UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
CURSO DE ENGENHARIA MECATRÔNICA

GABRIEL SARTORI MAÇANEIRO

DETECÇÃO DE SINGULARIDADES EM MANIPULADORES INDUSTRIAIS SERIAIS UTILIZANDO O ALGORITMO DE EVOLUÇÃO DIFERENCIAL

Joinville
2024

GABRIEL SARTORI MAÇANEIRO

DETECÇÃO DE SINGULARIDADES EM MANIPULADORES INDUSTRIAIS SERIAIS UTILIZANDO O ALGORITMO DE EVOLUÇÃO DIFERENCIAL

Trabalho apresentado como requisito parcial para obtenção do título de bacharel em Engenharia Mecatrônica, no Curso de Engenharia Mecatrônica, do Centro Tecnológico de Joinville, da Universidade Federal de Santa Catarina.

Orientador: Dr. Lucas Weihmann

Joinville
2024

Dedico este trabalho a minha família,
cujo amor e carinho me dão forças para seguir em busca dos meus sonhos.

AGRADECIMENTOS

Agradeço a Deus por me guiar ao longo dessa jornada, sendo a luz em meu caminho nos momentos mais difíceis. Agradeço pela saúde e pelo cuidado diário que sempre me sustentaram.

Aos meus pais e ao meu irmão, pelo apoio incondicional nos momentos mais cruciais da minha vida. Agradeço por todo o esforço e dedicação que tornaram possível a realização deste trabalho.

Agradeço ao meu orientador, Lucas Weihmann, pelo suporte e pela amizade durante toda a realização desse trabalho. Agradeço pela presença constante e pela dedicação ao ensino e à orientação.

Aos amigos e colegas de classe, com quem compartilhei momentos inesquecíveis ao longo do curso, que tornaram essa jornada mais leve e prazerosa.

Agradeço à Voltor, empresa que me proporcionou ensinamentos valiosos para a vida toda e forneceu os dados e materiais essenciais para a realização deste trabalho.

A todos que, de alguma forma, contribuíram para a concretização deste trabalho, deixo aqui minha gratidão.

A jornada de mil milhas começa com um único passo (Lao Tzu).

RESUMO

A automação industrial é fundamental para a eficiência e sustentabilidade dos processos produtivos. Nas categorias de automação flexível e programável, os manipuladores industriais são amplamente utilizados devido à sua precisão e repetibilidade, características essenciais para garantir a segurança e a qualidade na produção. Para esse propósito, a detecção de singularidades em manipuladores é crucial, pois essas configurações críticas podem representar riscos tanto para o ambiente quanto para as pessoas, caso não sejam devidamente evitadas. Nesse contexto, propõe-se neste trabalho um algoritmo genérico capaz de identificar configurações singulares e reconhecer padrões de juntas em manipuladores seriais industriais. Além disso, são realizadas simulações e comparações com outros estudos para validar esse algoritmo. Verificou-se que o algoritmo construído foi capaz de encontrar os mínimos globais e os padrões de juntas corretamente em todos os manipuladores avaliados.

Palavras-chave: robótica; evolução diferencial; detecção de singularidades.

ABSTRACT

Industrial automation is essential for the efficiency and sustainability of production processes. In flexible and programmable automation categories, industrial manipulators are widely used due to their precision and repeatability, essential characteristics for ensuring safety and quality in production. For this purpose, detecting singularities in manipulators is crucial, as these critical configurations can pose risks to both the environment and people if not properly avoided. In this context, this work proposes a generic algorithm capable of identifying singular configurations and recognizing joint patterns in serial industrial manipulators. Additionally, simulations and comparisons with other studies are conducted to validate this algorithm. It was verified that the developed algorithm was able to find the global minima and correctly identify the joint patterns in all evaluated manipulators.

Keywords: robotics; differential evolution; singularity analysis.

LISTA DE FIGURAS

Figura 1 – Punho esférico 16
Figura 2 – Mecanismo serial e paralelo 17
Figura 3 – Espaço de trabalho de um braço antropomórfico 17
Figura 4 – Posição e orientação de um corpo rígido 19
Figura 5 – Ilustração dos parâmetros de Denavit-Hartenberg 20
Figura 6 – Comportamento de uma submatriz do jacobiano de um manipulador antropomórfico 23
Figura 7 – Singularidade de limite em um robô planar de duas juntas 24
Figura 8 – Diagrama de rede dos algoritmos meta-heurísticos 26
Figura 9 – Função multimodal 27
Figura 10 – Fase de inicialização 29
Figura 11 – Mutação 30
Figura 12 – Estrutura de coordenadas dos manipuladores seriais 31
Figura 13 – Estrutura de coordenadas do UR5 32
Figura 14 – Máquina de estados finita 33
Figura 15 – Estrutura cinemática do AER-1 34
Figura 16 – Progressão do algoritmo número *Branch-and-prune* 35
Figura 17 – Fluxograma do algoritmo de detecção de singularidades 37
Figura 18 – Estrutura do robô planar de três juntas 39
Figura 19 – Estrutura do manipulador antropomórfico 40
Figura 20 – Manipulador Snake 42
Figura 21 – Estrutura do Snake 42
Figura 22 – Fluxograma da função de verificação de padrões 44
Figura 23 – Estrutura da lista de singularidades 45
Figura 24 – Visualização do robô planar 48
Figura 25 – Modelo IRB 4600 CoppeliaSIM 49
Figura 26 – Visualização do Snake 49
Figura 27 – Evolução do melhor indivíduo para o robô planar 50
Figura 28 – Padrões de juntas encontrados para o robô planar 52
Figura 29 – Postura singular para $\theta_2 = 0$ 53
Figura 30 – Posturas singulares para $\theta_2 = \pm\pi$ 53
Figura 31 – Evolução do melhor indivíduo para o manipulador antropomórfico . 54
Figura 32 – Padrões de juntas encontrados para o manipulador antropomórfico . 54
Figura 33 – Posições cartesianas das juntas para 30 ciclos 56
Figura 34 – Postura singular de punho com $\theta_5 = 0$ 57

Figura 35 – Postura singular de cotovelo com $\theta_3 = -1.4295$ 57
Figura 36 – Singularidade de ombro $\theta_2 = 0.5635rad$ e $\theta_3 = -2.6509rad$ 57
Figura 37 – Evolução do melhor indivíduo para o manipulador Snake 58
Figura 38 – Padrões de juntas encontrados para o manipulador Snake 60
Figura 39 – Postura singular com $\theta_2 = \pi/2$ 60
Figura 40 – Postura singular com $\theta_3 = 0$ 60
Figura 41 – Postura singular com $\theta_5 = 0$ 61

LISTA DE TABELAS

Tabela 1 – Parâmetros de Denavit-Hartenberg do robô planar 38
Tabela 2 – Tamanho dos elos do robô planar 39
Tabela 3 – Limites das juntas do robô planar 39
Tabela 4 – Parâmetros de Denavit-Hartenberg do manipulador antropomórfico 40
Tabela 5 – Tamanho dos elos do manipulador antropomórfico 41
Tabela 6 – Limites das juntas do manipulador antropomórfico 41
Tabela 7 – Parâmetros de Denavit-Hartenberg do Snake 43
Tabela 8 – Tamanho dos elos do Snake 43
Tabela 9 – Limites das juntas do Snake 43
Tabela 10 – Posturas singulares e resultado da evolução diferencial do robô planar 51
Tabela 11 – Posturas singulares e resultado da evolução diferencial do manipulador antropomórfico 55
Tabela 12 – Posturas singulares e resultado da evolução diferencial do manipulador Snake 59

SUMÁRIO

1 INTRODUÇÃO 12
1.1 OBJETIVOS 14
1.1.1 Objetivo Geral 14
1.1.2 Objetivos Específicos 14

2 FUNDAMENTAÇÃO TEÓRICA 15
2.1 MANIPULADORES INDUSTRIAIS 15
2.1.1 Estrutura de um robô industrial 15
2.1.2 Tipos de robôs industriais 16
2.1.3 Espaço de trabalho 17
2.1.4 Graus de liberdade 18
2.1.5 Cinemática 18
2.1.5.1 Postura de um corpo rígido 18
2.1.5.2 Transformação homogênea 19
2.1.5.3 Parâmetros de Denavit-Hartenberg 20
2.1.5.4 Cinemática direta 21
2.1.5.5 Jacobiano 21
2.1.6 Singularidades 23
2.1.6.1 Desacoplamento de singularidades 24
2.2 EVOLUÇÃO DIFERENCIAL 25
2.2.1 Otimização local e global 26
2.2.2 Estrutura da população 27
2.2.3 Inicialização 28
2.2.4 Mutação 28
2.2.5 Crossover 29
2.2.6 Seleção 30
2.3 ESTADO DA ARTE 31

3 MÉTODO 36
3.1 Etapa Inicial 36
3.1.1 Variáveis utilizadas 36
3.1.1.1 Robô planar de três juntas 38
3.1.1.2 Manipulador antropomórfico 40
3.1.1.3 Manipulador Snake 41
3.1.2 Função de verificação de padrões 43
3.1.3 Função callback 45

3.1.4 Função objetivo 46
3.2 Ciclo principal 46
3.3 Visualização das posturas 48

4 RESULTADOS 50
4.1 Robô planar de três juntas 50
4.2 Manipulador antropomórfico 52
4.3 Manipulador Snake 58

5 CONCLUSÕES 62
5.1 Trabalhos futuros 63

REFERÊNCIAS 65

1 INTRODUÇÃO

A indústria, sustentáculo da sociedade tecnológica moderna, passou por inúmeras inovações, entre as quais a mais impactante foi a automação. A automação fornece segurança e confiabilidade nos processos produtivos, utilização eficiente dos recursos, consequentemente, reduzindo emissões e aumentando a sustentabilidade e qualidade dos produtos. Por esses motivos, teve sua importância intensificada ao longo dos anos (Jämsä-Jounela, 2007).

A automação pode ser separada em três categorias. Na rígida, as máquinas são projetadas para um processo ou produção de um produto específico. Nesse caso, é empregada em busca de volume, padronização e alta produtividade. Nas categorias programável e flexível, visa-se a fabricação de pequenos ou médios lotes em série e a fabricação de lotes variáveis, respectivamente (Romano, 2002).

Nas categorias programável e flexível de automação, tem-se máquinas multifuncionais e reprogramáveis, sendo amplamente utilizados os robôs industriais (Romano, 2002). Para que esses robôs possam realizar as tarefas designadas e gerar trajetórias aceitáveis no efetuador final, é necessário um estudo da posição e velocidade desse efetuador e das juntas do robô, denominado cinemática (Hayat; Sadanand; Saha, 2015).

A cinemática de um robô pode ser dividida em duas etapas: cinemática direta e inversa. A direta envolve a determinação da posição e orientação do efetuador final com base nas posições das juntas do robô. A cinemática inversa consiste em encontrar a configuração das juntas necessária para posicionar o efetuador final em uma posição e orientação desejadas (Siciliano *et al.*, 2010).

Em sistemas robóticos de cadeia aberta, a cinemática inversa costuma ser mais complexa do que a direta, o que se deve ao fato de que, para uma posição e orientação específicas desejadas no efetuador final, pode existir mais de um conjunto de configurações das juntas que resolvam o problema, ou até mesmo nenhuma solução (Lynch; Park, 2017). Além disso, a presença de soluções infinitas para a cinemática inversa, uma consequência das singularidades cinemáticas, pode complicar ainda mais a resolução do problema (Siciliano *et al.*, 2010).

A detecção das singularidades em manipuladores é essencial pois são configurações em que se pode perder o controle do movimento do efetuador final em algumas direções, comprometendo seu desempenho. Além disso, variações de velocidade, força e torque limitados no efetuador final podem ocasionar variações infinitas desses parâmetros nas juntas, resultando em riscos de segurança. Os pontos de singularidade são, portanto, um problema para o algoritmo controlador desses robôs

(Spong; Hutchinson; Vidyasagar, 2005).

Vaezi *et al.* (2011) propuseram a detecção de singularidades em um Stäubli© TX40 decompondo o problema da cinemática inversa em dois - posição e orientação - e identificando os três tipos de singularidades possíveis para esse robô: limite do antebraço, interior do antebraço e singularidade do punho. Essa classificação de singularidades adotada pode ser generalizada para a maioria dos manipuladores seriais.

Com o intuito de evitar esses pontos e garantir um planejamento de trajetórias robusto, é possível formular essa dificuldade como um problema de otimização. Cada aplicação, dependendo de sua complexidade, pode exigir a adoção de um método de otimização diferente para se obter bons resultados (Rebouças Filho *et al.*, 2018). No campo da robótica, entre os métodos de otimização que podem ser utilizados, estão os algoritmos evolucionários.

Os algoritmos evolucionários são algoritmos de busca baseados no conceito de evolução natural. Representam um modelo de aprendizado coletivo dos indivíduos de uma população ao longo das gerações, que passam por processos aleatórios de recombinação, mutação e seleção para encontrar uma solução para o problema (Bäck, 1996). A evolução diferencial é uma subclasse dos algoritmos evolucionários muito utilizada atualmente e possui a vantagem de incorporar um processo aleatório já nas fases de recombinação e mutação sem a necessidade de uma distribuição de probabilidade em separado (Das; Suganthan, 2011).

Lora *et al.* (2018) trataram o planejamento de trajetórias como um problema de otimização, minimizando a função que representava o erro de trajetórias alternativas nos pontos de singularidade. Esse desenvolvimento foi feito para um manipulador planar com três juntas de revolução, utilizando quatro métodos distintos para comparar seus resultados, entre os quais estava o algoritmo de evolução diferencial. González, Blanco e Moreno (2009) também utilizaram a evolução diferencial para encontrar as trajetórias ótimas nas tarefas de um manipulador PUMA.

Lin *et al.* (2008) realizaram a caracterização de posturas singulares e a busca por trajetórias ótimas utilizando algoritmos genéticos, outra subclasse dos algoritmos evolucionários, em um manipulador paralelo de três juntas e seis graus de liberdade. Dessa maneira, embora existam diversos trabalhos com a aplicação da evolução diferencial na robótica, o desenvolvimento desses costuma ser feito com foco na otimização de trajetórias. Portanto, nesse trabalho, objetiva-se desenvolver um algoritmo especificamente para detecção de singularidades, pois não foi encontrado nenhum estudo na literatura que utilize a abordagem que será proposta.

A metodologia empregada consiste em identificar as configurações singulares a partir do cálculo do valor absoluto do determinante da matriz jacobiana, que relaciona as velocidades linear e angular do efetuador final do manipulador com as velocidades

das juntas (Siciliano *et al.*, 2010). Para alcançar esse objetivo, a hipótese de solução é tratar esse determinante como um problema de otimização. Assim, utilizando-se o algoritmo de evolução diferencial, deseja-se minimizar essa função para caracterizar posturas singulares.

Com base na análise dos resultados e na comparação com estudos de outros autores, verificou-se que o algoritmo de detecção de singularidades apresentou resultados coerentes. Exibe-se aqui o histórico dos valores absolutos do determinante do jacobiano ao longo das gerações, os valores das juntas do manipulador e os resultados da otimização pela evolução diferencial para fins de análise. Figuras representativas dos manipuladores em posturas singulares também são incluídas para visualização.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Desenvolver um algoritmo para detecção de singularidades de robôs industriais seriais utilizando o método de evolução diferencial.

1.1.2 Objetivos Específicos

- Implementar os modelos cinemáticos dos manipuladores industriais utilizando a abordagem de Denavit-Hartenberg;
- Desenvolver e implementar algoritmos baseados em evolução diferencial para detecção das singularidades no espaço de configuração dos manipuladores;
- Realizar simulações computacionais para validar a metodologia de detecção de singularidades;
- Comparar os resultados obtidos com outros estudos relevantes para validar a eficácia da metodologia proposta;
- Analisar os resultados das simulações e comparações, identificando as configurações singulares e avaliando as condições específicas que levam à singularidade.

2 FUNDAMENTAÇÃO TEÓRICA

A fim de demonstrar a base necessária ao desenvolvimento de um algoritmo capaz de detectar singularidades, este capítulo é composto por três partes: manipuladores industriais, evolução diferencial e o estado da arte. Na primeira parte, são apresentadas as definições fundamentais dos robôs industriais, como representá-los a partir de um modelo matemático, calcular sua cinemática direta, calcular sua matriz jacobiana e utilizá-la para detectar os pontos de singularidade. Na segunda parte, são detalhados os principais conceitos da evolução diferencial e como esse método pode ser empregado para resolver problemas de minimização. Por fim, na terceira parte, são apresentados estudos recentes com o intuito de apresentar avanços na área de detecção de singularidades.

2.1 MANIPULADORES INDUSTRIAIS

A Associação Brasileira de Normas Técnicas (ABNT), através da norma ABNT NBR ISO 10218-1:2018, que visa segurança especificamente no projeto e construção do robô, caracterizou um robô industrial como um "[...] manipulador multifuncional reprogramável, automaticamente controlado, programável em três ou mais eixos que pode ser fixo no local ou móvel para uso em aplicações de automação industrial [...]". (p. 3).

2.1.1 Estrutura de um robô industrial

Usualmente, os robôs são compostos por elos e juntas, formando um mecanismo articulado onde cada junta conecta exatamente dois elos (Lynch; Park, 2017). O elo é definido como um corpo rígido capaz de ser agregado a outros elos através de nós, que são os pontos de conexão entre elos. As juntas são a conexão entre os elos (nos seus nós) que permitem movimento relativo dos elos entre si (Norton, 2010).

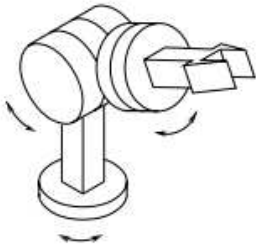
Tipicamente, mecanismos robóticos possuem dois tipos de juntas para realizar movimento: prismáticas e de revolução. As juntas prismáticas permitem movimento de translação relativo entre dois elos, enquanto as juntas de revolução permitem movimento rotacional relativo entre os elos. Em um robô serial, cada junta adicionada proporciona um grau de liberdade. Já em robôs paralelos isso não é necessariamente verdade devido às restrições impostas pela cadeia fechada (Siciliano *et al.*, 2010).

Um robô é classificado como manipulador, caso apresente a base fixa. Um manipulador é composto por um braço, um punho e um efetuador final. O braço permite

alcance e mobilidade e é responsável pelo posicionamento do punho (Siciliano *et al.*, 2010). O último elo deve ser equipado com um dispositivo adequado de acordo com a tarefa, denominado efetuador final. Comumente, são utilizadas garras robóticas que podem, por exemplo, realizar o movimento de pinça ou segurar e movimentar objetos através de sucção (Karabegović; Banjanović-Mehmedović, 2020).

O punho do robô serial é responsável por estabelecer a orientação desejada para realizar a tarefa e, normalmente, é composto por duas ou três juntas de revolução (Groover, 2015). Caso os eixos dessas juntas se interceptem em um único ponto, como na Figura 1, o punho é denominado esférico, o que facilita a coordenação entre braço e punho e desacopla posição e orientação do manipulador (Siciliano *et al.*, 2010).

Figura 1 – Punho esférico



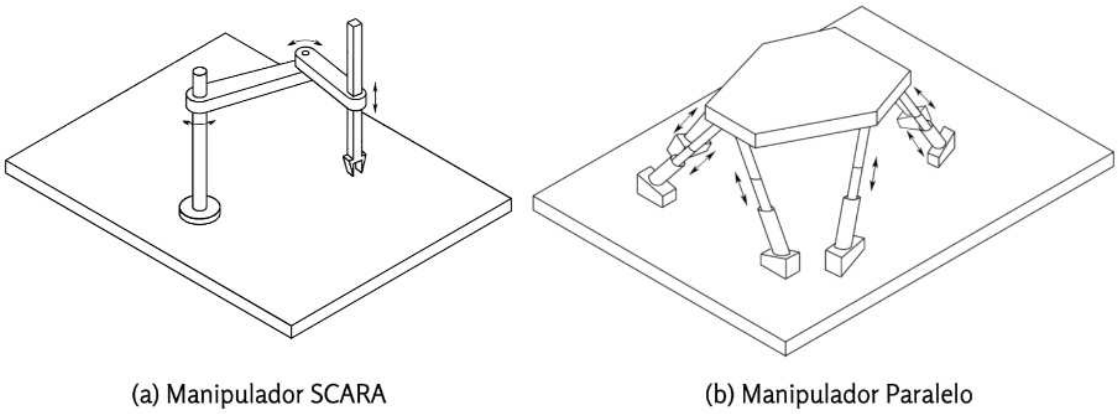
Fonte: Siciliano *et al.* (2010, p. 10).

A partir dos tipos de juntas e da disposição entre as mesmas, existem diversas combinações possíveis de configuração de um manipulador, as quais afetam diretamente as capacidades e as tarefas que o robô pode realizar (Groover, 2015). A estrutura dos robôs industriais serve como base para sua categorização, pois, de acordo com a distribuição dos elos e juntas, podem ser classificados em diferentes tipos.

2.1.2 Tipos de robôs industriais

Utilizando-se como critério a estrutura dos robôs de base fixa, é possível classificá-los em seriais ou paralelos. Robôs seriais (também conhecidos como de cadeia aberta) não apresentam um circuito fechado na composição de seus elos e juntas, como na Figura 2(a). Robôs paralelos (também conhecidos como de cadeia fechada) possuem um circuito fechado em sua composição, como é demonstrado na Figura 2(b), onde ressalta-se que o chão também é considerado um elo (Lynch; Park, 2017). Essa distinção entre os tipos de robôs influencia diretamente o espaço de trabalho que conseguem acessar.

