ISSN 1870-4069

Sintonización óptima de un controlador PD utilizando evolución diferencial con un modelo dinámico virtualizado en Unity 3D

Rosaura A. Suárez-Santillán, Gabriel Sepúlveda-Cervantes, Edgar A. Portilla-Flores, Maria Bárbara Calva-Yáñez, Cuauhtémoc Morales-Cruz, Noemi Hernández-Oliva

Instituto Politécnico Nacional, Centro de Innovación y Desarrollo Tecnológico en Cómputo, Grupo de Investigación e Innovación en Mecatrónica (GIIM), Ciudad de México, México
{rsantillans1800}@alumno.ipn.mx,
{gsepulvedac,aportilla,nhernandezo}@ipn.mx,
josechh@yahoo.com, {cumoralescruz}@gmail.com,{b_calva}@hotmail.com

Resumen. La creciente demanda de desarrollo científico tecnológico ha permitido la conjunción de diversos campos de la ciencia con el fin de resolver problematicas actuales .Permitiendo de esta manera cruzar los límites tradicionales entre varias disciplinas académicas para el surgimiento de nuevos conocimientos con base en las necesidades que anteriormente eran nulas. La creciente demanda de virtualizar los sistemas presentes en el mundo real ha obligado a desarrollar diversas herramientas que permiten recrear el mundo físico en una computadora. Actualmente, los sistemas físicos tienden a ser más complejos implicando que el modelo matemático tambien es complejo. En este trabajo se presenta la solución de un problema de optimización dinámico haciendo uso del motor de física NVIDIA PhysX para realidad virtual con el fin de reproducir el comportamiento de un sistema no lineal como caso de estudio demostrando la manipulación y recreación del comportamiento del sistema sin la necesidad de tener el modelo dinámico del mismo.

Palabras clave: sintonización óptima, modelado de sistemas, virtualización de sistemas, optimización, evolución diferencial, control PD.

Optimal Tuning of a PID Controller Using Differential Evolution with a Virtualized Dynamic Model in Unity 3D

Abstract. The growing demand for scientific and technological development has allowed the combination of diverse fields of science in order to solve current problems. This way, it is possible to cross the traditional limits between several academic disciplines for the emergence of new knowledge based on the needs that previously they were nil. The growing

demand to virtualize present systems in the real world has forced to develop several tools that allow recreating the physical world in a computer. Currently, physical systems tend to be more complex implying that the mathematical model is also complicated. In this paper, we present the solution of a dynamic optimization problem using the physics engine NVIDIA PhysX for virtual reality in order to reproduce the behavior of a nonlinear system as a case study demonstrating the manipulation and recreation of the behavior of the system without the need to have the dynamic model of it.

Keywords: differential evolution, Unity 3D, optimization, PD control.

1. Introducción

En la actualidad el uso de nuevas herramientas tecnológicas de virtualización para el manejo de sistemas ha permitido el avance y crecimiento en diferentes áreas del conocimiento. El uso de simuladores ha facilitado el desarrollo de nuevos productos a un menor costo, reduciendo así su tiempo de producción. Permitiendo que el desarrollo de los productos tenga un mejor enfoque antes de ser elaborado y su vida útil se prolongue. La tecnología computacional ha crecido a pasos agigantados brindado diversos enfoques para un mismo objetivo. La Realidad Virtual representa escenas o imágenes de objetos producidas por un sistema informático que permite crear la sensación de su existencia real para un usuario. Para lograr una buena aproximación de los fenómenos físicos que se representan se han desarrollado múltiples y muy diversos motores de física para calcular el comportamiento de los sistemas de interés [2,3,18].

La creación del modelo de manera virtual permite al usuario tener una mejor visualización del comportamiento del sistema sin tener el desgaste del mismo. Esta es una ventaja que brinda virtualizar el sistema. Unity 3D es una herramienta de desarrollo de juegos en 3D integral multiplataforma desarrollada por Unity Technologies [5]. Unity3D es más prominente en las capacidades de representación tridimensional, el detalle y el rendimiento sólido, admite 3DMAX, MAYA, Sketcheup, CAD y otros modelos tridimensionales [17]. La tecnología de recorrido virtual es una herramienta que se utiliza en el diseño y la presentación de arquitecturas. Integra gráficos de computadora, multimedia, inteligencia artificial, tecnología de sensores múltiples, redes, etc., nos brinda un sólido soporte para crear un mundo virtual [8]. Así mismo, al utilizar la computadora para generar un entorno de simulación, se puede recrear el mundo físico de una manera sencilla y darse cuenta de la interacción natural que se presenta con el entorno sin dañar el mismo. El modelar un sistema comienza por identificar sus elementos y las relaciones entre ellos. El modelar es multifacético o por perspectiva porque el modelo que se construya de un sistema real depende del objetivo del usuario que lo modela [19].

