

**UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS**

Ewerton Venancio Stanzani

Ajuste evolutivo de um controlador de voo de um quadricóptero

São Carlos

2018

Ewerton Venancio Stanzani

Ajuste evolutivo de um controlador de voo de um quadricóptero

Monografia apresentada ao Curso de Engenharia Elétrica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Área de concentração: Engenharia Elétrica

Orientador: Prof. Dr. Eduardo do Valle Simões

São Carlos
2018

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO,
POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS
DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da
EESC/USP com os dados inseridos pelo(a) autor(a).

Sa Stanzani, Ewerton Venancio
Ajuste evolutivo de um controlador de voo de um
quadricoptero. / Ewerton Venancio Stanzani; orientador
Eduardo do Valle Simões. São Carlos, 2018.

Monografia (Graduação em Engenharia Elétrica com
ênfase em Sistemas de Energia e Automação) -- Escola de
Engenharia de São Carlos da Universidade de São Paulo,
2018.

1. Algoritmo Evolutivo. 2. Quadricóptero. 3.
Controlador PID. I. Título.

FOLHA DE APROVAÇÃO

Nome: Ewerton Venancio Stanzani

Título: "Ajuste evolutivo de um controlador de voo de um quadricóptero"

Trabalho de Conclusão de Curso defendido e aprovado
em 25/06/2018,

com NOTA 9,0 (nove, zero), pela Comissão Julgadora:

Prof. Dr. Eduardo do Valle Simões - Orientador - SSC/ICMC/USP

*Mestre Paulo Roberto Ubaldo Guazzelli - Doutorando -
SEL/EESC/USP*

Mestre Jesimar da Silva Arantes - Doutorando - ICMC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

*Dedico esse trabalho a minha família e a minha namorada que sempre estiveram ao meu lado durante
esse caminho.*

AGRADECIMENTOS

A Wilson, Auda e Wellington, pelo apoio em todas as decisões que me fizeram chegar onde estou hoje. A Isis, minha namorada, por ter mudado a forma com que via minha graduação e meus objetivos de vida. A Universidade de São Paulo, por todas as oportunidades que tive até hoje. A República Várzea, pelas amizades e pelo aprendizado durante o tempo em que estive lá. E ao meu orientador Eduardo Simões, por todo o aprendizado, orientação e principalmente por ter feito tudo isso sempre com alegria e dedicação no que estava fazendo.

*“Your assumptions are your windows on the world.
Scrub them off every once in a while, or the light won’t come in.”*

Isaac Asimov

RESUMO

STANZANI, E. Ajuste evolutivo de um controlador de voo de um quadricóptero. 2018. 70p.
Monografia - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2018.

Esse trabalho apresenta o desenvolvimento de um algoritmo evolutivo para encontrar ganhos adequados de um controlador proporcional derivativo de um drone, implementado em um simulador no software MATLAB. Durante o desenvolvimento, foram analisados algoritmos evolutivos com alguns diferentes operadores genéticos, como mutação e predação, e analisado as suas respostas no simulador em que a aeronave esta modelada. A função aptidão, responsável pela eficácia do algoritmo, tinha como objetivo minimizar o overshoot e o tempo de estabilização da resposta do sistema. O algoritmo se mostrou viável como uma possível alternativa aos métodos que pilotos de drone utilizam atualmente para realizar o ajuste dos controladores PD de suas aeronaves.

Palavras-chave: Algoritmo Evolutivo, Quadricóptero, Controlador PD, Controle.

ABSTRACT

STANZANI, E. **Evolutionary adjustment of a flight controller of a quadricopter.** 2018. 70p.
Monografia - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2018.

This work presents the development of an evolutionary algorithm to find suitable gains of a derivative proportional controller of a drone, implemented in a simulator in the MATLAB software. During development, evolutionary algorithms were analyzed with some different genetic operators, such as mutation and predation, and their responses were analyzed in the simulator in which the aircraft is modeled. The fitness function, responsible for the efficiency of the algorithm, had the objective to minimizing the overshoot and settling time of the system response. The algorithm proved feasible as a possible alternative to the methods that drone pilots currently use to perform the tuning of their aircraft PD controllers.

Keywords: Evolutionary Algorithms. Drones. PD Controller. Control.

LISTA DE ILUSTRAÇÕES

Figura 1 – Tipos e Tamanhos Diferentes de Drones	22
Figura 2 – Ciclo de Tendência para Tecnologias Emergentes da Gartner, 2017	22
Figura 3 – Investimento do Setor de Agronegócios, por categoria, em 2016	23
Figura 4 – Fluxograma de um Algoritmo Genético Básico	28
Figura 5 – Gene, Indivíduo e População	29
Figura 6 – <i>Crossover</i> entre dois indivíduos	30
Figura 7 – Mutação ocorrendo em um indivíduo	31
Figura 8 – Predação ocorrendo em um indivíduo de uma geração a próxima	32
Figura 9 – Componentes Básicos de Um Sistema de Controle	33
Figura 10 – Controle em Malha Fechada através de um controlador PID	35
Figura 11 – Ação de um controlador PID. No tempo t , o termo proporcional depende do valor instantâneo do erro. A ação integral é baseada na integral do erro até o tempo t . A ação derivativa extrapola linearmente o erro afim de prever o seu valor no futuro.	35
Figura 12 – Parâmetros de resposta de um controlador a uma função degrau unitário como entrada	36
Figura 13 – Interface de Reposta do Simulador	40
Figura 14 – Sistema Implementado com um Algoritmo Evolutivo Básico	41
Figura 15 – Aptidão da População do Primeiro Sistema	42
Figura 16 – Aptidão da População do Primeiro Sistema em Detalhe	43
Figura 17 – Análise de Dispersão dos Indivíduos do Primeiro Sistema	43
Figura 18 – Sistema Implementado com um Algoritmo Evolutivo Mais Complexo	44
Figura 19 – Aptidão da População do Segundo Sistema	45
Figura 20 – Aptidão da População do Segundo Sistema em Detalhe	45
Figura 21 – Análise de Dispersão dos Indivíduos do Segundo Sistema	46
Figura 22 – Aptidão da População do Último Sistema	47
Figura 23 – Aptidão da População do Último Sistema em Detalhe	47
Figura 24 – Análise de Dispersão dos Indivíduos do Último Sistema	48
Figura 25 – Análise de Dispersão dos Indivíduos do Último Sistema	49
Figura 26 – Análise da Resposta do Melhor Indivíduo do Último Sistema	50

LISTA DE TABELAS

Tabela 1 – Parâmetros de Entrada do Primeiro Algoritmo	41
Tabela 2 – Parâmetros de Entrada do Segundo Algoritmo	44
Tabela 3 – Parâmetros de Entrada do Sistema	46
Tabela 4 – Resultados da Última Simulação	49

SUMÁRIO

1	INTRODUÇÃO	21
1.1	Motivação e Relevância do Trabalho	21
1.2	Revisão Bibliográfica	24
1.3	Objetivo	24
2	FUNDAMENTOS TEÓRICOS	27
2.1	Algoritmos Genéticos	27
2.1.1	População Inicial	28
2.1.2	Função Aptidão	29
2.1.3	Operadores Genéticos	29
2.1.3.1	<i>Crossover</i>	30
2.1.3.2	Mutação	30
2.1.3.3	Predação	31
2.1.3.4	Convergência	32
2.2	Sistemas de Controle do Quadricóptero	32
2.2.1	Controladores em Malha Fechada	33
2.2.2	Controladores PID	33
2.2.3	Ajuste do Controlador PID	35
2.2.4	Avaliação de um Controlador	36
3	DESENVOLVIMENTO	37
3.1	Algoritmo Genético	37
3.1.1	População Inicial	37
3.1.2	Função Aptidão	37
3.1.3	Seleção do mais apto	38
3.1.4	<i>Crossover</i>	38
3.1.5	Término	39
3.2	Simulador	39
3.2.1	Funcionamento	40
3.2.2	Modificações	41
3.3	Implementação	41
3.3.1	Sistema Inicial	41
3.3.2	Segundo Sistema	44
3.3.3	Sistema Final	46
3.3.3.1	Análise dos Resultados	49
4	CONSIDERAÇÕES FINAIS	51
4.1	Conclusão	51
4.2	Estudos Futuros	51

REFERÊNCIAS	53
APÊNDICES	55
APÊNDICE A – CÓDIGO PRINCIPAL	57
ANEXOS	61
ANEXO A – SIMULADOR	63
ANEXO B – FUNÇÃO PARÂMETROS E FUNÇÃO DO CONTROLADOR	65
ANEXO C – CONTROLADOR	67

1 INTRODUÇÃO

1.1 Motivação e Relevância do Trabalho

Tecnologia é o conjunto de técnicas e ferramentas que melhoram a indústria, as relações sociais, o comércio, ou seja, que melhoram a qualidade de vida e facilitam a vida humana. Cada vez mais vem acontecendo uma intensificação da evolução tecnológica, tanto em velocidade quanto em grau de sofisticação. Apesar de tecnologias trazerem diferentes benefícios, elas não são amplamente aceitas e difundidas logo no início. Pelo contrário, até uma tecnologia se estabelecer no mercado ela passa por diferentes ciclos de vida. Os quatro estágios do ciclo de vida de uma tecnologia, segundo Mamaghani e Zong (2015), são os seguintes:

- Estágio de Inovação;
- Estágio de Distribuição;
- Estágio de Difusão;
- Estágio de Substituição.

Estágio de Inovação: Esta etapa representa o nascimento de um novo produto, material de processo resultante de Pesquisa e Desenvolvimento. Nos laboratórios de P&D, novas ideias são geradas dependendo de necessidades e de conhecimento. Dependendo da alocação de recursos e também do elemento de mudança, o tempo gasto no estágio de inovação, bem como nos estágios subsequentes, varia amplamente.

Estágio de Distribuição: Esta etapa representa a demonstração e comercialização de uma nova tecnologia, como produto, material ou processo com potencial para utilização imediata. Muitas inovações ficam apenas na etapa de P&D. Apenas uma porcentagem muito pequena delas é comercializada. A comercialização de resultados de pesquisa depende de fatores técnicos e econômicos.

Estágio de Difusão: representa a penetração no mercado de uma nova tecnologia através da aceitação da inovação, pelos potenciais utilizadores da tecnologia. Mas os fatores de oferta e da demanda influenciam conjuntamente a taxa de difusão da tecnologia.

Estágio de Substituição: Este último estágio representa o declínio no uso e eventual extensão de uma tecnologia, devido à substituição por outra tecnologia. Muitos fatores técnicos e não técnicos influenciam a taxa de substituição. O tempo gasto nesse estágio depende da dinâmica do mercado.

A tecnologia estudada no presente projeto é a de drones ou VANT - Veículo Aéreo Não Tripulado. Essa tecnologia tem ganhado cada vez mais popularidade no mundo todo devido a suas diversas aplicações em lazer, segurança, vigilância, militar e até mesmo nos esportes.

Quando o tema drones é abordado, é necessário salientar que existem diferentes modelos para diferentes finalidades.

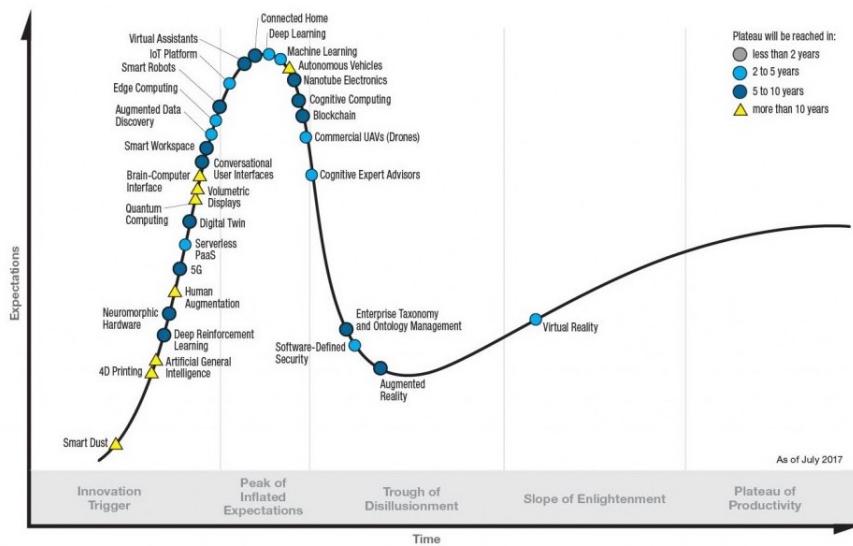
Figura 1: Tipos e Tamanhos Diferentes de Drones



Fonte: 3dinsider.com (2017)

Segundo GARTNER (2017), empresa americana líder em pesquisa de mercado e tendências em tecnologia, drones estão em uma fase avançada de maturidade e em até 5 anos alcançarão um patamar de alta produtividade movimentando muitas indústrias e o mercado. Eles mostraram que, em 2017, esse mercado não apenas movimentou aproximadamente USD 6 bilhões, como também está em franco crescimento.

