Memorias del Congreso Nacional de Control Automático
ISSN: 2594-2492

Comparación de dos técnicas evolutivas para la sintonización de PIDs en sistemas con retardo

René Martínez, Carlos A. Brizuela, L.A. Márquez-Martínez

CICESE Research Center. 3918 Carr. Ensenada-Tijuana.
22860 Ensenada B.C. Mexico.
Email:{rmartinez,cbrizuel,lmarquez}@cicese.edu.mx

Resumenen este trabajo se presentan dos técnicas de optimización meta-heurística para sintonizar un controlador PID. Se consideran los métodos evolución de diferencial y optimización por enjambre de partículas, utilizando la heurística de Ziegler-Nichols para establecer los espacios de búsqueda y con base a ellos realizar la minimización de diferentes índices de desempeño para una planta lineal de segundo orden con retardo en la entrada. Las simulaciones numéricas muestran un mejor desempeño de las meta-heurísticas comparado con el obtenido usando la función PIDTUNE de Matlab®.

Keywords: control PID, sintonización, indices de desempeño, algoritmos evolutivos, retardos, sistemas de segundo orden.

1. INTRODUCCIÓN

El controlador proporcional-integral-derivativo (PID) (Minorsky (1922)) es utilizado en más del 90 % de aplicaciones industriales (Åström and Hägglund (2001)), debido a su construcción simple y otras características como robustez y facilidad de implementación. Un obstáculo que puede surgir con este controlador es la sintonización de sus parámetros. Si se realiza una mala sintonización puede ocasionar un rendimiento deficiente o incluso provocar inestabilidad en el sistema de control.
A través de los años, se han desarrollado varios métodos para determinar los parámetros del controlador PID. En la literatura exsiten más de 1,500 fórmulas de sintonización (O'Dwyer (2009)). Un referente clásico es la heurística de Ziegler-Nichols (Z-N) (Ziegler and Nichols (1942)). Sin embargo, el rendimiento obtenido al utilizar Z-N no siempre es el adecuado. Por está razón, se han utilizado otros métodos para sintonizar el controlador PID.
En los últimos años se han implementado técnicas de optimización meta-heurística para encontrar los parámetros óptimos y mejorar el desempeño del controlador, entre ellos los algoritmos genéticos y evolución diferencial (Saad et al. (2012)), recocido simulado (Abrajan-Guerrero and Marquez-Martinez (2010)), explosión de mina (Majumdar et al. (2014)), y optimización por enjambre de partículas (Chen et al. (2017)).

Para definir la función objetivo, generalmente se utilizan los índices de desempeño que se basan en el cálculo del error entre la referencia y la respuesta del sistema. La integral del valor absoluto del error (IAE), la integral del error cuadrático (ISE), la integral del tiempo multiplicado por el valor absoluto del error (ITAE) y la integral del tiempo multiplicada por el error al cuadrado (ITSE), son algunos de los índices de desempeño clásicos. Existen otros como la integral del tiempo al cuadrado multiplicado por el error absoluto (ISTAE) y la integral del tiempo al cuadrado multiplicado por el error cuadrático (ISTSE) que no han sido muy utilizadas en la literatura.
En este trabajo se propone el uso de dos técnicas de optimización meta-heurística que han mostrado tener buen desempeño: evolución diferencial (Storn and Price (1997)) y optimización por enjambre de partículas (Kennedy and Eberhart (1995)), además del uso de los índices ISTAE e ISTSE como función objetivo. Las respuestas obtenidas son comparadas con las obtenidas utilizando la heurística de Z-N y la función *pidtune()* del programa MATLAB®.

2. PRELIMINARES

En esta sección se muestra la estructura del controlador PID que se va a utilizar, se explican los métodos de optimización que se usarán y los índices de desempeño que se implementarán como función objetivo.

2.1 Controlador proporcional-integral-derivativo (PID)

La estructura del controlador PID en el dominio de la frecuencia considerada es:

$$U_{PID}(s) = K_p \left(1 + \frac{1}{sT_i} + sT_d\right). \quad (1)$$

En el dominio del tiempo se considera:

$$u_{PID}(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \dot{e}(t) \right) \quad (2)$$

2.2 Técnicas consideradas

Se seleccionaron dos técnicas de optimización metaheurística: evolución diferencial (DE) y optimización por enjambre de partículas (PSO). Ambas fueron elegidas por haber mostrado un buen desempeño en otras aplicaciones y porque su implementación es sencilla. Además no es necesario que la función objetivo deba cumplir con características específicas como diferenciabilidad, convexidad e inclusive continuidad. Cada método se explica a continuación.

