UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS
UNIDADE ACADÊMICA DE GRADUAÇÃO
CURSO DE ENGENHARIA ELÉTRICA

EDGAR DIAS DE OLIVEIRA

SINTONIA DE CONTROLADORES PID ATRAVÉS DE META-HEURÍSTICAS

São Leopoldo
2019

EDGAR DIAS DE OLIVEIRA

SINTONIA DE CONTROLADORES PID ATRAVÉS DE META-HEURÍSTICAS

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de Bacharel em Engenharia Elétrica, pelo Curso de Engenharia Elétrica da Universidade do Vale do Rio dos Sinos - UNISINOS

Orientador: Prof. Ms. João Olegário de Oliveira de Souza

São Leopoldo

2019

Dedico este trabalho à Sofia. Pois nada me motiva mais a ser uma pessoa melhor e a tornar o mundo um lugar melhor do que minha filha.

AGRADECIMENTOS

Agradeço aos meus pais, Carlos Antônio Soares de Oliveira e Teresinha Aparecida dos Santos Dias, por todo o apoio dado a mim durante toda a minha vida, as dificuldades enfrentadas e os obstáculos superados para garantirem um bom futuro e uma boa educação aos filhos, e por sempre acreditarem em mim. Agradeço também aos meus irmãos, Alan Dias Klazer e Aline Dias de Oliveira que contribuíram em tudo aquilo que lhes foi possível para a realização deste trabalho.

Agradeço também aos meus professores pela dedicação no ensino ao longo de todo o curso, especialmente ao meu orientador, João Olegário de Oliveira de Souza, por ter sido sempre muito presente e muito solícito, contribuindo de maneira significativa para a realização deste trabalho.

Agradeço aos meus amigos que também contribuíram, seja com a discussão de ideias sobre o trabalho, seja por propiciar momentos de lazer após incansáveis horas de estudo.

“Somewhere, something incredible is waiting to be known.”

Carl Sagan

RESUMO

Este trabalho apresenta a metodologia empregada para a sintonia de controladores PID utilizando meta-heurísticas. É feita uma comparação entre quatro meta-heurísticas, *Genetic Algorithm* – GA, *Particle Swarm Optimization* – PSO, *Ant Colony Optimiization* – ACO, e *Artificial Bee Colony* – ABC, com o objetivo de minimizar os parâmetros da resposta ao degrau do sistema no domínio do tempo, tempo de subida, tempo de acomodação e sobressinal.

A planta proposta para aplicação das meta-heurísticas é o controle de posição de um motor de corrente contínua com ímãs permanentes utilizando um sistema SISO (Single Input Single Output) em cascata, com três malhas de controle, uma malha interna para a corrente, uma malha intermediária para a velocidade e uma malha externa para a posição do motor CC.

Foram realizadas simulações a fim de comparar e encontrar os parâmetros ótimos para o problema proposto. As meta-heurísticas também foram submetidas à sintonia dos três controladores de duas maneiras, primeiramente de forma sequencial e, após, de forma simultânea, para comparação dos resultados obtidos.

Todas as meta-heurísticas se mostraram satisfatórias na sintonia dos controladores PID, e a sintonia simultânea, mesmo sendo a mais complexa para os algoritmos, foi a que encontrou o menor custo para a função objetivo proposta, reduzindo o tempo de subida de 443,3 ms para 335,83 ms e o tempo de acomodação de 977,2 ms para 715,5 ms. O tempo de simulação das meta-heurísticas também foi medido, e foi constatado que algumas meta-heurísticas podem levar menos da metade do tempo que outras para a mesma sintonia. Os resultados são discutidos ao final do trabalho.

Palavras-chave: PID. Meta-heurística. *Genetic Algorithm*. *Particle Swarm Optimization*. *Ant Colony Optimization*. *Artificial Bee Colony*.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE SIGLAS

ABC	*Artificial Bee Colony* (Colônia Artificial de Abelhas)
ACO	*Ant Colony Optimization* (Otimização por Colônia de Formigas)
AVR	*Automatic Voltage Regulator* (Regulador Automático de Tensão)
BA	*Bat Algorithm* (Algoritmo dos Morcegos)
CA	Corrente Alternada
CC	Corrente Contínua
CLP	Controlador Lógico Programável
DDC	*Direct Digital Control* (Controle Digital Direto)
EEE	Erro em Estado Estacionário
FFA	*FireFly Algorithm* (Algoritmo dos Vagalumes)
FOPID	*Fractional Order PID* (PID de Ordem Fracionária)
GA	*Genetic Algorithm* (Algoritmo Genético)
IAE	*Integral of the Absolute Error* (Integral do Erro Absoluto)
ISE	*Integral of the Squared Error* (Integral do Erro Quadrático)
ITAE	*Integral of Time multiplied by Absolute Error* (Integral do Erro Absoluto Ponderado pelo Tempo)
ITSE	*Integral of Time multiplied by Squared Error* (Integral do Erro Quadrático Ponderado pelo Tempo)
LQR	*Linear Quadratic Regulator* (Regulador Linear Quadrático)
MP	*Maximum Peak* (Pico Máximo ou Sobressinal)
NP	*Nondeterministic Polynomial time* (tempo Polinomial Não-Determinístico)
NPC	*Nondeterministic Polynomial time Complete* (tempo Polinomial Não-Determinístico Completo)
P	Polynomial time (tempo Polinomial)
PID	Controlador Proporcional – Integral – Derivativo
PV	*Process Variable* (Variável de Processo)
PSO	*Particle Swarm Optimization* (Otimização por Enxame de Partículas)
PWM	*Pulse Width Modulation* (Modulação por Largura de Pulso)
TA	Tempo de Assentamento
TS	Tempo de Subida

SUMÁRIO

1 INTRODUÇÃO

Os sistemas de controle são uma parte fundamental da ciência moderna, estando presentes desde processos simples, como o controle de temperatura de uma usina, até mais complexos, como satélites que orbitam o espaço. Além desses processos, o controle de processos é encontrado também na natureza, como o nosso corpo faz o controle de açúcar no sangue, ou os níveis de adrenalina no nosso corpo em situações de estresse (NISE, 2013).

O processo de seleção dos parâmetros do controlador PID a fim de otimizar o desempenho requerido por determinado processo é conhecido como sintonia do controlador PID (OGATA, 2011). Há métodos convencionais e modernos para a sintonia dos controladores, o mais conhecido é o método de Ziegler & Nichols, o qual propôs um método para calcular os parâmetros do controlador mesmo sem conhecer os parâmetros do processo controlado.

Quando buscamos otimizar um processo, como minimizar custos ou maximizar lucros, torna-se interessante fazer uso de métodos de otimização, como são as meta-heurísticas, algoritmos baseados em inteligência artificial que têm se mostrado eficazes para sintonizar controladores PID. Neste trabalho, quatro meta-heurísticas serão estudadas e testadas para a sintonia de um sistema em cascata com três controladores para controle de posição de um motor de corrente contínua. Primeiro, cada controlador será sintonizado individualmente, de maneira sequencial, e após, serão sintonizados de maneira simultânea, sendo um desafio muito maior para as meta-heurísticas.

1.1 TEMA

O presente trabalho trata do estudo e da utilização de meta-heurísticas para a sintonia de controladores PID (Proporcional-Integral-Derivativo). Serão apresentadas quatro meta-heurísticas para o cálculo de otimização dos parâmetros do controlador PID, Algoritmo Genético (*Genetic Algorithm* – GA), Otimização por Enxame de Partículas (*Particle Swarm Optimization* – PSO), Otimização por Colônia de Formigas (*Ant Colony Optimiization* – ACO), e Colônia Artificial de Abelhas (*Artificial Bee Colony* – ABC).

É feita a modelagem matemática de um motor de corrente contínua para ser utilizado como modelo para aplicação das meta-heurísticas, e são realizadas simulações para o controle da posição do motor CC através de um sistema em cascata com três controladores.

1.2 DELIMITAÇÃO DO TEMA

Estudo e análise de meta-heurísticas para a sintonia de controladores PID para controle de posição de um motor de corrente contínua através de simulações.

1.3 PROBLEMA

O controlador PID é a estratégia de controle mais utilizada atualmente, visto que cerca de 95% dos problemas de controle o utiliza (ÅSTRÖM e HÄGGLUND, 1995). Além das malhas de controle que operam sem controle algum, encontramos malhas que operam distantes do seu ponto ótimo, precisando ser ajustadas. Com o avanço de estudos em métodos de otimização baseados em meta-heurísticas, os métodos convencionais de sintonia estão se tornando cada vez mais obsoletos. Este trabalho visa a aprofundar os estudos de meta-heurísticas para a sintonia de controladores PID.

1.4 OBJETIVOS

1.4.1 Objetivo Geral

O objetivo deste trabalho é realizar comparações entre meta-heurísticas a fim de encontrar uma boa solução para a sintonia de controladores PID em malhas de controle. Diversos estudos têm mostrado que meta-heurísticas apresentam resultados muito satisfatórios quando comparados com os métodos convencionais presentes nas bibliografias e mais utilizados atualmente no mercado.

1.4.2 Objetivos Específicos

a) Implementar as meta-heurísticas ABC, ACO, GA e PSO para a sintonia de controladores PID;

b) Realizar o ajuste dos parâmetros das meta-heurísticas a fim de encontrar os que melhor se ajustam à planta proposta;
c) Analisar comparativamente os resultados obtidos pelas meta-heurísticas na configuração SISO (*Single Input Single Output*) e cascata em um motor CC (corrente contínua);
d) Realizar a sintonia sequencial e simultânea dos controladores do sistema em cascata.

1.5 JUSTIFICATIVA

Há uma ampla aplicação de controle PID na indústria, devido à sua fácil implementação, baixo custo, versatilidade e capacidade de alterar o regime transitório e permanente dos sistemas controlados (ÅSTRÖM e HÄGGLUND, 1995). A maioria dos processos automatizados por CLP's (Controlador Lógico Programável, hoje, possui algum algoritmo PID implementado em sua malha de controle. A expansão da indústria, acompanhando o avanço da tecnologia, exige controladores cada vez mais robustos, possuindo critérios de desempenho diferentes para cada tipo de sistema. As meta-heurísticas são capazes de otimizar diversos critérios de desempenho de acordo com a necessidade do sistema, se tornando muito mais atrativas do que os métodos convencionais para sistemas complexos.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo é apresentada uma introdução aos controladores PID, seus princípios de funcionamento e a função de cada uma de suas ações, proporcional, integral e derivativa. São descritos os critérios de desempenho mais utilizados para controladores PID e o método de controle pelo sistema em cascata. Também é apresentada uma visão geral sobre métodos de otimização, mínimos e máximos locais e globais e função objetivo. São apresentadas as meta-heurísticas Colônia Artificial de Abelhas, Otimização por Colônia de Formigas, Algoritmo Genético e Otimização por Enxame de Partículas, suas inspirações e as equações que descrevem seu funcionamento. Ao final são apresentados o princípio de funcionamento do motor de corrente contínua, bem como sua modelagem matemática.

2.1 CONTROLADOR PID

Por volta do ano de 1750, ganhos proporcionais através de realimentação foram utilizados para regular a velocidade de moinhos de vento. Em 1788, James Watt utilizou um sistema semelhante para fazer o controle de velocidade de motores a vapor. Somente algum tempo depois se passou a utilizar ação proporcional e integral (ÅSTRÖM e HÄGGLUND, 1995).

O controlador PID, na forma como é conhecido hoje, surgiu entre os anos 1915 e 1940, coincidindo com o desenvolvimento de grandes companhias, como Bristol, Fisher, Taylor Instrument, etc. A ação integral era geralmente chamada de *reset* automático, já que ela substituiu uma ferramenta utilizada na ação proporcional conhecida como *reset* manual para obter o estado estacionário correto. Maxwell, em 1968, foi quem escreveu uma análise matemática para uma máquina a vapor com regulador, a qual mostrou claramente a diferença entre a ação proporcional e integral.

Em 1935, Ralph Clarridge, da Taylor Instrument, apresentou um controlador com ação derivativa. A ideia vinha sendo discutida desde a década de 1920, sobre uma ação capaz de antecipar erros futuros, porém pouco aceita. A ação passou a ser chamada de *pre-act*.

A partir de então, o controlador PID avançou junto com o desenvolvimento acelerado de novas tecnologias, passando por controladores pneumáticos na década de 1940, até serem substituídos por componentes eletrônicos, com o surgimento dos amplificadores operacionais, em 1950. Em 1960 foi a era dos controladores computacionais, com o Controle Digital Direto (*Direct Digital Control* – DDC), onde os computadores controlavam diretamente o atuador. Em 1970 surgiram os microprocessadores, os quais conduziram o desenvolvimento de sistemas de controle distribuído para controle de processo. E com o aumento do poder computacional dos microprocessadores, foi possível introduzir sintonia e adaptação para controladores de malha simples, o que seguiu até a década de 1990 (ÅSTRÖM e HÄGGLUND, 1995).

Controladores PID, e suas diversas adaptações, são provavelmente os controladores mais utilizados em todo o mundo (JOHNSON e MORADI, 2005). Até mesmo as plantas mais avançadas possuem como base de controle principal controladores PID. Os controladores PID têm uma longa história, tendo sobrevivido desde a era analógica até os computadores digitais, tendo sido o primeiro controlador a ter produção em massa para atender ao grande volume de mercado demandado pelas indústrias.

Segundo Johnson e Moradi (2005), o controlador PID continua importante até os dias de hoje devido a três fatores: passado de sucesso; ampla disponibilidade; e simplicidade de uso.

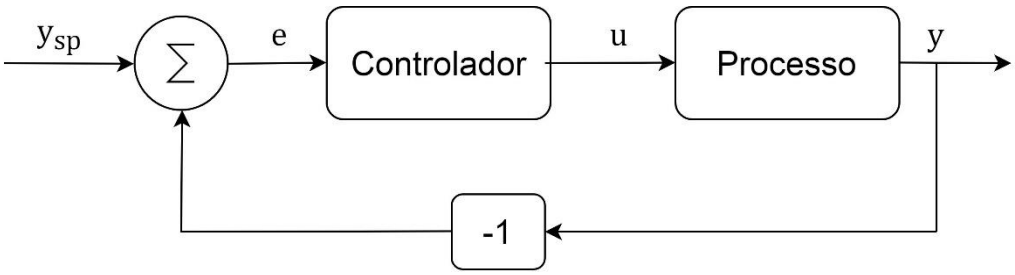
2.1.1 Estrutura

Os componentes principais de um sistema são o controlador e o processo a ser controlado, conforme é mostrado na Figura 1. O processo fornece uma resposta de saída y para uma entrada y_{sp} (*set point*) do sistema. Um sistema de malha fechada (*feedback system*) é utilizado para comparar a saída do sistema com o valor desejado da entrada (*set point*). Quando há uma diferença entre o valor desejado e o valor real da saída do sistema, um sinal de erro *e* é gerado, conforme a Equação 3.1.

$$e = y_{sp} - y \tag{3.1}$$

O sinal de erro gerado é enviado ao controlador, que busca corrigir esse erro, enviando um sinal u para o processo. Pode-se notar que quando a saída do sistema é igual à entrada, ou seja, $y = y_{sp}$, o sinal de erro é nulo e, portanto, o controlador não toma nenhuma ação. Já quando a saída do sistema é maior ou menor que a entrada, um sinal de erro é enviado ao controlador e a saída do controlador então envia um sinal, ajustando a saída do processo (ÅSTRÖM e HÄGGLUND, 1995).

Figura 1 - diagrama de blocos simples de um sistema em malha fechada



Fonte: Elaborado pelo autor.

A função de transferência que relaciona a saída com a entrada do controlador é dada pela equação 3.2:

$$u(t) = K\left[e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d\frac{d}{dt}e(t)\right] \tag{3.2}$$

onde u é a variável de controle, e é o sinal de erro, e os parâmetros do controlador são o ganho proporcional K, o tempo integral T_i e o tempo derivativo T_d (ÅSTRÖM e HÄGGLUND, 1995).

Essa equação mostra as três componentes do controlador PID, a ação proporcional, integral, e derivativa, as quais serão explicadas a seguir. Pode-se reescrever essa mesma equação no domínio das frequências, utilizando a Transformada de Laplace, como mostrado na equação 3.3.

$$U(s) = K\left(1 + \frac{1}{sT_i} + sT_d\right) \tag{3.3}$$

Porém, a forma mais comum de se representar a função de transferência do controlador PID é fazendo as seguintes modificações (ÅSTRÖM e HÄGGLUND, 1995):

$$K_p = K$$

$$K_i = \frac{K}{T_i}$$

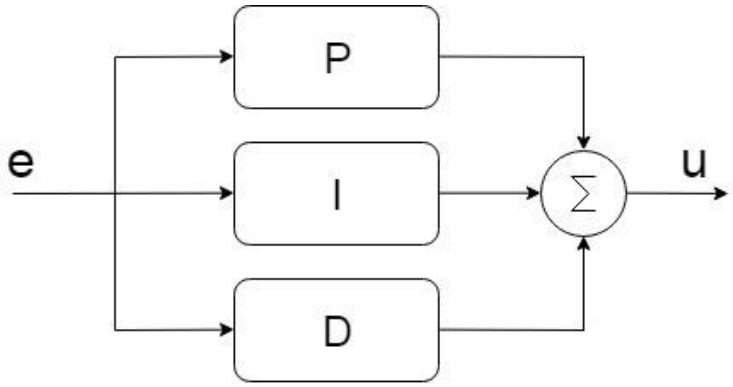
$$K_d = KT_d$$

Dessa forma, se tem a equação 3.4.

$$G_C(s) = K_p + \frac{K_i}{s} + K_d s \tag{3.4}$$

Esse tipo de representação pode causar confusão com a equação 3.3, porém a equação 3.4 é muito útil em cálculos analíticos porque os parâmetros aparecem linearmente. Outra vantagem é que é possível obter ação proporcional, integral ou derivativa pura com valores finitos dos parâmetros (ÅSTRÖM e HÄGGLUND, 1995). A estrutura simplificada de um controlador PID é mostrada na Figura 2.

Figura 2 - Estrutura simplificada de um controlador PID



Fonte: Elaborado pelo autor.

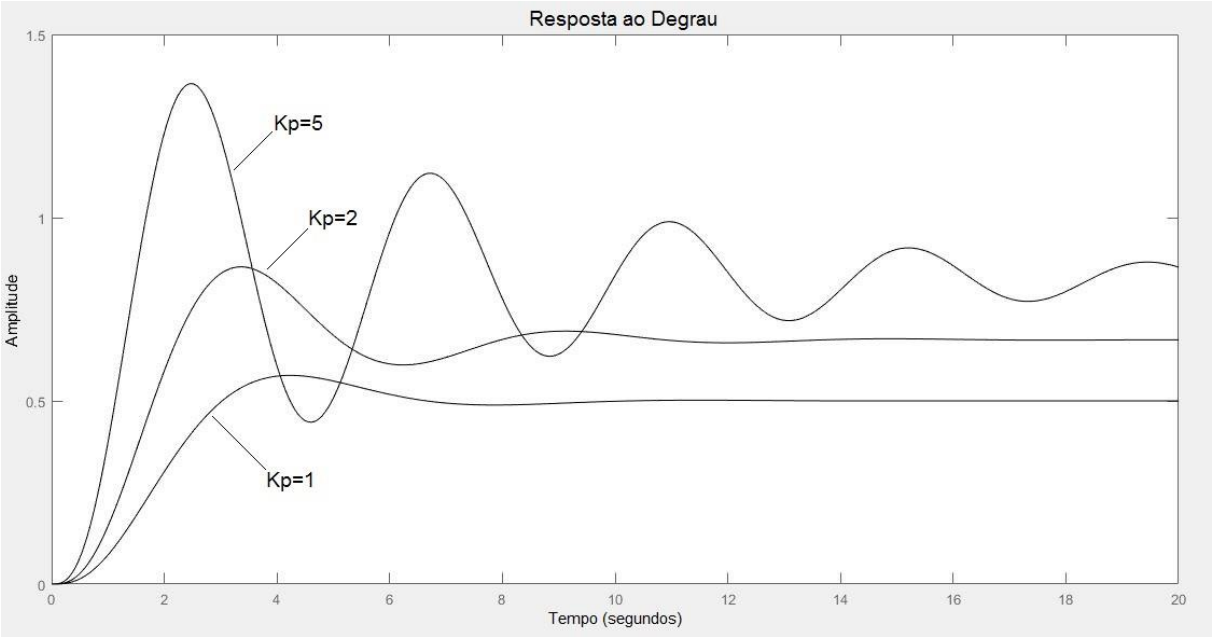
2.1.1.1 Ação proporcional

A ação proporcional simplesmente faz com que a saída de controle seja proporcional ao erro, conforme a equação 3.5.

$$u(t) = Ke(t) + u_b \tag{3.5}$$

O sinal u_b é um erro do sistema (bias), o qual deve ser ajustado para reduzir o erro em estado estacionário. Quando o erro do sistema é zero, se tem que $u(t) = u_b$. Assim, o valor de u_b pode ser ajustado manualmente para eliminar o erro em estado estacionário. A ação proporcional reduz, mas não zera, o erro em estado estacionário e faz com que a resposta seja mais rápida, porém, um incremento muito grande do ganho proporcional pode resultar em uma oscilação muito grande do sistema. A Figura 3 mostra a resposta de amplitude pelo tempo do efeito do aumento do ganho proporcional em um sistema aplicando diferentes valores de K_P.

Figura 3 - Resposta a um degrau unitário de um sistema em malha fechada com diferentes valores de ganhos proporcionais



Fonte: Elaborado pelo autor.

2.1.1.2 Ação integral

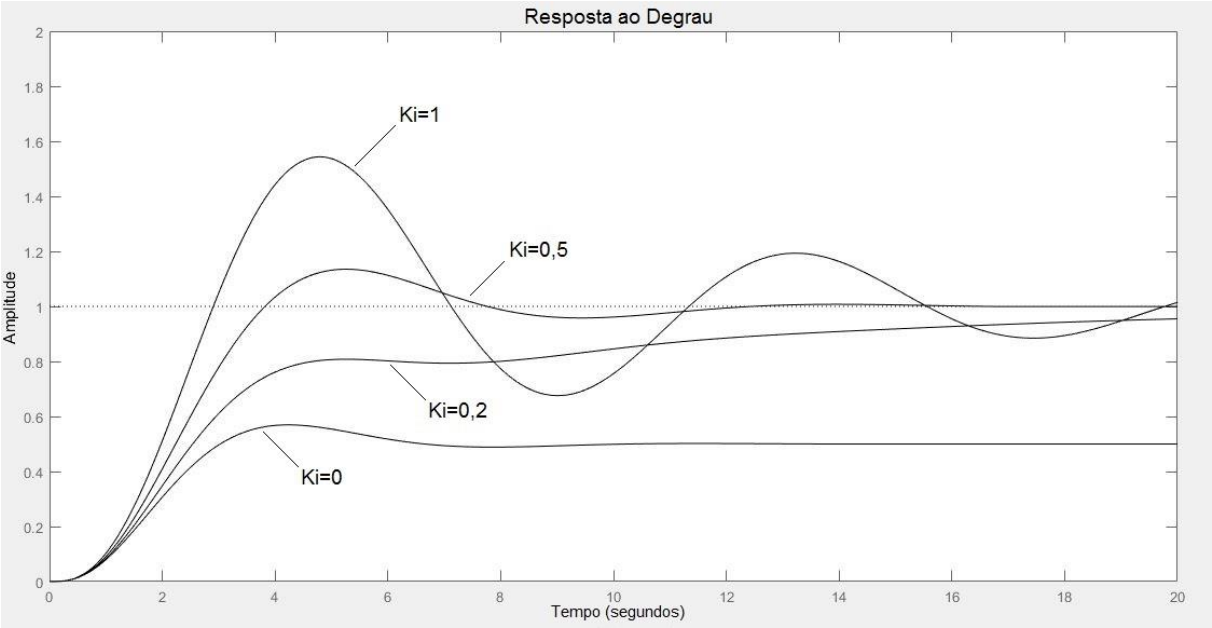
A ação integral tem por objetivo zerar o erro em estado estacionário, ou seja, fazer com que a saída seja igual à entrada. Por menor que seja o erro em estado estacionário, ele sempre irá gerar um sinal de controle através da ação integral, que guiará o sinal de saída para o *set point*. A partir da equação 3.2, considerando um sinal de controle constante e um sinal de erro constante, se obtém a equação 3.6, a qual mostra que, enquanto o erro for diferente de zero, o sinal de controle não assumirá um valor constante. Sendo assim, todo controlador com ação integral sempre terá erro estacionário nulo (ÅSTRÖM e HÄGGLUND, 1995).

$$u_0 = K(e_0 + \frac{e_0}{T_i}t) \tag{3.6}$$

Com a utilização da ação integral, o *bias* inserido no controle proporcional e ajustado manualmente se torna desnecessário, a ação integral passou a funcionar como um ajuste automático do valor do *bias* que zera o erro em estado estacionário.

A Figura 4 mostra a resposta de amplitude pelo tempo do comportamento de um processo com um controlador PI, considerando o ganho proporcional em 1, alterando o valor da ação integral K_I.

Figura 4 - Resposta a um degrau unitário de um sistema em malha fechada com diferentes valores da ação integral.



Fonte: Elaborado pelo autor.

2.1.1.3 Ação derivativa

A ação derivativa tem como principal objetivo aumentar a estabilidade do sistema. Na maioria dos sistemas dinâmicos, a informação de uma mudança na variável de controle levará um tempo até ser notada na variável de processo, isso faz com que sempre se corrija o erro com atraso. A ação derivativa atua como se pudesse predizer o erro que virá para corrigi-lo. Pode-se dizer que um controlador PD corrige proporcionalmente o erro que virá. Isso é feito extrapolando o erro pela sua curva tangente (ÅSTRÖM e HÄGGLUND, 1995).

