JAIC
Journal of Applied Instrumentation and Control

Controlador *Fuzzy-PID* Adaptativo Otimizado por PSO para Controle de Ventoinha em uma Bancada Didática

Mario Henrique Bigai, Gabriel Franco Harteman, Adriele Ortiz Gomes, Marcella Scoczynski Ribeiro Martins e Fernanda Cristina Corrêa

***Resumo*—Nesse trabalho foi projetado um controlador *Fuzzy-PID* adaptativo otimizado por enxame de partículas (PSO), onde usou-se variar os limites dos universos de discurso das variáveis de entrada e saída do sistema de controle *fuzzy*, mantendo as respectivas funções de transferências proporcionais, com objetivo de obter o menor tempo de estabilização possível da ventoinha contida na placa PICGenius, e acionada por um PWM e pelo Microcontrolador PIC 18F4550. Os resultados foram comparados ao controlador PID convencional, onde foi possível observar que o controlador *Fuzzy-PID* proporcionou melhorou resultados no sistema controlado.**

***Palavras-chaves*—PSO, Sistema de Controle Fuzzy, Inteligência Artificial.**

I. Introdução

Controladores PID são frequentemente utilizados em diversas aplicações industriais e podem ter seu desempenho aprimorado com a aplicação de técnicas adaptativas, que reúnem dados do ambiente operacional do sistema realizando ajustes quando há mudanças de condição previamente estabelecidas. Assim, a tomada de decisão no sistema de controle será baseada na condição encontrada atualmente, minimizando, portanto, falhas no sistema. Uma dessas técnicas adaptativas é a lógica Fuzzy, que pode auxiliar sistemas dinâmicos em ambientes instáveis. Nesse sentido, a união de um PID a um controlador Fuzzy resulta em um controlador híbrido conhecido como controlador *Fuzzy-PID* [1].

O controlador Fuzzy – PID integra a vantagem de ambas as estruturas de controle e, como consequência uma melhoria no desempenho tanto em termos do transitório quanto ao estacionário [2].

Porém, para que o controlador fuzzy funcione corretamente, é necessário ter pleno conhecimento do sistema a ser controlado, pois se trata de um controle totalmente empírico, os valores utilizados na simulação e implementação podem não ser os ideais. Para resolver este problema, um algoritmo para otimizar as funcionalidades pode ser aplicado. Uma dessas técnicas é a otimização por enxame de partículas (PSO), que cria um universo de resultados possíveis e, em poucas interações, pode trazer melhores parâmetros para o controlador [1].

O objetivo deste artigo é desenvolver um controlador adaptativo *Fuzzy-PID* para uma aplicação de controle de uma ventoinha, acionada por PWM a fim de comparar seu desempenho com um PID discreto convencional. Além disso, o controlador *Fuzzy-PID* é sintonizado pelo PSO visando a redução do tempo de acomodação na situação de partida da ventoinha, ou seja de 0 à 98% de sua velocidade. O estudo foi baseado no trabalho de [3] onde foi desenvolvido um controlador PID discreto para uma planta similar.

Recentemente os controladores *Fuzzy-PID* tem sido utilizados em diversas aplicações em várias áreas da engenharia, como por exemplo em controle de motores do tipo BLDC para o aumento de vida útil [4], em processos químicos de um sistema de biomassa para uma caldeira a vapor [5] e em outras aplicações complexas com funções de transferências de ordem superiores não lineares. Além disso, alguns trabalhos já aplicaram o PSO como otimizador, por exemplo em [6], onde os autores propõe a sintonia online dos parâmetros do controlador através de um mecanismo de adaptação baseado em algoritmo em PSO. Também em [7] é apresentado um método de busca baseado no PSO para otimização de um controlador fuzzy aplicado à um sistema de suspensão, e em [8], onde o PSO é usado para ajustar um controlador fuzzy de um sistema robótico. Diante dos promissores resultados aplicando o PSO, este trabalho também visa incorporar este

algoritmo de otimização para explorar didaticamente o controlador adaptativo *Fuzzy-PID* em uma ventoinha.

Como a ventoinha é uma planta relativamente simples e está contida no kit didático PicGenius, o intuito do trabalho é desenvolver um controlador adaptativo *Fuzzy-PID* otimizado e apresentar de forma simples as vantagens deste controlador em relação a um PID discreto convencional, como a melhoria do desempenho tanto em regime transitório como em regime estacionário. E ainda, é possível explorar as vantagens de utilizar o algoritmo de otimização PSO nos projetos de controle.

O projeto dos controladores PID e *Fuzzy-PID* adaptativo, bem como o algoritmo de otimização, foram feitos em *Python*, através da IDE online *Google Collaboratory*, que é uma alternativa aos softwares convencionais de projeto de controle, e permite a execução de comandos em um ambiente em nuvem. Para o projeto dos controladores foram utilizadas as bibliotecas *control* e *scikitfuzzy*.

II. Lógica Fuzzy

Com o propósito de auxiliar na busca da solução de problemas específicos, a lógica difusa ou comumente conhecida como lógica *Fuzzy* procura se aproximar do pensamento humano, saindo da lógica booleana e buscando respostas lidando com o conceito de verdade parcial. Isto ocorre, pois a base dos sistemas *Fuzzy* é a teoria dos conjuntos *Fuzzy* no qual seus elementos possuem um grau de pertinência associado que é determinado pela análise das funções de pertinência [9].

Uma das principais vantagens de se utilizar o controlador *Fuzzy* é a de que o projetista pode realizar o controle de um sistema com base no comportamento que o mesmo possui, sem a necessidade de obtenção do seu modelo matemático [10]. Porém, para que isso ocorra é necessário que o projetista tenha um pleno conhecimento do funcionamento do sistema de maneira que consiga ajustar os parâmetros de forma correta fazendo com que o esforço de controle trate de maneira específica cada entrada com o propósito de atingir os objetivos esperados.

