

Comunicaciones del CIMAT

**Optimización de la Estructura-Control de un
Robot Paralelo Planar 3RRR Usando Algoritmos
Evolutivos, Matlab-Simulink y SimWise 4D**

M. Infante-Jacob, S.Ivvan Valdez, Eusebio Hernández,
E. Chávez-Conde y S. Botello-Aceves

Comunicación del CIMAT:I-17-02/20.07.2017
(CC/CIMAT)



CIMAT

Optimización de la Estructura-Control de un Robot
Paralelo Planar 3RRR usando Algoritmos
Evolutivos, Matlab-Simulink y SimWise 4D

M. Infante-Jacobo, S. Ivvan Valdez, Eusebio Hernández,
E. Chavez-Conde y S. Botello-Aceves

19 de enero de 2017

Agradecimientos

Se le agradece al Centro de Investigación en Matemáticas (CIMAT) y al Consejo Nacional de Ciencia y Tecnología, particularmente al Proyecto de Ciencia Básica SEP CONACYT CB-2011/169132: *Optimización de la estructura-control de robots paralelos basada en algoritmos poblacionales* por los apoyos otorgados para la realización de este trabajo.

Índice general

1. Manual SimWise 4D y Matlab-Simulink	3
1.1. CAD: Creación, e importación	3
1.1.1. Modelo de sólidos	3
1.1.2. Importar el archivo CAD con SimWise 4D	4
1.1.3. Envío y recepción de datos	9
1.1.4. Configuración de parámetros de simulación	10
1.2. Matlab, Simulink y SimWise 4D	11
1.2.1. Acoplar el modelo a Simulink	11
1.2.2. Ejecutar Simulink desde Matlab	13
2. Descripción cinemática del robot 3RRR	17
2.1. Cinemática Inversa	18
2.2. Cinemática diferencial	20
3. Optimización	23
3.1. Optimización del espacio regular de trabajo	23
3.1.1. Resultados obtenidos	24
3.2. Optimización de la destreza del robot para trayectorias definidas	25
3.2.1. Función objetivo	26
3.2.2. Resultados obtenidos	27
3.3. Optimización de parámetros de control dinámico	27
3.3.1. Función objetivo 1: Minimización simultanea del error y energía consumida	29
3.3.2. Función objetivo 2: Minimización del error en los eslabones actuados	34
3.3.3. Función objetivo 3: Minimización consumo de energía	38
4. Conclusiones	42

Capítulo 1

Manual SimWise 4D y Matlab-Simulink

Este Capítulo contiene las instrucciones para la configuración de forma adecuada un modelo virtual en el simulador de SimWise 4D, y poder recibir y adquirir datos a partir de este modelo desde Matlab.

1.1. CAD: Creación, e importación

En esta Sección se muestra una forma de importar un archivo CAD al SimWise 4D, el cual se ocupará para la realización de simulaciones dinámicas del mecanismo que se considera estudiar, en nuestro caso será del robot paralelo planar 3RRR.

1.1.1. Modelo de sólidos

Antes de empezar a simular con SimWise es muy importante hacer mención que el diseño del mecanismo jugará un papel especial al momento de la simulación dinámica. Esta importancia se verá reflejada al momento de establecer las configuraciones de movimiento y fijado definidas en SimWise 4D. Una vez hecha esta observación, empecemos a describir el procedimiento de acomodo del modelo de sólidos.

Un modelo para la manufactura puede llegar a ser muy complejo, pues lleva una gran cantidad de piezas y subensambles, los cuales consisten en elementos de máquina, elementos de sujeción y piezas del mismo mecanismo, es por esto que a veces es conveniente realizar un diseño simplificado paralelo a éste, que nos permita con mayor facilidad realizar el ensamble en SimWise 4D para su posterior simulación.

En nuestro caso, para propósitos demostrativos, se optó por un diseño simplificado, en el que muchos de los elementos diseñados para la manufactura fueron retirados y/o modificados, e.g. las secciones y ejes que conforman los eslabones pasivos se conside-

raron como una sola pieza, los eslabones tanto pasivos como actuados y la plataforma móvil se diseñaron con fines ilustrativos.

Se debe procurar que el ensamble del mecanismo se realice de forma tal, que el eje Z esté orientado en forma convencional, es decir, que esté de manera vertical, en contra del sentido de la gravedad (aunque la gravedad puede ser modificada a placer en SimWise 4D, por convención es preferible hacerlo de la forma indicada anteriormente).

Otro dato muy importante para el ensamble del archivo CAD antes de importarlo con SimWise 4D es el punto o posición cero del sistema, es decir, su posición de descanso o a partir de donde se empezará a realizar las mediciones. Es preferible ensamblar nuestro mecanismo en el software CAD de manera que sea mas fácil definir nuestras referencias de medición en SimWise 4D. En la Fig. 1.1 se muestra la forma en que se ensambló el robot considerado como ejemplo en este trabajo. Una vez que se ha hecho el ensamble del archivo CAD, generado con algún software de diseño asistido por computadora, podemos pasar al siguiente paso.

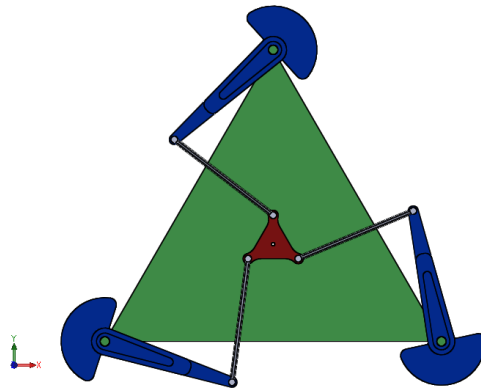


Figura 1.1: Posición cero del robot 3RRR

1.1.2. Importar el archivo CAD con SimWise 4D

SimWise 4D maneja un gran catálogo de compatibilidad para la importación de modelos CAD, entre ellos figuran Catia, Nx, AutoCAD, Solidworks, SolidEdge y otros. En este trabajo, se utilizó Solidworks para el diseño del modelo virtual.

Ya con el modelo ensamblado de manera correcta, abriremos el software SimWise 4D y daremos clic en File → Open para seleccionar nuestro archivo, tal como se muestra en la Fig. 1.2.

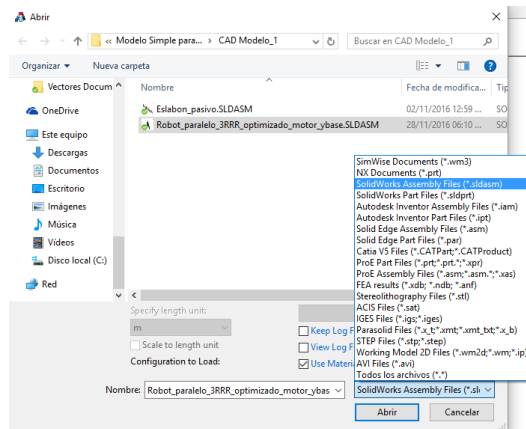


Figura 1.2: Posición cero del robot 3RRR

Después de abrir nuestro archivo, SimWise 4D se encargará de convertir el modelo a una versión que pueda interpretar. Realizado todo esto, procedemos a guardar nuestro archivo con extensión .wm3 en la dirección de nuestra elección (no importa que no esté en el mismo lugar que los archivos CAD originales). El archivo deberá quedar de la misma forma que en el software de diseño, como se muestra en la Fig. 1.3.

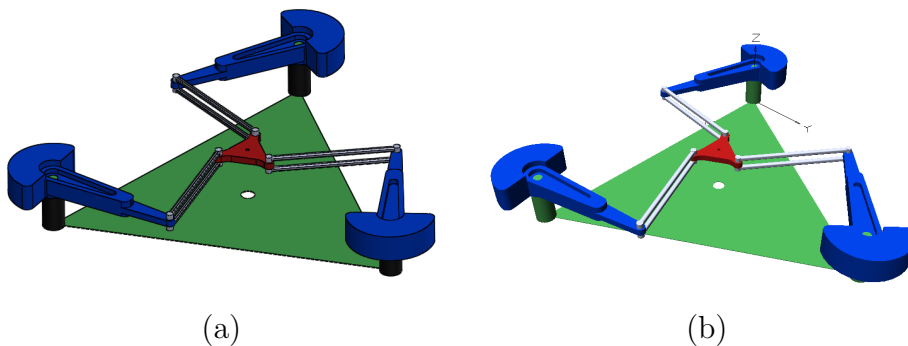


Figura 1.3: Ensamble del robot 3RRR. (a) En SolidWorks. (b) En SimWise 4D.

Configuración de parámetros

Después de haber importado nuestro archivo, todavía no es posible realizar simulación alguna, pues SimWise 4D no respeta las relaciones de posición que se le hayan dado con el software de diseño (a menos que se exporte directamente desde algunos softwares especiales, pero esto complica el adecuado posicionamiento de sistemas coordenados, es por esto que se recomienda realizar manualmente la configuración del modelo), por lo que se procede a darle coordenadas de posición a cada elemento del ensamble.

Abrimos nuestro modelo .wm3 guardado previamente. Se procederá a fijar cada elemento que no se deba mover en la estructura, en nuestro caso, será la base del manipulador y los actuadores, como se muestra en la Fig. 1.4.

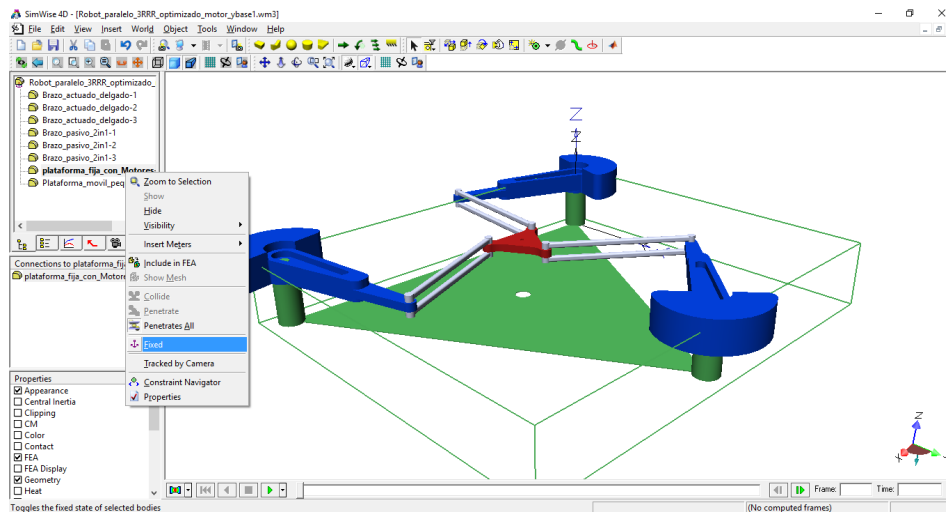


Figura 1.4: Fijando la Base del robot.

Después de esto, necesitamos establecer sistemas coordenados a cada unión del sistema, es decir, un eslabón binario tendría 2 ejes coordenados, un eslabón ternario tendría 3, uno cuaternario tendría 4 y así sucesivamente.

Para esto se da clic en: Insert → Coord y se selecciona la circunferencia del eje de rotación o la cara donde se desee que esté el eje coordenado, como se muestra en la Fig. 1.5 (a), y así para cada pieza del sistema.

Al final, cada par cinemático deberá constar de dos ejes coordenados ubicados en el mismo lugar como se muestra en la Fig. 1.5 (b), de forma que nos permita relacionarlos entre sí fácilmente.

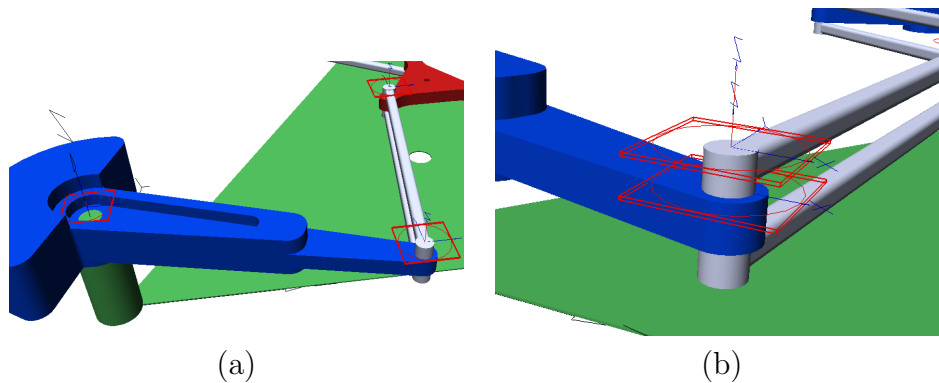


Figura 1.5: Estableciendo ejes coordenados. (a) Coordenadas por eslabón. (b) Coordenadas por par cinemático, una del brazo actuado y otra del brazo pasivo (en esta imagen, las coordenadas deberían estar en la misma posición pero se separaron con el fin de observar ambas).

Después de esto, se modifican los parámetros de cada eje coordenado respecto al eje principal, o respecto al eje a partir del cual toma mediciones dicho eje coordenado, para esto, se da doble clic en la coordenada deseada y en la sección Pos, se cambian los valores de posición y orientación según sea el caso, tal como se muestra en la Fig. 1.6.

