

Controlo de navegação de um robô móvel diferencial através de algoritmos evolutivos

RICARDO JOSÉ SOUSA BARBOSA
Setembro de 2021

CONTROLO DE NAVEGAÇÃO DE UM ROBÔ MÓVEL DIFERENCIAL ATRAVÉS DE ALGORITMOS EVOLUTIVOS

Ricardo José Sousa Barbosa

Departamento de Engenharia Eletrotécnica

Mestrado em Engenharia Eletrotécnica e de Computadores

Área de Especialização em Automação e Sistemas

Relatório elaborado para satisfação parcial dos requisitos da Unidade Curricular de
Tese/Dissertação do Mestrado em Engenharia Eletrotécnica e de Computadores

Candidato: Ricardo José Sousa Barbosa, Nº 1150685, 1150685@isep.ipp.pt

Orientação científica: Professor Ramiro Sousa Barbosa, rsb@isep.ipp.pt



Departamento de Engenharia Eletrotécnica

Mestrado em Engenharia Eletrotécnica e de Computadores

Área de Especialização em Automação e Sistemas

2021

Agradecimentos

A elaboração da presente dissertação não seria possível sem o apoio de alguns intervenientes. Por essa razão, gostaria de dirigir alguns agradecimentos.

Agradeço ao orientador Ramiro Barbosa pela sua dedicação e disponibilidade presencial e *online* ao longo destes meses de trabalho. Por todas as explicações e revisões realizadas no desenvolvimento técnico e científico do sistema de controlo, com o intuito de melhorar e valorizar o meu projeto.

Por último, o meu maior agradecimento vai para os meus familiares e para a minha namorada, com quem eu partilhei os bons e maus momentos e que em todas as circunstâncias sempre me deram todo o seu apoio, carinho e motivação para concretização desta etapa e, conseqüente, finalização deste curso. Particularmente, agradeço aos meus pais o esforço monetário que realizaram, para que eu frequentasse este curso e conseguisse concretizar este sonho.

Resumo

O planeamento de trajetórias e o controlo de navegação são atividades fundamentais na robótica móvel e possibilitam ao robô deslocar-se de forma autónoma. Dessa forma, este projeto apresenta um sistema de controlo da navegação de um robô móvel de tração diferencial, tendo como objetivo reduzir o erro de posição do robô quando este se move ao longo da trajetória, considerando os limites dinâmicos do mesmo.

O modelo de robô móvel proposto foi implementado em um modelo computacional segundo as relações cinemáticas e dinâmicas do robô KHEPERA IV. A tarefa de movimento do mesmo é uma tarefa de movimento ponto a ponto em um ambiente interno, sendo que o planeamento da trajetória foi realizado usando três algoritmos de controlo evolutivo, especificamente o AG, o PSO e o CS.

A ferramenta de programação utilizada para o desenvolvimento da solução foi o *software* MATLAB/Simulink, uma vez que permite a criação de um ambiente de modelação interativo e a visualização dos resultados obtidos para as simulações em estudo e, ainda, uma *toolbox* de otimização. Uma estrutura de controlo baseada em PID é projetada no *software* para controlo da trajetória descrita a partir dos parâmetros do controlador otimizados pelos algoritmos evolutivos.

Para avaliar o desempenho do sistema de controlo, foram simuladas três trajetórias diferentes e a capacidade de execução dessas trajetórias é analisada numericamente através de três índices de desempenho, sendo estes o IAE, ITAE e ISE.

A aplicação de várias trajetórias no sistema demonstrou que o projeto de controlo foi capaz de seguir a trajetória de referência com um erro aceitável e, por essa razão, o robô móvel tem capacidade para reproduzir qualquer percurso desejado com uma boa aproximação.

Palavras-Chave

Robô móvel diferencial, Algoritmos evolutivos, Controlo PID, Controlo da trajetória.

Abstract

Trajectory control and navigation control are fundamental activities in mobile robotics and allow the robot to move autonomously. Thus, this project presents a navigation control system for a differential wheeled mobile robot, aiming to reduce the robot's position error when it moves along the trajectory, considering its dynamic limits.

The proposed mobile robot model was implemented in a computational model according to the kinematic and dynamic relationships of the KHEPERA IV robot. Its motion task is a point-to-point motion task in an indoor environment, and the trajectory control was performed using three evolutionary control algorithms, specifically AG, PSO and CS.

The programming tool used to develop the solution was the MATLAB/Simulink software, as it allows the creation of an interactive modeling environment and the visualization of the results obtained for the simulations under study, as well as an optimization toolbox. A PID-based control structure is designed in the software to control the trajectory described from controller parameters optimized by evolutionary algorithms.

To evaluate the performance of the control system, three different trajectories were simulated and the capacity to execute these trajectories is numerically analyzed through three performance indices, these being the IAE, ITAE and ISE.

The application of multiple trajectories in the system demonstrated that the control design was able to follow the reference trajectory with an acceptable error and, therefore, the mobile robot can reproduce any desired path with a good approximation.

Keywords

Differential wheeled mobile robot, Evolutionary algorithms, PID control, Tracking trajectory.

Índice

AGRADECIMENTOS.....	I
RESUMO	III
ABSTRACT	V
ÍNDICE	VII
ÍNDICE DE FIGURAS	IX
ÍNDICE DE TABELAS	XIII
ACRÓNIMOS	XV
1. INTRODUÇÃO	1
1.1.CONTEXTUALIZAÇÃO	1
1.2.OBJETIVOS	3
1.3.CALENDARIZAÇÃO.....	3
1.4.ORGANIZAÇÃO DO RELATÓRIO.....	4
2. NAVEGAÇÃO ROBÓTICA MÓVEL COM APLICAÇÃO DE ALGORITMOS EVOLUTIVOS	7
2.1.INTRODUÇÃO	8
2.2.ALGORITMOS EVOLUTIVOS	10
2.2.1.ALGORITMOS GENÉTICOS	10
2.2.2.REDES NEURONAIS ARTIFICIAIS.....	15
2.2.3.OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS	18
2.2.4.OTIMIZAÇÃO POR COLÓNIAS DE FORMIGAS	21
2.2.5.OTIMIZAÇÃO POR COLÓNIA DE BACTÉRIAS.....	25
2.2.6.ALGORITMO DE COLÓNIA DE ABELHAS ARTIFICIAIS	29
2.2.7.ALGORITMO DE PESQUISA DE CUCO	32
3. PLATAFORMAS DE ROBÔS MÓVEIS	37
3.1.TIPOS DE SISTEMAS DE LOCOMOÇÃO	38
3.2.TIPOS DE ROBÔS MÓVEIS.....	40
3.2.1.ROBÔ DIFERENCIAL.....	41
3.2.2.ROBÔ TRICICLO	44
3.2.3.ROBÔ ACKERMAN	47
3.2.4.ROBÔ SÍNCRONO	50
3.2.5.ROBÔ OMNIDIRECIONAL	52

3.3. COMPARAÇÃO ENTRE OS VÁRIOS TIPOS DE ROBÔS MÓVEIS	58
3.4. MODELAÇÃO DINÂMICA DO ROBÔ MÓVEL COM TRAÇÃO DIFERENCIAL	60
4. IMPLEMENTAÇÃO.....	63
4.1. ESQUEMA EM DIAGRAMA DE BLOCOS DA SIMULAÇÃO.....	63
4.2. PLANEAMENTO DE TRAJETÓRIA	65
4.3. MODELO DE CONVERSÃO PARA AS VELOCIDADES ANGULARES DAS RODAS EM SIMULINK	67
4.4. MODELO DO ATUADOR EM SIMULINK	68
4.5. MODELO DINÂMICO EM SIMULINK	70
4.6. MODELO CINEMÁTICO EM SIMULINK	71
4.7. ALGORITMO EVOLUTIVO	72
4.7.1. FICHEIRO <i>SCRIPT</i> PRINCIPAL	72
4.7.2. FUNÇÃO OBJETIVO	74
5. TESTES E RESULTADOS	77
5.1. METODOLOGIA DOS TESTES.....	78
5.1.1. ÍNDICES DE DESEMPENHO.....	79
5.2. RESULTADOS OBTIDOS.....	80
5.3. DISCUSSÃO DE RESULTADOS	89
6. CONCLUSÕES.....	95
6.1. OBJETIVOS REALIZADOS	95
6.2. CONCLUSÃO.....	96
6.3. DESENVOLVIMENTOS FUTUROS	97
REFERÊNCIAS DOCUMENTAIS.....	99
ANEXO A. PROGRAMAÇÃO MATLAB PARA O PLANEAMENTO DA TRAJETÓRIA.....	111
ANEXO B. MODELO DE OTIMIZAÇÃO DO ALGORITMO CS	113
ANEXO C. CÓDIGO MATLAB PARA CÁLCULO DOS ÍNDICES DE DESEMPENHO.....	117
ANEXO D. PARÂMETROS DO CONTROLADOR PROVENIENTES DAS SIMULAÇÕES	119

Índice de Figuras

Figura 1 - Classificação de abordagens de navegação de robôs móveis [7]	10
Figura 2 - Diferença entre Gene, Cromossoma e População	12
Figura 3 - Ponto de cruzamento	13
Figura 4 - Alteração dos genes dos pais	13
Figura 5 - Novos indivíduos a introduzir na população	13
Figura 6 - Processo de mutação	14
Figura 7 - Arquitetura da rede neuronal artificial [7]	16
Figura 8 - Movimento de uma partícula i entre duas posições [25]	20
Figura 9 - Princípio de pesquisa de alimentos pela colónia de formigas [30]	23
Figura 10 - Movimento de uma bactéria	26
Figura 11 - Robô Urbie [61]	39
Figura 12 - Robô Marvin Mark II [62]	39
Figura 13 - Robô Sepios [64]	40
Figura 14 - Cinemática do robô diferencial [66]	41
Figura 15 - Cinemática do robô triciclo [59]	44
Figura 16 - Cinemática do robô Ackerman [59]	48
Figura 17 - Cinemática do robô síncrono [59]	51
Figura 18 - Roda <i>omni</i> ou <i>poly</i> [79]	53

Figura 19 - Roda <i>Mecanum</i> [81]	53
Figura 20 - Movimentos do robô omnidirecional [82]	54
Figura 21 - Cinemática do robô omnidirecional [59]	55
Figura 22 - Modelo dinâmico do robô móvel [66]	61
Figura 23 - Diagrama de blocos	64
Figura 24 - Trajetória em círculo	66
Figura 25 - Trajetória em forma de oito	66
Figura 26 - Trajetória em espiral	67
Figura 27 - Conversão para as velocidades angulares das rodas	68
Figura 28 - Modelo do atuador	70
Figura 29 - Modelo dinâmico	71
Figura 30 - Modelo cinemático	72
Figura 31 - Resultado da simulação da trajetória em círculo	84
Figura 32 - Resultado da simulação para a trajetória em oito	86
Figura 33 - Resultado da simulação para a trajetória em espiral	88
Figura 34 - Binário dos atuadores para a trajetória em círculo	88
Figura 35 - Binário dos atuadores para a trajetória em forma de oito	89
Figura 36 - Binário dos atuadores para a trajetória em espiral	89
Figura 37 - Evolução do IAE durante a simulação	90
Figura 38 - Evolução do ITAE durante a simulação	90

Figura 39 - Evolução do ISE durante a simulação	91
Figura 40 - Evolução da função objetivo através do CS	92
Figura 41 - Evolução da função objetivo através do AG	93
Figura 42 - Evolução da função objetivo através do PSO	93

Índice de Tabelas

Tabela 1 - Calendarização do projeto	4
Tabela 2 - Comparação dos tipos de robôs móveis	58
Tabela 3 - Parâmetros do robô móvel diferencial	65
Tabela 4 - Trajetórias aplicadas ao sistema	65
Tabela 5 - Principais características do atuador	68
Tabela 6 - Parâmetros de inicialização do AG	78
Tabela 7 - Parâmetros de inicialização do PSO	79
Tabela 8 - Parâmetros de inicialização do CS	79
Tabela 9 - Análise estatística das funções objetivo das simulações	81
Tabela 10 - Análise dos índices de erro para a simulação com menor função objetivo	82
Tabela 11 - Trajetória em círculo com AG	119
Tabela 12 - Trajetória em círculo com PSO	120
Tabela 13 - Trajetória em círculo com CS	121
Tabela 14 - Trajetória em forma de oito com AG	122
Tabela 15 - Trajetória em forma de oito com PSO	123
Tabela 16 - Trajetória em forma de oito com CS	124
Tabela 17 - Trajetória em espiral com AG	125
Tabela 18 - Trajetória em espiral com PSO	126

Acrónimos

ABC	–	<i>Artificial Bee Colony</i>
ACO	–	<i>Ant Colony Optimization</i>
AEPSO	–	<i>Area Extension Particle Swarm Optimization</i>
AG	–	Algoritmos Genéticos
BFO	–	<i>Bacterial Foraging Optimization</i>
CC	–	Corrente Contínua
CS	–	<i>Cuckoo Search</i>
DPSO	–	<i>Darwin Particle Swarm Optimization</i>
IAE	–	<i>Integral Absolute Error</i>
ICR	–	<i>Instantaneous Center of Rotation</i>
ITAE	–	<i>Integral of Time Multiplied by Absolute Error</i>
ISE	–	<i>Integral Square Error</i>
MATLAB	–	<i>Matrix Laboratory</i>
PSO	–	<i>Particle Swarm Optimization</i>
P	–	Proporcional
PI	–	Proporcional-Integral
RDPSO	–	<i>Robotic Darwin Particle Swarm Optimization</i>
RPSO	–	<i>Robotic Particle Swarm Optimization</i>

1. INTRODUÇÃO

Neste capítulo é realizada a apresentação do projeto desenvolvido explicando o seu enquadramento com o curso em que está inserido e com os temas que serão abordados. Seguidamente, são identificados os objetivos a alcançar no final do projeto, bem como a calendarização das tarefas executadas. O capítulo termina com uma breve descrição da organização do relatório, indicando os temas retratados em cada capítulo.

1.1. CONTEXTUALIZAÇÃO

Este projeto foi realizado no âmbito da unidade curricular de Tese/Dissertação do 2º ano do Mestrado em Engenharia Eletrotécnica e de Computadores na área de especialização de Automação e Sistemas, do Instituto Superior de Engenharia do Porto. A unidade

curricular tem como propósito integrar e aprofundar conhecimentos das várias áreas científicas estudadas ao longo do curso.

O tema escolhido consiste na análise e implementação de algoritmos de controlo evolutivos com o objetivo de realizar o controlo da navegação do modelo de um robô móvel. Para isso, será desenvolvida uma aplicação em MATLAB/Simulink que possibilita a modelação do robô e a visualização da sua navegação dependendo do algoritmo aplicado.

Atualmente, a robótica é uma área de vasta pesquisa, uma vez que um robô pode ser mais preciso, rápido e forte que um humano, porém ele apresenta pouca ou praticamente nenhuma inteligência. Ainda assim, com o recurso a uma programação adequada, o robô consegue efetuar atividades complexas com sucesso. Especificamente, os robôs móveis são capazes de realizar uma maior variedade de tarefas mais complexas que os robôs industriais e, por essa razão, dentro da robótica, são alvos de maior atenção.

A capacidade de um robô móvel planear o seu movimento e interagir com o ambiente, sem intervenção humana, é essencial para caracterizar a sua autonomia. Desta forma, a navegação é uma das tarefas mais importantes e complexas no problema de locomoção de robôs. Conceitualmente, a navegação baseia-se na condução de um robô por um percurso no espaço de trabalho que possa ser percorrido sem colisão com os obstáculos e que possibilite ao robô deslocar-se entre a posição inicial e a posição final. Esta complexa tarefa pode ser dividida em subproblemas tais como a localização do robô no espaço de pesquisa, o planeamento de trajetórias e a execução destas corretamente.

Em particular, este projeto é focalizado na navegação de um robô móvel com transmissão diferencial, isto é, um veículo com duas rodas traseiras motrizes e uma roda dianteira de eixo livre. Assim sendo, para o planeamento de trajetórias, aplicaram-se várias soluções baseadas em algoritmos de controlo evolutivos e nas restrições cinemáticas e dinâmicas do robô, passando por uma análise de desempenho através do cálculo dos índices que permitem verificar o movimento do robô em comparação com o pretendido.

1.2. OBJETIVOS

O principal objetivo deste trabalho é a implementação de um conjunto de algoritmos de controlo evolutivos capazes de determinarem os ganhos do controlador, que proporcionam a um robô móvel diferencial a capacidade de navegação desde um ponto inicial até ao destino. Dada a complexidade inerente a este projeto, foram definidos diversos objetivos adjacentes, que se encontram apresentados de seguida:

- Levantamento do estado da arte, como forma de contextualização sobre o tema;
- Análise dos vários tipos de robôs móveis terrestres com rodas, com o objetivo de estudar o modelo cinemático de cada um;
- Elaborar e testar o modelo do robô móvel com tração diferencial em ambiente MATLAB/Simulink;
- Definir o tipo de controlador a ser utilizado na malha de realimentação do sistema de controlo, bem como na malha de realimentação interna do mesmo.
- Implementar os algoritmos de controlo no sistema com recurso a ferramentas do *software* MATLAB;
- Analisar os resultados obtidos através do cálculo de índices de desempenho, que permitam verificar o correto movimento do robô na execução das trajetórias pretendidas e comparação dos diferentes algoritmos de controlo.

1.3. CALENDARIZAÇÃO

O cronograma da Tabela 1 caracteriza o desenvolvimento do presente projeto, onde são apresentadas as respetivas tarefas, bem como o seu período de execução e o momento de finalização.

Tabela 1 - Calendarização do projeto

Tarefas	Meses												
	Set	Out	Nov	Dez	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set
Revisão bibliográfica de algoritmos evolutivos													
Revisão bibliográfica de plataformas robóticas													
Estudo e implementação do modelo do robô móvel diferencial													
Implementação das trajetórias de referência													
Implementação dos algoritmos evolutivos no sistema de controlo													
Simulação e testes													
Aquisição de resultados e conclusão													
Elaboração do relatório													

1.4. ORGANIZAÇÃO DO RELATÓRIO

Esta dissertação está dividida em seis capítulos, seguindo os anexos.

No primeiro capítulo, Introdução, é realizada uma pequena contextualização sobre o tema no qual o projeto se enquadra, além da motivação que levou à realização do projeto e os objetivos que se pretendem atingir. Para terminar este capítulo, é apresentada a calendarização do projeto durante a sua fase de execução.

Seguindo o capítulo dois, Navegação Robótica Móvel com Aplicação de Algoritmos Evolutivos, é realizada uma apresentação ao tema de navegação robótica usando algoritmos evolutivos, incluindo a exposição de alguns projetos baseados nos diferentes algoritmos.

No terceiro capítulo, Plataformas de Robôs Móveis, são descritos os vários tipos de sistemas de locomoção, com ênfase em sistemas móveis terrestres. Neste capítulo explora-

se as diferentes configurações das plataformas de robôs móveis com rodas, englobando as equações do modelo cinemático. Seguidamente, compara-se as configurações apresentadas por meio da enumeração das principais capacidades de desempenho das mesmas. No fim do capítulo, são expostas as equações do modelo dinâmico referente à plataforma de robô móvel escolhida para o desenvolvimento do projeto.

No capítulo quatro, Implementação, é descrita toda a implementação do sistema de controlo do robô móvel. O capítulo inicia com a explicação do esquema em diagrama de blocos da simulação e, em seguida, é detalhada a forma como cada modelo foi desenvolvido no *software* MATLAB/Simulink, bem como a implementação dos algoritmos evolutivos no sistema de controlo.

No capítulo cinco, Testes e Resultados, são apresentados os resultados obtidos ao longo dos testes efetuados ao sistema de controlo. Primeiramente, é elaborada uma descrição dos testes realizados e dos índices de desempenho utilizados para medir a qualidade da solução. Seguidamente, é realizada uma análise numérica dos resultados alcançados nas simulações e, dessa forma, destacam-se quais os algoritmos evolutivos com melhor desempenho para cada trajetória. Na fase seguinte, são ilustrados graficamente os melhores resultados do sistema para os diferentes cenários. No fim do capítulo, é desenvolvida uma discussão de resultados acerca da estabilização dos diferentes algoritmos de controlo.

As principais conclusões deste projeto são descritas no capítulo 6, Conclusões, assim como a especificação dos objetivos atingidos e as melhorias futuras a desenvolver como temas a complementar o projeto.

Por último, é apresentado em Anexo a informação complementar ao projeto, como o código implementado em MATLAB e os ganhos do controlador para todas as simulações realizadas.

2. NAVEGAÇÃO ROBÓTICA MÓVEL COM APLICAÇÃO DE ALGORITMOS EVOLUTIVOS

No presente capítulo é inicialmente feita uma contextualização à navegação robótica móvel, explicando alguns conceitos básicos sobre o tema. De seguida, são apresentados alguns algoritmos evolutivos, referindo em certos casos as respetivas analogias com os sistemas biológicos em que são baseados, bem como as equações matemáticas que caracterizam o seu procedimento. Adicionalmente, em cada algoritmo são apresentados casos de estudo que expõem a sua implementação na resolução de um dado problema de controlo.

2.1. INTRODUÇÃO

A navegação de robôs móveis caracteriza-se pela habilidade de um robô locomover-se de uma posição inicial até à posição de destino com base nas informações da sua posição física e do ambiente onde se encontra. Esta técnica tem por base os seguintes critérios de otimização [1]:

- Minimizar o tempo de execução;
- Minimizar a energia (ou esforço do atuador);
- Minimizar as colisões.

Nomeadamente, um sistema de navegação pode ser definido basicamente por três questões: “Onde estou?”, “Para onde vou?”, “Como chego lá?” [2] que resumem as três competências fundamentais da navegação, sendo estas [3]:

1. Auto-localização;
2. Planeamento da trajetória;
3. Construção e interpretação de mapas.

A competência de localização representa a capacidade do robô de determinar a sua posição e orientação no ambiente. Seguidamente, no planeamento da trajetória utilizam-se as informações da localização da posição atual do robô e da posição do objetivo para traçar um caminho que deverá ser percorrido pelo robô móvel. A interpretação dos mapas permite que através das informações vindas dos sensores seja possível determinar a posição do robô no ambiente e com o recurso das informações do planeamento seja possível atingir o objetivo esperado [3].

No desenvolvimento de robôs móveis a navegação é o principal problema enfrentado, uma vez que exige conhecer a posição do robô com um nível adequado de precisão. Para isso, a tarefa de navegação pode ser decomposta em tarefas básicas como mapeamento e planeamento de trajetórias [4].

O mapeamento traduz-se na representação do ambiente através da utilização de mapas obtidos pelo sistema de sensores incorporados na estrutura do robô ou previamente armazenados. Enquanto, o planeamento de trajetória atribui aos robôs a competência de idealizar os seus próprios movimentos, sem a necessidade da interferência humana. Esta

tarefa de planeamento pode ser considerada como uma simplificação na qual se estuda em profundidade as principais questões, referidas anteriormente, de forma isolada antes de se analisar as dificuldades adicionais. Por conseguinte, considera-se que o único objeto móvel no ambiente é o robô, bem como este ser o único objeto rígido, ou seja, não apresenta partes móveis, como braços, pernas, atuadores finais, entre outros e a sua deslocação é apenas limitada pelos obstáculos presentes no espaço de trabalho. Em suma, o planeamento da trajetória em navegação robótica pode ser definido como a procura de um caminho contínuo livre de obstáculos que guie o robô desde o início até à configuração do objetivo [4][5].

De um modo geral, o planeamento de trajetórias tem duas categorias principais: o planeamento de trajetórias global e o planeamento de trajetórias local, estas vertentes devem ser utilizadas em simultâneo pelo robô para que se possa ter um melhor desempenho. O planeamento global significa que o ambiente é completamente conhecido pelo robô móvel e estático, permitindo-lhe delinear um trajeto por um local mesmo que este não seja detetado pelos seus sensores. Por outro lado, na navegação local o ambiente é total ou parcialmente desconhecido para o robô móvel que controla o seu movimento através dos dados adquiridos pelos sensores [4][6]. As abordagens de planeamento de trajetórias para um ambiente conhecido são baseadas em abordagens clássicas, como Decomposição de Células, *Roadmaps*, Campo Potencial Artificial. No entanto, estes algoritmos são tradicionais e possuem inteligência limitada. Por sua vez, as abordagens de navegação local são consideradas abordagens reativas, pois são mais inteligentes e capazes de executar um plano autonomamente. Exemplos deste tipo de abordagens são Algoritmos Genéticos, Algoritmo de Colónia de Formigas, Otimização de Enxame de Partículas, entre outros.

A Figura 1 ilustra a classificação de algumas abordagens de navegação de robôs móveis.

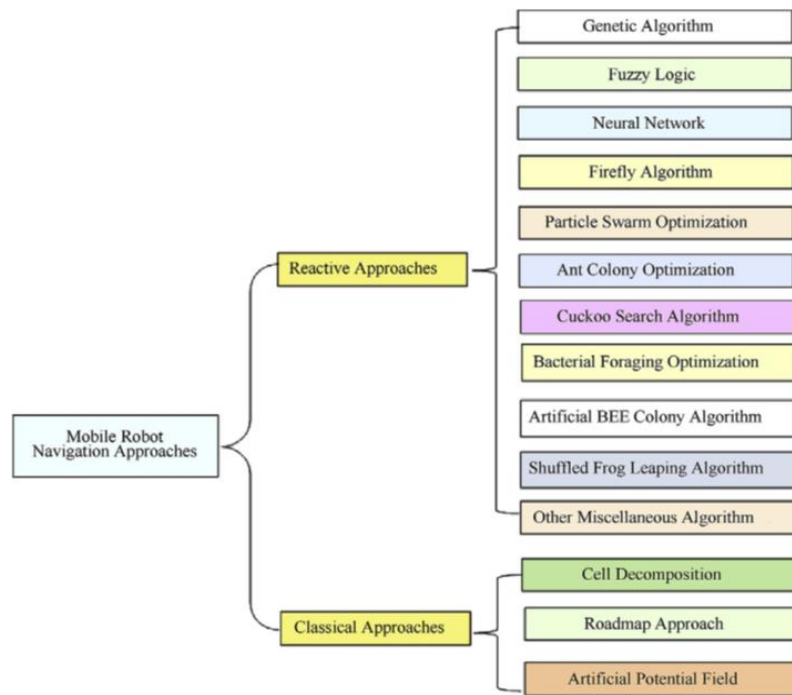


Figura 1 - Classificação de abordagens de navegação de robôs móveis [7]

2.2. ALGORITMOS EVOLUTIVOS

Os algoritmos evolutivos são técnicas de inteligência artificial capazes de resolver problemas da qual os algoritmos clássicos teriam muita dificuldade ou até não seriam capazes de resolver. Muitas das técnicas são baseadas no funcionamento de sistemas biológicos e da forma como estes resolvem problemas, sendo depois transpostos para solucionar problemas de controlo.

2.2.1. ALGORITMOS GENÉTICOS

Algoritmos genéticos (AG) são algoritmos populares de otimização baseados em pesquisa, que seguem o princípio de seleção natural e da genética de Charles Darwin [8]. De acordo com a teoria de Darwin, o princípio de seleção privilegia os indivíduos mais aptos para serem selecionados para reprodução, com o intuito de produzir descendentes que herdaram as características para a próxima geração. Por sua vez, os indivíduos com mais

descendentes têm mais probabilidade de preservarem o seu código genético nas gerações seguintes. O código genético caracteriza a identidade de cada indivíduo e é integrado nos cromossomas [9].

A aplicação desta ferramenta em campos da ciência da computação, para a pesquisa de uma solução ótima de um determinado problema, foi apresentada pela primeira vez por Holland no ano de 1975 [7]. Nos algoritmos computacionais, os algoritmos genéticos são aplicados como uma simulação por computador, em que uma população de possíveis soluções aleatórias codificadas em cromossomas artificiais é selecionada para sofrer uma evolução por meio de sucessivas gerações em busca de melhores soluções. A cada geração, a aptidão de todas as soluções é avaliada, posteriormente, alguns indivíduos são escolhidos para progredir para a próxima geração e sofrem cruzamento ou mutação para formarem uma nova população. Esta nova população é utilizada como entrada na iteração seguinte do algoritmo [9][10].

Neste algoritmo um cromossoma artificial é uma estrutura de dados que ilustra uma das possíveis soluções dentro do espaço de pesquisa e o seu processo de evolução é composto pelas seguintes etapas:

População inicial

O algoritmo genético inicia-se a partir de um conjunto de indivíduos criados chamado População. Cada indivíduo caracteriza uma solução possível para o problema de otimização em questão [8].

Um indivíduo é descrito por um conjunto de variáveis conhecidas como genes, que por sua vez são agrupados em uma *string*, a fim de formar um cromossoma, ou seja, uma solução.

No AG, o cromossoma de um indivíduo é uma cadeia binária, isto é, uma sequência de 0's e 1's com um comprimento fixo e cada *bit* desta sequência é inicializado aleatoriamente [11].

A Figura 2 esclarece os conceitos integrantes desta etapa.

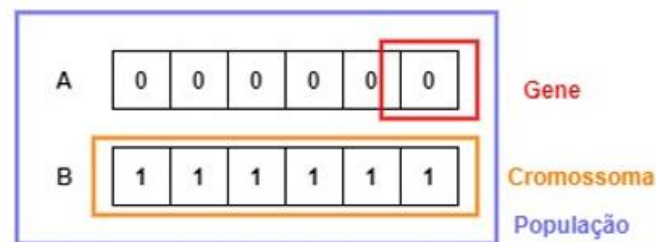


Figura 2 - Diferença entre Gene, Cromossoma e População

Avaliação

A avaliação é realizada, durante o processo evolutivo, através de uma função que representa cada problema especificadamente e tem como objetivo determinar a aptidão de cada indivíduo na população. A aptidão é dimensionada por meio de uma pontuação de condicionamento físico, onde a probabilidade, p_r , de um individuo ser selecionado para a reprodução é apoiada [9].

Seleção

O processo de seleção em algoritmos genéticos é baseado no nível de aptidão para selecionar os indivíduos e permitir que eles transmitam os seus genes para a próxima geração através da reprodução [9].

Dessa forma, os indivíduos mais aptos têm maior probabilidade de serem escolhidos para a reprodução, ou seja, a probabilidade de uma solução i ser selecionada é proporcional à sua aptidão.

Cruzamento

O processo de cruzamento é considerado o operador genético predominante de um algoritmo genético. Pares de progenitores são escolhidos aleatoriamente durante a reprodução, tendo por base o valor da aptidão, para se recombinarem sexualmente através do operador de cruzamento com uma probabilidade p_c , com o objetivo de criar indivíduos devido à troca de genes entre os seus progenitores. Por essa razão, os descendentes formados serão diferentes dos seus pais, mas com características genéticas herdadas de ambos [9].

Para explicação deste processo, considere o ponto de cruzamento da Figura 3.