Para que ocorra o processamento das variáveis numéricas emitidas ao controlador, por exemplo um sinal enviado por um sensor, é necessária a execução de um processo que consiste em transformar estes valores numéricos em variáveis linguísticas para realizar a tomada de decisão com base em regras pré-estabelecidas, regras estas que estão associadas a um valor numérico necessário para efetuar o controle da planta. Estas etapas podem ser definidas como: fuzzificação, inferência e defuzzificação [11], conforme representado na Figura 1.

A etapa de fuzzificação consiste em transformar os dados de entrada, que são variáveis numéricas, em variáveis linguísticas, onde ocorre um pré-processamento de categorias com a finalidade de reduzir o número de processos. Já na inferência ocorrem as decisões com base no condicional Se-Então, em inglês *If-Then*, definidas por uma base de regras previamente estabelecidas definido as ações a serem tomadas em determina ocasião. A última etapa do processamento do sinal, a defuzzificação, assegura a interpretação exata das variáveis linguísticas obtidas na fase da inferência em valores numéricos [11].

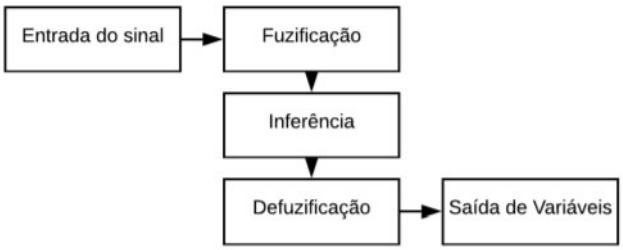

Fig. 1. Estrutura de um controlador de lógica fuzzy

III. Controlador Híbrido *Fuzzy*-PID

Apesar de ser robusto e de fácil aplicação o controlador PID possui algumas limitações em determinadas aplicações, como por exemplo em sistemas que não se comportam de maneira linear ou em situações em que a dinâmica da planta varie constantemente, tais situações podem impactar no tempo de resposta do controlador. Uma das alternativas para tal situação seria a união do controlador PID e a lógica *Fuzzy*.

Neste controlador, a aplicação da lógica *Fuzzy* é acoplada a um controlador PID para ajustar os seus parâmetros automaticamente em um processo on-line (Figura 2), ou seja, se houver modificações na dinâmica da planta, como por exemplo variações de carga, os ganhos do controlador PID são ajustados por meio da lógica *Fuzzy* para se adaptarem à essa mudança em tempo real [2].

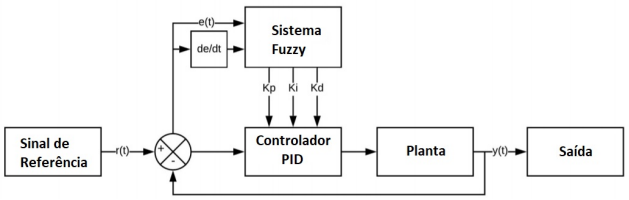

Fig. 2. Estrutura básica de um controlador híbrido *Fuzzy*-PID.

Esta união pode tornar o sistema estável mais rapidamente que um controlador PID clássico, resultando uma diminuição do tempo de acomodação para se atingir o estado estacionário [12].

IV. Otimização por Enxame de Partículas

A. Visão Geral

A Otimização por Enxame de Partículas, PSO da sigla em inglês, é uma técnica de otimização baseada no comportamento social de animais, como cardume de peixes ou bandos de pássaros, navegando pelo espaço. Ela foi desenvolvida por J. Kennedy e R. Eberhart em 1995 [13]. O PSO é muito utilizado em otimização de controladores do tipo *Fuzzy*-PID como pode ser visto em [4], [14], [15]

O PSO tem uma população inicial de indivíduos (partículas) e uma função de avaliação que mede como cada indivíduo no enxame está em relação a otimização do problema, sendo a interação social dos indivíduos e a troca de conhecimento entre as partículas o principal artifício para a solução da otimização.

A função do PSO é evoluir o enxame de partículas inicialmente randômicas, para que em cada iteração cada partícula seja atualizada de modo a seguir os melhores valores para o problema, e que todas as partículas do enxame cheguem a essa solução levando em conta as iterações anteriores. Cada partícula tem uma posição x e uma velocidade v. A posição representa a solução do problema e a velocidade representa o deslocamento da partícula no espaço de soluções.

Um algoritmo PSO padrão pode ser representado pelo fluxograma contido na figura 3

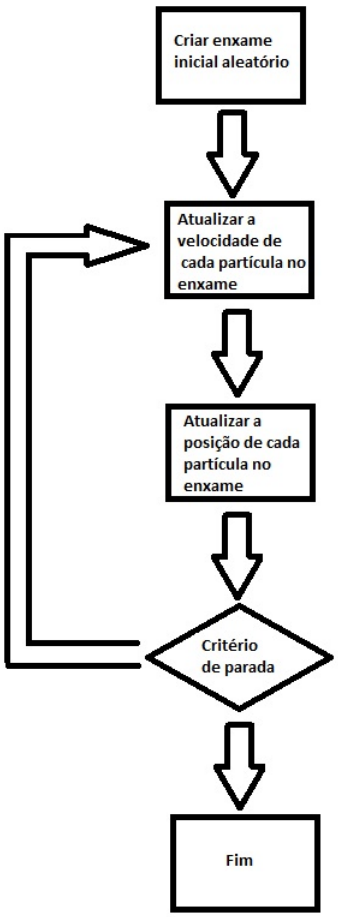

Fig. 3. Fluxograma de um PSO padrão.