Figura 2 – Mecanismo serial e paralelo

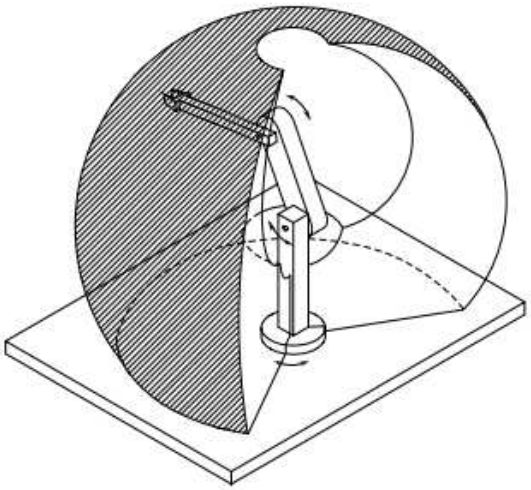


Fonte: Adaptado de Siciliano *et al.* (2010, p. 7 e 8).

2.1.3 Espaço de trabalho

O espaço de trabalho é definido como a região em que o efetuador final pode se movimentar e realizar as tarefas de forma que sua única restrição é causada pela própria estrutura mecânica, ou seja, suas limitações físicas (Karabegović; Banjanović-Mehmedović, 2020). Na Figura 3, observa-se o espaço de trabalho de um braço antropomórfico, o qual possui três juntas de revolução e, com base nos limites das juntas, resulta em um setor esférico como espaço de trabalho (Siciliano *et al.*, 2010).

Figura 3 – Espaço de trabalho de um braço antropomórfico



Fonte: Siciliano *et al.* (2010, p. 7).

Independentemente da tarefa, a estrutura do manipulador determina o espaço de trabalho, que deve ser alcançado por pelo menos uma configuração das juntas do

robô (Lynch; Park, 2017). A forma do espaço de trabalho está diretamente relacionada aos graus de liberdade do robô, pois estes definem o número de movimentos independentes que pode realizar.

2.1.4 Graus de liberdade

Norton (2010) define os graus de liberdade, também chamados de mobilidade, como a quantidade de coordenadas independentes necessárias para determinar a posição de um mecanismo ou, em outras palavras, para estabelecer uma saída previsível. Tratando-se dos graus de liberdade de juntas, é possível afirmar que juntas prismáticas ou de revolução adicionam um grau de liberdade entre dois elos ou adicionam cinco limitações no movimento dos elos, pois um corpo rígido no espaço tem, naturalmente, seis graus de liberdade, três de posição e três de orientação (Lynch; Park, 2017).

2.1.5 Cinemática

Para que um manipulador realize as tarefas designadas, é necessário relacionar o movimento de suas juntas com a posição e a orientação do efetuador final. Obter essa relação é o objetivo da cinemática, que pode ser classificada em cinemática de posição e cinemática diferencial (Karabegović; Banjanović-Mehmedović, 2020). A cinemática de posição, que abrange a cinemática direta e inversa, relaciona a postura do efetuador final com as variáveis de junta, conforme a seção 2.1.5.4 (Siciliano *et al.*, 2010).

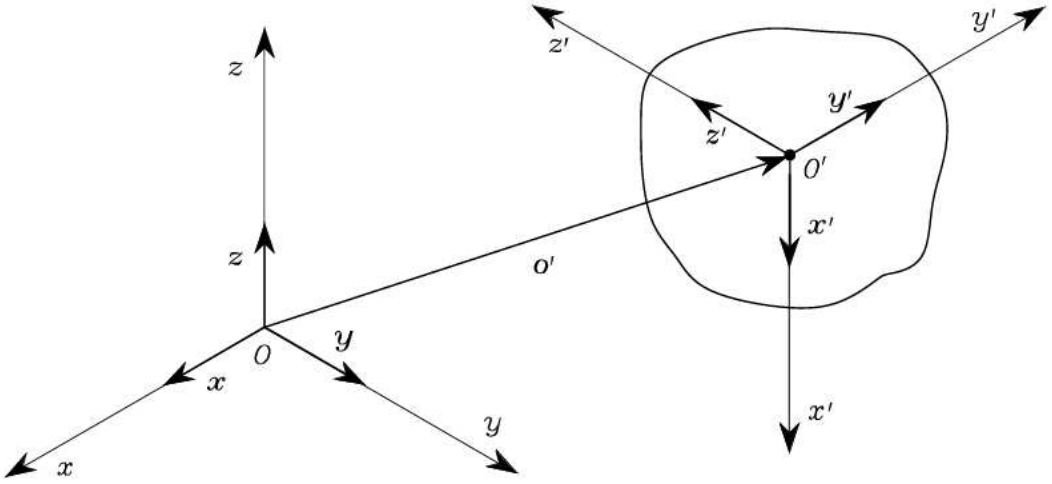
Na cinemática diferencial, relacionam-se as velocidades lineares e angulares do efetuador final e as velocidades das juntas, conforme seção 2.1.5.5 (Siciliano *et al.*, 2010). Nesta seção são abordados a postura de um corpo rígido, como realizar transformações entre os sistemas de coordenadas das juntas dos robôs, o que são e como calcular as matrizes homogêneas e como calcular a cinemática direta e a matriz jacobiana de um manipulador.

2.1.5.1 Postura de um corpo rígido

A partir da posição e da orientação, um corpo rígido é representado no espaço em relação a um sistema de coordenadas de referência. A posição *o'* de um ponto do corpo é representada pela Equação 1, onde ***x***, ***y*** e ***z*** são os vetores unitários do sistema de referência, multiplicados pelos termos o'_x, o'_y e o'_z que são as componentes do vetor ***o'***. Esse vetor é exibido na Figura 4 (Siciliano *et al.*, 2010).

$$\mathbf{o}' = o'_x\mathbf{x} + o'_y\mathbf{y} + o'_z\mathbf{z} \tag{1}$$

Figura 4 – Posição e orientação de um corpo rígido



Fonte: Siciliano *et al.* (2010, p. 40).

Da mesma forma, a orientação é descrita ao fixar um sistema ortonormal em um ponto e referenciá-lo a outro sistema. Conforme ilustrado na Figura 4, $O' - x'y'z'$ é um sistema com origem em O' e vetores unitários ***x'***, ***y'*** e ***z'***. Esses vetores são expressos em relação ao sistema $O - xyz$ por meio da Equação 2, onde suas componentes representam os cossenos diretores dos eixos de $O' - x'y'z'$ em relação a $O - xyz$ (Siciliano *et al.*, 2010).

$$\begin{aligned} \mathbf{x}' &= x'_x\mathbf{x} + x'_y\mathbf{y} + x'_z\mathbf{z} \\ \mathbf{y}' &= y'_x\mathbf{x} + y'_y\mathbf{y} + y'_z\mathbf{z} \\ \mathbf{z}' &= z'_x\mathbf{x} + z'_y\mathbf{y} + z'_z\mathbf{z} \end{aligned} \tag{2}$$

2.1.5.2 Transformação homogênea

A transformação de coordenadas entre sistemas arbitrários pode ser descrita por uma matriz de transformação homogênea. Considerando-se um sistema de coordenadas $O_1 - x_1y_1z_1$, é possível relacioná-lo a outro sistema $O_0 - x_0y_0z_0$ a partir da matriz da Equação 3. Na Equação 3, $\boldsymbol{o}_1^0$ é o vetor que representa a origem e $\boldsymbol{R}_1^0$ a matriz de rotação, ambos relacionam o sistema 1 ao sistema 0 (Siciliano *et al.*, 2010).

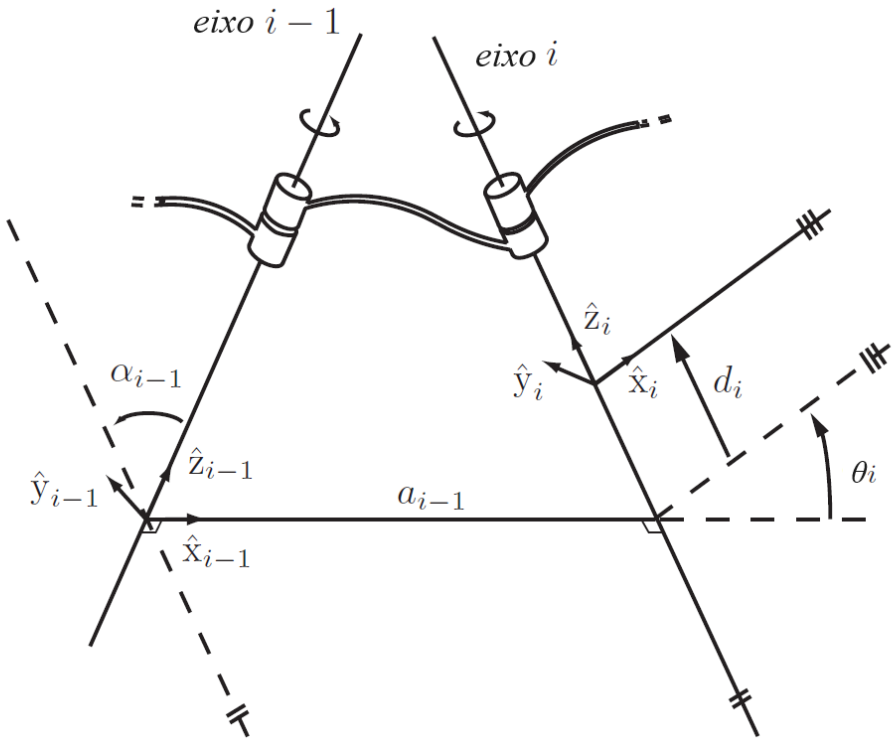
$$\mathbf{A}_1^0 = \begin{bmatrix} \mathbf{R}_1^0 & \mathbf{o}_1^0 \\ \mathbf{0^T} & 1 \end{bmatrix} \tag{3}$$

2.1.5.3 Parâmetros de Denavit-Hartenberg

A convenção de Denavit-Hartenberg propõe uma abordagem procedural de posicionar sistemas de coordenadas nas juntas do robô, permitindo o cálculo das matrizes de transformação homogênea e, consequentemente, a definição da cinemática direta. Com base nessa convenção obtêm-se os sistemas de coordenadas *i* e *i-1* representados na Figura 5. Além disso, fornece os parâmetros α, θ, a e d, que descrevem completamente a cinemática direta de um manipulador serial com juntas prismáticas ou de revolução e são definidos como: (Lynch; Park, 2017).

- a_{i-1}: Distância das linhas mutuamente perpendiculares dos sistemas;
- d_i: Distância da intersecção de $\hat{x}_{i-1}$ e z_i para origem do sistema *i*;
- α_{i-1}: Ângulo de $\hat{z}_{i-1}$ a $\hat{z}_i$ medido em torno de $\hat{x}_{i-1}$;
- θ_i: Ângulo de $\hat{x}_{i-1}$ a $\hat{x}_i$ medido em torno de $\hat{z}_i$;

Figura 5 – Ilustração dos parâmetros de Denavit-Hartenberg



Fonte: Adaptado de Lynch e Park (2017, p. 586).

A abordagem de Denavit-Hartenberg é consolidada na literatura e interessados podem consultar Lynch e Park (2017). Como os parâmetros mencionados são o suficiente para descrever a cinemática direta, após posicionar os sistemas de coordenadas em cada junta do manipulador, aplica-se sucessivas rotações e translações em torno dos eixos definidos com a convenção de Denavit-Hartenberg $\hat{x}$

e $\hat{z}$, como demonstrado na Equação 4. Dessa forma, encontram-se as matrizes de transformação homogênea de cada sistema adjacente a partir da Equação 5 (Siciliano *et al.*, 2010).

$$\mathbf{A_i^{i-1}} = \mathbf{Rot}(\hat{z}, \theta_i)\mathbf{Trans}(\hat{z}, d_i)\mathbf{Trans}(\hat{x}, a_i)\mathbf{Rot}(\hat{x}, \alpha_i) \tag{4}$$

$$\mathbf{A_i^{i-1}} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

2.1.5.4 Cinemática direta

O cálculo da cinemática direta envolve encontrar a postura do efetuador final de um manipulador em função das suas variáveis de juntas q_i. Considerando-se, então, um sistema de referência na base do manipulador $O_b - x_b y_b z_b$, representa-se a postura do efetuador final através da matriz de transformação homogênea da Equação 6. Nessa equação, $\boldsymbol{n_e^b}$, $\boldsymbol{s_e^b}$ e $\boldsymbol{a_e^b}$ simbolizam os vetores unitários do sistema do efetuador final em relação à base e $\boldsymbol{p_e^b}$ a posição da origem do efetuador final também em relação à base (Siciliano *et al.*, 2010).

$$\mathbf{T_e^b(q)} = \begin{bmatrix} \mathbf{n_e^b(q)} & \mathbf{s_e^b(q)} & \mathbf{a_e^b(q)} & \mathbf{p_e^b(q)} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Para um manipulador serial que possui apenas juntas de revolução ou prismáticas, obtém-se a expressão da Equação 6 após sucessivas multiplicações de matrizes homogêneas, onde cada uma representa a relação do sistema de coordenadas de uma junta para outra. Desse modo, para um manipulador com *n* juntas, a Equação 7 descreve a relação do efetuador final à base, onde $\boldsymbol{T_0^b}$ representa a transformação do sistema *0* à base e $\boldsymbol{T_e^n}$ a transformação do efetuador final ao sistema *n* (Siciliano *et al.*, 2010).

$$\mathbf{T_e^b(q)} = \mathbf{T_0^b A_1^0(q_1) A_2^1(q_2)...A_n^{n-1}(q_n) T_e^n} \tag{7}$$

2.1.5.5 Jacobiano

Na cinemática diferencial, a matriz jacobiana é utilizada para relacionar a velocidade das juntas do manipulador com a velocidade linear e angular do efetuador

final. Essa matriz é dependente das variáveis de juntas e, para um manipulador com n juntas, está representada pela Equação 8, onde J_P são as velocidades de juntas $\dot{q}$ que contribuem para a velocidade linear e J_O para a velocidade angular do efetuador final (Siciliano *et al.*, 2010).

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{P_1} & & \mathbf{J}_{P_n} \\ & \dots & \\ \mathbf{J}_{O_1} & & \mathbf{J}_{O_n} \end{bmatrix} \tag{8}$$

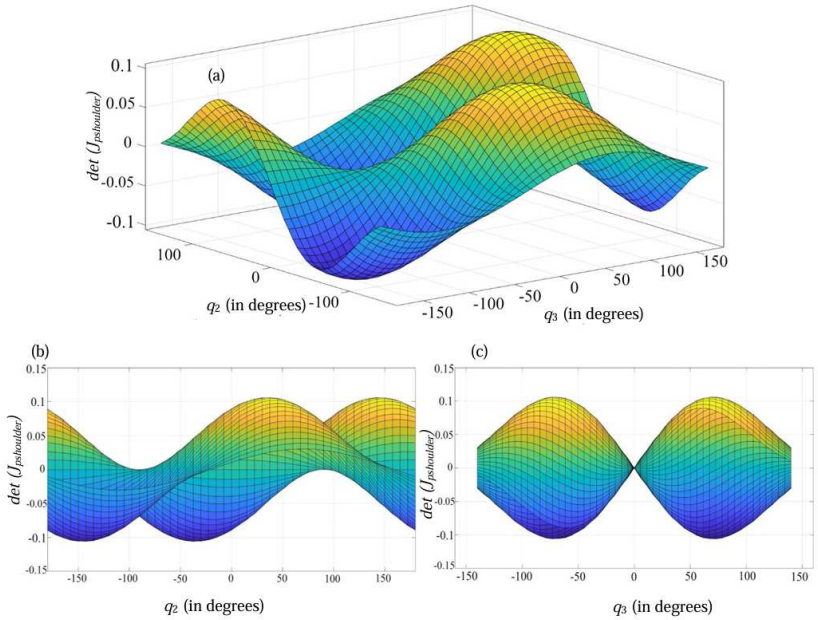
Para a construção da matriz na Equação 8 através do método do jacobiano geométrico, utiliza-se as componentes da Equação 9 que são baseadas na cinemática direta. A componente pe representa a posição do efetuador final em relação à base. As componentes p_{i-1} representam a posição e z_{i-1} a orientação em z de cada uma das juntas em relação à base (Siciliano *et al.*, 2010).

$$\begin{bmatrix} \mathbf{J}_{P_i} \\ \mathbf{J}_{O_i} \end{bmatrix} = \begin{cases} \begin{bmatrix} \mathbf{z}_{i-1} \\ \mathbf{0} \end{bmatrix} & \text{para juntas prismáticas,} \\ \begin{bmatrix} \mathbf{z}_{i-1} \times (\mathbf{p}_e - \mathbf{p}_{i-1}) \\ \mathbf{z}_{i-1} \end{bmatrix} & \text{para juntas de revolução.} \end{cases} \tag{9}$$

O cálculo detalhado de cada um dos parâmetros da Equação 9 encontra-se em Siciliano *et al.* (2010). Esses parâmetros são dependentes das variáveis de junta e das características construtivas do manipulador. Dessa forma, podem ser utilizados para o cálculo da velocidade linear e angular de qualquer junta do robô (Siciliano *et al.*, 2010). A determinação do jacobiano é essencial para a identificação dos pontos de singularidade.

O jacobiano de um manipulador pode se tornar uma função complexa, dependendo de sua estrutura. Pode estar em função de diversas variáveis e possuir múltiplos mínimos globais, como apresentado por Aboelnasr, Baha e Mokhiamar (2021, p.14). Na Figura 6, observa-se o comportamento do determinante de uma submatriz do jacobiano de um manipulador antropomórfico, a qual está relacionada às singularidades de ombro, que serão discutidas na subseção de desacoplamento na próxima seção sobre singularidades.

Figura 6 – Comportamento de uma submatriz do jacobiano de um manipulador antropomórfico



Fonte: Aboelnasr, Baha e Mokhiamar (2021, p. 14).

2.1.6 Singularidades

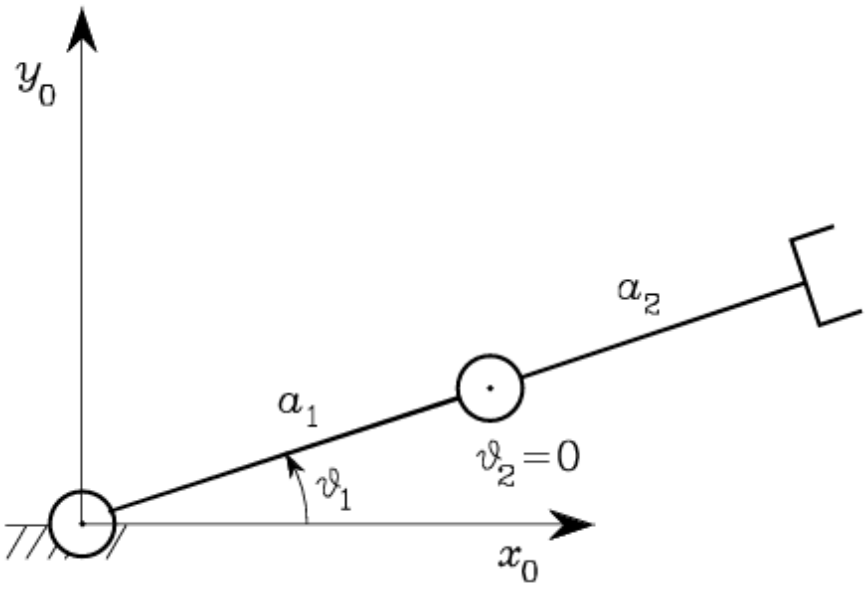
Define-se a singularidade cinemática como uma postura em que a mobilidade do efetuador final, ou seja, a capacidade de gerar velocidades espaciais em uma ou mais direções, é reduzida. Matematicamente, quando o manipulador está em uma posição singular, o posto da matriz jacobiana encontra-se abaixo de seu máximo (Lynch; Park, 2017). O posto de uma matriz é o número de colunas ou linhas independentes e, portanto, estar abaixo de seu máximo significa que ao menos uma coluna ou linha se tornou dependente de outras (Strang, 2009).

Ainda segundo Lynch e Park (2017), essa singularidade é independente do método escolhido para o jacobiano e também da escolha do sistema de coordenadas do efetuador final. No caso do manipulador encontrar-se em uma configuração singular, podem existir infinitas soluções para a cinemática inversa, e, caso esteja próximo dessa configuração, pequenas velocidades no espaço operacional representam grandes velocidades no espaço das juntas (Siciliano *et al.*, 2010). Essas velocidades desestabilizam o sistema de controle, o que pode ocasionar danos físicos ao robô e ao ambiente, além de representar riscos à segurança no local (Azizi; Zadeh, 2014).

É possível dividir as singularidades em duas classes, de limite e internas. As singularidades de limite ocorrem quando o manipulador se encontra totalmente estendido ou retraído, pode-se evitá-las apenas impedindo que trabalhe no limite de seu espaço de trabalho. As singularidades internas ocorrem em configurações específicas ou quando há o alinhamento de dois ou mais eixos dos sistemas de coordenadas.

Essas podem ser encontradas em qualquer ponto no interior do espaço de trabalho (Siciliano *et al.*, 2010). Na Figura 7, observa-se um exemplo de singularidade de limite para um robô planar de duas juntas totalmente estendido.

Figura 7 – Singularidade de limite em um robô planar de duas juntas



Fonte: Siciliano *et al.* (2010, p. 117).

Uma abordagem eficaz para caracterizar posturas singulares em robôs é avaliar o determinante de sua matriz jacobiana: quando esse determinante é zero, o robô encontra-se em uma condição de singularidade. No entanto, em manipuladores de estrutura complexa, essa abordagem pode ser difícil de resolver e exigir alta capacidade computacional. Para contornar essa questão em manipuladores com punho esférico, é possível dividir a análise em duas etapas: calcular as singularidades do braço, representado pelas três primeiras juntas, e as singularidades do punho, composto pelas três juntas do punho esférico (Siciliano *et al.*, 2010).

2.1.6.1 Desacoplamento de singularidades

O método do desacoplamento de singularidades consiste em subdividir a matriz jacobiana dos manipuladores conforme exemplificado na Equação 10, para um manipulador de seis juntas. Como as singularidades são inerentes à estrutura do manipulador e não dependem da escolha dos sistemas de coordenadas, é conveniente posicionar o efetuador final no centro do punho esférico, o que torna a matriz J_{12} nula e transforma o jacobiano em uma matriz triangular inferior (Siciliano *et al.*, 2010).