Figura 2: Ciclo de Tendência para Tecnologias Emergentes da Gartner, 2017



Fonte: (GARTNER, 2017)

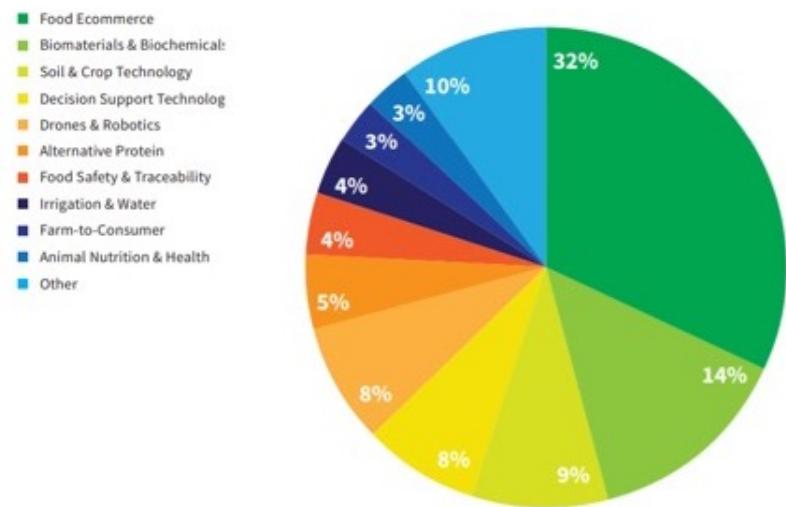
Similar ao mercado global, o mercado brasileiro movimentou uma grande quantidade de dinheiro. Segundo Emerson Granemann, em 2017, o Brasil movimentou R\$ 300 milhões, sendo 50% a mais que o ano anterior. Segundo ele, o agronegócio é o grande representante desse investimento no Brasil, sendo responsável por 40% dessa receita.

Drones são utilizados no setor de agronegócios, pois podem fazer o monitoramento aéreo, também conhecido como sensoriamento remoto, das condições das plantações. Podem detectar problemas de irrigação, variação do solo, desmatamento, mudanças na pecuária, erosão, infestações por pragas e fungos e outras informações que podem não ser facilmente aparentes no nível do solo. As câmeras presentes nos drones podem capturar imagens multiespectrais, capturando imagens infravermelhas e do espectro visual. Estas imagens podem ser sequenciadas para mostrar mudanças nas plantações.

O tamanho do mercado global de alimentos e agronegócios, de acordo com o relatório da McKinsey & Co., “Pursuing the Global Opportunity in Food and Agribusiness”, é de USD 5 trilhões (GOEDDE; HORII; SANGHVI,). Desde 2004, investimentos corporativos e de risco focados em alimentos e agronegócios aumentaram significativamente, triplicando para mais de USD 100 bilhões em 2013. Investimentos corporativos e de capital de risco em empresas de tecnologia agrícola em todo o mundo totalizaram USD 25 bilhões em 2015 (SINGH, 2016). Em 2017, 13 fundos de investimentos foram dedicados à tecnologia agrícola (BURWOOD-TAYLOR, 2016).

Dado o tamanho de mercado e seu crescimento, é importante analisar onde os investimentos deste setor são feitos. A figura 3, mostra os principais investimentos do setor de alimentos e agronegócios por categoria em 2016.

Figura 3: Investimento do Setor de Agronegócios, por categoria, em 2016



Fonte: AgTech Investing Report, 2017, citado por Srivastava (2017)

Analizando este contexto, é possível observar que drones obtiveram investimentos, desenvolvimento, e, consequentemente diminuição no custo de aquisição (SRIVASTAVA, 2017). Devido a isso, assim como a diversos outros fatores, teve-se uma popularização de tal tecnologia. Hoje em dia, é possível comprar um quadricóptero por um preço acessível, o que levanta diversos questionamentos sobre o assunto. Pois, é necessário a preocupação com a segurança no voo desde já para facilitar a regulação e legalização dessas aeronaves.

Devido a tal popularidade e a possibilidade de customização de alguns desses modelos, existem muitos que possuem sistemas de controles de baixa qualidade, ou mal regulados, que podem causar diversos tipos de acidente e até mesmo matar pessoas, o que os torna uma grande ameaça a segurança pública. Existem diversos relatos de drones invadindo espaços aéreos de aeronaves de porte maior, como por exemplo aeroportos, e se envolvendo em acidentes com aviões.

Um sistema eficaz de controle é fundamental para o bom funcionamento de tal aeronave e para a segurança de todos envolvidos com isso. Uma aeronave que possui uma resposta mais rápida e controlada é fundamental para que se tenha mais segurança e. Outros exemplos podem ser citados nos quais é importante o investimento em tecnologias que tornam o voo mais seguro.

1.2 Revisão Bibliográfica

Atualmente, drones vem sendo aplicados em diversos setores, como o da agricultura por exemplo, para monitoramento e mapeamento aéreo, despejo de químicos, etc. Tal aplicação pode envolver um investimento alto, em que uma falha pode resultar em um custo muito alto. Dado isso, é fundamental que tenha-se um sistema de controle que garanta segurança no voo desse tipo de aeronave. Para isso, deve-se estudar diversas possibilidades e entender qual tipo de conceito pode ser aplicado para melhorar a segurança e assim, diminuir os danos. Drones, em geral, são controlados por diversos controladores PID (Proporcional, Integrador e Derivativo) implementados digitalmente com objetivos diversos, mas todos afim de melhorar o comportamento da aeronave em voo.

Dado isso, a proposta desse projeto é a de desenvolver um algoritmo evolutivo capaz de encontrar parâmetros adequados para controladores PD (Proporcional e Derivativo) de um quadricóptero para otimizar o comportamento em voo de uma aeronave em um simulador, pois esse é um dos sistemas de controle mais utilizados nesse tipo de aeronave.

Em VANT domésticas, os usuários configuram manualmente esses parâmetros através de tentativa e erro, o qual exige um voo teste para testar os parâmetros pré-configurados. Esse processo, assim como exemplificado anteriormente, pode provocar a queda da aeronave devido a resposta inadequada do sistema. Quando uma falha de controle acontece em aeromodelos e drones em voo, essas aeronaves podem ficar a deriva e/ou desempenhar movimentos arriscados, tornando-se perigosos a qualquer pessoa ao redor.

Além do desenvolvimento do algoritmo evolutivo capaz de otimizar os parâmetros do controlador para um melhor controle de voo do drone, é necessário que todos esses testes sejam executados dentro de um simulador, afim de obter um controle o mais próximo possível de uma situação real, para assim testar em um modelo real.

Com a aplicação dos algoritmos evolutivos, pode-se otimizar os parâmetros de forma automática e atingir valores melhores do que os parâmetros obtidos manualmente, resultando em um voo mais estável e seguro.

1.3 Objetivo

O objetivo desse projeto é o de desenvolver um algoritmo evolutivo capaz de encontrar parâmetros adequados para controladores PD (Proporcional e Derivativo) de um quadricóptero para otimizar o comportamento dessa aeronave. Tal algoritmo será feito e testado dentro de um simulador feito no

programa MATLAB na própria linguagem nativa do software. Primeiramente, o algoritmo evolutivo será validado como funcional e a partir disso testado algumas variações no próprio algoritmo para entender como isso pode auxiliar na busca por soluções melhores. Caso o algoritmo mostre uma solução viável para o problema, sendo isso julgado pela curva de resposta, a alternativa sera considerada como viável em solucionar o problema de ajuste do controlador PD.

2 FUNDAMENTOS TEÓRICOS

A computação evolutiva é um ramo de pesquisa em que se encontram uma família de algoritmos inspirados na evolução biológica, mais especificamente na Seleção Natural apresentada por Darwin em A Origem das Espécies. Ela compreende a busca por uma solução ótima através de um conjunto de técnicas de otimização que avaliam indivíduos que competem pela sobrevivência e passam suas características para uma nova geração. Como visto em Pozo et al. (s.d.), as técnicas de computação evolutiva incluem programação evolucionária, estratégias evolucionárias, algoritmos genéticos e programação genética (BANZHAF et al., 1998). Cada vez mais esse tipo de método vem sendo utilizado para obter modelos de inteligência computacional (BARRETO, 1997).

Nesse trabalho, será visto como algoritmos evolutivos podem auxiliar na busca de uma solução ótima para um problema de controle.

2.1 Algoritmos Genéticos

Apresentados por John Henry Holland, professor Ph.D. da Universidade de Michigan (HOLLAND, 1975), algoritmos genéticos são uma área de pesquisa dentro da Computação Evolutiva. Holland tinha como objetivo estudar os fenômenos relacionados a evolução e adaptação das espécies e de encontrar uma maneira de aplicar esses conceitos a computadores.

A principal ideia do algoritmo genético está em tratar possíveis soluções de um determinado problema como indivis de uma população, que a cada geração irá evoluir e caso bem sucedida, convergir para uma solução adequada do problema. Os algoritmos genéticos possuem aplicação em muitas áreas científicas, principalmente em otimização de soluções, *machine-learning*, modelagem de diversos problemas, dentre outros.

Algoritmos genéticos são muito bons em determinar máximos e mínimos de uma função. Essa vantagem se dá devido ao fato de que a população inicial criada através do algoritmo é aleatória e através de técnicas de seleção ela tende a convergir para um ponto ótimo da função.

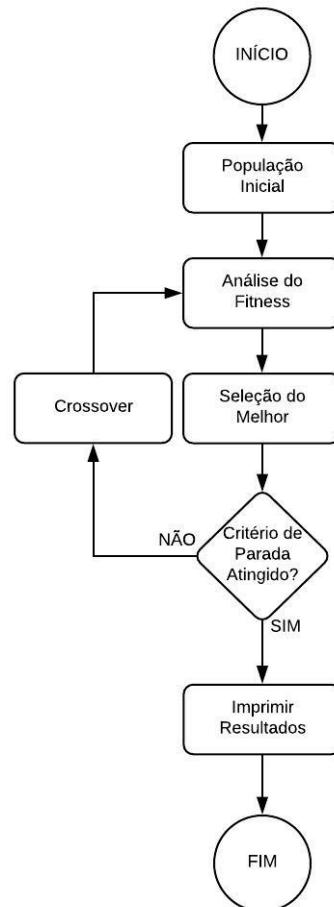
Segundo Pozo et al. (s.d.), a execução de um algoritmo genético pode ser resumida aos seguintes passos:

- Inicialmente a população inicial é criada, normalmente de forma aleatória.
- A população é avaliada indivíduo a indivíduo, segundo algum critério que avalia a performance do indivíduo, chamado de função aptidão.
- Em seguida, entra um operador que seleciona os indivíduos de melhor valor através de um processo de seleção baseado na performance avaliada no passo anterior. Esses serão os indivíduos base para a próxima seleção.
- A partir do melhor indivíduo, a nova geração é obtida através da mistura entre indivíduos selecionados. Isso ocorre através dos operadores genéticos de *crossover* e mutação.
- O algoritmo repete-se até que uma solução aceitável seja encontrada, ou um limite pré-

estabelecido tenha sido atingido. Por exemplo, um número máximo de gerações.

O processo pode ser resumido de acordo com o fluxograma abaixo:

Figura 4: Fluxograma de um Algoritmo Genético Básico



Fonte: Elaborado pelo autor

2.1.1 População Inicial

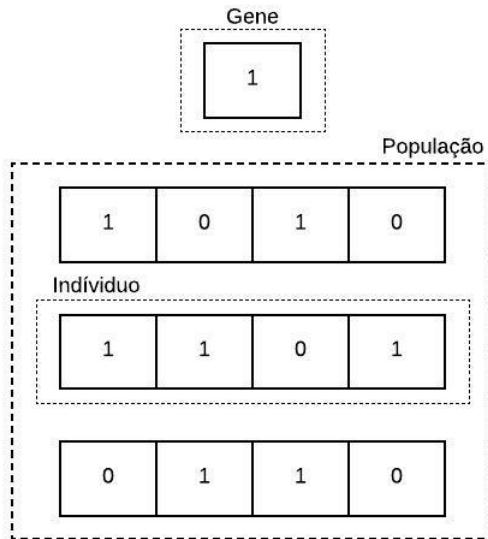
A população inicial é a representação das possíveis soluções do problema. Ela é definida aleatoriamente dentro dos limites estabelecidos para que o espaço de busca por soluções seja grande o suficiente para englobar uma solução ideal para o problema.

O tamanho da população varia de acordo com o problema, pois pode afetar o desempenho global e a eficiência dos algoritmos evolutivos. Populações pequenas tendem a ser pouco diversas e como consequência tendem a varrer um espaço menor de possíveis soluções. Porém, populações muito grandes podem afetar consideravelmente o desempenho computacional na execução do algoritmo evolutivo e dessa forma ter uma redução muito grande na eficiência em resolver o problema.