Para calcular a posição e velocidade em cada iteração são usadas as equações 1 e 2.

$$x_{k+1}^{i} = x_{k}^{i} + v_{k+1}^{i} \tag{1}$$

$$v_{k+i}^{i} = w_k \cdot {v_k}^{i} + c_1 \cdot r_1 \cdot (p_k^i - x_k^i) + c_2 \cdot r_2 (p_k^g - x_k^i) \tag{2}$$

Sendo:

x_k^i = Posição da partícula i no instante k;

v_k^i= Velocidade da partícula i no instante k;

p_k^i= Melhor posição individual da partícula i no instante k;

p_k^g= Melhor posição global de todas as partículas no instante k;

w_k= Constante de inércia;

c_1, c_2= Termos cognitivos e de aprendizagem social das partículas;

r_1, r_2= Números aleatórios entre 0 e 1.

A velocidade é o parâmetro que guia o processo de otimização. Ela reflete o conhecimento individual de cada partícula e o conhecimento total do grupo. O conhecimento individual de cada partícula é representado pela componente cognitiva e o conhecimento total é representado pela componente de aprendizado social. Com essa informação, é possível observar que a velocidade depende de 3 termos: O termo de inércia que considera a velocidade anterior e dificulta as mudanças abruptas de direção, o termo cognitivo que direciona o indivíduo para a melhor posição achada por ele e a componente social que direciona os indivíduos para a melhor posição encontrada pelo enxame. Desse modo, as partículas convergem juntas para a melhor solução da otimização.

O valor do coeficiente de inércia w_k permite ajustar o busca local e global pela melhor solução, sendo que quanto mais próximo de 1 mais abrangente é a busca (global) e quanto mais próximo de 0 menos abrangente é a busca (local). Para garantir uma exploração maior do espaço de soluções e evitar que o PSO fique travado em melhores locais são introduzido os números randômicos r_1 e r_2. Assim, mesmo que o algoritmo convergir para uma solução satisfatória, ele continua executando para verificar se há melhores soluções.

Para definir a importância do comportamento social e cognitivo são introduzidos as constantes c_1 e c_2. Quando c_1 =c_2=0 , significa que os componentes social e individual tem a mesma importância. No modelo PSO social (c_1 = 0 e c_2 maior que zero) a componente social tem uma importância maior que a componente cognitiva. No PSO cognitivo (c_1 maior que zero e c_2 = 0) a componente individual tem uma importância maior que a social.

Os valores w_k, c_1, c_2, r_1 e r_2 são ajustados de acordo com o problema que será otimizado. Para essa otimização foi utilizada os valores w_k = 0.5, c_1 = 1, and c_2 = 2. O algoritmo utilizado nesse projeto foi baseado no algoritmo PSO de Nathan Rooy [16].

B. Função de Avaliação

A função de avaliação tem o propósito de medir o desempenho de cada partícula e avaliar o quão perto ela está da melhor combinação de parâmetros para o controlador, assim como definir o melhor indivíduo local e global.

Essa avaliação é feita considerando o tempo de estabilização como uma pontuação a ser minimizada. Essa pontuação é gerada pela função de 5 parâmetros do controlador adaptativo *Fuzzy-PID* de cada partícula, de modo a minimizar o tempo de estabilização. Também foram acrescentadas punições à pontuação relacionadas ao percentual de sobressinal e ao erro estacionário. Se o percentual de sobressinal for maior que 2% a pontuação é aumentada em 10% do valor do percentual de sobressinal. Se o erro estacionário for maior que 0.5 a pontuação aumentada em 40% do valor do erro estacionário.

Garantindo assim que o melhor global tenha o menor tempo de estabilização possível sem desrespeitar os requisitos de

projetos estabelecidos e também possibilitando que o algoritmo do PSO varie livremente os parâmetros e identifique a desestabilização do sistema.

C. Parâmetros de Otimização

Para a otimização foi considerado inicialmente o sistema com as funções de pertinência das figuras 6, 7, 8, 9 e 10. Como o controlador *Fuzzy*-PID adaptativo foi programado considerando o universo de discurso sempre simétrico, ele se inicia em -x e termina em x, assim como todos as funções de pertinência triangulares são definidos proporcionalmente aos valores do universo de discurso, mantendo assim o formato do controlador inicial, mas alterando o os limites do universo de discurso. Dessa forma, os parâmetros que serão variados pelo PSO são o universo de discurso no intervalo entre 0.01 e 40 para o erro e a derivada do erro, no intervalo entre 0.0001 e 10 para os ganhos kp e kd e no intervalo entre 10^{-6} e 10 para ki.

Seguindo esses parâmetros para o PSO, o algoritmo foi executado 10 vezes. Em cada execução gerou-se uma combinação de parâmetros, com tempo de estabilização, percentual de sobressinal e erro estacionário associado a ela. Salvou-se esses resultados em uma planilha do Google, implementada no *Python* e no *Google Colaboratory ID*E via biblioteca *gspread*.

V. Modelagem da ventoinha

Para o desenvolvimento de controladores do tipo PID é necessário conhecer o modelo matemático da planta. A obtenção do modelo matemático da planta foi realizada em duas etapas: coleta de dados e aproximação do sistema. Coletou-se os dados através de um sensor e um microcontrolador, com isso foi possível fazer a identificação do sistema para então, aproximar a resposta da planta a uma função de transferência.