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{11} & \mathbf{J}_{12} \\ \mathbf{J}_{21} & \mathbf{J}_{22} \end{bmatrix} \tag{10}$$

Uma vez que o jacobiano é transformado em uma matriz triangular inferior, o cálculo do determinante é simplificado, conforme ilustrado na Equação 11. A primeira condição, $det(J_{11}) = 0$, representa as singularidades do braço, enquanto a segunda condição, $det(J_{22}) = 0$, representa as singularidades do punho. Vale ressaltar que essa abordagem não pode ser utilizada para relacionar as velocidades das juntas com a velocidade do efetuador final, como ocorre no jacobiano tradicional (Siciliano *et al.*, 2010).

$$det(\mathbf{J}) = det(\mathbf{J_{11}}) \cdot det(\mathbf{J_{22}}) \tag{11}$$

Um exemplo de desacoplamento ocorre para um manipulador antropomórfico apresentado por Siciliano *et al.* (2010). Nesse exemplo, são explicitadas as condições para a singularidade de punho, enquanto as singularidades do braço são subdivididas em duas condições: singularidade de cotovelo e de ombro, que são aplicadas na seção de resultados deste trabalho. A propriedade do determinante da matriz jacobiana fundamenta a hipótese deste trabalho, que propõe tratar a identificação de singularidades como um problema de otimização, possibilitando o uso do método de evolução diferencial para sua resolução.

2.2 EVOLUÇÃO DIFERENCIAL

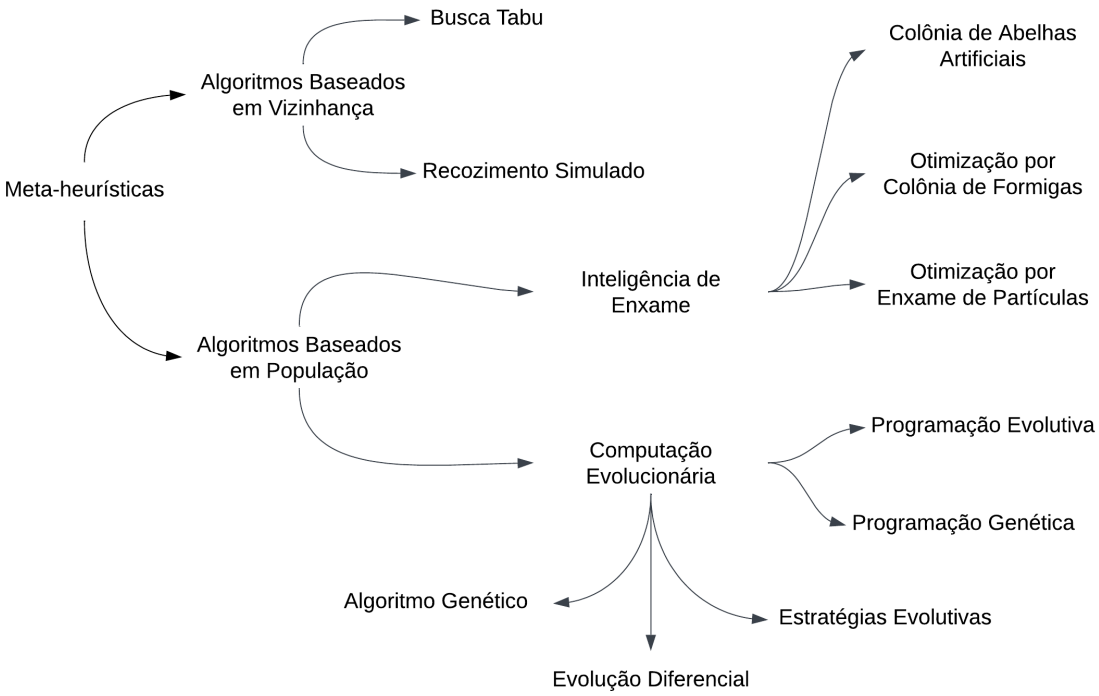
Denominam-se problemas de otimização aqueles em que se deseja maximizar determinadas propriedades enquanto se minimizam outras propriedades indesejadas. Essas propriedades dependem do sistema, que pode ser modelado através de uma função. Se essa função representa o objetivo de otimização, é chamada de função objetivo. Alternativamente, pode ser denominada função custo, caso o objetivo de otimização seja minimizar o valor que pode assumir em um determinado intervalo (Price; Storn; Lampinen, 2014).

Existem muitas aplicações na engenharia em que não é possível definir claramente a natureza de um problema de otimização. Nesses casos, encontrar o método apropriado para resolver o problema torna-se um desafio. Com o intuito de solucionar esse desafio, diversos métodos genéricos foram desenvolvidos, chamados meta-heurísticos. Esses métodos focam na avaliação de determinadas funções e não dependem de parâmetros ou restrições específicas do problema (Bilal *et al.*, 2020).

Algoritmos meta-heurísticos podem ser classificados em dois subconjuntos de acordo com suas respectivas características. O primeiro subconjunto, os algoritmos

baseados em vizinhança, é utilizado para buscas locais, pois utiliza os resultados de seus vizinhos para atingir o objetivo. O segundo subconjunto é composto por algoritmos baseados em população, que empregam o conceito de gerações populacionais para alcançar o objetivo. Um diagrama de rede representando essas classificações pode ser visto na Figura 8 (Bilal *et al.*, 2020).

Figura 8 – Diagrama de rede dos algoritmos meta-heurísticos



Fonte: Adaptado de Bilal *et al.* (2020, p. 2).

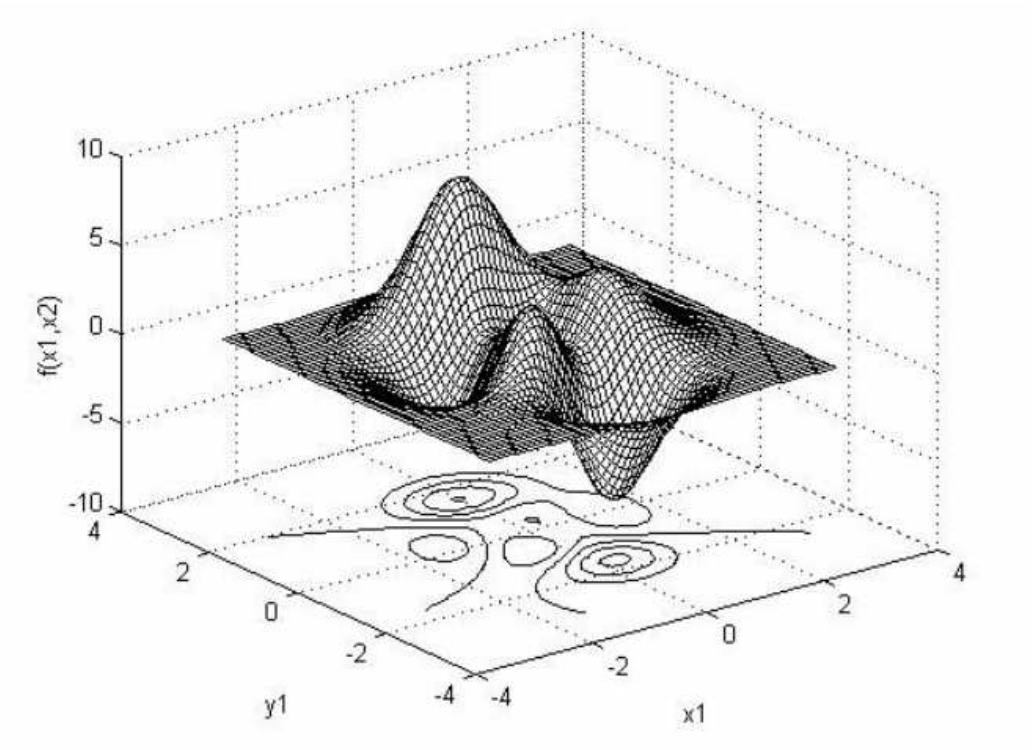
A evolução diferencial é uma subclasse da computação evolucionária, também conhecida como algoritmos evolutivos, baseada no conceito da evolução natural das espécies (Bilal *et al.*, 2020). Esse método se destaca pela simplicidade e robustez nas variáveis de controle, pela capacidade de lidar com alto custo computacional, pela convergência consistente e pela capacidade de lidar com funções objetivo não diferenciáveis, não lineares e, especialmente, aquelas que apresentam múltiplos mínimos locais (Storn; Price, 1997).

2.2.1 Otimização local e global

Uma função objetivo é caracterizada como multimodal quando possui mais de um mínimo local - o menor valor da função em uma determinada vizinhança. Esse cenário gera um desafio conhecido como problema do ponto de partida. Dependendo

do critério de busca do algoritmo utilizado em questão e do ponto inicial da busca, o algoritmo pode ser atraído para um mínimo local, que pode não corresponder à solução ideal. Nesse contexto, a solução ideal é o mínimo global, o menor valor que a função assume considerando todo o seu domínio. A representação de uma função objetivo multimodal pode ser visualizada na Figura 9 (Price; Storn; Lampinen, 2014).

Figura 9 – Função multimodal



Fonte: Price, Storn e Lampinen (2014, p. 17).

A evolução diferencial combate o problema do ponto de partida avaliando a função objetivo em múltiplos e aleatórios pontos iniciais escolhidos dentro de limites predefinidos (Price; Storn; Lampinen, 2014). As próximas subseções serão apresentadas de forma a detalhar os principais conceitos e o funcionamento do algoritmo de evolução diferencial na seguinte sequência: estrutura da população, inicialização, mutação, crossover e seleção.

2.2.2 Estrutura da população

O algoritmo de evolução diferencial trabalha com o conceito de gerações. Em cada geração g, existe uma população atual, simbolizada por P_x. Essa população é composta por N_p vetores, e cada um desses vetores é composto por D parâmetros. A cada vetor da população atual é atribuído um índice i, e a cada um de seus parâmetros, um índice j. A Equação 12 representa esse conceito (Price; Storn; Lampinen, 2014).

$$\begin{aligned} \mathbf{P}_{x,g} &= (\mathbf{x}_{i,g}), i = 0, 1, ..., N_p - 1, g = 0, 1, ..., g_{max}, \\ \mathbf{x}_{i,g} &= (x_{j,i,g}), j = 0, 1, ..., D - 1 \end{aligned} \tag{12}$$

Além da população atual, em cada geração existe uma população intermediária, também conhecida como mutante, P_v, a qual é gerada a partir das mutações do algoritmo. Essa população intermediária segue a mesma estrutura da população atual e está representada na Equação 13 (Price; Storn; Lampinen, 2014).

$$\begin{aligned} \mathbf{P}_{v,g} &= (\mathbf{v}_{i,g}), i = 0, 1, ..., N_p - 1, g = 0, 1, ..., g_{max}, \\ \mathbf{v}_{i,g} &= (v_{j,i,g}), j = 0, 1, ..., D - 1 \end{aligned} \tag{13}$$

Por fim, em cada geração é realizada uma recombinação dos vetores da população atual com os vetores da população mutante, gerando uma população de teste P_u. A população de teste está representada na Equação 14 (Price; Storn; Lampinen, 2014). É relevante mencionar que a população atual pode ser composta, tanto por vetores resultados de recombinações e seleções passadas, quanto por pontos iniciais como será explicado na próxima subseção.

$$\begin{aligned} \mathbf{P}_{u,g} &= (\mathbf{u}_{i,g}), i = 0, 1, ..., N_p - 1, g = 0, 1, ..., g_{max}, \\ \mathbf{u}_{i,g} &= (u_{j,i,g}), j = 0, 1, ..., D - 1 \end{aligned} \tag{14}$$

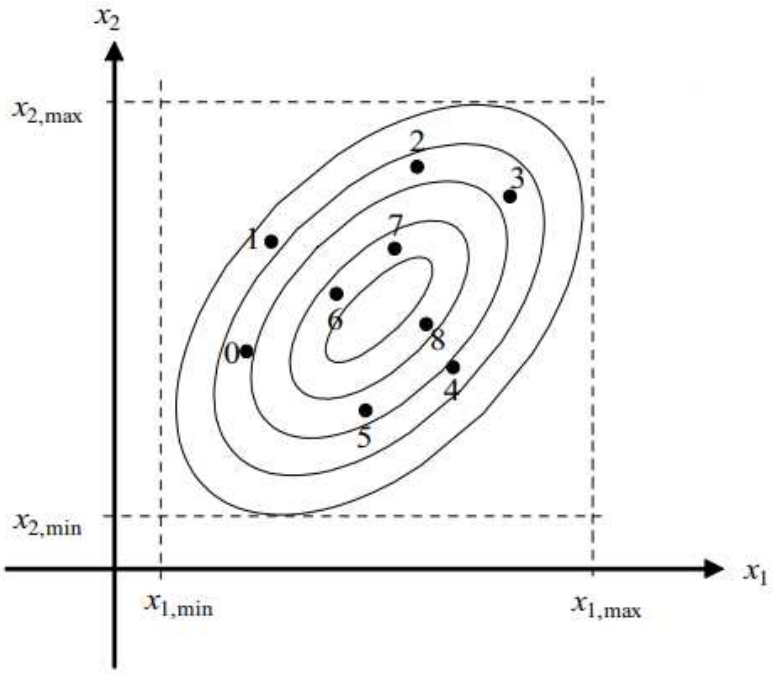
2.2.3 Inicialização

Na fase de inicialização, são utilizados vetores de limite, que possuem os limites inferior b_l e superior b_u para cada um dos D parâmetros, de cada um dos N_p vetores da população atual na geração zero. A partir dos limites, é gerada uma população de vetores contendo valores aleatórios para cada um dos parâmetros. A Figura 10 retrata um exemplo da população inicial de um problema, onde cada ponto representa um vetor de dois parâmetros x_1 e x_2 (Price; Storn; Lampinen, 2014).

2.2.4 Mutação

A mutação na evolução diferencial consiste em uma operação vetorial que tem por objetivo criar novos vetores, que representam indivíduos, com o objetivo de explorar o espaço de soluções. A operação que representa essa mutação pode ser vista na Equação 15, a qual consiste em realizar uma subtração com um peso F de dois vetores $x_{r1,g}$ e $x_{r2,g}$ escolhidos aleatoriamente e adicioná-los a um vetor base $x_{r0,g}$ que pode ser escolhido com diferentes métodos (Price; Storn; Lampinen, 2014).

Figura 10 – Fase de inicialização



Fonte: Adaptado de Price, Storn e Lampinen (2014, p. 31).

$$\mathbf{v}_{i,g} = \mathbf{x}_{r0,g} + F \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \tag{15}$$

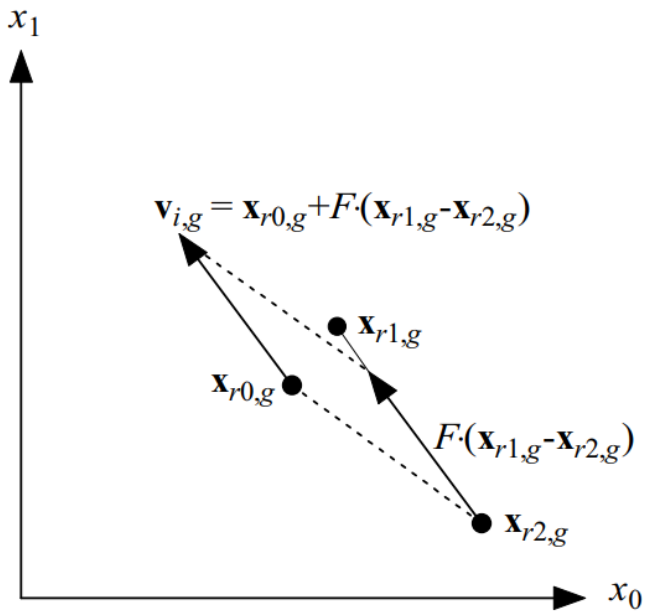
Outra forma de visualização dessa mesma operação é apresentada na Figura 11, onde é possível observar graficamente a disposição dos vetores de exemplo. Para complementar essa operação, após a mutação, ocorre a etapa de crossover, também conhecida como recombinação, que combina informações de dois vetores para formar um terceiro vetor teste (Price; Storn; Lampinen, 2014).

2.2.5 Crossover

Nessa etapa é criado o vetor teste conforme descrito na Equação 16. Cada parâmetro j do vetor teste é obtido a partir do vetor da população atual $x_{i,j,g}$ ou do vetor mutante $v_{i,j,g}$. Essa escolha é definida por um valor aleatório: caso o valor seja menor que a probabilidade de crossover Cr, definida pelo usuário, o valor do vetor mutante será utilizado, caso contrário, será utilizado o valor do vetor da população atual (Price; Storn; Lampinen, 2014).

$$\mathbf{u}_{i,g} = u_{j,i,g} = \begin{cases} v_{j,i,g}, & \text{se } (\text{rand}_j(0,1) \leq Cr \text{ ou } j = j_{rand}) \\ x_{j,i,g}, & \text{caso contrário} \end{cases} \tag{16}$$

Figura 11 – Mutação



Fonte: Adaptado de Price, Storn e Lampinen (2014, p. 39).

Há também um índice j_{rand}, gerado aleatoriamente, para garantir que pelo menos um parâmetro do vetor teste seja herdado do vetor mutante. Sem isso, haveria uma probabilidade de o vetor teste ser idêntico ao vetor da população atual, o que não contribuiria para o progresso da busca. A etapa final de cada geração, denominada seleção, compara os vetores da população atual com os vetores teste gerados, determinando quais avançarão para as próximas gerações (Price; Storn; Lampinen, 2014).

2.2.6 Seleção

Na etapa de seleção, o vetor que avançará para as próximas gerações é determinado pela Equação 17. O vetor de testes e o vetor da população atual passam pela avaliação da função objetivo e, em problemas de minimização, aquele que obtiver o menor resultado, é selecionado para ir adiante. Portanto, esses processos de mutação, crossover e seleção são repetidos geração por geração até que encontrem um ponto mínimo ou outro critério de parada seja atingido, como por exemplo, número máximo de gerações g_{max} (Price; Storn; Lampinen, 2014).

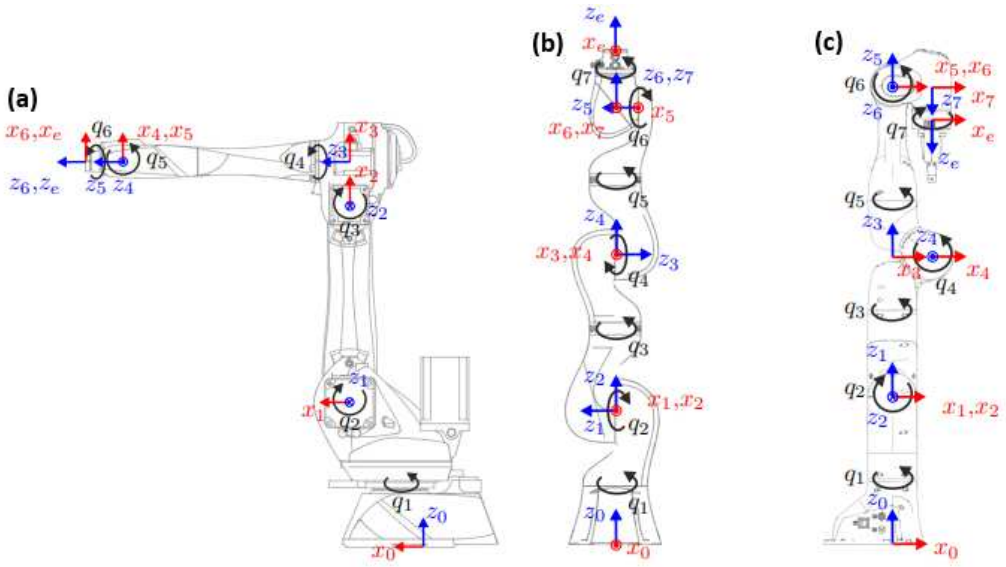
$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g}, & \text{se } f(\mathbf{u}_{i,g}) \leq f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g}, & \text{caso contrário} \end{cases} \tag{17}$$

2.3 ESTADO DA ARTE

As singularidades são um problema fundamental dos manipuladores industriais (Oetomo; ANG JUNIOR, 2009). Compreender a estrutura do manipulador e suas posturas singulares é essencial para evitar perdas de controle e garantir segurança durante suas trajetórias (Zaplana; Hadfield; Lasenby, 2022). Nesse contexto, um estudo recente propôs uma nova abordagem de prevenção de singularidades em tempo real (Beck *et al.*, 2023).

Nesse estudo, Beck *et al.* (2023) iniciam com a explicação da cinemática direta e a definição dos parâmetros de Denavit-Hartenberg de três manipuladores seriais: o Comau Racer 7-1.4 (a), o KUKA LBR iiwa 14 R820 (b) e o Franka Emika Panda (c), seus esquemáticos podem ser vistos na Figura 12. Em seguida, definem e demonstram, a partir do manipulador Comau Racer 7-1.4, como calcular a matriz jacobiana e utilizá-la para identificar posturas singulares (Beck *et al.*, 2023).

Figura 12 – Estrutura de coordenadas dos manipuladores seriais



Fonte: Adaptado de Beck *et al.* (2023, p. 2 e 3).

Além disso, foram feitas as devidas considerações para os manipuladores KUKA LBR iiwa 14 R820 e Franka Emika Panda, uma vez que são redundantes e exigem adaptações para encontrar posturas singulares a partir do jacobiano. Uma breve explicação sobre a dinâmica desses manipuladores foi apresentada e, posteriormente, a formulação do problema como uma otimização de trajetórias foi realizada de duas formas diferentes, utilizando o método dos pontos interiores como solução (Beck *et al.*, 2023).