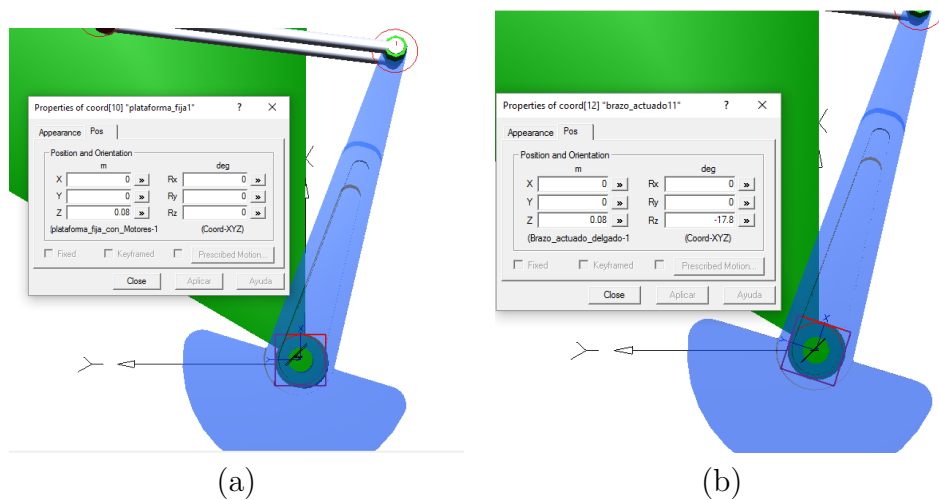


Figura 1.6: Orientando coordenadas. (a) Coordenada primaria y fija. (b) Coordenada secundaria, gira alrededor de la primaria

Restricciones de movimiento

Las restricciones de movimiento permiten que el mecanismo se quede anclado a ciertos puntos, pero que a su vez tenga libertad de movimiento. En general, existen dos tipos de restricciones, las restricciones pasivas como las uniones de revoluta, uniones esféricas o las uniones prismáticas; y las restricciones actuadas o activas, como son los motores, actuadores lineales, uniones por banda o por engranes.

Para generar estas restricciones basta con seleccionar un eje coordenado y darle clic derecho, a continuación, seleccionar Create Constraint, en esta ventana, seleccionar la otra coordenada con la que estará referenciada y el tipo de restricción que se requiere, como se muestra en la Fig. 1.7, es decir, para la unión entre el eje del motor y el eslabón actuado deberá ser una unión del tipo Revolute Motor, y para las uniones pasivas, bastará con seleccionar uniones del tipo Revolute Joint como se observa en la Fig. 1.8.

El resultado final de agregar todas las restricciones se muestra en la Fig. 1.9. Para verificar que el modelo quedó bien configurado, se pueden hacer ejecuciones en lazo abierto del mecanismo estableciendo una velocidad deseada de rotación en los actuadores y observar que el sistema se mueva de manera esperada, en caso de que el sistema se desensamble o se caiga, deberá revisar que todas las uniones estén debidamente restringidas.

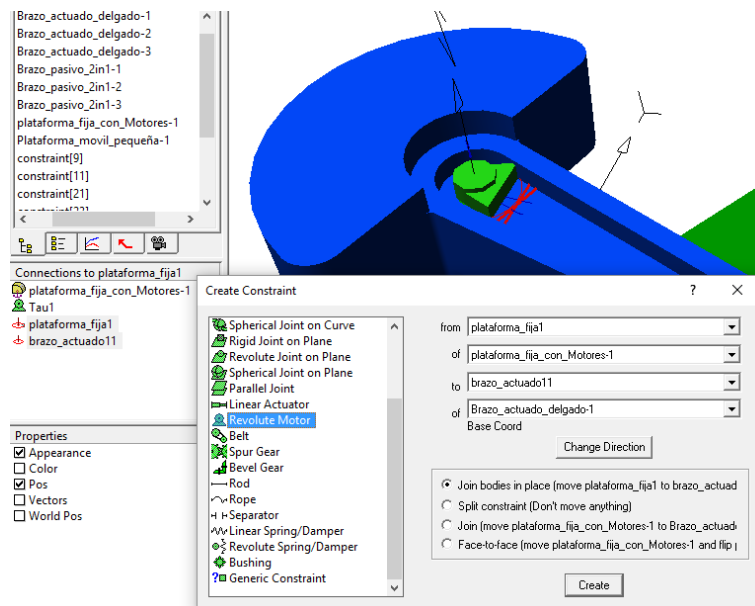


Figura 1.7: Restricción activa (Motor).

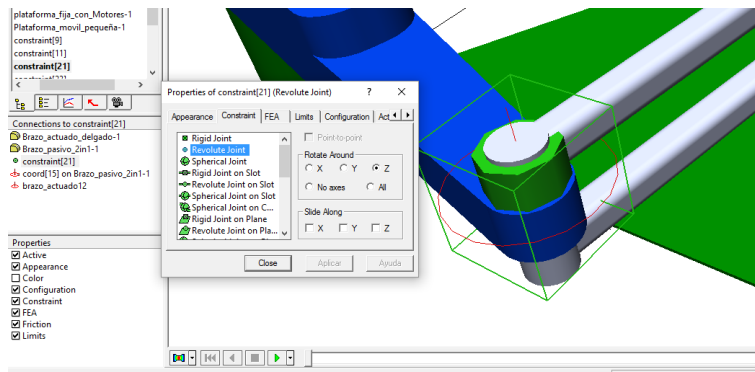


Figura 1.8: Restricción pasiva (Unión de revoluta)

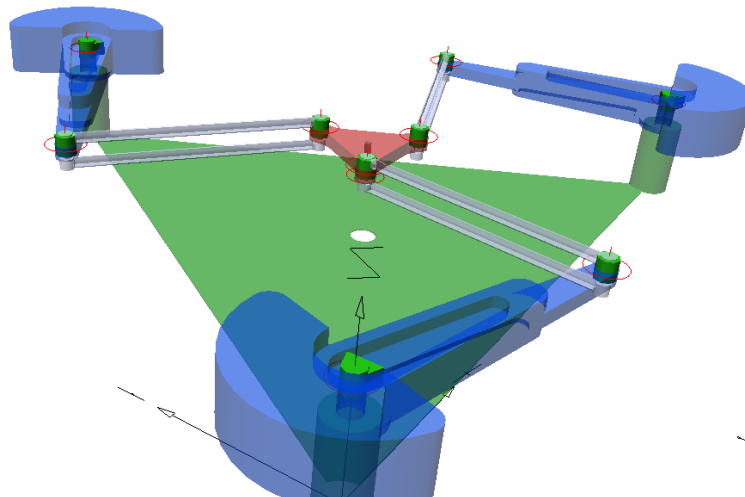


Figura 1.9: Mecanismo con todas las restricciones de movimiento.

1.1.3. Envío y recepción de datos

Con todo lo anterior el sistema ya se puede mover dentro del software SimWise 4D, aunque aún no es posible controlar su movimiento desde Matlab Simulink, por lo que se hace necesario establecer parámetros de sensado y de control dentro del modelo. Para definir las entradas de control se procede de la siguiente manera:

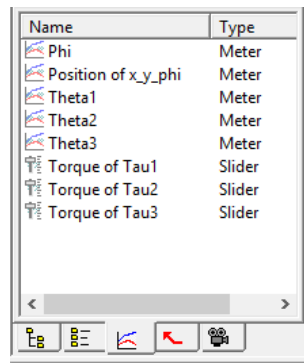
Clic en el actuador que se desee controlar, después de haberlo seleccionado, clic en la pestaña Insert → Control → Torque → Slider.

Esto se realiza para cada actuador que se desee controlar de manera externa a SimWise 4D.

Para definir las salidas de datos, se realiza lo siguiente:

Clic en la restricción o coordenada que se desee leer, ahora en la pestaña Insert →

Meter \rightarrow Orientation o Position según sea el caso. Esto se deberá realizar para cada sección del sistema del que se necesite o se desee saber información de la posición, orientación, velocidad o aceleración. Al final, deberemos tener un total de 3 entradas de control (τ_1, τ_2 y τ_3), 4 medidores de orientación ($\theta_1, \theta_2, \theta_3$ y ϕ), y un medidor de posición cartesiano (G_x y G_y), como se observa en la Fig. 1.10.



Name	Type
Phi	Meter
Position of x_y_phi	Meter
Theta1	Meter
Theta2	Meter
Theta3	Meter
Torque of Tau1	Slider
Torque of Tau2	Slider
Torque of Tau3	Slider

Figura 1.10: Variables de control y medición del sistema.

1.1.4. Configuración de parámetros de simulación

La adecuada configuración de parámetros es muy importante al momento de realizar las simulaciones, estos parámetros comprenden desde el tipo de unidades que usará el modelo, el método de integración, el paso de integración, tipo de material, etc. A continuación se muestran algunas configuraciones necesarias para poder llevar a cabo la ejecución del programa exitosamente. Para configurar las unidades en que se planea trabajar con el modelo se realiza lo siguiente: clic en SimWise Settings \rightarrow Display Settings \rightarrow Units.

Para configurar los parámetros de simulación, de igual forma nos situaremos en la ventana SimWise Settings \rightarrow Simulation Settings. En este punto es importante resaltar que el Animation Frame Rate deberá ser igual al paso de muestreo que se ocupará en Simulink, además, es preferible configurar el paso de integración como fijo para evitar problemas en la simulación. En la Fig. 1.11 se muestra la ventana de configuración.

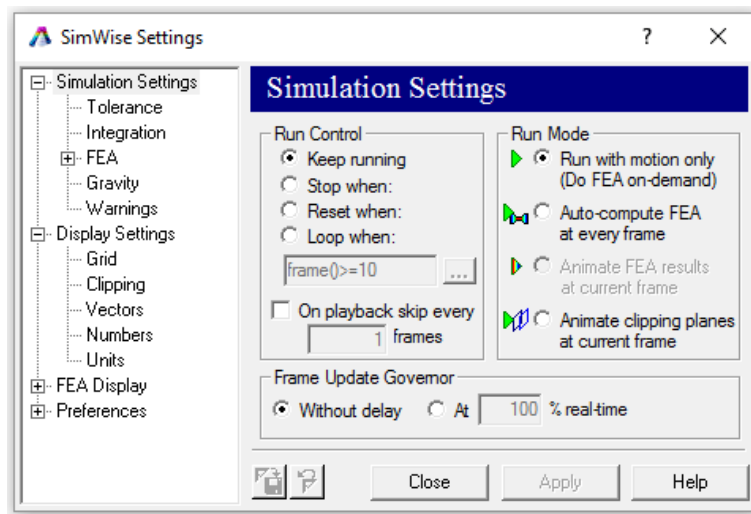


Figura 1.11: Ventana de configuración de SimWise 4D.

1.2. Matlab, Simulink y SimWise 4D

1.2.1. Acoplar el modelo a Simulink

En esta Sección se muestra cómo configurar nuestro modelo utilizando Simulink para su posterior simulación. Con todos los pasos anteriores, el modelo se encuentra listo para recibir y enviar información, por lo que podemos abrir Matlab Simulink, una vez aquí, buscamos en la librería de Simulink el bloque swPlant, que se encuentra en la sección SimWise 4D (como se muestra en la Fig. 1.12(a)), la cual se instala automáticamente al instalar SimWise 4D en la computadora, siempre y cuando esta cuente con una versión compatible de Matlab.

Después de esto, para configurar las entradas y salidas de la planta damos doble clic sobre el bloque y buscamos el archivo .wm3 correspondiente al modelo de estudio, después, seleccionamos de las entradas y salidas, que previamente definimos en el modelo de SimWise 4D, las que se vayan a ocupar para realizar la simulación, o de las que se requiera saber su información. Tal como se muestra en la Fig. 1.12(b), del sistema de estudio.

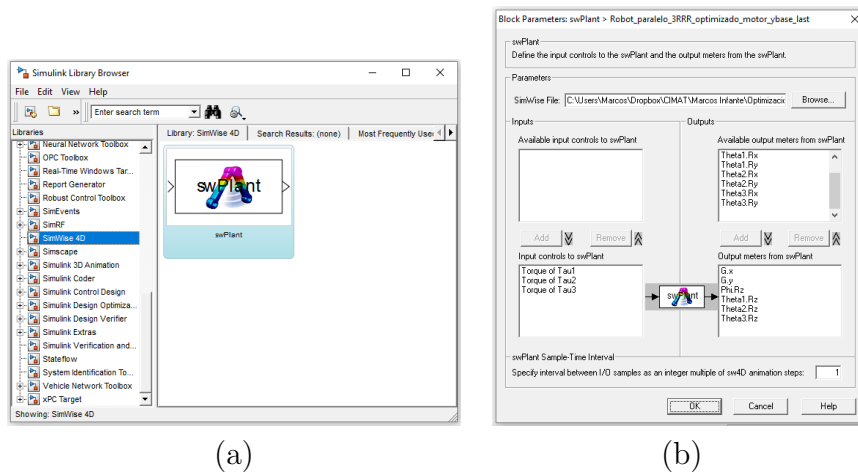


Figura 1.12: (a) Bloque Simulink de SimWise 4D. (b) Configuración de entradas y salidas.

1.2.1.1. Control con Simulink

Ahora que ya tenemos preparada la planta, es fácil definir un lazo de control para cada eslabón actuado del robot 3RRR, con el cual realizar simulaciones para el seguimiento de trayectorias. En la Fig. 1.13 se muestra el sistema implementado para el control del modelo.

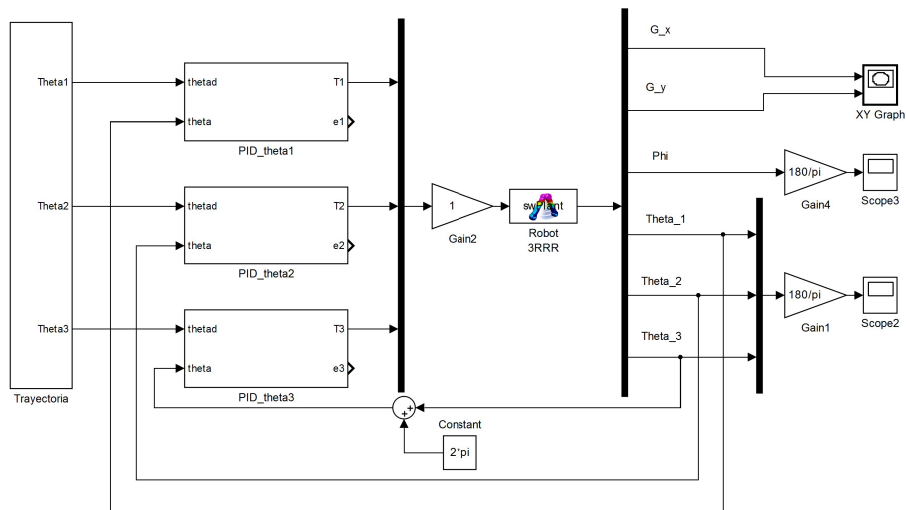


Figura 1.13: Sistema controlado en Simulink.