Para a coleta de dados foi utilizada a placa didática Kit PICGenius que possui um microcontrolador da família PIC18F e um sensor infravermelho capaz de ler a rotação da ventoinha. Para a leitura da rotação através do sensor foi utilizado o programa MPLabX, onde programou-se o timer 0 do microcontrolador para gerar um pulso PWM e criou-se um arquivo com os dados em pás por 100 milisegundos (pas/100ms). Por meio do processo de identificação do sistema, chegou-se a uma função de transferência de primeira ordem apresentada na equação 3.

$$G(s) = \frac{80}{0.89 \cdot s + 1} \tag{3}$$

A técnica de controle é pensada para o microcontrolador, para isso a planta deve passar pelo processo de discretização. Para um processo de discretização eficiente, é necessário escolher o período de amostragem adequado. Para esse projeto definiu-se o período de amostragem sendo 100 ms, já que é o mesmo período de coleta de dados do sensor infravermelho.

Escolheu-se o método *Zero-Order-Hold (ZOH)* para a discretização porque ele mantém as características de estabilidade do sistema. Esse método se baseia em um segurador de amostras de ordem zero, isso significa que o valor das saídas do sistema equivale ao valor no instante da amostragem. Nesse processo, os polos são discretizados seguindo uma transformação exponencial. A função de transferência discretizada é apresentada em na equação 4. A figura 4 contém a resposta ao degrau da função de transferência discreta do modelo obtido, com tempo de estabilização de 4 segundos.

$$G(z) = \frac{8.502}{z - 0.8937} \tag{4}$$

Para o projeto do controlador *Fuzzy-PID* adaptativo em linguagem de programação *Python* é necessário implementar a equação a diferenças da planta, pois desse modo é possível obter todos os valores de saída da simulação individualmente. A equação a diferenças da ventoinha está contida na equação 5.

$$u_k = u_{k-1} \cdot 0.8937 + e_{k-1} \cdot 8.502 \tag{5}$$

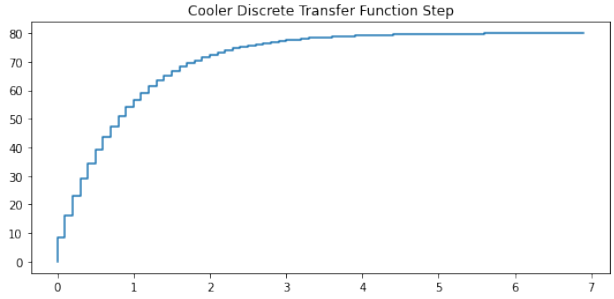

Fig. 4. Resposta ao degrau da ventoinha.

VI. Controlador PID Discreto Inicial

O projeto do controlador PID discreto inicial foi baseado no modelo exato representado na equação 6. Os requisitos de projeto para o controlador inicial são percentual de sobressinal menor que 15 %, tempo de estabilização menor que 3 segundos e erro estacionário nulo.

$$K(z) = \frac{(kp + kd + ki) \cdot z^2 - (kp + 2 \cdot kd) \cdot z + kd}{z \cdot (z - 1)} \tag{6}$$

Como pode ser visto na equação 6 a função de transferência do controlador PID é caracterizado por um pólo fixo em 1, um pólo fixo em zero e um zero complexo conjugado cujo os valores variam de acordo com os ganhos kp, ki e kd. Portanto para a obtenção dos valores dos pólos foi feita a função de transferência em malha fechada do controlador discreto com a planta discreta e variou-se os valores dos ganhos até que os requisitos de projeto fossem atendidos. Esse método é análogo a ferramenta *SISOTOOL* do software *MATLAB*. O valor dos ganhos kp, ki e kd para o controlador inicial são respectivamente: $0,02858$, $0,00751$ e $0,07291$.

Como o controle PID será implementado em um microcontrolador é necessário obter a equação a diferenças do controlador. Utilizando o mesmo período de amostragem da discretização da função de transferência da ventoinha e substituindo os valores dos ganhos obtidos no projeto do controlador PID, obteve-se a equação 7.

$$u_k = u_{k-1} + e_k \cdot 0.109 - e_{k-1} \cdot 0.1744 + e_{k-2} \cdot 0.07291 \quad (7)$$

A resposta ao degrau do conjunto ventoinha e controlador PID, ambos em equação a diferenças está contida na figura 5. Os parâmetros de desempenho são percentual de sobressinal de 12.72 %, tempo de estabilização de 2.8 segundos e erro estacionário de $9.19 \cdot 10^{-11}$. O controle estabiliza em 1 pois o esforço de controle é o *duty cycle* aplicado ao PWM para acionar a ventoinha.

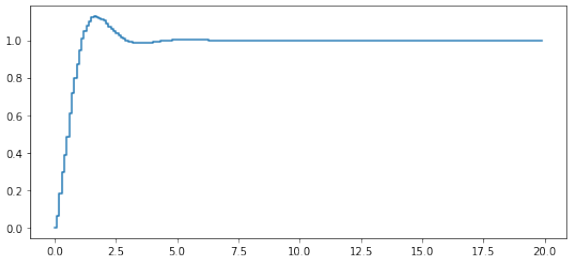

Fig. 5. Resposta ao degrau do PID inicial.

VII. Controlador *Fuzzy*-PID Adaptativo

A. Sistema de Inferência Fuzzy

Para implementar o sistema *fuzzy* foi utilizada a biblioteca *scikit fuzzy* em *Python*. Essa biblioteca implementa o sistema *fuzzy* por meio de objetos de entradas, saídas e regras. Com isso, as funções de pertinências são contidas nos objetos de entrada e saída por meio das variáveis linguísticas e as regras relacionam as variáveis linguísticas por meio de um objeto de simulação.

Um controlador adaptativo *Fuzzy-PID* é composto por um controlador PID e um sistema de inferência *fuzzy*. Para o projeto, considerou-se como as entradas o erro (E) e a derivada do erro (dE) da planta, e o sistema de inferência tem como saída os ajustes dos ganhos kp, ki, e kd do PID que influenciam na estabilidade, percentual de sobressinal, e tempo de estabilização do sistema.

