Sistemas de Integração e Automação Industrial

Professor Dr. Orides Morandin Jr.

Robô Cartesiano no Eixo Z: Um Estudo Comparativo de Técnicas de Controle

Algoritmo Genético

Bruna Zamith Santos, RA: 628093
Matheus Vrech Silveira Lima, RA: 727349

São Carlos,
Dezembro de 2019

1. Proposta

O projeto consiste no desenvolvimento de um Robô Cartesiano no Eixo Z. Ele atua movendo uma carga, no eixo Z, até uma posição desejada (definida pelo operador do robô). O objetivo principal é fazer um estudo comparativo do uso de três diferentes técnicas de controle implementadas neste robô, a saber: PID, PID Adaptativo e Lógica Fuzzy.

Para avaliar essas técnicas, são considerados três parâmetros de desempenho:

- *Overshooting*: O *overshooting* é um dos efeitos indesejados mais severos. Se assumirmos um sistema de primeira ordem, pode-se considerar que o *overshooting* é mínimo. No entanto, ao variar a carga (força peso atuante) no robô, o comportamento muda e essa hipótese anterior deixa de ser válida. Então, fica ainda mais evidente a necessidade de analisar outras técnicas de controle;
- Erro em regime: Representa a diferença entre a posição esperada do robô e a posição obtida. Idealmente, esse erro é zero. Na prática, é definida uma porcentagem de erro admissível (a ser apresentada na Seção 3.1).
- Tempo de resposta: É o parâmetro que define o tempo que o robô leva para atingir a posição desejada (dentro da porcentagem de erro admissível). Quanto menor o tempo de resposta, melhor a produtividade do robô.

Além disso, o robô foi desenvolvido levando em consideração 3 usuários:

- Operador: Responsável por operar o robô, colocando a carga e definindo a posição desejada;
- Manutenção: Responsável por realizar testes periódicos no robô;
- Desenvolvedor: Responsável por desenvolver e otimizar o robô.

Cada um dos três níveis de usuário tem permissão para acessar diferentes funcionalidades, dependendo do caso de uso.

O projeto não se restringe ao estudo comparativo do uso das três técnicas de controle, mas também avalia a implementação de funcionalidades que seriam interessantes para o uso do robô na indústria. Uma dessas funcionalidades, que é o foco do presente relatório, é o Algoritmo Genético para otimização da técnica de controle PID.

De acordo com [1], uma pesquisa com mais de 11,000 controladores utilizados nas indústrias químicas mostraram que 97% usa a técnica do PID. E desses, a maioria utiliza o método de Ziegler-Nichols para sintonizar os parâmetros de entrada Kp, Ki e Kd. O Ziegler-Nichols foi proposto em 1942 e é um método controlado de tentativa-e-erro [2]. Ele possui algumas desvantagens importantes, como o fato de ser um método demorado e de exigir que levemos o processo à

beira da instabilidade enquanto procuramos o ganho final. Esses dois aspectos podem ser indesejáveis para a indústria quando pensamos numa aplicação prática.

De outro ponto de vista, o problema de definição dos parâmetros de entrada Kp, Ki e Kd é um problema combinatório. Sendo assim, cabe o estudo de uma metaheurística de busca para otimização da definição desses parâmetros, como o Algoritmo Genético (AG). Eles têm sido aplicados em praticamente todas as áreas de otimização, design e aplicações, e existem diferentes variantes e modelos híbridos de AG, como os apresentados em [3].

Nesse contexto, o desenvolvedor seria o usuário responsável por acionar a otimização dos parâmetros de entrada pelo AG. Por tratar-se de um etapa crucial do projeto e cujo conhecimento sobre a construção do robô é fundamental, não permitiremos que essa etapa seja delegada a outros usuários.

Pensando em termos de aplicação na indústria, essa otimização deverá ocorrer em duas situações: Antes da entrega do produto ao cliente e em manutenções periódicas. A segunda situação é relevante porque depois de algum tempo, o robô pode sofrer desgaste e os valores ótimos das constante Kp, Ki e Kd vão, consequentemente, mudar.

Na Seção 2 a seguir, será feita uma breve introdução de como o AG funciona e então quais os parâmetros considerados para sua implementação e funcionamento no projeto do Robô Cartesiano no Eixo Z.

2. Algoritmo Genético

2.1 Funcionamento Geral

Como exposto anteriormente, o Algoritmo Genético é uma metaheurística de busca e atua como um algoritmo de otimização, buscando mínimos (ou máximos) locais de uma *fitness function* definida.