Dentro dessa população, os indivíduos devem ser representados de uma maneira boa o suficiente para que o algoritmo trabalhe efetivamente com eles. Isso varia de acordo com cada problema. Usualmente

os indivíduos são representados como uma sequência dos seus atributos e cada atributo é representado de uma forma binária. Apesar de ser uma das formas mais utilizadas, podem ser representados de outras formas, como através de números reais. Os exemplos a seguir representam de forma binária para fins didáticos. Porém, nesse trabalho é usada a representação dos indivíduos como números reais.

Figura 5: Gene, Indivíduo e População



Fonte: Elaborado pelo autor

2.1.2 Função Aptidão

A cada geração, um ou mais indivíduos são selecionados através de um processo de seleção que visa mostrar quem é o melhor, ou o mais apto, daquela população. Em geral, os métodos de seleção avaliam os indivíduos um a um para descobrirem quem é o melhor. Porém, alguns métodos avaliam apenas uma amostra da população, dado o fato que o processo como um todo pode demandar bastante tempo e performance de um sistema.

Uma função aptidão bem definida é fundamental para o bom funcionamento do algoritmo evolutivo, afinal ela mede quão melhor um indivíduo é naquele ambiente e nos permite comparar dois ou mais indivíduos para descobrir quem é o melhor. Em casos em que não é possível definir bem qual função utilizar para a solução do problema, outros métodos podem ser utilizados. Pode-se citar também outros métodos de seleção comumente usados em algoritmos genéticos como o Método da Roleta, Seleção por torneio, dentre outros.

2.1.3 Operadores Genéticos

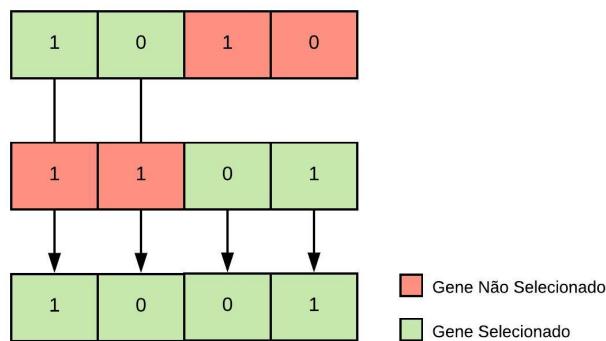
A partir da avaliação dos indivíduos e seleção dos mais aptos, os operadores genéticos definem e criam a geração seguinte.

2.1.3.1 Crossover

Para cada novo indivíduo a ser criado, um par de indivíduos anteriores é selecionado para que suas características sejam transmitidas a ele. Usualmente, o conceito do elitismo define que o indivíduo mais apto sempre sobreviva de uma geração para a outra. Para garantir que o gene do melhor indivíduo sempre passe para a próxima geração, pode-se utilizar o melhor indivíduo como um dos indivíduos no crossover. Dessa forma, os genes desse indivíduo mais apto são mesclados com o restante dos indivíduos para que assim surja uma nova geração.

Essa operação tem como objetivo a criação de um novo indivíduo que combine as características dos indivíduos pais e potencialmente tenha uma aptidão melhor do que eles.

Figura 6: *Crossover* entre dois indivíduos



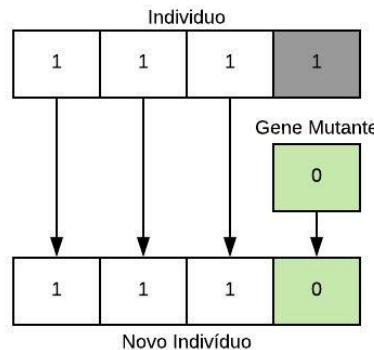
Fonte: Elaborado pelo autor

2.1.3.2 Mutação

Esse operador genético tem como objetivo modificar aleatoriamente alguma característica do indivíduo da nova geração. A mutação não tem uma periodicidade definida, ou seja, não ocorre necessariamente em todos os indivíduos.

A mutação é importante, pois ajuda a potencialmente trazer novas características que não estavam presentes na população anteriormente e assim aumentar o escopo de busca do algoritmo. Dessa forma, mesmo que uma população converja e esteja concentrada em torno de um valor de solução, a mutação ainda permite que buscas continuem sendo feitas afim de garantir que a solução encontrada não seja um mínimo ou máximo local da solução e continue buscando por mínimos e máximos globais.

Figura 7: Mutação ocorrendo em um indivíduo



Fonte: Elaborado pelo autor

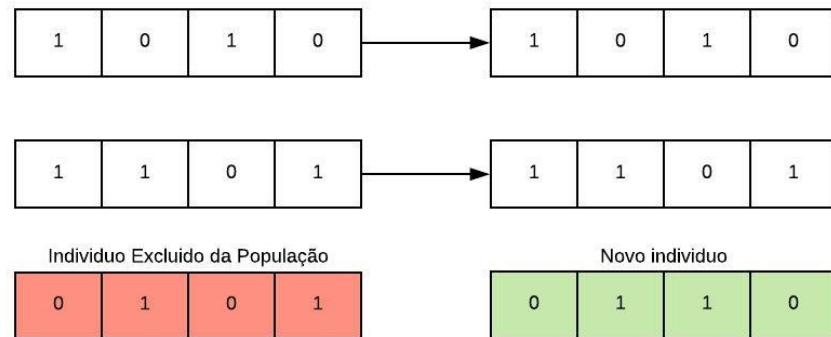
2.1.3.3 Predação

Esse operador genético tem como objetivo remover alguns indivíduos da população e inserir novos indivíduos e dessa forma potencializar ainda mais a busca por novas soluções.

Esse operador foi introduzido por Simões e Barone (2002) e é pouco utilizado em algoritmos genéticos em geral. Porém, traz um ganho em sempre trazer uma aleatoriedade grande no sistema. Deve-se sempre tomar o cuidado de que ao remover alguns indivíduos, não será descartado o mais apto, pois até o momento ele é a melhor solução apresentada ao problema definido.

Analogamente a mutação, esse operador evita que o algoritmo converja para um mínimo ou máximo local de uma função, pois sempre quando o mesmo ocorrer, uma nova potencial solução passa a ser avaliada e comparada com as já existentes. A convergência para um mínimo ou máximo local é um problema característico dos algoritmos genéticos, uma vez que eles tendem a convergir para valores de solução similares ao do melhor indivíduo, o que pode não ser uma solução ótima ainda. Assim, como dito em Simões e Barone (2002), com a estratégia de predação, a população sempre terá um novo material genético sendo introduzido, aumentando as chances de conversão para um mínimo ou máximo global.

Figura 8: Predação ocorrendo em um indivíduo de uma geração a próxima



Fonte: Elaborado pelo autor

2.1.3.4 Convergência

Como visto em Busetti (2001), a aptidão do mais apto e a média da aptidão de todos os indivíduos a cada geração tendem a convergir para um mínimo ou máximo global de uma função. Diz que um algoritmo evolutivo convergiu quando a maioria dos seus indivis se encontram perto da solução do indivíduo mais apto. Ao longo do tempo a população tende a convergir enquanto a aptidão médio se aproxima da aptidão do mais apto.

A grande vantagem do algoritmo evolutivo não está em garantir que será encontrada uma solução ótima do problema, mas sim uma solução aceitável que seja o suficiente para resolver aquele problema. A partir disso, diversas abordagens podem ser feitas para adicionar um algoritmo evolutivo na solução de um problema comum a fim de se buscar melhorias.

Ao longo desse trabalho, será mostrado como cada um desses conceitos foi aplicado e desenvolvido para a solução do problema apresentado.

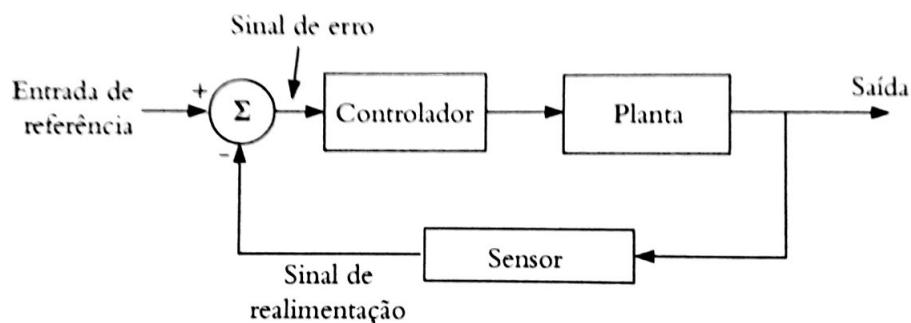
2.2 Sistemas de Controle do Quadricóptero

Como definido em Åström e Murray (2004), controle é o uso de algoritmos e a técnica de realimentação em sistemas de engenharia. Tais algoritmos não se limitam apenas ao mundo computacional. Antigamente, controladores eram feitos apenas com elementos eletrônicos sem a necessidade de um sistema embarcado. Foi apenas em 1959 que um computador foi utilizado para controle de uma planta industrial. Isso ocorreu em uma refinaria da empresa Texaco no Texas (BUSINESS WEEK, 1959 apud VILLAÇA; SILVEIRA, 2013). Assim, através das aplicações dessas técnicas espera-se que o sistema a ser controlado tenha um comportamento esperado. As técnicas de controle são aplicadas após a modelagem do sistema em análise. Essa modelagem, em geral, resume o sistema em um bloco de entrada e saída que descreve o comportamento do sistema (saída) a um estímulo (entrada).

2.2.1 Controladores em Malha Fechada

Um sistema de controle é basicamente composto de um controlador, algo a ser controlado (chamado de planta) e um sensor. A planta pode representar diversos sistemas, como por exemplo um ar condicionado, um motor, um processo químico, um robô dentre outros e o seu comportamento é um resultado das suas características físicas e de influências do ambiente. O sensor é responsável por monitorar o estado da planta e enviar essa informação para o controlador. Dessa forma, ele compara com a referência definida e decide se deve atuar para corrigir o valor de leitura ou não. O controlador tem como principal objetivo minimizar o erro do sistema, ou seja, garantir que a saída do sistema se comporte de acordo com a referência de entrada.

Figura 9: Componentes Básicos de Um Sistema de Controle



Fonte: (NIKU, 2014)

Sistemas em que o sinal de saída não tem efeito sobre o sinal de entrada são chamados de sistemas de malha aberta. Porém, em muitos casos deseja-se que o sinal de saída de um sistema influencie em sua entrada. Assim, utiliza-se os chamados sistemas de malha fechada, em que a saída tem influência no sinal de entrada e o sistema é realimentado, como na Figura 9.

Existem inúmeras vantagens na utilização de controladores em malha fechada. Por exemplo: controle sobre perturbações de um sistema, melhora de estabilidade, controle sobre sensibilidade do sistema, entre outros. Existem diversas abordagens dentro desse tópico, mas esse trabalho entrará no detalhe de um tipo específico de controlador, o controlador PID, que foi empiricamente verificado por ser uma estrutura extremamente útil (ÅSTRÖM; HAGGLUND, 1994).

2.2.2 Controladores PID

Os controladores Proporcional-Integral-Derivativo (comumente chamados de controladores PID) são controladores desenhados para um melhor controle do sistema atuando no erro de três formas, como diz o seu nome. São controladores altamente utilizados na indústria em processos de automação, assim como em diversas outras aplicações como em aquecedores, drones, dentre outros. Segundo Desborough e Miller (2002), uma pesquisa nas indústrias de refinaria, química e papel e celulose, em que envolveu mais de onze mil controladores 97% deles utilizavam controladores PID.

O Controlador Proporcional tem a intenção de que a sua atuação no sistema seja proporcional

ao erro que existe nele, onde k_p é o ganho proporcional. Dessa forma, diferente de um controlador Liga-Desliga, ele atua proporcionalmente, evitando muitas oscilações. Porém, esse efeito apenas tem uma grande desvantagem, ele faz com que o sistema não fique em linha com sua referência, o que é chamado de erro de regime permanente. Por exemplo, caso o sistema necessite de algum sinal do controlador proporcional para funcionar. Para isso o erro precisa ser diferente de zero.

Quando o controlador proporcional é adicionado, o sistema fica dependente desse ganho, dessa forma é possível projetar um valor para que o sistema se comporte de uma maneira particular (NIKU, 2014). Esse conceito pode ser aplicado a todos controladores que serão apresentados nesse trabalho. Ele é fundamental, pois através disso pode-se conseguir um sistema se comporte de uma forma mais adequada.

$$u(t) = k_p \cdot e(t) \quad (2.1)$$

A fim de evitar o erro de regime permanente, existe o Controlador Integral. Ele faz com que a ação de controle seja proporcional a integral do erro. Dessa forma, erros de regime permanente passam a não ter efeito no sistema, onde k_i é o ganho integral.