As funções de pertinência para as variáveis são baseadas no artigo *"Study on Fuzzy Self-Adaptive PID Control System of Biomass Boiler Drum Water"* [5]. Nesse artigo, as funções de pertinência usadas para entrada e saída são compostas por 7 funções triangulares com as seguintes variáveis linguísticas: PB (positivo grande), PM (positivo médio), PS (positivo pequeno), ZO (zero), NS (negativo pequeno) e NB (negativo grande) que podem ser vistas nas figuras 6, 7, 8, 9 e 10.

As regras para o sistema *Fuzzy* também foram baseadas em [5]. As regras utilizadas para o controlador estão contidas nas tabelas I, II e II.

B. Estratégia de Controle Adaptativo Fuzzy-PID descrita por Equação a Diferenças

Para interagir um sistema de controle *Fuzzy* com um PID discreto em um ambiente de programação em *Python* é necessário adotar uma estratégia de controle por equação a diferenças, pois para cada passo de simulação o controlador

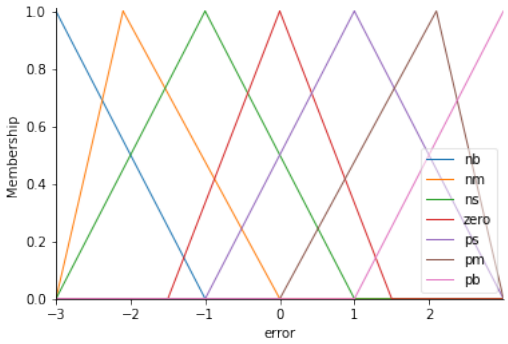

Fig. 6. Função de pertinência da variável erro.

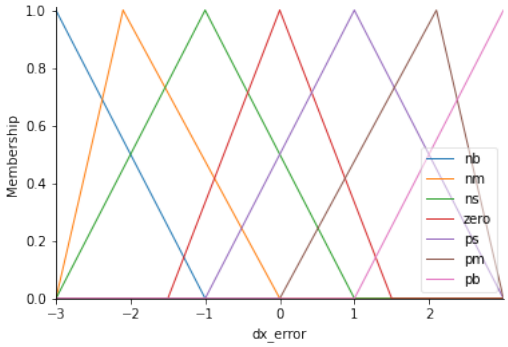

Fig. 7. Função de pertinência da variável derivada do erro.

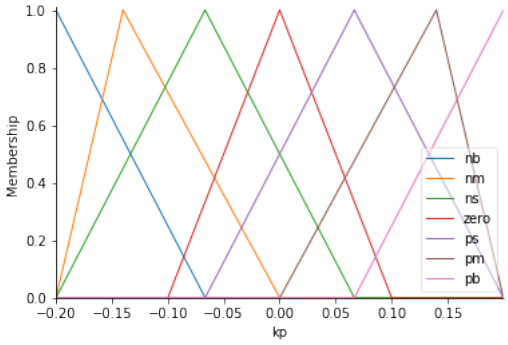

Fig. 8. Função de pertinência da variável Kp.

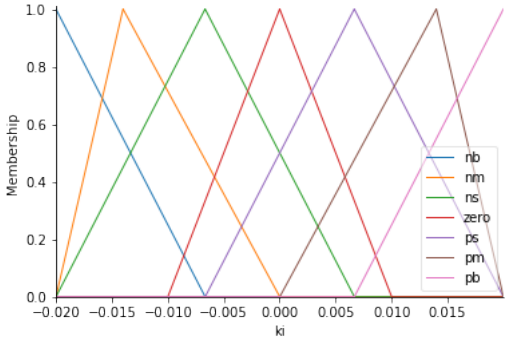

Fig. 9. Função de pertinência da variável Ki.

Fuzzy irá prover ajustes nos ganhos do controlador PID. Isso

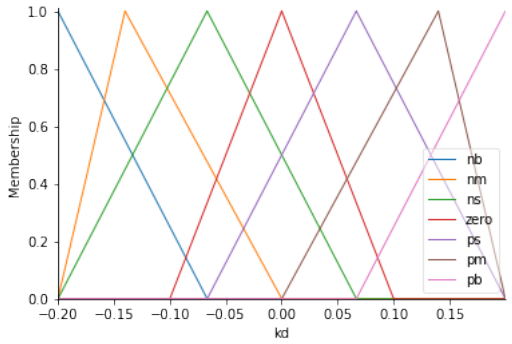

Fig. 10. Função de pertinência da variável Kd.

TABLE I
CONJUNTO DE REGRAS PARA KP

	Error						
dxError	**NB**	**NM**	**NS**	**ZO**	**PS**	**PM**	**PB**
NB	PB	PB	PM	PM	PS	ZO	ZO
NM	PB	PB	PM	PS	PS	ZO	NS
NS	PM	PM	PM	PS	ZO	NS	NS
ZO	PM	PM	PS	ZO	NS	NM	NM
PS	PS	PS	ZO	NS	NS	NM	NM
PM	PZ	ZO	NS	NM	NM	NM	NB
PB	ZO	ZO	NM	NM	NM	NB	NB

TABLE II
CONJUNTO DE REGRAS PARA KI

	Error						
dxError	**NB**	**NM**	**NS**	**ZO**	**PS**	**PM**	**PB**
NB	NB	NB	NM	NM	NS	ZO	ZO
NM	NB	NB	NM	NS	NS	ZO	ZO
NS	NB	NM	NS	NS	ZO	PS	PS
ZO	NM	NM	NS	ZO	PS	PM	PM
PS	NM	NS	ZO	PS	PS	PM	PB
PM	ZO	ZO	PS	PS	PM	PB	PB
PB	ZO	ZO	PS	PM	PM	PB	PB

