Projeto de Graduação

DEE DEPARTAMENTO DE ENGENHARIA ELÉTRICA

11 de julho de 2019

VALIDAÇÃO EXPERIMENTAL DE METODOLOGIAS DE CONTROLE PARA MANIPULADORES COOPERATIVOS

Gustavo Bertagna Peixoto Barbosa

PUC RIO

www.ele.puc-rio.br

Projeto de Graduação

DEE DEPARTAMENTO DE ENGENHARIA ELÉTRICA

VALIDAÇÃO EXPERIMENTAL DE METODOLOGIAS DE CONTROLE PARA MANIPULADORES COOPERATIVOS

Aluno: Gustavo Bertagna Peixoto Barbosa

Orientador: Antonio Candea Leite

Trabalho apresentado com requisito parcial à conclusão do curso de Engenharia Elétrica na Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil.

Agradecimentos

Agradeço aos meus pais, Angela Maria Bertagna e Augusto Peixoto Barbosa, por sempre priorizar meus estudos e me darem o suporte necessário para a conclusão do curso de graduação.

Agradeço a minha namorada, Eline Correia Ramos, por sempre me dar amor e apoio durante todo o curso da graduação. Graças a ela pude enfrentar diversas dificuldades, e seguir em frente da melhor maneira possível.

Agradeço a minha irmã Natalia Bertagna Peixoto Barbosa e minha avó Alda Peixoto Barbosa, que torceram por mim, e me deram forças para me tornar um engenheiro.

Agradeço ao meu orientador, Antonio Candea Leite, por sempre acreditar em mim, e me dar diversas oportunidades para crescer profissionalmente e como pessoa.

Agradeço ao meu amigo, Adalberto Oliveira, por me dar apoio durante todo o meu projeto, me ajudando na parte técnica, teórica e com a escrita dessa dissertação.

Resumo

Neste projeto, considera-se o problema de modelagem cinemática e controle cinemático de braços robóticos cooperativos para manipulação bimanual de objetos. Desse modo, utiliza-se a abordagem denominada *Symmetric Formulation*. Essa técnica propõe a utilização de *Virtual Sticks*, e define as variáveis do sistema em termos de movimentos absolutos e relativos. Essas variáveis podem ser calculadas diretamente da posição e orientação dos sistemas de coordenadas dos *Virtual Sticks* de cada manipulador serial. A verificação e a validação dos algoritmos de controle propostos são realizados por meio de simulações numéricas em MATLAB/Simulink e ensaios experimentais com dois robôs manipuladores PhantomX Pincher Arm Kit utilizando a plataforma Robot Operating System (ROS).

Palavras-chave: Controle,Algoritmos,Cinemático, Cooperativo

Abstract

In this paper, It is considered the problem of kinematic modelling and kinematic control of cooperative manipulator robots for bimanual manipulation of objects. In this way, the Symmetric Formulation is proposed. This approach uses Virtual Sticks and define variables in terms of absolute and relative movements of the cooperative system, which can be calculated directly from the position and orientation of the coordinator systems of the virtual stick of each serial manipulator. The verification and validation of the proposed control algorithms are performed through numerical simulations in MATLAB/Simulink and experimental tests with two PhantomX Pincher Arm Kit manipulators using the Robot Operating System (ROS) platform.

Keywords: Control,Algorithms,Kinematic,Cooperative

Sumário

Lista de Figuras

1 Introdução

Os manipuladores robóticos estão sendo amplamente utilizados em diversas áreas que há trabalho insalubre, perigoso e penoso, como no ambiente fabril. Dessa forma, o principal local aonde os braços robóticos são aplicados, é na indústria, pois são capazes de produzir peças cada vez mais complexas e maiores que não são possíveis serem feitas por um ser humano. Esses robôs oferecem diversas vantagens para o aprimoramento da linha de produção, como a padronização do produto, agilidade na fabricação, maior qualidade no resultado final, maior precisão e repetibilidade nas tarefas.

Contudo, em algumas tarefas são necessários dois ou mais braços que atuem de maneira conjunta e síncrona. Desse modo, utilizando um sistema robótico cooperativo, é possível ampliar a precisão e a capacidade de carga se comparado a um simples manipulador. Uma aplicação muito comum desse tipo de sistema, é o uso de dois robôs tele operados. Com isso, um dos manipuladores se encontra em um ambiente seguro junto com o operador, e o segundo encontra-se em um ambiente nocivo ao homem. Dessa maneira, o segundo braço imita os movimentos do primeiro, sem que o operador esteja em risco na área perigosa. Essa arquitetura é chamada de *Master-Slave*, e são vantajosas em ambientes, como no espaço, no fundo do oceano, na medicina, ou em usinas nucleares.

Portanto, nesse projeto será estudado sobre o tipo de sistema robótico detalhado no parágrafo anterior, no contexto de manipulação bi-manual cooperativa de cargas e objetos. Com isso, por meio dos conceitos estudados, será possível implementar um algoritmo em que os manipuladores executem movimentos seguindo uma trajetória desejada e manuseando um objeto. Através dessa aplicação, a capacidade de carga dos braços cooperativos será ampliada, se comparada com somente um braço robótico.

Nesse contexto, para implementar a manipulação de um objeto de forma cooperativa, serão estudadas algumas metodologias para a modelagem cinemática e o controle cinemática de sistemas robóticos desse tipo. Primeiramente, será investigado uma abordagem chamada de *Symmetric Formulation*, onde as variáveis dos manipuladores cooperativos serão divididas em absolutas e relativas. Com isso, serão utilizadas essas variáveis para implementar algumas leis de controle para controlar o sistema. Esses conceitos serão utilizados simulados e validados com uso de dois braços robóticos *PhantomX Pincher Arm* e com o auxílio da plataforma ROS *(Robot Operating System)*.

Desso forma, a presente dissertação está organizada da seguinte forma: No capítulo 2 será apresentado sobre o estado da arte do tema estudado. Portanto, serão citadas algumas pesquisas que utilizaram os principais tópicos desse projeto. No capitulo 3, serão apresentados os conceitos fundamentais utilizados para a realização do projeto, como a modelagem e o controle cinemático de um simples manipulador, a conversão de sistemas contínuos para sistemas discretos, e por último sobre a plataforma ROS. No capitulo 4, é feito um estudo da modelagem e do controle cinemático de um sistema robótico cooperativo com dois manipuladores utilizando alguns conceitos abordados no capítulo anterior. No capítulo 5 é estudado profundamente sobre o manipulador PhantomX Pincher Arm, e seus componentes. No capítulo 6 será realizada uma análise dos resultados da simulação numérica feita no Matlab/Simulink. No capítulo 7, serão apresentados os resultados da validação do algoritmo de controle implementado, e será feita sua análise. Por ultimo, no capítulo 8, é feita a conclusão do projeto.

2 Revisão do estado da Arte

Nesta seção, serão abordados algumas pesquisas que utilizaram os principais tópicos desse projeto. Dentre eles, podem ser citados, Modelagem e Controle de Sistemas Cooperativos. Dessa maneira, na próxima subseção, serão apresentadas duas pesquisas relacionadas a modelagem de robôs cooperativos. Na primeira, é apresentada uma abordagem que realiza a modelagem de robôs cooperativos utilizando as restrições cinemáticas impostas [1]. A segunda pesquisa, descreve como é feita a modelagem de um robô Baxter. Dessa maneira, utiliza-se o ROS *(Robot Operating System)*e o Matlab/Simulink para validar o modelo através de uma simulação gráfica [2]. Em seguida, na terceira pesquisa, será discutido uma abordagem de controle utilizando Realimentação de força, para que se tenha maior robustez ao fazer a manipulação bi-manual de um objeto [3]. Por último, será apresentada uma pesquisa que utiliza controle adaptativo, para estimar parâmetros cinemáticos e dinâmicos do sistema [4].

a Modelagem de Sistemas Robóticos

Nessa primeira pesquisa, sobre modelagem, é proposta uma abordagem que utiliza as restrições do sistema. Essas restrições são impostas pela manipulação de um objeto por dois robôs trabalhando de forma cooperativa. Dessa maneira, devido ao contato constante entre os robôs e o objeto, assume-se que o sistema seja um mecanismo paralelo. Com isso, assume-se que a distância entre o End-effector dos manipuladores e o centro de massa do objeto, sejam dois elos que estão conectados. Também é assumido que o contato entre o robô e o objeto, é associado a uma junta passiva. Desse jeito, é criada uma restrição, pois a pose do centro de massa do objeto deve ser igual para os dois robôs, em relação a um sistema de coordenadas localizado na base comum para os dois manipuladores. Depois que é encontrado o modelo matemático da restrição cinemática, é encontrado o Jacobiano de restrição do sistema. Dessa forma, utiliza-se os Jacobianos de cada manipulador e o Jacobiano de restrição para encontrar o Jacobiano para o sistema cooperativo, em função somente das juntas ativas.

Na segunda pesquisa, é modelado um robô humanoide chamado de Baxter, que pode ser visto como um sistema de manipuladores cooperativos, devido ao seus dois braços que podem ser encarados como dois manipuladores. Esse robô possui 7 juntas e 8 elos em cada braço. Dessa forma, nesse trabalho, é utilizado a notação de Denavit-Hartenberg para encontrar as equações da cinemática de cada braço do Baxter. Depois de encontrar as equações cinemática direta do manipulador, é utilizado os software Matlab/Simulink e o ROS (Robot Operation System) para simular o robô e verificar a validação da modelagem cinemática.

b Controle de Sistemas Robóticos

Nessa pesquisa, é proposto um esquema de controle híbrido de posição/força, que utiliza a dinâmica e a cinemática do sistema robótico cooperativo para seguir uma trajetória desejada, manipulando um objeto. Com isso, primeiro, para descrever o sistema, utiliza-se a abordagem de Task Space Formulation, que modela o sistema em função de variáveis absolutas e relativas ao movimento dos manipuladores. Dessa maneira, utilizando essa abordagem, não é preciso saber as reais dimensões do objeto manipulado. Contudo, para obter um sistema mais robusto, para que os robôs não exerçam

uma força exagerada que poderia danificar o objeto, utiliza-se um controle de força. Desse jeito, utiliza-se o controle cinemático, aliado ao controle de força para que os manipuladores sigam uma trajetória desejada.

