8º Congreso Nacional de Mecatrónica
Noviembre 26 y 27, 2009. Veracruz, Veracruz.

Plataforma Gráfica de Simulación Basada en un Pendubot

Enrique López-Pérez, Pablo Sánchez-Sánchez, Fernando Reyes-Cortés, Antonio Michua-Camarillo, Benjamín Calderón-Flores, Jaime Cid-Monjaraz, Juan Carlos Torres-Monsivais y José Arturo Mancilla-Morales
Universidad Autónoma de Puebla, F. C. E., Grupo de Robótica *oocelo*
lepable@ece.buap.mx y freyes@ece.buap.mx

Resumen **- El principal objetivo de este trabajo es presentar una plataforma gráfica de simulación evaluando una estructura de control no lineal utilizando un el índice de desempeño, adicionalmente, el simulador tiene la capacidad de describir el comportamiento del sistema de acuerdo a una estructura de control, y se muestran los resultados en forma gráfica.**

I. INTRODUCCIÓN

Básicamente los robots industriales son dispositivos de posicionamiento y se utilizan en varios procesos. Para controlarlos se requiere del conocimiento de un modelo matemático que nos describa el comportamiento del robot. El modelo matemático se obtiene mediante la aplicación de leyes físicas que gobiernan la dinámica del robot [1]. Los robots ofrecen retos interesantes en teoría y práctica ya que propicia el desarrollo de estructuras de control no lineal debido a la naturaleza propia del robot. Desde un punto de vista práctico, la implementación de los algoritmos de control en un robot viene limitado a la existencia física del mismo, ya que éstos son costosos y la sintonización de las ganancias es un peligroso experimento si no se tiene mucha experiencia, por lo que el riesgo para el robot aumenta entre cada sintonización de ganancias, de ahí que es importante considerar al simulador como una herramienta que evita daños por sintonización y no es tan costoso como un sistema real.

Este trabajo se enfoca a la elaboración de una plataforma gráfica de simulación, en la cual se implementa una estructura de control no lineal, para evaluar este algoritmo de control obtenemos el índice de desempeño al aplicar la norma L^2.

El tipo de control que se utiliza para originar el movimiento del sistema es el bien conocido *control de posición*, cuyo objetivo es tomar el efector final del robot desde una posición inicial q_0, hasta una posición deseada q_d [2].

Una plataforma gráfica de simulación, también llamada simulador reproduce el comportamiento del sistema ya que al implementar el simulador se utilizan ecuaciones matemáticas que describen propiamente al sistema, además se encuentra el hecho de que el simulador crea una atmósfera gráfica para una fácil interpretación de los datos que se muestran al usuario final. Por esta razón los simuladores son herramientas que se utilizan en el diseño y desarrollo de prototipos. Este trabajo se encuentra organizado como sigue: en la sección 2 se describe la dinámica que rige al robot y su principal propiedad, después se describe el algoritmo de control, se analiza su estabilidad mediante la teoría de Lyapunov, se compara al simulador con otro simulador comercial como lo es SIMNON© para verificar su funcionalidad, y finalmente en la sección 3 se concluye el trabajo.

II. PLATAFORMA GRÁFICA DE SIMULACIÓN

En general, un simulador es una herramienta que utiliza un conjunto de elementos para recrear el comportamiento de un sistema en un ambiente de 3D.

Estos componentes son:

- El modelo dinámico. Que describe el comportamiento del sistema ante un estimulo.
- La estructura de control. La cual hace que el objetivo del control de posición se cumpla y el efector final del robot llegue a la posición deseada.
- La representación 3D. la cual nos brinda una perspectiva de profundidad y crea el ambiente gráfico.
- La programación. Se utiliza C++ y relaciona todos los elementos anteriores para formar un todo.

Un simulador se diferencia de una animación en el hecho de que en un simulador se utiliza toda la dinámica de un sistema y una estructura de control para posicionar los eslabones del robot, mientras que en una animación simplemente es una sucesión de imágenes para dar la impresión de movimiento.

