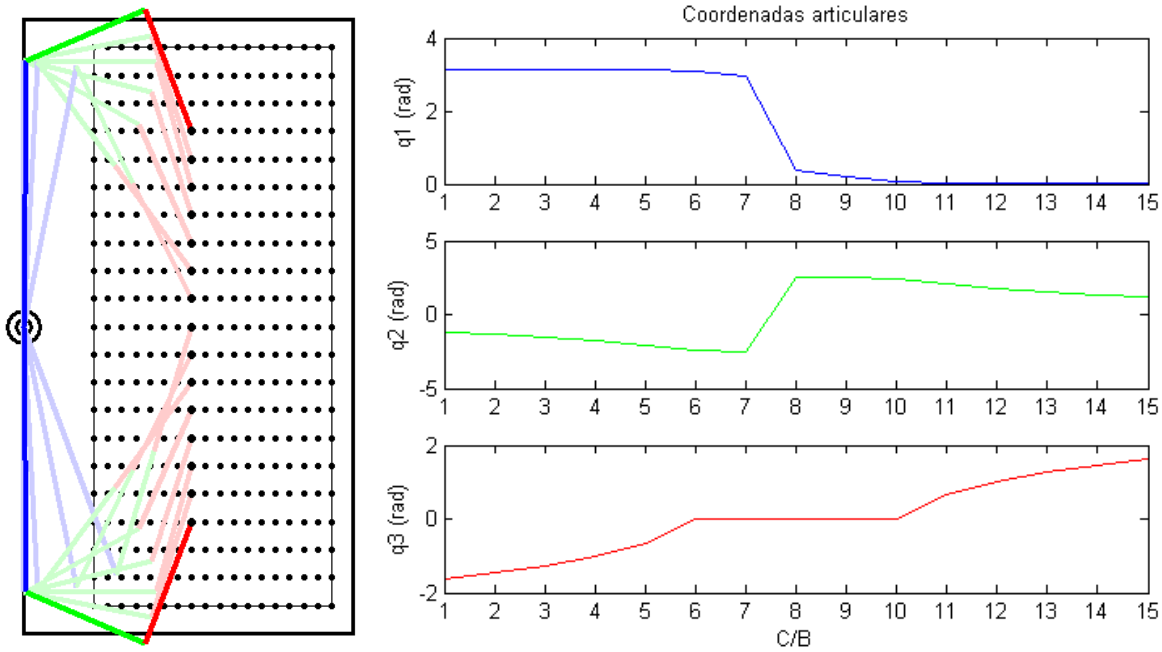

Fig. 4-8: Simulación 8 (Criterio II – Configuraciones preferidas)

4.1.3 Criterio III a. – Mínimo tiempo invertido (Motores en paralelo)

4.1.3.1 Simulación 9

Condiciones: trayectoria horizontal, fila: 18, columna inicial: 17, columna final: 4, con restricciones articulares, configuración de partida: $q\_ant = [\pi/2 \ \ \pi/2 \ \ \pi/2]$ (Fig. 4-9).

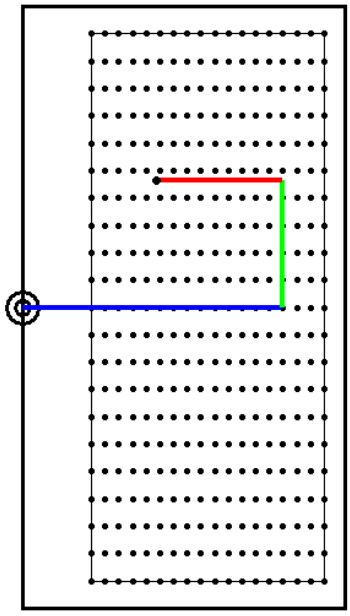

Fig. 4-9: Configuración de partida $q\_ant = [\pi/2 \ \ \pi/2 \ \ \pi/2]$

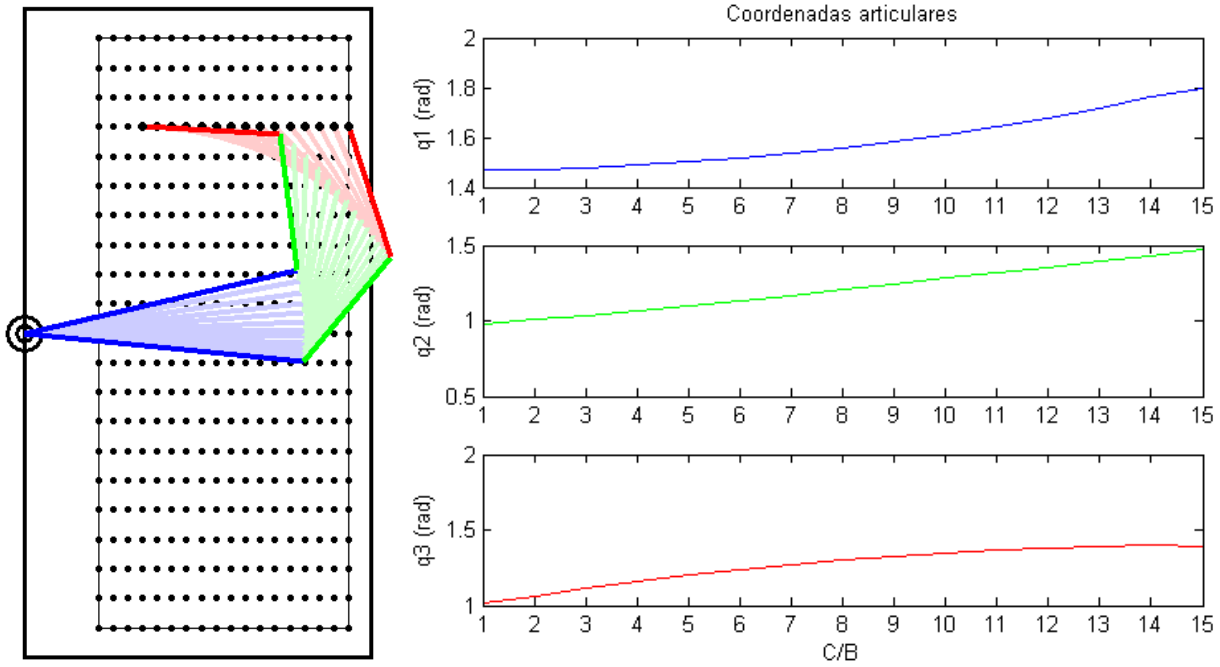

Fig. 4-10: Simulación 9 (Criterio III – Mín. tiempo invertido, motores en paralelo)

4.1.3.2 Simulación 10

Condiciones: trayectoria vertical, columna: 8, fila inicial: 18, fila final: 4, con restricciones articulares, configuración de partida: $q\_ant = [\pi/2 \ \ \pi/2 \ \ \pi/2]$ (Fig. 4-9).

