



INSTITUTO TECNOLÓGICO VALE



**Programa de Pós-Graduação em Instrumentação, Controle e
Automação de Processos de Mineração (PROFICAM)
Escola de Minas, Universidade Federal de Ouro Preto (UFOP)
Associação Instituto Tecnológico Vale (ITV)**

Dissertação

**ALGORITMOS EVOLUTIVOS PARA SOLUÇÃO DA CINEMÁTICA INVERSA DE
ROBÔS COM ATÉ SETE GRAUS DE LIBERDADE: UM ESTUDO DA
APLICABILIDADE EM PROCESSOS DE MINERAÇÃO**

Matheus Alves e Farnese

**Ouro Preto
Minas Gerais, Brasil
2020**

Matheus Alves e Farnese

**ALGORITMOS EVOLUTIVOS PARA SOLUÇÃO DA CINEMÁTICA INVERSA DE
ROBÔS COM ATÉ SETE GRAUS DE LIBERDADE: UM ESTUDO DA
APLICABILIDADE EM PROCESSOS DE MINERAÇÃO**

Dissertação apresentada ao Programa de Pós-Graduação em Instrumentação, Controle e Automação de Processos de Mineração da Universidade Federal de Ouro Preto e do Instituto Tecnológico Vale, como parte dos requisitos para obtenção do título de Mestre em Engenharia de Controle e Automação.

Orientador: Prof. Gustavo Pessin, D.Sc.

Coorientador: Prof. Gustavo Medeiros Freitas,
D.Sc.

Ouro Preto
2020

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

F235a Farnese, Matheus Alves e.

Algoritmos evolutivos para solução da cinemática inversa de robôs com até sete graus de liberdade [manuscrito]: um estudo da aplicabilidade em processos de mineração. / Matheus Alves e Farnese. - 2020.

117 f.: il.: color., gráf., tab.. + Códigos de computador.

Orientador: Dr. Gustavo Pessin.

Coorientador: Prof. Dr. Gustavo Medeiros Freitas.

Dissertação (Mestrado Profissional). Universidade Federal de Ouro Preto. Programa de Mestrado Profissional em Instrumentação, Controle e Automação de Processos de Mineração. Programa de Pós-Graduação em Instrumentação, Controle e Automação de Processos de Mineração.

Área de Concentração: Engenharia de Controle e Automação de Processos Mineraiis.

1. Robótica. 2. Algoritmos computacionais - Algoritmos evolutivos. 3. Minas e recursos mineraiis. I. Freitas, Gustavo Medeiros. II. Pessin, Gustavo. III. Universidade Federal de Ouro Preto. IV. Título.

CDU 681.5:622.2

Bibliotecário(a) Responsável: Sione Galvão Rodrigues - CRB6 / 2526



FOLHA DE APROVAÇÃO

Matheus Alves e Farnese

Algoritmos Evolutivos para Solução da Cinemática Inversa de Robôs com até Sete Graus de Liberdade: Um Estudo da Aplicabilidade em Processos de Mineração

Membros da Banca

Gustavo Pessin – Doutor - Instituto Tecnológico Vale Mineração

Gustavo Medeiros Freitas – Doutor - Universidade Federal de Minas Gerais

José Alberto Naves Cocota Júnior – Doutor - Universidade Federal de Ouro Preto

Fernando Santos Osório – Doutor - Universidade de São Paulo

Versão final

Aprovado em 08/09/2020

De acordo,

Agnaldo José da Rocha Reis (p/ Gustavo Pessin).



Documento assinado eletronicamente por **Agnaldo Jose da Rocha Reis, COORDENADOR(A) DO CURSO DE PÓS-GRADUAÇÃO EM INSTRUMENTAÇÃO, CONTROLE E AUTOMAÇÃO DE PROC DE MINERAÇÃO**, em 11/12/2020, às 12:43, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0114178** e o código CRC **59D9CE73**.

”A engenharia é a arte de dirigir as grandes fontes de poder na natureza para o uso e conveniência do homem!”

Thomas Tredgold

”O que sabemos é uma gota, o que ignoramos é um oceano!”

Isaac Newton

*Aos engenheiros Joaquim Alves
Borges e Judegar Farnese, que
sabiam toda a técnica e foram
graduados Mestres na escola da
vida! Aos meus pais!*

Agradecimentos

O universo não é aleatório, existe uma energia suprema que o determina, e podemos conhecê-la como Deus. Sinto, e sou muito grato, por essa energia estar sempre trabalhando em meu favor. Obrigado Deus!

Ainda que essa energia me guie, a jornada é de provações, o caminho não é fácil, requer evolução. De outra forma não haveria razão para voltarmos a este plano. E é extremamente gratificante e reconfortante saber que em todo caminho, nos bons e maus momentos, eu tive uma base sólida me sustentando. Minha família sempre acreditou em mim e me apoiou, eles tornaram tudo possível. Em especial, minha mãe Neiva, meu pai Rogério e minha irmã Anna Clara, minha gratidão eterna, mais uma vitória nossa.

Meus agradecimentos se estendem à Laiane Andrade, que foi quem esteve mais próxima durante essa etapa. Foi ela quem me viu de perto sonhar, lutar, ceder, recomeçar, acreditar, fraquejar, reerguer e conquistar, sempre com uma palavra amiga e um gesto carinhoso que me preenchia uma vez mais. Obrigado por ter sido tão importante. A gratidão por compartilhar e acreditar nesse sonho comigo nunca acabará.

Deixo um profundo e sincero agradecimento aos meus orientadores, Gustavo Pessin e Gustavo Freitas, grandes engenheiros e pesquisadores, que mediante sua incrível capacidade escolheram o caminho mais nobre que um sábio pode tomar, o de transmitir o seu conhecimento. Me sinto privilegiado por pessoas tão capazes terem dedicado parte do seu conhecimento e tempo em prol do meu sucesso.

Agradeço os meus amigos, que são poucos, mas me entendem, por isso são os melhores. Em especial ao Mestre Guilherme Brito, um irmão de longa data, que a vida me deu o privilégio de ter ao meu lado mais uma vez nesse desafio, muito obrigado meu amigo pelo apoio incircunstancial, minha conquista é muito sua também. Minha imensa gratidão também ao Alex Ferreira e ao Higor Milânes que acreditam nas mesmas coisas que eu acredito, que sonham assim como eu sonho, e que trabalham incansavelmente de maneira incrível. O apoio de vocês foi essencial nessa conquista.

Fico mais uma vez grato à Escola de Minas, e ao espírito vivo de Gorceix, que me tornou engenheiro e mestre. Mais do que nunca creio que “Cum Mente et Malleo” faremos a diferença para nossas pessoas e nosso país. É uma honra poder dizer mais uma vez que eu pertenço aos nobres que fazem parte dessa história.

Uma empresa são pessoas, cada uma delas. E a produção de riquezas é a razão dessas

peessoas trabalharem com um objetivo em comum. Aquelas pessoas que acreditam em um ideal e batalham para que a Vale seja o que ela deve ser são dignas de reverências. Assim como nosso solo rico, que cria tanta oportunidades, deve ser sempre protegido e louvado. Portanto, obrigado Vale por conseguir reunir pessoas que acreditam em grandes coisas e por transformar nossa riqueza em oportunidades, o que me permitiu chegar aqui.

Em especial agradeço a Instituto Tecnológico Vale (ITV), por cumprir sua nobre missão de criar opções de futuro por meio de pesquisa científica e desenvolvimento de tecnologias. A integração entre a indústria e o conhecimento científico é um caminho promissor ao desenvolvimento de nossa nação, e poder fazer isso criando oportunidades é um trabalho admirável. Sou extremamente grato por ter participado dessa iniciativa.

Muito obrigado a toda a turma da automação de Minas de Carajás, que quando precisei me concentrar nos livros, permaneceram competentes, dedicados e se esforçaram por mim, para que eu pudesse buscar meus sonhos. Vocês são uma grande equipe e merecem todo o sucesso que conquistam.

Renovo meus votos de gratidão à república Senzala, que claro, voltou a ser meu teto quando precisei. Mas muito além disso, obrigado a todos os escravos que me acolheram de volta e sobretudo obrigado por continuarem mantendo o espírito que mantém essas paredes de pé para que possamos voltar pra casa quando queremos ou precisamos.

Por fim, são muitas as palavras, mas elas nunca serão suficientes para agradecer o quanto eu deveria a todos aqueles que contribuíram nessa etapa. Tudo isso foi só pra dizer que cada um de vocês será sempre lembrado por mim, vocês foram fundamentais em minha vida.

Resumo

Resumo da Dissertação apresentada ao Programa de Pós Graduação em Instrumentação, Controle e Automação de Processos de Mineração como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ALGORITMOS EVOLUTIVOS PARA SOLUÇÃO DA CINEMÁTICA INVERSA DE ROBÔS COM ATÉ SETE GRAUS DE LIBERDADE: UM ESTUDO DA APLICABILIDADE EM PROCESSOS DE MINERAÇÃO

Matheus Alves e Farnese

Setembro/2020

Orientadores: Gustavo Pessin
Gustavo Medeiros Freitas

O cálculo da cinemática inversa de um manipulador robótico consiste em determinar a posição das juntas do robô de forma que seu efetuador alcance uma posição e orientação definida. Conforme a arquitetura do robô e a pose desejada, o problema pode possuir uma, múltiplas, inúmeras ou até mesmo nenhuma solução. Essa dissertação apresenta o desenvolvimento e aplicação de técnicas de algoritmos genéticos para determinação da cinemática inversa de um manipulador robótico de até sete graus de liberdade. Buscando melhor desempenho para a aplicação específica, são investigadas diferentes configurações de algoritmos genéticos, considerando os efeitos da população inicial, número de gerações, métodos de seleção, crossover e mutação. São apresentados também os resultados da ponderação da função objetivo do algoritmo genético para melhorar a eficiência energética da movimentação do robô. Para levar o manipulador até a pose determinada pelo AG é proposto e implementado um controle de trajetória através das velocidades das juntas, calculadas através um polinômio de quinto grau. Todas as soluções propostas são simuladas em ambientes virtuais utilizando os manipuladores IRB120 e IRB6700. Para validar a solução com 6DoF, os algoritmos desenvolvidos no MatLab são integrados ao RobotStudio. Já as simulações para validação das soluções de 7DoF e o controle de velocidade são realizadas integrando o MatLab ao CoppeliaSim. Por fim, estão apresentados nessa dissertação testes dos resultados alcançados aplicados em um robô real.

Palavras-chave: Robótica, Algoritmos Evolutivos, Mineração

Macrotema: Mina; **Linha de Pesquisa:** Robótica Aplicada à Mineração; **Tema:** Eficiência Energética.

Abstract

Abstract of Dissertation presented to the Graduate Program on Instrumentation, Control and Automation of Mining Process as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

EVOLUTIONARY ALGORITHMS FOR SOLUTION OF INVERSE KINEMATICS OF ROBOTS WITH UP TO SEVEN DEGREES OF FREEDOM: A STUDY OF APPLICABILITY IN MINING PROCESSES

Matheus Alves e Farnese

September/2020

Advisors: Gustavo Pessin
Gustavo Medeiros Freitas

The inverse kinematics of a robotic manipulator consists on determining the robot's joints positions so that its end-effector reaches a defined position and orientation. Depending on the robot's architecture and the desired pose, the problem may have one, multiple, countless or even no solution. This article presents the development and application of genetic algorithm techniques to determine the inverse kinematics of a robotic manipulator with six degrees of freedom. Focusing on the specific application, we investigate different genetic algorithms configurations, considering the effects of the initial population, number of generations, selection methods, crossover and mutation. They are exposed too, the results of AG's objective function weighting for improve the power efficiency of the robot. In order to move the robot until the desired pose are propose and implemented a path planning based on joint control velocity, calculated by a fifth order polynomial. All the solutions are simulated in virtual environment using the manipulators IRB120 and IRB6700. To validate the 6DoF solution the genetic algorithms are implemented in MatLab, which is integrated with RobotStudio. And, for validate the 7DoF solution and the control of velocity the algorithms is integrated with teh CoppeliaSim. Finally, the results are validated on real robot.

Keywords: Robotic, Genetic Algorithms, Mineration

Macrotheme: Mine; **Research Line:** Robotics Applied to Mining; **Theme:** Energy Efficiency.

Lista de Figuras

Figura 1.1	Exemplos de robô (braços manipuladores) em áreas operacionais da Vale (a) Robô para lavagem de equipamentos em Carajás. (b) Robô para soldagem de vagões de carga em São Luís. (c) Robô para preparação de amostras para análise de produto em Carajás. (d) Robô para preparação de amostras para análise de produto em Itabira.	24
Figura 2.1	Translação e rotação de um corpo rígido.	35
Figura 2.2	(a) Posição da junta (b) Velocidade da junta (c) Aceleração da junta (SPONG <i>et al.</i> , 2005).	40
Figura 2.3	Manipulador IRB 120 (ABB, 2019b).	41
Figura 2.4	Características físicas e cinemáticas do IRB120 Adaptado (VARHEGY <i>et al.</i> , 2017).	42
Figura 2.5	Espaço de trabalho do IRB6700 (ABB, 2019c).	43
Figura 2.6	Dimensões do IRB6700 (Adaptado (ABB, 2019c)).	43
Figura 3.1	Possíveis soluções para a cinemática inversa de um manipulador com 6DoF (SILIANO <i>et al.</i> , 2010).	59
Figura 3.2	Esquema de integração MatLab / Robot Studio (CORBACHO, 2014).	63
Figura 3.3	Interface gráfica do MatLab (GUI).	65
Figura 3.4	Cena do CoppeliaSim com o IRB120 sob o trilho.	67
Figura 4.1	Resultado do algoritmo genético executado no MatLab.	69
Figura 4.2	Efeitos da população inicial e do número de gerações.	70
Figura 4.3	Efeitos do elitismo e da sobreposição de populações.	71
Figura 4.4	Efeitos da variação da taxa de crossover	72
Figura 4.5	Efeitos do processamento de população em paralelo.	73
Figura 4.6	Efeito dos métodos de seleção, <i>crossover</i> e mutação.	74
Figura 4.7	Efeito dos métodos de seleção, <i>crossover</i> e mutação.	75
Figura 4.8	Efeitos do doping da população inicial.	76
Figura 4.9	Resultado do algoritmo para cinemática inversa de um manipulador com 7DoF.	76
Figura 4.10	Resultado do algoritmo, com população inicial dopada, para cinemática inversa de um manipulador com 7DoF.	77

Figura 4.11	Simulações sem ponderação da função objetivo.	78
Figura 4.12	Simulações com função objetivo ponderada.	79
Figura 4.13	Deslocamento do primeiro eixo com função objetivo não ponderada.	80
Figura 4.14	Deslocamento do primeiro eixo com função objetivo ponderada.	80
Figura 4.15	(a) Posições, (b) velocidades e (c) acelerações das juntas durante a trajetória.	80
Figura 4.16	(a) Robô pra frente com cotovelo para cima (b) Robô para frente com cotovelo para baixo (c) Robô para trás com cotovelo pra baixo.	82
Figura 4.17	Comparação entre solução no MatLab e as variáveis de junta do robô no <i>FlexPendant</i>	82
Figura 4.18	(a) Posição Inicial “Home” (b) Simulação sem ponderação da função objetivo com voltado para trás (c) Simulação com ponderação da função objetivo com voltado para frente.	83
Figura 4.19	Simulação do controle de posição para manipulador com 7DoF.	84
Figura 4.20	Simulação do controle de trajetória para manipulador com 7DoF.	85
Figura 4.21	Simulação do AG aplicado ao IRB6700.	86
Figura 4.22	Simulação do AG aplicado ao IRB6700, com população inicial dopada.	86
Figura 4.23	Simulações com função objetivo não ponderada.	87
Figura 4.24	Simulações com função objetivo ponderada.	88
Figura 4.25	Deslocamento da junta 1 para o AG com a função objetivo não ponderada.	88
Figura 4.26	Deslocamento da junta 1 para o AG com a função objetivo ponderada.	89
Figura 4.27	Cotação da licença PC-Interface com a ABB.	89
Figura 4.28	Simulação no IRB6700.	90
Figura 5.1	Forno tipo grelha móvel (MOURÃO, 2017).	91
Figura 5.2	Carros de grelha (MOURÃO, 2017).	92
Figura 5.3	Impacto pela parada do forno de grelha móvel (MOURÃO, 2017).	92
Figura 5.4	Robô em sobreposição ao esforço ergonômico.	93
Figura 5.5	Lavagem manual e robotizada de equipamentos de grande porte.	95

Lista de Tabelas

Tabela 2.1	AG's aplicados à Robótica	33
Tabela 2.2	Limites de ângulos das juntas do IRB 120 (ABB, 2019b).	42
Tabela 2.3	Torque das juntas do IRB 120 (ABB, 2019b).	42
Tabela 2.4	Limites de ângulos das juntas do IRB6700 (ABB, 2019c).	44
Tabela 2.5	Torque das juntas do IRB6700 (ABB, 2019c).	44
Tabela 3.1	Tabela D.H. do IRB120.	53
Tabela 3.2	Tabela D.H. do IRB120 com 7DoF.	60
Tabela 4.1	Configurações do AG1.	68
Tabela 4.2	Configurações do AG1.	70
Tabela 4.3	Configurações do AG2.	71
Tabela 4.4	Configurações do AG3.	72
Tabela 4.5	Configurações do AG4.	73
Tabela 4.6	Configurações do AG5.	73
Tabela 4.7	Configurações do AG6.	74
Tabela 4.8	Configurações do AG7.	75
Tabela 4.9	Tabela D.H. IRB6700 com sétimo eixo.	85

Lista de Siglas e Abreviaturas

Ψ Ângulo de Orientação

τ Torque

Φ Ângulo de Orientação

Θ Ângulo de Orientação

W Peso de Ponderação

ac Aceleração

AG Algoritmos Genéticos

D.H. *Denavit-Hartenberg*

DoF *Degree of Freedom*

f Função objetivo

F.O. Função Objetivo

fval Valor final da função fitness

GAOT *Genetic Algorithms Optimization Toolbox*

GUI *Guide User Interface*

p Posição

PID *Proporcional-Integral-Derivativo*

PROC *Procedure*

q Junta

R Matriz de Rotação

RNA *Rede Neural Artificial*

ROS *Robot Operating System*

Rot *Rotação*

RPY Ângulos *Roll, Pitch, Yaw*

T Transformada Homogênea

t Tempo

th Melhor indivíduo determinado pelo algoritmo genético

Trans *Translação*

v Velocidade

Lista de Símbolos

Ψ Ângulo de Orientação

τ Torque

Φ Ângulo de Orientação

Θ Ângulo de Orientação

W Peso de Ponderação

ac Aceleração

AG Algoritmos Genéticos

D.H. *Denavit-Hartenberg*

DoF *Degree of Freedom*

f Função objetivo

F.O. Função Objetivo

fval Valor final da função fitness

GAOT *Genetic Algorithms Optimization Toolbox*

GUI *Guide User Interface*

p Posição

PID *Proportional-Integral-Derivativo*

PROC *Procedure*

q Junta

R Matriz de Rotação

RNA *Rede Neural Artificial*

ROS *Robot Operating System*

Rot *Rotação*

RPY Ângulos *Roll, Pitch, Yaw*

T Transformada Homogênea

t Tempo

th Melhor indivíduo determinado pelo algoritmo genético

Trans *Translação*

v Velocidade

Sumário

1	Introdução	23
1.1	Motivação	25
1.2	Objetivo	26
1.3	Organização do Trabalho	27
2	Revisão Bibliográfica	28
2.1	Cinemática Inversa	28
2.2	Trabalhos Relacionados	30
2.3	Robótica	33
2.3.1	Conceitos Fundamentais	33
2.3.2	Espaço Euclidiano	34
2.3.3	Movimento de Corpos Rígidos	34
2.3.4	Transformadas Homogêneas	35
2.3.5	Ângulos de Orientação	36
2.3.6	Cinemática Direta	37
2.3.7	Convenção de Denavit-Hartenberg	37
2.3.8	Planejamento de Trajetórias	38
2.3.9	Manipuladores Robóticos da ABB	40
2.4	Algoritmos Genéticos	44
2.4.1	Representação Genética	45
2.4.2	Cruzamento e Mutação	45
2.4.3	Elitismo	46
2.4.4	Seleção	46
2.4.5	Função Objetivo	46
2.4.6	Crerios de Parada	46
2.4.7	Genetic Algorithm Optimization ToolBox	47
2.5	Métodos de Integração e Simulação	47
3	Metodologia Proposta	50
3.1	Algoritmos Genéticos no MatLab	50
3.1.1	Representação Genética	50

3.1.2	Representação Genética	51
3.1.3	População Inicial	51
3.1.4	Função Objetivo	51
3.1.5	Restrições e Critérios de Parada	54
3.1.6	Configuração do Algoritmo Genético	54
3.2	Planejamento de Trajetória no MatLab	57
3.3	Cinemática Inversa para Robôs com 7DoF	58
3.4	Eficiência Energética	61
3.5	Métodos de Simulação	62
3.5.1	Simulação com o RobotStudio	63
3.5.2	Interface Gráfica de Usuário (GUI)	64
3.5.3	Simulação com o CoppeliaSim	65
4	Resultados	68
4.1	Algoritmos Genéticos para Resolução da Cinemática Inversa de 6DoF	68
4.2	Investigação dos Algoritmos Genéticos	69
4.3	Algoritmos Genéticos para Resolução da Cinemática Inversa de 7DoF	75
4.4	Função Objetivo Ponderada	77
4.5	Planejamento de Trajetória no MatLab	80
4.6	Simulações	81
4.6.1	Simulações no RobotStudio	81
4.6.2	Simulações no Coppelia	83
4.7	Simulações com o IRB6700	85
5	Aplicações da Solução - Estudos de Viabilidade de Casos	91
5.1	Montagem Robotizada de Carro de Grelha	91
5.2	Célula Robotizada de Lavagem de Equipamentos	94
6	Conclusão	98
6.1	Contribuições	100
6.2	Trabalhos Futuros	101
	Referências Bibliográficas	103
	Apêndices	107
A	Algoritmo Genético no MatLab	108
A.1	Cinemática Direta	108
A.2	Função para Geração da População Aleatória	111
A.3	Função Objetivo	112
A.4	Restrições	113

A.5	Algoritmo Genético	113
B	Controle de Trajetória no MatLab	115
C	MatLab Integrado com Simuladores	117
C.1	Integração MatLab - RobotStudio	117
C.2	Integração MatLab - Coppelia	118
D	Código RobotStudio	121
E	Links Vídeos de Simulação	123

1. Introdução

A Revolução Industrial, que emergiu no século XVIII, provocou uma série de reviravoltas na sociedade humana, e ainda que seja dividida em etapas, é uma revolução permanente. Os grandes marcos dessa revolução, além de permitirem uma série de novos experimentos, trouxeram transformações imprevistas na vida cotidiana, na mentalidade das pessoas, no processo produtivo e nas relações de trabalho (HARARI, 2018). A primeira fase, iniciada na Inglaterra se caracterizou pela máquina a vapor, o tear mecânico e as locomotivas. No século seguinte a energia elétrica trouxe um novo patamar ao sistema de produção. Mais recentemente, a criação de computadores e da internet representaram um terceiro marco importante para a sociedade e sobretudo para a indústria.

Essas etapas são marcadas por um interesse em comum, a melhoria de processos para facilitação da mão de obra humana e aumento de produtividade. A revolução permanece viva, e esses interesses ainda norteiam os principais processos industriais ao redor do mundo. Atualmente, está se consolidando um quarto marco dessa revolução que, não por acaso, foi denominado indústria 4.0. Nessa nova fase os princípios fundamentais são preservados, aprimorar processos, facilitar a mão de obra e aumentar a produtividade ainda são conceitos balizadores da evolução da indústria e consequentemente da sociedade. E assim como em outros tempos a evolução vem aliada à inovação tecnológica, como comunicabilidade entre máquinas, inteligência artificial para tomada de decisão e ciências de análise massiva de dados.

A robótica, apesar de não ser um conceito novo, assume protagonismo dentro dessa nova indústria. O estudo de sistemas robotizados surgiu e se desenvolveu ao longo do século XX junto aos computadores e a internet. Tanto é que os primeiros robôs datam da década de 20 e 30, quando apareceram modelos como o Televox e o Elektro. Entretanto, os conceitos trazidos pela nova onda tecnológica ampliaram os horizontes de pesquisa e trabalho com sistemas robotizados.

Um dos problemas clássicos relacionados a robótica, o cálculo da cinemática inversa, ainda é alvo de muito estudo. É comum encontrar pesquisas tratando desse problema, buscando identificar uma solução genérica e robusta para determinar as variáveis das juntas associadas a uma determinada posição e orientação do efetuador (VARGAS, 2013). A determinação da cinemática inversa permite conhecer a pose de um robô para que o efetuador alcance determinado ponto no espaço. Esse problema é essencial para que muitas tarefas possam ser executadas de maneira automatizada, e ganha relevância com o surgimento de novos conceitos de visão computacional. Novas técnicas permitem ao robô uma maior interação com o espaço ao seu redor, ampliando sua gama de aplicações.

Partindo dessas premissas, é fácil vislumbrar no horizonte futuro de curto, médio prazo que sistemas robotizados tendem a ganhar autonomia, sobretudo com a capacidade de tomada de decisão. Isso quer dizer que, além de mecanizar e automatizar tarefas, os robôs passarão a ser capazes de interagir com o ambiente e decidir suas ações a partir dos dados coletados.

Um exemplo recente, que deixa claro esse avanço, é o envio do robô móvel “*Curiosity*” para exploração em Marte (WELCH *et al.*, 2013). Considerando esses fatores, a fusão entre as técnicas de aprendizado de máquina e robotização está cada vez mais em voga. Algoritmos de aprendizagem contribuem com o robô acrescentando um nível de inteligência, permitindo que partes das tarefas executadas possam acontecer a partir de uma tomada de decisão da própria máquina.

A medida que ganham em robustez as soluções robotizadas tendem a ocupar setores industriais cada vez mais diversificados, mesmo aqueles que historicamente investem menos em aplicações tecnológicas de ponta, como o setor de exploração mineral (MESQUITA *et al.*, 2017). O uso de sistemas e ferramental tecnológico, como robôs, podem contribuir com a produtividade, eficiência e segurança operacional desse setor, minimizando os riscos de acidentes pessoais e de impactos socioambientais (VILLELA, 2017).

É nesse contexto de investimentos em tecnologia para ganho de produtividade, e sobretudo para segurança operacional, que a Vale¹ tem fomentado cada vez mais o uso de recursos tecnológicos em suas operações, com destaque para a robotização de processos. O intuito da empresa é padronizar e aumentar a eficiência dos processos, assim como reduzir custos e riscos para seus empregados. Sistemas robóticos são utilizados em manutenções de vagões, laboratórios de análise química e também para lavagem de equipamentos de grande porte (COTA *et al.*, 2017). A Figura 1.1 demonstra alguns dos robôs em operação na Vale.

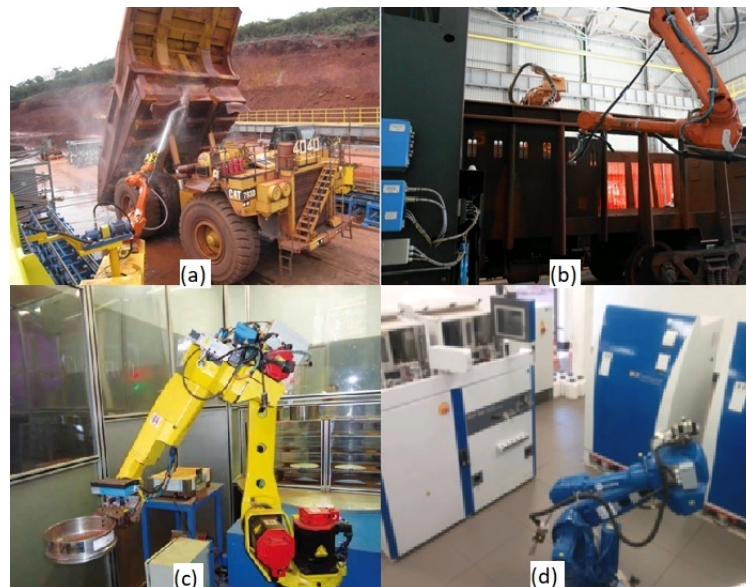


Figura 1.1: Exemplos de robô (braços manipuladores) em áreas operacionais da Vale (a) Robô para lavagem de equipamentos em Carajás. (b) Robô para soldagem de vagões de carga em São Luís. (c) Robô para preparação de amostras para análise de produto em Carajás. (d) Robô para preparação de amostras para análise de produto em Itabira.

Conhecendo o crescente cenário de aplicação de robôs na indústria de beneficiamento

¹<http://www.vale.com/>; <http://www.itv.org/>

mineral, investigar técnicas que possam tornar esses processos mais robustos e inteligentes pode trazer uma contribuição em termos de segurança e eficiência. Esse tipo de trabalho pode melhorar a adaptação dos robôs às especificidades das tarefas que lhes são propostas, considerando a diversidade de ambientes em que é possível encontrar esses equipamentos, desde oficinas de manutenção até usinas de pelotização.

1.1. Motivação

A evolução da robotização de processos vem ultrapassando as barreiras de mecanização e automatização. Com o avanço das técnicas de inteligência artificial é possível conferir maior dinamismo aos processos robotizados, se valendo de recursos como algoritmos genéticos, redes neurais e outros métodos de otimização. Assim sendo, integrar dois campos de conhecimento convergentes em uma única solução é um dos aspectos motivadores desse projeto.

A aplicação de algoritmos de inteligência artificial permite otimizar soluções, como da cinemática inversa. Considerando que um robô de 6 graus de liberdade tem múltiplas soluções viáveis dentro do espaço de trabalho, encontrar uma formatação ótima pode significar eficiência energética e ganho de tempo produtivo. Essa premissa se reforça se forem acrescentados mais graus de liberdade, como por exemplo um trilho, que permite a translação lateral de toda a estrutura do robô.

Alguns algoritmos evolutivos tem ainda a característica de não necessitarem de conhecimento prévio do problema, uma vez que podem partir de uma população inicial aleatória. Esse fator, em tese, permite uma maior adaptabilidade a processos diversos, flexibilizando a aplicação da solução e melhorando sua replicabilidade. Essas características podem ser bastante úteis ao ser aplicadas em robôs, isso porque a técnica se concentra na posição desejada do efetuador no espaço e a disposição angular de cada eixo, independente do processo em que está sendo aplicado. A hipótese é que essa solução permite flexibilizar a disposição de equipamentos e ferramentas dentro do espaço de trabalho do robô. Isso pode trazer ganhos como por exemplo para o processo de lavagem robotizada na oficina de manutenção da Vale em Carajás. Nessa solução existe uma grande dificuldade em adaptar a lavagem quando o modelo de equipamento a ser lavado é alterado. Além disso, um erro mínimo de posicionamento de um equipamento de grande porte pode gerar consequências graves como a colisão do robô.

Em termos operacionais, aplicar algoritmos de aprendizado de máquina para determinar a cinemática inversa de robôs pode trazer um ganho de eficiência considerável. Em um processo produtivo de larga escala esse ganho pode ser diretamente refletido no resultado financeiro. Um exemplo claro está relacionado ao robô utilizado para montagem de carro de grelha móvel, na usina de pelotização em Vitória/ES. A eficiência desse robô está diretamente ligada a produtividade do forno de queima de pelotas, e uma parada delongada desse forno pode prejudicar a produção de pelotas. Nesse mesmo sentido, outros processos robotizados dentro da Vale podem ter interferência significativa na eficiência operacional. O processo robotizado de emenda

de correias pode ser otimizado de forma a reduzir o tempo parado do transportador de longa distância diminuindo o impacto na produtividade. Da mesma forma, a parada do caminhão fora de estrada na célula de lavagem robotizada, afetará mais a produção, quão mais longa for.

Esses impactos operacionais são revertidos diretamente em resultados financeiros. Em comum todos os processos citados afetam diretamente na produção de minério de ferro, e segundo o relatório anual de produção da Vale publicado em seu site, foram produzidos 55,3 milhões de toneladas de pelotas apenas em 2018 (VALE, 2019). O mesmo relatório enfatiza que o minério de ferro é o principal produto da empresa, responsável pela maior parte de seu fluxo de caixa. Considerando isso, estudar e desenvolver aplicações tecnológicas que poderão contribuir com a maior produtividade e competitividade da empresa é a principal motivação desse trabalho.

1.2. Objetivo

O principal objetivo desse trabalho é estudar técnicas de otimização para resolução da cinemática inversa de um robô com até 7 graus de liberdade. A ideia é desenvolver algoritmos genéticos para determinar valores ótimos para as variáveis de junta que levarão o manipulador à pose desejada.

Mais especificamente, os objetivos contemplados por essa dissertação são:

- Proposta, desenvolvimento e investigação de técnicas de otimização para solução do problema da cinemática inversa para robôs de 6 graus de liberdade.
- Proposta, desenvolvimento e investigação de técnicas de otimização para solução do problema da cinemática inversa para robôs de 6 graus de liberdade com a inserção de um eixo externo.
- Desenvolvimento de técnica para melhorar a eficiência energética durante o deslocamento, da pose inicial até a pose final, do manipulador robótico.
- Desenvolvimento de técnica para planejamento de trajetórias.
- Integração do algoritmo com o ambiente virtual RobotStudio da ABB, e com o ambiente de simulação virtual CoppeliaSim, para efeito de simulação.
- Realizar estudo acerca da aplicabilidade do algoritmo genético no processo robotizado de montagem de carro de grelha, e na lavagem robotizada de equipamentos de grande porte, na Vale.

1.3. Organização do Trabalho

Essa dissertação está organizada em mais 5 capítulos, além deste primeiro, que introduz o tema e apresenta as motivações e os objetivos. O capítulo 2, traz uma revisão bibliográfica, abordando de maneira ampla diversas técnicas utilizadas para solução da cinemática inversa, como métodos geométricos, algébricos, iterativos e heurísticos. Na sequência são apresentados mais detalhadamente os trabalhos relacionados a esse, que utilizam algoritmos genéticos para resolver problemas acerca da robotização de processos. Ainda nesse capítulo, é colocado um referencial teórico fundamental ao desenvolvimento da proposta, tratando temas de robótica, algoritmos genéticos e métodos de simulação.

O terceiro capítulo aborda a metodologia desenvolvida para alcançar os objetivos propostos. Em resumo, é exposto o desenvolvimento do algoritmo genético no MatLab, o planejamento de trajetória utilizando polinômios de quinta ordem e a simulação das soluções no RobotStudio e no CoppeliaSim.

Os resultados e as discussões a eles relacionadas são expostos no quarto capítulo; nesse capítulo são apresentadas as investigações realizadas sobre as diversas variações dos algoritmos desenvolvidos e a efetividade destes para a resolução da cinemática inversa. Além disso, são apresentados os resultados relacionados à solução para melhorar a eficiência energética durante a movimentação do manipulador. Na sequência é tratado sobre o planejamento de trajetória baseada no controle de velocidade das juntas a partir de um polinômio de quinta ordem. São apresentados ainda nesse capítulo todas as simulações realizadas nos ambientes virtuais e no robô real.

No quinto capítulo é realizado um estudo sobre a aplicabilidade da solução nos processos robotizados de montagem de carro de grelha e de lavagem de equipamentos de grande porte, ambos operacionais dentro da Vale. Esse capítulo mostra ainda os possíveis ganhos que a aplicação do algoritmo genético pode trazer a esses processos.

Por fim, no último capítulo estão apresentadas as conclusões e contribuições dessa dissertação e levanta as possibilidades de trabalhos futuros que podem ser realizados a partir dos resultados apresentados.

2. Revisão Bibliográfica

Esse capítulo está dividido em cinco seções: a primeira trata de maneira ampla sobre as diversas formas de solucionar o problema da cinemática inversa. A segunda seção discorre sobre os trabalhos relacionados à proposta apresentada nessa dissertação, que é solucionar o problema da cinemática inversa através de algoritmos evolutivos. A terceira e quarta seções apresenta referenciais teóricos sobre robótica e algoritmos genéticos, que servem como embasamento ao conteúdo dessa dissertação. E a quinta seção trata sobre métodos de integração com ambientes virtuais de simulação de robótica.