1.2.2. Ejecutar Simulink desde Matlab

En esta Sección se aborda el tema de ejecutar un modelo de Simulink desde el entorno del editor de Matlab, así como la modificación de los valores de cada bloque. Esto es muy importante, pues el modelo de SimWise está en el entorno de bloques de Simulink, y si quisiéramos mandar a evaluar la planta en múltiples ocasiones, tendríamos que correrlo manualmente cada vez. A continuación se muestran algunas instrucciones importantes para este propósito:

1. Instrucción para buscar el modelo en una dirección

- `find_system('Name' , 'Nombre del archivo');`

Ejemplo: `find_system('Name' , 'Control_3RRR');`

2. Instrucción para abrir el modelo una vez que se ha encontrado

- `open_system('Nombre del archivo');`

Ejemplo: `open_system('Control_3RRR');`

3. Instrucción para cambiar el valor de un parámetro de un bloque

- `set_param('Nombre del archivo/Subsistema/.../Nombre del Bloque', 'Tipo de bloque', 'Valor a asignar');`

Ejemplo: `set_param('Control_3RRR/PID_theta1/Gainkp1', 'Gain', '1');`

4. Iniciar simulación de un modelo en Simulink y activar el guardado de datos.

- `simOut=sim('Nombre del Archivo', 'SimulationMode', 'Modo de simulación', ... 'StopTime', 'Tiempo', 'SaveOutput', 'on', 'OutputSaveName', 'Nombre de los datos guardados');`

Ejemplo: `simOut=sim('Control_3RRR', 'SimulationMode', 'normal', ... 'StopTime', '3', 'SaveOutput', 'on', 'OutputSaveName', 'Datos');`

5. Guardar datos de salida del modelo.

- `Variable = simOut.get('Nombre de los datos guardados');`

Ejemplo: `Salida = simOut.get('Datos');`

1.2.2.1. Función de ejemplo

Esta es una función en MATLAB que modifica los valores PID del modelo a bloques que aparece en la Fig. 1.14 y retorna el valor de los errores de posición de los eslabones durante el seguimiento de una trayectoria propuesta, estos datos se adquieren del modelo Simulink mediante el bloque 'To Workspace'.

```
%% Inicia la función

function [salida1,salida2,salida3]=funcion_simulink_matlab(alpha)

find_system('Name','Control_3RRR'); %Buscar el modelo
open_system('Control_3RRR'); %Abre el modelo en simulink

kp=alpha(1); %Valor numérico asignado
ki=alpha(2); %Valor numérico asignado
kd=alpha(3); %Valor numérico asignado
%% Asignar el valor de las ganancias PID a cada controlador

%-----PID1-----%
set_param('Control_3RRR/PID_theta1/Gainkp1',...
    'Gain',char(num2str(kp)));%dar valores a la ganacia kp
set_param('Control_3RRR/PID_theta1/Gainki1','Gain',...
    char(num2str(ki)));%dar valores a la ganacia ki
set_param('Control_3RRR/PID_theta1/Gainkd1','Gain',...
    char(num2str(kd)));%dar valores a la ganacia kd
%-----PID2-----%
set_param('Control_3RRR/PID_theta2/Gainkp2','Gain',...
    char(num2str(kp)));%dar valores a la ganacia kp
set_param('Control_3RRR/PID_theta2/Gainki2','Gain',...
    char(num2str(ki)));%dar valores a la ganacia ki
set_param('Control_3RRR/PID_theta2/Gainkd2','Gain',...
    char(num2str(kd)));%dar valores a la ganacia kd
%-----PID3-----%
set_param('Control_3RRR/PID_theta3/Gainkp3','Gain',...
    char(num2str(kp)));%dar valores a la ganacia kp
set_param('Control_3RRR/PID_theta3/Gainki3','Gain',...
    char(num2str(ki)));%dar valores a la ganacia ki
set_param('Control_3RRR/PID_theta3/Gainkd3','Gain',...
    char(num2str(kd)));%dar valores a la ganacia kd
%% Inicia la Simulación del modelo en Simulink
```



```

simOut=sim('Control_3RRR','SimulationMode','normal','StopTime','10',...
    'SaveOutput','on','OutputSaveName','Datos1',...
    'SaveOutput','on','OutputSaveName','Datos2',...
    'SaveOutput','on','OutputSaveName','Datos3');
salida1 = simOut.get('Datos1');%Guarda el valor de Datos1 en salida1
salida2 = simOut.get('Datos2');%Guarda el valor de Datos2 en salida2
salida3 = simOut.get('Datos3');%Guarda el valor de Datos3 en salida3
%% %% Finaliza la Función

```

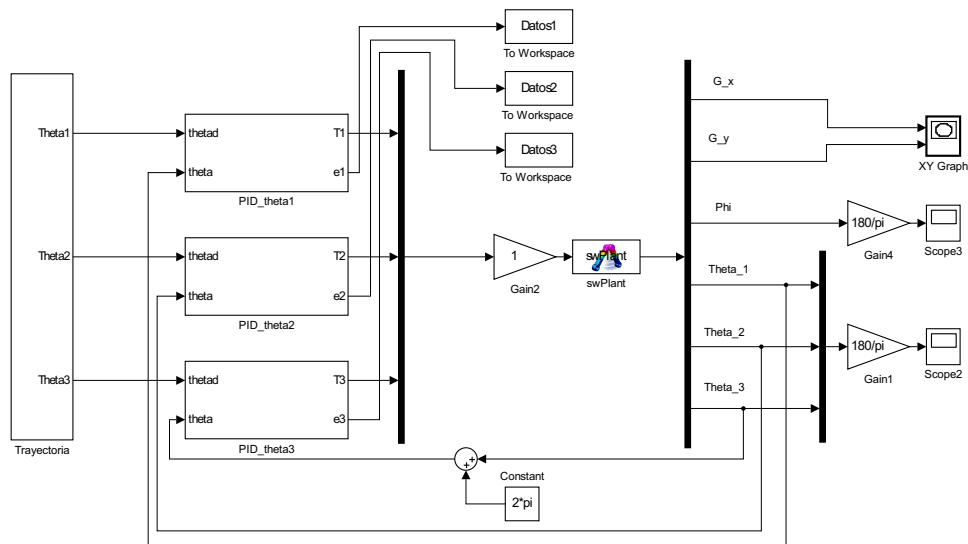
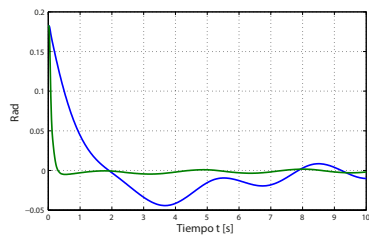


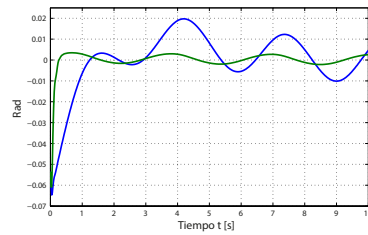
Figura 1.14: Sistema controlado con envío de datos

1.2.2.2. Resultados de simulación

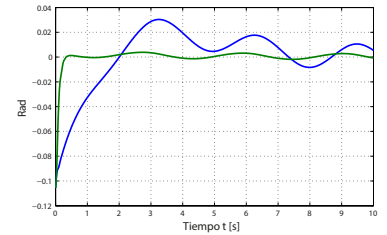
En la Fig. 1.15 se muestran algunos resultados de la ejecución de la función de ejemplo mostrada anteriormente con el nombre de: *funcion_simulink_matlab(α)* para dos vectores α de ganancias PID propuestos.



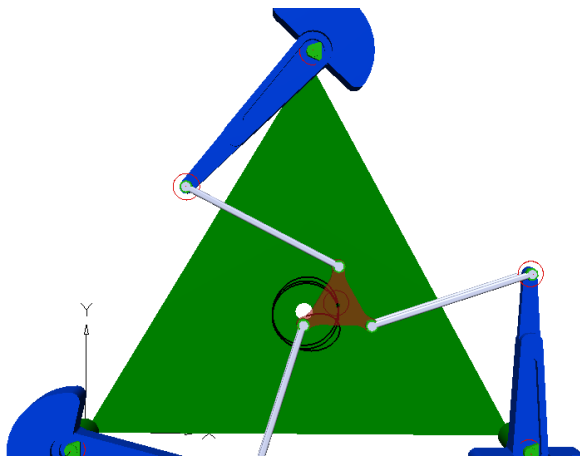
(a)



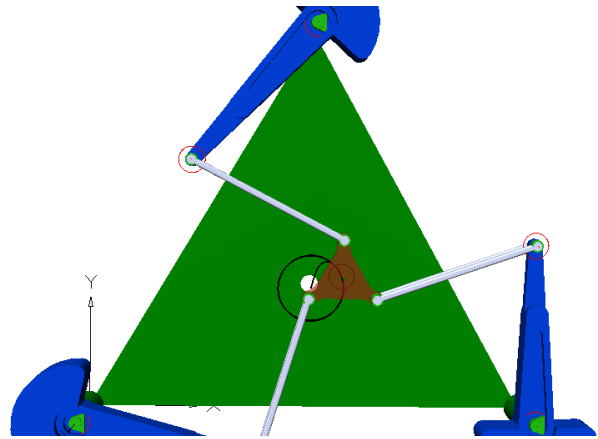
(b)



(c)



(d)



(e)

Figura 1.15: (a) Error de θ_1 (b) Error de θ_2 (c) Error de θ_3 , azul para $\alpha = [0.5, 0.2, 0.5]^T$ y verde para $\alpha = [5, 1, 0.5]^T$. (d) Trayectoria de salida para $\alpha = [0.5, 0.2, 0.5]^T$. (e) Trayectoria de salida para $\alpha = [5, 1, 0.5]^T$.

Capítulo 2

Descripción cinemática del robot 3RRR

En este Capítulo se muestran las ecuaciones que rigen la cinemática inversa del mecanismo, así como la matriz Jacobiana de éste.

El robot Planar 3RRR (Ver Fig. 2.1) de estudio está compuesto por 6 brazos del tipo revoluta anclados entre sí al centro del mecanismo, de los cuales, la mitad son actuados y la otra mitad son pasivos. Para el análisis del robot se empleó el método de ecuación de cerradura (loop-closure equation).

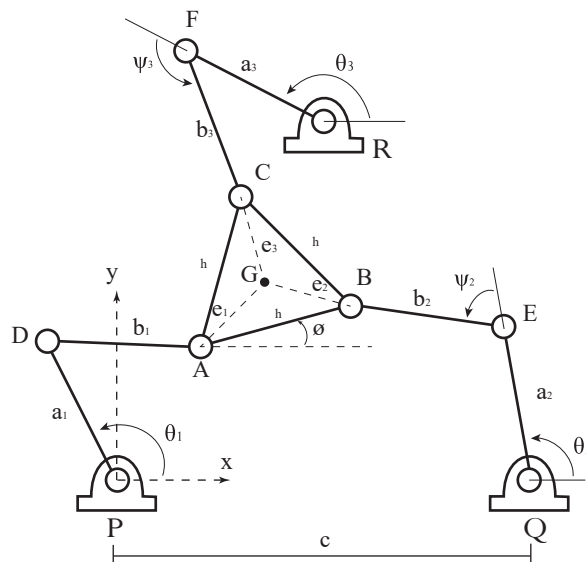


Figura 2.1: Robot paralelo manipulador 3RRR.

De la geometría de la Fig. 2.1, puede ser escrita una ecuación de cerradura como:

$$\overline{OA} = \overline{OP} + \overline{PD} + \overline{DA} \quad (2.1)$$

para el primer miembro, la cual da como resultado:

$$\begin{aligned} x_A &= x_p + a_1 \cos(\theta_1) + b_1 \cos(\theta_1 + \psi_1) \\ y_A &= y_p + a_1 \sin(\theta_1) + b_1 \sin(\theta_1 + \psi_1) \end{aligned} \quad (2.2)$$

Sumando el cuadrado de ambas ecuaciones anteriores, se obtiene la siguiente ecuación, con la que desaparece el ángulo pasivo ψ .

$$x_A^2 + y_A^2 - 2x_A a_1 \cos(\theta_1) - 2y_A a_1 \sin(\theta_1) + a_1^2 - b_1^2 = 0. \quad (2.3)$$

De igual manera, para los otros dos miembros restantes, las ecuaciones que describen su geometría son:

$$\begin{aligned} &x_A^2 + y_A^2 - 2x_A x_Q - 2y_A y_Q + x_Q^2 + y_Q^2 + h^2 + a_2^2 - b_2^2 + 2x_A h \cos(\phi) \\ &+ 2y_A h \sin(\phi) - 2x_A a_2 \cos(\theta_2) - 2y_A a_2 \sin(\theta_2) - 2a_2 h \cos(\phi) \cos(\theta_2) - 2x_Q h \cos(\phi) \\ &- 2y_Q h \sin(\phi) + 2x_Q a_2 \cos(\theta_2) + 2y_Q a_2 \sin(\theta_2) - 2a_2 h \sin(\phi) \sin(\theta_2) = 0 \end{aligned} \quad (2.4)$$

$$\begin{aligned} &x_A^2 + y_A^2 - 2x_A x_R - 2y_A y_R + x_R^2 + y_R^2 + h^2 + a_3^2 - b_3^2 + 2x_A h \cos(\phi + \frac{\pi}{3}) \\ &+ 2y_A h \sin(\phi + \frac{\pi}{3}) - 2x_A a_3 \cos(\theta_3) - 2y_A a_3 \sin(\theta_3) - 2a_3 h \cos(\phi + \frac{\pi}{3}) \cos(\theta_3) \\ &- 2x_R h \cos(\phi + \frac{\pi}{3}) - 2y_R h \sin(\phi + \frac{\pi}{3}) + 2x_R a_3 \cos(\theta_3) + 2y_R a_3 \sin(\theta_3) \\ &- 2a_3 h \sin(\phi + \frac{\pi}{3}) \sin(\theta_3) = 0 \end{aligned} \quad (2.5)$$

2.1. Cinemática Inversa

Para la Cinemática Inversa, x_A , y_A y ϕ son conocidos, y los ángulos de los eslabones motores θ_1 , θ_2 y θ_3 , deben ser calculados. Esto puede ser resuelto de la siguiente manera.