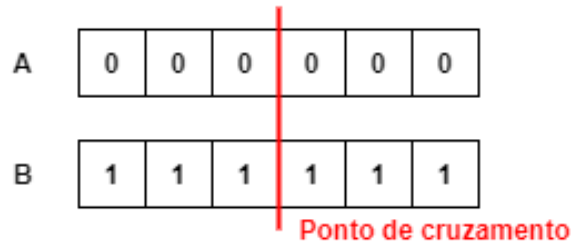


Figura 3 - Ponto de cruzamento

O ponto de cruzamento divide os dois progenitores em uma posição aleatória, de seguida os filhos são criados devido à alteração dos genes dos pais entre si até atingir o ponto de cruzamento [8] (Figura 4).

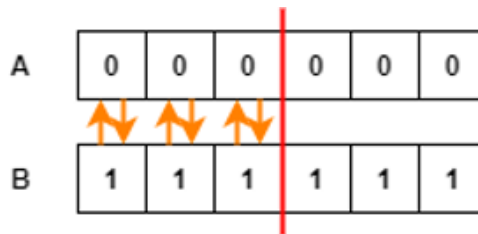


Figura 4 - Alteração dos genes dos pais

Por fim, estes novos indivíduos gerados, Figura 5, são introduzidos na população.

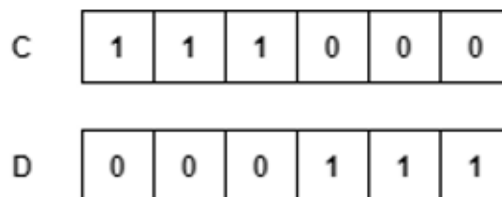


Figura 5 - Novos indivíduos a introduzir na população

Mutação

Em alguns indivíduos formados a partir do operador de cruzamento são submetidos à operação de mutação com uma probabilidade dada pela taxa de mutação p_m [9]. O valor desta probabilidade geralmente é baixo, uma vez que o operador de mutação é um operador genético secundário [10].

O operador de mutação é necessário para aumentar a diversidade genética na população e impedir a convergência prematura, invertendo arbitrariamente o conteúdo de um ou mais genes escolhidos de um cromossoma [7]. Esta inversão modifica o código genético dos indivíduos, possibilitando a introdução de elementos com novas características na população.

A Figura 6 demonstra o processo de mutação.

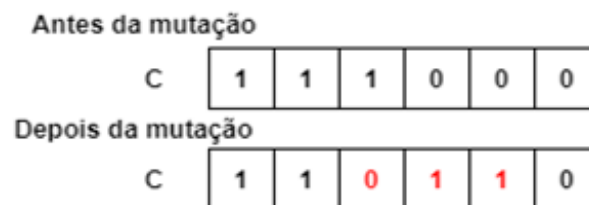


Figura 6 - Processo de mutação

Em termos de implementação, a mutação é benéfica na medida em que permite que a probabilidade de se chegar a todos os pontos do espaço de busca nunca seja zero.

Encerramento

O algoritmo genético termina quando se verifica que as condições de encerramento da evolução foram atingidas, isto é, se a população convergiu e não introduz descendentes com características significativamente diferentes da geração anterior. Neste caso, o algoritmo gerou um conjunto de soluções adequadas para o problema em estudo [8].

Atualmente, os algoritmos genéticos podem ser aplicados em diversas áreas da ciência e tecnologia, incluindo a navegação por robôs. A aplicação de AG em problemas de navegação de robôs móveis foi iniciada pelos autores Takanori Shibata e Toshio Fukuda para um ambiente estático. No artigo *"Intelligent Motion Planning by Genetic Algorithm with Fuzzy Critic"* [12], os autores propuseram uma estratégia que conjuga um algoritmo genético para otimização do planejamento de trajetórias com lógica difusa na função de aptidão física para avaliar o movimento planejado. Esta função de aptidão física é designada de Critério Difuso e avalia planos, similar a populações no AG, em relação a múltiplas variáveis. A análise desta estratégia foi realizada por meio de simulações que determinam

um caminho para o robô móvel seguir desde o ponto inicial até ao ponto de destino, com a presença de apenas um obstáculo poligonal no espaço de pesquisa.

A navegação de robôs móveis em um ambiente desconhecido com obstáculos alteráveis é apresentada por Pu Shi e Yujiee Cui no estudo denominado *“Dynamic Path Planning for Mobile Robot Based on Genetic Algorithm in Unknown Environment”* [13]. O comprimento das cadeias binárias que representam os pontos de uma trajetória, bem como o tamanho do espaço de pesquisa determina o tempo computacional para uma função de aptidão. Dessa forma, para diminuir o comprimento das cadeias, é apresentado neste artigo um método de codificação da trajetória em que a função de aptidão e os operadores genéticos são criados no algoritmo. Esta técnica de codificação exclusiva diminui a complexidade computacional convencional do algoritmo genético e através da transformação dos dados bidimensionais em dados unidimensionais é possível reduzir o tamanho do espaço de pesquisa, aumentando assim a velocidade de execução da procura. Por fim, os resultados das simulações realizadas pelos autores afirmaram que o algoritmo genético é bastante eficaz sob complexos ambientes dinâmicos.

2.2.2. REDES NEURONAIS ARTIFICIAIS

As redes neurais artificiais são técnicas computacionais para solucionar problemas que, tal como o nome indica, são inspiradas na estrutura neuronal biológica e na forma como o cérebro analisa e processa informações [14][15].

Este algoritmo de otimização é composto da mesma forma que o cérebro humano, ou seja, incorpora nós de neurónios interconectados como uma teia. Neste caso, os neurónios artificiais são chamados de unidades de processamento e contemplam unidades de entrada e de saída. Essas unidades são conectadas umas às outras por canais de comunicação, com o intuito de simular as sinapses que ocorrem em uma rede de neurónios biológica e cada conexão possui ainda associado um determinado peso, que determina a influência na saída da unidade. Quando são detetados sinais à entrada do neurónio, estes são multiplicados pelo seu peso e o valor real do estímulo é determinado através da soma ponderada de todos os sinais. Em seguida, o resultado da função soma é aplicado à função

de ativação que decide se o sinal deve continuar para a etapa seguinte ou não, conforme um valor limite estabelecido [14][15][16].

O procedimento descrito anterior pode ser representado pela equação (2.1) e pela Figura 7. Se o valor real do estímulo ultrapassar o limite μ da função de ativação, o neurónio ativa o valor 1 na saída binária y , caso contrário a saída mantém-se em $y = 0$. Esta comparação entre os dois valores é realizada através da função de Heaviside, isto é, $\theta(x) = 1$ se $x \geq 0$ e $\theta(x) = 0$ se $x < 0$ [17].

$$y = \theta\left(\sum_{j=1}^D w_j x_j - \mu\right) \quad (2.1)$$

Em que, D é o número total de entradas e w_j representa o peso do neurónio processador x_j .

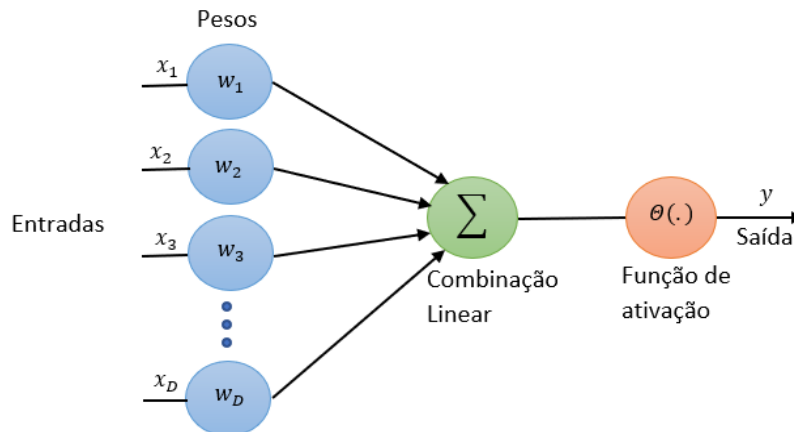


Figura 7 - Arquitetura da rede neuronal artificial [7]

Arquiteturas neuronais são tipicamente organizadas em camadas, onde usualmente podemos definir 3 grupos. Mais precisamente, as camadas de entrada que é onde os padrões são fornecidos às unidades de processamento, as camadas intermédias ou ocultas que é onde é realizado o processamento através das conexões ponderadas e, por fim, a camada de saída que conclui e apresenta o resultado à rede para ser utilizado [15].

O método de propagação da informação recebida é uma categorização fundamental da rede neuronal e podem ser distinguidas duas abordagens, sendo estas as redes diretas ou também chamadas *feedforward* e as redes realimentadas. No caso das redes diretas, o

fluxo de informação é unidirecional através das camadas intermédias até à saída. Por outro lado, as redes realimentadas têm ligações entre os neurónios de forma cíclica sem restrições, isto é, ocorre realimentação do sinal de saída para a entrada [17][16].

Assim que definida a rede neuronal, esta tem de ser treinada para reconhecer padrões de dados. A habilidade de aprender e melhorar o seu desempenho é a propriedade mais importante de uma rede neuronal e o processo de treino é realizado através de um processo iterativo de modificação dos pesos entre as unidades de processamento. O algoritmo de ensino da rede é representado por um conjunto de regras estipuladas para a solução de um problema, estas regras são chamadas de retropropagação, isto é, uma propagação reversa de erros que permite às redes aperfeiçoar suas camadas intermédias em situações em que o resultado não corresponde ao esperado. Por sua vez, existem vários tipos de algoritmos de aprendizagem para diferentes modelos de redes neuronais, sendo estas [15][18][16]:

- Aprendizagem supervisionado: a rede neuronal utiliza um agente externo que comunica à rede a resposta pretendida tendo em consideração o padrão de entrada;
- Aprendizagem não supervisionado: a rede neuronal opera sobre um padrão de entrada sem um treino prévio e fornece na saída o conjunto dos padrões mais semelhantes.

No que toca aos algoritmos de aprendizagem supervisionado, o modelo mais usual é o *Perceptron*, que consiste em ajustar os pesos e os limites das suas unidades de processamento através da abordagem de retropropagação. Por outro lado, para as redes neuronais artificiais de aprendizagem não supervisionado o modelo mais comum é o *Kohonen*, que é considerado um mapa auto-organizável capaz de reconhecer aproximações entre os grupos de dados e formar conjuntos dos pontos similares, com o intuito de obter centros de processamento separados em um plano bidimensional [16][19].

As características das redes neuronais artificiais como a capacidade de generalização, a representação distribuída, a capacidade de aprendizagem e tolerância a falhas tornam-na muito apreciável no campo da navegação de robôs móveis.

No artigo *“Neural Networks in Mobile Robot Motion”* [20] o autor Danica Janglová apresentou uma abordagem utilizando redes neurais artificiais, para planeamento de trajetórias e controlo inteligente de um robô móvel com rodas em um ambiente parcialmente desconhecido, que pode ainda conter vários obstáculos diferentes passíveis de movimentação. O método de construção de um caminho sem colisões é baseado em duas redes neurais, sendo que a primeira é usada para determinar o espaço livre usando dados de sensor ultrassom e a segunda rede encontra uma trajetória segura, evitando o obstáculo mais próximo.

Com o decorrer do tempo a implementação das redes neurais foi evoluindo, com o intuito de melhorar o seu desempenho. Por essa razão, os autores Usman Ahmed Say, Faraz Kunwar e Mazhar Iqbal no estudo denominado *“Guided Autowave Pulse Coupled Neural Network based real time path planning and an obstacle avoidance scheme for mobile robots”* [21] apresentaram uma modificação das redes neurais para formar o *Guided Autowave Pulse Coupled Neural Network* e atingir rápida convergência dos parâmetros para um robô móvel em um ambiente estático e dinâmico. O modelo proposto introduz o controlo direcional de microondas e o disparo acelerado de neurónios com base na estratégia de limiar dinâmico. Os resultados de simulação e experimentais demonstraram que esta abordagem é uma técnica de planeamento de trajetórias com eficiência de tempo para encontrar caminhos ideais em ambientes estáticos e dinâmicos.

2.2.3. OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS

No início dos anos 90, alguns estudos sobre o comportamento social de bando de animais expuseram que os animais de um certo grupo, como por exemplo insetos, pássaros e peixes, são capazes de comunicar entre si e essa capacidade confere-lhes uma grande vantagem de sobrevivência. Inspirados por esses trabalhos, James Kennedy e Russell Eberhart apresentaram em 1995 o algoritmo de otimização de enxame de partículas [22].

O algoritmo de otimização de enxame de partículas (PSO) é uma técnica de otimização estocástica baseada no comportamento social dos pássaros, porém não necessita que nenhum líder dentro do bando alcance o alvo. Os bandos criam uma forma cooperativa de

encontrar alimentos e cada membro destes altera o seu padrão de procura tendo em consideração quer as suas experiências aprendidas, quer as dos restantes membros [7][23][24].

Note-se que, quando um enxame de pássaros voa sobre um local, eles devem encontrar um ponto para pousar e, conseqüentemente, a decisão de qual o ponto onde o bando deve pousar é um problema complexo. Mais concretamente, todas as aves de um bando que procuram um bom local para pousar têm a capacidade de avaliar um local usando vários critérios de sobrevivência ao mesmo tempo, até que o melhor local seja encontrado por um dos membros do bando, para isso cada ave do bando guia-se pela sua experiência de conhecimento pessoal e experiência de conhecimento de bando, também designado conhecimento geral. Este movimento descrito por parte do bando só acontece quando todos os membros são capazes de partilhar informações entre si, caso contrário cada animal pousaria em um ponto diferente e em um momento diferente [22].

O problema de encontrar um ponto seguro para pousar caracteriza-se num problema de otimização, uma vez que o melhor lugar para pousar depende da maximização da disponibilidade de alimentos e da diminuição do risco de existência de predadores.

Com base nisso, o algoritmo PSO é um tipo de processo de pesquisa baseada em bando, no qual cada indivíduo é designado de partícula e movem-se no espaço de pesquisa de um problema de otimização, de acordo com fórmulas matemáticas simples sobre a posição e velocidade da partícula [24]. A posição de cada partícula retrata uma potencial solução do problema de otimização em questão [23]. Esta procura por melhores locais no ambiente de pesquisa e o movimento de cada partícula é influenciado pela sua posição de local mais conhecida caracterizado como o seu conhecimento de experiência pessoal (*pbest*), bem como pela posição mais conhecida pelo bando, isto é, conhecimento de experiência geral (*gbest*), que são destacadas como posições melhores quando encontradas por várias partículas. Em cada iteração, todas as partículas alteram o seu estado constantemente no espaço de pesquisa até alcançar o equilíbrio ou o estado ideal [24].

Uma partícula i é definida pelo seu vetor de posições, x_i , e pelo seu vetor de velocidade, v_i . Em cada iteração, todas as partículas atualizam a sua posição e velocidade de acordo com as equações (2.2) e (2.3), respetivamente [25]:

$$x_i^{t+1} = x_i^t + v_i^t \times t \quad (2.2)$$

$$v_i^{t+1} = wv_i^t + c_1r_1(pbest_i^t - x_i^t) + c_2r_2(gbest_i^t - x_i^t) \quad (2.3)$$

Em que:

- $pbest$ e $gbest$ indicam a melhor posição de cada partícula individual e a melhor posição do bando;
- w é peso da inércia usado para equilibrar a busca local e a busca global;
- c_1 e c_2 são duas constantes positivas chamadas coeficiente de aceleração individual e coeficiente de aceleração global, respetivamente;
- r_1 e r_2 são variáveis aleatórias distribuídas uniformemente dentro do intervalo $[0,1]$;
- t representa o número da iteração atual.

A Figura 8 ilustra o movimento de uma partícula i entre duas posições.

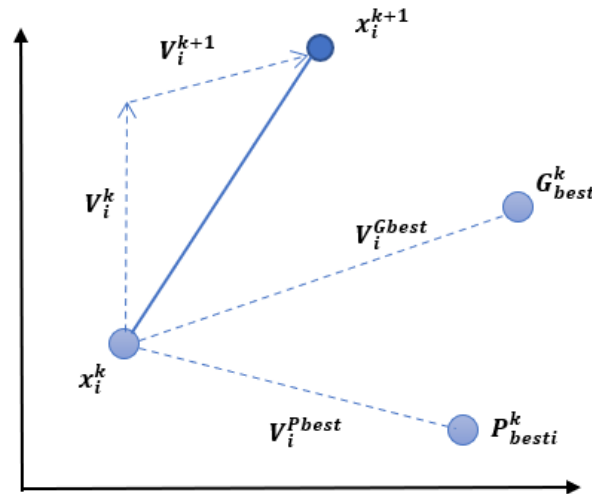


Figura 8 - Movimento de uma partícula i entre duas posições [25]

Atualmente, o algoritmo PSO é bastante utilizado no campo de navegação de robôs móveis. Como exemplo disso, podemos observar o artigo denominado de “*Applying Area Extension PSO in Robotic Swarm*” publicado pelos autores Adham Atyabi, Somnuk Phon-Amnuaisuk e Chin Kuan Ho que desenvolveram o *Area Extension PSO* como uma extensão do PSO básico, com o objetivo de abordar problemas de restrições de dinâmicas e dependentes do tempo na simulação da pesquisa robótica. A abordagem AEPSO é baseada na utilização de

um conjunto de robôs móveis simulados e foi implementada com sucesso em missões de resgate de sobreviventes, bem como no desarmamento de bombas.

Seguidamente, podemos constatar as modificações feitas por Micael Couceiro, Rui Rocha e Nuno Ferreira no estudo “*A PSO multi-robot exploration approach over unreliable MANETs*” [27]. Estes autores propuseram duas extensões do sistema PSO e Darwin PSO (DPSO), respetivamente designadas como *Robotic PSO* (RPSO) e *Robotic DPSO* (RDPSO), de forma a adaptar essas técnicas de inspiração biológica a sistemas multi-robô para evitar obstáculos e problemas de comunicação. No algoritmo RDPSO, os conceitos de exclusão e inclusão social são aplicados como um mecanismo de “punir e recompensar”, aumentando dessa forma a capacidade de encontrar os locais ótimos. Os autores verificaram através de resultados experimentais com plataformas reais que são necessários apenas 4 robôs para alcançar a missão proposta e uma otimização global em, aproximadamente, 90% dos experimentos, independentemente da distância máxima de comunicação.

O artigo “*Path planning for mobile robot using self-adaptive learning particle swarm optimization*” desenvolvido pelos autores Guangsheng LI e Wusheng Chou [28] apresenta uma abordagem *Self-Adaptive Learning Particle Swarm Optimization*, para resolver o problema de planeamento do caminho do robô móvel em um ambiente complexo sob várias restrições. Inicialmente, transformaram o problema de planeamento de trajetórias em um problema de otimização com vários objetivos de minimização e desenvolveram a função objetivo com três componentes, sendo estas o comprimento do percurso, grau de risco de colisão e suavidade. Em seguida, um novo modelo de aprendizagem auto-adaptativo foi desenvolvido para selecionar adaptativamente as melhores estratégias de pesquisa em diferentes momentos do processo de otimização, aumentando assim a capacidade de pesquisa do PSO.

2.2.4. OTIMIZAÇÃO POR COLÓNIAS DE FORMIGAS

O algoritmo de otimização de colónias de formigas (ACO), é proposto por Marco Dorigo em 1992 na sua tese de doutoramento. Este algoritmo é introduzido com base no

comportamento das formigas e a sua capacidade de encontrar o caminho mais curto desde a sua colónia até a uma fonte de alimento [7].

Inicialmente, o algoritmo ACO foi implementado para resolver o problema do vendedor ambulante que é um problema que tenta determinar a menor rota para percorrer uma série de cidades, visitando cada uma apenas uma vez, e retornar à cidade de partida. Atualmente, já está aplicado em vários campos de ciência e engenharia, como por exemplo problemas de atribuição quadrática, coloração gráfica e problemas de navegação de robôs móveis para otimização do caminho percorrido [7][29].

O princípio básico da ACO é que cada formiga liberta uma secreção chamada feromona no caminho que percorreu como referência e também consegue detetar as secreções libertadas por outras formigas ao longo da procura de alimentos. Através destas feromonas, as formigas constituintes da colónia podem comunicar umas com as outras e estabelecer um caminho a percorrer [30].

Quando uma formiga encontra alguma quantidade de comida, ela transporta o máximo que consegue e ao retornar deposita feromonas no chão de acordo com a quantidade e qualidade dos alimentos. À medida que a quantidade de feromonas em um caminho é maior do que em outro, a probabilidade de optar por esse caminho também é maior e quanto mais formigas seguirem o caminho mais secreções serão libertadas no percurso, deste modo a concentração de feromonas se torna mais elevada atraindo as restantes formigas da colónia, formando um mecanismo de *feedback* positivo. Geralmente, o caminho que adquire a maior concentração de feromonas é o mais curto, por sua vez as feromonas nos restantes caminhos vão gradualmente reduzir até que sejam extintas. Neste ponto, toda a colónia de formigas está reunida no caminho ótimo [30].

A Figura 9 apresenta o princípio descrito.

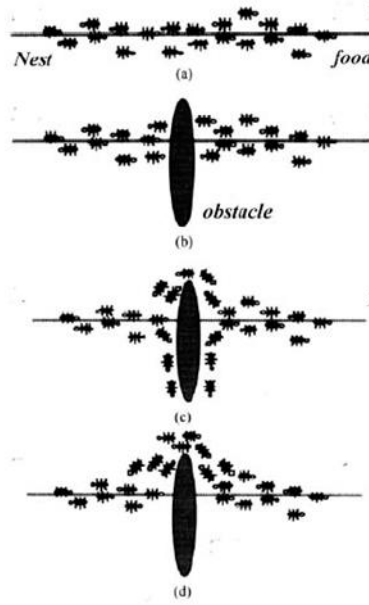


Figura 9 - Princípio de pesquisa de alimentos pela colônia de formigas [30]

O primeiro algoritmo proposto na literatura seguindo o princípio da ACO foi o *Ant System* [31][32]. A principal característica é que em cada iteração, os valores das feromonas associadas às arestas do grafo são alterados em tempo de execução por todas as formigas da colônia que completaram o percurso. Esta atualização pode ser realizada a cada movimento de uma formiga de um vértice para outro ou apenas depois que todas as formigas completaram o seu percurso. Por sua vez, a feromona τ_{ij} associada à aresta que une os vértices i e j , é atualizada segundo a equação (2.4) [33].

$$\tau_{ij} \leftarrow (1 - \rho) \times \tau_{ij} + \sum_{k=1}^m \Delta\tau_{ij}^k \quad (2.4)$$

Onde $\rho \in [0,1]$ é a taxa de evaporação da feromona e $\Delta\tau_{ij}^k$ é a quantidade de feromona depositada na aresta i e j pela k -ésima formiga calculada conforme a equação (2.5) [33].

$$\Delta\tau_{ij}^k = \begin{cases} \frac{1}{L_k}, & \text{Se formiga } k \text{ usou a aresta } (i,j) \text{ no seu trajeto} \\ 0, & \text{caso contrário} \end{cases} \quad (2.5)$$

Em que L_k é a duração do trajeto construído pela k -ésima formiga.

Nomeadamente, na construção da solução as formigas percorrem o gráfico de construção e utilizam iterativamente uma função probabilística para decidir se incluem ou não determinada aresta na solução. A probabilidade de transição $p(c_{ij}|s_k^p)$ da k -ésima formiga que se move do vértice i para o j é dada pela equação (2.6) [33].

$$p(c_{ij}|s_k^p) = \begin{cases} \frac{\tau_{ij}^\alpha \times \eta_{ij}^\beta}{\sum_{c_{il} \in N(s_k^p)} \tau_{il}^\alpha \times \eta_{il}^\beta}, & \text{se } j \in N(s_k^p) \\ 0, & \text{caso contrário} \end{cases} \quad (2.6)$$

Onde $N(s_k^p)$ é o conjunto de componentes que ainda não constituem a solução parcial s^p da formiga k . Os parâmetros α e β definem a importância relativa da feromona em relação à informação heurística n_{ij} que é determinada segundo a equação (2.7) [33].

$$n_{ij} = \frac{1}{d_{ij}} \quad (2.7)$$

Sendo que, d_{ij} é o comprimento do componente c_{ij} , ou seja, a distância entre o vértice i e j .

A navegação para robôs móveis utilizando o ACO é introduzido por Shirong Liu, Linbo Mao e Jinshou Yu no artigo “*Path Planning Based on Ant Colony Algorithm and Distributed Local Navigation for Multi-Robot Systems*” [34], onde apresentaram uma estratégia de prevenção de colisões para vários sistemas de robôs que consistia na atribuição de uma penalidade na intensidade de feromonas do percurso quando uma formiga virtual encontrasse um canto morto e, assim, evitar a indecisão no caminho do robô. O processo de busca de alimentos por parte das formigas é semelhante ao planeamento de trajeto dos robôs na medida em que, enquanto existirem formigas no ninho, elas encontrarão o trajeto mais curto até à comida para evitar obstáculos.

2.2.5. OTIMIZAÇÃO POR COLÔNIA DE BACTÉRIAS

Kevin Passino no ano de 2002 [35] apresentou o algoritmo *Bacterial Foraging Optimization* (BFO) inspirado no comportamento de forrageamento da bactéria *Escherichia coli*. Esta bactéria é composta por uma membrana plasmática, parede celular e cápsula que contém o citoplasma e o nucleóide. Adicionalmente, apresenta vários flagelos distribuídos aleatoriamente em torno da sua parede celular.

A *Escherichia coli* consegue locomover-se através da rotação dos flagelos, sendo que estes são capazes de girar na mesma direção em cerca de 100-200 rotações por segundo. Quando os flagelos giram no sentido horário, cada flagelo puxa a célula resultando em uma “queda”, isto é, uma mudança de direção. No caso contrário, quando os flagelos giram no sentido anti-horário, os seus efeitos se acumulam formando uma força que faz a bactéria avançar rapidamente em uma direção [36].

As bactérias conseguem sentir a concentração de nutrientes no ambiente e a alternância entre “correr” e “cair” possibilita não só que a bactéria pesquise a área nutritiva em direções aleatórias, mas também se afaste da área venenosa. Nomeadamente, a corrida é realizada quando a bactéria se aproxima de uma concentração de alimentos e a queda quando a mesma se afasta de alguns alimentos para procurar mais [36][37].

A quimiotaxia bacteriana é uma combinação complexa de corrida e queda que permite manter as bactérias em locais de maior quantidade de nutrientes. Inicialmente, as bactérias medem a concentração de alimentos em um local inicial, mudam de direção aleatoriamente, correm para uma nova posição e medem novamente a concentração de alimentos. Quando a concentração do novo local for maior, elas deslocam-se outra vez nessa direção. Por outro lado, quando a concentração do novo local for inferior, elas caem para encontrar outra direção e movem-se nessa nova direção [37][38].

A Figura 10 ilustra como é descrito o movimento de uma bactéria.

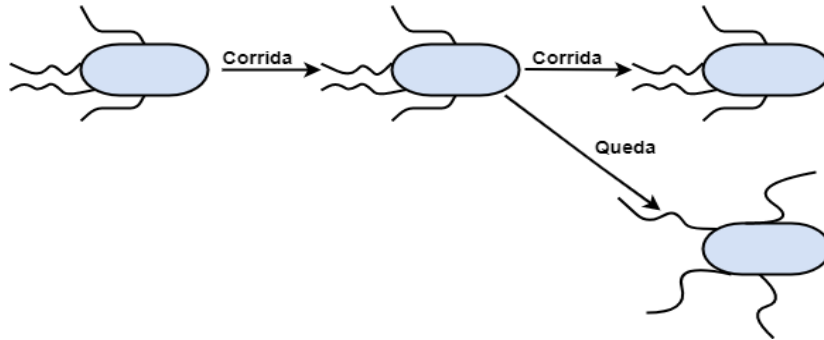


Figura 10 - Movimento de uma bactéria

As bactérias percorrem o ambiente em busca de locais com uma elevada concentração de alimentos. Quando estas obtêm alimento suficiente aumentam o seu comprimento e se dividem em duas partes iguais para reproduzir a próxima geração, enquanto as bactérias de regiões menos nutritivas se dispersam e morrem. As bactérias presentes em uma região com elevada concentração de alimentos atraem as restantes bactérias através de fenômenos químicos e aquelas que estão em uma região menos nutritiva emitem um sinal de aviso utilizando um sinal específico [7].

O algoritmo original BFO consiste em quatro mecanismos principais e estes são apresentados de seguida.

Quimiotaxia

A quimiotaxia é a principal motivação do processo de forrageamento das bactérias, que consiste em quedas e corridas consecutivamente. Supondo que $\theta^i(j, k, l)$ representa a bactéria na j -ésima etapa quimiotática, k -ésima reprodução e l -ésima etapa de eliminação-dispersão e $C(i)$ é o tamanho do passo quimiotático durante cada corrida ou queda, ou seja, a unidade de duração do deslocamento. Assim sendo, o movimento da bactéria pode ser descrito de acordo com a equação (2.8) [37].

$$\theta^i(j+1, k, l) = \theta^i(j, k, l) + C(i) \frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \quad (2.8)$$

Onde $\Delta(i)$ é a direção do vetor da j -ésima etapa quimiotática. Quando a bactéria executa uma corrida, $\Delta(i)$ é o mesmo que na última etapa quimiotática, caso contrário, $\Delta(i)$ é um vetor aleatório com valores entre $[-1, 1]$ [37].

Enxame

O comportamento social que as bactérias exercem, de atrair as outras bactérias quando encontram grandes quantidades de nutrientes ou de repelir as mesmas quando estão em perigo, é modelado pela equação (2.9) [39].

$$Jcc(\theta, \theta^i(j, k, l)) = \sum_{i=1}^S \left[-d_{attract} \exp(-w_{attract} \sum_{f=1}^p (\theta_f - \theta_f^i)^2) \right] + \sum_{i=1}^S \left[h_{repellant} \exp(-w_{repellant} \sum_{f=1}^p (\theta_f - \theta_f^i)^2) \right] \quad (2.9)$$

Em que:

- $\theta = [\theta_1, \dots, \theta_p]^T$ é a bactéria no domínio de otimização;
- θ_f^i caracteriza o f -ésimo componente da i -ésima posição da bactéria θ^i ;
- $Jcc(\theta, \theta^i(j, k, l))$ é o valor de comunicação célula a célula que será adicionado ao resultado da função de adequação na fase j de quimiotaxia;
- p é o tamanho da dimensão do problema;
- S é o número de bactérias;
- $d_{attract}, w_{attract}, h_{repellant}, w_{repellant}$ são os diferentes coeficientes que representam a força de atração ou repulsão.

Reprodução

Como o indivíduo menos saudável morre, é realizada uma etapa de produção após todas as etapas quimiotáticas. A saúde das bactérias é calculada através da soma da adequação de todas as etapas de toda a sua vida, ou seja, equação (2.10) [37].

$$\sum_{j=1}^{Nc} J(i, j, k, l) \quad (2.10)$$

Onde N_c representa o número de etapas em um processo de quimiotaxia.

As bactérias são classificadas de ordem decrescente da quantidade de nutrientes adquiridos nos processos quimiotáticos. Por essa razão, considera-se que as bactérias da primeira metade da população tenham conseguido obter nutrientes suficientes para se reproduzirem. Assim cada uma das bactérias mais saudáveis se divide em duas bactérias colocadas na mesma posição, enquanto as restantes morrem e são removidas da população, mantendo constante o tamanho do enxame [40].