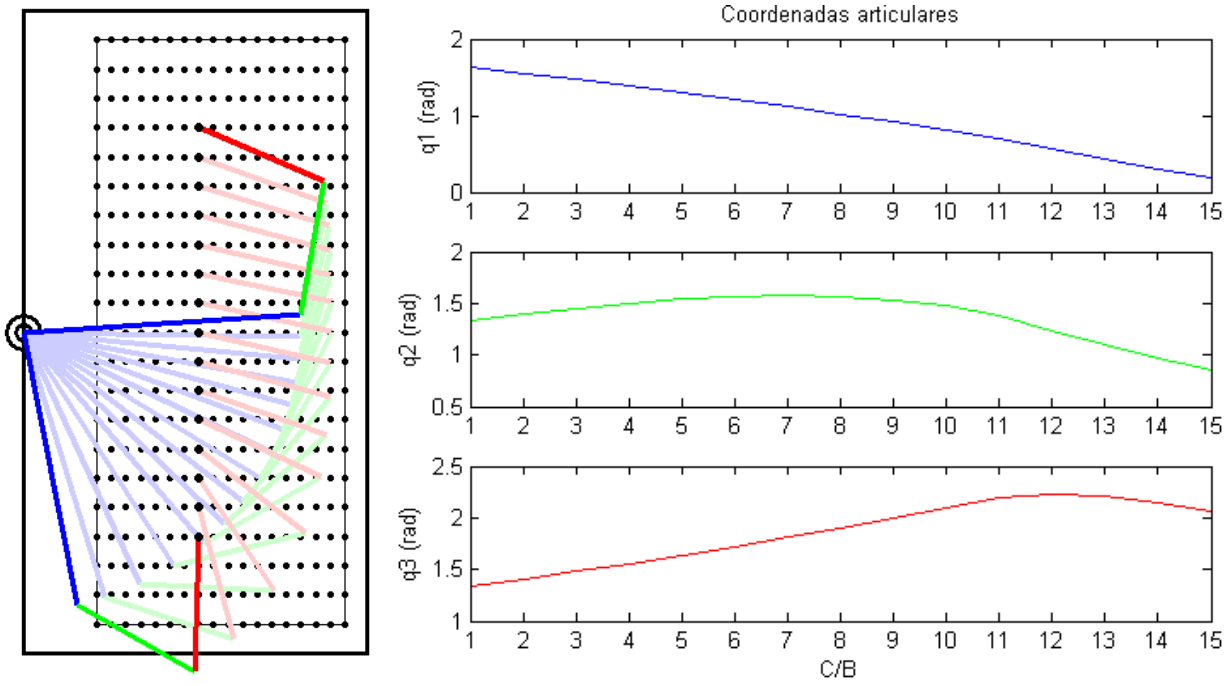

Fig. 4-11: Simulación 10 (Criterio III – Mín. tiempo invertido, motores en paralelo)

4.1.3.3 Simulación 11

Condiciones: trayectoria horizontal, fila: 18, columna inicial: 17, columna final: 4, con restricciones articulares, configuración de partida: $q\_ant = [\pi/2 \ -\pi/2 \ -\pi/2]$ (Fig. 4-12).

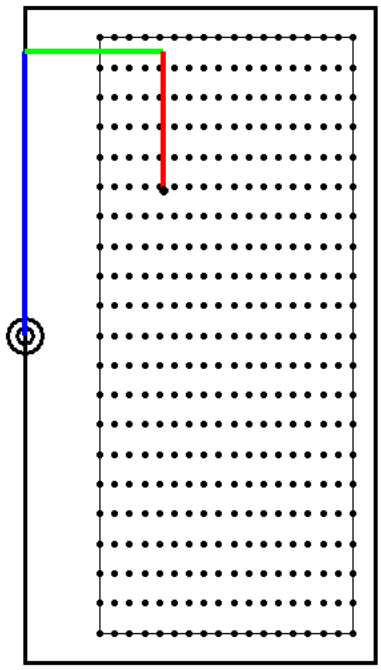

Fig. 4-12: Configuración de partida $q\_ant = [\pi/2 \ -\pi/2 \ -\pi/2]$

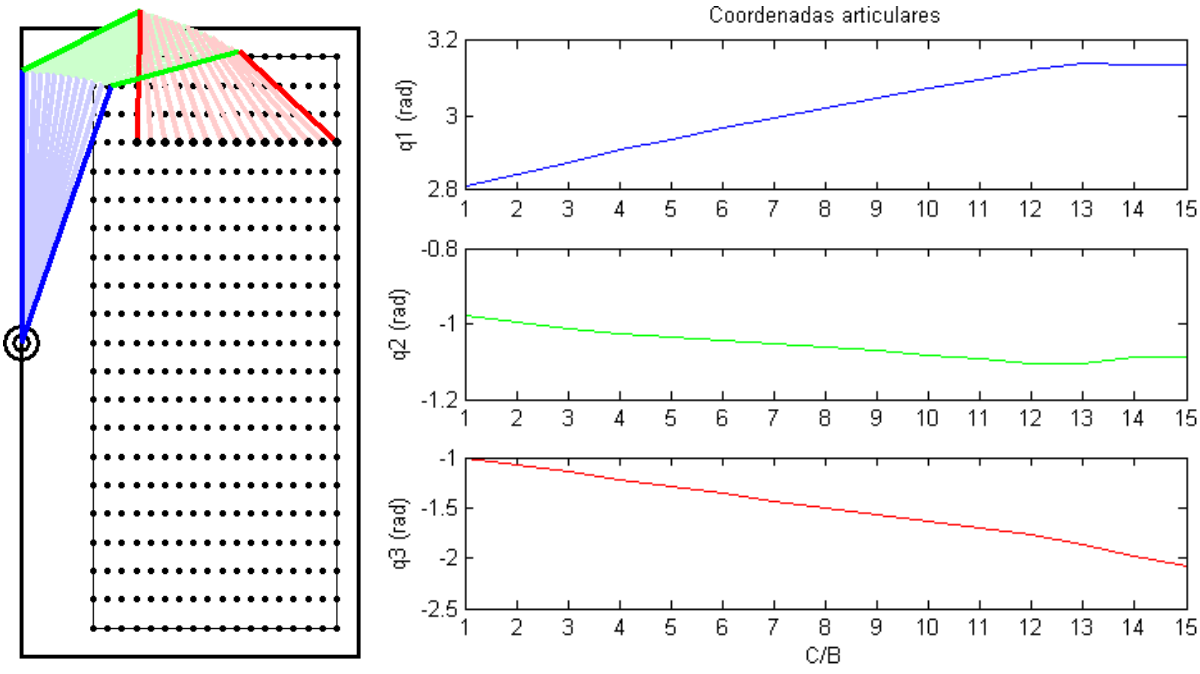

Fig. 4-13: Simulación 11 (Criterio III – Mín. tiempo invertido, motores en paralelo)

4.1.3.4 Simulación 12

Condiciones: trayectoria vertical, columna: 8, fila inicial: 18, fila final: 4, con restricciones articulares, configuración de partida: $q\_ant = [\pi/2 \ -\pi/2 \ -\pi/2]$ (Fig. 4-12).