Ele é construído em cima de alguns conceitos, como:

- Cromossomo: Possível solução do problema;
- *Fitness Function*: Ou função de aptidão. Avalia o resultado que um cromossomo consegue obter;
- População: Conjunto de indivíduos, cada um representa uma solução;

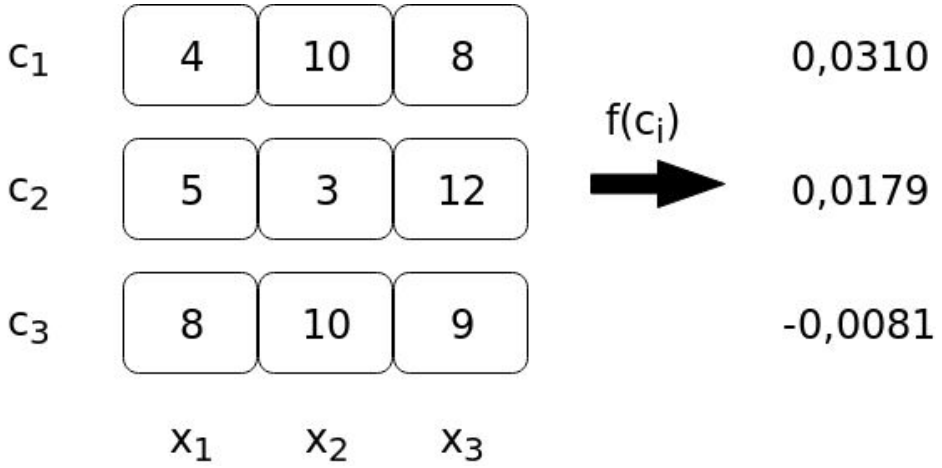

Figura 1: Exemplo de uma população (cromossomos ci) e aplicação de uma *fitness function* f(ci) sobre ela.

Definido o domínio do problema de acordo com os conceitos apresentados, é possível aplicar o AG. Ele é um algoritmo iterativo, que funciona de acordo com o ciclo representado na Figura 2, cujas etapas principais são:

- Seleção: Baseados na aptidão, os indivíduos mais aptos devem possuir maior probabilidade de seleção;
- Cruzamento: Os indivíduos selecionados são cruzados dois a dois;
- Mutação: Modifica parte do cromossomo (aleatoriamente ou por heurística). Gera soluções inéditas;
- Elitismo: Seleciona uma porcentagem dos melhores cromossomos para a nova população, evitando que se percam os melhores indivíduos após o cruzamento e a mutação.

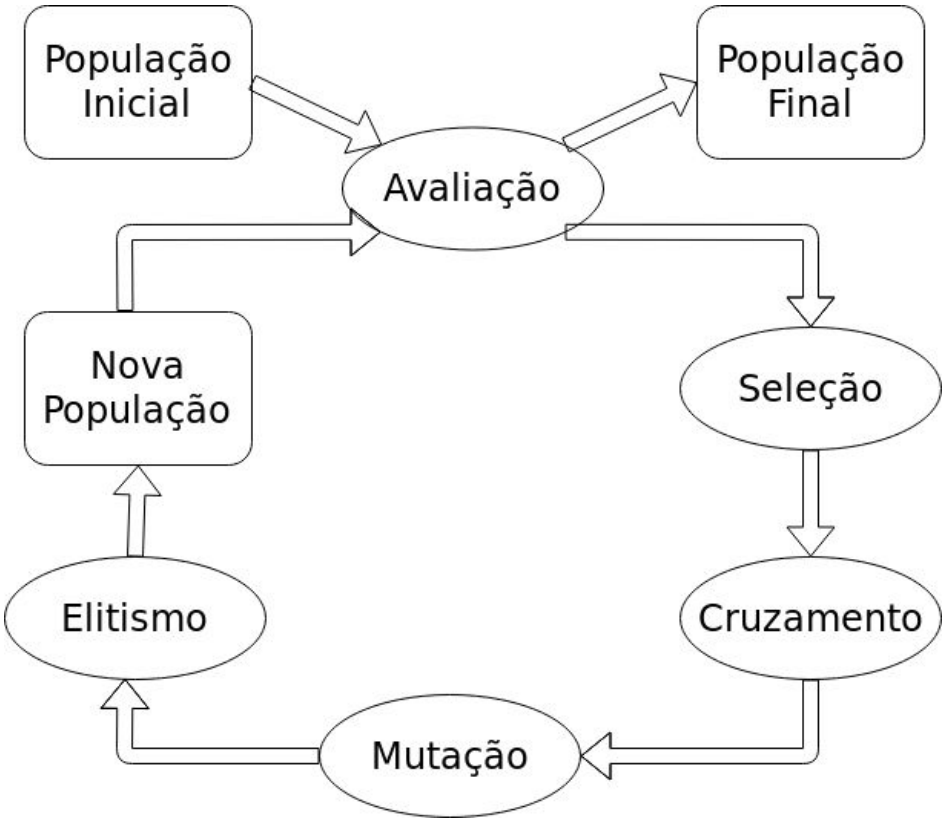

Figura 2: Ciclo convencional de um AG.

Figura 3: Exemplo de cruzamento e mutação dos cromossomos.