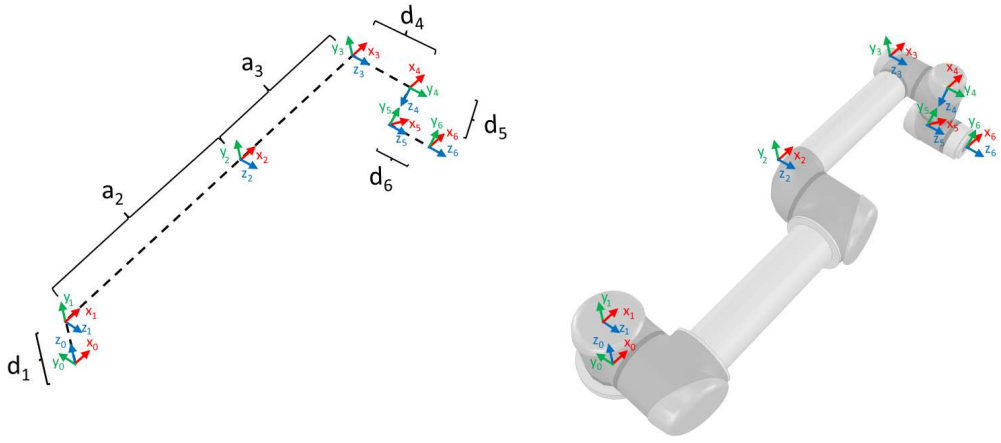
A primeira abordagem consistiu em restringir a medida de manipulabilidade do robô acima de um determinado valor mínimo. Entretanto, essa solução não previne

todas as singularidades e é custosa computacionalmente. Portanto, a nova abordagem apresentada consistiu em formular a função objetivo como uma somatória das distâncias entre as configurações das juntas e suas respectivas configurações singulares. Essas configurações são previamente conhecidas por meio do jacobiano analítico (Beck *et al.*, 2023).

Realizaram, então, uma comparação entre essas duas abordagens utilizando simulações com o algoritmo de Monte Carlo. O método de prevenção de singularidades proposto apresentou, em média, um desempenho superior ao método de maximização da manipulabilidade. A diferença foi especialmente expressiva no Franka Emika Panda, devido à sua complexidade estrutural. Por outro lado, o método de maximização da manipulabilidade obteve melhor performance no Comau Racer 7-1.4, pois, segundo Beck *et al.* (2023), existem menos possibilidades de trajetórias alternativas para evitar singularidades em robôs não redundantes.

Outro estudo realizou o cálculo da cinemática inversa com prevenção de singularidades de um robô UR5, um manipulador serial de seis graus de liberdade da Universal Robots, utilizando uma máquina de estados finita. Nesse trabalho, Villalobos, Sanchez e Martell (2022) iniciam definindo os parâmetros de Denavit-Hartenberg e calculando a cinemática direta do UR5, cuja estrutura de coordenadas é apresentada na Figura 13. Em seguida, determinam o jacobiano analítico desse manipulador e, com base nele, classificam as possíveis posturas singulares de forma análoga às de robôs antropomórficos convencionais: ombro, cotovelo e punho.

Figura 13 – Estrutura de coordenadas do UR5

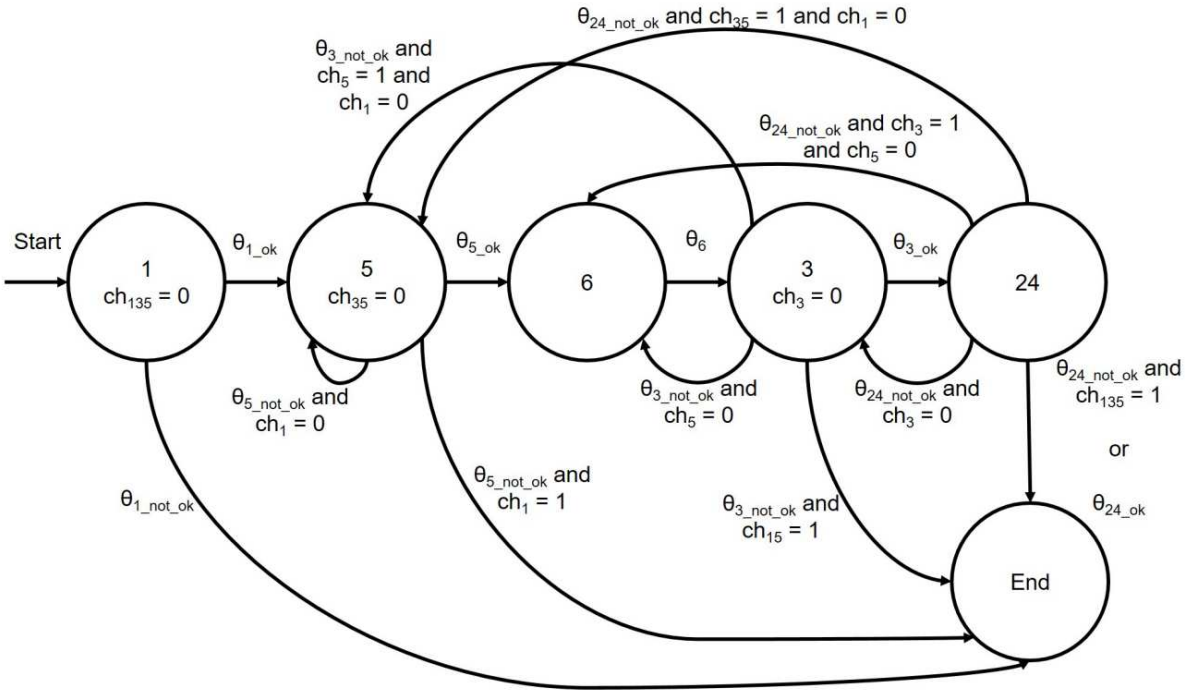


Fonte: Villalobos, Sanchez e Martell (2022, p. 4).

Após essas definições, Villalobos, Sanchez e Martell (2022) apresentam dois algoritmos para solução da cinemática inversa. O primeiro algoritmo calcula todas as posturas que resolvem a cinemática e escolhe a que proporciona os movimentos mais

eficientes, ou seja, que minimiza o deslocamento das juntas. O segundo algoritmo é representado pela máquina de estados finita ilustrada na Figura 14. Cada estado verifica os valores dos ângulos obtidos na cinemática inversa, garantindo que nunca assumam valores inviáveis ou que levem o robô a posturas singulares (Villalobos; Sanchez; Martell, 2022).

Figura 14 – Máquina de estados finita



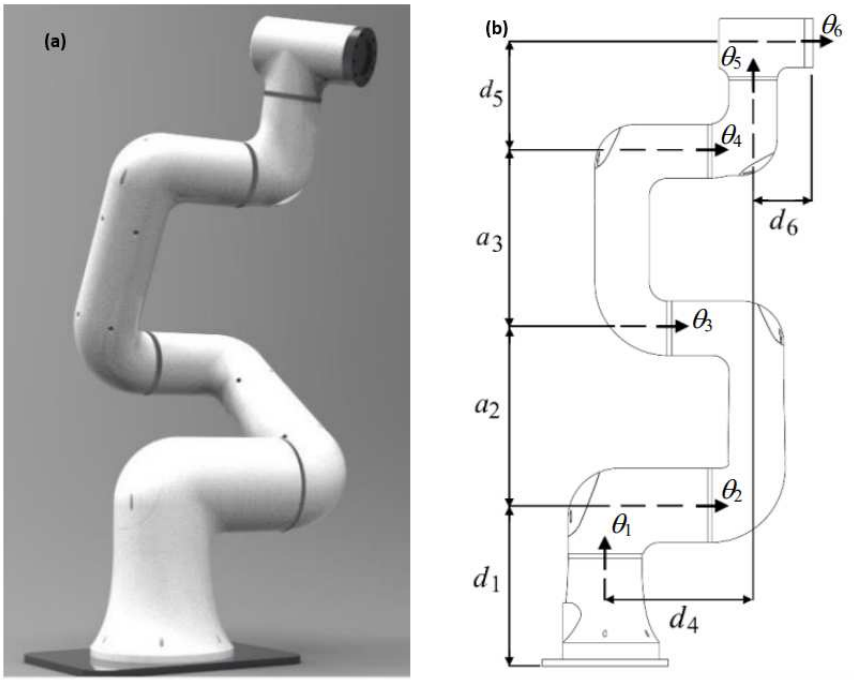
Fonte: Villalobos, Sanchez e Martell (2022, p. 9).

Por fim, realizaram uma comparação entre os dois algoritmos com base em simulações que envolveram medições dos ângulos para dez posturas diferentes, sendo oito singulares e duas regulares. Os resultados indicaram que o segundo algoritmo é mais adequado para aplicações com baixo custo computacional e maior flexibilidade em relação aos resultados fornecidos pela cinemática inversa (Villalobos; Sanchez; Martell, 2022).

Em outro trabalho, Aboelnasr, Baha e Mokhiamar (2021) apresentam a estrutura do AER-1, um braço robótico serial com seis graus de liberdade. A Figura 15 exibe tanto o modelo 3D (a) quanto a estrutura cinemática (b) desse braço. Assim como nos outros estudos, Aboelnasr, Baha e Mokhiamar (2021) realizam a modelagem do robô utilizando os parâmetros de Denavit-Hartenberg para calcular sua cinemática direta.

Após definir a estrutura do robô, Aboelnasr, Baha e Mokhiamar (2021) aplicaram o algoritmo de Monte Carlo para gerar o espaço de trabalho do manipulador. Para cada conjunto de ângulos aleatórios gerados, a posição do efetuador final foi calculada por

Figura 15 – Estrutura cinemática do AER-1



Fonte: Aboelnasr, Baha e Mokhiamar (2021, p. 2).

meio da cinemática direta. Em seguida, formularam as condições de singularidades, baseadas no cálculo do determinante da matriz jacobiana analítica. Durante a geração do espaço de trabalho, o determinante era continuamente avaliado com os ângulos aleatórios, identificando posturas singulares. As posições cartesianas que resultaram em um determinante igual a zero foram ilustradas a partir de uma figura (Aboelnasr; Baha; Mokhiamar, 2021).

Por fim, após identificarem analiticamente os ângulos que levavam o robô às singularidades, Aboelnasr, Baha e Mokhiamar (2021) desenvolveram um algoritmo para explorar todos os valores dentro dos limites das juntas, com incrementos de 1°. A partir desses dados, foram gerados gráficos e um mapa de calor para facilitar a visualização do comportamento do determinante da matriz jacobiana. Assim, demonstraram que é possível utilizar métodos numéricos e a análise da matriz jacobiana para caracterizar as singularidades de um manipulador (Aboelnasr; Baha; Mokhiamar, 2021).

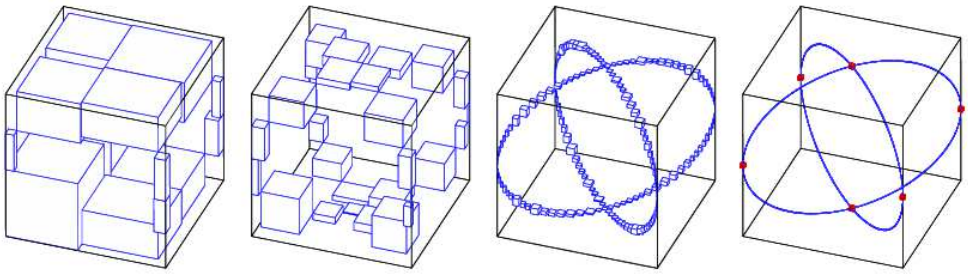
Bohigas *et al.* (2014) apresentaram um trabalho com o intuito de desenvolver um método capaz de caracterizar e identificar diferentes tipos de singularidade em robôs com qualquer arquitetura relevante. A origem dessa abordagem e sua relação com outros estudos são discutidas na introdução do artigo, porém, conforme mencionam Bohigas *et al.* (2014), nesse artigo a abordagem de caracterização e identificação de singularidades é apresentada detalhadamente.

Primeiramente, Bohigas *et al.* (2014) definem as configurações singulares e demonstram, através de sistemas de equações que dependem dos valores das

juntas e de um vetor de velocidades, como caracterizar todas essas configurações em um robô, classificando-as em dois tipos: o problema da cinemática instantânea direta e o problema da cinemática instantânea inversa. Desse modo, Bohigas *et al.* (2014) apresentam as definições e as particularidades de cada um dos seis tipos possíveis de singularidades para um robô, derivados da classificação das configurações mencionadas acima.

Em seguida, Bohigas *et al.* (2014) aplicam os conceitos apresentados em um estudo de caso simples que pode ser resolvido de forma analítica para facilitar o entendimento. Para robôs complexos, são necessários algoritmos numéricos para a solução dos sistemas de equações. Então, Bohigas *et al.* (2014), apresentaram um exemplo de algoritmo numérico adaptado, denominado *Branch-and-prune*, no qual uma caixa de limites é criada e iterativamente remove as porções que não representam soluções, para isolar cada tipo de singularidade, representado pelos pontos vermelhos, conforme ilustrado na Figura 16.

Figura 16 – Progressão do algoritmo número *Branch-and-prune*



Fonte: Bohigas *et al.* (2014, p. 7).

Bohigas *et al.* (2014) também demonstraram que o algoritmo é eficaz para robôs com qualquer estrutura cinemática, apresentando apenas um aumento no tempo de computação, como é comum em outros algoritmos, mas sem grandes dificuldades para o método. Apesar de não terem sido encontrados estudos que utilizem a abordagem de detecção de singularidades proposta neste trabalho, os estudos existentes contribuem para a compreensão da metodologia que será apresentada para a caracterização de posturas singulares.

3 MÉTODO

Para o desenvolvimento do algoritmo de detecção de singularidades em três robôs propostos, visando um mapeamento das condições que causam posturas singulares, neste capítulo são apresentados os materiais e métodos utilizados para alcançar os objetivos propostos. Considerando a necessidade de restringir o escopo do trabalho para validar a metodologia, o algoritmo geral desenvolvido foi particularizado para ser testado em três manipuladores seriais: o robô planar de três juntas rotativas, um robô antropomórfico e um robô de estrutura não convencional.

A metodologia segue o fluxograma apresentado na Figura 17. Na etapa inicial, são introduzidas as declarações e definições das variáveis e funções essenciais detalhadas na seção 3.1. Na sequência, define-se o ciclo principal, que realiza sucessivas iterações em busca de posturas singulares e, por fim, os resultados são apresentados de forma automatizada. As ferramentas utilizadas no desenvolvimento do algoritmo foram o Visual Studio Code e a linguagem Python. Para visualização dos dados, além do Python, utilizou-se o software CoppeliaSIM e uma simulação em javascript própria do terceiro manipulador, que serão detalhados na seção 3.3.

O Visual Studio Code é um editor de texto com suporte para diversas linguagens de programação, adequado tanto para uso privado quanto comercial, que permite a depuração e testes de códigos. Por oferecer uma documentação abrangente e bom suporte, foi escolhido como ambiente de desenvolvimento para a criação do algoritmo de detecção de singularidades (Code, 2024).

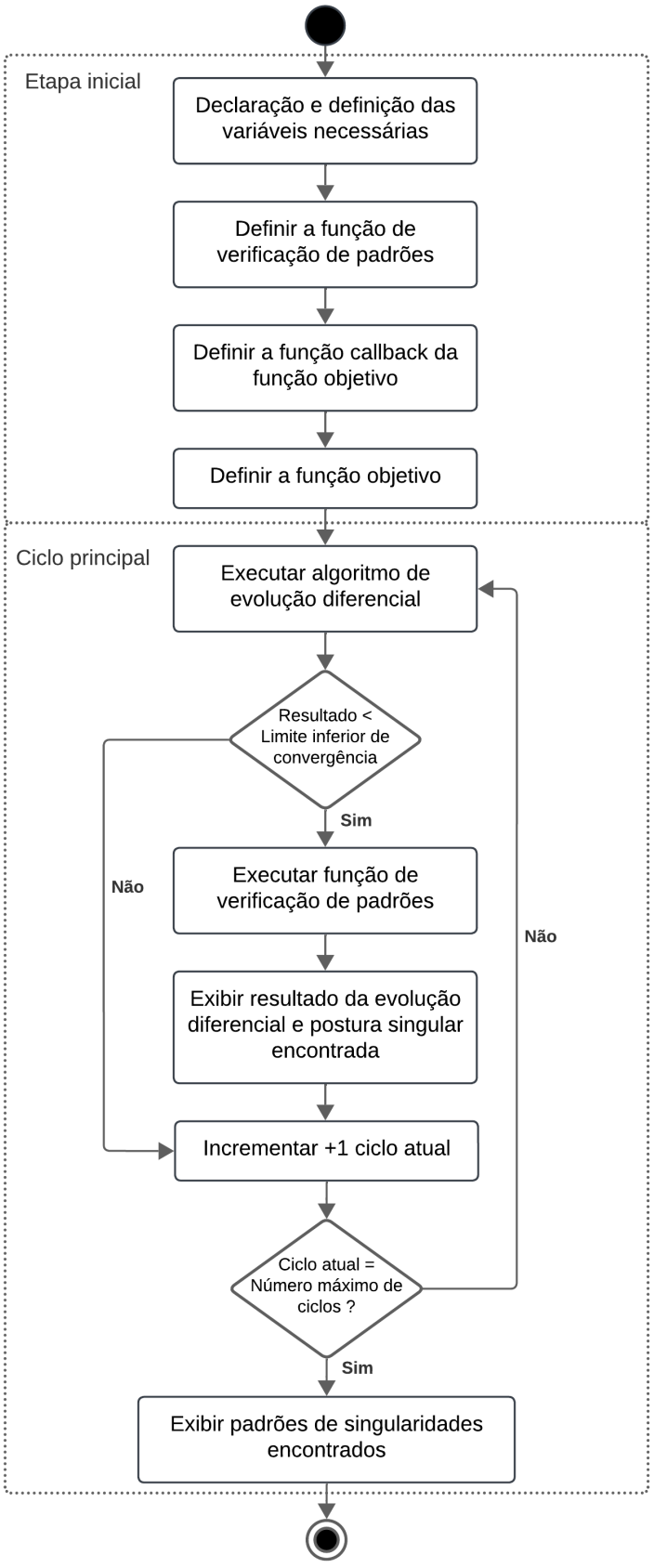
A linguagem Python foi adotada por disponibilizar bibliotecas bem documentadas para álgebra linear, gráficos e robótica, facilitando a modelagem e os cálculos necessários para os manipuladores (Corke, 2024). O software CoppeliaSIM pode ser utilizado para simulação e validação de sistemas robóticos complexos. Neste trabalho, esse software foi empregado para inspeção visual das posturas singulares encontradas (COPPELIA ROBOTICS AG, 2024), assim como a simulação javascript.

3.1 ETAPA INICIAL

3.1.1 Variáveis utilizadas

A primeira etapa de elaboração do algoritmo consiste em declarar e definir as variáveis que serão utilizadas. A primeira delas é a lista de limites das juntas do manipulador, que especifica os valores mínimos e máximos para cada junta, de forma a delimitar o espaço de busca da evolução diferencial. Também é necessário definir o número máximo de ciclos - o número de vezes que o ciclo principal irá repetir realizando

Figura 17 – Fluxograma do algoritmo de detecção de singularidades



Fonte: Autor (2024).

a busca da evolução diferencial - pois o algoritmo é iterativo e precisa de uma condição de parada. Esse valor deve ser grande o suficiente para permitir a identificação dos padrões de singularidade, mas não tão grande a ponto de dificultar a análise dos resultados. Neste trabalho, testes indicaram que 30 ciclos são adequados para atender a ambos os critérios.

Além disso, são necessárias duas listas adicionais. A primeira armazena objetos de uma classe que representa os valores das juntas em posturas singulares identificadas, sendo utilizada na função de verificação de padrões. Essa classe, além do valor da junta, também armazena um nome para identificação da junta e o número de vezes que o valor se repetiu ao longo dos ciclos de iteração. A segunda lista contém os objetos reconhecidos como padrões pela função, permitindo a penalização de determinados valores durante a execução da evolução diferencial, como será detalhado na explicação da função objetivo.

Ainda na etapa de definição de variáveis, é fundamental realizar a modelagem do manipulador que será utilizado e determinar sua tabela de Denavit-Hartenberg, que será empregada no posterior cálculo de seu jacobiano. Na pesquisa foram modelados três manipuladores como estudo de caso: um robô planar com três juntas rotativas, um manipulador antropomórfico e um manipulador de estrutura não convencional. A modelagem desses exemplos é apresentada nas seções 3.1.1.1, 3.1.1.2 e 3.1.1.3.

3.1.1.1 Robô planar de três juntas

Na Figura 18, o primeiro robô utilizado como estudo de caso, é composto por três juntas rotativas paralelas. A direção do eixo x_0 é arbitrária, enquanto os eixos x_1 e x_2 foram dispostos na mesma direção dos elos correspondentes. Como todas as juntas são rotativas, os parâmetros θ_i representam as variáveis de junta, ou seja, os cálculos subsequentes e a busca por soluções na evolução diferencial serão realizados em função dessas variáveis. Os parâmetros de Denavit-Hartenberg desse modelo estão dispostos na Tabela 1 (Siciliano *et al.*, 2010).