2.1. Cinemática Inversa

A solução da cinemática inversa é de fundamental importância para transformar as especificações de movimento, atribuídas ao efetuador no espaço operacional, nos correspondentes movimentos espaciais conjuntos, que permitem a execução do movimento desejado (SICILIANO *et al.*, 2010). Existem maneiras diversas de calcular a cinemática inversa para um manipulador robótico. Quanto maior o número de graus de liberdade mais complexo se torna o problema. A análise geométrica da posição do efetuador para encontrar os valores angulares das juntas se traduz em um conjunto extenso de equações não lineares. O problema analisado dessa forma pode ter múltiplas soluções, ou até mesmo não possuir soluções admissíveis (SICILIANO *et al.*, 2010).

A resolução da cinemática inversa não é trivial; uma das formas tradicionais de resolver o problema é desacoplar as juntas de posição e de orientação, simplificando a solução. Essa técnica é aplicável apenas para manipuladores de seis graus de liberdade que utilizam punho do tipo esférico (SPONG *et al.*, 2005). Dessa maneira é possível através da análise geométrica da posição do efetuador estipular primeiro os ângulos de posicionamento θ_1 , θ_2 e θ_3 . Na sequência, é possível calcular os ângulos de orientação θ_4 , θ_5 e θ_6 , de maneira independente, baseado na matriz de rotação do frame 6 em relação ao frame 3.

Além da abordagem geométrica, existem métodos algébricos para encontrar a solução da cinemática inversa. Um dos métodos consiste na utilização da base de Grobner. Esse método se trata de um algoritmo algébrico que pode ser aplicado a um dado conjunto de polinômios não nulos, produzindo um conjunto finito de geradores. Assim sendo, ao encontrar soluções para os geradores, é possível determinar a solução de todo o conjunto. Como as variáveis de juntas podem ser encontradas através de 12 polinômios não nulos, ao conciliar a base de Grobner com a convenção de Denavit-Hatemberg de um dado robô, é possível determinar sua pose. A resolução se mostra computacionalmente eficiente, uma vez que as equações produzidas para determinação das seis variáveis de junta são matematicamente mais simples de serem resolvidas (DA SILVA *et al.*, 2019).

Outra maneira de determinar a pose do robô para que o efetuador alcance determinado

ponto no espaço, é através de métodos iterativos. Uma aproximação linear da solução pode ser feita através da matriz Jacobiana. Essa é uma matriz de derivadas parciais que modela os movimentos do efetuador a partir da variação de ângulos de uma articulação. Dessa forma, é possível determinar as variáveis de juntas que levarão o efetuador à pose final através da inversa da matriz Jacobiana.

Esse método possui restrições; além de não ter soluções únicas, a matriz Jacobiana J pode não ser quadrada ou invertível. E mesmo que seja invertível, pode não funcionar bem em pontos próximos a singularidade. Várias abordagens derivadas desse método foram propostas a fim de contornar esses problemas. Dentre estes, pode-se destacar o método da Pseudo-Inversa, da Jacobiana Transposta, da Decomposição em valores singulares (SVD), o algoritmo DLS (no inglês, damped leastsquare), o método FIK (no inglês, feedback inverse kinematics) e o método CCD, dentre outros (ARISTIDOU e LASENBY, 2011). A maioria dessas técnicas apresentadas são baseadas em métodos numéricos ou de otimização. Em geral, o desempenho das soluções apresentadas é avaliado de acordo com a estabilidade, custo computacional e robustez às singularidades.

Em especial, uma solução alternativa para o problema é baseada em um algoritmo que estima a inversa da matriz Jacobiana dinamicamente. Esse algoritmo, denominado inversa filtrada, fornece uma solução mesmo quando são consideradas poses inviáveis ou inalcançáveis, ou seja, que estão próximas do ponto de singularidade. O método evita que seja necessário a inversão da matriz jacobiana para cada ponto da trajetória, fato que além de demandar menos recursos computacionais evita inconsistências matemáticas (VARGAS, 2013).

Outra forma de resolver o problema da cinemática inversa é conhecido como subproblemas de Paden-Kahan. Esse método consiste em utilizar técnicas de análise geométrica, baseado nas rotações de cada eixo, para simplificar o problema, eliminando dependências de algumas das variáveis de juntas conforme o modelo do robô (MURRAY *et al.*, 1994). Basicamente a técnica simplifica o movimento de rotação e translação de um corpo rígido em três subproblemas clássicos de resolução conhecida. Como a posição do efetuador pode ser definida como a resultante dos movimentos de rotação e translação de cada junta, a aplicação dos subproblemas de Paden-Kahan pode determinar a cinemática inversa do robô.

Todos os métodos apresentados possuem certas desvantagens que se tornam complicadores para determinadas circunstâncias. Os métodos geométricos se tornam mais complexos quanto mais elaborada for a geometria do robô. Já os métodos iterativos possuem soluções únicas que dependem necessariamente do ponto de partida. Em outras palavras, para tratar o problema da cinemática inversa de maneira genérica, para manipuladores com n graus de liberdade, os métodos tradicionais são menos eficientes por terem formulações matemáticas um tanto quanto mais complexas (KÖKER *et al.*, 2004).

Uma solução viável, sobretudo para robôs mais complexos, são os algoritmos de aprendizado de máquina. Por exemplo, uma rede neural de retro propagação, com função de ativação sigmoideal, pode ser usada para resolver problemas de cinemática inversa. Nesse método são

amostrados um conjunto de valores angulares que atendem a um determinado ponto no espaço utilizando um polinômio cúbico. Esse conjunto de dados é utilizado para treinar uma rede neural artificial (RNA). O algoritmo é treinado até que o erro de distância no espaço entre o efetuador e o ponto desejado seja mínimo. Essa abordagem do problema gera soluções adequadas (KÖKER, 2013).

E variações desse tipo de técnica podem tornar os resultados da cinemática inversa ainda mais satisfatórios. Uma rede neural configurada em paralelo apresenta menores erros médios das juntas e um melhor rendimento em detrimento das redes neurais simples. Nessa variante é adicionado um direcionador de padrões de treinamento que tem a finalidade de identificar qual RNA deve ser ativada para um padrão de treinamento específico. A escolha de qual RNA é atuada se dá através de um estudo de densidade de distribuição dos pontos no espaço de trabalho do robô (NUNES, 2016).

Entretanto as técnicas de redes neurais também possuem algumas desvantagens. Um primeiro limitador está relacionado com a amostragem de dados que deve ser feita para o treinamento das redes neurais. Para torná-la possível é necessário um conhecimento prévio do problema. Além disso quanto maior o número de dados amostrados melhor a qualidade do treinamento da rede neural, portanto, por vezes é necessária uma coleta extensa para resultados adequados. Importante citar ainda sobre a demanda de recursos computacionais e o tempo para treinamento dessas redes neurais. Em seu trabalho, Nunes (2016) cita que a rede neural configurada em paralelo demandou um treinamento de 10.000 épocas com duração de 27 horas, considerando um manipulador de 3 graus de liberdade. Köker (2013) afirma que “como as redes neurais são aproximadores universais com uma precisão arbitrária, a precisão da solução obtida requer aprimoramento para aplicações críticas”. O problema é agravado se o espaço de trabalho do robô for grande e conseqüentemente as possibilidades de decisões for aumentada.

Uma maneira de contornar essas dificuldades é utilizando algoritmos genéticos para melhorar a precisão dos resultados. Esses podem ser tomados como população inicial para um algoritmo genético de forma a minimizar o erro de posição do efetuador. Através dessa combinação de estratégias a precisão da solução é significativamente aprimorada para os níveis de micrômetro. Esta abordagem pode ser especialmente útil em aplicações que requerem alta precisão como micro-montagem, corte, perfuração e operações médicas (KÖKER, 2013). Além da aplicação em conjunto com as redes neurais, os algoritmos evolutivos podem, por si só, apresentar soluções ao problema. A próxima seção tratará mais detalhadamente sobre esse tipo de método na solução da cinemática inversa, demonstrando alguns trabalhos relacionados a esse assunto.

2.2. Trabalhos Relacionados

Os algoritmos genéticos (AGs) são inspirados na teoria da seleção natural, onde a partir de uma população inicial, são cruzados os melhores indivíduos a fim de encontrar o indivíduo

ótimo, ou seja, a melhor solução do problema (HOLLAND, 1992). Nesse tipo de algoritmo os indivíduos trocam informações entre si através de operadores genéticos de cruzamento e mutação, evoluindo em novas gerações de indivíduos mais aptos. Esses indivíduos são selecionados por uma função objetivo, *fitness*, que os atribui valores de custo de acordo com sua aptidão para resolução do problema. Esse processo é executado até que os critérios de parada pré-definidos sejam alcançados. Os pormenores dessa técnica será detalhados na seção 2.4.

Os AGs podem ser aplicados aos mais diversos tipos de problemas; em especial a utilização destes para soluções em robótica tem se tornado mais recorrente. Mais especificamente, nesse tipo de aplicação cada indivíduo é uma possível configuração de juntas que formam uma pose para levar o robô a uma pose desejada. A representação genética é dada por n ângulos, sendo n o grau de liberdade do robô. A população passa por um processo evolutivo, através de cruzamentos entre indivíduos e mutações, por um número determinado de gerações. Cada geração é produzida de acordo com a estratégia de seleção aplicada, e o processo é repetido até que o critério de convergência seja atendido. O objetivo final é encontrar os ângulos necessários para atingir a posição desejada, o que se traduz em encontrar o indivíduo mais apto em termos genéticos (MOMANI *et al.*, 2016). Dessa forma o algoritmo tende a evoluir gerando como resultado uma pose que provoque o menor deslocamento angular do robô para alcançar a posição desejada com menor erro possível.

Um exemplo é a utilização de algoritmos genéticos com imigrantes aleatórios para propiciar a interação de robôs móveis com ambientes dinâmicos. Nesse método ocorre a substituição do pior indivíduo da população corrente e de seus vizinhos próximos por novos indivíduos aleatórios. O intuito é permitir que os indivíduos escapem dos ótimos locais presentes na superfície de *fitness*, induzidos pelas mudanças inerentes ao problema não estacionário (TINÓS, 2007).

Ainda acerca de robôs móveis, outra aplicação é no controle do caminhar de robôs com pernas (HEINEN, 2007). Os AGs são utilizados para otimizar os parâmetros de técnicas de controle como tabelas de ângulos e posição para autômatos, parâmetros de elipse em funções cíclicas e pesos sinápticos de redes neurais. A função *fitness* é estipulada a partir do deslocamento total do robô, além de considerar sensores que indicam o toque das patas no chão e a estabilidade do robô. Um segundo algoritmo genético é utilizado para evoluir a morfologia mais adequada do robô para permitir o caminhar que, associado ao primeiro, apresenta soluções satisfatórias.

Os AGs também podem ser aplicados para robôs com outras configurações. Uma das possibilidades é a resolução da cinemática inversa de um robô planar com 3DoF minimizando o deslocamento angular das juntas (NUNES, 2007). Uma variação dessa solução é a utilização de algoritmos genéticos contínuos. Esse método possui o mesmo objetivo dos AGs convencionais, entretanto são aplicados a nível do espaço das juntas, ao contrário do convencional onde os operadores são aplicados a nível do espaço de trabalho. Esse tipo de estratégia é útil na suavização da trajetória do robô, mas requer um tempo mais elevado para execução do algoritmo (MO-

MANI *et al.*, 2016).

Uma abordagem que se assemelha às estratégias anteriores é a utilização de um algoritmo genético híbrido. Nesse caso dois algoritmos são utilizados: o primeiro faz uma exploração dentro do espaço de trabalho do robô a fim de identificar boas áreas, ou nichos, onde pode estar a solução. Na sequência o outro algoritmo busca a solução ótima dentro das áreas previamente determinadas. Essa estratégia contribui com a rápida convergência para soluções precisas, além de flexibilizar o uso para manipuladores diversos, bastando para tanto adequar as equações de cinemática direta (PAVAN *et al.*, 2018).

Outra vantagem dos algoritmos evolutivos na resolução da cinemática inversa é que essa pode ser resolvida considerando mais objetivos para alcançar a pose desejada. Simultaneamente o algoritmo pode minimizar a energia despendida, o tempo gasto, o número de rotações necessárias, além de minimizar o erro de posição no espaço. A redução do dispêndio de energia é baseada no consumo energético de cada motor do manipulador durante a movimentação ($W.s/^\circ$). Para a redução do tempo da trajetória é considerada a especificação da velocidade de rotação de cada junta ($^\circ/s$). A quantidade de rotações considera o deslocamento angular de todas as juntas ($^\circ$). Já o erro de posição no espaço é baseado nas equações de distância euclidianas (mm). Dessa forma o *fitness* é definido como uma somatória de cada um desses objetivos de otimização, que ao atingir valores de ótimo global apresenta bons resultados de minimização de tempo e gasto energético (ŠTEVO *et al.*, 2014).

A presença de vários objetivos em um problema gera um conjunto soluções ótimas, em vez de uma única solução ideal. Por esse motivo, assim como proposto por Števo *et al.* (2014), Rubrecht *et al.* (2012) e Ferrentino *et al.* (2019), os algoritmos evolutivos têm sido amplamente utilizados na otimização desse tipo de problema, inclusive para a resolução da cinemática inversa, que, conforme exposto na seção 2.1, pode ter inúmeras soluções conforme geometria e grau de liberdade do robô. Isso porque algoritmos desse gênero, inclusive os algoritmos genéticos, são muito bem adaptados para otimização em vastos espaços de pesquisa contínua. Um exemplo de aplicação dessa solução é na evolução da morfologia de um manipulador para ambientes altamente restritos, como um túnel de uma máquina de perfuração. Nesse caso, técnicas de algoritmos genéticos são utilizadas a fim de minimizar o erro linear máximo, o número de graus de liberdade e o tamanho do manipulador, assim como o número de colisões por segmento a cada interação, de um manipulador que cobrirá o espaço de trabalho restrito (RUBRECHT *et al.*, 2012).

Os AGs são úteis também na otimização do planejamento de trajetórias de manipuladores robóticos. Um exemplo é a aplicação para otimização do tempo da trajetória quando o manipulador tem movimentos restritos, mas se conhece o caminho exato a ser percorrido. Ou seja, desde que o manipulador não seja redundante, o caminho do espaço articular relacionado pode ser escolhido a partir de um conjunto finito de poses e ser otimizado com relação ao tempo. Para tanto o problema é parametrizado em termos de velocidade e distância em relação a variação de tempo da trajetória, definindo dessa forma a representação genética dos indivíduos.

A função *fitness* nesse caso é determinada como minimização de tempo, considerando para isso a área sob a curva da trajetória (FERRENTINO *et al.*, 2019).

Outra possibilidade é construir a função objetivo considerando o desvio acumulado entre a trajetória representada por um determinado indivíduo e a trajetória pretendida. Sendo a representação de cada indivíduo um conjunto de várias poses que formam uma trajetória, e o gene dessa representação as variáveis de junta para cada uma delas. Além disso, são avaliadas pelo algoritmo a discrepância entre a trajetória desejada e a trajetória realizada no início e no fim. Esse tipo de estratégia mostra uma boa repetibilidade para essa aplicação, pois apesar de apresentar trajetórias finais diferentes, todas mostram qualidade semelhante (PIRES, 1998). A tabela 2.1 apresenta um compilado das referências encontradas sobre a aplicação de algoritmos genéticos no campo da robótica:

Tabela 2.1: AG's aplicados à Robótica

Aplicação	Abordagem	Autor
Cin. Inversa	Algoritmos genéticos contínuos	(MOMANI <i>et al.</i>, 2016)
Cin. Inversa	Algoritmos genéticos híbridos	(PAVAN <i>et al.</i>, 2018)
Cin. Inversa	Fitness multiobjetivo	(ŠTEVO <i>et al.</i>, 2014)
Robôs Móveis	AG's com imigrantes aleatórios	(TINÓS, 2007)
Robôs Móveis	Otimização de técnicas de controle	(HEINEN, 2007)
Robôs Planares	Minimização do deslocamento	(NUNES, 2007)
Morfologia	Otimização da morfologia do robô	(RUBRECHT <i>et al.</i>, 2012)
Trajetória	Tempo em trajetória conhecida	(FERRENTINO <i>et al.</i>, 2019)
Trajetória	Minimização de erro da trajetória	(PIRES, 1998)

2.3. Robótica

Nessa seção são apresentados alguns conceitos fundamentais de robótica. Primeiro serão tratados assuntos como distância euclidiana e movimentos de corpos rígidos, que culminarão na apresentação de conceitos de cinemática de robôs. Na sequência, será tratado sobre o planejamento de trajetórias para manipuladores robóticos, baseado no conceito de polinômios de quinta ordem. Por fim, serão apresentados dois manipuladores da ABB relevantes para esse projeto. O intuito é evidenciar suas características físicas e cinemáticas que serão importantes para a implantação da solução proposta, além de explicar sobre a linguagem de programação RAPID.

2.3.1. Conceitos Fundamentais

Nessa dissertação é relevante conhecer conceitos sobre o aspecto construtivo do manipulador, seus graus de liberdade e seu espaço de trabalho. Robôs são equipamentos compostos por elos unidos por juntas em uma cadeia cinemática aberta ou fechada. Essas juntas podem ser de

rotação (R) ou prismáticas (P). De acordo com as disposições das juntas esses robôs podem ser divididos em 5 tipos distintos, Articulados (RRR), como os modelos antropomórficos da ABB utilizados como objeto de estudo desta dissertação, Esférico (RRP), Scara (RRP), Cilíndrico (RPP) e Cartesiano (PPP). Cada uma dessas juntas concede uma possibilidade de movimento ao robô, portanto o somatório do número de juntas corresponde ao seus graus de liberdade, do inglês Degrees of Freedom (DoF). E o volume total percorrido pelo efetuador, dadas todas as possibilidades de execução de movimento das juntas, determina o espaço de trabalho do manipulador. Diretamente relacionado à geometria do robô, o espaço de trabalho contém todos os pontos que o efetuador do robô pode alcançar Siciliano *et al.* (2010).

2.3.2. Espaço Euclidiano

Espaço euclidiano é um espaço vetorial real de dimensão finita e permite entender o estudo das relações entre ângulos e distâncias no espaço (CALLIOLI *et al.*, 1993). O espaço euclidiano pode ser interpretado para n dimensões, entretanto nesse trabalho o estudo será restringido ao espaço tridimensional que é o interesse na área de robótica.

Um ponto no espaço pode ser representado pelas coordenadas cartesianas x , y e z . Da mesma forma um espaço tridimensional pode ser representado pelos vetores \vec{i} , \vec{j} , \vec{k} e suas respectivas componentes conforme equação 2.1:

$$\vec{v} = v_1\vec{i} + v_2\vec{j} + v_3\vec{k}. \quad (2.1)$$

A norma Euclidiana de um vetor é definida como a distância da origem até o ponto que o vetor representa. Portanto a distância no espaço Euclidiano pode ser definida pelo Teorema de Pitágoras, da geometria euclidiana, conforme a equação 2.2:

$$|v| = \sqrt{v_1^2 + v_2^2 + v_3^2}. \quad (2.2)$$

2.3.3. Movimento de Corpos Rígidos

No espaço Euclidiano o movimento de uma partícula é descrito pela sua posição, a cada instante, com respeito a um sistemas de coordenadas inercial. Quando a distância entre duas partículas pertencentes ao mesmo corpo rígido não se altera, mesmo havendo movimento do corpo, ou que nele seja aplicado uma força, é possível afirmar que estas formam um corpo rígido. Em um corpo rígido pode ser aplicado movimentos de rotação e translação. Para referenciar esse movimentos são estipulados um sistema de coordenadas, cujo origem define a posição do corpo no espaço, e os eixos cartesianos a sua orientação em relação a uma referência inercial, conforme mostrado na Figura 2.1.

O movimento de rotação pode ser descrito a partir da matriz de cossenos diretores, considerando as rotações de cada eixo cartesiano com respeito a origem. A partir desse conceito

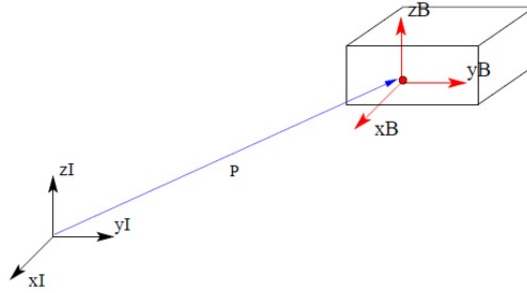


Figura 2.1: Translação e rotação de um corpo rígido.

é possível definir as matrizes de rotações elementares R_z , R_y e R_x , que representa as rotações em torno dos respectivos eixos de coordenadas z, y e x, conforme matrizes 2.3, 2.4, 2.5:

$$R_z(\alpha) = \begin{bmatrix} \cos(\alpha) & -\text{sen}(\alpha) & 0 \\ \text{sen}(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (2.3)$$

$$R_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \text{sen}(\beta) \\ 0 & 1 & 0 \\ -\text{sen}(\beta) & 0 & \cos(\beta) \end{bmatrix}, \quad (2.4)$$

$$R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\gamma) & -\text{sen}(\gamma) \\ 0 & \text{sen}(\gamma) & \cos(\gamma) \end{bmatrix}. \quad (2.5)$$

Em outras palavras, essas matrizes descrevem a rotação ao redor de um eixo para alinhar o sistema de coordenadas a que pertence ao do referencial inercial. É possível compor movimentos de rotações sucessivos dos corpos rígidos através da multiplicação matricial das diferentes matrizes de rotação ao redor do sistema de coordenadas correntes, conforme equação 2.6. Ou seja, uma rotação do terceiro frame, ou frame 3, em respeito a base, ou frame 0 (R_{03}), é dada pela rotação do frame 1 em relação ao frame 0 (R_{01}), multiplicado pelo rotação do frame 2 em relação ao frame 1 (R_{12}), por fim multiplicando a rotação do frame 3 em relação ao frame 2 (R_{23}):

$$R_{03} = R_{01}R_{12}R_{23}. \quad (2.6)$$

2.3.4. Transformadas Homogêneas

A posição de um corpo rígido no espaço pode ser descrito através de sua translação em respeito ao frame de referência, e sua orientação pode ser expressa através das componentes do vetor unitário anexado ao corpo com referência ao mesmo frame (SICILIANO *et al.*, 2010). Uma maneira simplificada e compacta de representar a posição e a orientação do corpo rígido

é através da transformada homogênea. Essa notação matemática é uma matriz que agrega a matriz de rotação do corpo em respeito ao frame de referência e sua respectiva translação no espaço, conforme matriz 2.7:

$$T_1^0 = \begin{bmatrix} R_1^0 & o_1^0 \\ 0^T & 1 \end{bmatrix}, \quad (2.7)$$

nesse caso a transformada homogênea T_1^0 representa a rotação do corpo rígido do frame1 em relação ao frame0 (R_1^0), e a translação nas coordenadas x, y e z, do frame1 em relação a origem (o_1^0).

2.3.5. Ângulos de Orientação

Um método tradicional de representar a orientação de um corpo são os ângulos *Roll*(Φ), *Pitch*(Θ) e *Yaw*(Ψ), RPY. Muito comum no campo aeronáutico, esses ângulos representam um movimento de rolagem em torno do eixo z, arfagem em torno do eixo y e guinada em torno do eixo x. Esse método de representação é relevante para esse projeto, pois é o sistema de orientação utilizado pelos manipuladores da ABB.

Mais especificamente, definindo a sequência de rotação como x, y, z, em respeito ao frame fixo, a orientação resultante é calculada pela pré multiplicação matricial das matrizes de rotações elementares R_z , R_y e R_x , assim como demonstrado pela equação 2.8 (SICILIANO *et al.*, 2010):

$$R = R_z(\Phi)R_y(\Theta)R_x(\Psi) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \quad (2.8)$$

onde R é a denotação genérica da matriz de rotação do efetuador em respeito a base. Daí, é possível determinar através de métodos geométricos os ângulos de orientação, Φ , Θ e Ψ , conforme as equações 2.9, 2.10, 2.11:

$$\Phi = \text{atan2}(r_{21}, r_{11}), \quad (2.9)$$

$$\Theta = \text{atan2}(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}), \quad (2.10)$$

$$\Psi = \text{atan2}(r_{32}, r_{33}). \quad (2.11)$$

Apesar de não ser a abordagem utilizada para desenvolvimento da solução apresentada nessa solução, os controladores da ABB também interpretam orientações pelo sistema de representação de quatérnios. Esses elementos matemáticos foram propostos por Sir William Rowan Hamilton em estudos que buscavam estender o plano complexo ao espaço de três dimensões, e aplicados para representar orientação por A. Cayley. A menor complexidade da estrutura matemática dos quatérnios exige um menor tempo de processamento em relação as

manipulações de matrizes de rotações, por esse motivo tem se tornado cada vez mais comum a utilização desses recursos para representar a orientação de robôs (FRANQUEIRA, 1993). Dessa forma os quatérnios passaram a ser relevantes no estudo de manipuladores robóticos de forma que a equação 2.12:

$$Q_{n-1}^n = e^{\frac{\theta_n}{2}\vec{i}} e^{\frac{\alpha_n}{2}\vec{k}}, \quad (2.12)$$

representa as matrizes de rotações 2.3, 2.4 e 2.5 de acordo com seu respectivo eixo de rotação. Onde, Q_{n-1}^n representa o quatérnio de rotação do eixo n em respeito ao eixo $n-1$, θ_n e α_n são ângulos de rotação, \vec{i} e \vec{k} são vetores unitários de um sistema de coordenadas ortogonal.

2.3.6. Cinemática Direta

O problema da cinemática direta diz respeito a pose do efetuador dada a configuração das juntas. Essas são os ângulos entre dois links do robô para caso de juntas rotacionais, ou o deslocamento linear para o caso de juntas prismáticas (SPONG *et al.*, 2005). Considerando que cada junta fornece um grau de mobilidade associado a uma variável angular, é possível determinar a posição e orientação do efetuador em respeito a base do robô através da rotação e translação provocado por cada junta. Para tanto é possível descrever essa relação matematicamente através das transformadas homogêneas. Conforme demonstra a equação 2.13, a multiplicação matricial entre todas as transformada homogêneas da cadeia cinemática, $T_{01}, T_{12} \dots T_{(n-1)n}$, resultará na transformada homogênea do efetuador em respeito a base, T_{0n} , o que descreve o problema da cinemática direta:

$$T_{0n} = T_{01}(\theta_1)T_{12}(\theta_2)T_{23}(\theta_3)T_{34}(\theta_4) \dots T_{(n-1)n}(\theta_n). \quad (2.13)$$

Em suma o objetivo dessa análise é determinar o efeito acumulado das variáveis de junta na posição e orientação do efetuador final. Para isso é possível simplificar a solução introduzindo convenções, como a representação de Denavit-Hartenberg (SPONG *et al.*, 2005).

2.3.7. Convenção de Denavit-Hartenberg

A convenção de Denavit-Hartenberg baseia-se no fato de que para determinar a posição relativa de duas retas no espaço, são necessários somente dois parâmetros. O primeiro é a distância medida ao longo da normal comum entre duas retas e o segundo é o ângulo de rotação em torno da normal comum, que uma das retas deve girar, de forma que fique paralela. Consequentemente para definir a posição relativa de dois sistemas de coordenadas são necessários quatro parâmetros (CABRAL, 2016). A intercessão dos eixos de um sistema de coordenadas define a origem do mesmo. Portanto, a partir da definição da posição relativa entre dois eixos de dois sistemas de coordenadas, pode-se descrever a posição relativa entre os próprios sistemas. Dessa forma, desde que o eixo x seja perpendicular ao eixo z do frame anterior e que eles

se interceptem é possível determinar a transformada homogênea entre esses frames a partir da equação 2.14:

$$A_{i-1}^i = Rot_z(\theta_i)Trans_z(d_i)Trans_x(a_i)Rot_x(\alpha_i), \quad (2.14)$$

onde, θ_i são as variáveis de junta, d_i o deslocamento ao longo do eixo z, a_i o deslocamento ao longo do eixo x e α_i a rotação entre os frames. O método propõe, como forma de facilitar a análise, a construção de uma tabela considerando a associação desses valores, conhecida como tabela D.H.

2.3.8. Planejamento de Trajetórias

Para o caso particular em que são conhecidas as poses inicial e final do robô, e não existam obstáculos em seu espaço de trabalho, é possível determinar os ângulos das juntas entre as duas poses, através de um polinômio de terceira ordem (KÖKER *et al.*, 2004). Para isso é necessário entender que, ao longo de um período de tempo determinado, a pose do robô varia a cada instante durante a trajetória. Assim sendo é possível afirmar que a variação dos ângulos de juntas podem ser descritos conforme equações 2.15 e 2.16:

$$\theta(0) = \theta_0, \quad (2.15)$$

$$\theta(t_f) = \theta_f, \quad (2.16)$$

ou seja no instante de tempo inicial as posições das juntas são iguais θ_0 , e θ_f para o instante final. Consequentemente é possível afirmar, pela derivada de primeira ordem, que as velocidades no instante inicial e final são iguais a 0, conforme equações 2.17 e 2.18:

$$\frac{\partial \theta(0)}{\partial t} = 0, \quad (2.17)$$

$$\frac{\partial \theta(t_f)}{\partial t} = 0, \quad (2.18)$$

a partir dessas premissas a trajetória pode ser descrita pelo polinômio cúbico da equação 2.19:

$$\theta(t) = a_0 + a_1(t) + a_2(t^2) + a_3(t^3), \quad (2.19)$$

a partir desta, é possível definir a velocidade e a aceleração de θ em determinado instante de tempo de acordo com as derivadas de primeira e segunda ordem respectivamente, conforme as equações 2.20 e 2.21:

$$\frac{\partial \theta}{\partial t} = a_1 + 2a_2(t) + 3a_3(t^2), \quad (2.20)$$

$$\frac{\partial^2 \theta}{\partial t^2} = 2a_2(t) + 6a_3(t). \quad (2.21)$$

Manipulando as equações 2.20 e 2.21, é possível encontrar o polinômio cúbico que define a posição do robô ao longo do tempo a partir da posição inicial conforme equação 2.22:

$$\theta_i(t) = \theta_{i0} + \frac{3}{t_f^2}(\theta_{if} - \theta_{i0})t^2 - \frac{2}{t_f^3}(\theta_{if} - \theta_{i0})t^3, \quad (2.22)$$

onde $i = 1, 2, \dots, n$; é o número de juntas do robô. t_f é o tempo definido do percurso em segundos, t é o instante de tempo atual em segundos, θ_{i0} é o ângulo da junta na posição inicial, θ_{if} é o ângulo da junta na posição final e n é o número de juntas do manipulador.

Essa estratégia é precisa para determinar a pose do robô em cada instante da trajetória e apresenta velocidades contínuas para as juntas. Entretanto o polinômio cúbico esbarra em um problema de descontinuidade nas acelerações das juntas, provocando movimentos impulsivos ao manipulador.

Esse é um prolema relevante porque quão mais complexo for o trabalho a ser executado pela manipulador melhor deve ser o controle da trajetória que este cumprirá até chegar a pose final com precisão. Essa premissa se reforça se houver obstáculos a serem evitados em seu espaço de trabalho, porque é necessário garantir que o robô chegue a pose final com precisão, por um caminho bem definido.

Para tanto é possível realizar um controle em malha fechada durante o movimento do manipulador de forma a absorver as perturbações em seu movimento. Essas perturbações podem ser oriundas dos acoplamentos dinâmicos e variações inerciais existentes do robô. Dessa forma, através de sensores, o sinal de saída, contendo as perturbações provocadas pela dinâmica do atuador, são realimentadas no controlador do sistema (MARIANO *et al.*, 2013).

Uma das estratégias de controle que pode ser utilizada nesse caso é o controle Proporcional Integral Derivativo (PID). A utilização de um controlador desse gênero permite que o deslocamento da junta do robô seja suavizado ao longo do tempo para um dado sinal de referência enviado à junta, atingindo um erro em regime estacionário igual a zero. Isso permite com que o manipulador se desloque até a pose desejada com precisão (SPONG *et al.*, 2005).

É possível também planejar a trajetória utilizando um método em malha aberta, no qual a saída do manipulador não é enviada como um feedback ao controlador. A estratégia consiste em incorporar restrições às acelerações iniciais e finais das juntas, tal qual para a posição e velocidade, conforme equação 2.22. Essas restrições são necessárias para evitar variações instantâneas de comando das juntas, provocando movimentos bruscos do robô. Dessa forma são necessárias seis variáveis e um polinômio de quinta ordem como na equação 2.23 (SPONG *et al.*, 2005):

$$q(t) = a_0 + a_1t + a_2t^2 + a_3t^3 + a_4t^4 + a_5t^5, \quad (2.23)$$

a partir deste, a velocidade é determinada pela derivada da posição resultando em um polinômio de quarta ordem, como demonstrado pela equação 2.24. Em consequência a aceleração é expressa por uma função de terceira ordem, conforme equação 2.25. Essa estratégia evita os

pontos de singularidade, com acelerações estipuladas em zero para os instantes iniciais e finais. A Figura 2.2 demonstra o comportamento da posição, velocidade e aceleração das juntas ao longo de uma trajetória definida em 2 segundos.

$$\frac{\partial \theta}{\partial t} = a_1 + 2a_2(t) + 3a_3(t^2) + 4a_4(t^3) + 5a_5(t^4), \quad (2.24)$$

$$\frac{\partial^2 \theta}{\partial t^2} = 2a_2 + 6a_3(t) + 12a_4(t^2) + 20a_5(t^3). \quad (2.25)$$

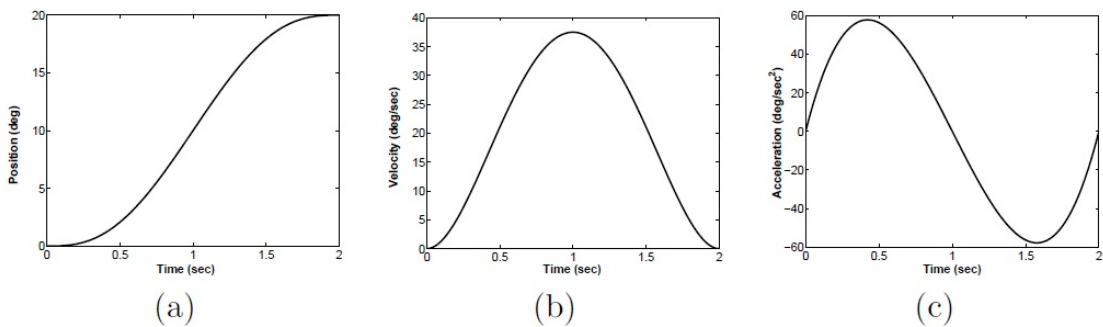


Figura 2.2: (a) Posição da junta (b) Velocidade da junta (c) Aceleração da junta (SPONG *et al.*, 2005).

2.3.9. Manipuladores Robóticos da ABB

Nessa seção serão apresentados dois manipuladores da ABB que serão objetos de estudo deste trabalho. Para ambos são apresentadas as principais características que serão necessárias para os cálculos cinemáticos do robô. Além disso, conhecer esses manipuladores será importante para a construção da população inicial, função objetivo e restrições do algoritmo genético como será mostrado na seção 3.1. A escolha pelos robôs da ABB foi baseada no fato que grande parte das aplicações que rodam na Vale atualmente utilizam esses equipamentos.

O IRB 120 é um manipulador robótico da geração mais recente da ABB. Esse é um robô de menor porte, com capacidade de carga de até 3Kg e ideal para manuseio de materiais e aplicações de montagem. Um equipamento mais ágil, compacto e leve, com uma boa precisão de trajetória (ABB, 2019a). Esse robô foi escolhido como um dos objetos de estudo desse trabalho por ser um modelo em escala dos robôs antropomórficos de punho esférico da ABB. Além disso, esse é o equipamento disponível no laboratório do Instituto Tecnológico Vale (ITV) em Ouro Preto, o que permitirá que uma série de testes sejam realizados, antes que a solução possivelmente progrida para aplicação industrial. O fato de ser um modelo em escala dos demais manipuladores também facilitará a adequação dessa solução para processos diversos dentro da própria Vale. Isso porque os cálculos cinemáticos para os demais modelos são os mesmo, bas-

tando para tanto adaptar os parâmetros da tabela de Denavit-Hartenberg com as características do robô que será utilizado.