II-A. Dinámica del robot

Para diseñar controladores, es necesario conocer su dinámica que lo gobierna, esto lo obtenemos mediante el modelo dinámico del robot utilizando leyes físicas. Para obtenerlo utilizamos las ecuaciones de movimiento de Lagrange [3, 4, 5, 6]. Por lo que las ecuaciones de movimiento para un sistema conservativo están dadas como sigue:

$$\frac{d}{dt}\left[\frac{\partial \mathcal{L}(q,\dot{q})}{\partial \dot{q}}\right] - \frac{\partial \mathcal{L}(q,\dot{q})}{\partial q} = \tau - f(\dot{q},\tau) \qquad (1)$$

donde $q, \dot{q} \in R^{n\times 1}$ y son vectores de desplazamiento y velocidad respectivamente, $f(\tau,\dot{q}) \in R^{n\times 1}$ es el vector de fricción y el Lagrangiano $L(q,\dot{q})$ es la diferencia entre la energía cinética y la energía potencial [3,4,5,6]

$$\mathcal{L}(q,\dot{q}) = \mathcal{K}(q,\dot{q}) - \mathcal{U}(q) \qquad (2)$$

Como es bien conocido en ausencia de fricción y otras perturbaciones, la dinámica de un robot de n eslabones es:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau \qquad (3)$$

donde $q, \dot{q}, \ddot{q} \in R^{n\times 1}$ son los vectores de los desplazamientos,

velocidad y aceleración, respectivamente; $M(q)\in R^{n\times n}$ es la matriz definida positiva de masas e inercias; $C(q,\dot{q})\in R^{n\times n}$ es la matriz de Coriolis y fuerza Centrípeta y finalmente $g(q)\in R^{n\times 1}$ es el vector gravitacional. Aunque la ecuación de movimiento es compleja, tiene propiedades fundamentales que pueden ser explotadas para facilitar el diseño de controles. Por lo que se muestra a continuación las siguientes propiedades:

Propiedad 1: La matriz de inercias $M(q)$ es una matriz definida positiva $M(q)>0$, es simétrica y esta acotada por [6]:

$$\mu_1(q)I \le M(q) \le \mu_2(q)I \tag{4}$$

donde *I* es la matriz identidad.

Propiedad 2: La matriz $\frac{1}{2}\dot{q}^T[\dot{M}(q)-2C_m(q,\dot{q})]\dot{q}\equiv 0$ es la antisimétrica, tal que [6]:

$$\dot{M}(q) = C(q,\dot{q}) + C(q,\dot{q})^T \tag{5}$$

Propiedad 3: El par gravitacional $g(q)\in \square^{nx1}$ se obtiene mediante el gradiente de la energía potencial $\mathrm{U}(q)$ del sistema.

$$g(q) = \frac{\partial \mathcal{U}(q)}{\partial q} \tag{6}$$

Satisface [6]:

$$\left\| \frac{\partial g(q)}{\partial q} \right\| \le k_g \tag{7}$$

II-B. Estructura de control

En esta sección se presenta el análisis de estabilidad de la estructura de control. Típicamente proponemos controladores utilizando el método de moldeo de energía [4,5,6,7], ya que nos ayuda en el diseño, este método considera al modelo dinámico sin fricciones y otras perturbaciones [4,5,6,7]. Utilizamos el siguiente esquema de control:

$$\tau = \nabla \mathcal{U}(k_p,\tilde{q}) - f_v(k_v,\dot{q}) + g(q) + f(\tau,\dot{q}) \tag{8}$$

donde $\tilde{q}$ es el error de posición en coordenadas cartesianas; $\mathrm{U}(k_p,\tilde{q})$ es la energía potencial artificial descrita por:

$$\mathcal{U}(k_p,\tilde{q}) = \frac{f(\tilde{q})k_p f(\tilde{q})^T}{2} \tag{9}$$