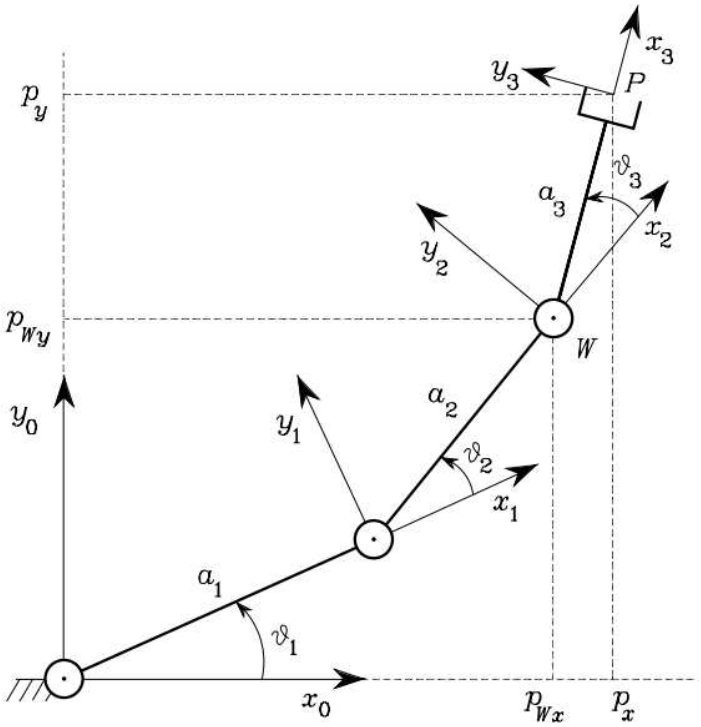
Tabela 1 – Parâmetros de Denavit-Hartenberg do robô planar

Junta	α	a	d	θ
1	0	a_1	0	θ_1
2	0	a_2	0	θ_2
3	0	a_3	0	θ_3

Fonte: Siciliano *et al.* (2010, p. 69).

Dado que a evolução diferencial realiza a busca a partir das variáveis de junta θ_i, os demais parâmetros precisam ser definidos numericamente, e não apenas em termos algébricos. Dessa forma, os tamanhos dos elos foram escolhidos para se

Figura 18 – Estrutura do robô planar de três juntas



Fonte: Siciliano *et al.* (2010, p. 69).

assemelhar aos do segundo manipulador utilizado neste estudo, embora qualquer outro conjunto de valores pudesse ter sido adotado. Além disso, os limites das juntas foram determinados considerando valores típicos de uma junta de revolução desconsiderando colisões, mas poderiam ser ajustados conforme a aplicação desejada. Esses dados estão apresentados nas Tabelas 2 e 3.

Tabela 2 – Tamanho dos elos do robô planar

Elo	Tamanho
a_1	1.095
a_2	0.495
a_3	0.175

Fonte: Autor (2024).

Tabela 3 – Limites das juntas do robô planar

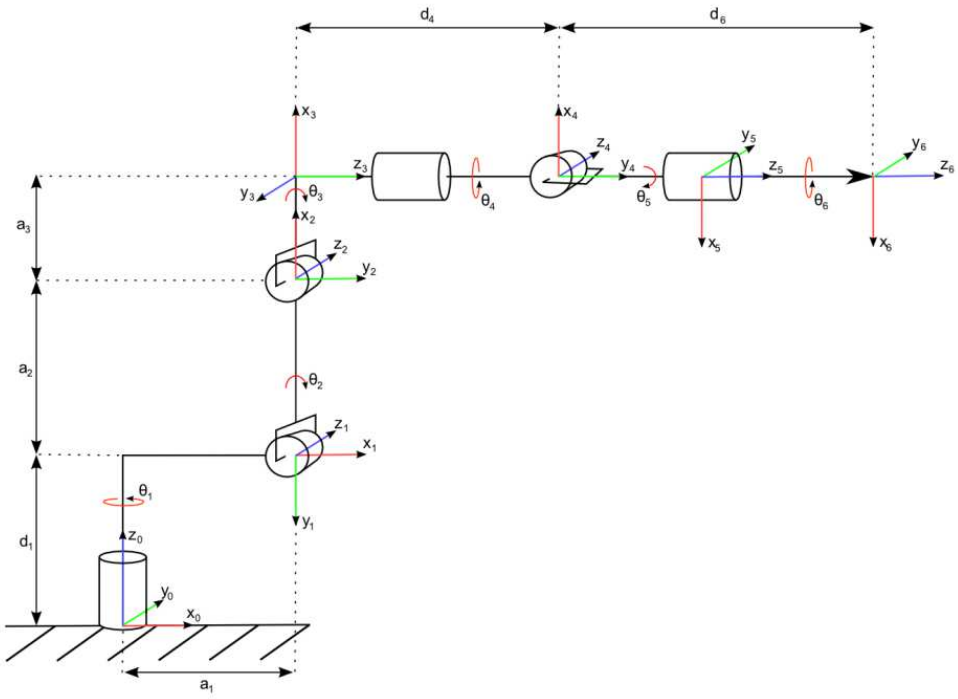
Junta	θ_{min}	θ_{max}
1	$-180°$	$180°$
2	$-180°$	$180°$
3	$-180°$	$180°$

Fonte: Autor (2024).

3.1.1.2 Manipulador antropomórfico

Na Figura 19, apresenta-se a estrutura do segundo manipulador de estudo de caso, o ABB IRB4600. Esse robô possui seis juntas de revolução, com as três últimas configuradas como um punho esférico. Assim como no exemplo anterior, as variáveis de junta são representadas pelos parâmetros θ_i. Os parâmetros de Denavit-Hartenberg desse manipulador estão dispostos na Tabela 4.

Figura 19 – Estrutura do manipulador antropomórfico



Fonte: Kvernberg (2015, p. 16).

Tabela 4 – Parâmetros de Denavit-Hartenberg do manipulador antropomórfico

Junta	α	a	d	θ
1	$-\frac{\pi}{2}$	a_1	d_1	θ_1
2	0	a_2	0	$\theta_2 - \frac{\pi}{2}$
3	$-\frac{\pi}{2}$	a_3	0	θ_3
4	$\frac{\pi}{2}$	0	d_4	θ_4
5	$\frac{\pi}{2}$	0	0	$\theta_5 + \pi$
6	0	0	d_6	θ_6

Fonte: Kvernberg (2015, p. 17).

No caso do manipulador antropomórfico, a busca da evolução diferencial também é realizada em função das variáveis de junta θ_i. Os comprimentos dos elos e os

limites das juntas foram definidos com base na variante ABB IRB4600 20/2.50, conforme especificado no manual de produto (ABB, 2024). Esses dados estão organizados nas Tabelas 5 e 6.

Tabela 5 – Tamanho dos elos do manipulador antropomórfico

Elo	Tamanho
a_1	0.175
a_2	1.095
a_3	0.175
d_1	0.495
d_4	1.2305
d_6	0.085

Fonte: Adaptado de ABB (2024, p. 72).

Tabela 6 – Limites das juntas do manipulador antropomórfico

Junta	θ_{min}	θ_{max}
1	$-180°$	$180°$
2	$-90°$	$150°$
3	$-180°$	$75°$
4	$-400°$	$400°$
5	$-120°$	$120°$
6	$-400°$	$400°$

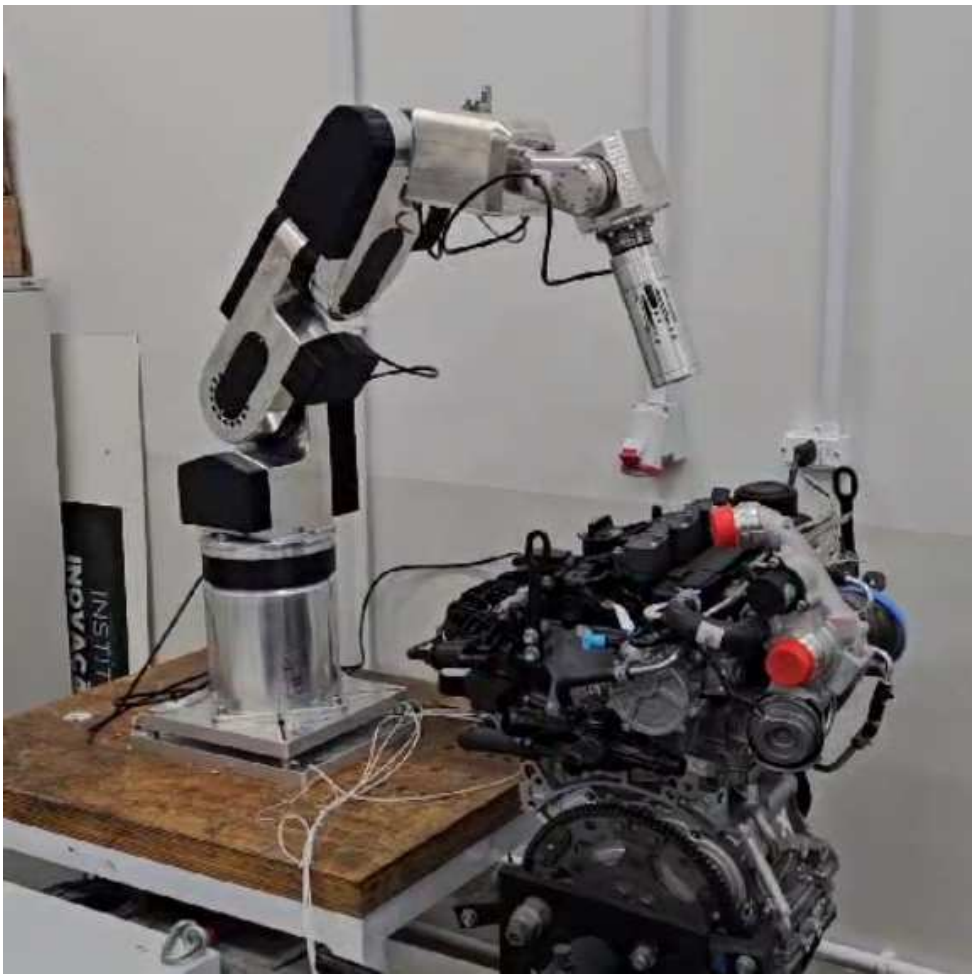
Fonte: Adaptado de ABB (2024, p. 51).

3.1.1.3 Manipulador Snake

O terceiro exemplo é um manipulador de estrutura não convencional denominado Snake. Esse robô foi desenvolvido em parceira entre a Voltor LTDA e o Serviço Nacional de Aprendizagem Industrial (SENAI) para a General Motors (GM). Sua estrutura modular foi projetada para simplificar tanto a construção quanto a flexibilidade, permitindo a adição ou remoção de graus de liberdade conforme necessário. Essa flexibilidade é possível devido ao alinhamento ortogonal entre suas juntas iniciais. A Figura 20 ilustra esse manipulador com uma configuração de sete graus de liberdade.

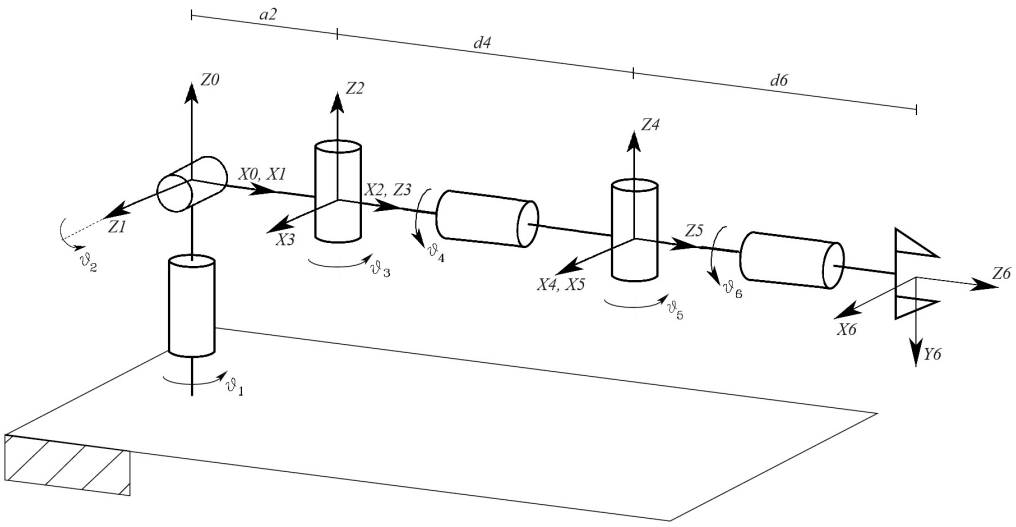
Neste trabalho, utiliza-se a configuração do Snake com seis graus de liberdade. A Figura 21 ilustra a estrutura do manipulador nessa configuração. Todas as juntas que o compõem são de revolução: as três primeiras juntas estão dispostas ortogonalmente entre si, enquanto as três últimas juntas em uma configuração de punho esférico. As variáveis de juntas são representadas pelos termos θ_i, e os parâmetros de Denavit-Hartenberg do Snake estão dispostos na Tabela 7.

Figura 20 – Manipulador Snake



Fonte: Voltor (2024).

Figura 21 – Estrutura do Snake



Fonte: Autor (2024).

Tabela 7 – Parâmetros de Denavit-Hartenberg do Snake

Junta	α	a	d	θ
1	$\frac{\pi}{2}$	0	0	θ_1
2	$-\frac{\pi}{2}$	a_2	0	θ_2
3	$-\frac{\pi}{2}$	0	0	$\theta_3 - \frac{\pi}{2}$
4	$\frac{\pi}{2}$	0	d_4	θ_4
5	$-\frac{\pi}{2}$	0	0	θ_5
6	0	0	d_6	θ_6

Fonte: Autor (2024).

No caso do Snake, o algoritmo de evolução diferencial também utiliza as variáveis de junta θ_i para minimizar a função objetivo. Os comprimentos dos elos e os limites das juntas estão organizados nas Tabelas 8 e 9. Com todas as variáveis devidamente estabelecidas, é possível definir as três principais funções do algoritmo, que serão detalhadas na próxima subseção.

Tabela 8 – Tamanho dos elos do Snake

Elo	Tamanho
a_2	0.5
d_4	0.65
d_6	0.42

Fonte: Voltor (2024).

Tabela 9 – Limites das juntas do Snake

Junta	θ_{min}	θ_{max}
1	$-360°$	$360°$
2	$0°$	$180°$
3	$-90°$	$90°$
4	$-180°$	$180°$
5	$-90°$	$90°$
6	$-180°$	$180°$

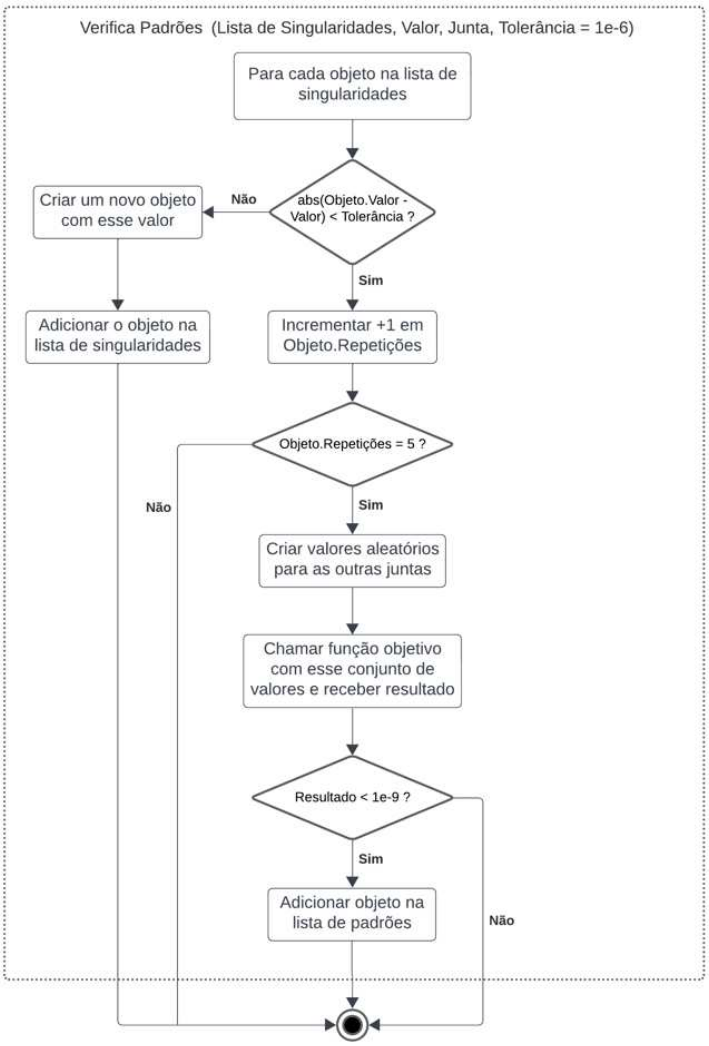
Fonte: Voltor (2024).

3.1.2 Função de verificação de padrões

A função de verificação de padrões tem como objetivo identificar padrões de singularidade. Sua criação tornou-se necessária porque, durante os testes iniciais, observou-se que o algoritmo encontrava os mesmos tipos de singularidades repetidamente em todos os ciclos. Essas singularidades estão relacionadas a

características específicas das juntas, conforme descrito por Siciliano *et al.* (2010), como o manipulador estar completamente esticado ou retraído, ou o alinhamento de dois eixos. Assim, essa função foi desenvolvida para que o algoritmo armazene esses valores e os penalize em ciclos futuros, evitando redundâncias. O fluxograma da função é apresentado na Figura 22.

Figura 22 – Fluxograma da função de verificação de padrões



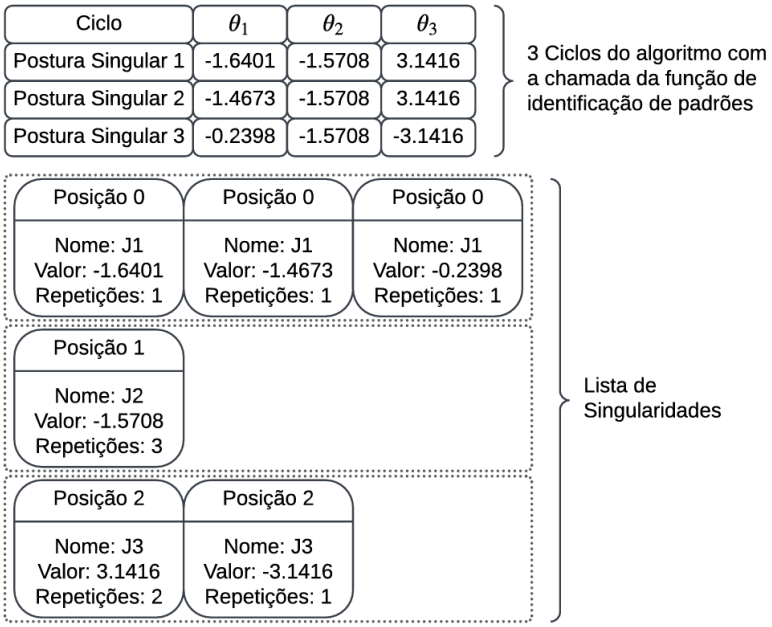
Fonte: Autor (2024).

A função recebe como parâmetros:

- a lista de objetos que representam os valores de juntas de posturas singulares identificadas anteriormente;
- o valor da junta correspondente à postura singular detectada no ciclo atual;
- um identificador que indica a qual junta o valor pertence;
- um parâmetro de tolerância para a comparação do valor de junta atual pelos valores de juntas identificados anteriormente;

Na Figura 23 é apresentada a estrutura da lista de objetos após a execução de três ciclos do algoritmo com o robô planar para exemplificação. A tolerância foi introduzida após testes indicarem que pequenas variações nos valores das juntas poderiam acarretar na mesma singularidade ao longo dos ciclos.

Figura 23 – Estrutura da lista de singularidades



Fonte: Autor (2024).

A função executa iterativamente em todos os objetos da lista de singularidades. Ao receber um valor de uma determinada junta, é comparado com todos os valores armazenados anteriormente para essa mesma junta. Se a diferença absoluta entre eles for menor que a tolerância, o contador de repetições é incrementado. Quando o número de repetições atinge cinco, um valor que se mostrou eficaz nos testes iniciais, são gerados valores aleatórios para as demais juntas, e a função objetivo é chamada com esses valores como parâmetro. Caso a função objetivo retorne um valor inferior a 1×10^{-9}, conclui-se que o valor dessa junta, isoladamente, pode gerar singularidades. Nesse caso, o objeto que representa esse valor é armazenado na lista de padrões para ser penalizado nos ciclos subsequentes.

3.1.3 Função callback

A segunda função a ser implementada é denominada *callback* e é um parâmetro chamado ao final de cada geração na *differential_evolution*, que foi a função de evolução diferencial escolhida do Python e será detalhada na seção do ciclo principal. Caso a função *callback* retorne *True*, a otimização global será interrompida. Além disso,

o parâmetro *disp* pode ser ativado para exibir o valor da função objetivo a cada geração.

Durante testes com o parâmetro *disp*, foi observado que a evolução diferencial continuava a busca mesmo após encontrar um valor zero para a função objetivo. Assim, foi criada uma função *callback* que verifica o valor da função objetivo a cada geração e, caso seja zero, retorna *True* e encerra a busca. Essa abordagem trouxe um ganho significativo de tempo para o algoritmo.

3.1.4 Função objetivo

A última função a ser implementada é a função objetivo, utilizada pela evolução diferencial para minimizar seu valor de retorno. Essa função recebe como parâmetro as variáveis de junta que são ajustadas pelo algoritmo durante o processo de otimização. O primeiro passo da função objetivo é percorrer o vetor de padrões para verificar se algum dos valores atuais das juntas corresponde a um padrão previamente identificado. Caso um desses valores seja encontrado, a função retorna um valor elevado 1×10^6, que pode ser interpretado pela evolução diferencial como uma penalização.

Como o objetivo é minimizar o valor de retorno, essa penalização força o algoritmo a buscar um novo conjunto de variáveis que evite configurações singulares já reconhecidas. Assim, o sistema se mantém distante das singularidades detectadas. A tolerância para considerar dois valores próximos foi definida em $0,1rad$, pois não representou um problema para os estudos de caso utilizados, porém esse valor pode ser ajustado conforme necessidade.

Caso os valores recebidos como parâmetro não correspondam a um padrão previamente reconhecido, a função executa dois passos: calcula o jacobiano do robô e retorna o valor absoluto de seu determinante. O cálculo do jacobiano é realizado por meio da função *jacob0* da biblioteca *roboticstoolbox* (Corke, 2024), que utiliza o jacobiano geométrico, no qual a velocidade do efetuador final é expressa em termos de componentes linear e angular, conforme descrito por Siciliano *et al.* (2010).