Evolución diferencial. Éste método básicamente realiza una búsqueda global en una población de soluciones candidatas, que se mutan y combinan para crear nuevos individuos que serán seleccionados dependiendo de su aptitud dada por la función objetivo. El algoritmo se detiene al llegar a un número determinado de generaciones.
El algoritmo se divide en 4 etapas:

1. Inicialización. Se genera una población inicial aleatoria de NP vectores de dimensión D, dónde D es el número de parámetros a optimizar y NP es igual a $D * 10$ ó $D * 15$. Se define
$$x_{i,G}, i = 1, 2, ..., NP, \quad (3)$$
como una población para cada generación G.
2. Mutación. Para cada vector objetivo $x_{i,G}, i = 1, ..., NP$, se genera un vector mutado de acuerdo a la siguiente ecuación,
$$v_{i,G+1} = x_{a_1,G} + F(x_{a_2,G} - x_{a_3,G}), \quad (4)$$
donde $a_1, a_2, a3 \in 1, 2, ..., NP$, son índices aleatorios enteros y $F > 0$. Los índices seleccionados al azar deben ser diferentes del índice i del vector objetivo. F es un factor constante y real $\in [0, 2]$ que actúa como ganancia de la diferencia $(x_{a_2,G} - x_{a_3,G})$.
3. Cruzamiento. Para ampliar el espacio buscado alrededor de los vectores mutados, se utiliza el cruzamiento, donde por medio de la ecuación 5 se crea el vector prueba.
$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & \text{si } (randb(j) \leq CR) \\ & \text{o } j = rnbr(i) \\ x_{ji,G} & \text{si } (randb(j) > CR) \end{cases}, \quad (5)$$
$$j = 1, 2, ..., D.$$
donde $randb(j) \in [0, 1]$ es la evaluación de un generador de números aleatorios con distribución uniforme. $CR \in [0, 1]$ es la constante de cruzamiento, y $rnbr(i) \in [1, 2, ..., D]$ es un índice seleccionado al azar que asegura al menos un parámetro del vector mutado en el vector prueba.
4. Selección: Para decidir si el vector prueba $u_{i,G+1}$ debe convertirse en miembro de la generación $G + 1$, se compara con el vector objetivo $x_{i,G}$, usando el criterio voraz. El cual consiste en evaluar el vector $u_{i,G+1}$, y si se obtiene un valor en la función de costo menor al obtenido con $x_{i,G}$, entonces $x_{i,G+1}$ será $u_{i,G+1}$, sino, se conserva el vector objetivo $x_{i,G}$.

Optimización por enjambre de partículas: Éste método al igual que evolución diferencial, es un optimizador estocástico. La inspiración de su desarrollo surgió a raíz de la elaboración de modelos de conductas sociales de los patrones de movimiento del vuelo de las aves y de los bancos de peces. PSO comparte similitudes con otras técnicas de computación evolutiva, ya que se inicializa con una población de soluciones candidatas aleatorias y busca el valor óptimo al avanzar en las generaciones, aunque no utiliza los operadores de mutación y cruzamiento.
En el algoritmo PSO a las soluciones candidatas se les llama "partículas", las cuales están en constante movimiento en un espacio de búsqueda multidimensional de un problema de optimización siguiendo a las partículas mejor evaluadas, momento a momento. La posición de cada partícula representa una solución candidata al problema en cuestión.
Para utilizar este método debe generarse una población inicial de NP vectores (partículas).
$$P_i, i = 1, 2, ..., NP \quad (6)$$
Cada partícula será inicializada con una posición ($p_i \in \mathbb{R}^D$) aleatoria uniformemente distribuida, utilizando como valores máximos y mínimos, los respectivos límites inferiores y superiores del espacio de búsqueda de cada parámetro involucrado en el problema de optimización. También se inicializa la velocidad ($v_i \in \mathbb{R}^D$) en cero. Después de inicializar, se crea una memoria de la mejor posición conocida de la partícula a su posición inicial ($p_{best_i} = p_i$) y evaluando la posición de cada partícula de la población $f(p_i)$ se encuentra la mejor partícula, i.e., la más pequeña $f(p_i)$ que cumple con $f(p_i) < f(g_{best})$, donde g_{best} es la mejor posición global conocida.
Hecho esto, se realiza un proceso iterativo que mientras no se cumpla un criterio de paro actualizará la velocidad y la posición de las partículas. La velocidad se actualiza mediante la siguiente ecuación,
$$v_i^{k+1} = w * v_i^k + c_1 * r_1 * (p_{best_i} - p_i^k) + c_2 * r_2 * (g_{best} - p_i^k). \quad (7)$$
Con esta nueva velocidad se actualiza la posición,
$$p_i^{k+1} = p_i^k + v_i^{k+1} \quad (8)$$
De nuevo se evalua cada partícula y si $f(p_i) < f(p_{best_i})$, se actualiza p_{best_i} y si $f(p_{best_i}) < f(g_{best})$, se actualiza g_{best}. El algoritmo regresa g_{best} una vez que la condición de paro se satisface.