Para el eslabón 1, reescribimos la Ec. (2.3) de la forma siguiente:

$$e_1 \sin(\theta_1) + e_2 \cos(\theta_1) + e_3 = 0 \quad (2.6)$$

Donde:

$$\begin{aligned} e_1 &= -2y_A a_1, \\ e_2 &= -2x_A a_1, \\ e_3 &= x_A^2 + y_A^2 + a_1^2 - b_1^2. \end{aligned}$$

Sustituyendo las identidades trigonométricas de tangente de medio ángulo $\sin(\theta_i) = \frac{2t_i}{1+t_i^2}$ y $\cos(\theta_i) = \frac{1-t_i^2}{1+t_i^2}$, donde $t_i = \tan(\frac{\theta_i}{2})$ dentro de la Ec. (2.6), se obtiene:

$$(e_3 - e_2)t_1^2 + 2e_1 t_1 + (e_3 + e_2) = 0 \quad (2.7)$$

Resolviendo la Ec. (2.7) mediante la formula general para polinomios cuadrados, resulta:

$$\theta_1 = 2 \tan^{-1} \left(\frac{-e_1 \pm \sqrt{e_1^2 + e_2^2 - e_3^2}}{e_3 - e_2} \right) \quad (2.8)$$

De forma similar calculamos θ_2 y θ_3 para cada eslabón actuado.

$$\theta_2 = 2 \tan^{-1} \left(\frac{-e_4 \pm \sqrt{e_4^2 + e_5^2 - e_6^2}}{e_6 - e_5} \right) \quad (2.9)$$

$$\theta_3 = 2 \tan^{-1} \left(\frac{-e_7 \pm \sqrt{e_7^2 + e_8^2 - e_9^2}}{e_9 - e_8} \right) \quad (2.10)$$

Donde:

$$\begin{aligned} e_4 &= 2y_Q a_2 - 2y_A a_2 - 2a_2 h \sin(\phi) \\ e_5 &= 2x_Q a_2 - 2x_A a_2 - 2a_2 h \cos(\phi) \\ e_6 &= x_A^2 + y_A^2 - 2x_A x_Q - 2y_A y_Q + x_Q^2 + y_Q^2 + h^2 + a_2^2 - b_2^2 \\ &\quad + 2x_A h \cos(\phi) + 2y_A h \sin(\phi) - 2x_Q h \cos(\phi) - 2y_Q h \sin(\phi) \end{aligned}$$

$$\begin{aligned} e_7 &= 2y_R a_3 - 2y_A a_3 - 2a_3 h \sin\left(\phi + \frac{\pi}{3}\right) \\ e_8 &= 2x_R a_3 - 2x_A a_3 - 2a_3 h \cos\left(\phi + \frac{\pi}{3}\right) \\ e_9 &= x_A^2 + y_A^2 - 2x_A x_R - 2y_A y_R + x_R^2 + y_R^2 + h^2 + a_3^2 \\ &\quad - b_3^2 + 2x_A h \cos\left(\phi + \frac{\pi}{3}\right) + 2y_A h \sin\left(\phi + \frac{\pi}{3}\right) \\ &\quad - 2x_R h \cos\left(\phi + \frac{\pi}{3}\right) - 2y_R h \sin\left(\phi + \frac{\pi}{3}\right) \end{aligned}$$

Una vez obtenidos los valores de θ_1 , θ_2 y θ_3 , los ángulos ψ_1 , ψ_2 y ψ_3 pueden ser calculados a partir de la Ec. (2.2) y sus homólogas para los otros eslabones.

2.2. Cinemática diferencial

Para la obtención del Jacobiano, se procede usando el método de cadena de vectores de velocidad o velocity vector loop.

A partir de la Fig. 2.1, se pueden escribir las ecuaciones de cadena cerrada de cada eslabón de la siguiente forma:

Para el primer eslabón, la ecuación de cerradura es:

$$\overline{PG} + \overline{GA} = \overline{PD} + \overline{DA} \quad (2.11)$$

Donde sus componentes x e y son:

$$\begin{aligned} G_x - e_1 \cos\left(\phi + \frac{\pi}{6}\right) &= a_1 \cos(\theta_1) + b_1 \cos(\theta_1 + \psi_1) \\ G_y - e_1 \sin\left(\phi + \frac{\pi}{6}\right) &= a_1 \sin(\theta_1) + b_1 \sin(\theta_1 + \psi_1) \end{aligned}$$

Para el segundo eslabón:

$$\overline{PG} + \overline{GB} = \overline{PQ} + \overline{QE} + \overline{EB} \quad (2.12)$$

Donde sus componentes x e y son:

$$\begin{aligned} G_x + e_2 \cos\left(-\phi + \frac{\pi}{6}\right) &= x_Q + a_2 \cos(\theta_2) + b_2 \cos(\theta_2 + \psi_2) \\ G_y - e_2 \sin\left(-\phi + \frac{\pi}{6}\right) &= y_Q + a_2 \sin(\theta_2) + b_2 \sin(\theta_2 + \psi_2) \end{aligned}$$

Para el tercer eslabón:

$$\overline{PG} + \overline{GC} = \overline{PR} + \overline{RF} + \overline{FC} \quad (2.13)$$

Donde sus componentes x e y son:

$$\begin{aligned} G_x - e_3 \sin(\phi) &= x_R + a_3 \cos(\theta_3) + b_3 \cos(\theta_3 + \psi_3) \\ G_y + e_3 \cos(\phi) &= y_R + a_3 \sin(\theta_3) + b_3 \sin(\theta_3 + \psi_3) \end{aligned}$$

Derivando respecto al tiempo las Ec. (2.11), (2.12) y (2.13), obtenemos las ecuaciones de cerradura de velocidad:

$$\mathbf{b}_1 \cdot \mathbf{v}_g + \dot{\phi} \mathbf{k} \cdot (\mathbf{e}_1 \times \mathbf{b}_1) = \dot{\theta}_1 \mathbf{k} \cdot (\mathbf{a}_1 \times \mathbf{b}_1) \quad (2.14)$$

$$\mathbf{b}_2 \cdot \mathbf{v}_g + \dot{\phi} \mathbf{k} \cdot (\mathbf{e}_2 \times \mathbf{b}_2) = \dot{\theta}_2 \mathbf{k} \cdot (\mathbf{a}_2 \times \mathbf{b}_2) \quad (2.15)$$

$$\mathbf{b}_3 \cdot \mathbf{v}_g + \dot{\phi} \mathbf{k} \cdot (\mathbf{e}_3 \times \mathbf{b}_3) = \dot{\theta}_3 \mathbf{k} \cdot (\mathbf{a}_3 \times \mathbf{b}_3) \quad (2.16)$$

Reescribiendo las Ec. (2.14), (2.15) y (2.16) en forma matricial, podemos definir el sistema como:

$$J_x \dot{\mathbf{x}} = J_q \dot{\mathbf{q}}, \quad (2.17)$$

$$\dot{\mathbf{q}} = J_q^{-1} J_x \dot{\mathbf{x}} \quad (2.18)$$

Donde:

$$J_x \dot{\mathbf{x}} = \begin{bmatrix} b_{1x} & b_{1y} & e_{1x}b_{1y} - e_{1y}b_{1x} \\ b_{2x} & b_{2y} & e_{2x}b_{2y} - e_{2y}b_{2x} \\ b_{3x} & b_{3y} & e_{3x}b_{3y} - e_{3y}b_{3x} \end{bmatrix} \begin{bmatrix} v_{gx} \\ v_{gy} \\ \dot{\phi} \end{bmatrix}$$

$$J_q \dot{\mathbf{q}} = \begin{bmatrix} a_{1x}b_{1y} - a_{1y}b_{1x} & 0 & 0 \\ 0 & a_{2x}b_{2y} - a_{2y}b_{2x} & 0 \\ 0 & 0 & a_{3x}b_{3y} - a_{3y}b_{3x} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}$$

Nótese que la matriz Jacobiana definida en la Ec. (2.18) para el manipulador paralelo corresponde al Jacobiano inverso de un robot serial.

Esta definición de la matriz Jacobiana difiere un poco de la que tradicionalmente se usa para los robot manipuladores seriales, donde el Jacobiano realiza una transformación a partir de las velocidades de las uniones a la velocidad del efector final, este cambio en la definición del Jacobiano para los robots paralelos se deriva a partir de la dualidad existente entre los manipuladores seriales y paralelos (ver [9]), y se realiza por conveniencia.

2.2.0.3. Homogeneización de la matriz Jacobiana

Debido a que el mecanismo de estudio presenta grados de libertad traslacionales y rotacionales, la matriz Jacobiana previamente obtenida presenta disparidad en sus dimensiones, por lo que se hace necesario realizar algún procedimiento para lograr homogeneizar dicha matriz. Lipkin y Duffy en [6], mencionan que usar el numero de condición de una matriz Jacobiana no homogénea en procesos de control u optimización, puede ocasionar problemas, pues dicho numero se ve modificado al variar el tamaño en escala del mecanismo, con lo que hace inviable el uso de dicho numero.

Varios investigadores han trabajado en la obtención de matrices Jacobianas dimensionalmente homogéneas, Gosselin en [2], propone expresar la velocidad del efector final

mediante la descripción de dos puntos ubicados en el efector, obteniendo así una matriz Jacobiana homogénea de (4×3) . Kim y Ryu en [5], proponen una nueva formulación para el cálculo de la matriz Jacobiana homogénea para manipuladores paralelos, usando tres puntos ubicados en el efector final en lugar de dos, esta metodología puede ser aplicada a robots con mas de tres grados de libertad y resulta en una matriz Jacobiana dimensionalmente homogénea, cuyo tamaño es de (6×9) para manipuladores de 6-GDL o (3×6) para 3-GDL. Otro método utilizado por Hosseini, Daniali y Taghirad en [4], y Lou, Liu y Li en [10], muestran una forma distinta y mas práctica de lograr homogeneizar la matriz Jacobiana, basada en la división de las columnas no uniformes (con unidades) sobre un factor de ponderación que desaparezca dichas no uniformidades. Siguiendo el método propuesto en [10], podemos reescribir la matriz Jacobiana (2.18) obtenida en el capítulo anterior de la siguiente forma:

$$\hat{J} = \begin{bmatrix} \frac{b_{1x}}{a_{1x}b_{1y}-a_{1y}b_{1x}} & \frac{b_{1y}}{a_{1x}b_{1y}-a_{1y}b_{1x}} & \frac{\mathbf{k} \cdot (\mathbf{e}_1 \times \vec{b}_1) |\vec{e}_1|}{(a_{1x}b_{1y}-a_{1y}b_{1x}) |\vec{e}_1|} \\ \frac{b_{2x}}{a_{2x}b_{2y}-a_{2y}b_{2x}} & \frac{b_{2y}}{a_{2x}b_{2y}-a_{2y}b_{2x}} & \frac{\mathbf{k} \cdot (\mathbf{e}_2 \times \vec{b}_2) |\vec{e}_2|}{(a_{2x}b_{2y}-a_{2y}b_{2x}) |\vec{e}_2|} \\ \frac{b_{3x}}{a_{3x}b_{3y}-a_{3y}b_{3x}} & \frac{b_{3y}}{a_{3x}b_{3y}-a_{3y}b_{3x}} & \frac{\mathbf{k} \cdot (\mathbf{e}_3 \times \vec{b}_3) |\vec{e}_3|}{(a_{3x}b_{3y}-a_{3y}b_{3x}) |\vec{e}_3|} \end{bmatrix} \quad (2.19)$$

Con esto, obtenemos una matriz Jacobiana \hat{J} escalada cinemáticamente, la cual es dimensionalmente homogénea, y con la que es posible evaluar la manipulabilidad y/o destreza del mecanismo sin importar su escalamiento en tamaño.

Capítulo 3

Optimización

En general, la optimización de parámetros tanto cinemáticos como dinámicos es importante en el desarrollo de mecanismos, pero se vuelve un problema obligatorio si es que se desea explotar al máximo la capacidad de una plataforma paralela. En este Capítulo se presentan tres casos principales de optimización, 1. Optimización del espacio de trabajo, 2. Optimización de la destreza del robot para trayectorias definidas y 3. Optimización de parámetros de control dinámico para el seguimiento de trayectorias definidas. Para el proceso de optimización, se utilizaron los algoritmos CMA-ES [3], BUMDA [8] y NSGA-II [7], los cuales son una estrategia evolutiva, un estimador de distribución y un algoritmo genético respectivamente, todos, aunque cada uno tiene su manera particular de trabajar, se agrupan dentro de la categoría de algoritmos evolutivos. Para mas información, favor de revisar la bibliografía señalada.

3.1. Optimización del espacio regular de trabajo

En esta Sección se aborda el problema de maximizar el espacio de trabajo del robot paralelo planar 3RRR, para llevar a cabo esto, se propone encontrar el máximo espacio de trabajo regular, donde la suma de las longitudes de los eslabones de cada brazo del robot está normalizada a la unidad. Este problema está sujeto a restricciones que garanticen un funcionamiento adecuado del sistema, y que lo mantengan alejado de puntos singulares.

Para su fácil construcción y manufactura, se considera una estructura simétrica, donde todos los eslabones actuados son de igual longitud, y todos los eslabones pasivos son idénticos entre sí, además, las bases actuadas están a la misma distancia una de la otra, formando un triangulo equilátero, al igual que la plataforma móvil, es decir, refiriéndonos al esquema del robot mostrado en la Fig. 2.1, $a_i = a, b_i = b, e_i = e$ para $i = 1, 2, 3$ y $\|A_i A_j\| = c$, para $i, j = 1, 2, 3$, donde $i \neq j$.