Eliminar e dispersar

As ocorrências de algumas situações podem resultar na eliminação-dispersão, como por exemplo, as bactérias ficarem presas nos locais iniciais ou ótimos. Dessa forma, a diversidade do BFO pode modificar-se, a fim de eliminar os acidentes de aprisionamento em locais.

Este evento ocorre após um certo número de processos de reprodução, de seguida algumas bactérias são escolhidas, conforme uma probabilidade predefinida, P_{ed} , para serem mortas e novas bactérias são geradas aleatoriamente dentro do espaço de pesquisa para substituição [37][39].

A implementação do algoritmo BFO na navegação de robôs moveis foi introduzido por Leandro Coelho e Cezar Sierakowski no estudo designado "*Bacteria colony approaches with variable velocity applied to path optimization of mobile robots*" [41]. Estes autores testaram novas abordagens do algoritmo de otimização de colónias de bactérias com velocidade variável baseada nas distribuições uniformes de Gauss e Cauchy, tendo a verificação desta abordagem em problemas de planeamento de trajetórias ter sido realizada em ambientes estáticos.

Com o decorrer do tempo mais técnicas foram surgindo, como é o caso do estudo intitulado como "*Mobile robot path planning based on adaptive bacterial foraging algorithm*" desenvolvido pelos autores Liang Xiao-dan, Li Liang-yu, Wu Ji-gang e Chen Han-ning [42], onde utilizam a mesma estratégia de navegação em um ambiente estático, mas com múltiplos obstáculos integrados. Por conseguinte, neste modelo proposto, o robô imita o

comportamento das bactérias para determinar o caminho ótimo e sem colisões nos vários obstáculos entre o ponto inicial e o ponto objetivo no espaço de pesquisa. Em termos de simulação, foram admitidos dois cenários de teste com diferentes números de obstáculos para avaliar o desempenho da abordagem proposta. Os autores puderam assim concluir que o robô consegue adaptar-se com boa precisão e estabilidade às trajetórias planeadas em ambientes complexos.

2.2.6. ALGORITMO DE COLÓNIA DE ABELHAS ARTIFICIAIS

O algoritmo *Artificial Bee Colony* (ABC) introduzido por Dervis Kharaboga em 2005 [43], é um algoritmo meta-heurístico baseado em enxame, especificamente, nas atividades das abelhas na procura de alimento.

Para aplicação deste algoritmo, o problema de otimização é convertido no problema de encontrar o melhor vetor de parâmetros que minimiza a função objetivo. As abelhas artificiais localizam aleatoriamente um conjunto de soluções iniciais e, seguidamente, aperfeiçoam-nas iterativamente através das estratégias de análise da vizinhança de fonte de alimento em memória e abandono das soluções más. Esta inclusão de fontes ricas em alimentos por parte das abelhas e o abandono de fontes pobres é caracterizado como *feedback* positivo e *feedback* negativo, respetivamente [44].

No ABC, a colónia de abelhas artificiais contém três tipos de abelhas: abelhas empregadas ou forrageiras, abelhas espectadores e abelhas escoteiras [45]. Na fase inicial do processo de forrageamento, as abelhas escoteiras exploram aleatoriamente o ambiente de pesquisa, a fim de descobrir as posições das fontes de alimentos. Quando uma fonte de alimento é encontrada, o néctar dessa fonte é explorada pelas abelhas forrageiras que memorizam a sua localização. Se for encontrada alguma fonte de alimento mais vantajosa, estas são incorporadas e compartilhadas com as abelhas espectadoras. Por sua vez, as abelhas espectadoras, que esperam na colmeia, escolhem probabilisticamente a melhor fonte de alimento, de acordo com a informação dada pelas abelhas forrageiras, no momento de descarga na colmeia, que descreve a sua qualidade do néctar. Nomeadamente, quando

uma fonte de alimento é esgotada, a abelha forrageira que a explorou torna-se uma abelha escoteira em busca de novas fontes de alimento [46][47].

Em termos de implementação, no algoritmo ABC a localização de uma fonte de alimento corresponde a uma possível solução para o problema, por sua vez o número de soluções é igual ao número de abelhas forrageiras, pois cada abelha forrageira representa uma fonte de alimento específica. Ainda, a quantidade do néctar existente numa fonte de alimento corresponde à qualidade da solução do problema [48].

A colónia de abelhas artificiais pode ser dividida nas seguintes etapas:

- Fase de inicialização: todos os vetores da população de fontes de alimento são inicializados ($m = 1$ até tamanho máximo da população) por abelhas escoteiras. Como cada fonte de alimento, $\vec{x}_m$, é um vetor de solução para o problema de otimização, cada vetor deste tipo contém n variáveis ($x_{mi}, i = 1 \dots n$), que devem ser otimizadas para minimizar a função objetivo. A inicialização pode ser realizada a partir da equação (2.11) [44].

$$x_{mi} = x_{min} + rand(0,1) \times (x_{max} - x_{min}) \quad (2.11)$$

Onde:

- x_{min} e x_{max} são o limite inferior e superior do parâmetro x_{mi} , respetivamente;
- $rand(0,1)$ é um valor aleatório entre 0 e 1.
- Fase de abelha empregada: cada abelha empregada procura novas fontes de alimento ($\vec{v}_m$) com mais néctar na vizinhança da fonte de alimento em sua memória. Seguidamente, elas encontram a nova fonte de alimento vizinha e avaliam a sua lucratividade deslocando-se para esta quando a quantidade de néctar da nova fonte é mais elevada em comparação com a anterior. A fórmula dada pela equação (2.12) permite determinar a nova fonte de alimento vizinha [44].

$$v_{mi} = x_{mi} + \phi_{mi}(x_{mi} - x_{ki}) \quad (2.12)$$

Onde:

- $\vec{x}_k$ é uma fonte de alimento selecionada aleatoriamente;
- i é o índice de parâmetro escolhido aleatoriamente;
- $\emptyset_{mi}$ é um número aleatório entre o intervalo $[-1, 1]$.

Depois de estabelecer a nova fonte de alimento, é calculada a sua adequação e é determinada uma preferência entre $\vec{v}_m$ e $\vec{x}_m$.

O valor de adequação da solução pode ser calculado para o algoritmo ABC através da equação (2.13) [44].

$$fit_m(\vec{x}_m) = \begin{cases} \frac{1}{1 + f_m(\vec{x}_m)}, & \text{Se } f_m(\vec{x}_m) \geq 0 \\ 1 + |f_m(\vec{x}_m)|, & \text{Se } f_m(\vec{x}_m) < 0 \end{cases} \quad (2.13)$$

Em que $f_m(\vec{x}_m)$ é o valor da função objetivo da solução $\vec{x}_m$.

- Fase de abelhas espectadoras: nesta fase, as abelhas espectadoras escolhem a fonte de alimentos dependendo dos valores de probabilidade calculados, tendo por base os valores de aptidão. A probabilidade p_m com que $\vec{x}_m$ é escolhido por uma abelha espectadora pode ser determinada usando a equação (2.14) [44].

$$p_m = \frac{fit_m(\vec{x}_m)}{\sum_{m=1}^{SN} fit_m(\vec{x}_m)} \quad (2.14)$$

- Fase de abelhas escoteiras: por fim, as abelhas forrageiras cujas fontes de alimento não são melhoradas dentro de um número predeterminado de ensaios, designado como critério de abandono, tornam-se abelhas escoteiras e as suas soluções são abandonadas. Por outras palavras, as fontes de alimento que são pobres ou que se tornaram pobres devido à exploração são abandonadas e o comportamento de *feedback* negativo ocorre de forma a equilibrar o *feedback* positivo. Consequentemente, as abelhas convertidas iniciam a busca de novas soluções para substituir a solução removida [44].

A introdução do algoritmo ABC na navegação de robôs móveis em um ambiente estático é apresentada por Marco Contreras-Cruz, Victor Ayala-Ramirez e Uriel Hernandez-Belmonte

no estudo designado por “*Mobile robot path planning using artificial bee colony and evolutionary programming*” [49], onde a abordagem proposta conjuga o algoritmo de colônia de abelhas artificiais como procedimento de busca e o algoritmo de programação evolutiva para aperfeiçoar o caminho de solução encontrado.

Uma abordagem semelhante em um ambiente estático é apresentada no artigo “*Bee colony algorithm for real-time optimal path planning of mobile robots*” por M. Saffari e M. Mahjoob [50], em que propuseram um método constituído por duas etapas. Na primeira etapa, é estabelecido um caminho inicial livre de colisão desde o ponto de partida até ao destino através de uma regra simples e na segunda etapa é otimizado o caminho inicial com recurso do algoritmo ABC. Os resultados das simulações comprovam que este método é eficaz e pode ser aplicado no planeamento de trajeto em tempo real.

Com o intuito de conseguir a navegação de robôs móveis em um ambiente dinâmico em tempo real, os autores Qianzhi Ma e Xiujuan Lei expuseram no artigo denominado “*Dynamic Path Planning of Mobile Robots Based on ABC Algorithm*” [51] uma abordagem hibridizada que combina o algoritmo ABC com a estratégia de janela deslizante no tempo. O caminho global foi substituído por vários caminhos locais, dentro de uma série de janelas deslizantes, idealizados pelo algoritmo ABC. Os resultados da simulação desta abordagem demonstram uma elevada adequação para resolução de problemas, bem como uma grande eficiência e precisão.

2.2.7. ALGORITMO DE PESQUISA DE CUCO

O algoritmo *Cuckoo Search* (CS) é um algoritmo meta-heurístico apresentado por Xin-She Yang e Suash Deb [52] no ano de 2009. Este algoritmo tem por base o comportamento preguiçoso na reprodução de pássaros de cuco.

Os cucos são uma espécie de ave com uma estratégia reprodutiva única em comparação com outras espécies de ave. Em particular, algumas espécies de cucos põem os ovos em ninhos comunais e podem ainda retirar os ovos de outras aves para aumentar a probabilidade de eclosão dos seus ovos. Por outro lado, outras espécies de cucos praticam

o método de parasitismo de ninhadas para colocar os seus ovos nos ninhos de outros pássaros hospedeiros [53].

Os cucos parasitas colocam os ovos em um ninho hospedeiro que geralmente choca mais rápido que os restantes ovos no ninho. Quando isso ocorre, o cuco parasita empurra para fora do ninho os ovos não chocados, com o intuito de reduzir a probabilidade de os ovos legítimos chocarem. Adicionalmente, a cria de cuco estrangeiro pode imitar as crias hospedeiras para conseguir mais alimento [53].

No entanto, há uma possibilidade de o cuco hospedeiro descobrir que um dos ovos do seu ninho é estrangeiro. Em consequência desse acontecimento, o cuco pode-se livrar do ovo hospedeiro ou abandona o ninho por completo e constrói um ninho completamente novo em outro lugar [53][7].

Com base nesses fenômenos, os autores Yang e Deb desenvolveram o algoritmo de otimização CS baseados nas três regras a seguir [54]:

- Cada cuco põe um ovo de cada vez em um ninho escolhido aleatoriamente;
- Os melhores ninhos com elevada qualidade de ovos irão transitar para a próxima geração;
- O número de ninhos hospedeiros disponíveis é fixo e o ovo depositado por um cuco é descoberto pelo pássaro hospedeiro com uma probabilidade $p_a \in [0,1]$. Nessa situação, o cuco hospedeiro pode desfazer-se do ovo ou abandonar o ninho e construir um novo em outro lugar.

No sentido da implementação, cada ovo em um ninho representa uma solução e cada cuco pode representar uma nova solução. O objetivo consiste em utilizar novas e melhores soluções para substituírem uma solução menos boa nos ninhos [53]. Na forma original do algoritmo CS, apenas um ovo está em cada ninho, deste modo, não há distinção entre um ovo, um ninho ou um cuco. Assim sendo, cada ninho equivale a um ovo que, por sua vez, equivale a um cuco [54].

Este algoritmo ainda utiliza uma combinação equilibrada de um passeio aleatório local e um passeio aleatório de exploração global, que é controlado pelo parâmetro de comutação p_a . O passeio aleatório local é caracterizado pela equação (2.15) [55].

$$x_i^{t+1} = x_i^t + \beta s \otimes H(pa - \varepsilon) \otimes (x_j^t - x_k^t) \quad (2.15)$$

Em que:

- x_j^t e x_k^t são duas soluções diferentes escolhidas aleatoriamente por permutação aleatória;
- H é a função de Heaviside;
- ε é um número aleatório retirado de uma distribuição uniforme;
- s é o tamanho do passo;
- $\otimes$ é o produto de entrada de dois vetores;
- β é um pequeno fator de escala.

No que toca ao passeio aleatório global, este é determinado usando voos de Lévy através da equação (2.16) [55].

$$x_i^{t+1} = x_i^t + \alpha \otimes L(s, \lambda) \quad (2.16)$$

Onde $\alpha > 0$ é o fator de escala do tamanho do passo e $L(s, \lambda)$ são os comprimentos dos passos que são distribuídos conforme a distribuição de probabilidade dada pela equação (2.17) [55].

$$L(s, \lambda) = \frac{\lambda \Gamma(\lambda) \sin(\frac{\pi \lambda}{2})}{\pi} \frac{1}{s^{1+\lambda}} \quad (2.17)$$

Este algoritmo de otimização é bastante aplicado em problemas de otimização em engenharia devido à sua elevada eficiência e taxa de convergência. Uma das áreas de aplicação é a navegação por robôs móveis, onde Prases Mohanty e Dayal Parhi apresentaram o algoritmo de navegação de um robô de rodas no artigo “*Optimal path planning for a mobile robot using cuckoo search algorithm*” [56]. Neste estudo apresentaram simulações e experiências em tempo real em um ambiente desconhecido ou parcialmente conhecido, composto por vários obstáculos estáticos. Os resultados experimentais e práticos provaram uma boa concordância, uma vez que, o erro de desvio é muito pequeno.

Outra abordagem de planeamento de trajetórias é o algoritmo apresentado pelos autores Gaige Wang, Lihong Guo, Hong Duan, Heqi Wang, Luo Liu e Mingzhen Shao no artigo “*A Hybrid Metaheuristic DE/CS Algorithm for UCAV Three-Dimension Path Planning*” [57]. Estes autores combinaram o algoritmo de evolução diferencial com o CS para otimizar problemas de planeamento de trajetórias tridimensionais para veículos aéreos de combate autónomos. A evolução diferencial é aplicada para melhorar o processo de seleção de cucos do algoritmo CS durante o processo de atualização do cuco no ninho. Neste algoritmo os cucos podem agir como agentes na procura do caminho aéreo ideal, permitindo ao robô aéreo encontrar o percurso seguro percorrendo os nós escolhidos, evitando as ameaças e consumindo o mínimo combustível. Esta abordagem aumenta a velocidade de convergência global, resultando assim numa melhor exploração do ambiente tridimensional.

3. PLATAFORMAS DE ROBÔS MÓVEIS

No atual capítulo são identificados os diversos tipos de sistemas de locomoção, ilustrando cada um com um exemplo. Posteriormente, são apresentadas as principais configurações de plataformas de robôs móveis com rodas e as mesmas são explicadas detalhadamente através das equações do modelo cinemático. O capítulo encerra com a comparação de todas as configurações apresentadas, onde são abordadas as principais capacidades de desempenho de robôs móveis.

3.1. TIPOS DE SISTEMAS DE LOCOMOÇÃO

No mundo da robótica móvel, várias investigações estão sendo, permanentemente, realizadas com o objetivo de construir robôs e controlá-los em diferentes ambientes. Os robôs móveis são alvos de um elevado interesse devido à grande diversidade de tarefas que estes podem realizar e a adequação dos robôs da melhor forma possível às exigências das tarefas, isto é, precisão e exatidão, vem sendo um trabalho imprescindível.

O sistema de locomoção é uma das principais características de um robô para a definição das técnicas de controlo e planeamento de trajetórias. Dessa forma, o tipo do sistema de locomoção do robô é um fator importante no projeto do robô móvel e depende quer do meio em que o robô se move, quer da sua manobrabilidade, controlabilidade, eficiência e estabilidade [58].

De acordo com o ambiente, os robôs móveis podem ser classificados nas seguintes categorias principais [59]:

- Sistemas móveis terrestres: são tipos de plataformas móveis que tem por base veículos móveis com rodas, lagartas, pernas ou robôs que simulam outro tipo de locomoção animal como rastejar.
- Sistemas móveis aéreos: são plataformas voadoras como aviões, helicópteros, *drones*, foguetes, sistemas voadores que imitam animais, que voam em um espaço aéreo ou orbitam a Terra ou algum satélite.
- Sistemas móveis aquáticos: esta categoria contempla as plataformas móveis que se deslocam em cima e dentro de água, como barcos, navios e submarinos.

O robô Urbano, também designado de Urbie, é um exemplo de um robô móvel terrestre com locomoção através de lagartas, desenvolvido em conjunto pela Jet Propulsion Laboratory, iRobot Corporation, Institute of Carnegie Mellon University e University of Southern California Robotics Research Laboratory [60].

O Urbie foi desenvolvido com o objetivo de auxiliar no reconhecimento militar móvel em terrenos acidentados e de difícil acesso, porém tornou-se útil também para a polícia,

equipas de emergência e de resgate para ser utilizado na busca e salvamento em zonas de desastre.

A Figura 11 apresenta o robô Urbie.

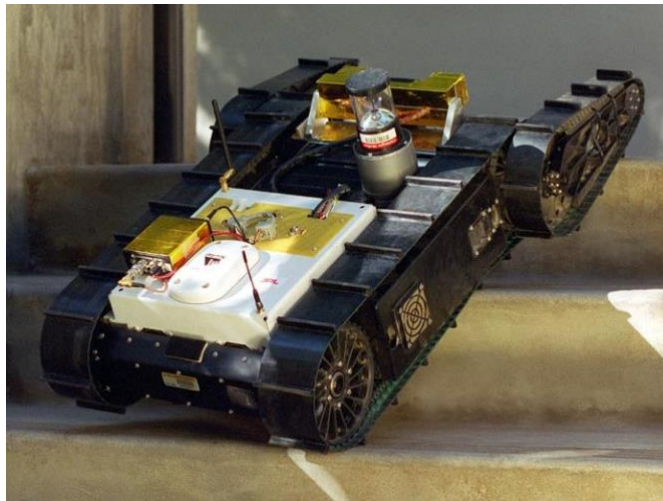


Figura 11 - Robô Urbie [61]

Como exemplo de um robô de um sistema móvel aéreo identifica-se o robô Marvin MarkII, Figura 12. O Marvin é um robô aéreo com navegação inteligente desenvolvido pela Technische Universität Berlin [62].

Este equipamento foi construído com o propósito de desenvolver um sistema autônomo para participação no projeto COMETS e, tendo por base um modelo de helicóptero disponível no mercado, este veículo multifuncional permite a exploração de ambientes através da acoplação de um sistema de aquisição de imagens.



Figura 12 - Robô Marvin Mark II [62]

Por fim, um exemplo de um robô móvel aquático é o Sepios [63]. Este robô foi desenvolvido por um grupo de estudantes do Swiss Federal Institute of Technology com o objetivo de construir uma arquitetura de controlo robótico baseada em um tipo especial de barbatana inspirada nos chocos.

O Sepios distingue-se dos restantes robôs aquáticos clássicos, pois apresenta uma característica omnidirecional, elevada manobrabilidade e baixa perturbação para os peixes, o que permite a aproximação dos mesmos, tornando-se um bom equipamento para filmagens da vida marinha.

A Figura 13 ilustra o robô móvel Sepios.



Figura 13 - Robô Sepios [64]

3.2. TIPOS DE ROBÔS MÓVEIS

De entre os sistemas de locomoção descritos na secção anterior, este projeto tem mais incidência sobre os sistemas móveis terrestres, mais precisamente, nos veículos móveis com rodas. As combinações de diversas disposições de rodas permitem a construção de uma grande variedade de robôs móveis com vários graus de manobrabilidade distintos. De seguida, serão apresentadas as principais características de algumas configurações de robôs móveis com rodas.

3.2.1. ROBÔ DIFERENCIAL

Os robôs móveis utilizam frequentemente um mecanismo de acionamento designado por acionamento diferencial. Os robôs de direção diferencial apresentam duas rodas motrizes montadas em um eixo comum, onde a velocidade de cada roda é controlada independentemente por um motor separado, e uma ou mais rodas livres para apoiar o robô e conferir estabilidade [59][65].

Embora seja possível variar a velocidade de cada roda, ao longo do movimento de rotação o robô deve girar em torno de um ponto imaginário que se localiza ao longo do eixo comum das rodas, como ilustrado na Figura 14. Este ponto imaginário é designado de *Instantaneous Center of Rotation* (ICR). Para o estudo do robô móvel diferencial foi escolhido o caso particular de um robô no qual o centro geométrico é coincidente com o centro de massa [65].

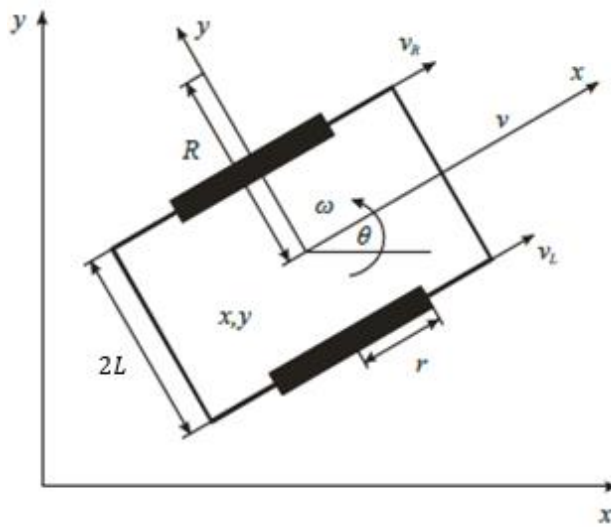


Figura 14 - Cinemática do robô diferencial [66]

O sistema de coordenadas Oxy define o sistema de coordenadas do mundo. Metade do comprimento do eixo entre as rodas do robô é representado pela variável L , r caracteriza o raio das mesmas e R é o raio instantâneo da curva medido a partir do meio do robô até ao ponto ICR. A configuração do robô móvel nas cinco coordenadas generalizadas é dada pelo vetor $q = [x, y, \theta, \theta_r, \theta_l]$, sendo que a posição e a orientação do robô são definidas pelas coordenadas x e y do seu centro de massa e pelo ângulo de direção θ . Seguidamente, as variáveis θ_r e θ_l representam, respetivamente, o ângulo de rotação da roda direita e da

roda esquerda. A roda livre do robô serve apenas para manter o equilíbrio do mesmo e, por essa razão, o seu efeito não é contabilizado [67][68].

As trajetórias que o robô móvel percorre podem ser alteradas variando as velocidades das rodas motrizes. Para determinação das velocidades em cada instante de tempo, existem relações matemáticas entre as velocidades angulares e as velocidades lineares da roda esquerda, v_L , equação (3.1), e da roda direita, v_R , equação (3.2) [69].

$$v_L(t) = r\omega_L(t) \quad (3.1)$$

$$v_R(t) = r\omega_R(t) \quad (3.2)$$

A velocidade linear $v(t)$ e a velocidade angular $\omega(t)$ podem ser obtidas conforme as equações (3.3) e (3.4), respetivamente [69].

$$v(t) = \frac{(v_R(t) + v_L(t))}{2} \quad (3.3)$$

$$\omega(t) = \frac{(v_R(t) - v_L(t))}{2L} \quad (3.4)$$

Adicionalmente, é possível representar as velocidades do robô a partir das velocidades angulares das rodas motrizes juntamente com os parâmetros geométricos do robô, de acordo com as equações (3.5) e (3.6) [70].

$$v(t) = r \frac{(\omega_R(t) + \omega_L(t))}{2} \quad (3.5)$$

$$\omega(t) = r \frac{(\omega_R(t) - \omega_L(t))}{2L} \quad (3.6)$$

As velocidades lineares de cada roda, equações (3.7) e (3.8), podem também ser representadas segundo as velocidades do robô da seguinte forma [69]:

$$v_L(t) = v(t) - \frac{\omega(t) \times 2L}{2} \quad (3.7)$$

$$v_R(t) = v(t) + \frac{\omega(t) \times 2L}{2} \quad (3.8)$$

O robô móvel encontra-se, em qualquer instante, numa determinada posição do espaço com uma direção que gera um ângulo, θ , com o eixo x da estrutura de coordenadas do mundo. À medida que o robô se move no espaço, a estrutura de coordenadas do robô move-se com ele, logo o θ é o ângulo entre o eixo x da estrutura do mundo e o eixo x da estrutura do robô [71]. Desse modo, tendo em consideração as relações acima, a cinemática do robô nas coordenadas locais pode ser expressa através da equação (3.9) [70].

$$\begin{bmatrix} \dot{x}_m(t) \\ \dot{y}_m(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} \frac{r}{2} \cos(\theta(t)) & \frac{r}{2} \cos(\theta(t)) \\ \frac{r}{2} \sin(\theta(t)) & \frac{r}{2} \sin(\theta(t)) \\ \frac{r}{2L} & -\frac{r}{2L} \end{bmatrix} \begin{bmatrix} \omega_R(t) \\ \omega_L(t) \end{bmatrix} \quad (3.9)$$

Enquanto, a cinemática externa do robô, ou seja, nas coordenadas globais é dada pela equação (3.10).

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} \cos(\theta(t)) & 0 \\ \sin(\theta(t)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \quad (3.10)$$

Cinemática Direta

O posicionamento de um robô em um determinado momento de tempo, t , é calculado a partir da integração do modelo cinemático da equação (3.10) e é designado como odometria, equações (3.11), (3.12) e (3.13). Adicionalmente, a determinação da posição do robô para um determinado valor das velocidades é chamada de cinemática direta [59].

$$x(t) = \int_0^t v(t) \cos(\theta(t)) dt \quad (3.11)$$

$$y(t) = \int_0^t v(t) \sin(\theta(t)) dt \quad (3.12)$$

$$\theta(t) = \int_0^t \omega(t) dt \quad (3.13)$$

Cinemática Inversa

No que toca à cinemática inversa, esta é aplicada para determinar as variáveis de velocidades que correspondem a uma determinada posição do robô ou a uma trajetória do caminho desejado. Dessa forma, a velocidade linear e a velocidade angular do robô podem ser calculadas a partir das equações (3.14) e (3.15) [69].

$$v(t) = \pm \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2} \quad (3.14)$$

$$\omega(t) = \frac{\dot{x} \times \ddot{y} - \dot{y} \times \ddot{x}}{\dot{x}^2 + \dot{y}^2} \quad (3.15)$$

3.2.2. ROBÔ TRICICLO

O sistema de locomoção de um robô triciclo baseia-se em duas rodas convencionais fixas sobre um eixo comum e uma roda convencional, centrada e orientável, que contém as funções de tração e direção. Esta configuração de rodas é muito utilizada na robótica móvel, graças às três rodas que tornam o robô estável na direção vertical. No entanto, quando em movimento, por vezes o centro de gravidade do robô localiza-se nos limites da superfície de equilíbrio, o que resulta numa perda momentânea de tração e, por sua vez, torna-se uma fonte de erro no momento de determinação da posição do robô [72].

A Figura 15 ilustra a cinemática do robô triciclo.

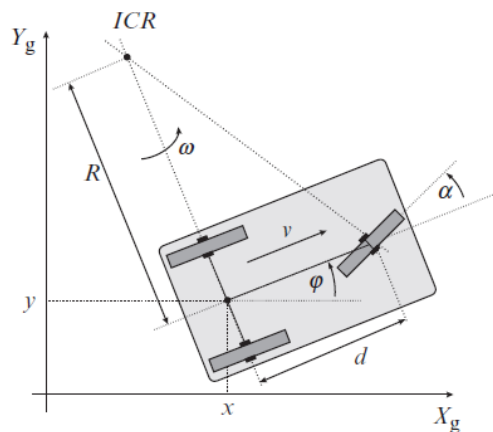


Figura 15 - Cinemática do robô triciclo [59]

As variáveis de controlo deste tipo de acionamento são o ângulo da direção do volante, α , e a velocidade angular da roda dianteira em torno do seu eixo, ω_s , em função do tempo [59]. O ponto ICR é localizado em linha com o eixo das rodas traseiras e é definido pela interseção do eixo da roda dianteira com o eixo das rodas traseiras fixas [73]. Ao longo da execução de uma curva, o robô circula em torno do ponto ICR com uma velocidade angular ω , raio R e distância entre as rodas d [59].

A dimensão do raio da curva é calculada conforme a equação (3.16) [59].

$$R(t) = d \tan\left(\frac{\pi}{2} - \alpha(t)\right) = \frac{d}{\tan(\alpha(t))} \quad (3.16)$$

E a velocidade angular com que a roda dianteira gira em torno do ponto ICR é dada pela equação (3.17).

$$\omega(t) = \dot{\phi} = \frac{v_s(t)}{\sqrt{d^2 + R^2}} = \frac{v_s(t)}{d} \sin(\alpha(t)) \quad (3.17)$$

Onde a velocidade linear da roda pode ser descrita pela equação (3.18), sendo que r representa o raio da roda dianteira.

$$v_s(t) = \omega_s(t)r \quad (3.18)$$

Por sua vez, o modelo cinemático na estrutura do robô de três rodas é obtido a partir das equações (3.19), (3.20) e (3.21) [59].

$$v(t) = \dot{x}_m = v_s(t) \cos(\alpha(t)) \quad (3.19)$$

$$\dot{y}_m = 0 \quad (3.20)$$

$$\dot{\phi} = \frac{v_s(t)}{d} \sin(\alpha(t)) \quad (3.21)$$

Enquanto que, o modelo cinemático na estrutura do mundo, isto é, o modelo cinemático de postura pode ser dado conforme as equações (3.22), (3.23) e (3.24).

$$\dot{x} = v_s(t) \cos(\alpha(t)) \cos(\phi(t)) \quad (3.22)$$

$$\dot{y} = v_s(t) \cos(\alpha(t)) \sin(\varphi(t)) \quad (3.23)$$

$$\dot{\varphi} = \frac{v_s(t)}{d} \sin(\alpha(t)) \quad (3.24)$$

Alternativamente, pode ser escrita em formato matricial conforme a equação (3.25).

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \cos(\varphi(t)) & 0 \\ \sin(\varphi(t)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \quad (3.25)$$

Cinemática Direta

A determinação da cinemática direta de um robô triciclo é semelhante ao caso de mobilidade diferencial. Mais concretamente, tem-se em consideração a equação (3.25) e integra-se, obtendo assim as equações anteriores (3.11), (3.12) e (3.13) [59].

Cinemática Inversa

A cinemática inversa é geralmente mais complexa de solucionar, devido a esse facto aplica-se uma estratégia de movimento com dois padrões. O primeiro padrão de movimento, especifica que o robô móvel só pode avançar retilineamente ($\alpha(t) = 0$) e o segundo padrão de movimento define que o robô só pode rodar no local ($\alpha(t) = \pm \frac{\pi}{2}$) [59].