Como as singularidades podem ser descritas em função do determinante do jacobiano igualado a zero (Siciliano *et al.*, 2010), opta-se por considerar seu valor absoluto, seguindo a abordagem proposta por Aboelnasr, Baha e Mokhiamar (2021), pois assim é possível tratá-lo como um problema de otimização, já que esse determinante pode assumir valores negativos. Logo, o problema é tratado como uma minimização por meio de evolução diferencial. Dessa forma, após todas as variáveis e funções preliminares terem sido definidas, pode-se iniciar o ciclo principal de operação.

3.2 CICLO PRINCIPAL

O ciclo principal opera de forma iterativa, começando pela execução do algoritmo de evolução diferencial. Esse método é um bom candidato para a detecção

de singularidades, pois, conforme apontam Price, Storn e Lampinen (2014), é capaz de lidar com funções multimodais e encontrar mínimos globais, explorando grandes espaços de busca. Essas características estão presentes no estudo do jacobiano, conforme demonstrado na seção 2.1.5.5.

A função de evolução diferencial utilizada no algoritmo é chamada *differential_evolution* e pertence à biblioteca *Scipy*, dentro do pacote *Optmize*. Essa biblioteca, amplamente utilizada para manipulação de dados, foi escolhida por ser de código aberto e oferecer comandos de alto nível (SciPy, 2024). Os parâmetros necessários fornecidos para a função incluem a função objetivo, a lista com os limites do manipulador e a função *callback*. Os demais parâmetros aceitos pela função, que podem ser conferidos em (SciPy, 2024), foram mantidos nos valores padrões, uma vez que já apresentavam bons resultados.

Entre esses valores padrões, tem-se a estratégia denominada *best1bin*. Nessa estratégia, calcula-se a diferença ponderada pelo peso F entre dois vetores aleatórios, que é então somada ao vetor $x_{r0,g}$, conforme Equação 15, que no caso dessa estratégia é o melhor indivíduo da população, o que justifica a nomenclatura *best1*. Durante a fase de crossover, utiliza-se uma distribuição binomial para cada um dos elementos do vetor, por isso o sufixo *bin*, para gerar um valor aleatório que define se o vetor teste será composto pelo elemento do vetor mutante ou do vetor original, conforme descrito na Equação 16. Em comparação com essa equação, na *best1bin* não há um j_{rand}, porém, o último valor do vetor durante o crossover é sempre proveniente do vetor mutante, garantindo que o vetor teste não seja idêntico ao vetor da população atual (Qiang; Mitchell, 2014).

Além da estratégia, a função possui um número máximo padrão de 1000 gerações e o tamanho da população em cada geração é de 15 indivíduos. A função também utiliza uma tolerância absoluta de zero e tolerância relativa de 0,01, ambas responsáveis por encerrar a busca ao considerar que o algoritmo convergiu. O fator F utilizado na mutação é definido por uma tupla padrão *(0,5; 1)*, o que significa que, a cada geração, é gerado aleatoriamente um valor dentro desse intervalo. A probabilidade padrão de crossover C_r é 0,7. Uma descrição detalhada dos demais parâmetros e suas funcionalidades está apresentada em SciPy (2024).

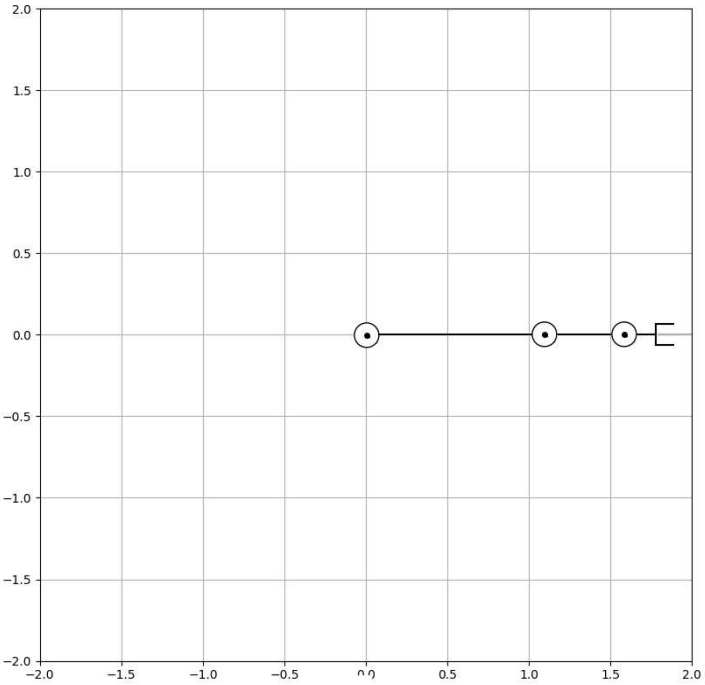
Após a execução do algoritmo de evolução diferencial, verifica-se se o determinante da matriz jacobiana é zero, considerando uma tolerância absoluta de 1×10^{-18}. Isso se deve ao fato de o algoritmo nem sempre convergir exatamente para zero ao encontrar uma postura singular, como será apresentado no capítulo de resultados. Caso o valor obtido seja inferior a essa tolerância, a função de verificação de padrões é chamada para cada junta do manipulador. Após essa função, são exibidos o resultado do algoritmo de evolução diferencial, ou seja, qual o valor mínimo encontrado para o determinante do jacobiano, e as variáveis de junta θ_i que obteve. Desse modo, o

ciclo principal continua até atingir o número máximo de iterações e, ao final, os padrões de juntas são exibidos.

3.3 VISUALIZAÇÃO DAS POSTURAS

Para visualização das posturas singulares encontradas no robô planar de três juntas, foi utilizada a biblioteca *Matplotlib.pyplot*, desenvolvida para realizar um trabalho semelhante ao feito no software de cálculo numérico *MATLAB* (Matplotlib, 2024). Primeiramente, por meio da cinemática direta, calcula-se a posição cartesiana das juntas e do efetuador final. Em seguida, nas posições das juntas, desenham-se círculos e segmentos de reta conectando-os. Por fim, na posição do efetuador final, representa-se uma garra para ilustrar o manipulador. Essa representação é exibida na Figura 24.

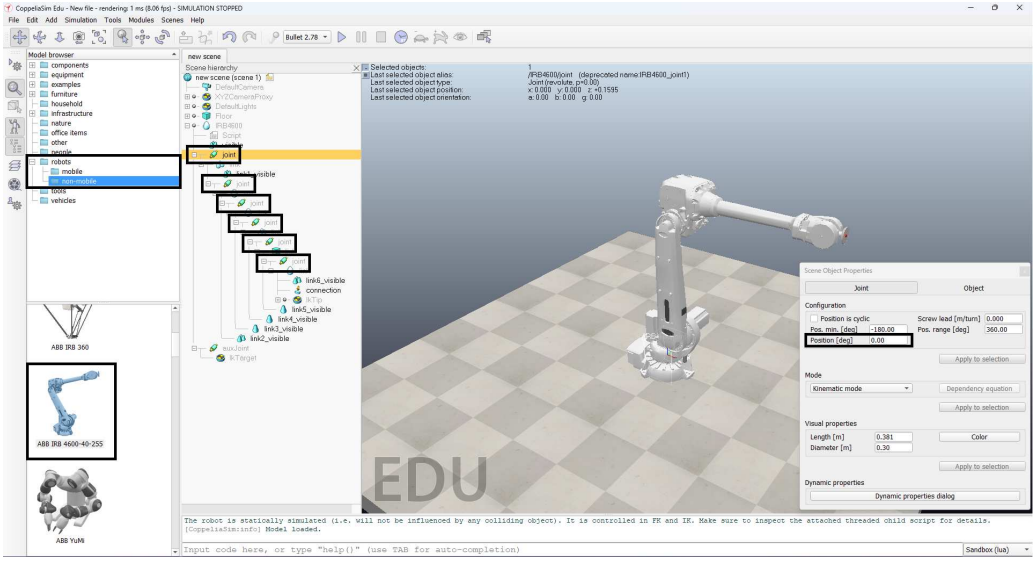
Figura 24 – Visualização do robô planar



Fonte: Autor (2024).

No software CoppeliaSIM, é possível visualizar as posturas singulares do manipulador antropomórfico. Primeiramente, seleciona-se o modelo *ABB IRB 4600-40-255* na aba *Model Browser/robots/non-mobile*, como apresentado na Figura 25. Em seguida, para cada uma das juntas presentes na aba *Scene hierarchy/IRB4600*, é possível ajustar seus valores utilizando a opção *Position [deg]*, também destacada na Figura 25. Dessa forma, os valores encontrados pelo algoritmo podem ser inseridos manualmente para visualização das posturas singulares.

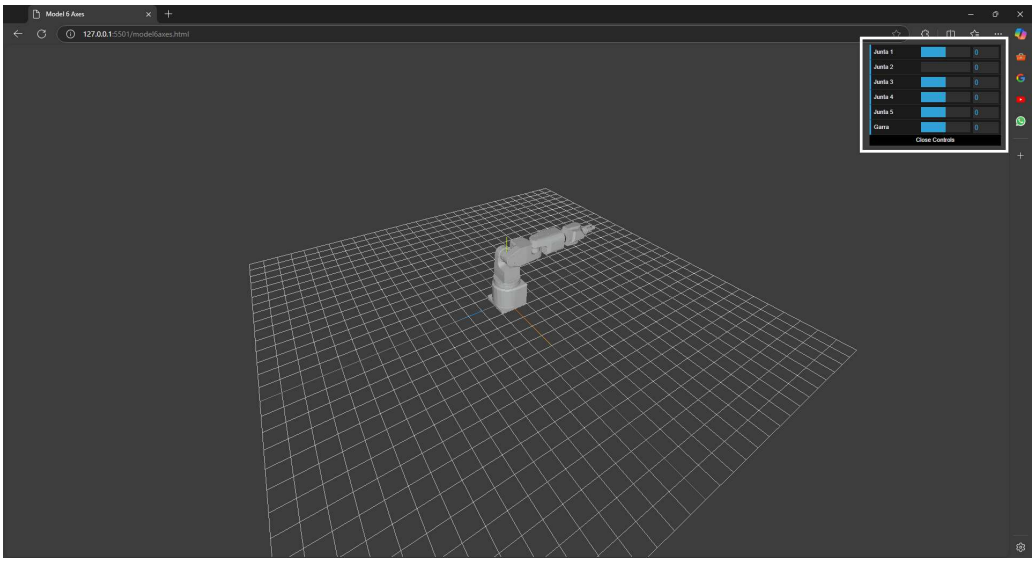
Figura 25 – Modelo IRB 4600 CoppeliaSIM



Fonte: Autor (2024).

Para visualizar as posturas singulares do manipulador Snake, foi utilizado um algoritmo desenvolvido em *JavaScript* fornecido pela Voltor. Esse algoritmo cria um servidor local acessível via navegador web, conforme apresentado na Figura 26. Assim, através da aba de controles no canto superior direito, é possível inserir manualmente os valores encontrados para cada uma das juntas.

Figura 26 – Visualização do Snake



Fonte: Autor (2024).

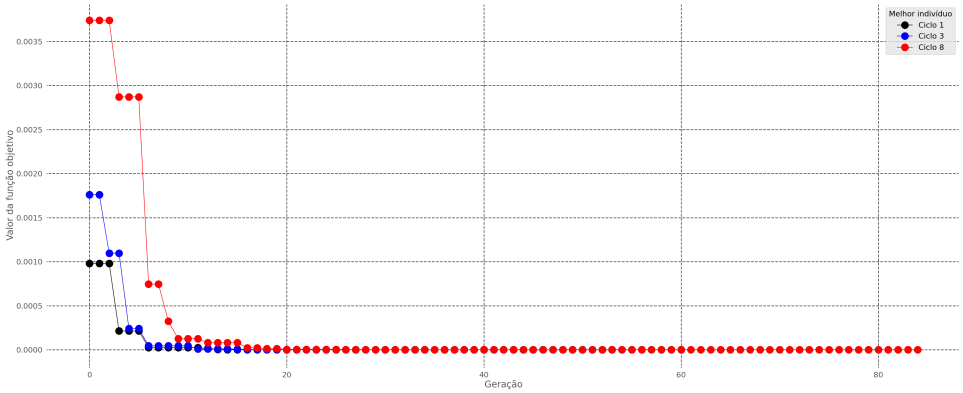
4 RESULTADOS

Neste capítulo, a metodologia proposta é aplicada aos três robôs modelados anteriormente, e os resultados obtidos são apresentados. A escolha desses manipuladores como estudo de caso visa validar a metodologia, desenvolvida de forma genérica para ser aplicável a qualquer robô serial. Inicialmente, é exibido o valor do determinante do jacobiano referente ao melhor indivíduo ao longo das gerações em três ciclos da evolução diferencial. Em seguida, são apresentados os resultados da busca conduzida pela evolução diferencial, juntamente com os valores das variáveis de junta θ_i para cada um dos trinta ciclos realizados. Por fim, são discutidos os padrões de junta identificados e é fornecida uma visualização do manipulador em posturas singulares.

4.1 ROBÔ PLANAR DE TRÊS JUNTAS

Com o intuito de acompanhar a evolução do melhor indivíduo ao longo das gerações, foram obtidos os valores apresentados na Figura 27. Esses dados correspondem a três ciclos distintos. Observa-se que, para o robô de três juntas planares, o algoritmo converge mais rapidamente em relação aos outros manipuladores, os quais serão apresentados na sequência. Para os três ciclos, o determinante atingiu valores na ordem de 1×10^{-6} em menos de 21 gerações, como indicado por quedas acentuadas no início do processo. No total, o ciclo mais longo exigiu 85 gerações e o ciclo mais curto 57 gerações para alcançar um determinante igual a zero.

Figura 27 – Evolução do melhor indivíduo para o robô planar



Fonte: Autor (2024).

Além de acompanhar o desempenho do melhor indivíduo ao longo das gerações, é possível também identificar as posturas singulares e analisar a

convergência da evolução diferencial ao longo dos 30 ciclos definidos para o algoritmo. Esses dados estão apresentados na Tabela 10. Vale destacar que, embora os cálculos tenham sido realizados com a precisão padrão da linguagem Python, os valores foram apresentados na tabela com apenas quatro casas decimais para facilitar a leitura e interpretação.

Tabela 10 – Posturas singulares e resultado da evolução diferencial do robô planar

Ciclo	θ_1	θ_2	θ_3	$Resultado$
1	-0.1838	0.0	-0.6464	0.0
2	-2.0865	0.0	1.8716	0.0
3	2.2885	3.1416	1.4526	0.0
4	2.8666	0.0	-2.3110	0.0
5	1.2111	0.0	-0.1340	0.0
6	-0.0478	0.0	0.013	0.0
7	-2.0004	3.1416	-1.9973	0.0
8	2.4934	-3.1416	-0.3343	0.0
9	-2.2379	-3.1416	1.9436	0.0
10	-2.0704	-3.1416	1.9015	0.0
11	-2.2392	3.1416	-1.5755	0.0
12	-2.4784	-3.1416	1.3303	0.0
13	-1.1709	-3.1416	1.9422	0.0
14	-2.1325	3.1416	1.9110	0.0
15	0.8551	3.1416	1.6847	0.0
16	2.6778	-0.1000	-1.5989	0.0086
17	-0.9384	0.1000	1.8558	0.0086
18	0.0769	-0.1000	-3.0311	0.0086
19	-1.6062	-3.0416	0.7919	0.0087
20	1.4463	-0.1000	-1.3965	0.0086
21	-2.8651	3.0416	-3.0618	0.0087
22	0.279593	-0.1000	-2.1374	0.0086
23	0.6370	3.0416	-0.9797	0.0087
24	1.7655	3.0416	-0.1927	0.0087
25	2.2574	-0.1000	-1.5908	0.0086
26	1.3155	-0.1000	-1.0082	0.0086
27	-1.8226	-0.1000	2.6889	0.0086
28	1.4432	-0.1000	-0.0414	0.0086
29	1.4301	-0.1000	1.5972	0.0086
30	-0.3687	-0.1000	1.7014	0.0086

Fonte: Autor (2024).

Para validar a coerência dos resultados obtidos, é possível utilizar a matriz jacobiana do robô de três juntas, fornecida por Siciliano *et al.* (2010, p. 114). Com o auxílio da biblioteca simbólica *Sympy*, deriva-se a Equação 18, que expressa o determinante do jacobiano desse manipulador. Considerando a_1, a_2 os parâmetros

construtivos e θ_2 o ângulo da segunda junta do robô da Figura 18, com comprimento dos elos descritos na Tabela 2, os únicos valores dentro dos limites estipulados na Tabela 3 que levam o robô a uma postura singular são $\theta_2 = 0$ e $\theta_2 = \pm\pi$.

$$det(\mathbf{J}) = a_1 \cdot a_2 \cdot sin(\theta_2) \tag{18}$$

Essas condições foram precisamente encontradas pelo algoritmo nas primeiras quinze gerações. Com a verificação de padrões implementada, observa-se que nenhuma dessas condições se repetiu mais de cinco vezes. A partir da geração quinze, não foram mais identificadas posturas singulares, e o valor do determinante passou a ser condicionado ao tamanho da penalização definida, evidenciada pela diferença de $\pm 0.1 rad$ na variável de junta θ_2 em relação às condições previamente encontradas. Ao final da execução do algoritmo, são exibidos os padrões de juntas identificados, conforme apresentado na Figura 28.

Figura 28 – Padrões de juntas encontrados para o robô planar

```
J2: 0.0 (Repetições: 5)
J2: 3.141592653589793 (Repetições: 5)
J2: -3.141592653589793 (Repetições: 5)
```

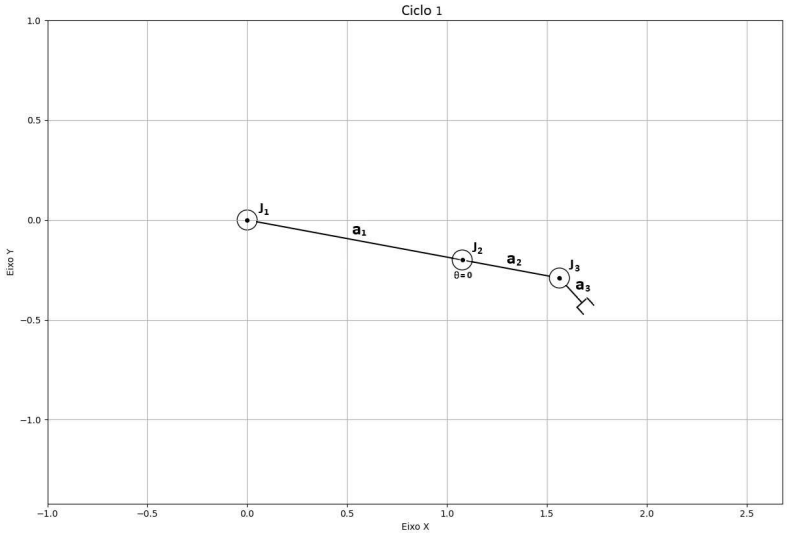
Fonte: Autor (2024).

Além disso, é possível visualizar o manipulador em suas posturas singulares. A Figura 29 ilustra o robô em uma postura singular na condição $\theta_2 = 0$, enquanto a Figura 30 apresenta o robô em posturas singulares para as condições $\theta_2 = \pm\pi$, que essencialmente representam a mesma posição. Essas figuras exemplificam uma das possíveis posturas singulares, pois os valores θ_1 e θ_3 podem variar, já que não influenciam o determinante e, portanto, o robô possui infinitas singularidades. Essa análise não visa identificar novas posturas singulares, mas sim validar o algoritmo de detecção de singularidades em um robô amplamente conhecido e estudado.

4.2 MANIPULADOR ANTROPOMÓRFICO

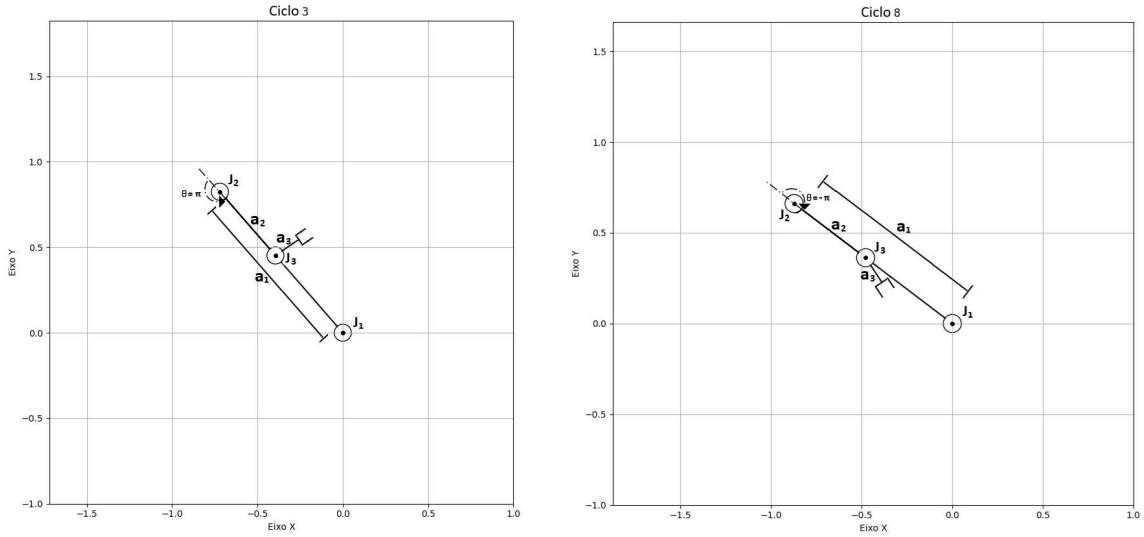
Assim como ocorre no robô planar de três juntas, no manipulador antropomórfico é possível observar a convergência do determinante do jacobiano do melhor indivíduo ao longo das gerações. A Figura 31 ilustra esse comportamento para três ciclos distintos. Esse manipulador também apresenta uma queda acentuada no erro, levando 40 gerações para atingir a ordem de 1×10^{-6} no ciclo mais longo e 12 gerações no ciclo mais curto. No total, o ciclo mais longo exigiu 247 gerações e o mais curto 103 gerações para convergir ao valor zero.