A equação 3.7 mostra o comportamento do controlador PD e a equação 3.8 mostra uma aproximação por série de Taylor para a estimativa do tempo T_d à frente.

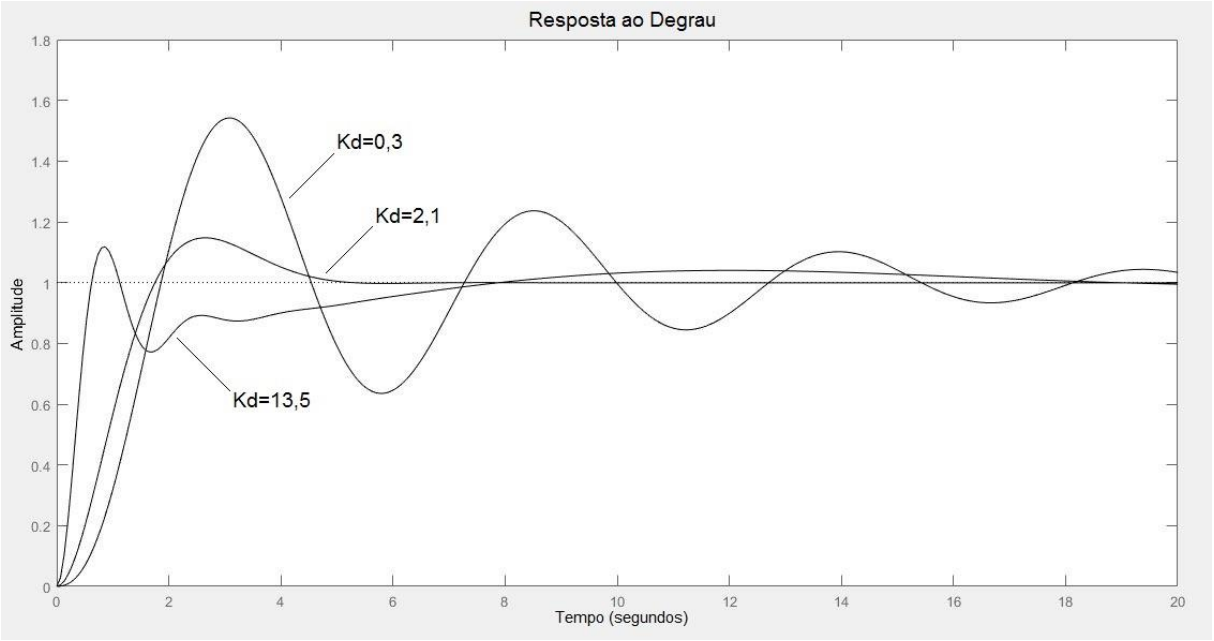
$$u(t) = K[e(t) + T_d \frac{d}{dt} e(t)] \tag{3.7}$$

$$e(t + T_d) = e(t) + T_d \frac{d}{dt} e(t) \tag{3.8}$$

O sinal de controle responde, então, a uma aproximação do erro no tempo $(t + T_d)$. A Figura 5 mostra a resposta de amplitude pelo tempo do efeito da ação derivativa para um sistema, variando os valores da ação derivativa K_d para um

controlador PID. O valor do ganho proporcional é 3 e a ação integral é 1,5, constantes em todos os casos.

Figura 5 - Resposta a um degrau unitário de um sistema em malha fechada com diferentes valores da ação derivativa



Fonte: Elaborado pelo autor.

2.1.2 Sintonia

O controlador PID utilizada três parâmetros para melhorar a resposta de sistemas, proporcional, integral e derivativo, onde cada ação age de uma maneira sobre a resposta. A escolha dos três parâmetros do controlador é chamada sintonia.

Segundo Johnson e Moradi (2005), os dois passos para a sintonia do controlador PID são a escolha da estrutura do controlador e a determinação dos parâmetros.

2.1.2.1 Escolha da Estrutura

O tipo de controlador depende diretamente do modelo e da complexidade da planta a ser controlada, podendo ser utilizado um controlador do tipo P, PI, PID ou PD, esse último menos comum. A Tabela 1 mostra a ação de cada parâmetro a ser escolhido para a sintonia do controlador PID na resposta transitória e em estado estacionário do sistema.

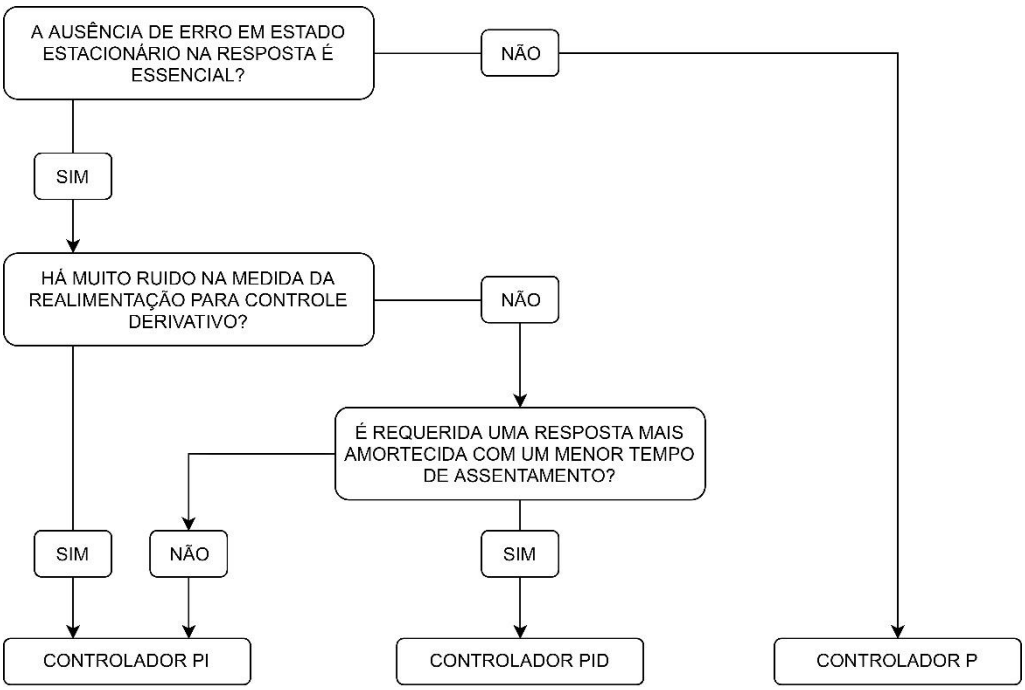
Tabela 1 - Efeitos da sintonia dos parâmetros do controlador PID

	Resposta transitória	Estado estacionário
P	Aumentar o valor de Kp aumenta a velocidade de resposta	Aumentar o valor de Kp reduz mas não elimina o erro de estado estacionário
I	Aumentar o valor de Ki possui uma grande variedade de tipos de resposta	Aumentar o valor de Ki elimina o erro em estado estacionário do sistema
D	Aumentar o valor de Kd possui uma grande variedade de tipos de resposta e pode ser utilizado para amortecer a resposta	Aumentar o valor de Kd não possui efeito na resposta em estado estacionário

Fonte: Adaptado de Johnson e Moradi (2005).

A Figura 6 mostra um fluxograma da escolha de estrutura do controlador baseado nos efeitos da Tabela 1.

Figura 6 - Fluxograma de decisão da estrutura do controlador PID



Fonte: Adaptado de Johnson e Moradi (2005).

Seguindo o fluxograma, é possível fazer a escolha da estrutura do controlador, utilizando um controlador PI quando não pode haver erro em estado estacionário, e um PID quando é necessário amortecer a resposta. Já nos casos mais simples, um controlador P é adequado.

2.1.2.2 Escolha dos Parâmetros

Inicialmente a sintonia do controlador era feita empiricamente, mas isso só era possível porque a indústria ainda lidava com processos simples, onde esse método de sintonia gerava uma resposta aceitável. Em 1942 foram sugeridos dois métodos para a sintonia dos parâmetros por Ziegler & Nichols, pioneiros nos métodos de sintonia. Muitos métodos de sintonia surgiram depois, como o método dos relés de Åström & Hägglund (1985), e também muitos estudos e novas versões do método de Ziegler & Nichols (JOHNSON e MORADI, 2005).

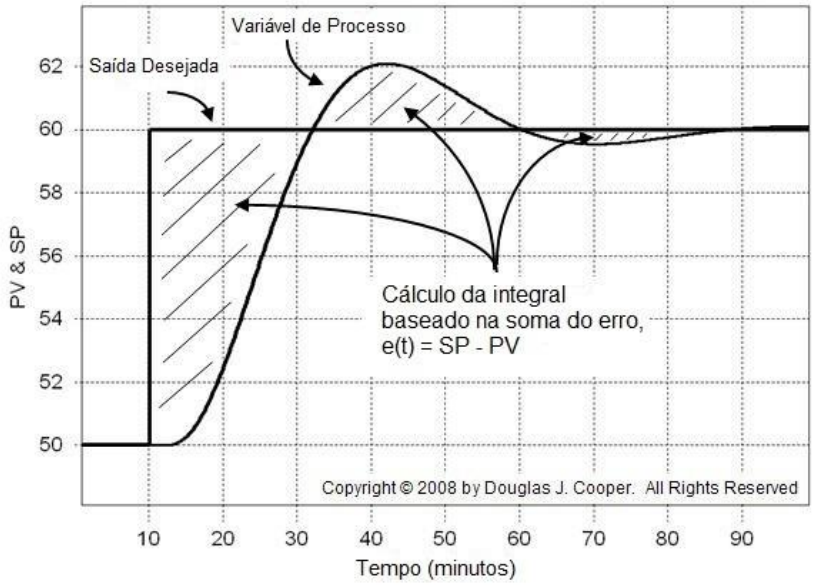
Atualmente há muitos estudos na utilização de meta-heurísticas para a sintonia de controladores PID, com resultados satisfatórios, como Saad, Jamaluddin e Darus (2012) e Silva (2005).

2.1.3 Critérios de Desempenho

Os métodos de sintonia de PID sempre buscam encontrar parâmetros ótimos para a resposta do sistema, sejam eles o sobressinal, o tempo de subida ou o erro em estado estacionário. Nesses casos, quanto menor o valor, mais próximo do ideal se está (SILVA, 2005). Em outras palavras, mede-se o desempenho do PID no quanto ele é capaz de minimizar esses efeitos indesejáveis. Muitas vezes, a planta na qual se está trabalhando exige uma resposta rápida, ou um sobressinal pequeno ou nulo. Dessa forma, o desenvolvedor precisa ser capaz de trabalhar com esses critérios.

Entre os critérios de desempenho dos controladores PID mais conhecidos estão os baseados na integral do erro do sistema. A seguir serão apresentados o *IAE*, o *ITAE*, o *ISE* e o *ITSE*. A Figura 7 mostra a representação gráfica da integral do erro do sistema, a parte sombreada do gráfico, sendo a diferença entre a saída desejada (*Set Point - SP*) e a variável de processo (*Process Variable - PV*).

Figura 7 - Representação gráfica da integral do erro do sistema



Fonte: Adaptado de Cooper (2008).

Além de minimizar os efeitos indesejados da resposta do sistema, um bom critério de desempenho deve ter uma boa seletividade, ou seja, o critério de desempenho deve sofrer grandes variações para pequenas variações dos parâmetros do controlador, para que seja fácil distinguir o ponto ótimo de pontos não ótimos (SILVA, 2005).

2.1.3.1 Integral do Erro Absoluto - IAE

Integral do Erro Absoluto (*Integral of the Absolute Error*). Esse índice é bastante útil e também fácil de ser implementado. Apresenta pesos iguais para erros independente do tempo em que ocorrem e apresenta baixa seletividade. (SILVA, 2005). O IAE é calculado pela equação 3.9:

$$IAE = \int_0^T |e(t)|dt \tag{3.9}$$

onde T é o tempo suficiente para o sistema entrar em regime estacionário.

2.1.3.2 Integral do Erro Absoluto ponderada pelo Tempo - ITAE

Integral do Erro Absoluto ponderada pelo Tempo (Integral of Time multiplied by Absolute Error). Diferentemente do IAE, o ITAE perdoa erros com tempo curto, os

quais são mais elevados, e penaliza erros com tempo elevado, Dentre todos os índices é o que apresenta melhor seletividade (DORF e BISHOP, 2001). O ITAE é calculado pela equação 3.10.

$$ITAE = \int_0^T t|e(t)|dt \tag{3.10}$$

2.1.3.3 Integral do Erro Quadrático - ISE

Integral do Erro Quadrático (Integral of the Squared of the Error). Elevando o erro ao quadrado, perdoa erros pequenos e pune erros grandes. O valor do índice varia consideravelmente com o amortecimento do sistema, de subamortecido para superamortecido. Para reduzi-lo, é preciso ter um amortecimento equilibrado (DORF e BISHOP, 2001). Apresenta uma curva de seletividade com mínimo quase indefinido. O ISE é calculado pela equação 3.11.

$$ISE = \int_0^T e^2(t)dt \tag{3.11}$$

2.1.3.4 Integral do Erro Quadrático ponderado pelo Tempo - ITSE

Integral do Erro Quadrático ponderado pelo Tempo (Integral of Time multiplied by the Squared Error). O ITSE é calculado pela equação 3.12.

$$ITSE = \int_0^T te^2(t)dt \tag{3.12}$$

Esse índice penaliza fortemente erros grandes associados a tempos elevados, e perdoa erros pequenos em tempos curtos. Na definição dos parâmetros, possui uma seletividade melhor que o ISE (FERMINO, 2014).

2.1.4 Critérios de Robustez

Ao sintonizar um controlador, além de atender as especificações do sistema, o projetista deve se preocupar também com a robustez do sistema. A robustez mede

o quanto o sistema é capaz de sofrer alterações nos seus parâmetros sem se tornar instável, e é medida por dois critérios: Margem de Ganho e Margem de Fase.

Nise (2013) define esses critérios da seguinte maneira:

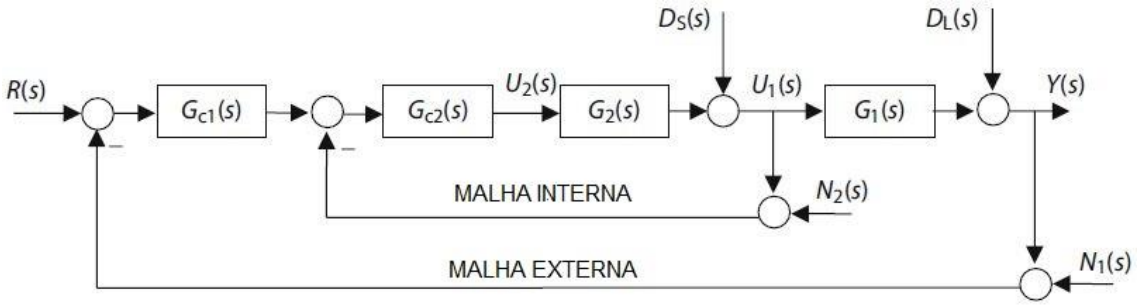
> Margem de fase A quantidade de defasagem adicional em malha aberta necessária no ganho unitário para tornar o sistema em malha fechada instável.
>
> Margem de ganho A quantidade adicional de ganho em malha aberta, expressa em decibéis (dB), necessária na defasagem de 180° para tornar o sistema em malha fechada instável.

Esses dois parâmetros obedecem aos critérios de estabilidade de Nyquist. Quanto maior a margem de ganho e de fase, mais robusto é o sistema, e margens de ganho ou de fase negativas indicam que o sistema é instável (NISE, 2013). Segundo Ogata (2011), um sistema robusto deve ter margem de fase entre 30° e 60°, e margem de ganho maior que 6 dB.

2.1.5 Controle em Cascata

Viu-se como uma simples malha de realimentação pode ser simples e ao mesmo tempo extremamente útil no controle de um sistema, porém, dependendo da complexidade da planta, ela não nos garante uma resposta em um tempo satisfatório. Uma das possíveis soluções para esse tipo de planta é a utilização do controle em cascata, ou seja, divide-se a planta por partes e aplica-se um controlador a cada parte da planta, tornando o sistema mais robusto (SEBORG, EDGAR e MELLICHAMP, 2004). A Figura 8 apresenta uma malha de controle em cascata, com uma malha interna e uma malha externa. Pode-se notar que a planta foi dividida em $G_1(s)$ e $G_2(s)$, onde a saída da planta interna é a entrada da planta externa. Cada planta possui uma realimentação negativa e um controlador próprio. O controlador $G_{C1}(s)$ é responsável pela planta $G_1(s)$ e o controlador $G_{C2}(s)$ é responsável pela planta $G_2(s)$. A entrada do sistema é $R(s)$ e a saída é $Y(s)$.

Figura 8 - Malha de controle em cascata

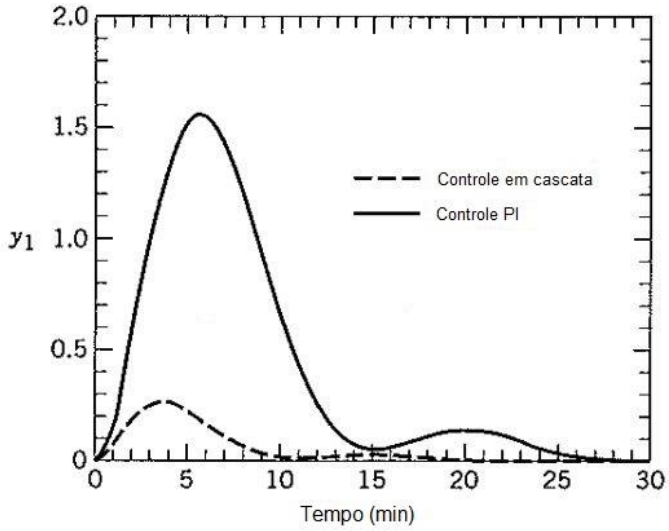


Fonte: Adaptado de Johnson e Moradi (2005).

Pode-se notar que alguma perturbação que ocorra em uma malha interna será corrigida pelo controlador interno e, portanto, terá um efeito reduzido na variável de processo, tornando o sistema mais estável e robusto a perturbações. Na Figura 8, uma perturbação em $D_S(s)$ seria detectado e corrigido pelo controlador $G_{C2}(s)$ antes que pudesse causar maiores impactos na saída do sistema.

Para o correto funcionamento do sistema em cascata, a malha interna deve responder mais rapidamente que a malha externa. Assim, o sistema será mais estável, e será possível utilizar maiores valores de ganho proporcional na malha externa. Além disso, a malha secundária geralmente é composta por um controlador um P ou um PI, a ação derivativa raramente é utilizada na malha interna. Já a malha externa, normalmente é um controlador PI ou PID (SEBORG, EDGAR e MELLICHAMP, 2004). A Figura 9 mostra a resposta de um sistema aplicando um distúrbio na malha interna, com controladores em cascata, e com um único controlador PI. O sistema em cascata é muito menos afetado por distúrbios internos.

Figura 9 - Reposta de um sistema a um distúrbio na malha de controle interna com e sem controle em cascata



Fonte: Adaptado de Seborg, Edgar e Mellichamp (2004).

2.2 MÉTODOS DE OTIMIZAÇÃO

Um dos princípios fundamentais da natureza é a busca por um estado ótimo. Desde o princípio da vida, os átomos começaram a formar ligações para maximizar a utilização de energia de seus elétrons, até as formas moleculares, as quais buscavam formar estruturas cristalinas de forma a minimizar a energia necessária para as ligações moleculares. O princípio biológico da sobrevivência do mais apto também atuou durante milhares de anos, criando formas de vida cada vez melhores e mais adaptadas (WEISE, 2009).

Genericamente, tudo o que recebe um alto grau de importância, se busca uma fórmula matemática para otimização. Os seres humanos buscam o maior grau de felicidade com o menor esforço possível, empresas buscam maximizar lucros e minimizar custos. Segundo Weise (2009), a Otimização Global tem por objetivo encontrar as melhores soluções x^* para um conjunto $\mathbb{X}$ de acordo com um conjunto de objetivos $F = \{f_1, f_2, \dots, f_n\}$. Esse conjunto de funções é chamado de função objetivo. A função objetivo $F : \mathbb{X} \rightarrow Y \; com \; Y \subseteq \mathbb{R}$ é a função matemática a qual se deseja minimizar (WEISE, 2009).

2.2.1 Classes de Problemas

Muitos problemas podem ser resolvidos por algoritmos simples que encontram soluções exatas, porém isso não é possível para todos os problemas. Quanto mais difícil é o problema, melhor deve ser o algoritmo para solucioná-lo, e quanto mais rápido o algoritmo solucioná-lo, mais eficaz ele é. Para tanto, há classes de dificuldade, aplicadas a problemas de decisão. A dificuldade de um problema é avaliada pelo tempo necessário para solucioná-lo, quanto maior o tempo necessário, mais difícil ele é (CORMEN, LEISERSON, *et al.*, 2009). São três as classes dos problemas de decisão: P; NP; e NPC ou NP-Completo, os quais são explicados a seguir.

Os problemas de classe P (*Polynomial Time*) são aqueles que podem ser solucionados em um tempo polinomial $O(n^k)$, para uma constante k, onde n é o número de variáveis do problema, ou seja, o tempo para solução é conhecido (CORMEN, LEISERSON, *et al.*, 2009).

Os problemas de classe NP (*Nondeterministic Polynomial Time*) são os problemas cuja solução é verificável em um tempo polinomial $O(n^k)$. Isso significa que a solução exata de um problema não pode ser encontrada em um tempo conhecido, porém, dada uma solução para o problema, se pode verificar se essa solução é ou não é viável para o problema em um tempo conhecido e calculado. Os problemas NP somente são solucionados através de polinômios não-determinísticos. Os problemas de classe P estão contidos em NP, já que podem ser solucionados em um tempo conhecido, podem também ser verificados. Dessa forma pode-se afirmar que $P \subset NP$ (CORMEN, LEISERSON, *et al.*, 2009).

Os problemas de classe NPC, também conhecidos como NP-Completo, são os problemas de classe NP para os quais não há solução nenhuma dentro da classe NP mais difíceis de serem solucionados que eles, ou seja, são tão difíceis quanto os NP mais difíceis. Dessa forma, se pode afirmar que, se há uma solução viável em tempo polinomial para um problema NP-Completo, então há uma solução viável em tempo polinomial para qualquer problema NP (CORMEN, LEISERSON, *et al.*, 2009).

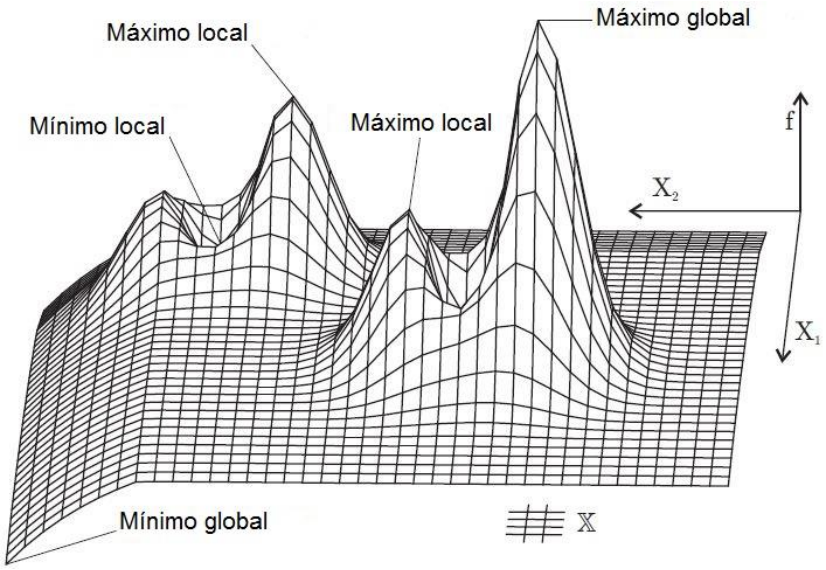
Apesar de muitas pesquisas já terem sido feitas sobre os problemas NP-Completo, até hoje não foi encontrada uma solução em tempo polinomial conhecido para eles; porém, também não foi encontrada nenhuma prova de que não há uma solução viável em tempo polinomial para eles. Segundo Cormen, Leiserson, *et al*.

(2009), ao estar de frente com um problema de classe NP-Completo, se deve abrir mão de buscar uma solução exata para o problema e buscar algoritmos com soluções aproximadas para o problema. Em geral, meta-heurísticas, são utilizadas apenas em problemas de classe NP, para os quais não é possível encontrar uma solução exata em tempo computacional viável. Além dessas três classes, há problemas determinados NP-Difícil, problemas para os quais não há um algoritmo não-determinístico conhecido para solucioná-los, porém podem ser reduzidos a problemas NP-Completo. Problemas NP-Difícil são tão difíceis quanto problemas NP-Completo (CORMEN, LEISERSON, *et al.*, 2009).

2.2.2 Otimização Global

Em otimização global, buscamos encontrar, em um espaço de busca determinado, o valor mínimo que uma função f pode ter (WEISE, 2009). Algumas funções, porém, possuem valores ótimos locais, que podem gerar confusão para o algoritmo de busca. A Figura 10 mostra o espaço de busca de uma função bidimensional que possui picos de valores. Os picos com maior e menor valor são os Máximo e Mínimo Global, enquanto todos os outros picos são Máximos e Mínimos Locais. Como as meta-heurísticas possuem um grau de aleatoriedade na busca, e não conhecem a solução ótima, elas podem muitas vezes encontrar mínimos locais e ficarem presas, sendo incapazes de encontrar o mínimo global.

