TRABALHO DE GRADUAÇÃO

MODELAGEM, IDENTIFICAÇÃO E CONTROLE PARA O ROBÔ MANIPULADOR UR3

PEDRO SAMAN D. N. CESARINO

Brasília, Dezembro de 2020

ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

MODELLING, IDENTIFICATION AND CONTROL FOR THE ROBOTIC MANIPULATOR UR3

PEDRO SAMAN D. N. CESARINO

ORIENTADOR: PROF. GEOVANY ARAÚJO BORGES

TRABALHO DE GRADUAÇÃO EM ENGENHARIA
DE CONTROLE E AUTOMAÇÃO

PUBLICAÇÃO: FT. TG-nº 09

BRASÍLIA/DF: DEZEMBRO - 2020

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

MODELAGEM, IDENTIFICAÇÃO E CONTROLE PARA O ROBÔ MANIPULADOR UR3

PEDRO SAMAN D. N. CESARINO

TRABALHO DE GRADUAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM ENGENHARIA.

APROVADA POR:

_____

Prof. Geovany Araújo Borges – ENE/Universidade de Brasília
Orientador

_____

Prof. Henrique Cezar Ferreira – ENE/Universidade de Brasília
Examinador Interno

_____

Prof. João Yoshiyuki Ishihara – ENE/Universidade de Brasília
Examinador Interno

BRASÍLIA, 09 DE DEZEMBRO DE 2020.

FICHA CATALOGRÁFICA

CESARINO, PEDRO SAMAN D. N.
Modelagem, Identificação e Controle para o Robô Manipulador UR3 [Distrito Federal] 2020.
xiv, 88p., 210 x 297 mm (ENE/FT/UnB, Bacharel em Engenharia, Engenharia de Controle e Automação, 2020).
Trabalho de Graduação – Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. Modelagem Dinâmica	2. Identificação
3. Controle Dinâmico	4. Manipuladores Robóticos
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

CESARINO, PEDRO SAMAN D. N. (2020). Modelagem, Identificação e Controle para o Robô Manipulador UR3 . Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT. TG-nº 09, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 88p.

CESSÃO DE DIREITOS

AUTOR: Pedro Saman D. N. Cesarino
TÍTULO: Modelagem, Identificação e Controle para o Robô Manipulador UR3 .
GRAU: Bacharel em Engenharia ANO: 2020

Pedro Saman D. N. Cesarino

Departamento de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

Dedicatória

Aos meus pais, José Aprigio e Maria Saman, que sempre me apoiaram e permitiram que eu pudesse me dedicar aos estudos.

AGRADECIMENTOS

Gostaria de agradecer primeiramente: aos meus pais que sempre me deram um ambiente acolhedor e as melhores condições para que eu estudasse sem preocupações extras, às minhas irmãs Catarina Saman e Erika Saman que cresceram ao meu lado e ao meu primo Arthur Cesarino que considero um irmão e conheço desde que nasci por sempre me ajudar e ser um grande companheiro da vida. Sem vocês eu não chegaria tão longe.

Também quero agradecer aos professores que eu tive durante minha vida e que me transmitiram incontáveis conhecimentos. Em especial o professor Geovany Araújo Borges que me aceitou como aluno de trabalho de graduação e ao Rafael Ramos por ter me ajudado muito durante esse um ano e meio de projeto.

Agradeço a todos os meus amigos que com certeza me ajudaram a chegar onde eu estou: André Marques, Augusto Freitas, Bernardo Carsten, Caio Oliveira, Camila Sidersky, Daniel Bemerguy, Fernando Sobral, Ian Nery, Khalil Carsten, Marcelo Araújo, Rafael Deflon, Rafael Torres, Renato Nobre, Ricardo Nunes e Rodrigo Navarro.

RESUMO

A utilização de robôs manipuladores em situações de cooperação com humanos ou outros robôs requer um extenso conhecimento sobre a sua dinâmica para que seja possível garantir algum grau de segurança para as pessoas ou equipamentos envolvidos. Além das equações que relacionam as forças aplicadas com o movimento obtido, é preciso conhecer os parâmetros dinâmicos do robô caso não sejam fornecidos, para, por fim, ser possível realizar controladores que busquem garantir tal segurança. Neste trabalho foi utilizada a dinâmica de Newton-Euler para escrever as equações dinâmicas do robô manipulador UR3. Então foi feita a simplificação das equações em formato simbólico para ser possível separar os termos lineares dos não lineares para finalmente realizar o processo de identificação dos parâmetros dinâmicos desconhecidos utilizando experimentos feitos pelo Pedro Garcia. Por fim foram apresentadas duas arquiteturas de controle cinemático de trajetória que levam em consideração a dinâmica do robô: Controle do ângulo de rotação das juntas e controle de posição e orientação da ferramenta. Ambos os controladores foram testados apenas em ambiente de simulação sendo que o primeiro obteve resultados satisfatórios, enquanto o segundo obteve problemas com a componente de controle de orientação. A partir destes controladores de posição será possível criar novas arquiteturas para realizar o controle de torque nas juntas e de força de contato da ferramenta em trabalhos futuros, possibilitando a cooperação entre robôs ou entre humanos e robôs de forma mais segura.

Palavras Chave: Modelagem Dinâmica, Identificação, Controle Dinâmico, Manipuladores Robóticos

ABSTRACT

To be able to use robotic manipulators in situations that require human-robot or robot-robot cooperation is necessary to have an extensive knowledge about its dynamic as to be possible to ensure some degree of safety to the people or equipment involved. Beyond the equations that connect the applied force with the obtained movement it's necessary to know the dynamic parameters of the robot if those are not provided to finally be able to design controllers that seek to ensure the said safety. In this work it was utilized the Newton-Euler parametrization to write the UR3 robot dynamic equations, then was made the symbolic simplification to separate the linear terms form the nonlinear for then perform the unknown parameters identification process using Pedro Garcia's experimental results. Lastly, two different controllers architecture was presented: joint angle position control and tool position and orientation control. Both controllers ware tested only in simulation environment. The first one achieved good results while the last one had problems with the orientation component. From those controllers it's possible to design a joint torque controller and a tool contact force controller enabling a safer cooperation between humans and robots.

Key Words: Dynamic Modelling, Identification, Dynamic Control, Robotic Manipulators

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Gregos

ξ	Resíduo
Θ	Matriz de parâmetros

Símbolos Latinos

$\mathcal{F}$	Vetor de Força-torque
J	Função de custo
Y	Matriz de regressores

Subscritos

i	Identificação do número do link ou junta

Sobrescitos

X^T	Matriz transposta de X
X^{-1}	Matriz inversa de X
X^{-T}	Matriz inversa da transposta de X
$\ddot{y}$	A segunda derivada da de uma função $y = f(x)$
$\dot{y}$	A primeira derivada da de uma função $y = f(x)$

Siglas

K_i	Ganho da integral do erro de posição	
K_p	Ganho do erro de posição	
K_v	Ganho do erro de velocidade	
R^i_{i+1}	Rotação do sistema de coordenadas i+1 para o i	
τ	Torque	[N.m]
c(q)	Cosseno de um ângulo q	
DH	Denavit Hartenberg	
kg	Quilograma	[kg]
LARA	Laboratório de Automação e Robótica	
m	Metro	[m]
PID	Proporcional Integral Derivativo	
ROS	Robot Operating System	
RPY	Sistema de orientação Roll Pitch Yaw	
s(q)	Seno de um ângulo q	
t	Instante de tempo	[segundos]

1 INTRODUÇÃO

A robótica é uma área de conhecimento humano que envolve diversas áreas da engenharia, como teoria de controle, sensoriamento, tratamento de dados, cinemática e dinâmica de corpos, instrumentação etc. e por isso é tão complexa e fascinante. Existem tipos de robôs com arquiteturas diversas: os que tentam imitar partes humanas como braços, pernas, mãos, os que tentam imitar animais como os humanoides, quadrúpedes ou em formato de serpente, os que são veículos ou drones, possuindo variados objetivos como se locomover em ambientes acidentados, voar, nadar, manipular objetos, realizar soldas, realizar pinturas, entreter pessoas [1]. A variedade de atividades que robôs podem fazer é grande, sendo cada vez mais necessárias com a crescente demanda por produtos industrializados [2] e seu uso médico [3].

Nas últimas décadas, a robótica tem avançado consideravelmente, fazendo presente no nosso dia a dia. Porém é na indústria que essa tecnologia mais causa impacto, automatizando processos e garantindo mais segurança. De fato, robôs conseguem realizar tarefas repetitivas e de alta precisão com menos falhas e de forma mais rápida que humanos conseguem. Mesmo assim, muitos desses robôs ainda são pré-programados e raramente interagem diretamente com humanos, precisando ficar isolados em áreas de segurança para evitar acidentes.

Por outro lado, é interessante que humanos façam parte do processo produtivo. Seja por sua capacidade de adaptar a linha de produção de forma rápida para fazer um produto personalizado ou pela sua capacidade cognitiva, que robôs, de forma geral, ainda não possuem.

Neste capítulo introdutório vai ser dado uma pequena motivação por trás do trabalho, explicação dos softwares e ambientes de desenvolvimento utilizados e uma visão geral dos próximos capítulos.

1.1 A COOPERAÇÃO HUMANO-ROBÔ

Uma forma de tornar o processo mais flexível e ao mesmo tempo sem perder as características desejáveis de um processo automático é a integração do humano no processo produtivo na forma de cooperação humano-robô em que o homem ajuda com a sua capacidade cognitiva e capacidade motora flexível enquanto a máquina faz o trabalho pesado, repetitivo e que necessita de alta precisão.

Uma grande vantagem de utilizar tecnologias que permitam a interação segura entre humanos e robôs é permitir que pessoas continuem trabalhando em ambientes onde existam

processos automáticos acontecendo sem a necessidade de parar tais processos. Por outro lado, os trabalhadores precisariam se acostumar com robôs trabalhando ao seu lado e aprender a utilizar tal cooperação da melhor forma possível.

Além da justificativa industrial, outra aplicação desta cooperação segura é oferecendo suporte para uma pessoa debilitada que não possua mais a capacidade de fazer algo sozinha, como um idoso a levantar da cama ou no auxílio de fisioterapeutas na recuperação de pacientes com dificuldades motoras [4].

O grande desafio da cooperação humano-robô é garantir um ambiente seguro e confortável para o trabalhador, ou paciente, especialmente quando existe a necessidade de contato físico entre os dois [5]. Outras dificuldades incluiriam o planejamento de tarefas e os protocolos de segurança.

Para que essa cooperação seja possível é necessário que os robôs não só tenham a capacidade de perceber o ambiente em sua volta para evitar colisões, rastrear objetos, detectar a intenção humana, etc. mas também consigam trabalhar em contato direto com humanos ou outros robôs, sem que ocorram acidentes. Para obter essas características é necessário o uso de sensores, como câmeras, juntamente com técnicas de controle que levem este problema em consideração.

O número de pesquisas relacionadas com a cooperação homem-robô vem crescendo bastante nas últimas décadas [6], destacando a relevância e interesse no desenvolvimento de tecnologias que permitam maior cooperação entre nós e as máquinas de forma segura e viável [7].

1.2 O MANIPULADOR UR3

O manipulador robótico UR3 é um robô de seis graus de liberdade rotativos, ou seja, ele tem seis juntas capazes de realizar movimentos rotatórios. Ele é um robô comercial vendido pela *Universal Robots* e existem diversos modelos com arquitetura parecida. Esse robô é indicado para realizar tarefas que não necessitem de forças elevadas, carga de trabalho máxima indicada de 3 kg, por isso ele é usado principalmente para soldagem, aparafusamento, *pick and place* de peças leves, etc.

O robô foi projetado para ser usado por técnicos de forma bem intuitiva e rápida, por isso a fabricante não permite realizar qualquer tipo de comando. Por exemplo, não é possível enviar um comando direto de corrente para as juntas, mas apenas uma velocidade e aceleração angulares de referência para cada motor. Além disso, a forma de programação dele é feita via uma linguagem própria da empresa a *URScript*. A partir dela é possível ler os diversos sensores e mandar os comandos de velocidade e aceleração.

Na figura 1.1 podemos ver o UR3 com uma garra instalada em sua última junta. Ele foi instalado no Laboratório de Automação e Robótica (LARA) e aparafusado à mesa. Justamente por ser um robô comercial, a empresa não informa totalmente as informações internas de construção do robô. Não fornecendo, por exemplo, a equação dinâmica ou então a matriz de inercia de cada junta.

Figura 1.1 – Foto do manipulador robótico UR3 instalado no LARA.

1.3 OBJETIVOS

Apesar de fornecer facilidades para implementação de aplicações para o robô, como aplicações pré-programadas e a possibilidade de aprendizado online, ou seja manualmente mostrar um trajeto para o robô realizar, a *Universal Robotics* não disponibiliza dados importantes para o desenvolvimento de aplicações avançadas, como controle complacente ou controle por impedância que seriam um bom ponto de partida para o desenvolvimento de aplicações de cooperação humano-robô.

Para realizar tais controladores é preciso que: o modelo dinâmico do robô seja levantado, os parâmetros dinâmicos desconhecidos sejam identificados, a estrutura dos controladores seja programada e, antes de se aplicar no robô real, simulações sejam feitas para garantir o correto funcionamento do projeto.

Para isso, o primeiro objetivo desse trabalho foi verificar as informações disponíveis sobre a cinemática do robô. O segundo passo foi desenvolver um algoritmo para definir e simplificar as equações dinâmicas, relação entre movimento e as forças necessárias para realizá-lo, de forma que fosse genérico o suficiente para que fosse possível ser utilizado em outros robôs com poucas modificações. Então, fazer a identificação dos parâmetros dinâmicos desconhecidos para que seja possível fazer simulação do robô para finalmente ser possível criar as arquiteturas de controladores e definir os seus parâmetros.

No capítulo 2 será feita uma introdução teórica de todos os procedimentos realizados no trabalho. No capítulo 3 será feita a explicação e serão mostrados os sistemas e códigos feitos para os implementar. O capítulo 4 será focado na apresentação dos resultados experimentais obtidos e na sua análise técnica. Por fim, o capítulo 5 será composto pela conclusão do estudo e sugestões de temas para continuar o trabalho.

1.4 SOFTWARE

Durante o desenvolvimento do trabalho foram usados os Softwares Matlab e Maple além da biblioteca do Robot Operating System (ROS). A seguir será feita uma breve descrição sobre eles, seus usos e como foram utilizados.

1.4.1 Matlab

O Matlab [8] é uma plataforma de programação feita para ser usada por cientistas e engenheiros de forma intuitiva e rápida para que o foco da programação seja a resolução do problema e não a forma de implementação. Neste programa é possível implementar códigos, importar e analisar massas de dados, realizar *debug* de código, implementar sistemas etc. Ele possui integração com diversas outras ferramentas, como Arduino e ROS, além de ter diversas *Toolbox* prontas para serem usadas, como as de Identificação, Controle PID e Redes Neurais.

Esse software foi utilizado, na sua versão 2018b, neste trabalho para a implementação da maioria dos códigos e análise de dados pela familiaridade com o programa e pela possibilidade de fácil análise de código e erros de implementação que ele fornece. Outro motivo para sua utilização foi o Simulink [9]. Ele possibilita a implementação de sistemas em um ambiente computacional com interface gráfica por blocos permitindo a modelagem, simulação e análise em tempo real, permitindo a construção de um sistema de forma computacional antes de ser necessária a implementação física.

1.4.2 Maple

O Maple [10] é um software de computação simbólica matemática. Ele tem ferramentas de resolução e simplificação de expressões matemáticas extremamente sofisticadas. Assim como o Matlab, o Maple também é um ambiente de implementação de algoritmos e análise de dados, possuindo *Toolbox* de modelagem física e geração de códigos. Foi utilizada a sua poderosa biblioteca de simplificação de expressões simbólicas para diminuir e separar expressões dinâmicas geradas no Matlab.

1.4.3 ROS

O ROS [11] é um sistema de código aberto para operar robôs. Ele oferece a capacidade de implementar aplicações específicas de robótica fornecendo abstração de hardware, controle de periféricos, implementação de operações de baixo nível, transmissão de mensagens etc. O ROS tem a característica de ser flexível, altamente modular e ter a característica de ser *Plug and Play*, ou seja, existe a possibilidade de desenvolver uma aplicação para um simulador robótico, se for possível o uso de ROS nele, e, com poucas modificações, implementar para o robô físico. Neste trabalho foi utilizado um programa feito pelo aluno Rafael Ramos, via ROS, para enviar comandos e ler os sensores do UR3 para gerar os dados de identificação.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados os conceitos teóricos importantes que foram utilizados durante o desenvolvimento deste projeto. Começando pelos conceitos mais básicos de cinemática, representação de eixos e rotações, passando pela detalhada modelagem dinâmica de manipuladores e identificação de sistemas dinâmicos para, por fim, descrever as arquiteturas de controladores utilizadas.

2.1 MODELAGEM CINEMÁTICA

A cinemática é a ciência que estuda o movimento de corpos, ou seja sua posição, velocidade e aceleração, sem se preocupar com as forças necessárias para gerá-las. A cinemática inversa de um manipulador calcula quais ângulos de junta o robô precisa estar para chegar em uma posição e orientação desejadas para o efetuador terminal enquanto a cinemática direta faz o contrário, dados os ângulos ou posições das juntas qual a posição e orientação o efetuador está.

Uma das formas de se determinar tais modelos cinemáticos é utilizando a matriz de Denavit-Hartenberg (DH) do robô. Essa matriz é escrita a partir de quatro parâmetros para cada junta que descrevem a relação entre os links de um robô. Dois desses parâmetros descrevem o link em si e dois explicam como cada junta se conecta. Uma característica desse procedimento é que três desses parâmetros são constantes para dado robô, pois dependem de como ele foi construído enquanto o último parâmetro depende apenas do ângulo de rotação que uma junta realizou, então, apenas com os valores de rotação das juntas é possível descrever a cinemática de um manipulador.

Um passo importante na construção da matriz DH é a determinação da localização dos eixos de coordenada para cada junta do robô e do eixo fixo de referência. O livro do Craig [12] tem uma descrição detalhada de como escolher a localização de cada eixo. De forma geral o eixo zero é arbitrário, deve haver um sistema de eixos coordenados para cada junta do robô buscando orientar a componente z de tal forma que a rotação da junta seja em torno dele.

A figura 2.1 mostra a escolha das posições dos eixos de coordenada utilizado durante o trabalho para o robô UR3. Tanto a posição dos eixos quanto a tabela 2.1 com os parâmetros DH foram obtidos das informações dadas pela fabricante do UR3 em [13]. Os valores de distância estão em metros.

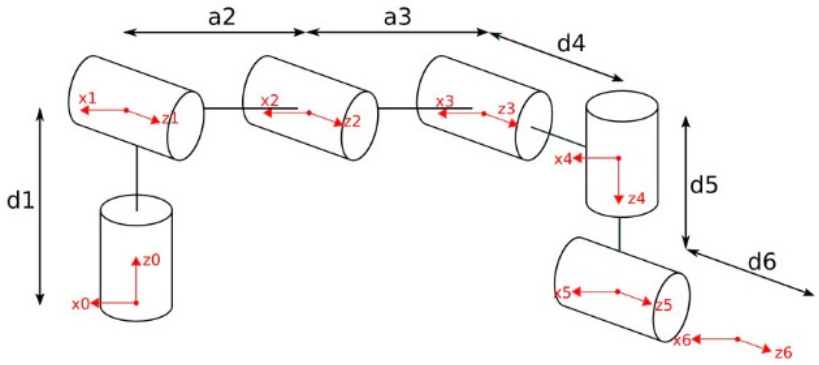

Figura 2.1 – Escolha da posição dos eixos de coordenada para as juntas do UR3.

Tabela 2.1 – Tabela com os Parâmetros de Denavit-Hartenberg do UR3

i	a_i	α_i	d_i	θ_i
1	0	$-\pi/2$	0,152	q_1
2	0,244	0	0	q_2
3	0,213	0	0	q_3
4	0	$-\pi/2$	0,112	q_4
5	0	$\pi/2$	0,085	q_5
6	0	0	0,082	q_6

2.1.1 Rotação entre Eixos

É necessário definir o que são as matrizes de rotação entre sistemas de eixos de coordenadas e a notação que será utilizada no trabalho [14]. Considere um vetor $v(x, y, z)$ qualquer definido inicialmente em um sistema $x_0y_0z_0$ e precisa ser representado em um segundo sistema $x_1y_1z_1$. Considere que houve as seguintes rotações do primeiro para o segundo sistema: γ graus em torno x, β graus em torno y e α em torno z. É possível realizar uma transformação de rotação entre os sistemas de eixos via a matriz de rotação mostrada em 2.1 em que c(q) significa cosseno de q e s(q) significa seno de q. Portanto temos: $v_1 = R_0^1 v_0$. Se houvesse um terceiro eixo de coordenadas que rotacionou em relação ao segundo via transformação R_1^2, poderíamos escrever v como: $v(x_2, y_2, z_2) = R_1^2 v_1 = R_1^2 R_0^1 v_0$.

$$R_0^1 = \begin{bmatrix} c(\alpha) & -s(\alpha) & 0 \\ s(\alpha) & c(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c(\beta) & 0 & s(\beta) \\ 0 & 1 & 0 \\ -s(\beta) & 0 & c(\beta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\gamma) & -s(\gamma) \\ 0 & s(\gamma) & c(\gamma) \end{bmatrix} \tag{2.1}$$

Em um manipulador com todas suas juntas rotativas, ou seja, existe apenas movimento rotatório entre seus links, é possível representar a rotação entre dois links usando apenas a rotação em torno de z, que é o movimento gerado pelo motor daquela junta ao rotacionar

um ângulo q, e a rotação em torno de x que é o ângulo fixo α_i da tabela de DH. A matriz de rotação entre duas juntas de um manipulador fica como na equação 2.2.

$$R_i^{i+1} = \begin{bmatrix} c(q) & -s(q) & 0 \\ s(q) & c(q) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\alpha) & -s(\alpha) \\ 0 & s(\alpha) & c(\alpha) \end{bmatrix} \tag{2.2}$$

2.1.2 Cinemática Direta

Para calcular a posição e orientação no espaço 3D do efetuador terminal partindo dos ângulos atuais das juntas de um robô manipulador basta calcular a matriz de transformação T, a partir da multiplicação de cada matriz de transformação T_i de cada frame do manipulador partindo da base até a ponta. Cada matriz de transformação é calculada como na equação 2.7 das quatro transformações mostradas de 2.3 até 2.6, para cada link do manipulador [14].

$$Rot_{z,\theta_i} = \begin{bmatrix} c(\theta_i) & -s(\theta_i) & 0 & 0 \\ s(\theta_i) & c(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

$$Trans_{z,d_i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

$$Trans_{x,a_i} = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

$$Rot_{x,\alpha_i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c(\alpha_i) & -s(\alpha_i) & 0 \\ 0 & s(\alpha_i) & c(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

$$T_i = Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i} \tag{2.7}$$

$$T_i = \begin{bmatrix} c(\theta_i) & -s(\theta_i)c(\alpha_i) & s(\theta_i)s(\alpha_i) & a_i c(\theta_i) \\ s(\theta_i) & c(\alpha_i)c(\theta_i) & -c(\theta_i)s(\alpha_i) & a_i s(\theta_i) \\ 0 & s(\alpha_i) & c(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

Substituindo as matrizes 2.3 até 2.6 na equação 2.7 para cada um dos seis links da tabela 2.1 e os multiplicando na ordem de T_1 à T_6 obtemos a matriz de transformação que representa a translação e rotação do último frame, onde fica a ferramenta, em relação a base.

Para se obter a posição p(x,y,z) basta fazer:

- $x = T(1,4)$
- $y = T(2,4)$
- $z = T(3,4)$

Já a orientação da ferramenta pode ser descrita pela matriz 3x3 interna à T na posição T(1:3,1:3) (primeiras 3 linhas e primeiras 3 colunas) que representa a rotação existente entre o frame da base e o frame do efetuador terminal. Existem diversas maneiras de se descrever tal rotação, entre elas: ângulos Z–Y–X de Euler, ângulos de rotação fixa X-Y-Z, também chamados de "*roll, pitch, yaw*" (RPY) e a representação por quatérnios. Neste trabalho foi usado, por conveniência, a representação RPY. A rotação pode ser determinada pela equação 2.9, e, usando a notação simplificada 2.10, podemos obter os ângulos β, α e γ resolvendo as equações 2.11, 2.55 e 2.13.

$${}^{A}_{B}R_{XYZ}(\alpha,\beta,\gamma) = \begin{bmatrix} c(\alpha)c(\beta) & c(\alpha)s(\beta)s(\gamma) - s(\alpha)c(\gamma) & c(\alpha)s(\beta)c(\gamma) + s(\alpha)s(\gamma) \\ s(\alpha)c(\beta) & s(\alpha)s(\beta)s(\gamma) + c(\alpha)c(\gamma) & s(\alpha)s(\beta)c(\gamma) - c(\alpha)s(\gamma) \\ -s(\beta) & c(\beta)s(\gamma) & c(\beta)c(\gamma) \end{bmatrix} \tag{2.9}$$

$${}^{A}_{B}R_{XYZ}(\alpha,\beta,\gamma) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.10}$$

$$\beta = Atan2(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}) \tag{2.11}$$

$$\alpha = Atan2(r_{21}/c\beta, r_{11}/c\beta) \tag{2.12}$$

$$\gamma = Atan2(r_{32}/c\beta, r_{33}/c\beta) \tag{2.13}$$

2.1.3 Cinemática Inversa

Já o problema da cinemática inversa é definido como: a partir de um ponto no espaço 3D desejado, quais ângulos de junta levam o efetuador terminal a tal localização?

Apesar de existir modos analítico algébrico e geométrico para se determinar as equações que resolvem esse problema foi usado a função *ikine* da *toolbox* de robótica do Matlab que, dado o modelo do robô definido usando o DH, calcula numericamente o vetor $p(x, y, z, roll, pitch, yaw)$. Esse método tem algumas desvantagens, por exemplo a necessidade de se definir um ponto de partida, uma escolha ruim pode inviabilizar a determinação do vetor p.

Outro problema é que não é possível escolher uma posição particular para o braço. Apesar de tais desvantagens, como a cinemática inversa seria utilizada apenas para gerar pontos para teste de congruência de resultados para o controle de posição, esse foi o método escolhido. Informações detalhadas de como usar a função pode ser achada no site da *Toolbox* [1].

Em modos gerais a função depende da definição de uma estrutura de corpo rígido que representa as ligações do robô usando a matriz DH, um ponto do espaço desejado e uma configuração de juntas inicial para o algoritmo começar a busca. A escolha desse ponto inicial impacta diretamente o resultado.

2.1.4 Planejamento de Trajetórias

Planejamento de trajetória consiste em definir o caminho que cada junta, ou então a ferramenta no efetuador terminal, vai seguir entre os pontos inicial e final definidos para uma dada tarefa. Ao se definir o caminho é, também, designado uma aceleração e uma velocidade.

Existem alguns métodos de se definir tal trajetória como polinômios cúbicos, polinômios cúbicos com pontos de passagem, polinômios de ordem superior, trajetos lineares, trajetos lineares com suavização parabólica. Neste trabalho foi utilizado esta última abordagem.

A suavização parabólica, exemplo na figura 2.2, [12], tem o objetivo de tornar uma trajetória inicialmente linear em um conjunto de três partes em que o meio é uma trajetória linear, o início é uma trajetória parabólica com curvatura positiva e o final é uma trajetória parabólica com curvatura negativa. Tornando, assim, o perfil de velocidade e posição contínuos no início e no final do movimento.

[1] <https://petercorke.com/toolboxes/robotics-toolbox/>

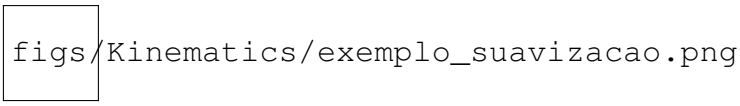

Figura 2.2 – Exemplo de suavização parabólica.

Para conseguir gerar a trajetória da forma proposta é preciso fazer as seguintes considerações:

- Durante as partes parabólicas a aceleração é constante;
- As partes parabólicas devem se conectar com a parte central linear;
- As partes parabólicas tem a mesma duração;
- A velocidade ao final da primeira parte parabólica é igual a velocidade da parte linear;
- É preciso definir os pontos final e inicial, aceleração na parte não linear e o tempo desejado para a trajetória;
- A aceleração deve ser suficientemente alta para que exista solução.

A equação 2.14 garante que a velocidade no final da primeira porção parabólica seja igual a velocidade na parte linear. Considerando que a velocidade inicial é zero e a aceleração é constante, a equação 2.15 define qual a posição no momento inicial da parte linear. Combinando as equações 2.14 e 2.15 e considerando que $x_h = \frac{x_f+x_0}{2}$ e $t_f = 2t_h$ temos a equação 2.16. Resolvendo essa equação para t_b temos a equação 2.17 havendo a restrição para $\ddot{x}_b$ igual 2.18. Quanto maior o valor da aceleração mais próximo de uma reta a trajetória se torna, quando 2.18 se torna uma igualdade não há parte linear e a trajetória se torna a junção de duas parábolas.

Assim sendo temos as equações 2.14 a 2.18, em que o ponto inicial x_0, o ponto final x_f, a aceleração da parte parabólica $\ddot{x}_b$, o tempo desejado t_f, posição central x_h, tempo central t_h, tempo do final e posição da primeira parte parabólica t_b e x_b.

$$\ddot{x}_b t_b = \frac{x_h - x_b}{t_h - t_b} \tag{2.14}$$

$$x_b = \int \dot{x} dt = \int \ddot{x}_b t dt = x_0 + \frac{\ddot{x}_b t_b^2}{2} \tag{2.15}$$

$$\ddot{x}_b t_b^2 - \ddot{x}_b t_f t_b + (x_f - x_0) \tag{2.16}$$

$$t_b = \frac{t_f}{2} - \frac{\sqrt{\ddot{x}_b^2 t_f^2 - 4\ddot{x}_b(x_f - x_0)}}{2\ddot{x}_b} \tag{2.17}$$

$$\ddot{x}_b \geq \frac{4(x_f - x_0)}{t_f^2} \tag{2.18}$$

Finalmente, é possível escrever a função da trajetória como o conjunto 2.19.

$$x(t) = \begin{cases} x_0 + \frac{\ddot{x}_b t^2}{2} & \text{para } 0 \leq t \leq t_b \\ x_0 + \ddot{x}_b t_b (t - \frac{t_b}{2}) & \text{para } t_b < t \leq (t_f - t_b) \\ x_f - \frac{\ddot{x}_b (t - t_f)^2}{2} & \text{para } (t_f - t_b) < t < t_f \end{cases} \tag{2.19}$$

Esse método gera perfis de posição, velocidade e aceleração como na figura 2.3 em que os dados de entrada foram iguais a tabela 2.2.

Tabela 2.2 – Tabela com parâmetros de entrada para a trajetória exemplo.

x_0	-0,112 m
x_f	0,112 m
t_f	25 s
$\ddot{x}_b$	0,004 m/s^2

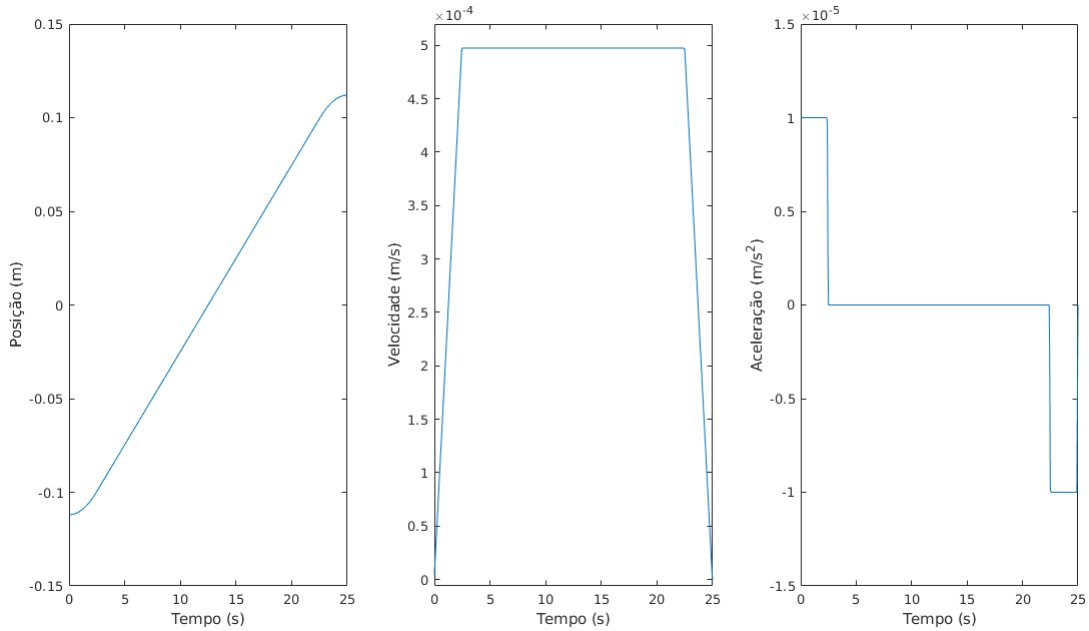

Figura 2.3 – Trajetória exemplo gerada pelo algoritmo de trajetória linear com suavização parabólica.