Figura 29 – Postura singular para $\theta_2 = 0$



Fonte: Autor (2024).

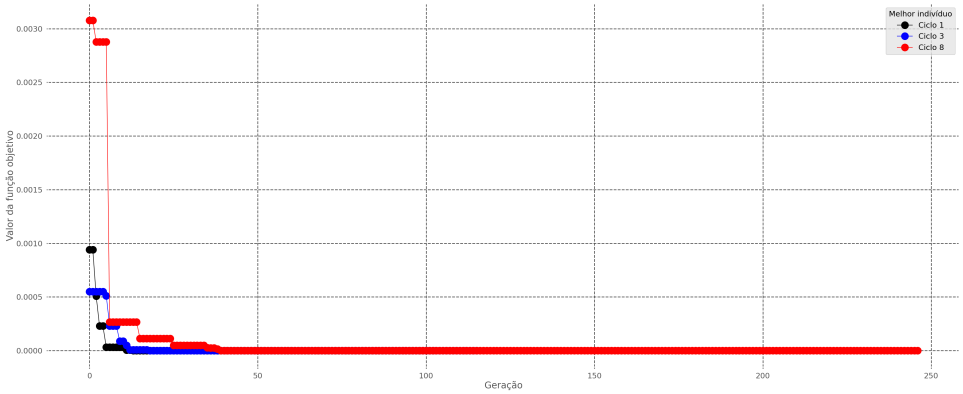
Figura 30 – Posturas singulares para $\theta_2 = \pm\pi$



Fonte: Autor (2024).

A Tabela 11 apresenta o histórico de convergência dos 30 ciclos definidos para o manipulador antropomórfico. Um padrão de junta conhecido em manipuladores com punho esférico é descrito em siciliano para as condições $\theta_5 = 0$ e $\theta_5 = \pi$. Assim, a condição que atende os limites da Tabela 6 é $\theta_5 = 0$, reconhecida pelo algoritmo nos cinco primeiros ciclos. Para avaliar a coerência dos demais resultados obtidos, utiliza-se o conceito do jacobiano geométrico e o desacoplamento de singularidades em Siciliano *et al.* (2010), e os parâmetros de Denavit-Hartenberg definidos na Tabela 4. Com base nesses elementos, deriva-se o determinante do jacobiano, conforme representado pela Equação 19.

Figura 31 – Evolução do melhor indivíduo para o manipulador antropomórfico



Fonte: Autor (2024).

$$det(\mathbf{J}) = a_2(d_4 \cdot c_3 + a_3 \cdot s_3)(a_1 + a_2 \cdot s_2 + a_3 \cdot s_{23} + d_4 \cdot c_{23}) \tag{19}$$

A Equação 19 representa apenas o determinante do jacobiano de posição devido ao desacoplamento. Os termos c_3, s_2 e s_3 representam o $\cos(\theta_3)$, $\sin(\theta_2)$ e $\sin(\theta_3)$ respectivamente. Já os termos s_{23} e c_{23} representam o $\sin(\theta_2 + \theta_3)$ e $\cos(\theta_2 + \theta_3)$. Observa-se, nessa equação, outras duas condições que anulam o determinante. Considerando as variáveis dessa equação conforme descritas na Figura 19 e na Tabela 5, a primeira condição que respeita os limites da Tabela 6 é $\theta_3 = -1.4295$, também identificada logo nos cinco primeiros ciclos. Esses padrões de junta são exibidos ao fim da execução do algoritmo, conforme ilustrado na Figura 32.

Figura 32 – Padrões de juntas encontrados para o manipulador antropomórfico

```
Singularidades de Juntas Encontradas:
J3: -1.4295250894293638 (Repetições: 5)
J5: -4.557018651318856e-11 (Repetições: 5)
```

Fonte: Autor (2024).

A segunda condição que anula o determinante na Equação 19 depende dos parâmetros θ_2 e θ_3, o que impede sua caracterização como uma singularidade de junta. Isso explica por que os resultados da Tabela 11 convergiram para zero, mesmo após o algoritmo identificar os dois tipos de singularidade de junta, diferentemente do robô planar, em que o determinante ficou condicionado ao valor da penalização. Para essa condição, estudos mais detalhados podem ser conduzidos para identificar o padrão desse comportamento.

Um exemplo de estudo que pode ser realizado é observar o comportamento

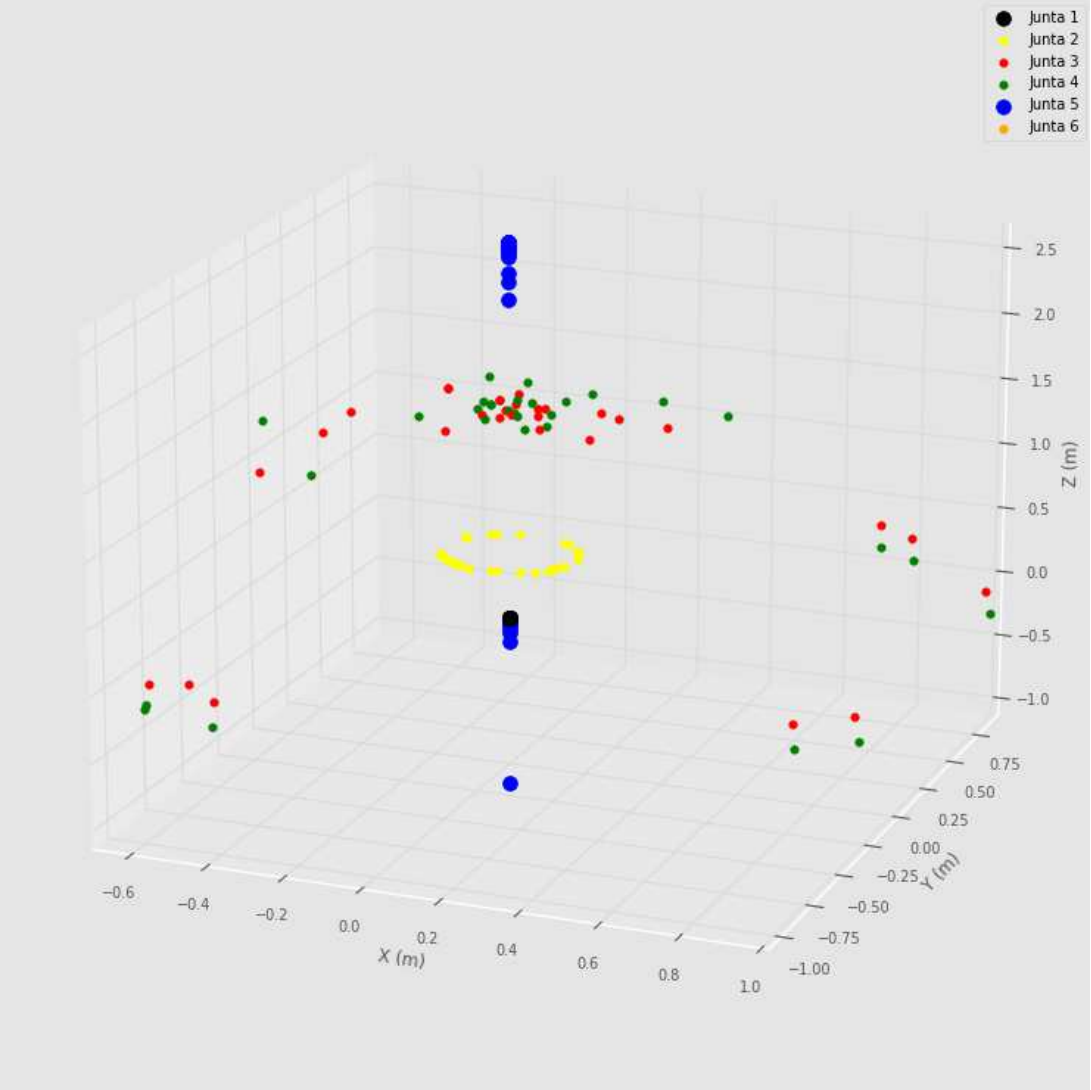
Tabela 11 – Posturas singulares e resultado da evolução diferencial do manipulador antropomórfico

Ciclo	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	Resultado
1	0.5471	-0.0749	-1.4295	4.6896	-0.0000	-3.2317	0.0
2	2.1673	-0.0749	-1.4295	0.9095	-0.0000	5.5829	0.0
3	-0.3419	-0.0749	-1.4295	1.9535	-0.0000	-1.3153	0.0
4	-1.5139	-0.0749	-1.4295	-3.4042	0.0000	-0.8456	0.0
5	-2.0478	-0.0749	-1.4295	-5.4589	-0.0000	2.7265	0.0
6	0.3735	-0.6266	-0.4177	-1.8493	-0.2057	-4.0094	0.0
7	-0.6595	1.8708	1.2245	-0.4027	0.1124	2.0006	0.0
8	-0.6057	0.1417	-1.8396	-6.3951	0.3735	-5.3797	0.0
9	-1.8552	0.0921	-1.7454	5.3045	-0.1191	5.5469	0.0
10	-0.4906	1.9201	1.1116	-4.9599	0.1932	2.7830	0.0
11	-1.8575	1.8712	1.2237	0.3755	-0.1282	3.5093	0.0
12	-2.6352	-0.5540	-0.5470	5.5048	-0.1678	-2.6129	0.0
13	-2.7145	0.2054	-1.9612	0.4304	-0.1601	-4.5727	0.0
14	-2.3036	-0.1496	-1.2895	2.2636	-0.1790	-3.6756	0.0
15	-2.4507	-0.3967	-0.8319	0.8621	-0.5231	-5.8736	0.0
16	-2.0412	1.8911	1.1769	-1.9957	-0.1202	4.2821	0.0
17	-1.4189	-0.2460	-1.1096	1.5726	0.1687	4.6250	0.0
18	-2.1592	-0.2471	-1.1077	-5.0917	-0.1016	-4.9344	0.0
19	-2.5038	2.4830	-0.0233	2.1490	-0.1014	5.9084	0.0
20	0.9907	1.8583	1.2549	3.7390	0.1284	-0.3539	0.0
21	-1.0832	-0.3135	-0.9847	1.9245	0.1413	-3.2925	0.0
22	2.5636	0.0687	-1.7009	2.6766	0.1723	3.6260	0.0
23	0.8791	1.8611	1.2481	-1.2928	-0.1130	0.6181	0.0
24	0.1120	0.0965	-1.7537	-5.3769	-0.1752	5.4140	0.0
25	-2.1701	0.5635	-2.6509	5.6073	0.1396	3.7544	0.0
26	2.5838	0.0658	-1.6954	3.1659	0.2305	5.6154	0.0
27	1.7541	-0.0167	-1.5392	-0.8291	-0.5563	-1.2786	0.0
28	-0.8797	0.0339	-1.6349	-5.4166	-0.1674	4.5117	0.0
29	0.4859	1.8827	1.1965	-5.9493	0.1120	-2.6529	0.0
30	2.0788	-0.1959	-1.2029	6.2969	0.1241	-1.1131	0.0

Fonte: Autor (2024).

da posição cartesiana de cada uma das juntas do manipulador. Com essas posições, é possível gerar uma nuvem de pontos, conforme ilustrado na Figura 33. Nela, observa-se que a Junta 2 apresenta um padrão circular, devido ao fato de sua posição depender apenas do valor da Junta 1. As demais juntas não parecem exibir um padrão, com exceção da Junta 5. Vale ressaltar que a Junta 6 está sobreposta pela Junta 5, a qual representa o centro do punho esférico, o que ocorre por conta da forma como a estrutura foi modelada, e ambas exibem um padrão linear. Conforme descrito por Siciliano *et al.* (2010, p. 120) e pela Equação 19, essas singularidades dependem de θ_2 e θ_3 e ocorrem quando há um alinhamento do punho esférico com o eixo z_0 da Figura 19.

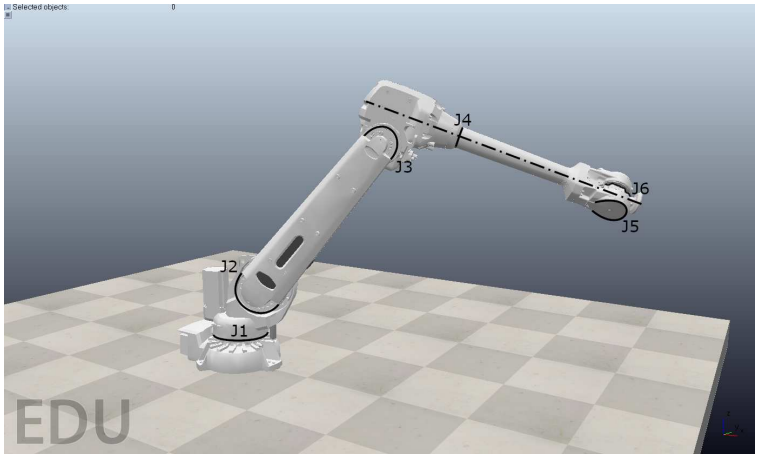
Figura 33 – Posições cartesianas das juntas para 30 ciclos



Fonte: Autor (2024).

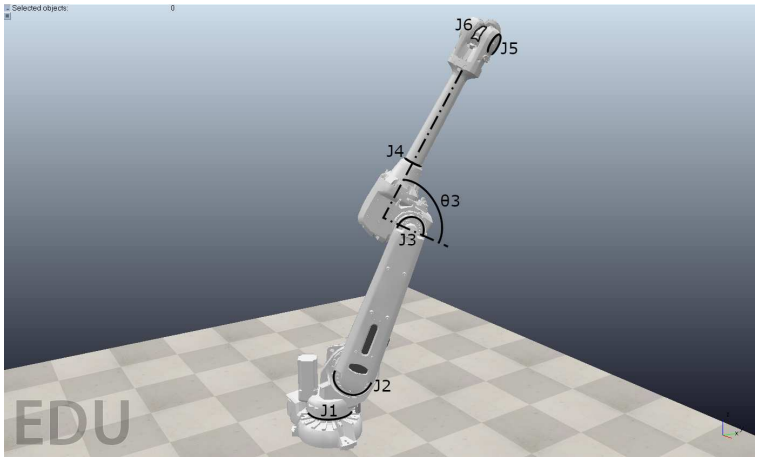
Com o auxílio do software CoppeliaSIM, é possível visualizar o manipulador nas condições de posturas singulares encontradas, com valores aleatórios atribuídos às demais juntas para isolar cada condição singular. A singularidade de punho esférico é ilustrada na Figura 34, enquanto a singularidade de cotovelo é apresentada na Figura 35. Na Figura 36, o manipulador é exibido em um exemplo de singularidade de ombro, onde o centro do punho esférico se alinha com o eixo z_0. Esse manipulador também possui infinitas singularidades, pois θ_1, θ_4 e θ_6 podem variar sem influenciar o determinante de seu jacobiano. Esta análise também não visa identificar novas singularidades, mas sim validar o algoritmo de detecção em um robô conhecido e estudado, porém mais complexo que o robô planar de três juntas.

Figura 34 – Postura singular de punho com $\theta_5 = 0$



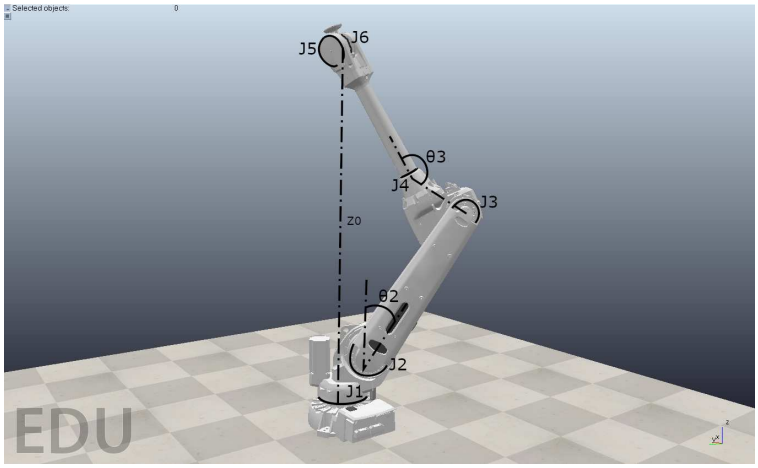
Fonte: Autor (2024).

Figura 35 – Postura singular de cotovelo com $\theta_3 = -1.4295$



Fonte: Autor (2024).

Figura 36 – Singularidade de ombro $\theta_2 = 0.5635rad$ e $\theta_3 = -2.6509rad$

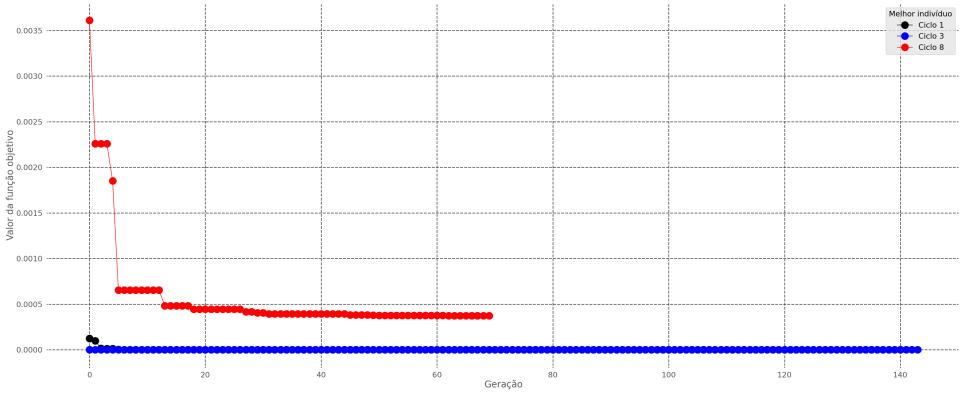


Fonte: Autor (2024).

4.3 MANIPULADOR SNAKE

A Figura 37 apresenta o histórico de convergência do determinante do jacobiano do melhor indivíduo ao longo das gerações do manipulador Snake em três ciclos distintos. Observa-se que, em um dos ciclos, o manipulador não convergiu para zero, o que será discutido em seguida com a apresentação da tabela de posturas singulares. Nos ciclos em que o manipulador convergiu para zero, em um deles foram necessárias 6 gerações, enquanto no outro o algoritmo atingiu a ordem de 1×10^{-6} logo na primeira geração. No total, o ciclo mais longo durou 144 gerações e o mais curto, 70 gerações, até que a evolução diferencial encerrasse a busca.

Figura 37 – Evolução do melhor indivíduo para o manipulador Snake



Fonte: Autor (2024).

A Tabela 12 apresenta os resultados da minimização da evolução diferencial e as posturas identificadas para os 30 ciclos definidos no algoritmo. Um padrão observável refere-se aos valores da penúltima junta, que compõe o punho esférico. Conforme descrito por Siciliano *et al.* (2010), nota-se que, dentro dos limites definidos na Tabela 9, há o padrão de junta $\theta_5 = 0$. Além desse padrão, para validar os outros comportamentos observados, com base no desacoplamento descrito em Siciliano *et al.* (2010), deriva-se a Equação 20, o determinante do jacobiano do manipulador Snake.

$$det(\mathbf{J}) = -a_2 \cdot d_4 \cdot \cos(\theta_2) \cdot \sin(\theta_3) \cdot (a_2 + d_4 \cdot \cos(\theta_3)) \tag{20}$$

A Equação 20 ilustra outras três condições potenciais para anular o determinante. Considerando os valores de acordo com a Figura 21 e a Tabela 8, e dentro dos limites estabelecidos pela Tabela 9, observa-se que os valores $\theta_2 = \pi/2$ e $\theta_3 = 0$ caracterizam posturas singulares. A terceira condição, no entanto, não leva a uma singularidade, pois o termo $a_2 + d_4 \cdot \cos(\theta_3)$ não pode ser anulado devido ao comprimento dos elos. Todas essas condições foram identificadas e exibidas pelo

Tabela 12 – Posturas singulares e resultado da evolução diferencial do manipulador Snake

Ciclo	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	Resultado
1	2.8960	1.5708	-0.0000	-0.8685	0.0000	-3.1121	0.0
2	-3.5269	1.5708	-0.0000	-2.9560	0.0000	-0.8052	0.0
3	3.0413	1.5708	0.0000	-2.8750	0.0000	-0.6412	0.0
4	2.0733	1.5708	-0.0000	-2.4635	0.0000	1.8342	0.0
5	1.0107	1.5708	0.0000	0.9851	0.0000	1.5088	0.0
6	2.4322	1.4708	0.1000	2.2610	0.1000	-1.9212	0.0004
7	2.4539	1.4708	0.1000	1.5313	-0.1000	2.6261	0.0004
8	-0.1545	1.6708	-0.1000	-1.6961	-0.1000	-1.2115	0.0004
9	3.2178	1.4708	-0.1000	1.3612	0.1001	-0.9635	0.0004
10	-0.7409	1.6709	0.1000	1.9743	-0.1000	1.3716	0.0004
11	-2.2823	1.6708	-0.1000	-0.0344	0.1000	0.0520	0.0004
12	-4.3251	1.6708	0.1001	2.2723	-0.1000	1.3475	0.0004
13	-4.2729	1.4707	-0.1000	0.6145	-0.1000	0.1876	0.0004
14	-5.3481	1.4707	-0.1000	2.8815	-0.1001	0.4251	0.0004
15	-2.4797	1.4708	0.1000	1.3847	-0.1000	-0.6241	0.0004
16	-3.1222	1.4708	0.1000	1.5209	0.1000	-0.5258	0.0004
17	-0.0112	1.4708	0.1000	-1.1395	0.1000	-2.0957	0.0004
18	5.8001	1.6708	0.1000	1.9434	0.1000	2.9262	0.0004
19	-3.5076	1.4707	0.1000	-1.8222	-0.1000	-3.0889	0.0004
20	2.0304	1.6708	0.1000	-2.9870	0.1000	-1.8770	0.0004
21	-3.5854	1.6708	-0.1000	1.1842	-0.1001	2.7087	0.0004
22	-6.2660	1.6709	-0.1000	1.3797	0.1000	3.1413	0.0004
23	5.0415	1.4708	-0.1000	2.1379	0.1000	0.2637	0.0004
24	1.1745	1.6708	-0.1000	-0.0649	0.1000	-2.6187	0.0004
25	-2.4331	1.6708	-0.1000	-0.6483	-0.1000	-0.2296	0.0004
26	-0.7389	1.4707	0.1000	-2.2163	-0.1001	-1.6931	0.0004
27	-5.0952	1.6709	-0.1000	0.1733	-0.1000	1.8282	0.0004
28	3.2355	1.6708	-0.1000	0.5275	0.1000	1.7057	0.0004
29	-0.6704	1.6710	0.1000	-1.5473	0.1000	1.1572	0.0004
30	2.0305	1.6708	0.1000	-2.3171	0.1000	2.7566	0.0004

Fonte: Autor (2024).

algoritmo, conforme Figura 38 e Tabela 12. Além disso, assim como ocorre com o robô planar de três juntas, nota-se na Tabela 12 que o determinante está condicionado à magnitude da penalização, também evidenciada pela diferença de $\pm 0.1 rad$ nas juntas θ_2, θ_3 e θ_5 em relação às condições mencionadas.