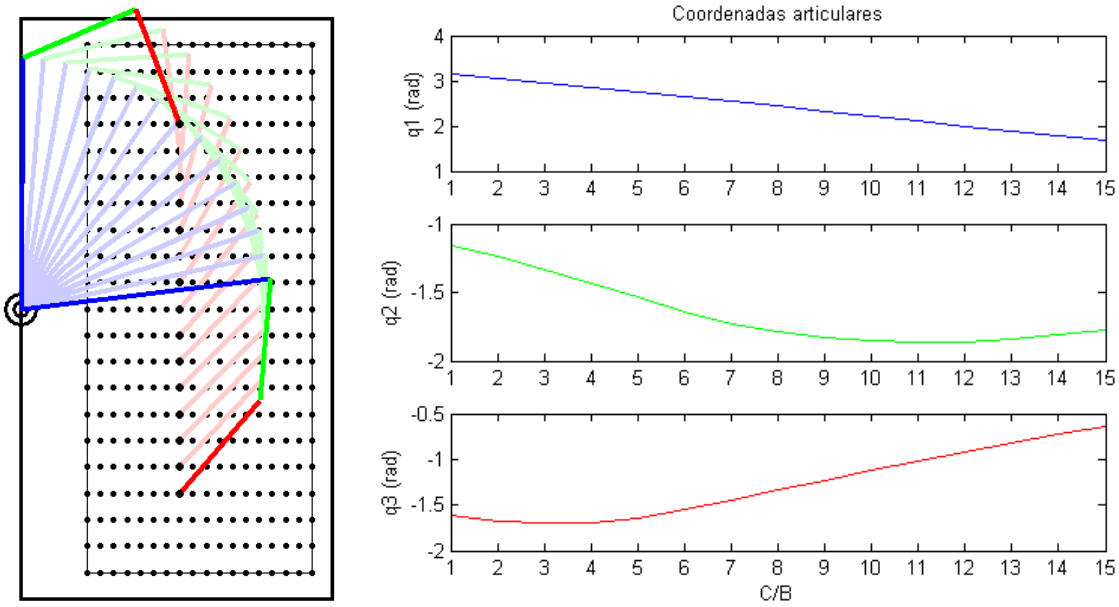

Fig. 4-14: Simulación 12 (Criterio III – Mín. tiempo invertido, motores en paralelo)

4.1.4 Criterio III b. – Mínimo tiempo invertido (Motores en serie)

4.1.4.1 Simulación 13

Condiciones: trayectoria horizontal, fila: 18, columna inicial: 17, columna final: 4, con restricciones articulares, configuración de partida: $q\_ant = [\pi/2 \ \pi/2 \ \pi/2]$ (Fig. 4-9).

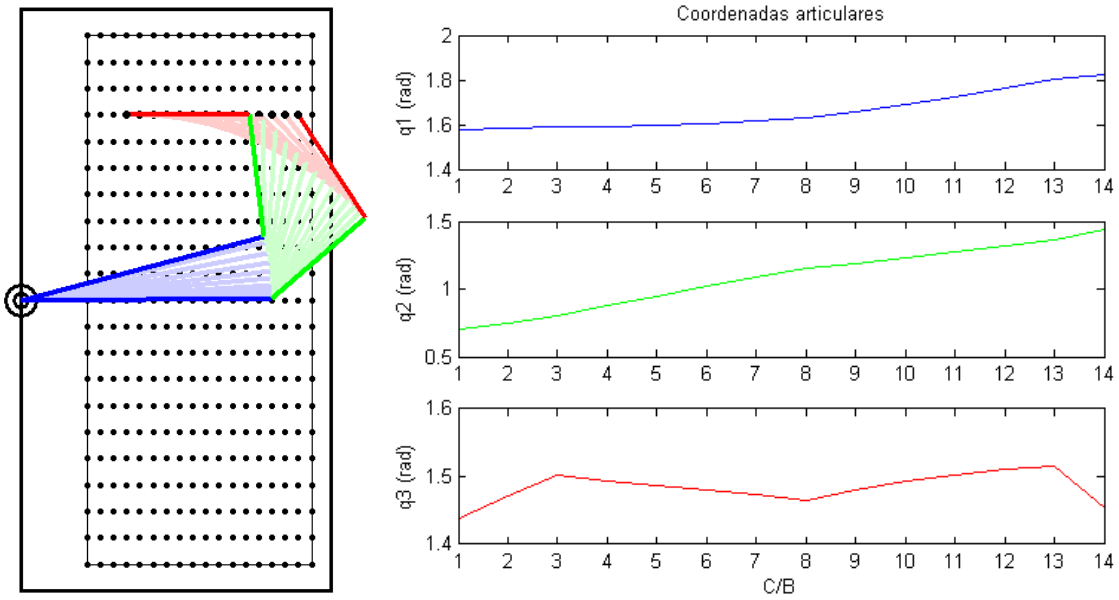

Fig. 4-15: Simulación 13 (Criterio III – Mín. tiempo invertido, motores en serie)

4.1.4.2 Simulación 14

Condiciones: trayectoria vertical, columna: 8, fila inicial: 18, fila final: 4, con restricciones articulares, configuración de partida: $q\_ant = [\pi/2 \ \ \pi/2 \ \ \pi/2]$ (Fig. 4-9).