2.2 Parâmetros do Algoritmo

Para o domínio do problema, anteriormente exposto na Seção 1, foram definidos os seguintes parâmetros no que tange a implementação do AG:

- Cada cromossomo é constituído por três genes, que representam os valores de Kp, Ki e Kd;
- A *Fitness Function* é o tempo de resposta e o objetivo é sua minimização. Isso porque consideramos que os parâmetros da técnica de controle PID

implementada já consideram a redução de *overshooting* e de erro em regime (esses pontos vão ser melhor discutidos na Seção 3.1). Assim, com o AG podemos reduzir o tempo de resposta, aumentando a produtividade do robô;

- O ciclo do AG se repete até atingir um número pré-definido de iterações. Experimentos ([4]) indicaram que a partir de 300 gerações, não há mais melhora significativa dos resultados. Logo, é razoável considerar 300 como sendo o valor máximo de gerações;
- Quanto maior a população inicial, melhor. Referências como [4] e [5] sugerem não ultrapassar 100 cromossomos por se preocuparem com tempo de execução. Para nosso projeto, esta não é uma questão prioritária, pois assume-se que o software do AG será executado apenas pelo desenvolvedor do sistema e em situações específicas (como apresentado na Seção 1). Todavia, para fins de teste, vamos trabalhar com populações de tamanho 100;
- A seleção é baseada no método do Torneio. Ele trabalha escolhendo com a mesma probabilidade N indivíduos. E então, seleciona o cromossomo com a maior aptidão dentre esses N escolhidos. Esses dois passos são repetidos até preencher o tamanho da população. O tamanho N é definido sendo 15;
- O cruzamento é realizado apenas sobre o gene Kp;
- A mutação é realizada apenas sobre o gene Ki;
- Ainda em [4], não se recomenda realizar mutações com tanta frequência devido à degradação da capacidade de busca. Por essa razão, definimos a probabilidade de mutação como sendo 10%;
- É preciso receber um intervalo aceitável para cada constante. Esse intervalo deve, na prática, ser encontrado pelo método de Ziegler-Nichols. Como será detalhado futuramente na Seção 3.1, para fins de teste, foram definidos os seguintes intervalos: Kp [1.5; 100], Ki [1; 5] e Kd [1; 1.5]. Esses valores foram obtidos a partir de experimentos, buscando intervalos que minimizam o *overshooting*.

A Tabela 1 resume os parâmetros expostos:

Parâmetro	**Informação/Valor**
Cromossomo	3 Genes: Kp, Ki, Kd
Fitness Function	Tempo de Resposta (Minimização)
Número de Iterações	300
Tamanho da População	100
Tamanho da População no Torneio	15

Gene de Cruzamento	Kp
Gene de Mutação	Ki
Probabilidade de Mutação	0.1
Intervalo de Kp	[1.5; 100]
Intervalo de Ki	[1; 5]
Intervalo de Kd	[1; 1.5]

Tabela 1: Resumo dos parâmetros do AG definidos.

Desses parâmetros, os únicos que são alteráveis por interface ou diretamente por chamado de função são os intervalos de Kp, Ki e Kd. Isso porque assumimos que em casos de manutenção periódica após desgaste eventual do robô, os intervalos aceitáveis também mudam e deverão ser ajustados. Os outros parâmetros podem e devem ser fixos durante todo o ciclo de vida do robô.

3. Implementação

Todo o código do AG foi desenvolvido em linguagem de programação C++. Os parâmetros todos são ajustáveis no cabeçalho do programa.

3.1 Parâmetros de Desempenho

Como introduzido na Seção 1, estamos trabalhando com 3 parâmetros de desempenho:

- *Overshooting*: Assumimos que a estratégia de controle na qual o AG vai atuar (nesse caso, o PID) já levou em consideração a minimização do *overshooting* nas escolhas dos intervalos de Kp, Ki e Kd. Para a simulação a ser descrita na Seção 4, consideramos intervalos que não permitissem que o *overshooting* ultrapassasse 30%;
- Erro em regime: Assim como para o *overshooting*, assumimos que o erro em regime foi considerado na implementação da técnica de controle. Para efeitos de teste, foi definida uma margem de erro de 5%.
- Tempo de resposta: É o parâmetro que é levado em consideração na otimização pelo AG, isto é, sua *fitness function*. Buscamos a minimização do tempo de resposta, o que levaria a um aumento da produtividade do robô.

3.2 Diagramas

A seguir são apresentados 4 diagramas definidos para a implementação do Algoritmo Genéticos: Casos de Uso, Atividades, Classes e Sequências.