Entretanto, o sistema pode não estar sujeito apenas a erros de regime constante. Pode, por exemplo, estar sujeito a uma oscilação. Mesmo assim, controladores PI's são amplamente usados pela capacidade de resolver diversos problemas que tem a característica de possuir erros de regime permanente.

$$u(t) = k_i \cdot \int_0^t e(\tau) d\tau \quad (2.2)$$

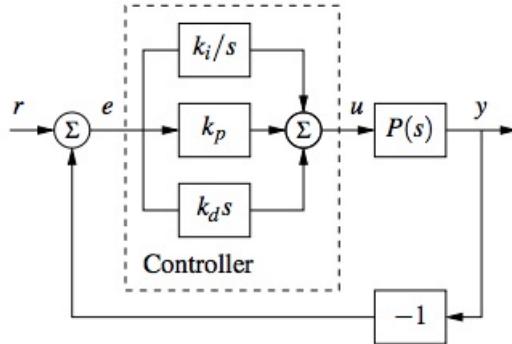
O controlador derivativo, traz mais uma possível melhora para o sistema. Ele adiciona ao controlador a capacidade de antecipar o erro, usando uma predição do valor que o erro terá através de uma extrapolação linear.

Assim, combinando os três efeitos obtém-se o controlador PID que pode ser representado da forma a seguir, de acordo com Åström e Murray (2004).

$$u(t) = k_p \cdot e(t) + k_i \cdot \int_0^t e(\tau) d\tau + k_d \cdot \frac{\partial e(t)}{\partial t} \quad (2.3)$$

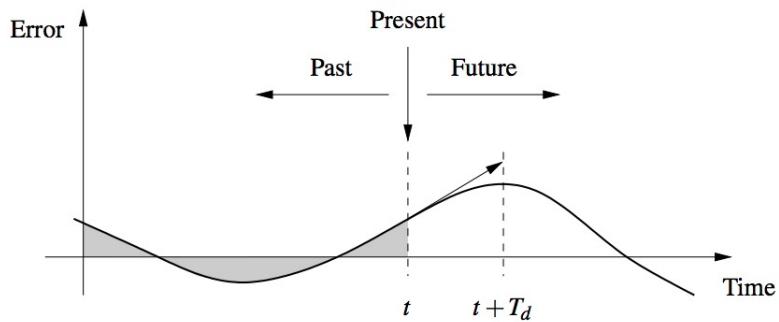
A figura 10 representa um controlador PID atuando sobre o erro (e), comparando a referência (r) com a saída (y) do sistema. Já a figura 11 mostra como um controlador PID atua nos seus diferentes componentes, olhando ao presente, passado e tentando prever o futuro.

Figura 10: Controle em Malha Fechada através de um controlador PID



Fonte: (ÅSTRÖM; MURRAY, 2004)

Figura 11: Ação de um controlador PID. No tempo t , o termo proporcional depende do valor instantâneo do erro. A ação integral é baseada na integral do erro até o tempo t . A ação derivativa extrapola linearmente o erro afim de prever o seu valor no futuro.



Fonte: (ÅSTRÖM; MURRAY, 2004)

2.2.3 Ajuste do Controlador PID

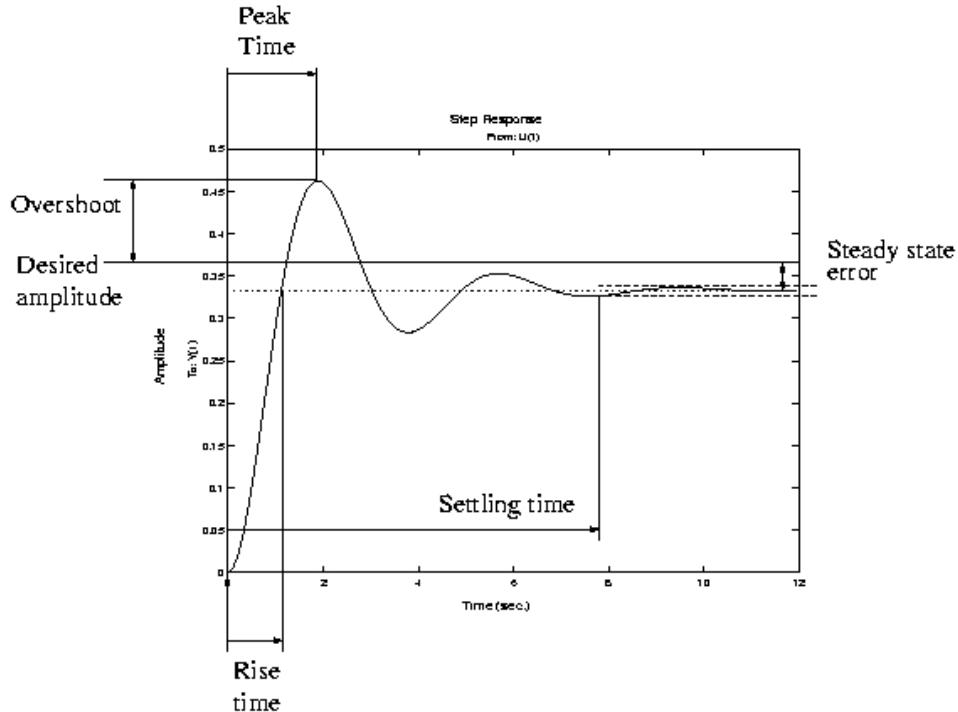
Ao escolher o controlador PID como o tipo de controlador para o sistema, deve-se ter a preocupação em escolher os valores de ganho de cada componente (k_p , k_i e k_d) a fim de obter uma resposta satisfatória para o sistema, ou seja, um comportamento adequado. Esse desafio pode ser resolvido de diversas formas, pode-se modelar o sistema e encontrar os lugares nas raízes ou utilizar de outros métodos que existem na literatura, como por exemplo o método de Ziegler and Nichols (ZIEGLER; NICHOLS., 1942).

Durante este trabalho, será levantada uma outra forma para encontrar parâmetros ideais para o controle do nosso sistema e será feira a avaliação da resposta encontrada, para verificar se ela foi satisfatória.

2.2.4 Avaliação de um Controlador

Existem diversas formas de se avaliar a resposta um sistema e sua performance. Dificilmente sistemas respondem instantaneamente a uma perturbação na entrada, dessa forma, é necessário medir e avaliar algumas das características da resposta, quando comparada a referência de entrada.

Figura 12: Parâmetros de resposta de um controlador a uma função degrau unitário como entrada



Fonte: (GOODWINE, 2003)

Neste trabalho, o foco maior será em analisar o Sobresinal (*Overshoot*) e o Tempo de Acomodação(*Settling Time*), mostrados na figura 12. O sobresinal é o valor de pico do sinal medido a partir do valor de referência. O tempo de acomodação é o tempo necessário para que a partir dali a curva resposta se mantenha dentro de uma faixa ao redor do valor final, usualmente em torno de 1% a 5%.

3 DESENVOLVIMENTO

Este capítulo, descreve o desenvolvimento do trabalho e mostra como um algoritmo evolutivo pode ser aplicado em resolver um problema de controle. Descreve também as mudanças e adaptações feitas no algoritmo e no simulador para que se adaptassem a esse problema e ao final encontrassem ganhos do controlador PD implementado através do simulador. O simulador não foi implementado com um controlador integrador pois no modelo não precisamos nos preocupar com erros de regimes permanentes pois o modelo da aeronave é conhecido.

3.1 Algoritmo Genético

Existem duas formas de utilização dos conceitos de algoritmos genético em um sistema de controle.

Off-line: utiliza-se um algoritmo evolutivo para projetar um controlador que é depois utilizado para controlar um sistema real através das soluções encontradas.

On-line: utiliza-se o algoritmo evolutivo de forma ativa para um controlador sujeito a mudanças drásticas no ambiente.

Uma vantagem de um controlador evolutivo é que ele pode ser utilizado em ambientes dinâmicos, necessitando de um poder computacional compatível com a atividade a ser executada. Existem diversas outras vantagens no uso de algoritmos genéticos que podem ser diretamente aplicáveis a esse problema. Algoritmos genéticos são muito bons em determinar máximos e mínimos globais de uma função. Essa vantagem se dá devido ao fato de que a população inicial criada através do algoritmo é aleatória e através de técnicas de seleção ela tende a convergir para um ponto ótimo da função.

Como dito anteriormente, um algoritmo evolutivo simples consiste de quatro etapas: a definição da população inicial; a função de aptidão; a seleção do indivíduo mais apto; e o *Crossover*. Neste capítulo, será discutido como cada uma delas se aplica ao nosso problema.

3.1.1 População Inicial

A população inicial é a representação das variáveis a serem controladas do sistema em análise, no caso os ganhos do controlador PD formarão um indivíduo dessa população. Todos os indivíduos dessa população são definidos aleatoriamente, respeitando-se um limite de valores definidos no início do algoritmo.

O tamanho da população também será definido no início do algoritmo. O valor escolhido para a população foi o de 10 indivíduos e os indivíduos são representados como números reais e não de forma binária.

3.1.2 Função Aptidão

A partir de uma população criada, é necessário definir uma função que julgue quem desses indivíduos tem a melhor performance no ambiente ou problema a ser estudado. Nesse caso, pode-se

determinar quem dos indivíduos responde mais rápido ao comando dado, ou o indivíduo que possui menos oscilações na resposta, ou até mesmo uma combinação destes dois fatores. A função aptidão deve ser escolhida com cuidado, pois ao utilizar uma função errada estaria selecionando indivíduos errados como os mais aptos.

Nesse trabalho, será utilizado uma função que considera o sobresinal e o tempo de resposta do sistema, conforme a equação 3.1. Como o nosso objetivo é o de minimizar o sobresinal e o tempo de acomodação, nossa função de aptidão será uma função de minimização. Dessa forma, o melhor indivíduo será aquele que tiver uma função aptidão com o menor valor. Ambas as funções são dependentes dos valores dos indivíduos (ganho k_p e k_d).

$$Sobresinal(i) = \begin{cases} 0 & , Sobresinal < 0 \\ Sobresinal + 1 & , Sobresinal \geq 0 \end{cases} \quad (3.1)$$

$$Aptidao(i) = TempoDeAcomodacao(i) + Sobresinal(i) \quad (3.2)$$

O índice i representa um indivíduo da população e a correção na função sobresinal se dá para valores de sobresinal menores do que zero, para que ele passe a considerar o sobresinal como zero e somando um aos valores para que tenha um valor mínimo de 1.

Quando o programa não converge no tempo máximo de 5 segundos, o programa não pode avaliar sobresinal e Tempo de Resposta da curva e retorna o valor NaN (*Not a Number*) como aptidão. Dessa forma, um valor grande o suficiente é atribuído a ele para que receba uma nota ruim e assim dificilmente seja escolhido como o melhor indivíduo.

3.1.3 Seleção do mais apto

A partir da população estabelecida e da função aptidão, é possível julgar o indivíduo de melhor performance. Este será o indivíduo mais apto e o que passará suas características para a próxima geração. A seleção ocorre comparando a aptidão de todos os indivíduos e encontrando o melhor. Quando a aptidão melhor que o anterior é encontrado, o algoritmo marca o indivíduo i como sendo melhor que os anteriores.

```

1   for i=1:tamPop
2       if fitness(i)≤fitness(melhor)
3           melhor = i;
4       end
5   end

```

Após a seleção do mais apto, ocorre uma troca de posição onde o indivíduo de melhor fitness é reposicionado na posição 1 dos vetores de indivíduos.

3.1.4 Crossover

Uma vez selecionado o indivíduo mais apto, os seus parâmetros devem ser passados para os demais indivíduos. Usualmente, o *Crossover* ocorre pegando o atributo de cada um dos indivíduos, juntamente ao do mais apto e calculando a média de valores entre eles, porém podem-se usar outras formas

quando numeros reais são utilizados, como por exemplo a média ponderada e a razão (GUAZZELLI, 2017 apud DEB; KALYANMOY, 2001). É importante que os parâmetros do indivíduo mais apto, chamado agora de pai, não se altere de valor. Dessa forma, após o *crossover* de todos os indivíduos com o mais apto, obtem-se uma nova geração.

$$Atributo_{novo}(i) = \frac{Atributo_{geracaoAnterior}(i) + Atributo_{melhorIndividuo}}{2}, i \geq 2 \quad (3.3)$$

A nova geração será avaliada nos mesmos critérios que a geração anterior. E assim o programa continuará até tentar convergir para um indivíduo adequado a solução do problema. O *crossover* não altera em nada os atributos do indivíduo (i) de índice 1, pois ele representa o melhor indivíduo da geração atual.