2.3 Función de costo.

Para el diseño de un controlador eficiente, el objetivo es mejorar la respuesta del sistema minimizando los parámetros en el dominio del tiempo, como el sobrepaso (M_p), el tiempo de asentamiento (t_s) y el error en estado estacionario (e_{ss}). Esto se puede lograr minimizando el error entre la referencia y la salida del sistema, expresada de la siguiente forma,
$$e(t) = y_{ref}(t) - y(t) \quad (9)$$

el cual es utilizado en los índices de desempeño, los que a su vez son usados como función de costo. En este trabajo se utilizaron los siguientes índices:

Integral del tiempo multiplicado por el valor absoluto del error

$$ITAE = \int_0^\infty t|e(t)|dt \tag{10}$$

Integral del tiempo al cuadrado multiplicado por el error absoluto

$$ISTAE = \int_0^\infty t^2|e(t)|dt, \tag{11}$$

Integral del tiempo al cuadrado multiplicado por el error cuadrático

$$ISTSE = \int_0^\infty t^2e^2(t)dt, \tag{12}$$

Se seleccionaron estos índices ya que presentaron una buena relación entre sus óptimos y el del tiempo de asentamiento o del sobreimpulso.

2.4 Planta considerada

La planta nominal seleccionada es de segundo orden con retardo, conocida como SOPTD (ver ec. 13). Se tomó del trabajo de (Chen et al. (2017)), y presenta un retardo de $0{,}2s$. La sintonización del controlador PID la llevaron a cabo utilizando el método de PSO con algunas modificaciones.

$$G_1(s) = \frac{0{,}5e^{-0{,}2s}}{0{,}08s^2 + 0{,}68s + 1{,}45}, \tag{13}$$

3. CONFIGURACIÓN EXPERIMENTAL Y RESULTADOS

Para comprender mejor el comportamiento de búsqueda de las dos técnicas evolutivas, se realiza un conjunto de experimentos de simulación, donde se compara los distintos resultados obtenidos y se analiza su rendimiento. Los parámetros utilizados en los algoritmos y los resultados se muestran en esta sección.

Los algoritmos de optimización se programaron en el lenguaje de programación interpretado Python versión 2.7.12. La simulación de la planta con retardo se realizó en lenguaje C++. Las simulaciones se llevaron a cabo en una computadora con Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz y 16GB de memoria.
Para las simulaciones se utilizó un tamaño de población de 30 individuos para DE y 30 partículas para PSO, es decir, $NP = 30$. Para el número de generaciones o iteraciones máximas se estableció un valor de 50, es decir, $G = I_{max} = 50$, y para el espacio de búsqueda se utilizó $S = [[0, 10], [0, 1], [0, 0{,}2]]$, en ambos algoritmos para las constantes K_p, T_i y T_d, respectivamente. Para las variables particulares de DE como mutación y cruzamiento, se utilizó: $F = 0{,}5$ y $C = 0{,}85$, respectivamente. Y para las variables de PSO, como el peso de inercia, constante cognitiva y constante social, se usó: $w = 0{,}2$, $c_1 = 2$ y $c_2 = 2$. Los valores de los parámetros NP, I_{max}, S, w, c_1 y c_2 fueron tomados de (Chen et al. (2017)) para hacer una comparación adecuada.