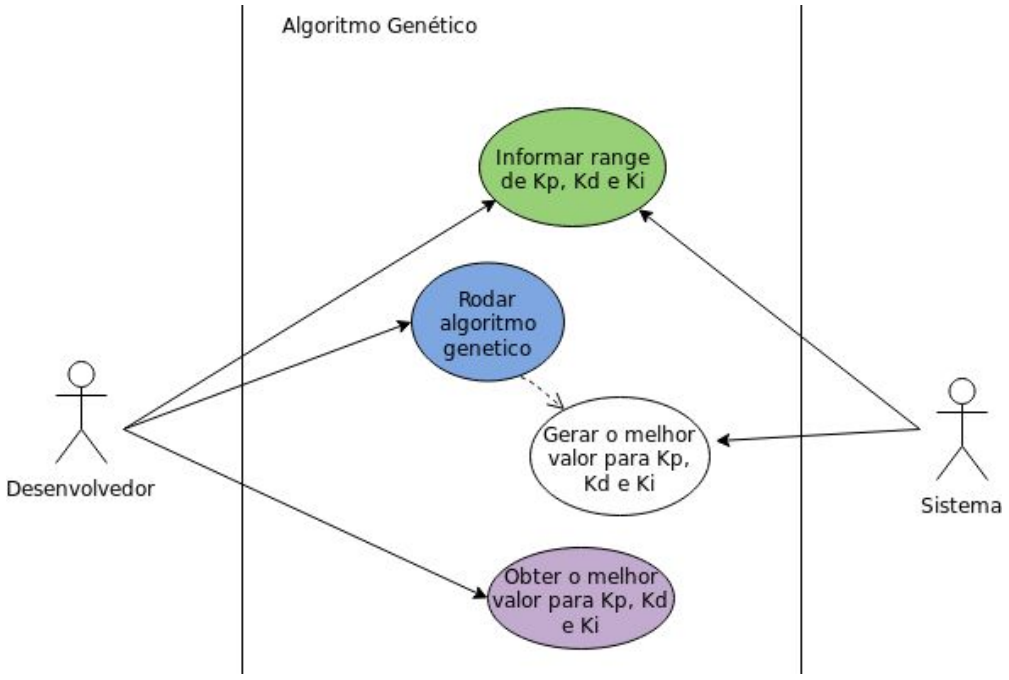

Figura 4: Diagrama de Casos de Uso.

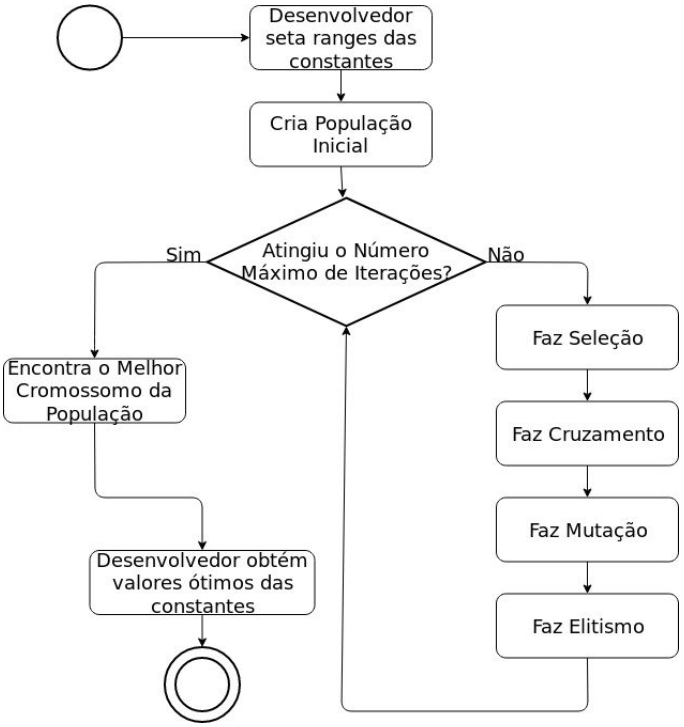

Figura 5: Diagrama de Atividades.