Un sistema físico debe ser representado en un modelo matemático para poder simular su comportamiento. Sin embargo, entre más complejo sea el sistema el modelo matemático también incrementa su complejidad. Es por ello que diversos

sistemas existentes en el mundo físico aun no tienen un modelo que lo describa. El proceso de modelado analítico se divide en dos grandes etapas importantes para cualquier sistema. La primera es delimitar el modelo en función de los fenómenos físicos relevantes a la problemática a resolver. En esta etapa, la sistematización del sistema no es fácilmente aplicable y requiere de una vasta experiencia en relación con el sistema a modelar. La siguiente etapa consiste en formalizar las relaciones constitutivas y estructurales asociadas respectivamente a los fenómenos considerados y a la forma en que estos se disponen dentro del sistema. En los sistemas físicos, estas relaciones constitutivas y estructurales encuentran su expresión matemática en las leyes fundamentales de los dominios de la física asociados a los fenómenos mencionados. Los modelos matemáticos describen las relaciones existentes entre las magnitudes caracterizantes del sistema, vinculando de esta manera variables matemáticas representativas de las señales en el sistema [6,7,9].

De manera general todos los trabajos necesitan la planta o un modelo para realizar las pruebas necesarias de las teorías que se estén desarrollando. En el caso de sistemas de control es una práctica habitual modelar el sistema de forma matemática y realizar el análisis a lazo abierto y a lazo cerrado en conjunto con el controlador que se esta evaluando [10,11,13].

Actualmente se utilizan herramientas computacionales que permiten utilizar el sistema físico en un ambiente virtual sin comprometer el desempeño ni la vida útil del mismo. El ambiente virtual NVIDIA PhysX es una tecnología especialmente diseñada para su aceleración por hardware a través de procesadores de alta capacidad dotados de cientos de núcleos. Los cuales proporcionan un crecimiento exponencial de la potencia de cálculo de la física que da lugar a entornos de juego increíblemente ricos y realistas gracias a mejoras como personajes de geometrías complejas y perfectamente articuladas para proporcionar interacciones y movimientos más naturales. Una de las herramientas utilizadas es Unity ya que tiene el mejor motor de física, lo que implica que al ser aplicada a sistemas permite un realismo impresionante tanto en la visualización como en el comportamiento del mismo.

De esta manera, Unity brinda la ventaja más importante del desarrollo: no se requiere tener el modelo matemático del sistema para poder recrearlo, incluso algún modelo complejo que aún no tenga su representación matemática puede ser implementado en esta herramienta computacional. Con ello permite configurar las dimensiones de los elementos del sistema de manera precisa y fácil, y reconfigurar el sistema cuando se desee. Modificar no solo sus dimensiones sino los parámetros que puede tener cualquier sistema físico. Unity brinda al usuario la facilidad de realizar cálculos de colisiones, fuerzas ejercidas, torques y fricciones del sistema. Al trabajar con este tipo de herramientras se brinda libertad para trabajar en el control de la posición y velocidad, ya que sus características son ajenas a cualquier ley de control aplicada al sistema.

Otra de las mayores ventajas que presenta el uso de herramientas como NVIDIA PhysX es el tener instancias múltiples del sistema con el que se está trabajando. Es decir, se emula cierta cantidad de sistemas a la par con las mismas

características pero aplicandole diferente control.

Por otro lado, la optimización es un proceso que pretende encontrar una solución o conjunto de soluciones que maximice o minimice lo que se conoce como función objetivo, es decir, una o varias funciones que en su conjunto permitan obtener el resultado deseado. En el área de optimización en ingeniería, ésta se define como el acto de obtener los mejores resultados bajo ciertas circunstancias [12]. Algunos otros autores definen la optimización como el proceso de buscar la mejor solución posible a un problema, bajo ciertas circunstancias [16]. Los métodos de optimización son conocidos como técnicas de programación matemática, las cuales son aplicables a problemas de toma de decisiones, así como al establecimiento de las mejores soluciones posibles.