Assim sendo, para o movimento retilíneo do robô, as velocidades simplificam-se nas equações (3.26) e (3.27).

$$v(t) = v_s(t) \quad (3.26)$$

$$\omega(t) = 0 \quad (3.27)$$

Tendo em consideração estas velocidades com a equação (3.25), as equações de movimento convertem-se nas equações (3.28), (3.29) e (3.30) [59].

$$x(t) = x(0) + v_s \cos(\varphi(0))t \quad (3.28)$$

$$y(t) = y(0) + v_s \sin(\varphi(0)) t \quad (3.29)$$

$$\varphi(t) = \varphi(0) \quad (3.30)$$

No caso do movimento de rotação, as velocidades do robô alteram-se conforme as equações (3.31) e (3.32).

$$v(t) = 0 \quad (3.31)$$

$$\omega(t) = \frac{v_s(t)}{d} \quad (3.32)$$

E quando associadas com a equação anterior (3.25) resultam no modelo de movimento dado pelas equações (3.33), (3.34) e (3.35) [59].

$$x(t) = x(0) \quad (3.33)$$

$$y(t) = y(0) \quad (3.34)$$

$$\varphi(t) = \varphi(0) + \frac{v_s(t)}{d} t \quad (3.35)$$

A combinação desta solução possibilita através das equações de movimento dos dois padrões determinar as variáveis de controlo necessárias para o movimento durante cada fase.

3.2.3. ROBÔ ACKERMAN

O robô quadriciclo, também conhecido como Ackerman, surgiu para satisfazer a necessidade de solucionar o problema do erro na determinação da posição do robô triciclo [72].

A configuração Ackerman baseia-se no princípio de direção de um carro, na qual as quatro rodas possuem os seus eixos posicionados de forma que os eixos das duas rodas frontais se intercetem num ponto, ICR. No entanto, como as rodas traseiras são fixas, este ponto tem de pertencer à linha reta definida pelo prolongamento do eixo comum das rodas traseiras. Para obter este facto, é necessário que numa mudança de direção a roda

dianteira interna (mais próxima do ICR) possua um ângulo de viragem maior do que o da roda dianteira externa, o que leva a uma minimização do deslizamento das rodas e que a roda interna se mova com uma velocidade mais lenta do que a oposta [59][72].

Nomeadamente, a grande vantagem desta configuração caracteriza-se pelo facto de as forças laterais serem neutralizadas devido às restrições de deslizamento. Todavia, esta geometria apresenta uma limitação no ângulo de direcção, o que resulta em restrições de movimento. Especificamente, o robô não consegue rodar sobre só próprio e, em mudanças de direcção que o raio da curva é superior ao espaço disponível, este necessita de executar manobras repetidamente até conseguir concluir a mudança de direcção [74].

Na Figura 16 podemos verificar que a roda dianteira direita encontra-se no lado interno e, dessa forma, apresenta um ângulo de orientação maior em relação à roda dianteira esquerda.

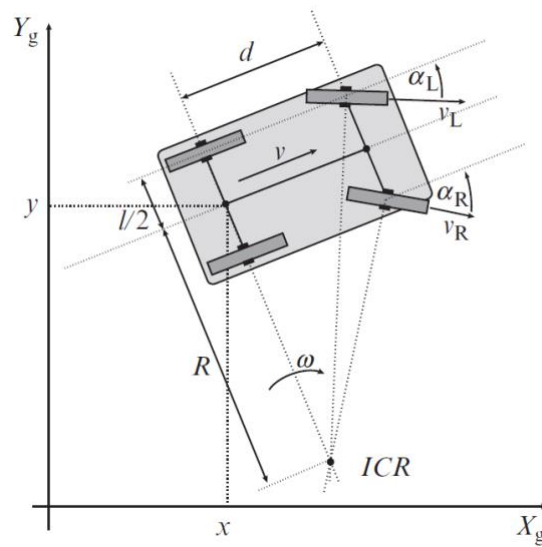


Figura 16 - Cinemática do robô Ackerman [59]

A condição de direcção sem deslizamento, ou seja, condição de Ackerman é expressa pela equação (3.36) [75].

$$\cot(\alpha_o) - \cot(\alpha_i) = \frac{l}{d} \quad (3.36)$$

Onde:

- α_o e α_i representam o ângulo de direção da roda externa e interna, respetivamente;
- l representa a distância entre o centro das rodas dianteiras;
- d caracteriza a distância entre o eixo frontal e o eixo traseiro.

Os ângulos da roda dianteira esquerda, α_L , e direita, α_R , ainda podem ser calculados através da equação (3.37) e da equação (3.38), respetivamente [59].

$$\alpha_L = \frac{\pi}{2} - \tan^{-1} \left(\frac{R + \frac{l}{2}}{d} \right) \quad (3.37)$$

$$\alpha_R = \frac{\pi}{2} - \tan^{-1} \left(\frac{R - \frac{l}{2}}{d} \right) \quad (3.38)$$

O centro de massa do robô móvel aquando da ocorrência de uma mudança de direção realizará uma curva com raio dado pela equação (3.39) [75].

$$R = \sqrt{\frac{d^2}{2} + d^2 \cot^2 \alpha} \quad (3.39)$$

Sendo que, o ângulo α ilustra o ângulo de rotação do robô e é determinado pela média do ângulo de Ackerman, equação (3.40) [59].

$$\alpha = \frac{\pi}{2} - \tan^{-1} \left(\frac{R}{d} \right) \quad (3.40)$$

Por sua vez, as duas rodas traseiras circulam em torno do ponto ICR com a mesma velocidade angular, enquanto as velocidades lineares das rodas dianteiras esquerda e direita são dadas conforme a equação (3.41) e a equação (3.42), respetivamente.

$$v_L = \omega \left(R + \frac{l}{2} \right) \quad (3.41)$$

$$v_R = \omega \left(R - \frac{l}{2} \right) \quad (3.42)$$

O modelo de movimento do mecanismo Ackerman pode ser descrito usando a mesma cinemática da configuração de triciclo. Dessa forma, o modelo cinemático na estrutura do robô é traduzido pelas equações anteriores (3.19), (3.20) e (3.21) e o modelo cinemático na estrutura do mundo pelas equações (3.22), (3.22) e (3.24), onde é utilizada a média do ângulo de Ackerman determinada com recurso à equação (3.40) [59].

Consequentemente, a determinação da cinemática direta também segue o princípio de integração semelhante ao caso do robô diferencial, sendo assim descrita pelas equações (3.11), (3.12) e (3.13) [59].

Por outro lado, a cinemática inversa deste tipo de mobilidade é bastante complexa, não existindo expressões matemáticas explícitas que a descreva.

3.2.4. ROBÔ SÍNCRONO

A capacidade de um robô móvel direcionar cada uma das suas rodas em torno do eixo vertical de forma síncrona, ou seja, as rodas apresentarem sempre a mesma direção e girarem em uníssono, é chamada de transmissão síncrona. A configuração típica deste tipo de transmissão possui três rodas distribuídas simetricamente ao longo da estrutura do robô, tendo por base os vértices de um triângulo equilátero [59]. A direção das rodas e a sua rotação na mesma velocidade é controlada pelo veículo, o que geralmente é conseguido com o recurso de um conjunto de correias que conectam fisicamente as rodas [76]. Dado que, todas as rodas permanecem paralelas, o robô de transmissão síncrona gira em torno do seu centro e o ponto ICR situa-se no infinito [73][59].

O veículo com transmissão síncrona é composto por um arranjo mecânico de dois motores independentes, sendo um motor implementado para impor a velocidade das três rodas e o segundo para controlar a rotação das mesmas rodas em torno dos seus eixos verticais [77]. Este controlo independente da velocidade linear e da rotação do robô móvel simplifica

o controle geral do mesmo e permite um modelo conveniente para experiências de simulação [76].

Os robôs móveis com esta configuração de transmissão são sensíveis às mais pequenas variações no plano do solo, uma vez que as oscilações provocam problemas de alinhamento das rodas devido à distância que cada roda percorre ser diferente [76].

A Figura 17 apresenta a disposição das rodas em torno da estrutura do robô.

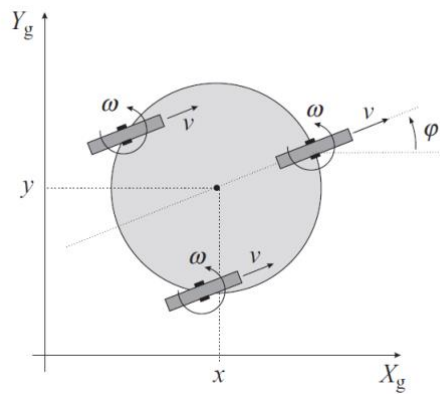


Figura 17 - Cinemática do robô síncrono [59]

As variáveis de controle deste mecanismo são a velocidade angular das rodas, ω , e a velocidade linear, v , em função do tempo [59].

A cinemática do robô síncrono é semelhante à cinemática do robô diferencial, porém no caso em questão as variáveis de controle podem ser controladas independentemente e na transmissão diferencial isso não é possível. Por essa razão, o modelo cinemático do robô é descrito pela equação (3.10) [59].

Cinemática Direta

A cinemática direta possibilita a determinação da posição do robô móvel a partir de certos valores das variáveis de controle e esta é obtida pela integração do modelo cinemático. Dessa forma, a cinemática direta do robô móvel síncrono é também conseguida pelas equações (3.11), (3.12) e (3.13).

Cinemática Inversa

Como as variáveis de controlo são controladas independentemente, a mudança de direção é completamente dissociada da translação e essa particularidade torna a cinemática inversa de um veículo com transmissão síncrona idêntica ao caso especial da cinemática inversa do robô com transmissão diferencial [76].

Por outras palavras, a cinemática inversa é resolvida para o caso especial em que o robô gira no mesmo local ou move-se retilineamente em uma direção. Quando o robô gira no lugar com velocidade angular, ω , constante por algum tempo Δt , a orientação deste altera-se ao longo de $\omega\Delta t$ e com velocidade linear nula. No caso do movimento retilíneo, se o veículo se mover com velocidade linear, v , constante, por algum tempo Δt , este desloca-se ao longo de uma distância $v\Delta t$ no sentido da sua orientação atual e com velocidade angular nula [59][76].

3.2.5. ROBÔ OMNIDIRECIONAL

Nas configurações de rodas descritas anteriormente, apenas eram utilizadas rodas simples, ou seja, rodas que só se podem deslocar na direção da sua orientação. No entanto, para alcançar um movimento de deslocação omnidirecional é necessária uma roda mais complexa. Um exemplo dessa roda é a roda *omni* ou *poly* (Figura 18). Estas rodas são compostas por uma série de rolos externos emborrachados presos a toda a circunferência do aro, de modo que os seus eixos estejam perpendiculares ao eixo da roda, permitindo assim que a roda realize a rotação normal, mas também o movimento livre na direção do eixo da roda [59][78].



Figura 18 - Roda *omni* ou *poly* [79]

Outro exemplo de uma roda omnidirecional é a roda *Mecanum* ou também conhecida como roda sueca, em que os rolos passivos são implementados em volta do aro da roda com um ângulo de 45° em relação ao eixo da roda [59]. A plataforma *Mecanum* é composta por quatro rodas, das quais duas rodas devem ser *left-handed* (com os rolos a uma inclinação de $+45^\circ$) e as outras duas *right-handed* (com os rolos a uma inclinação de -45°), onde as rodas da mesma diagonal são do mesmo tipo [78]. Esta característica permite ao robô ser um tipo de robô holonômico, quer isto dizer que tem a capacidade de se deslocar simultaneamente e de forma independente em várias direções de movimento diferentes ou, por outras palavras, movimentar-se para todos os lados sem precisar de rodar em torno do seu próprio eixo [80].

Este tipo específico de rodas pode-se visualizar através da Figura 19.



Figura 19 - Roda *Mecanum* [81]

O controlo de cada uma das rodas é feito de forma individual, tendo cada uma das quatro rodas um motor acoplado. A força produzida pela rotação das rodas é transferida para os rolos, onde nestes a força pode ser dividida em dois vetores. Mais concretamente, um vetor perpendicular ao eixo do rolo e outro paralelo ao mesmo. A combinação das forças resultantes especifica o movimento global do veículo [78].

O movimento para a frente ou para trás de um robô omnidirecional é conseguido através do ajuste de todas as rodas principais na mesma velocidade. No caso em que as velocidades das rodas principais de um lado são opostas às velocidades das rodas do outro lado, o robô roda. Por fim, para a execução do movimento lateral da plataforma é necessário que as rodas em uma diagonal tenham velocidade oposta às rodas na outra diagonal. A partir da combinação dos procedimentos descritos, é possível que o robô móvel consiga efetuar o movimento em qualquer direção e com qualquer rotação [59].

A Figura 20 representa os principais movimentos do robô com configuração omnidirecional, especificando o sentido de rotação de cada uma das rodas.

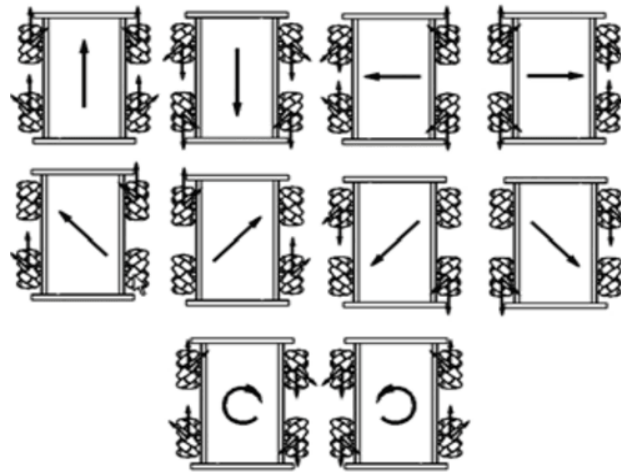


Figura 20 - Movimentos do robô omnidirecional [82]

Cinemática do robô com quatro rodas *Mecanum*

O modelo cinemático do robô com quatro rodas *Mecanum* é representado na Figura 21, identificando as seguintes variáveis [59]:

- v_1, v_2, v_3, v_4 são as velocidades lineares das quatro rodas;
- R representa o raio das rodas;
- v_R é a velocidade perpendicular ao eixo do rolo;
- φ é a posição angular do robô;
- l caracteriza a distância entre o centro do robô até ao eixo que passa no centro da roda;
- d representa metade do comprimento do eixo entre as rodas do robô;

- x_m, y_m são as coordenadas do robô;
- γ retrata o ângulo entre o eixo do robô e o eixo da roda.

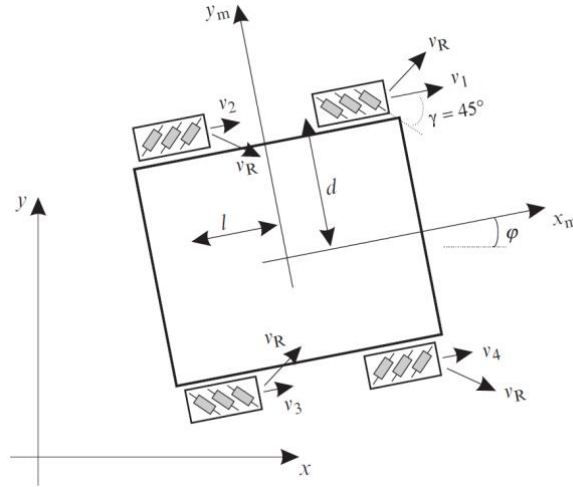


Figura 21 - Cinemática do robô omnidirecional [59]

A velocidade da roda 1 nas coordenadas do robô é determinada a partir da velocidade linear da roda $v_1(t)$ e da velocidade dos rolos $v_R(t)$. Como descrito anteriormente, quando se move as rodas *Mecanum*, os rolos convertem uma fração da força de rotação da roda em uma força perpendicular aos eixos dos mesmos. Tendo em consideração que os eixos passivos dos rolos apresentam um ângulo de $\gamma = 45^\circ$ em relação ao eixo da roda principal, consegue-se definir a velocidade total da roda na direção x_m e y_m da estrutura de coordenadas do robô. As equações (3.43) e (3.44) determinam essas velocidades para o caso da roda 1 [59].

$$v_{m1x} = v_1(t) + v_R(t) \cos\left(\frac{\pi}{4}\right) = v_1(t) + \frac{v_R(t)}{\sqrt{2}} \quad (3.43)$$

$$v_{m1y} = v_R(t) \sin\left(\frac{\pi}{4}\right) = \frac{v_R(t)}{\sqrt{2}} \quad (3.44)$$

Seguidamente, a velocidade da roda principal é obtida através da equação (3.45).

$$v_1(t) = v_{m1x} - v_{m1y} \quad (3.45)$$

Por outro lado, a velocidade da roda na direção da estrutura de coordenadas do robô também pode ser expressa não só por meio da velocidade de translação do robô, equação (3.46), mas também pela velocidade angular $\dot{\phi}(t)$, equação (3.47) e (3.48) [59].

$$v_m(t) = \sqrt{\dot{x}_m^2(t) + \dot{y}_m^2(t)} \quad (3.46)$$

$$v_{m1x} = \dot{x}_m(t) - \dot{\phi}(t)d \quad (3.47)$$

$$v_{m1y} = \dot{y}_m(t) - \dot{\phi}(t)l \quad (3.48)$$

A partir das equações das velocidades lineares e angulares da direção de cada roda descritas, as velocidades das quatro rodas podem ser deduzidas pela velocidade do corpo do robô, equações (3.49) - (3.52) [59][78].

$$v_1(t) = \dot{x}_m(t) - \dot{y}_m(t) - (l + d)\dot{\phi}(t) \quad (3.49)$$

$$v_2(t) = \dot{x}_m(t) + \dot{y}_m(t) - (l + d)\dot{\phi}(t) \quad (3.50)$$

$$v_3(t) = \dot{x}_m(t) - \dot{y}_m(t) + (l + d)\dot{\phi}(t) \quad (3.51)$$

$$v_4(t) = \dot{x}_m(t) + \dot{y}_m(t) + (l + d)\dot{\phi}(t) \quad (3.52)$$

Cinemática Inversa

Tendo em consideração as equações anteriores, a cinemática inversa nas coordenadas locais é apresentada segundo a forma matricial a partir da equação (3.53) [59].

$$\begin{bmatrix} v_1(t) \\ v_2(t) \\ v_3(t) \\ v_4(t) \end{bmatrix} = \begin{bmatrix} 1 & -1 & -(l + d) \\ 1 & 1 & -(l + d) \\ 1 & -1 & (l + d) \\ 1 & 1 & (l + d) \end{bmatrix} \begin{bmatrix} \dot{x}_m(t) \\ \dot{y}_m(t) \\ \dot{\phi}(t) \end{bmatrix} \quad (3.53)$$

Esta equação matricial pode ser traduzida para o formato compacto, de acordo com a equação (3.54).

$$v(t) = J\dot{q}_m(t) \quad (3.54)$$

Em que:

- v^T representa o vetor $[v_1, v_2, v_3, v_4]^T$;
- $\dot{q}_m^T$ simboliza o vetor $[\dot{x}_m, \dot{y}_m, \dot{\phi}]^T$.

No caso da obtenção da cinemática inversa nas coordenadas globais, esta pode ser conseguida com o recurso da equação (3.55) no formato compacto [59].

$$v(t) = JR_G^L \dot{q}(t) \quad (3.55)$$

Onde a matriz de rotação ortogonal R_G^L , equação (3.56), é definida para mapear a estrutura do mundo para a estrutura do robô. Dito de outro modo, representa a orientação das coordenadas locais em relação às coordenadas globais.

$$R_G^L = \begin{bmatrix} \cos \varphi & \sin \varphi & 0 \\ -\sin \varphi & \cos \varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.56)$$

Cinemática Direta

A equação da cinemática direta do robô omnidirecional para as coordenadas locais é descrita pela equação (3.57) [59].

$$\dot{q}_m(t) = J^+ v(t) \quad (3.57)$$

Sendo $J^+ = (J^T J)^{-1} J^T$ a matriz pseudo inversa de J , isto é, uma generalização da matriz inversa.

Na equação (3.58) encontra-se representada na forma matricial a equação da cinemática direta do robô omnidirecional para as coordenadas locais [59].

$$\begin{bmatrix} \dot{x}_m(t) \\ \dot{y}_m(t) \\ \dot{\varphi}(t) \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{(l+d)} & \frac{1}{(l+d)} & \frac{1}{(l+d)} & \frac{1}{(l+d)} \end{bmatrix} \begin{bmatrix} v_1(t) \\ v_2(t) \\ v_3(t) \\ v_4(t) \end{bmatrix} \quad (3.58)$$

Por sua vez, a cinemática direta nas coordenadas globais é determinada mediante a equação (3.59).

$$\dot{q}(t) = (R_G^L)^T J^+ v(t) \quad (3.59)$$

3.3. COMPARAÇÃO ENTRE OS VÁRIOS TIPOS DE ROBÔS MÓVEIS

Tendo em consideração os tipos de robôs móveis descritos anteriormente, na Tabela 2 está apresentada uma comparação entre todos, em termos de características positivas e negativas.

Tabela 2 - Comparação dos tipos de robôs móveis

Tipo de robô móvel	Características positivas	Características negativas
Robô Diferencial	• Estável em superfícies uniformes; • Contacto constante com o piso; • Baixa força de fricção durante a mudança de direção; • Rotação em torno de um eixo vertical; • Altamente direcionável; • Controlo simples.	• Instável em superfícies irregulares; • Alteração da direção do robô com a mínima variação das rodas.
Robô Triciclo	• Estável em superfícies uniformes; • Contacto constante com o piso; • Altamente direcionável; • Rotação em torno de um eixo vertical; • Alteração de direção sem efeitos colaterais.	• Instável em superfícies irregulares; • Alta força de fricção durante a mudança de direção; • Perda de tração em superfícies irregulares; • Possibilidade de erro na determinação da posição.

Robô Ackerman	• Estável quer em superfícies uniformes, quer em superfícies irregulares; • Forças laterais neutralizadas; • Alteração imprevista da velocidade das rodas não causa erros na direção do robô.	• Limitação no ângulo de direção; • Alta força de fricção durante a mudança de direção.
Robô Síncrono	• Estável em superfícies uniformes; • Todas as rodas apresentam sempre a mesma velocidade e direção; • Utilização de 2 motores simplifica o controlo geral do robô.	• Pequenas variações no solo provocam problemas de alinhamento das rodas; • Instável em superfícies irregulares.
Robô Omnidirecional	• Direção omnidirecional; • Estável em superfícies planas; • Alta capacidade de manobra do robô; • Deslocação em diferentes direções com pouco atrito.	• Alteração imprevista da velocidade das rodas provoca erros no percurso do robô; • Alta sensibilidade às irregularidades do piso.

As capacidades principais analisadas no desempenho de robôs móveis com rodas são a estabilidade, controlabilidade e manobrabilidade. Da análise da tabela comparativa, pode-se constatar que, em geral, existe uma correlação inversa entre a controlabilidade e a manobrabilidade, isto é, se o veículo é fácil de controlar, então é menos manobrável e vice-versa. No que toca à estabilidade, pode-se verificar que um robô com três rodas em

contacto com o solo obtém estabilidade vertical. No entanto, esta estabilidade é sensível a irregularidades no plano do solo, e por essa razão, a estabilidade aprimorada é apenas conseguida por arquiteturas de locomoção com 4 ou mais rodas [83].

Em suma, não existe um sistema de locomoção perfeito que otimize simultaneamente a capacidade de estabilidade, controlabilidade e manobrabilidade. Assim sendo, a configuração do robô móvel mais conveniente a uma aplicação deve ser escolhida tendo em consideração todos os compromissos e dificuldades em questão.

Para o desenvolvimento deste projeto, escolheu-se implementar um robô móvel com acionamento diferencial. Esta configuração é amplamente utilizada em superfícies planas e uniformes, uma vez que demonstra-se ser estável nestes ambientes, requer uma estratégia de controlo simples e é capaz de se deslocar em torno de um eixo vertical, alcançando satisfatoriamente todas as capacidades elementares de um robô.

3.4. MODELAÇÃO DINÂMICA DO ROBÔ MÓVEL COM TRAÇÃO DIFERENCIAL

A dinâmica é o ramo da física que estuda o movimento de um sistema mecânico sob ação das diferentes forças. Esta subdivisão da Física compreende a cinemática, que analisa matematicamente o movimento dos corpos sem ter em consideração as forças que promovem o movimento, e a cinética que relaciona estas forças com os movimentos produzidos [67].

As leis básicas da dinâmica podem ser expressas a partir de vários princípios, tais como as leis de Newton, o princípio de D'Alembert, as equações de Lagrange e o princípio de Hamilton. No entanto, todas estas formulações estão fundamentadas nas leis de Newton e no princípio do trabalho virtual, isto é, o trabalho virtual das forças externas é equivalente ao trabalho virtual das forças internas. Na área da robótica, aplicam-se essencialmente as formulações de Lagrange e de Newton-Euler [68]. Nomeadamente, neste projeto foi escolhido implementar a abordagem da segunda Lei de Newton, equação (3.60), que integra parâmetros como os efeitos da massa do robô e do momento de inércia [70].

$$F_i = ma_i \quad (3.60)$$

Onde m é a massa do robô móvel e a a aceleração do mesmo.

Com base nas relações das equações, é possível desenvolver o modelo dinâmico do robô móvel com tração diferencial, Figura 22, sendo que este é assentado nos binários de tração do motor de cada uma das rodas.

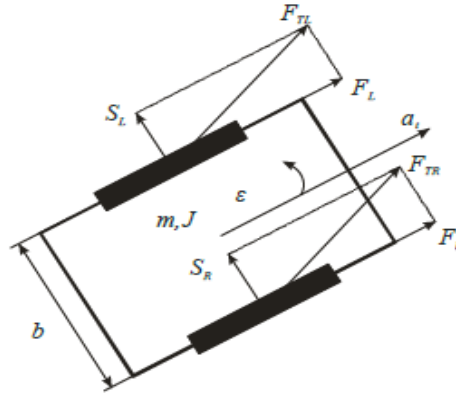


Figura 22 - Modelo dinâmico do robô móvel [66]

Em termos das forças aplicadas às rodas, a aceleração linear pode ser expressa conforme a equação (3.61).

$$\dot{v} = \frac{F_R}{m} + \frac{F_L}{m} \quad (3.61)$$

Em que, F_L e F_R são as forças aplicadas à roda esquerda e à roda direita e a partir destas é também possível definir-se os binários das rodas, equações (3.62) e (3.63). No entanto, o binário total do robô é especificado como a soma dos binários de cada uma das rodas, tal como é exposto na equação (3.64) [70].

$$\tau_R = F_R \times \frac{2L}{2} \quad (3.62)$$

$$\tau_L = -F_L \times \frac{2L}{2} \quad (3.63)$$

$$\tau_{Robot} = \tau_R + \tau_L \quad (3.64)$$

Em que, τ_R caracteriza o binário da roda direita e τ_L é o binário da roda esquerda.

No que toca à aceleração angular, esta pode ser expressa em função das forças de tração e do momento de inércia do robô, J , da seguinte forma [70]:

$$\dot{\omega} = \frac{F_R \times 2L}{2J} - \frac{F_L \times 2L}{2J} \quad (3.65)$$

Por fim, com a implementação da segunda Lei de Newton, consegue-se obter o binário total do robô a partir da aceleração angular segundo a equação (3.66) [70].

$$\tau_{Robot} = J \times \dot{\omega} \quad (3.66)$$

4. IMPLEMENTAÇÃO

No presente capítulo é apresentada a forma de implementação não só dos modelos expostos no capítulo anterior, mas também do modelo dos atuadores do robô móvel. Ainda neste capítulo, é explicada a estrutura geral do sistema e o método de otimização do mesmo.

4.1. ESQUEMA EM DIAGRAMA DE BLOCOS DA SIMULAÇÃO

O modelo matemático do robô móvel com tração diferencial é uma aproximação do robô móvel real, que se baseia em um modelo de controlo da trajetória em conjunto com o

modelo dos atuadores do robô, o modelo dinâmico e o modelo cinemático do robô. O diagrama de blocos do robô está apresentado na Figura 23.

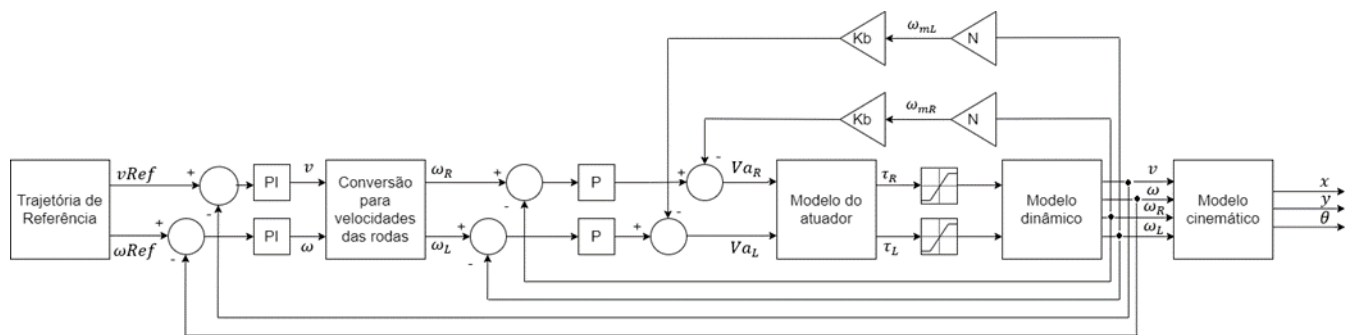


Figura 23 - Diagrama de blocos

As entradas de referência do sistema são as velocidades lineares e as velocidades angulares desejadas calculadas a partir dos valores de posição para as diferentes trajetórias de referência. Posteriormente, as velocidades de referência são comparadas com os valores das velocidades reais obtidas na saída do subsistema do modelo dinâmico e o erro de cada variável é controlado por um controlador PI. Este controlador foi estipulado como PI, para que o erro em regime permanente da malha de realimentação conseguisse ser nulo e, dessa forma, o sistema atingisse os valores de referência na saída.

Os sinais de saída do controlador são depois convertidos para as velocidades angulares das rodas. Neste ponto do sistema foi desenvolvida uma realimentação interna com o intuito de suprimir a dinâmica do robô, portanto as velocidades angulares das rodas fornecidas pelo controlador são comparadas com as velocidades angulares reais. O controle do erro em regime permanente gerado na realimentação interna foi implementado com recurso a dois controladores apenas com a componente P para facilitar o processamento da simulação do sistema. A saída deste controlador fornece as tensões de armadura dos motores aplicados no robô e, seguidamente, esta grandeza é também comparada com as tensões de armadura calculadas a partir das velocidades das angulares na saída do subsistema do modelo dinâmico.

A partir do subsistema do modelo dos atuadores, converte-se as tensões de armadura para os binários das rodas do robô móvel que afetam a dinâmica deste com o objetivo de percorrer uma trajetória, determinada no planeamento de trajetória.

Por fim, a saída da realimentação interna são as velocidades na saída do modelo dinâmico que depois são transformadas pelo modelo cinemático na posição real do robô.

No desenvolvimento do presente projeto foi aplicada a especificação real do robô Khepera IV [84] para a simulação do modelo obter os resultados mais realistas possíveis, especificamente os parâmetros cinemáticos e dinâmicos são os apresentados na Tabela 3.