Los valores K_p, T_i y T_d que se muestran en las siguientes tablas, son los parámetros óptimos de un muestreo de 30 pruebas, tomando en cuenta el menor sobrepaso y el menor valor de costo.

En las Tablas 1 y 2 se muestran los parámetros óptimos obtenidos por ambas meta-heurísticas para cada función de costo. Las Figuras 1 y 3 muestran las respuestas al escalón del sistema con el controlador PID sintonizado, en estas gráficas se incluye la respuesta generada por KPSO de (Chen et al. (2017)) como comparación. En las Figuras 2 y 4 se muestra la señal de control para cada una de las respuestas.

Tabla 1. Resultados para $G_1(s)$ optimizando con DE.

Función de costo	K_p	T_i	T_d	$M_p(\%)$	$t_s(seg)$
ITAE	3.765724	0.645460	0.147523	0.03	0.6
ISTAE	3.392363	0.604862	0.139409	0.006	0.68
ISTSE	3.968908	0.672873	0.151770	0.0	1.14

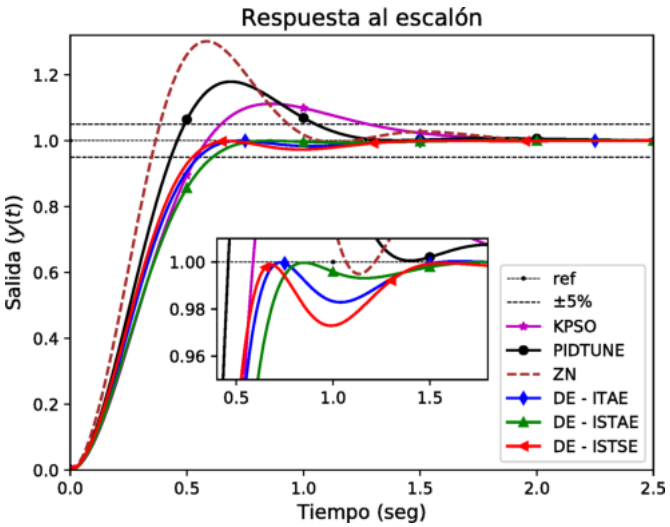

Figura 1. Respuesta al escalón para $G_1(s)$ sintonizando con DE.

Tabla 2. Resultados para $G_1(s)$ optimizando con PSO.

Función de costo	K_p	T_i	T_d	$M_p(\%)$	$t_s(seg)$
ITAE	3.019586	0.566363	0.132067	0.14	0.77
ISTAE	3.486443	0.633824	0.154874	0.02	1.06
ISTSE	3.850574	0.652922	0.149674	0.11	0.58

4. DISCUSIONES

Los mejores resultados se obtuvieron con los índices ITAE e ISTAE, particularmente, al optimizar con evolución diferencial, el índice ISTAE arroja mejores resultados. Sin

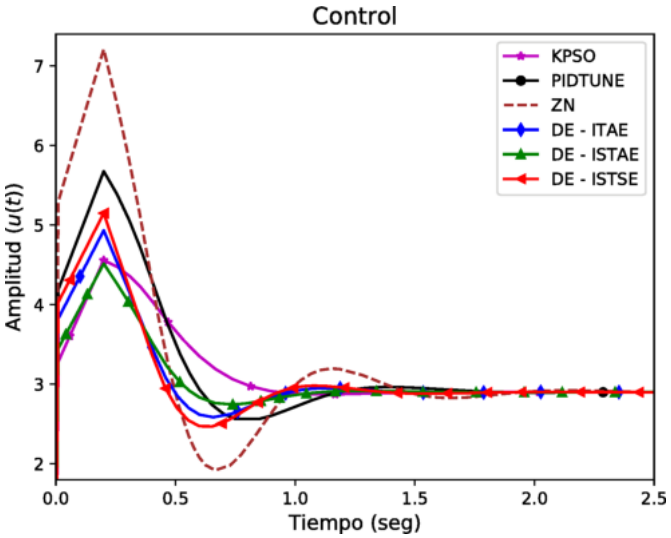

Figura 2. Señal de control para $G_1(s)$ sintonizando con DE.