En este trabajo se presenta la solución de un problema de optimización dinámico haciendo uso de motor de física para realidad virtual con el fin de reproducir el comportamiento del sistema de estudio sin la necesidad de tener el modelo dinámico del sistema. El problema de optimización planteado representa el problema de sintonización óptima de un controlador PD. La solución se obtiene al aplicar una técnica metaheurśtica para obtener las ganancias apropiadas para conseguir que el controlador lleve al sistema a la posición deseada. Adicionalmente, se presenta el modelo matemático del sistema y la solución del mismo problema haciendo uso del modelo. La información mostrada en este artículo se estructura de la siguiente forma: en la sección II se presenta el modelo dinámico del sistema no lineal del péndulo simple con el cual se va a trabajar, en la sección III se presenta la virtualización del sistema en el entorno de Unity3D, la sección IV muestra el problema de sintonización como un problema de optimización, la sección V expone el algoritmo implementado y finalmente en la sección V y VI se presentan los resultados y las conclusiones.

2. Modelo dinámico del péndulo simple

Sea el sistema mecánico denominado péndulo simple como se muestra en la Fig. 1, donde m denota el elemento mecánico situado en el extremo de una barra de masa despreciable con una longitud l. Por otro lado τ define el par aplicado y el coeficiente de fricción viscosa es b, el cual se debe a la fricción en los sistemas mecánicos de rodamiento.

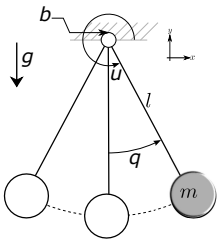


Fig. 1. Péndulo simple.

De acuerdo con el Método de Euler-Lagrange, las ecuaciones dinámicas para un sistema de n coordenadas generalizadas (q), están dadas por la ecuación (1).

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) + \frac{\partial D}{\partial \dot{q}} - \frac{\partial L}{\partial q} = \tau, \tag{1}$$

donde:
L = Lagrangiano del sistema,
τ = par,
D = función de disipación de Rayleigh para sistemas amortiguados.

El Lagrangiano del sistema y la función de disipación de Rayleigh, están dados por la ecuación (3):

$$L = K - U, \tag{2}$$

$$D = \frac{1}{2} b\dot{q}^2, \tag{3}$$

donde:
$U = -mglcos(q)$, energía potencial,
$K = \frac{1}{2} m\dot{q}^2 l^2$, energía cinética.

Debido a que el movimiento de la masa está limitada a permanecer en un círculo de radio igual a la longitud de la barra. El ángulo θ se considera como la variable de salida, y se designa a q como la coordenada generalizada, por lo tanto $q = \theta$, la cual indica la posicíon de la masa del péndulo. Por lo que al sustituir U y K, en el Lagrangiano se obtiene la ecuación de movimiento que describe al sistema del péndulo simple, la cual se presenta en la ecuación (4):

$$ml^2\ddot{q} + b\dot{q} + mglsin(q) = \tau. \tag{4}$$

A partir de la ecuación de movimiento descrita en la ecuación (4), el vector de variables de estado es el siguiente $x = [x_1, x_2]^T = [q, \dot{q}]^T$ y la entrada del sistema es τ, la dinámica en el espacio de estados se presenta en la ecuación (5):

$$\begin{aligned} \dot{x} &= f\left(\boldsymbol{x}, \tau(t), t\right), \\ \dot{x} &= \begin{bmatrix} x_2 \\ \dfrac{\tau}{ml^2} - A\dot{x_2} - Csin(x_1) \end{bmatrix}, \end{aligned} \tag{5}$$

donde:

$$A = \frac{b}{ml^2},$$
$$C = \frac{g}{l}.$$

En [15] se realizó la construcción del sistema real de péndulo simple para comparar el comportamiento del sistema virtualizado con el físico. En este sistema el único parámetro medido fué el peso del péndulo y se hizo por medio de una báscula para después dividir ese valor entre la gravedad (g) y finalmente obtener la masa del sistema. Por otro lado, el resto de los párametros del sistema no se asumen o se miden de forma directa debido a que no se consideró que la distribución de la masa fuera homogénea y la construcción del péndulo fuera completamente simétrica. Por lo tanto, se implementó un estimador de parámetros para calcular la posición del centro de masa (l) y la constante de fricción viscosa (b) como se muestra en la Fig. 2.