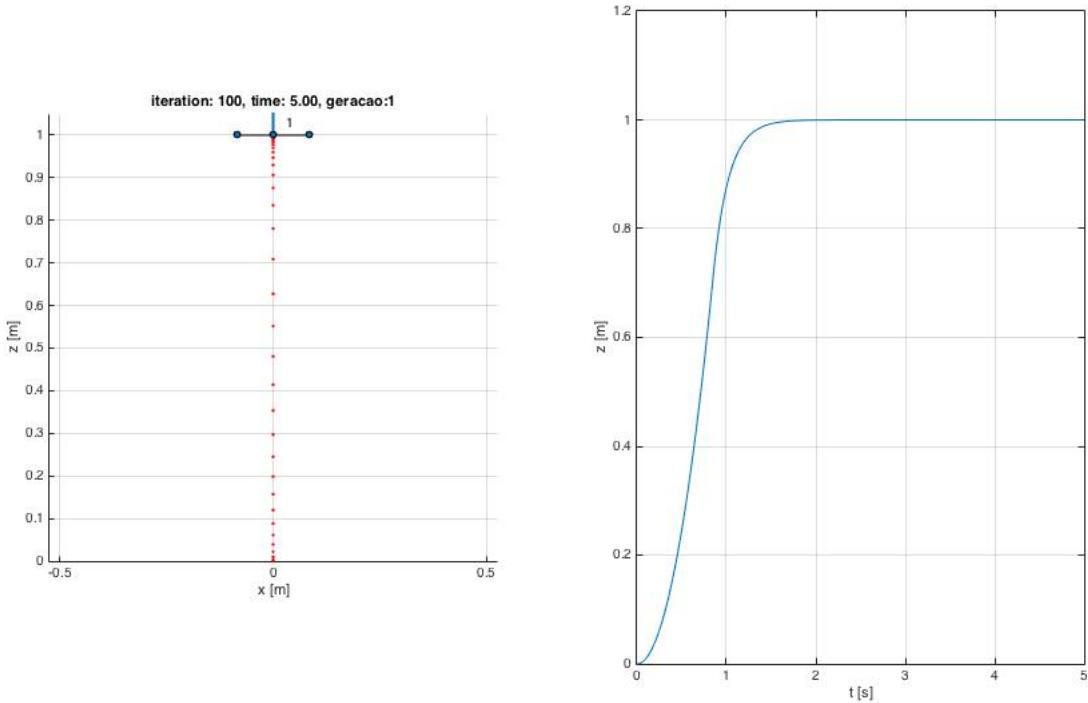
3.1.5 Término

Uma vez com uma nova geração, o processo de seleção do mais apto e *crossover* é repetido até um critério de parada definido. Este critério pode ser definido em tempo por exemplo ou até mesmo número de gerações. Dessa forma ao atingir o critério de parada, obteve-se os parâmetros que melhor foram avaliados durante as gerações. Dessa forma, no primeiro algoritmo, foi implementado o algoritmo evolutivo básico representado na Figura 4.

3.2 Simulador

O simulador utilizado nesse trabalho foi inteiramente desenvolvido pela *Penn Engineering* da *University of Pennsylvania* nos Estados Unidos. Esse simulador foi construído no software MATLAB e é apresentado pelo professor PhD Vijay Kumar durante o curso online Aerial Robotics (KUMAR, s.d.). O simulador tem a capacidade de simular um eixo (altitude) de controle do quadricóptero o que é o suficiente para testar o algoritmo evolutivo. Ele depois traça uma curva de como a aeronave respondeu ao comando de entrada, como é visto na figura 13. Através dessa curva, podem-se extrair dados que serão utilizados como métricas de testes para o algoritmo, como sobresinal e tempo de subida.

Figura 13: Interface de Resposta do Simulador



Fonte: Elaborado pelo autor

3.2.1 Funcionamento

O simulador utiliza como dados de entrada a gravidade, ganhos k_p e k_d do controlador PD de altitude e os parâmetros construtivos massa e tamanho do braço do quadricóptero. A partir disso e da modelagem feita pela universidade, ocorre a simulação do quadricóptero tentando atingir a altura desejada.

Os códigos utilizados e modificados nesse trabalho se encontram na seção de apêndices e anexos, onde temos o código principal do controlador do simulador, feito pela *Penn Engineering* e os códigos do algoritmo genético usado para avaliar a resposta do simulador aos indivíduos utilizados como entrada. Dessa forma, o objetivo da simulação é a de obter ganhos k_p e k_d que minimizem o erro na trajetória do quadricóptero. De acordo com as equações abaixo.

$$e(t) = x_d(t) - x(t) \quad (3.4)$$

O erro é dado pela diferença entre a posição desejada e a posição real do quadricóptero. Dessa forma, deseja-se obter valores de ganhos do controlador PD que levem o valor do erro o mais rápido possível para zero, para que assim ele fique o mais próximo possível de sua trajetória desejada.

$$u(t) = \ddot{x}_d(t) + k_d \cdot \dot{e}(t) + k_p \cdot e(t) \quad (3.5)$$

O sistema pode ser descrito através da fórmula 3.5, que representa a entrada como uma função do erro e dos ganhos do controlador.

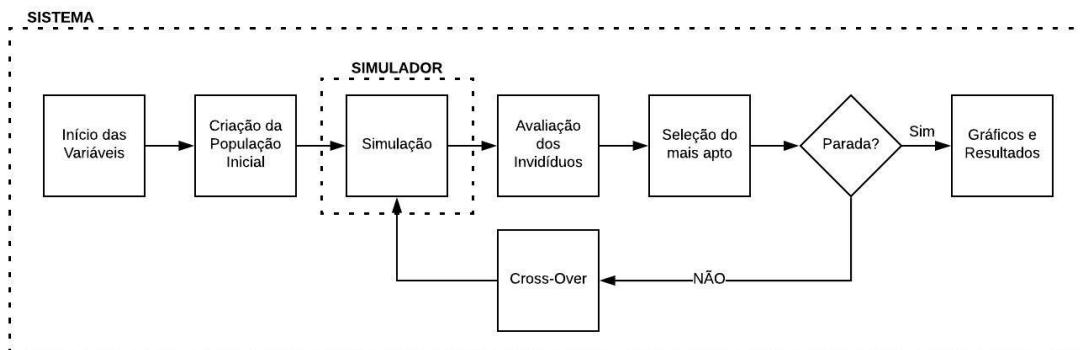
3.2.2 Modificações

Algumas alterações foram feitas no simulador para que pudesse ser utilizado nesse trabalho. A principal delas foi a que os ganhos do controlador deixaram de ser um parâmetro de entrada e podem agora ser configurados pelo próprio programa, para que assim possa ser utilizado pelo Algoritmo Evolutivo.

3.3 Implementação

Assim como o simulador, o algoritmo evolutivo foi implementado no software MATLAB para facilitar a comunicação e integração entre eles. O sistema representado abaixo pela figura 14 foi o primeiro a ser desenvolvido. Durante a implementação da ideia inicial, surgiu a necessidade de melhorias que posteriormente foram aplicadas e vieram a fazer parte do sistema.

Figura 14: Sistema Implementado com um Algoritmo Evolutivo Básico



Fonte: Elaborado pelo autor

3.3.1 Sistema Inicial

O primeiro sistema foi implementado conforme a figura 4, um algoritmo evolutivo básico. A população foi definida como sendo de 10 indivíduos e foi feito um teste para um número pequeno de gerações para avaliar o sistema.

Tabela 1: Parâmetros de Entrada do Primeiro Algoritmo

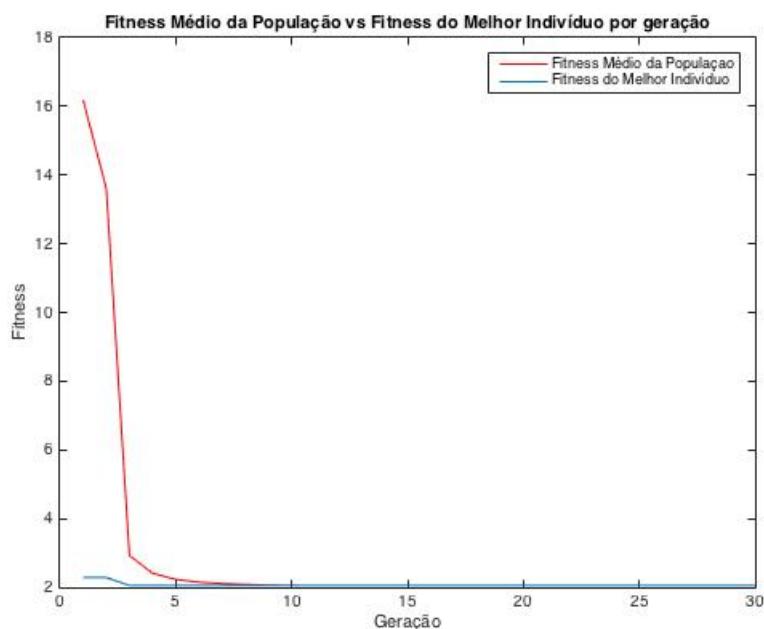
Variável	Valor
Tamanho da População (Indivíduos)	10
Gerações	30
Valor Máximo dos Atributos	5000

O objetivo inicial era de validar o algoritmo como viável, analisar a resposta dos indivíduos e definir possíveis melhorias que pudessem ser implementadas em seguida para uma conversão mais rápida dos indivíduos e uma busca em um espaço maior de soluções.

O sistema se mostrou viável e boas soluções foram encontradas quando o população convergia para um determinado valor. Porém, uma vez definidos os indivíduos eles buscavam o melhor dentro do alcance do individuo de maior e menor valor, somando-se ocasionalmente a ação da mutação. Dessa forma foram implementadas melhorias no algoritmo para que o sistema ainda tenha a aleatoriedade como forma de buscar sempre um máximo global da solução.

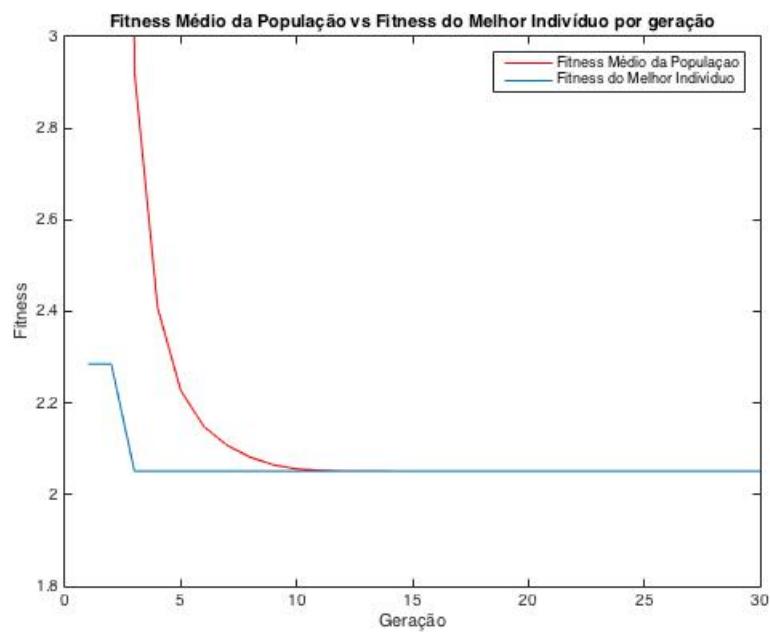
As figuras 15 e 16 mostram como se comportou o fitness dos indivíduos da população, convergindo para um valor aceitável de forma rápida. Na figura 17, é mostrado o universo de busca do algoritmo genético em todas as gerações através da dispersão dos indivíduos de todas as gerações.

Figura 15: Aptidão da População do Primeiro Sistema



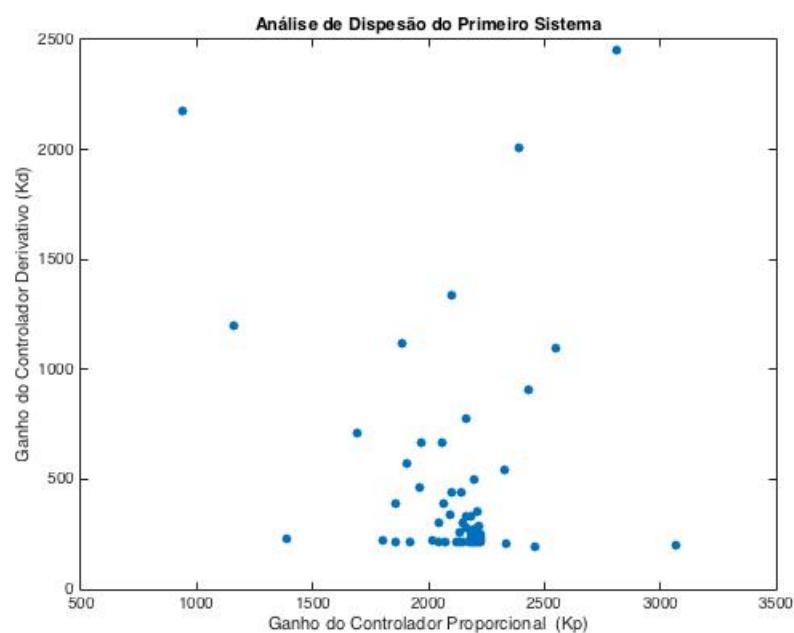
Fonte: Elaborado pelo autor

Figura 16: Aptidão da População do Primeiro Sistema em Detalhe



Fonte: Elaborado pelo autor

Figura 17: Análise de Dispersão dos Indivíduos do Primeiro Sistema

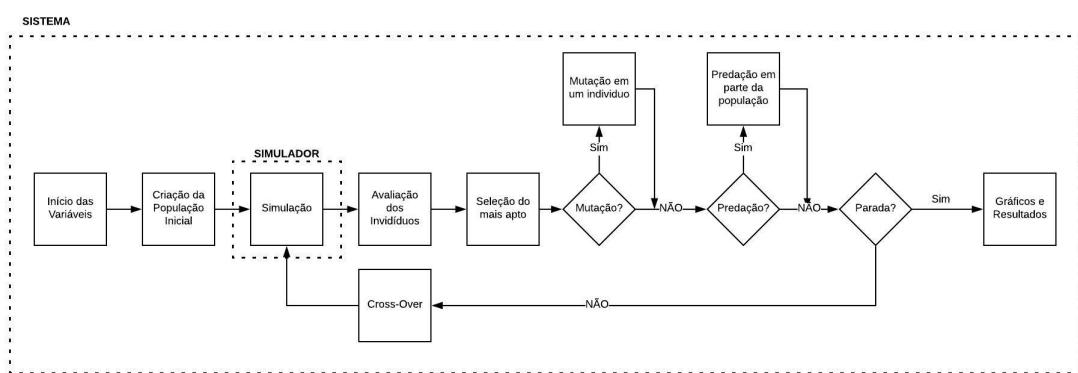


Fonte: Elaborado pelo autor

3.3.2 Segundo Sistema

No segundo sistema, foram adicionados dois operadores genético ao algoritmo do primeiro sistema. Primeiramente, a mutação, apresentada anteriormente, tinha como objetivo adicionar uma pequena aleatoriedade na busca por soluções com a periodicidade de 10 gerações e atingindo um indivíduo. Além da mutação um novo operador genético foi introduzido. Esse novo operador é chamado de predação e ocorre periodicamente assim como a mutação e substitui 2 indivíduos a cada vez que acontece.