Y el término $f_v(k_v,\dot{q})$ de la ecuación (8) es la acción derivativa. Utilizamos el siguiente esquema de Lyapunov:

$$V(\dot{q},\tilde{q}) = \frac{\dot{q}^T M(q)\dot{q}}{2} + \mathcal{U}(k_p,\tilde{q}) \tag{10}$$

La metodología del moldeo de energía consiste en encontrar una función $\mathrm{U}(k_p,\tilde{q})$ que cumpla las siguientes condiciones de Lyapunov:

$$\begin{array}{ll} V(0,0)=0 & \forall\ \dot{q},\tilde{q}=0 \\ V(\dot{q},\tilde{q})>0 & \forall\ \dot{q},\tilde{q}\neq 0 \end{array} \tag{11}$$

Al realizar la derivada de la función de Lyapunov, se obtiene [9]:

$$V(\dot{q},\tilde{q}) = \dot{q}^T M(q)\ddot{q} + \frac{\dot{q}^T \dot{M}(q)\dot{q}}{2} - \frac{\partial \mathcal{U}(k_p,\tilde{q})^T}{\partial \tilde{q}}\dot{q} \tag{12}$$

cumpliendo con la condición:

$$V(\dot{q},\tilde{q}) \le 0 \tag{13}$$

Por último se comprueba estabilidad asintótica global con el empleo del teorema de LaSalle

$$V(\dot{q},\tilde{q}) < 0 \tag{14}$$

II-B1. Controlador propuesto

$$\tau = K_p\psi_{\tilde{q}} - K_v\psi_{\dot{q}} + g(q) \tag{15}$$

donde $\psi_{\tilde{q}} = \frac{\sinh(2\tilde{q})\mathrm{sech}^2(\tilde{q})}{2}$ y $\psi_{\dot{q}} = \frac{\sinh(2\dot{q})\mathrm{sech}^2(\dot{q})}{2}$. Además $\tilde{q}$ es el error de posición, señala la diferencia entre la posición deseada q_d y la posición real *q*; K_p es la ganancia del control proporcional; K_v es la ganancia del control derivativa y $g(q)$ es el par gravitacional.

La ecuación en lazo cerrado del sistema es obtenido combinando la ecuación del modelo del robot ecuación (5) y el esquema de control ecuación (15):

$$\frac{d}{dt}\begin{bmatrix} \tilde{q} \\ \dot{q} \end{bmatrix}\begin{bmatrix} -\dot{q} \\ M(q)^{-1}[K_p\psi_{\tilde{q}} - K_v\psi_{\dot{q}} - C(q,\dot{q})\dot{q}] \end{bmatrix} \tag{16}$$

la cual es una ecuación diferencial y el origen en el espacio de estado es el punto de equilibrio y es único. Para el análisis de la ecuación (16) proponemos la siguiente función de Lyapunov

basándonos en la metodología de moldeo de energía. Se propone la siguiente función de Lyapunov [9,10]:

$$V(\dot{q},\tilde{q}) = \frac{\dot{q}^T M(q)\dot{q}}{2} + \begin{bmatrix} \sqrt{ln(\cosh(\tilde{q}_1))} \\ \sqrt{ln(\cosh(\tilde{q}_2))} \\ \vdots \\ \sqrt{ln(\cosh(\tilde{q}_n))} \end{bmatrix}^T K_p \begin{bmatrix} \sqrt{ln(\cosh(\tilde{q}_1))} \\ \sqrt{ln(\cosh(\tilde{q}_2))} \\ \vdots \\ \sqrt{ln(\cosh(\tilde{q}_n))} \end{bmatrix} \quad (17)$$

El primer termino de $V(\dot{q},\tilde{q})$ es una función definida positiva con respecto a $\dot{q}$ ya que $M(q)$ es una matriz definida positiva, de igual manera para el segundo termino es también definido positivo pero ahora con respecto a $\tilde{q}$ debido a que K_p es una matriz definida positiva.