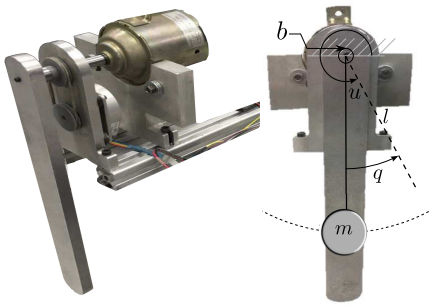


Fig. 2. Sistema: péndulo Simple.

Finalmente todos los parámetros descritos en la Fig. 2 del sistema se muestran en la Tabla 1.

Tabla 1. Parámetros.

Parámetros	Valor
l	0.2326 m
b	0.0015 Nm/s
m	0.04617 kg
g	9.81 m/s

3. Sistema del péndulo en Unity3D

El sistema de péndulo simple fue configurado en una escena de Unity3D. El primer paso fue realizar una base para sostener el péndulo la cual se diseñó de dos partes: una base con forma cuadrada y un eslabón fijo perpendicular a la base con la altura necesaria para ver el comportamiento del sistema. Se fijó el punto de unión entre la base y el péndulo. Se añadió el péndulo como se visualiza en la Fig. 2, al cual se le proporcionó las mismas caracteristicas dandole el valor de la fricción, la masa presente en el péndulo y la cinemática del sistema. Así

mismo se le agregó la opción de uso de gravedad para tener un sistema idéntico al físico pero de manera virtual, como se muestra en la Fig. 3.

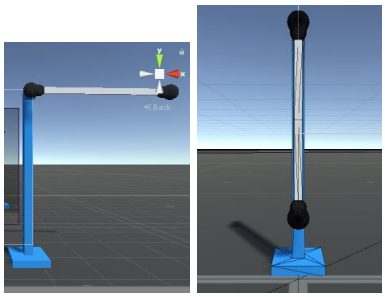


Fig. 3. Péndulo simple virtualizado.

4. Enfoque de optimización

Este enfoque permite el cálculo de cada una de las constantes del controlador para que el sistema de péndulo simple realice la acción de control de posición.

4.1. Función objetivo

En la ecuación (7) se expresa la función objetivo la cual evalua el rendimiento del sistema es la sumatoria del error entre el comportamiento deseado y la respuesta de la planta y una función de sobretiro. Esta función evalúa la integral del error cuadrático como resultado de un conjunto dado de parámetros y un tiempo de simulación de t_0 a t_f. En este caso, el vector $\boldsymbol{p}$ contiene las variables de diseño y n indica la dimensionalidad del problema.

4.2. Variables de diseño

La solución del problema de optimización debe obtener el conjunto de parámetros para satisfacer los requisitos de diseño. Por lo tanto, el vector de diseño en (6) contiene el conjunto de parámetros mencionado. Los componentes de este vector son las ganancias del controlador PD: la constante del controlador de proporcionalidad (K_p) y la constante del controlador derivativo (K_d). El problema se debe acotar con los valores límite para cada una de las variables en (9)-(10). Estas cotas representan la energia del sistema definida por el usuario:

$$\begin{aligned} &\boldsymbol{p} \in \mathbb{R}^2, \\ &\boldsymbol{p} = [K_p, K_d]^T. \end{aligned} \tag{6}$$

4.3. Problema de optimización

El problema descrito anteriormente se resume en las ecuaciones (7)-(10), el cual se trata de un problema de optimización dinámico mono-objetivo sin restricciones y las variables de diseño se acotan a valores mínimos y máximos.

$$\underset{\boldsymbol{p}\in\mathbb{R}^2}{min} \quad f_1(\boldsymbol{p}) = \int_{t_0}^{t_f} e^2 dt + F_{sobretiro}, \tag{7}$$

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}(u,\boldsymbol{p},t), u(\boldsymbol{p},t), t), \tag{8}$$

$$considerando \quad las \quad siguientes \quad cotas:$$

$$0 < K_p \leq 40, \tag{9}$$

$$0 < K_d \leq 40, \tag{10}$$

$$\tag{11}$$

$$F_{sobretiro} = \begin{cases} 0 & \text{si } sobretiro < 0 \\ sobretiro & \text{si } sobretiro > 0 \end{cases} \tag{12}$$

$$sobretiro = q_{max} - q_d(t_{qmax}), \tag{13}$$

siendo t_{qmax} el instante donde la existe el sobretiro máximo.