O outro manipulador que será estudado é o IRB6700. Esse é um robô de escala industrial, com payload de 150 a 300Kg dependendo de sua versão e com alcance de até 3,20m. O estudo desse robô nessa dissertação tem o intuito de aproximar a aplicação dos resultados a uma escala mais próxima dos processos industriais. Além disso, conforme será exposto na seção 4.7, esse manipulador foi um modelo encontrado disponível para testes em ambiente industrial.

IRB120

Esse modelo possui 6 juntas rotacionais, em cadeia aberta, configuração antropomórfica e punho esférico conforme demonstra a Figura 2.3 (ABB, 2019b).

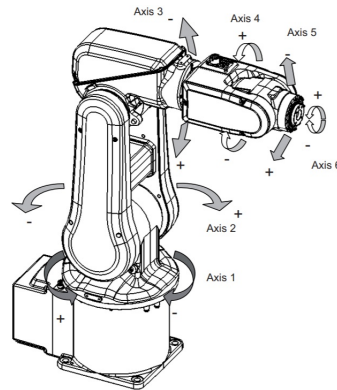


Figura 2.3: Manipulador IRB 120 (ABB, 2019b).

Para a utilização desse robô uma das características fundamentais a serem conhecidas são suas dimensões. Essa informação, aliada com o tipo das juntas e suas configurações, é importante para o cálculo da cinemática direta, assim como da inversa. A Figura 2.4, mostra todas as dimensões relevantes para os cálculos cinemáticos.

Outras informações relevantes são os limites das juntas do manipulador. O espaço de trabalho do robô é definido pela formatação que essas juntas podem alcançar. Em especial para o cálculo da cinemática inversa utilizando algoritmos heurísticos é essencial conhecer esses limites para formatar a população inicial, bem como delimitar a faixa de valores em que a solução poderá constar. A tabela 2.2 especifica os valores mínimos e máximos para todas as 6 juntas do IRB120.

Outros parâmetros relativos a estrutura física do robô, relevantes para os objetivos propostos nessa dissertação, são os torques dos motores de cada junta. Cada motor deve exercer determinada quantidade de esforço conforme a posição da junta. Isso porque as juntas da base devem ter força o suficiente para movimentar toda a estrutura do robô. Em contrapartida, os motores do punho esférico, que orientam a ferramenta, devem suportar um esforço menor, portanto sua capacidade de torque é mais baixa. Os valores de torque para cada uma das juntas foram extraídos do manual do IRB120 (ABB, 2019b), conforme mostrado na tabela 2.3. Para

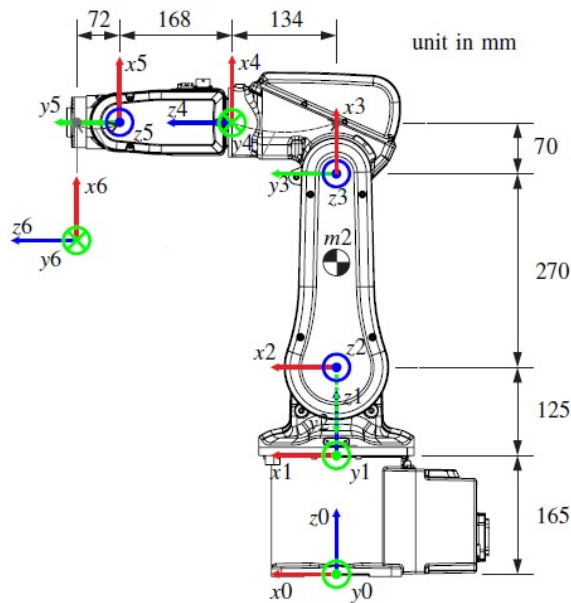


Figura 2.4: Características físicas e cinemáticas do IRB120 Adaptado (VARHEGY *et al.*, 2017).

Tabela 2.2: Limites de ângulos das juntas do IRB 120 (ABB, 2019b).

$Junta_i$	θ_{min}	θ_{max}
1	-165°	165°
2	-110°	110°
3	-110°	70°
4	-160°	160°
5	-120°	120°
6	-400°	400°

esse manipulador em específico os motores das juntas 1 e 2 são iguais, assim como os motores das juntas 4 e 5.

Tabela 2.3: Torque das juntas do IRB 120 (ABB, 2019b).

$Junta_i$	Torque(τ)
1	± 295 Nm
2	± 295 Nm
3	± 85 Nm
4	± 4.8 Nm
5	± 4.8 Nm
6	± 2.2 Nm

IRB6700

Por ser um robô desenvolvido para aplicações industriais, o IRB6700 possui dimensões bem maiores que o IRB120. Isso torna seu alcance e seu espaço de trabalho bem maiores, conforme evidenciado pela Figura 2.5 (ABB, 2019c).

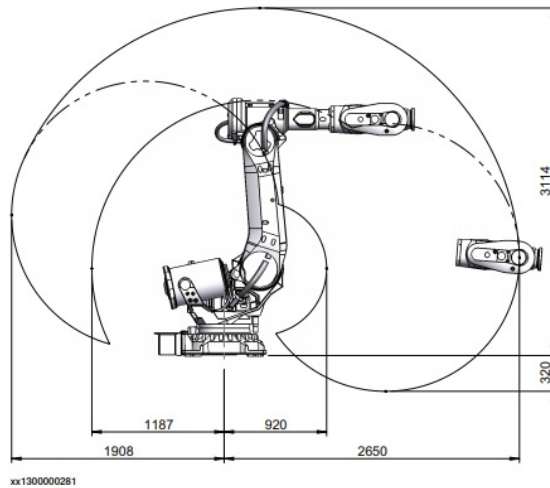


Figura 2.5: Espaço de trabalho do IRB6700 (ABB, 2019c).

Em comum com o modelo apresentado anteriormente, o IRB6700 também é um modelo antropomórfico, com 6 graus de liberdade e punho esférico. Pelos mesmos motivos apresentados para o IRB120, conhecer seus parâmetros físicos e cinemáticos também será importante para alcançar os objetivos propostos. Assim sendo as dimensões do IRB6700 são demonstradas na Figura 2.6. Na Tabela 2.4 constam os limites dos ângulos e na Tabela 2.5 os torques para cada uma das juntas do IRB6700.

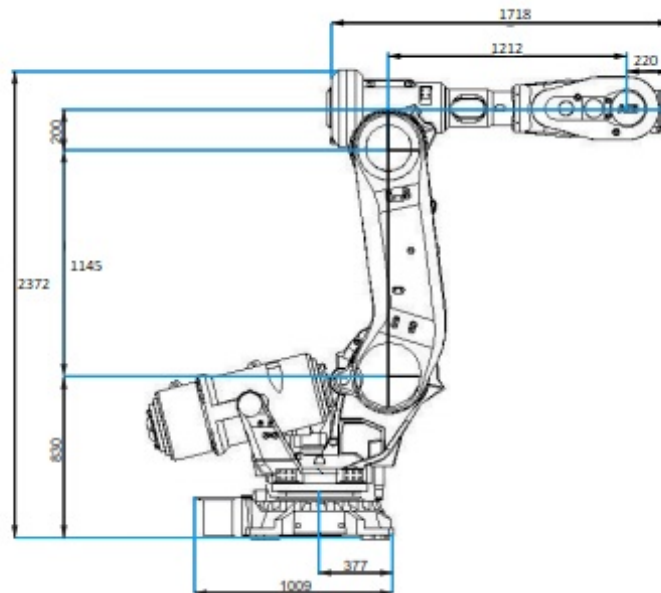


Figura 2.6: Dimensões do IRB6700 (Adaptado (ABB, 2019c)).

Tabela 2.4: Limites de ângulos das juntas do IRB6700 (ABB, 2019c).

$Junta_i$	θ_{min}	θ_{max}
1	-170°	170°
2	-65°	85°
3	-180°	70°
4	-300°	300°
5	-130°	130°
6	-360°	360°

Tabela 2.5: Torque das juntas do IRB6700 (ABB, 2019c).

$Junta_i$	Torque
1	± 21 kNm
2	± 21 kNm
3	± 5 kNm
4	± 1135 Nm
5	± 1135 Nm
6	± 570 Nm

Linguagem RAPID

Os manipuladores mais modernos da ABB tem um controlador que executa o sistema operacional IRC5. A linguagem de programação interpretada por esse sistema operacional é a RAPID. Essa é uma linguagem de programação flexível e de alto nível, que apesar de possuir comandos simples e de fácil entendimento, permitem a criação de aplicações altamente sofisticadas (ABB, 2018).

Através dessa linguagem é programada a movimentação do robô baseado em variáveis do espaço cartesiano e do espaço das juntas. Permite também programar tarefas adjacentes como rotinas de busca, interações com entradas e saídas digitais e criação de sockets de comunicação. Essa é uma linguagem proprietária da ABB utilizada para programar seus controladores de robôs. Apesar de que alguns estudos têm sido desenvolvidos para encontrar formas de realizar o controle dos robôs utilizando outros softwares, ainda assim todos as soluções necessitam minimamente de execução de códigos em Rapid, pelo menos para criar o socket de comunicação que permitirá o comando externo.

2.4. Algoritmos Genéticos

Algoritmos genéticos são mecanismos heurísticos de otimização baseados em conjuntos bem definidos de objetivos e restrições, ou regras, destinados a realizar uma busca estocástica polarizada durante um número finito de etapas, a fim de apresentar uma solução ótima. Esse método de otimização foi inspirado nos mecanismos de evolução de populações de seres vivos. A teoria proposta pelo naturalista Charles Darwin, em sua obra “A origem das espécies” publi-

cada em 1859, revela que em um ambiente natural a espécie mais apta têm maiores chances de sobreviver e estarão mais propensas a gerar descendentes (LACERDA e CARVALHO, 1999).

Holland (1992) foi o precursor na tradução da ideia de Darwin para o conceito de inteligência artificial. A teoria propõe que um conjunto de dados gerados aleatoriamente, ou não, sejam testados pelo algoritmo e os que trazem melhores resultados, ou mais aptos, são selecionados através de uma função matemática que atribui custos maiores ou menores conforme a aptidão do indivíduo. Aqueles selecionados são recombinados entre si criando novas gerações que novamente serão avaliadas. Esse processo é repetido até que o resultado ótimo seja encontrado ou o número máximo de gerações seja findado (LACERDA e CARVALHO, 1999).

2.4.1. Representação Genética

As estruturas de dados, que são possíveis soluções ao problema, são representadas através de cromossomos que contém uma formatação que será interpretada e processada pelo algoritmo genético. Uma representação muito comumente utilizada é a binária, entretanto outras formas podem ser utilizadas, desde que carreguem informações coerentes ao problema. Um exemplo disso são os cromossomos utilizados na representação genética desse problema, que conforme será tratado na seção 3.1.2, representam as variáveis de junta do manipulador. As informações contidas dentro de cada cromossomo são denominadas genótipos. Já o conjunto de todas as configurações que o cromossomo pode assumir é denominado espaço de busca, ou seja, o agregado de todas possíveis soluções ao problema.

Já um determinado conjunto de cromossomos, ou indivíduos, formam uma população para iniciar o processamento do algoritmo. A partir dos indivíduos contidos na população inicial são iniciados os operadores genéticos, como cruzamento e mutação, que provocarão a evolução até um indivíduo ótimo. A população pode ser determinada aleatoriamente, ou a partir de determinados quesitos conhecidos a fim de diminuir o espaço de busca e melhorar a performance para convergência ao resultado.

2.4.2. Cruzamento e Mutação

O cruzamento é a troca de informações entre indivíduos selecionados para geração de novos indivíduos. A ideia principal é a propagação de características do melhor indivíduo. E esse operador é uma das principais características que distingue os algoritmos genéticos das demais técnicas de busca aleatória. Existem diversas formas diferentes de realizar o cruzamento, dentre elas o cruzamento de um ponto, de dois pontos, de multi pontos, cruzamento heurístico, aritmético, dentre outros.

Já a mutação é a técnica de alterar arbitrariamente uma ou mais informações genéticas do indivíduo permitindo uma variação da população, garantindo sua diversidade. A introdução de material genético novo, permite ao algoritmo explorar toda o espaço de soluções, evitando que o problema se estabeleça em um mínimo local.

2.4.3. Elitismo

Elitismo é o processo de seleção de melhores soluções da população combinada, a fim de evitar a degradação de qualquer boa solução (RUBRECHT *et al.*, 2012). Em outras palavras, essa estratégia transfere um cromossomo de uma geração para outra sem alterações. Isso evita que uma possível solução ótima seja perdida entre uma geração e outra ao sofrer um cruzamento com outro indivíduo ou uma mutação.

2.4.4. Seleção

Assim como descreve a lei de seleção natural, onde o indivíduo mais apto tem mais chance de sobreviver e de propagar sua espécie, o operador de seleção em algoritmos genéticos também busca selecionar os melhores indivíduos. A ideia é dar aos indivíduos mais aptos preferência na propagação e reprodução durante o processo evolutivo. Isso permite que esse indivíduo mantenha ou passe suas características para as próximas gerações (HEINEN, 2007). Os métodos de seleção podem ser diversos, como o método por torneio, método por roleta, dentre outros. Não é possível definir um melhor método em detrimento de outro, cada um pode ter pior ou melhor comportamento conforme o problema que está sendo otimizado. O conceito fundamental é que o critério de seleção deve ser bem definido de forma que os melhores indivíduos sejam propagados e a solução encontrada.

2.4.5. Função Objetivo

Para que seja realizada a seleção é utilizada uma função matemática, denominada de função objetivo, ou *fitness*. Essa função é a responsável por atribuir custos conforme a aptidão de cada indivíduo, conseqüentemente dita como o algoritmo genético vai evoluir. Por esse motivo é fundamental que seja construída de maneira correta para que o resultado convirja para os valores ótimos esperados. Importante ressaltar também que a complexidade da função objetivo afeta diretamente no desempenho do algoritmo genético. Dessa forma, simplificar a função pode reduzir a demanda por recursos computacionais e também o tempo de convergência.

2.4.6. Critérios de Parada

Os indivíduos são testados pela função objetivo, selecionados e evoluem até que um determinado critério de parada seja alcançado. O algoritmo pode ser interrompido ao atingir determinado número de gerações, quando alcança o valor ótimo ou quando não há evoluções durante algumas gerações (LACERDA e CARVALHO, 1999). Os critérios de parada podem estar baseados ainda no cumprimento de determinadas restrições que o indivíduo deve obedecer para ser considerado solução ao problema. Nesse caso o algoritmo só será interrompido caso os critérios da restrição sejam alcançados.

2.4.7. Genetic Algorithm Optimization ToolBox

Não é fácil traduzir conceitos da ciência genética para uso em programas de computador. O principal problema é a construção de um código genético que pode representar a estrutura de diferentes problemas (HOLLAND, 1992). Considerando esse fator, nas últimas décadas diversas linguagens de programação têm sido desenvolvidas e estudadas para o desenvolvimento de algoritmos genéticos. Uma das pioneiras para tratar do problema foi a linguagem FORTRAN. Mais recentemente a linguagem C, ou PHYTON tem sido utilizadas de maneira recorrente para este tipo de aplicação.

Um dos softwares que permite o desenvolvimento de algoritmos genéticos é o MATLAB da desenvolvedora MathWorks. O software em questão apresenta um método de toolboxes com funções específicas para os mais diversos tipos de aplicações, desde sintonia de controladores até simulações de robôs. Uma das ferramentas desse software é o Global Optimization Toolbox que fornece funções que pesquisam soluções globais para problemas que contêm vários máximos ou mínimos. Dentre diversas técnicas para problemas de otimização que existem dentro do toolbox, uma delas é o algoritmo genético.

O Genetic Algorithm Optimization ToolBox, conhecido como GAOT, pode ser usado para problemas de otimização em que a função objetivo ou restrição é contínua, descontínua, estocástica, não possui derivadas ou inclui simulações ou funções de caixa preta. O software permite criar algoritmos genéticos personalizados, definindo as funções desejadas como o tipo de cruzamento, mutação, tamanho da população, dentre outras possibilidades. Mais especificamente o GAOT cria um algoritmo genético para resolver problemas de otimização suaves ou não suaves com qualquer tipo de restrição, incluindo restrições de número inteiro. Esses modelos de algoritmos são estocásticos de base populacional, que pesquisam a solução ótima aleatoriamente através de métodos de seleção entre membros da população que tem sua diversidade garantida por métodos de cruzamento e mutação (MATHWORKS, 2020).

Uma das principais vantagens da utilização desse recurso é a facilidade de implantação uma vez que não é necessário criar linhas de código para realizar as funções do AG, como a seleção, cruzamento ou mutação. Todas essas funções já são nativas do sistema, bastando para tanto configurar o algoritmo genético conforme o desejado. Esse fator contribui sobretudo com a avaliação de diferentes formatações do algoritmo genético, pois sem alterar a estrutura do algoritmo construído é possível mudar os métodos de seleção, cruzamento, mutação, bem como as taxas de cruzamento, tamanho da população, ou até mesmo inserir funções como elitismo e sobreposição de população sem grandes dificuldades.

2.5. Métodos de Integração e Simulação

No desenvolvimento de aplicações com robôs é essencial simular os resultados teóricos. Entretanto, em geral, os robôs de alta performance são caros e escassos no ambiente de pesquisa,

o que muitas vezes dificulta testes e experimentos para os pesquisadores. Nesse contexto, as simulações em ambientes virtuais ganham importância, e para isso existem diversos softwares que podem trazer bons resultados (PERALTA *et al.*, 2016).

Conforme já foi descrito nesse capítulo, essa dissertação se concentra em aplicações com robôs da ABB, por esse motivo é necessário entender métodos de integração de softwares diversos com o RobotStudio. E uma das principais dificuldades em lidar com esse software é justamente a integração com outros softwares ou dispositivos que não sejam do mesmo fabricante. As interações com outras ferramentas podem propiciar mais recursos, para aplicações mais complexas, como os próprios algoritmos de aprendizado de máquina, ou por exemplo aplicações de técnicas de visão computacional. Assim sendo, algum esforço tem sido empregado para desenvolver soluções que possam prover essa integração e não é raro encontrar pesquisas relacionadas a esse tema.

Uma dessas soluções propõe a integração do RobotStudio, com o Robot Operation System (ROS) (ROS, 2018). O ROS é uma ferramenta flexível, de código aberto, que incentiva a implementação colaborativa de softwares para sistemas robóticos, de forma a simplificar a tarefa de criação de robôs complexos e robustos. Nessa estratégia o ROS passa a ser o sistema central de controle do robô. A ferramenta “*MoveIt!*” calcula a trajetória e envia através de um socket ao controlador IRC5 que colocará o robô em movimento conforme comando. A flexibilidade é a principal vantagem dessa estratégia, sobretudo por facilitar a integração com outros dispositivos robóticos (TORRE, 2017).

Outra possibilidade viável é a integração do RobotStudio com o software LabView. Nesse método o LabView recebe os sinais do robô através de sensores para coordenar os movimentos necessários para o funcionamento da célula robotizada. Além disso, esse tipo de integração permite a supervisão do robô, inclusive de maneira remota via WEB (QUESADA, 2014).

Um terceiro método é controlar os manipuladores da ABB utilizando o MATLAB. Nessa hipótese é criada uma interface entre os dois softwares utilizando protocolo TCP/IP e ferramentas gráficas de interface com o usuário. A estratégia gera bons resultados sobretudo para controle no nível do espaço cartesiano, isto é, enviando pontos de destino da ferramenta (CORBACHO, 2014). E como já foi citado na seção 2.4.7, o MatLab possui ferramentas específicas para a utilização de algoritmos heurísticos. Portanto a estratégia de integração do RobotStudio com o MatLab, conforme proposto por Corbacho (2014), pode trazer grandes vantagens para o cálculo da cinemática inversa utilizando AGs, que é o principal objetivo proposto nessa dissertação.

Contudo, o RobotStudio se trata de um software comercial que, para utilização de todos os seus recursos, requer uma licença cujo valores são, possivelmente, inacessíveis a pesquisadores não financiados. A versão gratuita do software, apesar de permitir a comunicação com o MatLab, cerceia a utilização de mais recursos de simulação, como por exemplo a inserção de novos elementos externos ao robô. Sem a aquisição da licença de desenvolvimento não é

possível inserir novas ferramentas, trilhos, esteiras ou qualquer outro elemento que não seja o próprio robô da ABB. Por esse motivo, é necessário conhecer a integração com outros tipos de softwares de simulação que permitam de maneira gratuita uma gama maior de recursos para o desenvolvimento da aplicação.

Um dos softwares que possibilitam essas simulações é o CoppeliaSim. Esse software é utilizado para o desenvolvimento rápido de algoritmos, simulações de automação de fábrica, prototipagem, verificação rápidas e educação relacionada à robótica. Esse simulador possui uma versão para estudantes que pode ser acessada de maneira gratuita e que permite a construção de simulações elaboradas, inclusive utilizando manipuladores da ABB. Outra vantagem desse software é seu ambiente de desenvolvimento integrado, baseado em uma arquitetura de controle distribuído. Isso torna o CoppeliaSim muito versátil e ideal para aplicações de robótica, podendo estas serem desenvolvidas em diversos tipos de linguagens, inclusive MatLab (COPPELIA, 2018a).

3. Metodologia Proposta

Para alcançar o objetivo de determinar a cinemática inversa de um manipulador de até sete graus de liberdade, a proposta desse trabalho é desenvolver e investigar técnicas de algoritmos genéticos através da ToolBox GAOT do MatLab, apresentada na seção 2.4.7. A ideia principal é aproveitar a flexibilidade dessa ferramenta para propor diversas variações do algoritmo a fim de determinar aquele que traz melhor resultado ao problema.

Posteriormente a proposta é simular a solução em ambientes virtuais que permitam certificar a eficácia do algoritmo desenvolvido, tal qual descrito na seção 2.5. A ideia inicial é integrar a solução do MatLab com o RobotStudio, uma vez que todas as simulações serão feitas utilizando manipuladores da ABB. Essa integração permitirá verificar a eficácia de manipuladores com até 6 graus de liberdade. O passo seguinte dessa metodologia será integrar o algoritmo genético desenvolvido ao CoppeliaSim. Nesse simulador será possível acrescentar um trilho que propiciará um deslocamento lateral ao robô, conferindo ao sistema um sétimo grau de liberdade. A simulação será realizada considerando o controle de posição das setes juntas para alcançar a pose determinada pelo AG, como também será simulado o controle de trajetórias através da velocidade das juntas.

3.1. Algoritmos Genéticos no MatLab

Conforme exposto na seção 2.4, os algoritmos genéticos são aqueles inspirados na teoria da seleção natural, onde a partir de uma

3.1.1. Representação Genética

inicial, são cruzados os melhores indivíduos a fim de encontrar o indivíduo ótimo, ou seja, a melhor solução do problema. Para o caso em questão o objetivo é determinar as variáveis de juntas que levarão o robô à pose desejada, com o menor deslocamento angular possível, com margens de erro, de posição e orientação, aceitáveis.

Uma das principais vantagens da utilização desse tipo de algoritmo é que não necessitam de conhecimento prévio do problema a ser tratado (NUNES, 2007). Isso porque, todos os indivíduos iniciais, ditos população inicial, são gerados aleatoriamente sem, portanto, interessar inicialmente qual a disposição desejada no final. Existe a possibilidade de que a população seja alterada e inserido indivíduos conhecidamente mais próximos da solução final, essa técnica pode ajudar numa conversão mais rápida para o resultado esperado. Esse método será detalhado na seção 4.2.

3.1.2. Representação Genética

Como o interesse final do AG é encontrar a melhor pose para um robô antropomórfico com até sete graus de liberdade, é necessário determinar as variáveis de juntas que resultarão na pose desejada. Por esse motivo os indivíduos deve ter como representação genética os ângulos, das juntas do manipulador, que serão aleatoriamente determinados. Essa determinação aleatória está condicionada que todos os ângulos estejam dentro dos limites de juntas expostos na tabela 2.2. Tratar indivíduos que estejam fora dos limites de operação do robô pode gerar erros no resultado, retornando uma solução fora do espaço de trabalho. Dessa forma o modelo padrão do cromossomo é representado pela equação 3.1.

$$i = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6). \quad (3.1)$$

Da mesma maneira, para o caso em que é inserido uma junta prismática como um sétimo eixo que provoque um deslocamento lateral ao robô, o indivíduo deve conter mais um fenótipo que indique em mm o deslocamento deste eixo. Assim sendo, passa a além dos seis ângulos das juntas rotativos do manipulador, passa a conter uma variável d_0 relativa ao deslocamento lateral conforme indicado na equação 3.2.

$$i = (d_0, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6). \quad (3.2)$$

3.1.3. População Inicial

Conhecidos a representação genética que atende à solução do problema proposto, é possível determinar a população inicial utilizando a função de biblioteca aberta, “crtpr”, exposta no Anexo A.2, para geração de indivíduos aleatoriamente. São determinados para essa função o tamanho de cada cromossomo, que no caso é um conjunto de 6 genes que representam os ângulos de cada junta do robô. Além dos ângulos são estabelecidos os limites dentro dos quais os indivíduos podem ser gerados aleatoriamente, que correspondem aos limites dos ângulos para cada junta. E por fim, o tamanho da população, variável que será objeto de investigação dessa dissertação.

3.1.4. Função Objetivo

A determinação da função objetivo do problema é uma etapa crucial e não trivial, que diz respeito à atribuição matemática de custos relacionados à aptidão de cada indivíduo da população para ser a solução ótima. Para esse caso os objetivos são três: a minimização do deslocamento angular, do erro de posição e de orientação da pose final. O intuito é encontrar a pose final que acarrete o menor deslocamento angular possível ao robô. Para os três casos

podem ser utilizados conceitos de distância euclidiana expostos na seção 2.3.2. Dessa forma as equações 3.3, 3.4 e 3.5 representam os objetivos para solução da cinemática inversa:

$$f_1 = \sqrt{(x - p_x)^2 + (y - p_y)^2 + (z - p_z)^2}, \quad (3.3)$$

$$f_2 = \sqrt{(\Phi - \Phi_d)^2 + (\Theta - \Theta_d)^2 + (\Psi - \Psi_d)^2}, \quad (3.4)$$

$$f_3 = \sqrt{(\theta_1 - \theta_{1i})^2 + (\theta_2 - \theta_{2i})^2 + \dots + (\theta_6 - \theta_{6i})^2}, \quad (3.5)$$

onde f_1 corresponde à minimização do erro de distância de posição no espaço, considerando x , y e z o ponto que se deseja alcançar e p_x , p_y e p_z a posição do efetuador resultante das variáveis de juntas do indivíduo testado pelo algoritmo. Já f_2 diz respeito à minimização do erro de orientação, onde Φ , Θ e Ψ são os ângulos, *roll*, *pitch* e *yaw*, resultantes do indivíduo em avaliação e Φ_d , Θ_d e Ψ_d é a orientação desejada. Por fim, f_3 é a função com o objetivo de minimizar o deslocamento angular, sendo $\theta_1, \theta_2 \dots \theta_6$ as variáveis de juntas que compõe o indivíduo em avaliação pelo algoritmo e $\theta_{1i}, \theta_{2i} \dots \theta_{6i}$ as variáveis de juntas na posição inicial. Assim sendo a função objetivo (F.O.) é dada pelo somatório das três equações, conforme equação 3.6:

$$[F.O.] = f_1 + f_2 + f_3. \quad (3.6)$$

Com essa formatação o algoritmo genético passa a ser multiobjetivo e o resultado ótimo se dará pela minimização das três variáveis propostas. Apesar da função objetivo somar valores de grandezas distintas, esse não é um fator impeditivo para a resolução do problema. Em suma, a interpretação dos números é realizada de maneira adimensional. Minimizando o total das três equações para um valor ótimo global, será garantido o menor deslocamento angular, com erros de posição e orientação dentro dos parâmetros aceitos pelas restrições do problema. É possível ponderar a função objetivo concedendo maiores ou menores pesos para cada uma das três parcelas. Dessa forma se o interesse maior for minimizar o deslocamento das juntas deve-se atribuir um fator multiplicando f_3 maior que as demais parcelas de forma que a minimização do deslocamento seja reforçada. O mesmo vale para as parcelas de minimização do erro.

O deslocamento angular, calculado pela equação 3.5, é uma métrica baseada no deslocamento calculado eixo a eixo do robô. Assim sendo, quanto menor for o deslocamento para cada junta, melhor será a otimização alcançada. Já para a minimização do erro de posição e de orientação, é considerada a distância de cada indivíduo que está sendo testado em relação à pose desejada. Dessa forma, como a representação genética dos indivíduos são os ângulos da junta, é necessário calcular a cinemática direta para determinar a posição do efetuador para cada indivíduo em teste.

Conforme exposto na seção 2.3.7, uma das maneiras mais convencionais de calcular a cinemática direta de um manipulador é através da convenção de Denavit-Hartenberg. Baseado nos modelos cinemáticos do IRB120 apresentado na seção 2.3.9 foi possível construir a

Tabela 3.1, proposta pela convenção DH, onde a_i a distância ao longo de x_i , da origem O_i à interseção dos eixos x_i e z_{i-1} , d_i a distância ao longo de z_{i-1} , de O_{i-1} à interseção dos eixos x_i e z_{i-1} , α_i é o ângulo do eixo z_{i-1} para o eixo z_i , medido em torno de x_i , θ_i o ângulo do eixo x_{i-1} para o eixo x_i , medido em torno de z_{i-1} :

Tabela 3.1: Tabela D.H. do IRB120.

$Junta_i$	$a_i(\text{mm})$	$d_i(\text{mm})$	$\alpha_i(^{\circ})$	$\theta_i(^{\circ})$
1	0	290	-90	θ_1
2	270	0	0	$\theta_2 - 90$
3	70	0	90	θ_3
4	0	302	-90	θ_4
5	0	0	90	θ_5
6	0	72	0	$\theta_6 + 180$

A partir da tabela é possível determinar a transformada homogênea do efetuador em respeito a base, T_{06} , conforme equação 2.14. Ao conciliar o resultado da convenção D.H. com o modelo da transformada homogênea padrão, exposta na matriz 2.7, é possível determinar as coordenadas cartesianas, p_x , p_y e p_z , alcançadas pelo efetuador a partir das variáveis de junta relativas ao indivíduo sendo testado pelo AG, como demonstrado pela matriz 3.7:

$$T_{06} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (3.7)$$

onde T_{06} é uma matriz ($\in SO(3)$), dada por uma rotação ($\in SO(3)$) mais uma translação ($\in R^3$). Daí, é possível aplicar p_x , p_y e p_z na equação 3.3 para determinar o valor f_1 da primeira função objetivo. De T_{06} também é possível extrair a matriz de rotação da transformada homogênea, tal qual a matriz 2.8, e então aplicar as equações dos ângulos RPY, 2.9, 2.10, 2.11, para determinar a orientação da ferramenta relativa às variáveis de juntas do indivíduo que está sendo analisado pelo algoritmo. Dessa forma é possível determinar os ângulos Φ , Θ e Ψ que aplicados na função 3.4 retornarão o valor f_2 para o segundo objetivo.

Em resumo, para cada indivíduo é determinada a posição, a orientação e o deslocamento angular, que são avaliados pela função objetivo. A função atribuirá custos menores ao indivíduo quanto menor for o erro de posição, orientação e o deslocamento angular em relação ao ponto desejado. Um menor custo aumenta a probabilidade das características genéticas desse indivíduo serem propagadas para as próximas gerações. Os cálculos da função objetivo implementados no MatLab estão expostos nos Anexos A.1 e A.3.

3.1.5. Restrições e Critérios de Parada

Ao buscar a solução ótima é preciso garantir que os erros de posição e orientação do efetuador do robô estejam dentro de parâmetros aceitáveis, o mais próximo possível de zero. Em outras palavras, o algoritmo pode se deparar com uma solução de ótimo local e terminar sua busca entendendo que tenha encontrado a melhor solução, ainda que essa esteja distante da posição e orientação desejadas. Por esse motivo, os erros de posicionamento e orientação do robô são tratados como uma restrição para a solução. Ou seja, para cada possível solução o algoritmo checa a correspondência com a pose desejada, sendo essa uma das condições de interrupção da busca do algoritmo. Dessa forma a evolução acontecerá até que as condições da restrição sejam atendidas, garantindo que o efetuador chegue ao ponto determinado com orientação desejada.

Nesse caso, são estipulados limites aceitáveis de erro de posição de 1 milímetro(mm) e de orientação de 1 grau(°). O objetivo de estipular a margem de erro na restrição é reduzir o custo computacional e tempo necessário para encontrar a solução perfeita onde todos os erros são iguais a 0. A julgar pelas aplicações industriais que serão analisadas no capítulo 5, é possível afirmar que erros máximos de tais magnitudes não causarão impactos na efetividade da solução. As implementações matemáticas que representam as restrições de posicionamento, em milímetros, são demonstradas pelas equações 3.8, e de orientação, em graus, pela equações 3.9, essas também estão baseadas nos conceitos de espaço euclidiano demonstrados na seção 2.3.2:

$$\sqrt{x^2 - p_x^2} \leq 1, \quad \sqrt{y^2 - p_y^2} \leq 1, \quad \sqrt{z^2 - p_z^2} \leq 1, \quad (3.8)$$

$$\sqrt{\Phi^2 - \Phi_d^2} \leq 1, \quad \sqrt{\Theta^2 - \Theta_d^2} \leq 1, \quad \sqrt{\Psi^2 - \Psi_d^2} \leq 1, \quad (3.9)$$

as variáveis p_x , p_y , p_z , Φ , Θ e Ψ , assim como para a função objetivo, são determinados pelo cálculo da cinemática direta. A maneira como são calculadas são exatamente iguais a mostrada na seção anterior. Os cálculos das restrições desenvolvidos no MatLab são demonstrados no Anexo A.4.

Um possível questionamento seria sobre a necessidade de minimizar o erro de posição e orientação uma vez que estes já são restrições ao problema. Mas, caso a minimização não ocorra, o algoritmo busca deslocamentos angulares mínimos ignorando a importância de minimizar o erro em relação à posição e orientação final. Dessa forma, as condições de restrição nunca são atendidas, impossibilitando uma solução viável.

3.1.6. Configuração do Algoritmo Genético

Nas seções anteriores foram explicadas as características da população inicial e realizada as demonstrações matemáticas da função objetivo e das restrições. O passo seguinte é construir o algoritmo genético, determinando os parâmetros de seleção, cruzamento, mutação,

número de gerações, elitismo, sobreposição de populações e demais parâmetros relevantes, no MatLab. A toolbox GAOT permite personalizar a execução do algoritmo determinando cada um desses parâmetros de maneira independente. Para tanto é utilizada a função “gaoptimset”, cujo parâmetros de entrada são justamente as configurações desejáveis ao algoritmo. Portanto, é necessário indicar o parâmetro a ser determinado e na sequência qual o valor a ele atribuído, como demonstrado no Anexo A.5. No método proposto nessa dissertação, foram determinados os seguintes parâmetros à função:

- *PopulationSize*: Determina o tamanho da população inicial que deve ser o mesmo tamanho da população gerada pela função “crtrp”. Este parâmetro é um dos objetos de investigação dessa dissertação; para tal foram determinadas variações do tamanho da população inicial com 10,25 e 40 indivíduos, conforme será tratado na seção 4.2.
- *InitialPopulation*: utilizado para povoar a população inicial, cujo tamanho foi determinado pelo parâmetro anterior. Para tanto, o conjunto de indivíduos gerados aleatoriamente pela função “crtrp” foi denominado “pop” e atribuído a esse parâmetro.
- *Generations*: Através deste parâmetro é determinado o número máximo de gerações que poderá conter o AG. Não necessariamente o número de gerações estipulado será executado, uma vez que o algoritmo será interrompido caso seja encontrado um valor ótimo que atenda às restrições para a solução. A variação desse parâmetro também será objeto de investigação dessa dissertação. Para isso, conforme exposto na seção 4.2, foram realizados testes considerando 25,40 e 55 gerações.
- *TolCon*: diz respeito à tolerância aceitável para que a restrição seja atendida. Na prática, as equações de restrição foram igualadas a 0 de forma que o algoritmo busque a solução perfeita, e o parâmetro TolCon determinado como 1. Dessa forma, caso as restrições de erro de posição e de orientação atinjam erros menores que 1mm e 1°, respectivamente, o algoritmo é interrompido conforme previsto pela modelagem matemática das restrições, nas equações 3.8 e 3.9.
- *SelectionFcn*: Nesse parâmetro é determinado o modo de seleção que será executado pelo algoritmo genético. Conforme o problema, os métodos de seleção podem ser mais ou menos efetivos, diminuindo o tempo de processamento, ou aumentando a robustez do algoritmo para escapar de valores de ótimo local. Por esse motivo, serão investigados nesse trabalho o método de seleção por torneio, por roleta e o método de seleção uniforme.
- *CrossoverFcn*: permite a seleção do tipo de cruzamento que o algoritmo genético vai fazer. A forma de cruzamento afeta diretamente a maneira como o algoritmo evolui ao resultado. Para a investigação são definidos os métodos de cruzamento heurístico, aritmético e single point.