Tabela 3 - Parâmetros do robô móvel diferencial

Massa do robô (m)	0,9 kg
Momento de inércia (J)	0,01 kgm ²
Raio das rodas (r)	0,021 m
Metade da distância entre os eixos (L)	0,0725 m

4.2. PLANEAMENTO DE TRAJETÓRIA

No planeamento da trajetória, vários trajetos diferentes foram implementados [69] para serem introduzidos no modelo do controlador como trajetória de referência com o objetivo de analisar as capacidades do sistema. Os percursos utilizados estão descritos na Tabela 4.

Tabela 4 - Trajetórias aplicadas ao sistema

Trajétória em círculo	$x(t) = 0,3 \sin(0,3t)$ $y(t) = 0,3 \cos(0,3t)$
Trajétória em forma de oito	$x(t) = 0,5 \sin(0,3t)$ $y(t) = 0,5 \sin(0,3t) \cos(0,3t)$
Trajétória em espiral	$x(t) = 0,13t \cos(0,3t)$ $y(t) = 0,13t \sin(0,3t)$

As várias trajetórias de referência implementadas estão representadas graficamente na Figura 24, na Figura 25 e na Figura 26.

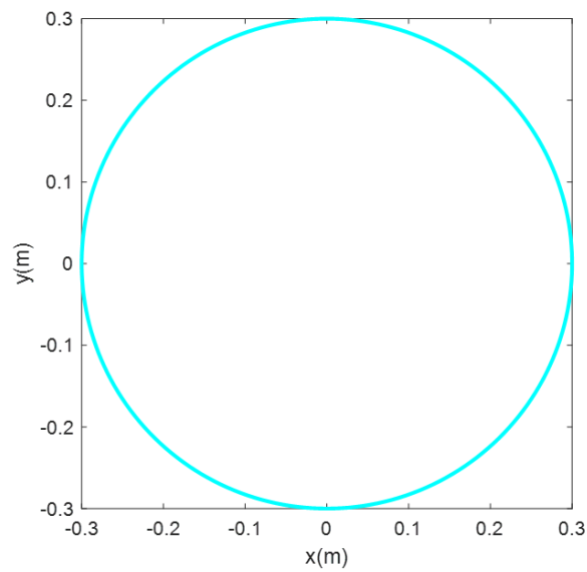


Figura 24 - Trajetória em círculo

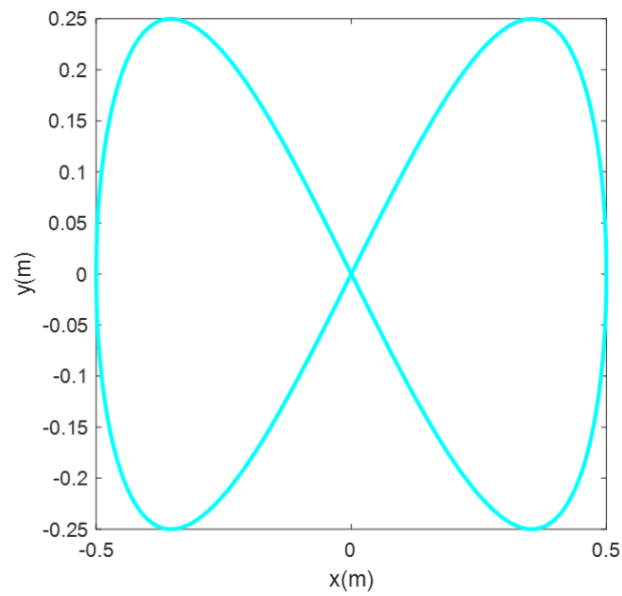


Figura 25 - Trajetória em forma de oito

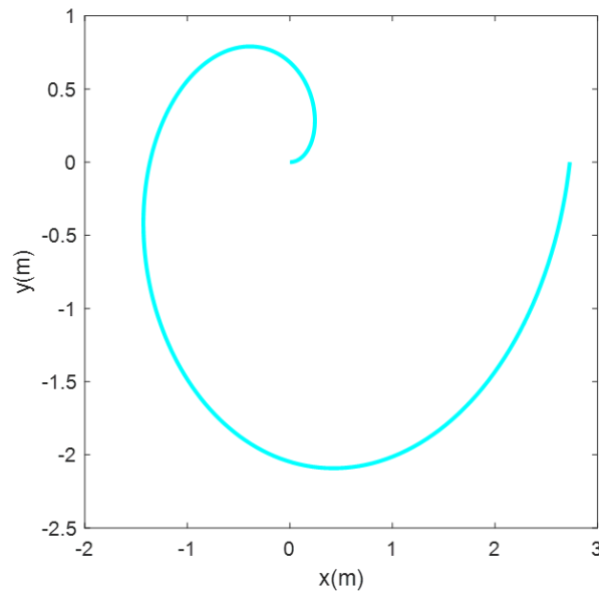


Figura 26 - Trajetória em espiral

A partir das coordenadas das posições $x(t)$ e $y(t)$, as equações (3.14) e (3.15) foram deduzidas através de programação, com o intuito de determinar as velocidades lineares, $v(t)$, e as velocidades angulares, $\omega(t)$, do robô. Seguidamente, os valores dessas velocidades são introduzidos no sistema.

A programação MATLAB desenvolvida para o planeamento da trajetória pode ser encontrada no Anexo A.

4.3. MODELO DE CONVERSÃO PARA AS VELOCIDADES ANGULARES DAS RODAS EM SIMULINK

A partir das velocidades geradas na saída do controlador foi realizada a transformação destas para as velocidades angulares da roda direita, $\omega_R(t)$, e da roda esquerda, $\omega_L(t)$. Para isso, neste bloco foram descritas em Simulink as equações (3.1), (3.2) e, ainda, as equações (3.7) e (3.8), como mostra a Figura 27.

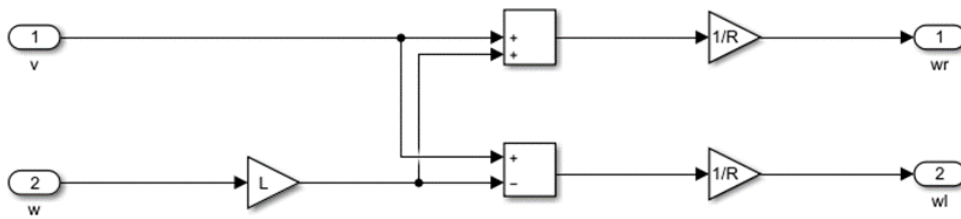


Figura 27 - Conversão para as velocidades angulares das rodas

4.4. MODELO DO ATUADOR EM SIMULINK

O modelo dos atuadores do robô móvel possibilita a simulação da dinâmica dos atuadores, mais especificamente motores CC, para uma maior aproximação da realidade.

Os motores de CC de ímã permanente são geralmente usados para acionar as rodas do sistema de um robô móvel diferencial. O motor selecionado para a implementação foi o 2619 SR da Faulhaber [85]. Este atuador foi escolhido tendo como critério a tensão nominal de 12 V e um binário de saída capaz de aguentar a massa do robô descrita na Tabela 3.

A Tabela 5 apresenta as características principais do atuador escolhido.

Tabela 5 - Principais características do atuador

Tensão nominal (V)	12 V
Resistência de armadura (R_a)	36,5 Ω
Indutância de armadura (L_a)	0,0022 H
Constante de binário (K_t)	0,019 Nm/A
Constante de força contraelectromotriz (K_b)	0,019 Vs/rad
Razão de engrenagem (N)	112:1
Saturação do binário do motor (M_{max})	0,093 Nm

Em um motor CC controlado por armadura, as equações (4.1) - (4.4) são aplicadas para representação do circuito da armadura, sendo que a tensão da armadura é a entrada de controlo [67].

$$V_a = R_a i_a + L_a \frac{di_a}{dt} + e_a \quad (4.1)$$

$$e_a = K_b \omega_m \quad (4.2)$$

$$\tau_m = K_t i_a \quad (4.3)$$

$$\tau = N \tau_m \quad (4.4)$$

Onde:

- V_a é a tensão da armadura;
- R_a e L_a ilustram a resistência e a indutância da armadura, respetivamente;
- i_a caracteriza a corrente da armadura;
- e_a representa a força contraelectromotriz;
- K_t e K_b simbolizam as constantes de binário e de força contraelectromotriz, respetivamente;
- ω_m é a velocidade angular do rotor;
- τ_m e τ representam o binário de motor e o binário de saída aplicado às rodas, respetivamente;
- N é a razão de transmissão.

Tendo em consideração que, nos robôs móveis os motores são conectados mecanicamente à roda através de engrenagens, pode concluir-se que as equações mecânicas do movimento dos motores estão relacionadas diretamente com a dinâmica mecânica do robô diferencial. Portanto, ao motor da roda direita e ao motor da roda esquerda aplicam-se as equações (4.5) e (4.6), respetivamente [67].

$$\omega_{mR} = N \omega_R \quad (4.5)$$

$$\omega_{mL} = N \omega_L \quad (4.6)$$

A Figura 28 apresenta a forma como o modelo dos atuadores foi incorporado no sistema geral.

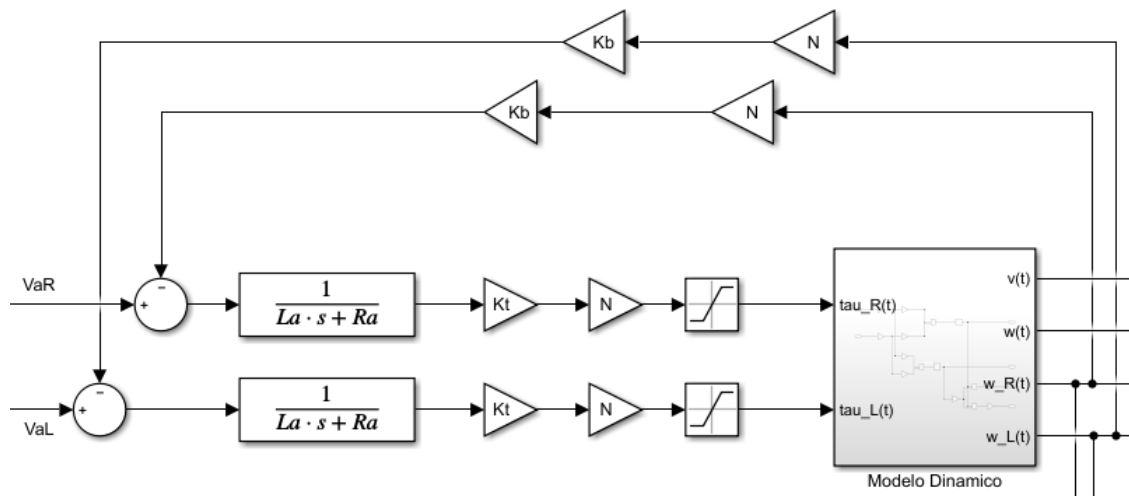


Figura 28 - Modelo do atuador

4.5. MODELO DINÂMICO EM SIMULINK

O modelo dinâmico do sistema permite que o movimento do corpo do robô seja definido tendo em consideração os binários aplicados às rodas.

Dessa forma, as entradas deste subsistema são o binário da roda direita, τ_R , e o binário da roda esquerda, τ_L . Com recurso às equações (3.62), (3.63), (3.61), (3.65), (3.5) e (3.6) apresentadas no capítulo anterior, foi possível deduzir a velocidade linear e a velocidade angular a partir do binário das rodas, sendo estas aplicadas ao modelo cinemático como entrada do subsistema.

A construção deste modelo em Simulink está representada na Figura 29.

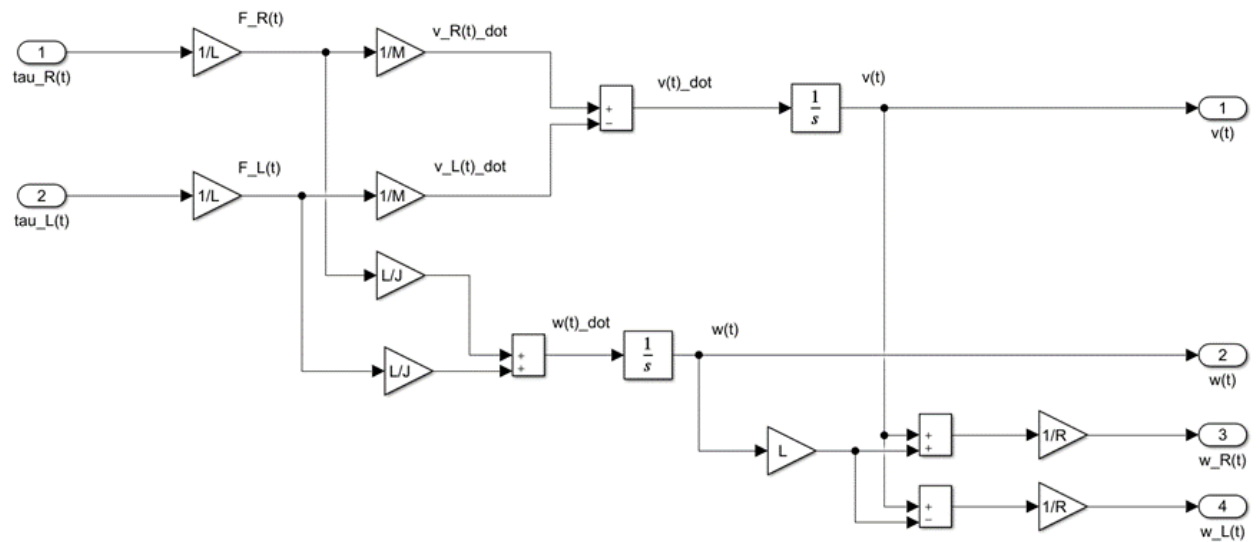


Figura 29 - Modelo dinâmico

4.6. MODELO CINEMÁTICO EM SIMULINK

O modelo cinemático aplicado garante que a posição do robô em um determinado momento de tempo seja determinada a partir da velocidade linear e da velocidade angular do robô no mesmo instante.

Por conseguinte, as entradas do subsistema do modelo cinemático são a velocidade linear e a velocidade angular do robô e, ainda, as velocidades angulares de cada uma das rodas. As saídas do subsistema são as coordenadas reais da posição do robô, $x(t)$ e $y(t)$, e a orientação do mesmo, $\theta(t)$.

Para a construção do modelo foram modeladas em Simulink as equações (3.11), (3.12) e (3.13) descritas anteriormente, conforme a Figura 30.

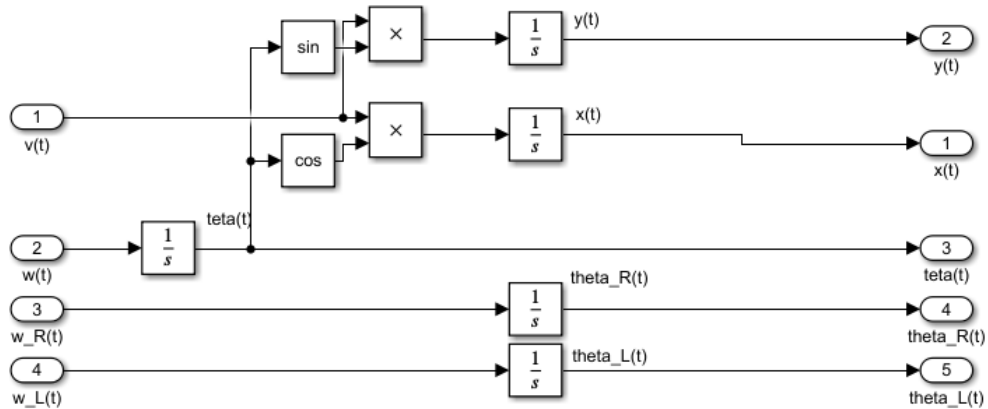


Figura 30 - Modelo cinemático

4.7. ALGORITMO EVOLUTIVO

A implementação de cada um dos algoritmos evolutivos é realizada com recurso a dois ficheiros, nomeadamente, o ficheiro *script* principal e o ficheiro da função objetivo, que são utilizados separadamente para encontrar a solução ótima do sistema.

4.7.1. FICHEIRO *SCRIPT* PRINCIPAL

No ficheiro principal do algoritmo é onde são chamadas as outras funções adjacentes ao programa.

A implementação do AG e do PSO foi efetuada com recurso à *toolbox* de otimização incorporada no *software* MATLAB denominada “*optimtool*”. Por essa razão, a configuração dos parâmetros destes algoritmos foi realizada de forma semelhante.

A configuração dos parâmetros do AG e sua execução foi realizada conforme o código a seguir apresentado.

```
FuncObj= @ObjectiveFunction_GA_Motor;
nvars= 6;
lb= [0 0 0 0 0 0];
ub= [100 100 100 100 100 100];
```

```
options = optimoptions('ga','PopulationSize',50,
'MaxGenerations',60,'MaxStallGenerations',60);

[xopt, fval]=
ga(FuncObj,nvars,[],[],[],[],lb,ub,[],options)
```

O código descrito pode ser explicado da seguinte forma:

- `FuncObj` é a indexação do ficheiro da função objetivo;
- `nvars` representa o número de variáveis a controlar;
- `lb` e `ub` são os vetores que retratam os limites inferiores e superiores das variáveis independentes, respetivamente;
- `options` são as definições que se pretende para o AG, em substituição das definições estipuladas por omissão no MATLAB. Em particular, `PopulationSize` é o tamanho da população do AG, `MaxGenerations` representa o número de gerações que o algoritmo executa e `MaxStallGenerations` impede o algoritmo de terminar antes de realizar todas as gerações.
- `xopt` ilustra a solução para os ganhos do controlador;
- `fval` é o valor da função objetivo calculada para a solução `xopt`.

Seguidamente, para a configuração dos parâmetros do PSO, a diferença na programação acontece apenas na chamada da função de otimização e no vetor de opções do algoritmo. O código referente à configuração dos parâmetros do PSO, bem como a sua execução é apresentado a seguir.

```
FuncObj= @ObjectiveFunction_PSO_Motor;
nvars= 6;
lb= [0 0 0 0 0 0];
ub= [100 100 100 100 100 100];
options =
optimoptions('particleswarm','SwarmSize',50,'MaxI
terations',60,'MaxStallIterations',60);

[xopt,fval]=
particleswarm(FuncObj,nvars,lb,ub,options)
```

Em que, no vetor de opções do algoritmo, `SwarmSize` é o número de partículas do enxame, `MaxIterations` ilustra o número máximo de iterações do algoritmo e, ainda, `MaxStallIterations` garante que o algoritmo execute o número de iterações pretendidas.

Por sua vez, o algoritmo CS foi implementado a partir de um modelo denominado “cuckoo_search_new”, que foi desenvolvido pelo próprio autor deste algoritmo e que pode ser livremente acedido [86]. Assim sendo, no ficheiro *script* principal, o modelo do CS é chamado segundo a sintaxe seguinte:

```
[xopt,fval]=cuckoo_search_new(50)
```

Sendo que, o valor de entrada para o modelo representa o número de ninhos do algoritmo. Os restantes parâmetros de configuração do CS são implementados no ficheiro da função de otimização através do excerto de código exposto a seguir. A programação completa deste ficheiro pode ser consultada no Anexo B.

```
N_IterTotal=60;  
%% Simple bounds of the search domain  
% Lower bounds  
nd=6;  
Lb=0*ones(1,nd);  
% Upper bounds  
Ub=100*ones(1,nd);
```

Onde, à semelhança dos algoritmos evolutivos descritos anteriormente, *N_IterTotal* caracteriza o número de iterações do algoritmo, *nd* é o número de variáveis a controlar e *Lb* e *Ub* representam, respetivamente, os vetores que correspondem aos limites inferiores e superiores das variáveis a controlar.

4.7.2. FUNÇÃO OBJETIVO

A função objetivo representa o critério de minimização do erro do sistema, ou seja, é a função que ajusta os ganhos do controlador PID para minimização de um determinado índice.

Para a otimização do modelo é utilizada como função objetivo a distância Euclidiana entre as posições de referência e as posições reais do robô de tração diferencial ao longo da simulação. Dessa forma, a distância Euclidiana foi calculada segundo a equação (4.7).

$$d = \sqrt{\sum_{k=1}^{N_a} (\Delta x_k^2 + \Delta y_k^2)} \quad (4.7)$$

Onde Δx_k é a diferença entre a posição k de referência e a posição k real do robô no eixo horizontal, Δy_k é a diferença segundo o eixo vertical da mesma posição de referência e posição real e N_a é o número de amostras ao longo da trajetória simulada.

Para a implementação da função objetivo, a equação (4.7) foi modelada através de uma função local que é indexada no ficheiro *script* principal. Todavia, para o algoritmo CS esta função não é implementada em uma função local, dado que no ficheiro da função de otimização deste algoritmo é identificada uma secção para desenvolvimento da função objetivo do sistema.

Todos os algoritmos seguem a mesma função objetivo, logo a programação correspondente a esta função é igual em todos os casos de estudo e é apresentada em seguida.

```
global temp xr yr kpv kiv kpw kiw Kpr Kpl

kpv = k(1);
kiv = k(2);
kpw = k(3);
kiw = k(4);
Kpr = k(5);
Kpl = k(6);

[t,xx,x,y,teta,v,w]=sim('Modelo_sim_Motor.slx',temp);

d = sum((xr(:)-x(:)).^2 + (yr(:)-y(:)).^2);
J = sqrt(d)
```


5. TESTES E RESULTADOS

Neste capítulo são apresentados um conjunto de testes realizados para os diferentes algoritmos, bem como para as diferentes trajetórias que têm por base a avaliação do sistema implementado. Seguidamente, são aplicados alguns índices de erro para determinação da sua capacidade para atingir uma solução ótima. Por fim, este capítulo termina com a exposição dos resultados obtidos ao longo dos testes e a posterior análise entre os mesmos e os resultados esperados.

5.1. METODOLOGIA DOS TESTES

De entre os algoritmos estudados, foram testados no controlo de navegação do robô móvel três algoritmos de otimização, especificamente o AG, o PSO e o CS.

Para simular a estrutura de controlo, é necessário determinar os parâmetros do algoritmo e, dessa forma, a sua determinação foi realizada tendo por base o Algoritmo Genético, mais concretamente, foi simulado este algoritmo com várias configurações dos parâmetros da população e das gerações. Seguidamente, foi selecionada uma configuração que apresentasse não só bons resultados finais da função objetivo, mas também convergência da mesma ao longo da simulação e, ainda, um tempo de simulação satisfatório. Por conseguinte, foi definida para simulação a configuração de 50 indivíduos e 60 gerações.

Uma vez estabelecidos os parâmetros a utilizar, estes foram implementados nos restantes algoritmos para comparação da solução. O desempenho de cada algoritmo simulado em MATLAB foi avaliado para as diferentes trajetórias através de vários critérios de desempenho.

A Tabela 6, a Tabela 7 e a Tabela 8 apresentam os parâmetros utilizados para inicialização do AG, do PSO e do CS, respetivamente.

Tabela 6 - Parâmetros de inicialização do AG

Parâmetro	Valor
Tamanho da população	50
Número de gerações	60
Número de variáveis (<i>nvars</i>)	6
Limite inferior das variáveis (<i>lb</i>)	0
Limite superior das variáveis (<i>ub</i>)	100
Tolerância da função	$1,0 \times 10^{-6}$
Probabilidade de cruzamento (p_c)	0,8
Número de elites	3

Tabela 7 - Parâmetros de inicialização do PSO

Parâmetro	Valor
Tamanho do enxame	50
Número de iterações	60
Número de variáveis ($nvars$)	6
Limite inferior das variáveis (lb)	0
Limite superior das variáveis (ub)	100
Tolerância da função	$1,0 \times 10^{-6}$
Extensão inicial do enxame	2000
Probabilidade mínima de vizinhos	0,25
Peso de ajuste individual (c_1)	1,49
Peso de ajuste social (c_2)	1,49
Intervalo de inércia (w)	[0,1; 1,1]

Tabela 8 - Parâmetros de inicialização do CS

Parâmetro	Valor
Número de ninhos (n)	50
Número de iterações ($N_IterTotal$)	60
Número de variáveis (nd)	6
Limite inferior das variáveis (Lb)	0
Limite superior das variáveis (Ub)	100
Probabilidade de o pássaro hospedeiro descobrir o ovo estrangeiro (pa)	0,25

5.1.1. ÍNDICES DE DESEMPENHO

Os índices de desempenho aplicados para avaliar a qualidade da resposta do sistema de controlo são o *Integral Absolute Error* (IAE), *Integral of Time Multiplied by Absolute Error* (ITAE) e *Integral Square Error* (ISE). As expressões dos índices de desempenho são descritas pelas equações (5.1), (5.2) e (5.3).

$$IAE = \int_0^{tsim} |e_x| dt + \int_0^{tsim} |e_y| dt \quad (5.1)$$

$$ITAE = \int_0^{tsim} |e_x| t dt + \int_0^{tsim} |e_y| t dt \quad (5.2)$$

$$ISE = \int_0^{tsim} e_x^2 dt + \int_0^{tsim} e_y^2 dt \quad (5.3)$$

Em que:

- $tsim$ representa o tempo de simulação;
- $e_x = x_r - x$ é a diferença entre a referência desejada (x_r) e o valor atual da posição (x) no eixo Ox.
- $e_y = y_r - y$ é a diferença entre a posição de referência no eixo Oy (y_r) e o valor real da mesma (y).

O índice IAE fornece o somatório do e_x e do e_y no tempo de forma linear. Enquanto, o índice ITAE insere o tempo como um fator de medida, ou seja, conforme t avança a penalização nos erros será maior, resultando em sistemas com *overshoot* pequeno e oscilações bem amortecidas. O ISE soma o quadrado de e_x e de e_y no tempo, o que realça os erros de maior grandeza e menospreza os erros de menor grandeza, caracterizando o sistema com uma rápida diminuição do erro inicial elevado.

O código em MATLAB que possibilita o cálculo dos erros da simulação é apresentado no Anexo C.

5.2. RESULTADOS OBTIDOS

Devido ao algoritmo evolutivo ser estocástico, considerou-se uma amostra de 25 simulações para cada algoritmo em cada trajetória, com o objetivo de analisar se a solução do algoritmo tende sempre para o mesmo resultado perante valores iniciais diferentes. As simulações foram realizadas com um tempo de simulação de 21 segundos para cada uma

das trajetórias e com um tempo de amostragem de 0,1 segundos. Nomeadamente, para todos os trajetos estipulados a posição inicial do robô está localizada no ponto inicial da trajetória de referência.

Ao longo das simulações anotou-se, para todos os algoritmos, todos os parâmetros do controlador para uma posterior análise. Estes parâmetros estão apresentados detalhadamente no Anexo D.

De forma a avaliar a dispersão das funções objetivo obtidas na amostra de simulações foi calculada a função objetivo mínima e máxima, bem como a média da amostra e o desvio-padrão. Na Tabela 9 resumiu-se estes valores para os vários algoritmos para uma melhor comparação entre os mesmos.

Tabela 9 - Análise estatística das funções objetivo das simulações

Trajetórias	Algoritmo evolutivo	Função objetivo (d)			
		Máxima	Mínima	Média	Desvio-padrão
Trajetória em círculo	GA	0,0109	0,0094	0,0096	$3,0564 \times 10^{-4}$
	PSO	0,0094	0,0094	0,0094	$8,5559 \times 10^{-8}$
	CS	0,0095	0,0094	0,0094	$1,9129 \times 10^{-5}$
Trajetória em forma de oito	GA	0,232	0,0126	0,0184	0,0029
	PSO	0,0197	0,0132	0,0177	0,0022
	CS	0,0193	0,0145	0,0166	0,0015
Trajetória em espiral	GA	0,0295	0,0191	0,0243	0,0030
	PSO	0,0336	0,0200	0,0249	0,0038
	CS	0,0274	0,0204	0,0239	0,0027

Pela análise da Tabela 9 é possível perceber que os resultados estatísticos dos três algoritmos são bastante semelhantes, porém o Algoritmo Genético, em geral, exibe pior resultado do que os restantes. Para a trajetória em círculo, o algoritmo que apresentou melhor resultado foi o PSO, enquanto, que para as restantes trajetórias foi o CS.

Seguidamente, com o intuito de complementar a análise estatística, foi realizada a análise segundo os índices de desempenho, tendo por base a simulação com a função objetivo mínima de cada algoritmo. Neste estudo, o melhor controlador em termos do erro será aquele que apresenta um menor valor para os índices calculados. Para um melhor estudo destes índices, a Tabela 10 permite visualizar os respectivos valores e comparar entre os diversos algoritmos.

Tabela 10 - Análise dos índices de erro para a simulação com menor função objetivo

Trajetórias	Algoritmo evolutivo	Índices de desempenho			Parâmetros do controlador					
		IAE	ITAE	ISE	kp _v	ki _v	kp _w	ki _w	kp _r	kp _l
Trajetória em círculo	GA	0,0156	0,1588	$8,8445 \times 10^{-6}$	99,97	99,99	69,42	99,93	7,06	95,95
	PSO	0,0157	0,1603	$9,3207 \times 10^{-6}$	99,99	100	67,75	100	7,19	99,99
	CS	0,0156	0,1593	$8,8264 \times 10^{-6}$	100	100	72,78	100	7,15	99,99
Trajetória em forma de oito	GA	0,0406	0,5038	$6,3469 \times 10^{-5}$	82,86	88,76	7,88	39,91	13,41	29,09
	PSO	0,0876	1,1283	$2,9168 \times 10^{-4}$	45,49	100	0	45,42	19,05	67,02
	CS	0,0600	0,7195	$1,1298 \times 10^{-4}$	0	98,85	7,87	34,25	22,25	100
Trajetória em espiral	GA	0,1212	1,4685	$5,0927 \times 10^{-4}$	65,51	93,40	51,33	67,68	71,12	25,19
	PSO	0,1134	1,2970	$4,2816 \times 10^{-4}$	48,28	96,62	38,02	63,18	100	29,58
	CS	0,1475	1,8555	$7,7162 \times 10^{-4}$	92,50	100	50,75	63,18	56,02	25,86

A partir da análise da Tabela 10, conclui-se que o algoritmo que demonstrou ter melhor desempenho para o planeamento das trajetórias em círculo e em forma de oito foi o AG. Especificamente, na trajetória em círculo, os índices de desempenho não apresentam grande discrepância entre os algoritmos, porém o AG exibiu um menor valor para o IAE e para o ITAE, querendo isto dizer que o resultado do controlador originou um menor erro entre a posição real do robô e a posição de referência ao longo do tempo de simulação. Para a trajetória em forma de oito, o AG destacou valores menores para todos os índices de desempenho calculados, ou seja, este algoritmo demonstrou maior capacidade de atingir uma solução ótima.