2.1.5 Jacobiana

A matriz jacobiana é uma matriz formada pelas derivadas parciais de primeira ordem de uma função vetorial. Em robótica essa matriz é usada para relacionar, pela expressão 2.20, a velocidade das juntas $\dot{q}$ de um manipulador e as velocidades angulares ω e lineares $\dot{p}$ do efetuador terminal. Essa matriz será necessária posteriormente no desenvolvimento de

controladores. Ela pode ser escrita em sua forma analítica como em 2.21 ou, como foi usada neste trabalho, em sua forma geométrica [15].

$$v_{(6x1)} = \begin{bmatrix} \dot{p} \\ \omega \end{bmatrix}_{(6x1)} = J(q)_{(6xn)} \dot{q}_{(nx1)} \tag{2.20}$$

$$J_F(x_1, ..., x_n) = \frac{\partial(F_1, ..., F_n}{\partial(x_1, ..., x_n)} \tag{2.21}$$

Para se determinar a Jacobiana Geométrica de um manipulador vamos dividir a matriz J em duas partes: $J_{p(3xn)}$ e $J_{o(3xn)}$, equação 2.22. Onde $J_{p(3xn)}$ representa a contribuição de $\dot{q}$ sobre a velocidade linear do efetuador terminal e $J_{o(3xn)}$ representa a contribuição de $\dot{q}$ sobre a velocidade angular.

$$\begin{bmatrix} \dot{p} \\ \omega \end{bmatrix}_{(6x1)} = \begin{bmatrix} J_{P(3xn)} \\ J_{O(3xn)} \end{bmatrix}_{(6xn)} \dot{q}_{(nx1)} \tag{2.22}$$

Como $n = 6$ e todas juntas do robô são de revolução, a jacobiana pode ser escrita da forma 2.23 com $J_{P(i)}$ sendo definido como 2.24 e $J_{O(i)}$ como 2.25.

$$J = \begin{bmatrix} J_{P1} & J_{P2} & J_{P3} & J_{P4} & J_{P5} & J_{P6} \\ J_{O1} & J_{O2} & J_{O3} & J_{O4} & J_{O5} & J_{O6} \end{bmatrix} \tag{2.23}$$

$$J_{Pi} = z_{i-1} \times (P - p_{i-1}) \tag{2.24}$$

$$J_{Oi} = z_{i-1} \tag{2.25}$$

Onde P é a posição do efetuador terminal com respeito ao frame fixo da base, p_{i-1} é a posição de cada junta com respeito ao frame da base e pode ser obtida dos três primeiros elementos da quarta coluna da matriz homogênea 2.7, z_{i-1} é o eixo em que ocorre a rotação de cada junta em respeito ao frame fixo e pode ser definida pegando os três primeiros valores da terceira coluna da matriz homogênea. Por fim i é o índice que varia de 1 a 6 e representa cada junta.

Um algoritmo realizando todos os cálculos para definição do valor numérico da jacobiana geométrica para o robô UR3 dependente um dado conjunto de valores dos ângulos das juntas foi desenvolvido em ambiente Matlab.

2.2 MODELAGEM DINÂMICA

A modelagem dinâmica consiste em escrever as equações que descrevem a relação entre as forças aplicadas em um corpo e o movimento resultante. A dinâmica direta parte do histórico do movimento e de uma força sendo aplicada e calcula a aceleração atual. Já a dinâmica inversa parte do conhecimento do movimento atual para calcular a força que foi aplicada. Neste trabalho foi utilizada a formulação clássica de Newton-Euler para escrever tais equações dinâmicas. Foi utilizada a convenção de variáveis apresentada da equação 2.26 até 2.44.

$$a_{c,i} = \text{aceleração linear do centro de massa do link i} \tag{2.26}$$
$$a_{e,i} = \text{aceleração linear do fim do link i, começo do link i+1} \tag{2.27}$$
$$\omega_i = \text{velocidade angular do frame i com respeito ao frame i} \tag{2.28}$$
$$\alpha_i = \text{aceleração angular do frame i com respeito ao frame i} \tag{2.29}$$
$$z_i = \text{eixo de rotação do frame i com relação ao frame i} \tag{2.30}$$
$$g_i = \text{aceleração da gravidade com relação ao frame i} \tag{2.31}$$
$$f_i = \text{força exercida pelo link i} \tag{2.32}$$
$$\tau_i = \text{torque exercido pelo link i} \tag{2.33}$$
$$f_e = \text{força externa} \tag{2.34}$$
$$\tau_e = \text{torque externo} \tag{2.35}$$
$$m_i = \text{massa do link i} \tag{2.36}$$
$$I_i = \text{matriz de inercia referente ao link i} \tag{2.37}$$
$$l_i = \text{distância entre o frame i e o i+1} \tag{2.38}$$
$$lc_i = \text{distância entre o frame i e o centro de massa da junta i} \tag{2.39}$$
$$q_i = \text{posição angular da junta i} \tag{2.40}$$
$$\dot{q}_i = \text{velocidade angular da junta i} \tag{2.41}$$
$$\ddot{q}_i = \text{aceleração angular da junta i} \tag{2.42}$$
$$rc_{i,i} = \text{distância entre o frame i+1 e o centro de massa da junta i} \tag{2.43}$$
$$B_i = \text{Coeficiente de atrito viscoso da junta i} \tag{2.44}$$

2.2.1 Dinâmica Inversa

A parametrização de Newton-Euler [14] parte do princípio das leis de movimento de corpos rígidos de Euler e da lei da ação e reação de Newton:

- Ação e reação: Toda ação tem uma reação de mesma magnitude e direção oposta. Ou seja, se o link 1 exerce um torque τ no link 2, o segundo exerce um torque $-\tau$ no link 1.
- A variação do momento linear é igual a força resultante em link.
- A variação do momento angular é igual ao torque resultante em um link.

As equações 2.45 e 2.46 podem ser reescritas como em 2.47 e 2.48, considerando que a massa de um link é constante, fazendo a transformação do torque e momento de inercia em relação ao frame inercial para o frame fixo no centro de massa do próprio link [16].

$$\sum f = m\dot{v} \tag{2.45}$$

$$\sum \tau_0 = \frac{d(I_0\omega_0)}{dt} \tag{2.46}$$

$$\sum f = ma \tag{2.47}$$

$$\sum \tau = \omega \times (I\omega) + I\dot{\omega} \tag{2.48}$$

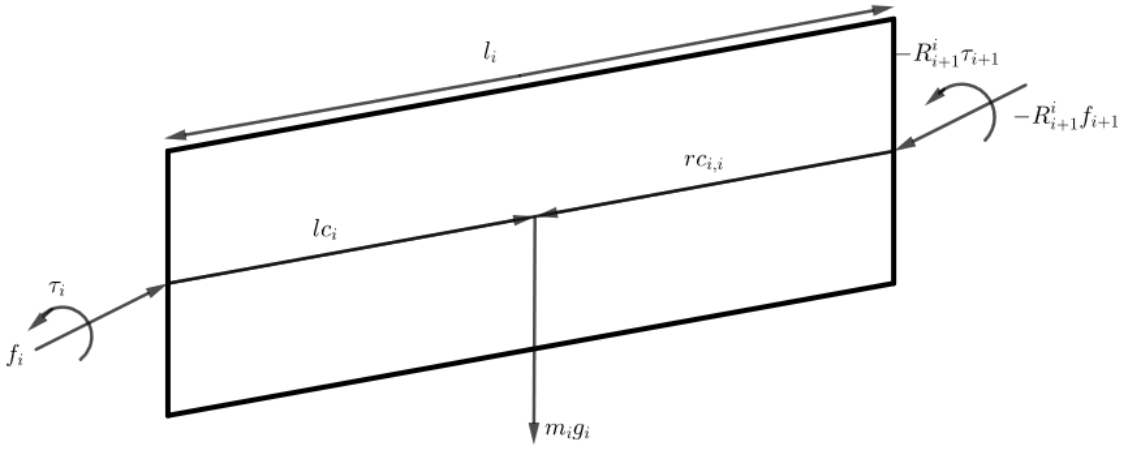

Figura 2.4 – Forças e torques aplicados em um link i qualquer.

As forças que atuam em um link i são: A força que o próprio link f_i está exercendo, a força de reação do link posterior f_{i+1} e a força da gravidade m_ig. Então, considerando que a aceleração do link aparece no centro de massa dele, a expressão para a força exercida pelo link atual, partindo da equação 2.47 fica como em 2.50.

$$f_i - R^i_{i+1} f_{i+1} + m_i g = m_i a_{c,i} \tag{2.49}$$

$$f_i = R^i_{i+1} f_{i+1} + m_i a_{c,i} - m_i g \tag{2.50}$$

Utilizando o balanço de torques para calcular o torque exercido pelo link atual τ_i, partindo da equação 2.48, temos que levar em consideração: o torque de reação do link posterior τ_{i+1}, o torque gerado pela força f_i vezes o a distância do início do link i para o seu centro de massa e o torque resultante da força de reação do link posterior vezes a distância do link posterior até o centro de massa do link atual. Obtemos a equação 2.52 para o τ_i.

$$\tau_i - R^i_{i+1}\tau_{i+1} + f_i \times lc_i - R^i_{i+1} f_{i+1} \times rc_{i,i} = \omega_i \times (I\omega_i) + I\dot{\omega}_i \tag{2.51}$$

$$\tau_i = R^i_{i+1}\tau_{i+1} - f_i \times lc_i + (R^i_{i+1} f_{i+1}) \times rc_{i,i} + \omega_i \times (I\omega_i) + I\alpha_i \tag{2.52}$$

Algo que não é considerado na equação 2.52 é o torque de reação contrário ao movimento resultante do atrito presente nas conexões mecânicas entre o motor e o resto do braço. Uma forma simples de se incorporar o atrito no movimento é considerar apenas o atrito viscoso proporcional a velocidade angular da junta [17]. Adicionado esse componente com sinal oposto ao movimento na equação 2.52 obtemos a equação 2.53 para o torque final.

$$\tau_i = R^i_{i+1}\tau_{i+1} - f_i \times lc_i + (R^i_{i+1} f_{i+1}) \times rc_{i,i} + \omega_i \times (I\omega_i) + I\alpha_i + B_i\dot{q}_i \tag{2.53}$$

O objetivo final é calcular o torque aplicado por cada junta i partindo dos conhecidos torques e forças externas aplicados pelo link anterior mais as leituras de posição, velocidade e aceleração de cada junta providos por seus sensores. Para isso é preciso expressar as componentes ω, α e $a_{c,i}$, da equação 2.53, em termos de q, $\dot{q}$ e $\ddot{q}$.

Para ω partimos do fato de a velocidade angular de um link ser igual a velocidade angular do frame anterior, rotacionado para o eixo de coordenadas do frame atual, mais a rotação feita pela junta atual, equação 2.54. Usando a segunda lei de Newton em um eixo rotativo [18] a derivada da expressão 2.54 fica como mostrado na equação 2.55.

$$\omega_i = R^i_{i-1}\omega_{i-1} + z_i\dot{q}_i \tag{2.54}$$

$$\alpha_i = R^i_{i-1}\alpha_{i-1} + z_i\ddot{q}_i + \omega_i \times z_i\dot{q}_i \tag{2.55}$$

Enfim, para $a_{c,i}$, partimos da expressão 2.56 da velocidade linear do centro de massa, considerando que a distância entre o link anterior e o centro de massa do link atual é constante, temos a equação 2.58 para a aceleração.

$$v_{c,i} = v_{e,i-1} + \omega_i \times r_{i-1,ci} \tag{2.56}$$

$$a_{e,i} = R^i_{i-1} a_{e,i-1} + \dot{\omega}_i \times r_{i-1,i} + \omega_i \times (\omega_i \times r_{i-1,ci}) \tag{2.57}$$

$$a_{c,i} = R^i_{i-1} a_{e,i-1} + \dot{\omega}_i \times lc_i + \omega_i \times (\omega_i \times lc_i) \tag{2.58}$$

Com isso finalizamos a formulação proposta que relaciona a posição, velocidade e aceleração de cada junta e o torque exercido para realizar aquele movimento. Como pode ser visto essa formulação é recursiva, ou seja, para um manipulador com n juntas, o cálculo do torque da junta 1 depende do torque das $n-1$ juntas posteriores, por outro lado o cálculo da última junta depende da definição da aceleração dos centros de massa de todos links anteriores. Dessa forma o cálculo pode ser divido assim:

- **Recursão Direta**:

  Considerando as condições iniciais $\omega_0 = \alpha_0 = a_{c,0} = a_{e,0} = 0$, isto significa dizer que o manipulador tem movimento nulo em relação a um referencial, é possível resolver as equações 2.54, 2.55, 2.57 e 2.58, nesta ordem, partindo do link 1 até o link n.

- **Recursão Inversa**:

  Considerando as condições inicias $f_{n+1} = 0$ e $\tau_{n+1} = 0$, ou seja não há forças ou torques externos sendo aplicados no efetuador terminal, é possível resolver as equações 2.50 e 2.53 para obter o torque que está sendo realizado em cada uma das n juntas.

Como é possível ver pelas equações, a realização do cálculo depende, além das variáveis de junta, da definição das seguintes constantes para cada uma das n juntas: distância nas 3 direções entre a origem do frame da junta e o seu centro de massa, matriz de inércia $3x3$, massa e coeficiente de atrito viscoso. Portanto o conhecimento desses valores para cada junta se torna necessária para se realizar o cálculo numérico do torque.

Um algoritmo que implementa todos os passos das recursões direta e inversa descritos acima foi implementado no ambiente Matlab de forma simbólica utilizando como base a tese de mestrado [19].

2.2.1.1 Reescrita da Equação Dinâmica

As saídas da recursão inversa do algoritmo de Newton-Euler são as equações que descrevem o torque de cada junta do manipulador, porém essas saídas não estão em uma forma

que facilite os procedimentos de identificação e controle. Para resolver isso foi feito um algoritmo, em Matlab, que reescreve equação 2.53 no formato matricial da equação 2.59.

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + B\dot{q} \tag{2.59}$$

Onde

- M é a matriz inercial
- C é a matriz com termos centrífugos e de Coriolis
- G é a matriz gravitacional
- B é o vetor de atrito viscoso
- q é o vetor de variáveis de junta
- τ é o vetor de torques

A matriz inercial é uma matriz 6x6 em que as linhas representam o torque de cada junta e as colunas representam a aceleração de cada junta. Portanto o termo $m_{i,j}$, i e j variando de 1 a 6, representa os termos do torque da junta i que são proporcionais a aceleração da junta j. Para o isolar basta substituir $\ddot{q}_j = 1$ na equação de τ_i enquanto os outros $\ddot{q}$, todos os $\dot{q}$ e g são igualados a zero. Dessa forma os únicos termos que sobram da expressão τ_i são aqueles que multiplicam $\ddot{q}_j$.

A matriz gravitacional é uma 6x1 em que cada linha i representa a expressão de dependência de τ_i com a gravidade g e o ângulo q das juntas. Para calcular cada g_i é preciso igualar todos os valores de $\ddot{q}$ e $\dot{q}$ a zero em τ_i.

Por fim, C e B foram unidas em uma só matriz 6x1 tendo em vista que ambas são multiplicadas por $\dot{q}$. Por sua complexa escrita e dependência de q e $\dot{q}$, a definição dessa matriz conjunta foi feita de acordo com a equação 2.60. Vale ressaltar que essa matriz já contempla a multiplicação de $\dot{q}$.

$$CoBdQ = \tau - M\ddot{q} - G \tag{2.60}$$

2.2.1.2 Dinâmica no Espaço Cartesiano

Até agora as equações dinâmicas foram todas desenvolvidas em termos do espaço de juntas, porém, como será visto mais adiante no capítulo de controladores, é interessante

também descrever a dinâmica no espaço cartesiano. Esse novo sistema descreve a aceleração do efetuador terminal em relação as forças e torques aplicados nele.

O nosso objetivo é transformar a equação 2.61, que é a mesma equação 2.59 em que o termo $\dot{q}$ que multiplica $C(q, \dot{q})$ e a matriz $B\dot{q}$ foram incorporadas na própria matriz $C(q, \dot{q}$, em uma equação na forma de 2.62.

$$\tau = M(q)\ddot{q} + C(q, \dot{q}) + G(q) \tag{2.61}$$

$$\mathcal{F} = M_{\mathcal{X}}(q)\ddot{\mathcal{X}} + C_{\mathcal{X}}(q, \dot{q}) + G_{\mathcal{X}}(q) \tag{2.62}$$

O termo $\mathcal{F}$ é um vetor 6x1 de força-torque aplicado na efetuador terminal e o vetor $\mathcal{X}$ e as matrizes $M_{\mathcal{X}}(q)$, $C_{\mathcal{X}}(q, \dot{q})$ e $G_{\mathcal{X}}(q)$ são análogas às respectivas matrizes no espaço das juntas.

A equação 2.63 é uma forma de relacionar o vetor $\mathcal{F}$ com os torques resultantes das juntas de um manipulador [12].

$$\tau = J^T(\theta)\mathcal{F} \tag{2.63}$$

Pré multiplicando as equações 2.61 e 2.63 por $J^{-T(\theta)}$ obtemos

$$J^{-T}(\theta)\tau = J^{-T}(\theta)M(q)\ddot{q} + J^{-T}(\theta)C(q, \dot{q}) + J^{-T}(\theta)G(q) \tag{2.64}$$

$$J^{-T}(\theta)\tau = \mathcal{F} \tag{2.65}$$

Realizando a igualdade acima temos

$$\mathcal{F} = J^{-T}(\theta)M(q)\ddot{q} + J^{-T}(\theta)C(q, \dot{q}) + J^{-T}(\theta)G(q) \tag{2.66}$$

Utilizando a definição da jacobiana mostrada em 2.20, notando que v daquela equação equivale a $\dot{\mathcal{X}}$, diferenciando essa equação em relação ao tempo pela regra da cadeia temos a equação 2.68 que descreve a aceleração da ferramenta. Por fim, isolando o termo $\ddot{\theta}$ temos a equação 2.69.

$$\dot{\mathcal{X}} = J\dot{\theta} \tag{2.67}$$

$$\ddot{\mathcal{X}} = \dot{J}\dot{\theta} \tag{2.68}$$

$$\ddot{\theta} = J^{-1}\ddot{\mathcal{X}} - J^{-1}\dot{J}\dot{\theta} \tag{2.69}$$

Substituindo 2.69 em 2.66 temos

$$\mathcal{F} = J^{-T}(\theta)M(q)(J^{-1}\ddot{\mathcal{X}} - J^{-1}\dot{J}\dot{\theta}) + J^{-T}(\theta)C(q,\dot{q}) + J^{-T}(\theta)G(q) \tag{2.70}$$

Agrupando os termos de forma que fiquem na forma da equação 2.62, temos as matrizes

$$M_{\mathcal{X}}(\theta) = J^{-T}(\theta)M(\theta)J^{-1}(\theta) \tag{2.71}$$

$$C_{\mathcal{X}}(\theta,\dot{\theta}) = J^{-T}(\theta)(C(\theta,\dot{\theta}) - M(\theta)J^{-1}(\theta)\dot{J}(\theta)\dot{\theta}) \tag{2.72}$$

$$G_{\mathcal{X}}(\theta) = J^{-T}(\theta)G(\theta) \tag{2.73}$$

É notável que as matrizes na forma cartesiana dependem das matrizes no espaço das juntas. Não sendo preciso a determinação analítica dessas novas matrizes, já que durante uma simulação é possível calcular seus valores numéricos dado o valor das matrizes do espaço de juntas e do valor da jacobiana para aquele instante de tempo.

2.2.2 Dinâmica Direta

Para a formulação da Dinâmica direta, equação 2.74, basta manipular a equação dada em 2.59 isolando o termo $\ddot{q}$.

$$\ddot{q} = M^{-1}(\tau - C(q,\dot{q})\dot{q} - G(q) - B\dot{q}) \tag{2.74}$$

$$\dot{q} = \int \ddot{q}dt \tag{2.75}$$

$$q = \int \dot{q}dt \tag{2.76}$$

Essa equação dinâmica é utilizada principalmente para se fazer simulações de manipuladores já que em um ambiente prático real quem fornece a aceleração e suas integrais é o próprio sistema físico.

2.3 IDENTIFICAÇÃO DE SISTEMAS DINÂMICOS

Identificação de sistemas é um processo que busca achar o modelo matemático que melhor explica como que um vetor de entrada $u(t)$ se relaciona com um vetor de saída $y(t)$. Ou seja, ele busca calcular um vetor de parâmetros que mais aproxima dados de entrada com dados de saída [20].

No caso da identificação de um manipulador, buscamos calcular os valores das constantes das matrizes M, C, G e B da equação 2.59 usando o torque medido como saída e as variáveis de junta medidas como entrada. É comum que tais constantes sejam determinadas via identificação dinâmica quando não disponibilizadas pelo fabricante do robô ou, então, quando não há confiança sobre os valores fornecidos.

O modelo dinâmico do torque é extremamente não linear pela presença de diversos termos de seno e cosseno que aparecem na equação devido às rotações das juntas e, portanto, impossibilita-se o uso das técnicas lineares de identificação. Para que essas técnicas pudessem ser utilizadas foi necessário manipular a equação para que ela se tornasse linear nos parâmetros.

2.3.1 Linearização

A equação do torque, na forma na forma da equação 2.59, pode ser reescrita de forma que se torne linear nos parâmetros [14], ou seja, existe a possibilidade de separar os termos em duas matrizes, como na equação 2.77, em que $Y(q, \dot{q}, \ddot{q})$ contém a parte não linear e Θ que contém os termos lineares.

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) + B\dot{q} = Y(q, \dot{q}, \ddot{q})\Theta = \tau \tag{2.77}$$

A matriz $Y(q, \dot{q}, \ddot{q})$ é formada por uma complexa relação não linear, porém depende unicamente de q, $\dot{q}$ e $\ddot{q}$ que em um experimento são valores medidos das variáveis de junta. Portanto tomando um experimento com todas essas medidas é possível, a cada instante de tempo, calcular cada valor da matriz Y. Dessa forma, mesmo que não seja possível determinar cada parâmetro dinâmico individual, como a massa de cada link, é possível utilizar técnicas de identificação lineares para determinar Θ. Tornando-se possível estimar o torque aplicado pelas juntas a partir de medições disponíveis de suas variáveis.

Para que fosse possível separar os termos da expressão para o torque de cada junta, foi feito um algoritmo no software *Maple* que importa as expressões simbólicas geradas no Matlab e faz a manipulação algébrica para simplificar e separar conjuntos de multiplicações dos termos lineares dos não lineares. O comando feito pode ser visto, na notação necessária

do *Maple*, em 2.78.

$$\begin{aligned}\tau_i := collect(combine(expand(\tau_i), trig),\\ (m_1, m_2, m_3, m_4, m_5, m_6, l_{1y}, l_{2x}, l_{2z}, l_{3x}, l_{3z}, l_{4y}, l_{5y}, l_{c1y}, l_{c2x}, l_{c2z}, l_{c3x}, l_{c3z}, l_{c4y}, l_{c5y}, l_{c6y},\\ beta_1, beta_2, beta_3, beta_4, beta_5, beta_6, I_{6x}, I_{6y}, I_{6z}, I_{5x}, I_{5y}, I_{5z}, I_{4x}, I_{4y}, I_{4z}, I_{3x}, I_{3y}, I_{3z},\\ I_{2x}, I_{2y}, I_{2z}, I_{1x}, I_{1y}, I_{1z}), distributed)\end{aligned} \quad (2.78)$$

Então, a saída do dessa função foi analisada manualmente para montar as matrizes $Y(q, \dot{q}, \ddot{q})$ e Θ. Os parâmetros dinâmicos escritos como segundo argumento da função *collect* são apenas aqueles que foram usados para o caso do UR3, que será mostrado na seção 3.

Uma vez com o sistema corretamente separado na forma $\tau = Y(q, \dot{q}, \ddot{q})\Theta$ é possível usar técnicas de identificação para sistemas lineares, desde que seja possível substituir a matriz Y por medições a cada instante de um experimento. A técnica escolhida para ser usada no trabalho foi a identificação por mínimos quadrados clássico por sua simples implementação e pelo bom resultado obtido.

2.3.2 Mínimos Quadrados

Dado uma função $\tau = f(y)$ que depende de um vetor θ de tamanho n, $f(y) : \mathbb{R}^n \to \mathbb{R}$, pode-se escrever essa relação, em um instante i, na forma $\tau_i = f(y_i, \theta)$. Se forem tomadas N observações de τ e y temos a relação matricial da equação 2.79. Em que Y é chamada matriz de regressores e θ é o vetor de parâmetros.

$$\begin{gathered}\tau = Y\theta \\ \begin{bmatrix}\tau_1 \\ \tau_2 \\ \vdots \\ \tau_N\end{bmatrix} = \begin{bmatrix}y_1^1 & y_1^2 & \dots & y_1^n \\ y_2^1 & y_2^2 & \dots & y_2^n \\ \vdots & \vdots & \dots & \vdots \\ y_N^1 & y_N^2 & \dots & y_N^n\end{bmatrix} \begin{bmatrix}\theta_1 \\ \theta_2 \\ \vdots \\ \theta_n\end{bmatrix}\end{gathered} \quad (2.79)$$

Para que essa relação matricial seja válida é preciso que a função f seja invariante no tempo, para que de fato o vetor θ seja o mesmo em diferentes observações da função. Outra restrição é a função ser linear nos parâmetros, ou seja, os termos do modelo são valores constantes ou multiplicações de um parâmetro linear com uma função preditora que pode ser não linear.

Assumindo uma estimativa dos valores reais de θ como $\hat{\theta}$, podemos definir a equação

$\tau = Y\hat{\theta} + \xi$ em que ξ é o resíduo ou erro de regressão. Temos, então, que $\xi = \tau - Y\hat{\theta}$ é uma medida de quão bem os valores escolhidos para os parâmetros explica a relação de entrada e saída.

É desejável que $\hat{\theta}$ seja tal que minimize o valor de ξ. Para isso é definida a função de custo que quantifica o ajuste de $Y\hat{\theta}$ sobre a curva τ e depende do vetor $\hat{\theta}$ estimado e do conjunto de dados Z^N utilizado.

$$J_{MQ}(\hat{\theta}, Z^N) = \sum_{i=1}^{N} \xi^2(i) = \xi^T \xi = ||\xi||^2 \tag{2.80}$$

Reescrevendo a função de custo para explicitar os parâmetros e medidas temos 2.81. Como o objetivo da identificação é minimizar o valor da função de custo, calculamos a derivada com respeito a $\hat{\theta}$, equação 2.82, igualando o resultado a zero e resolvendo para $\hat{\theta}$ obtemos a solução mostrada na equação 2.83 [20].

$$\begin{aligned} J_{MQ} &= (\tau - Y\hat{\theta})^T(\tau - Y\hat{\theta}) \\ &= \tau^T\tau - \tau^T Y\hat{\theta} - \hat{\theta}^T Y^T \tau + \hat{\theta}^T Y^T Y\hat{\theta} \qquad (2.81) \\ \frac{\partial J_{MQ}}{\partial \hat{\theta}} &= -(\tau^T Y)^T - Y^T + (Y^T Y + Y^T Y)\hat{\theta} \\ &= -Y^T\tau - Y^T\tau + 2Y^T Y\hat{\theta} \qquad (2.82) \\ \hat{\theta} &= [Y^T Y]^{-1} Y^T \tau \qquad (2.83) \end{aligned}$$

Por fim, para se verificar se a equação 2.83 é um ponto de mínimo ou máximo, calcula-se a segunda derivada de J_{MQ}, equação 2.84. Como o termo $Y^T Y$ é, por construção, uma matriz positiva definida, a derivada é maior que zero e, portanto, a expressão obtida é de fato um ponto mínimo.

$$\frac{\partial^2 J_{MQ}}{\partial \hat{\theta}^2} = 2Y^T Y > 0 \tag{2.84}$$

2.3.3 Geração dos Dados

Um dos passos importantes no processo de identificação de sistemas dinâmicos é a definição do conjunto de dados que serão usados para gerar as restrições do processo de minimização. A equação 2.80 indica que tal conjunto de dados influencia no resultado da função de custo.

A trajetória escolhida deve garantir que o robô faça um movimento seguro, não ocorram

colisões entre o manipulador e o ambiente e com ele mesmo. Ela deve, também, ser uma excitação suficiente para ser acurada e rápida mesmo na presença de ruído e perturbações nas medições.

A sensibilidade dos parâmetros identificados em relação aos erros, seja de modelagem ou ruídos de medição, pode ser determinado a partir do número de condição da matriz de observação Y [21] e é calculado via 2.85 e 2.86. Já o valor da norma $Cond_2(Y)$ é calculado pela decomposição de valores singulares (SVD) da matriz de regressores Y via equação 2.87, em que σ_1 é o maior valor singular e σ_N é o maior.

$$\frac{||\partial\hat{\theta}||}{\hat{\theta}} \leq Cond(Y)\frac{||\partial\hat{\tau}||}{\hat{\tau}} \quad \text{,com } \partial Y = 0 \tag{2.85}$$

$$\frac{||\partial\hat{\theta}||}{\hat{\theta}+\partial\hat{\theta}} \leq Cond(Y)\frac{||\partial\hat{Y}||}{\hat{Y}} \quad \text{,com } \partial\tau = 0 \tag{2.86}$$

$$Cond(Y) = \frac{\sigma_1}{\sigma_N} \tag{2.87}$$

Existem diversas abordagens para se determinar as curvas de entrada para identificação, dentre elas o sinal pseudo-randômico, séries finitas de Fourier e composição de séries de Fourier e polinômios. Neste trabalho foi utilizado o conjunto de experimentos realizado pelo aluno Pedro Garcia da Universidade de Brasília no final do ano de 2019.

Esse conjunto de experimentos foi realizado utilizando a parametrização proposta por Park [22], em que a trajetória é definida como a soma de uma componente polinomial e uma componente de série de Fourier e é dada pelas equações 2.88 a 2.90. Em que ω_0 é a frequência de oscilação da trajetória e i é o índice da junta variando de 1 a n para um robô com n juntas.

$$q_i(t) = \lambda_i(t) + \delta_i(t) \tag{2.88}$$

$$\lambda_i(t) = \sum_{j=1}^{5} \lambda_{ij} t^j \tag{2.89}$$

$$\delta_i(t) = \sum_{l=1}^{N} a_{il} cos(l\omega_0 t) \tag{2.90}$$

As condições de contorno do movimento podem ser definidas como mostrado de 2.91 a 2.93, em que 0 é o início da trajetória e t_f é o tempo final.

$$q_i(0) = \lambda_i(0) + \delta_i(0), \quad q_i(t_f) = \lambda_i(t_f) + \delta_i(t_f) \tag{2.91}$$

$$\dot{q}_i(0) = \dot{\lambda}_i(0) + \dot{\delta}_i(0), \quad \dot{q}_i(t_f) = \dot{\lambda}_i(t_f) + \dot{\delta}_i(t_f) \tag{2.92}$$

$$\ddot{q}_i(0) = \ddot{\lambda}_i(0) + \ddot{\delta}_i(0), \quad \ddot{q}_i(t_f) = \ddot{\lambda}_i(t_f) + \ddot{\delta}_i(t_f) \tag{2.93}$$

Dessa forma, escolhidos as condições de contorno, é executado, sobre a equação 2.88, um algoritmo de otimização que minimize o valor de condição $Cond(Y)$ ao mesmo tempo que se garante que as juntas não possam entrar nem em uma disposição de colisão nem passar dos seus valores máximos.

2.4 CONTROLE ROBÓTICO

Controle robótico consiste em a partir de uma referência e leituras de estados atuais comandar os sinais de excitação para conseguir ditar alguma variável de saída. Normalmente tal controle é realizado ajustando a corrente nas juntas para se obter uma posição, velocidade ou força de interação desejada.

2.4.1 Controle de um Sistema de 2º Ordem

Para começar a discussão sobre controladores será descrito primeiramente o comportamento de um sistema simples de segunda ordem composto por uma massa m, mola com coeficiente elástico k e sujeito a um atrito com a superfície de coeficiente b. A equação 2.94 descreve a movimentação desse sistema, dado pelo princípio fundamental da dinâmica de Newton, enquanto 2.95 mostra sua transformada de Laplace.

$$m\ddot{x} + b\dot{x} + kx = 0 \tag{2.94}$$

$$X(s)(ms^2 + bs + k) = 0 \tag{2.95}$$

A resposta do sistema dinâmico apresentado depende das suas raízes, equação 2.96. Essas raízes, também chamadas de polos do sistema, dependem diretamente dos valores numéricos de m, k e b e ditam a natureza da movimentação do bloco.

$$s_1 = -\frac{b}{2m} + \frac{\sqrt{b^2 - 4mk}}{2m}$$
$$s_2 = -\frac{b}{2m} - \frac{\sqrt{b^2 - 4mk}}{2m} \tag{2.96}$$

Como os valores das constantes são necessariamente positivos, a resposta dinâmica do sistema será sempre estável e os seus polos poderão ter uma das três formas:

- **Reais e iguais**: Caso $b^2 = 4mk$ o sistema é chamado de criticamente amortecido e tem a resposta sem oscilação mais rápida possível.
- **Reais e diferentes**: Caso $b^2 > 4mk$ o sistema é chamado de sobre amortecido e tem a resposta lenta.
- **Complexas conjugadas**: Caso $b^2 < 4mk$ o sistema é chamado de sub amortecido e tem uma resposta oscilatória decrescente.

Supondo agora que a resposta dinâmica natural do sistema não seja da forma que se deseja. Se for possível aplicar uma força no sistema, via atuadores, e ler a resposta, via sensores, será possível modificar o comportamento como desejado. A figura 2.5 mostra o sistema físico na presença de uma força f.

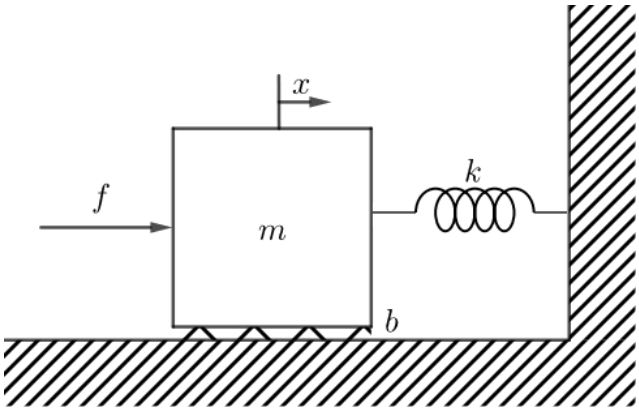

Figura 2.5 – Sistema massa-mola com atrito.