- *CrossoverFraction*: determina a taxa de recombinação entre indivíduos em uma dada geração. Quanto maior esse valor, mais os indivíduos selecionados pela função de seleção transferem carga genética entre si. Os efeitos da variação da taxa de recombinação também serão estudados nessa dissertação, para tanto o parâmetro é analisado como 0,5(50%) e 0,8(80%).
- *MutationFcn*: Nesse parâmetro é determinado o tipo de mutação que o algoritmo realiza. Uma das desvantagens do GAOT é que não é possível determinar a taxa de mutação do algoritmo. Portanto essa taxa é definida de maneira default pelo MatLab conforme o método escolhido. Na investigação são considerados os métodos de mutação “adaptação viável”, Gaussiano e Uniforme.
- *MigrationFraction*: determina a sobreposição de populações. Ou seja, é possível determinar qual a porcentagem de indivíduos que necessariamente migrarão para a próxima geração. Durante as investigações são considerados a sobreposição de população de 0,2(20%) e 0,25(25%).
- *EliteCount*: estabelece as condições de elitismo, ou seja, quantos indivíduos serão migrados para a geração seguinte sem sofrer cruzamento ou mutação. As análises são realizadas considerando os efeitos da migração de 2 indivíduos para a geração seguinte sem alterações.
- População em Paralelo: Esse é um parâmetro binário que pode ser ativado ou não. Caso seja selecionado, o algoritmo processa mais de uma população paralelamente na busca de ganhar eficiência no processo de evolução. Será investigada uma hipótese de algoritmo onde duas populações são processadas em paralelo.

Determinados todos os parâmetros, é necessário realizar a chamada do algoritmo utilizando a função “ga”. Como parâmetros dessa função devem ser determinados, além das variáveis estabelecidas pela função “gaoptimset”, a função objetivo (“*fitness*”) e a função de Restrição (“*Constraint*”), conforme definidas nas seções anteriores. Além desses parâmetros, é necessário ainda determinar, para a função “ga”, o número de variáveis da solução, que para o problema são os 6 ângulos de juntas. Devem ser determinados também os limites inferiores e superiores aos quais a solução pode pertencer, assim como foi feito para a população inicial. Essa definição garante que a solução constará no espaço de trabalho do robô. A chamada da função “ga” também é demonstrada no Anexo A.5.

O resultado do algoritmo genético é dado por um vetor de três elementos “th”, “fval” e “exitflag”. O primeiro corresponde efetivamente ao objetivo desse trabalho, ou seja, o melhor indivíduo encontrado pelo processo de evolução. Para o caso, “th” corresponde ao vetor com às seis variáveis de junta que resultarão na pose otimizada. Já “fval” é o valor final da função de *fitness*. Nessa proposta o valor da minimização é pouco representativo por si tratar de um

número resultado de um somatório de três funções de dimensões diferentes conforme explicado na seção 3.1.4. O importante é entender que quanto menor esse valor melhor foi o processo de otimização, ou seja, o deslocamento angular, o erro de posição e de orientação, foram mais efetivamente minimizados. Por fim “exitflag” discrimina o motivo pelo qual o algoritmo genético foi interrompido. O ideal é que assuma sempre valor igual a 1, que significa que o valor ótimo foi encontrado. Entretanto o algoritmo pode ser interrompido por outros motivos, como por atingir o limite de gerações, nesse caso “exitflag” será igual a 0, ou -2, caso não seja encontrada uma solução viável. São 7 outras possibilidades de interrupção do algoritmo, menos comuns de acontecerem, que são descritas no *help* da função.

Com o retorno da função “ga” é findada a construção do algoritmo genético no MatLab. Após construir as funções objetivo, as funções de restrição, gerar uma população inicial aleatória e determinar os parâmetros desejados, ao executar o algoritmo genético são encontrados o indivíduo com as 6 variáveis de junta que levarão o manipulador à pose otimizada, o valor da função *fitness* e o motivo de interrupção do algoritmo. Os códigos completos relativos as funções citadas são demonstrados no Anexo A. O próximo passo do trabalho é construir um algoritmo que possa estipular uma trajetória ao robô, que o leve de uma pose inicial determinada até a pose final encontrada pelo AG.

3.2. Planejamento de Trajetória no MatLab

Conforme exposto na seção 2.3.8 uma das maneiras possíveis de traçar a trajetória de um robô, uma vez que são conhecidos os pontos iniciais e finais, é utilizando um polinômio cúbico. Nessa proposta a pose inicial é definida sempre como a posição de “Home”, onde todas as variáveis de junta são iguais a 0. Já as 6 variáveis de juntas da pose final são aquelas determinadas através da otimização pelo AG. Assim sendo, basta aplicar as variáveis de junta das poses inicial e final na equação 2.22, que será determinada a pose do robô para cada um dos instantes da trajetória com tempo definido t_f .

Dessa forma é gerada uma determinada quantidade de poses intermediárias que levarão o robô da posição inicial até a posição final determinada pelo algoritmo genético. Maior será o número poses intermediárias o quão maior for a resolução indicada pelo algoritmo ao longo do tempo t_f . Essa solução é mais simples, retornando um conjunto de vetores de 6 posições, cujo composição são os ângulos das 6 juntas para cada pose intermediária da trajetória.

Entretanto, conforme tratado na seção 2.3.8, essa solução possui descontinuidade na aceleração das juntas, o que provoca movimentos bruscos ao robô, tornando a solução ineficiente para ser aplicada em robôs reais. Portanto, nessa dissertação, a proposta de solução para o planejamento de trajetórias é utilizar o polinômio de quinta ordem para aplicar restrições à aceleração evitando esse tipo de problema. Assumindo que o robô parte do repouso e termina a trajetória em repouso, é possível afirmar que as velocidades e acelerações iniciais são iguais a zero, conforme prevem as equações 2.17 e 2.18. O tempo da trajetória t_f pode ser definido con-

forme a necessidade do processo e o instante inicial t_0 é igual a zero. Conforme é demonstrado por Spong *et al.* (2005) é possível encontrar os fatores $a_0, a_1 \dots a_6$ da equação 2.23 resolvendo a equação matricial 3.12, que representa a multiplicação da inversa da matriz 3.10 pela transposta da matriz 3.11:

$$M = \begin{bmatrix} 1 & (t_0) & (t_0)^2 & (t_0)^3 & (t_0)^4 & (t_0)^5 \\ 0 & 1 & 2(t_0) & 3(t_0)^2 & 4(t_0)^3 & 5(t_0)^4 \\ 0 & 0 & 2 & 6(t_0) & 12(t_0)^2 & 20(t_0)^3 \\ 1 & (t_f) & (t_f)^2 & (t_f)^3 & (t_f)^4 & (t_f)^5 \\ 0 & 1 & 2(t_f) & 3(t_f)^2 & 4(t_f)^3 & 5(t_f)^4 \\ 0 & 0 & 2 & 6(t_f) & 12(t_f)^2 & 20(t_f)^3 \end{bmatrix}, \quad (3.10)$$

$$B = \begin{bmatrix} p_0(\theta_1) & v_0(\theta_1) & ac_0(\theta_1) & p_1(\theta_1) & v_1(\theta_1) & ac_1(\theta_1) \\ p_0(\theta_2) & v_0(\theta_2) & ac_0(\theta_2) & p_1(\theta_2) & v_1(\theta_2) & ac_1(\theta_2) \\ p_0(\theta_3) & v_0(\theta_3) & ac_0(\theta_3) & p_1(\theta_3) & v_1(\theta_3) & ac_1(\theta_3) \\ p_0(\theta_4) & v_0(\theta_4) & ac_0(\theta_4) & p_1(\theta_4) & v_1(\theta_4) & ac_1(\theta_4) \\ p_0(\theta_5) & v_0(\theta_5) & ac_0(\theta_5) & p_1(\theta_5) & v_1(\theta_5) & ac_1(\theta_5) \\ p_0(\theta_6) & v_0(\theta_6) & ac_0(\theta_6) & p_1(\theta_6) & v_1(\theta_6) & ac_1(\theta_6) \end{bmatrix}, \quad (3.11)$$

$$a_{nq} = (M^{-1})(B^T) = \begin{bmatrix} a_{11} & a_{21} & a_{31} & a_{41} & a_{51} & a_{61} \\ a_{12} & a_{22} & a_{32} & a_{42} & a_{52} & a_{62} \\ a_{13} & a_{23} & a_{33} & a_{43} & a_{53} & a_{63} \\ a_{14} & a_{24} & a_{34} & a_{44} & a_{54} & a_{64} \\ a_{15} & a_{25} & a_{35} & a_{45} & a_{55} & a_{65} \\ a_{16} & a_{26} & a_{36} & a_{46} & a_{56} & a_{66} \end{bmatrix}, \quad (3.12)$$

onde a matriz M é a composição das variações angulares, de velocidade e aceleração das juntas em relação ao instante inicial da trajetória, t_0 , e o tempo da trajetória, t_f , conforme as equações 2.24 e 2.25. E a matriz B é composta pelas posições inicial(p_0) e final(p_f), velocidades inicial(v_0) e final(v_f) e acelerações inicial(ac_0) e final(ac_f) para cada uma das seis juntas do robô. Essa equação matricial pode ser solucionada pelo MatLab utilizando recursos simples conforme demonstrado no Anexo B.1.

Os elementos da matriz a_{nq} corresponde aos fatores a_n , $a_1, a_2 \dots a_6$, das equações 2.23, 2.24 e 2.25 para cada uma das seis juntas q_n , $q_1, q_2 \dots q_6$. Assim sendo é possível calcular a posição, velocidade e aceleração para cada uma das juntas ao longo da trajetória de tempo estipulado.

3.3. Cinemática Inversa para Robôs com 7DoF

O problema da cinemática inversa para um robô antropomórfico de seis graus de liberdade tem quatro soluções bem definidas que podem ser facilmente encontradas através dos

métodos tradicionais (SICILIANO *et al.*, 2010). Conforme exposto na Figura 3.1, para dada posição e orientação o robô assumirá as poses: pra frente com cotovelo para cima; para frente com cotovelo para baixo; para trás com cotovelo pra cima; e para trás o cotovelo para baixo.

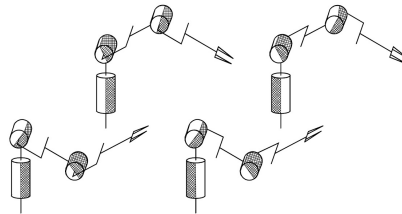


Figura 3.1: Possíveis soluções para a cinemática inversa de um manipulador com 6DoF (SICILIANO *et al.*, 2010).

Em contrapartida, ao anexar a esse tipo de robô uma junta externa que confira algum movimento de translação ou rotação, esse passa a ter um sétimo grau de liberdade e, por consequência, uma grande quantidade de soluções para a cinemática inversa. O sétimo eixo pode ser, por exemplo, um trilho ou uma esteira que suporte o robô, e que confira a esse uma translação latitudinal ou longitudinal. Recursos como estes não são difíceis de serem encontrados em ambientes industriais, e no capítulo 5 será demonstrada uma aplicação desse gênero em operação dentro da Vale.

O fato é que para esse tipo de configuração com sete graus de liberdade os métodos tradicionais para resolução da cinemática inversa perdem eficiência por se basearem em um conjunto de equações complexas sem soluções bem definidas. Em outras palavras para cada posição que a sétima junta prismática surge novas poses para as juntas rotativas do robô. Dessa forma conforme a precisão do robô e o alcance do sétimo eixo, o conjunto de possíveis soluções ao problema da cinemática inversa assume um número elevado de possibilidades. Os métodos tradicionais para solução desse problema podem ser agrupados em duas categorias principais, os métodos analíticos e os métodos numéricos.

Os métodos analíticos por se basearem na geometria da cadeia cinemática (método geométrico) ou na equação cinemática (método algébrico) tem sua aplicação dificultada quanto maior for o número de graus de liberdade do robô. Suas limitações estão na restrição da solução aos robôs solucionáveis e na complexidade de se obter as expressões específicas de cada robô. Em geral este método funciona bem para robôs com até seis graus de liberdade (ERTHAL, 1992).

Já os métodos numéricos independem da estrutura cinemática do robô tornando a abordagem do problema mais genérica. Entretanto por se basearem em cálculo iterativos, o aumento do número de graus de juntas do robô eleva o número de iterações a serem realizadas, tornando a solução potencialmente mais complexa, consumindo mais recursos computacionais e consequentemente aumentando o tempo para alcançar o resultado. Além disso pesa contra esse método o fato de levarem a uma única solução dentre todas possíveis (ERTHAL, 1992).

Em contrapartida, em cadeias cinemáticas com mais de seis graus de liberdade, métodos heurísticos como algoritmos genéticos podem ser facilmente adaptados para resolver o problema. De maneira simples, ao adaptar a tabela D.H. considerando um sétimo eixo nos cálculos de cinemática direta expostos na seção 3.1.4, e a representação genética dos indivíduos, a busca pela solução ótima ocorre da mesma maneira que para o problema do manipulador de 6 DoF.

Importante ressaltar ainda que os algoritmos genéticos aproveitam bem o fato do número de soluções possíveis ser aumentado com a inserção do sétimo eixo. Isso porque passam a existir mais resultados possíveis, que atendam as restrições colocadas ao problema, dentro do espaço de busca. Além disso, conforme é proposto por Števo *et al.* (2014), é possível considerar na busca pelo resultado da cinemática inversa uma função multiobjetivo. Isso permite que sejam otimizados outros fatores de interesse para que a pose final seja alcançada, como a minimização do deslocamento, do dispêndio energético, do erro em relação à pose desejada, dentre outros. Essa é uma possibilidade que os métodos analíticos e numéricos tendem a não considerar.

Assim sendo, o método proposto é adaptar o algoritmo genético exposto na seção 3.1 para resolver o problema da cinemática inversa para um manipulador com 7 graus de liberdade. A ideia é acrescentar um trilho, como uma junta prismática, que provoque ao manipulador uma translação ao longo do eixo y, servindo como um sétimo eixo ao robô.

Para tanto o primeiro passo é alterar a representação genética dos indivíduos para que esses, além de possuírem seis variáveis de junta angulares, tenham ainda um sétimo gene que expresse a translação da junta prismática em milímetros. Por consequência, é necessário estabelecer mais um limite para geração aleatória do sétimo gene do indivíduo. Nesse caso, como o trilho está sendo inserido somente na simulação e seu tamanho é adaptável, como referência foi tomado o trilho utilizado na célula de lavagem robotizada de Carajás, que possui aproximadamente 5000mm, conforme será demonstrado na seção 5.2. Dessa forma, a função “crtrp” é ajustada para que gere uma população com indivíduos com sete genes e a função “ga” é alterada para que retorne como solução um indivíduo com essa mesma característica.

Além disso, é necessário recalcular a cinemática direta para os cálculos da função objetivo e das restrições. Para isso, a tabela de Denavit Hatenberg do IRB120 é alterada de forma a considerar mais uma linha onde constarão as informações relativas aos sete eixos, conforme tabela 3.2.

Tabela 3.2: Tabela D.H. do IRB120 com 7DoF.

$Junta_i$	$a_i(\text{mm})$	$Translaçãolinear_i(\text{mm})$	Rotação em torno do eixo x(°)	Variável de Junta(°)
1	0	d_1	-90	0
2	0	290	-90	θ_2
3	270	0	0	$\theta_3 - 90$
4	70	0	90	θ_4
5	0	302	-90	θ_5
6	0	0	90	θ_6
7	0	72	0	$\theta_7 + 180$

Considerando a nova tabela adaptada é possível aplicar a equação matricial 2.14 proposta pela convenção D.H. Como o eixo da base se trata de uma junta prismática, a transformada homogênea será uma matriz identidade acrescida do fator d_1 de translação horizontal ao longo do eixo y, conforme demonstrado pela matriz 3.13:

$$T_{01} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (3.13)$$

Dessa forma, ao acrescentar T_{01} à transformada homogênea T do efetuador em respeito a base, será possível determinar a cinemática inversa para o manipulador com sete graus de liberdade como demonstrado na equação 3.14:

$$T_{07} = T_{01}T_{12}T_{23}T_{34}T_{45}T_{56}T_{67}. \quad (3.14)$$

Adaptada a cinemática direta e os limites considerando a junta prismática, é necessário alterar a função objetivo, f_3 , para contemplar a minimização do deslocamento da junta da base. Como a função é multiobjetivo é válido lembrar que as duas primeira parcelas da função objetivo não são alteradas. Isso porque, dizem respeito ao erro de posição no espaço cartesiano e ao erro de orientação. Ao adaptar a transformada homogênea acrescentando a sétima variável de junta, conseqüentemente a posição e orientação do robô será afetada. Já a terceira parcela da função objetivo, conforme demonstrado em na equação 3.5, considera o deslocamento de cada junta individualmente de acordo com os conceitos de distância euclidiana. Para a junta prismática anexada a base do robô não funciona diferente. Dessa forma a sétima variável de junta é agregada à função para contemplar a minimização da translação da base, conforme equação 3.15:

$$f_3 = \sqrt{(d_1 - d_{1i})^2 + (\theta_2 - \theta_{2i})^2 + (\theta_3 - \theta_{3i})^2 + \dots + (\theta_7 - \theta_{7i})^2}. \quad (3.15)$$

3.4. Eficiência Energética

Conforme citado na seção anterior, para o problema com seis graus de liberdade existem 4 soluções bem definidas, dessa forma ao executar o algoritmo genético as possíveis poses determinadas serão sempre as mesmas e portanto a quantidade de movimento de cada junta sempre semelhantes. Ao acrescentar o sétimo eixo o número de soluções é aumentado substancialmente, passando a ter inúmeras possibilidades para determinação da cinemática inversa, portanto as juntas podem se movimentar de maneira distinta para cada possível solução. Considerando esse fato, é possível ponderar a função objetivo do problema de forma a atribuir custos elevados para as soluções que provoquem deslocamentos maiores às juntas de maior torque, que

carregam um maior peso. Dessa forma é possível diminuir o deslocamento das juntas que gastam mais energia e, conseqüentemente, reduzir a quantidade total de energia despendida para chegar à pose desejada. Essa solução ganha relevância quanto maior for o robô, uma vez que as juntas necessitam de mais força para carregar o peso da estrutura.

Para melhorar a eficiência energética para locomover o robô à pose final determinada pelo AG, foram considerados como parâmetro os torques de cada junta expostos na Tabela 2.3. A ideia é que ao se locomover para a pose desejada o movimento de todas as juntas são responsáveis por 100% da energia despendida durante o movimento. Assim sendo, considerando τ o torque de cada junta, é possível determinar a porcentagem de influência que cada uma provoca no gasto energético total até a pose final, conforme equações 3.16 e 3.17:

$$\tau_t = \tau_1 + \tau_2 + \tau_3 + \tau_4 + \tau_5 + \tau_6, \quad (3.16)$$

$$W_i = \tau_i / \tau_t, \quad (3.17)$$

onde τ_t é o somatório do torque de todas as juntas e W é o fator de ponderação na função objetivo para cada junta i . Dessa forma a função 3.5, que é a parcela da função *fitness* que trata da minimização do deslocamento angular, pode ser reescrita conforme equação 3.18:

$$f3 = \sqrt{W_1(\theta_1 - \theta_{1i})^2 + W_2(\theta_2 - \theta_{2i})^2 + \dots + W_6(\theta_6 - \theta_{6i})^2}. \quad (3.18)$$

Entretanto, é necessário ainda contemplar a redução do dispêndio de energia considerando a sétima junta. Normalmente os motores das juntas que provocam o movimento lateral do manipulador são as que exigem mais torque, por carregarem todo o peso do robô. Assim sendo, o dispêndio de energia inerente do trabalho desse motor tem bastante relevância para o problema com 7DoF. O modo de cálculo segue a mesma regra, acrescentando um sétimo fator de ponderação à equação 3.15 que considera a minimização do deslocamento do sétimo eixo, conforme equação 3.19:

$$f3 = \sqrt{W_1(d_1 - d_{1i})^2 + W_2(\theta_2 - \theta_{2i})^2 + \dots + W_6(\theta_6 - \theta_{6i})^2 + W_7(\theta_7 - \theta_{7i})^2}. \quad (3.19)$$

3.5. Métodos de Simulação

O desenvolvimento no MatLab permite implementar de maneira adequada as soluções propostas para o problema da cinemática inversa. Entretanto, conforme ressaltado na seção 2.5, para confirmar a efetividade do método, é essencial realizar a simulação dos resultados. Para isso, a proposta é integrar o MatLab, onde foi desenvolvido o algoritmo genético, com ambientes de simulação virtuais de robótica. A ideia é realizar a simulação da solução para cinemática inversa do manipulador com 6 graus de liberdade através do RobotStudio, e utilizar o Coppelia-

Sim para simular a solução com 7DoF. Os códigos do MatLab para realizar as simulações estão expostas no Anexo C.

3.5.1. Simulação com o RobotStudio

Para simulação no RobotStudio é possível criar um *socket* de comunicação, para permitir a comunicação com clientes, como nesse caso, o MatLab. Através do *socket* é possível transmitir as variáveis de juntas, determinadas pelo algoritmo genético, ao módulo Rapid de programação, que interpreta as informações e comanda a execução de movimentos do robô. O software da ABB funciona como servidor dentro do *socket*, sendo o responsável por criar, abrir, aceitar e fechar as comunicações. Por sua vez, o MatLab funciona como um cliente, cujo função é apenas enviar e/ou receber informações do RobotStudio. O desenho esquemático dessa conexão é mostrado na Figura 3.2.

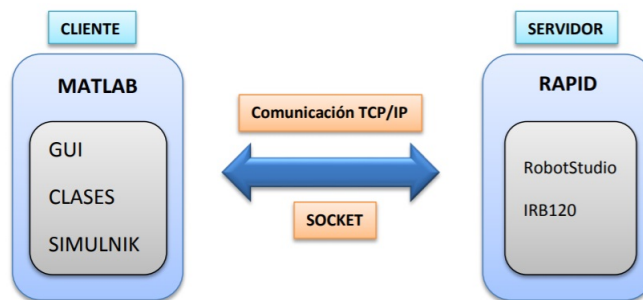


Figura 3.2: Esquema de integração MatLab / Robot Studio (CORBACHO, 2014).

A *procedure* (PROC) no módulo Rapid denominada “Conn” é responsável por gerar a conexão quando solicitada pelo cliente, utilizando a função “*SocketCreate*”, criando o *Socket* de comunicação. Na sequência são inseridas as informações de IP e porta através da função “*SocketBind*”, que para essa aplicação são o “127.0.0.1” e “55000”, respectivamente

A função “*SocketListen*” permite “ouvir” as requisições de conexão que posteriormente serão aceitas pela função “*SocketAccept*”. Uma vez que a conexão seja aceita uma variável booleana denominada “*connectionStatus*” assume valor verdadeiro para indicar que foi bem sucedida. Finalmente uma mensagem de “Conectado” é enviada ao MatLab, que confirmará a comunicabilidade entre os softwares.

É preciso também construir a rotina de comunicação dentro do executável do MatLab. Para isso é realizada uma chamada de função que cria uma instância “*tcpip*” denominada “*tc*”, onde são inseridos os mesmos parâmetros de IP, porta e nome do cliente estipulados no RobotStudio. Na sequência a função “*fopen*” solicita a abertura dessa conexão. Uma vez que essa seja bem sucedida a função “*fread*” lê a mensagem com a palavra “conectado” que o RobotStudio envia e plota essa mensagem na tela para visualização do usuário. Essa mensagem poderia ser escrita dentro do próprio MatLab, mas enviá-la do outro software garante que a conexão foi bem sucedida e está funcional.

Concluídas com sucesso todas essas etapas de conexão, a *procedure* “Conn” chama uma nova *procedure* denominada “DadosMovement” enquanto o valor da variável “ConnectionStatus” for verdadeiro. Esse recurso permite que o robô aceite comando de movimentos enquanto a conexão estiver ativa. A *procedure* “DadosMovement” é responsável por receber os dados oriundos do MatLab utilizando a função “SocketReceive”. Para tanto deve ser especificado o cliente que fornecerá as informações, nominar o pacote de dados recebidos, especificar o número de bytes a serem lidos e o tempo de espera para recebimento desses dados. Nesse projeto o MatLab é definido como cliente e denominado “client”, e o conjunto de informações recebidas recebe o nome de “dados”. Dentro do pacote “dados” o MatLab envia o vetor “D1” de 12 posições. Nesse vetor existem 6 posições que indicam as juntas calculadas pela cinemática inversa e 6 posições que trazem as variáveis “S1, S2, S3, S4, S5, S6” que indicam o sinal do valor da junta. Mais detalhadamente, as posições 1,3,5,7,9 e 11 do vetor carregam os valores de θ_1 até θ_6 respectivamente.

No MatLab, cada variável de junta é analisada por uma cláusula condicional, caso o valor da junta seja negativo a variável “S” assume valor 1, caso contrário, assume valor 2. Então as variáveis “S1” a “S6” são alocadas nas posições 0,2,4,6,8 e 10. Essa estratégia é necessária porque o RobotStudio não é capaz de interpretar valores do conjunto numérico real no pacote de dados transmitido. Por esse mesmo motivo, os valores das variáveis de junta são multiplicados por 10 no MatLab e posteriormente divididos por 10 no RobotStudio, dessa forma é possível interpretar valores não inteiros com precisão de uma casa decimal.

O vetor “dados” serve como entrada para a PROC “StringHandler” que é chamada dentro da própria PROC “DadosMovement”. A “StringHandler” recebe o pacote de dados e descompacta em um vetor dentro do RobotStudio. Através de um enlace *FOR* de passo 2 e de uma cláusula condicional *IF*, a *procedure* identifica se o valor da junta é positivo ou negativo e atribui os valores ao vetor do RobotStudio denominado “junta”. Nesse vetor estarão armazenados todos as variáveis de junta enviadas pelo MatLab, o que propiciará a movimentação do Robô.

Para realizar essa movimentação o RobotStudio faz uso da função “MoveAbsJ”. Essa função é utilizada para mover o robô a uma configuração definida por valores absolutos das juntas, levando o efetuador à pose desejada ao longo de um caminho não linear (ABB, 2018). O movimento ocorre conforme as curvas de velocidade e aceleração pré-definidas pelo fabricante. Essa função é chamada ao final da *procedure* “DadosMovement” logo após a atribuição dos valores ao vetor junta pela *procedure* “StringHandler”. Os códigos necessários para realizar essa integração entre MatLab e RobotStudio estão expostas nos Anexos C e D.

3.5.2. Interface Gráfica de Usuário (GUI)

Para efeitos de testes e simulação, foi desenvolvido um protótipo de interface com as funções básicas para interação com o RobotStudio. Dentre as funcionalidades criadas na interface estão: a conexão com o ambiente de simulação; campo para inserção das coordenadas car-

tesianas e ângulos de orientação; função de inserção de valores de ângulos para movimentação no espaço das juntas.

A Figura 3.3 mostra a interface criada onde é possível inserir o IP e a porta para criar a conexão, determinar coordenadas cartesianas e de orientação desejadas para simulação. É possível também utilizar o campo a direita para determinar de maneira direta as variáveis para movimentar o manipulador no espaço das juntas.

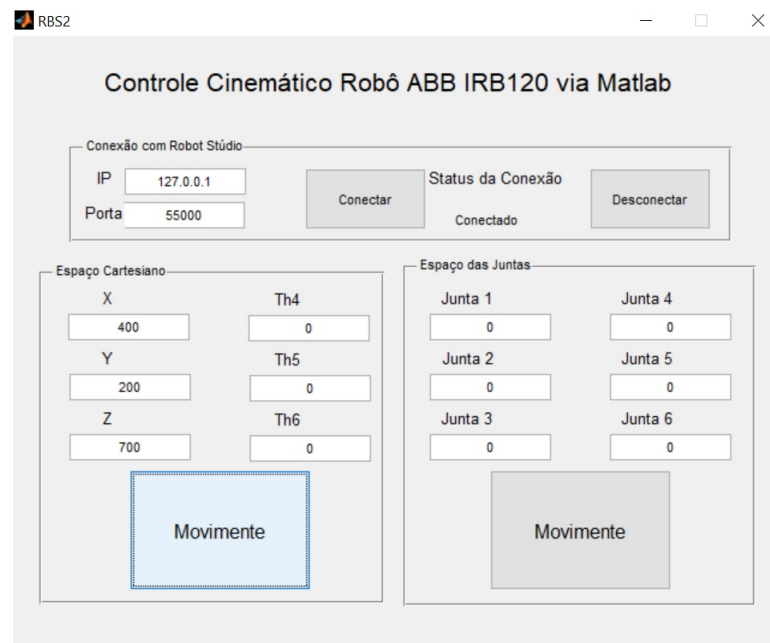


Figura 3.3: Interface gráfica do MatLab (GUI).

3.5.3. Simulação com o CoppeliaSim

Para simular a solução com sete graus de liberdade a proposta foi utilizar o ambiente de simulação virtual CoppeliaSim, que assim como descrito na seção 2.5 possui uma versão para estudantes com recursos mais vastos que a versão gratuita do RobotStudio. Nesse software é possível inserir diversos tipos de manipuladores de fabricantes variados, inclusive os modelos da ABB como o IRB120. Além disso é possível criar todo o ambiente com o qual o robô interage, criando peças, esteiras, obstáculos e até mesmo figuras humanas. Dessa forma, a proposta dessa dissertação foi agregar ao IRB120 um trilho que provoque um movimento lateral ao manipulador.

Apesar de esse software possuir uma linguagem de programação própria denominada LUA, ao contrário do RobotStudio, é possível realizar o controle do robô utilizando apenas o MatLab. Como demonstrado nesse capítulo o método proposto nesse trabalho é desenvolver o algoritmo genético utilizando os recursos do GAOT. Por esse motivo a integração entre o CoppeliaSim e o MatLab permitirá simular, com maior facilidade, os resultados da cinemática inversa para o manipulador considerando o sétimo grau de liberdade.

Para realizar essa integração basta acrescentar no main script da cena do CoppeliaSim uma única linha de código que determinará a porta de comunicação entre ambos os softwares. Para isso deve-se utilizar a função “simRemoteApi.start(1999)”, onde 1999 é a porta default utilizada para essa comunicação. A mesma definição do RemoteApi é utilizada no código do MatLab, onde é definida uma instância e então criado um cliente utilizando um IP definido como “127.0.0.1” e a porta de comunicação 1999. Dessa forma ao executar os códigos de ambos os softwares toda a informação gerada pelo cliente no MatLab será trafegada ao CoppeliaSim através do RemoteApi instanciado.

A API remota faz parte da estrutura da API CoppeliaSim. Essa funcionalidade permite a comunicação entre o CoppeliaSim e um aplicativo externo, é multiplataforma e suporta chamadas de serviço e fluxo de dados bidirecional (COPPELIA, 2018b). Existe uma API específica para cada tipo de software com o qual se deseja integrar o Coppelia, inclusive para o MatLab. Nessa API são disponibilizadas todas as funções suportadas para a integração entre os softwares. Dentre as principais, que são relevantes para esse trabalho, estão as funções de identificação de objeto, e as funções de determinação de posição e velocidade para as juntas.

A cena criada no CoppeliaSim é definida como uma estrutura hierárquica, onde todos os objetos estão dentro da hierarquia da própria cena, e alguns objetos podem estar sob hierarquia de outros. Para esse caso, o trilha é definido como o objeto principal dentro da cena, e dentro da hierarquia do trilha está colocada a junta prismática que proporcionará a translação lateral ao robô. Porém a junta não é tida pelo software como um elemento físico, por esse motivo dentro da hierarquia da junta é colocada um objeto denominada “Mesa”, que sustentará a estrutura física do robô. Dessa forma, todo o movimento definido para a junta prismática será revertido para a plataforma física. O manipulador IRB120 por sua vez está colocado dentro da hierarquia da base onde está suportado. Assim sendo, ao locomover a base, o robô também se locomoverá. Por fim, dentro da hierarquia do robô estão colocadas as 6 juntas rotacionais que o constitui e associado a cada junta está um elo físico que somados correspondem à estrutura do manipulador. A Figura 3.4 demonstra a cena construída no CoppeliaSim.

Cada uma das sete juntas citadas são referenciadas como um objeto no MatLab utilizando a função “*simxGetObjectHandle*”. Para isso basta definir o cliente ao qual a aplicação está conectada, especificar o nome da junta através de uma string que deve ser coincidente com o determinado no Coppelia e definir o modo de operação. Dessa forma, é possível enviar comandos através da RemoteApi para cada junta a partir do MatLab. Nessa dissertação serão considerados dois tipos de controle para o manipulador, o controle de posição das juntas e o controle de velocidade.

Controle de Posição

Nesse caso o objetivo é controlar a posição de cada junta que determinará ao robô a pose desejada. No método, os genes da solução do algoritmo genético, que são seis variáveis de juntas rotacionais e uma prismática, são trafegados diretamente ao CoppeliaSim, fazendo

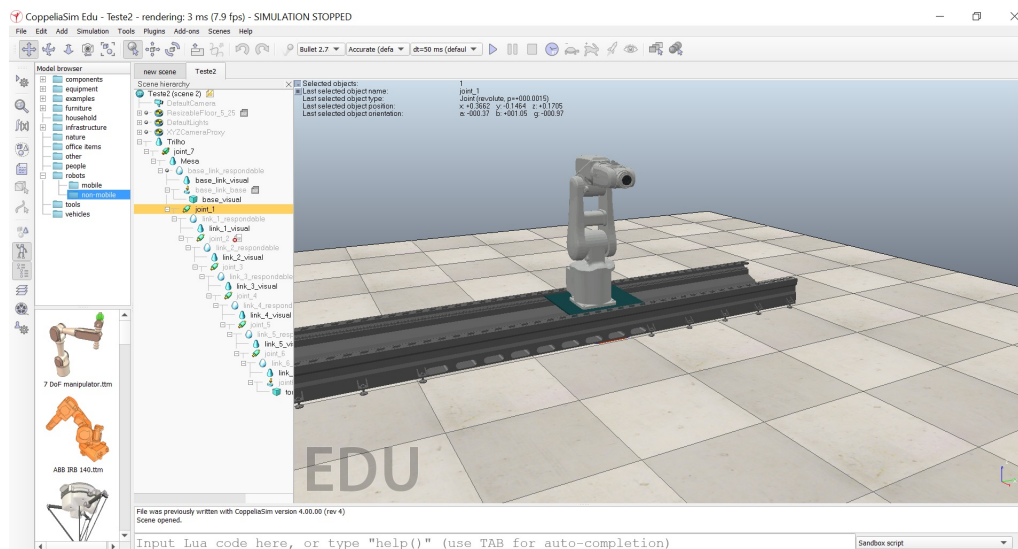


Figura 3.4: Cena do CoppeliaSim com o IRB120 sob o trilho.

com o robô se movimenta até a pose determinada pelo AG utilizando a velocidade padrão do simulador do Coppelia. Para isso é utilizada a função “*simXSetJointTargetPosition*” no MatLab, que estipula aos objetos criados para cada uma das 7 juntas, as variáveis resultantes do algoritmo genético.