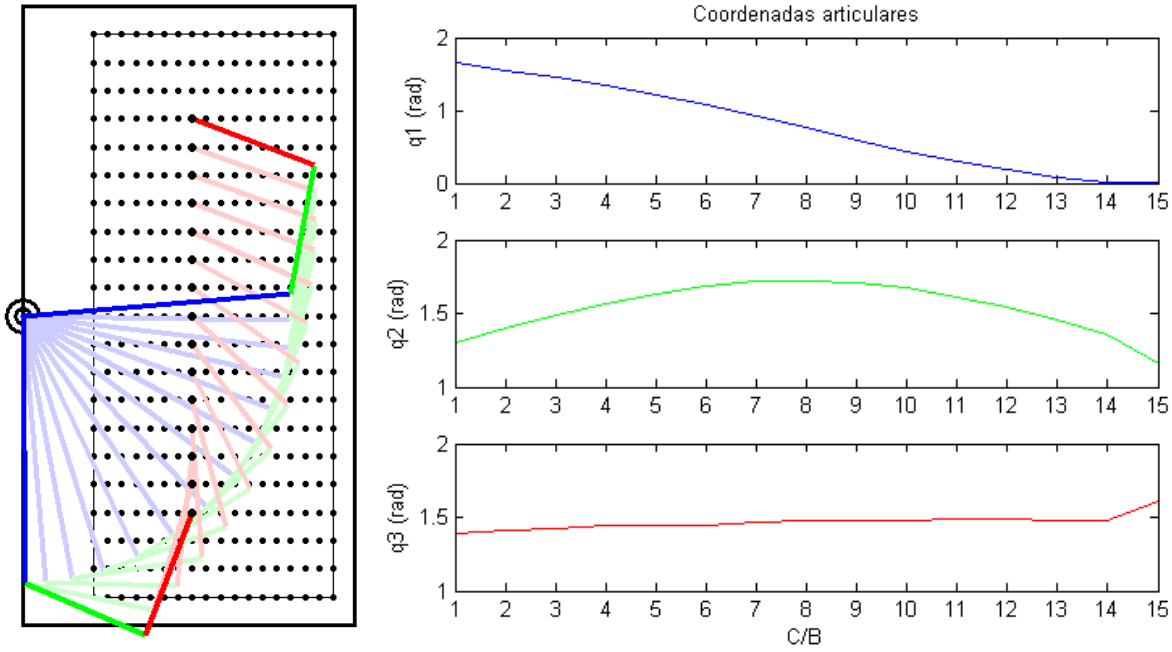

Fig. 4-16: Simulación 14 (Criterio III – Mín. tiempo invertido, motores en serie)

4.1.4.3 Simulación 15

Condiciones: trayectoria horizontal, fila: 18, columna inicial: 17, columna final: 4, con restricciones articulares, configuración de partida: $q\_ant = [\pi/2 \ \ -\pi/2 \ \ -\pi/2]$ (Fig. 4-12).

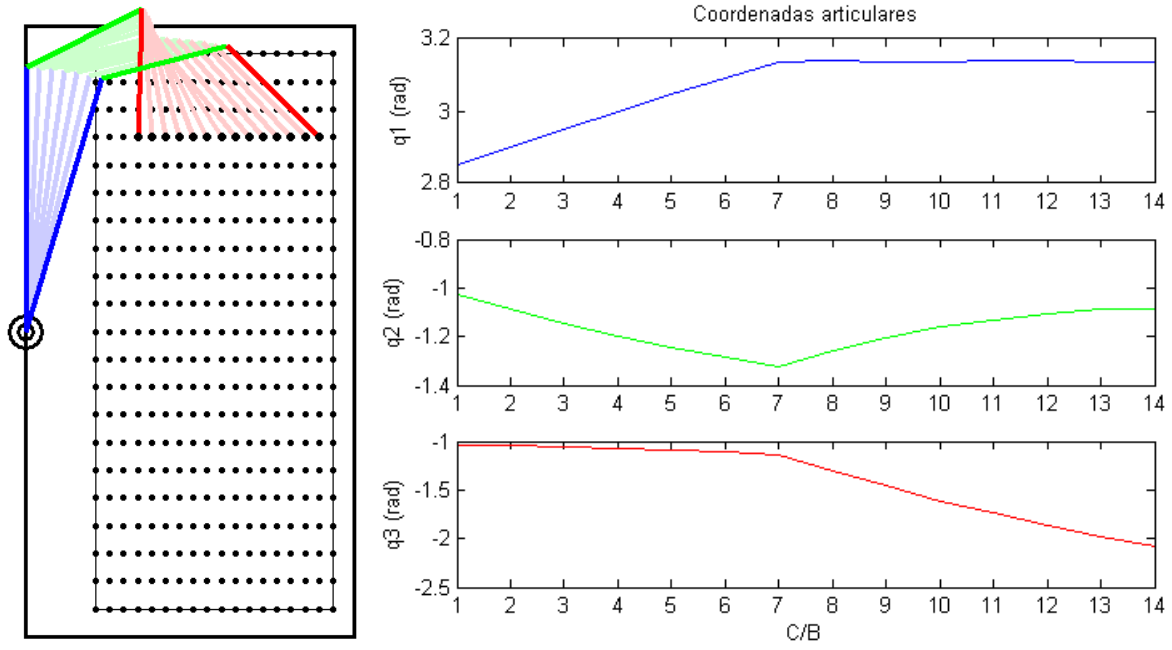

Fig. 4-17: Simulación 15 (Criterio III – Mín. tiempo invertido, motores en serie)

4.1.4.4 Simulación 16

Condiciones: trayectoria vertical, columna: 8, fila inicial: 18, fila final: 4, con restricciones articulares, configuración de partida: $q\_ant = [\pi/2 \ -\pi/2 \ -\pi/2]$ (Fig. 4-12).

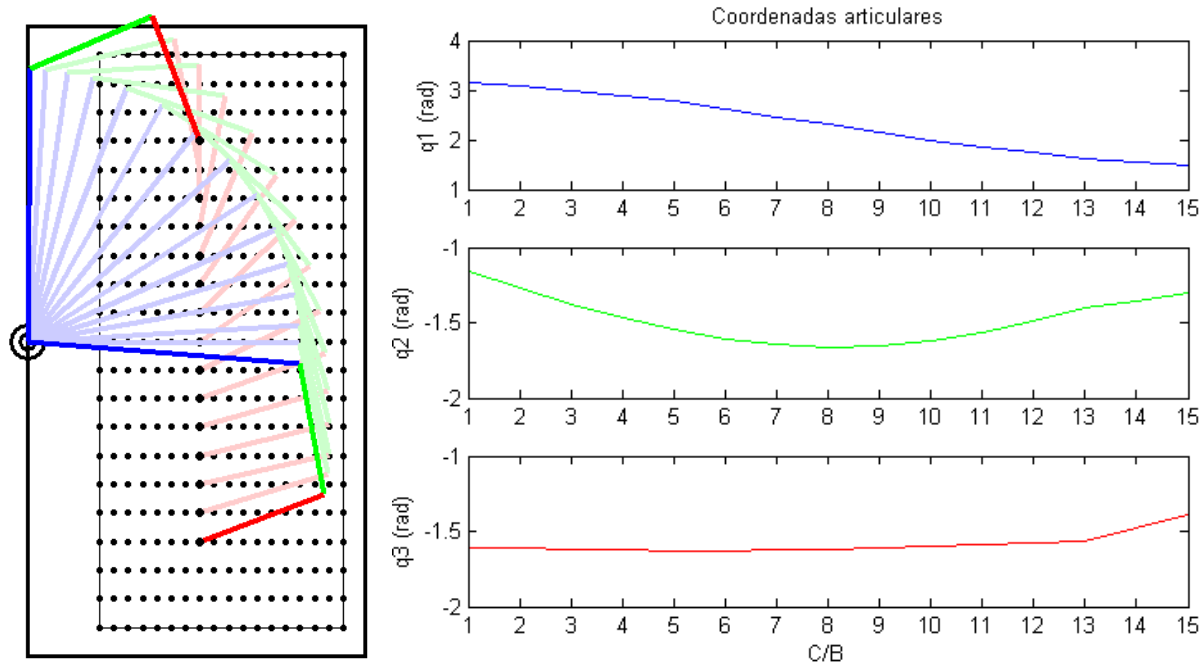

Fig. 4-18: Simulación 16 (Criterio III – Mín. tiempo invertido, motores en serie)

4.2 Conclusiones y consideraciones futuras

La aplicación del criterio I produce una respuesta articular muy irregular. Cada pico en las gráficas de las trayectorias articulares supone un cambio brusco en el sentido de giro de uno de los motores. A largo plazo, este comportamiento puede resultar perjudicial para los motores. Además, dado que el pulsador de cada C/B tiene un diámetro de 10.2 mm y estamos usando un *umbral_error* de 1 mm, podemos garantizar que estamos trabajando con una precisión suficiente sin necesidad de aplicar este criterio de optimización

El criterio II proporciona un resultado un poco mejor que el anterior. Sin embargo, no funciona bien cuando el manipulador trabaja por columnas o en las filas intermedias de C/Bs. En estos casos, se producen cambios bruscos entre $q_1 \sim 0°$ y $q_1 \sim 180°$, lo que al final resulta contraproducente para el motor de la base.