Propõe-se um sistema de controle regulador de posição na forma da equação 2.97 que calcula a força a ser aplicada no bloco dado a leitura de posição e velocidade a fim de manter a sua posição original.

$$f = -K_p x - K_v \dot{x} \tag{2.97}$$

Adicionando essa força f ao lado direito da equação 2.94 temos uma nova equação que descreve o movimento do bloco, equação 2.98.

$$m\ddot{x} + (b + K_v)\dot{x} + (k + K_p)x = 0 \tag{2.98}$$

É possível via escolha dos ganhos de controle K_p e K_v posicionar os polos de forma a ter a resposta dinâmica de interesse. É comum que sistemas de controle tenham a característica de ser criticamente amortecidos já que esta garante resposta mais rápida e sem oscilação possível. Sendo assim basta escolher os ganhos de forma que $(b + K_v) = 2\sqrt{m(k + K_p)}$. É preciso tomar cuidado para que tanto $b + K_v$ quanto $k + K_p$ sejam positivos, pois caso contrário o sistema se torna instável fazendo com que o erro de posição aumente.

2.4.2 Particionamento da Lei de Controle

Para facilitar o projeto de controladores mais complexos nas próximas seções, vai ser proposto uma nova estrutura de controlador. Essa estrutura é formada por duas partes: a parte baseada em modelo e a parte do servo [12].

O objetivo dessa nova estrutura é modificar a equação de malha aberta 2.99 em duas partes. Na parte baseada em modelo vamos assumir o conhecimento de m, b e k e usa-los para reescrever a equação, como em 2.100, de forma a reduzir o sistema para que ele pareça um formado por uma simples massa unitária, ou seja, como se não houvesse a presença de mola e atrito.

$$m\ddot{x} + b\dot{x} + kx = f \tag{2.99}$$

$$f = \alpha f' + \beta \tag{2.100}$$

Onde α e β são escolhidos de tal forma que se f' for tomado como a nova entrada do sistema ele vai parecer ser uma massa unitária. Substituindo 2.100 em 2.99, temos a equação 2.101.

$$m\ddot{x} + b\dot{x} + kx = \alpha f' + \beta \tag{2.101}$$

Agora é visível que para se obter o sistema de massa unitária é preciso fazer $\alpha = m$ e $\beta = b\dot{x} + kx$ obtendo a equação 2.102.

$$f' = \ddot{x} \tag{2.102}$$

Já para a parte do servo assumimos a mesma metodologia explicada para a equação 2.97 para controlar um sistema de massa unitária.

$$f' = -K_p x - K_v \dot{x} \tag{2.103}$$

Combinando 2.103 com 2.102 temos a equação 2.104. Para que a resposta seja criticamente amortecida basta tomar a relação de K_v e K_p igual a equação 2.105.

$$\ddot{x} + K_v \dot{x} + K_p x = 0 \tag{2.104}$$

$$K_v = 2\sqrt{K_p} \tag{2.105}$$

2.4.3 Controle de Acompanhamento das Variáveis de Junta

Variáveis de junta são os valores, para cada junta, de posição, velocidade e aceleração angulares. Desenvolver o controle dessas variáveis é de interesse como um primeiro passo para se desenvolver controles mais sofisticados.

Manipuladores robóticos em movimento estão sujeitos às forças da gravidade, inerciais, de Coriolis, centrípeta e de atrito 2.59. Essa equação pode ser utilizada para desenvolver um sistema de controle que prediz o torque necessário a ser aplicado para que certo movimento seja realizado [17]. Dessa forma, torna-se visível quais forças e torques o manipulador precisa fazer para realizar um movimento desejado.

Assumindo que os torques necessários, τ, são dados pela saída do sistema de controle, u, pode-se reescrever a equação original 2.59 como 2.106.

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + B\dot{q} = u \tag{2.106}$$

Reescrevendo a equação 2.106 isolando o termo $\ddot{q}$, temos a expressão 2.107 que define a aceleração obtida pelas juntas do braço robótico dado o sinal de controle calculado. Calculando a primeira e a segunda integral dessa equação no tempo, define-se, respectivamente, as velocidades e posições das juntas. Em que, para tornar a notação mais compacta, a expressão $C(q,\dot{q})\dot{q} + G(q) + B\dot{q}$ foi substituída por $n(q,\dot{q})$.

$$\ddot{q} = M^{-1}(q)(u - n(q,\dot{q})) \tag{2.107}$$

Para realizar esse cálculo é necessário se assumir os valores verdadeiros dos parâmetros dinâmicos do robô, tal suposição não será necessária em eventuais testes reais em robôs, pois as velocidades e posições dado os sinais de controle são lidos pelo mecanismo real ao invés

de serem calculados.

Assumindo que a matriz $\hat{M}(q)$ representa uma aproximação dos valores verdadeiros de $M(q)$ e $\hat{n}(q, \dot{q})$ é uma aproximação dos valores verdadeiros de $n(q, \dot{q})$, o valor do sinal de controle para o torque é calculado pela equação 2.108.

$$u = \hat{M}(q)y + \hat{n}(q, \dot{q}) \tag{2.108}$$

Na figura 2.6, pode-se ver a representação em diagrama de blocos do sistema de controle composto pela combinação das equações 2.107 e 2.108.

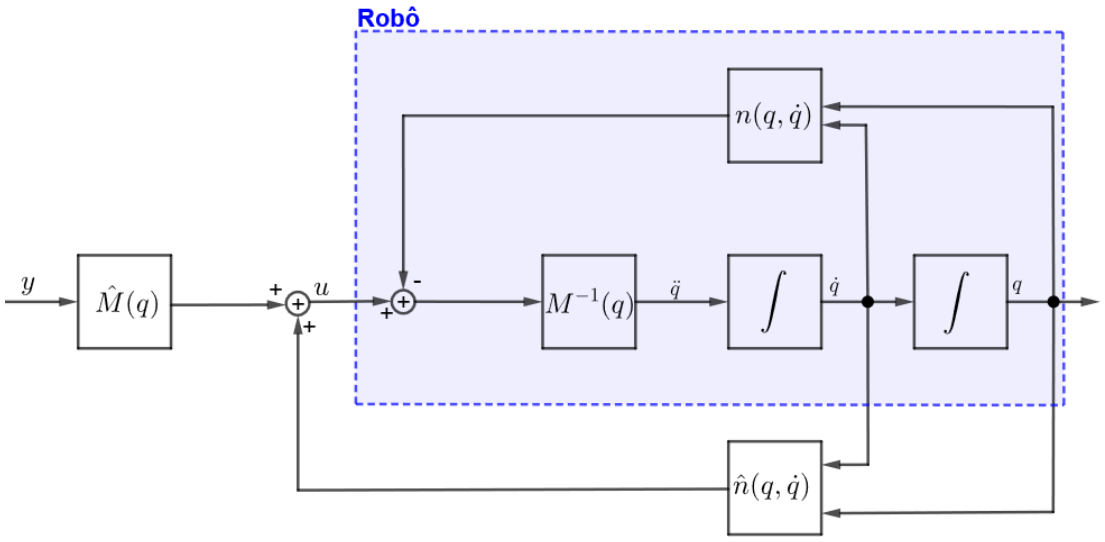

Figura 2.6 – Diagrama de blocos do sistema de controle do sistema linearizado.

Se os valores estimados de $M(q)$ e $n(q, \dot{q})$ forem iguais aos respectivos valores reais, as matrizes $n(q, \dot{q})$ e $\hat{n}(q, \dot{q})$ se subtraem e a multiplicação das matrizes $\hat{M}(q)$ pela inversa de $M(q)$ resulta na identidade. Assim, temos um sistema linear, a saída q está relacionada com a entrada y apenas via dois integradores, e de-copulado, isto é, cada valor de entrada de y influência unicamente em uma saída de $\ddot{q}$. Ou seja, o vetor y tem característica de aceleração: $y = \ddot{q}$. Essa relação e a mesma obtida na equação 2.102 na subseção anterior, portanto é possível pensar em um sistema de controle parecido com o 2.103.

No cenário ideal de se conhecer exatamente os parâmetros do robô seria suficiente fornecer uma aceleração desejada como sinal de controle e o manipulador iria seguir a trajetória determinada [17]. Porém não é comum que se tenha modelos perfeitos. O erro na posição e na velocidade das juntas pode ser definido como nas equações 2.109 e 2.110.

$$\tilde{q} = q_r - q \tag{2.109}$$

$$\dot{\tilde{q}} = \dot{q}_r - \dot{q} \tag{2.110}$$

Incorporando estes valores de erro, multiplicados por ganhos, mais uma referência de aceleração para o valor de y temos a equação 2.111. Os sinais das equações 2.109 e 2.110 são responsáveis por corrigir os erros devidos aos incertezas de modelagem.

$$y = \ddot{q}_r + K_p(q_r - q) + K_v(\dot{q}_r - \dot{q}) \tag{2.111}$$

Unindo a equação 2.111 ao fato que $y = \ddot{q}$ temos a equação 2.112 em que $\ddot{\tilde{q}} = \ddot{q}_r - \ddot{q}$

$$\ddot{\tilde{q}} + K_v\dot{\tilde{q}} + K_p\tilde{q} = 0 \tag{2.112}$$

A escolha de K_v e K_p pode ser feita usando a equação 2.105 assim como explicado anteriormente. O diagrama de blocos que ilustra esse controlador pode ser visto na figura 2.7.

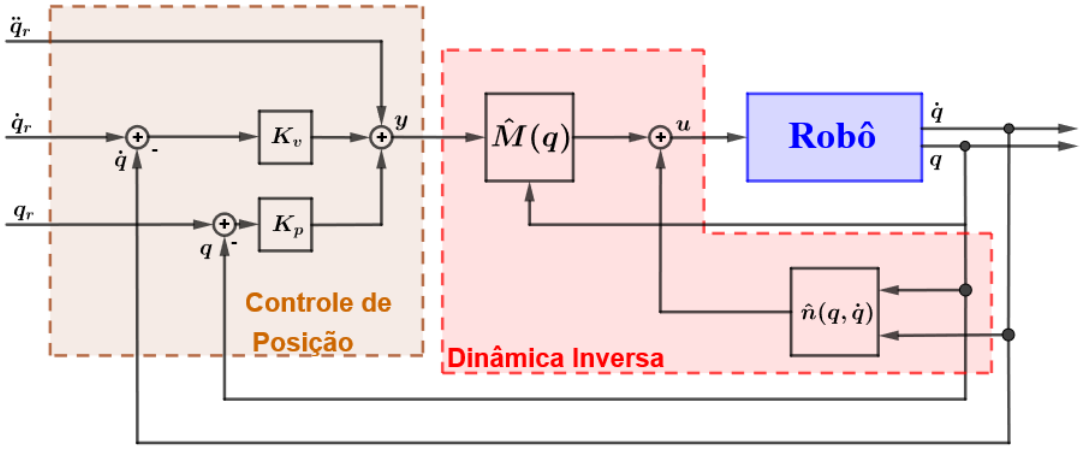

Figura 2.7 – Diagrama de blocos ilustrando o controlador das juntas.

Pela equação do erro, equação 2.112, podemos fazer a análise de regime permanente para verificar se o controlador é capaz de reduzir o erro de posição à zero. De fato, levando as derivadas dessa expressão à zero temos $K_p\tilde{q} = 0$ o que mostra que há erro nulo de posição no regime permanente. Porém isso não ocorrerá se for inserido um torque de perturbação no sistema.

Um problema da estrutura apresentada em 2.7 é a sua sensibilidade a perturbações ou ruídos, isto é, caso exista um torque de perturbação o sistema não terá erro de posição zero.

Fazendo a mesma análise anterior no regime permanente, mas agora considerando um vetor de torque τ_d sendo aplicado diretamente nas juntas do robô temos a nova equação 2.113 para erro de posição [12].

$$\ddot{\tilde{q}} + K_v\dot{\tilde{q}} + K_p\tilde{q} = \tau_d \tag{2.113}$$

Fazendo a análise de regime permanente na equação de erro 2.113 é constatado que esse novo sistema leva a um erro de posição igual à $\tilde{q} = \tau_d/k_p$ que depende da magnitude da perturbação e do termo K_p.

Uma forma de rejeitar o erro de posição devido à perturbação é a inclusão de um canal integral proporcional ao erro de posição multiplicado por um ganho K_i. A nova equação de controle de posição se torna 2.114 e a equação de erro 2.115. A figura 2.8 mostra a nova arquitetura com a presença do termo integral e da perturbação externa.

$$y = \ddot{q}_r + K_p(q_r - q) + K_v(\dot{q}_r - \dot{q}) + K_i \int (q_r - q)dt \tag{2.114}$$

$$\tau_d = \ddot{\tilde{q}} + K_v\dot{\tilde{q}} + K_p\tilde{q} + K_i \int \tilde{q}dt \tag{2.115}$$

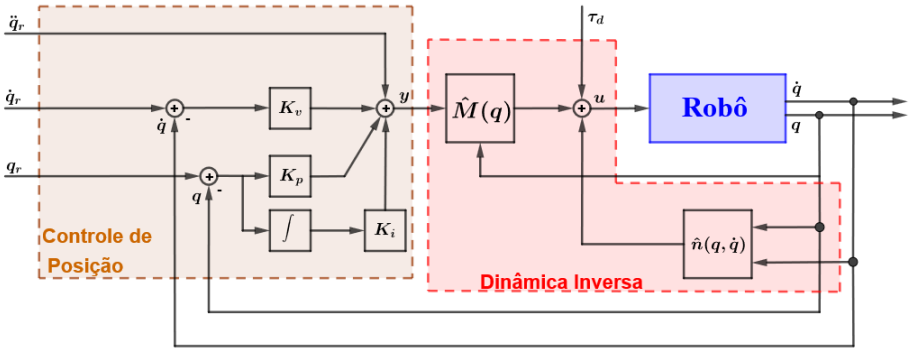

Figura 2.8 – Diagrama de blocos ilustrando o controlador das juntas com o canal integral.

Mais uma vez fazendo a análise de regime permanente é possível ver que o erro de posição se torna $\dddot{\tilde{q}} + K_v\ddot{\tilde{q}} + K_p\dot{\tilde{q}} + K_ie = \dot{\tau}_d$. Então o erro se torna $K_ie = 0$. Dessa forma é possível chegar em erro nulo em regime permanente.

2.4.4 Controle de Acompanhamento das Variáveis de Ferramenta

O controlador desenvolvido anteriormente foi baseado na manipulação das variáveis internas, ou seja, ângulos de junta e suas derivadas. É de interesse, principalmente quando se trata de manipuladores, que seja feita uma arquitetura de controle capaz de controlar di-

retamente as variáveis de ferramenta, também chamado de controle cartesiano, em que a referência é um ponto e orientação no espaço 3D para localização da ferramenta do robô.

Um modo de se realizar tal controle cartesiano é simplesmente realizar a transformação de variável de juntas para variável cartesiana na referência via uma conversão de trajetória, equações 2.116 a 2.118 em que INVKIN é a cinemática inversa. Porém, essa abordagem se mostrou ineficiente computacionalmente e mesmo que se resolva apenas a 2.116 e utilize métodos numéricos para obtenção da das suas derivadas isso costuma acumular muitos erros de estimativa [12].

$$q_d = INVKIN(\mathcal{X}_d), \tag{2.116}$$
$$\dot{q}_d = J^{-1}(q)\dot{\mathcal{X}}_d, \tag{2.117}$$
$$\ddot{q}_d = \dot{J}^{-1}(q)\dot{\mathcal{X}}_d + J^{-1}(q)\ddot{\mathcal{X}}_d \tag{2.118}$$

Uma forma de se evitar tal problema de conversão de trajetória é utilizar a transformação da equação dinâmica mostrada na seção de modelagem dinâmica na equação 2.62. Nessa equação é possível calcular, sem a necessidade de transformação de trajetória, o vetor de força-torque $\mathcal{F}$ e utilizá-lo como entrada da equação dinâmica.

Porém o que a equação de $\mathcal{F}$ calcula é: Se o robô está em uma configuração q e o efetuador terminal tem aceleração $\ddot{\mathcal{X}}$ o vetor força-torque no efetuador terminal é $\mathcal{F}$. É de interesse transformar $\mathcal{F}$ em um vetor que represente o torque em cada junta, sinal que é possível de fato aplicar no robô. Para isso basta utilizar a relação da equação 2.119. Também é necessário transformar o ângulo q em posição e orientação cartesiana e a velocidade angular $\dot{q}$ em velocidade linear $\dot{\mathcal{X}}$, equações 2.120 e 2.121.

$$\tau = J^T(\theta)\mathcal{F} \tag{2.119}$$
$$\mathcal{X} = KIN(q) \tag{2.120}$$
$$\dot{\mathcal{X}} = J(q)\dot{q} \tag{2.121}$$

A mesma análise para erro de posição feita no controlador de juntas pode ser feita aqui. O diagrama de blocos do controlador de variável cartesiana já com o canal integral e uma força de perturbação pode ser visto na figura 2.9.

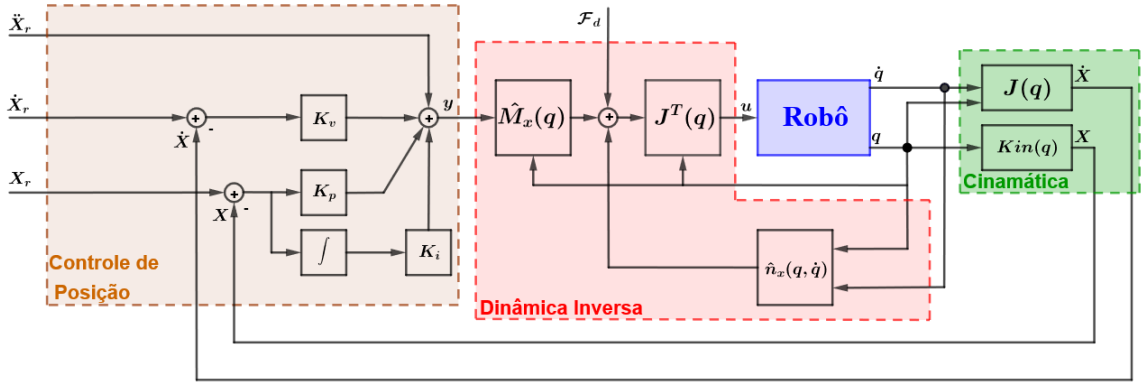

Figura 2.9 – Diagrama de blocos ilustrando o controlador em coordenadas cartesianas com o canal integral.

Esse esquema de controle não tem o problema computacional que teríamos caso fosse feito via conversão de trajetória das equações 2.116 a 2.118, pois nesse novo modelo não são feitos cálculos de inversão de matrizes nem o cálculo da cinemática inversa. Não sendo necessário o uso de técnicas numéricas para resolver essas contas e, portanto, não há os erros numéricos inerentes dessas operações.

3 DESENVOLVIMENTO DOS ALGORITMOS

Neste capítulo, é feito o detalhamento de como foram feitos: a função utilizada para o cálculo da cinemática, a função que escreve as equações dinâmicas do robô, as funções de simplificação e identificação dos parâmetros dinâmicos e por fim o diagrama de blocos que implementa os dois controladores apresentados em ambiente de simulação.

3.1 EQUAÇÕES DO MODELO CINEMÁTICO

As matrizes de rotação entre cada junta adjacente do manipulador, equações 3.1 a 3.6, podem ser calculadas como mostrado na equação 2.2 e a relação de rotação entre o sistema de coordenadas de uma junta e o eixo estacionário, equações 3.7 a 3.11, podem ser definidas a partir da explicação dada na seção 2.1.1.

$$R_0^1 = \begin{bmatrix} c(q_1) & -s(q_1) & 0 \\ s(q_1) & c(q_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(-pi/2) & -s(-pi/2) \\ 0 & s(-pi/2) & c(-pi/2) \end{bmatrix} \tag{3.1}$$

$$R_1^2 = \begin{bmatrix} c(q_2) & -s(q_2) & 0 \\ s(q_2) & c(q_2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

$$R_2^3 = \begin{bmatrix} c(q_3) & -s(q_3) & 0 \\ s(q_3) & c(q_3) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

$$R_3^4 = \begin{bmatrix} c(q_4) & -s(q_4) & 0 \\ s(q_4) & c(q_4) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(-pi/2) & -s(-pi/2) \\ 0 & s(-pi/2) & c(-pi/2) \end{bmatrix} \tag{3.4}$$

$$R_4^5 = \begin{bmatrix} c(q_5) & -s(q_5) & 0 \\ s(q_5) & c(q_5) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(pi/2) & -s(pi/2) \\ 0 & s(pi/2) & c(pi/2) \end{bmatrix} \tag{3.5}$$

$$R_5^6 = \begin{bmatrix} c(q_6) & -s(q_6) & 0 \\ s(q_6) & c(q_6) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

$$R_0^2 = R_0^1 R_1^2 \tag{3.7}$$
$$R_0^3 = R_0^2 R_2^3 \tag{3.8}$$
$$R_0^4 = R_0^3 R_3^4 \tag{3.9}$$
$$R_0^5 = R_0^4 R_4^5 \tag{3.10}$$
$$R_0^6 = R_0^5 R_5^6 \tag{3.11}$$

Uma função que calcula a cinemática direta foi escrita para definir a posição e orientação em RPY do efetuador terminal a partir dos ângulos de junta e a definição da matriz DH do robô usando as equações mostradas na seção 2.1.2. A *toolbox* de robótica do Matlab foi usada para verificar se a função criada estava coerente. Uma imagem do robô simulado, gerada pela *toolbox*, com todos ângulos iguais a zero pode ser vista na figura 3.1. Também foi feito uma função que implementa as equações mostradas na seção 2.1.5 para a determinação do valor numérico da matriz jacobiana para uma dada pose do UR3.

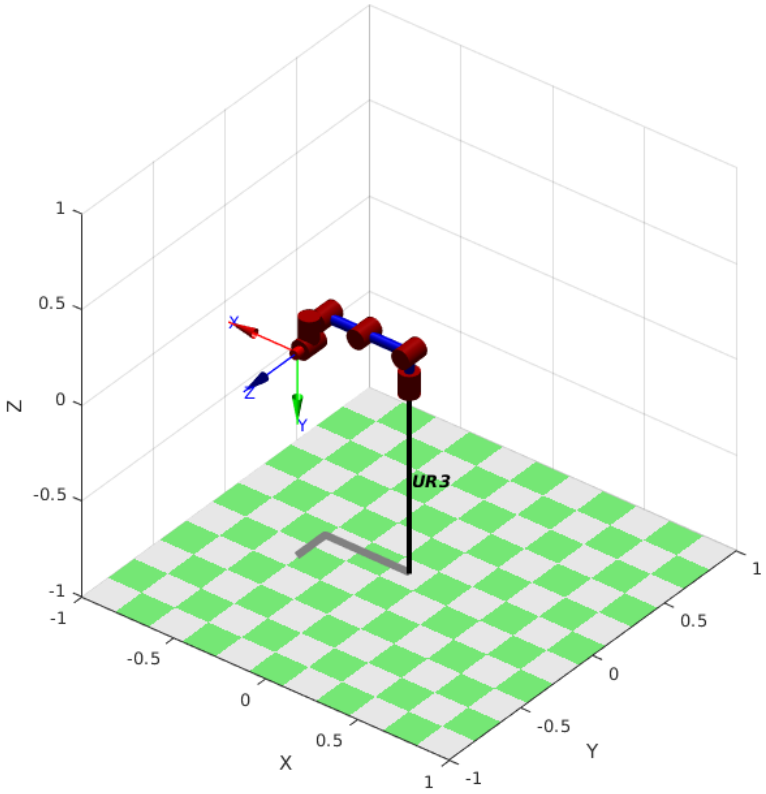

Figura 3.1 – Representação do UR3 usando a Toolbox de robótica do Matlab.

Como foi visto na seção 2.4.4, para implementar o controlador de trajetória da ferramenta é preciso utilizar a cinemática direta e a matriz jacobiana para que seja possível realimentar os sinais do robô para o controlador. A figura 3.2 mostra o diagrama feito utilizando Simulink que recebe as leituras q e $\dot{q}$ vindas do simulador e calcula x, $\dot{x}$, a jacobiana e a derivada

da jacobiana. O bloco *Direct Kinematics* é uma função em Matlab que chama as funções desenvolvidas que calculam a cinemática e a matriz jacobiana.

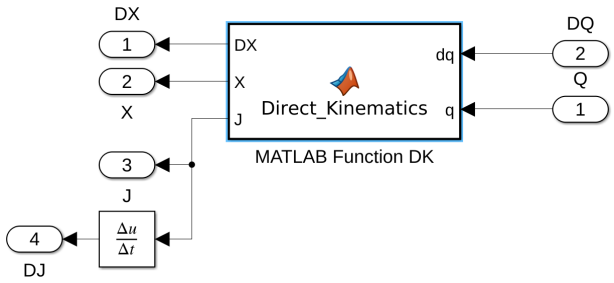

Figura 3.2 – Diagrama de blocos em Simulink que implementa a cinemática direta.

Um algoritmo que implementa as equações que determinam as trajetórias com suavização parabólica, mostrada na seção 2.1.4, que seriam utilizadas nos controladores foi desenvolvido em ambiente Matlab. Essa função recebe como entrada o tempo de execução, o período de amostragem, aceleração e os valores iniciais e finais de ângulo cada junta no caso do controle de juntas ou de posição espacial e orientação no caso do controle cartesiano. A saída da função é o valor de posição, velocidade e aceleração para cada instante de tempo do movimento. Esse algoritmo funciona de forma *offline* em relação ao controlador, ou seja, a trajetória é calculada antes da realização do controle.

3.2 ESCRITA DAS EQUAÇÕES DINÂMICAS

O próximo passo foi desenvolver um algoritmo que realizasse a modelagem dinâmica inversa do manipulador. Uma função que implementa todos os passos da seção 2.2.1 e reescreve a equação no formato clássico 2.59 foi escrita em Matlab. Como um dos objetivos do trabalho foi desenvolver um algoritmo que pudesse ser reutilizado para outros manipuladores a função necessita das seguintes definições:

- Orientação do eixo z_0 do sistema de coordenadas de referência,
- Matriz de rotação entre cada sistema de coordenada adjacente, R_i^{i+1},
- Matriz de rotação entre cada sistema de coordenada de junta e o sistema de coordenada fixo, R_0^i,
- O formato genérico de cada matriz de inercia das juntas,
- Quais dos três eixos devem ser considerados para cada comprimento de de link e para cada distância do centro de massa.

O último ponto dos itens acima serve para simplificar o resultado retirando complexidade desnecessária, por exemplo no caso do UR3 a distância entre o link 1 e o link 2, na nomenclatura definida l_1, pode ser considerado apenas o eixo y.

Para o UR3 foram considerados esses itens, na notação do capítulo 2.2.1, como:

- $z_0 = [0; 0; 1]$,
- Matrizes de rotação iguais as 3.1 a 3.6,
- Matrizes de rotação iguais as 3.7 a 3.11,
- Matrizes de inercia com apenas valores na diagonal principal, $I_i = \begin{bmatrix} I_{ix} & 0 & 0 \\ 0 & I_{iy} & 0 \\ 0 & 0 & I_{iz} \end{bmatrix}$,
- $l_1 = [0; l_{1y}; 0]$,
- $l_2 = [-l_{2x}; 0; l_{2z}]$,
- $l_2 = [-l_{2x}; 0; -l_{2z}]$,
- $l_3 = [-l_{3x}; 0; l_{3z}]$,
- $l_4 = [0; l_{4y}; 0]$,
- $l_5 = [0; -l_{5y}; 0]$,
- $lc_1 = [0; lc_{1y}; 0]$,
- $lc_2 = [-lc_{2x}; 0; lc_{2z}]$,
- $lc_2 = [-lc_{2x}; 0; -lc_{2z}]$,
- $lc_3 = [-lc_{3x}; 0; lc_{3z}]$,
- $lc_4 = [0; lc_{4y}; 0]$,
- $lc_5 = [0; -lc_{5y}; 0]$,
- $lc_5 = [0; 0; lc_{6z}]$

Foi feito em Simulink um bloco chamado de "Robo", mostrado na figura 2.74, que realiza a dinâmica direta com o objetivo de simular o manipulador. Esse bloco recebe como entrada um torque Tau que somado com uma perturbação e os sinais anteriores de velocidade e posição entra em uma função de Matlab que simplesmente calcula o valor numérico das matrizes G, M, C e B para determinar o valor da equação 2.74. Então, com o valor da

aceleração angular, simplesmente é realizada a integração obtendo como saída velocidade e posição angular.

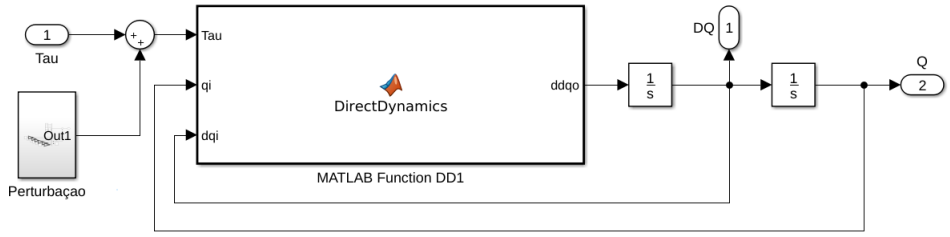

Figura 3.3 – Diagrama de blocos que simula o comportamento dinâmico do UR3.

Devido à pandemia provocada pelo corona-vírus, a suspensão das atividades presenciais não essenciais por parte da UnB e também por segurança pessoal, optou-se, em comum acordo com o orientador, pela condução das atividades em ambiente de simulação e para ter um grau mais realista nos testes feitos foi inserido um erro no cálculo da dinâmica direta no bloco "Robo". O erro foi definido como um multiplicador no valor final calculado da aceleração das juntas de modo a diminui-lo um valor percentual. Esse multiplicador foi escolhido como valor fixo igual ao *fitting* da identificação para cada junta individualmente.

3.3 IDENTIFICAÇÃO DOS PARÂMETROS DO MANIPULADOR

Uma vez em posse das expressões simbólicas do torque obtidas via o algoritmo de Newton-Euler foi feito o procedimento de simplificação utilizando as funções no software Maple como na equação 2.78. Então foi separado manualmente a partes lineares, parâmetros, e não lineares, regressores, para cada torque como na expressão matricial 2.77. A tabela 3.1 mostra quantos parâmetros dinâmicos cada junta tem após a simplificação do Maple.

Tabela 3.1 – Número de parâmetros de cada torque.

	Junta 6	**Junta 5**	**Junta 4**	**Junta 3**	**Junta 2**	**Junta 1**
Número de Parâmetros	3	14	20	26	31	28

Não é viável mostrar neste relatório as expressões para os regressores, pois cada um tem tamanho consideravelmente grande e, para ilustrar isso, vai ser mostrado os regressores da junta 5 no apêndice A. Já os parâmetros de todas as juntas vão ser explicitados nas tabelas com o resultado da identificação na seção 4.

Com as matrizes Y_i e θ_i definidas é possível escrever um algoritmo que implementa a equação dos mínimos quadrados para identificar os parâmetros lineares de cada uma das seis

juntas de forma independente. Isto é possível pelo fato das equações de cada junta estarem escritas de forma separada nas matrizes.

Dessa forma foi feito um algoritmo em Matlab que primeiramente carrega a matriz Y_i substituindo as medidas de q, $\dot{q}$ e $\ddot{q}$ em cada instante de tempo do experimento. Essa matriz Y_i tem tamanho o por m em que o é o número de instantes no experimento e m é o número de parâmetros de θ_i.

Após preencher as matrizes Y_i foi escrita a equação dos mínimos quadrados para cada torque partindo da junta 6 indo até a junta 1. Porém foi observado que alguns parâmetros dinâmicos são propagados da primeira junta para as seguintes no passo da recursão inversa do algoritmo de Newton-Euler como é mostrado na seção 2.2.1. Por exemplo o segundo parâmetro da junta 6 I_{6z} reaparece como decimo quarto parâmetro da junta 5.

Para que esses parâmetros propagados não fossem identificados mais de uma vez, possivelmente com valores diferentes, foram localizadas cada uma dessas repetições para utilizar tais valores como fixos iguais ao valor já identificado. Essa abordagem melhorou o resultado da identificação. Os parâmetros propagados podem ser vistos nas tabelas 4.4 a 4.9 da seção 4.2 de resultados.

Foi utilizado um dos experimentos realizados pelo aluno Pedro Garcia [23] como massa de dados de entrada. A figura 3.4 mostra a posição e a figura 3.5 mostra o torque lido de cada junta durante o experimento. Antes de utilizar esses dados foram retirados os pontos dos primeiros 5 segundos e dos últimos 2,5 segundos, pois há movimento nesses instantes. Após esse corte foi separado 70% do experimento para treinamento e os 30% restante para validação.

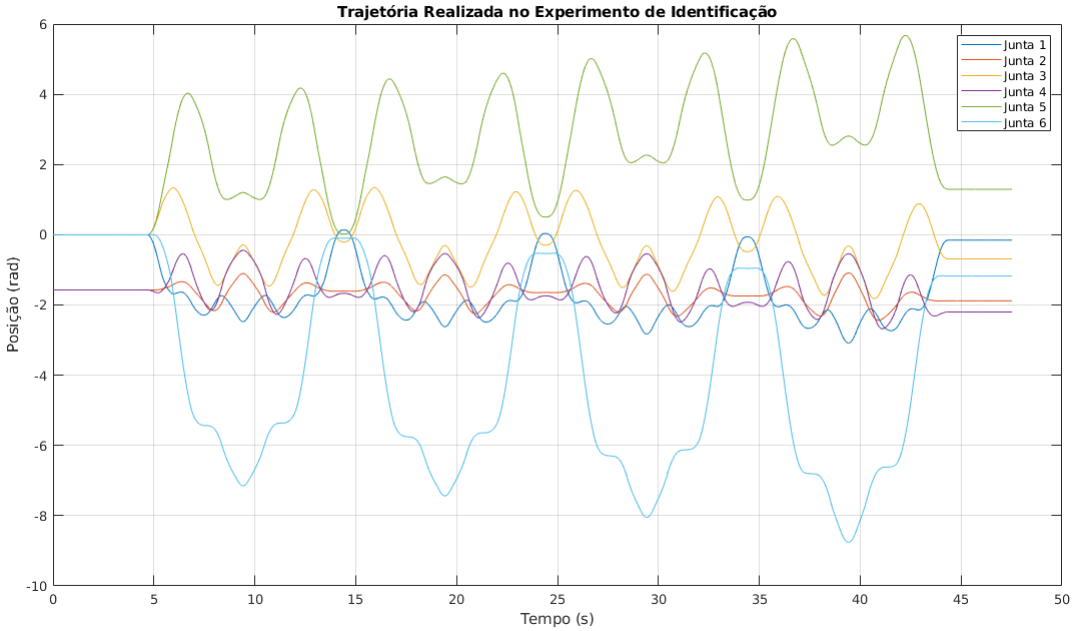

Figura 3.4 – Gráfico da trajetória feita, para cada junta, durante o experimento de identificação utilizado.