Para o controle de posição, como é estipulada apenas a posição final de cada junta, é preciso habilitar um controlador em suas propriedades dinâmicas para que essa consiga executar a trajetória até a posição estipulada. O CoppeliaSim oferece um controlador PID nativo que pode ser habilitado nas propriedades de controle das juntas para viabilizar esse tipo de simulação. Os códigos desenvolvidos no MatLab que permitem esse controle da posição considerando os 7 graus de liberdade são mostrados no Anexo C.

Controle de Velocidade

Conforme demonstrado nas seções 2.3.8 e 3.2, uma das maneiras de realizar o controle da trajetória é utilizando um polinômio de quinta ordem. Esse polinômio determina valores de velocidade para a junta ao longo do tempo t_f definido para a trajetória conforme equação 2.24. Uma das opções que o Coppelia oferece é controlar a velocidade de cada junta através do MatLab utilizando a função “*simxSetJointTargetVelocity*”.

Nesse caso, ao contrário do controle de posição, é preciso desabilitar o PID interno, de forma que todo o controle da trajetória seja feito através das velocidades das juntas determinadas pelo MatLab. Importante ressaltar que, como nos pontos inicial e final a velocidade é igual a zero, e o loop de controle interno não é realizado, é necessário alterar as propriedades dinâmicas das juntas para os movimentos dos motores sejam freados durante o estado de repouso e o manipulador não ceda à gravidade no início e no fim da trajetória.

4. Resultados

Esse capítulo expõe os resultados coletados após a implementação dos métodos propostos, que serão apresentados da seguinte forma: as seções 4.1 e 4.2 tratarão da investigação dos algoritmos genéticos, demonstrando os resultados para cada uma das variações propostas na seção 3.1.6. A seção 4.3 ampliará a análise demonstrando os resultados considerando a inserção do sétimo eixo. Os ganhos de eficiência energética através da ponderação da função objetivo serão expostos na seção 4.4. Na sequência a seção 4.5 abordará sobre os resultados relativos ao algoritmo de planejamento de trajetória através do controle de velocidade das juntas desenvolvido no MatLab. A seção 4.6 apresentará as simulações realizadas para constatação da eficácia das soluções propostas, conformes métodos apresentados na seção 3.5. Por fim, a seção 4.7 replicará alguns dos testes e simulações apresentadas no capítulo para o manipulador de padrão industrial IRB6700.

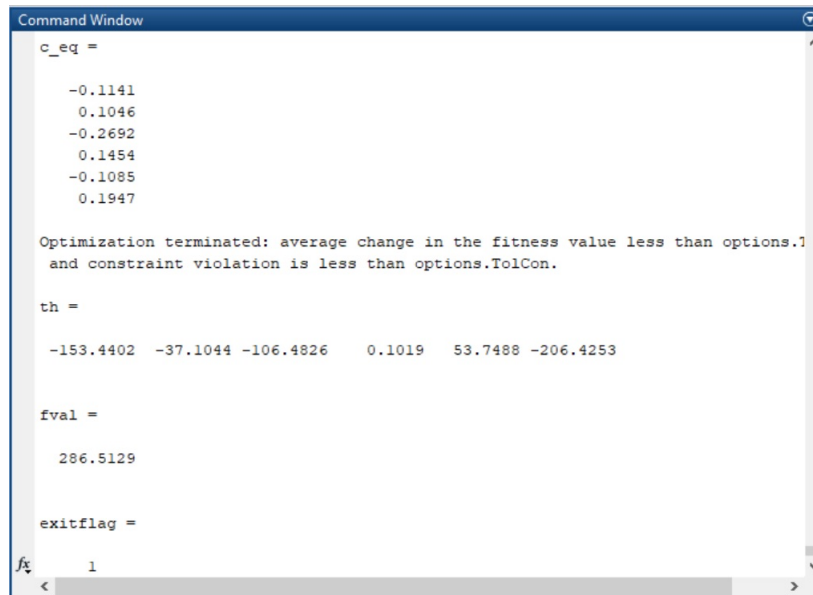
4.1. Algoritmos Genéticos para Resolução da Cinemática Inversa de 6DoF

Os primeiros resultados apresentados nessa dissertação são relativos aos testes realizados com o algoritmo genético para a solução da cinemática inversa para o robô de seis graus de liberdade. Esses resultados desconsideram as simulações, que serão apresentadas mais adiante nesse capítulo. O intuito principal foi investigar a eficácia do algoritmo para encontrar a solução do problema. Para isso, através da interface de usuário desenvolvida no MatLab, foi definido um ponto aleatório com coordenada cartesiana em x igual a 400mm, em y igual 200mm e em z igual a 700mm, dentro do espaço de trabalho do robô, considerando todos os ângulos de orientação iguais a 0°. A configuração inicial do robô foi a de Home, onde todas as variáveis de junta são iguais a 0. Os parâmetros da configuração do AG são expostos na Tabela 4.1, e o resultado alcançado é mostrado na janela de comando do MatLab conforme exposto na Figura 4.1.

Tabela 4.1: Configurações do AG1.

População Inicial	30
Número de Gerações	60
Método de seleção	Torneio
Crossover	Heurístico
Taxa de crossover	0,5
Mutação	AdaptFeasible

Analisando o resultado exposto na janela de comando é possível afirmar que o algoritmo interrompeu a execução ao encontrar o valor ótimo, uma vez que a variável “exitflag” assumiu valor igual 1. Nesse caso, o vetor “th” representa a solução ótima da cinemática inversa para o manipulador de 6 graus de liberdade. Vale ressaltar que os resultados para as seis juntas estão



```
Command Window
c_eq =
    -0.1141
     0.1046
    -0.2692
     0.1454
    -0.1085
     0.1947

Optimization terminated: average change in the fitness value less than options.TolFun
and constraint violation is less than options.TolCon.

th =
   -153.4402   -37.1044  -106.4826    0.1019    53.7488  -206.4253

fval =
    286.5129

exitflag =
    1
```

Figura 4.1: Resultado do algoritmo genético executado no MatLab.

dentro do limite indicado na tabela 2.2. Já a variável “fval” representa o valor final alcançado pelo função de *fitness*, que para essa dissertação, é meramente informativo, conforme discutido na seção 3.1.4. Por fim, o vetor “ceq” indica os erros de posição e orientação da solução, que por sua vez estão dentro das margens de erro estipuladas nas funções de restrição. Dessa forma é possível afirmar que a metodologia foi correta e o algoritmo desenvolvido teve o comportamento esperado.

Entretanto, a partir dessa primeira análise simples não é possível extrair conclusões sobre a eficiência do método; isso porque, apesar de ter chegado a um valor ótimo nessa execução, a repetição das simulações podem apresentar fragilidades do algoritmo. Esse, por exemplo, pode se mostrar pouco robusto para escapar de ótimos locais, não conseguindo atender as restrições, e por consequência o resultado final estar longe da pose desejada. Ou ainda, o algoritmo pode levar um longo tempo de execução para alcançar o valor de ótimo global e/ou consumir muitos recursos computacionais. Por esse motivo é necessária uma investigação mais detalhada, utilizando técnicas estatísticas, para melhor interpretar a eficácia e robustez do método proposto.

4.2. Investigação dos Algoritmos Genéticos

O objetivo do algoritmo desenvolvido é minimizar o deslocamento angular com margens de erro, de posicionamento e orientação, aceitáveis. Para identificar a melhor solução são analisados três fatores fundamentais: o tempo de execução do algoritmo até atingir o valor ótimo, o erro de médio de posição e o erro máximo dos ângulos de orientação, conforme equações de distância euclidiana. Além desses três, são avaliados também os critérios de parada, indicando as interrupções em valores ótimos com atendimento as restrições, por mínimos locais

irreversíveis ou por limite de gerações.

Na análise são considerados os efeitos da população inicial e do número máximo de gerações. Entendendo indivíduos como “i”, e gerações como “g”, foram estipuladas as seguintes variações: 10i25g, 25i25g, 25i40g, 40i40g e 40i55g. Para cada combinação foram coletadas 8 amostras de resultados. Além disso, foram analisadas variações na configuração do AG, como método de seleção, *crossover* e sua taxa, mutação, *population overlapping*, elitismo e processamento de populações em paralelo. É importante citar que todas as análises foram feitas para o mesmo ponto, uma vez que como o problema busca minimizar o deslocamento angular, pontos diferentes podem interferir nos resultados comparativos. O ponto utilizado, é o mesmo do teste anterior, e tem coordenadas cartesianas x, y e z iguais a 400mm, 200mm e 700mm e os ângulos de orientação são iguais a 0°. Para todos os casos foi considerado como configuração inicial a posição de *Home*, onde todas as variáveis de juntas são iguais a zero.

Efeitos da População Inicial e Número de Gerações

A primeira análise foi realizada considerando somente o efeito do tamanho da população inicial e o número máximo de gerações sob o resultado da otimização. As configurações do algoritmo para a primeira simulação (AG1) são descritas na Tabela 4.2. A Fig. 4.2, demonstra os resultados.

Tabela 4.2: Configurações do AG1.

Método de seleção	Torneio
<i>Crossover</i>	Heurístico
Taxa de <i>crossover</i>	0,5
Mutação	AdaptFeasible

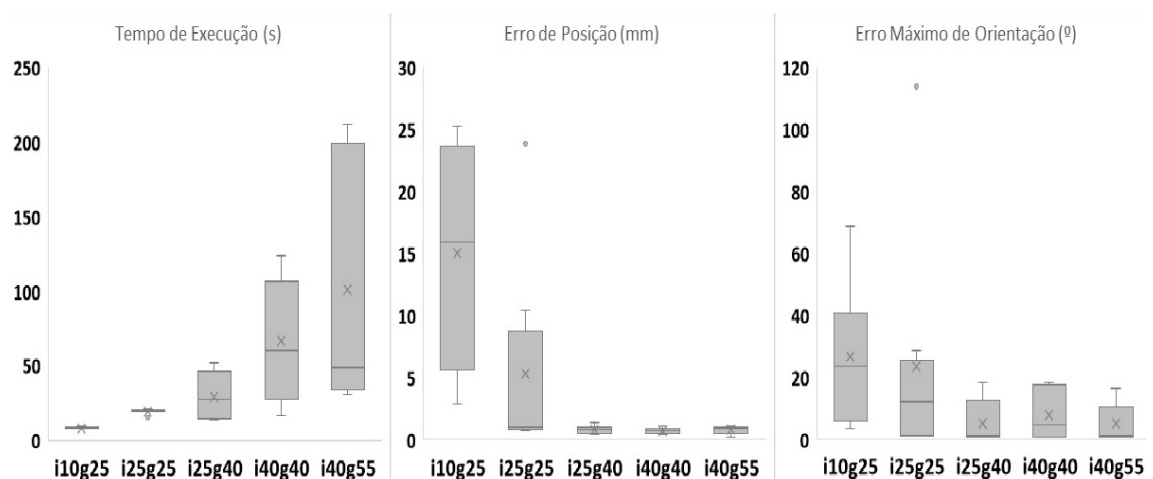


Figura 4.2: Efeitos da população inicial e do número de gerações.

É possível perceber que para os números menores de indivíduos e gerações o algoritmo converge rapidamente para a região da solução ótima, entretanto não há gerações o suficiente

para encontrar o ótimo global. Em consequência, as restrições não são atendidas, e os erros de posição e orientação são grandes. Em contrapartida, ao aumentar o número de indivíduos e gerações o tempo se eleva, mas os erros de posição e orientação alcançam níveis satisfatórios. Existe ainda o problema no qual o algoritmo fica preso em mínimos locais, sem alcançar um valor de ótimo global que atenda as restrições. Nas 40 amostras expostas nesses gráficos 32,5% foram interrompidas ao atingir o limite de gerações, 27,5% ficaram presas em mínimos locais e 40% convergiram a um valor de ótimo global. Para o caso de 40 indivíduos e 55 gerações o tempo médio dos testes que convergiram foi de 40s. Esse tempo foi considerado elevado pensando em uma possível aplicação que exija uma tomada de decisão em tempo real. Por esse motivo a opção foi por não aumentar mais o número de indivíduos e gerações.

Efeitos do Elitismo e Sobreposição de Populações

Para a sequência dos testes foram mantidas todas as configurações do AG anterior e acrescentado uma taxa de 25% de sobreposição de população e elitismo para 2 indivíduos. A Tabela 4.3 expõe a configuração do algoritmo para a segunda simulação (AG2) e os gráficos da Fig. 4.3 mostram os resultados alcançados.

Tabela 4.3: Configurações do AG2.

Método de seleção	Torneio
Crossover	Heurístico
Taxa de crossover	0,5
Mutação	AdaptFeasible
Sobreposição de População	0,25
Elitismo	2

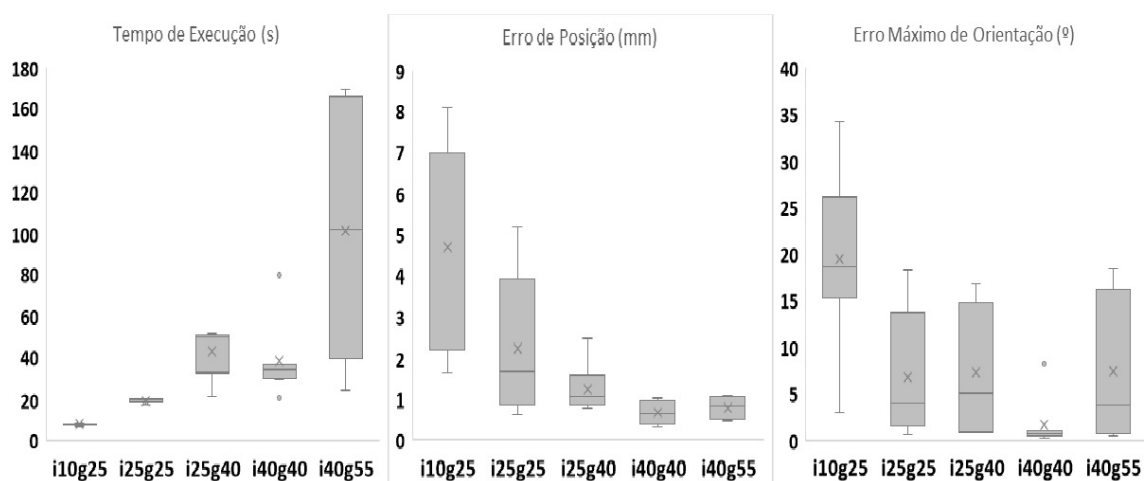


Figura 4.3: Efeitos do elitismo e da sobreposição de populações.

Para esse caso o aumento do tempo e minimização do erro conforme a elevação do número de indivíduos e gerações se manteve. Mas diferente do que foi observado para o teste

anterior, os erros de posição e de orientação foram menores para menos indivíduos e gerações. Mas ainda que os erros tenham sido menores nos testes com poucas gerações, estes continuaram sendo interrompidos sem encontrar o ótimo global. O ponto negativo foi que para os indivíduos que convergiram ao ótimo global o tempo de execução foram significativamente maiores. Este fator pode ser prejudicial para aplicações que necessitem de uma tomada de decisão mais ágil. Das 40 simulações apresentadas nessa seção 37,5% foram interrompidas ao atingir o limite de gerações, 25% ficaram presas em mínimos locais e 37,5% convergiram para um valor de ótimo global.

Foi avaliada ainda uma variação da taxa de *crossover* para 0.8 (AG3), mantendo as demais configurações, conforme indicado na Tabela 4.4. Essa variação não trouxe melhoras aos resultados. Não houve redução dos níveis de erro e os tempos de execução foram elevados. Apenas 35% dos testes convergiram a um valor de ótimo global, ao passo que 27,5% ficaram presos em ótimos locais e 37,5% foram interrompidos pelo limite de gerações. Os resultados são apresentados nos gráficos da Figura 4.4.

Tabela 4.4: Configurações do AG3.

Método de seleção	Torneio
<i>Crossover</i>	Heurístico
Taxa de <i>crossover</i>	0,8
Mutação	AdaptFeasible
Sobreposição de População	0,25
Elitismo	2

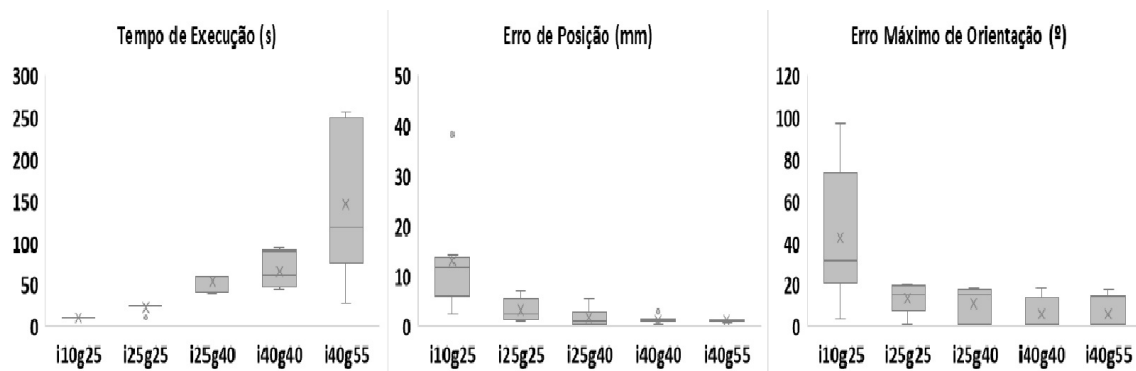


Figura 4.4: Efeitos da variação da taxa de crossover

Efeito de Populações em Paralelo

Foram considerados também os efeitos do processamento de populações em paralelo. Nesse caso foram retiradas as opções de elitismo e de sobreposição de população, mas mantidas todas as demais configurações. A Tabela 4.5 mostra as configurações desta simulação (AG4) e a Fig. 4.5 mostra os gráficos de tempo e erros, para processamento de populações em paralelo.

Tabela 4.5: Configurações do AG4.

Método de seleção	Torneio
Crossover	Heurístico
Taxa de crossover	0,5
Mutação	AdaptFeasible
Populações em Paralelo	Sim

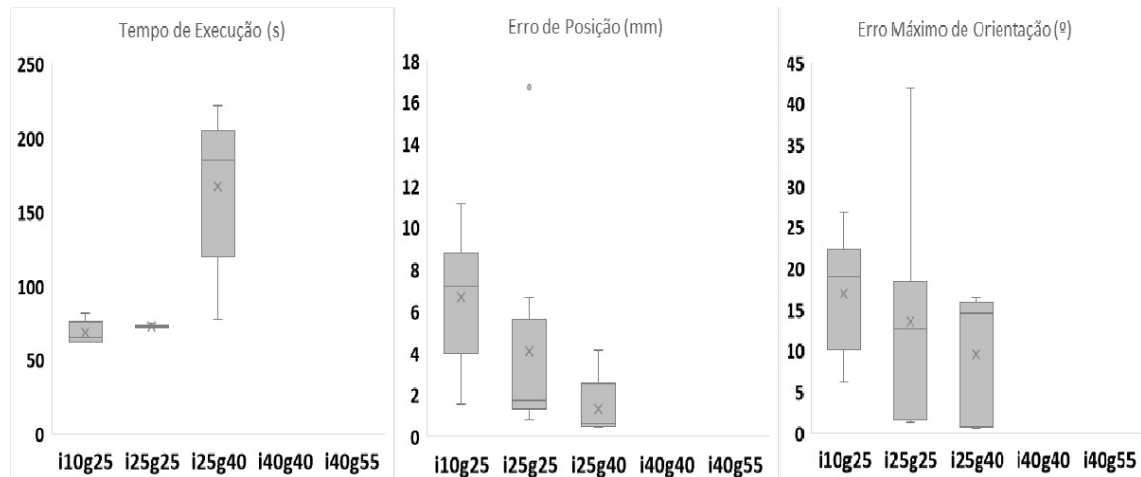


Figura 4.5: Efeitos do processamento de população em paralelo.

Foi possível perceber que os tempos de execução aumentaram de maneira expressiva, sem que isso significasse uma melhora dos índices de erros. O tempo médio dos testes com 25 indivíduos e 40 gerações foi de 166s e apenas 2 destes convergiram ao valor de ótimo global. Portanto a análise foi interrompida precocemente e já considerada inviável.

Efeito dos Métodos de Seleção, *crossover* e Mutação

Foi investigado o comportamento do algoritmo genético quando variado o método de seleção, mutação e *crossover*. Para a quarta simulação (AG5) os três parâmetros foram alterados, mantendo a taxa de *crossover*, a sobreposição de população e o elitismo conforme a Tabela 4.6. Os gráficos das Fig. 4.6 mostram os resultados dessa variação.

Tabela 4.6: Configurações do AG5.

Método de seleção	Roleta
Crossover	Aritmético
Taxa de crossover	0,5
Mutação	Gaussiana
Sobreposição de População	0,2
Elitismo	2

Nessa hipótese foi possível notar uma melhora na taxa de convergência para valores ótimos. Apesar de 47,5% dos algoritmos terem sido interrompidos pelo limite de gerações,

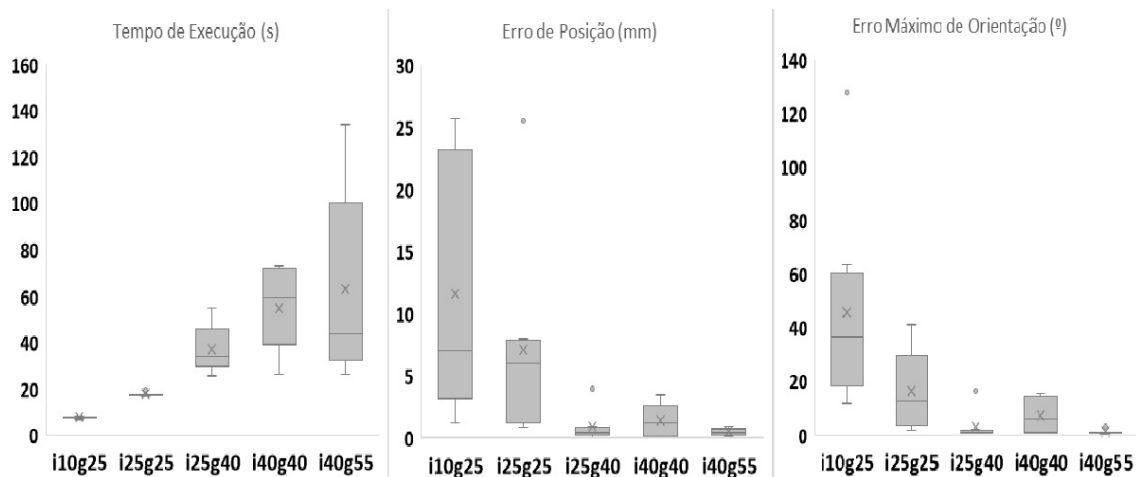


Figura 4.6: Efeito dos métodos de seleção, *crossover* e mutação.

45% deles convergiram a um valor de ótimo global. Consequentemente apenas 7,5% ficaram presos em valores de ótimo local. Portanto essa configuração demonstra maior robustez para escapar de ótimos locais.

Outra configuração analisada foi considerando o método de seleção e mutação uniformes e o método de *crossover* como Single Point (AG6), conforme Tabela 4.7. Entretanto essa configuração teve o pior desempenho entre os testes. Isso porque, para qualquer número de indivíduos e/ou gerações, o algoritmo ficou preso em valores de ótimo local. Resultado que pode ser interpretado através do gráfico da Figura 4.7, onde nota-se que apesar dos tempos de processamento relativamente baixos em relação ao número de gerações, o erro sempre se manteve em patamares altos, indicando que o algoritmo não convergiu ao ótimo global.

Tabela 4.7: Configurações do AG6.

Método de seleção	Uniforme
<i>Crossover</i>	Single Point
Taxa de <i>crossover</i>	0,8
Mutação	Uniforme
Sobreposição de População	0,2
Elitismo	2

Efeito do Doping da População Inicial

Considerando que normalmente um robô antropomórfico não utiliza todo seu espaço de trabalho e que possivelmente as principais áreas de trabalho são conhecidas, foi utilizada uma estratégia de dopar a população inicial com um indivíduo conhecido, dentro de um espaço de trabalho restrito. Ou seja, o primeiro indivíduo da população inicial foi alterado de forma que as variáveis de junta fossem propositalmente mais próximas às da pose do robô no ponto desejado. Para realizar essa simulação (AG7) foram consideradas as melhores configurações

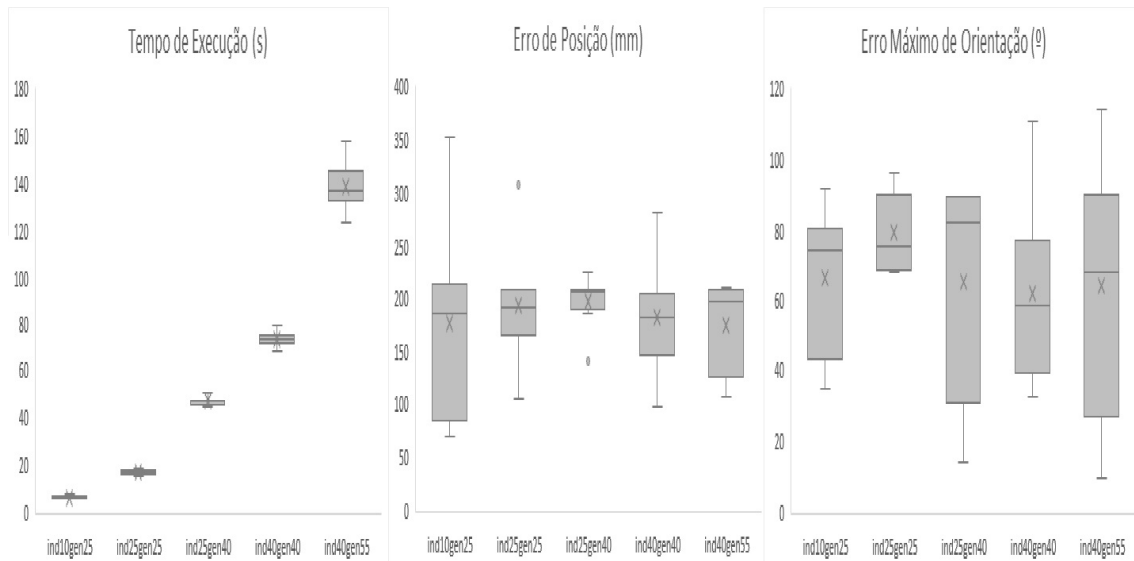


Figura 4.7: Efeito dos métodos de seleção, *crossover* e mutação.

avaliadas nos testes anteriores, conforme Tabela 4.8. Os resultados estão demonstrados nos gráficos da Fig. 4.8.

Tabela 4.8: Configurações do AG7.

Método de seleção	Roleta
<i>Crossover</i>	Aritmético
Taxa de <i>crossover</i>	0,5
Mutação	Gaussiana
Sobreposição de População	0,2
Elitismo	2

Os resultados alcançados com uma população inicial dopada foram excelentes. Em 100% dos testes houve convergência para um valor de ótimo global, em tempos significativamente baixos e com valores de erros dentro do limite aceito. Para populações com 10 indivíduos o algoritmo convergiu em um tempo médio de 5s, com erros atingindo a ordem de 0,12mm para posição e 0,62° para orientação. O melhor tempo de evolução foi de 3,57s.

4.3. Algoritmos Genéticos para Resolução da Cinemática Inversa de 7DoF

Conforme proposto na seção 3.3, o algoritmo genético foi adaptado para resolver o problema da cinemática inversa considerando uma junta prismática provocando uma translação lateral ao manipulador como um sétimo grau de liberdade. Para realizar esse teste foram consideradas as configurações que demonstraram os melhores resultados observados na seção anterior conforme Tabela 4.8. Para o primeiro teste foi desconsiderado os efeitos do doping da

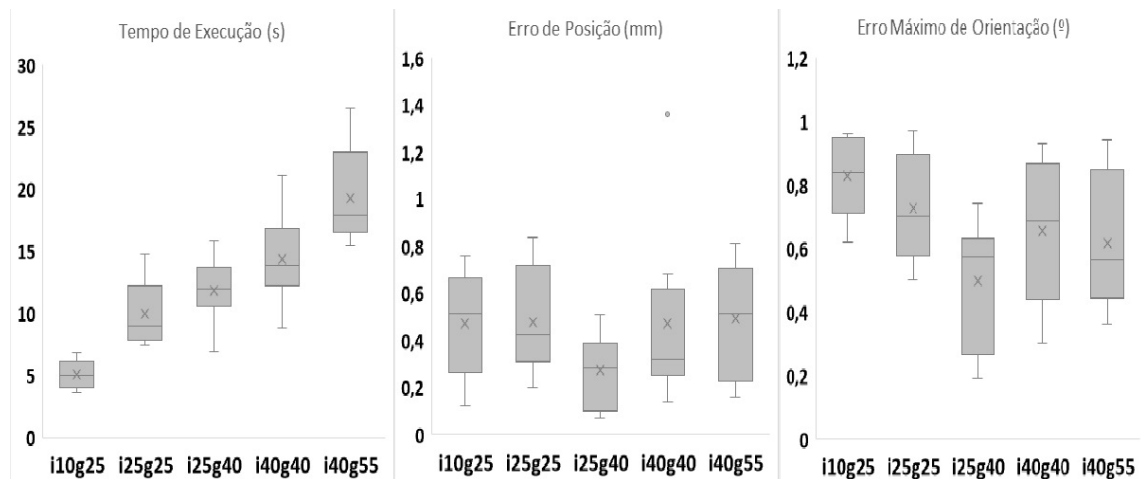


Figura 4.8: Efeitos do doping da população inicial.

população. Os pontos cartesianos utilizados para a simulação permaneceram os mesmo da solução de 6DoF, sendo x, y e z iguais a 400mm, 200mm e 700mm, assim como os ângulos de orientação desejados também foram todos iguais a 0. O intuito de manter as mesmas coordenadas foi comparar a diferença entre as poses finais alcançadas ao se acrescentar o sétimo eixo. A Figura 4.9 expõe o resultado da execução do algoritmo.

```

Command Window
400.0166 199.8198 700.0150 0.4675 0.1208 -0.2065

Optimization terminated: average change in the fitness value less than options.TolFun
and constraint violation is less than options.TolCon.

th =
-24.3198 33.8637 -42.3533 -0.2407 -81.4874 24.8230 380.2722

fval =
104.5741

exitflag =
1

TH =
-0.4245 0.5910 -0.7392 -0.0042 -1.4222 0.4332 380.2722

Elapsed time is 25.430639 seconds.
fx >>

```

Figura 4.9: Resultado do algoritmo para cinemática inversa de um manipulador com 7DoF.

Ao analisar o resultado a variável “exitflag” novamente foi igual a 1, o que comprova que a solução ótima foi encontrada. Mas para esse caso o vetor “th” é composto por sete elementos, sendo que o sétimo fator assume o valor 380, o que indica o deslocamento lateral em mm que é proporcionado ao robô. Esse resultado é coerente porque apesar dos limites do trilho terem sido estipulados em 5000mm, o IRB120 é um manipulador pequeno, com um alcance curto, e um deslocamento lateral muito elevado pode fazer com que o ponto que se deseja alcançar com o efetuador esteja fora do espaço de trabalho do robô.

Outra informação contida na Figura 4.9 é que o algoritmo demorou mais de 25s para convergir ao valor ótimo. A estratégia de dopagem da população inicial foi replicada para a

solução com 7DoF, alterando a população inicial com um indivíduo conhecidamente próximo a pose desejada. Foi inserido o indivíduo com as variáveis de juntas rotacionais 30°, 40°, -55°, 10°, -90° e -40° e variável de junta prismática de 50 mm. Os resultados são mostrados na Figura 4.10.

```

Command Window
0.2405

resultado =

    400.1531    199.9008    700.1566   -0.6148   -0.2122    0.9964

Optimization terminated: average change in the fitness value less than options.TolFun
and constraint violation is less than options.TolCon.

th =

    19.2102    29.1229   -35.3416    0.8697   -84.3196   -19.9133    61.6243

fval =

    118.7796

exitflag =

     1

Elapsed time is 7.186122 seconds.
f3 >>

```

Figura 4.10: Resultado do algoritmo, com população inicial dopada, para cinemática inversa de um manipulador com 7DoF.

O indivíduo dado como solução de fato é parecido com aquele utilizado para dopar a população inicial, atendendo as condições de restrição, com margens de erro aceitáveis. Mas o resultado mais expressivo observado foi que nesse teste o indivíduo convergiu ao ótimo global em 7,18s valor 3,5 vezes menor que a solução atingida com a solução não dopada. Isso deixa claro a principal vantagem desse método, que além de ser robusta para escapar de ótimos locais, torna a convergência ao resultado ótimo muito mais rápida.

4.4. Função Objetivo Ponderada

Conforme exposto na seção 3.4, ao ponderar os fatores da função de minimização do deslocamento angular é possível diminuir o dispêndio de energia para que o robô alcance a pose desejada. Para os manipuladores antropomórficos da ABB as juntas 1, 2 e 3, responsáveis pelo posicionamento do robô possuem uma quantidade de torque mais elevado que as juntas de orientação. Nesse caso, a expectativa é que, ao ponderar a equação, o deslocamento de cada junta seja reduzido quão maior for o torque do seus motores.

Para realizar os teste, a função f_3 que compõe a função multiobjetivo do algoritmo genético foi alterada conforme equação 3.18. Ao aplicar os valores de torque da tabela 2.3 nas equações 3.16 e 3.17 os fatores de ponderação encontrados foram 0,401 para as juntas 1 e 2; 0,175 para a junta 3; 0,01 para as juntas 4 e 5; e 0,005 para a junta 6. A fim de coletar resultados comparativos o algoritmo foi executado três vezes desconsiderando os fatores de ponderação, e outras três com os fatores inclusos. Novamente foram considerados os pontos cartesianos

e orientação utilizados nos testes anteriores. Os gráficos das Figuras 4.11 e 4.12 mostram o deslocamento total de cada junta para ambos os casos.

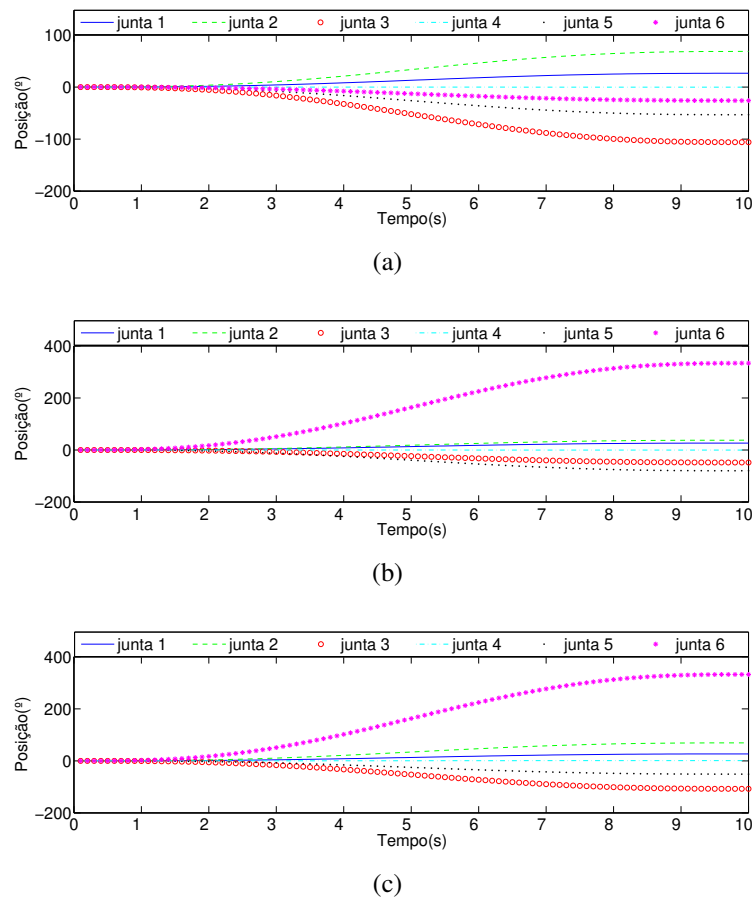
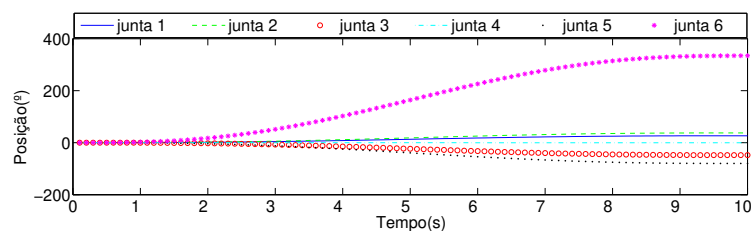


Figura 4.11: Simulações sem ponderação da função objetivo.