TABLE III
CONJUNTO DE REGRAS PARA KD

	Error						
dxError	**NB**	**NM**	**NS**	**ZO**	**PS**	**PM**	**PB**
NB	PS	NS	NB	NB	NB	NM	PS
NM	PS	NS	NB	NM	NM	NS	ZO
NS	ZO	NS	NM	NM	NS	NS	ZO
ZO	ZO	NS	NS	NS	NS	NS	ZO
PS	ZO	ZO	ZO	ZO	ZO	ZO	ZO
PM	PB	NS	PS	PS	PS	PS	PB
PB	PB	PM	PM	PS	PS	PS	PB

influencia no esforço de controle que o controlador PID atua na planta. A equação 8 apresenta a equação a diferenças do controlador adaptativo *Fuzzy-PID* por equação a diferenças.

$$u_k = u_{k-1} + e_k \cdot (0.109 + A) - e_{k-1} \cdot (0.1744 + B) + e_{k-2} \cdot (0.07291 + kd(i)) \quad (8)$$

Sendo:
$A = kp(i) + kd(i) + ki(i)$;
$B = kp(i) + 2 \cdot kd(i)$;
$kp(i)$ = O valor de saída kp para cada passo de simulação;
$ki(i)$ = O valor de saída ki para cada passo de simulação;
$kd(i)$ = O valor de saída kd para cada passo de simulação;

Então foi definido o sistema inicial com as funções de pertinências das respectivas variáveis contidas nas figuras 6, 7, 8, 9, e 10. A resposta ao degrau para o sistema inicial está contida na figura VII-B e os parâmetros de desempenho foram percentual de sobressinal 0.02%, tempo de estabilização de 1.9 segundos, erro estacionário de $1,73 \cdot 10^{-9}$.

Fuzzy-PID inicial.png

Fig. 11. Resposta ao degrau para *Fuzzy-PID* adaptativo inicial

Tratando-se do ambiente de programação em *Python* é possível criar a função que gera o controlador adaptativo *Fuzzy-PID* tendo como argumento da função os 5 limites dos universos de discurso das variáveis *Fuzzy*. Além de simular a resposta ao degrau, calcular os parâmetros de desempenho e retorná-los. Com isso, o algoritmo do PSO pode ser programado para variar os 5 parâmetros de entrada desta função, de modo a minimizar o tempo de resposta do sistema.

VIII. RESULTADOS DE OTIMIZAÇÃO

Após os 10 resultados serem obtidos, foram escolhidos dois pelos critérios de melhor tempo de estabilização e percentual de sobressinal. Os melhores resultados obtidos estão na Tabela IV.

TABLE IV
MELHORES RESULTADOS OBTIDOS PELO PROCESSO DE OTIMIZAÇÃO

Resultado	**Percentual de Sobressinal (%)**	**Tempo de Estabilização (s)**	**Erro Estacionário**
Melhor tempo de estabilização	$0,9619$	$0,7$	$4,9327 \cdot 10^{-9}$
Melhor Percentual de Sobressinal	$0,8617$	$0,8$	$4,4015 \cdot 10^{-9}$

Para ambos os controladores os resultados obtidos foram satisfatórios, pois o percentual de sobressinal é menor do que 1 e o erro estacionário é muito próximo a zero.

A comparação dos universos de discurso e as funções de pertinência das entradas (erro e derivada do erro) e das saídas (kp, ki e kd) dos dois controladores podem ser vistas nas figuras 12, 13, 14, 15 e 16.

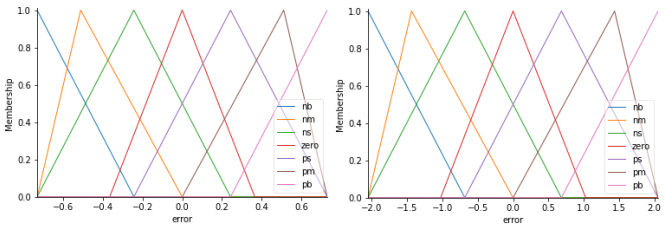

Fig. 12. Comparação das funções de pertinência otimizadas da variável erro de melhor tempo de estabilização e melhor percentual de sobressinal, respectivamente.

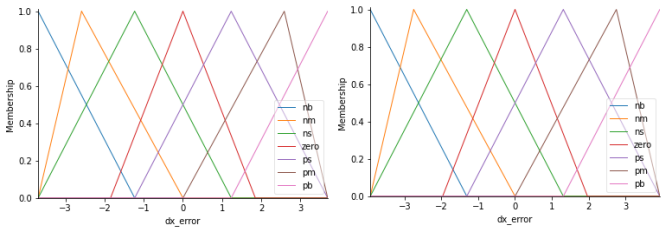

Fig. 13. Comparação das funções de pertinência otimizadas da variável derivada do erro de melhor tempo de estabilização e melhor percentual de Sobressinal, respectivamente.

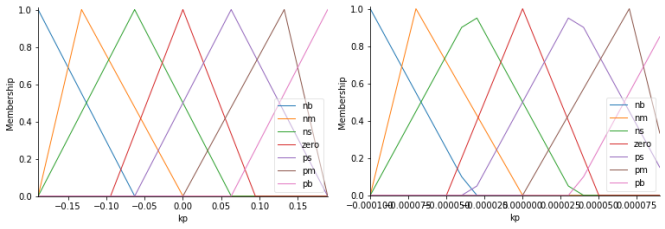

Fig. 14. Comparação das funções de pertinência otimizadas da variável Kp de melhor tempo de estabilização e melhor percentual de sobressinal, respectivamente.