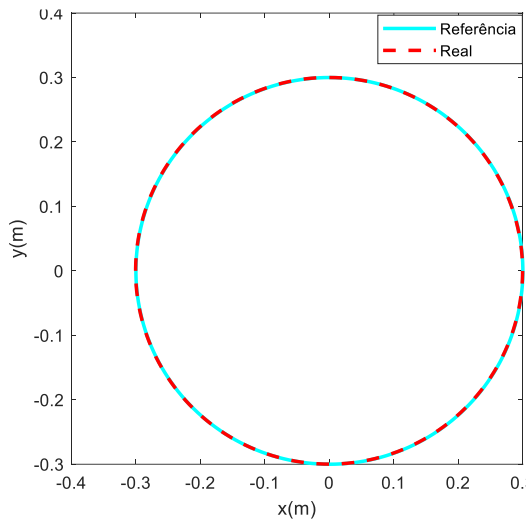
O AG evidenciou piores resultados estatísticos, porque os resultados da função objetivo na amostra de simulações são mais dispersos e tanto no PSO, como no CS estes resultados tendem sempre para um menor intervalo de valores. No entanto, o AG apresentou um

melhor resultado na análise dos índices de desempenho, uma vez que a simulação correspondente à função objetivo mínima gerou uma trajetória do robô com menor erro em relação à trajetória de referência.

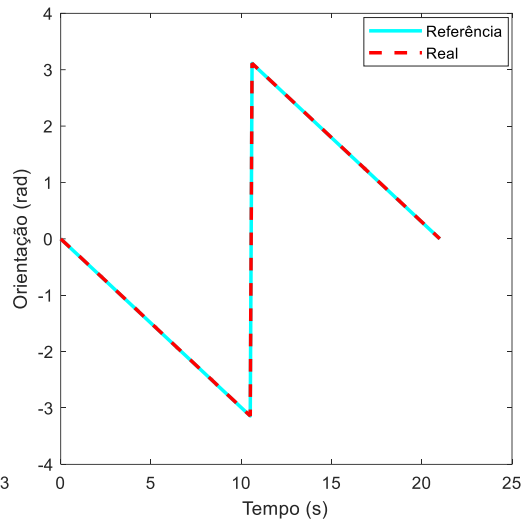
Adicionalmente, no estudo da trajetória em espiral o algoritmo que evidenciou melhor desempenho no controlo da navegação do robô móvel foi o PSO. As análises dos índices de desempenho deste algoritmo apresentaram melhores resultados em comparação com os restantes, podendo assim concluir-se que a solução é mais otimizada e, consequentemente, o erro entre os valores de referência e os valores reais no decorrer da simulação é menor.

Como forma de visualizar estas respostas do sistema, a Figura 31, a Figura 32 e a Figura 33 mostram a qualidade da solução do AG para os trajetos em círculo e em forma de oito e a qualidade da solução do PSO para a trajetória em espiral, respetivamente. Ainda para análise dos resultados, nestas figuras são apresentadas também as velocidades lineares e angulares quer do robô, quer das rodas na entrada e na saída do modelo do robô diferencial.

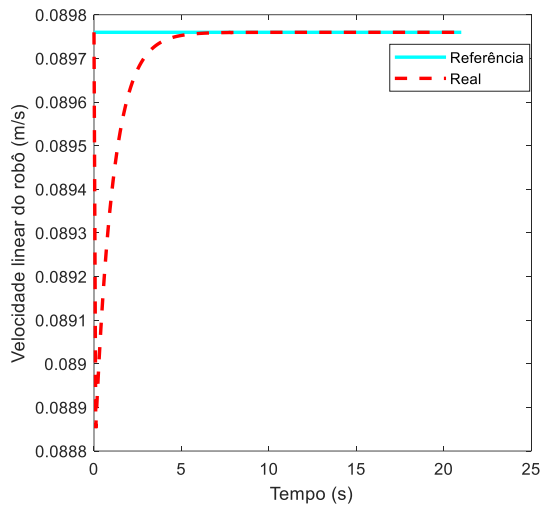
A Figura 31 mostra a resposta de saída do sistema para a trajetória em círculo. Através da Figura 31(a), pode-se concluir que o controlador do modelo tem um bom desempenho, uma vez que, não se consegue distinguir visualmente a trajetória de referência e a trajetória de robô, estando estas sobrepostas. Adicionalmente, a Figura 31(b) comprova que o robô se mantém sempre direcionado conforme a orientação desejada, logo este executa o trajeto posicionado de forma correta. Da Figura 31(c) à Figura 31(f), verificou-se que a velocidade linear e as velocidades angulares apresentam um pico correspondente ao erro inicial, no entanto, o sistema corrigiu-o e todas as velocidades seguiram o sinal de controlo de entrada constante. Particularmente, a velocidade angular da roda esquerda manteve-se maior do que a direita com o intuito de realizar uma curva contínua no sentido horário.



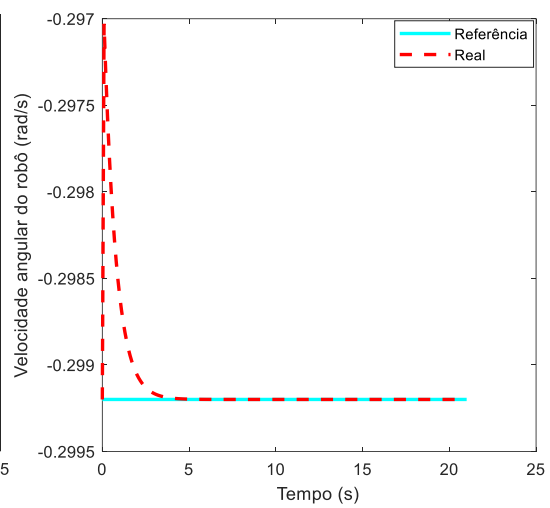
(a) Controlo da trajetória



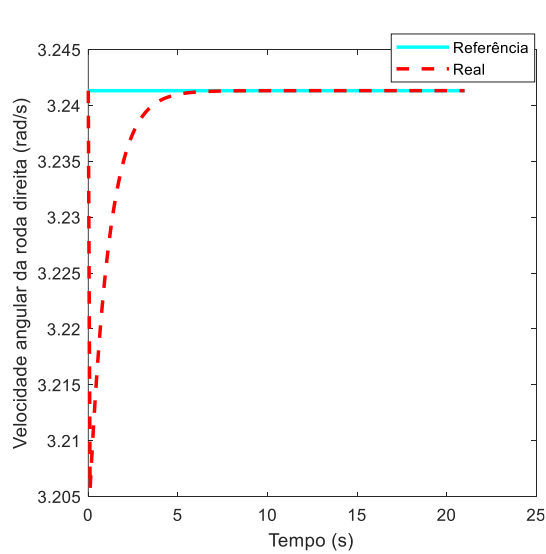
(b) Orientação do robô



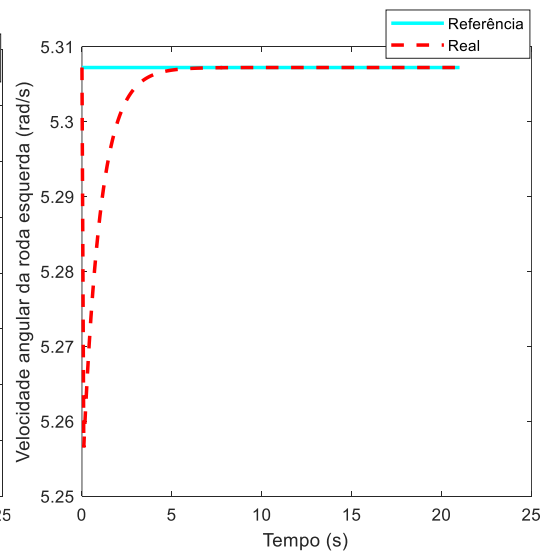
(c) Velocidade linear do robô



(d) Velocidade angular do robô



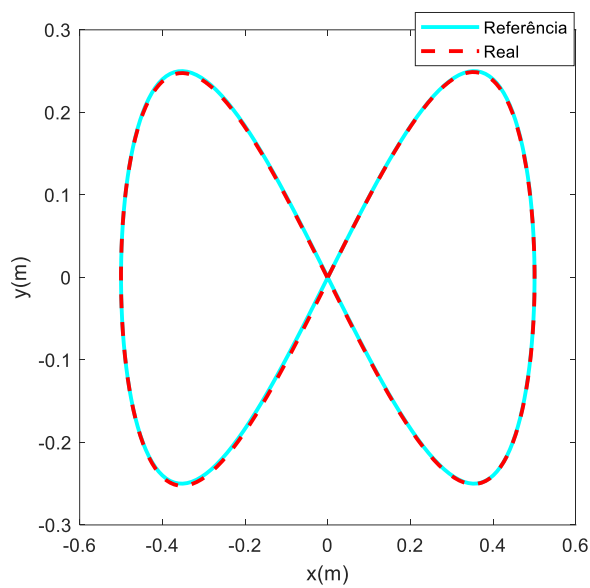
(e) Velocidade angular da roda direita



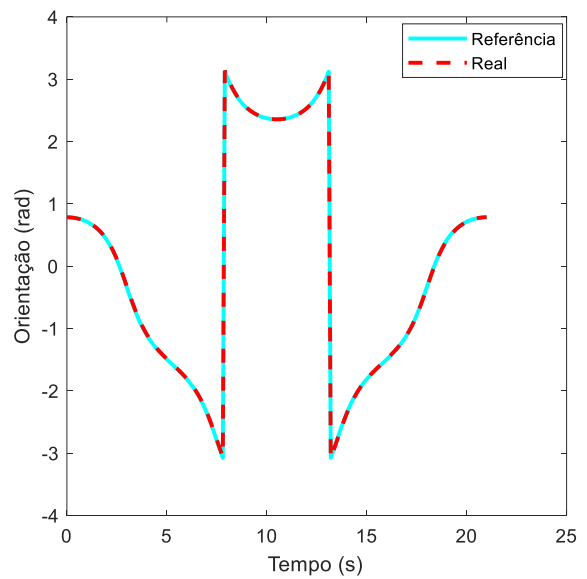
(f) Velocidade angular da roda esquerda

Figura 31 - Resultado da simulação da trajetória em círculo

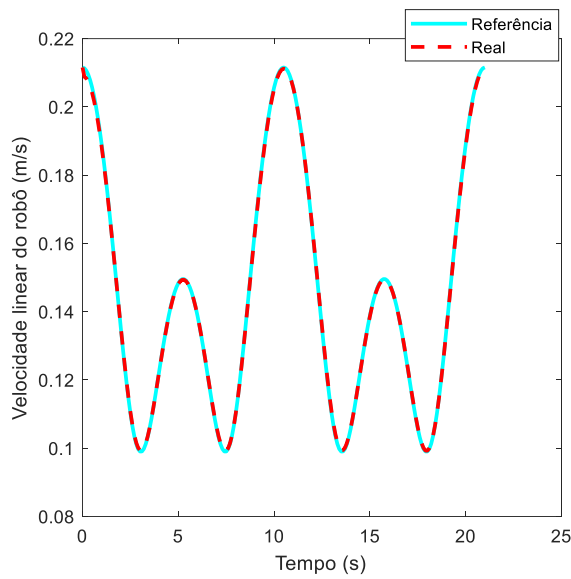
A trajetória em forma de oito descrita pelo sistema é apresentada na Figura 32. A partir da Figura 32(a), é possível perceber que a trajetória de saída do modelo tem alguns desvios em relação à trajetória de referência, porém a capacidade do robô móvel em percorrer a trajetória desejada é satisfatória e o controlador tem um bom desempenho. De acordo com a Figura 32(b), as variações da orientação do robô durante a trajetória coincidem com o esperado e, deste modo, conclui-se que o robô realiza a trajetória orientado na direção correta. Na Figura 32(c)-(f) pode-se observar que as mudanças de direção do robô originam um erro nas velocidades em relação aos valores de referência, mas o sistema evolui de forma a minimizar esse erro. Conforme ilustrado, na Figura 32(e) e na Figura 32(f), quando o robô móvel tem de virar para a esquerda, a velocidade angular da roda direita é maior do que a esquerda e vice-versa. No caso de o robô se deslocar em linha reta, as velocidades das rodas são iguais.



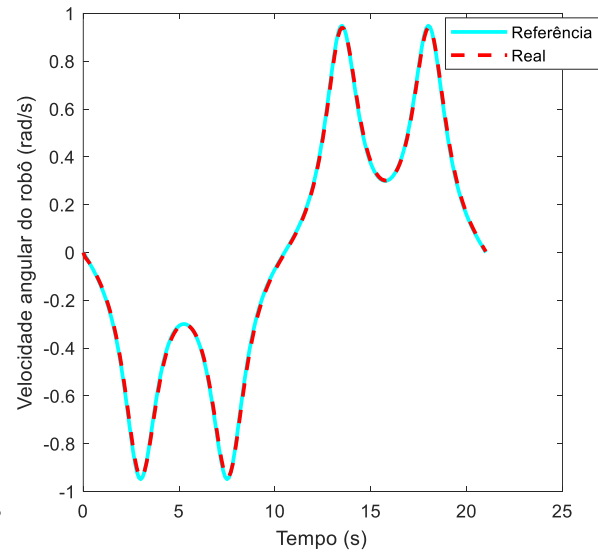
(a) Controle da trajetória



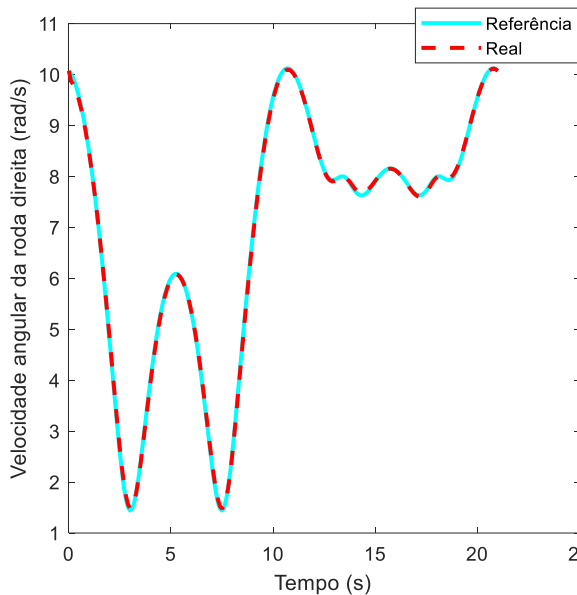
(b) Orientação do robô



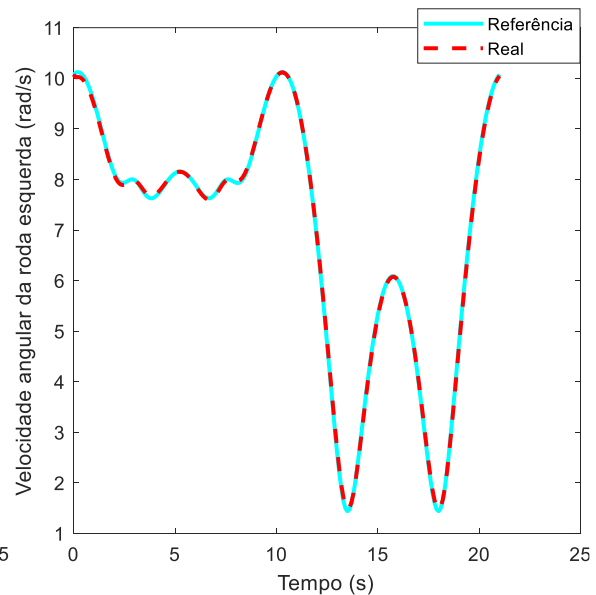
(c) Velocidade linear do robô



(d) Velocidade angular do robô



(e) Velocidade angular da roda direita

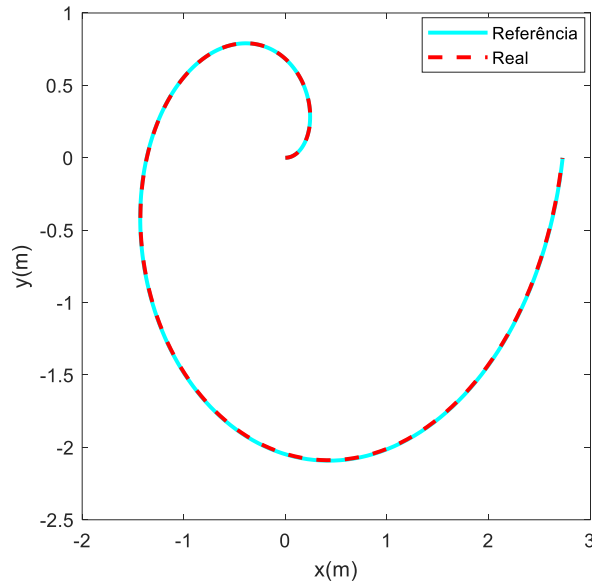


(f) Velocidade angular da roda esquerda

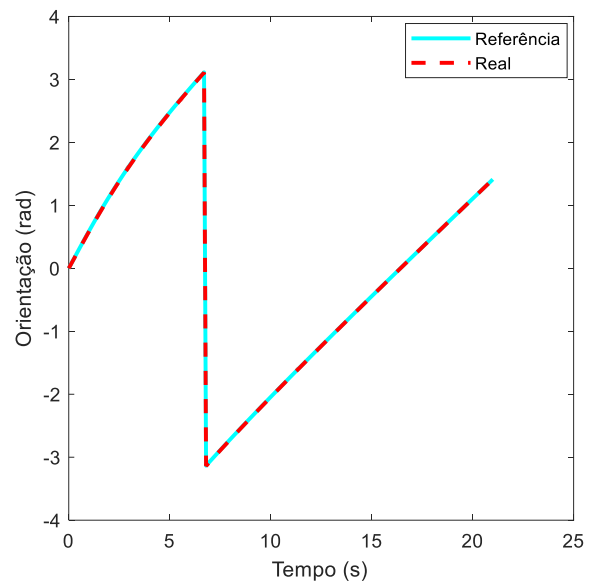
Figura 32 - Resultado da simulação para a trajetória em oito

Na Figura 33 podemos visualizar o controle para a trajetória em espiral. Pela análise da Figura 33(a) e da Figura 33(b), verificou-se que, ao longo do percurso, o rastreamento da trajetória não sofre muitas alterações em relação aos valores de entrada e que o robô se desloca orientado no sentido desejado. A velocidade linear do robô, Figura 33(c), é crescente durante o tempo de simulação, porque o robô móvel desloca-se mais rápido à medida que o raio de rotação do trajeto aumenta. Em contrapartida, a velocidade angular do robô, Figura 33(d), é decrescente, dado que o robô roda mais lentamente à medida que o

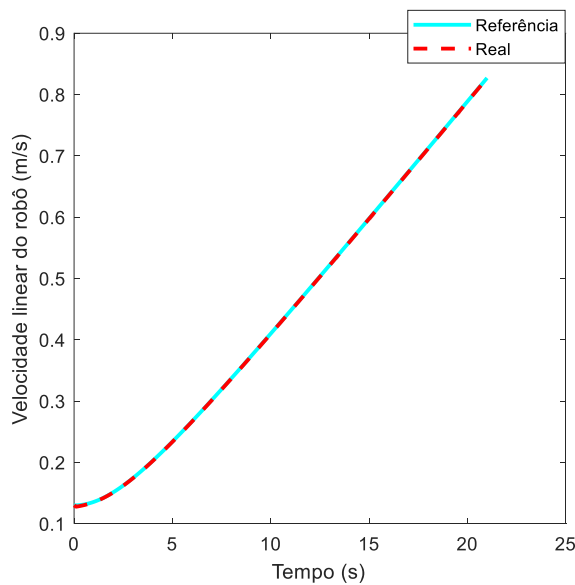
raio de rotação aumenta e esta velocidade apresenta um maior erro no instante inicial devido ao raio de rotação no início do percurso ser pequeno. Através da Figura 33(e) e da Figura 33(f), pode-se constatar que as velocidades angulares das rodas aumentam igualmente e que, para este caso de estudo, o controlador evidencia um ótimo resultado.



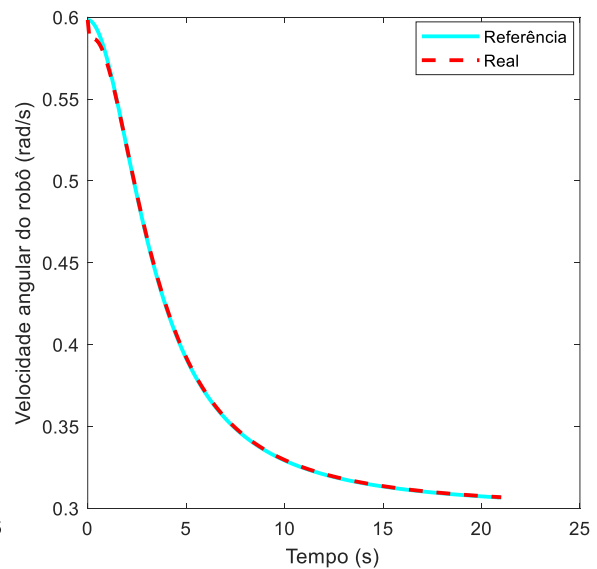
(a) Controle da trajetória



(b) Orientação do robô



(c) Velocidade linear do robô



(d) Velocidade angular do robô

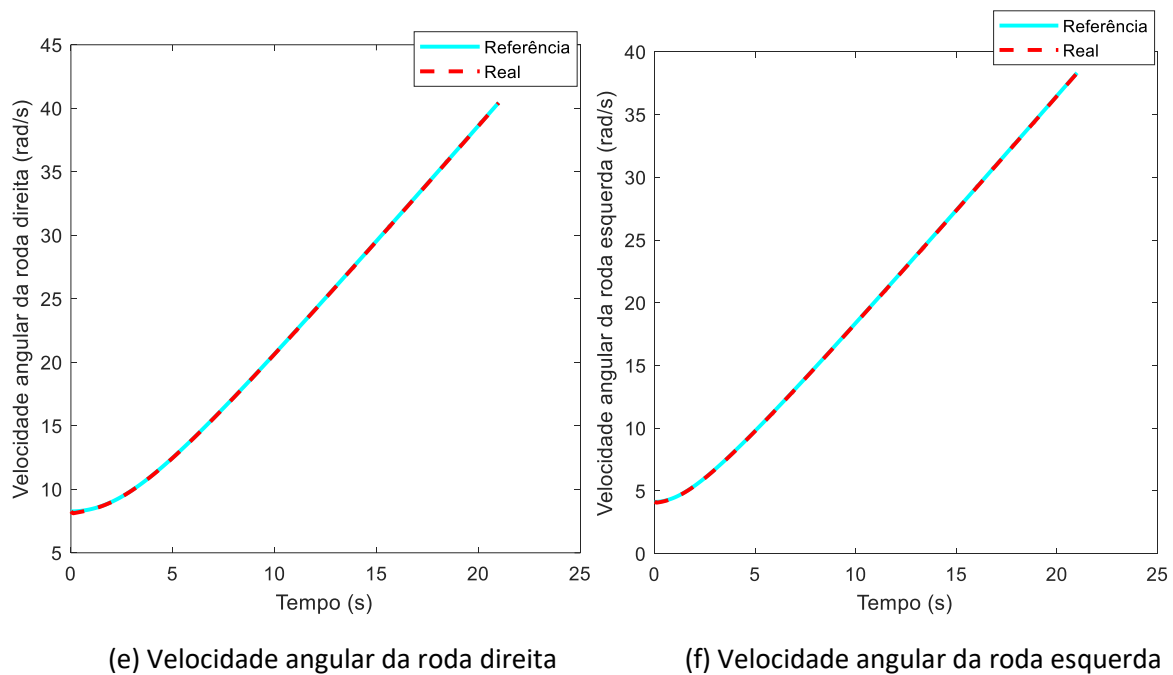


Figura 33 - Resultado da simulação para a trajetória em espiral

Para finalizar a análise da capacidade do robô móvel diferencial em realizar as trajetórias, segundo a solução do algoritmo evolutivo especificado anteriormente com o melhor desempenho para cada trajetória, visualizou-se o comportamento do binário dos atuadores do robô ao longo da trajetória. Dessa forma, a Figura 34, a Figura 35 e a Figura 36 permitem verificar que o robô tem capacidade para percorrer a trajetória em círculo, em forma de oito e em espiral, respectivamente, sem atingir a saturação dos seus atuadores.

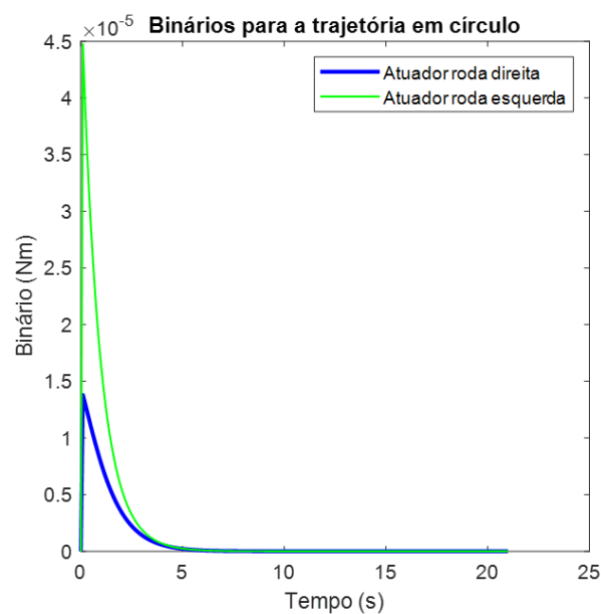


Figura 34 - Binário dos atuadores para a trajetória em círculo

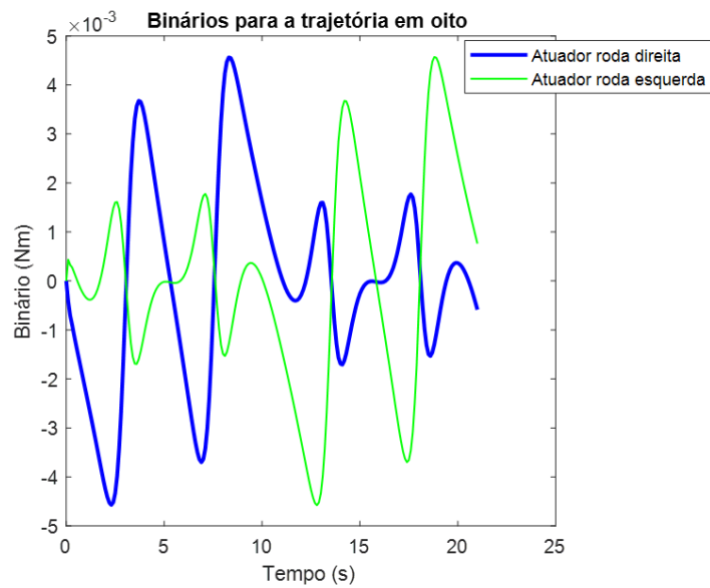


Figura 35 - Binário dos atuadores para a trajetória em forma de oito

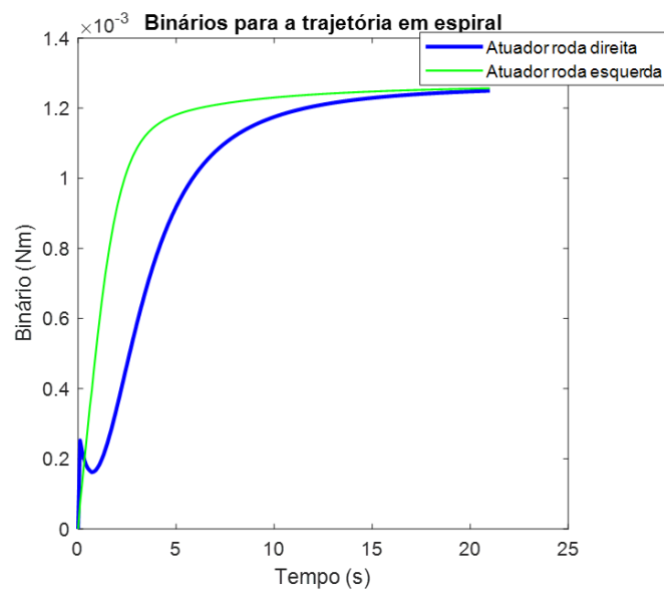


Figura 36 - Binário dos atuadores para a trajetória em espiral

5.3. DISCUSSÃO DE RESULTADOS

Depois da demonstração numérica dos resultados obtidos relativamente aos índices de desempenho, a Figura 37, a Figura 38 e a Figura 39 mostram o comportamento do IAE, do ITAE e do ISE, respetivamente, durante a simulação com a função objetivo mínima de cada algoritmo para a trajetória em forma de oito. Dessa forma, através destas figuras é possível

realizar uma comparação entre os três algoritmos evolutivos implementados e perceber o comportamento destes ao longo do tempo de simulação. Particularmente, foi escolhida a trajetória em forma de oito para a análise dos algoritmos, uma vez que é um trajeto desafiador e com bastantes mudanças de direção, no entanto, o comportamento dos diferentes algoritmos espera-se semelhante para qualquer uma das trajetórias.

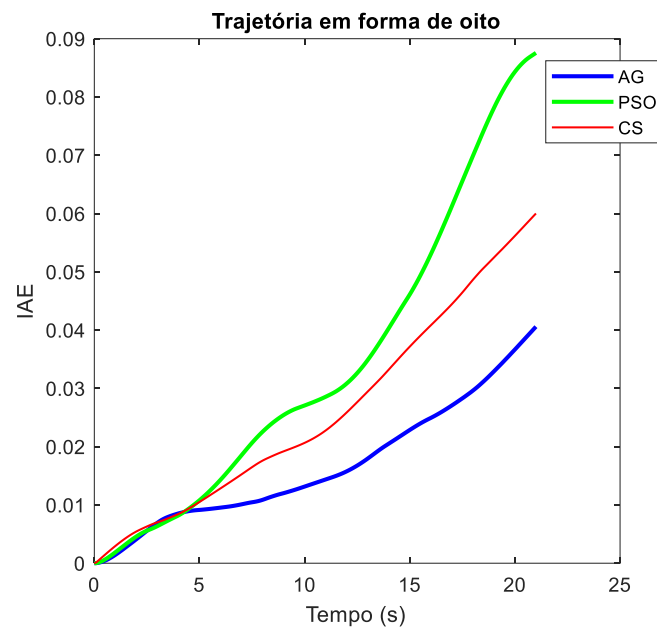


Figura 37 - Evolução do IAE durante a simulação

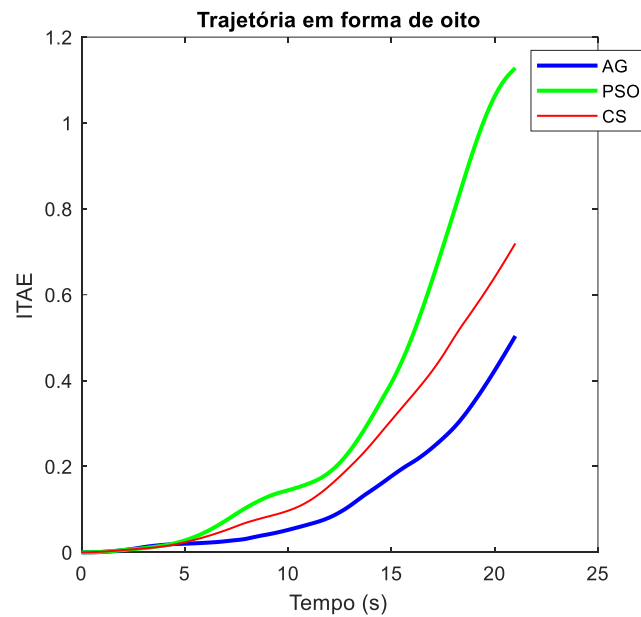


Figura 38 - Evolução do ITAE durante a simulação

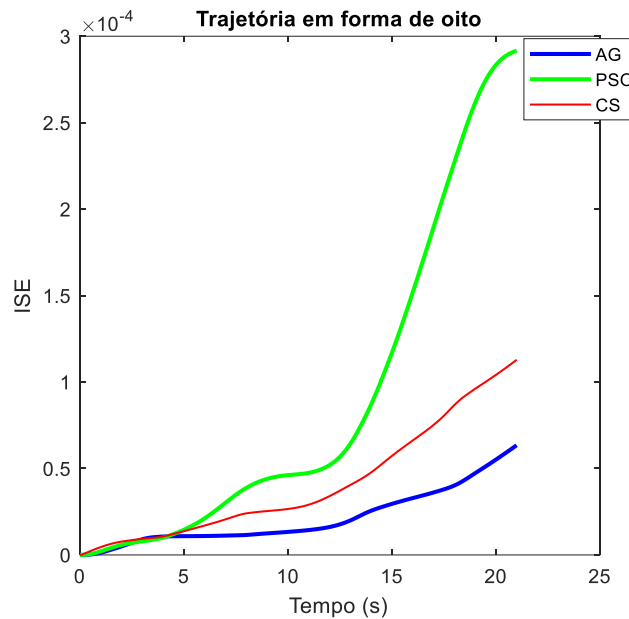


Figura 39 - Evolução do ISE durante a simulação

Para o índice IAE, seria de esperar um comportamento de forma linear ao longo da simulação e, por essa razão, pode-se verificar pela Figura 37 que além de o AG apresentar menor valor do erro, o algoritmo CS é que exibe um aumento do erro mais constante. Seguidamente, para o AG observa-se uma oscilação acentuada nos primeiros instantes de simulação o que representa uma diminuição da função objetivo e a partir desse ponto o erro do sistema aumenta de forma mais uniforme. No que toca ao PSO, visualiza-se um comportamento do erro mais oscilado, o que caracteriza várias melhorias da função objetivo no decorrer da simulação.