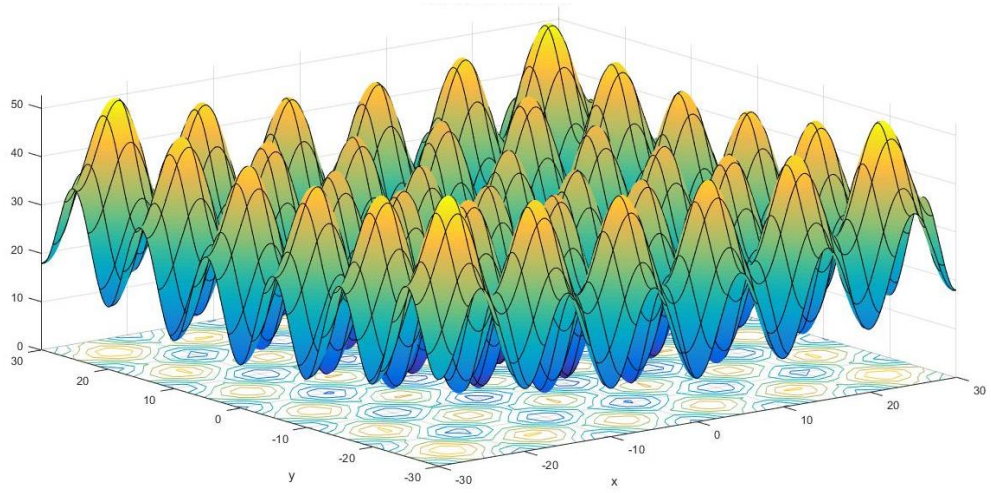
Figura 10 - Máximos e mínimos de uma função bidimensional



Fonte: Adaptado de Weise (2009).

Por definição, o mínimo local é um valor $x^* \in \mathbb{X} \mid f(x^*) \leq f(x)$ para todo x na vizinhança de x^*, sendo $f(x^*)$ o custo da função objetivo f no ponto x^*. Já o mínimo global é o valor $x^* \in \mathbb{X} \mid f(x^*) \leq f(x) \; \forall \, x \in \mathbb{X}$ (WEISE, 2009). Um bom algoritmo deve ser capaz de escapar de mínimos locais para encontrar o mínimo global. Uma função utilizada como parâmetro para validar a capacidade de um algoritmo de escapar de um mínimo local é a Função de Rastrigin (*Rastrigin Function*), mostrada na Figura 11, a qual possui muitos mínimos locais e um mínimo global em (0,0).

Figura 11 – Função de Rastrigin



Fonte: Elaborado pelo autor.

Além dos mínimos locais, há diversos outros obstáculos para os algoritmos de otimização, o que justifica a vasta gama de diferentes algoritmos existentes hoje, para se adaptarem ao tipo de problema proposto. Alguns desses obstáculos são (WEISE, 2009):

- Convergência prematura: falta de capacidade do algoritmo de explorar maiores áreas do espaço de busca, convergindo rapidamente a um mínimo local, geralmente causada por falta de diversidade;
- Enganação: quando a função objetivo guia fortemente o algoritmo para um mínimo local;
- Neutralidade: ocorre quando o mínimo global de uma função objetivo está em um espaço dominantemente plano, onde é muito difícil encontrá-lo;
- Epistasia: é definida como a dependência da contribuição de um gene para o valor da função objetivo dos outros genes.

- Ruído: são perturbações que podem ocorrer em um algoritmo. Ruídos são muito comuns em redes que dependem de treinamento, quando recebem dados inseridos incorretamente;

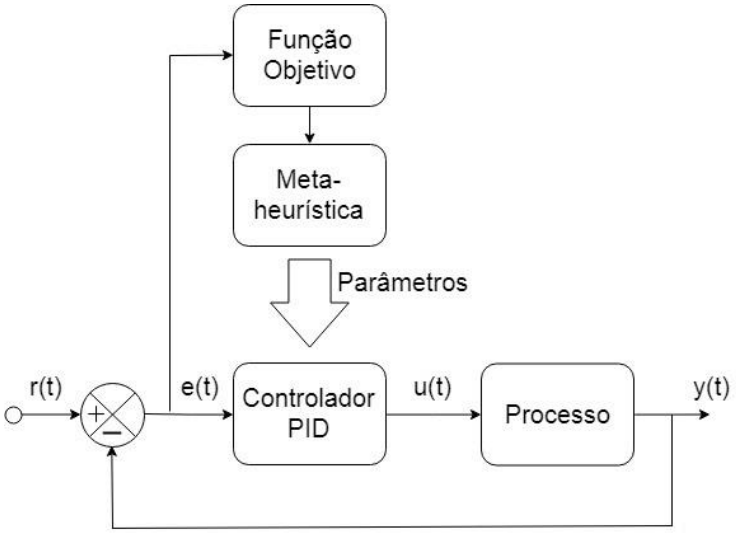
Quanto melhor uma meta-heurística for capaz de lidar com esses obstáculos, mais robusta ela é.

2.3 META-HEURÍSTICAS

Meta-heurística é o termo utilizado na ciência da computação para descrever um sub-campo da área de *Otimização Estocástica*. Otimização estocástica é a classe de algoritmos e técnicas utilizada para resolver problemas de difícil solução (NP-hard) empregando algum grau de aleatoriedade, buscando encontrar um valor ótimo, ou tão ótimo quanto possível (LUKE, 2013). A maior parte desses algoritmos é classificada como meta-heurística e é empregada para a solução de uma ampla variedade de problemas. A etimologia da palavra meta-heurística tem origem grega, com a junção do sufixo *meta*, que significa “metodologia de alto nível”, e da palavra *heuriskein*, cujo significado é “a arte de descoberta de novas estratégias para a solução de problemas” (TALBI, 2009). O termo meta-heurística foi utilizado pela primeira vez em 1986, por Fred Glover, em (GLOVER, 1986).

A diferença entre uma meta-heurística e uma heurística é que a heurística é criada para a solução de um problema específico e, portanto, se torna dependente desse problema, enquanto a meta-heurística é criada para a solução da maior variedade de problemas possível, sendo então independente do problema (CAMPOS, 2006). É importante notar, também, que a meta-heurística não garante encontrar a solução ótima para o problema, e sim uma boa aproximação, em um tempo computacional aceitável, como é comum em problemas estocásticos, onde a solução ótima é desconhecida (CAMPOS, 2006). A Figura 12 mostra o diagrama de blocos de como as meta-heurísticas determinam os parâmetros do controlador PID.

Figura 12 - Determinação dos parâmetros do PID através das meta-heurísticas



Fonte: Elaborado pelo autor.

A meta-heurística escolhe aleatoriamente os parâmetros do controlador PID, o erro do sistema é calculado e enviado para a função objetivo, que calcula o custo dos parâmetros selecionados pela meta-heurística. A meta-heurística, então, compara o custo de cada parâmetro selecionado e utiliza essa informação para buscar os parâmetros que minimizam o custo da função objetivo.

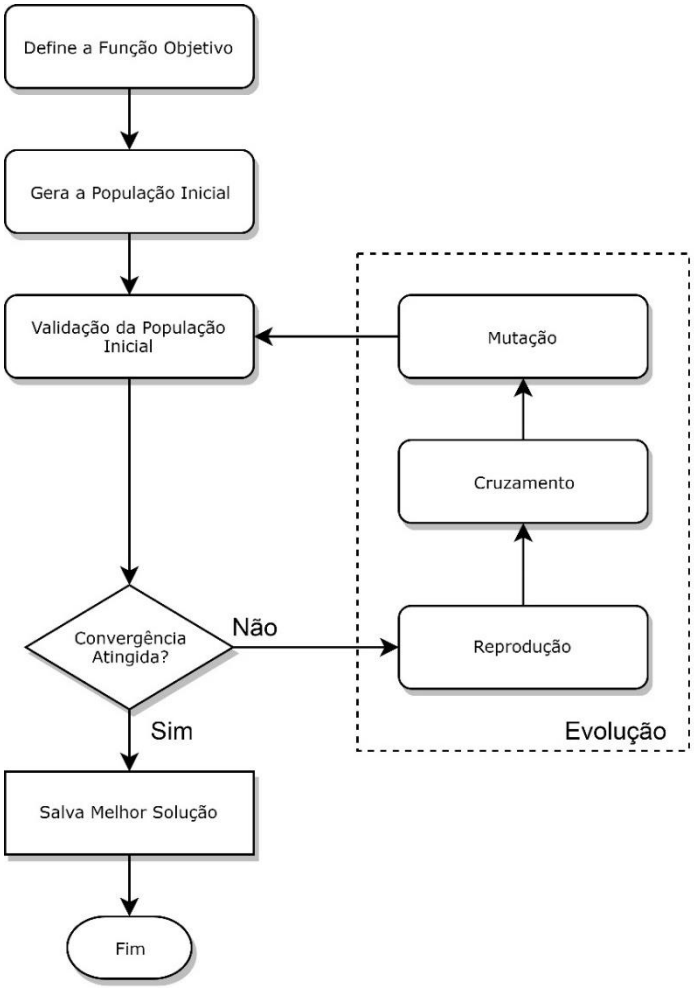
Os algoritmos meta-heurísticos são os mais diversos possíveis. Luke (2013) cita 139 algoritmos diferentes. Para a realização deste trabalho, serão utilizados quatro algoritmos diferentes para comparação: *Algoritmos Genéticos*; *Otimização por Enxame de Partículas*; *Otimização por Colônia de Formigas*; e *Colônia Artificial de Abelhas*. A explicação de cada algoritmo será apresentada a seguir.

2.3.1 Algoritmo Genético

Genetic Algorithm (GA) é um algoritmo de otimização introduzido em 1975 por John Holland (GOLDBERG, 1989), buscando imitar os processos observados na seleção natural e reproduzi-los em sistemas computacionais. Baseado nisso, o GA cria uma população inicial de indivíduos aleatórios, chamados cromossomos. Essa população passa por etapas de evolução, a partir da criação de novos indivíduos, otimizando a solução encontrada para o problema. Para um problema a ser minimizado com *n* incógnitas, cada indivíduo terá *n* genes e representará uma solução para a função objetivo a ser minimizada

A Figura 13 mostra o fluxograma do GA, e a seguir cada passo do fluxograma é explicado.

Figura 13 - Fluxograma do algoritmo genético



Fonte: Elaborado pelo autor.

2.3.1.1 Validação

Cada indivíduo é avaliado quanto a solução da função objetivo. O que tiver a melhor solução é o mais apto a resolver o problema. A cada iteração, todos os indivíduos da população passam por validação (*validation*), e o algoritmo é repetido até que algum critério de parada seja encontrado, podendo ser o valor da solução encontrada, o tempo de execução ou o número máximo de iterações, por exemplo. Enquanto o critério de parada não é atingido, o algoritmo, passa para a evolução. Segundo Goldberg (1989), um processo de evolução capaz de gerar boas soluções

para problemas diversos deve conter, ao menos, os operadores de *Reprodução*, *Cruzamento* e *Mutação*.

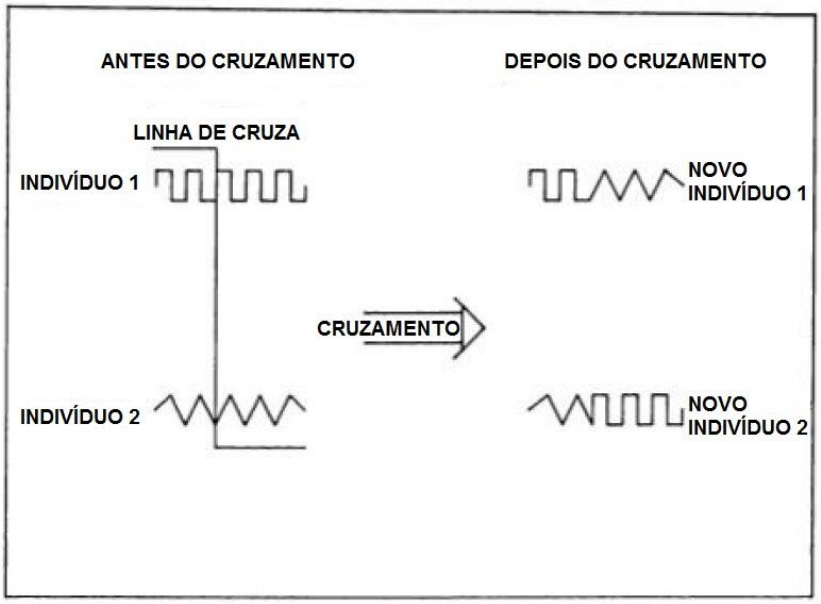
2.3.1.2 Reprodução

A reprodução (*Reproduction*) é o fator que garante a sobrevivência dos melhores indivíduos encontrados em cada geração, garantindo que uma geração sempre encontre uma solução melhor ou, no mínimo, igual à anterior (GOLDBERG, 1989). Esse processo funciona da seguinte forma. Após todos os indivíduos serem validados quando a aptidão para solucionar o problema, os indivíduos com melhor aptidão são mantidos na próxima geração, e os demais indivíduos são descartados para dar lugar a novos indivíduos na próxima geração (CAMPOS, 2006). Para uma população de n indivíduos, por exemplo, os k indivíduos com melhor aptidão são mantidos na próxima geração, onde $k<n$, e os n-k outros indivíduos dão lugar a novos indivíduos através do processo de evolução. Deve-se tomar cuidado ao selecionar quantos indivíduos passarão para a próxima geração, um número elevado fará o algoritmo convergir muito rapidamente, prejudicando sua evolução.

2.3.1.3 Cruzamento

Os indivíduos da geração atual são chamados de pais (*parents*) e os da geração futura, os filhos (*offspring*), são criados através da combinação de genes de dois indivíduos da geração anterior. O processo de cruzamento (*Crossover*) é mostrado na Figura 14. O número de genes que será alterado no cruzamento dos genes é calculado aleatoriamente, variando entre números inteiros de 1 até n-1, para um cromossomo com n genes, alterando todos os valores à direita deste (GOLDBERG, 1989). Cada par de cromossomos gera um par de filhos.

Figura 14 - Processo de cruzamento do algoritmo genético



Fonte: Adaptado de Goldberg (1989).

Nessa etapa, ocorre a seleção de quais indivíduos serão os pais das gerações futuras. Há diversos tipos de seleção, a seleção por roleta (*Roulette Wheel Selection*) é a mais utilizada (TALBI, 2009), onde indivíduos com maior aptidão para solucionar o problema têm maiores chances de serem selecionados. A probabilidade de cada indivíduo ser selecionado é dada pela equação 3.13:

$$p_i = \frac{\omega_i}{\sum_j^N \omega_j} \tag{3.13}$$

onde p_i é a probabilidade do indivíduo i ser escolhido, ω_i e ω_j representam a aptidão (*fitness*) do indivíduo i ou j, respectivamente, e N é o tamanho da população, ou seja, o número de indivíduos daquela população.

A soma de todas as probabilidades é igual a 1, ou 100%. Ao se colocar em um gráfico circular, se terão áreas maiores para indivíduos com aptidão melhores, conforme o exemplo dado por Goldberg (1989), de uma população com quatro indivíduos, com suas respectivas aptidões, mostrada na Tabela 2.

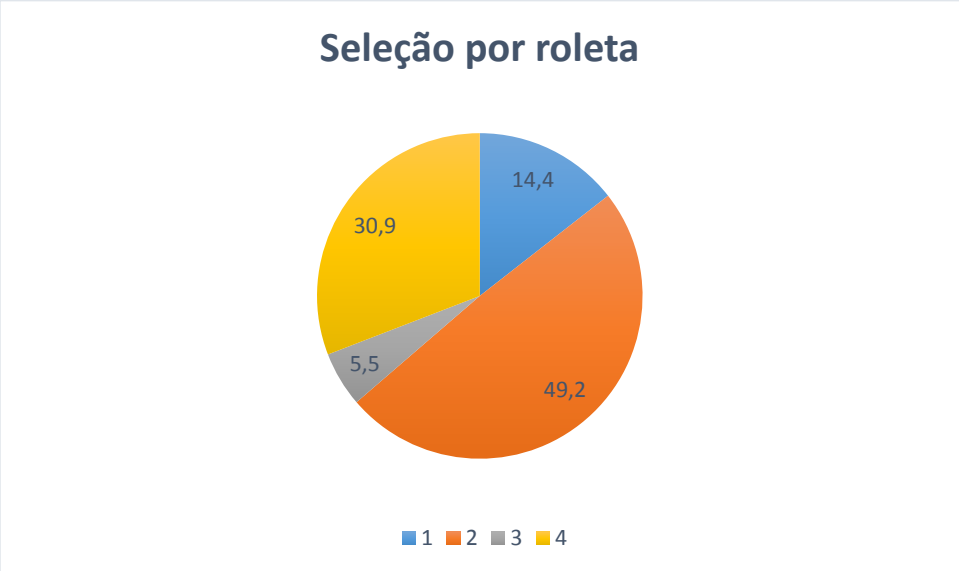
Tabela 2 - População de um algoritmo genético simples com valores de aptidão e probabilidade

Nº	Indivíduo	Aptidão	Probabilidade %
1	0 1 1 0 1	169	14,4
2	1 1 0 0 0	576	49,2
3	0 1 0 0 0	64	5,5
4	1 0 0 1 1	361	30,9
Total		1170	100,0

Fonte: Elaborado pelo autor.

O Gráfico 1 mostra a probabilidade de cada indivíduo dessa população ser selecionado, calculada conforme a equação 3.13. Nota-se que o indivíduo com maior aptidão, o segundo, tem mais chances de ser selecionado que os demais.

Gráfico 1 - Seleção por roleta de um algoritmo genético simples

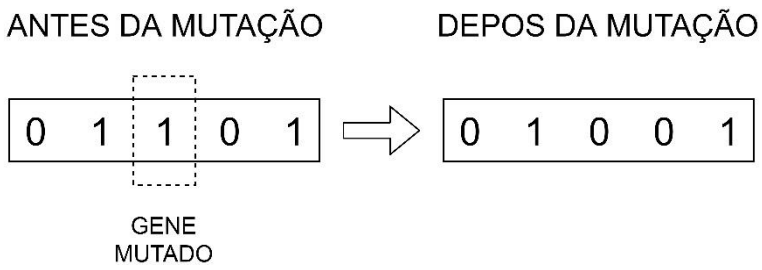


Fonte: Elaborado pelo autor.

2.3.1.4 Mutação

A mutação (*Mutation*) simula o que ocorre na evolução das espécies, um gene de um indivíduo aleatório sofre uma alteração. Esse processo tem o objetivo de diversificar a população e evitar que ela convirja precocemente, podendo ficar presa em um mínimo local (SAAD, JAMALUDDIN e DARUS, 2012). A Figura 15 mostra um indivíduo antes após sofrer mutação.

Figura 15 - Exemplo de mutação de um algoritmo genético



Fonte: Elaborado pelo autor.

A utilização de um fator de mutação muito baixo causa uma baixa diversificação da população, dificultando o encontro da solução ótima. Já um fator de mutação elevado pode dificultar a convergência do algoritmo (SAAD, JAMALUDDIN e DARUS, 2012).

2.3.2 Otimização por Enxame de Partículas

Otimização por Enxame de Partículas *(Particle Swarm Optimization* – PSO) é um algoritmo populacional apresentado pela primeira vez em 1995 por Russell Eberhart e James Kennedy, inspirado em duas metodologias: a vida artificial; e em bandos de pássaros, cardumes de peixes e teoria das partículas em geral (EBERHART e KENNEDY, 1995). O PSO cria partículas aleatórias no espaço de busca, onde cada partícula representa uma solução para o problema. Cada partícula contém duas informações muito importantes: *posição* e *velocidade*. As partículas vão se mover no espaço de busca a cada iteração, buscando por melhores soluções.

2.3.2.1 Atualização da posição e velocidade das partículas

Durante a execução do algoritmo, algumas informações devem ser guardadas. Ao se movimentar pelo espaço de busca, cada partícula irá passar por diversos pontos. Cada partícula deve manter na memória a posição na qual encontrou a melhor aptidão (*fitness*) para o problema, ou seja, a melhor posição individual. Também deve ser armazenada a melhor aptidão global das partículas, ou seja, a posição que encontrou a melhor aptidão dentre todas as partículas (EBERHART e KENNEDY, 1995). A velocidade da partícula será atualizada conforme a equação 3.14:

$$v_{ij}(t+1) = \omega v_{ij}(t) + r_1 c_1 \left(p_{ij}(t) - x_{ij}(t)\right) + r_2 c_2 \left(g_i(t) - x_{ij}(t)\right) \qquad (3.14)$$

onde $v_{ij}(t+1)$ é a velocidade da partícula na próxima iteração, ω é o coeficiente de inércia, $v_{ij}(t)$ é a velocidade atual da partícula, r_1 e r_2 são dois números aleatórios no intervalo $[0,1]$, c_1 e c_2 são os coeficientes de aceleração, $p_{ij}(t)$ é a melhor solução individual, $g_i(t)$ é a melhor solução global, e $x_{ij}(t)$ é a posição atual da partícula.

A nova velocidade da partícula será a soma de uma componente da velocidade atual, associada ao fator de inércia, com uma componente da melhor solução individual, e uma componente da melhor solução global, ambas associadas a um fator de aceleração.

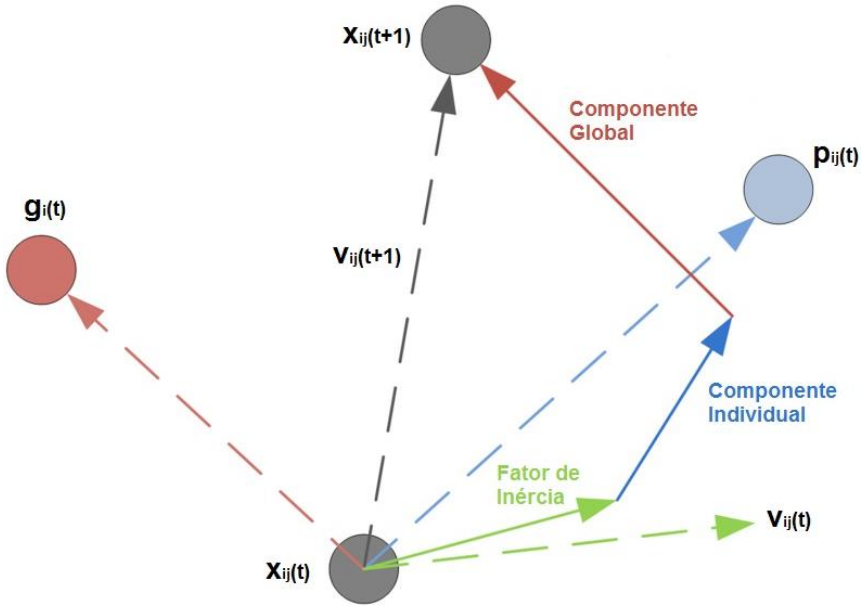
A nova posição da partícula será dada pela equação 3.15:

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \qquad (3.15)$$

onde $x_{ij}(t+1)$ é a posição da partícula na próxima iteração, $x_{ij}(t)$ é a posição atual da partícula, e $v_{ij}(t+1)$ é a velocidade da partícula na próxima iteração.

A Figura 16 mostra um exemplo de atualização dos parâmetros da partícula, conforme as equações 3.14 e 3.15.

Figura 16 - Atualização dos parâmetros de velocidade e posição das partículas



Fonte: Adaptado de Sowmya e Sunil (2013).

Seguindo essa regra, as partículas tendem a encontrar soluções cada vez melhores para o problema (SOWMYA e SUNIL, 2013).

2.3.2.2 Coeficientes de constrição

Após a publicação do artigo em 1995, houve um grande interesse da comunidade acadêmica no PSO, e muitos estudos surgiram nos anos seguintes, comprovando a eficácia do algoritmo para a solução de diversos problemas (SOWMYA e SUNIL, 2013). Porém, para Clerc e Kennedy (2002), a explicação do algoritmo não ficou clara pelos autores, fazendo com que muitas versões diferentes do algoritmo surgissem, sendo que nem todas elas eram capazes de achar as soluções ótimas.

Clerc e Kennedy (2002) fizeram um estudo mais aprofundado para explicar o funcionamento do algoritmo, envolvendo análise e sugestões de melhoria para o algoritmo original. Foram apresentados coeficientes de constrição, utilizados para controlar as características dinâmicas do algoritmo, fator de inércia e coeficientes de aceleração, a fim de levar o algoritmo a convergir mais rapidamente e ser capaz de resolver um maior número de problemas, tornando o algoritmo mais robusto. A relação entre os coeficientes de constrição, para $\varphi = \varphi_1 + \varphi_2 \geq 4$ é dada pela equação 3.16:

$$\chi = \frac{2k}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \tag{3.16}$$

Os valores sugeridos para os parâmetros são $k = 1$ e $\varphi_1 = \varphi_2 = 2{,}05$. Após, se calcula as componentes inercial e de aceleração do algoritmo da seguinte forma: $\omega = \chi$; $c_1 = \chi\varphi_1$ e $c_2 = \chi\varphi_2$. Ao utilizar os coeficientes de constrição sugeridos por Clerc e Kennedy (2002), se encontram os seguintes valores: $\omega = 0{,}7298$ e $c_1 = c_2 = 1{,}4962$.