É possível observar que para a variável erro houve uma grande mudança no universo de discurso entre o resultado com melhor tempo de estabilização e o melhor percentual de sobressinal. Já a variável derivada do erro permaneceu com o mesmo universo de discurso em ambos os casos.

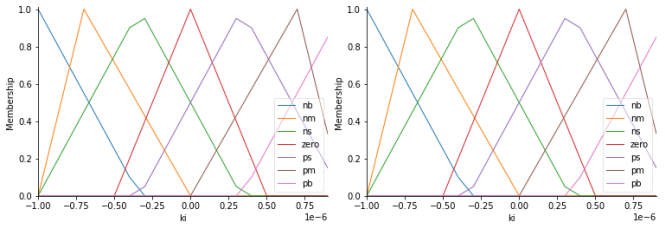

Fig. 15. Comparação das funções de pertinência otimizadas da variável Ki de melhor tempo de estabilização e melhor percentual de sobressinal, respectivamente.

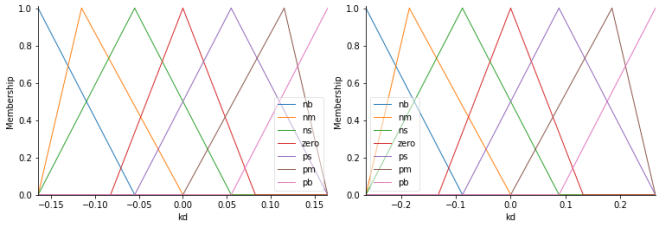

Fig. 16. Comparação das funções de pertinência otimizadas da variável Kd de melhor tempo de estabilização e melhor percentual de sobressinal, respectivamente.

Comparando as figuras observa-se que os ganhos proporcional (kp) e derivado (ki) tiveram uma pequena alteração no seu universo de discurso e nas funções de pertinência, já o ganho integral (ki) permaneceu com o mesmo universo de discurso em ambos os casos.

As respostas ao degrau para o melhor tempo de estabilização e para o melhor percentual de sobressinal obtidos em simulação estão apresentadas nas figuras 17 e 18, respectivamente.

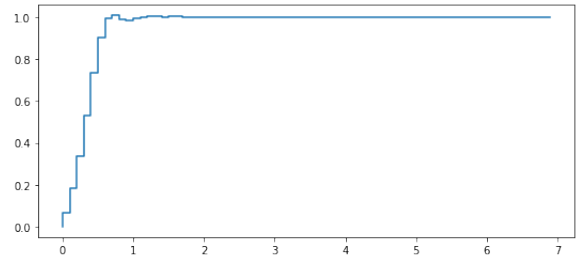

Fig. 17. Resposta ao degrau do controlador *Fuzzy*-PID adaptativo otimizado com o melhor tempo de estabilização.

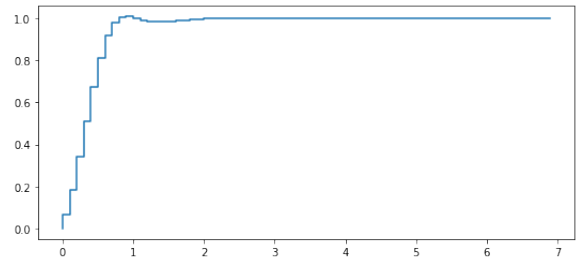

Fig. 18. Resposta ao degrau do controlador *Fuzzy*-PID adaptativo otimizado com o melhor percentual de sobressinal.

É possível observar que a resposta ao degrau em ambos os casos obteve um resultado bem parecido. Através dessas comparações observa-se a minuciosidade na sintonização do sistema *Fuzzy* das entradas e saídas utilizando o algoritmo PSO, com o objetivo de obter o resultado mais satisfatório de acordo com os parâmetros de projeto.

Na Tabela V é apresentada uma comparação entre os valores obtidos para os parâmetros de todos os controladores propostos neste trabalho.

TABLE V
COMPARAÇÃO DOS RESULTADOS OBTIDOS PELOS CONTROLADORES PROPOSTOS

Resultado	**Percentual de sobressinal (%)**	**Tempo de Estabilização (s)**	**Erro Estacionário**
PID Inicial	$12,72$	$2,8$	$9,19 \cdot 10^{-11}$
Fuzzy-PID	$0,02$	$1,9$	$1,73 \cdot 10^{-9}$
Fuzzy-PID Adaptativo (melhor tempo de estabilização)	$0,9619$	$0,7$	$4,9327 \cdot 10^{-9}$
Fuzzy-PID Adaptativo (melhor percentual de sobressinal)	$0,8617$	$0,8$	$4,4015 \cdot 10^{-9}$

Ao comparar o PID inicial ao *Fuzzy-PID* observa-se uma diminuição bem significativa no percentual de sobressinal. Esse parâmetro representa o pico da resposta ao degrau do controlador e em algumas aplicações é importante que ele seja bem baixo, portanto ao incluir o sistema de inferência *Fuzzy*

no projeto do controlador foi possível diminuir o valor desse parâmetro.

Ao comparar o PID Inicial e o *Fuzzy-PID* ao *Fuzzy-PID* adaptativo observa-se uma diminuição no tempo de estabilização. Em ambos os casos, melhor tempo de estabilização e melhor percentual de sobressinal, os parâmetros obtidos tiveram resultados bem parecidos.

Portanto, ao incluir o algoritmo PSO para a otimização das entradas é possível obter resultados muito satisfatórios se comparados ao PID puro e ao *Fuzzy*-PID.