5. Evolución diferencial

Este estudio usa el Algoritmo de Evolución Diferencial[14] para resolver el problema de optimización planteado. Esta herramienta ha sido usada ampliamente y se ha posicionado como una buena herramienta para la resolución de problemas de ingeniería [4]. Este algoritmo basa su comportamiento en la reproducción de individuos en una población y el reemplazo de individuos por sus hijos se da únicamente si el hijo es mejor que el padre respecto de la función objetivo. Para la solución de los problemas planteados en este trabajo se hizo uso de la versión de Evolución Diferencial: rand/1/bin el cual fue programado en C# haciendo uso del pseudocódigo mostrado en Fig. 4.

Los parámetros a obtener son los del controlador PD. Este controlador es utilizado para llevar al sistema de péndulo simple a la posición deseada y cumplir con la tarea de seguimiento es un PD (siglas de Proporcional y Derivativo). Este tipo de controlador se encarga de aplicar una acción correctora para que el error entre la señal de referencia y la señal de salida de la planta, se reduzca. Teniendo como ventajas la estabilidad general del sistema, es capaz de manejar procesos con retraso de tiempo y reduce el tiempo de asentamiento al mejorar la amortiguación y reducir el sobreimpulso. El PD convencional tiene solo dos parámetros para ajuste [1] como se describe en la ecuación (14):

$$u(t) = k_p e(t) + k_d \frac{de(t)}{dt}, \tag{14}$$

Algorithm 1: DE/rand/1/bin

```
create a random population ^0x_i i = 1, ...NI
evaluate initial fitness population f_0 = f(^0x_i)
for g := 1 to G_max do
    for i := 1 to NI do
        select randomly {r_1 ≠ r_2 ≠ r_3}
        for j := 1 to NV do
            if U(0, 1) < CR then
            v_ij = r_1j + F(r_2j − r_3j)
            else
            v_ij =^{g−1} x_ij
            end
        end
        if ^g v_i <<^{g−1} x_i then
        ^g x_ij =^g v_i
        else
        ^g x_ij =^{g−1} x_i
        end
    end
end
```

Fig. 4. Algoritmo ED/rand/1/bin

definiendo:

$$e(t) = q_d(t) - q(t), \tag{15}$$

donde:
$e(t)$ es el error existente entre la posición deseada y la posición actual.
q_d es la posición deseada a la cual debe de llegar el péndulo.
$q(t)$ es la posición actual en la que se encuentra el sistema.

6. Resultados

El algoritmo se ejecutó treinta veces con el fin de resolver el problema de optimización planteado en la ecuación (7). La inicialización de parámetros del algoritmo se estableció como se muestra en la Tabla 2.

Tabla 2. Parámetros de inicialización.

Descripción	Valor
Individuos	6
Generaciones (G_{max})	50
Factor cruza	0.8
Factor de mutación	0.5

En la Fig. 5 se muestra como el algoritmo de Evolución Diferencial en conjunto con Unity 3D corren los 6 individuos de forma paralela teniendo cada uno su gráfica del comportamiento en cada generación y los valores obtenidos al finalizar dicha generación. Así mismo, se observa como funciona cada uno de los individuos en cada evaluación que se realiza. El mostrar el valor obtenido de cada péndulo al paso de las generaciones permite al usuario observar las soluciones brindadas por el algoritmo de Evolución Diferencial y el comportamiento del péndulo en cada solución.

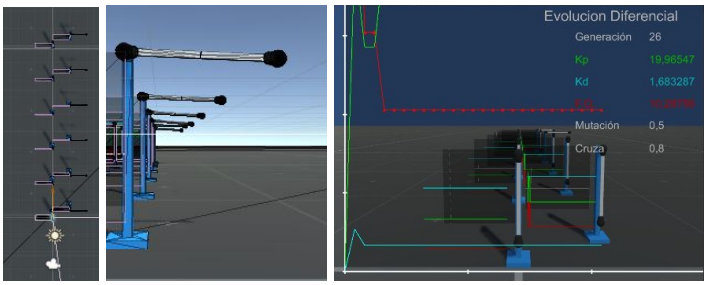


Fig. 5. Péndulos trabajando en paralelo.