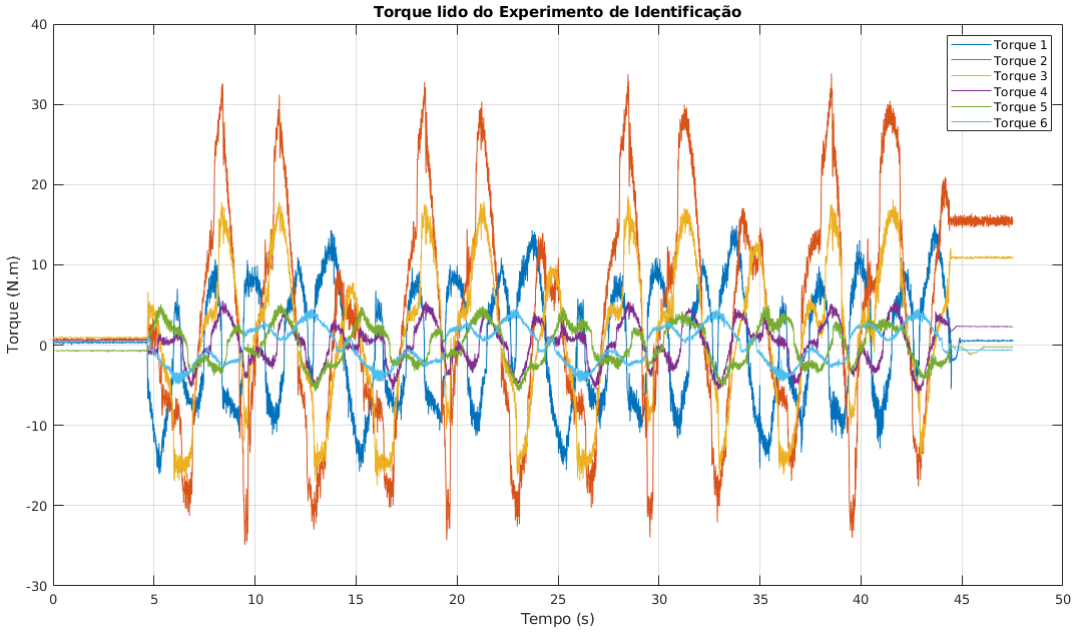

Figura 3.5 – Gráfico com o sinal de torque de cada junta durante o experimento de identificação utilizado.

Após o processo de identificação das equações foi utilizado a função 3.12 para calcular o *fitness* de cada conjunto de parâmetros. Para isso, foi feito o mesmo procedimento para preencher as matrizes Y_i, agora com os dados de validação, e, então, multiplicou-se pelos respectivos vetores identificados θ_i para gerar o vetor de torque estimado $\hat{\tau}_i$ então esses valores estimados são comparados com os vetores reais lidos τ_i.

A função 3.12 implementa o erro quadrático médio normalizado e é o padrão para determinação de *fitness* na *toolbox* de identificação do Matlab. Ela mostra, em porcentagem, o quão bem o modelo gerado minimiza os resíduos, justamente o que o algoritmo de mínimos quadrados busca fazer, além disso esse valor é normalizado para facilitar a comparação entre dados com escalas diferentes.

$$FIT = 100(1 - norm(\hat{\tau}_i - \tau_i)/norm(\hat{\tau}_i - mean(\hat{\tau}_i))); \tag{3.12}$$

Uma vez que o processo de identificação acha parâmetros considerados bons é preciso voltar a escrita da equação do torque para o formato clássico substituindo os parâmetros simbólicos pelos respectivos identificados. Para isso foi utilizado o mesmo algoritmo usado na seção 3.2. Esse procedimento é necessário para a implementação dos controladores propostos já que a arquitetura explicada na seção 2.4.3 e na seção 2.4.4 requerem a separação da matriz inercial das matrizes gravitacional e de Coriolis.

3.4 PROPOSTAS DE CONTROLADORES

Depois de obter a equação dinâmica e determinar o melhor conjunto de parâmetros dinâmicos é possível fazer esquemas de controle que levam a equação dinâmica em consideração. Apesar de ser possível fazer controle de posição sem esse conhecimento, é interessante desenvolver esquemas de controle de trajetória que sejam capazes de estimar a força que precisam aplicar para realizar o movimento, além de ser um passo importante para o desenvolvimento de arquiteturas de controle de força. Como foi impossibilitado o uso do robô real, todo o desenvolvimento e todos os testes tiveram que ser feitos em ambiente de simulação.

3.4.1 Controle de Acompanhamento de Trajetória das Juntas

O primeiro controlador desenvolvido foi um com objetivo de controlar o acompanhamento da trajetória das juntas, ou seja variáveis de junta deviam ser manipuladas de forma a acompanhar uma trajetória e não apenas uma referência fixa de posição ou velocidade. A arquitetura externa deste controlador pode ser vista na figura 3.6. Nela podemos ver da esquerda para direita: o bloco que recebe a trajetória da área de trabalho do Matlab, o esquema com o cálculo do erro de trajetória e do valor de aceleração desejado, um bloco de função Matlab que implementa as equações da dinâmica inversa e, por fim, o bloco de simulação do robô da figura 3.3.

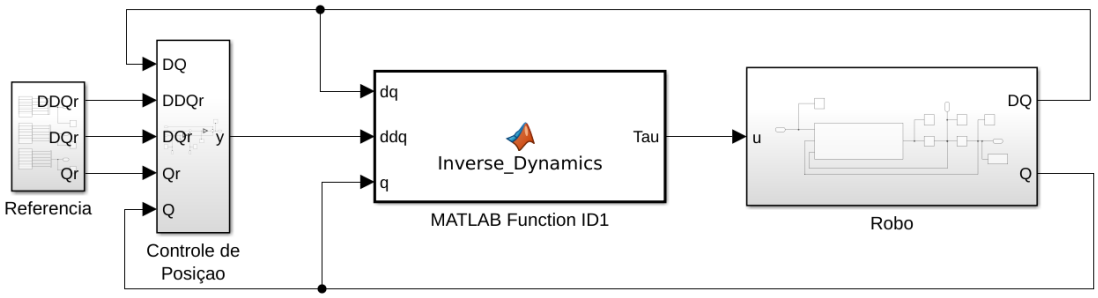

Figura 3.6 – Diagrama de blocos que implementa o controle de trajetória das juntas.

A figura 3.7 mostra o bloco que faz a importação da trajetória de referência que é composta pela posição, velocidade e aceleração desejadas para cada junta. Elas são importadas da área de trabalho do Matlab como uma estrutura de dados gerada pela função de geração de trajetória. Nessa estrutura de dados também é definido o vetor de tempo que foi amostrado a uma frequência escolhida e a disposição inicial das juntas que é necessário definir no bloco do robô como condição inicial do segundo integrador que calcula a posição do manipulador enquanto a condição inicial da velocidade foi determinada para zero.

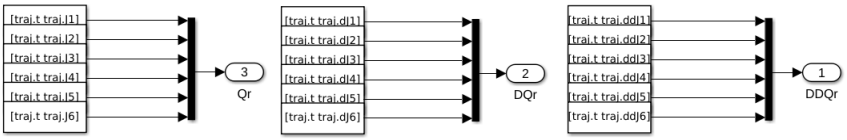

Figura 3.7 – Referências de trajetória para o controlador de juntas.

Já o bloco chamado de "controle de posição" pode ser visto com detalhes na figura 3.8. Nele é feito o cálculo do erro de posição, do erro de velocidade, da integral do erro de posição e feita a soma dos erros multiplicados pelos respectivos ganhos mais a aceleração de referência. Essa soma é então passada para o bloco chamado de "Inverse Dynamics" que recebe esse valor como uma aceleração desejada além da posição e da velocidade atuais do manipulador.

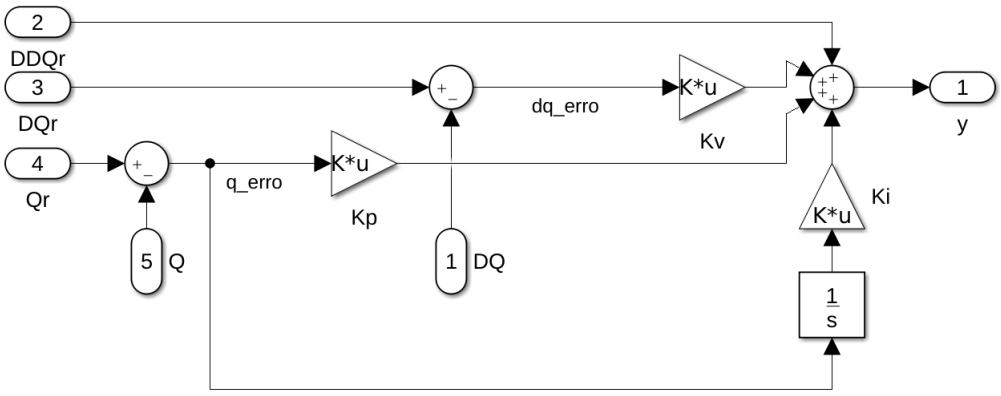

Figura 3.8 – Diagrama de blocos do controlador de trajetória das juntas.

A função "Inverse Dynamics" basicamente preenche os valores numéricos das matrizes G, M, C e B para usar na equação 2.106 determinando o torque a ser aplicado em cada junta. Dessa forma é possível ter uma noção da magnitude da força requerida para realizar a trajetória desejada. Por fim o valor de torque é passado para o bloco de simulação do robô que, por sua vez, calcula a aceleração obtida e suas integrais.

Seguindo o raciocínio desenvolvido na seção 2.4.2 foi escolhido para os experimentos sem canal integral os ganhos: $K_p = 49$ e $K_v = 14$. Porém em ambientes com a presença de perturbações no torque foi necessário o incremento da componente integral ao erro e com ele foi necessário a escolha de outros ganhos. Com o objetivo de manter o sistema criticamente amortecido e aproximadamente um sistema de segunda ordem foi escolhido a posição dos polos do controlador como = [-20 -2 -2]. Para que se tenha a equação da forma 2.115 com esses polos basta escolher os ganhos como $K_v = 24$, $K_p = 84$ e $K_i = 80$.

3.4.2 Controle de Acompanhamento de Trajetória da Ferramenta

Uma arquitetura de controladores mais interessante quando se trata de manipuladores é o controle de acompanhamento de trajetória da ferramenta ou também chamado de controle no espaço cartesiano. A figura 3.9 mostra o diagrama de blocos externo que implementa tal controlador no Simulink. Nela podemos ver que o formato é bem parecido com o controlador de juntas e, na realidade, o bloco de referência do controle de posição e o simulador do robô são iguais. A única mudança neles é a troca da variável de ângulo q e suas derivadas por posição e orientação x e as respectivas derivadas.

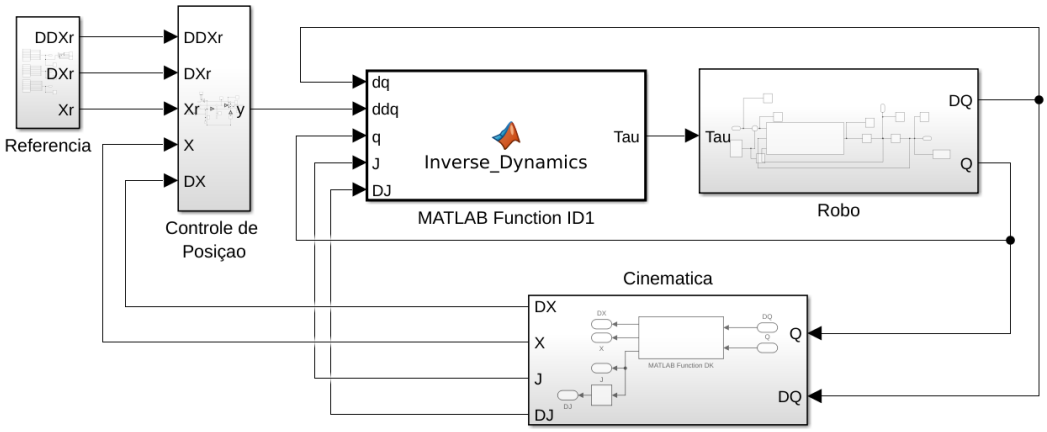

Figura 3.9 – Diagrama de blocos que implementa o controle de trajetória das juntas.

Nas figuras 3.10 e 3.11, pode-se ver o bloco de referência e o de controle de posição que são virtualmente idênticos aos do controlador de juntas e tem a mesma função.

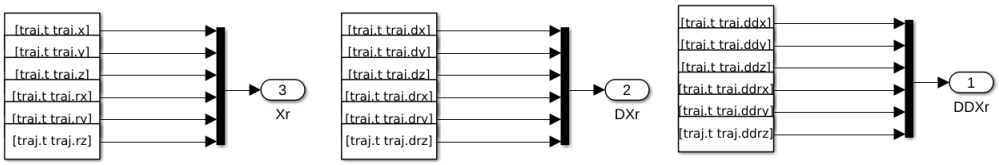

Figura 3.10 – Referências de trajetória para o controlador no espaço cartesiano.

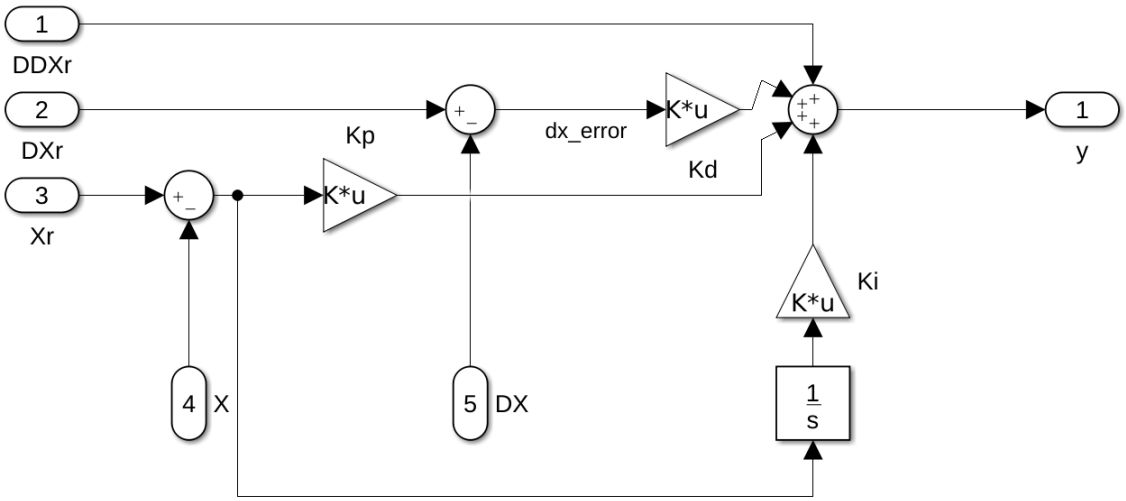

Figura 3.11 – Diagrama de blocos do controlador de trajetória da ferramenta.

O bloco "Inverse Dynamics" neste controlador tem o mesmo objetivo do anterior, porém agora é necessário o cálculo de transformação das matrizes dinâmicas do espaço das jutas para o espaço cartesiano utilizando equações 2.71, 2.72 e 2.73. Portanto esse bloco realiza as mesmas operações que já fazia no primeiro controlador mais essa transformação das matrizes

para calcular o vetor força-torque desejado e por fim o transforma em um vetor de torque nas juntas via a equação 2.119.

O último bloco necessário para o controle é o chamado "Cinematica". Ele tem os objetivos de transformar as variáveis de junta da saída do simulador do robô em variáveis cartesianas, utilizando as equações 2.120 e 2.121, e calcular o valor atual da matriz jacobiana e sua derivada para serem usados no bloco "Inverse Dynamics".

Os mesmos ganhos do controlador de juntas foram utilizados aqui, primeiramente como ponto de partida, porém o seu resultado foi satisfatório então decidiu-se manter os ganhos $K_p = 49$ e $K_v = 14$ para o caso sem integral e $K_p = 84$, $K_v = 24$ e $K_i = 80$ para o controlador com o ganho integral.

Durante o desenvolvimento deste controlador foi encontrado um problema com relação à forma de representar a orientação da ferramenta. Essa representação tem pontos de singularidade que acontecem quando β da equação 2.11 é igual a $\pm\frac{\pi}{2}$ fazendo com que o $cos(\beta)$ das equações 2.55 e 2.13 seja igual a zero e consequentemente tornem seus valores indefinidos. Quando isso ocorre é padrão escolher o valor de α igual a zero e $\gamma = Atan2(r_{12}, r_{22})$, além disso os valores dos ângulos α e γ são limitados a $\pm\pi$ e β limitado a $\pm\frac{\pi}{2}$.

Essas duas características da representação escolhida para a orientação da ferramenta impossibilitou o controle proporcional de seu erro, pois o seu valor mudava bruscamente entre os extremos como pode ser visto na figura 3.12 em que o controlador de apenas as variáveis de orientação foi desligado, pois do contrário a simulação daria erro e pararia. Uma forma de tentar contornar isso, apesar de não ideal, foi utilizar para o controle da orientação apenas o ganho K_v assim o controle seria proporcional apenas ao erro de velocidade.

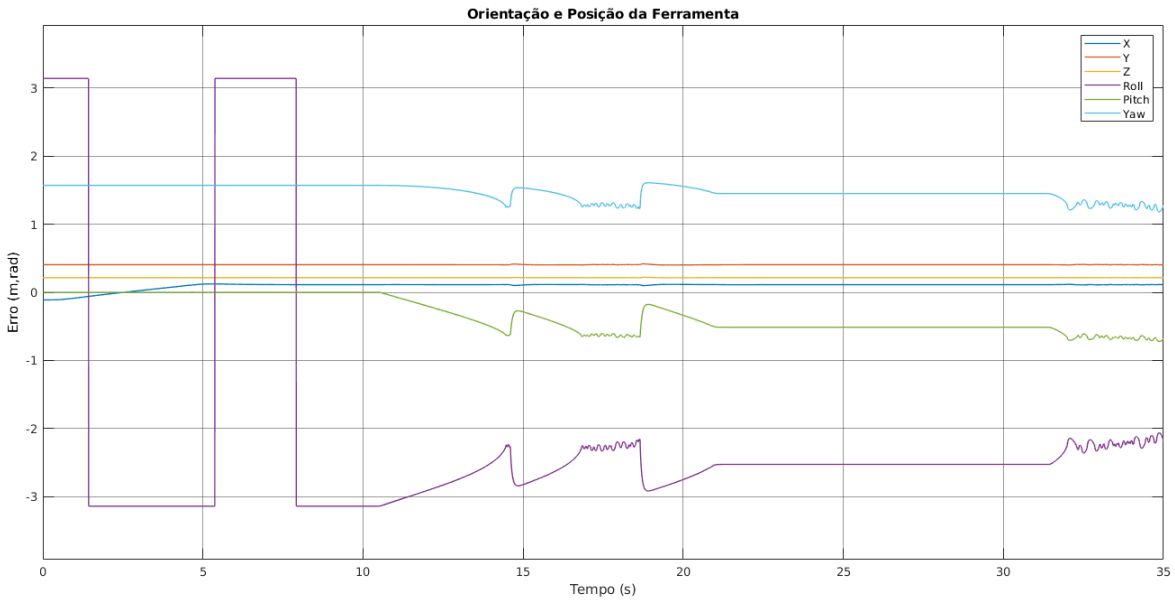

Figura 3.12 – Comportamento não desejado da orientação na representação utilizada.

Outro problema encontrado durante o desenvolvimento deste controlador foi a inserção de perturbação de torque. Ao se aplicar a perturbação o controlador ficava instável e explodia o valor das variáveis do robô. Por esse motivo também foi deixado de lado esse aspecto durante as simulações deste controlador.

4 RESULTADOS E ANALISE DOS EXPERIMENTOS

Neste capítulo são mostrados os experimentos realizados e suas análises para verificar a qualidade e exatidão das soluções propostas. Além de uma análise crítica dos pontos em que não se obteve bons resultados.

4.1 CINEMÁTICA

Com o objetivo de confirmar o algoritmo de cinemática direta, foram definidas três poses para o robô, tabela 4.1, para usar como dado de entrada para a função escrita e, então, comparou-se com a saída da *Toolbox* de robótica do Matlab. Na tabela 4.2, pode-se ver o resultado do algoritmo e nas figuras 4.1, 4.2 e 4.3, pode-se ver as figuras geradas pela *Toolbox*.

Tabela 4.1 – Conjunto de ângulos para cada junta escolhidos para os três experimentos.

	Junta 1	**Junta 2**	**Junta 3**	**Junta 4**	**Junta 5**	**Junta 6**
Posição 1	0º	0º	0º	0º	0º	0º
Posição 2	0º	-90º	0º	-90º	0º	0º
Posição 3	45º	22,5º	-45º	-157,5º	-90	0º

Tabela 4.2 – Posição do efetuador terminal para cada conjunto de ângulos definido.

	Posição 1	**Posição 2**	**Posição 3**
X	0.4570	0.0	0.2773
Y	0.1940	0.1940	0.4357
Z	0.0670	0.6940	0.2251
Roll	0	-180.0	-180.0
Pitch	0	0.0	45.0
Yaw	-90.0	-90.0	-90.0

Nas imagens da saída da *Toolbox* é possível ver os valores de posição e orientação dados os ângulos de junta. Podemos ver que o algoritmo desenvolvido acerta em todos os casos de teste. É possível que a *Toolbox* calcule esses valores de forma idêntica a implementada.

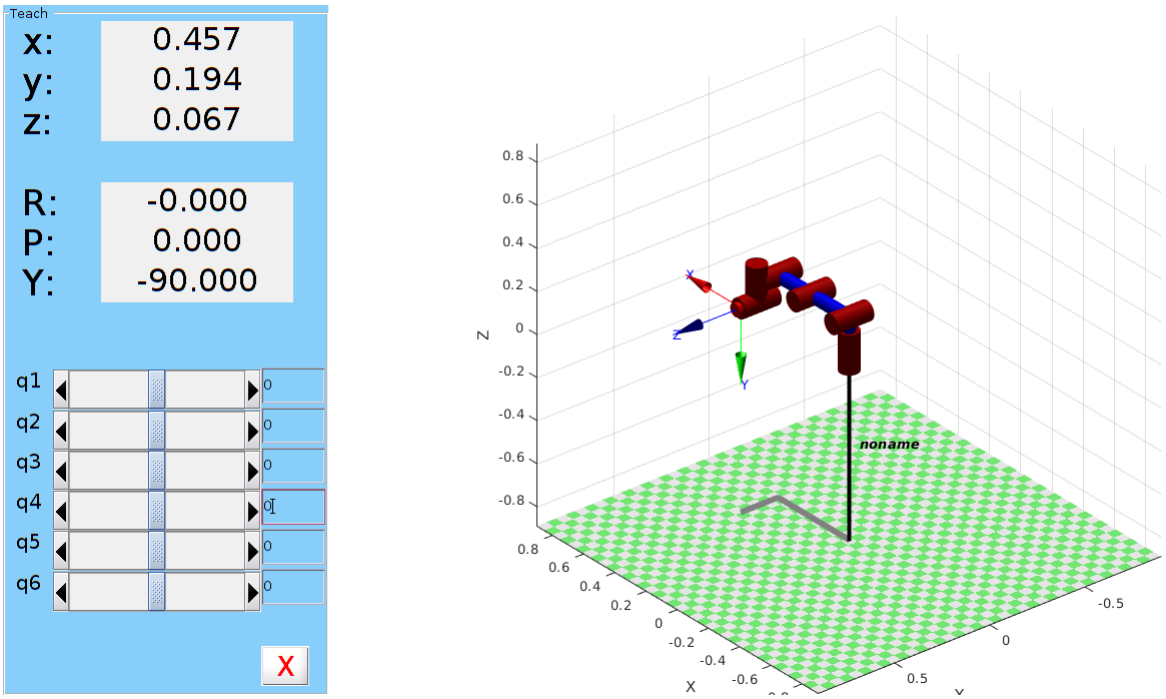

Figura 4.1 – Imagem do robô na posição 1 via *toolbox* de robótica do Matlab.

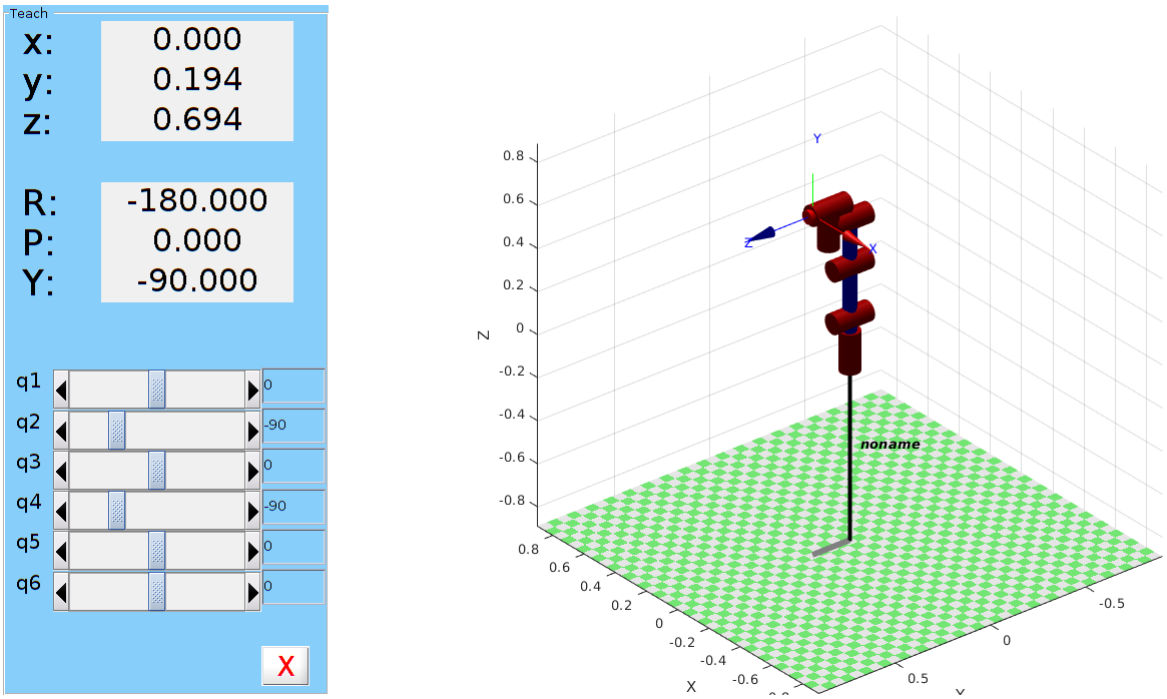

Figura 4.2 – Imagem do robô na posição 2 via *toolbox* de robótica do Matlab.

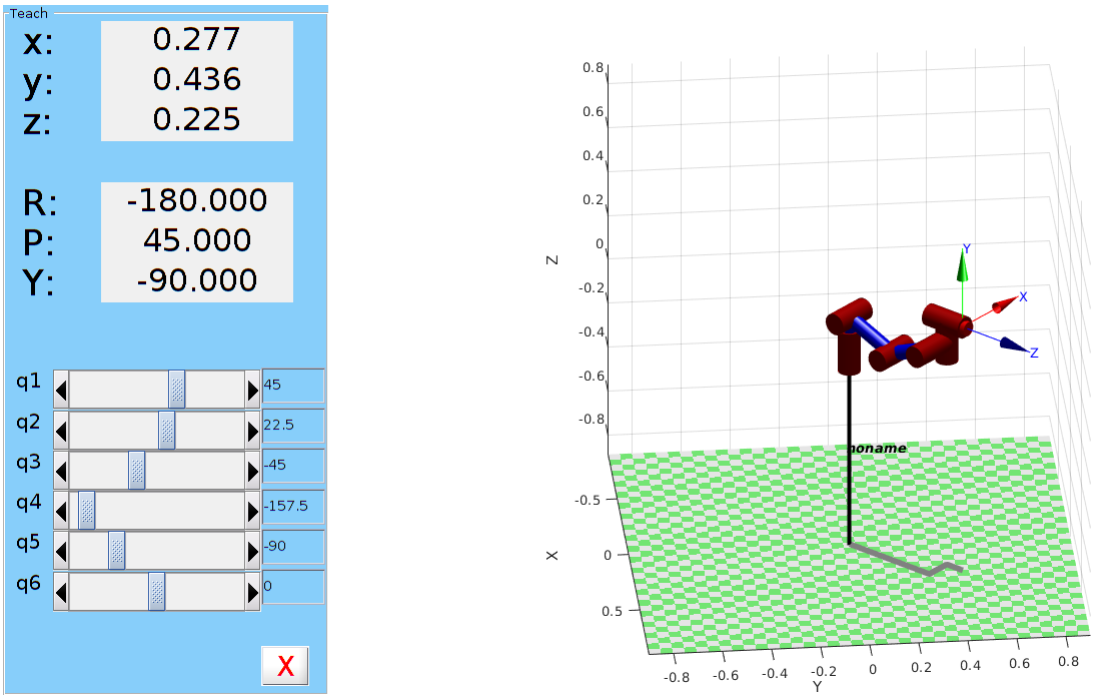

Figura 4.3 – Imagem do robô na posição 3 via *toolbox* de robótica do Matlab.

4.2 IDENTIFICAÇÃO

Os gráficos com a validações dos modelos identificados podem ser vistos nas figuras 4.4 a 4.9. As imagens mostram que o modelo explica bem o real comportamento do torque, mas, ao mesmo tempo, indicam que existem efeitos dinâmicos que não foram contemplados durante a modelagem. Por volta dos 34 segundos da curva de torque da junta 6, figura 4.9, é possível ver que há um pulo momentâneo no torque medido que o modelo não foi capaz de prever. Esses efeitos podem ser devidos algum comportamento físico não considerado ou então devido uma condição ruim durante o experimento. Uma possibilidade para essas anormalidades é que a base do robô não é 100% imóvel, ou seja, a força aplicada pelo próprio robô faz com que ele mova a mesa em que ele está aparafusado. Uma forma de resolver isso é trocar o local que o robô está instalado para uma torre com alta inercia.

O valor de *fitness* em porcentagem pode ser observado na tabela 4.3. Além disso os valores numéricos de cada um dos parâmetros estão nas tabelas 4.9 a 4.4.

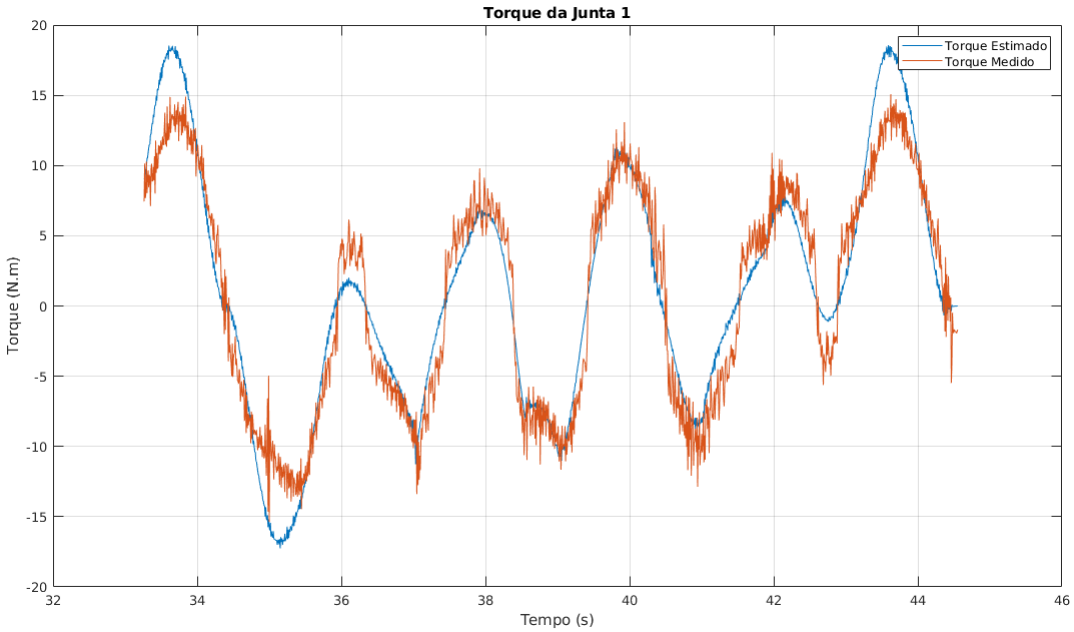

Figura 4.4 – Gráfico da validação da identificação para a junta 1 do robô.