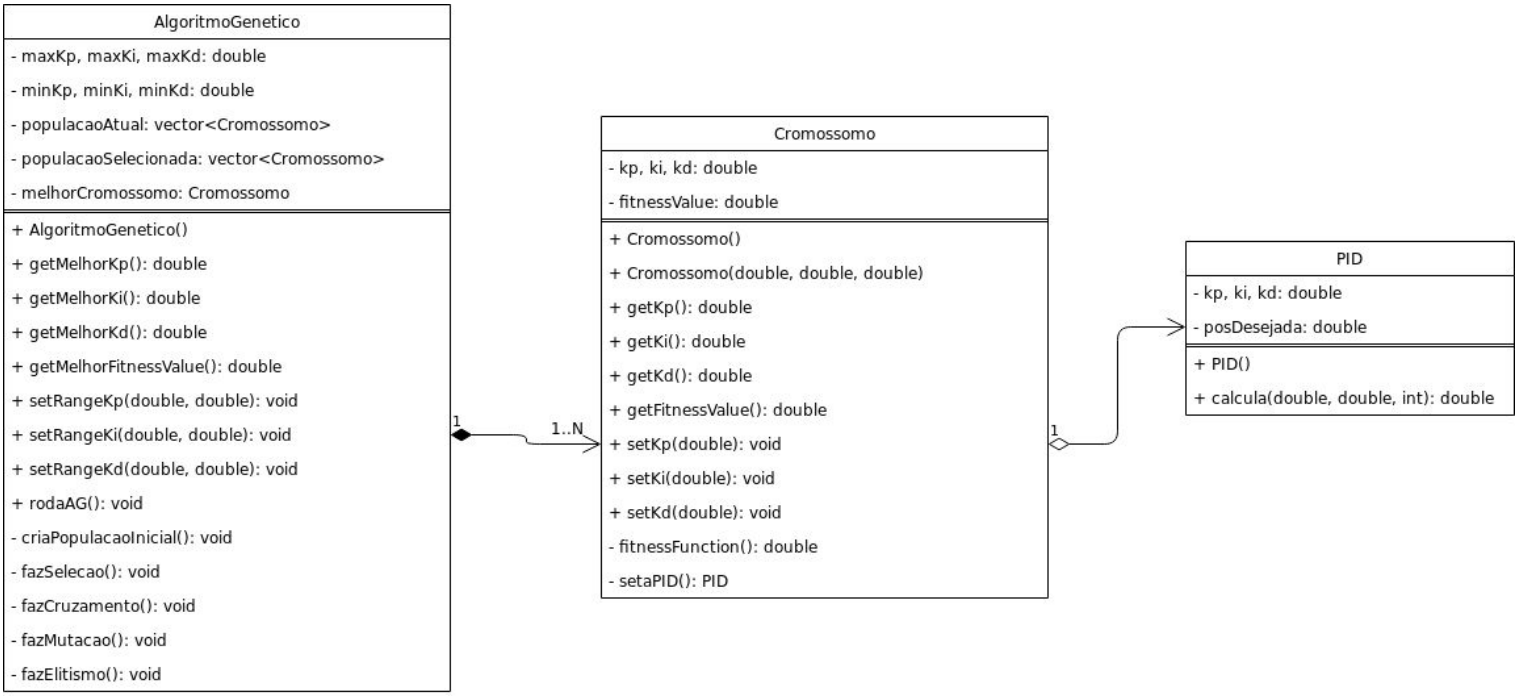

Figura 6: Diagrama de Classes.

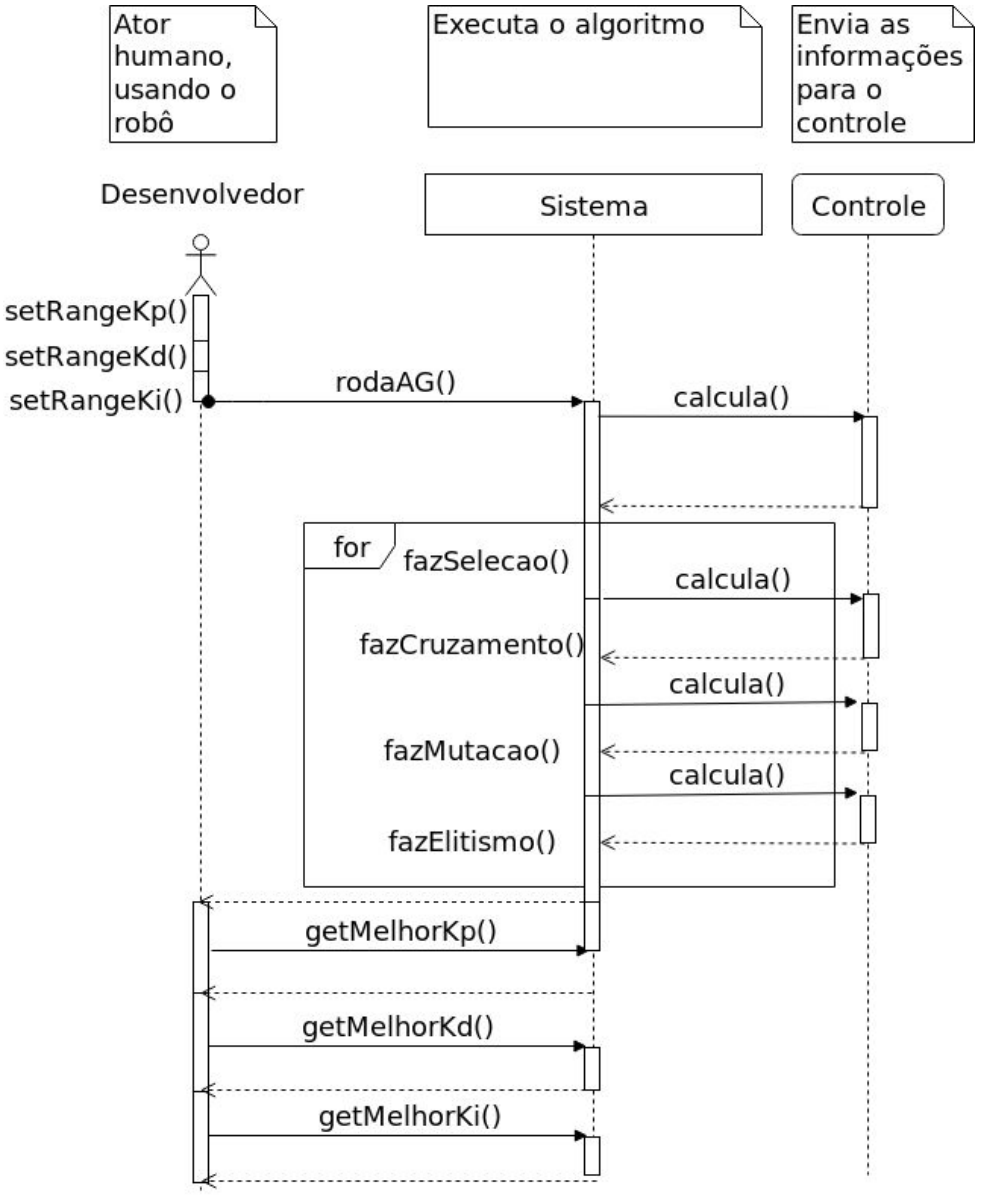

Figura 7: Diagrama de Sequências.