Debido a su tendencia a conservar la apariencia de la configuración previa, la aplicación del criterio III es la que produce una respuesta articular más suave y, por lo tanto, más sostenible a largo plazo por los motores. Sin embargo, este criterio no es completamente perfecto, ya que no garantiza que alguna de las articulaciones intermedias del robot no se salga del panel.

En futuros desarrollos, podría interesar sacrificar parte del carácter óptimo que aporta el criterio III, añadiendo las restricciones necesarias que garanticen que las uniones brazo-antebrazo y antebrazo-muñeca queden contenidas dentro de la superficie de trabajo. También, podría estudiarse la posibilidad de aplicar un criterio u otro en función de la secuencia de C/Bs con la que se trabaje. Un ejemplo de esta técnica podría consistir en aplicar el criterio II para las primeras y las últimas filas de C/Bs; y usar el Criterio 3 para las filas intermedias.

5 CÓDIGOS

5.1 cb_coord.m

```
%--------------------------------------------------------------------------
% GRÁFICO DEL PANEL DE C/Bs

% Coordenadas en metros y con origen en el C/B inferior izquierdo
%--------------------------------------------------------------------------

x_b = 0:0.02:0.02*17;
y_b = 0:0.04:0.04*20;

for j=1:21
    for i=1:18

        % Dibujar C/B (i,j):
        plot(x_b(i),y_b(j),'ko','MarkerFaceColor','k','MarkerSize',2)

%         % Descomentar para poner C/Bs en gris:
%         plot(x_b(i),y_b(j),'ko','MarkerEdgeColor',[0.5,0.5,0.5],...
%             'MarkerFaceColor',[0.5,0.5,0.5],'MarkerSize',2)

        hold on

    end
end

% Perímetro de los C/Bs:
rectangle('Position',[0 0 0.34 0.8],'LineWidth',0.5)

% Perímetro del panel:
rectangle('Position',[-0.1 -0.04 0.47 0.88],'LineWidth',2.5)

% Base del robot:
plot(-0.1,0.4,'ko','MarkerSize',20,'LineWidth',2.5)
plot(-0.1,0.4,'ko','MarkerSize',10,'LineWidth',2.5)

% Ajustar ejes:
ylim([-0.1 0.9])
axis equal
```