Na última pesquisa, é proposta uma lei de controle adaptativo para fazer a estimação de parâmetros dinâmicos e cinemáticos incertos dos manipuladores na tarefa de transporte de carga. Aliado a isso, também é proposto uma lei de controle adaptativo para estimar alguns parâmetros cinemáticos da carga, transportada pelo sistema robótico. Além de utilizar o esquema de controle adaptativo proposto para objetos que estão rigidamente anexados aos efetuadores, é proposto um esquema de controle adaptativo para a dinâmica e cinemática de manipuladores que estão sendo tele-operados

3 Conceitos fundamentais de cinemática para manipuladores

Os braços robóticos são compostos por juntas, que podem ser rotativas ou prismáticas, e por elos. Desse modo, as juntas unem os elos, que são corpos rígidos, e assim é formada uma cadeia cinemática. Um extremo dessa cadeia está fixado na base e o outro extremo está fixado no efetuador (*end-effector*). Com isso, cada junta do manipulador adiciona um grau de liberdade e está associada a uma variável da junta (ângulo ou deslocamento). Portanto, caso for desejado manipular um objeto no espaço, é necessário descrever a posição e a orientação do efetuador. Desse modo, é possível utilizar a cinemática para estudar a pose e a velocidade do *end-effector* e dos seus elos. Na figura 1, a seguir é possível visualizar uma cadeia cinemática:

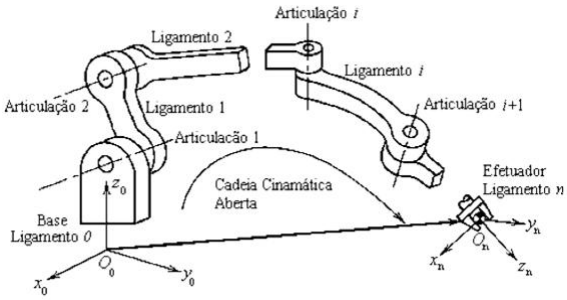

Figura 1: Cadeia Cinemática Aberta

Nas próximas subseções, serão abordados alguns conceitos fundamentais de cinemática aplicados em manipuladores robóticos seriais. Por meio desses conceitos, será possível realizar a modelagem e o controle dos manipuladores robóticos, como é desejado nesse projeto.

a Cinemática Direta

A cinemática direta tem o objetivo de calcular a posição do efetuador em função dos ângulos das juntas com relação a um sistema de coordenadas fixo na base. Dessa maneira, dado o manipulador da figura 2:

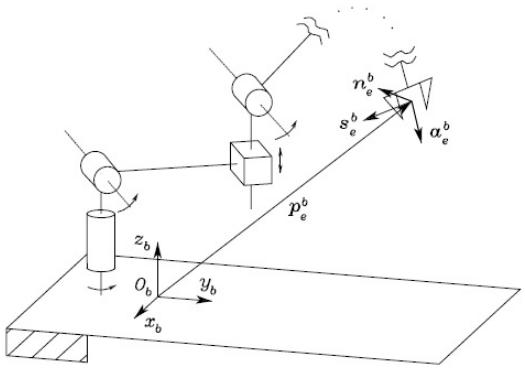

Figura 2: Descrição da pose do efetuador

Desse modo, é possível visualizar na figura acima, que o vetor p_e^b, que possui sua extremidade no *end-effector*, e sua origem na base, descreve a pose do efetuador em relação ao sistema de

coordenadas da base. Portanto, temos os seguintes sistemas de coordenadas:

$$\overline{E}_b = \begin{bmatrix} \vec{x_b} & \vec{y_b} & \vec{z_b} \end{bmatrix}, \tag{1}$$

$$\overline{E}_e = \begin{bmatrix} \vec{n} & \vec{s} & \vec{a} \end{bmatrix}, \tag{2}$$

Com isso, $\overline{E}_b$ descreve o sistema de coordenadas da base e $\overline{E}_e$ descreve o sistema de coordenadas do *end-effector*. O sistema de coordenadas do efetuador, é escolhido de acordo com a geometria da tarefa. Desse modo, quando o efetuador é um *gripper*, a origem do seu sistemas de coordenadas, é localizado no centro do *gripper*. O vetor $\mathbf{a}_e^b(q)$ é escolhido na direção de aproximação do objeto, o vetor $\mathbf{s}_e^b(q)$ é escolhido normal ao plano de deslizamento do *gripper*, e o vetor $\mathbf{n}_e^b(q)$ é escolhido normal aos dos outros vetores. Logo, utilizando esses conceitos, é possível escrever uma matriz de transformação homogênea , que descreve a cinemática direta do manipulador:

$$T_e^b(q) = \begin{bmatrix} n_e^b(q) & s_e^b(q) & a_e^b(q) & p_e^b(q) \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3}$$

Onde o vetor $q \in \mathbb{R}^{nx1}$ descreve as variáveis das juntas do manipulador robótico. Desse modo, por meio dessa matriz, é possível fazer a mudança de coordenadas de um sistema de coordenadas para um outro. Outro aspecto importante dessa matriz, é que sua última coluna representa a pose do efetuador em relação ao sistema de coordenadas da base. Dessa maneira, pode-se utilizar geometria básica para realizar o cálculo dessa matriz, como será visto agora na figura 3:

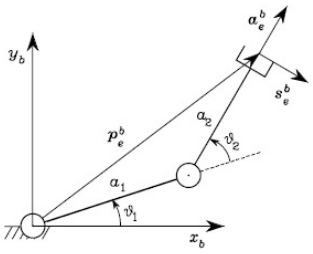

Figura 3: Exemplo cinemática direta robô planar 2DoF

Com isso, escolhendo as variáveis da junta e os sistemas de coordenadas da base e do efetuador de acordo com a figura acima, utiliza-se trigonometria, e decomposição de vetores para encontrar a matriz de transformação homogênea. Portanto, a matriz é dada por [5]:

$$\begin{bmatrix} 0 & s_{12} & c_{12} & a_1c_1 + a_{12}c_{12} \\ 0 & -c_{12} & s_{12} & a_1s_1 + a_{12}s_{12} \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4}$$

Contudo, se número de juntas e de elos for maior, o robô terá mais graus de liberdade, e não se movimentará somente em um espaço de 2 dimensões. Dessa maneira, é preciso que sejam usados métodos mais robustos que realizem o cálculo da matriz de transformação homogênea. Portanto, na próxima subseção, será abordado um método que executa essa tarefa.

b Convenção de Denavit-Hartenberg

Afim de computar a cinemática direta para um braço robótico serial, utiliza-se uma maneira recursiva para a obtenção da pose relativa entre dois elos consecutivos. Dessa forma, ao invés dos sistemas de coordenadas serem escolhidos de forma arbitrária, eles serão configurados de acordo com algumas regras impostas pela convenção de Denavit-Hartenberg. Desse modo, será mais fácil encontrar a transformação homogênea entre os dois elos. Portanto, na figura 4, a seguir, é possível ver uma parte de um manipulador, onde será aplicado essas regras:

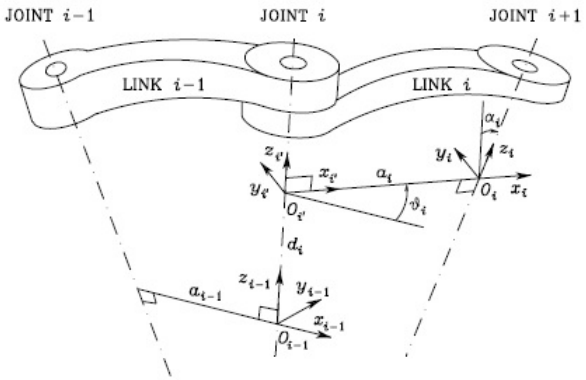

Figura 4: Exemplo da convenção de Denavit-Hartenberg

Com isso, será utilizada a convenção de Denavit-Hartenberg, para definir o sistema de coordenadas da junta i, que conecta os elos i e i-1. Logo, serão aplicadas as seguintes regras [5]:

1. Escolher o eixo z_i no sentido da junta i+1.
2. Localizar a origem de O_i, que é a intersecção com a normal comum dos eixos z_{i-1} e z_i. Também localizar $O_{i'}$ na interseção da normal comum com o eixo z_{i-1}.
3. Escolher o eixo x_i ao longo da normal comum enter os eixos z_{i-1} e z_i com direção da junta i para a junta i+1.
4. Escolher o eixo y_i para completar a regra da mão direita.

Há algumas exceções para a notação de Denavit-Hartenberg que são:

- Para o sistema de coordenadas da junta 0, somente z_0 é definido, portanto O_0 e x_0 podem ser escolhidos de forma arbitrária.
- Para o sistema de coordenadas n não há uma única configuração. Equanto x_n é normal ao eixo z_{n-1}. Geralmente, a junta n é de revolução, logo z_n deve ser alinhado com a direção z_{n-1}
- Quando há dois eixos que se interceptam, a direção de x_i é arbitrária.
- Quando a junta i é prismática, a direção de z_{i-1} é arbitrária.

Após realizar os procedimentos acima, para escolher os sistemas de coordenadas de todas as juntas, serão encontrados os parâmetros de Denavit-Hartenberg:

- a_i - Distância entre O_i e $O_{i'}$ (comprimento da normal comum).
- d_i - Coordenada de $O_{i'}$ ao longo de z_i (distância entre $O_{i'}$ e a normal comum).

- α_i - Ângulo entre os eixos z_{i-1} e z_i ao longo da direção do eixo x_i. A rotação será positiva quando for feita no sentido anti-horário.
- θ_i Ângulo entre os eixos x_{i-1} e x_i ao longo da direção do eixo z_i. A rotação será positiva quando for feita no sentido anti-horário.

Dois dos quatro parâmetros são constantes, e dependem somente da geometria da conexão entre duas juntas consecutivas, ligadas por um elo. Contudo, dois parâmetros dependem do tipo da junta, que são:

- a_i - Se a junta for de rotação, então θ_i é variável.
- a_i - Se a junta for prismática, então d_i é variável.

Após encontrar os parâmetros de Denavit-Hartenberg, é possível calcular a matriz de transformação utilizando eles:

$$A_i^{i-1}(q_i) = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5}$$

Dessa forma, é possível computar a transformação homogênea $T_n^0(q)$, fazendo a multiplicação $A_1^0 A_n^{n-1}$. Essa matriz representa a orientação e a posição do sistema de coordenadas da junta n, com respeito ao sistema de coordenadas da junta 0. Logo, é calculada a função da cinemática direta $T_e^b(q) = T_0^b T_n^0 T_e^n$. Com isso, encontra-se a matriz homogênea do *end-effector* com respeito ao sistema de coordenadas da base. A matriz $A_i^{i-1}(q_i) \in \mathbb{R}^{4\times4}$

c Cinemática Diferencial

Anteriormente, foi visto a cinemática direta, que relaciona a pose do efetuador, com as variáveis das juntas. Nessa seção, será abordada a cinemática diferencial, que relaciona as velocidades das juntas com as velocidades lineares e angulares do *end-effector*. A matriz que relaciona essas velocidades, é chamada de matriz Jacobiana. O Jacobiano do manipulador, pode ser calculado de duas maneiras: (1) analiticamente ou (2) geometricamente. A abordagem geométrica, utiliza as configurações do manipulador para calcular a matriz jacobiana. Por outro lado, a abordagem analítica utiliza a derivada da cinemática direta para calcular a matriz Jacobiana. O Jacobiano de um braço robótico é uma das relações mais importantes para a análise e controle do movimento de um robô manipulador. Desse modo, através dele, é possível ter as seguintes informações:

- Planejamento e execução de trajetórias suaves.
- Determinação da singularidade.
- Cálculo de equações dinâmica de movimento.
- Transformação de forças e torques do efetuador para as juntas.