De ahí que $V(\dot{q},\tilde{q})$ es una función definida positiva en forma global. La derivada de la función candidata de Lyapunov a lo largo de la trayectoria de la ecuación en lazo cerrado

$$\dot{V}(\dot{q},\tilde{q}) = \dot{q}^T M(q)\ddot{q} + \frac{\dot{q}^T \dot{M}(q)\dot{q}}{2} + \begin{bmatrix} \sqrt{ln(\cosh(\tilde{q}_1))} \\ \sqrt{ln(\cosh(\tilde{q}_2))} \\ \vdots \\ \sqrt{ln(\cosh(\tilde{q}_n))} \end{bmatrix}^T K_p + \begin{bmatrix} \frac{\tanh(\tilde{q})}{\sqrt{ln(\cosh(\tilde{q}))}} \end{bmatrix} \dot{\tilde{q}} \quad (18)$$

y después de aplicar algebra y la propiedad 2 la derivada puede ser escrita como:

$$\dot{V}(\dot{q},\tilde{q}) = -\dot{q}^T K_v \begin{bmatrix} \frac{\text{senh}(2\dot{q}_1)\text{sech}^2(\dot{q}_1)}{2} \\ \frac{\text{senh}(2\dot{q}_2)\text{sech}^2(\dot{q}_2)}{2} \\ \vdots \\ \frac{\text{senh}(2\dot{q}_n)\text{sech}^2(\dot{q}_n)}{2} \end{bmatrix} \leq 0 \quad (19)$$

La cual es una función semidefinida negativa y de ahí que concluimos estabilidad en el punto de equilibrio. Para probar estabilidad asintótica explotamos la naturaleza autónoma de la ecuación de lazo cerrado para aplicar el principio de invarianza de LaSalle:

$$\dot{V}(\dot{q},\tilde{q}) < 0 \quad (20)$$

En la region:

$$\Omega = \left\{ \begin{bmatrix} \tilde{q} \\ \dot{q} \end{bmatrix} \in R^n : V(\dot{q},\tilde{q}) = 0 \right\} \quad (21)$$

II-C. Descripción estructural del robot

Después de analizar el sistema y obtener la dinámica que lo describe, el siguiente paso consiste en definir tridimensionalmente al sistema, para lo cual se utilizan un programa CAD (como Autocad). La figura 1 describe el prototipo en 3D

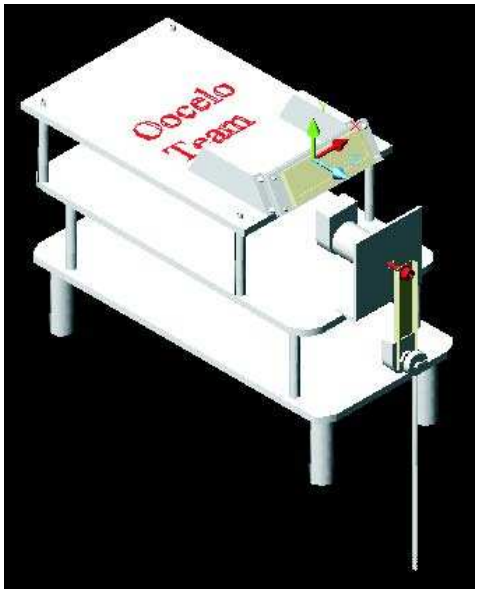

Figura 1. Bosquejo del sistema

La representación tridimensional del sistema es utilizado para la creación de una atmósfera gráfica en la cual el simulador hace de una interpretación amena de los datos mostrados en el simulador. Lo más importante cuando estamos dibujando el sistema en tercera dimensión es tener en mente cuales son las piezas que se se le darán el movimiento y ponerlas en el origen para un correcto movimiento. Una vez definidas las piezas las guardaremos en un archivo de datos para su posterior manipulación a través de rectángulos, de pixeles y puntos, en un formato en 3DS o Bitmap o PGM. Para manipular la imagen tales como la cabecera, identificador, y el tamaño del cuerpo de la imagen se encripta en un archivo de código ASCII, en este caso se transforma en un archivo 3DS.