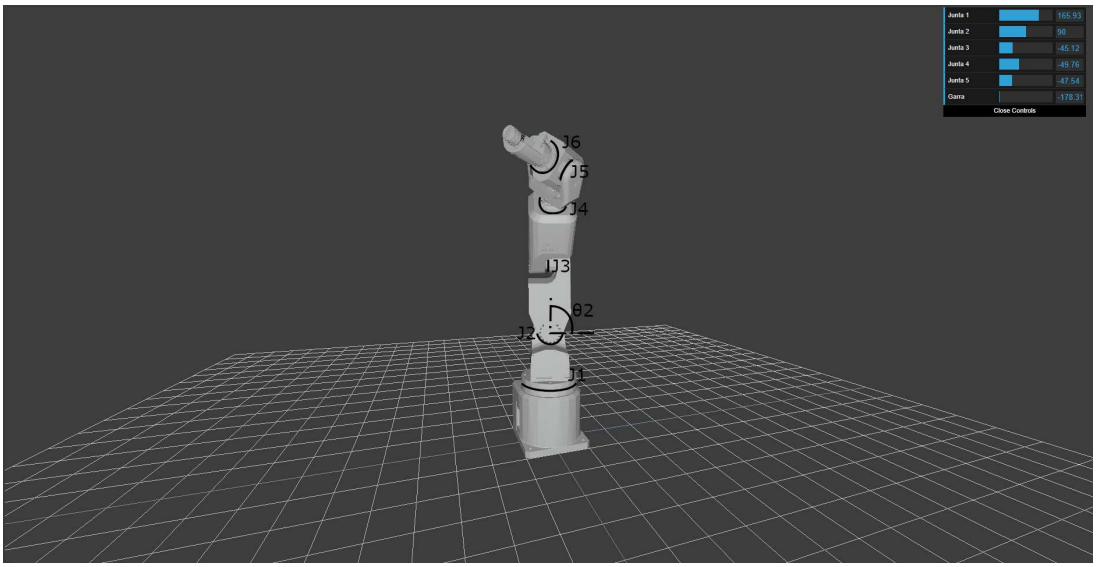
Com o auxílio da simulação em Javascript fornecida pela Voltor, é possível visualizar o manipulador Snake nas diferentes condições de singularidade. Cada figura representa uma condição específica, com valores aleatórios atribuídos às demais juntas para facilitar a identificação de cada caso. A Figura 39 exibe a condição em que $\theta_2 = \pi/2$, enquanto a Figura 40 apresenta a condição em que $\theta_3 = 0$. Já na Figura 41, é exibida a singularidade do punho esférico.

Figura 38 – Padrões de juntas encontrados para o manipulador Snake

```
Singularidades de Juntas Encontradas:
J2: 1.5707963267948994 (Repetições: 5)
J3: 6.6269501462164e-15 (Repetições: 5)
J5: 0.0 (Repetições: 5)
```

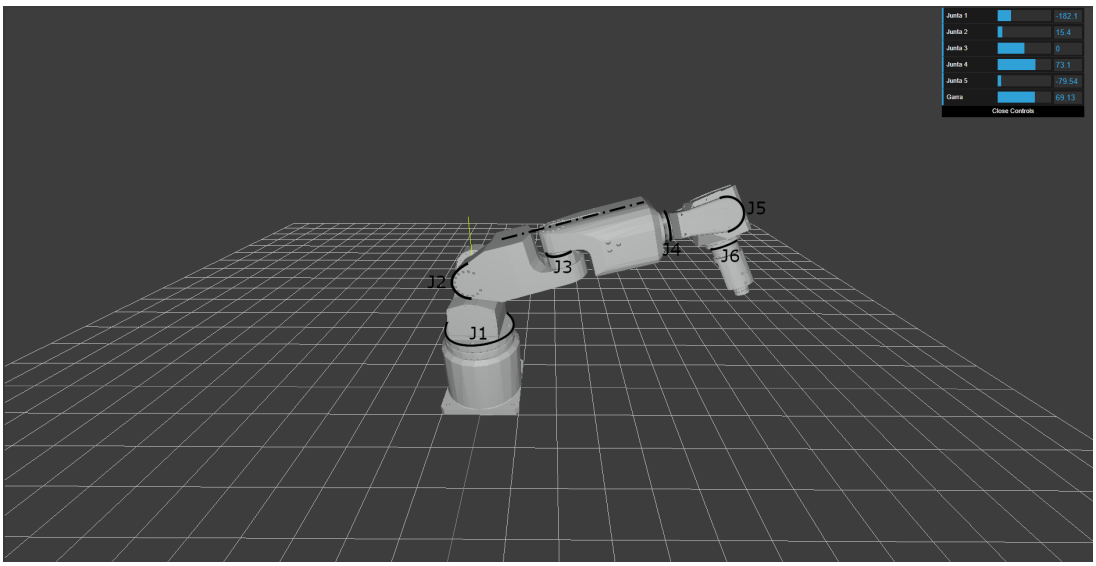
Fonte: Autor (2024).

Figura 39 – Postura singular com $\theta_2 = \pi/2$



Fonte: Autor (2024).

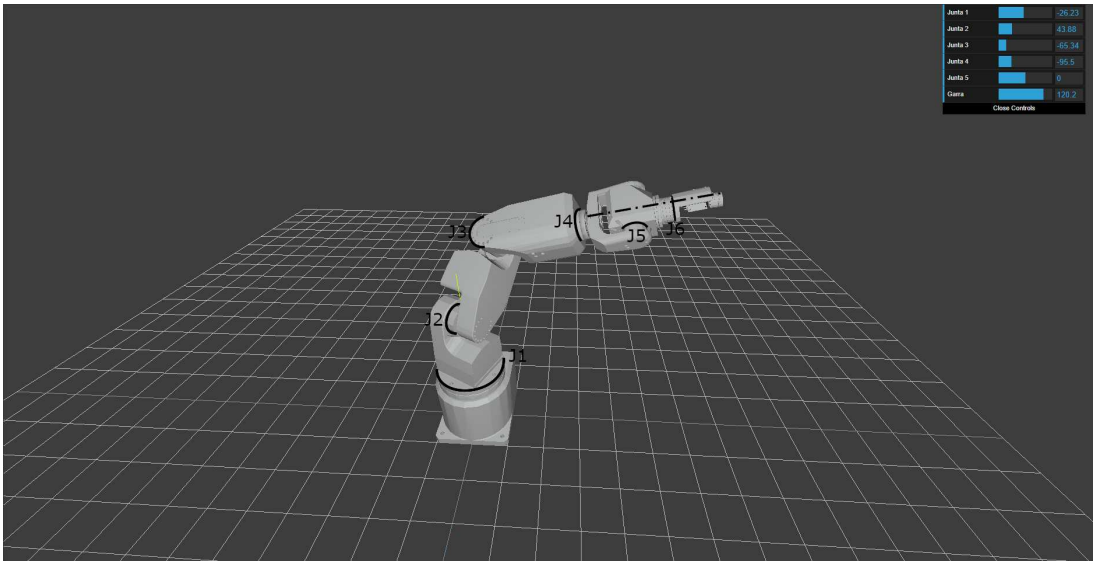
Figura 40 – Postura singular com $\theta_3 = 0$



Fonte: Autor (2024).

Esse manipulador apresenta infinitas singularidades, pois as variáveis θ_1, θ_4 e θ_6 não influenciam o determinante da matriz jacobiana. Assim, essa análise destaca como o algoritmo é capaz de lidar com manipuladores seriais de maneira genérica,

Figura 41 – Postura singular com $\theta_5 = 0$



Fonte: Autor (2024).

inclusive em casos de estrutura não convencional, evidenciando sua flexibilidade. Esses resultados descritos reforçam a eficácia do algoritmo na detecção de singularidades nos três robôs propostos, oferecendo a base para as conclusões gerais de seu desempenho.

5 CONCLUSÕES

Diante da crescente relevância dos manipuladores em processos de automação industrial e da importância do estudo das singularidades nesses robôs, este trabalho buscou desenvolver um algoritmo capaz de detectar posturas singulares em manipuladores industriais seriais. Neste estudo, utilizou-se a técnica de minimização do valor absoluto do determinante da matriz jacobiana por meio do algoritmo de evolução diferencial, buscando identificar automaticamente essas posturas críticas.

Para cumprir esse objetivo, foram implementados os modelos matemáticos que descrevem a estrutura e o comportamento cinemático dos manipuladores de estudo, conforme representado nas Figuras 18, 19 e 20. Esses modelos utilizam o método de Denavit-Hartenberg, que fornece uma representação padronizada das transformações entre as juntas, facilitando a análise e a simulação das posturas singulares de cada manipulador encontradas nas etapas subsequentes.

Em seguida, implementou-se o algoritmo para identificação de posturas singulares por meio do método de evolução diferencial. Esse algoritmo minimiza o valor absoluto do determinante do jacobiano, registrando os padrões de juntas para aplicação de penalizações posteriores. Em cada ciclo, o algoritmo identificou valores de determinante muito próximos de zero ou condicionados pela magnitude da penalização aplicada, conforme representado nas Tabelas 10, 11 e 12.

As tabelas, juntamente com a visualização dos manipuladores nos ambientes Python, Coppelia e Javascript, forneceram simulações que indicaram a coerência da metodologia proposta. No entanto, a confirmação dos resultados baseou-se principalmente nas Equações 18, 19 e 20. Essas equações, extraídas de Siciliano *et al.* (2010) e do método de desacoplamento de singularidades, permitiram verificar que o algoritmo identificou corretamente as condições de singularidade nos manipuladores.

Identificou-se que o robô planar apresenta três condições de singularidade dentro dos limites estabelecidos na Tabela 3: $\theta_2 = 0$ e $\theta_2 = \pm\pi$. O robô antropomórfico também possui três condições de singularidade nos limites da Tabela 6: $\theta_3 = -1.4295rad$, $\theta_5 = 0$, além de infinitos valores de θ_2 em conjunto com θ_3 que resultam no alinhamento do centro do punho esférico com o eixo z_0 desse manipulador. Já o robô Snake apresenta singularidades, nos limites da Tabela 9, nas seguintes configurações: $\theta_2 = \frac{\pi}{2}$, $\theta_3 = 0$ e $\theta_5 = 0$.

Assim, o desenvolvimento deste trabalho demonstrou a viabilidade do algoritmo de otimização por evolução diferencial para caracterização de posturas singulares. No entanto, embora tenha mostrado ser uma abordagem promissora, o algoritmo ainda possui limitações que precisam ser trabalhadas para seu aprimoramento. Além disso,

é possível sugerir novos tópicos para futuras melhorias, os quais serão abordados detalhadamente na próxima seção.

5.1 TRABALHOS FUTUROS

O objetivo desta seção é detalhar as limitações do algoritmo atual e sugerir possíveis melhorias para trabalhos futuros. Embora o algoritmo tenha demonstrado capacidade de identificar corretamente as singularidades das juntas e a busca por evolução diferencial tenha se mostrado eficiente, ainda não oferece a confiabilidade necessária para ser considerado uma ferramenta de estudo única, o que significa que ainda é necessário utilizá-lo como complemento de outras ferramentas. As possíveis melhorias e as razões para essa limitação serão discutidas a seguir.

Uma melhoria que poderia trazer um ganho significativo de flexibilidade ao algoritmo seria a implementação de procedimentos para incluir robôs paralelos. Esses robôs também apresentam posturas singulares que podem gerar problemas de controle e riscos de segurança para os operadores. Assim, uma abordagem comum para identificar essas singularidades é por meio da análise do jacobiano, avaliando a velocidade do manipulador. Para isso, analisa-se o determinante da matriz inversa do jacobiano (Merlet, 2006).

A principal limitação de confiabilidade do algoritmo está relacionada aos valores utilizados para acomodar imprecisões nos cálculos numéricos e o valor da penalização na função objetivo. Esses valores foram definidos com base em uma abordagem empírica de ajustes sucessivos apenas nos três manipuladores abordados nesse trabalho. Para aumentar a confiabilidade, poderiam ser realizados estudos de refinamento desses valores, que poderiam até ser ajustados dinamicamente, uma vez que manipuladores complexos tendem a gerar mais imprecisões do que aqueles com estruturas mais simples.

Outra melhoria significativa ao algoritmo seria implementar padrões de singularidades nos manipuladores, indo além dos padrões específicos das juntas. Assim, a função de detecção de padrões poderia identificar outras configurações que levassem o algoritmo a aplicar penalizações apropriadas. Por exemplo, essa função poderia utilizar a cinemática direta dos manipuladores para calcular as posições cartesianas das juntas e detectar possíveis padrões de singularidade. No caso do manipulador antropomórfico, o algoritmo seria então capaz de identificar todos os padrões de posturas singulares.

Um tópico interessante para trabalhos futuros seria a realização de um estudo comparativo entre diferentes técnicas de minimização e algoritmos meta-heurísticos aplicados ao cálculo do determinante do jacobiano, visando à detecção de singularidades em manipuladores. A comparação de métodos, como algoritmos

de Evolução Diferencial, Otimização por Enxame de Partículas (PSO) e Algoritmos Genéticos (GA), permitiria identificar quais são os mais eficientes em termos de tempo de convergência e precisão. Além disso, a aplicação de métodos capazes de encontrar múltiplos mínimos globais seria uma abordam coerente, uma vez que essa é uma outra limitação deste trabalho, contornada através da penalização de padrões reconhecidos.

REFERÊNCIAS

ABB. **Product manual - IRB 4600**. Zurique, 2024. Disponível em: https://search.abb.com/library/Download.aspx?DocumentID=3HAC033453-001&LanguageCode=en&DocumentPartId=&Action=Launch. Acesso em: 16 nov. 2024.

ABOELNASR, M.; BAHA, H.; MOKHIAMAR, O. Novel use of the monte-carlo methods to visualize singularity configurations in serial manipulators. **Journal of Mechanical Engineering and Sciences**, v. 15, n. 2, p. 7948–7963, jun. 2021.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO 10218-1**: Robôs e dispositivos robóticos — requisitos de segurança para robôs industriais parte 1: Robôs. Rio de Janeiro, 2018.

AZIZI, Y.; ZADEH, N. A. An adaptive control algorithm to improve singularity avoidance in 7-dof redundant manipulators. **Majlesi Journal of Mechatronic Systems**, v. 3, n. 1, mar. 2014.

BÄCK, T. **Evolutionary algorithms in theory and practice**: evolution strategies, evolutionary programming, genetic algorithms. New York: Oxford University Press, 1996.

BECK, F. *et al.* Singularity avoidance with application to online trajectory optimization for serial manipulators. **IFAC-PapersOnLine**, v. 56, n. 2, p. 284–291, 2023.

BILAL *et al.* Differential evolution: A review of more than two decades of research. **Engineering Applications of Artificial Intelligence**, v. 90, p. 103479, abr. 2020.

BOHIGAS, O. *et al.* A general method for the numerical computation of manipulator singularity sets. **IEEE Transactions on Robotics**, v. 30, n. 2, p. 340–351, abr. 2014.

CODE, V. S. **Visual Studio Code documentation**. 2024. Disponível em: https://code.visualstudio.com/docs. Acesso em: 20 out. 2024.

COPPELIA ROBOTICS AG. **CoppeliaSim user manual**: Create. compose. simulate. any robot. Zurique, 2024. Disponível em: https://manual.coppeliarobotics.com/. Acesso em: 16 nov. 2024.

CORKE, P. **Robotics toolbox for Python**. 2024. Disponível em: https://github.com/petercorke/robotics-toolbox-python. Acesso em: 20 out. 2024.

DAS, S.; SUGANTHAN, P. N. Differential evolution: a survey of the state-of-the-art. **IEEE Transactions on Evolutionary Computation**, v. 15, n. 1, p. 4–31, 2011.

GONZÁLEZ, C.; BLANCO, D.; MORENO, L. Optimum robot manipulator path generation using differential evolution. *In*: **Proceedings** of the 2009 IEEE CONGRESS ON EVOLUTIONARY COMPUTATION. Institute of Electrical and Electronics Engineers (IEEE), Trondheim, Noruega, p. 3322–3329, 2009. Disponível em: https://ieeexplore.ieee.org/document/4983366. Acesso em: 6 out. 2024.

GROOVER, M. P. **Automation, production systems, and computer-integrated manufacturing**. 4. ed. USA: Pearson, 2015.

HAYAT, A. A.; SADANAND, R. O. M.; SAHA, S. K. Robot manipulation through inverse kinematics. *In*: **Proceedings** of the 2015 CONFERENCE ON ADVANCES IN ROBOTICS. Association for Computing Machinery, New York, NY, USA, 2015. Disponível em: https://dl.acm.org/doi/10.1145/2783449.2783497. Acesso em: 4 abr. 2024.

JÄMSÄ-JOUNELA, S. L. Future trends in process automation. **Annual reviews in control**, v. 31, n. 2, p. 211–220, jan. 2007.

KARABEGOVIĆ, I.; BANJANOVIĆ-MEHMEDOVIĆ, L. **Industrial robots**: design, applications and technology. New York: Nova Science Publishers, Inc., 2020.

KVERNBERG, P. **Developing force control scenarios on ABB IRB 4600 with camera capture of dynamic motions**. Master of Science in Cybernetics and Robotics — Norwegian University of Science and Technology, 2015.

LIN, C. L. *et al.* Singularity characterization and path planning of a new 3 links 6-dofs parallel manipulator. **European Journal of Control**, v. 14, n. 3, p. 201–212, 2008.

LORA, M. *et al.* Metaheuristic techniques comparison to optimize robotic end-effector behavior and its workspace. **International Journal of Advanced Robotic Systems**, v. 15, out. 2018.

LYNCH, K. M.; PARK, F. C. **Modern robotics**: mechanics, planning, and control. Cambridge: University Press, 2017.

MATPLOTLIB. **Pyplot tutorial**. 2024. Disponível em: https://matplotlib.org/stable/tutorials/pyplot.html. Acesso em: 27 nov. 2024.

MERLET, J.-P. **Parallel robots**: Solid mechanics and its applications. 2. ed. France: Springer, 2006.

NORTON, R. L. **Cinemática e dinâmica dos mecanismos**. Porto Alegre: Mc Graw Hill, 2010.

OETOMO, D.; ANG JUNIOR, M. H. Singularity robust algorithm in serial manipulators. **Robotics and Computer-Integrated Manufacturing**, v. 25, n. 1, p. 122–134, fev. 2009.

PRICE, K. V.; STORN, R. M.; LAMPINEN, J. A. **Differential evolution**: A practical approach to global optimization. Alemanha: Springer, 2014.

QIANG, J.; MITCHELL, C. A unified differential evolution algorithm for global optimization. **IEEE Transactions on Evolutionary Computation**, jun. 2014. Disponível em: https://www.osti.gov/biblio/1163659.

Rebouças Filho, P. P. *et al.* Control of singularity trajectory tracking for robotic manipulator by genetic algorithms. **Journal of Computational Science**, v. 30, n. 1, p. 55–64, nov. 2018.

ROMANO, V. F. **Robótica industrial**: aplicação na indústria de manufatura e de processos. São Paulo: Blucher, 2002.

SCIPY. **Optimization and root finding**. 2024. Disponível em: https://docs.scipy.org/doc/scipy/index.html. Acesso em: 25 out. 2024.

SICILIANO, B. *et al.* **Robotics**: modelling, planning and control. London: Springer, 2010.

SPONG, M. W.; HUTCHINSON, S.; VIDYASAGAR, M. **Robot modelling and control**. Estados Unidos: Wiley, 2005.

STORN, R.; PRICE, K. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. **Journal of Global Optimization**, v. 11, p. 341–359, dez. 1997.

STRANG, G. **Introduction to linear algebra**. 4. ed. USA: Wellesley Cambridge Press, 2009.

VAEZI, M. *et al.* Singularity analysis of 6dof stäubli© tx40 robot. *In*: **Proceedings** of the 2011 IEEE INTERNATIONAL CONFERENCE ON MECHATRONICS AND AUTOMATION. Institute of Electrical and Electronics Engineers (IEEE), China, p. 446–451, 2011. Disponível em: https://ieeexplore.ieee.org/document/5985699. Acesso em: 10 abr. 2024.

VILLALOBOS, J.; SANCHEZ, I.; MARTELL, F. Singularity analysis and complete methods to compute the inverse kinematics for a 6-dof ur/tm-type robot. **Robotics**, v. 11, n. 6, p. 137, nov. 2022.

ZAPLANA, I.; HADFIELD, H.; LASENBY, J. Singularities of serial robots: Identification and distance computation using geometric algebra. **Mathematics**, v. 10, n. 12, p. 2068–2103, 2022.