IX. Conclusão

Com esse artigo, foi possível observar através do controle de um ventoinha realizado por um controlador *Fuzzy-PID* que a otimização é essencial para se obter um resultado melhor no controle PID. O resultado obtido foi satisfatório, demonstrando uma performance significativamente melhor do que o controlador PID inicial. Esse resultado pode ser observado através do tempo de estabilização e do percentual de sobressinal. Além disso, o PSO foi satisfatório para otimizar o controlador com os parâmetros pré estabelecidos.

O tipo de controlador projetado nesse artigo foi o controle ótimo. Essa técnica consiste em encontrar o melhor controle para um ponto de operação específico, nesse caso, a inicialização do ventoinha. Para encontrar um controlador com uma boa resposta para vários valores estabelecidos, com distúrbios e sistema dinâmico, seria necessário mudar os parâmetros do PSO para encontrar um controle robusto.

Outra possível mudança nesse projeto é sobre os parâmetros otimizados pelo PSO. Esse projeto otimiza apenas os universos de discurso da variável *fuzzy*, mas o PSO poderia otimizar também os formatos e os pontos das funções de pertinência, as regras do sistema de inferência *fuzzy* e seus pesos além de ganhos nas entradas e saídas do sistema, visando encontrar resultados ainda melhores.

O projeto foi executado no *Google Colaboratory IDE*, um IDE online para programação em *Python* com diversas aplicações online, porém com uma memória RAM limitada, assim, para ser possível rodar uma aplicação mais complexa é necessário implementar o código em um interpretador de *Python* nativo em um computador ou otimizar o código para não extrapolar o limite de memória RAM.

References

[1] C. d. C. C. Neto, L. D. Seixas, and F. C. Corrêa, "Retroactive control applied to a bldc motor," *Vibration Engineering and Technology of Machinery: Proceedings of VETOMAC XV 2019*, p. 233.

[2] R. Goswami and D. Joshi, "Performance review of fuzzy logic based controllers employed in brushless dc motor," *Procedia computer science*, vol. 132, pp. 623–631, 2018.

[3] F. Ferreira, D. Solak Castanho, H. Siqueira, and M. Kaster, "Modelagem e controle discreto digital aplicado a uma ventoinha," *SEA-Seminário de Eletrônica e Automação Ponta Grossa*, p. e.g, 2015.

[4] M. Lopez, P. Ponce, L. Soriano, A. Molina, and J. Rivas, "A novel fuzzy-pso controller for increasing the lifetime in power electronics stage for brushless dc drives," *IEEE Access*, vol. 7, pp. 47 841–47 855, abr 2019.

[5] J. Jin, H. Huang, J. Sun, and Y. Pang, "Study on fuzzy self-adaptive pid control system of biomass boiler drum water," *Journal of Sustainable Bioenergy Systems*, vol. 03, pp. 93–98, 01 2013.

[6] E. B. Costa and G. L. Serra, "Controle pid fuzzy adaptativo evolucionario baseado em especificacao linguística de margem de ganho e fase."

[7] J.-S. Chiou, S.-H. Tsai, and M.-T. Liu, "A pso-based adaptive fuzzy pid-controllers," *Simulation Modelling Practice and Theory*, vol. 26, pp. 49–59, 2012.

[8] Z. Bingül and O. Karahan, "A fuzzy logic controller tuned with pso for 2 dof robot trajectory control," *Expert Systems with Applications*, vol. 38, no. 1, pp. 1017–1031, 2011.

[9] F. M. De Azevedo, L. M. Brasil, and R. C. L. de Oliveira, *Redes neurais com aplicações em controle e em sistemas especialistas*. Visual Books, 2000.

[10] G. Feng, G. Lu, D. Sun, and S. Zhou, "A model reference adaptive control algorithm for fuzzy dynamic systems," in *Proceedings of the 4th World Congress on Intelligent Control and Automation (Cat. No. 02EX527)*, vol. 4. IEEE, 2002, pp. 3242–3246.

[11] K.-M. Choi, S.-W. Cho, D.-O. Kim, and I.-W. Lee, "Active control for seismic response reduction using modal-fuzzy approach," *International Journal of Solids and Structures*, vol. 42, no. 16-17, pp. 4779–4794, 2005.

[12] V. Geetha and S. Thangavel, "Performance analysis of direct torque controlled bldc motor using fuzzy logic," *International Journal of Power Electronics and Drive Systems*, vol. 7, no. 1, p. 144, 2016.

[13] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.

[14] S. Gharghory and H. Kamal, "Modified pso for optimal tuning of fuzzy pid controller," 2013.

[15] S. Bouallègue, J. Haggege, M. Ayadi, and M. Benrejeb, "Pid-type fuzzy logic controller tuning based on particle swarm optimization," *Engineering Applications of Artificial Intelligence*, vol. 25, pp. 484–493, 04 2012.

[16] N. Rooy. (2016, aug) Particle swarm optimization from scratch with python. [Online]. Available: https://nathanrooy.github.io/posts/2016-08-17/simple-particle-swarm-optimization-with-python/

Received: 11 February 2021;
Accepted: 20 April 2021;
Published: 22 May 2021;

PSO-Optimized Adaptive *Fuzzy-PID* Controller for Fan Control on a Teaching Bench

***Abstract*—In this work, an adaptive Fuzzy-PID controller optimized by particle swarm (PSO) was designed, where it was used to vary the limits of the speech universes of the input and output variables of the fuzzy control system, maintaining the respective proportional transfer functions, in order to obtain the shortest possible stabilization time of the fan contained in the PICGenius board, and driven by a PWM and the PIC 18F4550 Microcontroller. The results were compared to the conventional PID controller, where it was possible to observe that the Fuzzy-PID controller provided improved results in the controlled system.**

***Keywords*—PSO, Fuzzy Control Systems, Artificial Intelligence.**