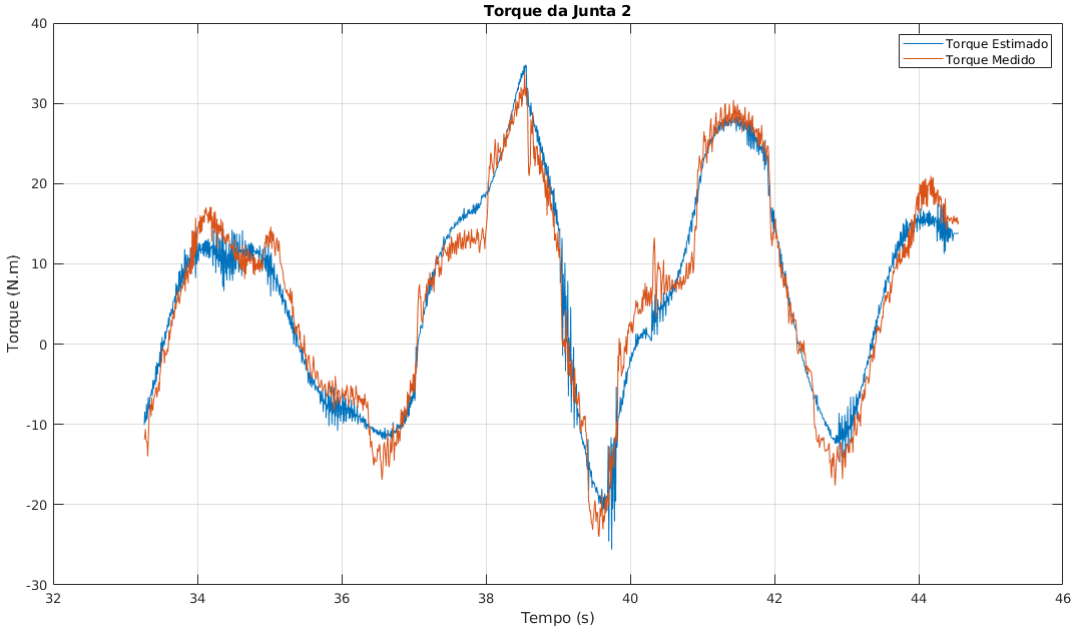

Figura 4.5 – Gráfico da validação da identificação para a junta 2 do robô.

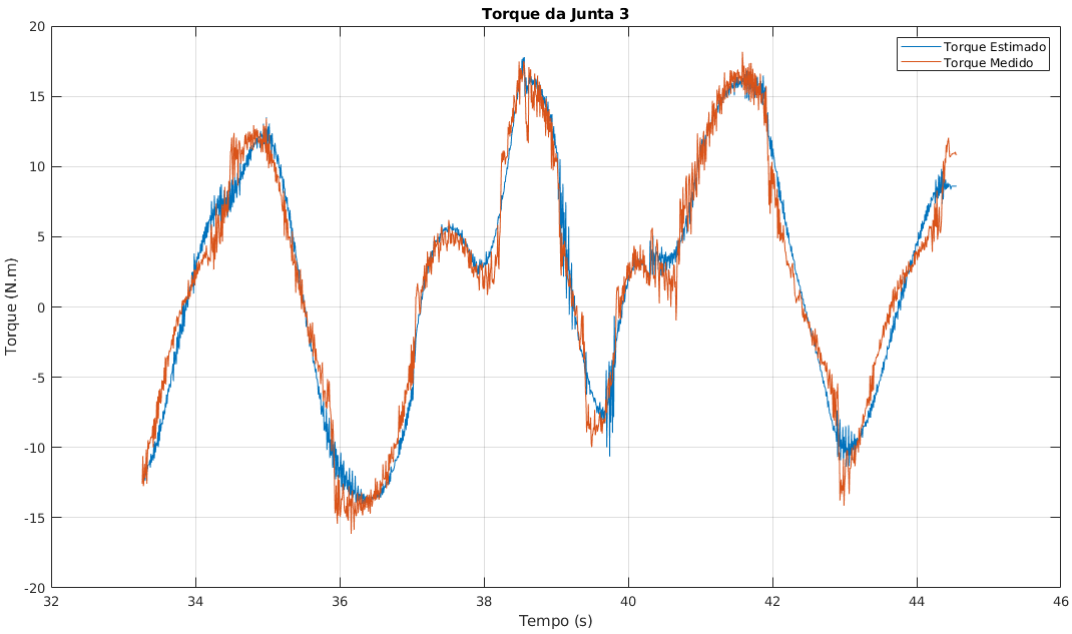

Figura 4.6 – Gráfico da validação da identificação para a junta 3 do robô.

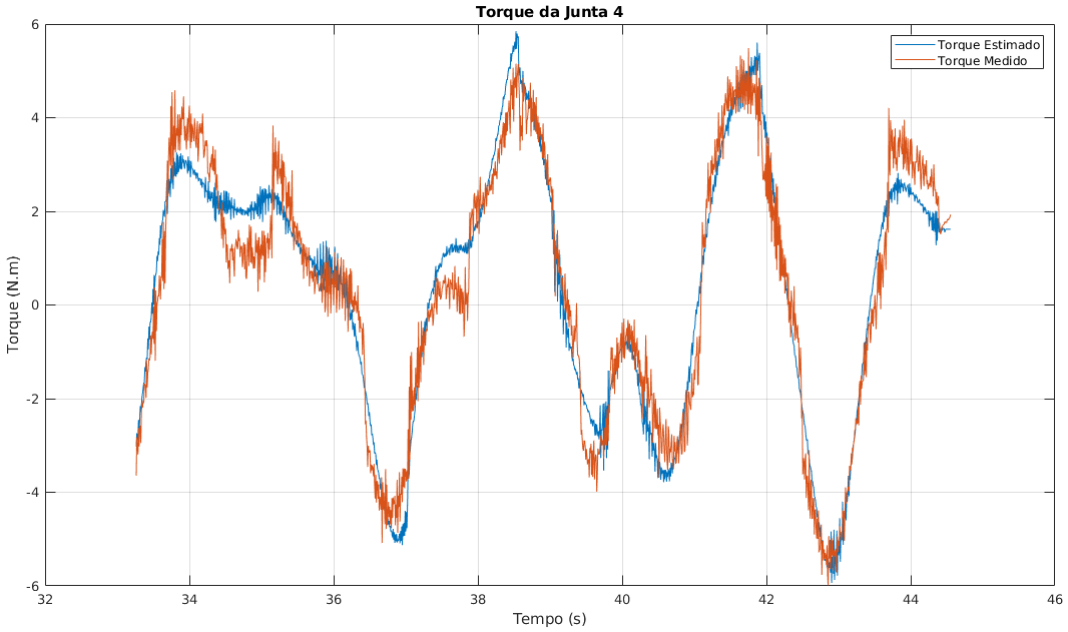

Figura 4.7 – Gráfico da validação da identificação para a junta 4 do robô.

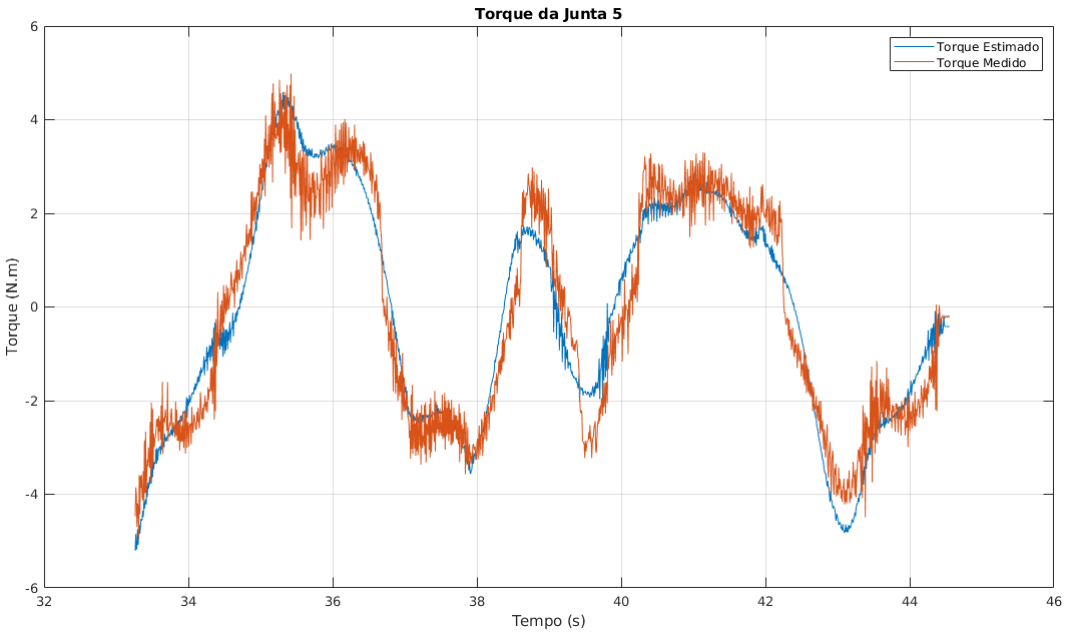

Figura 4.8 – Gráfico da validação da identificação para a junta 5 do robô.

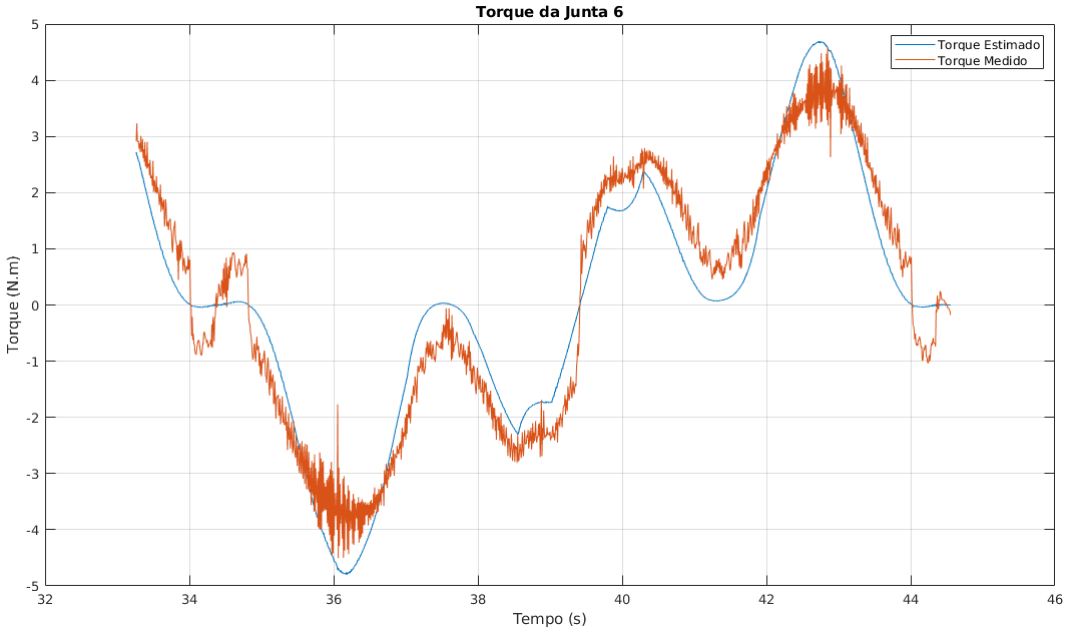

Figura 4.9 – Gráfico da validação da identificação para a junta 6 do robô.

Tabela 4.3 – Fitness para cada junta do robô obtida.

Junta 1	**Junta 2**	**Junta 3**	**Junta 4**	**Junta 5**	**Junta 6**
68,7391%	76,9917%	80,6167%	73,8611%	69,9604%	68,6581%

Tabela 4.4 – Valores identificados para os parâmetros dinâmicos presentes na expressão do torque da 6ª junta.

Parâmetro	**Valor**
$beta_6$	1,0892
I_{6z}	0,0021
$(I_{6x} - I_{6y})$	-0,0120

Tabela 4.5 – Valores identificados para os parâmetros dinâmicos presentes na expressão do torque da 5ª junta.

Parâmetro	**Valor**	**Parâmetro**	**Valor**
$m_6 * lc_{6z}^2$	0,0106	$beta_5$	1,2657
$m_6 * lc_{6z} * l_{3x}$	0,0354	I_{5x}	0,0083
$m_6 * lc_{6z} * l_{2x}$	0,0525	I_{5y}	0,0025
$m_6 * lc_{6z} * l_{3z}$	0,0114	I_{5z}	-0,0084
$m_6 * lc_{6z} * l_{5y}$	0,0065	I_{6x}	0,0030
$(m_6 * lc_{6z} * l_{2z} + m_6 * lc_{6z} * l_{4y})$	-0,0114	I_{6y}	0,0076
$g * lc_{6z} * m_6$	1,5420	I_{6z}	0,0021

Tabela 4.6 – Valores identificados para os parâmetros dinâmicos presentes na expressão do torque da 4ª junta.

Parâmetro	**Valor**	**Parâmetro**	**Valor**
$(m_6 * l_{5y}^2 + m_5 * lc_{5y}^2)$	0,0336	$m_6 * lc_{6z} * l_{3x}$	0,0354
$(l_{4y} * l_{5y} * m_6 + l_{5y} * l_{2z} * m_6 - l_{5y} * l_{3z} * m_6 + l_{4y} * lc_{5y} * m_5 + l_{2z} * lc_{5y} * m_5 - l_{3z} * lc_{5y} * m_5)$	0,0077	$beta_4$	1,7881
$(g * lc_{5y} * m_5 + g * l_{5y} * m_6)$	1,7078	$(I_{4z} - I_{4x})$	-0,0462
$g * lc_{6z} * m_6$	1,5420	I_{4y}	-0,0130
$(m_6 * lc_{6z} * l_{2z} + m_6 * lc_{6z} * l_{4y} - m_6 * lc_{6z} * l_{3z})$	-0,0228	I_{5x}	0,0083
$(m_5 * lc_{5y} * l_{2x} + m_6 * l_{5y} * l_{2x})$	0,0053	I_{5y}	0,0025
$(m_5 * lc_{5y} * l_{3x} + m_6 * l_{5y} * l_{3x})$	0,0251	I_{5z}	-0,0084
$m_6 * lc_{6z} * l_{2x}$	0,0525	I_{6x}	0,0030
$m_6 * lc_{6z} * l_{5y}$	0,0065	I_{6y}	0,0076
$m_6 * lc_{6z}^2$	0,0106	I_{6z}	0,0021

Tabela 4.7 – Valores identificados para os parâmetros dinâmicos presentes na expressão do torque da 3ª junta.

Parâmetro	**Valor**
$g * lc_{6z} * m_6$	1.5420
$(g * lc_{5y} * m_5 + g * l_{5y} * m_6)$	1.7078
$(g * lc_{3x} * m_3 + g * l_{3x} * m_5 + g * l_{3x} * m_4 + g * l_{3x} * m_6)$	8.2945
$(m_6 * l_{5y}^2 + m_5 * lc_{5y}^2)$	0.0077
$(l_{4y} * lc_{5y} * m_5 - l_{5y} * l_{3z} * m_6 + l_{5y} * l_{2z} * m_6 + l_{4y} * l_{5y} * m_6 - l_{3z} * lc_{5y} * m_5 + l_{2z} * lc_{5y} * m_5)$	0.0336
$(m_5 * lc_{5y}^2 + m_6 * l_{5y}^2)$	0.1016
$(m_4 * l_{3x}^2 + m_3 * lc_{3x}^2 + m_6 * l_{3x}^2 + m_5 * l_{3x}^2)$	0.0581
$(-l_{3x} * l_{3z} * m_4 + l_{3x} * l_{2z} * m_5 + l_{3x} * l_{4y} * m_5 - lc_{3x} * lc_{3z} * m_3 + l_{3x} * l_{2z} * m_6 + l_{3x} * l_{2z} * m_4 + l_{3x} * lc_{4y} * m_4 + l_{3x} * l_{4y} * m_6 - l_{3x} * l_{3z} * m_5 + l_{2z} * lc_{3x} * m_3 - l_{3x} * l_{3z} * m_6$	0.0106
$(m_6 * lc_{6z} * l_{3z} - m_6 * lc_{6z} * l_{2z} - m_6 * lc_{6z} * l_{4y}))$	0.0228
$m_6 * lc_{6z} * l_{2x}$	0.0525
$(m_5 * lc_{5y} * l_{2x} + m_6 * l_{5y} * l_{2x})$	0.0053
$m_6 * lc_{6z} * l_{3x}$	0.0354
$m_6 * lc_{6z} * l_{5y}$	0.0065
$(m_4 * l_{3x} * l_{2x} + m_5 * l_{3x} * l_{2x} + m_6 * l_{3x} * l_{2x} + m_3 * lc_{3x} * l_{2x})$	0.1640
$(m_5 * lc_{5y} * l_{3x} + m_6 * l_{5y} * l_{3x})$	0.0251
$beta_3$	4.1782
$(I_{3x} - I_{3y})$	-0.1116
I_{3z}	-0.0110
$(I_{4z} - I_{4x})$	0.0462
I_{4y}	-0.0130
I_{5x}	0.0083
I_{5y}	0.0025
I_{5z}	-0.0084
I_{6x}	0.0030
I_{6y}	0.0076
I_{6z}	0.0021

Tabela 4.8 – Valores identificados para os parâmetros dinâmicos presentes na expressão do torque da 2ª junta.

Parâmetro	**Valor**
$(l_{3x} * l_{4y} * m_6 + l_{3x} * lc_{4y} * m_4 - l_{3x} * l_{3z} * m_5 + l_{3x} * l_{2z} * m_6 - l_{3x} * l_{3z} * m_6 + l_{3x} * l_{2z} * m_5 + l_{3x} * l_{4y} * m_5 - l_{3x} * l_{3z} * m_4 - lc_{3x} * lc_{3z} * m_3 + l_{2z} * lc_{3x} * m_3 + l_{3x} * l_{2z} * m_4)$	0,0581
$(g * l_{3x} * m_4 + g * l_{3x} * m_5 + g * l_{3x} * m_6 + g * lc_{3x} * m_3)$	8,2945
$(m_6 * l_{2x}^2 + m_4 * l_{2x}^2 + m_2 * lc_{2x}^2 + m_3 * l_{2x}^2 + m_5 * l_{2x}^2)$	0,0571
$(m_6 * lc_{6z} * l_{4y} + m_6 * lc_{6z} * l_{2z} - m_6 * lc_{6z} * l_{3z})$	-0,0228
$(m_5 * l_{3x}^2 + m_6 * l_{3x}^2 + m_4 * l_{3x}^2 + m_3 * lc_{3x}^2)$	0,1016
$(g * l_{5y} * m_6 + g * lc_{5y} * m_5)$	1,7078
$(l_{2x} * l_{2z} * m_5 - l_{2x} * l_{3z} * m_5 + l_{2x} * l_{2z} * m_3 - l_{2x} * l_{3z} * m_4 + l_{2x} * l_{4y} * m_5 + l_{2x} * lc_{4y} * m_4 - l_{2x} * l_{3z} * m_6 + l_{2x} * l_{2z} * m_6 + l_{2x} * l_{2z} * m_4 - l_{2x} * lc_{3z} * m_3 + lc_{2x} * lc_{2z} * m_2 + l_{2x} * l_{4y} * m_6)$	0,1353
$(-l_{3z} * lc_{5y} * m_5 + l_{4y} * lc_{5y} * m_5 + l_{5y} * l_{2z} * m_6 + l_{4y} * l_{5y} * m_6 - l_{5y} * l_{3z} * m_6 + l_{2z} * lc_{5y} * m_5)$	0,0077
$(m_6 * l_{5y} * l_{3x} + m_5 * lc_{5y} * l_{3x})$	0,0251
$(m_6 * l_{5y}^2 + m_5 * lc_{5y}^2)$	0,0336
$(g * lc_{2x} * m_2 + g * l_{2x} * m_5 + g * l_{2x} * m_6 + g * l_{2x} * m_3 + g * l_{2x} * m_4)$	16,9322
$g * lc_{6z} * m_6$	1,5420
$m_6 * lc_{6z} * l_{5y}$	0,0065
$(m_5 * l_{3x} * l_{2x} + m_4 * l_{3x} * l_{2x} + m_6 * l_{3x} * l_{2x} + m_3 * lc_{3x} * l_{2x})$	0,1640
$(m_5 * lc_{5y} * l_{2x} + m_6 * l_{5y} * l_{2x})$	0,0053
$m_6 * lc_{6z}^2$	0,0106
$m_6 * lc_{6z} * l_{3x}$	0,0354
$m_6 * lc_{6z} * l_{2x}$	0,0525
$beta_2$	10,7669
$(I_{2y} - I_{2x})$	-0,0789
I_{2z}	0,1350
$(I_{3y} - I_{3x})$	0,1116
I_{3z}	-0.0110
$(I_{4z} - I_{4x})$	-0.0459
I_{4y}	-0.0130
I_{5x}	0.0083
I_{5y}	0.0025
I_{5z}	-0.0084
I_{6x}	0.0030
I_{6y}	0.0076
I_{6z}	0.0021

Tabela 4.9 – Valores identificados para os parâmetros dinâmicos presentes na expressão do torque da 1ª junta.

Parâmetro	**Valor**
$m_6 * lc_{6z} * l_{5y}$	0,0065
$m_6 * lc_{6z} * l_{2x}$	0,0525
$m_6 * lc_{6z} * l_{3x}$	0,0354
$(m_6 * lc_{6z} * l_{3z} - m_6 * lc_{6z} * l_{4y} + m_6 * lc_{6z} * l_{2z})$	0,0228
$m_6 * lc_{6z}^2$	0,0106
$(m_4 * l_{2x}^2 + m_3 * l_{2x}^2 + m_2 * lc_{2x}^2 + m_6 * l_{2x}^2 + m_5 * l_{2x}^2)$	0,0571
$(m_6 * l_{3z} * l_{5y} + m_5 * lc_{5y} * l_{3z} - m_5 * lc_{5y} * l_{4y} -$ $m_5 * lc_{5y} * l_{2z} - m_6 * l_{2z} * l_{5y} - m_6 * l_{5y} * l_{4y})$	-0,0388
$(m_6 * l_{3z} * l_{3x} + m_3 * lc_{3z} * lc_{3x} + m_5 * l_{3z} * l_{3x} +$ $m_4 * l_{3z} * l_{3x} - m_5 * l_{4y} * l_{3x} - m_6 * l_{4y} * l_{3x} - m_5 * l_{2z} * l_{3x} -$ $m_4 * lc_{4y} * l_{3x} - m_6 * l_{2z} * l_{3x} - m_4 * l_{2z} * l_{3x} - m_3 * lc_{3x} * l_{2z})$	-0,0500
$(m_5 * lc_{5y} * l_{3x} + m_6 * l_{5y} * l_{3x})$	0,0251
$(m_6 * l_{5y} * l_{2x} + m_5 * lc_{5y} * l_{2x})$	0,0053
$(m_4 * l_{3x} * l_{2x} + m_5 * l_{3x} * l_{2x} + m_6 * l_{3x} * l_{2x} + m_3 * lc_{3x} * l_{2x})$	0,1640
$(m_6 * l_{3z} * l_{2x} + m_5 * l_{3z} * l_{2x} + m_4 * l_{3z} * l_{2x} + m_3 * lc_{3z} * l_{2x} -$ $m_6 * l_{4y} * l_{2x} - m_3 * l_{2z} * l_{2x} - m_5 * l_{2z} * l_{2x} - m_4 * lc_{4y} * l_{2x} -$ $m_2 * lc_{2z} * lc_{2x} - m_4 * l_{2z} * l_{2x} - m_5 * l_{4y} * l_{2x} - m_6 * l_{2z} * l_{2x})$	-0,0759
$(m_5 * lc_{5y}^2 + m_6 * l_{5y}^2)$	0,0336
$(m_5 * l_{3x}^2 + m_6 * l_{3x}^2 + m_3 * lc_{3x}^2 + m_4 * l_{3x}^2)$	0,1016
$beta_1$	7,3285
$(-2 * l_{2z} * lc_{3z} * m_3 - 2 * l_{4y} * l_{3z} * m_6 - 2 * l_{2z} * l_{3z} * m_4 - 2 * l_{4y} * l_{3z} * m_5 +$ $2 * l_{4y} * l_{2z} * m_5 + 2 * l_{2z} * lc_{4y} * m_4 - 2 * l_{2z} * l_{3z} * m_6 - 2 * l_{2z} * l_{3z} * m_5 +$ $2 * l_{4y} * l_{2z} * m_6 - 2 * l_{3z} * lc_{4y} * m_4 + lc_{3z}^2 * m_3 + lc_{4y}^2 * m_4 +$ $l_{2z}^2 * m_4 + l_{4y}^2 * m_5 + l_{4y}^2 * m_6 + l_{2z}^2 * m_3 + l_{3z}^2 * m_4 +$ $l_{2z}^2 * m_5 + l_{2z}^2 * m_6 + lc_{2z}^2 * m_2 + l_{3z}^2 * m_6 + l_{3z}^2 * m_5 + I_{1y})$	-0,0383
I_{2x}	-0,0683
I_{2y}	0,0303
I_{3x}	0,0567
I_{3y}	-0.0949
I_{4x}	0,0148
I_{4z}	-0.0532
I_{5x}	0.0083
I_{5y}	0.0025
I_{5z}	-0.0084
I_{6x}	0.0030
I_{6y}	0.0076
I_{6z}	0.0021

Como pode ser visto nas imagens e pelos valores de *fitness* a identificação foi considerada suficientemente boa. O modelo dinâmico foi capaz de explicar boa parte do comportamento apresentado e para design de controladores não é necessário um modelo com 100% de precisão. Existiram alguns parâmetros dinâmicos com sinal que, a princípio, não fazem sentido como alguns momentos de inercia negativos ou como o sexto parâmetro do torque da quinta junta, $(m_6 * lc_{6z} * l_{2z} + m_6 * lc_{6z} * l_{4y})$, ser negativo. Uma possível explicação para os momentos de inercia negativos é a simplificação feita das matrizes de inercia serem diagonais. Já o parâmetro da quinta junta pode ter dado negativa por uma equivocada suposição da posição do centro de massa da junta.

O algoritmo de mínimos quadrados nem sempre resolve o sistema para um resultado com a melhor explicação física, mas sim acha um conjunto de parâmetros que torna o erro mínimo. Por isso as vezes alguns termos acabam por não ter um significado físico coerente. Isso se deve principalmente por conta de alguma incerteza de modelagem ou algum problema no conjunto de dados utilizado para aprendizado do modelo.

O algoritmo de identificação não forneceu os valores dos termos desconhecidos separadamente, mas sim combinações lineares destes termos. O conhecimento individual de cada termo não é necessário para a estimação do torque ou a implementação dos controladores tendo em vista que o modelo dinâmico desenvolvido foi reescrito de forma que depende apenas dos valores de cada matriz θ_i. Mesmo assim, caso seja interessante para simulação definir um robô com esses valores separados, as equações na forma matricial também aceitam esta forma de definição.

4.3 CONTROLADORES

Todos os experimentos propostos para os dois controladores a seguir foram feitos com o acréscimo de incerteza de modelagem como explicado na seção 3.2, porém para analisar o seu efeito no resultado foi feito primeiramente um experimento inicial com e sem tal incerteza no controlador de juntas. Nas figuras 4.10 e 4.11 é possível comparar a aceleração angular das juntas para uma mesma trajetória considerando a existência e a ausência da incerteza. O experimento realizado para gerar esses gráficos foi sem o canal integral e sem a presença de perturbações.

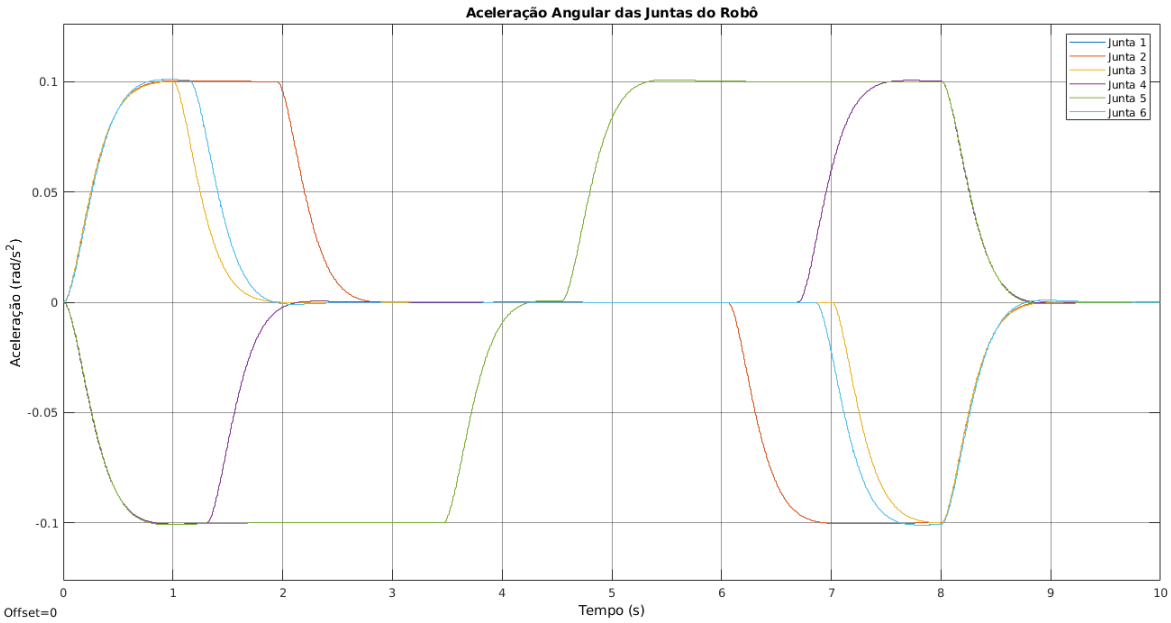

Figura 4.10 – Aceleração angular das juntas do robô no experimento com incerteza de modelagem.

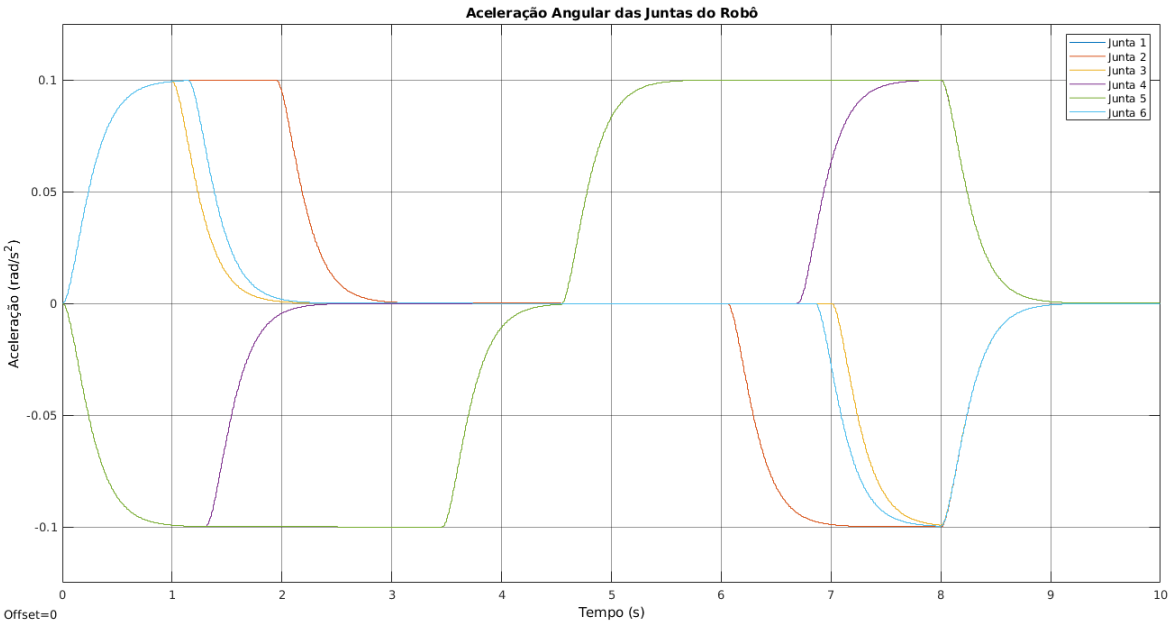

Figura 4.11 – Aceleração angular das juntas do robô no experimento sem incerteza de modelagem.

A diferença entre a presença e ausência do erro na aceleração das juntas apesar de ser perceptível é negligenciável. Já ao analisar os gráficos de referência de aceleração que o controle de posição calcula, figuras 4.12 e 4.13, vemos que o controlador compensa a incerteza de modelagem para tentar garantir que a trajetória desejada seja executada.

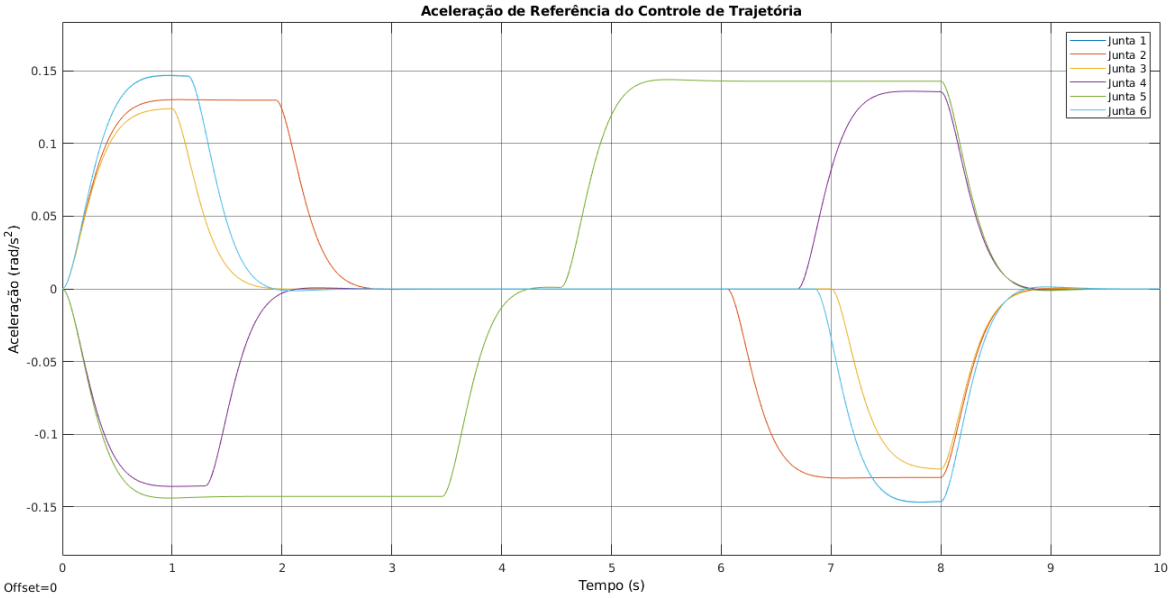

Figura 4.12 – Aceleração angular calculada pelo controle de posição no experimento com incerteza de modelagem.