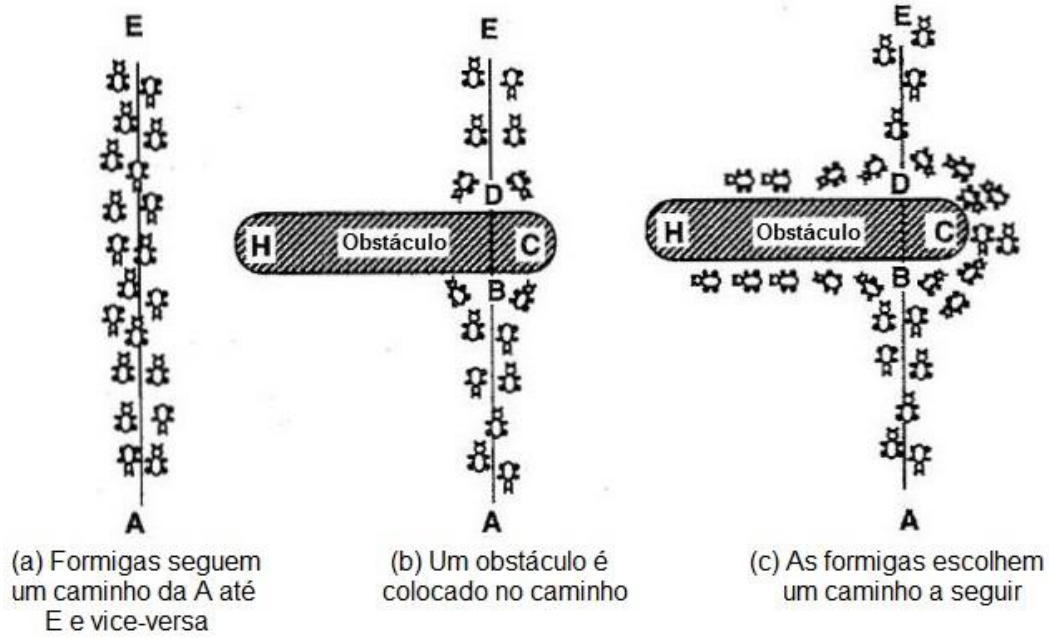
2.3.3 Otimização por Colônia de Formigas

Ant Colony Optimization (ACO) é um algoritmo baseado na maneira como as formigas buscam por comida (DORIGO e GAMBARDELLA, 1997). As formigas trocam informação através de feromônio depositado no chão, já não possuem uma boa visão. Enquanto caminham, as formigas depositam feromônio no chão, criando assim uma trilha por ela percorrida. O caminho que uma formiga faz para procurar comida sempre é baseado na quantidade de feromônio encontrado no caminho. Como, inicialmente, não há quantidade alguma, a busca por comida ocorre de forma aleatória e desordenada. À medida que as formigas retornam com comida, as trilhas de feromônio vão sendo criadas, as próximas formigas que saírem do formigueiro em busca de comida irão buscar o caminho com a maior quantidade de feromônio, até que todas as formigas sigam o mesmo caminho (DORIGO e GAMBARDELLA, 1997).

Um experimento realizado pode nos mostrar o comportamento de formigas reais na busca por comida (DORIGO, MANIEZZO e COLORNI, 1996). A Figura 17 (a) mostra formigas indo do caminho A, o formigueiro, até o caminho E, a fonte de comida, e voltando. O caminho já está repleto de feromônio por elas depositado. Na Figura 17 (b), um obstáculo é repentinamente colocado no caminho das formigas. As formigas que chegam aos pontos B e D precisam escolher entre virar para a esquerda ou para a direita para seguir até a comida, sempre optando pelo caminho com mais feromônio. Como inicialmente não há feromônio em nenhum dos dois caminhos, a primeira formiga que chegar aos pontos B e D tem a mesma probabilidade de escolher qual caminho seguir (DORIGO, MANIEZZO e COLORNI, 1996). Como o caminho BCD é menor que o caminho BHD, as formigas que fizeram

esse caminho chegarão primeiro ao ponto D, como visto na Figura 17 (c), aumentando a quantidade de feromônio nesse caminho. A partir de então, as formigas que estão voltando chegarem ao ponto D, encontrarão uma quantidade de feromônio maior no caminho DCB que no caminho DHB, aumentando assim a probabilidade de escolherem o caminho DCB. O mesmo ocorre para as formigas que estão indo do ponto B para o ponto D. Como consequência disso, a quantidade de feromônio depositada no caminho mais curto irá aumentar mais rapidamente que no caminho mais longo e, ao final, todas as formigas estarão escolhendo o caminho mais curto.

Figura 17 - Exemplo de obstáculo com formigas reais



Fonte: Adaptado de Dorigo, Maniezzo e Colorni (1996).

No começo a probabilidade de cada formiga escolher cada caminho será igual. À medida em que as formigas que escolheram o menor caminho voltarem, irão depositar mais feromônio naquele caminho, aumentando gradativamente a probabilidade das formigas escolherem aquele caminho. A partir de então, a maioria das formigas vai escolher o mesmo caminho, aumentando significantemente a quantidade de feromônio naquele caminho, já no caminho mais longo, com menos formigas, haverá menos feromônio. Outro fator importante a se destacar é que o feromônio evapora com o tempo, portanto, caminhos pouco escolhidos pelas formigas acabam por perder a quantidade de feromônio que tinham.

2.3.3.1 Atualização da posição

O ACO busca reproduzir as características das formigas em um algoritmo computacional a fim de resolver problemas de otimização combinatórios, como o problema do caixeiro viajante, por exemplo. Uma população de formigas é criada. Cada formiga inicia em um ponto aleatório e percorre todo o caminho. O caminho seguido por cada formiga é escolhido de acordo com a quantidade de feromônio contida em cada ponto daquele caminho (VILLAGRA e BARÁN, 2007), conforme a equação 3.17:

$$p_i^a = \begin{cases} \dfrac{\tau_i^\alpha . \eta_i^\beta}{\sum_j^N \tau_j^\alpha . \eta_j^\beta}, & se\ i \in N \\ 0, & outros\ casos \end{cases} \tag{3.17}$$

onde p_i^a é a probabilidade da formiga ir do ponto i para o ponto a, τ_i é a quantidade de feromônio no ponto i, η_i é a informação heurística (*heuristic information*), que representa a desejabilidade da formiga em escolher o ponto i, sendo calculada como o inverso da distância do ponto i ao ponto a, α representa o peso exponencial do feromônio, e β o peso exponencial da informação heurística, e N representa os pontos que ainda não foram visitados.

2.3.3.2 Atualização do feromônio

Uma quantidade inicial de feromônio é depositada em cada ponto a ser percorrido (DORIGO, MANIEZZO e COLORNI, 1996). Após o final de cada iteração, ocorre a atualização da quantidade de feromônio, conforme a equação 3.18:

$$\tau_i^a(it+1) = \rho\,.\,\tau_i^a(it) + \Delta\tau_i^a \tag{3.18}$$

onde $\tau_i^a(it+1)$ é a quantidade de feromônio da próxima iteração, ρ é o coeficiente de evaporação, tal que a evaporação é representada por (1- ρ), $\tau_i^a(it)$ é a quantidade de feromônio da iteração atual, e $\Delta\tau_i^a$ é a quantidade de feromônio depositado pela formiga durante o caminho que percorreu de i até a. A quantidade de feromônio é tanto maior quanto menor for o caminho percorrido pela formiga, e é dado pela equação 3.19:

$$\Delta\tau_i^a = \begin{cases} \frac{Q}{L_k}, & se\ (i,a) \in k \\ 0, & outros\ casos \end{cases} \quad (3.19)$$

onde Q é uma constante e L_k é o caminho percorrido pela formiga do ponto i ao ponto a, e k representa o caminho total percorrido pela formiga. Ou seja, se durante o trajeto da formiga, ela passou pelo caminho (i,a), será incrementado feromônio àquele caminho, caso contrário, o acréscimo de feromônio é nulo.

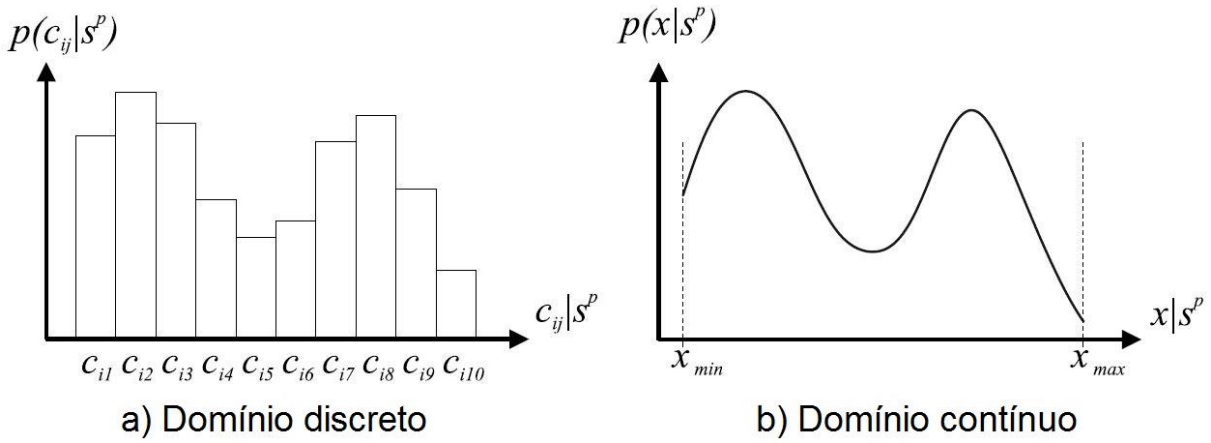
O coeficiente de evaporação é utilizado para aproximar o algoritmo ainda mais da situação real das formigas, já que o feromônio depositado pelas formigas evapora com o vento. A principal vantagem é fazer com que caminhos não mais utilizados sejam esquecidos com o tempo. O valor do coeficiente de evaporação deve ser menor do que 1, para evitar acúmulo ilimitado de feromônio em um determinado ponto (DORIGO, MANIEZZO e COLORNI, 1996).

2.3.3.3 ACO para Domínios Contínuos

O algoritmo ACO foi elaborado para resolver problemas de otimização combinatórios, os quais exigem que o problema seja dividido em uma variação finita de componentes, e o algoritmo busque a combinação ou permutação ideal desses componentes para encontrar a solução ótima (SOCHA e DORIGO, 2008). Porém, há uma classe de problemas que não se enquadra nesses quesitos, os problemas com variáveis contínuas, muito presentes na vida real. Socha e Dorigo (2008) fizeram então uma adaptação do ACO original para o domínio contínuo, sem perder suas características originais.

A primeira diferença entre os dois casos é a Distribuição de Probabilidade, que passa de um domínio discreto para um domínio contínuo. A Figura 18 (a) mostra uma distribuição de probabilidade discreta $P_d(C_{ij}/s^p)$ de componentes finitos $\{C_{i,1}, \dots, C_{i,10}\} \in N(s^p)$, e a Figura 18 (b) mostra uma distribuição de probabilidade contínua $P_c(x/s^p)$, com range $x \in [x_{min}, x_{max}]$.

Figura 18 - Distribuição de Probabilidade



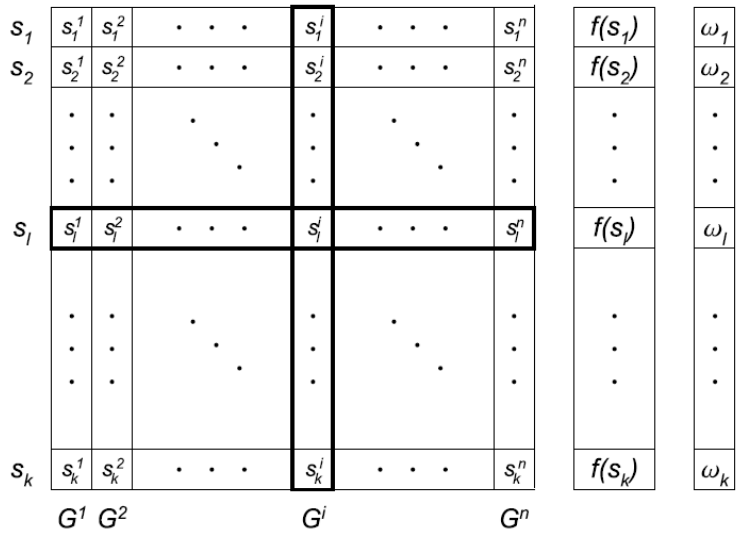
Fonte: Adaptado de Socha e Dorigo (2008).

Em ambas as distribuições, o eixo y representa a probabilidade p. Dessa maneira é possível utilizar o algoritmo no domínio contínuo.

2.3.3.3.1 Atualização do feromônio no domínio contínuo

Assim como no domínio discreto, as formigas deixavam trilhas de feromônio para serem seguidas, no domínio contínuo, elas também precisam se guiar por um parâmetro. Após a população de formigas ser criada, elas são avaliadas e organizadas de acordo com a aptidão para solucionar o problema, em um arquivo, ordenadas da melhor para a pior solução encontrada, como mostrado na Figura 19, onde $\{s_1, \dots, s_k\}$ representam as soluções, $\{\mathrm{G}^1, \dots, \mathrm{G}^n\}$ são as misturas gaussianas, uma para cada variável do problema, e $\{f_{(s_1)}, \dots, f_{(s_1)}\}$ e $\{\omega_1, \dots, \omega_k\}$ são a aptidão e o peso de cada formiga, respectivamente.

Figura 19 - Arquivo de soluções ordenadas do ACO para domínios contínuos



Fonte: (SOCHA e DORIGO, 2008).

Após o arquivo estar completamente preenchido, cada nova iteração irá gerar uma nova posição para as formigas. A atualização do feromônio consiste em armazenar a aptidão das posições das novas formigas no mesmo arquivo e excluir o número excedente de formigas no arquivo, ficando apenas o número da população de formigas original. Dessa forma, é garantido que as melhores soluções ficarão armazenadas no arquivo e as formigas serão guiadas para a solução ótima (SOCHA e DORIGO, 2008).

2.3.3.3.2 Atualização da posição no domínio contínuo

As misturas gaussianas (*Gaussian kernel*) $G^i(x)$ são formadas pela superposição de k gaussianas normais $g_l^i(x)$, onde k é o número de formigas, através da equação 3.20.

$$G^i(x) = \sum_{l=1}^{k} \omega_l g_l^i(x) = \sum_{l=1}^{k} \omega_l \frac{1}{\sigma_l^i \sqrt{2\pi}} e^{-\frac{\left(x-\mu_l^i\right)^2}{2\sigma_l^{i\,2}}} \tag{3.20}$$

Três parâmetros são importantes para parametrizar a mistura gaussiana (SOCHA e DORIGO, 2008): o vetor de pesos associados às gaussianas individuais ω, o vetor das médias μ^i, e o vetor dos desvios padrões σ^i. O vetor dos desvios padrões é calculado pela equação 3.21:

$$\sigma_l^i = \xi \sum_{e=1}^{k} \frac{\left|x_e^i - x_l^i\right|}{k-1} \tag{3.21}$$

onde *k* é o tamanho da população e o parâmetro $\xi > 0$ tem o mesmo efeito no domínio contínuo que a constante de evaporação no domínio discreto. Quanto maior o valor de ξ, menor a velocidade de convergência do algoritmo (SOCHA e DORIGO, 2008). O vetor das médias é preenchido com o valor das soluções, conforme a equação 3.22, e o vetor dos pesos é preenchido conforme a equação 3.23:

$$\mu^i = \{\mu_1^i, \dots, \mu_k^i\} = \{s_1^i, \dots, s_k^i\} \tag{3.22}$$

$$\omega_l = \frac{1}{qk\sqrt{2\pi}} e^{-\frac{(l-1)^2}{2q^2k^2}} \tag{3.23}$$

onde *q* é um parâmetro do algoritmo.

Quando o valor de *q* se aproxima de zero, somente a gaussiana associada à melhor solução é escolhida para gerar novas soluções e o sistema acaba convergindo prematuramente, já quando o valor de q é mais alto, mais soluções são utilizadas para gerar novas soluções, tornando o algoritmo mais robusto, embora aumente o tempo de convergência (SOCHA e DORIGO, 2008). Por fim, a probabilidade de seleção de cada formiga é feita em dois passos. O primeiro passo consiste em escolher uma das gaussianas que compõem a mistura gaussiana, através da equação 3.24:

$$p_l = \frac{\omega_l}{\sum_{r=1}^{k} \omega_r} \tag{3.24}$$

onde p_l é a probabilidade de uma gaussiana ser selecionada. O segundo passo consiste em gerar uma amostra aleatória da gaussiana escolhida. A amostra da gaussiana será, então, a nova posição da formiga (SOCHA e DORIGO, 2008).

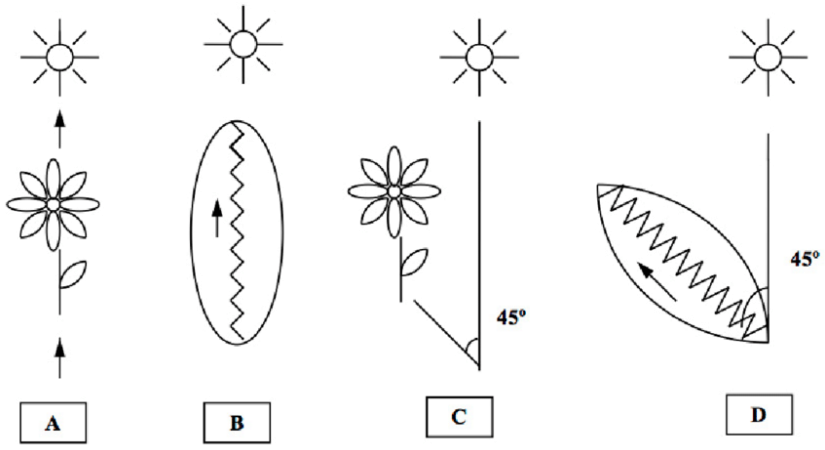
2.3.4 Colônia Artificial de Abelhas

Quando uma colmeia se instala em um lugar desconhecido, as abelhas operárias ativas (*employed bees*) saem em busca de comida, enquanto as abelhas operárias inativas (*onlooker bees*) aguardam em um lugar na colmeia chamado área de dança. Esse nome é dado porque quando as abelhas ativas retornam para a

colmeia com informações da fonte de comida, essas informações são passadas para as abelhas inativas através de uma dança (*waggle dance*), capaz de informar a direção, a distância e a quantidade de comida encontrada na fonte. As abelhas inativas, após ver a dança das abelhas ativas, podem então escolher uma fonte de comida para explorarem. Quanto melhor a fonte de comida informada através da dança, maiores as chances das abelhas inativas escolherem aquela fonte de comida (KARABOGA, 2005).

Após chamar a atenção das operárias inativas, a abelha ativa realiza a dança vibrando o seu corpo durante um intervalo de tempo em determinada direção, retornando ao ponto inicial, ora pela esquerda, ora pela direita, e vibrando novamente, repetindo esse processo algumas vezes. Tomando como base a posição do sol apontando na direção vertical para cima, o ângulo no qual a abelha faz a dança representa a direção da fonte de comida em relação ao sol. A Figura 20 mostra como a dança das abelhas operárias ativas transmite as informações para as operárias inativas.

Figura 20 - Comunicação das abelhas através da dança



Fonte: (BOMFIM, OLIVEIRA e FREITAS, 2017).

Se uma abelha encontrar uma fonte de alimento na mesma direção do sol, ela irá realizar a dança verticalmente para cima na área de dança, agora, se a fonte de comida estiver a um ângulo de 45º à esquerda do sol, a abelha fará a dança formando um ângulo de 45º à esquerda eixo vertical. O tempo que dura a vibração da abelha na dança indica a distância da colmeia até a fonte de comida, quanto mais tempo durar a dança, mais longe está a comida. E quanto mais ela vibrar e mais barulho fizer durante a dança, maior é a quantidade de comida encontrada (BOMFIM, OLIVEIRA e FREITAS, 2017).

Cada fonte de comida tem apenas uma abelha ativa, ou seja, o número de fontes de comida é igual ao número de abelhas ativas, e sempre que essa abelha retorna à colmeia, a dança é feita novamente para informar a quantidade remanescente de comida na fonte. A medida em que a fonte vai piorando, a abelha operária ativa se torna, então, uma abelha escoteira (*scout bees*). A abelha escoteira faz uma busca aleatória próxima à antiga fonte de comida que está se esgotando a fim de encontrar fontes mais fartas de comida. Quando ela encontra, novamente a informação é transmitida no para recrutar abelhas operárias inativas (KARABOGA, 2005).

2.3.4.1 Abelhas operárias ativas

O Artificial Bee Colony (ABC) é um algoritmo inspirado no comportamento das abelhas para a solução de problemas de otimização. Uma população de abelhas é gerada aleatoriamente, onde cada abelha representa uma solução para o problema. Cada abelha é avaliada quanto à aptidão (*fitness*) para solucionar o problema. Metade da população das abelhas com as melhores aptidões serão as abelhas operárias ativas (KARABOGA e BASTURK, 2007).
A posição das abelhas é calculada conforme a equação 3.25:

$$v_{ij} = x_{ij} + \varphi_{ij}\left(x_{ij} - x_{kj}\right) \tag{3.25}$$

onde $k \in \{1,2,\dots,SN\}$ e $j \in \{1,2,\dots,D\}$ são índices calculados aleatoriamente, φ_{ij} é um número aleatório entre [-1,1] SN é o número de operárias ativas e D é o número de variáveis do problema. Embora k seja calculado aleatoriamente, ele deve ser diferente de i (KARABOGA e BASTURK, 2007).

2.3.4.2 Abelhas operárias inativas’

Cada abelha está associada a uma aptidão (*fitness*) para solucionar o problema. A probabilidade de uma abelha operária inativa ocorre conforme a equação 3.26:

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \tag{3.26}$$

onde p_i é a probabilidade da abelha seguir a fonte de comida i, fit_i é a aptidão encontrada naquela fonte de comida, e SN é o número de abelhas operárias ativas. O processo de escolha segue a seleção por roleta. Quanto maior a aptidão de uma fonte de comida, maior a chance dela ser escolhida (JOHNSON e MORADI, 2005).

2.3.4.3 Abelhas escoteiras

Após uma abelha operária encontrar uma solução, ela passa a se movimentar conforme a equação 3.25 em busca de uma solução melhor em torno da que está. Se após um limite de tentativas ela não encontrar uma solução melhor, ela se torna uma abelha escoteira e sai em busca de uma solução melhor aleatoriamente, ou seja, uma posição aleatória no espaço de busca é atribuída para essa abelha e a quantidade limite de movimentos em busca de novas soluções é zerada. Essa nova posição se torna uma nova fonte de comida e ela passa a buscar novamente melhores soluções. O algoritmo inicia com uma abelha escoteira. Um número elevado de abelhas escoteiras aumenta a exploração no espaço de busca, enquanto um número baixo aumenta a busca nos pontos com melhores resultados (KARABOGA e BASTURK, 2007).

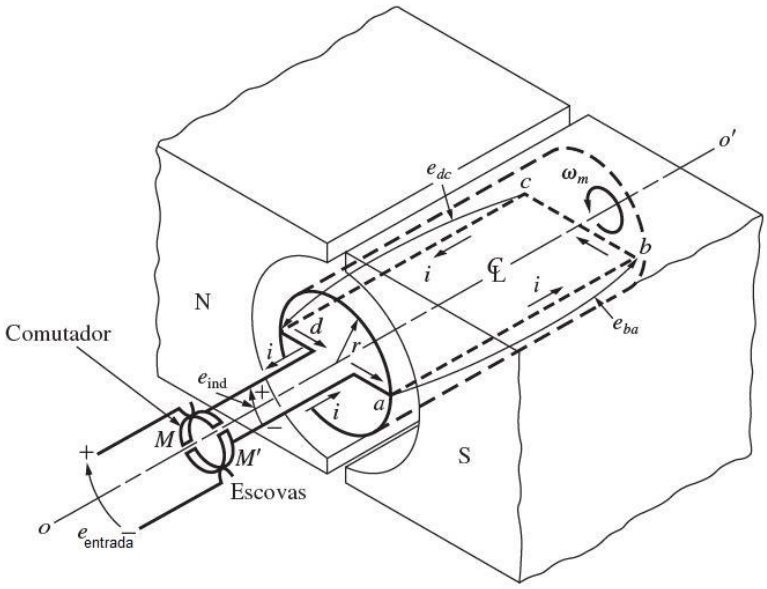
2.4 MOTOR DE CORRENTE CONTÍNUA

As primeiras máquinas de potência utilizadas nos Estados Unidos eram de corrente contínua (CC). Depois, por volta de 1890, a utilização corrente alternada (CA) começou a ganhar espaço. Apesar disso, diversas aplicações continuaram utilizando corrente contínua, como carros e tratores, por exemplo, onde já havia uma fonte CC instalada e não era vantajoso instalar uma fonte alternada de energia (CHAPMAN, 2013). Os motores de corrente contínua também mantiveram espaço no mercado por apresentar vantagens em alguns campos em relação aos motores de corrente alternada, como maior facilidade no controle de velocidade, por exemplo. Aplicações que exigiam um controle robusto e preciso de velocidade utilizavam motores de corrente contínua, antes da criação e popularização de retificadores e inversores com eletrônica de potência (CHAPMAN, 2013).

2.4.1 Princípio de Funcionamento

Os princípios de funcionamento do motor CC são os mesmos do motor CA. As correntes que circulam internamento no motor CC são alternadas, a diferença é que o motor CC possui um sistema que transforma a tensão em contínua na sua saída, esse sistema se chama comutador (CHAPMAN, 2013). O motor é composto por um rotor e um estator. O estator não se move, é responsável por gerar o campo magnético no qual está o rotor. O rotor é a parte girante do motor CC, nele fica o enrolamento da armadura do motor. Quando é aplicada uma tensão a uma espira a qual está imersa em um campo magnético, é induzido um torque nela, que a faz girar (CHAPMAN, 2013). A Figura 21 mostra o esquema de uma motor CC simples com uma espira e um estator de ímãs permanentes, com o sistema de comutação.

Figura 21 - Motor CC simples com uma espira e sistema de comutação



Fonte: Adaptado de Chapman (2013).