5.2 inv_kin_geometry.m

```
%--------------------------------------------------------------------------
% CINEMÁTICA INVERSA - MÉTODO GEOMÉTRICO

% p_obj => Punto que se desea alcanzar con el efector. El origen de
% coordenadas se sitúa en el 1er C/B (esquina inferior izquierda)

% q_ant => Configuración de partida en rad

% limite = 0 => NO hay restricciones articulares
% limite = 1 => Hay restricciones articulares

% Criterios de optimización:
% crit_opt = 1  => Máxima aproximación
% crit_opt = 2  => Configuraciones preferidas (q1 aprox. 0° o 180°)
% crit_opt = 31 => Mínimo tiempo invertido: movimiento simultáneo
% crit_opt = 32 => Mínimo tiempo invertido: movimiento secuencial

% Ejemplo:
% q_ant = [pi/2 pi/2 pi/2];
% [nsol, q] = inv_kin_geometry([0.2 0.2],q_ant,0,0)
%--------------------------------------------------------------------------

function [num_soluciones,qr] = inv_kin_geometry(p_obj,q_ant,limite,crit_opt)

% Coordenadas de la base del robot respecto al punto (0,0) (breaker esquina
% inf. izq.):

yb = 0.4; % 0.381;
xb = -0.1; %-0.10278;

% Se extraen las coordenadas cartesianas del punto objetivo (p -> panel):
xp = p_obj(1);
yp = p_obj(2);

% Posición respecto al eje local del robot (la base):
x = xp - xb;
y = yp - yb;

% Longitud eslabones:
le1 = 0.380;
le2 = 0.186;
le3 = 0.186;

% Máximo error admisible en la posición del manipulador:
umbral_error = 0.001;
% Resolución angular en la orientación en el punto objetivo:
resol_ang = 1;
% Valor mínimo de la función de optimización (inicialmente se le da un
% valor grande cualquiera):
J_min = 1000;
% Configuración óptima (inicialmente se iguala con la configuración de
% partida):
qmin = q_ant;
```

```
% Distancia cartesiana con el punto objetivo en la configuración óptima
% (inicialmente se le da un valor grande cualquiera):
dmin_objetivo = 1000;
% Contador del número de soluciones encontradas:
num_soluciones = 0;

if (check_pos(x,y)) % Si la posición(x,y)está dentro del panel

    % Calculamos para distintas orientaciones finales con las que llegar al
    % punto final:
    % for or_final = (-180+resol_ang):resol_ang:180
    for or_final = (-180+resol_ang-(resol_ang*limite)):resol_ang:180
        cabeceo=or_final*pi/180;

        % Se calculan las coordenadas de la base del eslabón 3 (x_b3, y_b3)
        % en función del punto objetivo y del ángulo de llegada a dicho
        % punto. Este ángulo se mide respecto a la horizontal y tomando el
        % vector con origen en la base del eslabón 3 y final en su extremo.

        inc_x = le3*cos(cabeceo);
        x_b3 = x - inc_x;
        inc_y = le3*sin(cabeceo);
        y_b3 = y - inc_y;

        % Fin cálculo coordenada base eslabón 3.

        % Si se puede alcanzar la posición (x_b3,y_b3), se prosigue con el
        % cálculo del ángulo alpha que deben formar entre sí los eslabones
        % 1 y 2 para alcanzar dicha posición. Deben obtenerse 2 soluciones.
        %
        % Para ello, se emplea el teorema del coseno:
        % Hipot^2 = le1^2 + le2^2 - 2*le1*le2*cos(alpha)
        %
        % En los cálculos se emplea la variable "cos_alpha" para el valor
        % cos(alpha).

        Hipot_cuad = (x_b3)^2 + (y_b3)^2;

        if (le1 + le2) >= sqrt(Hipot_cuad) % Si se puede alcanzar (x_b3,y_b3)

            cos_alpha = ((le1)^2 + (le2)^2 - Hipot_cuad)/(2*le1*le2);
            if (cos_alpha > 1.0)
                cos_alpha = 1.0;
            end
            if (cos_alpha < -1.0)
                cos_alpha = -1.0;
            end

            alpha = abs(acos(cos_alpha));

            % Fin cálculo ángulo alpha

            % Se aplica de nuevo el teorema del coseno para obtener el
            % ángulo beta, que forman el eslabón 1 con la línea que une
            % la base del robot a la base del eslabón 3 (Hipotenusa).

            cos_beta = (Hipot_cuad + (le1)^2 - (le2)^2)/...
                (2*le1*sqrt(Hipot_cuad));
```

```
            beta = abs(acos(cos_beta));

            angulo_b3 = atan2(y_b3,x_b3);

            q1_a = pi/2 + angulo_b3 + beta;
            q2_a = -pi + alpha;
            q3_a = cabeceo - q1_a - q2_a + pi/2;

            q1_b = pi/2 + angulo_b3 - beta;
            q2_b = pi - alpha;
            q3_b = cabeceo - q1_b - q2_b + pi/2;

            % Se calcula la posición cartesiana obtenida por la configuración
            % angular "a" ("codo hacia arriba"), así como la distancia entre
            % dicha posición y el punto objetico:
            p_a = direct_kin([q1_a q2_a q3_a]);
            d_a = dist_cart(p_a,[x y]);
            q_a = [q1_a q2_a q3_a];

            % Análogamente, para la configuración angular "b" ("codo hacia
            % abajo"):
            p_b = direct_kin([q1_b q2_b q3_b]);
            d_b = dist_cart(p_b,[x y]);
            q_b = [q1_b q2_b q3_b];

            % Para las dos soluciones, se comprueba si los ángulos cumplen
            % las restricciones y si el error cartesiano está dentro de
            % unos límites (en este caso, umbral_error = 0.001m = 1mm). De
            % ser así, se determina si es mejor solución que las anteriores,
            % intentando minimizar  una función de optimización.

            if (d_a < umbral_error)
                if (check_angles(q_a,limite))
                    num_soluciones = num_soluciones + 1;
                    J = func_opt(q_ant,q_a,d_a,crit_opt);
                    if (J < J_min)
                        dmin_objetivo = d_a;
                        J_min = J;
                        qmin = q_a;
                    end
                end
            end % Fin if (check_angles(q_a))

            if (d_b < umbral_error)
                if (check_angles(q_b,limite))
                    num_soluciones = num_soluciones + 1;
                    J = func_opt(q_ant,q_b,d_b,crit_opt);
                    if (J < J_min)
                        dmin_objetivo = d_b;
                        J_min = J;
                        qmin = q_b;
                    end
                end
            end % Fin if (check_angles(q_b))

        end % Fin if (le1 + le2) >= sqrt(Hipot_cuad)

    end % Fin for
```

```
else disp('Fuera del panel')

end % Fin if (check_pos(x,y))

qr = qmin;

% Descomentar si queremos ver:
% dmin_objetivo

end % Fin inv_kin_geometry


% Comprobar que la posición(x,y) está dentro del perímetro de C/Bs:
function ok = check_pos(x,y)

ok = 1;

% Límites:
x_max = 0.44 + eps;
x_min = 0.1 - eps;
y_max = 0.4 + eps;
y_min = -0.4 - eps;

% Nota: no se usan los límites exactos porque fallaba debido a errores de
% representación de punto flotante.

if (x > x_max)
    ok = 0;
elseif (x < x_min)
    ok = 0;
elseif (y > y_max)
    ok = 0;
elseif (y < y_min)
    ok = 0;
end

end


% Cinemática Directa:
function p = direct_kin(q)

% Longitud eslabones:
le1 = 0.380;
le2 = 0.186;
le3 = 0.186;

q1 = q(1);
q2 = q(2);
q3 = q(3);

x = real(le1*sin(q1)+le2*sin(q1+q2)+le3*sin(q1+q2+q3));
y = real(-le1*cos(q1)-le2*cos(q1+q2)-le3*cos(q1+q2+q3));
p = [x y];

end
```

```
% Cálculo de distancia entre 2 puntos:
function d = dist_cart(p1,p2)

d = sqrt((p1(1)-p2(1))^2+(p1(2)-p2(2))^2);

end


% Comprobación de restricciones articulares:
function ok = check_angles(q,limite)

ok = 1;
q1=q(1);
q2=q(2);
q3=q(3);

if (limite)

    % Limitaciones de articulación 1 (brazo):
    Lim_q1 = 180;
    if q1 > (Lim_q1*pi/180)
        ok = 0;
    end
    if q1 < (0*pi/180)
        ok = 0;
    end

    % Limitaciones de articulación 2 (antebrazo):
    Lim_q2 = 25;  % El limite real según CATIA: 20.76
    if q2 > ((180-Lim_q2)*pi/180)
        ok = 0;
    end
    if q2 < ((-180+Lim_q2)*pi/180)
        ok = 0;
    end

    % Limitaciones de articulación 3 (muñeca):
    Lim_q3 = 35; % El limite real según CATIA: 29.77
    if q3 > ((180-Lim_q3)*pi/180)
        ok = 0;
    end
    if q3 < ((-180+Lim_q3)*pi/180)
    ok = 0;
    end

    % Limitaciones para evitar choque:
    if 2*q2+q3 >= 345*pi/180 % sería exactamente 360 pero dejamos margen
        ok = 0;
    end

    if 2*q2+q3 <= -345*pi/180 % sería exactamente 360 pero dejamos margen
        ok = 0;
    end

end % Fin if (limite)

end
```

```
function J = func_opt(qa,qi,dist,criterio)

if criterio == 1 % Máxima aproximación al pto objetivo

    J = dist;

elseif criterio == 2 % Configuraciones preferidas

    J=abs(sin(qi(1)));

elseif criterio == 31 % Motores simultaneos

    J1 = abs(qa(1)-qi(1));
    J2 = abs(qa(2)-qi(2));
    J3 = abs(qa(3)-qi(3));

    J = max([J1 J2 J3]);

elseif criterio == 32 % Motores en serie

    J1 = abs(qa(1)-qi(1));
    J2 = abs(qa(2)-qi(2));
    J3 = abs(qa(3)-qi(3));

    J = J1+J2+J3;

else disp('Criterio incorrecto')

end

end
```