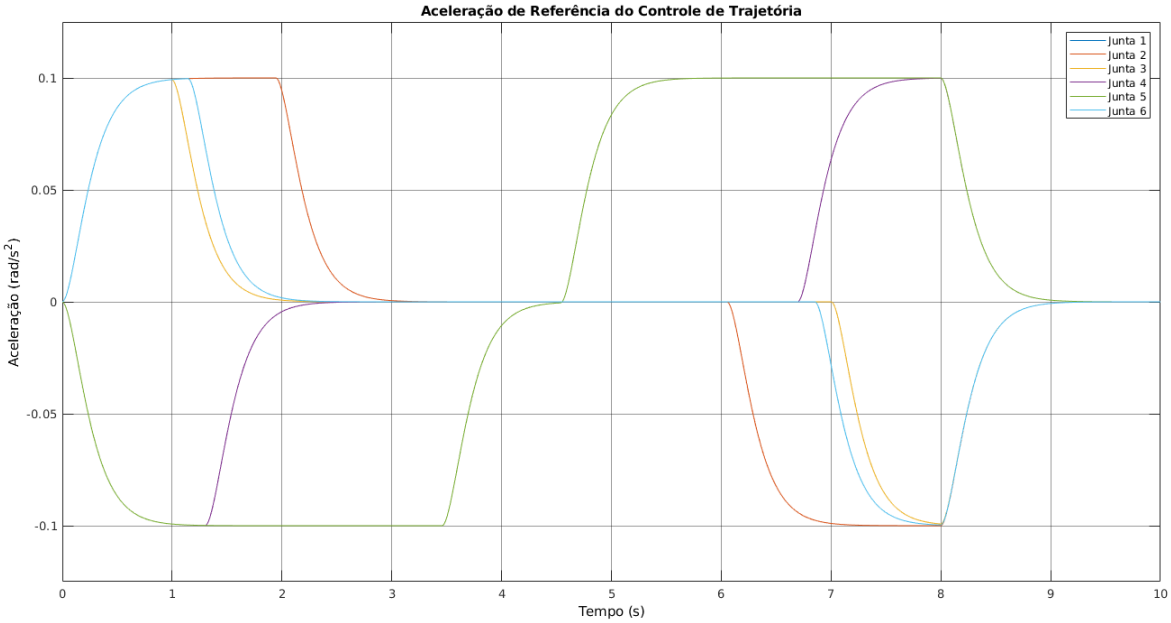

Figura 4.13 – Aceleração angular calculada pelo controle de posição no experimento sem incerteza de modelagem.

4.3.1 Controle de Acompanhamento de Trajetória das Juntas

Foram feitos os experimentos de acordo com a tabela 4.10 para o controlador de acompanhamento de trajetória das juntas desenvolvido utilizando a arquitetura da figura 2.8. Foram desenhados os gráficos com o sinal de erro de posição angular e com o sinal de controle de torque de todos os experimentos para duas trajetórias de referência.

Tabela 4.10 – Características de cada experimento realizado.

	Experimento 1	**Experimento 2**	**Experimento 3**	**Experimento 4**
Perturbação	Não	Sim	Não	Sim
Canal Integral	Não	Não	Sim	Sim

4.3.1.1 Trajetória 1

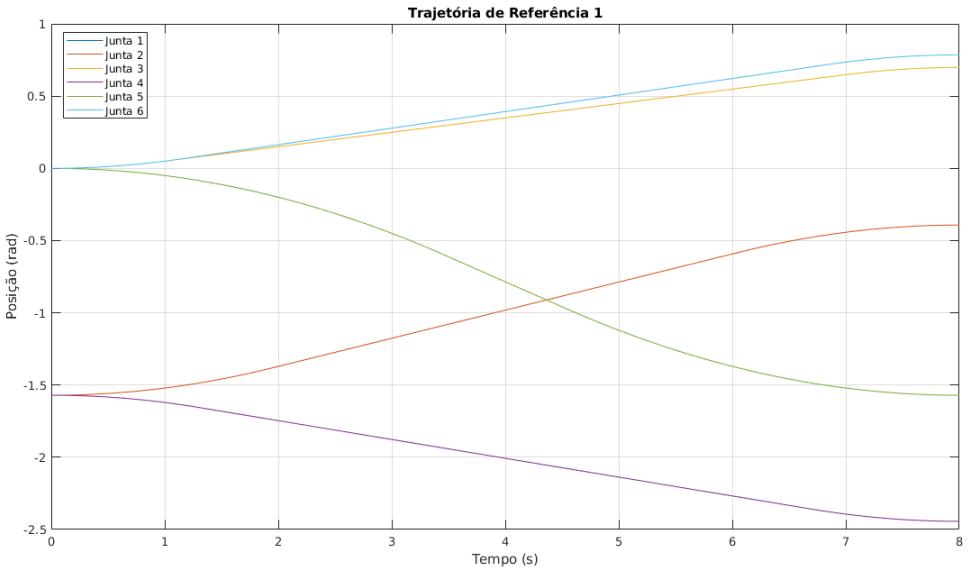

Figura 4.14 – Referências de posição para cada junta propostas para a primeira trajetória de controle das juntas.

A pose inicial e a pose final da trajetória 1, figura 4.14, podem ser vistas na figura 4.15 e suas características são:

- posição inicial em graus = [0 -90 0 -90 0 0],
- posição final em graus = [45 -22,5 40 -140 -90 45],
- tempo de execução = 8 segundos

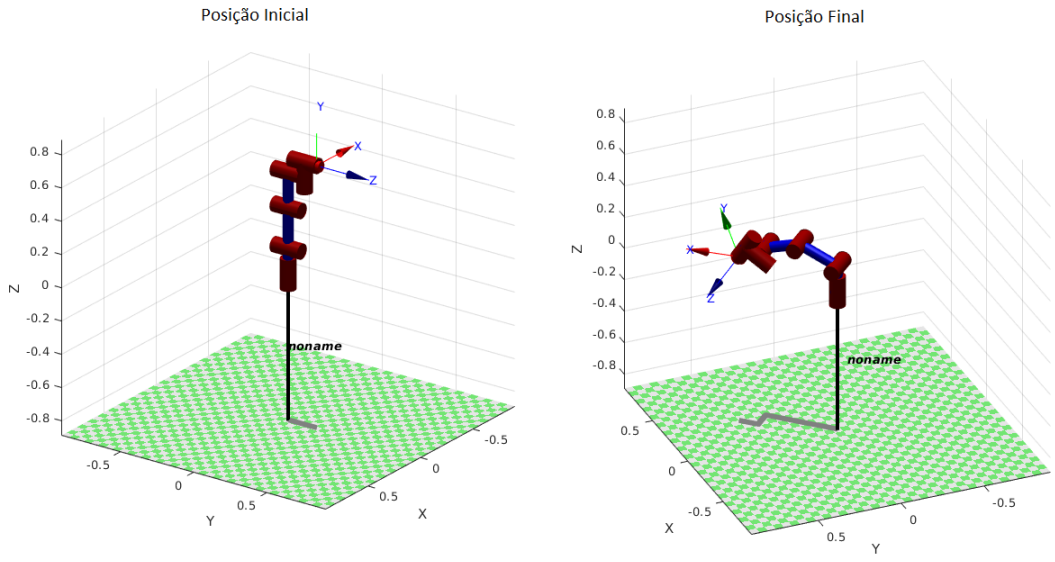

Figura 4.15 – Poses inicial e final da trajetória 1.

Experimento 1:

No primeiro experimento, como não há nem o canal integral nem perturbação, o erro de acompanhamento da posição, figura 4.16, aumenta enquanto a trajetória tem aceleração crescente, é constante enquanto a aceleração é nula e diminui enquanto a aceleração é negativa. Quanto maior os valores de K_p e K_v menor é o valor do erro em cada etapa, porém se forem valores muito altos o controlador se torna muito agressivo enviando comandos elevados de aceleração ao robô. Isso tornaria um problema caso houvesse por exemplo um toque não intencional no robô durante o movimento gerando uma perturbação no torque e uma reação violenta do controlador.

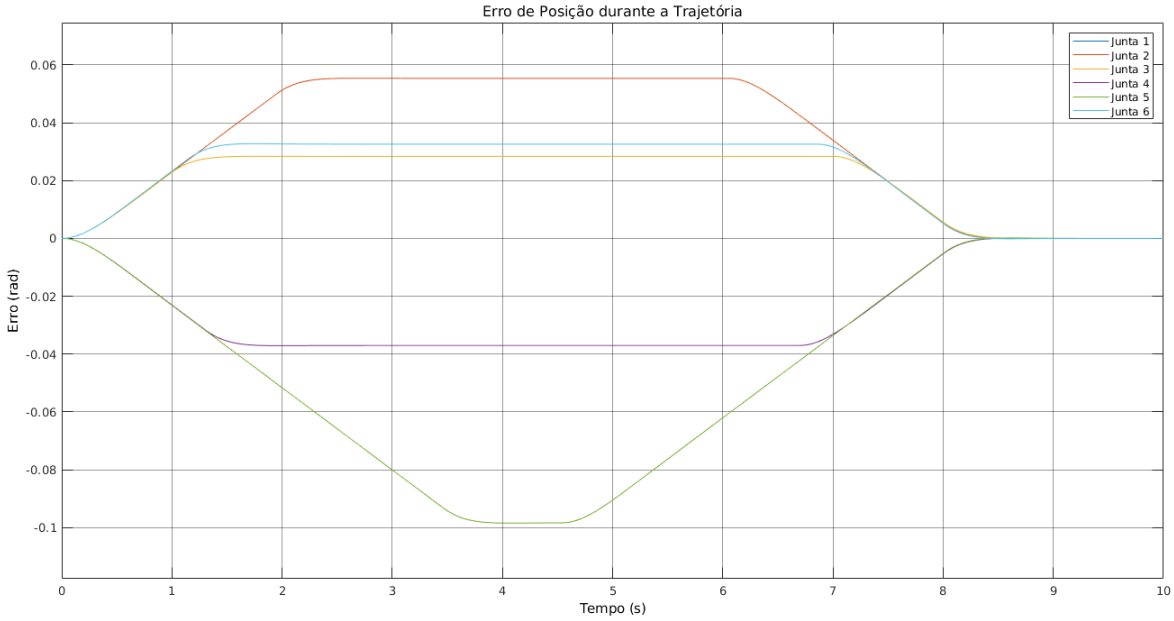

Figura 4.16 – Erro de acompanhamento da posição durante a trajetória 1 sem canal integral e sem perturbação.

Apesar de existir um erro de acompanhamento da posição da trajetória durante sua execução, dependendo da aplicação, tal erro não impactará de forma significativa. Nesta trajetória o erro foi de no máximo 5º em uma das juntas e, considerando que esse erro é durante um movimento do robô, esse erro não foi considerado grave.

Na figura 4.17 podemos ver o sinal de torque enviado para cada junta durante a execução do movimento. Como esperado, tendo em vista a posição inicial e final do manipulador, o torque começa em zero e as juntas 2 a 4 precisam segurar o peso do robô tendo que fazer assim as maiores forças para manter a posição desejada enquanto a junta 1 exerce uma força durante o movimento e ao final aplica torque nulo.

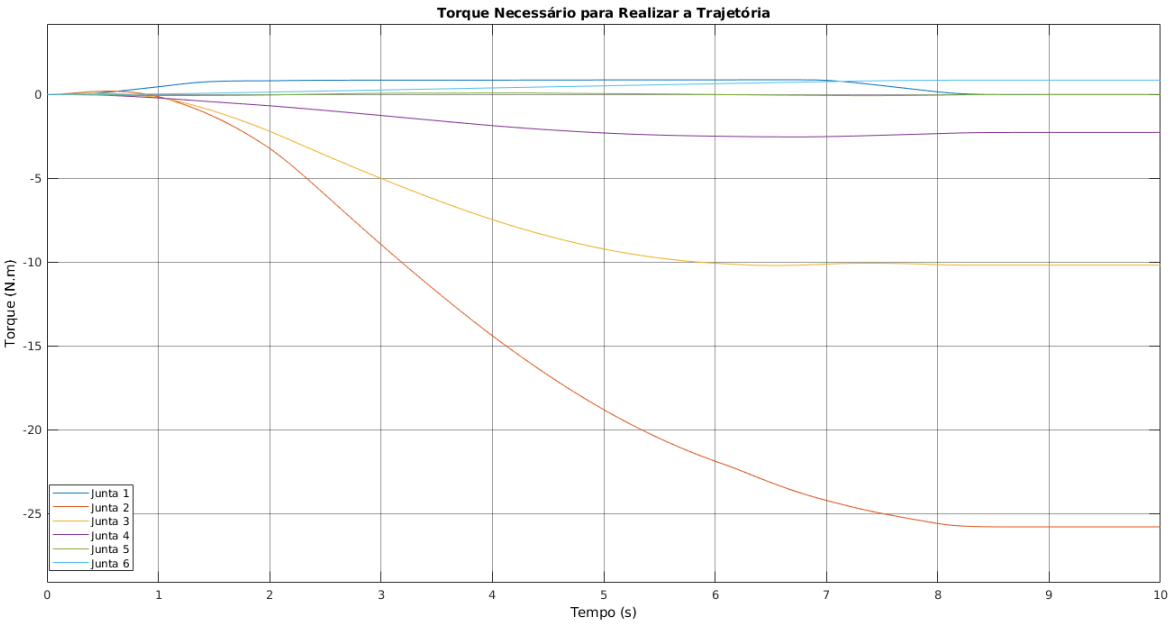

Figura 4.17 – Sinal de controle enviado ao simulador do robô durante a trajetória 1 para controle das juntas sem canal integral e sem perturbação.

Experimento 2:

Neste experimento foi mantido o controlador do experimento 1, porém foi incrementado uma perturbação no torque no local indicado na figura 3.3. Essa perturbação serviu para simular um toque não intencional no robô e tem a magnitude igual a figura 4.18. O formato de onda quadrada com início e fim em zero foi proposto para ser possível analisar como o sistema de controle iria lidar com a perturbação no momento do suposto toque e como seria a volta após essa perturbação cessar.

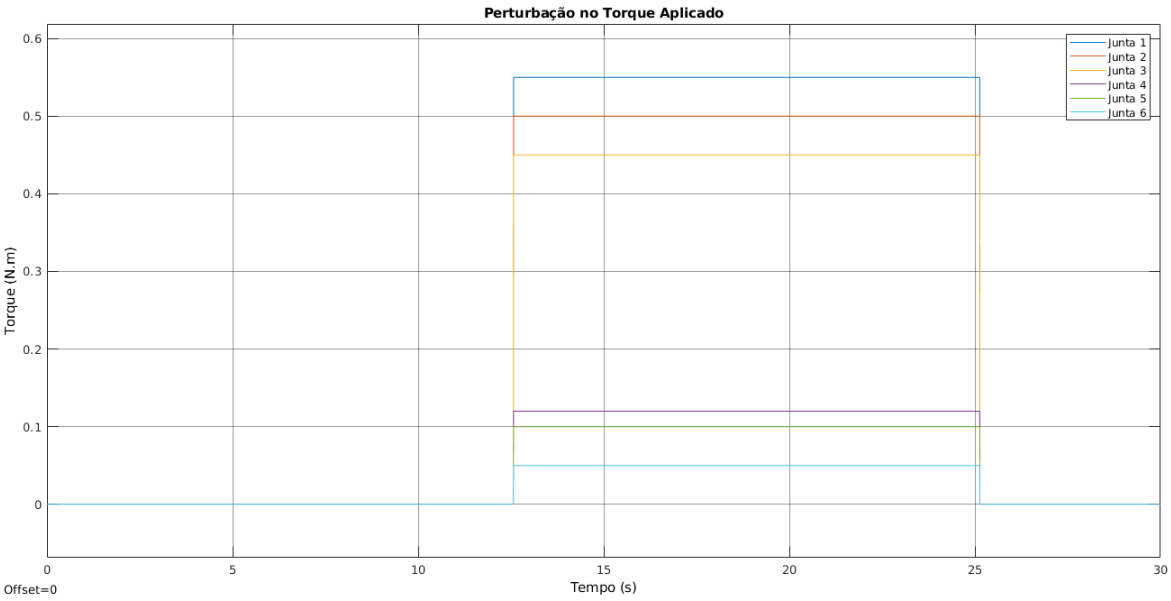

Figura 4.18 – Perturbação no torque durante a trajetória 1.

Da figura 4.19 vemos que este controlador apenas com os ganhos K_p e K_v não é capaz de manter o erro nulo. Também é possível ver que o erro tem um *overshoot* no momento da aplicação do toque, mas não apresenta tal resposta na retirada.

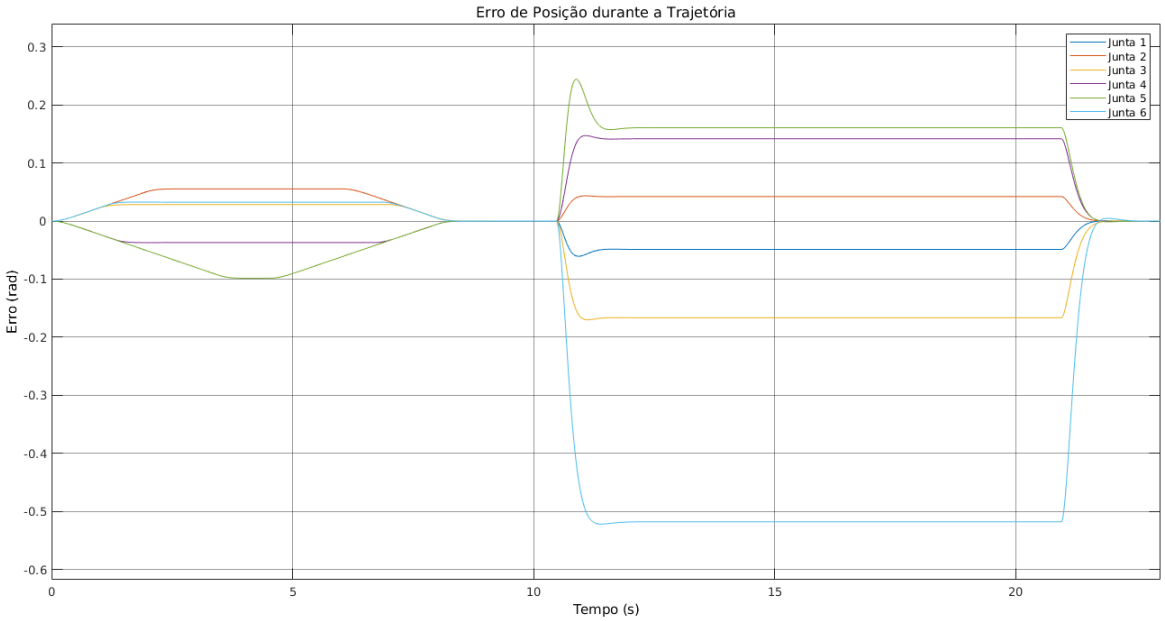

Figura 4.19 – Erro de acompanhamento da posição durante a trajetória 1 sem canal integral e com perturbação.

Já na figura 4.20 vemos o sinal de torque calculado para realizar a trajetória. Esse formato de resposta à perturbação com uma mudança no torque enviado e depois tendendo a um valor constante próximo ao valor anterior pode ser explicado pelo fato da referência, neste instante da trajetória, já estar mandando o controlador manter as juntas paradas, ou seja, ele tenta manter a velocidade nula e essa componente não permite que o erro de posição diminua apenas com o termo proporcional ao erro. Então o que esse controlador faz na presença de perturbação é apenas equilibrar essa força para manter as juntas paradas.

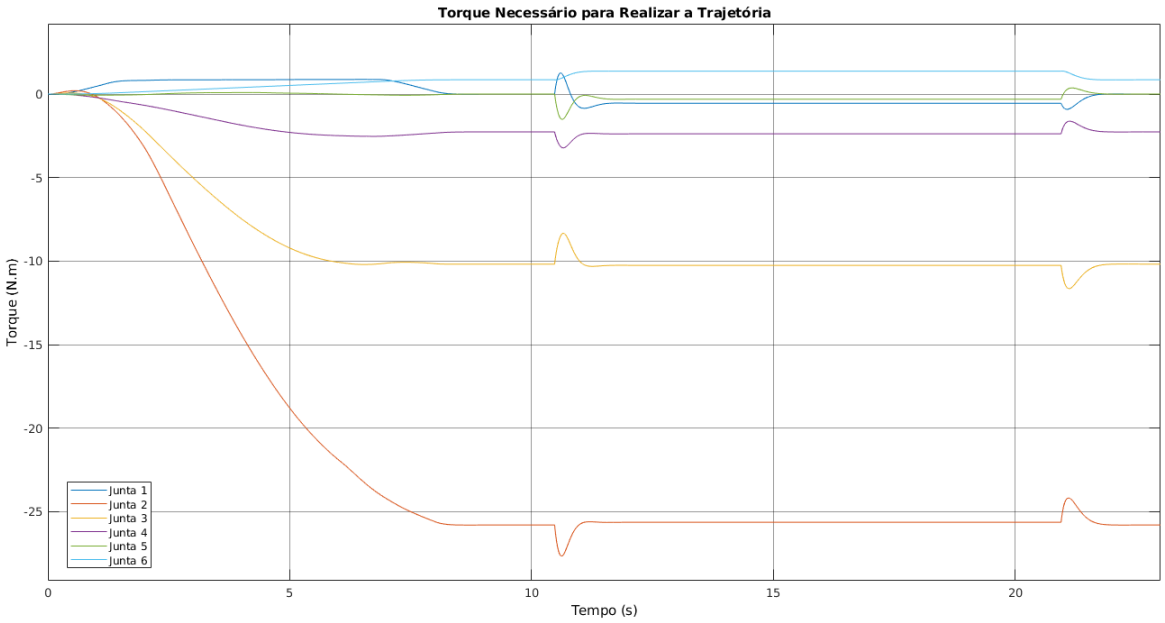

Figura 4.20 – Sinal de controle enviado ao simulador do robô durante a trajetória 1 para controle das juntas sem canal integral e com perturbação.

Experimento 3:

O terceiro experimento foi proposto para analisar como o canal integral modifica a resposta de operação normal da trajetória, isto é, como ela modifica o movimento na ausência de perturbações.

Com a adição do sinal integral ao erro de posição o controlador passa a tentar levar o erro que era não nulo durante a trajetória para zero. Na imagem 4.21 vemos o novo formato do erro de posição. Ao compará-lo com o erro da figura 4.16 é possível ver que o erro durante a trajetória inteira é significativamente menor. Mas em contrapartida a trajetória leva mais tempo para chegar ao ponto final. Neste exemplo deveria ser aos 8 segundos, mas só há convergência por volta dos 11,5 segundos. Já a figura 4.22 mostra o sinal de torque aplicado nas juntas nesse experimento e a única diferença visível ao comparar com a figura 4.17 do experimento sem canal integral é que o torque em regime permanente demora alguns segundos a mais para estabilizar.

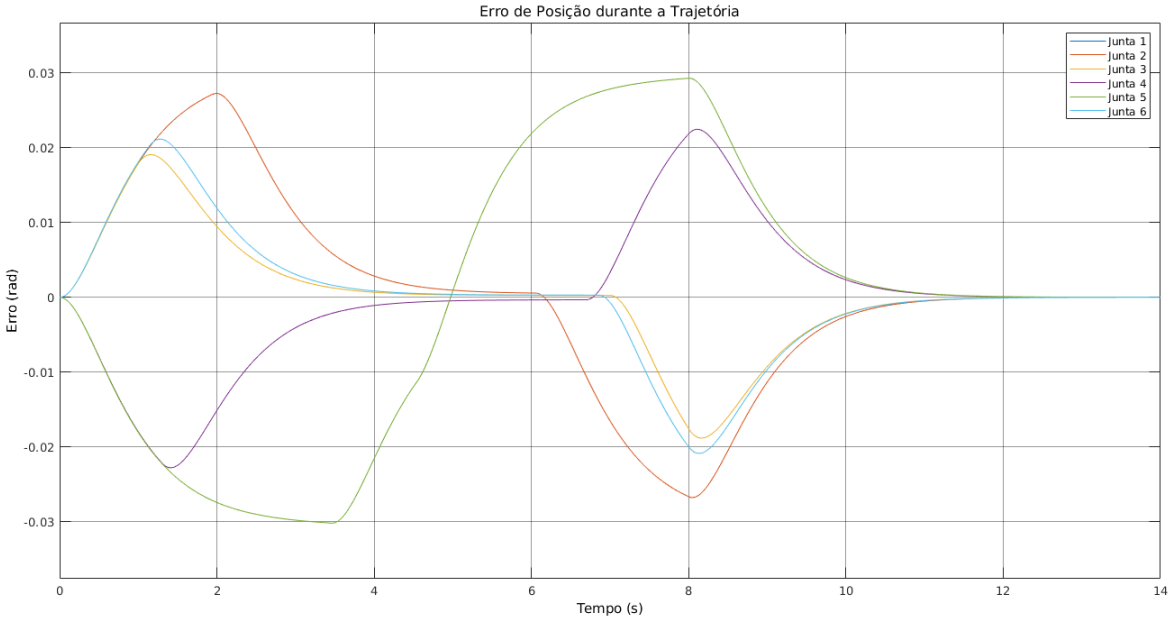

Figura 4.21 – Erro de acompanhamento da posição durante a trajetória 1 com canal integral e sem perturbação.

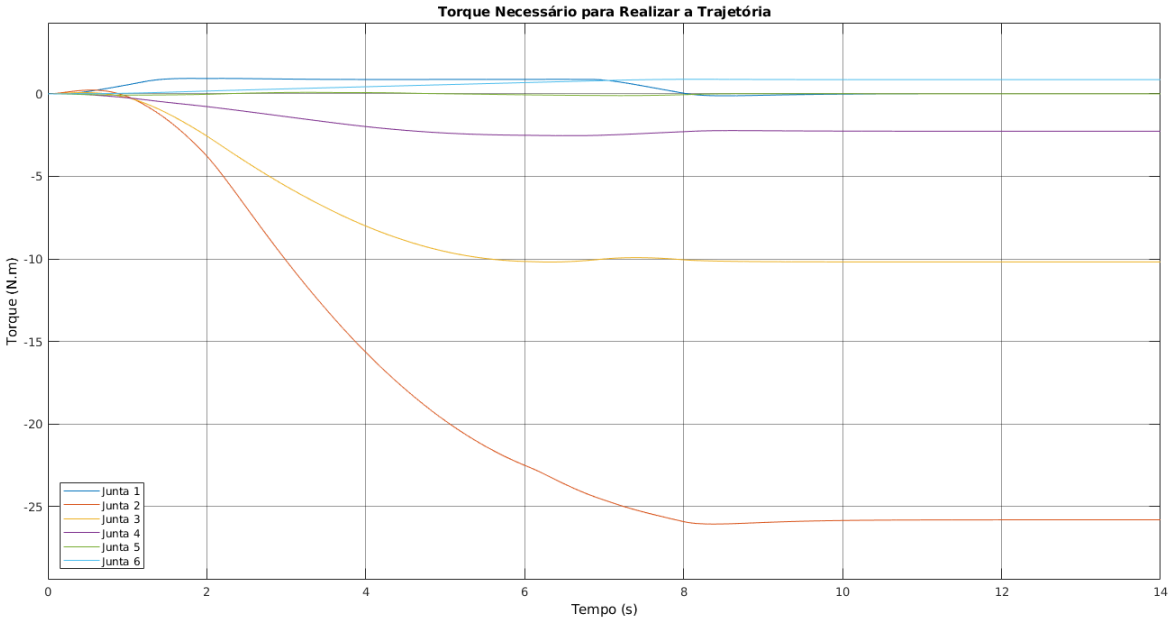

Figura 4.22 – Sinal de controle enviado ao simulador do robô durante a trajetória 1 para controle das juntas com canal integral e sem perturbação.

Experimento 4:

Por fim, o experimento 4 utiliza o mesmo controlador do experimento 3, mas é acrescentado uma perturbação de torque assim como no experimento 2. A figura 4.23 mostra claramente que esse controlador é capaz de reduzir erro de posição à zero e rejeita perturbações. Experimentos foram feitos aumentando o valor de ganho K_i e apesar de diminuir o tempo de convergência a zero tínhamos o mesmo problema de se escolher valores maiores

para K_p e K_v obtendo respostas violentas à perturbação. O sinal de torque enviado para as juntas neste experimento pode ser visto na figura 4.24 e, diferentemente do sinal do experimento 2, figura 4.20, é possível ver que o sinal de torque em regime permanente durante a perturbação tem um valor diferente do que aquele no regime permanente antes e depois da perturbação. Isso mostra que o canal integral de fato compensa a perturbação calculando um novo valor de torque.

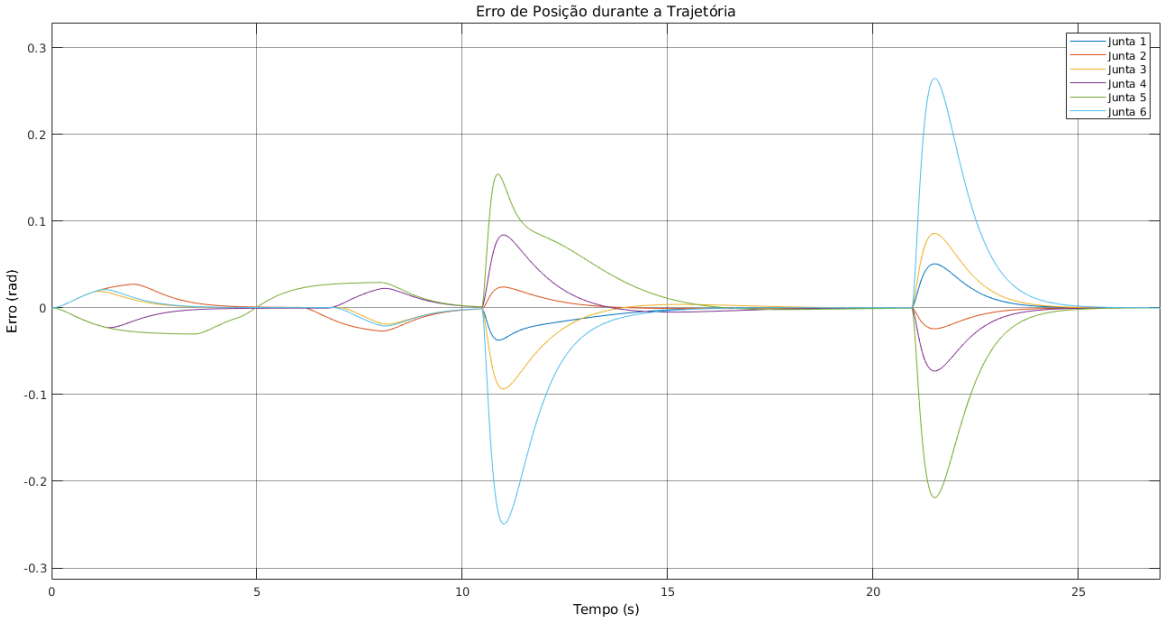

Figura 4.23 – Erro de acompanhamento da posição durante a trajetória 1 com canal integral e com perturbação.

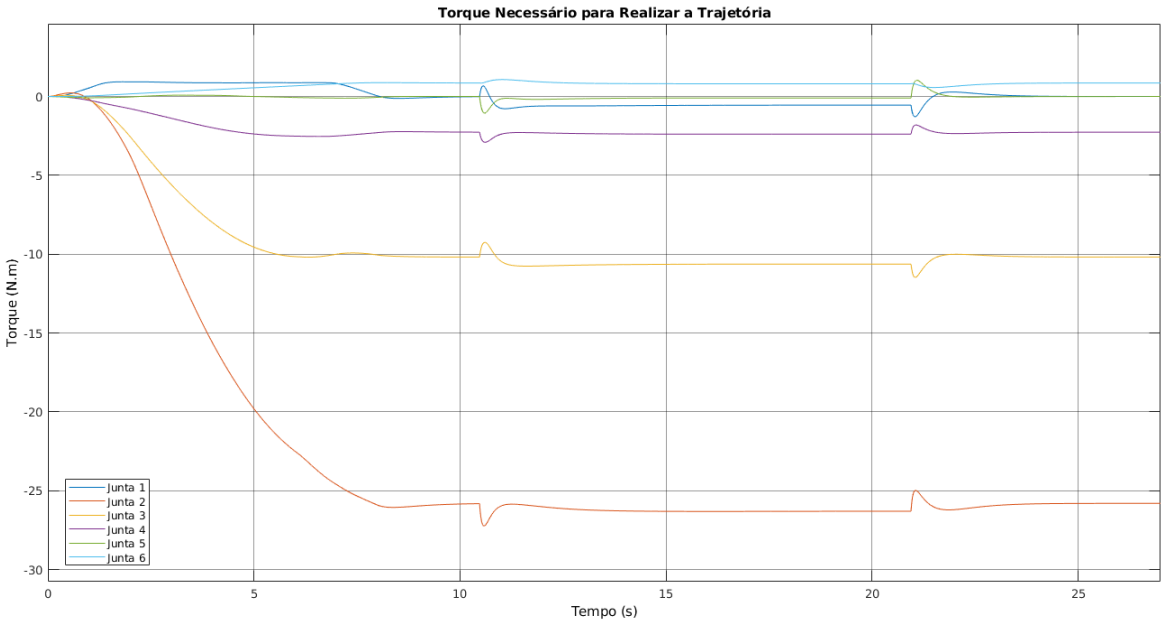

Figura 4.24 – Sinal de controle enviado ao simulador do robô durante a trajetória 1 para controle das juntas com canal integral e com perturbação.