4. Testes e Simulação

Na Seção 3.2 foram apresentados os diagramas referentes ao AG implementado. Como é possível ver no Diagrama de Classes (Figura 6) e no Diagrama de Sequências (Figura 7), foi criada uma função chamada "calcula" dentro da classe PID. Ela é responsável por executar a técnica de controle PID e retornar o tempo de resposta da execução.

A fim de ter um ambiente de testes interno do projeto do AG para o robô, foi desenvolvida uma simulação em Matlab/Simulink e uma classe em C++ (denominada SimulacaoMatlab) que serve de interface entre o AG e o projeto do Simulink. Ela pode ser colocada no lugar da função "calcula", e torna-se independente do projeto físico do robô.

A comunicação entre o código do AG e a Simulação do Matlab é representada na Figura 8.

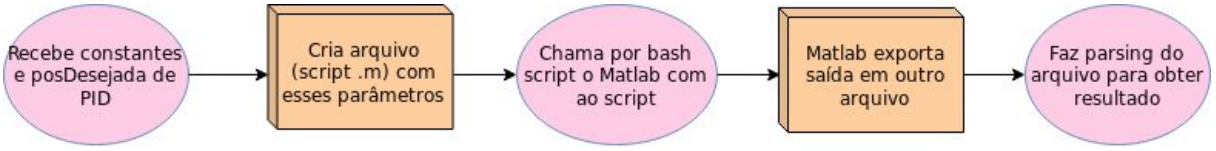

Figura 8: Representação da comunicação entre o AG e a Simulação.

A simulação foi toda construída por meio de diagrama de blocos em Simulink, exposto na Figura 9.

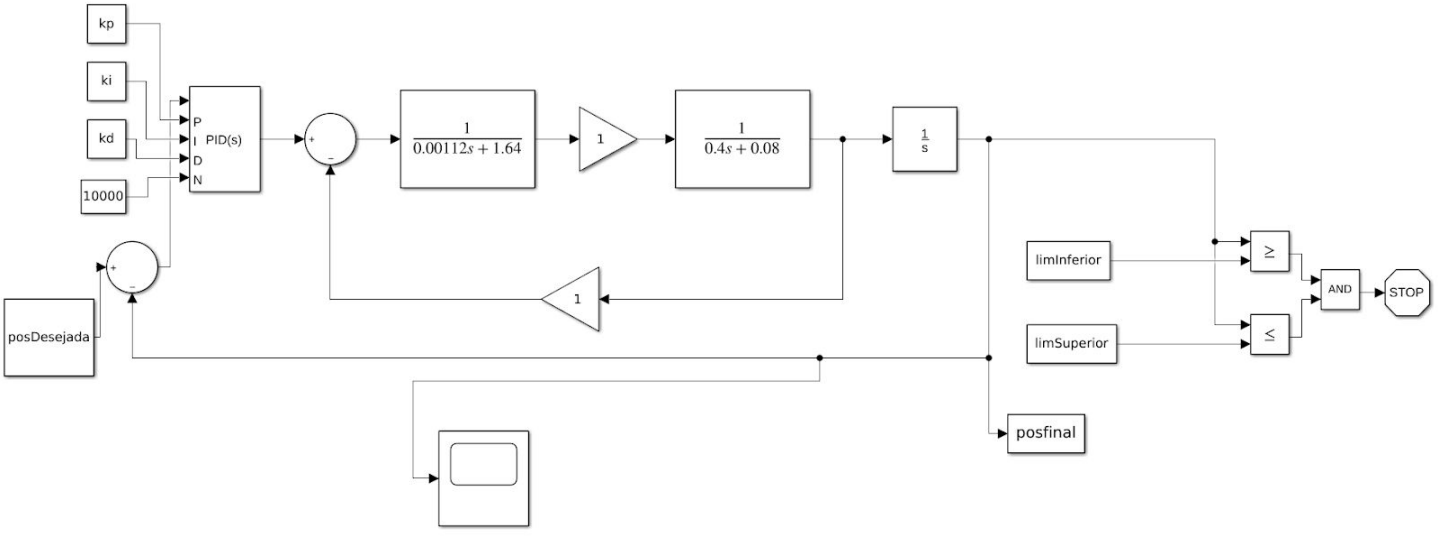

Figura 9: Diagrama de blocos da simulação da técnica de controle PID.

A simulação foi toda construída por meio de diagrama de blocos em Simulink, exposto na Figura 9. Os valores kp, ki, kd, limInferior (limite de erro inferior) e limSuperior (limite de erro superior) são definidos na chamada da simulação.

A função de transferência de um motor de corrente DC está representada no diagrama de blocos e envolve as variáveis: a resistência interna do motor (R), a indutância interna do motor (L), o momento de inércia do motor e da carga (J), e o atrito viscoso do motor e da carga (B).

Esses valores foram definidos de acordo com um motor real da fabricante Maxon, cujo R = 1.64 Ω e L = 1.12 mH.