II-D. Programación del simulador

El siguiente paso es conectar a través de un lenguaje de programación la parte matemática y los movimientos del sistema que responden a la dinámica del sistema para poder recrear el comportamiento del sistema en un ambiente controlado de 3D. Para poder recrear el movimiento del sistema utilizaremos una interfaz gráfica tal como lo es OpenGL, las librerías utilizadas incrementan el poder de las gráficas.

II-D1. OpenGL

OpenGL (Open Graphics Library) es una librería gráfica que nos permite hacer una gran variedad de cosas con gráficos. Con OpenGL podemos utilizar los resultados matemáticos obtenidos por el modelo, por el método numérico de Runge-Kuttan y por la estructura de control, y asociarlos con el modelo gráfico hecho en Autocad.

La librería OpenGL, por si sola, se encargará de reservar la memoria para los pixeles, así que esta parte no se programa, en este caso OpenGL contactará directamente con los elementos de la figura anterior a medida que sea necesario.

La librería se ejecuta a la par con el programa, independientemente de la capacidad gráfica de la computadora en la cual se ejecute la aplicación, aunque tener una tarjeta aceleradora de gráficos ayuda al apreciar el movimiento que generemos en OpenGL.

OpenGL es una especificación estándar que define una Interfaz de Programación de Aplicaciones (API por Application Programming Interface) multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos en 2D y 3D. Fue desarrollada por Silicon Graphics Inc. en 1992. Su nombre viene del inglés Open Graphics Library.

OpenGL contiene la Glu.h y la Glu32.dll, las cuales son compatibles con Borland C, Visual C++, Java, visual Fortran entre otros.

Para nuestro caso programaremos en el entorno de Visual C++, es por ello que debemos tomar en cuenta que debe estar incluida la librería glut32.dll en la carpeta de system y system32 de windows; y la glu.h en la librería visual C++, la relación de las librerías se muestra en la figura 2

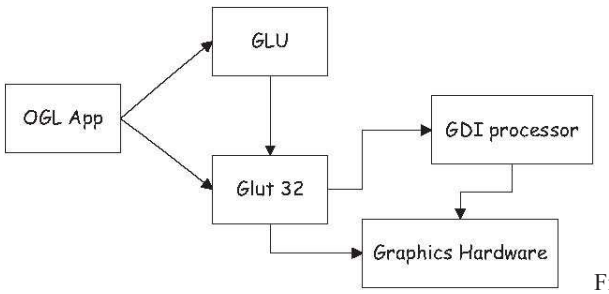

Figura 2. Relación de las librerías

II-D2. Visual C++

La programación orientada a objetos (POO) permite realizar programas mediante la unión de elementos simples, que pueden ser diseñados y comprobados de manera independiente del programa que va a usarlos. Esto es mediante clases. A estas clases, módulos o componentes, que interactúan entre sí cuando se ejecuta un programa, se les denomina objetos. Estos objetos contienen tanto datos como las funciones que actúan sobre esos datos. Cada uno de estos objetos corresponde a algún elemento que debe utilizar el programa. Como estructuras de datos: colas, pilas, etcétera. Durante la ejecución del programa, los objetos interactúan pasándose mensajes y respuestas.

Debemos tener en claro que:

- Objetos: Es una clase, que es el tipo de datos y funciones definidos por el programador
- Mensajes: Cuando un programa orientado a objetos se ejecuta, los objetos están recibiendo, y respondiendo a mensajes de otros objetos, por lo que se ejecuta alguna acción.
- Métodos: Un método se implementa en una clase y determina como tiene que actuar el objeto cuando recibe un mensaje. Es decir son las funciones.