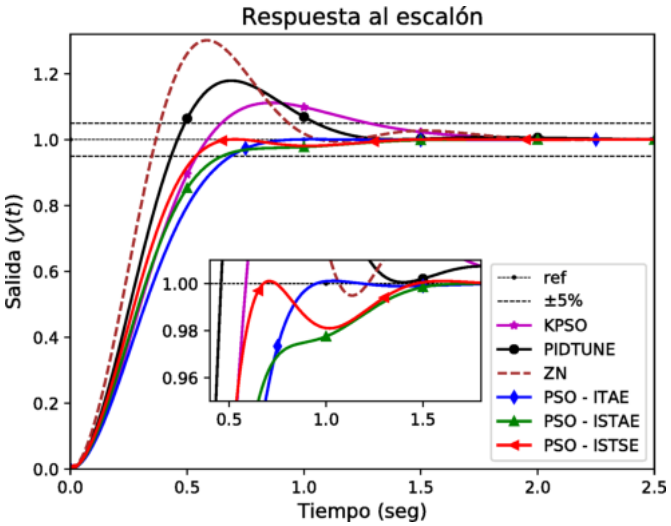

Figura 3. Respuesta al escalón para $G_1(s)$ sintonizando con PSO.

embargo al optimizar con PSO, ITAE tiene una mejor respuesta. Esto puede deberse a que ambos métodos son estocásticos y al inicializar sus poblaciones se hace de manera aleatoria, por lo que ambos pueden converger a un valor distinto. También debe tomarse en cuenta que se utiliza una condición que verifica que el sobrepaso sea cada vez menor, esto puede estancar al algoritmo en algún determinado valor. Comparando las señales de control, se puede observar que con los índices ITAE e ISTAE se obtienen señales que no tiene una amplitud muy grande comparadas con el otro índice y los otros métodos por lo que se sintonizó el controlador PID. En la Tabla 3 se muestran los parámetros para el controlador PID, el sobrepaso y el tiempo de asentamiento de los métodos contra los que se está comparando los algoritmos evolutivos.

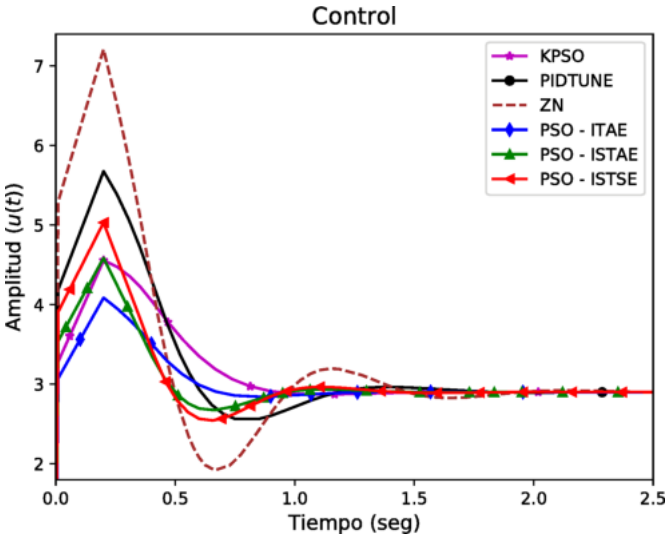

Figura 4. Señal de control para $G_1(s)$ sintonizando con PSO.

Para comparar el desempeño de ambas meta-heurísticas y verificar que las diferencias observadas son estadísticamente significativas, se llevó a cabo una prueba no paramétrica, la prueba de Wilcoxon, con un nivel de confianza del 99 %. En las tablas 4 y 5 se muestran la media y la desviación estándar de los resultados para la optimización de los índices de desempeño como función de costo y los parámetros que se obtuvieron en las respuestas al escalón, respectivamente, al utilizar los algoritmos evolutivos. La prueba de Wilcoxon muestra que las diferencias son efectivamente estadísticamente significativas. En ellas se puede inferir que en general la meta-heurística evolución diferencial presenta mejores resultados, ya que logra minimizar aún más los índices de desempeño que la optimización por enjambre de partículas, y además se puede observar que en los parámetros como el sobrepaso y el tiempo de asentamiento también se tiene mejores resultados, debido a que se obtiene un mejor conjunto de parámetros K_p, T_i y T_d y por ende una mejor respuesta al escalón. El tiempo promedio de ejecución para el algoritmo de evolución diferencial fue de 5.6 segundos y de 2.82 segundos para el de PSO.