En la Tabla 3 se muestran los resultados obtenidos al finalizar las treinta corridas del algoritmo de optimizacion. Se puede observar que los resultados obtenidos de despues de la ejecución del algoritmo de optimización treinta veces. Se muestra el mejor valor objetivo el cual es el que tiene menor valor, es decir el que implica que el error es casi cero y la función sobretiro es cero.

Tabla 3. Estadística.

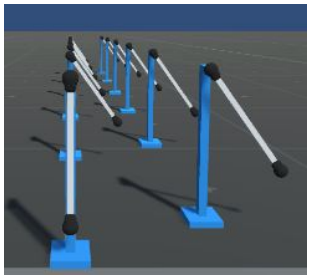
	Valor de la Función objetivo
Mejor	$3{,}8925x10^{-5}$ $[rad]$
Mediana	$4{,}0875x10^{-5}$ $[rad]$
Peor	$4{,}9456x10^{-5}$ $[rad]$
Desviación Estandar	$5{,}6026x10^{-6}$ $[rad^2]$

$$\boldsymbol{p}^* = [40{,}0000 \quad 23{,}439]^T, \tag{16}$$

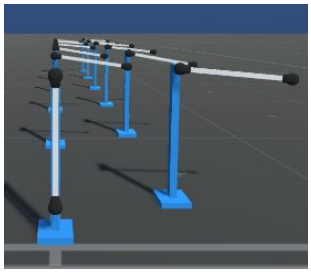
$$f(\boldsymbol{p}^*) = 3{,}8925x10^{-5}. \tag{17}$$

En la Fig. 6.(a) se muestra la inicializacion de los péndulos y la Fig. 6.(b) muestra que se están evaluando todos los hijos y todos los padres en el ambiente

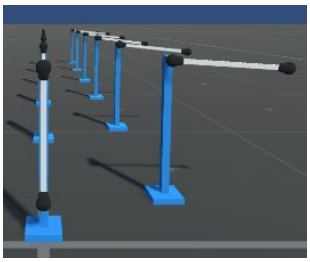
virtual. Por otro lado, en la Fig. 6.(c) se observa que a partir de la segunda generación únicamente los hijos son evaluados. En estas imagenes resulta sencillo observar las propiedades de paralelización con las que cuenta el uso del ambiente virtual de Unity y de esta manera se evaluan en un mismo *mundo* todos los individuos de la población.



(a) Inicio



(b) Padres



(c) Hijos

Fig. 6. Péndulos al paso de las generaciones.

7. Conclusiones

El presente trabajo muestra la posibilidad de hacer experimentación científica sobre fenómenos descritos por las leyes de Newton sin la necesidad de tener el modelo matemático del mismo. Esto se logra haciendo uso de los motores de física que hoy en dia se desarrollan para múltiples actividades los cuales resultan muy útiles para virtualizar el modelo de un sistema y tomar un camino alterno a la hora de correr simulaciones utilizando equipo de cómputo convencional. Este enfoque se probó al aplicarlo al virtualizado del sistema de un péndulo simple el cual sirvío para solucionar el problema de sintonización de un controlador PD para llevar al sistema a una posición deseada. La solución a este problema se obtuvó aplicando el algoritmo de Evolución Diferencial al problema de optimización resultante. De este modo se comprobó la viabilidad y aplicación de virtualizar sistemas mecánicos para la aplicación de técnicas de estudio. Esta herramienta representa un camino alterno para sistemas complejos

de modelar pero relativamente sencillos de virtualizar. Los resultados obtenidos fueron satisfactorios tanto para reproducir el comportamiento del sistema como para lograr la sintonización óptima de las ganancias de un controlador PD para llevar al péndulo a la posición deseada.

Agradecimientos. El primer, quinto y sexto autor agradecen al CONACyT el apoyo otorgado a través de la beca para estudios de doctorado. Todos los autores agradecen al Instituto Politécnico Nacional por el apoyo otorgado a través de la Secretaria de Investigación y Posgrado por medio de los proyectos SIP20196288 y SIP20190245.