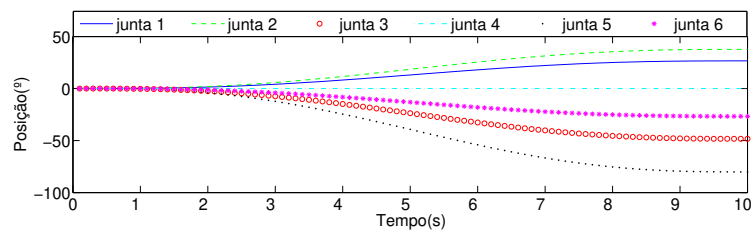
Como pode ser observado nas figuras, ao ponderar a função objetivo o deslocamento das juntas de posicionamento foram de fato menores. Apesar de o resultado ser demonstrado no gráfico da Figura 4.11(b) ser tão bom quanto os resultados com a equação ponderada, os outros dois casos mostram um deslocamento alto das juntas de posicionamento. Essa diferença entre resultados condiz com o comportamento esperado por se tratar de um método heurístico.

Para o caso da função objetivo ponderada, em nenhuma das três simulações as juntas 1, 2 e 3 se locomoveram mais do que 50° , ao passo que no pior caso da função não ponderada essas mesmas juntas se locomoveram em patamares próximos a 100° . Outra constatação desse resultado é que nos resultados com a função ponderada, a junta 3 sempre se locomove mais do que as juntas 1 e 2. Esse fato comprova a efetividade da solução uma vez que a junta com menor torque se locomoveu mais do que as juntas de maior torque.

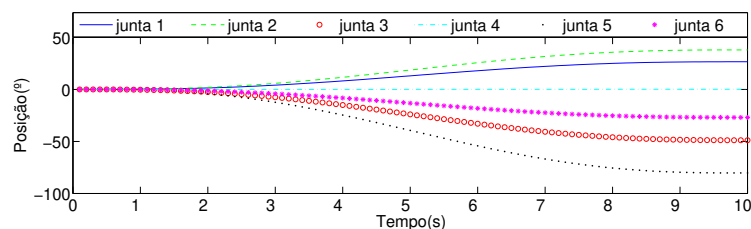
Esses resultados demonstram a eficácia da solução para o manipulador de 6DoF, entretanto, conforme tratado na seção 3.4, é necessário considerar a minimização do gasto energético da junta prismática que locomove todo o manipulador. Como o objetivo dessa dissertação não é tratar de nenhum processo robotizado em específico, apesar do capítulo 5 tratar de um exem-



(a)



(b)



(c)

Figura 4.12: Simulações com função objetivo ponderada.

plano prático em que a solução pode ser aplicada, não há nenhum motor utilizado como eixo da base para tomar como referência de torque. Por esse motivo foi colocado como premissa para realização das simulações que o motor que locomove a junta prismática tem torque 1,5 vezes maior que os motores de maior torque do manipulador, que são os das juntas 1 e 2. O valor hipotético estipulado foi de 292,5 Nm e pode ser perfeitamente ajustado caso exista algum estudo sobre o valor de referência real. Dessa forma os fatores de ponderação de $W_1, W_2 \dots W_7$ foram de 0,375; 0,25; 0,25; 0,109; 0,06; 0,06 e 0,03 respectivamente. Esses valores foram aplicados à equação 3.19, para determinar a função objetivo ponderada. Novamente foi realizada a comparação entre a função objetivo não ponderada e a função objetivo ponderada para evidenciar o ganho inerente a essa solução. Os resultados são mostrados nos gráficos das Figuras 4.13 e 4.14.

Como pode ser percebido através dos gráficos, para o caso em que a função objetivo não foi ponderada, o deslocamento da junta 1 foi 9 vezes maior que o caso em que houve a ponderação. As demais juntas também apresentaram movimentos angulares reduzidos conforme a quantidade de torque que aplicam, mas como a junta 1 é prismática e seu deslocamento é medido em mm, não há razões para realizar comparações no mesmo gráfico das outras juntas uma vez que o deslocamento dessas é medido em grandeza angular.

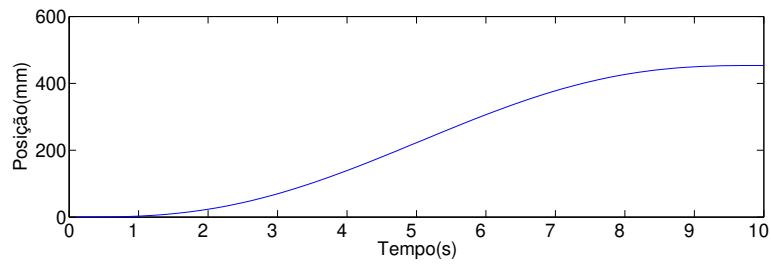


Figura 4.13: Deslocamento do primeiro eixo com função objetivo não ponderada.

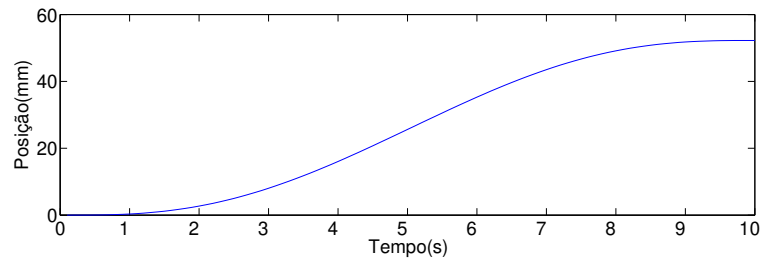


Figura 4.14: Deslocamento do primeiro eixo com função objetivo ponderada.

4.5. Planejamento de Trajetória no MatLab

Na seção 3.2, foi apresentada uma estratégia para planejamento de trajetórias através do controle de velocidade das juntas. Essa estratégia foi implementada no MatLab conforme a metodologia proposta. Dessa forma ao ser determinada a pose final otimizada conforme apresentado nas seções anteriores, e considerando a posição inicial a de “Home”, as variáveis de juntas foram tomadas como entrada para o planejamento de trajetória através do polinômio de quinta ordem. As curvas de posição, velocidade e aceleração das 6 juntas são apresentados na Figura 4.15.

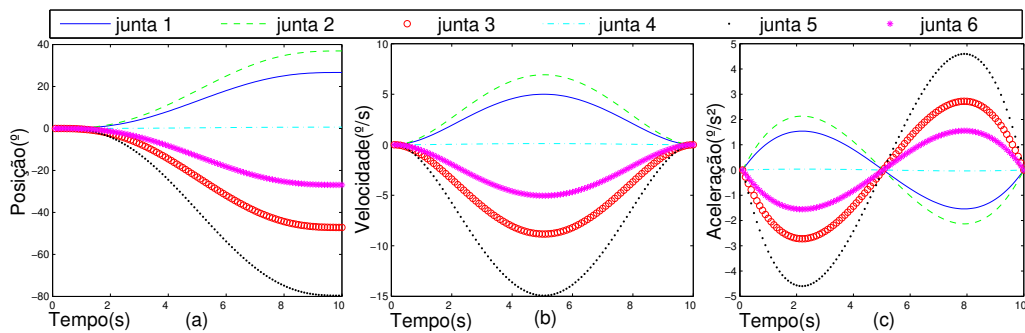


Figura 4.15: (a) Posições, (b) velocidades e (c) acelerações das juntas durante a trajetória.

De fato as curvas encontradas coincidem com aquelas propostas por Spong, demonstradas na Figura 2.2, o que mostra que a implantação do algoritmo para o planejamento da trajetória foi eficaz. Como pode ser observado na figura, as 6 juntas atingiram a posição desejada, com velocidades bem definidas durante a trajetória, mas sobretudo suavizando a curva de aceleração. Fica evidenciado que esse tipo de planejamento de trajetória incorpora restrições de

acelerações, evitando dessa forma variações instantâneas de comando das juntas, provocando movimentos bruscos do robô. Controlar a aceleração torna os movimentos mais suaves, portanto factíveis para aplicação em um robô real.

4.6. Simulações

Implementados todos os algoritmos propostos no MatLab, o próximo passo da proposta é simular os resultados em um ambiente virtual para comprovar sua eficácia. As simulações foram divididas em duas etapas distintas: a primeira relativa à simulação da cinemática inversa para o manipulador de 6DoF utilizando o RobotStudio. Na segunda etapa foram simulados os resultados do AG, para resolução da cinemática inversa para o problema com 7DoF, no CoppeliaSim. Utilizando esse mesmo software foi realizada a simulação do controle de velocidade das juntas para planejamento de trajetória.

4.6.1. Simulações no RobotStudio

Para verificar a efetividade do método de computação evolutiva para resolver a cinemática inversa, foi utilizada as configurações do melhor resultado exposto na seção 4.2, o AG7, integrado ao ambiente virtual do RobotStudio. Entretanto, buscando demonstrar que o método estocástico é capaz de encontrar todas as possíveis soluções do problema, foi desconsiderado o efeito do doping da população inicial. Caso contrário, o indivíduo propositalmente inserido interferiria no resultado, levando sempre à mesma configuração das juntas.

Para simular foram inseridos, através da interface gráfica, os pontos cartesianos $x = 200mm$, $y = 350mm$ e $z = 750mm$ e os ângulos de orientação desejados Φ_d , Θ_d e Ψ_d como 30° , 45° e 60° respectivamente. Essas coordenadas foram inseridas também no ambiente virtual de simulação do RobotStudio, que cria um frame na posição desejada com a devida orientação. Dessa forma é possível comprovar que as variáveis de juntas determinadas pelo AG levaram o robô à pose desejada. A Fig. 4.16 mostra as diferentes configurações das juntas do robô, resultando na mesma pose do efetuador.

Os resultados foram precisos e coincidem com as 4 possíveis configurações, apresentadas na Fig. 3.1, para a cinemática inversa de um manipulador de 6 DoF com punho esférico (SICILIANO *et al.*, 2010). Em cinco simulações, o robô alcançou 3 das 4 configurações básicas. A quarta configuração, que seria o robô para trás com o cotovelo para cima, está fora do espaço de trabalho.

Apenas com o intuito de garantir a funcionalidade do *socket* de comunicação criado, a Figura 4.17 mostra uma comparação do resultado gerado pelo MatLab com aqueles mostrados no *FlexPendant* virtual do robô. As variáveis de junta observadas no *FlexPendant* de fato correspondem àquelas enviadas pelo MatLab, garantindo que o processo de comunicação e tráfego de dados foi efetivo e a pose alcançada pelo manipulador foi de fato aquela determinada pelo

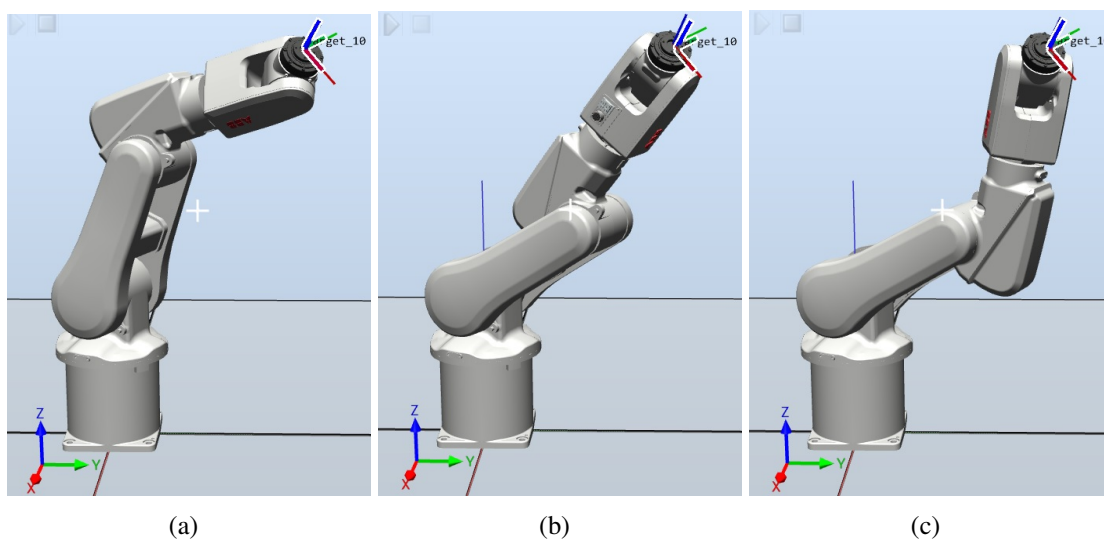


Figura 4.16: (a) Robô pra frente com cotovelo para cima (b) Robô para frente com cotovelo para baixo (c) Robô para trás com cotovelo pra baixo.

processo de otimização do AG

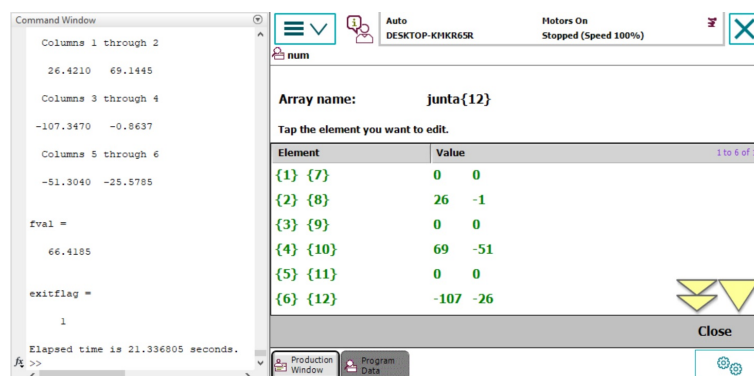


Figura 4.17: Comparação entre solução no MatLab e as variáveis de junta do robô no *FlexPendant*.

Foram realizadas ainda simulações considerando a função objetivo ponderada, com o intuito de comprovar a efetividade da solução para melhorar a eficiência energética do manipulador. Para essa simulação novamente os pontos cartesianos utilizados foram x, y e z iguais a 400mm, 200mm e 700mm, respectivamente e os ângulos de orientação iguais a zero. Todas as 10 simulações realizadas foram corretas alcançando a pose desejada. Entretanto, para o caso em que a função não foi ponderada, em 3 das 5 simulações o IRB120 atingiu uma pose voltado pra trás. Em contrapartida em todas as outras 5 simulações com a função ponderada, a pose final do robô foi voltada para frente, conforme exemplificado na Figura 4.18.

O traço vermelho que pode ser observado nas simulações indica a trajetória do efetuador para sair da posição inicial e alcançar a pose final desejada, que é marcada no ambiente virtual como “*Target₁₀*”. É notório que o movimento acumulado do robô é substancialmente reduzido

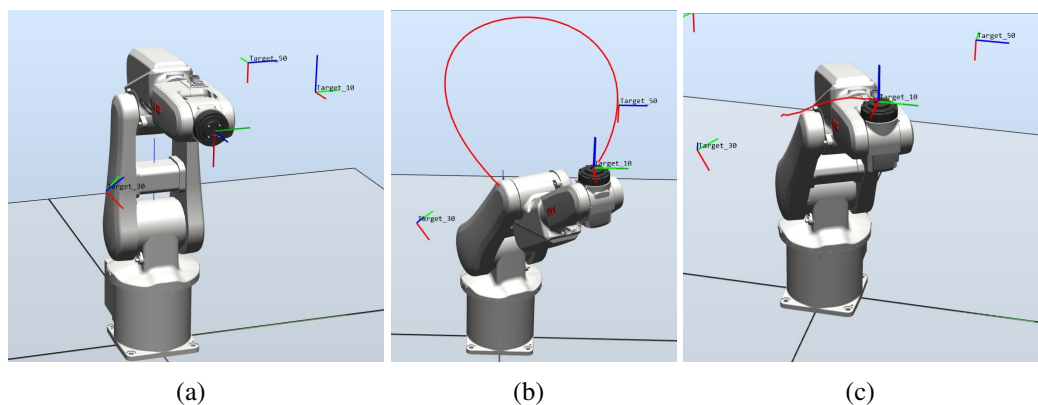


Figura 4.18: (a) Posição Inicial “Home” (b) Simulação sem ponderação da função objetivo com voltado para trás (c) Simulação com ponderação da função objetivo com voltado para frente.

quando a pose determinada é voltada para frente. Esse resultado comprova a eficácia do método, uma vez que reduzindo o deslocamento angular do manipulador, consequentemente o dispêndio de energia será menor.

4.6.2. Simulações no Coppelia

As simulações com o IRB120 no RobotStudio mostraram bons resultados. Entretanto conforme já foi explicitado, esse software, em sua versão gratuita não permite a inserção do sétimo eixo, para a simulação da cinemática inversa com 7DoF. Dessa forma, o algoritmo genético exposto na seção 4.3, foi simulado através do CoppeliaSim utilizando a cena mostrada na Figura 3.4. As simulações foram divididas em duas etapas, a primeira considerando apenas o controle de posição e a segunda utilização o controle de velocidade conforme proposto na seção 3.5. Importante ressaltar que as propriedades dinâmicas das juntas foram alteradas conforme a necessidade de cada teste. Para efeitos de comparação com as simulações anteriores, foi utilizado no CoppeliaSim as mesmas coordenadas cartesianas e de orientação.

Controle de Posição

As funções de controle através do API remoto mostrados na seção 3.5 foram construídas no MatLab permitindo que todo o controle fosse realizado através desse software. A Figura 4.19 mostra o resultado da simulação.

Como pode ser percebido através da figura, de fato o efetuador do robô alcançou o ponto desejado com a devida orientação, contudo diferente do RobotStudio o robô foi transladado lateralmente sob o trilho, portanto a pose final leva em consideração essa transladação, variando a configuração final das juntas. O que corrobora com a hipótese de que para essa simulação são muitas as soluções possíveis, e o método heurístico determinará poses distintas a cada execução.

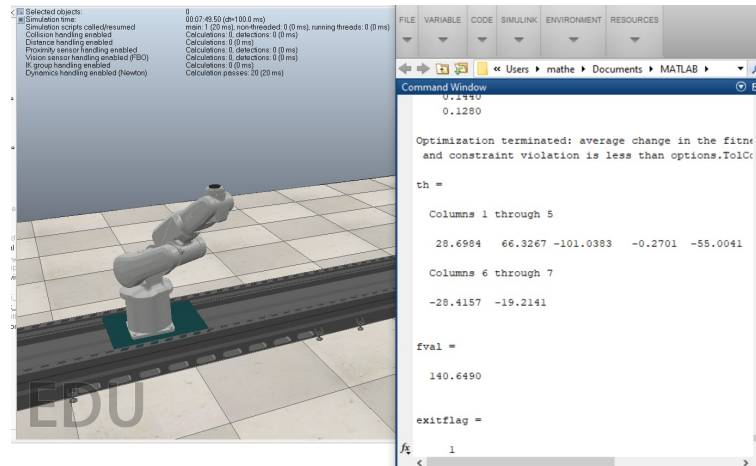


Figura 4.19: Simulação do controle de posição para manipulador com 7DoF.

Controle de Trajetória

A trajetória determinada utilizando o polinômio de quinta ordem foi estipulada para um tempo total de 10s. Como o *timestep* utilizado para simulação no CoppeliaSim é de 10ms, foi criada uma matriz com 7x100, onde são expressas as variáveis de junta a cada 10ms para o tempo total de 10s. Através de um enlace *for* essa matriz é destrinchada nas variáveis v_1, v_2, \dots, v_7 , que representam as velocidades “v” de cada junta para cada um dos instantes de tempo do *timestep* da simulação. Então esses valores são trafegados pela API remota até o CoppeliaSim utilizando a função “*simxSetTargetVelocity*” para que o controle de velocidade das juntas seja realizado.

De fato, as juntas tem sua velocidade comanda pelo algoritmo construído no MatLab. Dessa forma, elas se locomovem conforme a velocidade determinada pelo polinômio de quinta ordem, em cada instante de tempo. Para o tempo estipulado o robô realiza a trajetória até a pose final determinada pelo AG, comprovando a eficácia do método proposto, conforme demonstrado na Figura 4.20.

A simulação completa foi bastante efetiva. A pose final é determinada pelo algoritmo genético, que serve como *input* para o planejamento de trajetória utilizando o polinômio de quinta ordem. Determinadas as velocidades o efetuador alcança a pose desejada no tempo estipulado. Entretanto, é necessário ressaltar alguns pontos importantes, necessários ao sucesso dessa simulação: o MatLab deve enviar as informações de velocidade das juntas em uma taxa de tempo coincidente com o *timestep* da simulação, que para o caso foi determinada em 10ms. Assim sendo o envio das variáveis de velocidade são intervaladas em 0.01s utilizando a função *pause* no MatLab; além disso, como para o instante inicial e final as velocidades são iguais a 0 é essencial habilitar o freio dos motores para esses instantes. Caso contrário as juntas cederão à força da gravidade inviabilizando a simulação.

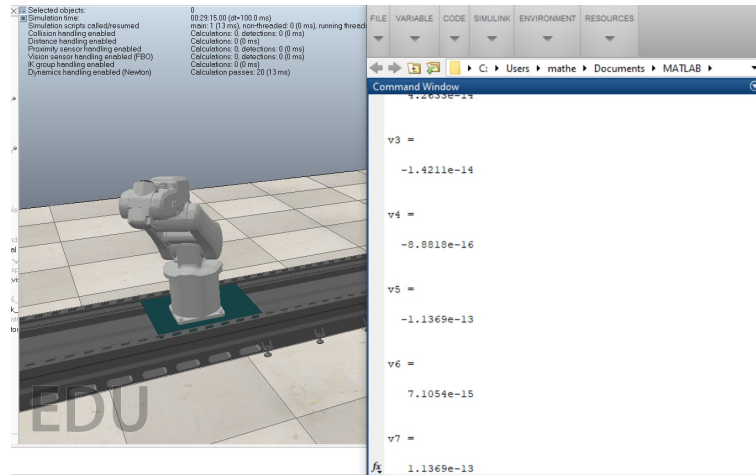


Figura 4.20: Simulação do controle de trajetória para manipulador com 7DoF.

4.7. Simulações com o IRB6700

A fim de comprovar a hipótese sobre a flexibilidade de adaptação do método e visando estender o estudo para manipuladores de padrão industrial, parte dos testes e simulações foram replicadas para o IRB6700. Em tese, ao se adaptar os cálculos de cinemática direta do manipulador, e ajustar os limites das juntas o algoritmo será capaz de resolver a cinemática inversa assim como para o IRB120. Portanto o primeiro passo é conhecer os parâmetros necessários à convenção de Denavit-Hantenberg extraídos das características físicas e cinemáticas mostradas na seção 2.3.9. A Tabela 4.9 mostra os parâmetros da convenção D.H para o IRB6700.

Tabela 4.9: Tabela D.H. IRB6700 com sétimo eixo.

$Junta_i$	$a_i(\text{mm})$	$d_i(\text{mm})$	$\alpha(^{\circ})$	$\theta(^{\circ})$
1	0	d_1	-90	0
2	370	780	-90	θ_2
3	1125	0	0	$\theta_3 - 90$
4	200	0	90	θ_4
5	0	-1142	-90	θ_5
6	0	0	-90	θ_6
7	0	200	0	$\theta_7 + 180$

Baseado nos parâmetros da tabela de D.H. é possível adaptar os cálculos da cinemática inversa expostos na seção 3.1.4. Aliado a essa adaptação, os limites das juntas também são alterados de acordo com os valores expostos na Tabela 2.4. Essas são as únicas modificações realizadas, mantendo a função objetivo, restrições e demais parâmetros do algoritmo genético inalterados. Nas simulações com o IRB6700, foram utilizados os pontos cartesianos x igual a 1600mm, y igual a 1000m e z igual a 2500m, os ângulos de orientação desejados foram mantidos todos iguais a 0. É possível notar que os valores dos pontos no espaço cartesiano são bem maiores que os utilizado para o IRB120. Isso se justifica pelo fato do espaço de trabalho

do IRB6700 ser bem mais amplo, conforme mostrado na Figura 2.5. As configurações do algoritmo genético foram mantidas as mesmas do melhor resultado encontrado na investigação, conforme tabela 4.8. A Figura 4.21, mostra o resultado da simulação realizada.

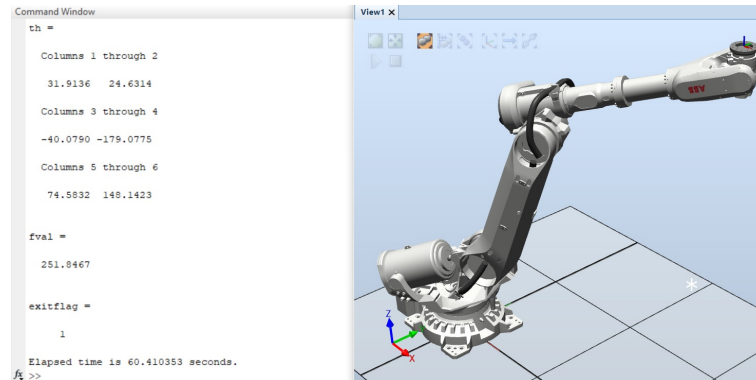


Figura 4.21: Simulação do AG aplicado ao IRB6700.

Nesse resultado novamente a variável “exitflag” foi igual a 1, indicando que o algoritmo alcançou o valor ótimo. Assim como para os testes do IRB120, “th” representa as variáveis de juntas para a pose que pode ser observada na simulação no RobotStudio e “fval” é o valor final da função objetivo. Chama a atenção nesses resultados o tempo que o algoritmo levou para alcançar o resultado ótimo. Isso se justifica também pelo fato do espaço de trabalho do IRB6700 ser consideravelmente maior, logo o espaço de busca do algoritmo também é aumentado.

Nesse caso, a estratégia de dopagem da população inicial, incluindo um indivíduo conhecidamente próximo da pose final pode contribuir na redução do tempo de processamento do algoritmo. Tal qual o realizado nas simulações do IRB120, o primeiro indivíduo da população foi alterado e as taxas de tempo da simulação chegaram a cair até três vezes conforme indica a Figura 4.22. Na simulação com a população não dopada as simulações demoraram em torno de 60s para convergir, ao passo que ao dopar a população a busca foi concluída num período de até 20s.

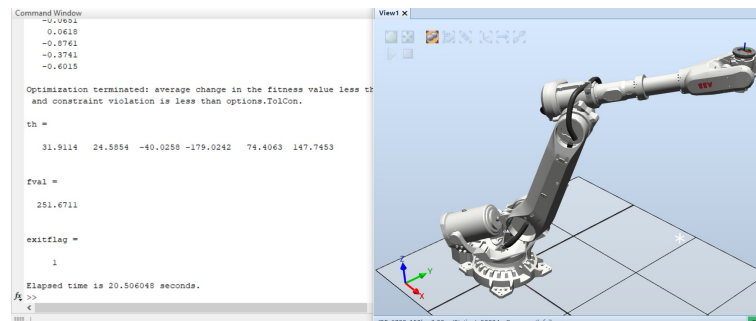


Figura 4.22: Simulação do AG aplicado ao IRB6700, com população inicial dopada.

Seguindo, conforme citado na seção 3.4, o ganho de eficiência energética pode ser maior conforme o porte do robô em que se aplica a solução. Como o IRB6700 é um robô industrial

com um alcance de mais de três metros, os torques dos motores das juntas necessários para movimentar toda a estrutura são substancialmente elevados. Assim sendo a ponderação da função objetivo com o intuito de otimizar o dispêndio energético do robô, ganha relevância nesse cenário. Novamente foram atribuídos pesos à função objetivo conforme descrito na seção 3.4. Para esse caso os fatores de ponderação $W_1, W_2 \dots W_6$ foram 0,421; 0,421; 0,100; 0,02; 0,02 e 0,01 respectivamente. A constatação dos ganhos mais uma vez foi realizada comparando o algoritmo no caso da função não ponderada, com o caso da função ponderada. Os resultados são demonstrados nos gráficos das Figuras 4.23 e 4.24.

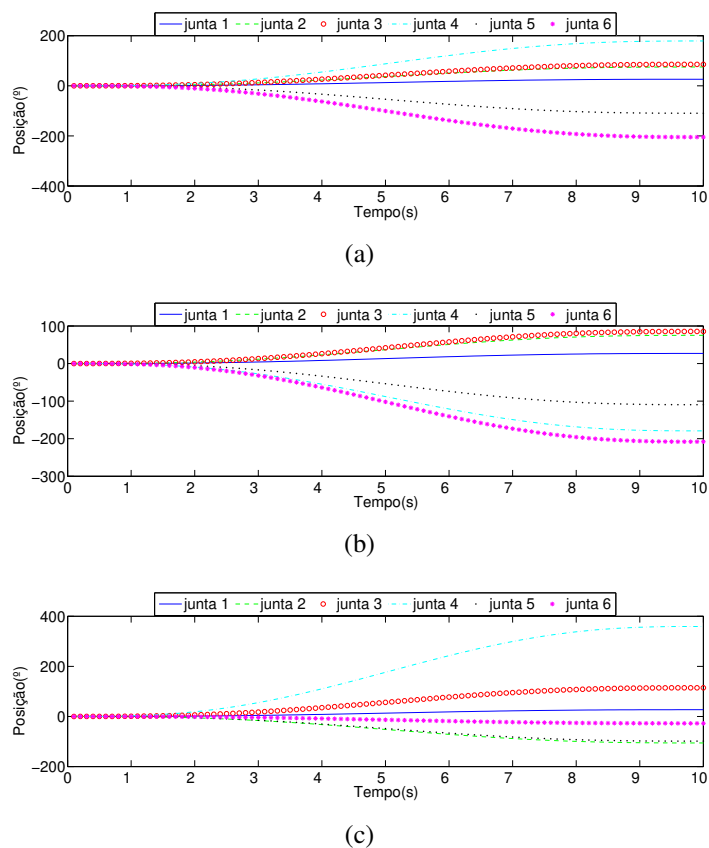
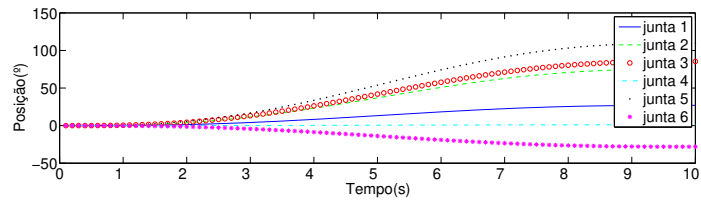


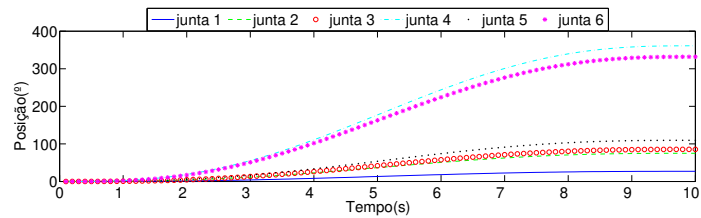
Figura 4.23: Simulações com função objetivo não ponderada.

Os resultados alcançados nos testes utilizando o IRB120 se repetiram para o manipulador IRB6700. As juntas de posicionamento tiveram um deslocamento menor em relação às juntas de orientação; novamente a junta 3 sempre se movimentou mais que as juntas 1 e 2, assim como a junta 6 sempre se locomoveu mais que as juntas 4 e 5. Nesse caso os principais ganhos foram em relação à junta 2 que efetivamente teve seu deslocamento angular diminuído com a função objetivo ponderada. Importante ressaltar que o processo de otimização em nenhum dos casos retornou uma pose final com o robô voltado para trás. Por esse motivo o impacto relacionado ao deslocamento da junta 1 foi pouco relevante nessas simulações.

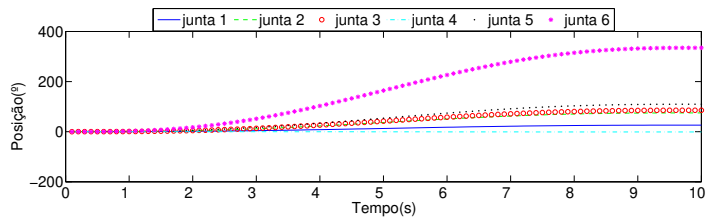
Considerando o porte do IRB6700, a junta prismática do eixo da base tem um impacto ainda maior em relação ao gasto energético para que o robô possa alcançar a pose final. Por



(a)



(b)



(c)

Figura 4.24: Simulações com função objetivo ponderada.

esse motivo os testes foram realizado considerando também a inserção do sétimo eixo. Assim como para a análise realizada com o IRB120, o torque do motor da junta prismática foi considerado 1,5 vezes maior que o torque da junta 1 do manipulador. Nessa situação hipotética o torque considerado foi de 31,5kNm e os resultados da otimização, comparando os resultados não ponderados com os ponderados são demonstrados nos gráficos das figuras 4.25 e 4.26.

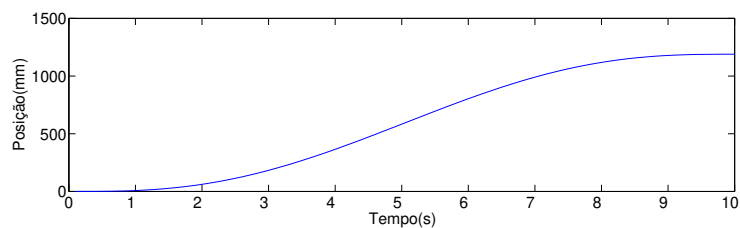


Figura 4.25: Deslocamento da junta 1 para o AG com a função objetivo não ponderada.

Como pode ser percebido através dos gráficos o deslocamento da junta pra o caso em que a função objetivo foi ponderada foi aproximadamente 3 vezes menor em relação ao caso convencional. Essa redução da quantidade de movimento é revertida diretamente em um menor dispêndio de energia pelo motor com maior torque da composição.

O intuito do o IRB6700 ter sido escolhido para as simulações foi sua disponibilidade

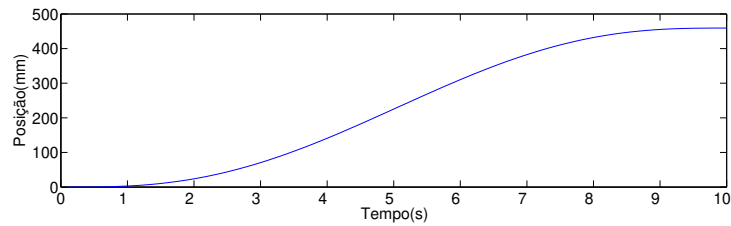


Figura 4.26: Deslocamento da junta 1 para o AG com a função objetivo ponderada.

para testes. Assim sendo os testes foram realizados para o manipulador real. A ideia era deixar de simular o controle de velocidade no CoppeliaSim e realizar os testes no robô real. Entretanto para que isso seja possível é necessário que o MatLab esteja em constante comunicação com o manipulador enviando os valores de velocidade para cada junta durante todo o tempo da trajetória. Entretanto, para viabilizar essa solução é necessário a função “*PC-Interface*” funcional no RobotStudio.

Essa função permite a comunicabilidade de um dispositivo externo, no caso o computador em que o MatLab está instalado, com o controlador do robô. Nos controladores virtuais criados nas simulações, essa função pode ser habilitada sem problemas acessando suas configurações. Em contrapartida, para os robôs reais essa função requer uma licença especial, adquirida a parte da licença do controlador. Essa licença possui um alto custo conforme demonstrado na Figura 4.27. Por se tratar de um robô aplicado em uma linha de produção industrial, sem objetivos acadêmicos e sem a necessidade de comunicação com um dispositivo externo, a licença para habilitar a função “*PC-Interface*” não foi adquirida, com o objetivo de reduzir custos.