Figura 18: Sistema Implementado com um Algoritmo Evolutivo Mais Complexo



Fonte: Elaborado pelo autor

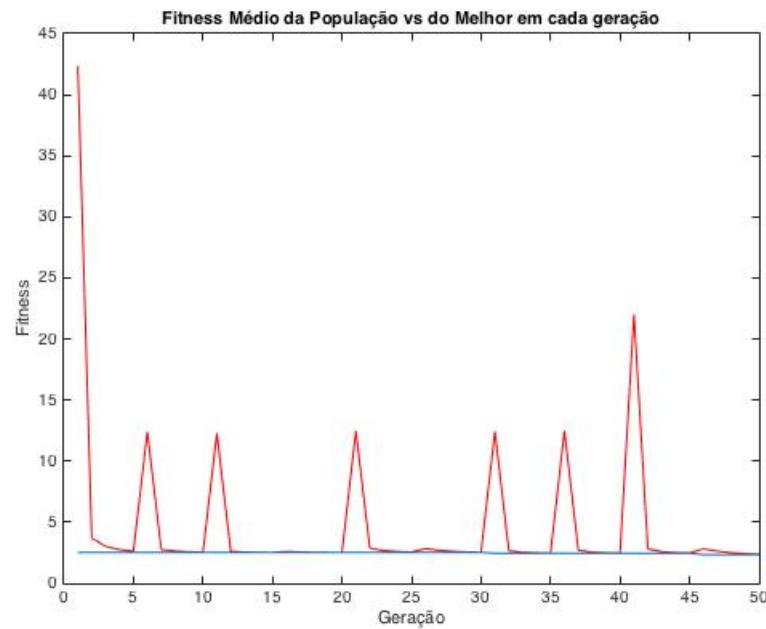
A predação, apresentada por Simões e Barone (2002), descarta alguns indivíduos e os substitui por indivíduos completamente novos. Assim, o sistema sempre tem indivíduos novos sendo inseridos a cada período de ocorrência dessa função. Dessa forma, o algoritmo agora evita de se concentrar em algum mínimo local e passa a buscar mais possibilidades de solução afim de encontrar mínimos globais, abrangendo potencialmente uma nova amostra de soluções. Nota-se na figura 21 que o algoritmo avaliou um universo maior de soluções do que quando comparado ao primeiro (Figura 17).

Tabela 2: Parâmetros de Entrada do Segundo Algoritmo

Variável	Valor
Tamanho da População (Indivíduos)	10
Gerações	50
Valor Máximo dos Atributos	5000
Periodicidade de Mutação (Gerações)	5
Periodicidade da Predação (Gerações)	10

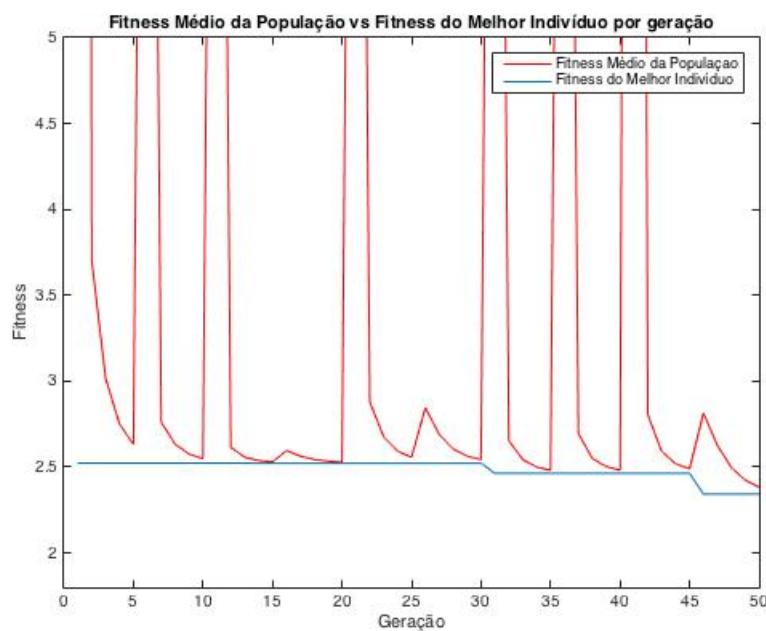
Essa mudança no sistema mostrou-se promissora e já em gerações iniciais avaliava novos indivíduos e os considerava como novas opções de solução. Isso é importante pois traz uma maior diversidade ao sistema. Na figura 19, nota-se que a introdução de indivíduos novos na população, deixou a aptidão média da população pior que a geração anterior. Esse efeito porém não pode ser analisado como um lado ruim da predação, uma vez que ela está introduzindo novos indivíduos na população que tem o potencial de unirem seu material genético com o indivíduo mais apto e se tornarem o novo indivíduo de melhor aptidão.

Figura 19: Aptidão da População do Segundo Sistema



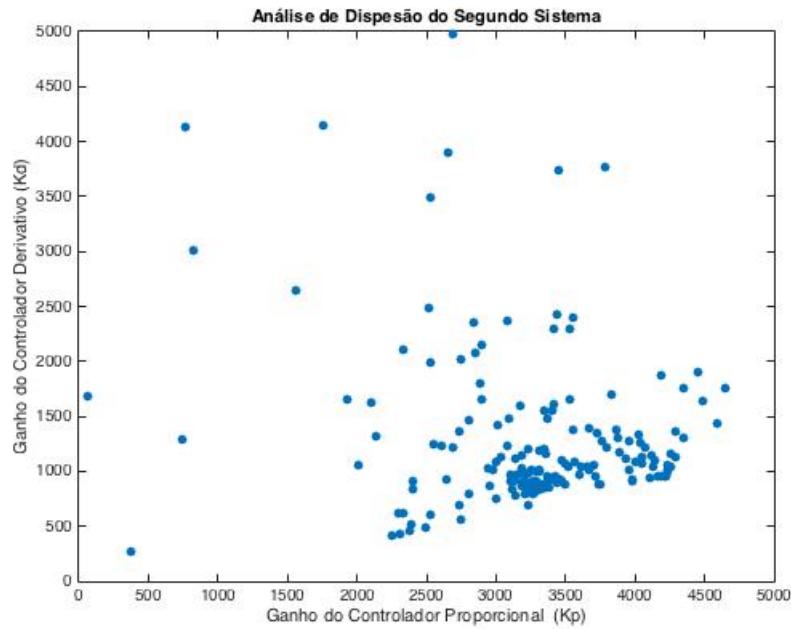
Fonte: Elaborado pelo autor

Figura 20: Aptidão da População do Segundo Sistema em Detalhe



Fonte: Elaborado pelo autor

Figura 21: Análise de Dispersão dos Indivíduos do Segundo Sistema



Fonte: Elaborado pelo autor

3.3.3 Sistema Final

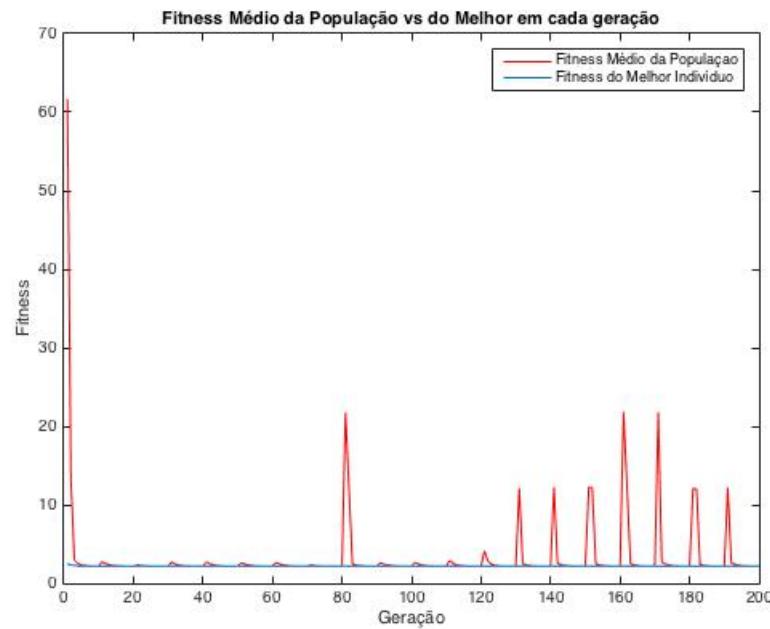
Com o segundo sistema funcionando e validado que ele se comportava melhor que o primeiro na busca de soluções, algumas configurações iniciais foram alteradas para testar como o algoritmo se comportava buscando soluções em um universo menor. O limite para as variáveis k_p e k_d foram consideravelmente reduzidos e o algoritmo foi avaliado novamente para um número grande de gerações.

Tabela 3: Parâmetros de Entrada do Sistema

Variável	Valor
Tamanho da População (Indivíduos)	10
Gerações	200
Valor Máximo dos Atributos	200
Periodicidade de Mutação (Gerações)	10
Periodicidade da Predação (Gerações)	20

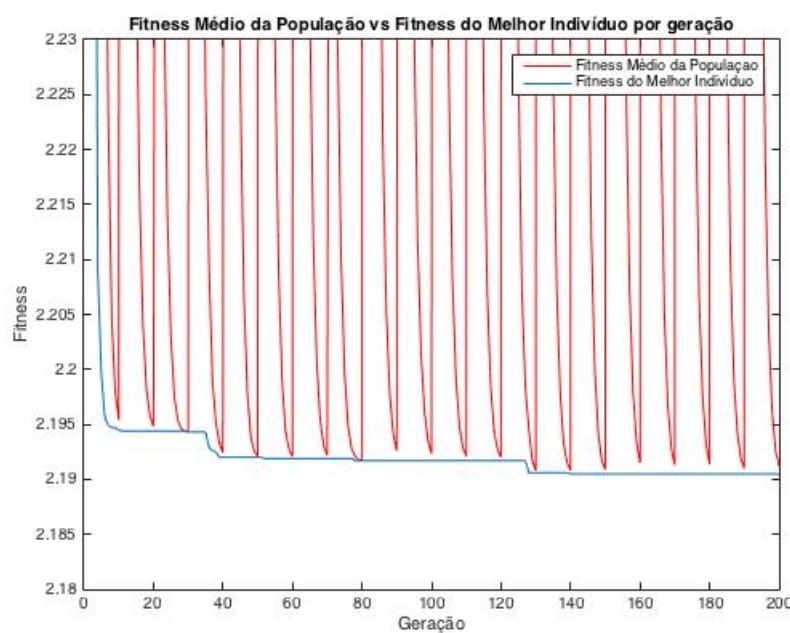
Nota-se na figura 19, que o comportamento do sistema na busca por soluções foi semelhante ao do segundo sistema, em que a introdução de novos indivíduos piorou a média da aptidão da população. Porém, após o surgimento de novos indivíduos, nota-se que ocasionalmente um novo melhor indivíduo é encontrado, pois a função aptidão do melhor indivíduo encontra um novo valor menor, como é visto em detalhe na figura 23.