A tensão induzida na espira faz com que ela se comporte como um eletroímã, polarizada, e seus polos se atraiam aos polos inversos do estator. Quando o polo norte da espira se aproxima do polo sul do estator, ele tende a ficar ali, portanto, é preciso inverter a polarização da espira para que ela continue girando. Para isso, é utilizado o sistema de comutação, composto pelos dois anéis estáticos em semicírculo, chamados comutadores, e as escovas, responsáveis por fazer o contato elétrico entre os comutadores e a espira. A tensão aplicada aos comutadores é contínua, logo, sempre haverá tensão positiva em um anel de comutação e negativa

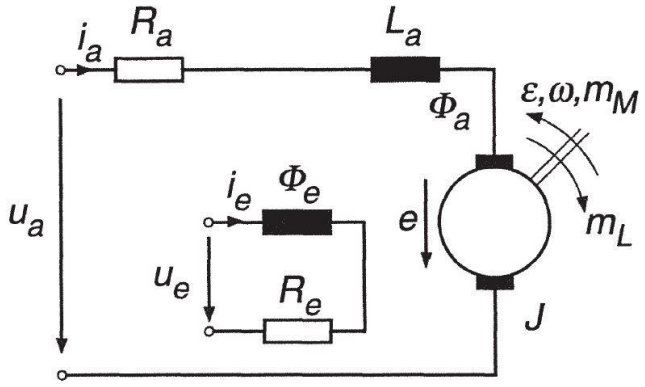
no outro. As espiras são ligadas às escovas, que recebem tensão através dos comutadores. A cada semiciclo. A cada semiciclo da espira, a escova que está em contato com o anel comutador altera, alterando assim o sentido de fluxo de corrente na espira e, consequentemente a polarização (CHAPMAN, 2013).

2.4.2 Modelagem Matemática

Por muitos anos, motores AC foram utilizados em sistemas de controle, porém, as dificuldades de se controlar os motores AC, como suas características não-lineares de resposta e difícil modelagem, incentivaram a busca por novas alternativas (GOLNARAGHI e KUO, 2009). Os motores CC são facilmente controláveis, principalmente posição e velocidade, pois apresentam uma resposta linear. Apesar de ter um custo mais elevado, depois do estudo e avanço no desenvolvimento de motores CC de ímã-permanente, eles alcançaram larga escala na indústria (GOLNARAGHI e KUO, 2009).

O circuito equivalente simplificado de um motor CC é mostrado na Figura 22. Pode-se ver o circuito da armadura com uma fonte e e a armadura representada por uma resistência R_A e uma indutância L_A. O circuito de campo é representado com uma resistência R_e e pelo fluxo nominal de campo Φ_e. O torque da máquina m_M se opõe ao torque da carga m_L (LEONHARD, 1996).

Figura 22 - Diagrama elétrico simplificado do motor de corrente contínua



Fonte: (LEONHARD, 1996).

O comportamento do motor elétrico CC é descrito pelas equações diferenciais 3.27 a 3.33 (LEONHARD, 1996).

$$R_a i_a + L_a \frac{di_a}{dt} + e = u_a \tag{3.27}$$

$$e = c_1 \Phi_e \omega \tag{3.28}$$

$$J \frac{d\omega}{dt} = m_M - m_L \tag{3.29}$$

$$m_M = c_2 \Phi_e i_a \tag{3.30}$$

$$R_e i_e + N_e \frac{d\Phi_e}{dt} = u_e \tag{3.31}$$

$$\Phi_e = f(\mathrm{i}_e) \tag{3.32}$$

$$\frac{d\varepsilon}{dt} = \omega \tag{3.33}$$

Após fazer a normalização pelos valores nominais sem carga, obtém-se as equações 3.34 a 3.37 (LEONHARD, 1996).

$$T_a \frac{d}{dt}\left(\frac{\mathrm{i}_1}{\mathrm{i}_{a0}}\right) = \frac{u_a}{u_{a0}} - \frac{i_a}{i_{a0}} - \frac{\omega}{\omega_0}\frac{\Phi_e}{\Phi_{e0}}, \qquad onde \qquad T_a = \frac{R_a}{L_a} \tag{3.34}$$

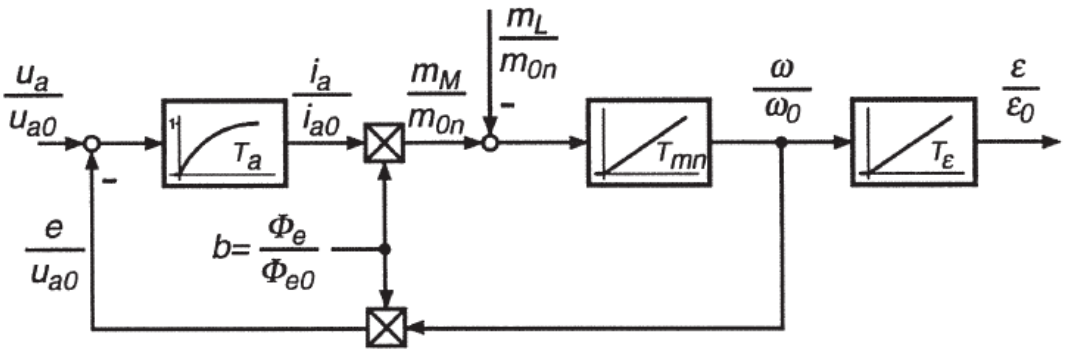
$$T_{e0} \frac{d}{dt}\left(\frac{\Phi_e}{\Phi_{e0}}\right) = \frac{u_e}{u_{e0}} - f_e\left(\frac{\Phi_e}{\Phi_{e0}}\right), \qquad onde \qquad T_{e0} = \frac{N_e \Phi_{e0}}{u_{e0}} \tag{3.35}$$

$$T_{mn} \frac{d}{dt}\left(\frac{\omega}{\omega_0}\right) = \frac{i_a}{i_{a0}}\frac{\Phi_e}{\Phi_{e0}} - \frac{m_L}{m_{L0}}, \qquad onde \qquad T_{mn} = \frac{J\omega_0}{m_0} \tag{3.36}$$

$$T_\varepsilon \frac{d}{dt}\left(\frac{\varepsilon}{\varepsilon_0}\right) = \frac{\omega}{\omega_0}, \qquad onde \qquad T_\varepsilon = \frac{\varepsilon_0}{\omega_0} \tag{3.37}$$

A Figura 23 mostra o diagrama de blocos de um motor CC com fluxo magnético constante e valores normalizados, considerando a tensão de armadura como entrada, o torque da carga como um distúrbio, e a posição como saída.

Figura 23 - Diagrama de blocos do motor CC com fluxo constante



Fonte: (LEONHARD, 1996).

Com o fluxo constante, obtém-se um sistema linear que pode ser representado por uma função de transferência, aplicando a transformada de Laplace na equações diferenciais da seguinte forma:

$$\mathcal{L}\left\{\frac{i_a}{i_{a0}}\right\} = I_a(s)$$

$$\mathcal{L}\left\{\frac{\omega}{\omega_0}\right\} = \Omega(s)$$

$$\mathcal{L}\left\{\frac{\varepsilon}{\varepsilon_0}\right\} = E(s)$$

E considerando:

$$\frac{\Phi_e}{\Phi_{e0}} = b \leq 1$$

As funções de transferência da corrente, velocidade e posição são representadas, respectivamente, pelas equações 3.38, 3.39 e 3.40.

$$\Omega(s) = \frac{bU_a(s) - (T_a s + 1)M_L(s)}{T_{mn}s(T_a s + 1) + b^2} \tag{3.38}$$

$$I_a(s) = \frac{T_{mn}sU_a(s) + bM_L(s)}{T_{mn}s(T_a s + 1) + b^2} \tag{3.39}$$

$$\mathrm{E}(s) = \frac{bU_a(s) - (T_a s + 1)M_L(s)}{T_\varepsilon T_{mn}s^2(T_a s + 1) + T_\varepsilon b^2 s} \tag{3.40}$$

3 ESTADO DA ARTE

Muitos estudos são realizados para a sintonia de controladores PID utilizando meta-heurísticas. Há diversas meta-heurísticas disponíveis hoje, as quais vêm sendo testadas para a solução de uma ampla variedade de sistemas de controle, apresentando resultados satisfatórios. Neste capítulo são apresentados estudos recentes na área de sintonia de controladores PID utilizando meta-heurísticas.

3.1 IMPLEMENTATION OF FRACTIONAL ORDER PID CONTROLLER FOR AN AVR SYSTEM USING GA AND ACO OPTIMIZATION TECHNIQUES

Suri babu e Chiranjeevi (2016) utilizaram as meta-heurísticas Algoritmos Genéticos e Otimização por Colônia de Formigas para sintonizar um PID de ordem fracionária (*Fractional Order PID* – FOPID) para controle de um regulador automático de tensão (*Automatic Voltage Regulator* – AVR).

O regulador de tensão AVR visa a absorver variações de tensão na entrada e manter a tensão constante na saída. A modelagem matemática do AVR é feita levando em consideração cinco componentes principais: amplificador; excitador; limites de excitação de tensão; gerador; e medidor e filtro. Nessa modelagem matemática são ignoradas a saturação e as não-linearidades do sistema. Cada componente é representado por uma função de primeira ordem com um ganho e uma constante de tempo, com exceção dos limites de excitação de tensão, que são representados por um relé. Todos os componentes são ligados em série.

O critério de desempenho utilizado como função objetivo é o ITAE. Os resultados utilizando as meta-heurísticas com FOPID são comparados com um PID. A partir dos dados coletados, é possível concluir que, utilizando meta-heurísticas, o controlador FOPID apresenta resposta mais estável e robusta que o controlador PID (SURI BABU e CHIRANJEEVI, 2016).

3.2 OPTIMAL PID CONTROLLER DESIGN FOR SPEED CONTROL OF A SEPARATELY EXCITED DC MOTOR: A FIREFLY BASED OPTIMIZATION APPROACH

Pal, Dey, *et al*. (2015) utilizaram um algoritmo inspirado nos vagalumes para controlar a velocidade de um motor CC de excitação independente utilizando um PID. O Algoritmo dos Vagalumes (*FireFly Algorithm* – FFA) se baseia no princípio da atração dos vagalumes através da luz. Os vagalumes emitem luz em intensidades diferentes, e, quanto maior o brilho, mais eles chamam a atenção das fêmeas (PAL, DEY, *et al.*, 2015). No algoritmo, porém, todos os vagalumes são unissexuais, a luz servirá como um sinal para atrair outros vagalumes e irá diminuir com a distância. Quanto maior o brilho de um vagalume, maior é sua aptidão (*fitness*) para a solução do problema, dessa forma, os vagalumes serão atraídos para regiões no espaço de busca onde está a melhor aptidão da população, em busca do ótimo global. Se um vagalume não encontrar nenhum outro vagalume brilhando por perto, ele irá se mover aleatoriamente (PAL, DEY, *et al.*, 2015).

Para validar o FFA, ele foi comparado com a técnica do Regulador Linear Quadrático (*Linear Quadratic Regulator* – LQR). Após modelar o motor, o sistema foi submetido a otimização pelos dois algoritmos para controle da velocidade. Através dos dados obtidos das simulações realizadas, o algoritmo dos vagalumes se mostrou eficaz na determinação dos parâmetros do controlador PID, em comparação com o método LQR, obtendo uma resposta mais rápida mantendo um baixo sobressinal. Além disso, o LQR consome um tempo muito maior para o cálculo dos parâmetros do PID (PAL, DEY, *et al.*, 2015).

3.3 A NEW MULTIOBJECTIVE PERFORMANCE CRITERION USED IN PID TUNING OPTIMIZATION ALGORITHMS

Para a sintonia de controladores PID, muitos algoritmos de inteligência artificial já foram propostos se mostrando muito eficientes. (MOUAYAD e BESTOUN, 2016). Tais algoritmos de otimização encontram os valores ótimos para o PID para minimizar uma determinada função custo, o grande desafio é determinar qual função custo será minimizada, ou seja, informar corretamente ao algoritmo o que é que

desejamos obter na resposta. Mouayad e Bestoun (2016) propõem a função custo mostrada na equação 2.1:

$$J(k) = \omega_{M_P} M_P(k) + \omega_{t_r} t_r(k) + \omega_{t_s} t_s(k) \quad (2.1)$$

onde $M_P(k)$ e ω_{M_P} são o sobressinal e seu respectivo peso, $t_r(k)$ e ω_{t_r} são o tempo de subida e seu respectivo peso, e $t_s(k)$ e ω_{t_s} são o tempo de assentamento e seu respectivo peso.

A determinação dos pesos (MOUAYAD e BESTOUN, 2016) interfere diretamente no resultado da sintonia, e não é uma tarefa fácil a ser feita, muitos autores resolvem esse problema de maneira empírica. Mouayad e Bestoun (2016) propõem também uma maneira analítica de calcular os pesos de forma que todos tenham a mesma contribuição. Para validar a função custo proposta, Mouayad e Bestoun (2016) a utilizam com a meta-heurística PSO para sintonizar um PID a fim de controlar um regulador de tensão AVR. Pode-se ver que a função proposta apresenta uma ótima solução para a sintonia do PID, em comparação com funções mono-objetivo. Além disso, em comparação com outras funções multiobjetivo, tem a grande vantagem de poder calcular de forma analítica os pesos de cada função objetivo (MOUAYAD e BESTOUN, 2016).

3.4 PID TUNING OF SERVO MOTOR USING BAT ALGORITHM

A fim de sintonizar um controlador PID para um servo motor, Singh, Vasant, *et al*. (2015) utilizaram um algoritmo de otimização chamado Algoritmo dos Morcegos (*Bat Algorithm* – BA). O BA foi introduzido por Xin-She Yang em 2010, inspirado na capacidade dos morcegos de se localizarem através do eco do som emitido por eles (SINGH, VASANT, *et al.*, 2015). O BA utiliza a frequência e a sonoridade para alcançar o ótimo global da função objetivo, e foi simplificado por três regras (SINGH, VASANT, *et al.*, 2015):

- Os morcegos utilizam eco localização para saber a distância aproximada de objetos e são capazes de diferenciar um obstáculo de uma presa;
- Os morcegos voam aleatoriamente com uma certa velocidade e em uma certa posição, com frequência constante, alterando o comprimento de onda e a sonoridade para procurar pela presa;

- Assume-se que a sonoridade varia de um valor alto e positivo para um valor mínimo pré-fixado.

A sintonia do controlador para o servo motor foi realizada com dois algoritmos, BA e PSO, ambos utilizando como função objetivo a ITSE. Para a comparação entre os dois algoritmos, foram realizadas cinco simulações com cada um e realizada a média aritmética dos valores obtidos para o tempo de subida e o tempo de assentamento (SINGH, VASANT, *et al.*, 2015). Os dados levantados mostram que o BA pode obter resultados melhores que o PSO para um servo motor, porém o estudo não leva em consideração distúrbios na modelagem do sistema, o que pode ser incluído em um novo estudo (SINGH, VASANT, *et al.*, 2015).

3.5 PID CONTROLLER TUNING USING ANT COLONY OPTIMIZATION FOR INDUCTION MOTOR

Motores de indução são máquinas difíceis de serem controladas, devido a suas características não lineares, exigem sintonia em tempo real para operar no melhor ponto possível a cada ponto de operação do sistema (DHIEB, YAICH, *et al.*, 2019). Dhieb, Yaich, *et al*. (2019) utilizaram o algoritmo ACO para fazer o controle de velocidade de um motor de indução gaiola de esquilo, utilizando um circuito de acionamento PWM.

Duas técnicas foram empregadas para a sintonia do controlador PID. Primeiramente se utilizou uma sintonia manual, a qual não apresentou bons parâmetros no domínio do tempo, ficando com alto sobressinal, e longos tempos de tempo de subida e tempo de assentamento (DHIEB, YAICH, *et al.*, 2019). Após, se realizou a sintonia utilizando ACO, a qual se mostrou eficiente para realizar o controle não linear de um motor de indução de gaiola de esquilo.

3.6 ANÁLISE

Os artigos apresentados nesse capítulo mostram os avanços obtidos na sintonia de controladores PID utilizando meta-heurísticas. A comparação entre as meta-heurísticas e os métodos convencionais de sintonia apresentam uma grande diferença no resultado, porém os métodos convencionais são suficientes para muitos controladores os quais não exijam especificações rigorosas. A sintonia com o uso de

meta-heurísticas é mais adequada para malhas de controle onde os critérios de desempenho exigidos não podem ser alcançados com facilidade pelos métodos convencionais. A comparação mais justa é feita somente entre meta-heurísticas.

4 METODOLOGIA

Este trabalho trata da utilização de meta-heurísticas para a sintonia de controladores PID. Para o seu desenvolvimento e correto funcionamento, algumas etapas devem ser realizadas, desde a pesquisa e escolha das fontes até a coleta a análise dos dados, escolha da função objetivo, sintonia sequencial e simultânea, entre outras. Neste capítulo serão descritos os detalhes para a elaboração deste trabalho.

4.1 AMBIENTE DE SIMULAÇÃO

Todas as simulações das meta-heurísticas para a determinação dos parâmetros do controlador e análise das respostas foram realizadas através do Simulink, uma ferramenta do Matlab para modelagem, simulação e análise de sistemas dinâmicos. O Simulink apresenta uma interface gráfica amigável para o usuário, permitindo trabalhar com diagramas de blocos, onde cada diagrama de bloco representa uma função de transferência ou uma operação matemática.

É importante ressaltar que o Simulink só funciona utilizando o *software* Matlab, da MathWorks®. A versão do Matlab utilizada é a R2016b. O computador utilizado para as simulações possui um processador Intel® Core™ i5-3337U CPU @ 1,80 GHz e 6 GB de memória RAM, e possui o sistema operacional Windows 8.1 de 64 bits.

A função objetivo minimizada é a ITAE acrescida de um fator de punição para sobressinal, visando a obter uma resposta rápida sem sobressinal para o controle de posição do motor CC. Todo erro gerado acima do *set point* será multiplicado por seis, aumentando assim a vantagem das respostas que não possuem sobressinal. O valor do fator de punição do sobressinal foi calculado empiricamente.

As funções que realizam as operações matemáticas de cada algoritmo, ABC, ACO, GA e PSO, foram fornecidas de modo *Open Source* pelo Yarpiz, um recurso de pesquisa acadêmica e científica, focado em inteligência artificial, *Machine Learning*, otimização em engenharia, pesquisa operacional e engenharia de controle. Os códigos estão disponíveis no site www.yarpiz.com.

As funções realizam apenas as operações matemáticas de cada meta-heurística, por isso, foram reescritas para realizar a sintonia de controladores PID, e

foram escritas no Matlab, em formato de *scripts*, assim como a função objetivo, e o diagrama de blocos do sistema de controle com processo, controladores, variável de erro e todos os parâmetros foram feitos no Simulink. As funções principais fazem a chamada da função objetivo, a qual se comunica com o diagrama de blocos do Simulink para a realização das simulações.

4.2 DETERMINAÇÃO DOS PARÂMETROS DAS META-HEURÍSTICAS

Cada uma das quatro meta-heurísticas possui parâmetros de inicialização os quais podem ser alterados de acordo com a necessidade do usuário, como o tamanho da população de indivíduos, partículas, formigas ou abelhas, por exemplo. Esses parâmetros influenciam diretamente no resultado obtido pelas meta-heurísticas. Os parâmetros de inicialização das meta-heurísticas geralmente são encontrados empiricamente (TALBI, 2009), para obter a melhor resposta para determinadas aplicações.

Para a escolha dos parâmetros de inicialização deste trabalho, cada parâmetro foi testado com três valores diferentes, realizando dez simulações com cada valor. Aquele que obteve a menor média aritmética da função objetivo foi o valor do parâmetro utilizado para as comparações entre as meta-heurísticas.

O algoritmo ABC foi testado quanto ao tamanho da população, o número de operárias inativas, o valor do coeficiente de abandono para uma operária ativa se tornar uma escoteira, e o coeficiente de aceleração das abelhas. O algoritmo ACO foi submetido a simulações no tamanho da população, no parâmetro q, relativo ao grau de aleatoriedade do algoritmo, e o valor do coeficiente de abandono ξ. Os parâmetros do GA simulados foram o tamanho da população, a taxa de cruzamento e a taxa de mutação. Já o PSO utilizou os coeficientes de constrição, descritos no item 2.3.2.2, e por isso foi testado somente quanto ao tamanho da população. Após a determinação dos parâmetros de inicialização das quatro meta-heurísticas, foi realizada a comparação entre elas, realizando trinta simulações, a fim de identificar a mais apta a minimizar a função objetivo.

Após a realização da sintonia sequencial dos controladores, foi utilizada a mesma função objetivo para sintonizar os três controladores de forma simultânea. Foram realizadas trinta simulações com cada meta-heurística e os dados obtidos foram comparados com a sintonia sequencial.

Primeiramente cada meta-heurística precisou encontrar dois parâmetros para o controlador PI de corrente, necessário para a modelagem do motor CC. Na segunda etapa, foi feita a sintonia do controlador de velocidade, também na configuração PI, e, por último, a sintonia do controlador de posição, na configuração PD.

4.3 MOTOR DE CORRENTE CONTÍNUA

O motor utilizado no experimento foi um motor de corrente contínua de ímãs permanentes do fabricante Bosch, modelo CPB 9.130.081.051, com tensão nominal de 12 V, potência de 105 W, rotação de 3950 RPM e torque de 0,95 Nm.

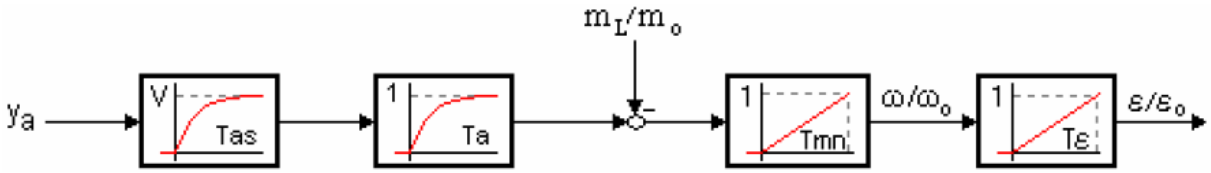
4.3.1 Modelagem Matemática

Para a realização da modelagem matemática do motor CC, os efeitos da força contra eletromotriz foram desprezados para simplificação do sistema. Além disso, foi acrescido ao modelo do motor um circuito de acionamento, para elevar a potência do sinal de controle do motor, com uma constante de tempo T_{as} e a função de transferência dada pela equação 4.1 (SOUZA, 2007):

$$\frac{G_R(S)}{V(S)} = \frac{1}{T_{as} + 1} \tag{4.1}$$

O modelo do motor CC simplificado é mostrado na Figura 24.

Figura 24 - Diagrama de blocos simplificado do motor CC



Fonte: (SOUZA, 2007).

4.3.1.1 Determinação dos parâmetros

A constante de tempo do circuito de acionamento T_{as} foi calculada pela equação 4.2 (SOUZA, 2007):

$$T_{as} = \frac{T_a}{10} \tag{4.2}$$

A constante de tempo da corrente T_a foi determinada seguindo os seguintes passos (SOUZA, 2007):

- Medir a corrente a vazio do motor;
- Medir a tensão de rotor travado até a corrente a vazio do motor;
- Montar o circuito do sensor de corrente;
- Aplicar um degrau de tensão no motor no valor da tensão de rotor travado e verificar a curva de resposta de corrente no osciloscópio.

O valor de T_a será o valor da constante de tempo da curva de corrente, ou seja, o tempo até atingir 63% do estado estacionário da curva.

A constante de tempo da velocidade T_{mn} foi determinada seguindo os seguintes passos (SOUZA, 2007):

- Montar o controle de corrente e o circuito de acionamento do motor;
- Montar o circuito do sensor de velocidade;
- Aplicar um degrau de corrente no motor e verificar a curva de resposta de velocidade no osciloscópio.

O valor de T_{mn} será a constante de tempo da curva de velocidade.

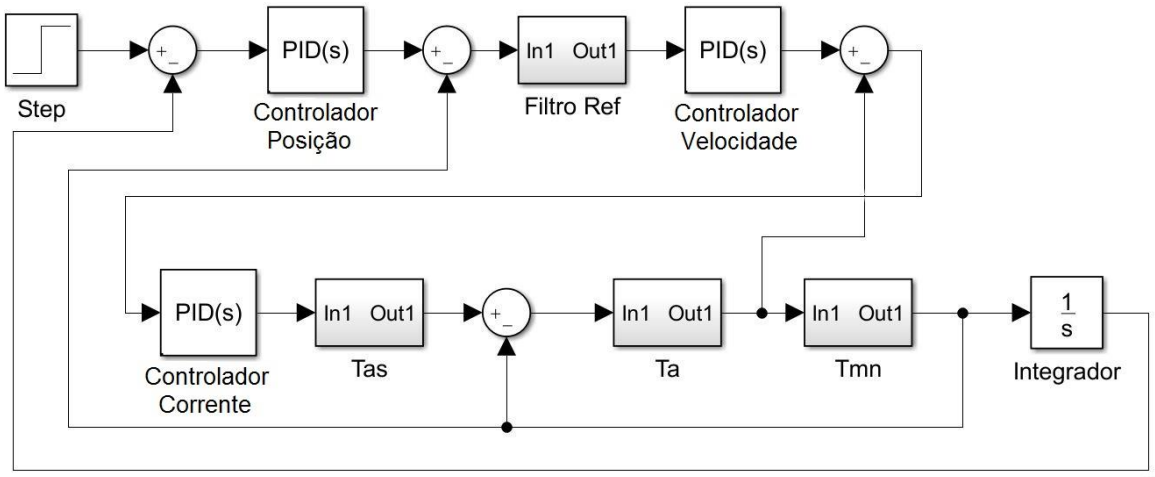
O modelo obtido após a modelagem do motor CC foi utilizado nas simulações para aplicação das meta-heurísticas.