3 PREMISSAS / REQUISITOS DO PROJETO								
3.1 DOCUMENTO(S) ENVIADO(S) PELO CLIENTE:								
- Serial do robô;								
66-78210								
4 INVESTIMENTO								
4.1 PREÇO								
Item do escopo	Descrição	Código de material / Serviço	NCM do produto	Preço Unitário sem Impostos (BRL)	ISS	ICMS	IPi	Preço Com Impostos (BRL)
2.1	PC Interface	Software	105	9.705,00	2%	-	-	10.935,00

Figura 4.27: Cotação da licença PC-Interface com a ABB.

Mediante a impossibilidade de comunicação com o dispositivo externo, a opção foi simplificar os testes. As variáveis de junta da pose determinada pelo AG foram inseridas manualmente no módulo Rapid do controlador e a execução dos movimentos foram realizadas utilizando a função “*MoveAbsJ*”, mantendo as curvas de velocidade e aceleração padrões Ro-

botStudio. De fato a solução com o algoritmo genético foi efetiva e o robô se locomoveu até a pose final desejada, a Figura 4.28 mostra a solução implantada.

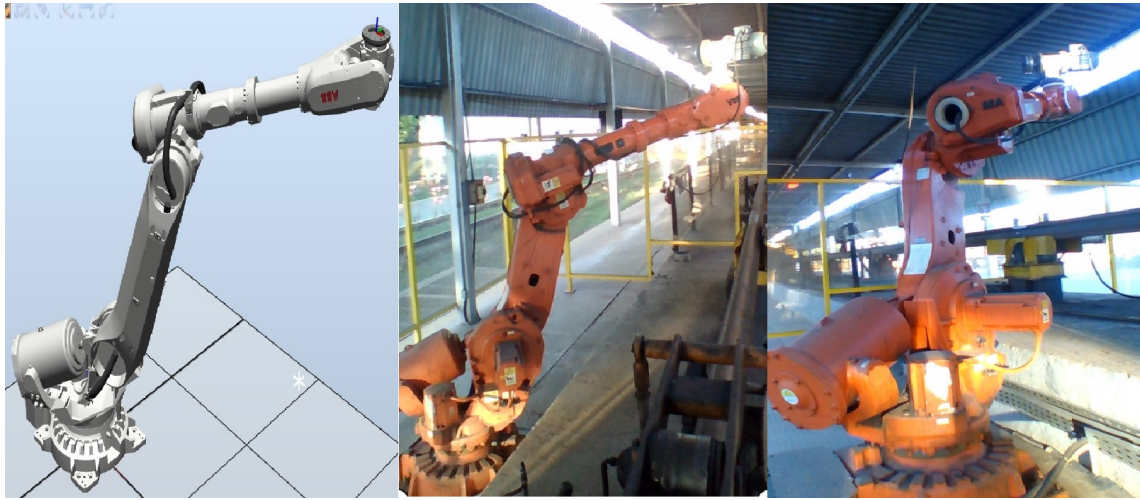


Figura 4.28: Simulação no IRB6700.

5. Aplicações da Solução - Estudos de Viabilidade de Casos

Conforme demonstrado nos capítulos anteriores, de fato os AGs podem ser uma solução viável e satisfatória para o problema da cinemática inversa. Portanto na sequência serão abordadas algumas aplicações reais em que a solução apresentada pode ser útil. Esse estudo foi realizado enfocando aplicações de células robotizadas já operacionais dentro da empresa Vale. O intuito é buscar entender a melhor forma dessa solução contribuir com os processos da empresa.

5.1. Montagem Robotizada de Carro de Grelha

A pelletização é uma etapa importante do processo de beneficiamento; esse processo consiste na aglomeração de minérios pobres, ultrafinos, impróprios para o uso direto nos fornos siderúrgicos. A transformação desse minério em esferas de diâmetro médio da ordem de 12mm, com propriedades químicas, físicas e metalúrgicas adequadas, possibilita a utilização desses na produção de ferro primário. O processo consiste basicamente de 3 etapas. A moagem é utilizada para cominuição a fim de diminuir a granulometria do minério. O pelletamento, onde a polpa misturada com aglomerantes e fundentes é transformada em pelotas ditas cruas (MOURÃO, 2017). Por fim a etapa mais importante do processo de pelletização é a queima das pelotas cruas, que confere ao produto resistência física para suportar o manuseio, transporte, esforços mecânicos e choques térmicos (DA SILVA CAVALCANTE *et al.*, 2017).

A queima das pelotas cruas acontecem em fornos de pelletização do tipo grelha móvel, como mostrado na Figura 5.1. Segundo Athayde (2013) esses fornos são equipamentos baseados em troca interna de calor, utilizados para secagem, queima e resfriamento das pelotas. Dentro desses fornos ocorre a sinterização dos grãos de minério.

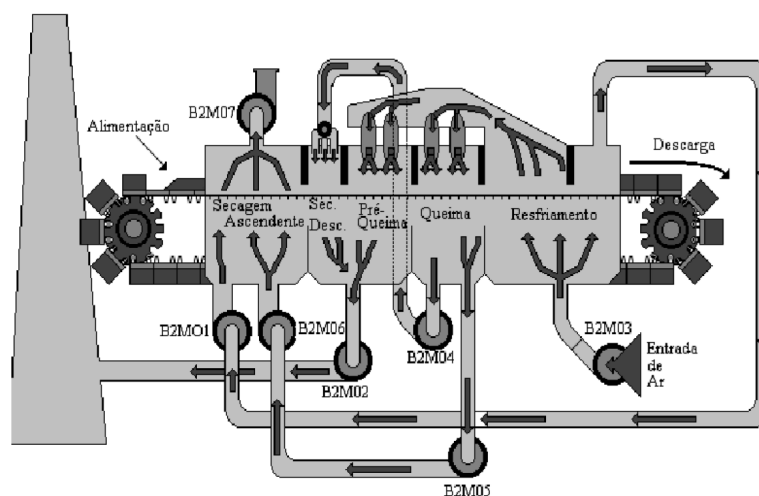


Figura 5.1: Forno tipo grelha móvel (MOURÃO, 2017).

Esses fornos são constituídos de uma esteira sem fim, composta de vários carros, cujo

quais no fundo são dispostas as barras de grelha fabricadas em aço inoxidável. Essa estrutura sustenta a carga de pelotas que transita pelo forno e permite a passagem de ar quente responsável pela reação termo-metalúrgica, conforme mostra a Figura 5.2 (TARPANI *et al.*, 2008).

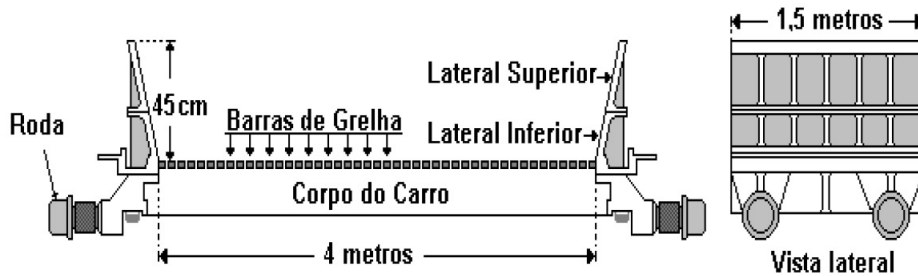


Figura 5.2: Carros de grelha (MOURÃO, 2017).

As barras que formam o carro de grelha são submetidas a uma temperatura de até 1100°C e a uma atmosfera oxidante que deteriora o aço de sua constituição. Por esses motivos tem vida útil determinada e precisam ser trocadas periodicamente. O processo produtivo é afetado diretamente em decorrência dessa troca, pois para que possa ser feita é obrigatória a parada do forno. Tais paralisações têm um grande impacto na produtividade da planta. Isso porque para retomada são necessárias uma série de medidas de respostas não imediatas, que demandam um período de tempo para retomada da plena operação (MOURÃO, 2017). O gráfico da Figura 5.3 exemplifica bem a situação.

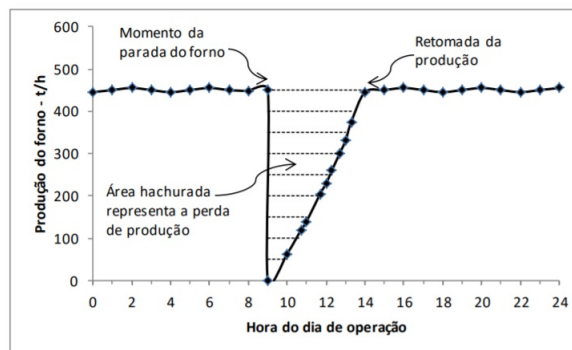


Figura 5.3: Impacto pela parada do forno de grelha móvel (MOURÃO, 2017).

Fica claro a importância da montagem e manutenção dos carros de grelha para o processo de produção de pelotas. Mourão (2017) enfatiza ao definir a grelha como a alma de um forno de pelletização convencional, uma vez que ela dita o ritmo e a qualidade do processo produtivo.

Um dos processos robotizados que é possível encontrar dentro da Vale, é justamente a montagem de carro de grelha para fornos de pelletização. A fim de aprimorar esse processo de montagem a Vale investiu em um robô antropomórfico da ABB, capaz de realizar a montagem dos carros de grelha de maneira automática. Antes realizada de maneira manual, a tarefa

consiste em retirar uma a uma as barras do estoque e colocar em locais determinados sob os carros de grelha. Esse mesmo movimento deve ser executado inúmeras vezes até que todo o carro seja montado. A Figura 5.4, mostra uma comparação entre a atividade manual e robotizada. A robotização desse tipo de atividade, além de contribuir com a preservação da saúde do funcionário, pode tornar o processo mais eficiente. Executá-lo de maneira mais rápida e precisa contribui diretamente com a redução do tempo em que o forno fica parado para manutenção.



Figura 5.4: Robô em sobreposição ao esforço ergonômico.

Os benefícios da utilização do robô são evidentes, contudo, existe um problema conceitual que torna esse robô ineficaz: a localização das barras de aço. Isso porque, o manipulador precisa localizar cada barra no estoque e colocá-lo na posição determinada sob o carro de grelha. Sem um método para que o robô possa fazer o reconhecimento, e passar a “enxergar” as peças, ele depende diretamente da intervenção humana para conseguir executar a tarefa. Dessa forma, é necessário que uma pessoa posicione as barras em locais pré determinados para que o robô consiga pegá-las. Obviamente isso não elimina o problema de movimentação repetitiva do executante, e a produtividade da célula volta a depender diretamente do trabalho humano.

Dai surge a necessidade de aprimorar a solução com métodos que consigam localizar a barra de maneira automática, e capturá-la sem depender de nenhum tipo de intervenção. O método de localização, apesar de não ser escopo desse trabalho, pode ser através de visão computacional, sensores lasers ou até mesmo soluções menos convencionais como sensores indutivos. O fato é que ao localizar a barra, é preciso determinar a cinemática inversa e trajetória do robô para a posição de captura. Uma das maneiras de realizar essa tarefa é utilizando as soluções expostas nesse trabalho.

A utilização de algoritmos evolutivos permitirá que seja encontrada uma pose otimizada, com erros espaciais e de orientação dentro de valores aceitáveis, para agarrar a peça. A principal vantagem da utilização desse método, é independer da posição inicial em que as peças sejam colocadas, desde que estejam dentro do espaço de trabalho do robô. Isso porque, conforme foi demonstrado, a população inicial do algoritmo genético é determinada de maneira aleatória, e convergirá para a solução ótima. Além disso, como é intuitivo pensar, nenhuma das peças a serem montadas estarão na mesma posição que a anterior, portanto a cada peça a ser retirada uma nova pose deve ser determinada. Caso seja delimitada uma área de aproximação onde as peças serão colocadas é possível que a população inicial do AG seja “dopada”, aumentando

a velocidade de conversão aos valores ótimos, o que possivelmente permitirá uma tomada de decisão em tempo real.

Além disso, o caráter repetitivo da tarefa, capturando e montando barras inúmeras vezes durante a manutenção, levam a crer que o dispêndio de energia do robô é elevado, sobretudo considerando seu porte. Portanto, a utilização da solução proposta nesse trabalho para melhorar a eficiência energética do robô pode trazer ganhos significativos, além de prorrogar sua vida útil.

Não é possível afirmar que esse seja de fato o melhor método a ser empregado, mas fica claro que a possibilidade de aplicação é plausível. Um estudo mais detalhado deve ser feito na planta já considerando o sistema de detecção, o espaço de trabalho e o formato das peças para determinar o quão eficaz a solução pode ser para esse caso em específico.

5.2. Célula Robotizada de Lavagem de Equipamentos

Outra solução robotizada dentro da Vale que chama atenção é a lavagem de equipamentos de grande porte na oficina de manutenção em Carajás. Uma das etapas essenciais ao processo de extração mineral é a manutenção do maquinário. Propiciar a essas máquinas uma maior disponibilidade física, reduzindo o tempo que ficam paradas em manutenção, é fator determinante para garantir a produtividade do processo.

Os procedimentos de manutenção em oficinas, dentro do ambiente de mineração normalmente começam com a lavagem dos equipamentos. Principalmente quando as intervenções acontecem em partes críticas, como motor, tanque de diesel, etc. Isso porque as condições de operação no ambiente de mina são bastante adversas. Fatores como a intensa suspensão de particulados, presença de lama, vazamentos de óleo devido a operações em condições extremas, entre outros fatores, contribuem para um acúmulo intenso de sujeira no equipamento. Dessa forma caso o equipamento não seja devidamente limpo, sistemas vitais como o de lubrificação, arrefecimento e de injeção de combustível podem ser contaminados.

Tradicionalmente essa lavagem era feita de maneira manual, com um funcionário operando um sistema hidráulico de alta pressão para conseguir limpar os equipamentos que podem chegar até 9m de altura e 12m de largura. A atividade realizada dessa forma, além de ser ineficiente, expõe seu executante a riscos de saúde, exigindo posturas inadequadas e contato direto com contaminantes químicos. Uma eventual falha do processo ou exposição a longo prazo pode provocar danos irreversíveis. Mediante esses fatores, foi realizado um investimento para substituição da mão de obra braçal por uma robô. A Figura 5.5 mostra uma comparação entre os dois métodos.

Como mostrado na figura, dois manipuladores de 6DoF da ABB foram equipados com bombas hidráulicas para aspergir água e sabão sobre os equipamento fora de estrada. Esses robôs foram montados sob uma plataforma móvel que permite o deslocamento lateral para atingir toda a extensão do objeto a ser lavado. Essa plataforma pode ser considerada um sétimo



Figura 5.5: Lavagem manual e robotizada de equipamentos de grande porte.

eixo para o robô que provoca um deslocamento ao longo do eixo y. Essa estrutura física e condição de operação é condizente com o problema analisado nesse trabalho. Um manipulador antropomórfico de cunho esférico da ABB, colocado sob um trilho que o translada lateralmente, propiciando ao sistema robotizado sete graus de liberdade.

Esse robô é programado pelo método de condução ponto a ponto. Nessa técnica o programador deve levar o manipulador nos pontos desejados um a um, então a informação de posição da junta é enviada especificamente nesses pontos da trajetória determinados. Posteriormente o robô se desloca por linha reta ou trajetórias circulares conforme indicado na programação (HENRIQUES *et al.*, 2002). Mais detalhadamente, são demarcados os pontos de maior relevância nos equipamentos fora de estrada, então o robô é movimentado manualmente a estes pontos e as coordenadas de posição e orientação são gravadas. Na sequência são utilizadas funções de deslocamento no espaço para que o robô faça uma varredura entre esses pontos demarcados, lançando água e sabão para limpar o equipamento.

A necessidade de determinar cada ponto torna o processo de programação mais moroso e menos preciso pois depende da acurácia da pessoa que vai levar o efetivador às posições desejadas. É relevante lembrar que essas máquinas possuem grandes dimensões e a interrupção de sua operação impacta diretamente a produção da empresa. Além disso a utilização dos robôs se torna menos flexível uma vez que para cada novo equipamento ou alterações relevantes nos já existentes é necessário refazer todo o processo.

Outro problema é que toda reprogramação do robô é realizada por mão de obra terceirizada. Isso dificulta novas configurações e reconfigurações para trabalhar com uma maior diversidade de equipamentos. Isso porque, em se tratando dos processos de contratação de fornecedores, esses costumam ser extremamente custosos, morosos e burocráticos. Outro fator importante é que renovações de frotas e modificações relevantes nos equipamentos, como a instalação do kit de autonomização de perfuratrizes, exigem uma maior flexibilidade na utilização da célula robotizada. Além disso, em virtude da célula de lavagem não ser uma estrutura móvel, a alteração dos equipamentos mantidos na oficina também demanda uma diversificação das rotinas de lavagem.

Importante citar o fator custo relativo à programação dos robôs. A ABB fornecedora exclusiva deste tipo de serviço, para a célula em questão, cobra valores altos para a criação de novas rotinas de lavagem. Os custos são preservados nesse trabalho por questões de confidencialidade corporativa. Mas para entendimento da ordem de grandeza desses valores, a cada nova programação, aproximadamente, seria possível comprar um manipulador novo, de mesmo porte. Em outras palavras, a programação dos robôs de lavagem impactam de maneira significativa o custo do processo.

Uma das maneiras de aprimorar esse processo é combinar técnicas de mapeamento a laser para determinação dos pontos relevantes, com o cálculo da cinemática inversa e consequente planejamento de trajetórias. E as soluções apresentadas nesse trabalho podem ser bastante úteis nesse aspecto. Também não é escopo desse trabalho determinar qual a melhor forma de se mapear o equipamento, mas hipoteticamente essa tarefa pode ser realizada utilizando sensores LIDAR. A partir do mapeamento multipontos a laser, é possível determinar os pontos relevantes para a lavagem, e a cinemática inversa pode ser determinada para que o efetuador atinja esses pontos utilizando os algoritmos genéticos apresentados nessa dissertação. Em linhas gerais, a partir da nuvem de pontos coletadas por um sensor de mapeamento é possível utilizar o método de RANSAC para identificar os planos dominantes necessários à lavagem. Uma vez que esses planos sejam encontrados, os pontos de vértice são reconhecidos e através do AG é determinado a melhor pose para atingir o ponto desejado.

Existem ainda diversas outras contribuições que essa técnica pode trazer para essa rotina robotizada. O primeiro ponto é que robô está colocado sob a plataforma móvel e portanto, conforme já foi demonstrado, possui uma gama extensa de soluções para a cinemática inversa. Conforme ficou constatado nessa dissertação os algoritmos genéticos podem ser efetivos para encontrar as soluções ótimas para esse manipulador de 7DoF.

Um segundo fator está relacionado com a dificuldade de posicionar equipamentos de tais proporções sempre no exato lugar para que o robô possa realizar a rotina pré programada. Atualmente são utilizadas marcas no chão e um manobreiro para auxiliar no posicionamento do equipamento. Erros nessa tarefa podem provocar problemas graves, como por exemplo a colisão do robô com o equipamento, ou a limpeza inadequada de algum ponto. Dessa forma, aliar técnicas de mapeamento a laser, com inteligência artificial como a proposta nessa dissertação, pode acrescentar autonomia ao robô para que ele se adapte às diferenças de posicionamento dos equipamentos.

Um ponto positivo a favor da solução utilizando algoritmos genéticos é que a adaptação do robô não precisa ser realizada em tempo real. Uma vez que o equipamento for parado, a cinemática inversa pode ser calculada para todos os pontos antes do início da rotina. Ou seja, se o AG for executado todos os ciclos de lavagem, é possível absorver diferenças de posicionamento em cada um deles. Considerando ainda que, apesar dos posicionamentos serem distintos, estarão sempre próximos, é possível dopar a população inicial para conseguir uma conversão rápida que não afetará o tempo total de ciclo.

Além disso, a solução para melhorar a eficiência energética desse robô também pode ser aplicada. Considerando que a composição possui sete graus de liberdade, e as poses para que o efetuador atinja o mesmo ponto são inúmeras, aplicar o algoritmo para diminuir a locomoção das juntas de maior torque pode trazer excelentes resultados. Essa hipótese se reforça ao ser considerado o porte dos robôs utilizados e a quantidade de ciclos realizadas diariamente, que podem variar entre 4 e 6 lavagens, com duração de 2 horas cada.

Mas, possivelmente, o ganho mais relevante está relacionado a flexibilização da lavagem de equipamentos. Considerando que atualmente apenas 4 tipos distintos de equipamentos são absorvidos pelo processo robotizado, muitos modelos ainda necessitam ser lavados pelo método manual. A aplicação dos AGs nesse processo pode possibilitar que o robô trabalhe normalmente independente do modelo de equipamento que se deseja lavar, com custos significativamente mais baixos.

Novamente, não é possível afirmar que a técnica apresentada seja a melhor forma de resolver o problema, mas a teoria se mostra bastante consistente para justificar testes nessa aplicação. É necessário desenvolver mais os conceitos, sobretudo sobre a aplicação de algoritmos de RANSAC para o problema e a sua integração com o AG. Além disso, é necessário fazer testes em campo para comprovar a eficácia do LIDAR e do algoritmo genético. De qualquer forma, esse trabalho apresenta uma contribuição relevante que pode trazer resultados positivos para os processos robotizados citados.

6. Conclusão

Nesse dissertação foi investigada a utilização de algoritmos genéticos para a solução da cinemática inversa de manipuladores robóticos. Foi testada e validada a eficácia do método para modelos antropomórficos, com punho esférico, com 6 e 7 graus de liberdade.

Os resultados demonstram que os AGs podem apresentar soluções precisas ao problema, comprovando algumas hipóteses levantadas como: de fato não é necessário um conhecimento prévio da atividade robotizada para aplicação da solução. A constatação está baseada no fato que a população inicial é gerada de maneira aleatória sem para tanto conhecer a atividade a ser executada, considerando apenas os limites das juntas do manipulador em uso; não são necessários muitos indivíduos na população inicial para que o algoritmo convirja para resultados ótimos, desde que o AG esteja configurado corretamente. Conforme mostrado na seção 4.2, as melhores configurações encontradas para o AG tendem a convergir de maneira rápida para a solução de ótimo global apenas com 10 indivíduos e 25 gerações. Aumentar o número de indivíduos não impedem que o resultado seja alcançado, entretanto pode elevar em muito o tempo necessário para encontrar o melhor resultado; em contrapartida, outra constatação realizada foi que AG's mal configurados podem causar impactos na obtenção dos resultados, ficando presos em ótimos locais, com margens de erro grandes, ou demorando tempos longos para atingir a solução ótima.

Mais uma conclusão importante extraída da investigação dos AGs foi que a precisão da pose final está diretamente relacionada ao número de gerações, portanto quanto menor o erro desejado, mais gerações são necessárias. Considerando isso, se fosse desconsiderada a margem de erro aceitável seriam necessárias muitas gerações para alcançar a solução perfeita onde os erros de posição e orientação são iguais a zero.

As investigações também mostraram que, em termos de convergência ao ótimo global com boas taxas de tempo, a melhor configuração do AG foi com seleção por roleta, crossover aritmético com taxa de 0,5 e mutação gaussiana. Essa conclusão diz respeito somente às configurações que foram analisadas nessa dissertação e não é possível afirmar que não existam configurações melhores. Para isso é necessário desenvolver e investigar outras hipóteses com diferentes combinações.

Outra constatação importante extraída dos resultados é que estratégias como a utilização de funções de elitismo e sobreposição de população também contribuem na conversão do resultado para valores ótimos globais. Para uma mesma configuração de seleção, crossover e mutação, a sobreposição de 20 a 25% da população e elitismo concedido a dois indivíduos por geração, melhorou a capacidade do algoritmo escapar de ótimos locais e também o tempo necessário para convergência.

É possível afirmar ainda que a estratégia que obteve melhores resultados foi dopar a população inicial. Quando se conhece o problema e é possível restringir o espaço de trabalho do efetuator, alterar a população inicial com indivíduos pertencentes ao espaço que se deseja

trabalhar é uma estratégia que garante a evolução para valores ótimos em taxas de tempo reduzidas. Essa estratégia é bastante útil sobretudo para processos industriais onde os robôs tem espaço de trabalho grande e conseqüentemente o espaço de busca do algoritmo também é aumentado. Ambos os processos robotizados citados na capítulo 5, servem como exemplo para utilização dessa estratégia. Para o caso do robô montador de carro de grelha a região onde o carro é colocado para manutenção pode ser pré determinada, de forma que a população inicial possa ser dopada com pontos aproximados aquela região. Para o caso do robô lavador de equipamentos fora de estrada fica ainda mais claro. A posição em que o robô é colocado em relação à região onde os equipamentos são estacionados restringe de maneira significativa seu espaço de trabalho, de forma que o robô sempre trabalhará voltado para frente.

Outra hipótese confirmada nesse trabalho foi a facilidade de adaptação da solução, o que permite aplicá-la a diferentes processos robotizados, bastando adaptar a cinemática do robô e seus limites de juntas. Isso foi comprovado realizando testes e simulações considerando dois robôs da ABB de portes bastante distintos. A solução de fato foi facilmente adaptada entre o IRB120 e IRB6700, de forma que foi necessário apenas alterar a tabela D.H conforme cada robô e ajustar os limites de cada junta de acordo com as especificações de produto. Esse é um resultado importante pois abre margens para que essa solução seja aplicada aos mais variados processos robotizados dentro da Vale citados por Cota *et al.* (2017).

Mais um dos objetivos proposto nesse trabalho que foi alcançado com sucesso foi a utilização do algoritmo genético para melhorar a eficiência energética da movimentação do robô, tanto para o solução da cinemática inversa de 6DoF, quanto para a de 7DoF. A função objetivo foi ponderada considerando a capacidade de torque de cada junta e os resultados demonstraram que de fato as juntas que possuem maior torque se locomoveram menos ao ser implementada essa solução. Importante observar que quanto maior o porte do robô, maior a economia de energia relacionada a essa solução. Portanto, ao considerar robôs indústrias de maior porte, que normalmente realizam a mesma tarefa de maneira repetitiva, ininterruptamente, a economia de energia a longo prazo pode ser de fato significativa com a utilização dessa solução.

Outro tema abordado nesse trabalho foi com relação ao planejamento de trajetória do robô a partir do resultado gerado pelo algoritmo genético. Foi proposta uma trajetória baseada no controle de velocidade das juntas, a partir de um polinômio de quinto grau. O algoritmo desenvolvido no MatLab permitiu coletar o resultado do AG e estipular a posição, velocidade e aceleração da junta para cada instante de tempo, para um trajetória com tempo total determinado. Os resultados teóricos foram precisos e os gráficos de posição, velocidade e aceleração de cada junta do robô coincidem ao que foi proposto por Spong *et al.* (2005). A estratégia utilizando o polinômio de quinta ordem é bastante útil pois acrescenta restrições às acelerações das juntas, evitando movimentações bruscas ao longo da trajetória, o que pode ser bastante eficaz sobretudo para aplicações em robôs reais.

Todas as soluções propostas e constatações realizadas foram confirmadas através de

simulações em ambiente virtual, utilizando tanto o software RobotStudio quanto o CoppeliaSim. Foram simuladas a solução da cinemática inversa para o robô com seis graus de liberdade, com sete graus de liberdade e considerando economia no dispêndio de energia durante a movimentação do robô, como também o controle de trajetória através da velocidade das juntas, e todas obtiveram êxito. Importante ressaltar ainda que o êxito da estratégia proposta, de utilizar algoritmos genéticos para determinação da cinemática inversa, foi validada em um robô real, ainda que de maneira restrita.

Por fim, foram realizados dois estudos de caso hipotéticos, de processos robotizados reais, que existem dentro da Vale: a montagem de carro de grelha móvel no processo de pelotização e a lavagem de equipamentos fora de estrada. No primeiro caso a montagem pode ser estruturada conciliando técnicas de visão computacional e algoritmos genéticos para determinação das poses necessárias. Já no segundo caso, a lavagem robotizada de equipamentos fora de estrada também pode ser otimizada utilizando a solução proposta para aumentar a flexibilidade durante a reprogramação da planta. Esses estudos são importantes pois levanta possibilidades de aplicação da solução desenvolvida em prol de uma melhoria operacional da empresa. Além disso permite criar uma ponte entre as soluções criadas no ambiente de pesquisa com a ponta do processo, onde acontece a atividade fim, utilizando manipuladores robóticos.

Em suma, é possível afirmar com a conclusão desse projeto que os algoritmos genéticos são eficazes para determinar a cinemática inversa de manipuladores robóticos. Aplicar esse tipo estratégia, além de retornar resultados precisos, ainda permite encontrar a melhor solução considerando objetivos diversos como a minimização do erro, do deslocamento angular e do dispêndio de energia. É possível afirmar também que a estratégia de planejar a trajetória, entre a pose inicial e a pose final determinada pelo AG, utilizando um polinômio de quinta ordem para determinar a velocidade das juntas é correta e traz resultados satisfatórios. Finalmente, é possível concluir que essa estratégia se mostra bastante flexível para adaptação aos mais diversos tipos de manipuladores e processos, e mostra robustez o suficiente para justificar experimentos em processos industriais .

6.1. Contribuições

Ao término dessa dissertação é possível elencar uma série de contribuições que esse projeto trouxe. A primeira e principal delas são os algoritmos evolutivos desenvolvidos para determinação da cinemática inversa de manipuladores com até sete graus de liberdade. Esses algoritmos são ainda adaptáveis para que a solução evolua para determinação da cinemática inversa de robôs com uma quantidade maior de juntas e morfologias distintas.

Uma segunda contribuição importante são os códigos de integração entre o MatLab e os simuladores virtuais de robótica, RobotStudio e CoppeliaSim. As soluções desenvolvidas são genéricas e podem ser replicadas pra diversas outras aplicações do campo da robótica.

Outra contribuição importante foi o desenvolvimento de um algoritmo que permite si-

mular o planejamento de trajetórias através do controle de velocidade das juntas. Esse algoritmo é genérico e pode ser aplicado a outros tipos de robôs em que se deseje realizar o planejamento de trajetória através da mesma estratégia.

Acerca dos algoritmos e códigos desenvolvidos, está sendo estudada pelo ITV a possibilidade de registro de programa junto ao Instituto Nacional de Propriedade Industrial (INPI).

Por fim, os resultados parciais relacionados a determinação da cinemática inversa para o robô com 6DoF e o controle de trajetórias pela velocidade das juntas foram apresentados e discutidos no artigo “Algoritmos Genéticos para Determinação da Cinemática Inversa de um Robô de 6DoF”, e serão publicados no XXIII Congresso Brasileiro de Automática de 2020.

6.2. Trabalhos Futuros

Apesar dos resultados alcançados nesse projeto terem sido bastante satisfatórios ainda são necessários avanços para acrescentar maior robustez à solução. A próxima etapa desse projeto é testar a solução em um manipulador que possua um controlador com todas as licenças necessárias, de forma a confirmar a eficácia da solução por completo.

Outra etapa essencial para validação dos resultados apresentados nessa dissertação será testar a solução em um manipulador robótico alocado sob um trilho que funcione como um sétimo eixo, como é o caso do lavador de equipamentos fora de estrada em Carajás. Esse tipo de teste é importante para certificar que não existam impedimentos para a solução que por ventura não tenham sido observados em ambientes virtuais.

Mais uma oportunidade é replicar a solução para outros tipos de manipuladores que não sejam da ABB, que possuam uma quantidade diferente de graus de liberdade, ou que possuam uma configuração de punho diferente da esférica. Os resultados aqui apresentados se restringiram aos robôs antropomórficos da ABB. Entretanto é possível que a solução seja viável para outros tipos de manipuladores, como o UR5, que também é um equipamento que está disponível no Instituto Tecnológico da Vale. Além disso, uma evolução do estudo pode acontecer para aplicação desse tipo de solução em outros tipos de robôs que não sejam antropomórficos, como robôs móveis (com rodas ou patas), robôs planares, drones ou até mesmo robôs humanoides.

Em relação aos Algoritmos Genéticos, apesar dos resultados alcançados terem sido bastante satisfatórios, utilizar técnicas de paralelização de AG's pode trazer contribuições significativas. Entendendo que os AG's demandam muitos recursos computacionais é possível paralelizar seu processamento de forma vários pontos no domínio do problema sejam pesquisados simultaneamente. Dividir a execução do algoritmo entre diversos processadores pode essencialmente reduzir o tempo de execução para cada geração. Mas além disso, essa técnica pode aumentar a cobertura do espaço de soluções aumentando a chance de encontrar a solução ótima, como também levar a uma convergência prematura quando poucos indivíduos mais aptos dominam a população (PESSINI, 2003).

Outro trabalho que contribuirá muito com essa solução é a integração com técnicas e/ou

dispositivos de visão computacional, ou mapeamento a laser. A cinemática inversa foi determinada de maneira correta pelo algoritmo genético, mas em todas os testes e simulações os pontos cartesianos no espaço e a orientação que eram desejados foram determinados de maneira manual pela interface gráfica ou pelas próprias linhas de código do MatLab. Integrar a essa solução dispositivos que permitam reconhecer o ambiente ao redor, permitirá que os pontos desejados sejam reconhecidos de maneira automática para execução do algoritmo. Isso acrescentará autonomia ao robô, aumentando a robustez em sua capacidade de tomada de decisão. Esse passo é fundamental também para que a solução seja aplicada nos processos robotizados da Vale que foram estudados nessa dissertação.

Além disso, ao utilizar técnicas para reconhecer o espaço de trabalho do robô, é possível identificar obstáculos que devem ser evitados. Esse tipo de situação é bastante comum em ambientes de produção reais, e é essencial que o manipulador consiga desviar de obstáculos a fim de evitar a colisão, inclusive com seres humanos que possivelmente possam adentrar seu espaço de trabalho. Para tanto é necessário aprimorar a técnica de planejamento de trajetórias de forma que uma vez que os obstáculos sejam detectados o caminho estipulado ao manipulador evite colisões.

É possível tratar o problema utilizando métodos offline onde a posição e a forma dos obstáculos são previamente conhecidas e estes estão imóveis ou as características do seu movimento são conhecidas. Dessa forma uma malha fechada de controle é criada a fim de minimizar o erro entre a pose do robô e a referência pré determinada para a trajetória naquele instante.

Uma outra forma é tratar o problema de maneira online, onde sensores são utilizados para reconhecer a posição e forma do obstáculo no exato instante da trajetória do robô. Por natureza esse método realiza o planejamento de trajetória e execução da trajetória acontecem de maneira simultânea, de forma que um sinal de controle é enviado a medida em que os obstáculos são reconhecidos. O método dos Campos Potenciais Artificiais é um exemplo dos métodos comumente utilizados para tratar esse problema.

Ainda considerando a segurança da execução do movimento do robô, um importante trabalho a ser desenvolvido é uma algoritmo que possa detectar possíveis colisões em ambiente de simulação antes que a trajetória seja designada a um robô real. É possível utilizando recursos dos próprios ambientes de simulação utilizados nessa dissertação, prever que o movimento do manipulador gerará uma colisão, permitindo que essa seja previamente tratada, evitando acidentes relacionados à operação do robô.

Por fim, será interessante estudar outros processos robotizados da Vale em que essa solução pudesse ser empregada. Nessa dissertação foram detalhados apenas dois casos, mas é possível que a solução possa contribuir com o robô utilizado para emendas de correia, para os robôs de soldagem de vagões ou para os robôs utilizados para análises químicas e físicas de amostras de minério. Conforme foi observado, a solução se mostrou flexível à adaptações portanto podem haver muitas outras oportunidades em que ganhos seriam observados.