Figura 22: Aptidão da População do Último Sistema



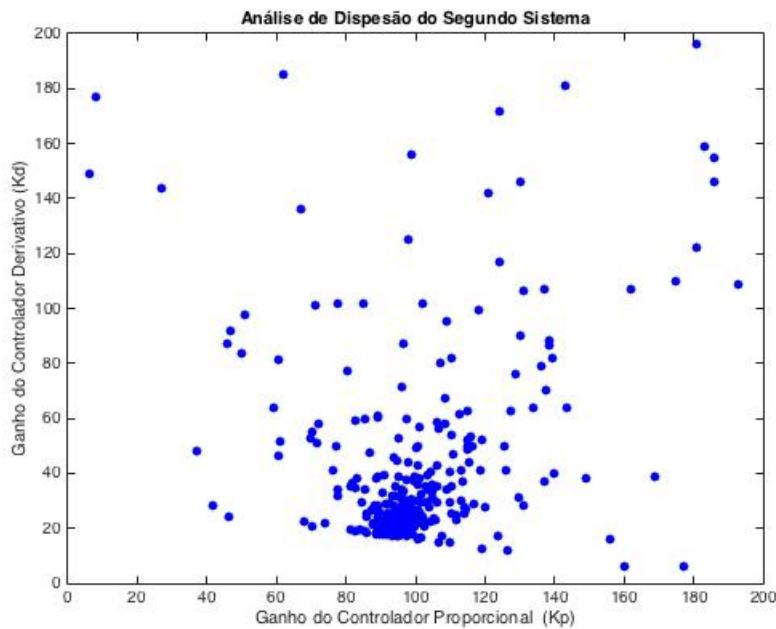
Fonte: Elaborado pelo autor

Figura 23: Aptidão da População do Último Sistema em Detalhe



Fonte: Elaborado pelo autor

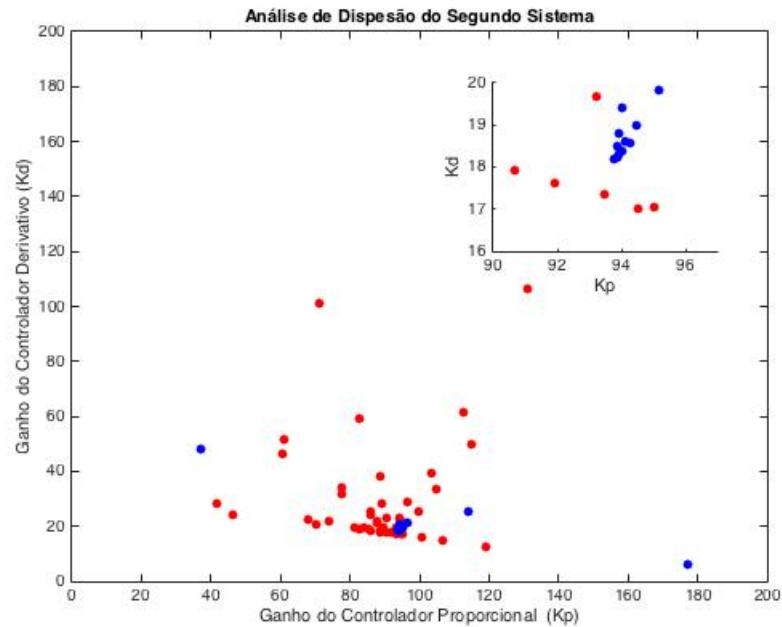
Figura 24: Análise de Dispersão dos Indivíduos do Último Sistema



Fonte: Elaborado pelo autor

A figura 25, mostra os indivíduos das cinco primeiras gerações (em vermelho) e os indivíduos das cinco últimas gerações (em azul). Podemos ver através dessa figura que o sistema converge em direção ao ponto de solução encontrado, porém ainda possui indivíduos buscando soluções em pontos distantes do ponto em que o atual melhor indivíduo se encontra, devido as técnicas de predação e mutação.

Figura 25: Análise de Dispersão dos Indivíduos do Último Sistema



Fonte: Elaborado pelo autor

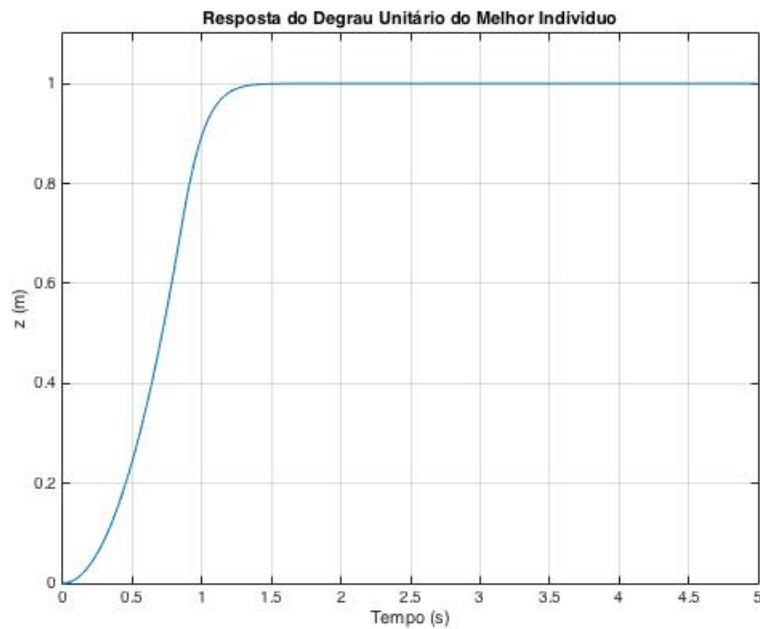
3.3.3.1 Análise dos Resultados

O indivíduo final teve uma resposta de acordo com a figura abaixo, mostrando um resultado satisfatório em relação resposta a entrada. O algoritmo se mostrou capaz de encontrar uma solução adequada dentro do universo estudado. Dessa forma, pode-se dizer que o algoritmo evolutivo desenvolvido é capaz de auxiliar na busca de ganhos de um controlador PD em um sistema.

Tabela 4: Resultados da Última Simulação

Variável	Valor
Ganho do Controlador Proporcional (k_p)	93,796
Ganho do Controlador Derivativo (k_d)	18,197
Sobresinal (%)	0.008
Tempo de Acomodação (s)	1,182

Figura 26: Análise da Resposta do Melhor Indivíduo do Último Sistema



Fonte: Elaborado pelo autor

Pode-se notar que o controlador exibiu um sobresinal baixo, mas superior a zero, ainda que a função aptidão tinha como objetivo minimizar o sobresinal juntamente ao tempo de estabilização. Pode-se interpretar isso como uma tentativa do algoritmo de minimizar o tempo de subida, sacrificando minimamente o fato de não haver sobresinal. Novos experimentos podem ser feitos variando a função aptidão e analisando novamente o comportamento, focando por exemplo em um controlador que potencialmente possua sobresinal, porém limitado a um valor, para dar mais liberdade ao algoritmo em encontrar novas soluções que permitam um tempo de estabilização mais rápido.

4 CONSIDERAÇÕES FINAIS

4.1 Conclusão

Neste trabalho foi desenvolvido um algoritmo evolutivo para encontrar ganhos adequados para um controlador Proporcional Derivativo de altitude de voo de um quadricóptero. Por meio da análise dos resultados conclui-se que essa é uma opção viável para encontrar soluções adequadas de ganhos de um controlador PD. O algoritmo evolutivo se mostrou viável ao tentar encontrar os ganhos para o sistema proposto, pois obteve um tempo de estabilização de 1,18 segundos e um sobresinal muito próximo a zero.

É importante observar que, para chegar nos resultados desejados, são feitas diversas simulações que demandam tempo e processamento. Desta forma, a nova metodologia proposta pode se tornar mais demorada comparada à métodos tradicionais de ajuste de PID quando se conhece a planta. Entretanto atualmente em drones, o ajuste as vezes é feito por tentativa e erro sem o conhecimento da planta e até mesmo com alteração da planta por instalação e remoção de objetos no drone, o que demanda tempo e traz um risco maior em segurança. Para realizar o ajuste dos ganhos em um drone real, usualmente os pilotos pousam o drone, mudam os parâmetros e decolam novamente testando a performance da aeronave, e assim sucessivamente até um resultado satisfatório ser encontrado. Uma alternativa possível é utilizar o algoritmo em um simulador para encontrar valores adequados para voo de um drone real e então implementar o algoritmo em um drone real, partindo de valores pré-definidos e configurar um universo menor de busca de soluções, para maximização da segurança.

Vale ressaltar que, neste projeto, o algoritmo foi implementado e testado em um simulador, o que traz uma segurança maior por não envolver testes em campo. As aplicações possíveis para um algoritmo evolutivo dentro de drones são várias. Uma possível aplicação para o algoritmo apresentado neste projeto é em testes de modelos de aeronave. Adicionalmente, se esse algoritmo rodar em paralelo em diversos computadores, é possível obter a resposta de forma rápida. Além disso esse método pode ser aplicado para configuração de diversos PID's aplicados em outras plantas que não apenas drones.

Todos os arquivos desse trabalho estarão disponíveis em: <<https://github.com/simoesusp/Hexacoptero/tree/master/Ewerton>>

4.2 Estudos Futuros

Atualmente, diversas aeronaves possuem um poder de processamento muito grande devido a dois fatores. O primeiro é o próprio hardware em que o sistema foi construído, que permite potencialmente que um algoritmo similar ao apresentado nesse trabalho possa ser aplicado e processado no hardware própria aeronave. Um segundo fator que pode ser explorado é de que aeronaves desse tipo podem se comunicar constantemente com um computador via telemetria, algumas inclusive são controladas dessa forma. Isso possibilita que algum algoritmo seja implementado no computador e que dessa forma controle a aeronave e/ou configure esses parâmetros em tempo real, fazendo testes o tempo todo.

Esse algoritmo pode hoje em dia ser implementado em diversos softwares presentes no mercado. Atualmente o software Mission Planner, de código aberto, possui capacidade de interpretar Python, uma

das linguagens mais utilizadas no mundo hoje por amadores e profissionais em diversas áreas. O algoritmo apresentado poderia ser implementado em Python e testado em uma aeronave real, desde que métodos de segurança sejam implementados juntos, pois tal aeronave não pode falhar e causar algum tipo de acidente.

Assim como implementado já nesse projeto, um algoritmo implementado em uma aeronave real teria também que levar em consideração os limites físicos do controlador existente na aeronave e dessa forma limitar os possíveis ganhos de tal controlador, para que um novo indivíduo aleatório da população não ultrapasse esses limites e cause algum tipo de dano.

REFERÊNCIAS

- BANZHAF, W. et al. **Genetic Programming: an introduction.** [S.l.]: Morgan Kaufmann, 1998.
- BARRETO, J. **Inteligência Artificial. No Limiar do Século XXI.** [S.l.: s.n.], 1997.
- BURWOOD-TAYLOR, L. **A Guide to the Investors Funding the Next Agricultural Revolution.** 2016. Disponível em: <<https://agfundernews.com/a-guide-to-investors-funding-the-next-agricultural-revolution.html>>.
- BUSETTI, F. **Genetic Algorithms Overview.** 2001.
- BUSINESS WEEK. **Texaco closes the loop.** New York: McGraw-Hill, 1959.
- DEB, K.; KALYANMOY, D. **Multi-Objective Optimization Using Evolutionary Algorithms.** New York, NY, USA: John Wiley & Sons, Inc., 2001. ISBN 047187339X.
- DESBOROUGH, L.; MILLER, R. **Increasing Customer Value of Industrial Control Performance Monitoring—Honeywell's Experience.** 2002.
- GARTNER. **Top Trends in the Gartner Hype Cycle for Emerging Technologies.** 2017. Disponível em: <<https://www.gartner.com/smarterwithgartner/top-trends-in-the-gartner-hype-cycle-for-emerging-technologies-2017/>>.
- GOEDDE, L.; HORII, M.; SANGHVI, S. **Pursuing the global opportunity in food and agribusiness.** Disponível em: <<https://www.mckinsey.com/industries/chemicals/our-insights/pursuing-the-global-opportunity-in-food-and-agribusiness>>.
- GOODWINE, B. **University of Notre Dame. Aerospace and Mechanical Engineering. AME 437: Control Systems Engineering.** 2003. Disponível em: <<https://controls.ame.nd.edu/ame437/S2003/hw1/>>.
- GUAZZELLI, P. R. U. **Controle preditivo de torque do motor de indução com otimização dos fatores de ponderação por algoritmo genético multiobjetivo.** 2017. Disponível em: <<http://www.teses.usp.br/teses/disponiveis/18/18153/tde-23032017-094031/>>.
- HOLLAND, J. H. **Adaptation in natural and artificial systems.** MIT Press Cambridge, MA, USA: [s.n.], 1975.
- KUMAR, V. **Robotics: Aerial Robotics by University of Pennsylvania.** s.d. Disponível em: <<https://www.coursera.org/learn/robotics-flight/home/info>>.
- MAMAGHANI, A.; ZONG, G. Viewpoint of neuromarketing technology assessment. **Proceedings of IAC-MEM 2015 in Vienna,** 2015.
- NIKU, S. B. **Introdução a Robótica: Análise, Controle e Aplicações.** Tradução de Sérgio Gilberto Taboada. Rio de Janeiro: LTC, 2014.
- POZO, A. et al. **Computação Evolutiva.** s.d. Disponível em: <<http://www.inf.ufpr.br/aurora/tutoriais/Ceapostila.pdf>>.
- SIMÕES, E. D. V.; BARONE, D. A. C. Predation: an approach to improving the evolution of real robots with a distributed evolutionary controller. **IEEE,** 2002.