Tabla 3. Resultados para $G_1(s)$ utilizando KPSO, PIDTUNE y ZN.

Método	K_p	T_i	T_d	M_p (%)	t_s (seg)
KPSO	3.2140	0.478943	0.119508	11.10	1.56
PIDTUNE	4.11	0.525	0.117	17.8	1.15
ZN	5.1939	0.5155	0.1289	29.92	1.65

5. CONCLUSIONES

En este trabajo se presentaron dos métodos de optimización meta-heurística fáciles de implementar para sintonizar un controlador PID. En las simulaciones, se comparó

tres formas de sintonizar PID, por medio de Z-N, utilizando el sintonizador de MATLAB®, *pidtune()*, y una variante de PSO, KPSO. Los resultados muestran significativamente una mejora tanto en sobrepaso como en tiempo de asentamiento. Los mejores resultados se obtuvieron al usar ITAE e ISTAE como funciones objetivo. Siendo el segundo el que mejor resultado presenta al comparar las señales de control. Lo interesante a destacar es que a pesar de sólo utilizar plantas lineales y el controlador PID, éstos métodos pueden utilizarse en plantas no lineales y con otros tipos de controladores, dónde se vean involucrados la sintonización de parámetros.

REFERENCIAS

Abrajan-Guerrero, R. and Marquez-Martinez, L.A. (2010). Automatic PID tuning , a heuristic optimization approach. *Amca*.

Åström, K. and Hägglund, T. (2001). The future of PID control. *Control Engineering Practice*, 9(11), 1163–1175.

Chen, J., Omidvar, M.N., Azad, M., and Yao, X. (2017). Knowledge-based particle swarm optimization for PID controller tuning. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, 1819–1826. IEEE.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. *Neural Networks, 1995. Proceedings., IEEE International Conference on*, 4, 1942–1948 vol.4. doi:10.1109/ICNN.1995.488968.

Majumdar, S., Mandal, K.K., and Chakraborty, N. (2014). Tuning of robust PID controller using mine blast algorithm. In *Proceedings of The 2014 International Conference on Control, Instrumentation, Energy and Communication (CIEC)*, 752–756. IEEE.

Tabla 4. Estadísticas de la optimización de las funciones de costo.

Función de costo	Estadística	DE	PSO
ITAE	Media	0.155694	0.201336
	Desv. estándar	0.001409	0.056923
ISTAE	Media	0.051917	0.093530
	Desv. estándar	0.000425	0.053576
ISTSE	Media	0.012982	0.015136
	Desv. estándar	0.000159	0.003347

Tabla 5. Estadísticas de los parámetros de las respuestas al escalón.

Parámetro	Función de costo	Estadística	DE	PSO
M_p (%)	ITAE	Media	0.07	0.37
		Desv. estándar	0.10	0.47
	ISTAE	Media	0.04	0.31
		Desv. estándar	0.01	0.26
	ISTSE	Media	0.63	0.63
		Desv. estándar	2.26	0.83
t_s (seg)	ITAE	Media	0.62	0.93
		Desv. estándar	0.07	0.20
	ISTAE	Media	0.68	0.93
		Desv. estándar	0.03	0.21
	ISTSE	Media	1.11	0.92
		Desv. estándar	0.02	0.23

Minorsky, N. (1922). Directional stability of automatically steered bodies. *Journal of ASNE*, 42(2), 280–309.

O'Dwyer, A. (2009). *Handbook of PI and PID controller tuning ruler*.

Saad, M.S.M., Jamaluddin, H., and Darus, I.Z.M. (2012). Implementation of PID controller tuning using differential evolution and genetic algorithms. *International Journal of Innovative Computing, Information and Control*, 8(11), 7761–7779.

Storn, R. and Price, K. (1997). Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4), 341–359.

Ziegler, J.G. and Nichols, N.B. (1942). Optimum settings for automatic controllers. *InTech*, 42(6), 94–100. doi: 10.1115/1.2899060.