Portanto, através do Jacobiano, tem-se a seguinte relação:

$$v_e = J(q)\dot{q}, \tag{6}$$

Aonde $\mathbf{v}_e \in \mathbb{R}^{6\times1}$, a matriz $\mathbf{J} \in \mathbb{R}^{6\times n}$, e o vetor de juntas $\dot{\mathbf{q}} \in \mathbb{R}^{6\times1}$. Contudo, a expressão acima pode ser dividida em duas:

$$\dot{p}_e = J_p(q)\dot{q}, \tag{7}$$

$$\omega_e = J_o(q)\dot{q}, \tag{8}$$

Com isso, tem-se que $\dot{p}_e \in \mathbb{R}^{3\times1}$, pois nele contém as velocidades lineares $\begin{bmatrix}\dot{x} & \dot{y} & \dot{z}\end{bmatrix}$. O jacobiano $\mathbf{J}_p \in \mathbb{R}^{3\times n}$. Desse modo, de forma similar, $\omega \in \mathbb{R}^{3\times1}$, já que nele está contido as velocidades angulares do efetuador $\begin{bmatrix}\dot{\phi} & \dot{\theta} & \dot{\psi}\end{bmatrix}$. O jacobiano $J_o \in \mathbb{R}^{3\times n}$. Com isso, o jacobiano geométrico do manipulador é dado por:

$$\begin{bmatrix}J_p\\J_o\end{bmatrix}, \tag{9}$$

Para computar esse Jacobiano,tem-se a seguinte regra caso a junta i seja prismática:

$$\begin{bmatrix}J_{p_i}\\J_{o_i}\end{bmatrix} = \begin{bmatrix}z_{i-1}\\0\end{bmatrix}, \tag{10}$$

Para o caso que a junta i seja rotativa, o seguinte jacobiano será utilizado:

$$\begin{bmatrix}J_{p_i}\\J_{o_i}\end{bmatrix} = \begin{bmatrix}z_{i-1}\times(p_e - p_{i-1})\\z_{i-1}\end{bmatrix}, \tag{11}$$

Além de calcular o Jacobiano de forma geométrica, também é possível calcular o Jacobiano de forma analítica, derivando as equações da cinemática direta:

$$\dot{p}_e = \frac{\partial p_e}{\partial q}\dot{q} = J_p(q)\dot{q}, \tag{12}$$

$$\dot{\phi}_e = \frac{\partial \phi_e}{\partial q}\dot{q} = J_\phi(q)\dot{q}, \tag{13}$$

Portanto, tem-se que o Jacobiano linear é dado por:

$$J_A(q) = \begin{bmatrix}J_p(q)\\J_\phi(q)\end{bmatrix}, \tag{14}$$

d Singularidade e Manipulabilidade

Antes de apresentar os conceitos de singularidade e manipulabilidade de um braço robótico, é preciso entender o que é o espaço de trabalho. O *workspace* representa a parte do ambiente que o *end-effector* pode acessar. Dessa forma, o volume e a forma da área acessível, depende da estrutura do manipulador, e também das limitações das juntas [5]. Portanto, se o robô chegar na fronteira do seu espaço de trabalho, ele entrará em uma configuração singular e assim, o seu Jacobiano geométrico perderá posto e se tornará não inversível. Com a perda de posto, o manipulador pode se tornar incontrolável. Logo, o conceito de singularidade é um estudo muito importante pelas seguintes razões:

- As singularidades representam configurações em que a mobilidade do robô fica reduzida, ou seja, não é possível impor um movimento arbitrário no efetuador.
- Quando o robô atinge a singularidade, há infinitas soluções para a cinemática inversa.
- Na vizinhança da singularidade, pequenas velocidades no espaço operacional, representam grandes velocidades no espaço das juntas.

O conceito visto acima, é chamado de singularidade de fronteira. Contudo, existe outras configurações, dentro do *workspace* em que o manipulador atinge a singularidade. Isso ocorre, quando dois ou mais eixos de movimentos se alinham, ou os efetuadores alcançam alguma particularidade. Essas configurações são mais problemática do que a singularidade de fronteira, pois ocorrem em qualquer lugar do espaço operacional. Esse tipo de singularidade é denominada de singularidades internas.

Desse modo, através da manipulabilidade, torna-se possível medir de maneira quantitativa o quão próximo o braço robótico está perto da singularidade. Essa variável é calculada da seguinte forma:

$$w(q) = \sqrt{det(J(q)J^T(q))}, \tag{15}$$

Portanto, tenta-se evitar valores de manipulabilidade muito próximos de zero.

e Controle Cinemático

Agora, depois de estudar a modelagem cinemática de um manipulador serial, será visto como realizar o controle cinemático desse tipo de robô, para que ele siga uma trajetória desejada. Dessa maneira, será desprezada a dinâmica do sistema. Essa hipótese, torna-se aceitável quando se tem as seguintes condições:

- Elevados fatores de redução nas engrenagens.
- São utilizadas baixas velocidades na realização da tarefa desejada.
- Existe uma malha de controle de velocidade de alto desempenho para cada junta.

Se for considerado que a maioria dos manipuladores possuem uma malha de controle de velocidade em nível de juntas, como na figura 5.

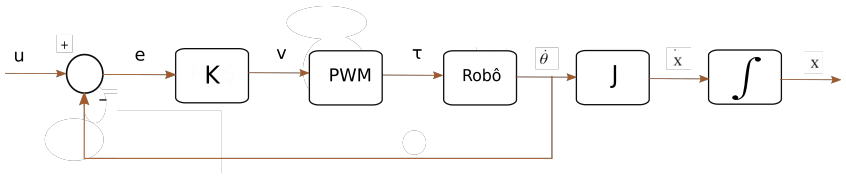

Figura 5: Malha de controle de velocidades em nível de juntas

Para uma entrada $u = \dot{q}_d$ e um controle de alto ganho, isto é, $\lim_{K\to\infty} e = 0$. Portanto:

$$u \approx \dot{q}, \tag{16}$$

Logo, a entrada $u \in \mathbb{R}^{n\times 1}$ igual as velocidades das juntas $\dot{q}$. Com isso tem-se o seguinte sistema de controle:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} \approx J(q)u, \tag{17}$$

Como dito acima, o objetivo do controle é é seguir uma trajetória desejada:

$$x \to x_d(t) \tag{18}$$

$$e = x_d - x \to 0, t \to \infty, \tag{19}$$

É possível calcular a dinâmica do erro, derivando-o:

$$\dot{e} = \dot{x_d} - \dot{x}, \tag{20}$$

Com isso, utilizando a equação (17), chega-se a seguinte expressão:

$$\dot{e} = \dot{x_d} - J(q)u, \tag{21}$$

Desse modo, é escolhido uma entrada do tipo:

$$u = J^{\dagger}(\dot{x_d} + Ke), \tag{22}$$

É escolhida uma matriz de ganho $K \in \mathbb{R}^{n \times 1}$ para multiplicar com erro. Essa matriz de ganho, deve ser positiva definida. Logo, utilizando esta lei de controle, tem-se que:

$$\dot{e} + Ke = 0, \tag{23}$$

Com essa lei de controle, a função do erro será:

$$e(t) = exp(-Kt)e_0, \tag{24}$$

Dessa maneira, por meio da função acima, percebe-se que quando $t \to \infty$, $e(\infty) \to 0$.

O nome do controlador utilizado, é proporcional com feedforward, e ele garante que o erro irá para 0 no regime permanente. Isso é possível, porque $\dot{x_d}$ evita a existência de um erro em estado estacionário. Caso fosse utilizado somente um controlador proporcional, o erro não tenderia a zero ao passar do tempo. Outra alternativa para eliminar o erro em regime permanente, é a implementação de controladores PI ou PID. A seguir é possível ver, na figura 6, o diagrama de blocos de um controle feedforward + proporcional:

f Discretização de Sistemas Contínuos

A grande maioria dos processos físicos são analógicos, porém, quando são utilizados computadores digitais, como os microcontroladores, na malha de controle, é necessário realizar uma discretização das variáveis. Desse modo, serão apresentados alguns conceitos para a implementação das leis

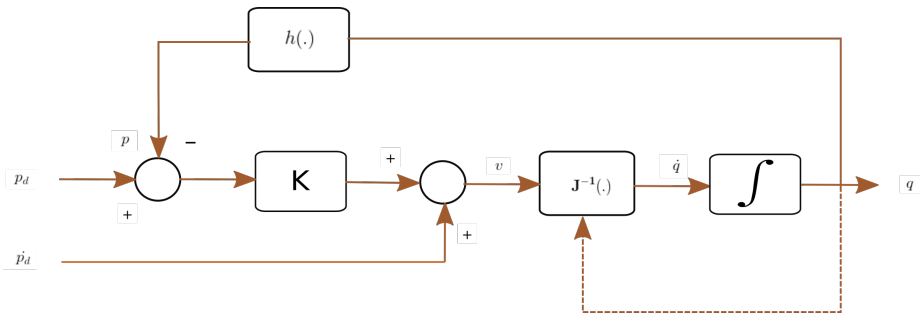

Figura 6: Diagrama de blocos do controle feedforward+proporcional

de controle em controladores digitais. Com isso, é preciso fazer a conversão de equações diferencias para equações algébricas. As técnicas mais utilizadas para executar esta tarefa são:

- Método de Euler Avançado (*Forward rectangular rule*)
- Método de Euler Atrasado (*Backward rectangular rule*)
- Aproximação de Tustin ou Aproximação Bilinear *(Trapezoid Rule)*

Para realizar a passagem de um sinal analógico para um sinal digital, são utilizados conversores A/D. Esses conversores transformam um sinal analógico, contínuo no tempo, em um sinal amostrado, discreto no tempo, quantizado dentro de um número finito de valores, que são determinados pela resolução do conversor em bits. Por exemplo, em um conversor de 10 bits, o sinal de entrada é transformado em amostras com valores entre 0 e 1024. Isso significa que o equipamento terá uma resolução de aproximadamente 0.1 %. A conversão dessa entrada, geralmente ocorre de forma repetida em períodos de T segundos de diferença. Desse modo, a letra h é utilizada para representar o período de amostragem em segundos.

Nesse contexto, uma das operações que será discretizada, é integração, pois não é possível realizá-la no mundo digital. Portanto, a seguir, será abordado como é será feita a discretização da integração usando os métodos de aproximação mencionados acima.

1. **Método de Euler Avançado (*Forward rectangular rule*)**

   Nesse método é assumido que:

$$s = \frac{z-1}{h}, \tag{25}$$

   Dessa maneira, admite-se que a entrada é uma função U(s) e a saída é a integral dessa função, denominada Y(s). A integração no sistema contínuo é dada por:

$$U(s) = sY(s), \tag{26}$$

   Utilizando a equação (22), é possível discretizar o sistema da seguinte maneira:

$$U(z) = \frac{(z-1)Y(z)}{h}, \tag{27}$$

   Simplificando a equação, tem-se que:

$$zY(z) = Y(z) + hU(z), \tag{28}$$

Utilizando a seguinte propriedade de transformada z:

$$zY(z) = y(k+1), \tag{29}$$

Através das equações (25) e (26), é obtido o seguinte resultado:

$$y(k+1) = y(k) + hu(k), \tag{30}$$

Portanto, a próxima amostra da integral, depende do valor atual dela e da sua derivada.

2. **Método de Euler Atrasado (*Backward rectangular rule*)**

   Nesse próxima método, assume-se que:

$$s = \frac{z-1}{hz}, \tag{31}$$

   Utilizando a equação (23) e aplicando nela a equação (28), tem se que:

$$zY(z) = Y(z) + hzU(z), \tag{32}$$

   Com isso, usando a propriedade (26), chega-se ao seguinte resultado:

$$y(k+1) = y(k) + hu(k+1), \tag{33}$$

   Logo, percebe-se que a integral da próxima amostra depende do valor atual da função e da próxima amostra da sua derivada.

   Propriedades desses métodos:

   - Podemos realizar mapeamento do plano S para o Z, de acordo com a figura 7 é:

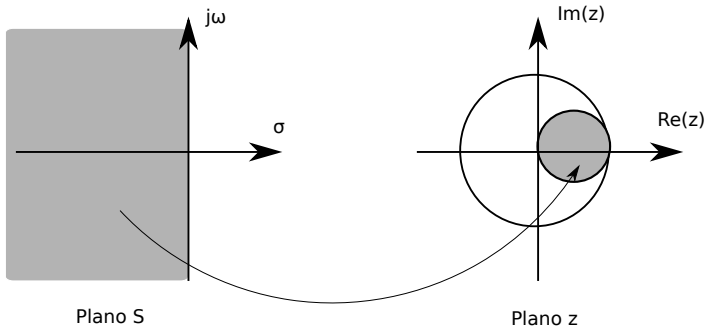

Figura 7: Mapeamento plano S - plano z método de Euler

   - Não preserva respostas ao impulso e em frequência.
   - Fácil Implementação.

3. **Aproximação de Tustin ou Aproximação Bilinear *(Trapezoid Rule)***

   Nesse último método, a variável s é dada por:

$$s = \frac{2(z-1)}{h(z+1)}, \tag{34}$$

   Substituindo essa equação, na equação (23), chega-se ao resultado:

$$zY(z) = Y(z) + \frac{h}{2}(zU(z) + U(z)), \tag{35}$$

   Dessa forma, aplicando a propriedade (26), a nova expressão será dada por:

$$y(k+1) = y(k) + \frac{h}{2}(u(k+1) + u(k)), \tag{36}$$

   Portanto, percebe-se que a amostra seguinte da integral, nesse método, depende da amostra atual da função e das amostras recentes e futuras da derivada da função. Em seguida, serão listadas algumas propriedades desse tipo de discretização

   Propriedades:

   - Não preserva respostas ao impulso e em frequência.
   - Transforma todo semi-plano esquerdo do plano s, no círculo unitário do plano z.