Para o índice ISE e ITAE, destaca-se um comportamento do erro idêntico ao IAE, contudo, o ISE evidencia um menor aumento do erro, visto que este índice menospreza os erros de menor grandeza do sistema. Em contrapartida, a inclusão do tempo como um fator de medida para o ITAE gera um maior acréscimo dos erros à medida que t avança.

Em suma, pelos resultados obtidos, pode-se constatar que a ordenação dos algoritmos implementados segundo a rapidez de estabilização da solução, ou seja, do algoritmo evolutivo que estabiliza primeiro para o que estabiliza por último é: CS - AG - PSO.

De forma a validar a estabilização dos algoritmos, da Figura 40 até à Figura 42 é ilustrado o progresso da função objetivo em relação às iterações durante uma simulação referente à trajetória em forma de oito.

O progresso da função objetivo ao longo da simulação segundo o algoritmo CS pode ser observado na Figura 40. A figura demonstra que este algoritmo encontrou a melhor solução da função objetivo ao fim de 5 iterações.

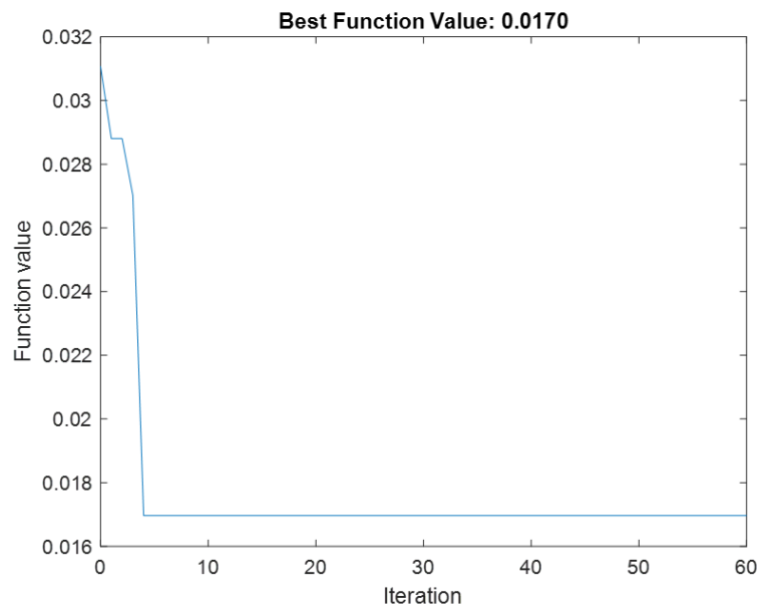


Figura 40 - Evolução da função objetivo através do CS

A Figura 41 descreve a evolução da função objetivo com recurso ao AG. Esta figura foi obtida a partir da função, incluída na *toolbox* utilizada para implementação do algoritmo, que representa os dados calculados pelo AG. Pode-se visualizar que a função objetivo diminuí de forma progressiva, tendo o algoritmo encontrado o valor mínimo ao fim de 20 gerações.

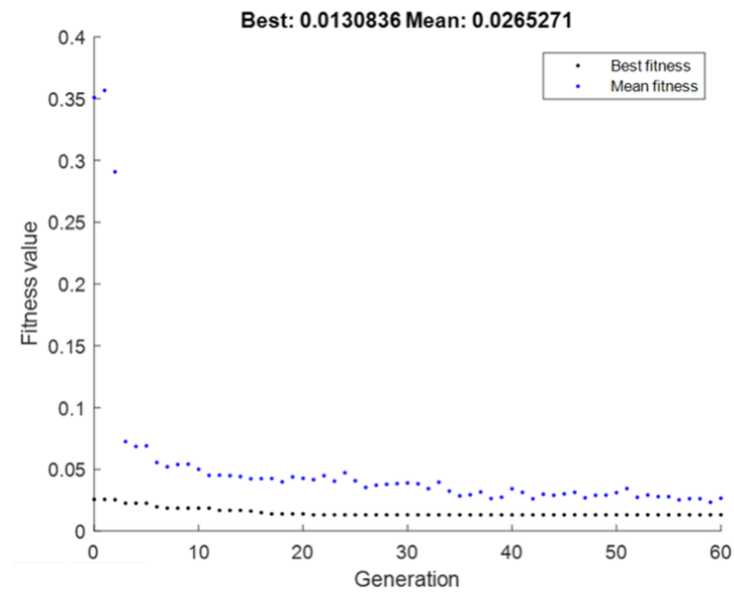


Figura 41 - Evolução da função objetivo através do AG

Por fim, o comportamento do PSO para encontrar o valor ótimo da função objetivo pode ser observado na Figura 42. Tal como para o AG, esta figura também foi extraída segundo a função presente na *toolbox* de otimização, que mostra graficamente os dados calculados pelo algoritmo. Pela análise desta figura, pode-se concluir que o PSO estabilizou a sua solução a partir de 33 iterações. Todavia, através deste algoritmo, a evolução da função objetivo não é gradual, isto é, o valor da função mantém-se constante por algumas iterações ao longo da simulação, seguindo uma função em degraus.

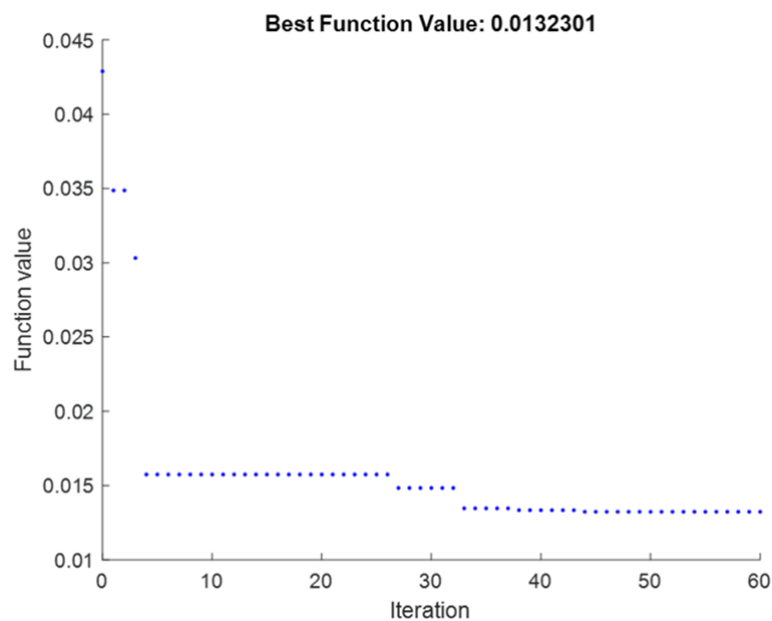


Figura 42 - Evolução da função objetivo através do PSO

6. CONCLUSÕES

Neste capítulo são apresentadas as principais conclusões acerca do trabalho desenvolvido e as competências adquiridas em relação ao tema que foi explorado. Por fim, são sugeridas algumas melhorias para desenvolvimento futuro deste projeto.

6.1. OBJETIVOS REALIZADOS

Desde uma fase inicial, destacaram-se os objetivos a alcançar na realização do presente projeto.

O principal objetivo passou por implementar um conjunto de algoritmos de controle evolutivos que possibilitassem um robô móvel diferencial navegar desde um ponto inicial

até ao ponto final de uma trajetória. Dessa forma, para a concretização deste objetivo, foi necessária a execução de um conjunto de metas, sendo estas:

- Implementação de um modelo dinâmico e cinemático para o robô;
- Realização de um sistema de controlo através do *software* MATLAB/Simulink;
- Implementação de diferentes trajetórias de referência;
- Desenvolvimento do modelo dos atuadores do robô diferencial;
- Implementação dos diferentes algoritmos evolutivos para encontrar uma solução ótima para o sistema;
- Análise do desempenho dos controladores no rastreamento da trajetória por meio do cálculo de vários índices de desempenho.

6.2. CONCLUSÃO

O problema de planeamento de trajetórias foi visto como um problema de otimização e três algoritmos evolutivos, especificamente o AG, o PSO e o CS, foram implementados com o objetivo de encontrar uma solução ótima para o sistema. Estes algoritmos de controlo otimizaram os parâmetros do controlador PID que, numa fase seguinte, são aplicados em uma estrutura de controlo baseada no modelo dinâmico do KHEPERA IV projetada para controlo da realização da trajetória.

Neste projeto, a função objetivo implementada tem por base a minimização do erro segundo o critério da distância Euclidiana. Os resultados das simulações para os vários trajetos evidenciaram que o modelo desenvolvido foi capaz de rastrear as trajetórias de referência com a devida precisão e que o sistema apresenta um ótimo desempenho. Por essa razão, pode-se constatar que os algoritmos evolutivos implementados demonstram ser uma abordagem apropriada para a sintonia de controladores PID.

A utilização de vários algoritmos invés de apenas um foi de grande importância para o estudo, dado que a melhor solução para as trajetórias especificadas não era alcançada através do mesmo algoritmo. Nomeadamente, de acordo com os índices de desempenho

analisados, a melhor solução para a trajetória em círculo e em forma de oito foi atingida pelo AG, enquanto, para a trajetória em espiral foi obtida pelo PSO.

Adicionalmente, os algoritmos implementados no projeto provaram ser bastante convenientes, uma vez que se pretendermos alterar a trajetória de referência basta executar o algoritmo para obter as configurações ótimas dos parâmetros do controlador para a nova trajetória. No entanto, no que toca ao tempo computacional, estes algoritmos não são a melhor alternativa, porque dependendo dos valores atribuídos para o número de iterações e tamanho da população, a simulação pode ser bastante demorada.

O *software* utilizado para o desenvolvimento do projeto, MATLAB, mostrou-se uma ótima ferramenta para implementação de sistemas de controlo devido à disponibilidade de funções de otimização para a sintonia dos controladores e à disponibilidade do ambiente Simulink, onde foi possível desenvolver o sistema de controlo.

6.3. DESENVOLVIMENTOS FUTUROS

Além de o sistema de controlo desenvolvido revelar um bom desempenho, ainda muitos fatores podem ser alterados para a aquisição de um modelo de simulação mais aproximado ao modelo real, tais como forças de atrito que existam no sistema e incrementação de perturbações externas. Estes fatores tornam-se importantes na medida em que os parâmetros externos ao controlo também interferem na resposta do sistema.

A análise do planeamento de trajetórias através de algoritmos evolutivos poderia ser aprofundada com a implementação de uma trajetória com vértices, a fim de perceber o comportamento do sistema para um trajeto com alterações de direções repentinas.

Futuramente, seria também benéfico a utilização de outros índices de desempenho como critério de sintonia dos parâmetros do controlador e, dessa forma, comparar a capacidade de navegação do robô móvel dependendo da função objetivo.

Referências Documentais

- [1] V. Z. A. Gasparetto, "A new method for smooth trajectory planning of robot manipulators," abril 2007. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0094114X06000887>. [Acedido em 5 maio 2020].
- [2] H. D.-W. John Leonard, "Mobile Robot Localization by Tracking Geometric Beacons," junho 1991. [Online]. Available: http://www-cgi.cs.cmu.edu/afs/cs.cmu.edu/Web/People/motionplanning/papers/sbp_papers/l/leonard_ieeeettoa19.pdf. [Acedido em 5 maio 2020].
- [3] A. V. P. Mário Luiz Tronco, "Sistema de Navegação para Robô Móvel Baseada em Visão Omnidirecional e Controlador Fuzzy," [Online]. Available: http://abricom.org.br/wp-content/uploads/2016/08/CBRN2005_093.pdf. [Acedido em 5 maio 2020].
- [4] S. C. B. d. Souza, "Planeamento de trajetória para um robô móvel com duas rodas utilizando um algoritmo A-Estrela modificado," dezembro 2008. [Online]. Available: <http://www.pee.ufrj.br/index.php/pt/producao-academica/dissertacoes-de-mestrado/2008-1/2008121701-2008121701/file>. [Acedido em 6 maio 2020].
- [5] S. G. Tzafestas, "Path Planning," [Online]. Available: <https://www.sciencedirect.com/topics/engineering/path-planning>. [Acedido em 5 maio 2020].
- [6] J. C. S. N. P. F. F. R. Fábio Silveira Vidal, "Sistema Dinâmico de Planejamento de Trajetória para Um Dirigível," 16 agosto 2006. [Online]. Available: <https://www.marinha.mil.br/spolm/sites/www.marinha.mil.br/spolm/files/arq0054.pdf>. [Acedido em 6 maio 2020].

- [7] G. B. L. A. P. D. P. A. J. B.K. Patle, "A review: On path planning strategies for navigation of mobile robot," agosto 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2214914718305130>. [Acedido em 7 maio 2020].

- [8] V. Mallawaarachchi, "Introduction to Genetic Algorithms — Including Example Code," 8 julho 2017. [Online]. Available: <https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3>. [Acedido em 11 maio 2020].

- [9] M. A. C. Pacheco, "Algoritmos Genéticos: Princípios e Aplicações," [Online]. Available: http://www.inf.ufsc.br/~mauro.roisenberg/ine5377/Cursos-ICA/MQ-intro_apost.pdf. [Acedido em 11 maio 2020].

- [10] "Algoritmos genéticos," [Online]. Available: <https://web.fe.up.pt/~ee98221/paginas/algo.htm#alg>. [Acedido em 12 maio 2020].

- [11] X. W. Y. H. Pengfei Guo, "The enhanced genetic algorithms for the optimization design," 18 outubro 2010. [Online]. Available: <https://ieeexplore.ieee.org/document/5639829>. [Acedido em 12 maio 2020].

- [12] T. F. Takanori SHIBATA, "Intelligent Motion Planning by Genetic Algorithm with Fuzzy Critic," [Online]. Available: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=397635>. [Acedido em 12 maio 2020].

- [13] Y. C. Pu Shi, "Dynamic path planning for mobile robot based on genetic algorithm in unknown environment," 28 maio 2010. [Online]. Available: <https://ieeexplore.ieee.org/document/5498349>. [Acedido em 12 maio 2020].

- [14] J. Frankenfield, "Artificial Neural Network (ANN)," 12 maio 2020. [Online]. Available: <https://www.investopedia.com/terms/a/artificial-neural-networks-ann.asp>. [Acedido em 14 maio 2020].

- [15] “Redes Neurais Artificiais,” [Online]. Available: <https://sites.icmc.usp.br/andre/research/neural/>. [Acedido em 13 maio 2020].
- [16] R. F. R. M. Ivan Luis Suptitz, “Análise comparativa de ferramentas de redes neurais artificiais,” 15 outubro 2015. [Online]. Available: http://www.abepro.org.br/biblioteca/TN_STO_213_265_26950.pdf. [Acedido em 14 maio 2020].
- [17] T. W. Rauber, “Redes neurais artificiais,” 23 maio 2014. [Online]. Available: https://www.researchgate.net/profile/Thomas_Rauber/publication/228686464_Redes_neurais_artificiais/links/02e7e521381602f2bd000000/Redes-neurais-artificiais.pdf. [Acedido em 15 maio 2020].
- [18] L. Dormehl, “What is an artificial neural network? Here's everything you need to know,” 6 janeiro 2019. [Online]. Available: <https://www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network/>. [Acedido em 14 maio 2020].
- [19] A. Andrade, “Classificação não supervisionada - Kohonen,” 6 novembro 2012. [Online]. Available: <https://pt.slideshare.net/aandrade1234/classificacao-no-supervisionada-kohonen>. [Acedido em 15 maio 2020].
- [20] D. Janglová, “Neural Networks in Mobile Robot Motion,” 2004. [Online]. Available: https://www.researchgate.net/publication/221702454_Neural_Networks_in_Mobile_Robot_Motion. [Acedido em 15 maio 2020].
- [21] F. K. M. I. Usman Ahmed Say, “Guided Autowave Pulse Coupled Neural Network (GAPCNN) based real time path planning and an obstacle avoidance scheme for mobile robots,” abril 2014. [Online]. Available: <https://www.sciencedirect.com/science/article/abs/pii/S0921889013002327>. [Acedido em 15 maio 2020].
- [22] V. C. L. Bruno Gomes de Almeida, “Particle Swarm Optimization: A Powerful Technique for Solving Engineering Problems,” 9 setembro 2019. [Online]. Available:

<https://www.intechopen.com/books/swarm-intelligence-recent-advances-new-perspectives-and-applications/particle-swarm-optimization-a-powerful-technique-for-solving-engineering-problems>. [Acedido em 15 maio 2020].

- [23] M. A. M. d. O. A. E. Marco Dorigo, "Particle swarm optimization," 13 outubro 2008. [Online]. Available: http://www.scholarpedia.org/article/Particle_swarm_optimization. [Acedido em 15 maio 2020].
- [24] D. T. L. L. Dongshu Wang, "Particle swarm optimization algorithm: an overview," 17 janeiro 2017. [Online]. Available: <https://link.springer.com/article/10.1007/s00500-016-2474-6#Sec1>. [Acedido em 15 maio 2020].
- [25] "Particle Swarm Optimization," 2017. [Online]. Available: <https://www.sciencedirect.com/topics/engineering/particle-swarm-optimization>. [Acedido em 15 maio 2020].
- [26] S. P.-A. C. K. H. Adham Atyabi, "Applying Area Extension PSO in Robotic Swarm," [Online]. Available: https://www.researchgate.net/publication/52001844_Applying_Area_Extension_PSO_in_Robotic_Swarm. [Acedido em 15 maio 2020].
- [27] R. R. N. F. Micael Couceiro, "A PSO multi-robot exploration approach over unreliable MANETs," 18 julho 2013. [Online]. Available: <https://www.tandfonline.com/doi/full/10.1080/01691864.2013.819605>. [Acedido em 16 maio 2020].
- [28] W. C. Guangsheng LI, "Path planning for mobile robot using self-adaptive learning particle swarm optimization," 15 novembro 2017. [Online]. Available: <http://engine.scichina.com/publisher/scp/journal/SCIS/61/5/10.1007/s11432-016-9115-2?slug=fulltext>. [Acedido em 16 maio 2020].

- [29] O. M. O. C. R. S. P. M. M.A. Porta Garcia, "Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation," junho 2009. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1568494609000349>. [Acedido em 18 maio 2020].
- [30] W.-m. L. A.-x. C. Han-ye Zhang, "Path Planning for the Mobile Robot: A Review," 1 outubro 2018. [Online]. Available: https://www.researchgate.net/publication/328033065_Path_Planning_for_the_Mobile_Robot_A_Review. [Acedido em 18 maio 2020].
- [31] V. M. A. C. Marco Dorigo, "Positive feedback as a search strategy," junho 1991. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=CC323B0695D7A2343333FD65DD2A914B?doi=10.1.1.52.6342&rep=rep1&type=pdf>. [Acedido em 18 maio 2020].
- [32] V. M. A. C. Marco Dorigo, "Ant system: optimization by a colony of cooperating agents," [Online]. Available: <https://ieeexplore.ieee.org/document/484436>. [Acedido em 18 maio 2020].
- [33] M. Dorigo, "Ant colony optimization," [Online]. Available: http://www.scholarpedia.org/article/Ant_colony_optimization#Explaining_ACO_through_an_example. [Acedido em 19 maio 2020].
- [34] L. M. J. Y. Shirong Liu, "Path Planning Based on Ant Colony Algorithm and Distributed Local Navigation for Multi-Robot Systems," [Online]. Available: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4026354>. [Acedido em 19 maio 2020].
- [35] K. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," 7 agosto 2002. [Online]. Available: <https://ieeexplore.ieee.org/document/1004010>. [Acedido em 20 maio 2020].

- [36] Y. Z. H. Z. H. C. B. N. Xiaohui Yan, "An Adaptive Bacterial Foraging Optimization Algorithm with Lifecycle and Social Learning," [Online]. Available: <https://www.hindawi.com/journals/ddns/2012/409478/>. [Acedido em 20 maio 2020].
- [37] Y. Z. K. H. Hanning Chen, "Cooperative Bacterial Foraging Optimization," [Online]. Available: <https://www.hindawi.com/journals/ddns/2009/815247/>. [Acedido em 20 maio 2020].
- [38] M. G. S. Devi, "Application of Modified Bacterial Foraging Optimization algorithm for optimal placement and sizing of Distributed Generation," maio 2014. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0957417413008245>. [Acedido em 20 maio 2020].
- [39] Y. L. G. W. Y.-F. Z. Q. X. J.-H. F. X.-T. C. Yu-Peng Chen, "A novel bacterial foraging optimization algorithm for feature selection," [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0957417417302646>. [Acedido em 21 maio 2020].
- [40] Y. Z. H. Z. H. C. B. N. Xiaohui Yan, "An Adaptive Bacterial Foraging Optimization Algorithm with Lifecycle and Social Learning," [Online]. Available: <https://www.hindawi.com/journals/ddns/2012/409478/>. [Acedido em 21 maio 2020].
- [41] C. A. S. Leandro dos Santos Coelho, "Bacteria colony approaches with variable velocity applied to path optimization of mobile robots," [Online]. Available: https://www.researchgate.net/profile/Leandro_Coelho/publication/228698741_BACTERIA_COLONY_APPROACHES_WITH_VARIABLE_VELOCITY_APPLIED_TO_PATH_OPTIMIZATION_OF_MOBILE_ROBOTS/links/551eb1ec0cf2a2d9e13e6e20.pdf. [Acedido em 22 maio 2020].
- [42] L. L.-y. W. J.-g. C. H.-n. Liang Xiao-dan, "Mobile robot path planning based on adaptive bacterial foraging algorithm," 2013. [Online]. Available:

<https://link.springer.com/content/pdf/10.1007/s11771-013-1864-5.pdf>. [Acedido em 22 maio 2020].

- [43] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," outubro 2005. [Online]. Available: <https://pdfs.semanticscholar.org/015d/f4d97ed1f541752842c49d12e429a785460b.pdf>. [Acedido em 25 maio 2020].
- [44] "Artificial bee colony algorithm," [Online]. Available: http://www.scholarpedia.org/article/Artificial_bee_colony_algorithm. [Acedido em 25 maio 2020].
- [45] L. B. A. S. Awatef Aouf, "Same Fuzzy Logic Controller for Two-Wheeled Mobile Robot Navigation in Strange Environments," 3 janeiro 2019. [Online]. Available: <https://www.hindawi.com/journals/jr/2019/2465219/>. [Acedido em 25 maio 2020].
- [46] D. K. Bahriye Akay, "A modified Artificial Bee Colony algorithm for real-parameter optimization," 1 junho 2012. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0020025510003336>. [Acedido em 25 maio 2020].
- [47] G. Y. Xianneng Li, "Artificial bee colony algorithm with memory," Abril 2016. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1568494616000090>. [Acedido em 25 maio 2020].
- [48] B. B. Dervis Karaboga, "On the performance of artificial bee colony (ABC) algorithm," janeiro 2008. [Online]. Available: <https://www.sciencedirect.com/science/article/abs/pii/S1568494607000531>. [Acedido em 25 maio 2020].
- [49] V. A.-R. U. H.-B. Marco Contreras-Cruz, "Mobile robot path planning using artificial bee colony and evolutionary programming," maio 2015. [Online]. Available:

<https://www.sciencedirect.com/science/article/abs/pii/S1568494615000885>.

[Acedido em 25 maio 2020].

- [50] M. J. M. M. H. Saffari, "Bee colony algorithm for real-time optimal path planning of mobile robots," [Online]. Available: <https://ieeexplore.ieee.org/document/5379462>. [Acedido em 25 maio 2020].
- [51] X. L. Qianzhi Ma, "Dynamic Path Planning of Mobile Robots Based on ABC Algorithm," [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-16527-6_34. [Acedido em 25 maio 2020].
- [52] S. D. Xin-She Yang, "Cuckoo Search via Lévy flights," [Online]. Available: <https://ieeexplore.ieee.org/document/5393690>. [Acedido em 27 maio 2020].
- [53] B. T. M. Mareli, "An adaptive Cuckoo search algorithm for optimisation," julho 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2210832717301679>. [Acedido em 27 maio 2020].
- [54] S. D. S. K. M. Xin-She Yang, "Multi-species Cuckoo Search Algorithm for Global Optimization," 30 junho 2018. [Online]. Available: <https://link.springer.com/article/10.1007/s12559-018-9579-4>. [Acedido em 27 maio 2020].
- [55] B. Z. S. Z. Jun Wang, "An Improved Cuckoo Search Optimization Algorithm for the Problem of Chaotic Systems Parameter Estimation," 12 janeiro 2016. [Online]. Available: <https://www.hindawi.com/journals/cin/2016/2959370/>. [Acedido em 27 maio 2020].
- [56] D. P. Prases Mohanty, "Optimal path planning for a mobile robot using cuckoo search algorithm," [Online]. Available: <https://www.tandfonline.com/doi/full/10.1080/0952813X.2014.971442>. [Acedido em 28 maio 2020].

- [57] L. G. H. D. H. W. L. L. M. S. Gaige Wang, "A Hybrid Metaheuristic DE/CS Algorithm for UCAV Three-Dimension Path Planning," 21 outubro 2012. [Online]. Available: <https://www.hindawi.com/journals/tswj/2012/583973/>. [Acedido em 28 maio 2020].
- [58] F. V. C. L.-A. Francisco Rubio, "A review of mobile robots: Concepts, methods, theoretical framework, and applications," 16 abril 2019. [Online]. Available: <https://journals.sagepub.com/doi/full/10.1177/1729881419839596>. [Acedido em 8 junho 2020].
- [59] I. Š. A. Z. S. B. Gregor Klancar, *Wheeled Mobile Robotics: From Fundamentals Towards Autonomous Systems*, Elsevier- Health Sciences Division, 2017.
- [60] "Urbie: Urban Robot," NASA, [Online]. Available: <https://www-robotics.jpl.nasa.gov/tasks/showTask.cfm?FuseAction=ShowTask&TaskID=107&tdaID=999992>. [Acedido em 8 junho 2020].
- [61] "Urbie, the Urban Robot," NASA, [Online]. Available: <https://www.jpl.nasa.gov/news/news.php?feature=485>. [Acedido em 8 junho 2020].
- [62] "TU Berlin Marvin MarkII," [Online]. Available: http://pdv.cs.tu-berlin.de/MARVIN/mark_ii_frameset_introduction.html. [Acedido em 9 junho 2020].
- [63] "sepios," [Online]. Available: <https://sepios.org/>. [Acedido em 9 junho 2020].
- [64] "Life Aquatic: Swiss students roll out cool, fish-like robot," 3 janeiro 2015. [Online]. Available: <https://www.rt.com/news/219591-sepios-nautical-robot-rollout/>. [Acedido em 9 junho 2020].

- [65] J. Dudek, "Differential Drive Robots," [Online]. Available: <http://www.cs.columbia.edu/~allen/F17/NOTES/icckinematics.pdf>. [Acedido em 11 junho 2020].
- [66] A. J. Peter Suster, "Tracking trajectory of the mobile robot Khepera II using approaches of artificial intelligence," Versita, 2011.
- [67] A. A. H. Rached Dhaouadi, "Dynamic Modelling of Differential-Drive Mobile Robots using Lagrange and Newton-Euler Methodologies: A Unified Framework," 2013. [Online]. Available: <https://www.hilarispublisher.com/open-access/dynamic-modelling-of-differentialdrive-mobile-robots-using-lagrange-and-newtoneuler-methodologies-a-unified-framework-2168-9695.1000107.pdf>. [Acedido em 1 agosto 2020].
- [68] P. C. Urbano J. Nunes, "Robôs Móveis: Dinâmica e Controlo de Trajectórias," Novembro 2001. [Online]. Available: https://www.researchgate.net/publication/299420315_Robos_Moveis_Dinamica_e_Controlo_de_Trajectorias. [Acedido em 1 Agosto 2020].
- [69] A. J. M. Amin Abbasi, "Trajectory tracking of two-wheeled mobile robots, using LQR Optimal Control Method, based on computational model of Khepera IV," Journal of Simulation & Analysis of Novel Technologies in Mechanical Engineering, 2017.
- [70] A. J. Jakub Cerkala, "Nonholonomic mobile robot with differential chassis mathematical modelling and implementation in simulink with friction in dynamics," Novembro 2015. [Online]. Available: <https://www.researchgate.net/publication/292177824>.
- [71] "Mobile Robot Kinematics," [Online]. Available: <https://www.usna.edu/Users/cs/crabbe/SI475/current/mob-kin/mobkin.pdf>. [Acedido em 11 junho 2020].