4.3.1.2 Trajetória 2

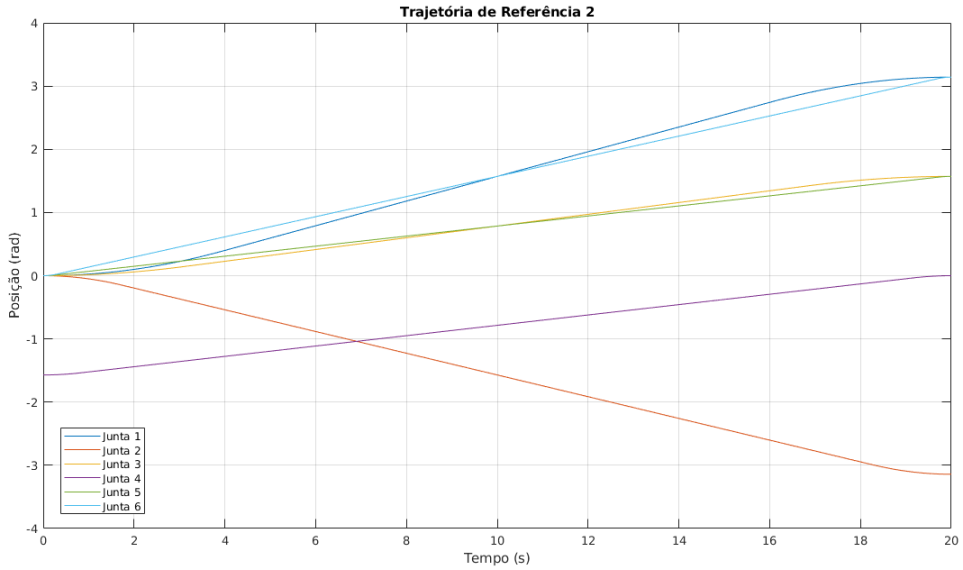

Figura 4.25 – Referências de posição para cada junta propostas para a segunda trajetória de controle das juntas.

A pose inicial e a pose final da trajetória 2, figura 4.25, podem ser vistas na figura 4.26. As características dessa trajetória são:

- posição inicial em graus = [0 0 0 -90 0 0],
- posição final em graus = [180 -180 90 0 90 180],
- tempo de execução = 20 segundos

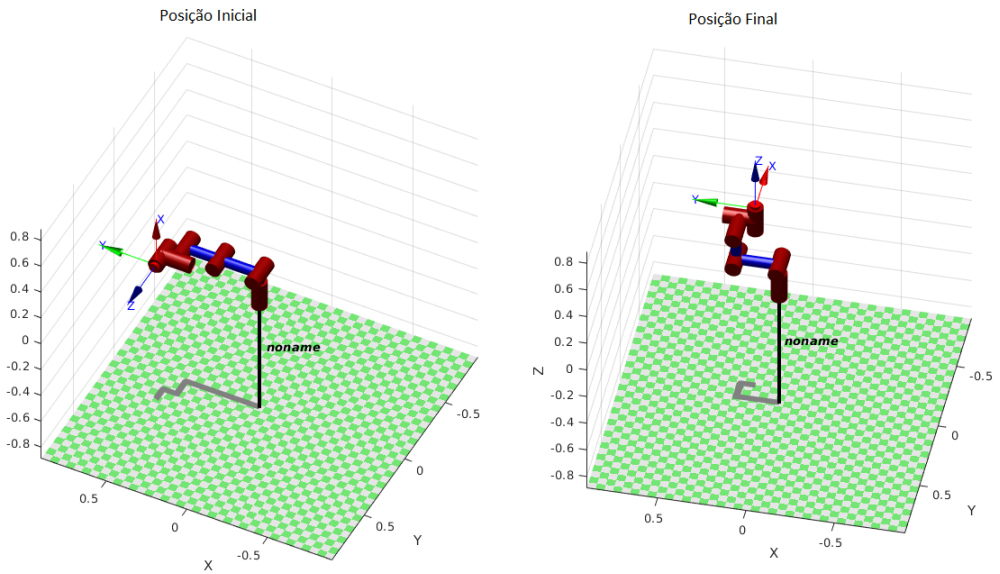

Figura 4.26 – Poses inicial e final da trajetória 2.

Experimento 1:

Os resultados do experimento 1 para a trajetória 2 foram muito parecidos com o da trajetória 1, em que existe um erro de acompanhamento da posição que cresce quando a velocidade é positiva e decresce quando é negativa. Na figura 4.27 podemos ver que nesta trajetória por ser mais longa passa um tempo maior na parte de erro constante.

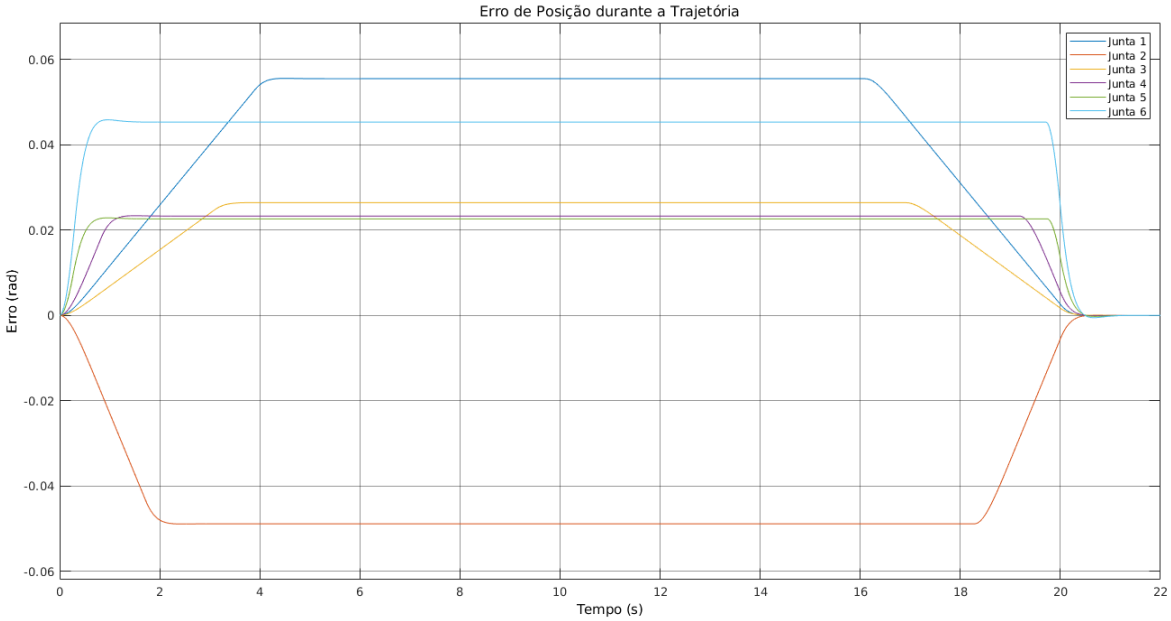

Figura 4.27 – Erro de posição durante a trajetória 2 sem canal integral e sem perturbação.

Experimento 2:

Com o mesmo controlador foi inserido no experimento 2 uma perturbação no torque, mas diferentemente da perturbação da trajetória 1 aqui ela foi inserida antes da trajetória finalizar. A figura 4.28 mostra que a perturbação começa aproximadamente aos 11 segundos e dura até os 22, logo ela começa antes da trajetória finalizar e termina após.

Na figura 4.29 é possível ver o erro de posição para esse experimento 2. Nele vemos que o controlador funciona muito mal durante a perturbação aumentando o erro até que o sinal de perturbação acabe. Outro ponto observado é que pela imagem do sinal de torque necessário, figura 4.30, o controlador está um pouco violento mandando um sinal de quase 80 N.m para uma junta e quase -100 N.m para outra.

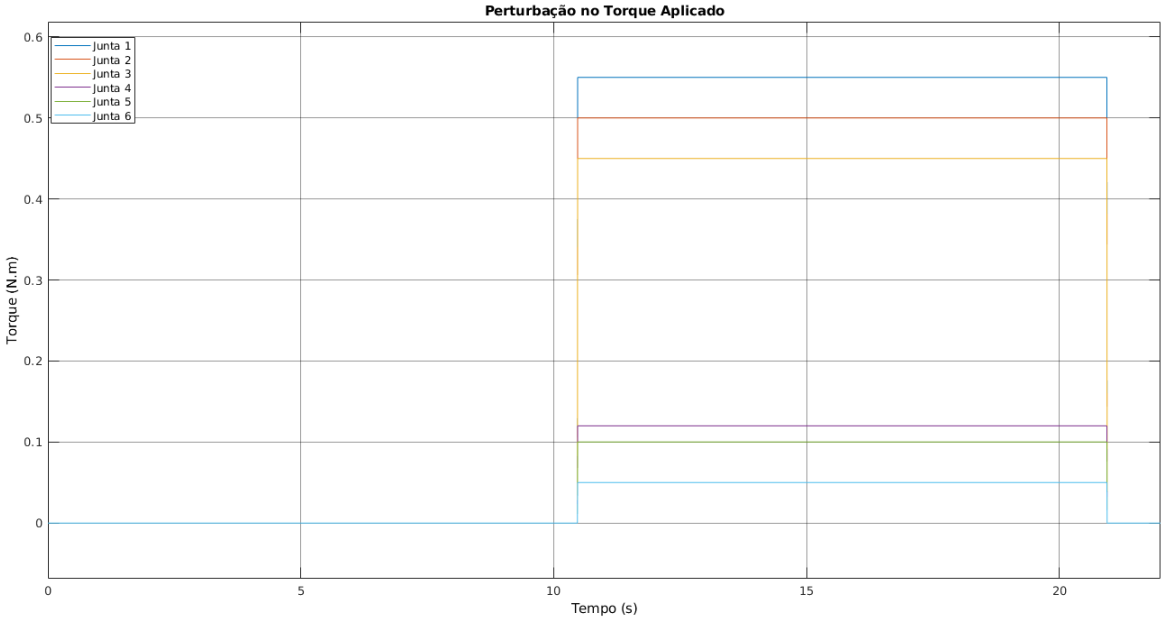

Figura 4.28 – Perturbação no torque durante a trajetória 2.

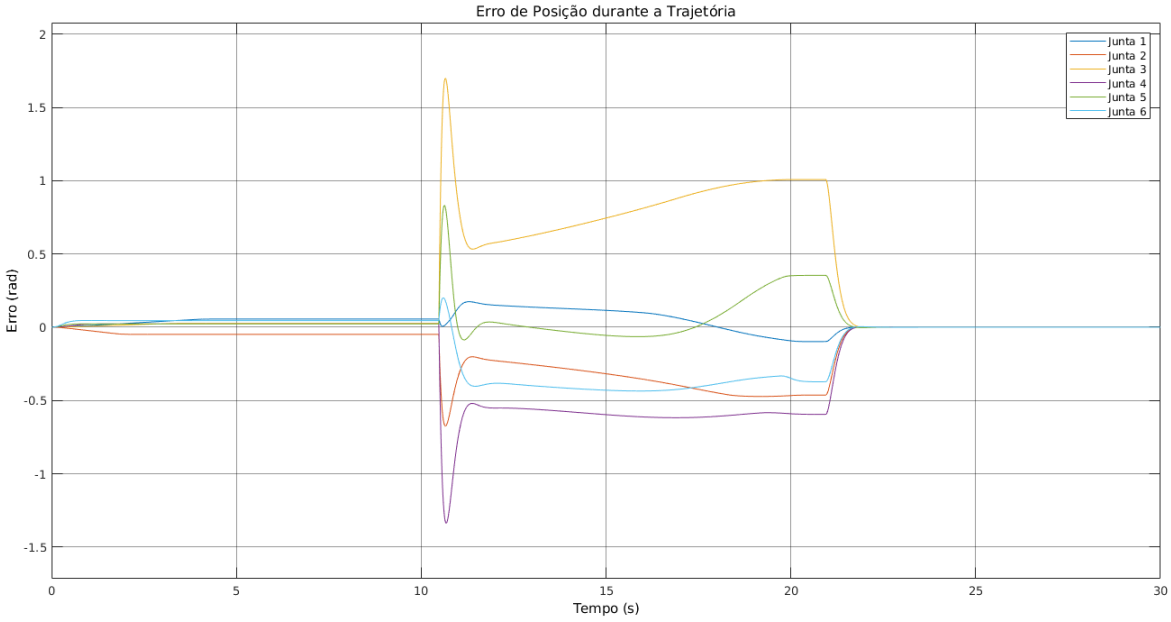

Figura 4.29 – Erro de posição durante a trajetória 2 sem canal integral e com perturbação.

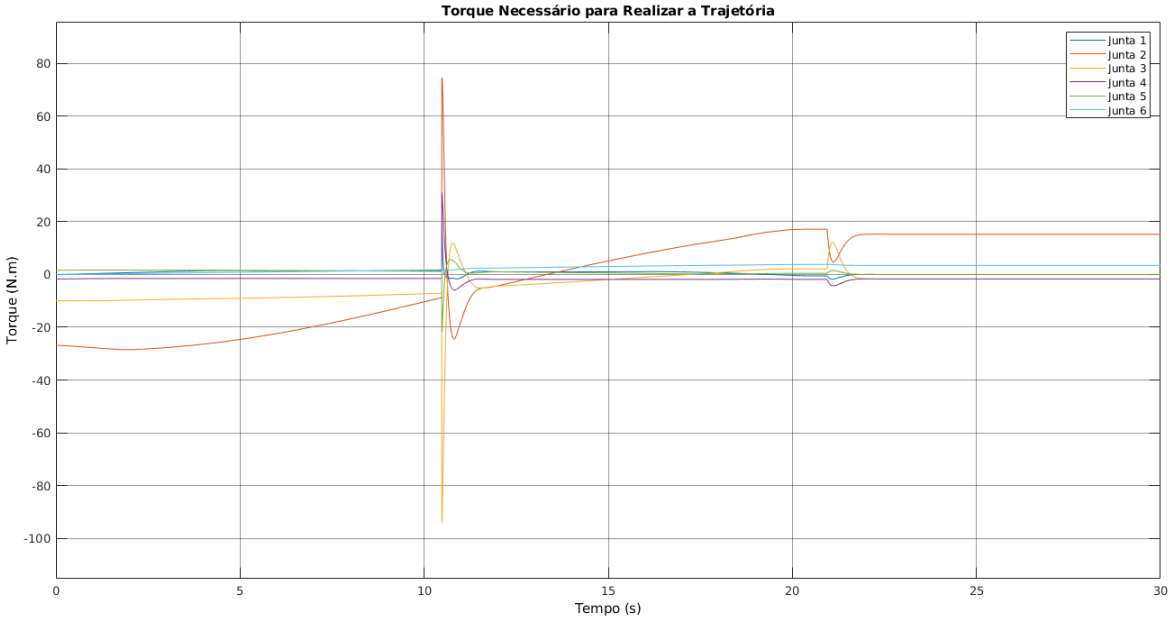

Figura 4.30 – Sinal de controle enviado ao simulador do robô durante a trajetória 2 para controle das juntas sem canal integral e com perturbação.

Experimento 3:

Este experimento reforçou o que se constatou no 3º experimento da trajetória 1, reduzindo o erro que era constante durante o trecho de aceleração zero e causando um *overshoot* nos momentos de desaceleração.

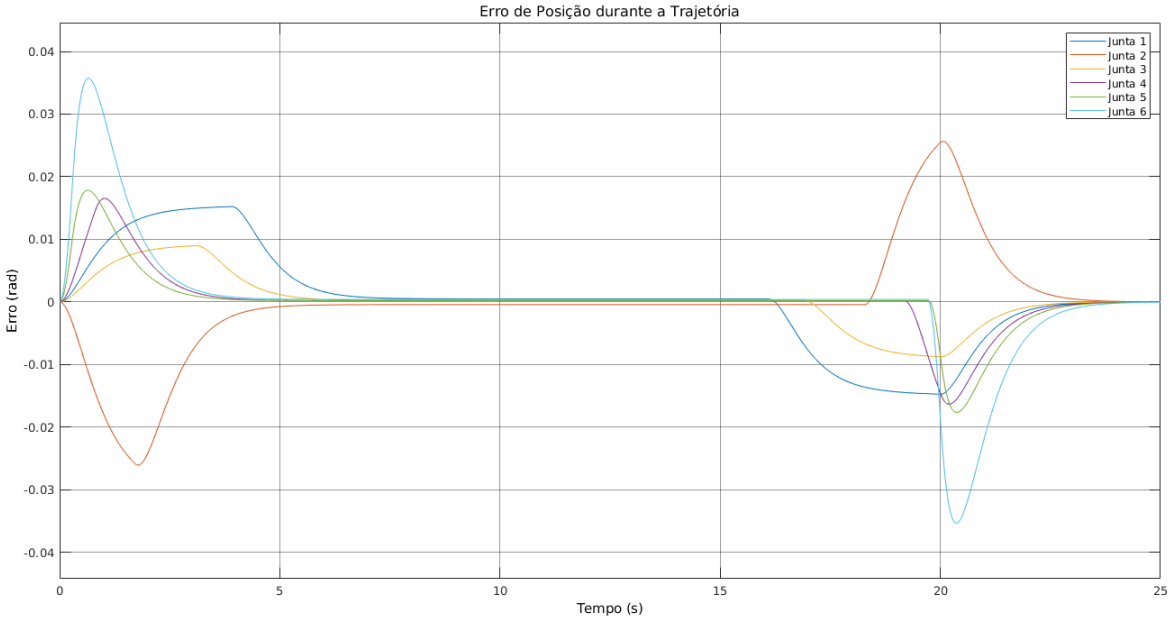

Figura 4.31 – Erro de posição durante a trajetória 2 com canal integral e sem perturbação.

Experimento 4:

O resultado do 4º e último experimento para a trajetória 2 do controle de juntas pode ser

visto nas figuras 4.32 e 4.33. Nessas figuras é possível ver que o controlador está o tempo todo tentando diminuir o erro de posição e a resposta à perturbação não foi violenta como no experimento 1.

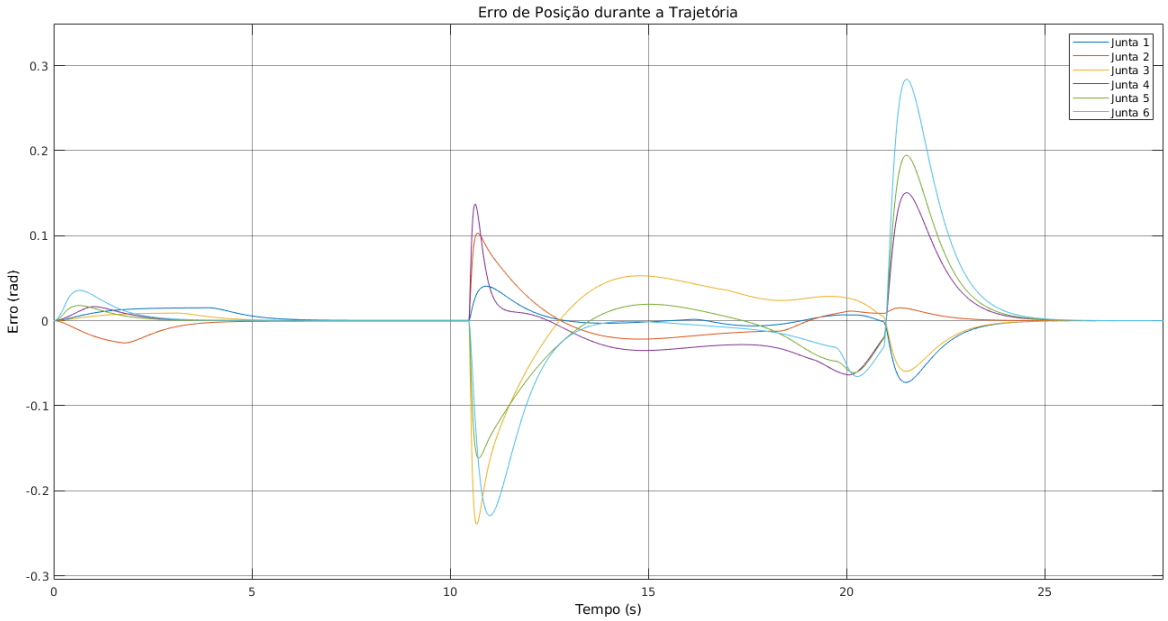

Figura 4.32 – Erro de posição durante a trajetória 2 com canal integral e com perturbação.

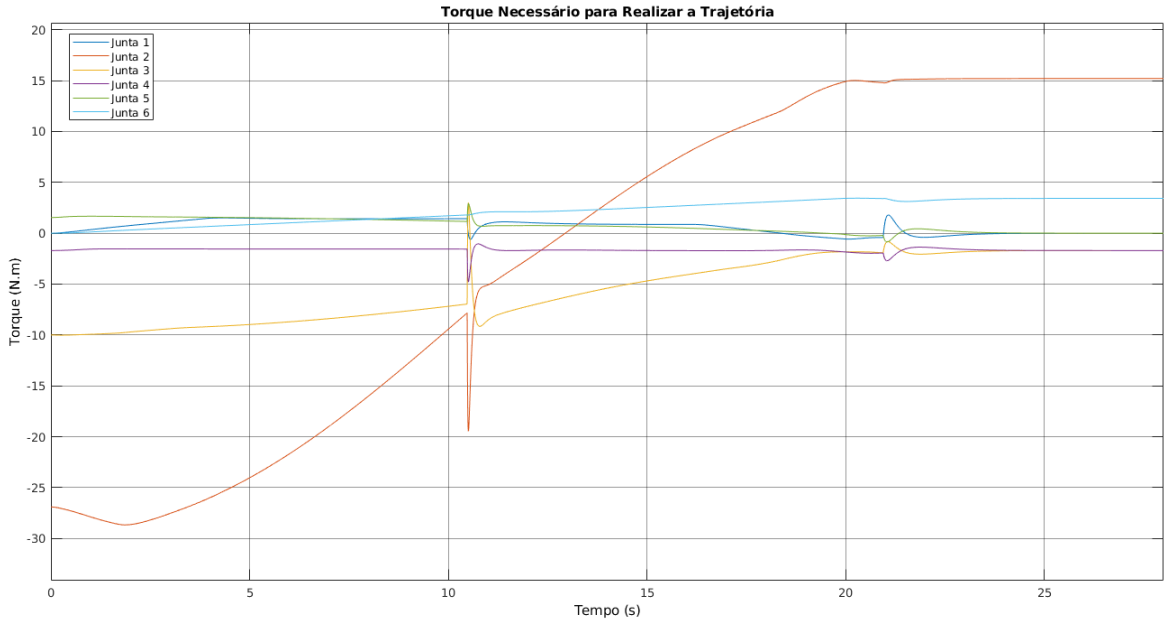

Figura 4.33 – Sinal de controle enviado ao simulador do robô durante a trajetória 2 para controle das juntas com canal integral e com perturbação.

4.3.2 Controle de Acompanhamento de Trajetória da Ferramenta

Os experimentos do controle de acompanhamento de trajetória da ferramenta, arquitetura mostrada na figura 2.9, foram feitos assim como os do controle anterior, porém, devido aos problemas com a representação da orientação relatados na seção 3.4.2, não foram considerados os experimentos com perturbação de torque e os ganhos K_p e K_i do controlador para as variáveis de orientação foram colocados em zero. Desta forma foram propostas duas trajetórias com dois experimentos um com e um sem canal integral para este controlador.

4.3.2.1 Trajetória 1

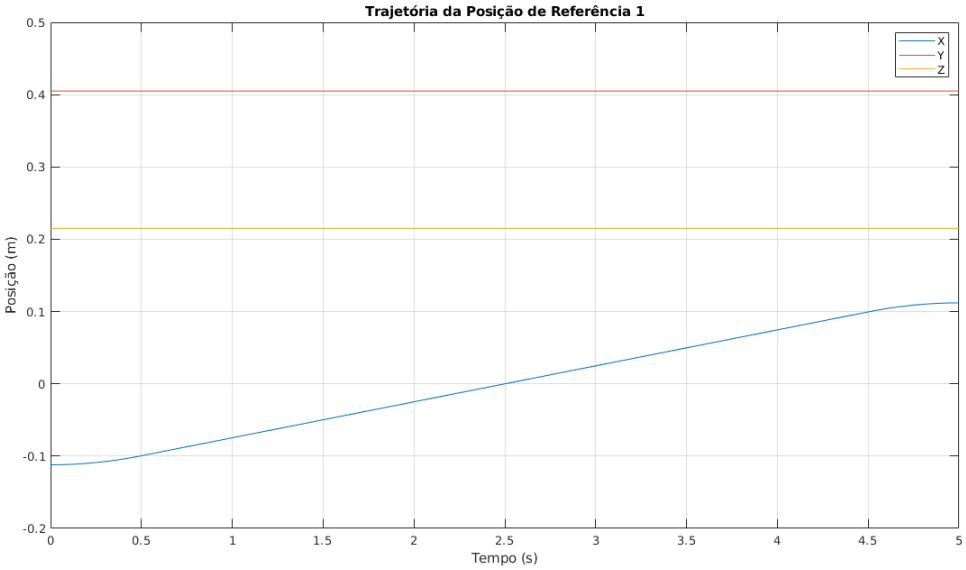

Figura 4.34 – Referências de posição cartesiana propostas para a primeira trajetória de controle da ferramenta.

A pose inicial e a pose final da trajetória 1, figura 4.34, podem ser vistas na figura 4.35. Essa trajetória foi escolhida para verificar se o robô conseguiria realizar um movimento em linha reta em uma dimensão sem se mover nas outras duas e suas características são:

- posição inicial em metros = [-0,112 0,405 0,215],
- posição final em metros = [0,112 0,405 0,215],
- orientação fixa em graus = [-pi/2 0 pi]
- tempo de execução = 5 segundos

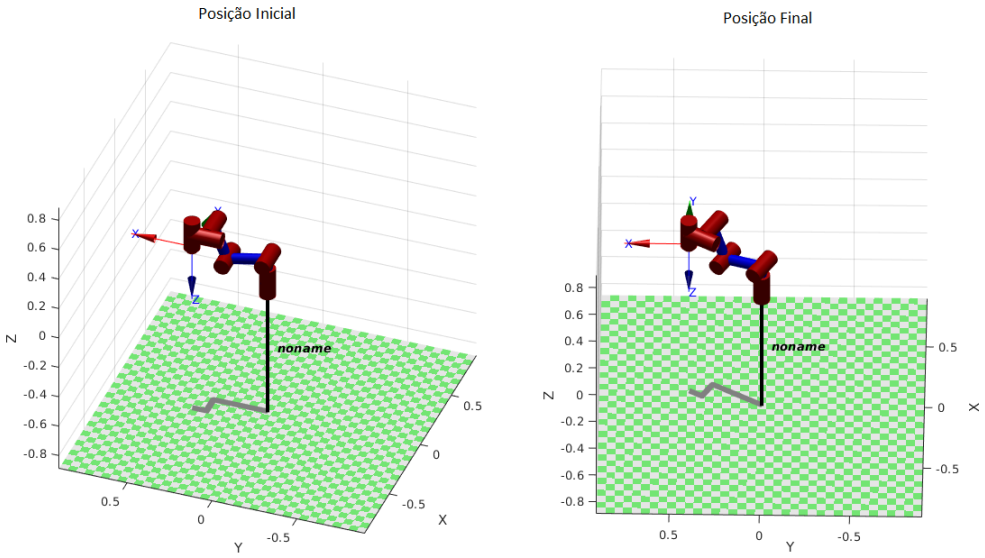

Figura 4.35 – Poses inicial e final da trajetória 1.

Experimento 1:

O erro de acompanhamento de posição para essa trajetória pode ser visto na figura 4.36. Já na figura 4.37 é possível ver o sinal de controle enviado ao simulador do robô.

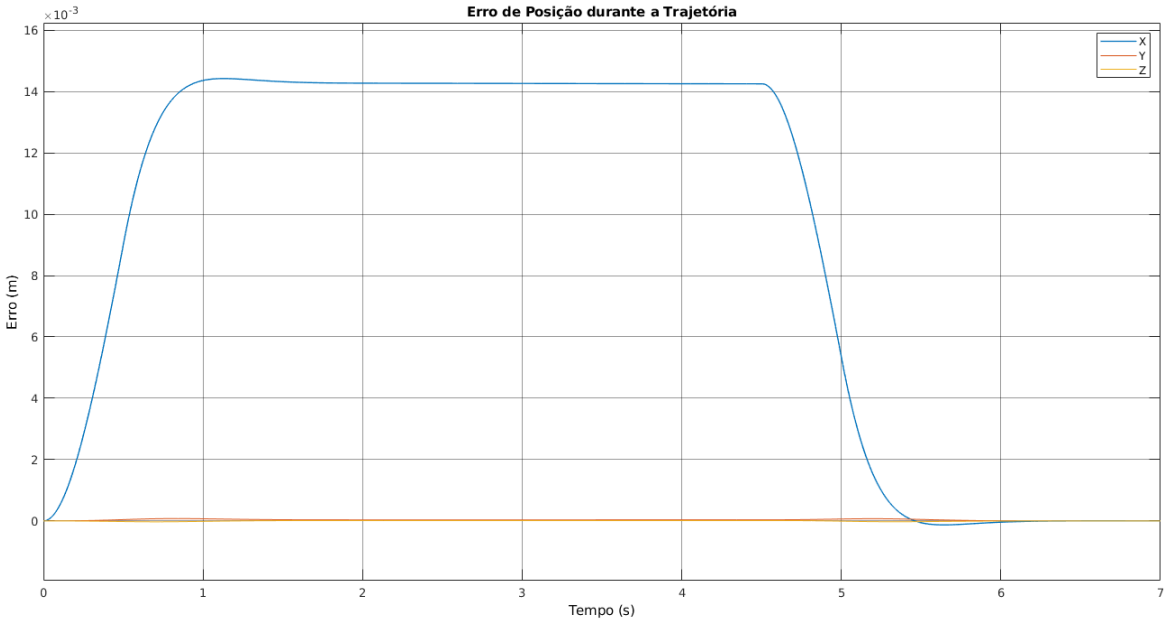

Figura 4.36 – Erro de posição durante a trajetória 1 sem canal integral.

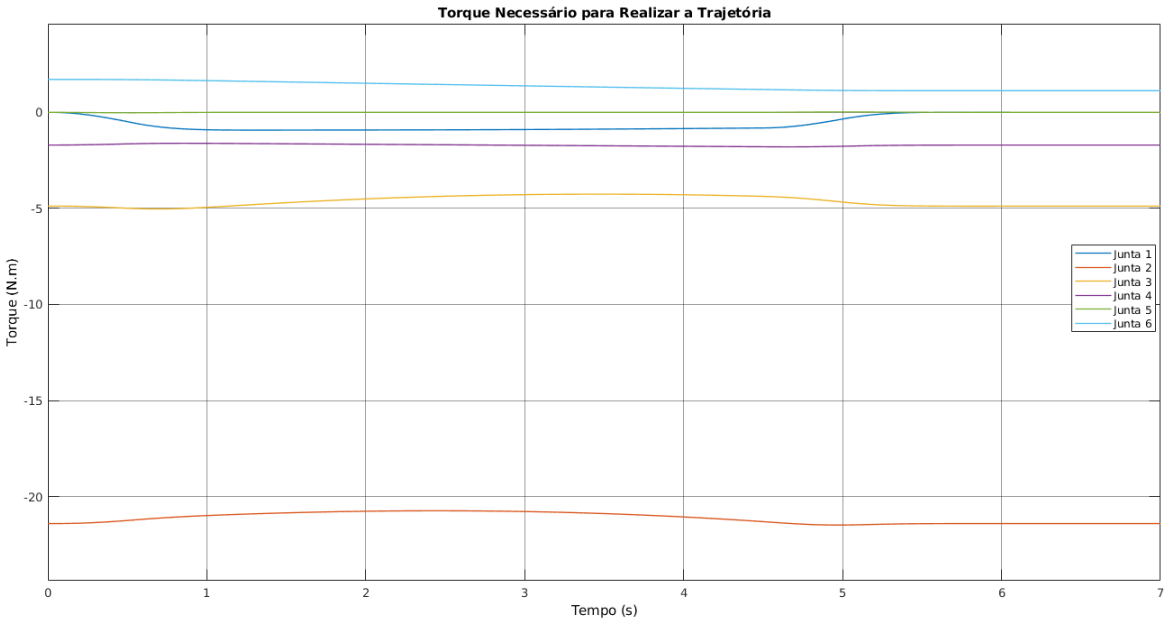

Figura 4.37 – Sinal de controle enviado ao simulador do robô durante a trajetória 1 para controle da ferramenta sem canal integral.

Do gráfico de erro de acompanhamento de posição é possível ver que o controlador de acompanhamento de trajetória cartesiana funciona tão bem quanto o de trajetória das juntas no sentido de que ele consegue levar a ferramenta a fazer o movimento pedido.

Experimento 2:

O erro de acompanhamento de posição para esse experimento pode ser visto na figura 4.38. Assim como o controlador das juntas o erro de posição da ferramenta quando há o canal integral é reduzido durante a realização da trajetória quando comparado com o controlador sem esse ganho.