     Consequentemente, pelo fato dessa discretização realizar o mapeamento de todo semi-plano esquerdo de s no círculo unitário z, ela se torna mais vantajosa que os métodos de Euler avançado e atrasado. Dessa forma, a seguir, é possível visualizar na figura 8 essa segunda propriedade:

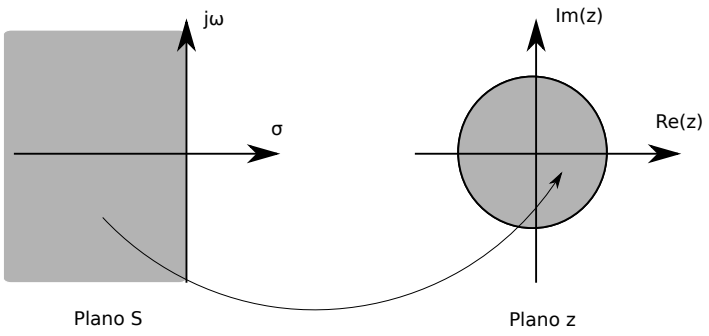

Figura 8: Mapeamento plano S - plano z método de Tustin

   Esse são os métodos utilizados para a discretização do sistema.

g ROS *(Robot Operating System)*

Após ser discutido sobre alguns conceitos de manipuladores e sistemas discretos, agora será apresentado o ROS, que é a plataforma utilizada para comunicar o computador com o robô. O ROS é um sistema operativo para robôs como é descrito pelo seu nome. Ele possui um *framework* muito flexível e contém várias ferramentas, bibliotecas, e convenções que facilitam as tarefas na programação de um robô. Esse sistema é baseado em Linux Ubuntu, e pode ser programado em C, C++ e Python. O

ROS é muito útil para sistemas robóticos que apresentam diversos módulos, como câmeras, encoders, sensores de ultrassom, etc e atuadores que se comunicam entre si. Desse modo, através do ROS, é possível obter uma arquitetura de processamento de dados em paralelo, para controlar o robô. Um exemplo disso, é caso o controlador, embarcado no sistema robótico, precise realizar cálculos pesados de processamento de imagem, porém não tem configurações suficientes para esta tarefa. Logo, é possível criar, por meio do ROS, um nó em um computador mais poderoso, que se comunicará através do protocolo TCP/IP com o robô.

Dessa forma, em uma rede que está rodando o ROS, sempre existirá uma máquina denominada *Master*. O *Master* sempre terá que rodar o comando roscore, que inicializará todos os serviços necessários para que os nós do ROS comuniquem entre si. Um nó no ROS é como um programa executável. A comunicação entre esses nós, é feita por meio do protocolo TCP/IP, como mencionado no parágrafo anterior. Cada um desses nós possui capacidade para publicar ou sub escrever nos tópicos. Um tópico pode ser visto como uma variável, ou seja, um nó pode escrever ou ler dados dessa variável. Logo, é assim que os nós se comunicam entre si. Através do ROS, é possível criar pacotes em C++ e Python, e cada um desses pacotes, possuem um número de nós.

A seguir serão explicados de forma mais detalhada esses conceitos do ROS, mencionados nos dois parágrafos acima, e também alguns comandos utilizados nessa plataforma.

1. **Nós:**

   Como dito no texto acima, um nó é como se fosse um executável. Alguns comandos que podem ser implementados no terminal do Linux, que possuem relação com os nós são:

   - roscore - Serve para inicializar o ROS na máquina do usuário. Através desse comando, serão criados um nó e dois tópicos.
   - rosnode - É um comando utilizado para obter mais informações sobre os nós que estão sendo executados pelo ROS nas máquinas locais ou remotas conectadas na mesma rede. Com isso, utilizando o comando rosnode list, é possível saber todos os nós que estão sendo executados na rede naquele momento. Para saber mais informações de um determinado nó, basta utilizar o comando rosnode info.
   - rosrun - Através desse comando, é possível inicializar um nó de qualquer pacote ROS do sistema que está o computador local.
   - roslaunch - Roda todos os nós ao mesmo tempo.

2. **Tópicos:**

   Os tópicos são semelhantes as variáveis globais. Dessa forma, um nó pode escrever em um tópico, ou pode ler dados de um tópico. Alguns comandos importantes para os tópicos são:

   - rostopic list - Esse comando é similar ao comando rosnode list. Dessa maneira, ele imprime uma lista de todos os tópicos que estão sendo publicados ou sub escritos no sistema.
   - rostopic type - Com esse comando, é possível visualizar o tipo dos dados que um certo tópico utiliza.
   - rostopic info - Esse comando mostra todas as informações de um certo tópico.
   - rostopic pub - Esse comando permite enviar dados para um certo tópico.

- rostopic echo - Esse comando permite visualizar os dados que estão sendo publicados em um certo tópico.

3. **Mensagens:**

   As mensagens são os dados que são lidos e escritos. Quando é feita uma publicação ou se subscreve em um tópico, é necessário especificar qual o tipo da mensagem utilizada, ou seja, qual tipo de dados que serão lidos ou escritos. Um comando relacionado com as mensagens é:

   - rosmsg - Informa todos os tipos de mensagem

4. **Subscribers** e **Publishers**

   Um ou mais *publishers* enviam mensagens para um certo tópico, por outro lado, um *Subscribers* serve para receber mensagens desse tópico.

5. Catkin

   É o diretório aonde será armazenado todos os pacotes criados pelos nós.

4 Conceitos de sistemas robóticos cooperativos

Após realizar um estudo sobre a modelagem e o controle de robôs seriais no capítulo anterior, esses conceitos serão ampliados para robôs cooperativos. Desse modo, ao invés de ser considerado que se tenha uma cadeia cinemática aberta, agora será considerado que existe uma cadeia cinemática fechada. Com isso, é possível chegar as equações da cinemática direta por mais de um caminho. Dessa forma, um sistema robótico cooperativo, em geral, é redundante, ou seja, possui mais graus de liberdade do que o necessário para a realização de uma certa tarefa. Essa característica faz com que esse tipo de sistema possua mais destreza, mais força e que não chegue aos limites das juntas tão facilmente. A seguir, é possível verificar uma ilustração, na figura 9, de braços robóticos cooperativos:

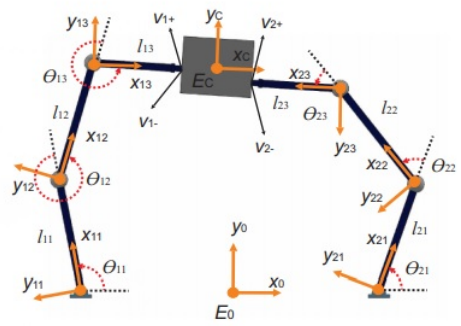

Figura 9: Exemplo de um sistema robótico cooperativo

a Modelagem de sistemas robóticos cooperativo

Sistemas robóticos cooperativos, requerem mais de um braço robótico, portanto ao invés de serem aplicados os conceitos que foram vistos na seção acima, em cada um dos manipuladores, é proposta uma técnica que mapeia a modelagem de todos os braços robóticos, utilizando uma abordagem mais centralizada. Nesse contexto, as variáveis do sistema são mapeadas em função de movimentos absolutos e relativos. Com isso, considera-se a seguinte imagem vista na figura 10:

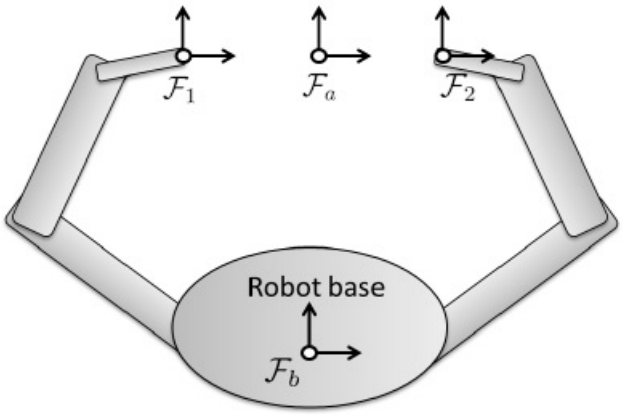

Figura 10: Sistemas de coordenadas de dois efetuadores e da base

Dessa maneira, na ilustração acima, existem dois manipuladores robóticos, onde os vetores

$x_1 = [p_1, \phi_1]$ e $x_2 = [p_2, \phi_2]$ representam as coordenadas dos efetuadores da esquerda e da direita em relação ao sistema de coordenadas da base F_b. O sistema de coordenadas absoluto F_a é posicionado em:

$$p_a = \frac{p_1 + p_2}{2}, \tag{37}$$

Aonde $p_a \in \mathbb{R}^{3\times1}$ é o vetor da posição absoluta expresso em relação ao sistema de coordenadas da base. O significado físico do sistemas de coordenadas F_a, é o ponto médio da posição dos efetuadores. Logo, caso os manipuladores estejam transportando uma carga, F_a será o sistema de coordenadas do centro de massa do objeto. Contudo, as variáveis absolutas não são suficientes para especificar o movimento dos robôs cooperativos, visto que, infinitas configurações dos efetuadores, chegam no mesmo vetor de posição p_a. Portanto, também serão utilizadas as variáveis relativas para descrever o movimento do sistema. A posição relativa é definida por:

$$p_r = p_2 - p_1, \tag{38}$$

Aonde $p_r \in \mathbb{R}^{3\times1}$ é calculada em relação ao sistema de coordenadas da base. Nesse contexto, é importante visualizar que a posição relativa desejada, é expressa no sistema de coordenadas absoluto F_a. Dessa forma, a posição relativa p_r é expressa em relação a base e p_r^d é expresso em relação as coordenadas absolutas. Com isso, é necessário fazer uma transformação de coordenadas para que p_r^d seja escrito em relação ao sistema de coordenadas da base. Essa variáveis podem ser verificadas através da figura 11:

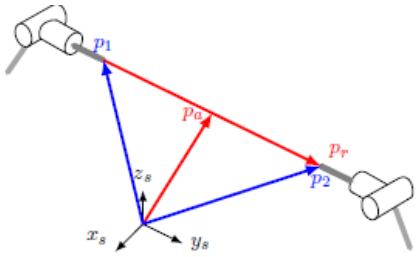

Figura 11: Imagem dos Vetores das posições absoluta e relativa e das posições dos efetuadores

Com isso, para encontrar os vetores das posições absolutas e relativas, primeiro são calculados os vetores das posições dos efetuadores através das equações da cinemática direta, vistas na seção anterior. Depois de obter esses vetores, utiliza-se as expressões (21) e (22) para obter as posições absolutas e relativas dos manipuladores. Assume-se que as orientações absolutas e relativas são:

$$\phi_a = \frac{\phi_1 + \phi_2}{2}, \tag{39}$$

$$\phi_r = \phi_2 - \phi_1, \tag{40}$$

Essa aproximação só é válida para a abordagem que será detalhada na próxima subseção. Desta forma, essas equações serão explicadas de forma mais clara posteriormente.