II-D3. Cargado del sistema

Para cargar el modelo dinámico en Visual C++ es necesario crear una nueva clase, la cual es llamada modelo dinámico, en esta clase utilizamos el algoritmo Runge-Kutta de cuarto orden, después se crea otra clase para dotar al sistema de una estructura de control y finalmente una clase para programar el índice de desempeño. Haciendo esto se muestra en la figura 2 el ambiente del simulador.

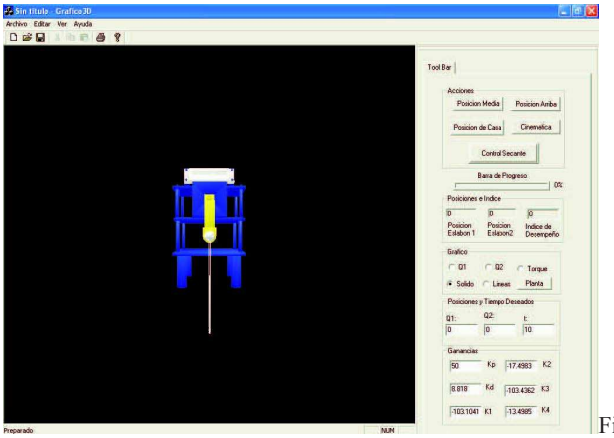

Figura 3. Simulador en la posición de casa

II-D4. Índice de desempeño

Para comparar el rendimiento de los controladores se utiliza la norma L^2 la cual es el promedio de todos los errores de posición

$\tilde{q}\ \tilde{q}$. Un valor pequeño de L^2 representa un error pequeño de posición y por lo tanto indica un mejor desempeño.

Una función vectorial $R^n \rightarrow R^n \in L^2$, si al evaluar:

$$\mathcal{L}^2 = \sqrt{\int_0^\infty \| f(t) \|^2 \, dt} < 0 \qquad (22)$$

es un escalar y donde $\| f(t) \|$ es la norma euclidiana de la función sobre el intervalo. Esta propiedad en funciones vectoriales es una medida para determinar la convergencia conforme el tiempo evolucione. Debido a que el tiempo de simulación es finito, debemos aplicar el concepto de valor efectivo para calcular la desviación de la función entre los intervalos de simulación, por lo cual definimos la norma L^2 como:

$$\mathcal{L}^2 = \sqrt{\frac{1}{T}\int_0^T \| \tilde{x} \|^2 \, dt} < \infty \qquad (23)$$

II-D5. Experimentos con el simulador

Ahora bien, para comprobar el funcionamiento del simulador lo comparamos con un simulador comercial como lo es SIMNON©. Para la estructura de control lineal las posiciones se manejan de la siguiente manera:

- Para el primer punto de equilibrio en donde las posiciones son para $q_1 = \pi$ y $q_1 = 0$, esto quiere decir que el eslabón 1 del sistema dará una vuelta de 360^o y el eslabón 2 quedará en 0^o con respecto al eslabón 1, por lo que el comportamiento en el eslabón 1 está descrito en las siguientes gráficas tomadas del simulador y de SIMNON©

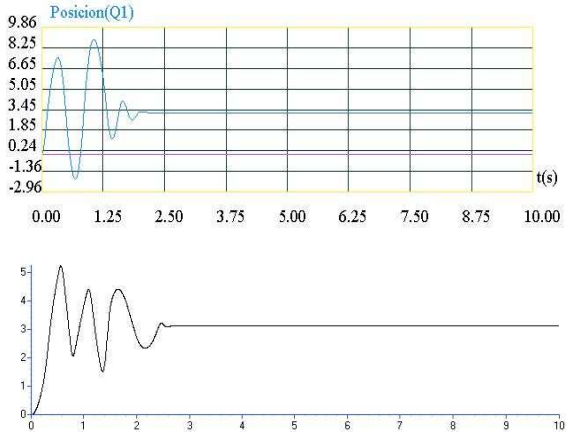

Figura 4. (A) simulador y (B) de SIMNON©

Como se puede observar en las gráficas de la figura 4, tanto el simulador con SIMNON© los eslabones tienen casi el mismo comportamiento y llegan a las posiciones deseadas y se establecen en el mismo tiempo. Para el segundo eslabón tenemos el siguiente comportamiento descrito en las gráficas de la figura 5:

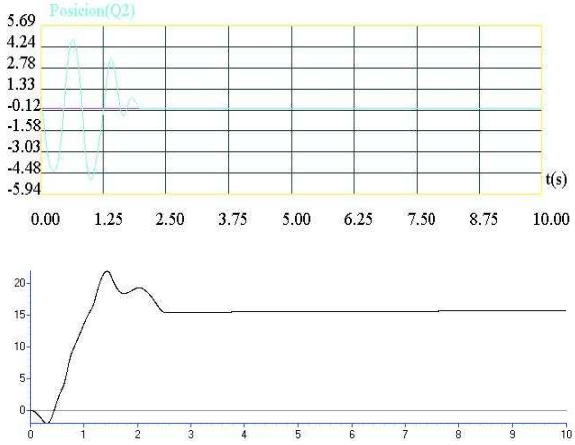

Figura 5. (A) simulador y (B) de SIMNON©

En la gráfica (B) de la figura 5, pareciera que no es igual a la del inciso (A), pero hay que tener en cuenta que la gráfica mostrada en (A), los datos son dobles y en la gráfica (B) son flotantes, por lo que el simulador ``dobla" a la gráfica pareciendo como si fuese otro comportamiento, pero es el mismo comportamiento en el simulador y en SIMNON©.

El índice de desempeño arrojado por el simulador es presentado en la figura 6

Figura 6. Índice de desempeño para el primer punto de equilibrio

El índice de desempeño marcado en la figura 6 es muy pequeño dada la posición que toma el sistema.

- Para el segundo punto de equilibrio las posiciones $q_1 = 2 * Pi$ y $q_2 = Pi$, dadas para el simulador y

SIMNON©. La posición articular q_1 tiene el siguiente comportamiento:

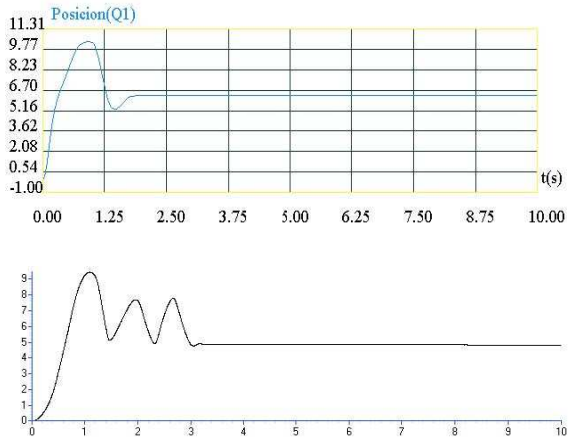

Figura 7. (A) simulador y (B) de SIMNON©

Observamos que para llegar a la posición de equilibrio el simulador da vuelta y media menos de lo que marca SIMNON©, esto supone que para llegar a la posición deseada al simulador le toma menos tiempo en establecerse en la posición de equilibrio.