5.3 n_sol.m

```
%--------------------------------------------------------------------------
% MAPA DE ACCESIBILIDAD (nº de soluciones para cada C/B)
%--------------------------------------------------------------------------

x_b = 0:0.02:0.02*17;
y_b = 0:0.04:0.04*20;
q_ant = [0 90*pi/180 -90*pi/180];
limite=0; % 0 => no hay límite de ángulos; 1 => hay límite de ángulos
criterio=1; % En este caso, se puede coger cualquiera de los criterios

acc = zeros(21,18);

disp('Calculando...')

% (i,j) son los índices del eje de coordenadas situado el breaker de la
% esquina inferior izquierda. (l,m) son las coordenadas de la matriz acc,
% que es un mapa del número de soluciones de cada posición.
```

```
for j=1:21
    l=22-j;
    for i=1:18
        [nsol] = inv_kin_geometry([x_b(i) y_b(j)],q_ant,limite,criterio);
        if nsol == 0
          disp('No hay solución para:')
          [x_b(i) y_b(j)]
        end
        m=i;
        acc(l,m)= nsol;
    end
end

figure
imagesc(acc)
colorbar

set(gca,'XTick', 1:18, 'YTick', 1:21)
axis xy

disp('Fin')
```

5.4 robot_pic.m

```
%--------------------------------------------------------------------------
% REPRESENTACIÓN GRÁFICA DEL ROBOT
%--------------------------------------------------------------------------

function robot_pic(q1,q2,q3) % q1, q2 y q3 en rad

l1=0.380; l2=0.186; l3=0.186; % Longitud eslabones
s=1; % Escala
d1=3; d2=3; d3=3; % Grosores eslabones
n=size(q1,2);

yb = -0.4;
xb = 0.1;

figure
axis equal
axis(s*[-0.15 0.44 -0.1 0.9])

cb_coord;

for j=1:1:n

    % Elección de colores:

    if j==1||j==n
        c1=[0 0 1]; c2=[0 1 0]; c3=[1 0 0]; % Colores eslabones
    else
        c1=[0.8 0.8 1]; c2=[0.8 1 0.8]; c3=[1 0.8 0.8];
    end
```

```
     % Eslabones:

    line([0 l1*sin(q1(j))]-xb,[0 -l1*cos(q1(j))]-yb,'Color',c1,...
    'LineWidth',d1);

    line([l1*sin(q1(j)) l1*sin(q1(j))+l2*sin(q1(j)+q2(j))]-xb,...
    [-l1*cos(q1(j)) -l1*cos(q1(j))-l2*cos(q1(j)+q2(j))]-yb,'Color',c2,...
    'LineWidth',d2);

    line([l1*sin(q1(j))+l2*sin(q1(j)+q2(j)) l1*sin(q1(j))+l2*sin(q1(j)+...
        q2(j))+l3*sin(q1(j)+q2(j)+q3(j))]-xb,[-l1*cos(q1(j))-...
        l2*cos(q1(j)+q2(j)) -l1*cos(q1(j))-l2*cos(q1(j)+q2(j))-...
        l3*cos(q1(j)+q2(j)+q3(j))]-yb,'Color',c3,'LineWidth',d3);

    % Efector:

    hold on
    plot(l1*sin(q1(j))+l2*sin(q1(j)+q2(j))+l3*sin(q1(j)+q2(j)+q3(j))-xb,...
    -l1*cos(q1(j))-l2*cos(q1(j)+q2(j))-l3*cos(q1(j)+q2(j)+q3(j))-yb,...
    'ko','MarkerFaceColor','k','MarkerSize',5)

end

j=1; %Para que la primera configuración quede encima

% Eslabones:

line([0 l1*sin(q1(j))]-xb,[0 -l1*cos(q1(j))]-yb,'Color',c1,'LineWidth',d1);

line([l1*sin(q1(j)) l1*sin(q1(j))+l2*sin(q1(j)+q2(j))]-xb,...
     [-l1*cos(q1(j)) -l1*cos(q1(j))-l2*cos(q1(j)+q2(j))]-yb,'Color',c2,...
     'LineWidth',d2);

line([l1*sin(q1(j))+l2*sin(q1(j)+q2(j)) l1*sin(q1(j))+l2*sin(q1(j)+q2(j))+...
    l3*sin(q1(j)+q2(j)+q3(j))]-xb,[-l1*cos(q1(j))-...
    l2*cos(q1(j)+q2(j)) -l1*cos(q1(j))-l2*cos(q1(j)+q2(j))-...
    l3*cos(q1(j)+q2(j)+q3(j))]-yb,'Color',c3,'LineWidth',d3);

% Efector:

hold on
plot(l1*sin(q1(j))+l2*sin(q1(j)+q2(j))+l3*sin(q1(j)+q2(j)+q3(j))-xb,...
    -l1*cos(q1(j))-l2*cos(q1(j)+q2(j))-l3*cos(q1(j)+q2(j)+q3(j))-yb,...
    'ko','MarkerFaceColor','k','MarkerSize',5)

end
```

5.5 q123_pic.m

```
%--------------------------------------------------------------------------
% REPRESENTACIÓN DE LAS COORDENADAS ARTICULARES q1, q2 y q3
%--------------------------------------------------------------------------

function q123_pic(q1,q2,q3,max_cb)

figure

subplot(3,1,1)
plot(q1)
xlim([1 inf])
title('Coordenadas articulares')
ylabel('q1 (rad)')
set(gca, 'XTick', (1:1:max_cb))

subplot(3,1,2)
plot(q2,'g')
xlim([1 inf])
ylabel('q2 (rad)')
set(gca, 'XTick', (1:1:max_cb))

subplot(3,1,3)
plot(q3,'r')
xlim([1 inf])
ylabel('q3 (rad)')
xlabel('C/B')
set(gca, 'XTick', (1:1:max_cb))

end
```