b Symmetric Formulation

Nessa subseção será considerado que o sistema robótico, composto por dois braços, está manipulando uma carga. Dessa forma, e técnica chamada *Symmetric Formulation* é utilizada para mapear esse tipo de problema. Nessa abordagem, é proposto um modelo usando *virtual sticks*, que são vetores de posição $r_i \in \mathbb{R}^3$. Com isso, considerando um sistemas de coordenadas F_c, fixado no objeto, o *virtual stick* é um elo virtual que conecta o sistema de coordenadas F_i do i-nésimo efetuador com o sistema de coordenadas F_c. Desse modo, a cinemática direta do sistema será determinada considerando os *virtual sticks* como parte do manipulador. O sistema de coordenadas de *stick* $F_{S,i}$ é o mesmo que o sistema de coordenadas do objeto F_c . A pose de cada *virtual stick* é dada por:

$$p_{S,1} = p_{S,2} = p_{S,c} \tag{41}$$

Um exemplo da utilização desse conceito é visto na figura 12 a seguir:

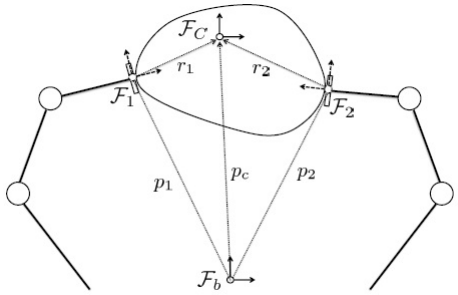

Figura 12: Exemplo de *Virtual Sticks*

Dessa maneira, considerando os *sticks* como um efetuador virtual, é possível utilizar as equações (21) e (22) da subseção anterior, para calcular as variáveis absolutas e relativas do sistema:

$$p_a = \frac{p_{S,1} + p_{S,2}}{2}, \tag{42}$$

$$p_r = p_{S,1} - p_{S,2}, \tag{43}$$

As orientações dos *virtual sticks* são dadas pelas equações (23) e (24). Essa aproximação só pode ser considerada se o deslocamento entre os sistemas de coordenadas dos *sticks* são pequenos. Caso o deslocamento entre os dois sistemas de coordenadas seja grande, essa aproximação não representa de forma fiel as orientações absolutas e relativas.

Essa abordagem de *Symmetric Formulation* só é válida caso não haja deformação do objeto, pois a configuração dos *virtual sticks* depende diretamente da pose do objeto. Dessa maneira, assume-se que o objeto e os manipuladores são muito rígidos e que o objeto está sendo apertado pelos efetuadores de tal forma, que não haja deslocamento entre os *end-effectors* e os pontos de contatos podem se tornar insignificantes.

c Cinemática Diferencial

Após estudar a cinemática direta dos robôs cooperativos na subseção anterior, agora será feita a análise de como relacionar as velocidades de cada manipulador serial, com as velocidades das variáveis do sistema. Dessa forma, serão encontrados os Jacobianos absoluto e relativo, que associam as variáveis das juntas dos dois braços robóticos, com as velocidades dos movimentos absolutos e relativos do sistema. Essas matrizes serão importantes para a realização do controle cinemático dos braços robóticos nas próximas análises. Essas matrizes são definidas da seguinte maneira:

$$J_a = \begin{bmatrix} 0.5J_1 & 0.5J_2 \end{bmatrix}, \tag{44}$$

$$J_r = \begin{bmatrix} -J_1 & J_2 \end{bmatrix}, \tag{45}$$

Aonde J_1 e J_2 são os Jacobianos dos manipuladores esquerdo e direito respectivamente. Com isso, a matrizes Jacobianas J_a e Jr definem as velocidades absolutas e relativas com as velocidades dos ângulos das juntas. Ambas possuem dimensões de $\mathbb{R}^{6\times 2n}$. Portanto, o Jacobiano total do sistema cooperativo é dado por:

$$J = \begin{bmatrix} J_a \\ J_r \end{bmatrix} \tag{46}$$

O Jacobiano total do sistema possui dimensões de $\mathbb{R}^{12\times 2n}$. Dessa maneira, por meio dele, é possível relacionar velocidades relativas e absolutas com a velocidade dos ângulos das juntas. Essa relação é definida da seguinte forma:

$$\begin{bmatrix} V_a \\ V_r \end{bmatrix} = J \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \end{bmatrix}, \tag{47}$$

Logo, os vetores $\dot{q_1}$ e $\dot{q_2}$ contém as velocidades das juntas dos manipuladores da esquerda e da direita respectivamente. Aliado a isso, o vetores v_a e v_r definem as velocidades absolutas e relativas do sistema e possuem dimensões $\mathbb{R}^{6\times 1}$ cada um.

d Controle Cinemático

Nessa subseção será discutida uma estratégia de controle cinemático, que tem como objetivo fazer com que os manipuladores sigam uma trajetória desejada carregando um objeto. Portanto, a abordagem de controle utilizada na seção anterior, será ampliada para que seja utilizada com a *Symmetric Formulation*. Com isso, a entrada do sistema é dada por:

$$u \approx \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \end{bmatrix}, \tag{48}$$

Desse modo, a entrada do sistema $u \in \mathbb{R}^{2n\times 1}$. O sistema de controle para robôs cooperativos é dado por:

$$\begin{bmatrix} v_a \\ v_r \end{bmatrix} \approx Ju, \tag{49}$$

O objetivo do controle é:

$$x_a \to x_{ad}, t \to \infty, \tag{50}$$

$$x_r \to 0, t \to \infty, \tag{51}$$

Logo, as equações dos erros absoluto e relativo são dadas por:

$$e_a = x_{ad} - x_a, \tag{52}$$

$$e_r = -x_r, \tag{53}$$

Com isso, a lei de controle proposta é:

$$u = J^{\dagger} \begin{bmatrix} \dot{x_{ad}} + K_a e_a \\ K_r e_r \end{bmatrix}, \tag{54}$$

Os vetores de erro $e_a \in \mathbb{R}^{6\times1}$ e $e_r \in \mathbb{R}^{6\times1}$. Aliado a isso, as matrizes de ganho escolhidas $K_a \in \mathbb{R}^{6\times6}$ e $K_r \in \mathbb{R}^{6\times6}$. Portanto, utilizando essa lei de controle, tem-se que:

$$\begin{bmatrix} \dot{e_a} + K_a e_a \\ K_r e_r \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{55}$$

Desse modo, além de utilizar o controlador feedforward + proporcional proposto na seção anterior, para controlar a pose absoluta dos manipuladores, utiliza-se também, controle proporcional para garantir que a pose relativa tenda a zero. Assim, o sistema robótico conseguirá seguir a trajetória desejada, manipulando um objeto. A seguir, é possível visualizar o diagrama de blocos de controle, na figura 13:

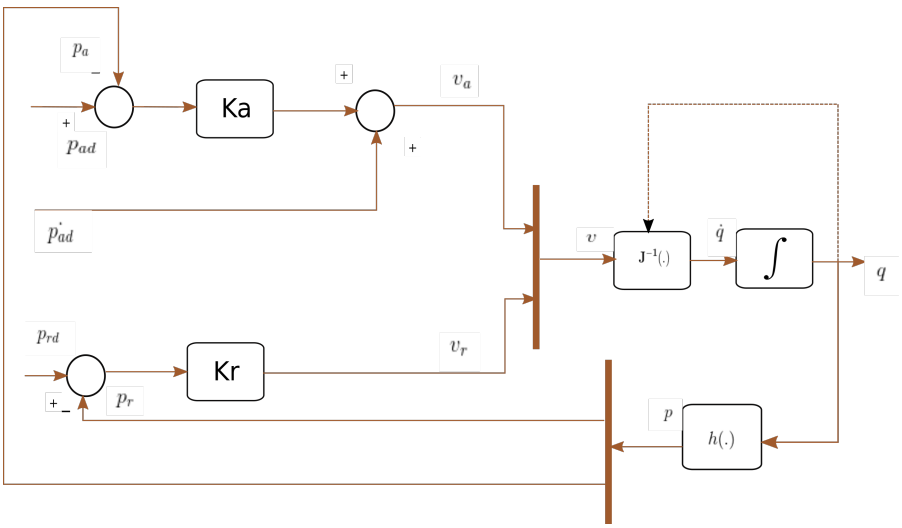

Figura 13: Diagrama de blocos para o controle cinemático de robôs cooperativos

5 PhantomX Pincher

Para a realização das simulações no Matlab/Simulink e dos ensaios experimentais da parte teórica, foram utilizados dois manipuladores PhantomX Pincher Arm. Portanto, nas próximas subseções, será feita uma melhor descrição desse tipo de braço robótico, como suas especificações técnicas, os atributos dos motores utilizados, as características dos *encoders* para a realização da leitura dos ângulos das juntas, e as equações da cinemática. Logo em seguida, essas equações serão estendidas para o sistema cooperativo com dois PhantomX Pincher. Desse modo, será possível ter uma melhor percepção sobre suas características que serão importantes para as próximas seções das simulações e testes experimentais.

a Descrição

O robô PhantomX Pincher AX-12 é um manipulador robótico que pertence a Trossen Robotics. Ele possui 5 graus de liberdade, porém, 1 grau de liberdade é relacionado ao *gripper*, então para uma perspectiva cinemática, ele possui 4 graus de liberdade. Com isso, por ele possuir menos de 6 graus de liberdade, significa que a ferramenta acoplada ao efetuador, não consegue rotacionar em todas as direções. O kit PhantomX Pincher Arm, que contém esses manipuladores, possui um microcontrolador ArbotiX que é similar a um Arduíno. No kit também possui suportes de montagem para câmeras e sensores. O robô também é equipado com servomotores da Dynamixel AX-12 Servos que possui 300 graus de movimento. Esses servomotores também possuem *encoders* que conseguem medir ângulos de -150 graus até 150 graus. O robô PhantomX Pincher também é facilmente adicionado a plataforma ROS, utilizando o pacote TurtleBot. Na figura 14 é possível ver uma figura do robô PhantomX Pincher.

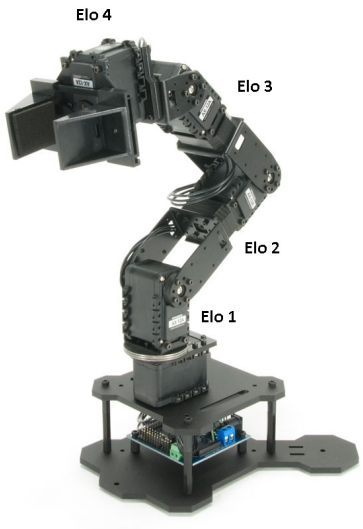

Figura 14: PhantomX Pincher Arm

Agora serão apresentados alguns parâmetros mecânicos importantes do robô. Desse jeito, foi montada uma tabela que apresenta algumas especificações do manipulador robótico:

Após a apresentação dos parâmetros mecânicos do manipulador, outros dados importantes, são os dos atuadores do robô. Os motores contidos no braço robótico são mecanismos rápidos e poderosos

Peso	550 g
Alcance Vertical	35 cm
Alcance Horizontal	30 cm
Capacidade da Garra	500 g
Capacidade de elevação em 15 cm	100 g
Capacidade de elevação em 20 cm	70 g
Capacidade de elevação em 25 cm	40 g

que possuem 300 graus de movimento. Esses servomotores possuem um microcontrolador embutido, que realiza a o controle de posição angular. Eles também só possuem somente 3 fios para conexão, que faz com que seja reduzido bastante a fiação do robô. A seguir, na figura 15, é possível ver uma imagem do motor:

Figura 15: Dynamixel AX-12A

As principais características desses motores são:

Tensão de Operação	12 V
Torque de travamento	15.3 Kg.cm
Velocidade sem carga	59 RPM
Peso	55 g
Dimensões	32 x 50 x 40 mm
Resolução	0.29º
Temperatura de operação interna	-5ºC a 70º C
Taxa de redução	1/254
Protocolo de Operação	TTL Half Duplex Async Serial
Ângulo de Operação	300º
Corrente Máxima	900 mA
Corrente de Standby	50 mA
Velocidade de comunicação	7343bps ~ 1Mbps
Feedback de posição	Possui
Feedback de temperatura	Possui
Feedback de tensão de entrada	Possui
PID	Possui
Material	Plástico na carenagem e engrenagens
Motor	Cored Motor