4.4 SISTEMA DE CONTROLE EM CASCATA

O método mais eficaz aceito pela comunidade científica para controle de posição de motores CC é o sistema em cascata (LEONHARD, 1996). O modelo do motor CC foi controlado por um sistema em cascata com três malhas, onde a primeira malha é do controle da corrente, utilizando um controlador PI, a segunda malha controla a velocidade, também com um controlador PI, e a terceira controla a posição do motor, utilizando um controlador PD.

A Figura 25 mostra o diagrama de blocos do sistema completo com os controladores e as malhas de realimentação.

Figura 25 - Diagrama de blocos completo do sistema



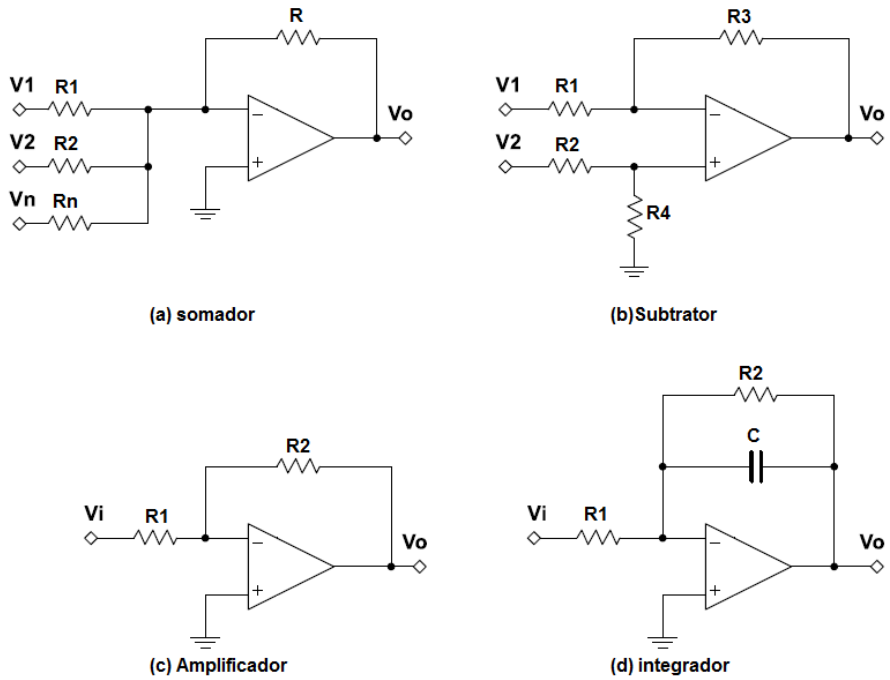
Fonte: Elaborado pelo autor.

Um degrau unitário é aplicado pelo bloco Step. Os blocos dos PID(s) são os controladores de posição, velocidade e corrente. A saída do controlador de posição alimenta a malha de velocidade, após passar pelo filtro de referência, o qual serve para gerar uma rampa de aceleração na entrada a fim de reduzir o pico de corrente na malha de velocidade. O bloco T_{as} é o circuito de acionamento do motor CC, responsável por amplificar o sinal de controle para alimentar o motor. Os blocos T_a, T_{mn} e Integrador fazem parte da modelagem do motor CC.

4.5 MONTAGEM DO CONTROLADOR

O controle do motor CC foi implementado através de circuitos eletrônicos analógicos, utilizando amplificadores operacionais. A Figura 26 mostra os quatro circuitos que foram utilizados para implementação dos controladores: (a) somador; (b) subtrator; (c) amplificador; e (d) integrador.

Figura 26 – Circuitos com amplificadores operacionais



Fonte: Elaborado pelo autor.

As funções de transferência dos circuitos somador, subtrator, amplificador e integrador ideal aparecem nas equações 4.3 a 4.6, respectivamente.

$$V_o = -\left(V_1 \frac{R}{R_1} + V_2 \frac{R}{R_2} + V_n \frac{R}{R_n}\right) \tag{4.3}$$

$$V_o = \frac{R_3}{R_1}(V_2 - V_1) \qquad para \qquad \frac{R_3}{R_1} = \frac{R_4}{R_2} \tag{4.4}$$

$$V_o = -V_i \frac{R_2}{R_1} \tag{4.5}$$

$$V_o = -V_i \frac{1}{sRC} \qquad para \quad R_2 = R_1 = R \tag{4.6}$$

O subtrator será utilizado para gerar o sinal de erro, ou seja, a diferença entre o *set point* e a saída do motor CC. Para o controlador, se utilizou o circuito amplificador como ganho proporcional e o circuito integrador como ganho integral, tendo seus sinais somados pelo circuito somador. Dessa forma, a transformação dos parâmetros do controlador para os componentes do circuito se dará pelas equações 4.7 e 4.8:

$$K_p = \frac{R_2}{R_1} \tag{4.7}$$

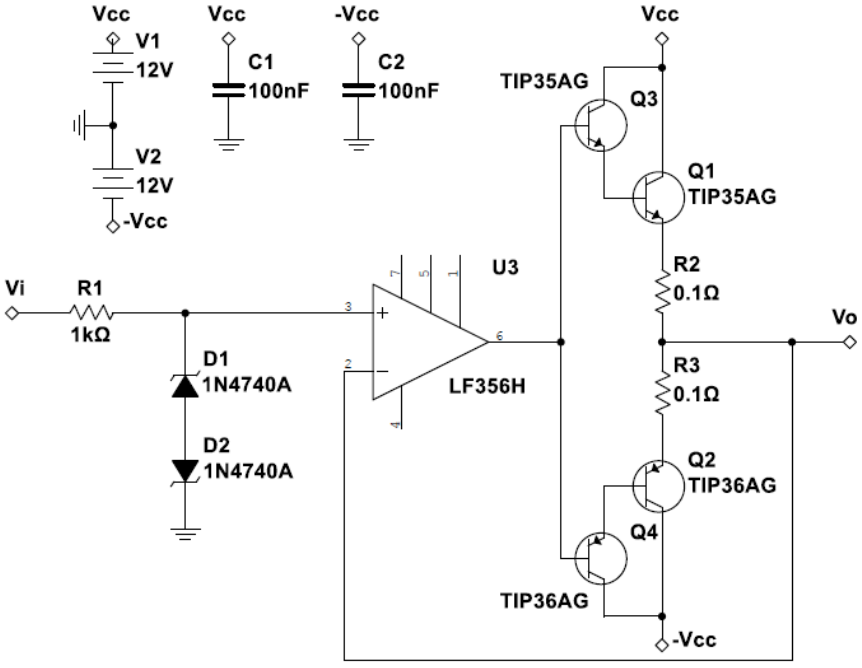
$$K_i = \frac{1}{RC} \tag{4.8}$$

e os componentes foram calculados a partir dos parâmetros do controlador encontrados pela meta-heurística.

4.5.1 Circuito de acionamento

Foi utilizado um circuito de acionamento para amplificar a potência do sinal de controle. Esse circuito é composto por quatro transistores na ligação Darlington, com a topologia push-pull, podendo fornecer uma corrente de 15 A e 40 V de tensão. Na alimentação dos transistores há um amplificador operacional LF356, como seguidor de tensão, e dois diodos zener na entrada do circuito para limitar a tensão entre -10 V e +10 V. A Figura 27 mostra o circuito de acionamento do motor CC.

Figura 27 - Circuito de acionamento do motor CC



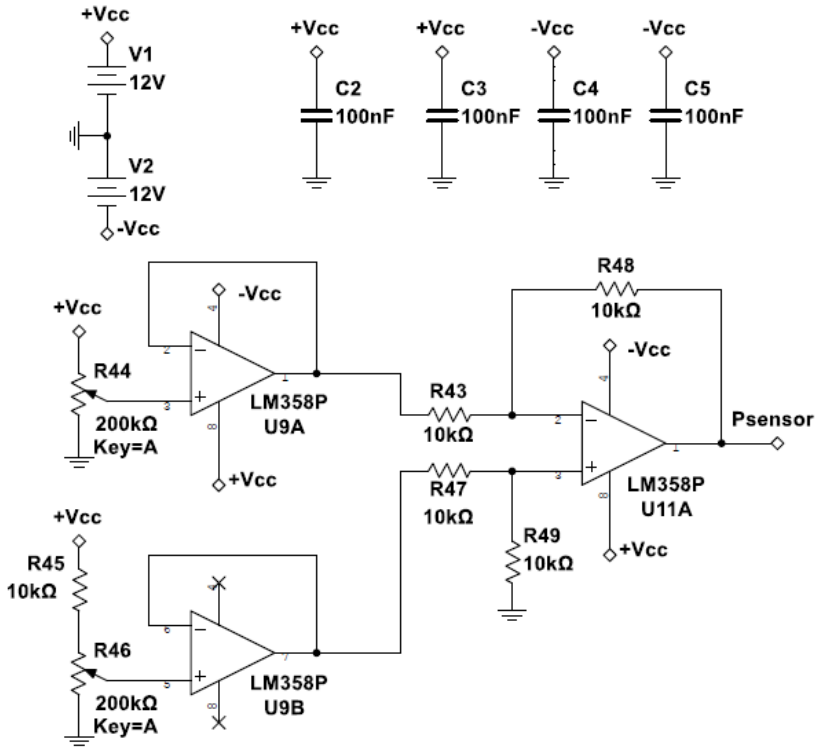
Fonte: Elaborado pelo autor.

4.5.2 Circuitos dos sensores

O circuito controlador deve sempre comparar o sinal de entrada do sistema, a referência, e a saída, para saber o quão distante está do desejado e poder tomar uma ação. Como o sinal de referência é fornecido em tensão elétrica, deve-se compará-lo com um sinal de tensão elétrica. Para isso, se utilizam circuitos transdutores, ou sensores, para converter o sinal desejado, como posição, velocidade ou corrente, em um sinal de tensão elétrica. Os circuitos eletrônicos dos sensores foram montados para fazer a modelagem do motor CC.

O circuito do sensor de posição utilizado é mostrado na Figura 28. Ele utiliza um potenciômetro multivoltas acoplado ao eixo do motor, seguido por um circuito seguidor de tensão. O sinal gerado possui um ajuste de offset, composto por um potenciômetro também ligado a um seguidor de tensão, e ambos ligados a um circuito subtrator.

Figura 28 - Circuito eletrônico do sensor de posição

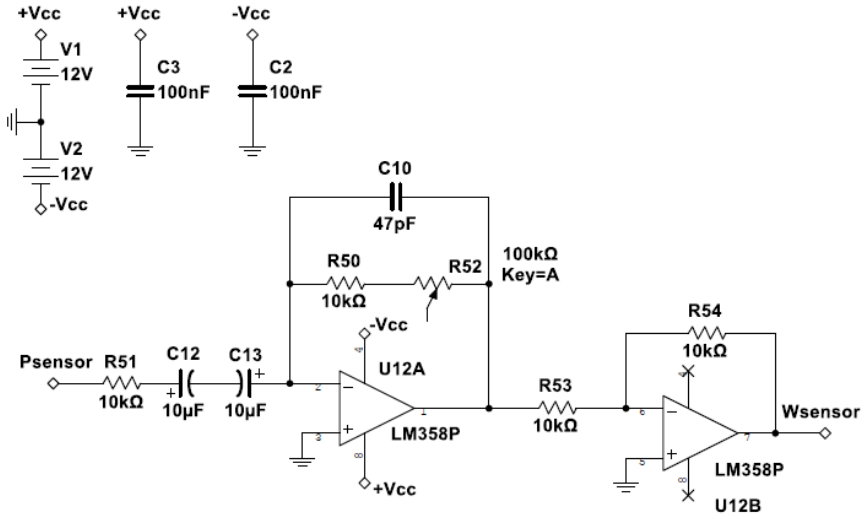


Fonte: Elaborado pelo autor.

O circuito do sensor de velocidade foi feito a partir da derivada do sinal de posição. A saída do sensor de posição foi ligada a um circuito derivativo e um amplificador inversor, obtendo assim o sensor de velocidade do motor. Os

componentes foram dimensionados de maneira empírica. O circuito é mostrado na Figura 29.

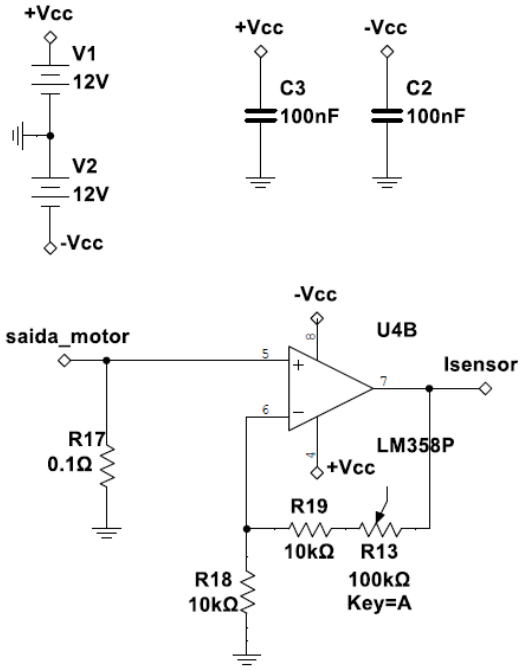
Figura 29 - Circuito eletrônico do sensor de velocidade



Fonte: elaborado pelo autor.

O circuito do sensor de corrente utilizou um resistor *shunt* em série com o motor CC. A corrente que circula pelo motor é a mesma do shunt, assim, a tensão do shunt é proporcional à corrente do motor. O circuito do controle de corrente é mostrado na Figura 30.

Figura 30 - Circuito eletrônico do sensor de corrente



Fonte: Elaborado pelo autor.

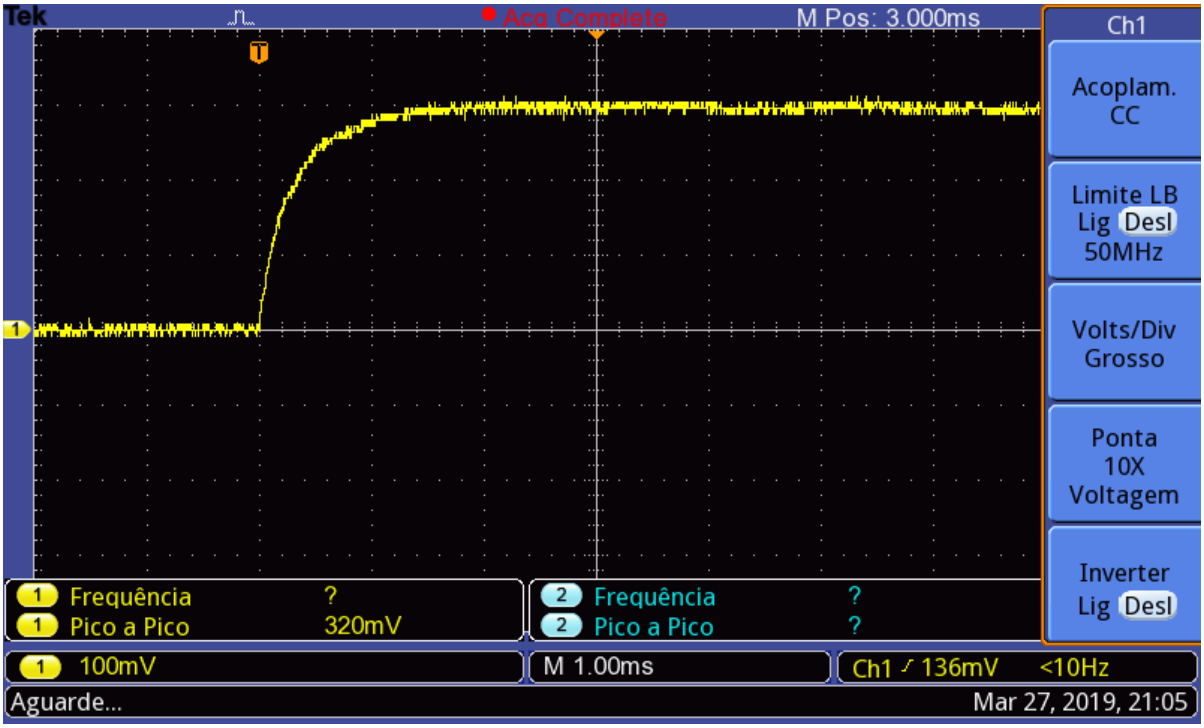
5 ANÁLISE DE RESULTADOS

Este capítulo trata da análise dos resultados de cada etapa do trabalho, verificando sua consistência com os dados esperados e os ajustes necessários para o correto funcionamento do trabalho.

5.1 DETERMINAÇÃO DA CONSTANTE DE TEMPO DA CORRENTE DO MOTOR CC

Para a determinação do primeiro parâmetro do motor CC, a constante de tempo da corrente T_a, foram seguidos os passos descritos no item 4.3.1.1. A corrente a vazio do motor CC encontrada foi de 1,43 A, e a tensão de rotor bloqueado foi de 3,1 V. Aplicando um degrau de 3,1 V ao motor CC, foi encontrada a resposta mostrada na Figura 31.

Figura 31 - Curva de corrente do motor CC a vazio



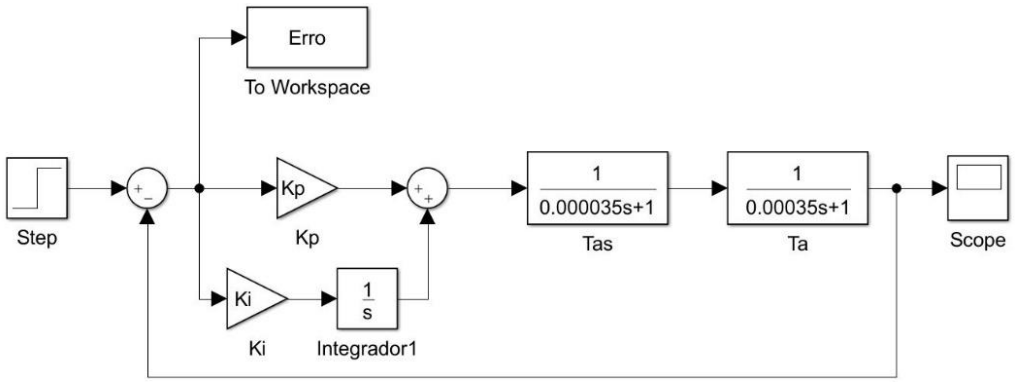
Fonte: Elaborado pelo autor.

A constante de tempo T_a encontrada foi de $350\ \mu s$, e, utilizando a equação 4,2, foi encontrada também a constante de tempo do circuito de acionamento T_{as}, de $35\ \mu s$.

5.2 ANÁLISE DOS PARÂMETROS DAS META-HEURÍSTICAS

Após a determinação da constante de tempo de corrente do motor CC, o sistema foi montado em diagrama de blocos no Simulink para simulação com as meta-heurísticas. O sistema pode ser visto na Figura 32.

Figura 32 - Diagrama de blocos do controle de corrente do motor CC



Fonte: Elaborado pelo autor.

É aplicado um degrau unitário na entrada do sistema através do bloco *Step* e a resposta do sistema pode ser visualizada pelo bloco *Scope*. O bloco *To Workspace* lê o valor do erro da resposta durante a simulação e envia o valor para a função objetivo. O objetivo das meta-heurísticas é encontrar os valores de K_P e K_I que obtenham o menor custo possível da função objetivo. O espaço de busca utilizado foi de [0,30] para o K_P e [0,40000] para o K_I. Para todos os sistemas foram realizadas 30 iterações.

5.2.1 Função Objetivo

Primeiramente foi definido utilizar a função ITAE como função objetivo. Porém, o erro calculado por essa função em tempos muito pequenos não gera tanta importância comparados aos erros com tempo maiores, e como a resposta da corrente é muito rápida, na ordem de microssegundos, a função objetivo produzia uma resposta com tempo de subida muito pequeno, aumentando consequentemente o sobressinal e a oscilação do sistema.

Para evitar esse efeito, foi incluído na função objetivo um fator de punição k para o sobressinal, multiplicando por seis o valor do erro acima do *set point*. Valores altos de k, como $k > 20$, anulam o sobressinal, porém geram erro em estado

estacionário, com tempo muito elevado para estabilização. Já um valor muito pequeno, próximo de um, não é capaz de amortecer o sobressinal do sistema.

5.2.2 Parâmetros da meta-heurística ABC

Para a determinação dos parâmetros da meta-heurística ABC, foram atribuídos três valores diferentes para cada um, mantendo os demais constantes, e realizadas dez simulações. Aquele que obtivesse a menor média aritmética da função objetivo seria o escolhido. Os parâmetros testados foram: número de operárias inativas; coeficiente de abandono; coeficiente de aceleração; e tamanho da população.

5.2.2.1 Abelhas Operárias Inativas

O número de abelhas operárias inativas foi testado com os valores 50%, 16,67%, e 10% da população, os demais parâmetros do algoritmo foram fixados em 60 para o tamanho da população, 1 para o coeficiente de abandono, e 0,2 para o coeficiente de aceleração. Um número elevado de operárias inativas aumenta a busca próxima à melhor aptidão já encontrada, porém um número muito elevado faz com que poucos pontos no espaço de busca sejam explorados, correndo o risco do algoritmo convergir para um ótimo local. A Tabela 3 mostra a média aritmética dos parâmetros da resposta ao degrau, sobressinal – MP, tempo de subida – t_r, tempo de assentamento – t_s, custo da função objetivo, e o valor de K_P e K_I para a melhor configuração encontrada pela variação do parâmetro. O menor custo da função objetivo foi obtido com 50% da população.

Tabela 3 - Parâmetros da resposta ao degrau variando as abelhas operárias inativas - ABC

Operárias Inativas	MP (%)	$t_r\ (\mu s)$	$t_s\ (\mu s)$	Custo	K_P	K_I
Pop/2	1,18	142,50	219,2	7,1192E-04	3,74	10693,88
Pop/6	1,08	144,00	222,5	7,3679E-04	3,70	10536,64
Pop/10	1,21	143,20	219,7	7,2662E-04	3,79	10765,98

Fonte: Elaborado pelo autor.

5.2.2.2 Coeficiente de Abandono

Mantendo agora o número de operárias inativas em 50% da população, se alterou então o coeficiente de abandono, que determina quanto tempo uma operária ativa se mantém em um ponto antes de se tornar uma escoteira e explorar aleatoriamente o espaço de busca. Os valores testados foram 0,2, 0,6 ou 1 vez o tamanho da população vezes o número de variáveis.

O comportamento do coeficiente de abandono é similar ao número de operárias inativas. Um valor elevado faz com que uma operária ativa passe mais tempo explorando o melhor ponto encontrado, podendo ficar preso em um mínimo local, e um valor baixo faz com que os pontos ótimos sejam abandonados rapidamente na busca por outros pontos no espaço de busca, sem explorar a região do ponto ótimo. A Tabela 4 mostra a média aritmética dos parâmetros da resposta ao degrau, sobressinal – MP, tempo de subida – t_r, tempo de assentamento – t_s, custo da função objetivo e o valor de K_P e K_I para a melhor configuração encontrada pela variação do coeficiente de abandono. O valor com o menor custo da função objetivo foi de uma vez o tamanho da população vezes o número de variáveis.

Tabela 4 - Parâmetros da resposta ao degrau variando o coeficiente de abandono - ABC

Coeficiente Abandono	MP (%)	t_r (μs)	t_s (μs)	Custo	K_P	K_I
0,2*nVar*Pop	1,18	142,50	219,2	7,1627E-04	3,79	10740,91
0,6*nVar*Pop	1,08	144,40	222,5	7,1437E-04	3,78	10768,57
1*nVar*Pop	1,21	143,20	219,7	7,0726E-04	3,77	10756,89

Fonte: Elaborado pelo autor.

5.2.2.3 Coeficiente de Aceleração

O coeficiente de aceleração foi testado com os valores 0,2, 0,6 e 1. Esse parâmetro influi no número de iterações para as operárias inativas se aproximarem das operárias ativas, que possuem as melhores aptidões. Valores altos convergem rapidamente, podendo ser em um mínimo local, já valores baixos podem não convergir ou exigir um tempo computacional muito maior. A Tabela 5 mostra a média aritmética dos parâmetros da resposta ao degrau, sobressinal – MP, tempo de subida – t_r, tempo de assentamento – t_s, custo da função objetivo e o valor de K_P e

K_I para a melhor configuração encontrada pela variação do coeficiente de aceleração. O menor custo da função objetivo foi encontrado com o coeficiente de aceleração 0,2.

Tabela 5 - Parâmetros da resposta ao degrau variando o coeficiente de aceleração - ABC

Coeficiente Aceleração	MP (%)	$t_r\ (\mu s)$	$t_s\ (\mu s)$	Custo	K_P	K_I
0,2	1,21	143,30	220,1	7,1175E-04	3,70	10538,00
0,6	1,65	136,50	240,5	7,4744E-04	3,76	10753,77
1	3,59	125,26	230,0	8,0296E-04	3,94	11218,03

Fonte: Elaborado pelo autor.

5.2.2.4 Tamanho da População

O tamanho da população foi testado com os valores 20, 60 e 100. Quanto maior a população, mais pontos no espaço de busca são explorados, porém valores muito altos para a população fazem com que a influência das melhores aptidões já encontradas não seja tão significativa, e o algoritmo exija um tempo computacional muito maior, por aumentar o tempo de cada simulação e necessitar mais iterações para convergir. A Tabela 6 mostra a média aritmética dos parâmetros da resposta ao degrau, sobressinal – MP, tempo de subida – t_r, tempo de assentamento – t_s, custo da função objetivo e o valor de K_P e K_I para a melhor configuração encontrada pela variação do tamanho da população. O menor custo da função objetivo foi obtido com uma população de 100 abelhas.

Tabela 6 - Parâmetros da resposta ao degrau variando o tamanho da população - ABC

População	MP (%)	$t_r\ (\mu s)$	$t_s\ (\mu s)$	Custo	K_P	K_I
20	1,32	141,20	234,7	7,3308E-04	3,68	10480,70
60	1,32	140,70	215,4	7,1330E-04	3,76	10747,27
100	1,14	144,60	222,7	7,0734E-04	3,77	10762,92

Fonte: Elaborado pelo autor.