Con lo anterior, podemos establecer los siguientes parámetros de diseño.

$$\alpha = [a, b, e, c]^T$$

Bajo el supuesto de que en la mayoría de máquinas es deseable un espacio regular de trabajo, ya sea un cubo, cilindro o esfera para sistemas tridimensionales o un cuadrado o círculo para sistemas planares, se considera un espacio de trabajo regular para el robot 3RRR en forma de cuadrado.

El punto central del espacio de trabajo estará centrado en $[\frac{c}{2}, \frac{c}{\sqrt{12}}]$, el cual coincide con el centro de la plataforma fija.

Con lo que la función objetivo utilizada es la mitad de un lado del cuadrado que se puede dibujar dentro de un área delimitada por el inverso del número de condición de la matriz Jacobiana, es decir:

$$\underset{\alpha}{\text{máx}} \quad \ell \tag{3.1}$$

sujeta a

$$\begin{aligned} \kappa(\hat{J}(x, \theta, \alpha)) &\geq 0.2; \\ -\frac{\pi}{3} &\leq \theta_1(x, \alpha) \leq \frac{\pi}{3} \\ \frac{\pi}{3} &\leq \theta_1(x, \alpha) \leq \pi \\ \pi &\leq \theta_1(x, \alpha) \leq \frac{5\pi}{3} \\ \phi &= 0^\circ \\ a + b + e &= 1 \\ a, b &\in [0.2, 1], e \in [0.1, 0.2], c \in [0.8, 2] \end{aligned}$$

3.1.1. Resultados obtenidos

Para el proceso de optimización del espacio de trabajo regular se ocuparon 3 algoritmos evolutivos para evaluar la función objetivo, tales algoritmos fueron: **CMAES**, **BUMDA** y **NSGA-II** Los mejores resultados obtenidos por cada algoritmo son los mostrados en la Tabla 3.1.

Como se puede ver en la Tabla 3.1, aunque los resultados son muy similares entre sí, el mejor resultado fue el obtenido por el algoritmo **CMAES**, cuyo espacio de trabajo regular forma un cuadrado con $2\ell = 0.46438$ por lado, dicho resultado se puede observar gráficamente en la Fig. 3.1.

Tabla 3.1: Mejores soluciones a la Ec. (3.1), longitud de los parámetros de diseño $\{a, b, e, c\}$ y el valor de la función objetivo $\mathcal{F}(\alpha)$.

AE	a	b	e	c	$\mathcal{F}(\alpha)$
BUMDA	0.476696	0.414219	0.109084	1.164001	0.230119
CMAES	0.481101	0.418833	0.100066	1.158919	0.232190
NSGA-II	0.480777	0.418933	0.100290	1.159674	0.231912

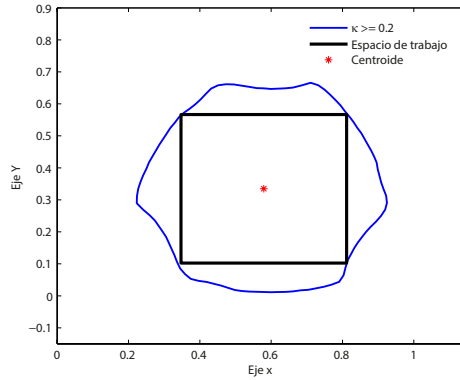


Figura 3.1: Espacio max. de trabajo regular con $\ell = 0.23219$, obtenido por CMAES.

3.2. Optimización de la destreza del robot para trayectorias definidas

Con el espacio de trabajo máximo obtenido en el caso de optimización anterior, se trazaron tres trayectorias discretas distintas lo mas grande posible en el espacio regular dado, y se procedió a realizar la optimización de las longitudes del robot tratando de maximizar la destreza del robot al desarrollar dicha trayectoria.

Las trayectorias propuestas fueron discretizadas en 360 puntos independientes del tiempo y son expresadas por las Ecs. (3.2), (3.3) y (3.4). Estas trayectorias se muestran gráficamente en la Fig. 3.2.

Ecuación del Circulo

$$\begin{aligned}
 x(p) &= Gx + \ell \cos(p) \\
 y(p) &= Gy + \ell \sin(p) \\
 \phi(p) &= 0
 \end{aligned} \tag{3.2}$$

Ecuación de la rosa de 3 pétalos

$$\begin{aligned} x(p) &= Gx + \ell \cos(p) \cos(3p) \\ y(p) &= Gy + \ell \sin(p) \cos(3p) \\ \phi(p) &= 0 \end{aligned} \quad (3.3)$$

Ecuación del caracol de Pascal

$$\begin{aligned} x(p) &= (0.905\ell + Gx) + 0.6005\ell \cos(p) - (1.8)(0.6005)\ell \cos(p)^2 \\ y(p) &= Gy + 0.6005\ell \sin(p) - (1.8)(0.6005)\ell \sin(p) \cos(p) \\ \phi(p) &= 0 \end{aligned} \quad (3.4)$$

Donde ℓ es el mejor resultado obtenido en el proceso de optimización anterior, mostrado en la Tabla 3.1.

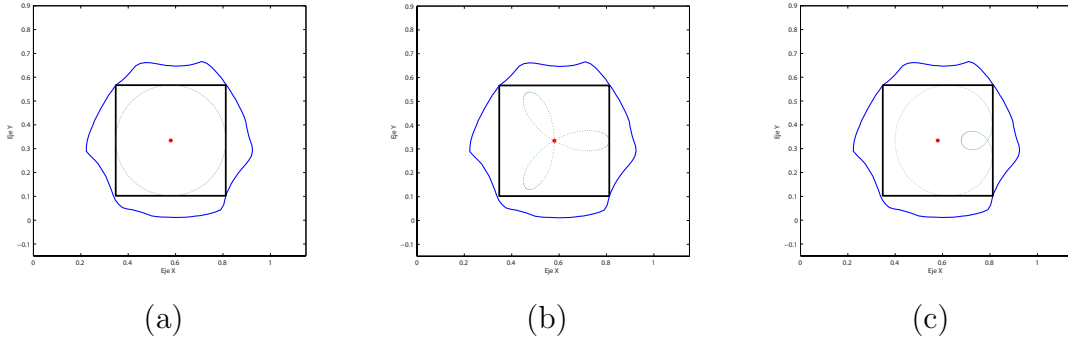


Figura 3.2: Trayectorias propuestas en el espacio de trabajo máximo para la Ec. (3.1). (a) Ec. (3.2) Circulo, (b) Ec. (3.3) Rosa de 3 petalos, (c) Ec. (3.4) Caracol de Pascal.

3.2.1. Función objetivo

Dada una trayectoria discretizada, se trata de maximizar la integral del inverso del numero de condición de la matriz Jacobiana evaluada en cada punto de dicha trayectoria, con esto, la función objetivo la definimos de la siguiente forma:

$$\max_{\alpha} \mathcal{F}(\alpha) = \int_{p=p_0}^{p_n} \kappa(\alpha) dp \quad (3.5)$$

sujeta a

$$\begin{aligned}
&\kappa(J(x, \theta, \alpha)) \geq 0.2; \\
&-\frac{\pi}{3} \leq \theta_1(x, \alpha) \leq \frac{\pi}{3} \\
&\frac{\pi}{3} \leq \theta_1(x, \alpha) \leq \pi \\
&\pi \leq \theta_1(x, \alpha) \leq \frac{5\pi}{3} \\
&a + b + e = 1 \\
&a, b \in [0.2, 1], e \in [0.1, 0.2], c \in [0.8, 2]
\end{aligned}$$

3.2.2. Resultados obtenidos

En la Tabla 3.2 se muestran los mejores resultados de la optimización de la Ec. (3.5) dados por el algoritmo BUMDA. Obsérvese que la trayectoria en forma de flor expresada por la Ec. (3.3) presenta un valor mayor en destreza que las otras dos trayectorias, esto debido a que pasa por el centro del espacio de trabajo del mecanismo, que es donde se encuentran los valores más grandes para el inverso del número de condición. En la Fig. 3.3 se muestran las curvas de nivel para cada configuración de este proceso de optimización mostrado en la Tabla 3.2.

Tabla 3.2: Soluciones a la Ec. (3.5) para la trayectorias mostradas, longitud de los parámetros de diseño $\{a, b, e, c\}$ y el valor de la función objetivo $\mathcal{F}(\alpha)$.

Trayectoria	a	b	e	c	$\mathcal{F}(\alpha)$
Mejores soluciones dadas por BUMDA					
Circulo, Ec. (3.2)	0.4733	0.4263	0.1004	1.0994	113.7561
Rosa de 3 petalos, Ec. (3.3)	0.4440	0.4549	0.1011	1.1572	186.7724
Caracol de Pascal, Ec. (3.4)	0.4783	0.4204	0.1013	1.2272	143.9334
Ecs. (3.2), (3.3) y (3.4) juntas.	0.4691	0.4299	0.1009	1.1626	433.7659

3.3. Optimización de parámetros de control dinámico

En esta Sección se aborda el problema de la obtención de las ganancias de control PID via la minimización de la energía aplicada al mecanismo y la minimización del error durante el seguimiento de una trayectoria definida. Para esto, dada una trayectoria dependiente del tiempo y el grupo de longitudes óptimas obtenidas para las tres trayectorias mezcladas mostrada en la Tabla 3.2 en el problema de diseño anterior,

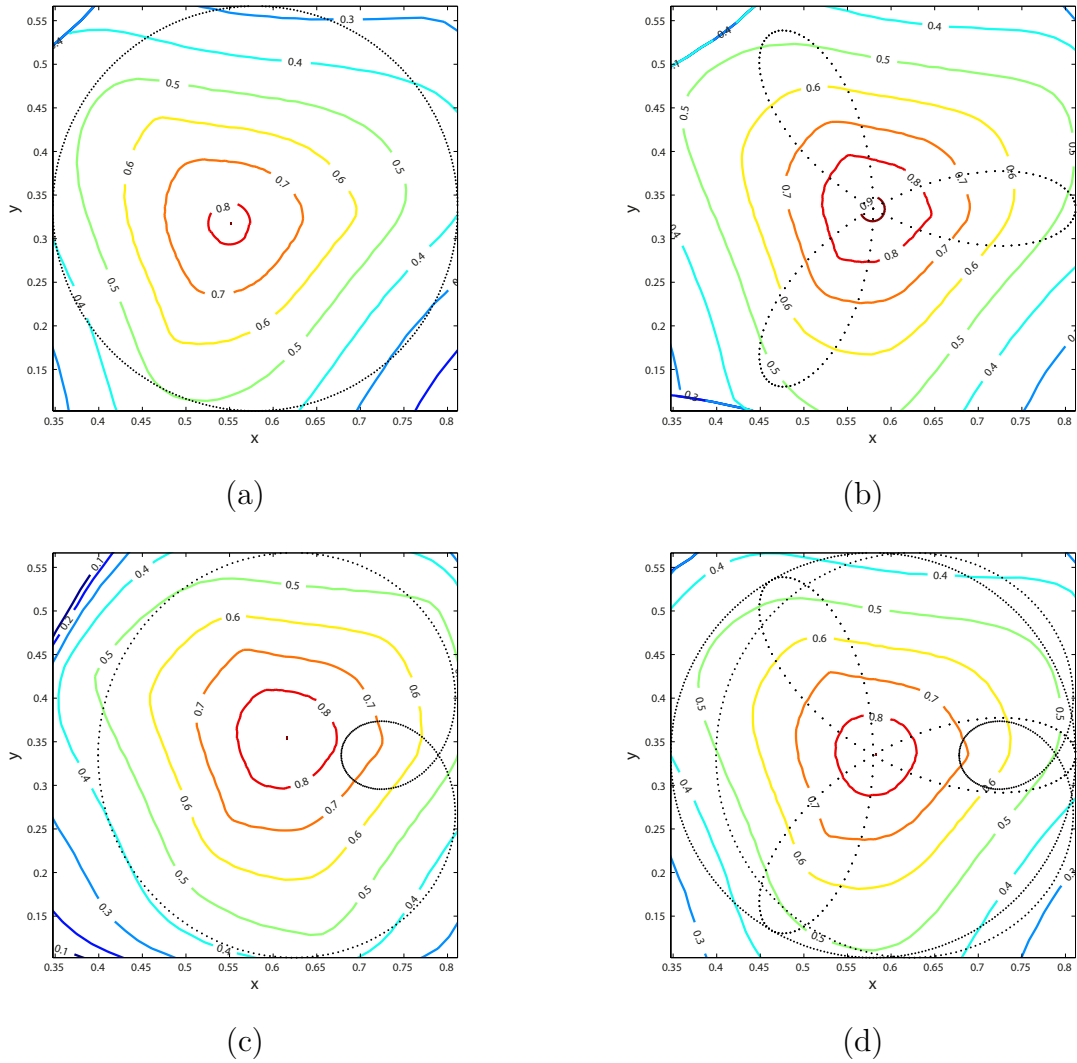


Figura 3.3: Curvas de nivel del inverso del numero de condición de la matriz Jacobiana asociadas a los parámetros óptimos mostrados en la Tabla 3.2 para: (a) Circulo Ec. (3.2), (b) Rosa de 3 pétalos Ec. (3.3) (c) Caracol de Pascal Ec. (3.4) (d) Ecs. (3.2), (3.3) y (3.4) juntas.

se obtienen un conjunto de parámetros de control optimizados que permiten un buen desempeño según la función objetivo propuesta.

Este problema lo podemos dividir en tres subproblemas, el primero, donde se requiere que el robot siga una trayectoria con un cierto grado de error, pero que a su vez use la menor cantidad de energía, el segundo caso, cuando lo único que nos interesa es el minimizar la cantidad de error entre la trayectoria deseada y la trazada por el sistema,

y por último, cuando lo que se requiere es que el robot gaste la menor cantidad de energía para realizar el seguimiento de una trayectoria.

Este trabajo se desarrollo bajo la metodología propuesta en [1], donde se lleva a cabo un caso de optimización similar, la diferencia radica principalmente, en que en este trabajo se evitó el cálculo del modelo dinámico del robot, haciendo uso de los resultados de simulación dinámica del modelo virtual implementado en SimWise 4D y Matlab-Simulink, como se mostró en el Capítulo 1.