No kit que contém o PhantomX Pincher, também contém um microcontrolador ArbotiX, que é similar a um microcontrolador Arduino ATMEGA644P de 16 MHz, como foi dito acima. Esse controlador

pode ser programado em uma linguagem “wiring” e em um ambiente de programação que roda no Windows, Linux e no MacOS. Para se comunicar a placa, utiliza-se uma interface USB que é similar a uma porta serial para o computador conectado. O ArbotiX possui 64k de memória ROM e uma memória RAM de 4k. Essas placas são alimentadas por uma tensão de 12 VDC. A seguir, é possível encontrar uma imagem, na figura 16, da placa Arbotix:

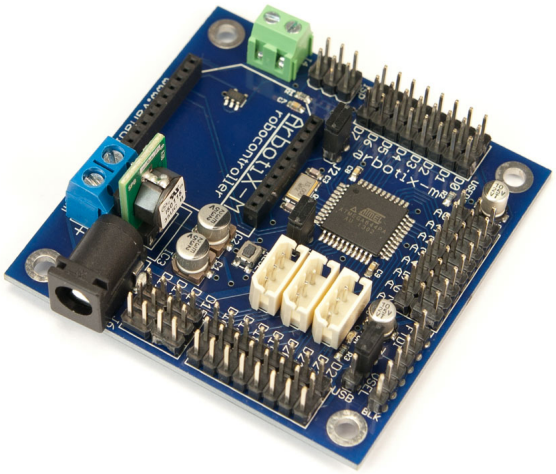

Figura 16: Arbotix

Como foi visto anteriormente, o robô utilizado, PhantomX Pincher Arm possui 4 graus de liberdade. Para isso, ele é composto por 4 juntas rotativas que conectam os elos do braço robótico. Essas juntas são:

Junta	Nome
1	Shoulder Pan
2	Shoulder Lift
3	Elbow
4	Wrist

Visualizando a figura acima, do manipulador robótico PhantomX Pincher, é possível ter uma noção de onde está cada junta. Dessa forma, a junta 1 é a que conecta o elo 1 com a base, e a junta 2 é a que se encontrar logo acima, conectando o elo 1 com o elo 2. Os movimentos que essas juntas realizam no robô, podem ser comparados ao do ombro. Em sequência, tem a junta 3, que pode ser associada ao cotovelo humano. Essa junta conecta o elo 2 com o elo 3. Por último, aparece a junta 4, que também pode ser associada ao corpo humano, como o punho, ligando o elo 3 com o elo 4. O comprimento dos elos do braço robótico serão muito importantes para o cálculo das equações da cinemática direta e diferencial dele. Na tabela a seguir, é possível visualizar os tamanhos de cada um desses elos:

Elo	Comprimento (mm)
1	40
2	105
3	105
4	105

Portanto, na próxima subseção, serão encontradas as equações da modelagem cinemática do PhantomX Pincher.

b Modelagem Cinemática

Para realizar o cálculo da cinemática direta do PhantomX Pincher, primeiro é preciso definir o sistema de coordenadas de cada junta. Portanto, utiliza-se a convenção de Denavit-Hartenberg para encontrar os eixos de todas as juntas. Dessa maneira, na figura 17, é possível ver a configuração do robô:

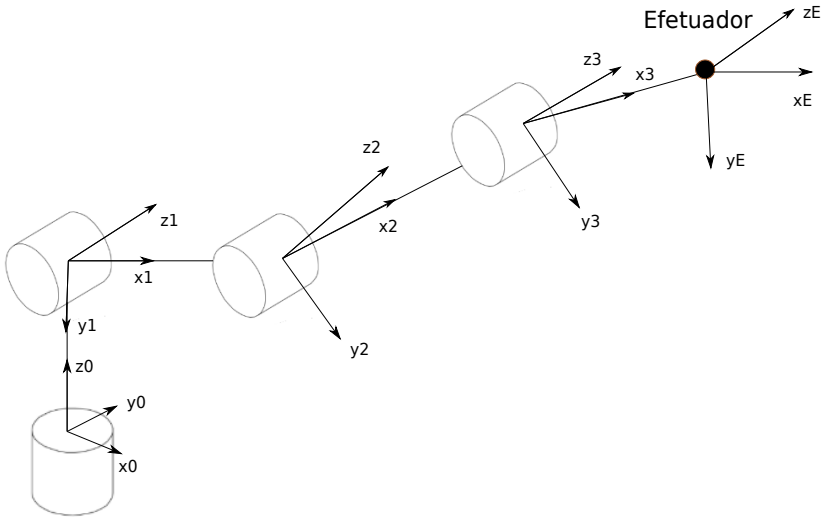

Figura 17: Sistema de Coordenadas das Juntas

Agora, é possível achar os parâmetros de Denavit-Hartenberg.

Elo	d_i	a_i	α_i	θ_i
1	40	0	$-pi/2$	θ_1
2	0	105	0	$\theta_2 - pi/2$
3	0	105	0	θ_3
4	0	105	0	θ_4

O manipulador PhantomX Pincher não possui a rotação da sua segunda junta de acordo com o conceito visto na subseção de Denavit-Hartenberg. Essa particularidade é devido a montagem do seu mecanismo. Com isso, ao invés de utilizar a variável θ_2, que seria o ângulo entre x_1 e x_0, o manipulador utiliza o complemento dessa variável. Logo, é feita uma adaptação nos parâmetros de Denavit-Hartenberg, e será usado $\theta_2 - pi/2$ para θ_i do segundo elo.

Agora, utilizando os parâmetros da tabela acima, é possível achar a matriz de transformação do efetuador, para o sistema de coordenadas da base. Contudo, primeiramente é necessário achar a matriz de transformação de cada sistema de coordenadas. Para isso, utiliza-se a matriz (5). Deste modo as matrizes são:

$$A_1^0 = \begin{bmatrix} cos(\theta_1) & 0 & -sen(\theta_1) & 0 \\ sen(\theta_1) & 0 & cos(\theta_1) & 0 \\ 0 & -1 & 0 & 40 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{56}$$

$$A_2^1 = \begin{bmatrix} sen(\theta_2) & cos(\theta_2) & 0 & 105sen(\theta_2) \\ -cos(\theta_2) & sen(\theta_2) & 0 & -105cos(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{57}$$

$$A_3^2 = \begin{bmatrix} cos(\theta_3) & -sen(\theta_3) & 0 & 105cos(\theta_3) \\ sen(\theta_3) & cos(\theta_3) & 0 & 105sen(\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{58}$$

$$A_4^3 = \begin{bmatrix} cos(\theta_4) & -sen(\theta_4) & 0 & 105cos(\theta_4) \\ sen(\theta_4) & cos(\theta_4) & 0 & 105sen(\theta_4) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{59}$$

Após encontrar as matrizes acima, é preciso realizar a seguinte operação para calcular a matriz de transformação entre o *end-effector* e a base do manipulador:

$$A_4^0 = A_1^0 A_2^1 A_3^2 A_4^3, \tag{60}$$

Será assumido que:

$$c_i = cos(\theta_i) \tag{61}$$

,

$$s_i = sen(\theta_i), \tag{62}$$

$$sij = sen(\theta_i + \theta_j), \tag{63}$$

$$cij = cos(\theta_i + \theta_j), \tag{64}$$

Logo, a matriz encontrada é:

$$A_4^0 = \begin{bmatrix} c_1 s_{234} & c_1 c_{234} & -s_1 & 105c_1(s_{234} + s_{23} + s_2) \\ s_1 s_{234} & s_1 c_{234} & c_1 & 105s_1(s_{234} + s_{23} + s_2) \\ c_{234} & -s_{234} & 0 & 105(c_{234} + c_{23} + c_2) + 40 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{65}$$

Utilizando a quarta coluna de A_4^0, é possível encontrar a posição do efetuador em relação ao sistema de coordenadas da base do manipulador. Desse modo, é obtido o seguinte vetor:

$$p_e^0 = \begin{bmatrix} x_e^0 \\ y_e^0 \\ z_e^0 \end{bmatrix} \begin{bmatrix} 105c_1(s_{234} + s_{23} + s_2) \\ 105s_1(s_{234} + s_{23} + s_2) \\ 105(c_{234} + c_{23} + c_2) + 40 \end{bmatrix}, \tag{66}$$

Contudo, nesse projeto é considerado que o manipulador realize movimento somente no plano xz. Portanto, é feita uma simplificação no modelo cinemático, e assim admite-se que $\theta_1 = 0$. Dessa forma, utilizando esse dado e a equação (51), o vetor de posição do efetuador será:

$$p_e^0 = \begin{bmatrix} x_e^0 \\ y_e^0 \\ z_e^0 \end{bmatrix} \begin{bmatrix} 105(s_{234} + s_{23} + s_2) \\ 0 \\ 105(c_{234} + c_{23} + c_2) + 40 \end{bmatrix}, \tag{67}$$

Porém, já que o movimento é no plano xz, serão utilizadas somente as coordenadas x e z. Logo, o novo vetor de posição será:

$$p_e^0 = \begin{bmatrix} x_e^0 \\ z_e^0 \end{bmatrix} \begin{bmatrix} 105(s_{234} + s_{23} + s_2) \\ 105(c_{234} + c_{23} + c_2) + 40 \end{bmatrix}, \tag{68}$$

A seguir, na figura 18, é possível visualizar uma imagem 2D do PhantomX Pincher no plano xz:

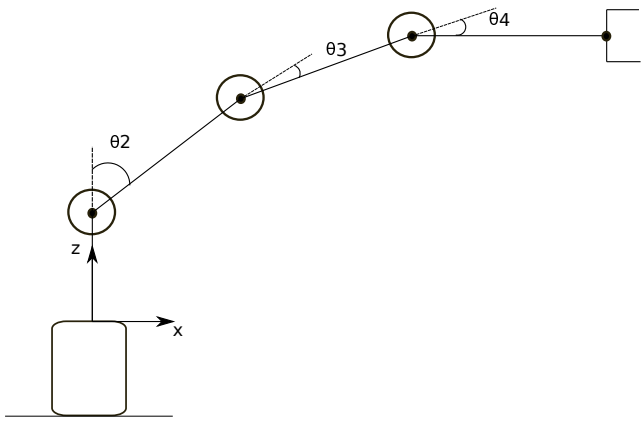

Figura 18: PhantomX Pincher 2D

Com isso, para calcular a orientação do efetuador utiliza-se a expressão:

$$\alpha_e = \theta_2 + \theta 3 + \theta 4, \tag{69}$$

A pose do efetuador é dada por:

$$\begin{bmatrix} p_e^0 \\ \alpha_e \end{bmatrix} = \begin{bmatrix} 105(s_{234} + s_{23} + s_2) \\ 105(c_{234} + c_{23} + c_2) + 40 \\ \theta_2 + \theta 3 + \theta 4 \end{bmatrix}, \tag{70}$$

Agora, para calcular o Jacobiano do sistema, utiliza-se as equações (12) e (13). Portanto, tem-se que:

$$J = \begin{bmatrix} 105(c_{234} + c_{23} + c_2) & 105(c_{234} + c_{23}) & 105c_{234} \\ -105(s_{234} + s_{23} + s_2) & -105(s_{234} + s_{23}) & -105s_{234} \\ 1 & 1 & 1 \end{bmatrix}, \tag{71}$$

Desso modo, encontra-se as equações da cinemática direta e da cinemática diferencial do PhantomX Pincher Arm. Na próxima subseção será abordado como foi feito o arranjo de dois PhantomX Pincher para a construção do sistema robótico cooperativo.

c Sistema Robótico Cooperativo com PhantomX Pincher

Nessa subseção será abordado como foi feito a montagem do sistema cooperativo com dois manipuladores PhantomX Pincher Arm. Depois serão calculadas as equações que descrevem o modelo cinemático em função das variáveis absolutas e relativas. A seguir, na figura 19, é apresentado a configuração dos braços robóticos:

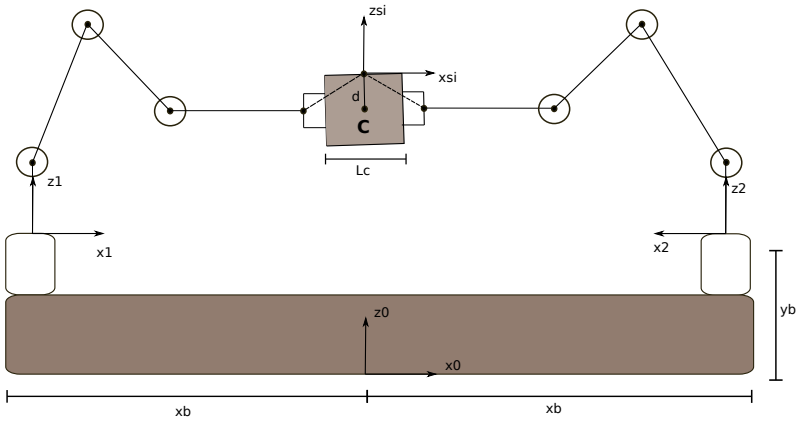

Figura 19: Sistema Robótico Cooperativo com PhantomX Pincher

O vetor de coordenadas da base dos manipuladores no sistema de coordenadas E_0 é dado por:

$$E_0 = \begin{bmatrix} E_1^0 \\ E_2^0 \end{bmatrix} \begin{bmatrix} x_1^0 \\ z_1^0 \\ x_2^0 \\ z_2^0 \end{bmatrix} = \begin{bmatrix} -x_b \\ y_b \\ x_b \\ y_b \end{bmatrix}, \tag{72}$$

A posição dos *end-effectors* em relação ao sistema de coordenadas da base de cada manipulador é dado por:

$$p_e = \begin{bmatrix} p_{e_1}^1 \\ p_{e_2}^2 \end{bmatrix} = \begin{bmatrix} 105(s_{234} + s_{23} + s_2) \\ 105(c_{234} + c_{23} + c_2) + 40 \\ 105(s_{678} + s_{67} + s_6) \\ 105(c_{678} + c_{67} + c_6) + 40 \end{bmatrix}, \tag{73}$$

Contudo, agora é preciso alterar o sistema as coordenadas do vetor acima para o sistema E_0. Com isso, é necessário fazer o seguinte cálculo:

$$p_e^0 = p_e + E_0 = \begin{bmatrix} p_{e_1}^0 \\ p_{e_2}^0 \end{bmatrix} = \begin{bmatrix} 105(s_{234} + s_{23} + s_2) - x_b \\ 105(c_{234} + c_{23} + c_2) + y_b \\ 105(s_{678} + s_{67} + s_2) + x_b \\ 105(c_{678} + c_{67} + c_6) + y_b \end{bmatrix}, \tag{74}$$

Dessa maneira, os ângulos θ_2,θ_3 e θ_4 são referentes as juntas do manipulador da esquerda e os ângulos θ_6,θ_7 e θ_8 são referentes as juntas do outro manipulador. Portanto, é possível perceber que as variáveis θ_1 e θ_5 não foram utilizadas, pois são relacionadas com as juntas *Shoulder Pan* de cada PhantomX Pincher, e por isso são sempre iguais a 0.

Desse modo, de forma similar como foi feito na equação (54), as orientações dos manipuladores serão dadas por:

$$\phi_{e_1} = \theta_2 + \theta_3 + \theta_4, \tag{75}$$

$$\phi_{e_2} = \theta_6 + \theta_7 + \theta_8, \tag{76}$$

Contudo, ao invés de ser considerado os efetuadores como a extremidade dos braços robóticos, o sistema cooperativo utilizará os *Virtual Sticks* como foi abordado acima. Logo, é preciso calcular a posição de cada *sticks* em relação ao sistema de coordenadas da base E_0. Para isso, primeiro é necessário descrever as características do objeto. Dessa forma, o objeto que será manipulado pelos robôs, possui lados com tamanho L_c e assume-se que ele é um quadrado. Consequentemente, as posições dos *Virtual Sticks* em relação ao sistema de coordenadas dos efetuadores são dadas por:

$$r_{s,1} = \begin{bmatrix} x_{s,1} \\ z_{s,1} \end{bmatrix} = \begin{bmatrix} Lc/2 \\ d \end{bmatrix}, \tag{77}$$

$$r_{s,2} = \begin{bmatrix} x_{s,2} \\ z_{s,2} \end{bmatrix} = \begin{bmatrix} -Lc/2 \\ d \end{bmatrix}, \tag{78}$$

A orientação dos *sticks* serão dadas por $\theta_{s,1}$ e $\theta_{s,2}$. Logo, para passar as coordenadas desse vetor para o sistema E_0, é feito o seguinte cálculo:

$$r_{s,1}^0 = R_y(\phi e_1) R_y(\theta_{s,1}) r_{s,1}, \tag{79}$$

$$r_{s,2}^0 = R_y(\phi_{e_2}) R_y(\theta_{s,2}) r_{s,2}, \tag{80}$$

Sendo R_y a matriz de rotação em y. É possível utilizar essa abordagem para realizar a rotação de um sistema de coordenadas para outro, pois é assumido que os manipuladores são planares, e então eles só possuirão rotação em uma direção. Após essa alteração de coordenada, é necessário fazer o acoplamento dos *Virtual Sticks* com os braços robóticos. Para isso, é feito:

$$p_{s,1} = r_{s,1}^0 + p_{e_1}^0, \tag{81}$$

$$p_{s,2} = r^0_{s,2} + p^0_{e_2}, \tag{82}$$

Agora, serão calculadas as matrizes Jacobianas de cada manipulador. Para isso, será utilizada a equação (56):

$$J_{m1} = \begin{bmatrix} 105(c_{234} + c_{23} + c_2) & 105(c_{234} + c_{23}) & 105c_{234} \\ -105(s_{234} + s_{23} + s_2) & -105(s_{234} + s_{23}) & -105s_{234} \\ 1 & 1 & 1 \end{bmatrix}, \tag{83}$$

$$J_{m2} = \begin{bmatrix} 105(c_{678} + c_{67} + c_6) & 105(c_{678} + c_{67}) & 105c_{678} \\ -105(s_{678} + s_{67} + s_6) & -105(s_{678} + s_{67}) & -105s_{678} \\ 1 & 1 & 1 \end{bmatrix}, \tag{84}$$

Logo em seguida, utilizando as equações (29),(30),(68),(69) as matrizes Jacobianas absoluta e relativa são dadas por:

$$J_a = 0.5 \begin{bmatrix} J_{m1} & J_{m2} \end{bmatrix}, \tag{85}$$

$$J_r = \begin{bmatrix} -J_{m1} & J_{m2} \end{bmatrix}, \tag{86}$$

Para calcular o Jacobiano total, é só utilizar a equação (31):

$$J = \begin{bmatrix} J_a \\ J_r \end{bmatrix}, \tag{87}$$

6 Simulação Numérica

Nessa sessão será discutido sobre a simulação numérica realizada no Matlab/Simulink. Desse modo, primeiramente serão definidas as condições iniciais do sistema, e em seguida serão escolhidos os parâmetros da simulação. Com isso, definidas essas variáveis, serão apresentados os resultados da simulação do algoritmo implementado e então, serão feitas as análises dos gráficos obtidos .

Logo, foi feita a simulação dos 2 métodos de integração vistos na seção de conceitos fundamentais, que são:

- Euler avançado
- Tustin

Também foram simuladas 3 leis de controle diferentes:

- Proporcional
- Proporcial + Integral
- Feedforward + Integral

a Condições Iniciais

Nessa parte da seção, serão conhecidas as condições inicias que foram determinadas e também como foi feita essa escolha. O primeiro parâmetro a ser estabelecido, foi o tamanho do objeto, que é dado por:

$$L_c = 50mm, \tag{88}$$

Com isso, a distância entre os efetuadores deve ser igual ao comprimento do objeto, como pode ser visto na figura 19.

Em seguida, foi feita uma análise para escolher os ângulos iniciais das juntas. Para essa escolha, foi investigada a manipulabilidade dos robôs, para que eles não fiquem próximos de alguma região de singularidade. Utilizando esse pretexto, o vetor de ângulos iniciais ficaram da seguinte forma:

$$q_0 = \begin{bmatrix} 0 & 0.595 & 1.76 & -0.785 & 0 & -0.595 & -1.76 & 0.785 \end{bmatrix}^T, \tag{89}$$

Sendo q_0 composto por:

$$q_0 = \begin{bmatrix} q_1 & q_2 \end{bmatrix}, \tag{90}$$

Dessa forma, q_1 são os ângulos das juntas do manipulador esquerdo e q_2 os ângulos das juntas do manipulador direito.

Utilizando a equação (15), é possível obter a manipulabilidade de cada braço robótico:

$$w_1 = w_2 = 5.11219 \times 10^6, \tag{91}$$

Portanto, percebe-se que as manipulabilidades são muito maiores que zero, então os robôs se encontram bem distantes das regiões singulares.

Agora, utilizando a equação (71), é possível obter as posições, em milímetro, dos *end-effectors* em relação a base cada robô, que são dadas por:

$$p_e = \begin{bmatrix} 238.2 & 52.88 & -238.2 & 52.88 \end{bmatrix}^T, \tag{92}$$

Logo, visualizando a geometria da figura 19, calcula-se o valor x_b:

$$x_b = 238.2 + 25 = 263.2mm, \tag{93}$$

A altura da base do sistema foi arbitrada de acordo com o sistema robótico real. Com isso, o valor da variável y_b é:

$$y_b = 122mm, \tag{94}$$

Desse modo, o vetor E_0 da equação (70) é dado por:

$$E_0 = \begin{bmatrix} -263.2 \\ 122 \\ 263.2 \\ 122 \end{bmatrix}, \tag{95}$$

Desse modo, fazendo a mudança de coordenadas dos efetuadores de acordo com o vetor (72), para a o sistema de coordenadas E_0, tem se que:

$$p_e^0 = \begin{bmatrix} -25 \\ 134,88 \\ 25 \\ 134,88 \end{bmatrix}, \tag{96}$$

Para calcular a orientação inicial dos efetuadores, utiliza-se as expressões (73) e (74):

$$\phi_{e_1} = 0.595 + 1.76 - 0.785 = 1.57rad, \tag{97}$$

$$\phi_{e_2} = -0.595 - 1.76 + 0.785 = -1.57rad, \tag{98}$$

Dessa maneira, acha-se as posições iniciais de cada efetuador em relação ao sistema de coordenadas do sistema e também a orientação inicial dos manipuladores. Essas informações serão importantes para o próximo passo, que é o cálculo dos valores dos *virtual sticks*.