5.2.3 Parâmetros da meta-heurística ACO

Os parâmetros da meta-heurística ACO foram obtidos pelo mesmo método que a ABC, cada parâmetro foi testado com três valores, mantendo os demais

constantes e realizando dez simulações com cada valor para cada parâmetro. Os parâmetros testados foram: tamanho da população; parâmetro *q*; e coeficiente de evaporação.

5.2.3.1 Tamanho da População

Os valores para teste do tamanho da população foram 20, 60 e 100, os demais parâmetros do algoritmo foram fixados em 1 para o parâmetro *q* e 1 para o coeficiente de evaporação. O comportamento do algoritmo com a variação da população é idêntico para todos os algoritmos, conforme descrito na meta-heurística ABC. A Tabela 7 mostra a média aritmética dos parâmetros da resposta ao degrau, sobressinal – MP, tempo de subida – t_r, tempo de assentamento – t_s, custo da função objetivo e o valor de K_P e K_I para a melhor configuração encontrada pela variação do tamanho da população. O menor custo da função objetivo foi obtido com uma população de 100 formigas.

Tabela 7 - Parâmetros da resposta ao degrau variando o tamanho da população - ACO

População	MP (%)	t_r (μs)	t_s (μs)	Custo	K_P	K_I
20	21,86	45,60	253,4	1,1353E-03	13,61	36388,47
60	4,50	108,47	265,1	1,0219E-03	3,88	10198,09
100	6,52	104,38	259,7	9,7786E-04	4,47	12478,33

Fonte: Elaborado pelo autor.

5.2.3.2 Coeficiente *q*

Segundo SOCHA e DORIGO (2008), o parâmetro *q* serve para determinar a relação Diversificação x Intensificação do sistema. Um algoritmo de otimização robusto deve possuir meios de diversificar a população para não ficar preso em um ótimo local, ao mesmo tempo que deve intensificar a busca em um ponto ótimo caso tenha encontrado o ótimo global. Quando o valor de *q* se aproxima de 0, somente a gaussiana com a melhor aptidão é utilizada para gerar novas soluções. Ao aumentar o valor de *q*, mais gaussianas com soluções razoavelmente boas são escolhidas, tornando o sistema mais diversificado e mais robusto, gerando, porém, um aumento no tempo computacional para convergência. A Tabela 8 mostra a média aritmética dos parâmetros da resposta ao degrau, sobressinal – MP, tempo de subida – t_r,

tempo de assentamento – t_s, custo da função objetivo e o valor de K_P e K_I para a melhor configuração encontrada pela variação do parâmetro *q*. O parâmetro *q* foi testado com os valores de 0,2, 0,6 e 1, obtendo o menor custo da função objetivo com o valor 0,6.

Tabela 8 - Parâmetros da resposta ao degrau variando o parâmetro *q* - ACO

Parâmetro q	MP (%)	t_r (μs)	t_s (μs)	Custo	K_P	K_I
0,2	9,03	89,34	257,7	1,0502E-03	4,78	13460,45
0,6	2,41	112,99	257,4	8,8771E-04	3,81	10525,00
1	2,06	130,28	252,0	9,6916E-04	3,49	10207,42

Fonte: Elaborado pelo autor.

5.2.3.3 Coeficiente de Evaporação

O coeficiente de evaporação afeta a predileção pelas piores soluções, diminui as chances delas serem escolhidas através das iterações. Quanto maior o coeficiente de evaporação, menos as piores soluções são escolhidas na geração das novas soluções, porém, maior o tempo necessário para convergência. Os valores testados para o coeficiente de evaporação foram 0,2, 0,6 e 1. A Tabela 9 mostra a média aritmética dos parâmetros da resposta ao degrau, sobressinal – MP, tempo de subida – t_r, tempo de assentamento – t_s, custo da função objetivo e o valor de K_P e K_I para a melhor configuração encontrada pela variação do coeficiente de evaporação. O menor custo da função objetivo se encontrou em 0,6.

Tabela 9 - Parâmetros da resposta ao degrau variando o coeficiente de evaporação - ACO

Coeficiente Evaporação	MP (%)	t_r (μs)	t_s (μs)	Custo	K_P	K_I
0,2	10,23	92,84	264,5	1,0262E-03	4,05	11118,98
0,6	7,17	96,40	252,2	1,0120E-03	4,09	11853,32
1	10,33	89,14	254,2	1,0737E-03	5,04	13126,55

Fonte: Elaborado pelo autor.

5.2.4 Parâmetros da meta-heurística GA

Os parâmetros testados para o GA foram: tamanho da população; taxa de cruzamento; e taxa de mutação, todos obtidos da mesma maneira que os demais

sendo testados com três valores, mantendo os demais constantes e realizando dez simulações com cada valor para cada parâmetro.

5.2.4.1 Tamanho da População

Os valores para teste do tamanho da população foram 20, 60 e 100, os demais parâmetros do algoritmo foram fixados em 70% para a taxa de cruzamento e 6% para a taxa de mutação. O comportamento do algoritmo com a variação da população é idêntico para todos os algoritmos, conforme descrito na meta-heurística ABC. A Tabela 10 mostra a média aritmética dos parâmetros da resposta ao degrau, sobressinal – MP, tempo de subida – t_r, tempo de assentamento – t_s, custo da função objetivo e o valor de K_P e K_I para a melhor configuração encontrada pela variação do tamanho da população. O menor custo da função objetivo foi obtido com uma população de 100 indivíduos.

Tabela 10 - Parâmetros da resposta ao degrau variando o tamanho da população - GA

População	MP (%)	t_r (μs)	t_s (μs)	Custo	K_P	K_I
20	17,95	62,77	253,0	1,0877E-03	4,71	12813,34
60	14,06	78,36	248,5	1,0228E-03	3,76	10677,55
100	8,11	103,38	247,6	9,1432E-04	3,81	10857,07

Fonte: Elaborado pelo autor.

5.2.4.2 Taxa de Cruzamento

A taxa de cruzamento no GA é utilizada para determinar quantos indivíduos da geração atual serão pais para as gerações futuras. Utilizando um método de seleção, os indivíduos com maior aptidão são escolhidos, intensificando a busca próximo às melhores soluções. Quanto maior a taxa de cruzamento, mais rápido o algoritmo converte, e maiores as chances de acabar em um mínimo local. Os valores testados foram 60%, 70% e 80% da população. A Tabela 11 mostra a média aritmética dos parâmetros da resposta ao degrau, sobressinal – MP, tempo de subida – t_r, tempo de assentamento – t_s, custo da função objetivo e o valor de K_P e K_I para a melhor configuração encontrada pela variação da taxa de cruzamento. O menor custo da função objetivo foi encontrado com a taxa de cruzamento de 60% da população.

Tabela 11 - Parâmetros da resposta ao degrau variando a taxa de cruzamento - GA

Taxa Cruzamento	MP (%)	$t_r\ (\mu s)$	$t_s\ (\mu s)$	Custo	K_P	K_I
0,6	3,33	131,16	227,9	8,3075E-04	3,80	10735,21
0,7	8,11	103,38	247,6	9,1432E-04	3,81	10857,07
0,8	5,69	115,52	232,5	8,8717E-04	3,69	10485,28

Fonte: Elaborado pelo autor.

5.2.4.3 Taxa de Mutação

Para equilibrar os efeitos da taxa de cruzamento, que intensifica as buscas próximas do melhor valor já encontrado, há a taxa de mutação, que altera aleatoriamente um indivíduo a fim de diversificar a população e encontrar valores melhores que os já obtidos até então em pontos diferentes no espaço de busca, funcionando como um meio de escapar de mínimos locais. Porém, uma taxa de mutação muito alta pode tornar o algoritmo muito aleatório, fazendo com que ele não seja capaz de convergir para um ótimo global. A Tabela 12 mostra a média aritmética dos parâmetros da resposta ao degrau, sobressinal – MP, tempo de subida – t_r, tempo de assentamento – t_s, custo da função objetivo e o valor de K_P e K_I para a melhor configuração encontrada pela variação da taxa de mutação. Os valores testados para a taxa de mutação foram 2%, 6% e 10% da população, e a melhor aptidão foi encontrada com a taxa de mutação de 10% da população.

Tabela 12 - Parâmetros da resposta ao degrau variando a taxa de mutação - GA

Taxa Mutação	MP (%)	$t_r\ (\mu s)$	$t_s\ (\mu s)$	Custo	K_P	K_I
0,02	4,25	114,78	248,9	9,1959E-04	3,81	11013,39
0,06	3,08	126,36	224,9	1,0498E-03	3,66	10259,45
0,1	12,97	77,52	255,1	9,0308E-04	3,87	10986,97

Fonte: Elaborado pelo autor.

5.2.5 Parâmetros da meta-heurística PSO

Para a meta-heurística PSO foram utilizados os coeficientes de constrição descritos no item 2.3.2.2. Dessa forma, o único parâmetro do PSO testado foi o tamanho da população, utilizando a mesma metodologia dos algoritmos anteriores, sendo testado com três valores e realizando dez simulações com cada valor.

5.2.5.1 Tamanho da População

Os valores para teste do tamanho da população foram 20, 60 e 100. O comportamento do algoritmo com a variação da população é idêntico para todos os algoritmos, conforme descrito na meta-heurística ABC. A Tabela 10 mostra a média aritmética dos parâmetros da resposta ao degrau, sobressinal – MP, tempo de subida – t_r, tempo de assentamento – t_s, custo da função objetivo e o valor de K_P e K_I para a melhor configuração encontrada pela variação do tamanho da população. O menor custo da função objetivo foi obtido com uma população de 100 partículas.

Tabela 13 - Parâmetros da resposta ao degrau variando o tamanho da população - PSO

População	MP (%)	t_r (μs)	t_s (μs)	Custo	K_P	K_I
20	11,38	85,35	241,1	9,6184E-04	3,72	10631,54
60	9,43	105,34	235,3	8,7472E-04	3,75	10718,86
100	3,23	134,86	225,5	7,4417E-04	3,75	10712,92

Fonte: Elaborado pelo autor.

5.3 SINTONIA DO CONTROLADOR DE CORRENTE

Após concluída a etapa de determinação dos parâmetros de cada meta-heurística, deve ser feita a comparação entre as quatro para saber qual obtém os melhores resultados na sintonia do controlador da corrente do motor CC. A Tabela 14 mostra os parâmetros finais de cada meta-heurística, os quais foram utilizados na comparação.

Tabela 14 - Parâmetros finais das meta-heurísticas

ABC		ACO		GA		PSO	
População	100	População	100	População	100	População	100
Operárias inativas	Pop/2	Parâmetro q	0,6	Taxa cruzamento	0,6	-	-
Coeficiente abandono	1	Coeficiente evaporação	0,6	Taxa mutação	0,1	-	-
Coeficiente aceleração	0,2	-	-	-	-	-	-

Fonte: Elaborado pelo autor.

A comparação entre as quatro meta-heurísticas foi feita realizando trinta simulações com cada uma, e escolhendo a que tivesse a menor média aritmética da função objetivo. A Tabela 15 mostra os valores médios encontrados pelas meta-heurísticas para o sobressinal MP, tempo de subida t_r, tempo de assentamento t_s, custo da função objetivo e o valor de K_P e K_I para a melhor configuração encontrada pelas meta-heurísticas.

Tabela 15 - Respostas do controle de corrente do motor CC por meta-heurística

Meta-heurística	MP (%)	$t_r\ (\mu s)$	$t_s\ (\mu s)$	Custo	K_P	K_I
ABC	1,1621	143,80	221,17	7,09E-04	3,77	10762,92
ACO	4,8979	109,97	252,43	9,66E-04	3,50	10045,49
GA	5,0309	119,02	236,00	8,87E-04	3,85	11000,49
PSO	1,1610	144,83	222,43	7,01E-04	3,75	10713,57

Fonte: Elaborado pelo autor.

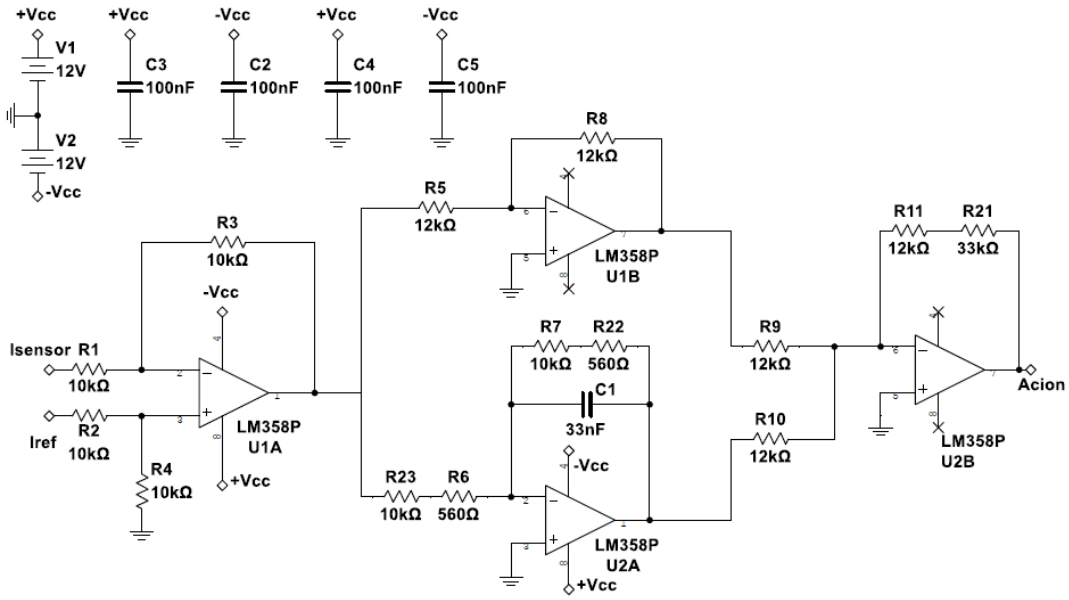
Como o objetivo final é o controle de posição do motor CC com o menor sobressinal possível, foi escolhida a meta-heurística PSO, que obteve o menor sobressinal e o menor custo da função objetivo. Os parâmetros da meta-heurística PSO serão utilizados para o controle de corrente do motor CC. Os parâmetros encontrados foram $K_P = 3{,}75$ e $K_I = 10713{,}57$.

5.4 MONTAGEM DO CONTROLADOR DE CORRENTE

Após a definição dos parâmetros do controlador serem escolhidos, os componentes do circuito eletrônico foram calculados utilizando as fórmulas 4.7 e 4.8. O circuito do controlador é composto por um subtrator para gerar o sinal de erro, seguido por um amplificador e um integrador em paralelo, e um somador. Os valores dos resistores e capacitores foram ajustados para valores da série E12. O ganho K_P foi colocado no somador.

O circuito final do controlador de corrente do motor CC é mostrado na Figura 33

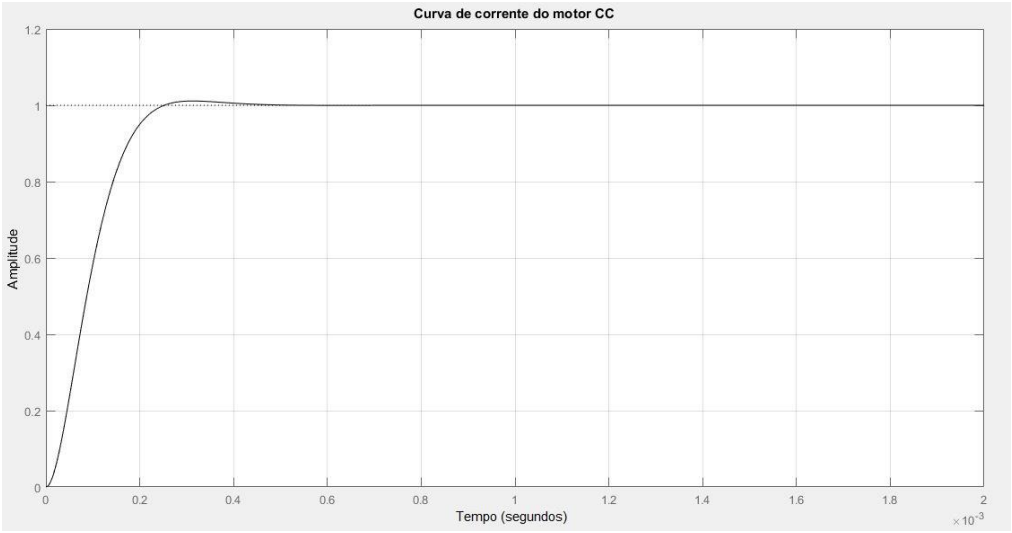
Figura 33 - Circuito de controle de controle do motor CC



Fonte: Elaborado pelo autor.

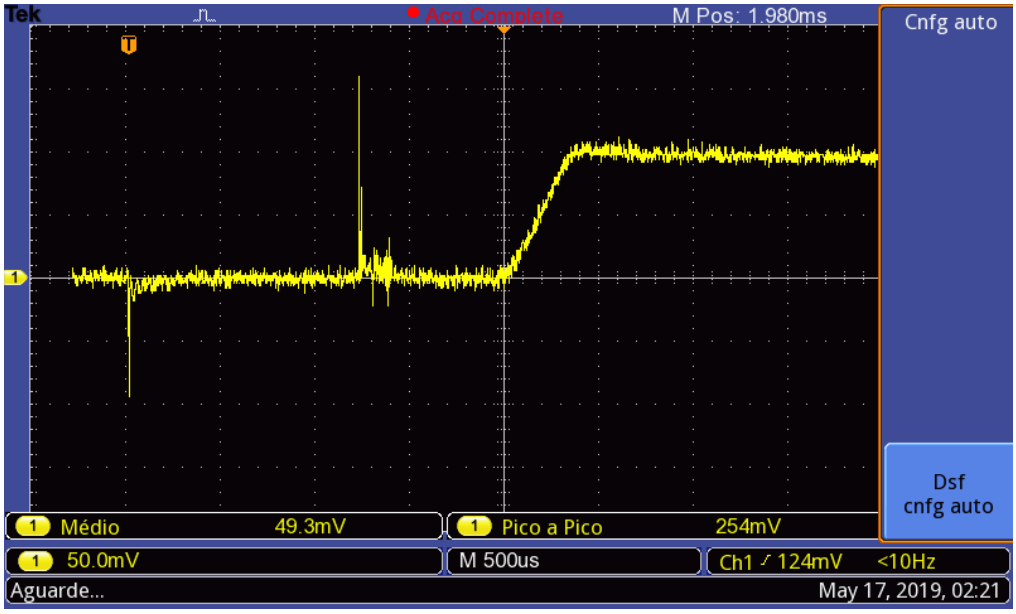
O circuito do controlador de corrente foi montado em *protoboard* e ligado ao motor CC para levantamento da curva de corrente. A Figura 34 e a Figura 35 mostram, respectivamente, a curva esperada para o controlador projetado e a curva obtida no osciloscópio com o controlador de corrente ligado ao motor CC.

Figura 34 – Resposta ao degrau do controle de corrente do motor CC



Fonte: Elaborado pelo autor.

Figura 35 – Curva de corrente obtida no osciloscópio

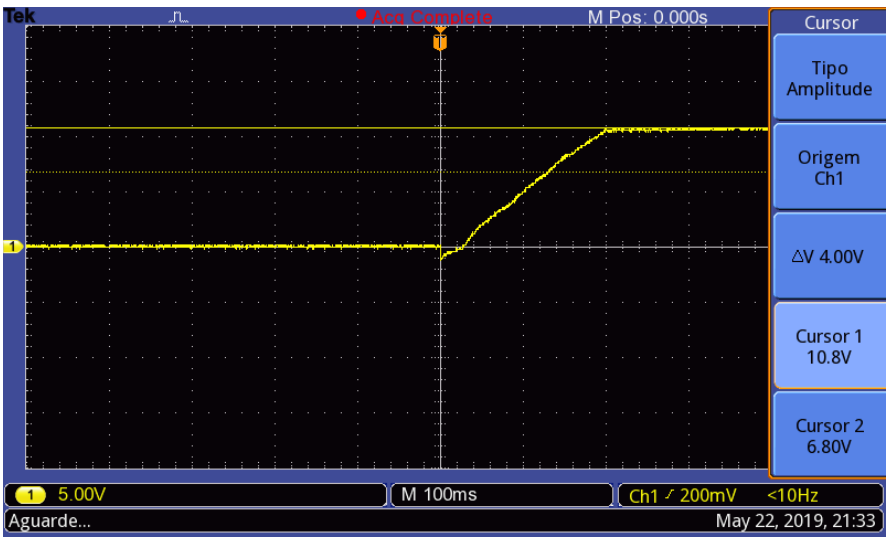


Fonte: Elaborado pelo autor.

5.5 DETERMINAÇÃO DA CONSTANTE DE TEMPO DA VELOCIDADE DO MOTOR CC

A constante de tempo da velocidade T_{mn} do motor CC foi determinada seguindo os passos descritos no item 4.3.1.1. A Figura 36 mostra a curva de velocidade obtida no osciloscópio aplicando um degrau de corrente no motor CC.

Figura 36 - Curva de velocidade do motor CC



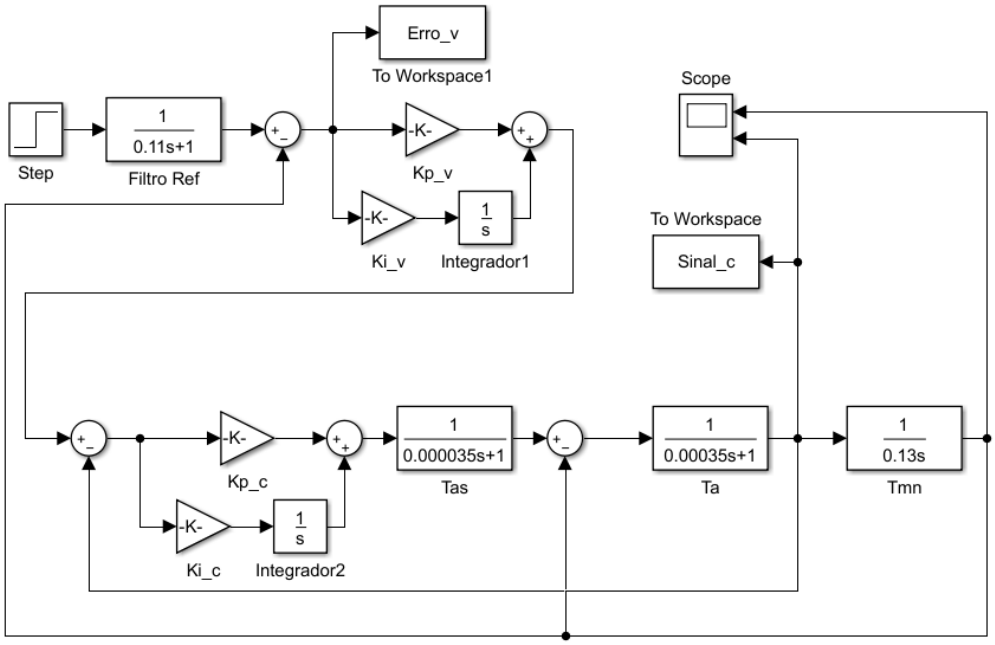
Fonte: Elaborado pelo autor.

O valor da constante de tempo T_{mn} encontrado foi de 130 ms.

5.6 SINTONIA DO CONTROLADOR DE VELOCIDADE

Para a realização da sintonia do controlador de velocidade, foi utilizado um controlador PI. Ao realizar o controle da velocidade do motor CC, um pico indesejado de corrente ocorre. Para amenizar esse efeito, duas ações foram tomadas. A primeira foi inserir no sistema um filtro de referência, com a função de gerar uma rampa de aceleração na entrada e diminuir, assim, o pico de corrente. A segunda foi alterar a função objetivo para, além de minimizar o erro da malha de velocidade, também minimizar o pico do sinal de corrente, com um fator de punição $k = 6$ para valores de corrente acima de 1. O espaço de busca utilizado foi de [0,10] para o parâmetro K_P e [0,50] para o K_I. A Figura 37 mostra o diagrama de blocos utilizado.

Figura 37 - Diagrama de blocos do controle de velocidade do motor CC



Fonte: Elaborado pelo autor.

O bloco do filtro de referência foi inserido após o degrau de entrada. Após o fechamento da malha de realimentação está o controlador de velocidade. O bloco Erro_v lê o sinal do erro de velocidade e envia para a função objetivo, assim como o bloco Sinal_c faz com a corrente. A saída do controlador de velocidade alimenta a malha do controle de corrente. O bloco t_{mn} foi inserido ao sistema do motor CC.

A comparação entre as quatro meta-heurísticas foi feita realizando dez simulações com cada uma, e escolhendo a que tivesse a menor média aritmética da

função objetivo. A Tabela 16 mostra os valores médios encontrados pelas meta-heurísticas para o tempo de subida t_r, tempo de assentamento t_s custo da função objetivo e o valor de K_P e K_I para a melhor configuração encontrada pelas meta-heurísticas. O sobressinal foi nulo para todas as comparações.