5.6 robot_movie.m

```
%--------------------------------------------------------------------------
% ANIMACIÓN DEL ROBOT
%--------------------------------------------------------------------------

function robot_movie(q1,q2,q3)

l1=0.380; l2=0.186; l3=0.186;
s=1; % Escala
d1=3; d2=3; d3=3; % Grosores eslabones
c1=[0 0 1]; c2=[0 1 0]; c3=[1 0 0]; % Colores eslabones
n=size(q1,2);

% Coordenadas de la base
yb = -0.4;
xb = 0.1;

% Preparación del vídeo:
writerObj = VideoWriter('out.avi'); % Nombre del vídeo
% Nº de frames por segundo (disminuir para ralentizar):
writerObj.FrameRate = 3;
open(writerObj);
```

```
for j=1:n

    pause(0.1);

    figure (j+2)
    axis equal
    %% Descomentar para fondo de C/Bs (tarda más):
    %cb_coordenadas;
    axis(s*[-0.15 0.44 -0.1 0.9])
    rectangle('Position',[0 0 0.34 0.8])
    grid

    line([0 l1*sin(q1(j))]-xb,[0 -l1*cos(q1(j))]-yb,'Color',c1,...
        'LineWidth',d1);

    line([l1*sin(q1(j)) l1*sin(q1(j))+l2*sin(q1(j)+q2(j))]-xb,...
     [-l1*cos(q1(j)) -l1*cos(q1(j))-l2*cos(q1(j)+q2(j))]-yb,'Color',c2,...
     'LineWidth',d2);

    line([l1*sin(q1(j))+l2*sin(q1(j)+q2(j)) l1*sin(q1(j))+l2*sin(q1(j)+...
        q2(j))+l3*sin(q1(j)+q2(j)+q3(j))]-xb,[-l1*cos(q1(j))-...
        l2*cos(q1(j)+q2(j)) -l1*cos(q1(j))-l2*cos(q1(j)+q2(j))-...
        l3*cos(q1(j)+q2(j)+q3(j))]-yb,'Color',c3,'LineWidth',d3);

    % Efector:
    hold on
    plot(l1*sin(q1(j))+l2*sin(q1(j)+q2(j))+l3*sin(q1(j)+q2(j)+q3(j))-xb,...
    -l1*cos(q1(j))-l2*cos(q1(j)+q2(j))-l3*cos(q1(j)+q2(j)+q3(j))-yb,...
    'ko','MarkerFaceColor','k','MarkerSize',5)

    frame = getframe(gcf);
    writeVideo(writerObj, frame);

    close (j+2)

end

close(writerObj); % Guarda vídeo

end

% No se puede llamar a la función cb_coord durante la grabación del vídeo,
% así que la creamos de forma local:

function cb_coordenadas
x_b = 0:0.02:0.02*17;
y_b = 0:0.04:0.04*20;

% figure

for j=1:21
    for i=1:18
        plot(x_b(i),y_b(j),'ko','MarkerFaceColor','k','MarkerSize',2)

        % C/Bs en gris:
        plot(x_b(i),y_b(j),'ko','MarkerEdgeColor',[0.5,0.5,0.5],...
            'MarkerFaceColor',[0.5,0.5,0.5],'MarkerSize',2)

        hold on
```

```
    end
end
% Perímetro de los C/Bs:
rectangle('Position',[0 0 0.34 0.8])
% Perímetro del panel:
rectangle('Position',[-0.1 -0.04 0.47 0.88],'LineWidth',2)
plot(-0.1,0.4,'ko','MarkerSize',20,'LineWidth',2)
plot(-0.1,0.4,'ko','MarkerSize',10,'LineWidth',2)

% si el origen fuera en el extremo superior izquierdo: axis ij. Pero es en
% el extremo inferior izquierdo.

ylim([-0.1 0.9])
axis equal
end
```

5.7 t_horizontal.m

```
%--------------------------------------------------------------------------
% TRAYECTORIA HORIZONTAL

% Descomentar función robot_movie para generar vídeo de la simulación
%--------------------------------------------------------------------------

close all

% Criterios y existencia de límites:
crit_opt = 32;
limite = 1;

% Trayectoria:
fila = 18;
col_ini = 17;
col_fin = 4;
dist_col = 0.02;
dist_fila = 0.04;

if  col_fin < col_ini % Movimiento de derecha a izquierda
    dist_col=-dist_col;
end

% Configuración inicial y representación gráfica:
q_ini = [pi/2 pi/2 pi/2];
% q_ini = [pi -pi/2 -pi/2];

robot_pic(q_ini(1),q_ini(2),q_ini(3))
q_ant = q_ini;

% Definición de las trayectorias articulares:
j=1;
q1=zeros(1,abs(col_fin-col_ini));
q2=zeros(1,abs(col_fin-col_ini));
q3=zeros(1,abs(col_fin-col_ini));
y = (fila-1)*abs(dist_fila);

x_inicial = (col_ini-1)*abs(dist_col);
```

```
x_final = (col_fin-1)*abs(dist_col);

for x = x_inicial:dist_col:x_final;
    [nsol,q_min] = inv_kin_geometry([x y],q_ant,limite,crit_opt);
    q_ant = q_min;
    q1(j) = q_ant(1);
    q2(j) = q_ant(2);
    q3(j) = q_ant(3);
    j = j+1;
end

% Superposición de las configuraciones del robot:
robot_pic(q1,q2,q3)

% % Vídeo (descomentar para generar .avi):
% robot_movie([q_ini(1) q1],[q_ini(2) q2],[q_ini(3) q3])

% Representación de q1, q2 y q3 para cada C/B:
max_cb = abs(col_ini-col_fin)+1;
q123_pic(q1, q2, q3, max_cb)
```

5.8 t_vertical.m

```
%--------------------------------------------------------------------------
% TRAYECTORIA VERTICAL

% Descomentar función robot_movie para generar vídeo de la simulación
%--------------------------------------------------------------------------

close all

% Criterios y existencia de límites:
crit_opt = 32;
limite = 1;

% Trayectoria:
columna = 8;
fila_ini = 18;
fila_fin = 4;
dist_fila = 0.04;
dist_col = 0.02;

if  fila_fin < fila_ini % Movimiento de arriba a abajo
    dist_fila = -dist_fila;
end

% Posición inicial y representación gráfica:
q_ini = [pi/2 pi/2 pi/2];
%q_ini = [pi -pi/2 -pi/2];

robot_pic(q_ini(1),q_ini(2),q_ini(3))
q_ant = q_ini;

% Definición de las trayectorias articulares:
j=1;
q1=zeros(1,abs(fila_fin-fila_ini));
q2=zeros(1,abs(fila_fin-fila_ini));
q3=zeros(1,abs(fila_fin-fila_ini));
```

```
x = (columna-1)*abs(dist_col);

y_inicial = (fila_ini-1)*abs(dist_fila);
y_final = (fila_fin-1)*abs(dist_fila);

for y = y_inicial:dist_fila:y_final;
    [nsol,q_min] = inv_kin_geometry([x y],q_ant,limite,crit_opt);
    q_ant = q_min;
    q1(j) = q_ant(1);
    q2(j) = q_ant(2);
    q3(j) = q_ant(3);
    j = j+1;
end

% Superposición de las configuraciones del robot:
robot_pic(q1,q2,q3)

% % Vídeo (descomentar para generar .avi):
% robot_movie([q_ini(1) q1],[q_ini(2) q2],[q_ini(3) q3])

% Representación de q1, q2 y q3 para cada C/B:
max_cb = abs(fila_ini-fila_fin)+1;
q123_pic(q1, q2, q3, max_cb)
```

REFERENCIAS

- Barrientos A., Peñín L. F., Balaguer C., Aracil R., "Fundamentos de robótica", McGraw-Hill, Madrid (2007), 2ª ed.

- Ollero Baturone A., "Robótica, manipuladores y robots móviles", Marcombo, Barcelona (2001)

- MathWorks - Makers of MATLAB and Simulink. https://es.mathworks.com/

- KR AGILUS sixx | KUKA AG - KUKA Robotics.
  https://www.kuka.com/en-in/products/robotics-systems/industrial-robots/kr-agilus

- Adept Python Linear Modules. http://www.adept.com/products/robots/scara/cobra-s600/general

- SCARA Robots – Epson. http://global.epson.com/products/robots/products/scara/

- Robot arms - Kinova Robotics.
  http://www.kinovarobotics.com/assistive-robotics/products/robot-arms/