Para la posición q_2, tenemos el siguiente comportamiento:

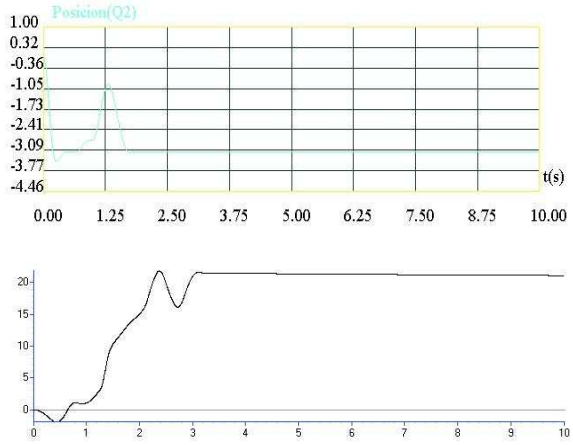

Figura 8. (A) simulador y (B) de SIMNON©

De igual manera como en el caso de q_2 para las posiciones $q_1 = Pi$ y $q_2 = 0$, tenemos el mismo efecto de los datos que grafican las posiciones, solamente cabe destacar que el simulador presenta menos oscilaciones que la simulación arrojada por SIMNON©. Finalmente el índice de desempeño arrojado por el simulador para el segundo eslabón se muestra en la figura 9

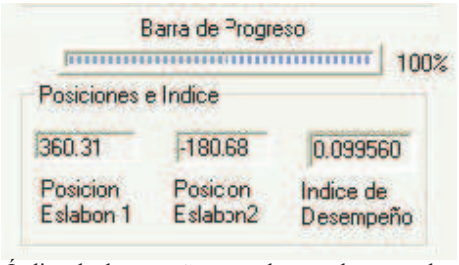

Figura 9. Índice de desempeño para el segundo punto de equilibrio

El índice de desempeño marcado en la figura 9 es un poco más grande que el índice de desempeño mostrado para el primer punto de equilibrio.

III. CONCLUSIONES

Hemos visto el comportamiento de un sistema de dos grados de libertad sub actuado en dos tipos de simuladores. La diferencia radica principalmente que no podemos observar el comportamiento del sistema de manera que veamos el movimiento de los eslabones de la planta en SIMNON© debido a que SIMNON© solamente nos proporciona las gráficas de las variables o posiciones de las articulaciones, mientras que en el simulador vemos el movimiento de los eslabones y podemos observar cómo evolucionan las posiciones articulares y el torque aplicado al sistema. Así mismo en el simulador podemos ver el índice de desempeño del controlador, su evolución en el tiempo y el valor final después de la simulación para los dos puntos de equilibrio.

REFERENCIAS

[1] J. J. Craig, Introduction to Robotics Mechanics and Control, (New York: Addison-Wesley, 1989).

[2] F. Reyes, J Cid and C. Campuzano, Development of an Experimental Plataform with open architecture for Robots Manipulators, Universidad Autónoma de Puebla, Puebla Mexico.

[3] A. Lona, On Output Feedback Control of Euler-Lagrange, (1996).

[4] H. Goldstein, Classical Dynamics, (Reading, MA.: Addison-Wesley, 1950).

[5] A. Barrientos, L. F. Peñin, C. Balaguer and R. Aracil, Fundamentos de Robótica, (Madrid, Espa~na: McGraw Hill, 1997).

[6] L. Sciavicco and B. Siciliano, Modeling and Control of Robot Manipulators, (Napoles, Italia: McGraw Hill, 1996).

[7] M. Takegaki and S. Arimoto, A NewFeedback Method for Dynamic Control of Manipulators, Journal of Dynamic System, Measurement and Control, 102(2), 1981, 119-125.

[8] R. Kelly and V. Santivañez, A class of global regulators with bounded control actions for robot manipulators, Proceedings of the 35th conference on decision and control, Kobe, Japan, 1996, 3382-3387.

[9] R. Kelly, R. Haber, R. Haber-Guerra and F. Reyes, Lyapunov Stable Control of Robot Manipulators: A Fuzzy Self-Tunning Procedure, Intelligent Automation and Soft Computing, 5(4), 1999, 313-326.

[10] F. Reyes and C. Campuzano, PD-Type Controller with Nonlinear Proportional Gain for Robot Manipulators, XX Congreso Internacional Académico de Ingeniería Electrónica, Puebla, México, 1998, 357-360.