Figura 10: Motor Maxon usado para obter as variáveis da função de transferência.

5. Resultados Obtidos

Os resultados a seguir expostos são referentes à execução do AG de acordo com os parâmetros detalhados nas seções anteriores. Eles serão divididos em 2 partes: Valores Intermediários e Valores de Saída.

Valores Intermediários			
Kp	**Ki**	**Kd**	**Tempo de Resposta**
87.4715	3.949	1.37173	0.0116
59.7574	2.31912	1.24185	0.0198
1.57936	3.85556	1.08629	0.6952
64.5367	2.43109	1.41749	0.0158
39.2399	2.33858	1.03754	0.0375
21.9679	4.89728	1.38978	0.0514
8.067	4.37681	1.44496	0.1505
51.8798	4.71915	1.37425	0.0206
Valores de Saída			
Kp	**Ki**	**Kd**	**Tempo de Resposta**
24.639	4.3299	1.17766	0.0099

Tabela 2: Resultados obtidos para uma execução do AG.

Um exemplo da tela com a saída do AG é apresentado na Figura 11.

```
$ export PATH=/usr/local/MATLAB/R2018b/bin/:$PATH; g++ *.cpp -o projeto.out; ./projeto.out
Cromossomo Inicial 0
Kp = 38.1707 | Ki = 3.01033 | Kd = 1.33988 | Fitness Value = 0.0378
Cromossomo Inicial 1
Kp = 32.0339 | Ki = 1.5906 | Kd = 1.3949 | Fitness Value = 0.0345
Cromossomo Inicial 2
Kp = 43.1092 | Ki = 2.82598 | Kd = 1.47957 | Fitness Value = 0.0232
Cromossomo Inicial 3
Kp = 23.362 | Ki = 4.99203 | Kd = 1.00067 | Fitness Value = 0.0665
Cromossomo Inicial 4
Kp = 3.19273 | Ki = 4.72423 | Kd = 1.07899 | Fitness Value = 0.4142
Cromossomo Inicial 5
Kp = 10.1041 | Ki = 2.63257 | Kd = 1.15817 | Fitness Value = 0.1464
Cromossomo Inicial 6
Kp = 59.9311 | Ki = 2.52597 | Kd = 1.26321 | Fitness Value = 0.0194
Cromossomo Inicial 7
Kp = 65.5382 | Ki = 1.52242 | Kd = 1.20001 | Fitness Value = 0.0185
```

Figura 11: Saída na tela para uma execução do AG.

Plotando o gráfico no osciloscópio dados os valores ótimos das constantes obtidos pelo AG:

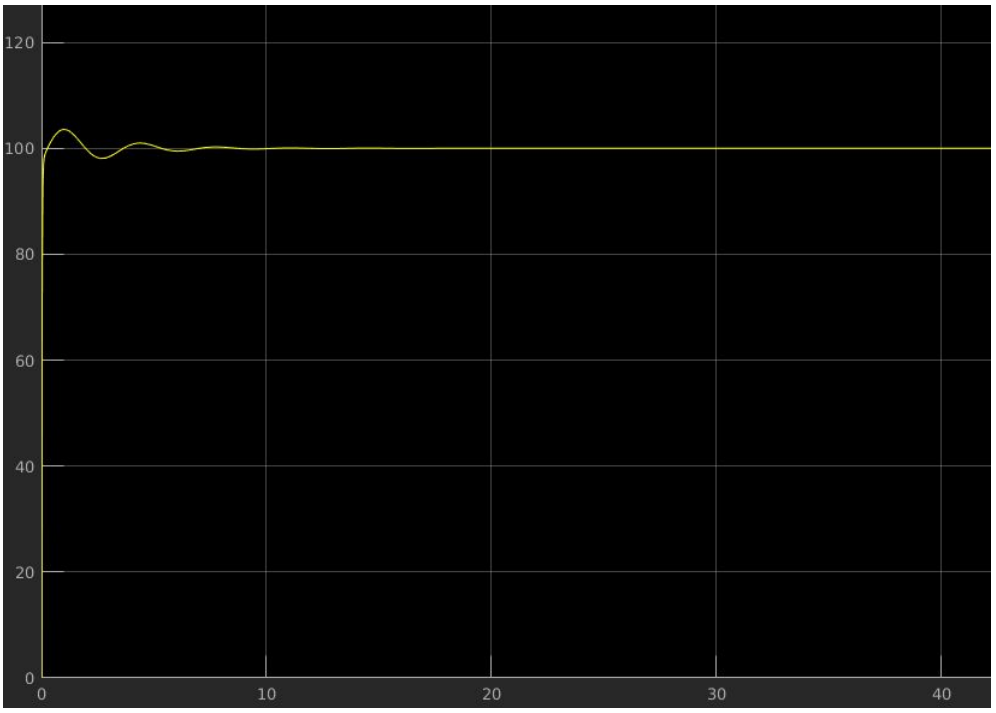

Figura 12: Saída no osciloscópio com os valores de saída obtidos.