- [72] C. V. R. Coutinho, “Robótica Móvel – Sistema de Condução Autónoma,” Instituto Superior de Engenharia de Lisboa, 2014.
- [73] P. L. Maria Isabel Ribeiro, “Kinematics Models of Mobile Robots,” abril 2002. [Online]. Available: <http://users.isr.ist.utl.pt/~mir/cadeiras/robmove/Kinematics.pdf>. [Acedido em 13 junho 2020].
- [74] D. M. d. S. Neves, “Modelo e Protótipos, à Escala, de AGVs para o Transporte de Material Numa Fábrica,” Porto, 2015.
- [75] R. G. Lorenzi, “Estudo Cinemático e Análise Dinâmica de uma Plataforma de Testes,” Rio de Janeiro, 2017.
- [76] M. J. Gregory Dudek, Computational Principles of Mobile Robotics, United States of America: Cambridge University Press, 2010.
- [77] “robot platform,” [Online]. Available: http://www.robotplatform.com/knowledge/Classification_of_Robots/wheel_control_theory.html. [Acedido em 22 junho 2020].
- [78] J. A. P. d. Oliveira, “Desenvolvimento de uma Plataforma Robótica Omnidirecional,” Porto, 2017.
- [79] “Nexus Automation Limited,” [Online]. Available: <http://www.microrobo.com/omni-wheels.html>. [Acedido em 25 junho 2020].
- [80] J. J. Z. R. L. W. I. J. W. Yong Liu, “Omni-directional mobile robot controller based on trajectory linearization,” 31 maio 2008. [Online]. Available: <https://www.sciencedirect.com/science/article/abs/pii/S0921889007001431>. [Acedido em 27 junho 2020].
- [81] “Omni Robot,” [Online]. Available: <https://omni-robots.com/products/304-8mm-12-inches-mecanum-wheel/>. [Acedido em 27 junho 2020].

- [82] “seeed,” [Online]. Available:
https://wiki.seeedstudio.com/4WD_Mecanum_Wheel_Robot_Kit_Series/.
[Acedido em 27 junho 2020].
- [83] S. Böttcher, “Principles of robot locomotion,” [Online]. Available:
<http://www2.cs.siu.edu/~hexmoor/classes/CS404-S09/RobotLocomotion.pdf>.
[Acedido em 30 junho 2020].
- [84] A. A. L. L. S. L. Ata Jahangir Moshayedi, “Path planning and trajectory tracking of a mobile robot using bio-inspired optimization algorithms and PID control,” [Online]. Available: <https://ieeexplore.ieee.org/document/9071596>. [Acedido em 15 Agosto 2020].
- [85] “Faulhaber,” 18 fevereiro 2020. [Online]. Available:
https://www.faulhaber.com/fileadmin/Import/Media/EN_2619_SR_DFF.pdf.
[Acedido em 10 junho 2021].
- [86] “MathWorks,” [Online]. Available:
https://www.mathworks.com/matlabcentral/fileexchange/29809-cuckoo-search-cs-algorithm?s_tid=mwa_osa_a. [Acedido em 21 maio 2021].

Anexo A. Programação MATLAB para o planeamento da trajetória

```
M= 0.9; % massa do robo
J= 0.01; % momento do inercia
L= 0.0725; % metade da distancia entre as rodas
R= 0.021; % raio da roda

La = 0.0022; % indutância do enrolamento da
armadura
Ra = 36.5; % resistência do enrolamento da
armadura
Kt = 0.019; % constante de binário
Kb = 0.019; % constante de força eletromotriz de
realimentacao
N = 112; % gear ratio
sat= 0.093; % saturacao do binario do motor

global temp
Ts = 0.1; % Sampling time
tempmax=21;
temp = 0:Ts:tempmax; % Simulation time
freq = 2*pi/tempmax;

T=3; % 1- Trajetoria em Circulo
      % 2- Trajetoria em Oito
      % 3- Trajetoria em Espiral

if T==1
    amp = 0.3;
    xRef = amp*sin(freq*temp); yRef =
amp*cos(freq*temp);
    dxRef = freq*amp*cos(freq*temp); dyRef = -
amp*freq*sin(freq*temp);
    ddxRef = -freq^2*amp*sin(freq*temp); ddyRef =
-amp*freq^2*cos(freq*temp);
elseif T==2
    amp = 0.5; %amplitude
    xRef = amp*sin(freq*temp); yRef =
amp*(sin(freq*temp).*cos(freq*temp));
```

```

    dxRef = freq*amp*cos(freq*temp); dyRef =
amp*freq*cos(2*freq*temp);
    ddxRef = -freq^2*amp*sin(freq*temp); ddyRef =
-(amp*2)*freq^2*sin(2*freq*temp);
else
    amp = 0.13;
    xRef = amp.*temp.*cos(freq*temp); yRef =
amp.*temp.*sin(freq*temp);
    dxRef = amp*cos(freq*temp) -
(amp*freq).*temp.*sin(freq*temp); dyRef =
amp*sin(freq*temp)+(amp*freq).*temp.*cos(freq*tem
p);
    ddxRef = -2*amp*freq*sin(freq*temp) -
(amp*freq^2).*temp.*cos(freq*temp); ddyRef=
2*amp*freq*cos(freq*temp) -
(amp*freq^2).*temp.*sin(freq*temp);
end

teta=atan2(dyRef, dxRef);
vRef = sqrt(dxRef.^2+dyRef.^2);
wRef = (dxRef.*ddyRef-
dyRef.*ddxRef)/(dxRef.^2+dyRef.^2);

xref = timeseries((xRef)', temp);
save('ref_x.mat','-v7.3', 'xref')
yref = timeseries((yRef)', temp);
save('ref_y.mat','-v7.3', 'yref')
tetaref = timeseries((teta)', temp);
save('ref_teta.mat','-v7.3', 'tetaref')
vref = timeseries((vRef)', temp);
save('ref_v.mat','-v7.3', 'vref')
wref = timeseries((wRef)', temp);
save('ref_w.mat','-v7.3', 'wref')

% condicoes iniciais
cinicial_x = xref.data(1);
cinicial_y = yref.data(1);
cinicial_teta = tetaref.data(1);
cinicial_v = vref.data(1);
cinicial_w = wref.data(1);

global xr yr kpv kiv kpw kiw Kpr Kpl
xr = (xRef)'; % Valores de referencia de posicao
yr = (yRef)';

```

Anexo B. Modelo de otimização do algoritmo CS

```
function [bestnest,fmin]=cuckoo_search_new(n)
if nargin<1,
% Number of nests (or different solutions)
n=25;
end
% Discovery rate of alien eggs/solutions
pa=0.25;

N_IterTotal=10;
%% Simple bounds of the search domain
% Lower bounds
nd=6;
Lb=0*ones(1,nd);
% Upper bounds
Ub=100*ones(1,nd);

% Random initial solutions
for i=1:n,
nest(i,:)=Lb+(Ub-Lb).*rand(size(Lb));
end

% Get the current best
fitness=10^10*ones(n,1);
[fmin,bestnest,nest,fitness]=get_best_nest(nest,nest,fitness);

N_iter=0;
%% Starting iterations
for iter=1:N_IterTotal,
    % Generate new solutions (but keep the current best)
    new_nest=get_cuckoos(nest,bestnest,Lb,Ub);

    [fnew,best,nest,fitness]=get_best_nest(nest,new_nest,fitness);

    % Update the counter
    N_iter=N_iter+n;
    % Discovery and randomization
    new_nest=empty_nests(nest,Lb,Ub,pa);

    % Evaluate this set of solutions
```

```

[fnew,best,nest,fitness]=get_best_nest(nest,new_nest,fitness);
    % Update the counter again
    N_iter=N_iter+n;
    % Find the best objective so far
    if fnew<fmin,
        fmin=fnew;
        bestnest=best;
    end
    disp(strcat('Total number of iterations=',num2str(iter)));
end %% End of iterations

%% Display all the nests
disp(strcat('Total number of iterations=',num2str(N_iter)));
fmin
bestnest

%% Get cuckoos by random walk
function nest=get_cuckoos(nest,best,Lb,Ub)
% Levy flights
n=size(nest,1);

beta=3/2;
sigma=(gamma(1+beta)*sin(pi*beta/2)/(gamma((1+beta)/2)*beta*2^((beta-1)/2)))^(1/beta);

for j=1:n,
    s=nest(j,:);
    %% Levy flights by Mantegna's algorithm
    u=randn(size(s))*sigma;
    v=randn(size(s));
    step=u./abs(v).^(1/beta);

    stepsize=0.01*step.*(s-best);

    s=s+stepsize.*randn(size(s));
    % Apply simple bounds/limits
    nest(j,:)=simplebounds(s,Lb,Ub);
end

%% Find the current best nest

```

```

function
[fmin,best,nest,fitness]=get_best_nest(nest,newnest,fitness)
% Evaluating all new solutions
for j=1:size(nest,1),
    fnew=fobj(newnest(j,:));
    if fnew<=fitness(j),
        fitness(j)=fnew;
        nest(j,:)=newnest(j,:);
    end
end
% Find the current best
[fmin,K]=min(fitness) ;
best=nest(K,:);

%% Replace some nests by constructing new solutions/nests
function new_nest=empty_nests(nest,Lb,Ub,pa)
n=size(nest,1);
% Discovered or not -- a status vector
K=rand(size(nest))>pa;

stepsize=rand*(nest(randperm(n),:)-nest(randperm(n),:));
new_nest=nest+stepsize.*K;
for j=1:size(new_nest,1)
    s=new_nest(j,:);
    new_nest(j,:)=simplebounds(s,Lb,Ub);
end

% Application of simple constraints
function s=simplebounds(s,Lb,Ub)
    % Apply the lower bound
    ns_tmp=s;
    I=ns_tmp<Lb;
    ns_tmp(I)=Lb(I);

    % Apply the upper bounds
    J=ns_tmp>Ub;
    ns_tmp(J)=Ub(J);
    % Update this new move
    s=ns_tmp;

```

```

%% You can replace the following by your own
functions
function J=fobj(k)

global temp xr yr kpv kiv kpw kiw Kpr Kpl

kpv = k(1);
kiv = k(2);
kpw = k(3);
kiw = k(4);
Kpr = k(5);
Kpl = k(6);

[t,xx,x,y,teta,v,w]=sim('Modelo_sim_CSMotor.slx',
temp);

d = sum((xr(:)-x(:)).^2 + (yr(:)-y(:)).^2);
J = sqrt(d);

```

Anexo C. Código MATLAB para cálculo dos índices de desempenho

```
%% carregar os ficheiros
clear all
for i=1:25
    n = 'simulacao%d';
    nome = sprintf(n,i);
    load (nome) % carregar os ficheiros
    funcobj(i)= fval;
end
%% Calculo dos dados estatisticos
Med = mean(funcobj)
Desvpadrao = std(funcobj)
Min = min(funcobj)
Max = max(funcobj)
%% Executar a simulacao com menor resultado
IndMin = find(funcobj == Min)
nome = sprintf(n,IndMin(1))
load (nome)

save('ref_v.mat','-v7.3', 'vref')
save('ref_w.mat','-v7.3', 'wref')

kpv = xopt(1);
kiv = xopt(2);
kpw = xopt(3);
kiw = xopt(4);
Kpr = xopt(5);
Kpl = xopt(6);

[t,xx,x,y,teta,v,w,wr,wl]=sim('Modelo_sim_Motor.s
lx',temp);

plot(xr,yr,'c','LineWidth',2)
hold on
plot(x,y,'--r','LineWidth',2)
axis square
legend('Referência','Real')

%% Indices de desempenho
ex = xr-x; ey = yr-y;
IAE_x = trapz(t,abs(ex));
```

```

IAE_y = trapz(t,abs(ey));
IAE = IAE_x + IAE_y

ITAE_x = trapz(t,abs(ex).*t);
ITAE_y = trapz(t,abs(ey).*t);
ITAE = ITAE_x + ITAE_y

ISE_x = trapz(t, ex.^2);
ISE_y = trapz(t, ey.^2);
ISE = ISE_x + ISE_y

```

Anexo D. Parâmetros do controlador provenientes das simulações

Tabela 11 - Trajetória em círculo com AG

Simulação	Função Objetivo	kpv	kiv	kpw	kiw	kpr	kpl
1	0,0096	93,0584	99,9742	88,9876	99,9007	7,4091	89,3805
2	0,0095	95,9505	99,9884	75,9090	99,9600	7,1988	71,8903
3	0,0095	99,9809	99,9025	82,7126	99,9455	7,1782	90,7108
4	0,0094	99,9518	99,9545	67,5991	99,9962	7,1525	92,4121
5	0,0096	93,0584	99,9742	88,9876	99,9007	7,4091	89,3805
6	0,0095	95,9505	99,9884	75,9090	99,9600	7,1988	71,8903
7	0,0095	99,9809	99,9025	82,7126	99,9455	7,1782	90,7108
8	0,0094	99,9518	99,9545	67,5991	99,9962	7,1525	92,4121
9	0,0095	98,4357	99,8970	82,2630	99,8229	7,7930	95,6798
10	0,0094	99,9724	99,9931	69,4187	99,9256	7,0518	95,9508
11	0,0094	99,8420	99,8151	75,0147	99,7813	7,2514	96,6200
12	0,0095	99,7021	99,9706	52,2775	99,9783	6,8999	80,1685
13	0,0095	99,9850	99,9998	63,7082	99,9994	7,0481	83,7243
14	0,0095	99,9619	99,3147	66,2524	99,3484	7,4423	99,6826
15	0,0095	90,8580	99,8929	57,1419	99,9190	7,1757	93,6652
16	0,0096	93,0584	99,9742	88,9876	99,9007	7,4091	89,3805
17	0,0095	99,5602	99,9303	88,4553	96,2544	6,9616	92,2814
18	0,0096	84,8431	99,9545	73,3559	99,4467	7,2955	99,8346
19	0,0099	99,0985	99,8192	38,4824	59,8817	4,7250	90,4449
20	0,0109	80,1931	99,9025	87,7442	99,7951	16,1614	97,2861
21	0,0095	99,9497	99,9935	79,6477	99,9533	7,0835	80,1257
22	0,0095	99,9796	99,9851	62,4783	99,9715	7,0333	86,6488
23	0,0098	63,7444	99,7772	38,3115	99,7574	8,0490	99,9345
24	0,0097	94,1212	99,9192	25,2841	99,8529	6,9633	71,6403
25	0,0094	99,9559	99,9674	56,5324	99,9095	7,0915	99,9162

Tabela 12 - Trajetória em círculo com PSO

Simulação	Função Objetivo	kpv	kiv	kpw	kiw	kpr	kpl
1	0,0094	99,9995	100	69,7420	100	7,1946	100
2	0,0094	99,9995	100	69,6854	100	7,1942	99,9998
3	0,0094	100	100	69,7227	100	7,1914	100
4	0,0094	99,9989	100	70,3173	99,9998	7,1852	99,9971
5	0,0094	99,9947	100	69,8435	100	7,1811	99,9992
6	0,0094	99,9998	100	69,4162	100	7,1907	100
7	0,0094	100	100	69,8584	99,9998	7,1988	100
8	0,0094	99,9997	99,9999	69,8075	100	7,1902	99,9995
9	0,0094	99,9993	100	69,6682	100	7,1829	100
10	0,0094	99,9963	99,9999	69,6615	99,9956	7,1940	99,9974
11	0,0094	99,9955	100	70,0271	100	7,1967	99,9996
12	0,0094	100	100	69,6420	100	7,1865	99,9998
13	0,0094	100	100	69,7936	100	7,2062	100
14	0,0094	100	99,9998	69,5163	100	7,1824	99,9994
15	0,0094	99,9997	100	69,9456	100	7,1936	100
16	0,0094	100	100	67,7385	99,9938	7,1304	100
17	0,0094	100	100	69,7592	99,9999	7,1922	100
18	0,0094	99,9997	100	67,7509	100	7,1935	99,9991
19	0,0094	100	100	69,8140	100	7,1965	99,9999
20	0,0094	100	99,9998	69,5163	100	7,1824	99,9994
21	0,0094	99,9997	100	69,9456	100	7,1936	100
22	0,0094	100	100	67,7385	99,9938	7,1304	100
23	0,0094	100	100	69,7592	99,9999	7,1922	100
24	0,0094	99,9997	100	67,7509	100	7,1935	99,9991
25	0,0094	100	100	69,8140	100	7,1965	99,9999

Tabela 13 - Trajetória em círculo com CS

Simulação	Função Objetivo	kpv	kiv	kpw	kiw	kpr	kpl
1	0,0094	100	100	71,9266	100	7,0835	100
2	0,0095	100	100	78,6280	96,0295	7,1711	100
3	0,0095	100	100	57,0603	99,9986	6,7257	92,3239
4	0,0094	100	100	84,2740	100	7,6139	100
5	0,0094	100	100	73,0279	100	7,0697	97,6926
6	0,0094	100	100	69,7842	100	7,3412	100
7	0,0094	100	100	71,1421	100	6,9827	99,8584
8	0,0094	100	100	75,7076	100	7,0668	100
9	0,0094	100	100	63,6815	99,9261	7,3168	100
10	0,0094	100	100	82,4299	100	7,0552	100
11	0,0094	100	100	70,6297	99,5321	7,2779	99,6561
12	0,0095	100	100	100	100	6,9782	99,9118
13	0,0094	99,2724	100	63,8276	100	7,5083	97,5728
14	0,0094	100	100	73,7892	100	7,0404	100
15	0,0094	100	100	69,7006	100	7,2843	96,7652
16	0,0094	100	100	67,6971	100	6,8497	100
17	0,0094	100	100	72,7797	100	7,1497	99,9923
18	0,0094	100	100	69,7006	100	7,2843	96,7652
19	0,0094	100	100	64,0091	100	7,0255	100
20	0,0094	100	100	72,7797	100	7,1497	99,9923
21	0,0094	100	100	77,2168	100	6,9486	96,9736
22	0,0095	100	100	100	100	7,4664	100
23	0,0095	100	99,9142	64,2091	99,9596	7,3232	87,5123
24	0,0095	100	99,9142	64,2091	99,9596	7,3232	87,5123
25	0,0094	100	100	64,0091	100	7,0255	100

Tabela 14 - Trajetória em forma de oito com AG

Simulação	Função Objetivo	kpv	kiv	kpw	kiw	kpr	kpl
1	0,0188	43,8645	67,1416	10,5709	33,2093	74,6313	59,7981
2	0,0182	17,6983	84,6157	25,1835	63,2871	32,3988	53,9763
3	0,0211	59,9914	46,6121	3,0439	42,4288	28,9747	46,3283
4	0,0168	95,8736	97,8722	89,9296	96,9677	50,5468	90,8228
5	0,0195	90,5807	83,5235	2,6188	85,5732	78,0424	84,4048
6	0,0188	43,8645	67,1416	10,5709	33,2093	74,6313	59,7981
7	0,0188	43,8645	67,1416	10,5709	33,2093	74,6313	59,7981
8	0,0182	17,6983	84,6157	25,1835	63,2871	32,3988	53,9763
9	0,0211	59,9914	46,6121	3,0439	42,4288	28,9747	46,3283
10	0,0168	95,8736	97,8722	89,9296	96,9677	50,5468	90,8228
11	0,0195	90,5807	83,5235	2,6188	85,5732	78,0424	84,4048
12	0,0166	3,3710	77,7161	40,7858	46,6396	62,0480	71,8245
13	0,0139	98,9160	81,7138	6,6556	32,1874	14,7162	47,3196
14	0,0232	37,9445	67,9300	74,1532	94,8463	11,4873	93,6601
15	0,0131	74,8483	79,4658	5,1028	33,8027	17,8982	95,2924
16	0,0159	27,1950	82,5960	20,0651	43,2276	63,9511	28,7454
17	0,0202	4,3207	97,4689	24,0436	99,9902	18,5248	19,0748
18	0,0126	82,8586	88,7637	7,8828	39,9105	13,4087	29,0862
19	0,0177	18,1819	63,8943	18,0824	44,7410	33,9939	73,4897
20	0,0202	0,6390	93,5878	90,1703	96,1489	34,4609	60,8321
21	0,0199	26,8645	94,4843	41,8374	70,7167	40,4809	52,6981
22	0,0225	66,2893	57,0607	15,6580	81,2396	70,1367	99,1365
23	0,0219	61,0534	51,6227	61,1318	83,9777	92,8996	97,8405
24	0,0203	88,4077	60,5959	12,5536	95,6345	56,9477	45,0883
25	0,0142	17,0836	89,7518	20,0102	41,0673	11,9669	59,4351

Tabela 15 - Trajetória em forma de oito com PSO

Simulação	Função Objetivo	kpv	kiv	kpw	kiw	kpr	kpl
1	0,0191	33,0146	100	1,7528	100	96,3391	59,6422
2	0,0141	78,0714	94,0676	6,5584	37,5311	53,2314	99,6732
3	0,0190	68,6840	86,0293	17,7543	99,6865	75,2485	96,6162
4	0,0194	99,2596	89,1853	85,1446	94,0992	59,1353	31,9023
5	0,0191	33,0146	100	1,7528	100	96,3391	59,6422
6	0,0197	26,5964	92,7020	49,5916	79,8158	45,6643	84,9446
7	0,0181	86,5774	84,0001	10,0792	96,3999	99,4994	88,8910
8	0,0132	45,4892	100	0	45,4153	19,0528	67,0193
9	0,0189	56,4345	99,3153	62,4009	99,8150	97,3385	97,1660
10	0,0191	33,0146	100	1,7528	100	96,3391	59,6422
11	0,0197	26,5964	92,7020	49,5916	79,8158	45,6643	84,9446
12	0,0181	86,5774	84,0001	10,0792	96,3999	99,4994	88,8910
13	0,0186	0,1696	100	13,9788	100	99,0097	99,9538
14	0,0143	55,2109	89,8045	18,2333	42,9139	83,5313	98,7645
15	0,0190	2,0132	100	16,1409	100	100	98,7161
16	0,0179	91,1680	96,9652	30,3533	100	74,3144	100
17	0,0196	100	61,3432	21,8239	69,0237	100	66,1053
18	0,0195	4,1569	94,3191	15,9045	100	99,0436	100
19	0,0191	33,0146	100	1,7528	100	96,3391	59,6422
20	0,0184	98,0652	69,2791	35,4370	99,9880	77,0702	99,8096
21	0,0184	83,3341	84,7518	100	98,2976	45,6229	87,7737
22	0,0142	64,0044	79,1522	11,0769	41,8862	29,6419	80,7788
23	0,0134	18,4988	98,1218	33,7127	51,5575	53,6482	88,6646
24	0,0184	4,4590	99,7003	17,8622	99,2920	98,4785	100
25	0,0145	100	82,1328	11,2908	38,1906	66,7721	99,5683

Tabela 16 - Trajetória em forma de oito com CS

Simulação	Função Objetivo	kpv	kiv	kpw	kiw	kpr	kpl
1	0,0167	18,0931	99,9994	0,3454	38,6481	36,4059	97,2756
2	0,0190	19,6828	96,3059	0	46,8760	9,5173	43,5020
3	0,0153	40,3835	100	25,1445	41,7573	15,7559	46,2670
4	0,0145	0,0001	98,8472	7,8723	34,2536	22,2525	100
5	0,0162	27,0720	78,2000	20,3738	59,9801	72,1077	100
6	0,0173	82,8299	83,8243	21,3717	47,6719	79,3512	79,8626
7	0,0160	55,2542	83,5098	23,1964	49,0939	74,5018	43,3992
8	0,0193	92,9415	54,8154	10,6945	41,0123	62,6222	92,1806
9	0,0170	25,9548	79,6325	27,9619	54,5618	100	69,5617
10	0,0152	25,3949	100	10,0524	46,0243	7,1143	12,5010
11	0,0188	96,7954	85,9786	21,3678	52,8491	65,0848	31,5038
12	0,0178	0	98,0626	15,4384	46,6650	32,4584	20,9510
13	0,0162	27,0720	78,2000	20,3738	59,9801	72,1077	100
14	0,0173	82,8299	83,8243	21,3717	47,6719	79,3512	79,8626
15	0,0178	0	96,6451	30,2626	50,2281	38,0982	52,9691
16	0,0160	55,2542	83,5098	23,1964	49,0939	74,5018	43,3992
17	0,0193	92,9415	54,8154	10,6945	41,0123	62,6222	92,1806
18	0,0170	25,9548	79,6325	27,9619	54,5618	100	69,5617
19	0,0160	100	100	0	44,6125	12,4084	34,0969
20	0,0153	36,7410	75,2695	19,6977	47,0134	30,0944	45,3486
21	0,0148	36,9532	85,8674	18,9662	45,6975	70,4690	34,5894
22	0,0146	13,1984	99,9350	15,9457	38,7249	21,1787	55,3274
23	0,0166	78,1699	100	25,4119	45,9889	72,2570	73,7063
24	0,0146	0	99,6272	26,4387	56,3335	10,1285	86,4526
25	0,0162	27,0720	78,2000	20,3738	59,9801	72,1077	100

Tabela 17 - Trajetória em espiral com AG

Simulação	Função Objetivo	kpv	kiv	kpw	kiw	kpr	kpl
1	0,0261	89,9014	94,8313	55,6228	90,9633	76,9739	23,3954
2	0,0261	89,9014	94,8313	55,6228	90,9633	76,9739	23,3954
3	0,0200	70,4646	78,5132	37,2400	62,9597	83,8977	26,7383
4	0,0290	42,6875	79,8710	25,3294	91,2710	78,4138	27,9642
5	0,0252	75,8271	75,7136	95,8078	88,4813	31,6946	17,5983
6	0,0205	88,6133	89,0161	24,1680	61,1288	78,9578	25,3832
7	0,0236	95,6196	84,0330	35,1265	60,4439	84,4001	23,4702
8	0,0224	40,1219	99,7616	92,2685	83,4275	91,3187	31,6071
9	0,0250	36,9350	75,5891	83,5259	78,9211	87,6118	32,4501
10	0,0220	92,1583	80,7435	64,0385	82,3247	63,9670	24,4675
11	0,0219	96,9287	87,7886	75,0553	78,6382	90,8500	25,2975
12	0,0257	86,7424	97,8328	67,5380	85,6003	98,5730	33,2481
13	0,0277	94,8020	96,9193	25,1015	90,0435	49,6780	25,0878
14	0,0225	97,7150	60,1691	75,5772	70,2974	87,7400	26,9279
15	0,0236	95,6196	84,0330	35,1265	60,4439	84,4001	23,4702
16	0,0295	79,7010	65,0811	77,9726	84,1602	96,8877	23,1043
17	0,0279	64,9581	96,7863	37,0788	90,3330	73,4384	27,7097
18	0,0218	68,0311	85,3400	45,1659	72,0820	76,5379	24,1630
19	0,0245	46,2346	82,5961	42,7446	81,2129	94,0298	28,7455
20	0,0191	65,5097	93,4014	51,3250	67,6777	71,1217	25,1872
21	0,0247	74,0119	64,7537	61,4430	72,1597	76,4724	24,1544
22	0,0274	74,1210	73,3605	27,6557	90,2468	67,8334	23,2152
23	0,0191	65,5097	93,4014	51,3250	67,6777	71,1217	25,1872
24	0,0247	74,0119	64,7537	61,4430	72,1597	76,4724	24,1544
25	0,0274	74,1210	73,3605	27,6557	90,2468	67,8334	23,2152

Tabela 18 - Trajetória em espiral com PSO

Simulação	Função Objetivo	kpv	kiv	kpw	kiw	kpr	kpl
1	0,0336	7,0773	98,6573	0,7321	100	94,9991	27,6966
2	0,0239	41,1381	90,9096	1,9145	61,1624	57,5539	22,2228
3	0,0263	96,2425	100	100	99,3474	97,6309	28,5149
4	0,0204	84,9985	97,7626	35,2562	57,3796	92,0130	26,4823
5	0,0200	48,2790	96,6166	38,0174	63,1791	100	29,5806
6	0,0284	59,7341	95,8973	50,1428	100	80,3322	25,6333
7	0,0204	87,0878	99,9375	51,3111	71,3595	96,6453	26,9199
8	0,0267	58,4629	94,4203	82,0565	99,9767	100	26,7755
9	0,0215	34,8951	87,5665	74,1757	74,2319	90,5252	29,0725
10	0,0234	97,5292	88,4470	16,8509	70,3109	58,4970	22,0006
11	0,0330	26,0099	97,6486	3,1429	100	84,6760	25,6924
12	0,0298	5,6415	99,9987	31,3083	99,3442	83,7614	26,9816
13	0,0206	96,9222	85,2262	40,7764	71,4039	39,6143	18,7388
14	0,0232	93,7193	85,1940	5,3747	63,2263	89,9957	26,9155
15	0,0255	66,6758	98,5555	59,1327	90,1912	81,7861	25,8550
16	0,0242	100	98,5362	98,8972	84,7594	100	26,0802
17	0,0262	85,8227	99,4362	67,7541	100	56,5505	21,7201
18	0,0246	32,2374	99,9729	94,5089	85,6542	97,4000	26,9315
19	0,0254	92,5616	100	63,9169	89,4241	89,6780	27,8743
20	0,0216	98,5581	83,4094	59,4642	77,2248	60,5508	22,2495
21	0,0261	80,0224	96,7706	73,2140	98,1103	89,7865	34,5183
22	0,0204	88,7536	90,4171	38,1914	68,6060	84,2914	27,7892
23	0,0280	6,7244	100	66,4510	99,1653	73,7114	28,0956
24	0,0277	99,9713	94,6434	49,2902	100	97,3398	27,9560
25	0,0210	48,0789	87,3765	42,2916	68,6175	67,9477	26,1874

Tabela 19 - Trajetória em espiral com CS

Simulação	Função Objetivo	kpv	kiv	kpw	kiw	kpr	kpl
1	0,0258	92,4323	100	58,5668	88,9361	59,6818	22,7115
2	0,0264	10,3548	72,0384	22,6966	59,6391	55,4555	21,7562
3	0,0210	65,6363	100	56,7153	61,8611	88,0700	25,2829
4	0,0264	10,3548	72,0384	22,6966	59,6391	55,4555	21,7562
5	0,0210	65,6363	100	56,7153	61,8611	88,0700	25,2829
6	0,0255	3,2509	99,6417	54,9316	85,6732	54,7424	24,6233
7	0,0228	100	95,4833	78,5285	64,2377	100	26,7779
8	0,0274	35,8188	99,9899	10,2727	59,0966	53,4098	19,9277
9	0,0204	92,4945	100	50,7457	63,1819	56,0171	25,8561
10	0,0264	10,3548	72,0384	22,6966	59,6391	55,4555	21,7562
11	0,0210	65,6363	100	56,7153	61,8611	88,0700	25,2829
12	0,0255	3,2509	99,6417	54,9316	85,6732	54,7424	24,6233
13	0,0228	100	95,4833	78,5285	64,2377	100	26,7779
14	0,0274	35,8188	99,9899	10,2727	59,0966	53,4098	19,9277
15	0,0204	92,4945	100	50,7457	63,1819	56,0171	25,8561
16	0,0264	10,3548	72,0384	22,6966	59,6391	55,4555	21,7562
17	0,0210	65,6363	100	56,7153	61,8611	88,0700	25,2829
18	0,0255	3,2509	99,6417	54,9316	85,6732	54,7424	24,6233
19	0,0228	100	95,4833	78,5285	64,2377	100	26,7779
20	0,0274	35,8188	99,9899	10,2727	59,0966	53,4098	19,9277
21	0,0204	92,4945	100	50,7457	63,1819	56,0171	25,8561
22	0,0208	20,6770	100	28,4435	60,3620	77,7220	27,7546
23	0,0264	10,3548	72,0384	22,6966	59,6391	55,4555	21,7562
24	0,0210	65,6363	100	56,7153	61,8611	88,0700	25,2829
25	0,0255	3,2509	99,6417	54,9316	85,6732	54,7424	24,6233