Para este proceso de optimización se escalaron las medidas de la geometría del robot, de manera que la distancia entre actuadores fuese igual a 50cm.

En la Tabla 3.3 se muestran los parámetros físicos del robot paralelo planar 3RRR utilizados durante la optimización de las ganancias para el control dinámico.

Tabla 3.3: Parámetros físicos del modelo virtual del robot 3RRR

Parámetros	Eslabón actuado	Eslabón pasivo	Plataforma móvil
Masa (kg)	0.953	0.0797	0.224
Longitud (m)	$a = 0.2017$	$b = 0.185$	$e = 0.043$
Inercia (kgm^2)	0.0037	0.00103	0.00016

3.3.1. Función objetivo 1: Minimización simultanea del error y energía consumida

En este caso, dada una trayectoria dependiente del tiempo, la función objetivo es definida como la suma de dos funciones independientes, una para el error de seguimiento y otra para el consumo de energía. Dichas funciones se describen a continuación:

La primera es la suma de las integrales de los errores entre el valor deseado y la posición angular de cada eslabón actuado, y la segunda es la suma de las integrales del torque aplicado por cada actuador, en ambos casos, durante un tiempo determinado.

$$\min_{\alpha} \mathcal{F}(\alpha) = \omega_1 \sum_{j=1}^d \int_{t=t_0}^{t_n} \|e^{(j)}(t)\| + \omega_2 \sum_{j=1}^d \int_{t=t_0}^{t_n} \|\Delta\tau^{(j)}(t)\| \quad (3.6)$$

sujeta a

$$\begin{aligned} -\frac{\pi}{3} &\leq \theta_1(x, \alpha) \leq \frac{\pi}{3} \\ \frac{\pi}{3} &\leq \theta_2(x, \alpha) \leq \pi \end{aligned}$$

$$\pi \leq \theta_1(x, \alpha) \leq \frac{5\pi}{3}$$

$$K_p, K_i, K_d \in [0, 11]$$

Donde ω_1 y ω_2 son pesos de la importancia de cada función, que deberá establecer el diseñador de acuerdo a sus requerimientos de que tanto desea que el robot sea capaz de seguir una trayectoria o que tanta energía sea capaz de ahorrar.

Obsérvese también, que este problema puede plantearse como un caso multiobjetivo, tratando de encontrar un equilibrio entre las dos funciones objetivo.

3.3.1.1. Resultados

En la Tabla 3.4 se muestran los resultados obtenidos de la evaluación de la función objetivo (3.6) con las trayectorias definidas por las Ecs. (3.2), (3.3) y (3.4) durante un tiempo de 3 seg. con una frecuencia de $\omega = \frac{2}{\pi}$ para el círculo y el caracol de pascal, y $\omega = \frac{1}{\pi}$ para la rosa de 3 pétalos utilizando el algoritmo BUMDA para el proceso de optimización.

Estos resultados, se pueden observar gráficamente en las Figs. 3.4, 3.5 y 3.6.

Tabla 3.4: Mejores soluciones a la Ec. (3.6), usando pesos de $\omega_1 = 0.99$ y $\omega_2 = 0.01$, ganancias del controlador PID $\{K_p, K_i, K_d\}$ y el valor de la función objetivo \mathcal{F} , Ec. (3.7)(\mathcal{F}_1) y Ec. (3.8)(\mathcal{F}_2)

Trayectoria	K_p	K_i	K_d	\mathcal{F}_1	\mathcal{F}_2	\mathcal{F}
Mejores soluciones dadas por BUMDA						
Círculo	9.1869	3.9540	0.4459	0.1942	5.3740	0.2460
Rosa de 3 pétalos	6.9383	5.1690	0.3965	0.1782	3.0804	0.2072
Caracol de Pascal	6.0973	0.1784	0.2749	0.3254	4.5408	0.3675

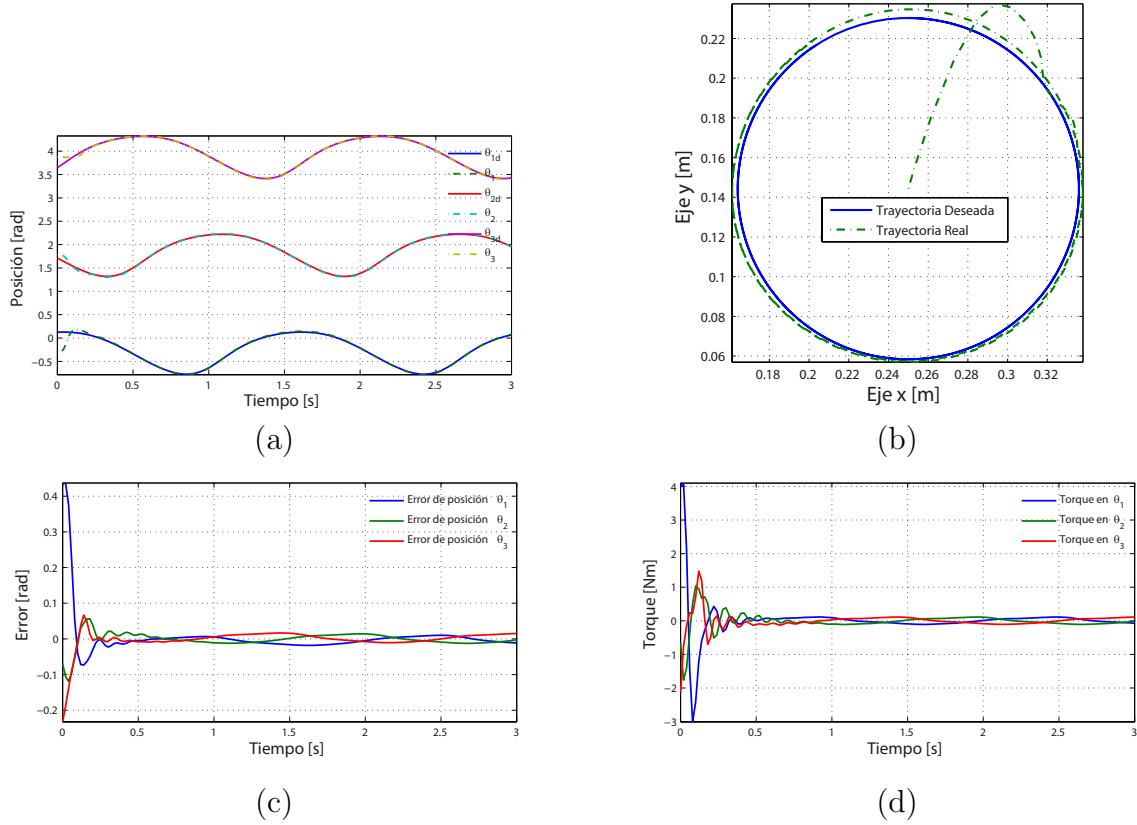


Figura 3.4: Gráficas de la mejor solución encontrada para la Ec. (3.6) mostrada en la Tabla 3.4 referentes a la trayectoria del círculo Ec. (3.2). (a) Posición deseada vs posición real de los eslabones actuados. (b) Trayectoria deseada vs trayectoria desarrollada por el efector final. (c) Error en la posición de los eslabones actuados. (d) Torque desarrollado por los eslabones actuados

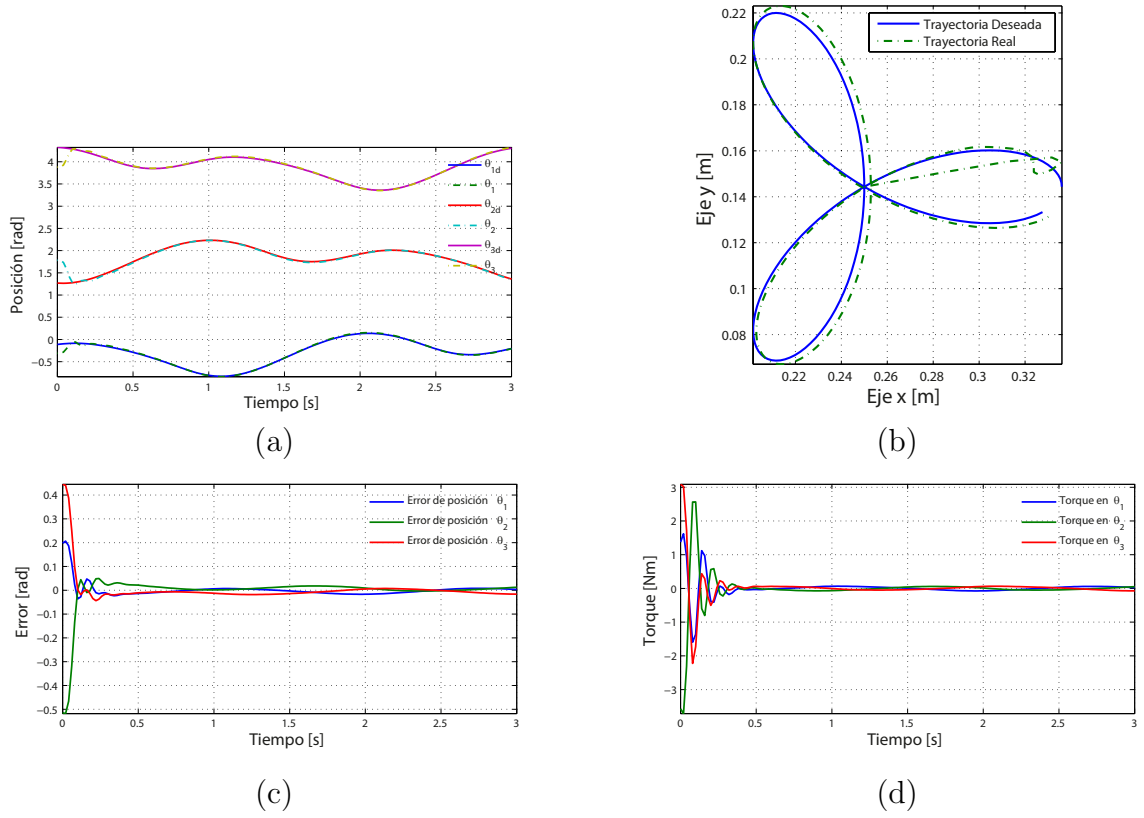


Figura 3.5: Graficas de la mejor solución encontrada para la Ec. (3.6) mostrada en la Tabla 3.4 referentes a la trayectoria de la rosa de 3 pétalos Ec. (3.3). (a) Posición deseada vs posición real de los eslabones actuados. (b) Trayectoria deseada vs trayectoria desarrollada por el efector final. (c) Error en la posición de los eslabones actuados. (d) Torque desarrollado por los eslabones actuados.

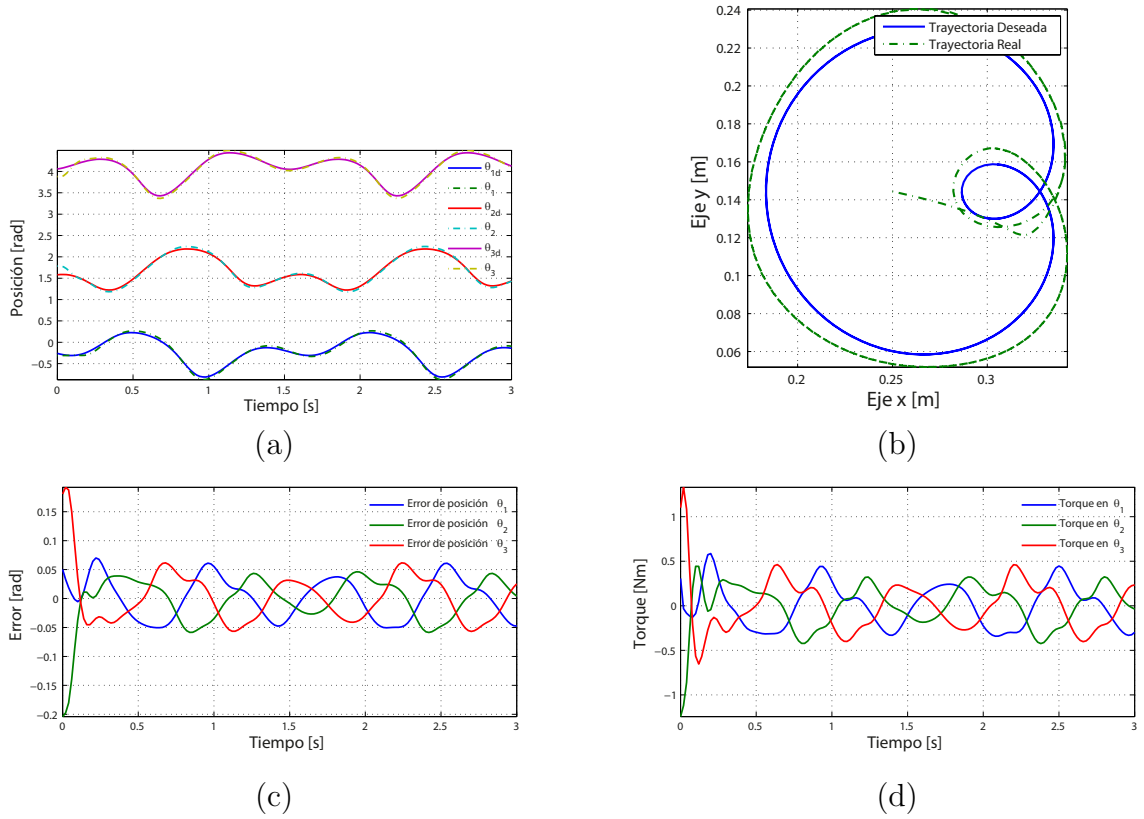


Figura 3.6: Gráficas de la mejor solución encontrada para la Ec. (3.6) mostrada en la Tabla 3.4 referentes a la trayectoria del caracol de Pascal Ec. (3.4). (a) Posición deseada vs posición real de los eslabones actuados. (b) Trayectoria deseada vs trayectoria desarrollada por el efector final. (c) Error en la posición de los eslabones actuados. (d) Torque desarrollado por los eslabones actuados.