Referencias

1. Asl, R.M., Pourabdollah, E., Salmani, M. (eds.): Optimal fractional order PID for a robotic manipulator using colliding bodies design. Soft Comput (2017)
2. Bose, M., Rajagopala, V.: Physics engine on reconfigurable processor — low power optimized solution empowering next-generation graphics on embedded platforms. In: 2012 17th International Conference on Computer Games (CGAMES). pp. 138–142 (July 2012)
3. Bzhikhatlov, I., Perepelkina, S.: Research of robot model behaviour depending on model parameters using physic engines bullet physics and ode. In: 2017 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM). pp. 1–4 (May 2017)
4. Cabrera, J., Simon, A., Prado, M.: Optimal synthesis of mechanisms with genetic algorithms. Mechanism and Machine Theory 37(10), 1165–1177 (2002), http://www.sciencedirect.com/science/article/pii/S0094114X02000514
5. He, Z., Shi, M., Li, C.: Research and application of path-finding algorithm based on unity 3d. In: 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS). pp. 1–4 (June 2016)
6. Ito, Y., Fujimoto, K., Tadokoro, Y., Yoshimura, T.: Risk-sensitive linear control for systems with stochastic parameters. IEEE Transactions on Automatic Control 64(4), 1328–1343 (April 2019)
7. Jadbabaie, A., , Morse, A.S.: Coordination of groups of mobile autonomous agents using nearest neighbor rules. IEEE Transactions on Automatic Control 48(6), 988–1001 (June 2003)
8. Jie, J., Yang, K., Haihui, S.: Study on the virtual natural landscape walkthrough by using unity 3d. In: 2013 International Conference on Computational and Information Sciences. pp. 1–4 (June 2013)
9. Lamperski, A., Ghusinga, K.R., Singh, A.: Analysis and control of stochastic systems using semidefinite programming over moments. IEEE Transactions on Automatic Control 64(4), 1726–1731 (April 2019)
10. Mohapatra, T.K., Sahu, B.K.: Design and implementation of ssa based fractional order pid controller for automatic generation control of a multi-area, multi-source interconnected power system. In: 2018 Technologies for Smart-City Energy Security and Power (ICSESP). pp. 1–6 (March 2018)
11. Nayak, P.C., Prusty, U.C., Prusty, R.C., Barisal, A.K.: Application of sos in fuzzy based pid controller for agc of multi-area power system. In: 2018 Technologies for Smart-City Energy Security and Power (ICSESP). pp. 1–6 (March 2018)

12. Rao, S.S.: Engineering optimization: Theory and practice. John Wiley and Sons, Inc. (New York, USA 1994)
13. Shutnan, W.A., Abdalla, T.Y.: Artificial immune system based optimal fractional order pid control scheme for path tracking of robot manipulator. In: 2018 International Conference on Advance of Sustainable Engineering and its Application (ICASEA). pp. 19–24 (March 2018)
14. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11(4), 341–359 (Dec 1997), https://doi.org/10.1023/A:1008202821328
15. Suárez-Santillán, R.A., Hernández-Oliva, N., Morales-Cruz, C., Calva-Yánez, M.B., Sepúlveda-Cervantes, G., Portilla-Flores, E.A.: Implementación en hardware-in-the-loop de un esquema completo de control para un sistema de péndulo simple utilizando un controlador pid.
16. Velázquez Reyes, J.: Evolución diferencial para problemas de optimización de espacios restringidos. Master's thesis, Sección de Computación, CINVESTAV-IPN (México 2006)
17. Xiaofeng, L., Tianhuang, C., Tingchao, Q.: Making a virtual sand table based on unity 3d technique. In: 2014 13th International Symposium on Distributed Computing and Applications to Business, Engineering and Science. pp. 278–281 (Nov 2014)
18. Yang, J., Hu, G.: A physics simulation engine for modeling autonomous motion through reverse engineering of sodaconstructor. In: 2009 IEEE/ACS International Conference on Computer Systems and Applications. pp. 287–292 (May 2009)
19. Zeigler, B.P., Muzy, A.: From discrete event simulation to discrete event specified systems (devs). IFAC-PapersOnLine 50(1), 3039–3044 (2017), http://www.sciencedirect.com/science/article/pii/S2405896317310601, 20th IFAC World Congress