Referências Bibliográficas

- ABB, R. “IRB 120 ABB’s 6 axis robot – for flexible and compact production”. <https://search-ext.abb.com/library/Download>, 2019a. Acessado: 11 maio 2019.
- ABB, R. “Product specification: IRB120”. <https://new.abb.com/products/robotics/industrial-robots/irb-120/irb-120-data>, 2019b. Acessado: 11 maio 2019.
- ABB, R. “RAPID Instructions, Functions and Data types”. <https://library.abb.com/pt>, 2018. Acessado: 20 junho 2019.
- ABB, R. “Product specification: IRB 6700”. <https://library.e.abb.com/public>, 2019c. Acessado: 08 agosto 2020.
- ARISTIDOU, A., LASENBY, J. “FABRIK: A fast, iterative solver for the Inverse Kinematics problem”, *Graphical Models*, v. 73, n. 5, pp. 243–260, set. 2011. doi: 10.1016/j.gmod.2011.05.003. Disponível em: <<https://doi.org/10.1016/j.gmod.2011.05.003>>.
- ATHAYDE, M. *Modelamento Fluidodinâmico da Zona de Queima de Fornos de Grelha Móvel para Pelotização de Minério de Ferro*. Tese de Mestrado, Universidade Federal de Minas Gerais, 2013.
- CABRAL, E. L. L. “Livro em Elaboração: Cap5 - Cinemática Direta de Robôs Manipuladores”. <http://sites.poli.usp.br/p/eduardo.cabral/Cinemática%20Direta.pdf>, 2016. Acessado: 18 abril 2020.
- CALLIOLI, C. A., DOMINGUES, H. H., COSTA, R. C. F. *Álgebra Linear e Aplicações*. Brasil, Atual, 1993. ISBN: 9788570562975.
- COPPELIA, R. “CoppeliaSim, Create. Compose. Simulate. Any Robot”. <https://www.coppeliarobotics.com/>, 2018a. Acessado: 30 julho 2020.
- COPPELIA, R. “CoppeliaSim, Create. Compose. Simulate. Any Robot”. <https://www.coppeliarobotics.com/helpFiles/en/remoteApiOverview.htm>, 2018b. Acessado: 06 agosto 2020.

- CORBACHO, A. G. “Desarrollo de una interfaz para el control del robot IRB120 desde Matlab”. 2014. Trabajo Fin de Grado (Grado en Ingeniería en Electrónica Y Automática Industrial), Universidad de Alcalá, España.
- COTA, E., TORRE, M. P., FERREIRA, J. A. T., et al.. “Robótica na Mineração”. Em: *ABM Proceedings*. Editora Blucher, out. 2017.
- DA SILVA, S. R. X., SCHNITMAN, L., FILHO, V. C. “Análise da Eficiência Computacional para Solução do Problema da Cinemática Inversa de Robôs Seriais Utilizando a Teoria de Bases de Gröbner”. Em: *Anais do 14º Simpósio Brasileiro de Automação Inteligente*. Galoa, 2019.
- DA SILVA CAVALCANTE, M. V., FIGUEIRA, R. M., NUNES, S. F. “RELAÇÃO ENTRE OS ESPAÇOS VAZIOS ENCONTRADOS NO LEITO DE PELOTAS EM UM FORNO DE GRELHA MÓVEL E OS CUSTOS COM ENERGÉTICOS E A PRODUTIVIDADE”. Em: *Anais dos Seminários de Redução, Minério de Ferro e Aglomeração*. Editora Blucher, set. 2017. doi: 10.5151/2594-357x-26818. Disponível em: <<https://doi.org/10.5151/2594-357x-26818>>.
- ERTHAL, J. L. *Estudo de Métodos para a Solução da Cinemática Inversa de Robôs Industriais para Implementação Computacional*. Tese de Mestrado, Universidade Federal de Santa Catarina, 1992.
- FERRENTINO, E., CIOPPA, A. D., MARCELLI, A., et al.. “An Evolutionary Approach to Time-Optimal Control of Robotic Manipulators”, *Journal of Intelligent & Robotic Systems*, v. 99, n. 2, pp. 245–260, nov. 2019.
- FRANQUEIRA, T. C. “Um Estudo Sobre Quatérnios e sua Aplicação em Robótica”, *I SBAI*, pp. 528,529, 1993.
- HARARI, Y. N. *Sapiens, Uma Breve História da Humanidade*, v. 1. 36 ed. Porto Alegre - Brasil, LPM, 8 2018. ISBN: 9788525432186.
- HEINEN, M. R. *Controle inteligente do caminhar de robôs móveis simulados*. Tese de Mestrado, Universidade do Vale do Rio dos Sinos, 2007.
- HENRIQUES, R., DUTRA, M., ROSÁRIO, J., et al.. *ROBÓTICA INDUSTRIAL: Aplicação na Indústria de Manufatura e de Processos - Cap6: Programação e Simulação de Robôs*. Brasil, Edgard Blücher, 2002. ISBN: 8521203152.
- HOLLAND, J. H. “Genetic algorithms”, *Scientific American*, v. 267, n. 1, pp. 66–73, 1992.
- KÖKER, R. “A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization”, *Information Sciences*, v. 222, pp. 528–543, fev. 2013.

- KÖKER, R., ÖZ, C., ÇAKAR, T., et al.. “A study of neural network based inverse kinematics solution for a three-joint robot”, *Robotics and Autonomous Systems*, v. 49, n. 3-4, pp. 227–234, dez. 2004.
- LACERDA, E. G. M., CARVALHO, A. C. P. D. L. F. “Introdução aos algoritmos genéticos”. Em: *Congresso Nacional da Sociedade Brasileira de Computação*. EntreLugar, 1999.
- MARIANO, D. T., BISSOCHI JUNIOR, C. A., GONÇALVES, F. A. D. S., et al.. “Desenvolvimento de um sistema de controle em malha fechada para um braço robótico do tipo puma.” *XI CEEL*, pp. 2,3, 2013.
- MATHWORKS. “Global Optimization Toolbox”. <https://www.mathworks.com/help/gads>, 2020. Acessado: 04 março 2020.
- MESQUITA, P. P. D., CARVALHO, P. S. L. D., OGANDO, L. D. “Desafios da mineração: desenvolvimento e inovação para redução dos impactos ambientais e sociais”. <https://www.bndes.gov.br/wps/portal/sitehome/conhecimento/noticias/noticia/inovacao-tecnologia-mineracao-metais>, 2017. Acessado: 17 junho 2019.
- MOMANI, S., ABO-HAMMOU, Z. S., ALSMAD, O. M. “Solution of Inverse Kinematics Problem using Genetic Algorithms”, *Applied Mathematics & Information Sciences*, v. 10, n. 1, pp. 225–233, jan. 2016.
- MOURÃO, J. M. *Aspectos Conceituais Relativos à Pelotização de Minérios de Ferro*. Brasil, ABM - Brazilian Association for Metallurgy, Materials and Mining, 2017.
- MURRAY, R. M., LI, Z., SASTRY, S. S. *A Mathematical Introduction to Robotic Manipulation*. California - USA, CRC Press, 1994. ISBN: 9780849379819.
- NUNES, L. E. N. D. P. *Geração e otimização de trajetórias de um manipulador robótico utilizando algoritmos genéticos*. Tese de Doutorado, Universidade Estadual Paulista, 2007.
- NUNES, R. F. *Mapeamento da cinemática inversa de um manipulador robótico utilizando redes neurais artificiais configuradas em paralelo*. Tese de Mestrado, Universidade Estadual Paulista, 2016.
- PAVAN, K. K., MURALI, M. J., SRIKANTH, D. “Generalized solution for inverse kinematics problem of a robot using hybrid genetic algorithms”, *International Journal of Engineering & Technology*, v. 7, n. 4.6, pp. 250, set. 2018.
- PERALTA, E., FABREGAS, E., FARIAS, G., et al.. “Development of a Khepera IV Library for the V-REP Simulator”, *IFAC-PapersOnLine*, v. 49, n. 6, pp. 81–86, 2016.

- PESSINI, E. C. *Algoritmos Genéticos Paralelos – Uma Implementação Distribuída Baseada em Javaspaces*. Tese de Mestrado, Universidade Federal de Santa Catarina, 2003.
- PIRES, E. J. S. *Algoritmos Genéticos: Aplicação à Robótica*. Tese de Mestrado, Universidade do Porto, 1998.
- QUESADA, R. C. *Projeto e Concepção de Células Robotizadas para Aplicações em Automação*. Tese de Mestrado, Universidade Estadual de Campinas, 2014.
- ROS. “About ROS”. <https://www.ros.org/about-ros/>, 2018. Acessado: 18 agosto 2020.
- RUBRECHT, S., SINGLA, E., PADOIS, V., et al.. *New Horizons in Evolutionary Robotics - Chapter 1: Evolutionary Design of a Robotic Manipulator for a Highly Constrained Environment*. Paris - France, Springer-Verlag Berlin Heidelberg 2011, 2012. ISBN: 978-3-642-18271-6.
- SICILIANO, B., SCIAVICCO, L., VILLANI, L., et al.. *Robotics: modelling, planning and control*. Advanced textbooks in control and signal processing. Napoli - Italy, Springer London, 2010. ISBN: 9781846286414.
- SPONG, M., HUTCHINSON, S., VIDYASAGAR, M. *Robot Modeling and Control*. USA, Wiley, 2005. ISBN: 9780471649908.
- ŠTEVO, S., SEKAJ, I., DEKAN, M. “Optimization of Robotic Arm Trajectory Using Genetic Algorithm”, *IFAC Proceedings Volumes*, v. 47, n. 3, pp. 1748–1753, 2014.
- TARPANI, J. R., MILAN, M. T., MALUF, O., et al.. “ANÁLISE DE FALHA DE UM COMPONENTE ESTRUTURAL DE PLANTA DE PELOTIZAÇÃO DE MINÉRIO DE FERRO”, *Tecnologia em Metalurgia e Materiais*, v. 5, n. 1, pp. 51–55, 2008. doi: 10.4322/tmm.00501010. Disponível em: <<https://doi.org/10.4322/tmm.00501010>>.
- TINÓS, R. “Comportamento auto-organizável em algoritmos genéticos aplicados a robôs móveis em ambientes dinâmicos”, *Sba: Controle & Automação Sociedade Brasileira de Automatica*, v. 18, n. 1, pp. 13–23, mar. 2007.
- TORRE, M. P. “Estratégias de integração entre braço manipulador ABB IRB 120 e mão robótica Barrett BH8-282 utilizando o ROS”. 2017. Monografia (Bacharel em Engenharia de Controle e Automação), UFOP (Universidade Federal de Ouro Preto), Ouro Preto, Brasil.
- VALE. “Resultados Financeiros da Vale”. <http://saladeimprensa.vale.com/Paginas/Segmentos.aspx?s=Financas&sID=5>, 2019. Acessado: 05 dezembro 2019.

- VARGAS, L. V. *Inversa Filtrada: Uma solução alternativa para a cinemática inversa de manipuladores robóticos*. Tese de Mestrado, Universidade Federal do Rio de Janeiro, 2013.
- VARHEGY, T., MELIK-MERKUMIANS, M., STEINEGGER, M., et al.. “A Visual Servoing Approach for a Six Degrees-of-Freedom Industrial Robot by RGB-D Sensing”, *OAGM&ARW Joint Workshop 2017*, p. 75, 2017.
- VILLELA, M. “Mineração cada vez mais na Era da Tecnologia”. <http://noticiasmineracao.mining.com/2017/05/22/mineracao-cada-vez-mais-na-era-da-tecnologia>, 2017. Acessado: 18 junho 2019.
- WELCH, R., LIMONADI, D., MANNING, R. “Systems engineering the Curiosity Rover: A retrospective”. Em: *2013 8th International Conference on System of Systems Engineering*. IEEE, jun. 2013. doi: 10.1109/sysose.2013.6575245. Disponível em: <<https://doi.org/10.1109/sysose.2013.6575245>>.

A. Algoritmo Genético no MatLab

A.1. Cinemática Direta

```
%%%% MATRIZ DE DENAVIT- HARTEMBERG MODELO IRB 120 %%%
a1 = 0; d1 = 290; alpha1 = -90; th1 = q(1);
a2 = 270; d2 = 0; alpha2 = 0; th2 = q(2) - 90;
a3 = 70; d3 = 0; alpha3 = -90; th3 = q(3);
a4 = 0; d4 = 302; alpha4 = 90; th4 = q(4);
a5 = 0; d5 = 0; alpha5 = -90; th5 = q(5);
a6 = 0; d6 = 72; alpha6 = 0; th6 = q(6) + 180;
a0 = 0; d0 = q(7); alpha0 = -90; th0 = 0;
% %%%MATRIZ DE DENAVIT- HARTEMBERG MODELO IRB 6700 %%%
a1 = 370; d1 = 780; alpha1 = -90; th1 = q(1);
a2 = 1125; d2 = 0; alpha2 = 0; th2 = q(2) - 90;
a3 = 200; d3 = 0; alpha3 = 90; th3 = q(3);
a4 = 0; d4 = -1142.5; alpha4 = -90; th4 = q(4);
a5 = 0; d5 = 0; alpha5 = -90; th5 = q(5);
a6 = 0; d6 = 200; alpha6 = 0; th6 = q(6) + 180;
a0 = 0; d0 = q(7); alpha0 = -90; th0 = 0;
% %%% MATRIZ DE TRANSFORMA O HOMOGNEA %%%

% T00 = [ cosd(th0) -cosd(alpha0)*sind(th0)
%         sind(alpha0)*sind(th0) a0*cosd(th0);
%         sind(th0) cosd(alpha0)*cosd(th0)
%         -sind(alpha0)*cosd(th0) a0*sind(th0);
%         0 sind(alpha0) cosd(alpha0) d0;
%         0 0 0 1];

T00 = [ 1 0 0 0;
        0 1 0 d0;
        0 0 1 0;
        0 0 0 1];

T01 = [ cosd(th1) -cosd(alpha1)*sind(th1)
        sind(alpha1)*sind(th1) a1*cosd(th1);
        sind(th1) cosd(alpha1)*cosd(th1)
        -sind(alpha1)*cosd(th1) a1*sind(th1);
        0 sind(alpha1) cosd(alpha1) d1;
```

```
0 0 0 1];
```

```
T12 = [cosd(th2) -cosd(alpha2)*sind(th2)
       sind(alpha2)*sind(th2) a2*cosd(th2);
       sind(th2) cosd(alpha2)*cosd(th2)
       -sind(alpha2)*cosd(th2) a2*sind(th2);
       0 sind(alpha2) cosd(alpha2) d2;
       0 0 0 1];
```

```
T23 = [cosd(th3) -cosd(alpha3)*sind(th3)
       sind(alpha3)*sind(th3) a3*cosd(th3);
       sind(th3) cosd(alpha3)*cosd(th3)
       -sind(alpha3)*cosd(th3) a3*sind(th3);
       0 sind(alpha3) cosd(alpha3) d3;
       0 0 0 1];
```

```
T34 = [cosd(th4) -cosd(alpha4)*sind(th4)
       sind(alpha4)*sind(th4) a4*cosd(th4);
       sind(th4) cosd(alpha4)*cosd(th4)
       -sind(alpha4)*cosd(th4) a4*sind(th4);
       0 sind(alpha4) cosd(alpha4) d4;
       0 0 0 1];
```

```
T45 = [cosd(th5) -cosd(alpha5)*sind(th5)
       sind(alpha5)*sind(th5) a5*cosd(th5);
       sind(th5) cosd(alpha5)*cosd(th5)
       -sind(alpha5)*cosd(th5) a5*sind(th5);
       0 sind(alpha5) cosd(alpha5) d5;
       0 0 0 1];
```

```
T56 = [cosd(th6) -cosd(alpha6)*sind(th6)
       sind(alpha6)*sind(th6) a6*cosd(th6);
       sind(th6) cosd(alpha6)*cosd(th6)
       -sind(alpha6)*cosd(th6) a6*sind(th6);
       0 sind(alpha6) cosd(alpha6) d6;
       0 0 0 1];
```

```
%T = T01*T12*T23*T34*T45*T56;
```

```
T = T00*T01*T12*T23*T34*T45*T56;
```

```
p = [T(1,4),T(2,4),T(3,4)];
```

```
%%%% Matriz de Rota o %%%
```

```
R = [T(1,1), T(1,2), T(1,3);  
     T(2,1), T(2,2), T(2,3);  
     T(3,1), T(3,2),T(3,3)];
```

```
%%%% Desmembramento da Matriz de Rota o em vari veis
```

```
r11 = R (1,1);  
r12 = R (1,2);  
r13 = R (1,3);  
r21 = R (2,1);  
r22 = R (2,2);  
r23 = R (2,3);  
r31 = R (3,1);  
r32 = R (3,2);  
r33 = R (3,3);
```

```
%%%%
```

```
%%%% C lculo da orienta o direta %%%
```

```
if (xf(5) > -90 && xf(5) <90)  
    Alpha = atan2(r21,r11);  
    Beta = atan2(-r31, sqrt(r32^2+r33^2));  
    Gama = atan2(r32, r33);  
elseif (xf(5) > 90)  
    Alpha = atan2(-r21,-r11);  
    Beta = atan2(-r31, -sqrt(r32^2+r33^2));  
    Gama = atan2(-r32, -r33);  
elseif(xf(5) == -90)  
    Beta = -pi/2;  
    A = [1 1; 1 -1];  
    b = [atan2(-r12,r22); 0];  
    sol = A^-1*b;  
    Alpha = sol(1);  
    Gama = sol(2);  
elseif(xf(5) == 90)  
    Beta = pi/2;
```



```

A = [1 -1; 1 1];
b = [atan2(r12 ,r22); 0];
sol = A^-1*b;
Alpha = sol(1);
Gama = sol(2);
end

```

```

ALPHA = Alpha * 180/pi; %Convers o de radianos em graus
BETA = Beta * 180/pi; %Convers o de radianos em graus
GAMA = Gama * 180/pi; %Convers o de radianos em graus

```

A.2. Função para Geração da População Aleatória

```

function chrom=crtrp(Nind,FieldDR)
% A random real value matrix is created coerced by upper and
% lower bounds
clc
%Nind = 10;
%FieldDR = [-165 -110 -110 -160 -120 -400; 165 110 70 160 120 400];
%FieldDR = [0 0 -60 -30 -90 -40; 60 60 0 30 20 40];
Nvar = size(FieldDR,2);
aux = rand(Nind,Nvar);
m=[-1 1]*FieldDR;
ublb=ones(Nind,1)*m;
lb=ones(Nind,1)*FieldDR(1,:);
chrom=ublb.*aux+lb;
% chrom(1,1) = 30;
% chrom(1,2) = 40;
% chrom(1,3) = -50;
% chrom(1,4) = 0;
% chrom(1,5) = -75;
% chrom(1,6) = -30;

% chrom(1,1) = 60;
% chrom(1,2) = 40;
% chrom(1,3) = 70;
% chrom(1,4) = -170
% chrom(1,5) = -140;
% chrom(1,6) = 120;

```

```
% chrom(1,7) = 300;
```

```
% chrom=round(chrom);
```

A.3. Função Objetivo

```
function y = myFitness(q,xf,qi)
```

```
%f1 - Minimiza o do erro de posição
```

```
f1 = (sqrt((xf(1) - p(1))^2 + (xf(2) - p(2))^2 + ...  
      (xf(3) - p(3))^2));
```

```
%f2 - Minimiza o do erro de orientação
```

```
f2 = (sqrt((GAMA - xf(4))^2 + (BETA - xf(5))^2 + ...  
      (ALPHA - xf(6))^2));
```

```
%f3 - Minimiza o do Deslocamento angular 6DoF
```

```
f3 = sqrt((q(1)-qi(1))^2 + (q(2)-qi(2))^2 + ...  
      (q(3)-qi(3))^2 + (q(4)-qi(4))^2 + ...  
      (q(5)-qi(5))^2 + (q(6)-qi(6))^2);
```

```
% f3 - 6DoF IRB120 Ponderada
```

```
f3 = sqrt(0.401*(q(1)-qi(1))^2 + 0.401*(q(2)-qi(2))^2 + ...  
      0.175*(q(3)-qi(3))^2 + 0.01*(q(4)-qi(4))^2 +  
      0.01*(q(5)-qi(5))^2 + 0.005*(q(6)-qi(6))^2);
```

```
% f3 - 6DoF IRB6700 Ponderada
```

```
f3 = sqrt(0.421*(q(1)-qi(1))^2 + 0.421*(q(2)-qi(2))^2 + ...  
      0.1*(q(3)-qi(3))^2 + 0.02*(q(4)-qi(4))^2 + ...  
      0.02*(q(5)-qi(5))^2 + 0.01*(q(6)-qi(6))^2);
```

```
% f3 - 7DoF Ponderada
```

```
f3 = sqrt((q(1)-qi(1))^2 + (q(2)-qi(2))^2 + ...  
      (q(3)-qi(3))^2 + (q(4)-qi(4))^2 + (q(5)-qi(5))^2 + ...  
      (q(6)-qi(6))^2 + ... (q(7)-qi(7))^2);
```

```
% f3 - 7DoF IRB120 Ponderada
```

```
f3 = sqrt(0.25*(q(1)-qi(1))^2 + 0.25*(q(2)-qi(2))^2 + ...  
      0.109*(q(3)-qi(3))^2 + 0.006*(q(4)-qi(4))^2 + ...  
      0.006*(q(5)-qi(5))^2 + 0.003*(q(6)-qi(6))^2 + ...  
      0.375*(q(7)-qi(7))^2);
```

```
% f3 - 7DoF IRB6700 Ponderada
```

```
f3 = sqrt(0.258*(q(1)-qi(1))^2 + 0.258*(q(2)-qi(2))^2 + ...  
      0.061*(q(3)-qi(3))^2 + 0.014*(q(4)-qi(4))^2 + ...
```

```

0.014*(q(5)-qi(5))^2 + 0.007*(q(6)-qi(6))^2 + ...
0.387*(q(7)-qi(7))^2);

```

```
F.O = y1 + y2 + y3;
```

A.4. Restrições

```

function [c, c_eq] = myConstraint(q,xf)
%RASTRIGINSFCN Compute the "Rastrigin" function.

erro = sqrt((xf(1) - p(1))^2 + (xf(2) - p(2))^2 + (xf(3) - p(3))^2);
resultado = [p(1), p(2), p(3), ALPHA, BETA, GAMA];
%%% Aqui s o efetivamente as retri es %%%
c_eq = [ sqrt((xf(1) - p(1))^2);
        sqrt((xf(2) - p(2))^2);
        sqrt((xf(3) - p(4))^2);
        sqrt((GAMA - xf(4))^2);
        sqrt((BETA - xf(5))^2);
        sqrt((ALPHA - xf(6))^2)];

c = [];
end

```

A.5. Algoritmo Genético

```

tic
nind = 25;
%Limites de junta IRB120
range = [-165 -110 -110 -160 -120 -400;
         165 110 70 160 120 400];
%Limites de junta IRB120 7DoF
range = [-165 -110 -110 -160 -120 -400 -25000;
         165 110 70 160 120 400, 2500];
%Limites de junta IRB6700 6DoF
range = [-170,-65,-180, -300, -130, -360;
         170, 85, 70, 300, 130, 360];
%Limites de junta IRB6700 7DoF
range = [-170,-65,-180, -300, -130, -360, -2500;
         170, 85, 70, 300, 130, 360, 2500];

```

```

%Posição Inicial
qi = [0; 0; 0; 0; 0; 0; 0];
%Limites Superiores e Inferiores %IRB120
lb = [-165,-110,-110, -160, -120, -400];
ub = [165,110,70, 160, 120, 400];
%Limites Superiores e Inferiores %IRB120 7DoF
lb = [-165,-110,-110, -160, -120, -400, -1000];
ub = [165,110,70, 160, 120, 400, 1000];
%Limites Superiores e Inferiores %IRB6700
lb = [-170,-65,-180, -300, -130, -360];
ub = [170, 85, 70, 300, 130, 360];
%Limites Superiores e Inferiores %IRB6700 7DoF
lb = [-170,-65,-180, -300, -130, -360, -2500];
ub = [170, 85, 70, 300, 130, 360, 2500];
%Definição dos parâmetros do AG
xf = [400,200,700,0,0,0];
pop = crtrp(nind, range)
Fitness = @(pop) myFitness(pop, xf, qi);
Constraint = @(pop) myConstraint(pop, xf);
opts = gaoptimset('PopulationSize', 25, ...
'InitialPopulation', pop, 'Generations', 40, ...
'TolCon', 1, 'CrossoverFcn', @crossoverarithmetic, ...
'CrossoverFraction', 0.5, ...
'MutationFcn', @mutationgaussian, ...
'SelectionFcn', @selectionroulette, ...
'MigrationFraction', 0.2, 'EliteCount', 2);
%Chamada do AG
[th, fval, exitflag] = ga(Fitness, 6, [], [], [], [], ...
lb, ub, Constraint, opts)
toc

```

B. Controle de Trajetória no MatLab

```
%variáveis de juntas inicial
q0 = [0,0,0,0,0,0,0,0];
%velocidade das juntas inicial
v0 = [0,0,0,0,0,0,0,0];
%aceleração das juntas inicial
ac0 = [0,0,0,0,0,0,0,0];
%Pose final determinada pelo AG
q1 = [-11.18,24.39,-28.64,-0.99,-85.75,11.93,277.86];
%velocidade das juntas final
v1 = [0,0,0,0,0,0,0,0];
%aceleração das juntas final.
ac1 = [0,0,0,0,0,0,0,0];
%instante inicial
t0 = 0;
%tempo da trajetória 10s
tf = 100;

%Resolução do polinômio de quinta ordem
t = linspace(t0,tf,1*(tf-t0));
c = ones(size(t));
M = [ 1 t0 t0^2 t0^3 t0^4 t0^5;
0 1 2*t0 3*t0^2 4*t0^3 5*t0^4;
0 0 2 6*t0 12*t0^2 20*t0^3;
1 tf tf^2 tf^3 tf^4 tf^5;
0 1 2*tf 3*tf^2 4*tf^3 5*tf^4;
0 0 2 6*tf 12*tf^2 20*tf^3];
%%
b = [q0(1) v0(1) ac0(1) q1(1) v1(1) ac1(1);
q0(2) v0(2) ac0(2) q1(2) v1(2) ac1(2);
q0(3) v0(3) ac0(3) q1(3) v1(3) ac1(3);
q0(4) v0(4) ac0(4) q1(4) v1(4) ac1(4);
q0(5) v0(5) ac0(5) q1(5) v1(5) ac1(5);
q0(6) v0(6) ac0(6) q1(6) v1(6) ac1(6);
q0(7) v0(7) ac0(7) q1(7) v1(7) ac1(7)];
b';
a = inv(M)*b';
```

```

for i = 1:7
    qd = a(1,i).*c + a(2,i).*t + a(3,i).*t.^2 + ...
        a(4,i).*t.^3 + a(5,i).*t.^4 + a(6,i).*t.^5;
    vd = a(2,i).*c + 2*a(3,i).*t + 3*a(4,i).*t.^2 + ...
        4*a(5,i).*t.^3 + 5*a(6,i).*t.^4;
    ad = 2*a(3,i).*c + 6*a(4,i).*t + ...
        12*a(5,i).*t.^2 + 20*a(6,i).*t.^3;

    %Velocidades individuais de cada junta
    for k = 1:100
        vdj(i,k) = [vd(k)]
    end
end

```

C. MatLab Integrado com Simuladores

C.1. Integração MatLab - RobotStudio

```
function varargout = RBS2(varargin)

function pushbutton1_Callback(hObject, eventdata, handles)
%IP = get(handles.edit1, 'String')
tc = tcpip('127.0.0.1', 55000, 'NetworkRole', 'client');
fopen(tc);
message=fread(tc);
char(message)';
set(handles.text1, 'String', char(message)');
handles.tc=tc
guidata(hObject, handles)

function pushbutton4_Callback(hObject, eventdata, handles)
tic %Inicia contagem de tempo
clc; %limpa a tela
tc = handles.tc

%Posição Efetuador
Xc = str2double(get(handles.edit9, 'String'));
Yc = str2double(get(handles.edit10, 'String'));
Zc = str2double(get(handles.edit11, 'String'));
alpha = str2double(get(handles.edit12, 'String'));
beta = str2double(get(handles.edit13, 'String'));
gama = str2double(get(handles.edit14, 'String'));

%AG
%Define o tamanho da população inicial
nind = 25;
%Limites de junta IRB120
range = [-165 -110 -110 -160 -120 -400;
         165 110 70 160 120 400];
%Limites de junta IRB6700
range = [-170, -65, -180, -300, -130, -360;
         170, 85, 70, 300, 130, 360];
```

```

%Posição Inicial
qi = [0; 0; 0; 0; 0; 0];
%Limites Superiores e Inferiores %IRB120
lb = [-165,-110,-110, -160, -120, -400];
ub = [165, 110, 70, 160, 120, 400];
%Limites Superiores e Inferiores %IRB6700
lb = [-170,-65,-180, -300, -130, -360];
ub = [170, 85, 70, 300, 130, 360];
xf = [Xc, Yc, Zc, alpha, beta, gama];
pop = crtrp(nind, range)
Fitness = @(pop) myFitness(pop, xf, qi);
Constraint = @(pop) myConstraint(pop, xf);
opts = gaoptimset('PopulationSize', 20,
'InitialPopulation', pop, 'Generations', 50, ...
'TolCon', 1, 'CrossoverFcn', @crossoverarithmetic, ...
'CrossoverFraction', 0.5, ...
'MutationFcn', @mutationgaussian, ...
'SelectionFcn', @selectionroulette, ...
'MigrationFraction', 0.2, 'EliteCount', 2);
[th, fval, exitflag] = ga(Fitness, 6, [], [], [], [],
lb, ub, Constraint, opts)
AN = th;
toc
end
D1=[S1 abs(AN(1)) S2 abs(AN(2)) S3 abs(AN(3))
S4 abs(AN(4)) S5 abs(AN(5)) S6 abs(AN(6))];
fwrite(tc, D1);
pause(1);
end

```

C.2. Integração MatLab - Coppelia

```

%Instanciando Conexão
sim=remApi('remoteApi');
sim.simxFinish(-1);
clientID=sim.simxStart('127.0.0.1',19999,true,true,5000,5);
sim.simxSynchronous(clientID,true);
sim.simxGetSimulationStepDone...
(client.simxDefaultSubscriber(simulationStepDone_CB));

```



```

sim.simxStartSimulation(client.simxDefaultPublisher());

%Checa a conexão com o Coppelia
if (clientID > -1)
    %plota em caso de sucesso
    disp('Connected to remote API server');

    %Cria o objeto das juntas tal qual no arquivo "vrep2"
    h=[0,0,0,0,0,0,0];
    [r,h(1)]= sim.simxGetObjectHandle...
        (clientID, 'joint_1', sim.simx_opmode_blocking);
    [r,h(2)]= sim.simxGetObjectHandle
        (clientID, 'joint_2', sim.simx_opmode_blocking);
    [r,h(3)]= sim.simxGetObjectHandle
        (clientID, 'joint_3', sim.simx_opmode_blocking);
    [r,h(4)]= sim.simxGetObjectHandle
        (clientID, 'joint_4', sim.simx_opmode_blocking);
    [r,h(5)]= sim.simxGetObjectHandle
        (clientID, 'joint_5', sim.simx_opmode_blocking);
    [r,h(6)]= sim.simxGetObjectHandle
        (clientID, 'joint_6', sim.simx_opmode_blocking);
    [r,h(7)]= sim.simxGetObjectHandle
        (clientID, 'joint_7', sim.simx_opmode_blocking);

    %Controle da trajetória "quebrada" em 100 velocidades
    for i = 1:100
        %Coleta as velocidades de cada uma das 6 juntas
        v1 = vdj(1,i)
        v2 = vdj(2,i)
        v3 = vdj(3,i)
        v4 = vdj(4,i)
        v5 = vdj(5,i)
        v6 = vdj(6,i)
        v7 = vdj(7,i)

        %Possibilidade 1 Controle de Posição das Juntas
        sim.simxSetJointTargetPosition...
            (clientID, h(1), joint_pos1(1), sim.simx_opmode_streaming);

```

```

sim.simxSetJointTargetPosition ...
    ( clientID ,h(2) ,joint_pos1 (2) ,sim.simx_opmode_streaming );
sim.simxSetJointTargetPosition ...
    ( clientID ,h(3) ,joint_pos1 (3) ,sim.simx_opmode_streaming );
sim.simxSetJointTargetPosition ...
    ( clientID ,h(4) ,joint_pos1 (4) ,sim.simx_opmode_streaming );
sim.simxSetJointTargetPosition ...
    ( clientID ,h(5) ,joint_pos1 (5) ,sim.simx_opmode_streaming );
sim.simxSetJointTargetPosition ...
    ( clientID ,h(6) ,joint_pos1 (6) ,sim.simx_opmode_streaming );
sim.simxSetJointTargetPosition
    ( clientID ,h(7) ,joint_pos1 (7) ,sim.simx_opmode_streaming );

```

%Possibilidade 2 Controle de Velocidade das Juntas

```

sim.simxSetJointTargetVelocity ...
    ( clientID ,h(1) ,v1 ,sim.simx_opmode_streaming );
sim.simxSetJointTargetVelocity ...
    ( clientID ,h(2) ,v2 ,sim.simx_opmode_streaming );
sim.simxSetJointTargetVelocity ...
    ( clientID ,h(3) ,v3 ,sim.simx_opmode_streaming );
sim.simxSetJointTargetVelocity ...
    ( clientID ,h(4) ,v4 ,sim.simx_opmode_streaming );
sim.simxSetJointTargetVelocity ...
    ( clientID ,h(5) ,v5 ,sim.simx_opmode_streaming );
sim.simxSetJointTargetVelocity ...
    ( clientID ,h(6) ,v6 ,sim.simx_opmode_streaming );
sim.simxSetJointTargetVelocity ...
    ( clientID ,h(7) ,v7 ,sim.simx_opmode_streaming );
%Sincroniza Timestep da Simula o
    pause(0.01);
end

else
    %Caso a conex o falhe
    disp('Failed connecting to remote API server ');
end

sim.delete(); % call the destructor!
disp('Program ended ');

```

D. Código RobotStudio

```
MODULE TCPIP
```

```
VAR socketdev server;  
VAR socketdev client;  
VAR bool connectionStatus := FALSE;  
VAR string message;  
VAR num junta {12};  
VAR rawbytes data;  
VAR num dados {1024};  
VAR jointtarget ...  
Home:=[[0,0,0,0,0,0],...  
[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]]];  
VAR num float;  
VAR num float1;  
VAR num contador:=0;  
VAR num transmission:=0;
```

```
PROC main()
```

```
MoveAbsJ [[0,0,0,0,0,0],...  
[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],...  
v1500,z1,tool0\WObj:=wobj0;  
Conn;  
SocketClose server;  
MoveAbsJ [[0,0,0,0,0,0],...  
[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],...  
v1500,z1,tool0\WObj:=wobj0;  
DadosMovement;  
ENDPROC
```

```
PROC Conn ()
```

```
SocketCreate server;  
SocketBind server, "127.0.0.1",55000;  
SocketListen server;  
SocketAccept server,client;  
connectionStatus := TRUE;
```

```

SocketSend client \Str := "Conectado";
WHILE connectionStatus DO
    DadosMovement;
ENDWHILE
ENDPROC

```

```

PROC DadosMovement()
    SocketReceive client \Data:= dados...
    \ReadNoOfBytes:=1024 \Time:= WAIT.MAX;
    stringHandler(dados);
    MoveAbsJ[[junta {2},junta {4},junta {6},...
    junta {8},junta {10},junta {12}],...
    [9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],...
    v1500 , z1 , too10 \WObj:= wobj0;
ENDPROC

```

```

PROC stringHandler(num Data2Process {*})
    FOR i FROM 1 TO 12 STEP 2 DO
        IF Data2Process{i} = 1 THEN
            junta{i + 1} := -(Data2Process{i + 1});
            !junta{i + 1} := -(Data2Process{i + 1});
        ELSEIF Data2Process{i} = 2 THEN
            junta{i + 1} := (Data2Process{i + 1});
            !junta{i + 1} := (Data2Process{i + 1});
        ENDIF
    ENDFOR
ENDPROC

```

```

ENDMODULE

```

E. Links Vídeos de Simulação

- <https://youtu.be/jfRJ8bL0RUQ>
- <https://youtu.be/-EwxK70GNzs>
- <https://youtu.be/PrIBlwSWUk>
- <https://youtu.be/Ff8Gh4Izz0>
- <https://youtu.be/pJxV7vYrWtM>