- SINGH, S. **Agricultural Tech Investment Rises to Record \$25 Billion.** 2016. Disponível em: <<https://www.bloomberg.com/news/articles/2016-10-25/agricultural-technology-investment-rises-to-record-25-billion>>.
- SRIVASTAVA, A. **Technology in Agribusiness: Opportunities to Drive Value.** 2017.
- VILLAÇA, M. V. M.; SILVEIRA, J. L. **Uma breve história do controle automático.** 2013. Disponível em: <<http://ilhadigital.florianopolis.ifsc.edu.br/index.php/ilhadigital/article/view/49/44>>.
- ZIEGLER, J. G.; NICHOLS., N. B. Optimum settings for automatic controllers. **ASME**, 1942.
- ÅSTRÖM, K. J.; HAGGLUND, T. **PID Controller: Theory, Design and Tuning.** [S.l.]: Library of Congress Cataloging-in-Publication Data, 1994.
- ÅSTRÖM, K. J.; MURRAY, R. M. **Feedback Systems: An Introduction for Scientists and Engineers.** [S.l.], 2004.

Apêndices

APÊNDICE A – CÓDIGO PRINCIPAL

O código abaixo representa a Implementação do Algoritmo Evolutivo Integrado ao Simulador

```

1 % 1. Inicio / Setup
2 clear all
3 close all
4 clc
5
6 global Kp Kv video printgeracao;
7
8 % Setup
9 video = false;
10 salvar = true;
11 plotar = true;
12
13 tamPop = 10; % Define o tamanho de populacao
14 geracoes = 200; % Define o numero de geracoes
15 maxVar = 200; %Define valor maximo das variaveis de inicio
16
17 mutacao = 10; % Define periodicidade em geracoes para ocorrer a taxa de ...
     mutacao
18 taxMut = 5; %Define a ordem de divisao da mutacao
19
20 preda = 10; % Define a periodicidade em geracoes da predacao
21
22 % Variaveis Auxiliares
23
24 i = 1;
25 melhor = 1;
26 printgeracao = 1;
27
28 % 2. Criar populacao Inicial
29 % Kp - Individuo1
30 % Kv - Individuo2
31
32 for i=1:tamPop
33
34     individuo1(i) = round( maxVar*rand() ) ;
35     individuo2(i) = round( maxVar*rand() ) ;
36
37 end
38
39 % 3. Loop
40

```

```
41 for geracaoAtual = 1:geracoes
42
43
44     for i=1:tamPop % 4.calculo do Fitness (avaliacao dos individuos)
45
46         Kp = individuo1(i);
47         Kv = individuo2(i);
48
49         run runsim;
50
51         settlingTime(i) = sim_info.SettlingTime;
52         overshoot(i) = (sim_info.Max-z_des)*100;
53
54
55         if overshoot(i) < 0;
56             overshoot(i) = 1;
57         else
58             overshoot(i) = overshoot(i) + 1;
59         end
60
61
62         fitness(i) = settlingTime(i) + overshoot(i) ;
63
64         if isnan( fitness(i) ) ;
65             fitness(i) = 100; %Fitness maximo de 100 (pior valor)
66         end
67
68
69         delete(findall(0,'Type','figure'))
70
71     end
72
73     % 5. selecao do Melhor
74
75     melhor = 1;
76
77     for i=1:tamPop
78         if fitness(i)≤fitness(melhor)
79             melhor = i;
80         end
81     end
82
83     % 5. Cross-Over
84
85     %Swapping
86     auxSwap1 = 1;
87     auxSwap2 = 1;
88
```

```

89     auxSwap1 = individuo1(melhor);
90     auxSwap2 = individuo2(melhor);
91
92     individuo1(melhor) = individuo1(1);
93     individuo2(melhor) = individuo2(1);
94
95     individuo1(1) = auxSwap1;
96     individuo2(1) = auxSwap2;
97
98         %Obs: O fitness e suas componentes nao estao "swapping"
99
100    %Cross-Over
101
102    for i=2:tamPop
103
104        individuo1(i)=( individuo1(i) + individuo1(1) )/2;
105        individuo2(i)=( individuo2(i) + individuo2(1) )/2;
106    end
107
108    % 6. predacao / Novos 2 Individuos (A cada "preda" geracoes)
109
110    if rem(geracaoAtual,preda) == 0;
111        individuo1(tamPop) = round( maxVar*rand() );
112        individuo2(tamPop) = round( maxVar*rand() );
113        individuo1(tamPop-1) = round( maxVar*rand() );
114        individuo2(tamPop-1) = round( maxVar*rand() );
115    end
116
117    % 7. mutacao (A cada "mutacao" geracoes sempre no segundo individuo)
118
119    if rem(geracaoAtual,mutacao) == 0;
120        individuo1(2) = individuo1(2) + round( maxVar*rand()/taxMut );
121        individuo2(2) = individuo2(2) + round( maxVar*rand()/taxMut );
122    end
123
124    %8. Salva o historico
125
126    aux_fit_historico = 1;
127
128    fit_grupo_historico(geracaoAtual) = mean(fitness);
129    fit_melhor_historico(geracaoAtual) = min(fitness);
130
131    printgeracao = printgeracao+1;
132
133    %Para grafico de dispersao
134
135    for i=1:tamPop
136        dispersao(1,geracaoAtual*tamPop+i-tamPop) = individuo1(i);

```

```
137     dispersao(2,geracaoAtual*tamPop+i-tamPop) = individuo2(i);  
138 end  
139  
140  
141  
142 end  
143 % 9. Voltar para *  
144  
145  
146  
147 %% Salvar historico e Resultados  
148  
149 for i=1:length(dispersao)  
150     scat1(i) = dispersao(1,i);  
151     scat2(i) = dispersao(2,i);  
152 end  
153  
154 if salvar  
155     csvwrite('scat1_200.csv',scat1)  
156     csvwrite('scat2_200.csv',scat2)  
157     csvwrite('dispersao_200.csv',dispersao)  
158     csvwrite('fit_grupo_historico_200.csv',fit_grupo_historico)  
159     csvwrite('fit_melhor_historico_200.csv',fit_melhor_historico)  
160 end
```

Anexos

ANEXO A – SIMULADOR

O código abaixo representa a função de controle do simulador e passa a referência de posição que a aeronave simulada deve alcançar.

```
1 % Hover
2 z_des = 1;
3
4 % Step
5 % z_des = 1;
6
7 % Given trajectory generator
8 trajhandle = @(t) fixed_set_point(t, z_des);
9
10 % This is your controller
11 controlhandle = @controller;
12
13 % Run simulation with given trajectory generator and controller
14 [t, z] = height_control(trajhandle, controlhandle);
15
16 % Sample code to get more info on the response
17 sim_info = lsiminfo(z, t, z_des);
18 %disp(['Settling time [s]: ', num2str(sim_info.SettlingTime)]);
19 %disp(['Overshoot [%]: ', num2str(max(0, (sim_info.Max-z_des)*100))]);
```


ANEXO B – FUNÇÃO PARÂMETROS E FUNÇÃO DO CONTROLADOR

Os códigos abaixo passam em funções os dados de configuração do simulador.

```

1 function [ params ] = sys_params_limit_thrust()
2
3 % Physical properties
4 params.gravity = 9.81;
5 params.mass = 0.18;
6 params.arm_length = 0.086;
7
8 % Actuator limits
9 params.u_min = 0;
10 params.u_max = 1.2*params.mass*params.gravity;
11
12 end

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21

```

```

1 function [ u ] = pd_controller(~, s, s_des, params)
2 %PD_CONTROLLER PD controller for the height
3 %
4 %    s: 2x1 vector containing the current state [z; v_z]
5 %    s_des: 2x1 vector containing desired state [z; v_z]
6 %    params: robot parameters
7
8 % persistent last_err
9 %
10 % if isempty(last_err)
11 %     last_err = 0;
12 % end
13
14 global Kp Kv;
15
16 err = s_des(1) - s(1);
17 d_err = s_des(2) - s(2);
18 u = params.mass*(Kp*err + Kv*d_err + params.gravity);
19
20 % last_err = err;
21 end

```


ANEXO C – CONTROLADOR

O código abaixo é o principal responsável pela simulação, gerando as curvas de simulação e de resposta a entrada.

```

1 function [t_out, z_out] = height_control(trajhandle, controlhandle)
2
3 global video printgeracao;
4
5 addpath('utils');
6
7 % video = false;
8 video_filename = 'height_control.avi';
9
10 params = sys_params;
11
12 % real-time
13 real_time = false;
14
15 %% ***** FIGURES *****
16 disp('Initializing figures... ')
17 if video
18     video_writer = VideoWriter(video_filename, 'Uncompressed AVI');
19     open(video_writer);
20 end
21 h_fig = figure;
22 sz = [1000 600]; % figure size
23 screensize = get(0, 'ScreenSize');
24 xpos = ceil((screensize(3)-sz(1))/2); % center the figure on the screen ...
25         horizontally
26 ypos = ceil((screensize(4)-sz(2))/2); % center the figure on the screen ...
27         vertically
28 set(h_fig, 'Position', [xpos ypos sz])
29
30 h_3d = subplot(1,2,1);
31 axis equal
32 grid on
33 view(0,0);
34 xlabel('x [m]'); ylabel('y [m]'); zlabel('z [m]');
35 h_2d = subplot(1,2,2);
36 plot_2d = plot(h_2d, 0, 0);
37 grid on;
38 xlabel('t [s]'); ylabel('z [m]');

```

```
39 quadcolors = lines(1);
40
41 set(gcf, 'Renderer', 'OpenGL')
42
43 %% ***** INITIAL CONDITIONS *****
44 max_iter = 100;           % max iteration
45 starttime = 0;            % start of simulation in seconds
46 tstep = 0.01;             % this determines the time step at which the ...
    solution is given
47 cstep = 0.05;             % image capture time interval
48 nstep = cstep/tstep;
49 time = starttime;         % current time
50
51 % Get start and stop position
52
53 des_start = trajhandle(0);
54 des_stop = trajhandle(inf);
55 stop_pos = des_stop(1);
56 x0 = des_start;
57 xtraj = nan(max_iter*nstep, length(x0));
58 ttraj = nan(max_iter*nstep, 1);
59
60 x = x0;                  % state
61
62 pos_tol = 0.01;
63 vel_tol = 0.01;
64
65 %% ***** RUN SIMULATION *****
66 disp('Simulation Running....')
67 % Main loop
68 for iter = 1:max_iter
69
70     timeint = time:tstep:time+cstep;
71
72     tic;
73     % Initialize quad plot
74     if iter == 1
75         subplot(1,2,1);
76         quad_state = simStateToQuadState(x0);
77         QP = QuadPlot(1, quad_state, params.arm_length, 0.05, ...
            quadcolors(1,:), max_iter, h_3d);
78         quad_state = simStateToQuadState(x);
79         QP.UpdateQuadPlot(quad_state, time);
80         h_title = title(h_3d, sprintf('iteration: %d, time: %4.2f', iter, ...
            time));
81     end
82
83     % Run simulation
```

```

84 [tsave, xsave] = ode45(@(t,s) sys_eom(t, s, controlhandle, trajhandle, ...
85     params), timeint, x);
86 x = xsave(end, :)';
87 % Save to traj
88 xtraj((iter-1)*nstep+1:iter*nstep,:) = xsave(1:end-1,:);
89 ttraj((iter-1)*nstep+1:iter*nstep) = tsave(1:end-1);
90
91 % Update quad plot
92 subplot(1,2,1)
93 quad_state = simStateToQuadState(x);
94 QP.UpdateQuadPlot(quad_state, time + cstep);
95 set(h_title, 'String', sprintf('iteration: %d, time: %4.2f, ...
96     geracao:%d', iter, time + cstep, printgeracao))
97 time = time + cstep; % Update simulation time
98
99 if video
100     writeVideo(video_writer, getframe(h_fig));
101 end
102
103 t = toc;
104 % Check to make sure ode45 is not timing out
105 if(t > cstep*50)
106     err = 'Ode solver took too long for a step. Maybe the controller is ...
107         unstable.';
108     disp(err);
109     break;
110 end
111 % Pause to make real-time
112 if real_time && (t < cstep)
113     pause(cstep - t);
114 end
115
116 end
117 % Check termination criteria
118 if norm(stop_pos - x(1)) < pos_tol && norm(x(2)) < vel_tol
119     err = [];
120 else
121     err = 'Did not converge';
122 end
123
124 disp('Simulation done');
125 if video
126     close(video_writer);
127 end

```

```
128
129 if ~isempty(err)
130 disp(['Error: ', err]);
131 end
132 t_out = ttraj(1:iter*nstep);
133 z_out = xtraj(1:iter*nstep,1);
134
135 end
```