3.3.2. Función objetivo 2: Minimización del error en los eslabones actuados

En este caso, se aborda el problema de la minimización del error en el seguimiento de trayectorias. Si en la Ec. (3.6) establecemos que $\omega_1 = 1$ y $\omega_2 = 0$, la función objetivo queda únicamente expresada en función del error acumulado en un lapso de tiempo, como se muestra en la Ec. (3.7) .

$$\min_{\alpha} \mathcal{F}(\alpha) = \sum_{j=1}^d \int_{t=t_0}^{t_n} \|e^{(j)}(t)\| \quad (3.7)$$

sujeta a

$$\begin{aligned} -\frac{\pi}{3} &\leq \theta_1(x, \alpha) \leq \frac{\pi}{3} \\ \frac{\pi}{3} &\leq \theta_1(x, \alpha) \leq \pi \\ \pi &\leq \theta_1(x, \alpha) \leq \frac{5\pi}{3} \\ K_p, K_i, K_d &\in [0, 11] \end{aligned}$$

Con esta función objetivo, se asegura que el robot será lo mas preciso posible en el seguimiento de la trayectoria, sin importarle cuanta energía se consuma.

3.3.2.1. Resultados

En la Tabla 3.5 se muestran los resultados obtenidos de la evaluación de la función objetivo (3.7) durante un tiempo de 3 seg. con el algoritmo BUMDA, dichos resultados se pueden observar gráficamente en las Figs. 3.7, 3.8 y 3.9.

Tabla 3.5: Mejores soluciones a la Ec. (3.7), Ganancias del controlador PID $\{K_p, K_i, K_d\}$ y el valor de la función objetivo \mathcal{F}_1 y Ec. (3.8)(\mathcal{F}_2)

Trayectoria	K_p	K_i	K_d	\mathcal{F}_1	\mathcal{F}_2
Mejores soluciones dadas por BUMDA					
Circulo	10.7131	0.2345	0.5403	0.1798	6.3779
Rosa de 3 pétalos	8.6487	0.3876	0.4704	0.1504	3.6416
Caracol de Pascal	7.7095	1.9882	0.3503	0.2787	5.5017

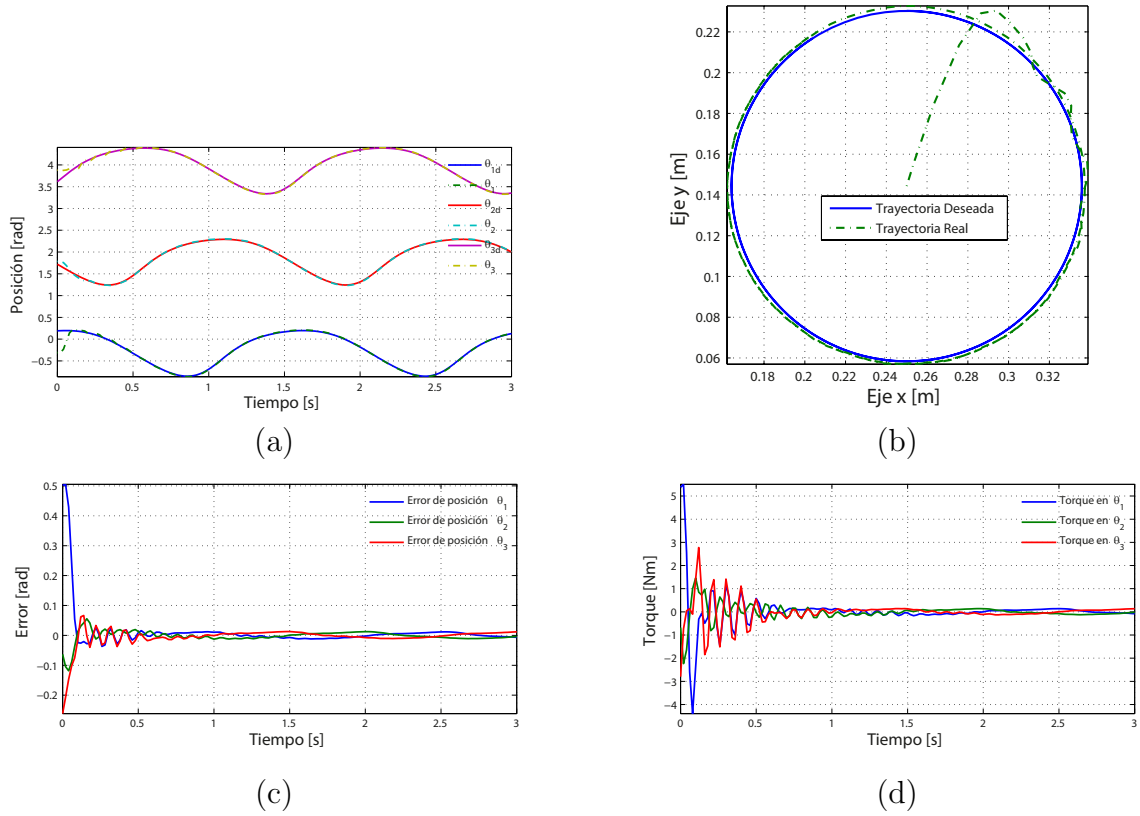


Figura 3.7: Graficas de la mejor solución encontrada para la Ec. (3.7) mostrada en la Tabla 3.5 referentes a la trayectoria del círculo Ec. (3.2). (a) Posición deseada vs posición real de los eslabones actuados. (b) Trayectoria deseada vs trayectoria desarrollada por el efector final. (c) Error en la posición de los eslabones actuados. (d) Torque desarrollado por los eslabones actuados

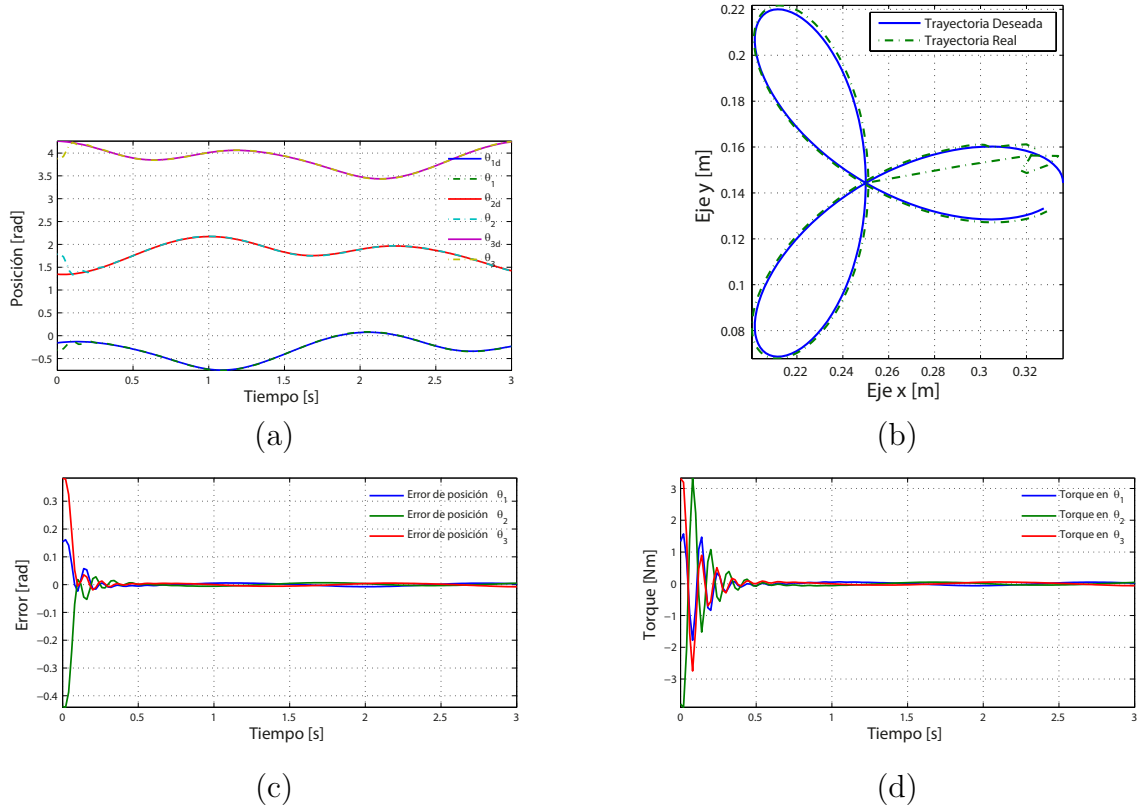


Figura 3.8: Gráficas de la mejor solución encontrada para la Ec. (3.7) mostrada en la Tabla 3.5 referentes a la trayectoria de la rosa de 3 pétalos Ec. (3.3). (a) Posición deseada vs posición real de los eslabones actuados. (b) Trayectoria deseada vs trayectoria desarrollada por el efector final. (c) Error en la posición de los eslabones actuados. (d) Torque desarrollado por los eslabones actuados.

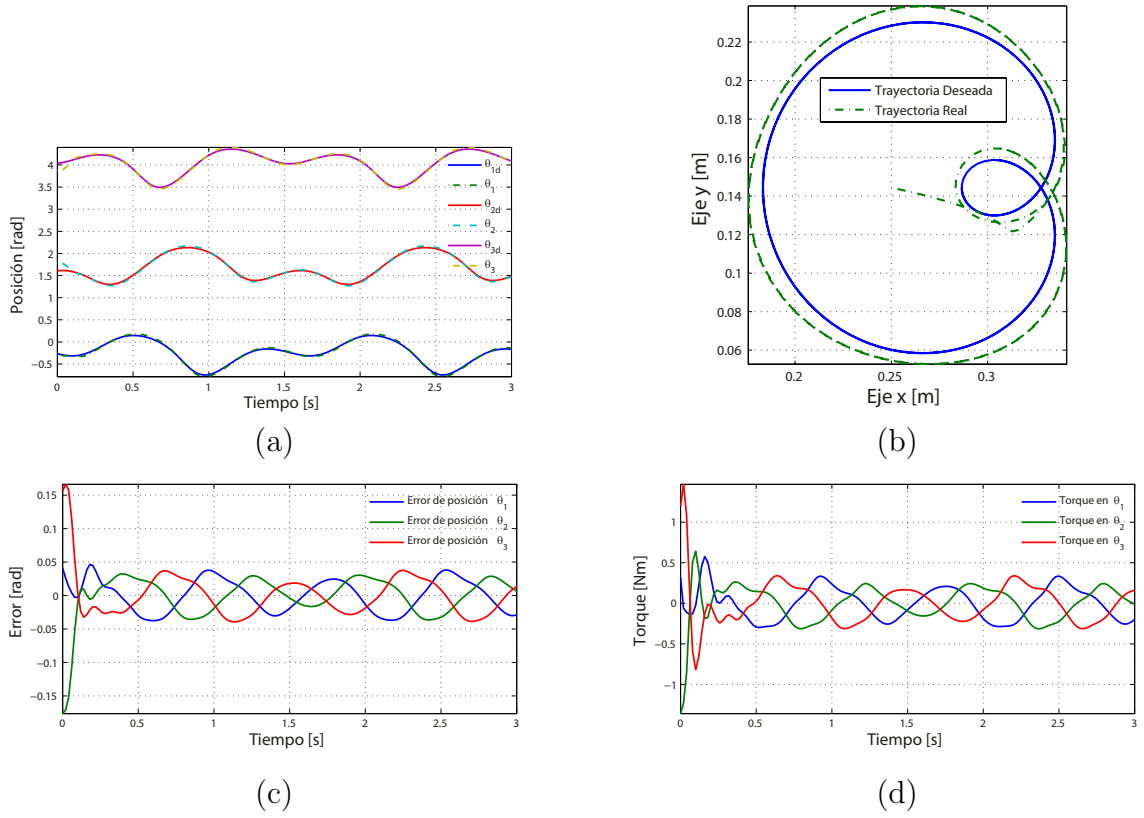


Figura 3.9: Gráficas de la mejor solución encontrada para la Ec. (3.7) mostrada en la Tabla 3.5 referentes a la trayectoria del caracol de Pascal Ec. (3.4). (a) Posición deseada vs posición real de los eslabones actuados. (b) Trayectoria deseada vs trayectoria desarrollada por el efector final. (c) Error en la posición de los eslabones actuados. (d) Torque desarrollado por los eslabones actuados.

3.3.3. Función objetivo 3: Minimización consumo de energía

De manera similar al caso anterior, si establecemos que en la Ec. (3.6), $\omega_1 = 0$ y $\omega_2 = 1$, desaparece la función encargada de sumar el error y solo queda la función encargada de sumar el torque implementado durante el seguimiento de la trayectoria. Al minimizar esta función, obtenemos un bajo consumo de energía en los actuadores, sin importarnos que tanto error se pueda generar durante el desarrollo de una trayectoria. Por lo que la función objetivo queda expresada como se muestra en la Ec. (3.8).

$$\min_{\alpha} \mathcal{F}(\alpha) = \sum_{j=1}^d \int_{t=t_0}^{t_n} \|\Delta\tau^{(j)}(t)\| \quad (3.8)$$

sujeta a

$$\begin{aligned} -\frac{\pi}{3} &\leq \theta_1(x, \alpha) \leq \frac{\pi}{3} \\ \frac{\pi}{3} &\leq \theta_1(x, \alpha) \leq \pi \\ \pi &\leq \theta_1(x, \alpha) \leq \frac{5\pi}{3} \\ K_p, K_i, K_d &\in [0, 11] \end{aligned}$$

3.3.3.1. Resultados

En la Tabla 3.6 se muestran los resultados obtenidos de la evaluación de la función objetivo (3.8) durante un tiempo de 3 seg. con el algoritmo BUMDA, dichos resultados se pueden observar gráficamente en las Figs. 3.10, 3.11 y 3.12.