Plotando o gráfico no osciloscópio dados os valores intermediários com maior tempo de resposta (Kp = 1.57936, Ki = 3.85556, Kd = 1.08629) obtidos:

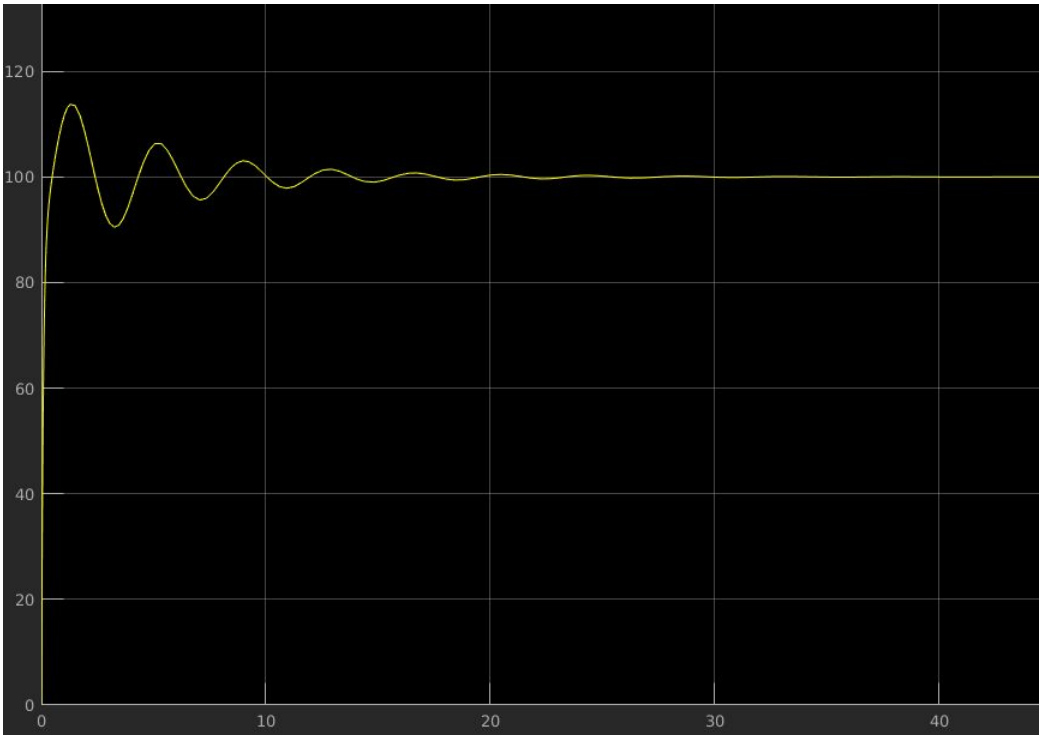

Figura 13: Saída no osciloscópio para os valores de saída obtidos.

6. Conclusão

É possível concluir que, a partir de um Algoritmo Genético como mecanismo de otimização para os valores de Kp, Ki e Kd de uma estratégia de controle PID, é possível obter uma minimização do tempo de resposta. Não só isso, pela comparação das Figuras 12 e 13, é possível perceber que essa otimização também influencia na porcentagem de overshooting e erro em regime. Sendo assim, é um mecanismo recomendado para utilização em sistemas de controle na indústria.

O tempo de execução do AG com simulação ainda é alto. No entanto, acredita-se que isso se deve à interface de comunicação C++/Simulink. Ainda que seja referente à implementação do AG propriamente dita, não representa impacto para o robô, dado que o AG só será executado antes da entrega do produto ao cliente e em manutenções periódicas. Ou seja, não ocorre enquanto o robô está em atividade.

Por se tratar de um algoritmo com vários parâmetros e condições de execução, é mais uma vez ressaltada a importância de sua operação apenas pelo desenvolvedor do robô e não por outro usuários.

O AG é um mecanismo que também permite que os desgastes do robô com o tempo de uso possam ser de certa forma superados, pois a otimização vai sempre agir em cima de como está o robô naquele momento, e os valores ótimos das constantes podem mudar.

Como desenvolvimento futuro, uma análise que seria interessante de ser feita é a plotagem do gráfico de convergência do AG. Ela permitiria entender se os parâmetros de entrada do AG estão coerentes ou se alguns outros ajustes são necessários. Ainda, poderíamos estender a aplicação para as técnicas de controle PID Adaptativo e Fuzzy. Para a primeira, o domínio do problema não mudaria (isto é, a configuração dos genes e cromossomos), deste modo, o código permite essa adaptação de maneira simples e direta.

Referências

[1] *Revisiting the Ziegler–Nichols step response method for PID control*. (2004) K.J.Astrom, T.Hagglund

[2] *Comparison among some well known control schemes with different tuning methods*. (2015). R.Kumar, S.K.Singla, V.Chopra

[3] *Genetic Algorithms*. (2011). Anke Meyer-Baese, Volker Schmid

[4] *PID Parameters Optimization by Using Genetic Algorithm*. (2012). Andri Mirzal, Shinichiro Yoshii, Masashi Furukawa

[5] *Genetic Algorithm Based PID Controller Tuning Approach for Continuous Stirred Tank Reactor*. (2014). A. Jayachitra, R. Vinodha