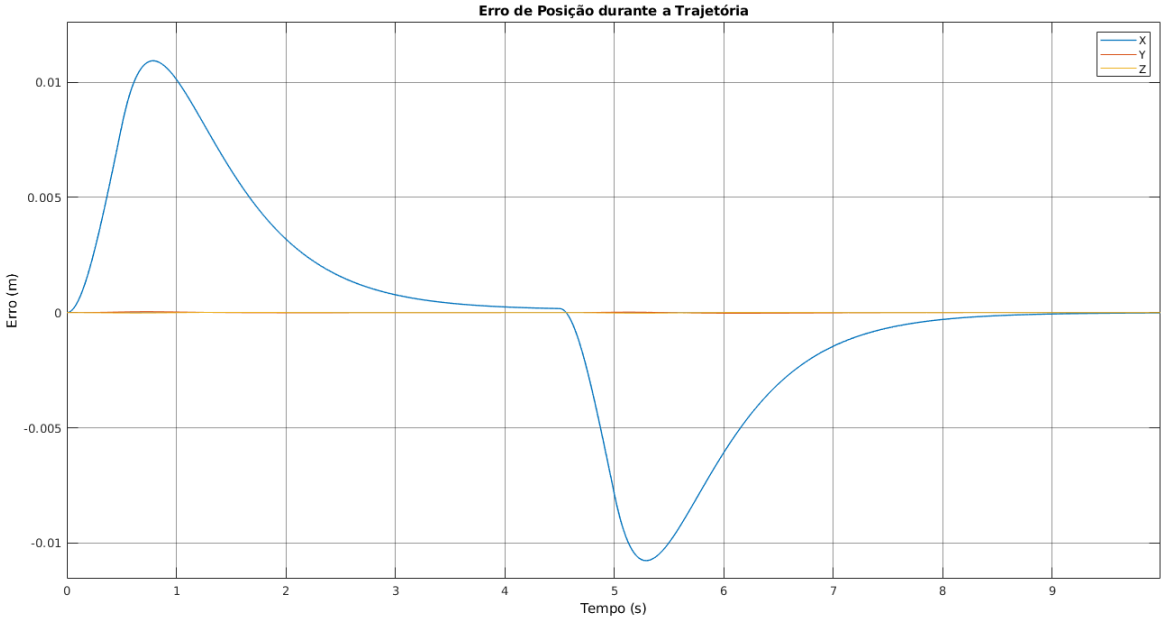

Figura 4.38 – Erro de posição durante a trajetória 1 com canal integral.

4.3.2.2 Trajetória 2

A segunda trajetória proposta pode ser vista na figura 4.39 e tem como objetivo avaliar se seria possível fazer com que a ferramenta siga uma referência nas três dimensões do espaço ao mesmo tempo.

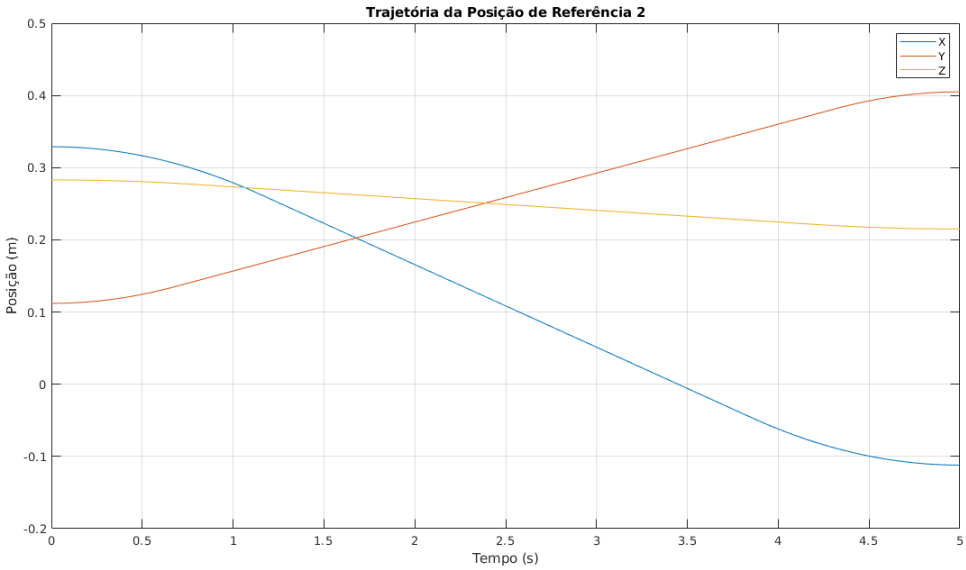

Figura 4.39 – Referências de posição cartesiana propostas para a segunda trajetória de controle da ferramenta.

A pose inicial e final do robô para esse movimento podem ser vistas na figura 4.40 e as características do movimento são:

- posição inicial em metros = [0,329 0,112 0,283],
- posição final em metros = [-0,112 0,405 0,215],
- orientação fixa em graus = [-pi/2 0 pi]
- tempo de execução = 5 segundos

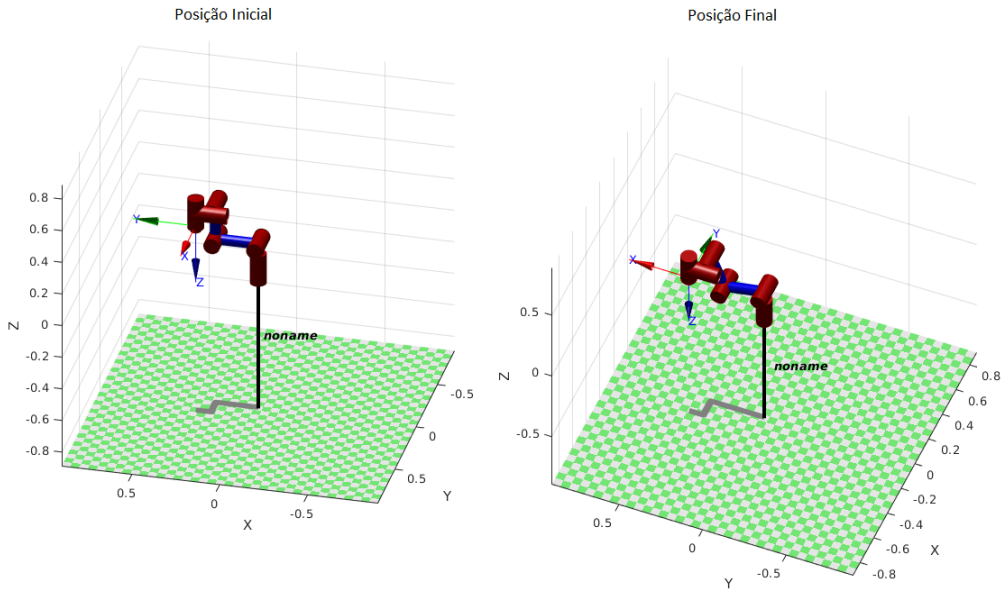

Figura 4.40 – Poses inicial e final da trajetória 2.

Experimento 1:

O padrão de erro deste segundo controlador é muito parecido com o do primeiro. Na figura 4.41 é possível ver que o erro de acompanhamento de posição no espaço tem o formato parecido com o erro de posição angular do controlador de juntas. Já o sinal de controle para realizar o movimento é visto na figura 4.42 e dele é possível ver que o controlador não calculou torques com valores elevados para realizar o acompanhamento da trajetória.

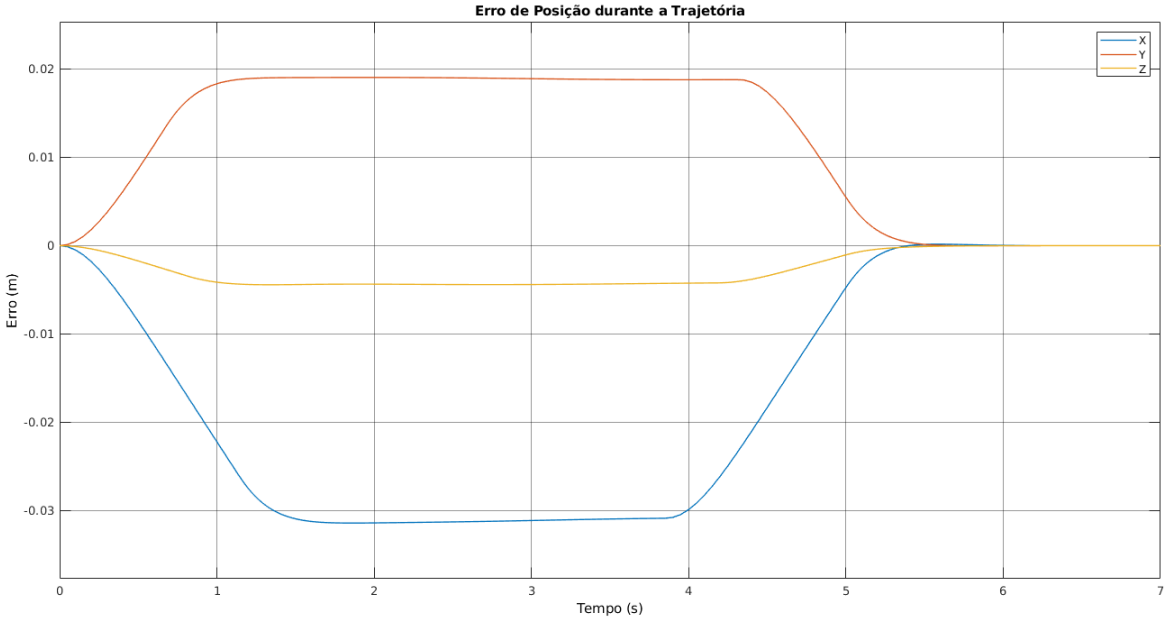

Figura 4.41 – Erro de posição durante a trajetória 2 sem canal integral.

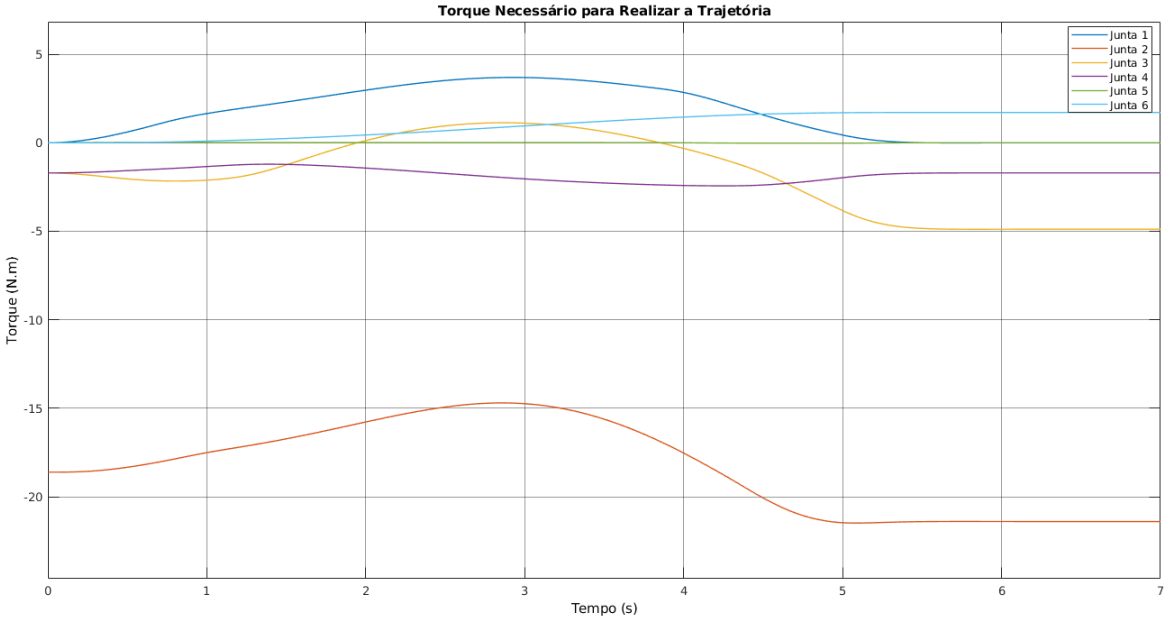

Figura 4.42 – Sinal de controle enviado ao simulador do robô durante a trajetória 2 para controle da ferramenta sem canal integral.

Experimento 2:

Por fim, para finalizar os experimentos feitos nesse trabalho, foi realizada a mesma trajetória anterior, porém agora com o canal integral ativo. O erro de acompanhamento da posição pode ser visto na figura 4.43 e dele vemos que o restultado foi o mesmo obtido para a primeira trajetória realizada para este controlador.

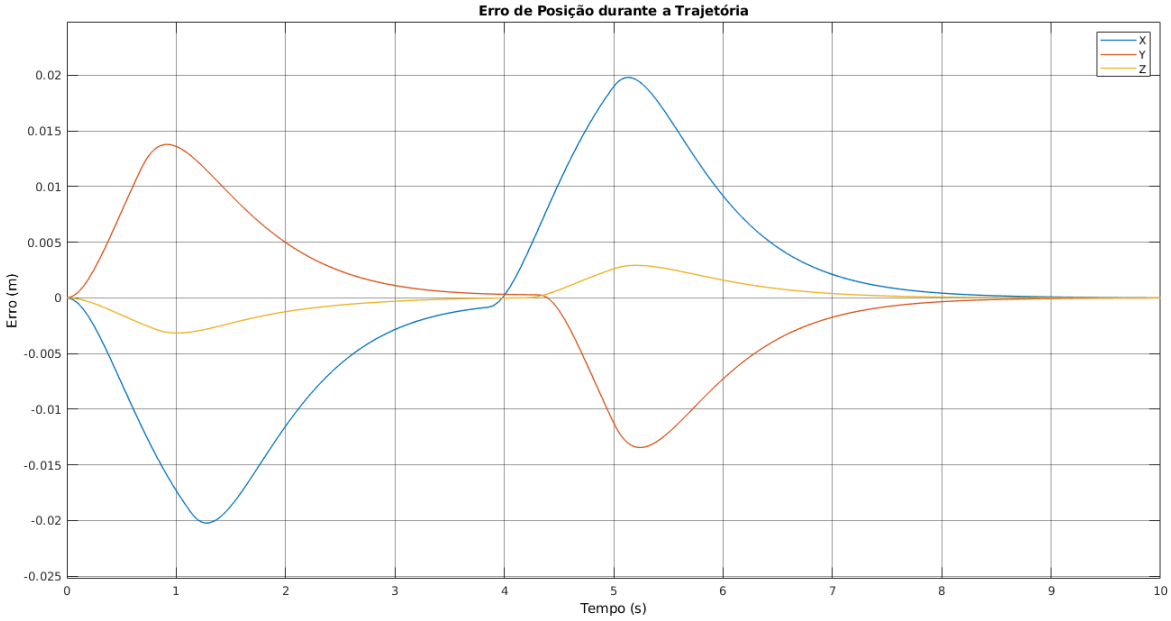

Figura 4.43 – Erro de posição durante a trajetória 2 com canal integral.

5 CONCLUSÃO

O desenvolvimento de aplicações avançadas que necessitam do uso de robôs em fábricas ou em ambientes médicos requerem um cuidadoso planejamento e implementação, tendo em vista que robôs podem levar a risco a integridade física das pessoas. Cada vez mais se torna necessário realizarmos tarefas com a ajuda de robôs na forma de atividades classificadas como cooperação humano-robô, como um robô que auxilia um idoso a levantar ou mesmo um robô que percebe caso entre em colisão com outros robôs ou com um humano.

Quando se trata de manipuladores, uma preocupação que se torna relevante é a força que o robô está aplicando em seu efetuador terminal. Para saber a magnitude da força sendo aplicada em uma atividade ou então o torque sendo aplicado nas juntas do robô durante um movimento, é preciso se ter o conhecimento das equações físicas que relacionam as variáveis que normalmente são medidas, como posição, velocidade e aceleração, e as variáveis de força. Além disso é necessário ter o conhecimento de algumas constantes que são específicas de cada robô ou então te-las estimadas.

Com esses conhecimentos é possível desenvolver aplicações que possam tentar garantir a segurança necessária para que robôs possam trabalhar em conjunto com humanos de forma segura e eficiente. Uma forma de fazer isso é programando o robô para controlar a força aplicada durante um movimento ou então saber qual a força que será aplicada durante uma trajetória para ser possível estimar potenciais colisões.

Este trabalho propôs começar esse processo de desenvolvimento de aplicações de cooperação para o novo manipulador UR3 adquirido recentemente pelo LARA. Foi desenvolvido a cinemática do robô, foram escritas as equações dinâmicas utilizando a parametrização de Newton-Euler, as equações obtidas foram reduzidas utilizando o software Maple e sua biblioteca de simplificação simbólica, foi feita a linearização nos parâmetros das equações dinâmicas utilizando também o Maple, foi realizado o processo de identificação dos parâmetros dinâmicos desconhecidos via algoritmo de mínimos quadrados e, por fim, foram propostos controladores de trajetória para as juntas e de trajetória da ferramenta localizada no efetuador terminal.

Apesar deste trabalho ter sido feito especificamente para o manipulador robótico UR3, os algoritmos feitos foram escritos de forma mais genérica possível para que futuros trabalhos com outras arquiteturas robóticas possam aproveitar do seu desenvolvimento.

A identificação, apesar de ter tido um *fitness* médio de 73,14%, foi considerada satisfatória, tendo em vista que o experimento disponível tinha alguns problemas com a fixação do robô na mesa e não tinha uma duração muito grande, apenas 33 segundos de aprendi-

zado e por volta de 12 segundos de validação. A maioria dos parâmetros físicos obtidos na identificação tem um significado físico condizente com a realidade, como massas positivas etc. Trocar as matrizes de inercia diagonais usadas por matrizes cheias e trocar o sentido da direção dos centro de massa das juntas pode resolver o problema dos parâmetros sem sentido físico identificados.

O controlador de juntas obteve um resultado extremamente satisfatório. Ele foi capaz de ignorar erros do modelo e perturbações aplicadas no torque durante e após o movimento terminar, mantendo o erro baixo e tendendo sempre ao erro nulo nos experimentos com canal integral. Ficando visível que o controlador com o ganho proporcional à integral do erro, apesar de mostrar um leve atraso na resposta, é o com melhor resultado.

Um problema que não foi possível ser resolvido durante o curso do projeto foi a representação da orientação da ferramenta. Infelizmente as limitações da representação utilizada foram observadas muito tardiamente, e a única solução que foi possível implementar, devido à falta de tempo foi ignorar esse aspecto do controlador da ferramenta. Durante alguns testes foi observado que se fosse ligado apenas o ganho de erro de velocidade era obtido um resultado minimamente satisfatório para o controle de orientação. Porém, mesmo com esse problema, o controle da posição no espaço 3D da ferramenta se mostrou útil e preciso.

Muitos testes e verificações poderão ser feitos quando o acesso ao laboratório e a utilização do robô voltarem. Novos testes com outras trajetórias podem ser feitos para explorar toda a área de trabalho do robô e verificar se um experimento mais longo gera resultados melhores para a identificação do manipulador. O controlador, principalmente o de controle das juntas que obteve melhor resultado, desenvolvido no ambiente de simulação Simulink, pode ser adaptado e escrito em C para ser implementado no robô.

Um novo método de representação da orientação da ferramenta pode ser usado no lugar do Roll, Pitch, Yaw tendo em vista que esse tem limitações que impediram o desenvolvimento completo do controlador de posição da ferramenta. Um bom ponto de partida seria estudar a representação via quatérnios duais [24], tornando necessário modificar algumas coisas desenvolvidas neste trabalho como a matriz Jacobiana.

5.1 LIMITAÇÕES DO PROJETO

Uma limitação da arquitetura dos controles desenvolvidos é que eles são baseados em calcular um torque necessário para ser aplicado nas juntas do robô de forma se obter a aceleração necessária para realizar a trajetória de referência. Tendo isso em vista, quando as implementações físicas no robô foram ser feitas é preciso ser possível injetar a corrente correspondente ao torque calculado pelo controlador, porém a arquitetura fechada do UR3 não

permite acesso direto à corrente dos motores. Então é necessário adaptar ou desenvolver um sistema de controle que consiga transformar os sinais de controle de torque para referências de velocidade e aceleração angulares para as juntas, variáveis que são possíveis de se controlar.

5.2 TRABALHOS FUTUROS

Com o objetivo de melhorar os parâmetros identificados para que todos tenham significado físico condizente com a realidade, é possível fazer mudanças na escrita das matrizes de inercia de cada junta do robô de forma que, ao invés de utilizar uma matriz com termos apenas na diagonal principal, utilizar uma matriz triangular. Outra mudança que pode ser feita é verificar se a mudança no sentido da direção dos centros de massa mudam os valores para ter significado físico.

O grande objetivo deste trabalho quando foi proposto era conseguir fazer o controle da força de contato do efetuador terminal do robô, porém isso necessita do uso de um simulador que tenha a capacidade de simular colisões e calcular a força de contato entre dois objetos. Essas informações seriam fornecidas via sensores do UR3 em uma implementação física. Utilizar ou desenvolver um simulador que tenha esse aspecto será necessário para desenvolver tais controladores. O artigo [25] mostra uma forma de calcular as forças de interação durante uma colisão.

Como sugestão de controladores de força a serem desenvolvidos, uma vez que seja possível testá-los, é o chamado Controle de Impedância [26] que consiste em assegurar uma certa interação mecânica entre um objeto e um ambiente incerto ou então o Controle de Impedância Variável [27] onde a impedância varia dependendo da leitura de sensores e predições que o robô faz durante o contato.

REFERÊNCIAS BIBLIOGRÁFICAS

1 SICILIANO, B.; KHATIB, O. *Springer handbook of robotics*. [S.l.]: Springer, 2016.

2 ALBERTIN, M. R. et al. Principais inovações tecnológicas da indústria 4.0 e suas aplicações e implicações na manufatura. *XXIV Simpósio de Engenharia de Produção. Anais..., Bauru*, 2017.

3 BEASLEY, R. A. Medical robots: current systems and research directions. *Journal of Robotics*, Hindawi, v. 2012, 2012.

4 BABAIASL, M. et al. A review of technological and clinical aspects of robot-aided rehabilitation of upper-extremity after stroke. *Disability and Rehabilitation: Assistive Technology*, Taylor & Francis, v. 11, n. 4, p. 263–280, 2016.

5 LIU, D.; KINUGAWA, J.; KOSUGE, K. A projection-based making-human-feel-safe system for human-robot cooperation. In: IEEE. *2016 IEEE International Conference on Mechatronics and Automation*. [S.l.], 2016. p. 1101–1106.

6 AJOUDANI, A. et al. Progress and prospects of the human–robot collaboration. *Autonomous Robots*, Springer, v. 42, n. 5, p. 957–975, 2018.

7 DOBRA, Z.; DHIR, K. S. Technology jump in the industry: human–robot cooperation in production. *Industrial Robot: the international journal of robotics research and application*, Emerald Publishing Limited, 2020.

8 MATHWORKS. *MATLAB - The Language Of Technical Computing*. [S.l.], 2020 (última vez acessado 09, 2020). <https://www.mathworks.com/products/matlab.html>.

9 MATHWORKS. *Simulink - Simulation and Model-Based Design*. [S.l.], 2020 (última vez acessado 09, 2020). <https://www.mathworks.com/products/simulink.html>.

10 MAPLESOFT. *Maple 20 - The Essential Tools for Mathematics and Modeling*. [S.l.], 2020 (última vez acessado 09, 2020). <https://www.maplesoft.com/products/maple/index.aspx>.

11 ROS - Robot Operating System. [S.l.], 2020 (última vez acessado 09, 2020). <https://www.ros.org/>.

12 CRAIG, J. J. *Introduction to robotics: mechanics and control, 3/E*. [S.l.]: Pearson Education India, 2009.

13 UNIVERSAL ROBOTICS. *Universal Robots e-Series User Manual*. 5.5. ed. [S.l.], 2019.

14 SPONG, M. W. et al. *Robot modeling and control*. [S.l.: s.n.], 2006.

15 ABDOLMALAKI, R. Y. Geometric jacobians derivation and kinematic singularity analysis for smokie robot manipulator & the barrett wam. *arXiv preprint arXiv:1707.04821*, 2017.

16 TRUESDELL, C. A. *A first course in rational continuum mechanics*. [S.l.]: Academic Press, 1992. v. 1.

17 BAJD, T. et al. *Robotics*. [S.l.]: Springer Science & Business Media, 2010. v. 43.

18 TAYLOR, J. R. *Classical mechanics*. [S.l.]: University Science Books, 2005.

19 HØIFØDT, H. *Dynamic modeling and simulation of robot manipulators: The newton-euler formulation*. Dissertação (Mestrado) — Institutt for teknisk kybernetikk, 2011.

20 AGUIRRE, L. A. *Introdução à identificação de sistemas–Técnicas lineares e não-lineares aplicadas a sistemas reais*. [S.l.]: Editora UFMG, 2004.

21 GAUTIER, M.; KHALIL, W. Exciting trajectories for the identification of base inertial parameters of robots. *The International journal of robotics research*, Sage Publications Sage CA: Thousand Oaks, CA, v. 11, n. 4, p. 362–375, 1992.

22 PARK, K.-J. Fourier-based optimal excitation trajectories for the dynamic identification of robots. *Robotica*, Cambridge University Press, v. 24, n. 5, p. 625, 2006.

23 GARCIA, P. *Estimação de Força de Interação no Braço Robótico UR3.* 2019. Trabalho de Graduação em Engenharia de Controle e Automação, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF.

24 PHAM, H.-L. et al. Position and orientation control of robot manipulators using dual quaternion feedback. In: IEEE. *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. [S.l.], 2010. p. 658–663.

25 LENS, T.; RADKHAH, K.; STRYK, O. von. Simulation of dynamics and realistic contact forces for manipulators and legged robots with high joint elasticity. In: IEEE. *2011 15th International Conference on Advanced Robotics (ICAR)*. [S.l.], 2011. p. 34–41.

26 SONG, P.; YU, Y.; ZHANG, X. Impedance control of robots: an overview. In: IEEE. *2017 2nd international conference on cybernetics, robotics and control (CRC)*. [S.l.], 2017. p. 51–55.

27 MARTÍN-MARTÍN, R. et al. Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks. *arXiv preprint arXiv:1906.08880*, 2019.

Apêndice

A APÊNDICE A

Neste apêndice vai ser mostrado a matriz Y_6 com os 3 regressores da sexta junta. Essa matriz de regressores é a menor e ocupa todo esse espaço, por isso foi escolhido não escrever todos no trabalho sendo possível ver os outros no GitHub do projeto. Os termos seguem a seguinte nomenclatura: q_i = ângulo medido da junta i, dq_i = velocidade angular medida da junta i, ddq_i = aceleração angular medida da junta i.

$Y_6(1)$ = $(dq6)$

$Y_6(2)$ = $(ddq6 - ddq1 * cos(q2 + q3 + q4 - q5)/2 + ddq1 * cos(q2 + q3 + q4 + q5)/2 - dq4 * dq5 * sin(q5) - dq2 * dq5 * sin(q5) - dq3 * dq5 * sin(q5) - dq1 * dq3 * sin(q2 + q3 + q4 + q5)/2 + dq1 * dq3 * sin(q2 + q3 + q4 - q5)/2 - dq1 * dq2 * sin(q2 + q3 + q4 + q5)/2 + dq1 * dq2 * sin(q2 + q3 + q4 - q5)/2 - dq1 * dq4 * sin(q2 + q3 + q4 + q5)/2 + dq1 * dq4 * sin(q2 + q3 + q4 - q5)/2 - dq1 * dq5 * sin(q2 + q3 + q4 + q5)/2 - dq1 * dq5 * sin(q2 + q3 + q4 - q5)/2 + ddq2 * cos(q5) + ddq3 * cos(q5) + ddq4 * cos(q5))$

$Y_6(3)$ = $(1/4*dq2^2*sin(2*q6)+3/16*dq1^2*sin(2*q4-2*q6+2*q2+2*q3)-3/16*dq1^2*sin(2*q4+2*q6+2*q2+2*q3)-1/8*dq1^2*sin(2*q6)-1/2*dq5^2*sin(2*q6)+1/16*dq1^2*sin(2*q6+2*q5)-1/16*dq1^2*sin(-2*q6+2*q5)-1/8*dq1^2*sin(2*q4+q5+2*q6+2*q2+2*q3)-1/8*dq1^2*sin(2*q4-q5+2*q6+2*q2+2*q3)-1/8*dq1^2*sin(2*q4+q5-2*q6+2*q2+2*q3)-1/8*dq1^2*sin(2*q4-q5-2*q6+2*q2+2*q3)+1/32*dq1^2*sin(2*q4+2*q2+2*q3-2*q6-2*q5)+1/32*dq1^2*sin(2*q4+2*q2+2*q3-2*q6+2*q5)-1/32*dq1^2*sin(2*q4+2*q2+2*q3+2*q6+2*q5)-1/32*dq1^2*sin(2*q4+2*q2+2*q3+2*q6-2*q5)-1/8*dq4^2*sin(2*q6+2*q5)+1/8*dq4^2*sin(-2*q6+2*q5)+1/4*dq4^2*sin(2*q6)-1/8*dq3^2*sin(2*q6+2*q5)+1/8*dq3^2*sin(-2*q6+2*q5)+1/4*dq3^2*sin(2*q6)-1/8*dq2^2*sin(2*q6+2*q5)+1/8*dq2^2*sin(-2*q6+2*q5)-1/8*dq4*dq1*sin(q2+q3+q4-2*q6-2*q5)-1/8*dq4*dq1*sin(q2+q3+q4+2*q6+2*q5)+1/8*dq4*dq1*sin(q2+q3+q4+2*q6-2*q5)+1/8*dq4*dq1*sin(q2+q3+q4-2*q6+2*q5)-1/8*dq3*dq1*sin(q2+q3+q4-2*q6-2*q5)-1/8*dq3*dq1*sin(q2+q3+q4+2*q6+2*q5)+1/8*dq3*dq1*sin(q2+q3+q4+2*q6-2*q5)+1/8*dq3*dq1*sin(q2+q3+q4-2*q6+2*q5)-1/8*dq2*dq1*sin(q2+q3+q4-2*q6-2*q5)-1/8*dq2*dq1*sin(q2+q3+q4+2*q6+2*q5)+1/8*dq2*dq1*sin(q2+q3+q4+2*q6-2*q5)+1/8*dq2*dq1*sin(q2+q3+q4-2*q6+2*q5)+1/4*dq1*dq5*sin(q3+q4+2*q6+q5+q2)+1/4*dq1*dq5*sin(q3+q4-2*q6+q5+q2)+1/4*dq1*dq5*sin(q3+q4+2*q6-q5+q2)-1/2*dq1*dq5*sin(-2*q6+q2+q3+q4)+1/4*dq1*dq5*sin(q3+q4-2*q6-q5+q2)+1/2*dq1*dq5*sin(2*q6+q2+q3+q4)+1/4*dq2*dq1*sin(q3+q4-2*q6-q5+q2)+1/4*dq2*dq1*sin(q3+q4+2*q6-q5+q2)-1/4*dq2*dq1*sin(q3+q4-2*q6+q5+q2)-1/4*dq2*dq1*sin(q3+q4+2*q6+q5+q2)+1/4*dq3*dq1*sin(q3+q4-2*q6-q5+q2)+1/4*$

$$dq3*dq1*sin(q3+q4+2*q6-q5+q2)-1/4*dq3*dq1*sin(q3+q4-2*q6+q5+q2)-1/4*dq3*dq1*sin(q3+q4+2*q6+q5+q2)+1/4*dq4*dq1*sin(q3+q4-2*q6-q5+q2)+1/4*dq4*dq1*sin(q3+q4+2*q6-q5+q2)-1/4*dq4*dq1*sin(q3+q4-2*q6+q5+q2)-1/4*dq4*dq1*sin(q3+q4+2*q6+q5+q2)+1/2*dq4*dq5*sin(q5+2*q6)+1/2*dq4*dq5*sin(q5-2*q6)+1/2*dq2*dq5*sin(q5+2*q6)+1/2*dq2*dq5*sin(q5-2*q6)+1/2*dq3*dq5*sin(q5+2*q6)+1/2*dq3*dq5*sin(q5-2*q6)-1/4*dq2*dq3*sin(2*q6+2*q5)+1/4*dq2*dq3*sin(-2*q6+2*q5)+1/2*dq2*dq3*sin(2*q6)-1/4*dq2*dq4*sin(2*q6+2*q5)+1/4*dq2*dq4*sin(-2*q6+2*q5)+1/2*dq2*dq4*sin(2*q6)-1/4*dq3*dq4*sin(2*q6+2*q5)+1/4*dq3*dq4*sin(-2*q6+2*q5)+1/2*dq3*dq4*sin(2*q6))$$

B APÊNDICE B

Todos os códigos, desenvolvimento e resultados obtidos neste trabalho podem ser visualizados e utilizados a partir do link do: <https://github.com/PedroSaman/UR3-Modelation>. Os códigos estão organizados de forma parecida com que foram explicados neste texto e com comentários do funcionamento de cada parte na língua inglesa. Cada parte do desenvolvimento do trabalho pode verificado nos arquivos:

- Dados MAT: MAT files, são arquivos com valores de experimentos e arquivos BAG do ROS.
- Parâmetros DH: Kinematics\DH\Toolbox.mat
- Cinemática direta: Kinematics\Direct\Kinematics.m
- Calculo da jacobiana: Kinematics\Jacobian.m
- Calculo das trajetórias: Kinematics\Trajectory
- Dinâmica do UR3: Dynamics\BuildUp_UR3\Symbolic_Dynamics\DOF6
- Reescrita das matrizes da dinâmica:
  Identification\Least Square Regression \M_G_C0Dq.m
- Preenchimento das matrizes Y: Identification\Least Square Regression \PsiMatrix.m
- Mínimos Quadrados: Identification\Least Square Regression \Identification6DOF.m
- Controlador de trajetória de juntas: Controler\Joint_Position_Control.slx
- Controlador de trajetória da ferramenta:
  Controler\External_coordinates_Position_Control.slx

Na pasta *Controler* existem os arquivos Simulink dos dois controladores usados e suas dependências, na pasta *Data Manipulations* existem algumas funções de auxilio para diferentes aplicações do projeto como a função do calculo das matrizes de rotação, na pasta *Dynamics* existe as funções de calculo das equações dinâmicas do robô UR3 com diferentes considerações de número de juntas, na pasta *Identification* existem as funções necessárias para implementar o algoritmo de mínimos quadrados e as equações necessárias para escrever as matrizes utilizadas no controlador além das funções feitas no Maple para simplificação matemática, na pasta *Kinematics* foram colocadas as funções de cinemática direta, a que calcula a matriz Jacobiana e as que calculam as trajetórias de referência utilizada no controlador, por fim, na pasta *MAT files* tem os arquivos MAT do Matlab com dados experimentais.