Tabla 3.6: Mejores soluciones a la Ec. (3.8), Ganancias del controlador PID $\{K_p, K_i, K_d\}$ y el valor de la función objetivo \mathcal{F}_2 y Ec. (3.7)(\mathcal{F}_1)

Trayectoria	K_p	K_i	K_d	\mathcal{F}_1	\mathcal{F}_2
Mejores soluciones dadas por BUMDA					
Circulo	2.2778	0.2001	0.1444	0.4275	1.9723
Rosa de 3 pétalos	1.1259	0.3157	0.0582	0.5779	0.8223
Caracol de Pascal	4.0419	0.6860	0.0783	0.5578	2.7456

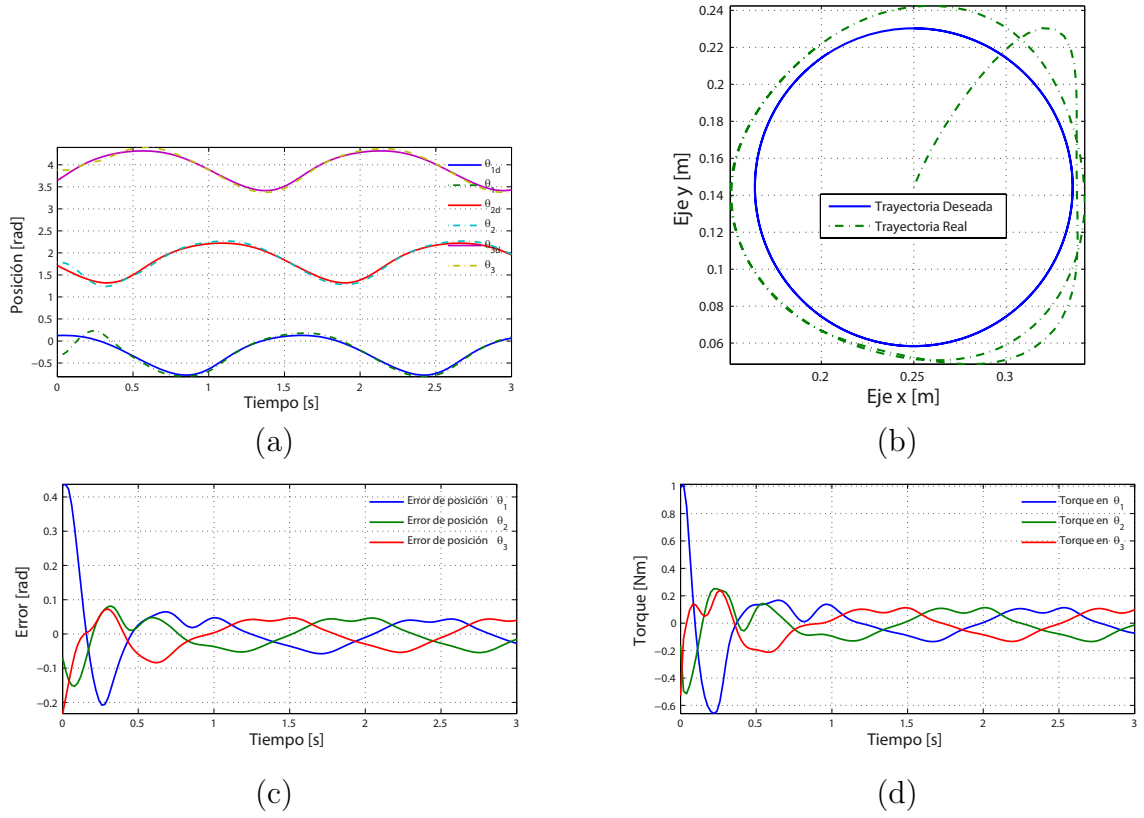


Figura 3.10: Gráficas de la mejor solución encontrada para la Ec. (3.8) mostrada en la Tabla 3.6 referentes a la trayectoria del círculo Ec. (3.2). (a) Posición deseada vs posición real de los eslabones actuados. (b) Trayectoria deseada vs trayectoria desarrollada por el efector final. (c) Error en la posición de los eslabones actuados. (d) Torque desarrollado por los eslabones actuados

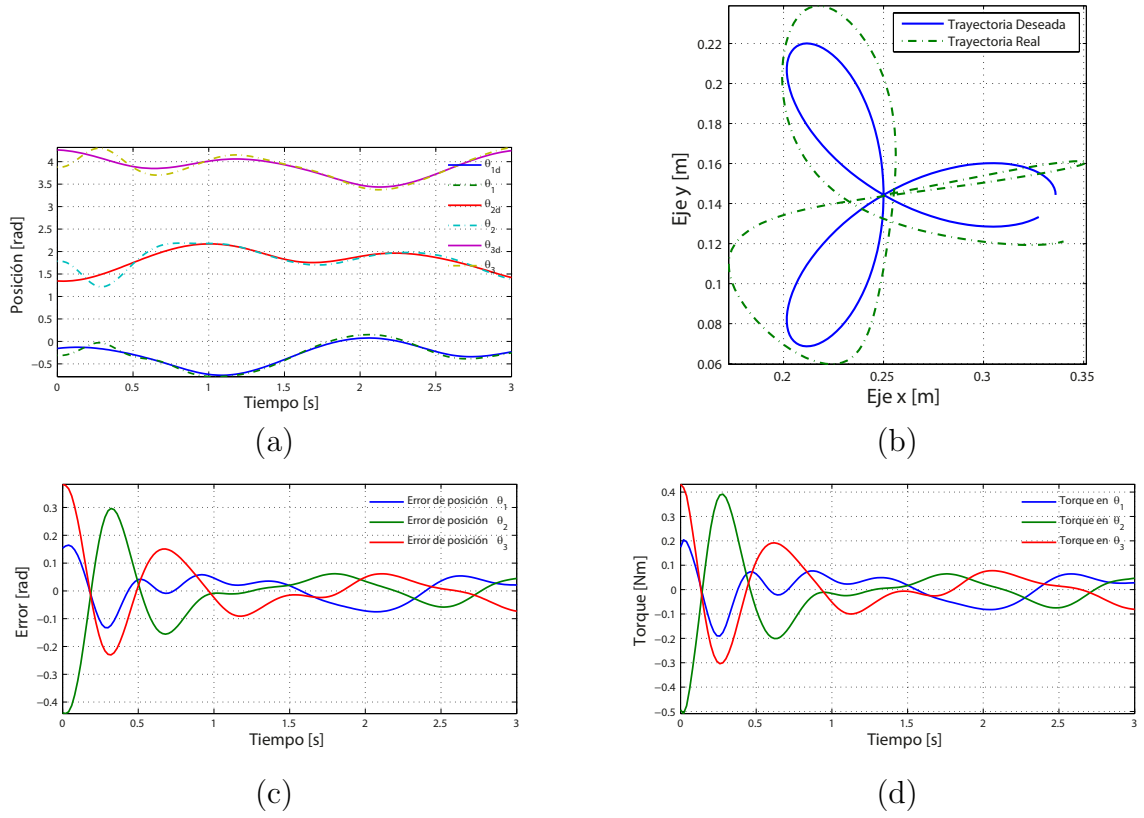


Figura 3.11: Gráficas de la mejor solución encontrada para la Ec. (3.8) mostrada en la Tabla 3.6 referentes a la trayectoria de la rosa de 3 pétalos Ec. (3.3). (a) Posición deseada vs posición real de los eslabones actuados. (b) Trayectoria deseada vs trayectoria desarrollada por el efector final. (c) Error en la posición de los eslabones actuados. (d) Torque desarrollado por los eslabones actuados.

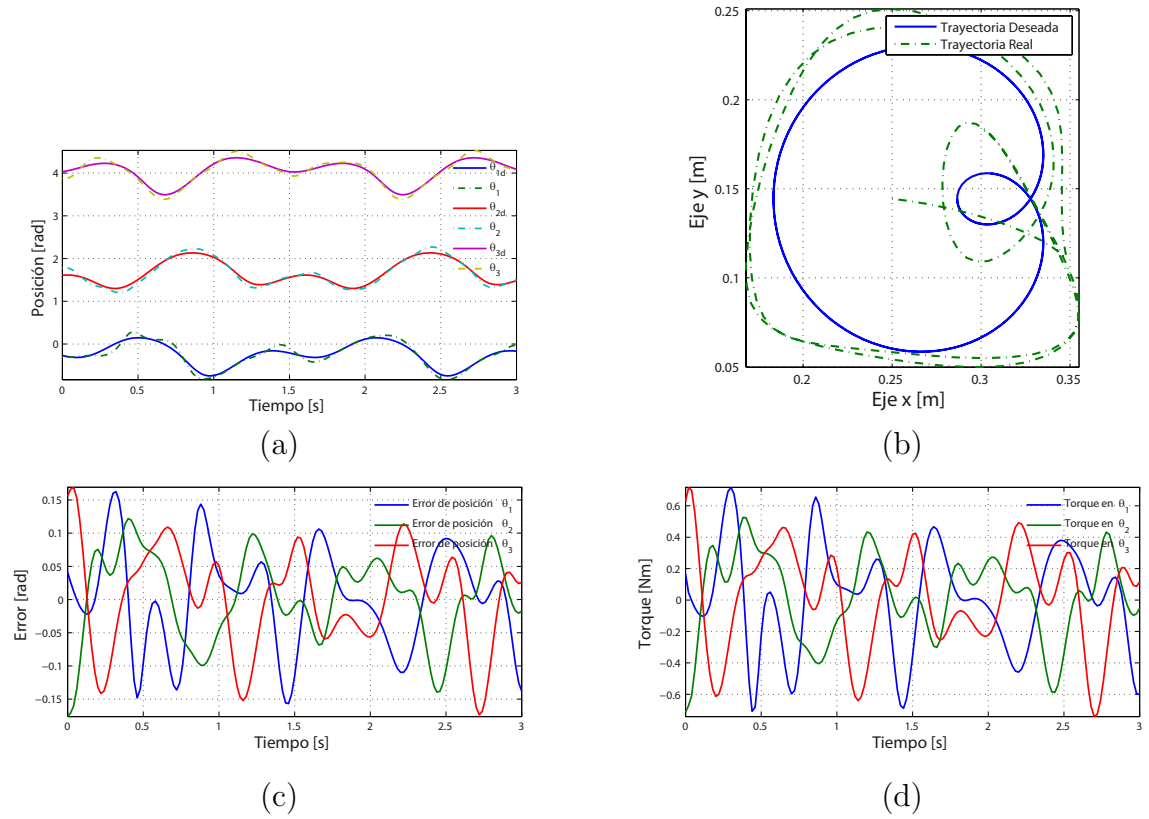


Figura 3.12: Gráficas de la mejor solución encontrada para la Ec. (3.8) mostrada en la Tabla 3.6 referentes a la trayectoria del caracol de Pascal Ec. (3.4). (a) Posición deseada vs posición real de los eslabones actuados. (b) Trayectoria deseada vs trayectoria desarrollada por el efector final. (c) Error en la posición de los eslabones actuados. (d) Torque desarrollado por los eslabones actuados.

Capítulo 4

Conclusiones

En este reporte se presentó una metodología para la implementación de simulaciones dinámicas de mecanismos virtuales, haciendo uso del programa SimWise 4D acoplado con Matlab y Simulink, evitando así el cálculo del modelo dinámico de manera analítica.

Otro punto que se mostró en este trabajo, fue la homogenización de la matriz Jacobiana del robot paralelo planar 3RRR, usando el método mostrado en [10], para la evaluación del número de condición de dicha matriz.

Por último, se presentó la optimización del robot paralelo planar 3RRR en tres secciones principales, la primera sección consistió en la maximización del espacio de trabajo utilizando tres algoritmos evolutivos distintos para la evaluación de la función objetivo, en la segunda sección se desarrolló la optimización de parámetros geométricos maximizando la destreza del robot al desarrollar ciertas trayectorias propuestas y por último, se abordó el problema de optimización de parámetros de control dinámico tratando de minimizar la energía utilizada, el error entre la trayectoria deseada y la real, o ambas, según la función objetivo propuesta, usando el programa Simwise 4D para evaluar cada individuo solución. En estas dos últimas fases, se implementó únicamente el algoritmo BUMDA para el proceso de optimización.

Nótese que estos resultados de control dinámico pueden mejorarse al realizar una buena planeación de la trayectoria que debe seguir el robot, implementando perfiles de velocidad y aceleración en dicho seguimiento.

Bibliografía

- [1] S. Botello-Aceves. Concurrent design optimization of kinematically complex mechanisms. Master's thesis, Center for Research in Mathematics, CIMAT, 2016.
- [2] C. Gosselin. Dexterity indices for planar and spatial robotic manipulators. In *IEEE Int. Conf. Robotics and Automation*, pages 650–655, Cincinnati, OH, 1990.
- [3] Nikolaus Hansen, André SP Niederberger, Lino Guzzella, and Petros Koumoutsakos. A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. *IEEE Transactions on Evolutionary Computation*, 13(1):180–197, 2009.
- [4] Mir Amin Hosseini, Hamid-Reza M. Daniali, and Hamid D. Taghirad. Dexterous workspace optimization of a tricept parallel manipulator. *Advanced Robotics*, 25:13–14, 2011.
- [5] S.-G. Kim and J. Ryu. New dimensionally homogeneous jacobian matrix formulation by three end-effector points for optimal design of parallel manipulators. *IEEE Transactions on Robotics and Automation*, 19:731–737, 2003.
- [6] H. Lipkin and J. Duffy. "hybrid twist and wrench control for a robotic manipulator". *ASME J. Mech. Trans. Automat. Des.*, 110:138–144, June 1988.
- [7] Nidamarthi Srinivas and Kalyanmoy Deb. Multiobjective optimization using non-dominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1994.
- [8] S Ivvan Valdez, Arturo Hernández, and Salvador Botello. A boltzmann based estimation of distribution algorithm. *Information Sciences*, 236:126–137, 2013.
- [9] K. Waldron and K. Hunt. "series-parallel dualities in actively coordinated mechanisms". *Proceedings of the 4th International Symposium on Robotic Research*, pages 175–181, 1988.
- [10] Y.J.Lou, G.F.Liu, and Z.X.Li. A general approach for optimal design of parallel manipulators. *IEEE Automation science and Engineering*, X, 2005.