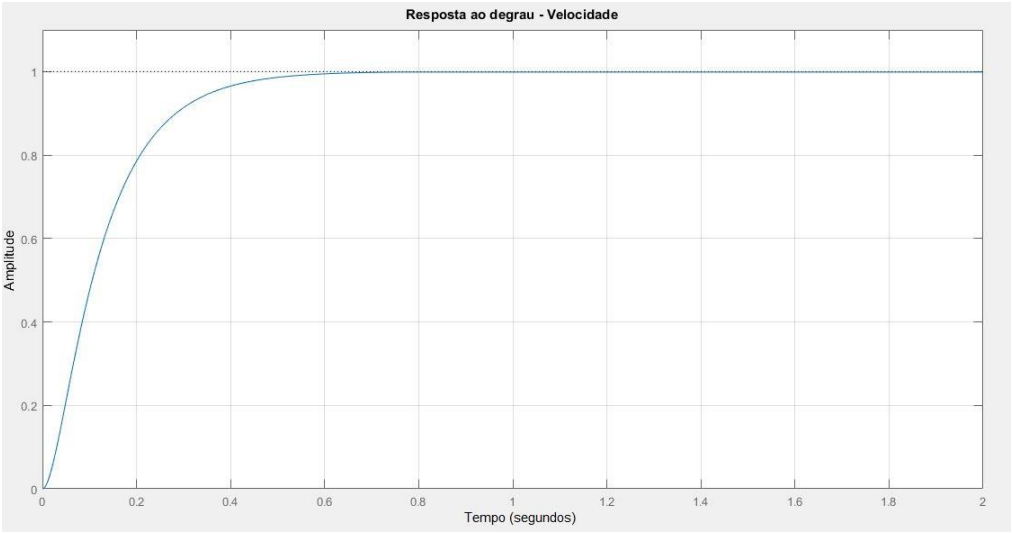
Tabela 16 - Respostas do controle de velocidade do motor CC por meta-heurística

Meta-heurística	$t_r\ (ms)$	$t_s\ (ms)$	Custo	K_P	K_I
ABC	254,0	461,0	2,14014	4,83	0,00
ACO	253,5	459,4	2,12649	4,82	0,05
GA	253,6	459,6	2,13304	4,82	0,04
PSO	250,0	451,0	2,18546	4,83	0,12

Fonte: Elaborado pelo autor.

Como pode-se ver na tabela, o PSO ficou com o menor valor para o tempo de subida e o tempo de assentamento, porém teve o maior custo da função objetivo. O menor custo da função objetivo foi obtido pelo ACO. Isso ocorre porque a resposta mais rápida causa um pico maior de corrente, aumentando assim o custo da função objetivo. Os parâmetros da meta-heurística ACO serão utilizados para o controle de corrente do motor CC. Os parâmetros encontrados foram $K_P = 4{,}82$ e $K_I = 0{,}05$. A Figura 38 mostra a resposta ao degrau do controle de velocidade do motor CC. O sobressinal encontrado foi de 0%, o tempo de subida de $254\ ms$ e o tempo de acomodação de $460\ ms$.

Figura 38 - Resposta ao degrau do controle de velocidade do motor CC



Fonte: Elaborado pelo autor.

5.7 SINTONIA DO CONTROLADOR DE POSIÇÃO

Segundo Leonhard (1996), para um sistema em cascata, o controle de posição pode ser feito por um controlador P. Para a realização da sintonia do controlador de posição, dois modelos foram utilizados, o primeiro utilizando um controlador P e o segundo com um controlador PD e feita a comparação, realizando dez simulações com cada um para cada meta-heurística. Foram realizadas simulações com um controlador PI também, porém as quatro meta-heurísticas encontraram valor nulo para a ação integral, portanto, não é necessário incluir ação integral no controlador de posição do sistema em cascata do motor CC.

A função objetivo do controle de posição seguiu o mesmo modelo do controle de velocidade, buscando minimizar o erro de posição e os erros de velocidade e de posição também. O espaço de busca utilizado foi de [0,10] para o parâmetro K_P nos dois modelos e [0,1] para o K_D no controlador PD. A Tabela 17 mostra a comparação entre o controlador P e o PD para controle da posição do motor CC.

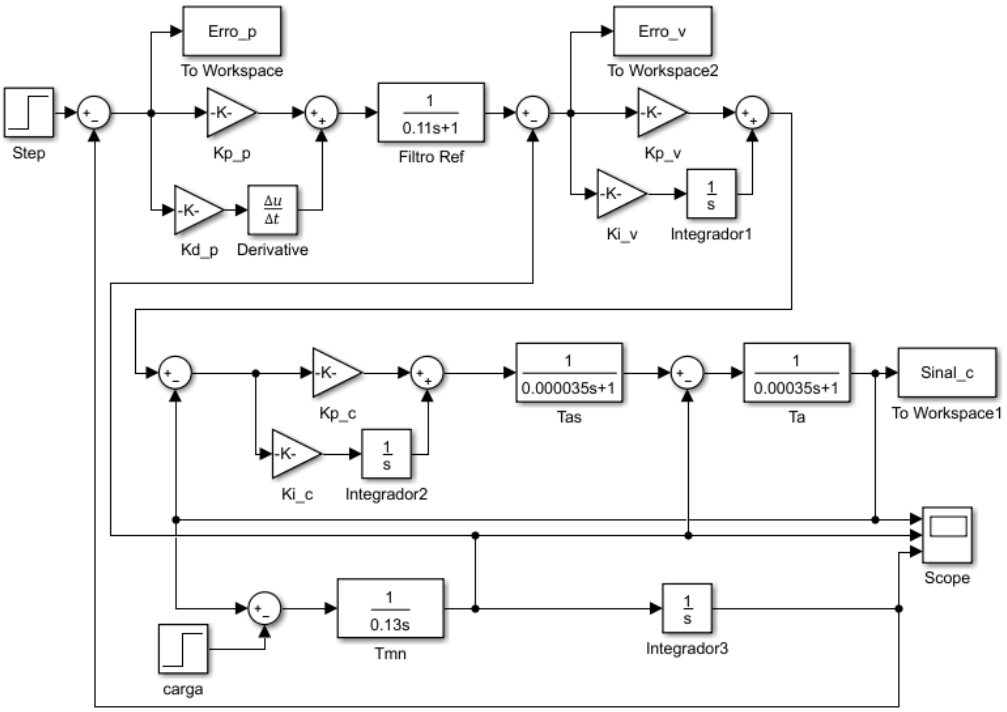
Tabela 17 - Controlador de posição P e PD para o motor CC

Controlador	Meta-heurística	$t_r\ (ms)$	$t_s\ (ms)$	Custo	K_P	K_D
P	ABC	476,5	775,9	116,4176	3,02	-
	ACO	483,4	866,6	116,4592	2,98	-
	GA	486,4	874,8	116,4767	2,94	-
	PSO	468,2	824,6	116,3559	2,97	-
PD	ABC	437,1	966,1	110,6827	5,30	0,96
	ACO	443,3	977,2	110,6373	5,30	0,96
	GA	456,2	1002,3	110,6688	5,29	0,97
	PSO	438,7	744,6	111,0769	5,29	0,96

Fonte: Elaborado pelo autor.

Como pode ser observado, para a mesma função objetivo, o controlador PD encontrou uma sintonia com um custo menor que o controlador P, por isso, esse sistema foi utilizado na simulação do controle de posição do motor CC. A Figura 39 mostra o diagrama de blocos do controle de posição do motor CC.

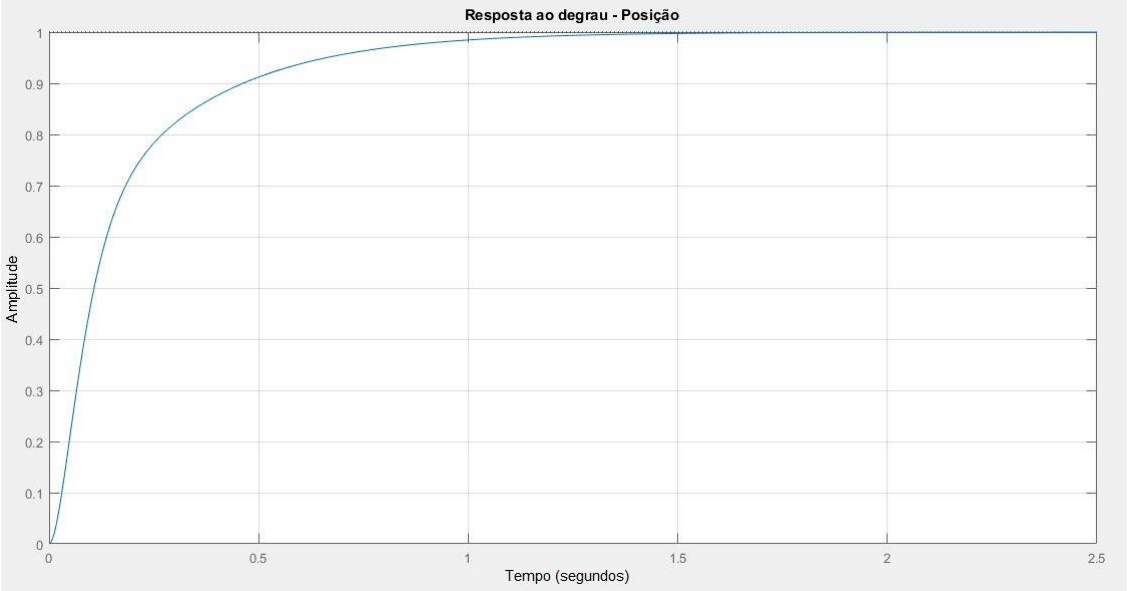
Figura 39 - Diagrama de blocos do controle de posição do motor CC



Fonte: Elaborado pelo autor.

Dentro do controle PD, a meta-heurística que obteve o menor custo da função objetivo foi a PSO. Os parâmetros do controlador escolhidos foram $K_P = 5{,}29$ e $K_D = 0{,}96$. A Figura 40 mostra a resposta ao degrau da posição controlada do motor CC com os parâmetros sintonizados pela meta-heurística. O sobressinal encontrado foi de 0%, o tempo de subida de $432\ ms$ e o tempo de acomodação de $925\ ms$.

Figura 40 - Resposta ao degrau do controle de posição do motor CC



Fonte: Elaborado pelo autor.

5.8 SINTONIA SIMULTÂNEA DOS CONTROLADORES

Após a sintonia sequencial dos controladores do motor CC, foi feita a sintonia simultânea, ou seja, as meta-heurísticas foram submetidas à sintonia dos seis parâmetros dos três controladores ao mesmo tempo, sendo um desafio muito maior para as meta-heurísticas. O diagrama de blocos é o mesmo do controle de posição do motor CC, assim como o espaço de busca de cada parâmetro. Foram realizadas trinta simulações com cada meta-heurística, e comparados o tempo de subida, tempo de assentamento e custo da função objetivo. O sobressinal foi nulo para todas as simulações. A Tabela 18 mostra o resultado das simulações.

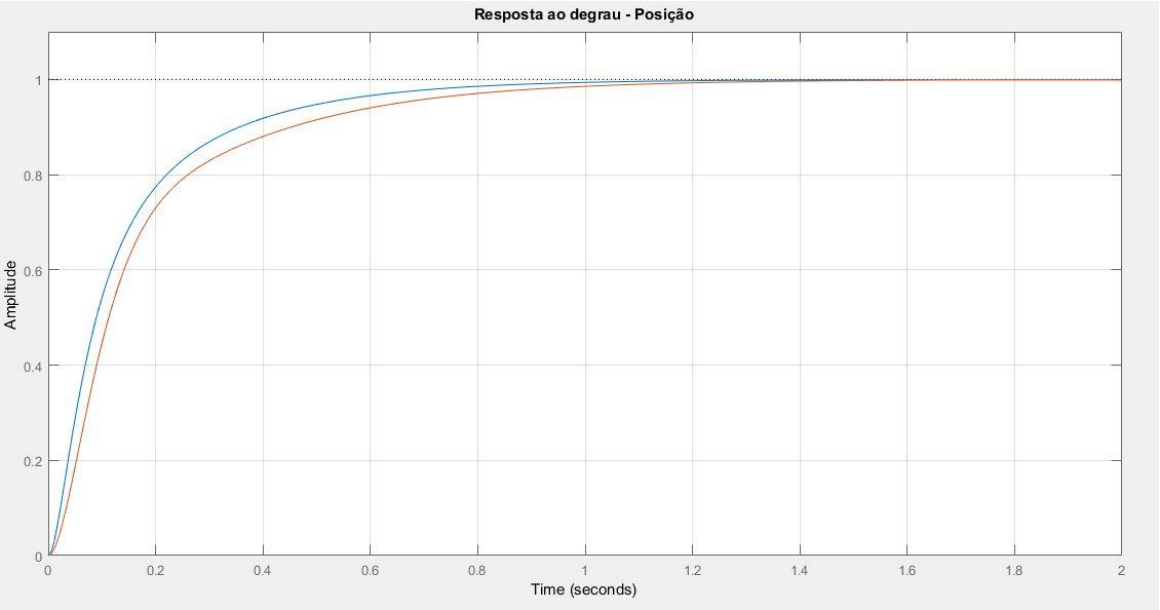
Tabela 18 - Respostas do controle simultâneo dos controladores do motor CC

Meta-heurística	t_r (ms)	t_s (ms)	Custo	K_P corrente	K_I corrente	K_P velocidade	K_I velocidade	K_P posição	K_D posição
ABC	338,27	718,6	105,7464	3,49	23507,27	10,00	45,41	6,19	0,81
ACO	335,83	715,5	105,0958	3,03	15946,39	10,00	50,00	6,61	0,93
GA	361,43	755,0	105,3752	2,21	19489,66	10,00	49,87	6,21	0,80
PSO	327,27	695,1	105,2526	2,51	18053,07	10,00	48,38	6,42	0,79

Fonte: Elaborado pelo autor.

A meta-heurística que atingiu o menor custo da função objetivo foi a ACO. Os parâmetros encontrados para o controlador de corrente foram $K_P = 3{,}03$ e $K_I = 15946{,}39$, para o controlador de velocidade foram $K_P = 10$ e $K_I = 50$, e para o controlador de posição foram $K_P = 6{,}61$ e $K_D = 0{,}93$. A Figura 41 mostra a resposta ao degrau da posição do motor CC sintonizado simultaneamente pela meta-heurística ACO, em azul, em comparação com a resposta ao degrau da sintonia sequencial, em vermelho. O sobressinal da resposta simultânea é nulo, o tempo de subida é de $254\ ms$ e o tempo de assentamento é de $724\ ms$.

Figura 41 - Respostas ao degrau sequencial e simultânea do controle de posição do motor CC



Fonte: Elaborado pelo autor.

Apesar da sintonia dos parâmetros de forma simultânea ser mais difícil, por ter seis variáveis ao invés de duas, aumentando a complexidade do processo, os resultados mostraram que as meta-heurísticas encontraram custos menores para a mesma função objetivo na sintonia simultânea. Isso ocorre devido a alguns fatores. O primeiro deles é que, tendo os dois primeiros controladores já sintonizados em valores definidos, o espaço de busca do controle de posição é menor e, portanto, alguns pontos de mínimos ficam de fora dele. Além disso, a função objetivo utilizada nos dois primeiros controladores tinha o objetivo de minimizar os erros de corrente e velocidade, sem se preocupar com a posição ao final do processo.

5.9 TEMPO DE SIMULAÇÃO

Ao falar-se sobre as meta-heurísticas, um dos pontos negativos é o tempo computacional necessário para que as executem. Quanto mais rápido um algoritmo resolve um problema, mais eficaz ele é. Os principais fatores que influenciam no tempo de execução de um algoritmo são o tamanho da população, o número de iterações, o número de variáveis e o tempo de simulação. As simulações comparativas desde trabalho contaram com uma população de cem indivíduos, trinta iterações, duas variáveis para os sequenciais e seis variáveis para o simultâneo. O tempo da simulação para a sintonia dos controladores PID deve ser tal que o

sistema atinja o estado estacionário. O controlador de corrente foi simulado por um tempo de $1{,}5\ ms$, o controlador de velocidade com $1\ s$ e o de posição e o simultâneo com $2\ s$, assim. A Tabela 19 mostra o tempo aproximado de simulação de cada meta-heurística para os controladores.

Tabela 19 - Tempo de simulação para as meta-heurísticas

Meta-heurística	Controle corrente	Controle velocidade	Controle posição	Controle simultâneo
ABC	03min 36s	08min 50s	10min 31s	28min 20s
ACO	02min 27s	06min 00s	10min 37s	23min 53s
GA	01min 44s	04min 22s	07min 50s	15min 36s
PSO	02min 26s	05min 58s	11min 06s	20min 21s

Fonte: Elaborado pelo autor.

A meta-heurística GA se mostrou a mais rápida em todas as comparações. Apesar de não ter apresentado o melhor resultado entre as meta-heurísticas, ficou muito próxima das demais, sendo apta a sintonizar controladores PID em um tempo menor que as demais. Já a ABC foi a mais lenta para sintonizar todos os controladores.

6 CONCLUSÃO E TRABALHOS FUTUROS

As quatro meta-heurísticas foram submetidas à sintonia do sistema de controladores PID em cascata, de forma sequencial e simultânea. A planta utilizada nas simulações foi a modelagem matemática de um motor CC real. Primeiramente foram encontrados os melhores parâmetros para cada meta-heurística, como tamanho da população, taxa de mutação e taxa de cruzamento no caso do GA, através da simulação com três valores diferentes para cada parâmetro. Como o processo é estocástico, foram realizadas dez simulações para cada valor do parâmetro, escolhendo os que obtiveram a menor média aritmética do custo da função objetivo.

Após a determinação dos parâmetros, foi realizada a comparação entre as meta-heurísticas, realizando entre dez e trinta simulações para os controladores de corrente, velocidade e posição, e os três de forma simultânea. Ao todo foram realizadas 720 simulações e os resultados foram comparados pela média aritmética da função objetivo. A função objetivo utilizada foi a ITAE acrescida de uma punição para sobressinal, a fim de obter uma resposta sem sobressinal, o que pode ser constatado pelas simulações.

Todas as meta-heurísticas se mostraram satisfatórias na sintonia dos controladores, encontrando respostas conforme os parâmetros desejados pela função objetivo. O projetista deve sempre ter em mente o que busca no controle, se é uma resposta rápida, precisa ou sem sobressinal, e saber passar essas informações para a função objetivo, e deve avaliar sempre que uma resposta diferente da desejada for encontrada com um custo baixo da função objetivo, isso significa que ela não está buscando o que se quer, portanto, é preciso alterá-la e recomeçar as simulações.

Para trabalhos futuros, sugere-se realizar novos testes ou aplicar os próprios parâmetros encontrados neste trabalho em um motor CC com as mesmas características para verificar na prática a eficácia dos parâmetros encontrados nas simulações. Outra sugestão é desenvolver uma função multiobjetivo que possa, através da alteração de um ou mais parâmetros, modificar ou determinar o grau de importância de cada parâmetro, tanto no domínio do tempo como no domínio das frequências complexas.

7 REFERÊNCIAS

ÅSTRÖM, K. J.; HÄGGLUND, T. **PID Controllers:** Theory, Design and Tuning. 2ª. ed. North Carolina: Instrument Society of America, 1995.

BOMFIM, I. G. A.; OLIVEIRA, M. O. D.; FREITAS, B. M. **BIOLOGIA DAS ABELHAS**. UNIVERSIDADE ESTADUAL DO CEARÁ – UECE. Fortaleza, p. 56. 2017.

CAMPOS, I. M. S. **Computação Evolucionária Aplicada a Problemas de Otimização Combinatória**. Natal: UFRN, 2006.

CHAPMAN, S. J. **Fundamentos de Máquinas Elétricas**. Tradução de Anatólio Laschuk. 5ª. ed. Porto Alegre: AMGH Editora Ltda., 2013.

COOPER, D. J. **Control Guru - Pratical Process Control**, 2008. Disponivel em: <https://controlguru.com/integral-reset-windup-jacketing-logic-and-the-velocity-pi-form/>. Acesso em: 21 Março 2019.

CORMEN, T. H. et al. **Introduction to Algorithms**. 3ª. ed. Cambridge: Massachusetts Institute of Technology, 2009.

DHIEB, Y. et al. PID Controller Tuning using Ant Colony Optimization for Induction Motor. **Journal of Electrical Systems**, Sfax, v. 15, n. 1, p. 133-141, Novembro 2019.

DORF, R. C.; BISHOP, R. H. **SISTEMAS DE CONTROLE MODERNOS**. Tradução de Bernardo Severo da Silva FIlho. 8ª. ed. Rio de Janeiro: LTC - Livros Técnicos e Científicos S.A., 2001.

DORIGO, M.; GAMBARDELLA, L. M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. **IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION**, v. 1, n. 1, p. 53-66, Abril 1997.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. Ant System: Optimization by a Colony of Cooperating Agents. **IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS**, v. 26, n. 1, p. 29-41, Fevereiro 1996.

EBERHART, R.; KENNEDY, J. A New Optimizer Using Particle Swarm Theory. **Proceedings of the Sixth International Symposium on Micro Machine and human Science**, Nagoya, p. 39-43, 1995.

FERMINO, F. **ESTUDO COMPARATIVO DE MÉTODOS DE SINTONIA DE CONTROLADORES PID**. Universidade de São Paulo - Escola de Engenharia de São Carlos. São Carlos, p. 82. 2014.

GLOVER, F. FUTURE PATHS FOR INTEGER PROGRAMMING AND LINKS TO ARTIFICIAL INTELLIGENCE. **Computers and Operation Research**, Colorado, v. 13, n. 5, p. 533-549, 1986.

GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization, and Machine Learning**. Massachusetts: Addison-Wesley Publishing Company, Inc., 1989.

GOLNARAGHI, F.; KUO, B. C. **Automatic Control Systems**. 9ª. ed. New York: JOHN WILEY & SONS, INC., 2009.

JOHNSON, M. A.; MORADI, M. H. **PID Control New Identification and Design Methods**. 1ª. ed. London: Springer-Verlag London, 2005.

KARABOGA, D. **AN IDEA BASED ON HONEY BEE SWARM FOR NUMERICAL OPTIMIZATION**. Erciyes University, Engineering Faculty. Kayseri. 2005.

KARABOGA, D.; BASTURK, B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. **Journal of Global Optimization**, v. 39, n. 3, p. 459-471, Novembro 2007.

LEONHARD, W. **Control of Electrical Drives**. 2ª. ed. Berlin: Springer-Verlag Berlin Heidelberg, 1996.

LUKE, S. **Essentials of Metaheuristics**. 2ª. ed. Fairfax: Lulu, 2013. Disponivel em: <http://cs.gmu.edu/~sean/book/metaheuristics/>.

MOUAYAD, S. A.; BESTOUN, A. S. A new multiobjective performance criterion used in PID tuning optimization algorithms. **Journal of Advanced Research**, Erbil, v. 7, p. 125-134, 2016.

NISE, N. S. **ENGENHARIA DE SISTEMAS DE CONTROLE**. Tradução de Jackson Paul Matsuura. 6ª. ed. Rio de Janeiro: LTC - Livros Técnicos e Científicos Ltda., 2013.

OGATA, K. **ENGENHARIA DE CONTROLE MODERNO**. Tradução de Heloísa Coimbra de Souza. 5ª. ed. São Paulo: Pearson Education do Brasil, 2011.

PAL, P. et al. OPTIMAL PID CONTROLLER DESIGN FOR SPEED CONTROL OF A SEPARATELY EXCITED DC MOTOR: A FIREFLY BASED OPTIMIZATION APPROACH. **International Journal of Soft Computing, Mathematics and Control**, Agartala, v. 4, n. 4, p. 39-48, Novembro 2015.

SAAD, M. S.; JAMALUDDIN, H.; DARUS, I. Z. M. IMPLEMENTATION OF PID CONTROLLER TUNING USING DIFFERENTIAL EVOLUTION AND GENETIC

ALGORITHMS. **International Journal of Innovative Computing, Information and Control**, Perlis, v. 8, n. 11, p. 7761-7779, Novembro 2012.

SEBORG, D. E.; EDGAR, T. F.; MELLICHAMP, D. A. **Process Dynamics and Control**. 2ª. ed. New Jersey: John Wiley & Sons, Inc., 2004.

SILVA, F. T. D. **SIMULATED ANNEALING APLICADO AO PROBLEMA DE SINTONIA DE PARÂMETROS DE CONTROLADORES PID**. Universidade Federal de Ouro Preto - Escola de Minas. Ouro Preto, p. 60. 2005.

SINGH, K. et al. PID Tuning of Servo Motor using Bat Algorithm. **Procedia Computer Science**, Seri Iskandar, v. 60, p. 1798-1808, 2015.

SOCHA, K.; DORIGO, M. Ant colony optimization for continuous domains. **European Journal of Operational Research**, v. 185, n. 3, p. 11551173, Dezembro 2008.

SOUZA, J. O. D. O. D. **Desenvolvimento e Implementação de Sistemas de Controle para Motores Elétricos**. Universidade do Vale do Rio dos Sinos - UNISINOS. São Leopoldo, p. 128. 2007.

SOWMYA, B.; SUNIL, M. P. Minimization of Floorplanning Area and Wire Length Interconnection Using Particle Swarm Optimization. **International Journal of Emerging Technology and Advanced Engineering**, v. 3, n. 8, p. 321-330, Agosto 2013.

SURI BABU, A. G.; CHIRANJEEVI, B. T. 2.1 IMPLEMENTATION OF FRACTIONAL ORDER PID CONTROLLER FOR AN AVR SYSTEM USING GA AND ACO OPTIMIZATION TECHNIQUES. **International Federation of Automatic Control**, Bhimavaram, v. 49, n. 1, p. 456-461, 2016.

TALBI, E.-G. **METAHEURISTICS FROM DESIGN TO IMPLEMENTATION**. New Jersey: John Wiley & Sons Inc., 2009.

VILLAGRA, M.; BARÁN, B. **Ant Colony Optimization with Adaptive Fitness Function for Satisfiability Testing**. 14th International Workshop, WoLLIC 2007. Rio de Janeiro: Springer-Verlag. 2007. p. 352-361.

WEISE, T. **Global Optimization Algorithms - Theory and Application**. 2ª. ed. Hefei: Auto Publicado, 2009. Disponível em: http://www.it-weise.de/projects/book.pdf.