A seguir, na figura 20, é possível visualizar uma imagem no Matlab do sistema robótico, após encontrar as condições acima:

A posição inicial do objeto é dada pelo seguinte vetor:

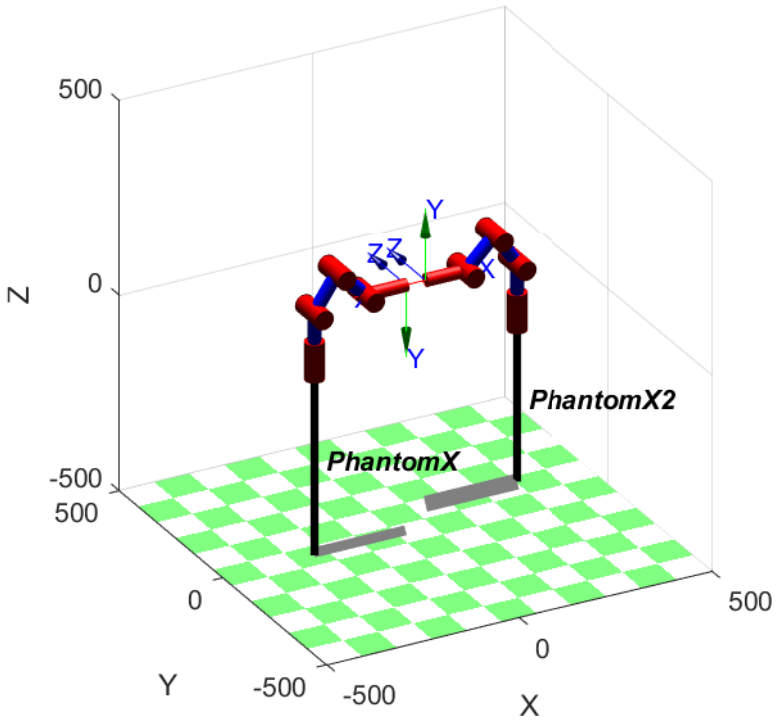

Figura 20: Sistema Robótico Cooperativo no Matlab

$$p_a^0 = \begin{bmatrix} 0 \\ 134,88 \end{bmatrix}, \tag{99}$$

Agora, serão calculados os parâmetros do *virtual sticks*. Desse modo, deseja-se que o centro de movimento seja na parte superior do objeto. Com isso, o valor de d nas equações (75) e (76) é a distância entre o lado mais elevado e o centro de massa, que é dada pela metade do comprimento do objeto, visto que ele é um quadrado. Utilizando as expressões mencionadas, tem se que:

$$r_{s,1} = \begin{bmatrix} 25 \\ 25 \end{bmatrix}, \tag{100}$$

$$r_{s,2} = \begin{bmatrix} -25 \\ 25 \end{bmatrix}, \tag{101}$$

Logo, o vetor do centro de movimento é dado por:

$$M = \begin{bmatrix} 0 \\ 159.86 \end{bmatrix}, \tag{102}$$

As orientações iniciais dos *virtual sticks* são encontradas por:

$$\theta_{s,1} = arctg(\frac{r_{s,1}(2)}{r_{s,1}(1)}) = -0.785rad, \tag{103}$$

$$\theta_{s,2} = arctg(\frac{r_{s,2}(2)}{r_{s,1}(2)}) = 2.35rad, \tag{104}$$

Para realizar o acoplamento dos *sticks* com os manipuladores, utiliza-se as equações (79) e (78). Desse modo, Acha-se os seguintes vetores:

$$p_{s,1} = \begin{bmatrix} 0 \\ 159.86 \end{bmatrix}, \tag{105}$$

$$p_{s,2} = \begin{bmatrix} 0 \\ 159.86 \end{bmatrix}, \tag{106}$$

b Parâmetros de Simulação

Nessa subseção serão apresentados alguns parâmetros importantes para a simulação. Os primeiros parâmetros a serem discutidos, serão os de tempo, que são:

- Tempo de amostragem.
- Frequência de amostragem.
- Tempo de simulação.

O tempo de amostragem do ROS é de 0.03s. Contudo, a simulação deve ser mais rápida que o ROS, para adquirir os dados dos sensores ou enviar comandos para o motor da forma correta. Desse modo, foi escolhido um valor 3 vezes menor para que isso ocorra. Portanto, o tempo de amostragem h é dado por:

$$h = 0.01s, \tag{107}$$

A frequência de amostragem é dada pelo inverso do período de amostragem:

$$f = 100hz, \tag{108}$$

O tempo de simulação foi escolhido empiricamente, da maneira que o sistema robótico se estabilizar e conseguir seguir a trajetória diversas vezes. Com isso, o valor escolhido foi:

$$T_{max} = 40s, \tag{109}$$

Após fazer a escolha desses valores, será feita uma análise dos parâmetros da trajetória, que será um círculo. Logo, precisa-se definir os seguintes valores:

- Frequência do sinal
- Raio da trajetória

A frequência do sinal escolhida deve ser muito menor que a frequência de amostragem, para não ocorrer problemas de *aliasing*. Além disso, também deve respeitar a limitação física do manipulador. Com isso, a frequência de Nyquist é dada por:

$$f_{Nyquist} = \frac{f}{2} = 50hz, \tag{110}$$

Logo, a frequência do sinal deve ser menor que a frequência de Nyquist. Para satisfazer essa condição foi feita a escolha do seguinte valor:

$$f_{sinal} = 0.05hz, \tag{111}$$

O período do sinal é:

$$T_{sinal} = 20s, \tag{112}$$

Assim, o sistema robótico completará cada volta do círculo em 20 segundos. Com isso, quando for feito os testes experimentais, será possível visualizar o sistema seguindo a trajetória perfeitamente.

Agora, será feita a escolhida do raio da trajetória. Esse valor não pode ser muito grande, pois se não os manipuladores iram chegar ao limite do seu *workspace*, e consequentemente ficarão singulares. Com isso, olhando as características do PhantomX Pincher, ele possui alcance vertical de 35 cm e o horizontal de 30 cm. Dessa maneira, devido as condições iniciais escolhidas, o braço já está alongado aproximadamente 24 cm na horizontal e 5 cm na vertical. Portanto o raio máximo da trajetória deve ser menor que 6 cm. Contudo, será escolhido um valor muito menor, para ter garantia que não haverá singularidade. Sendo assim, o raio escolhido foi:

$$R = 2.5cm, \tag{113}$$

Desse modo, o vetor posição da trajetória desejada é:

$$p_{ad}(t) = \begin{bmatrix} 25sen(\frac{\pi}{10}t) \\ 25cos(\frac{\pi}{10}t) + 159.86 \end{bmatrix}, \tag{114}$$

Pela expressão acima, é visto que o sistema robótico irá circular ao redor do centro de movimento calculado na expressão (98).

Deseja-se que os *sticks* não alterem a sua orientação, e sempre permaneça com o mesmo ângulo calculado nas equações (101) e (102). Contudo, é necessário calcular as orientações deles em relação ao sistema de coordenadas comum dos manipuladores. Dessa forma, os ângulos desejados são:

$$\phi^0_{s,1} = \phi_{e_1} + \theta_{s,1} = 0.785rad, \tag{115}$$

$$\phi^0_{s,2} = \phi_{e_2} + \theta_{s,2} = 2,36rad, \tag{116}$$

Utilizando a equação (37), o ângulo absoluto que deseja-se calcular é:

$$\phi_{ad} = 1.57rad, \tag{117}$$

Dessa maneira, o vetor da pose da trajetória desejada é:

$$x_{ad}(t) = \begin{bmatrix} 25sen(\frac{\pi}{10}t) \\ 25cos(\frac{\pi}{10}t) + 159.86 \\ 1.57 \end{bmatrix}, \tag{118}$$

Entretanto, além da pose da trajetória desejada, é necessário calcular a sua derivada no tempo, para o controle feedforward + proporcional:

$$\frac{dx_{ad}(t)}{dt} = \begin{bmatrix} \frac{25\pi}{10}cos(\frac{\pi}{10}t) \\ -\frac{25\pi}{10}sin(\frac{\pi}{10}t) \\ 0 \end{bmatrix}, \tag{119}$$

A seguir, na figura 21, é possível visualizar uma figura da trajetória desejada:

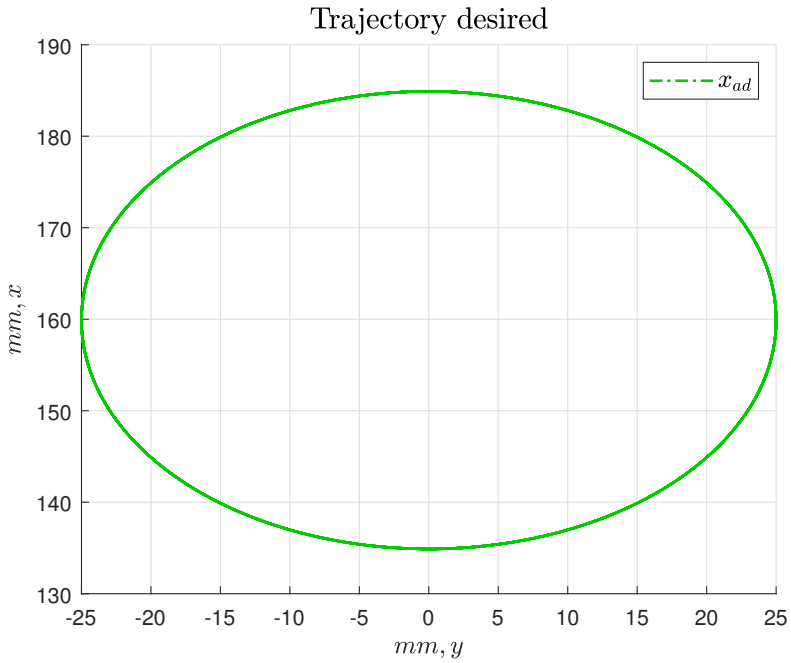

Figura 21: Trajetória desejada

Essa é a referência para o controle da pose absoluta, porém, também é necessário fazer a mesma análise para a pose relativa. Entretanto, deseja-se somente que os efetuadores permaneçam na mesma posição e orientação relativa durante a execução da tarefa. Portanto, o controle relativo deve manter a pose relativa inicial durante toda a trajetória. Com isso, utilizando a equação (36) e os vetores (103) e (104), tem-se o seguinte resultado para a posição relativa desejada:

$$p_{rd} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{120}$$

Para o cálculo da orientação relativa, utiliza-se a equação (38):

$$\phi_{rd} = -1.57rad, \tag{121}$$

Consequentemente, o vetor da pose relativa desejada é dado por:

$$p_{rd} = \begin{bmatrix} 0 \\ 0 \\ -1.57 \end{bmatrix}, \tag{122}$$

Após as escolhas dos parâmetros de simulação e de sinal, será feita a análise dos ganhos de controle. Como foi dito acima, os controles utilizados são P, PI e feedforward+P para a pose absoluta. Para controlar a pose relativa, um ganho relativo já é suficiente. Desse modo os ganhos que precisam ser calibrados são:

- Ganho proporcional absoluto
- Ganho proporcional relativo
- Ganho Integral

Os ganhos foram escolhidos de maneira empírica. Logo, eles foram aumentados até chegar a um bom resultado. Desse modo, os ganhos proporcionais foram incrementados até não ter mudanças significativas nos resultados, e também foi levado em conta a saturação dos atuadores. De acordo com a tabela de características do motor, a sua velocidade máxima sem carga, é de $6.2rad/s$. Aliado a isso, o ganho proporcional foi aumentado até obter os melhores resultados, e não houver problemas de *widup* ou *chattering*. Com isso, os valores escolhidos foram:

Ganhos	Valores
K_a	$10s^{-1}$
K_r	$25s^{-1}$
K_i	$1s^{-1}$

Desse modo, a velocidade máxima dos atuadores, com esses ganhos proporcionais foi igual a $2rad/s$. Contudo, é preciso que os motores aguentem o objeto e também o peso da sua própria estrutura. Desse modo, não é recomendado que aumente os valores desses ganhos. O ganho integral foi aumentado até obter bons resultados e não apresentar problemas de *chattering*. Caso ele for maior que $1s^{-1}$, o robô começa a vibrar muito.

c Resultados da simulação

Após a escolha das condições iniciais e dos parâmetros de simulação, agora serão apresentados os resultados das simulações realizadas. Desse modo, será feita uma análise dos gráficos apresentados para obter o melhor controlador e o melhor método de integração.

1. **Euler avançado com controlador proporcional**

   O primeiro teste a ser implementado, utilizou o método de Euler avançado para a integração numérica e um controlador proporcional para controlar a pose absoluta. Logo, os resultados obtidos

foram:

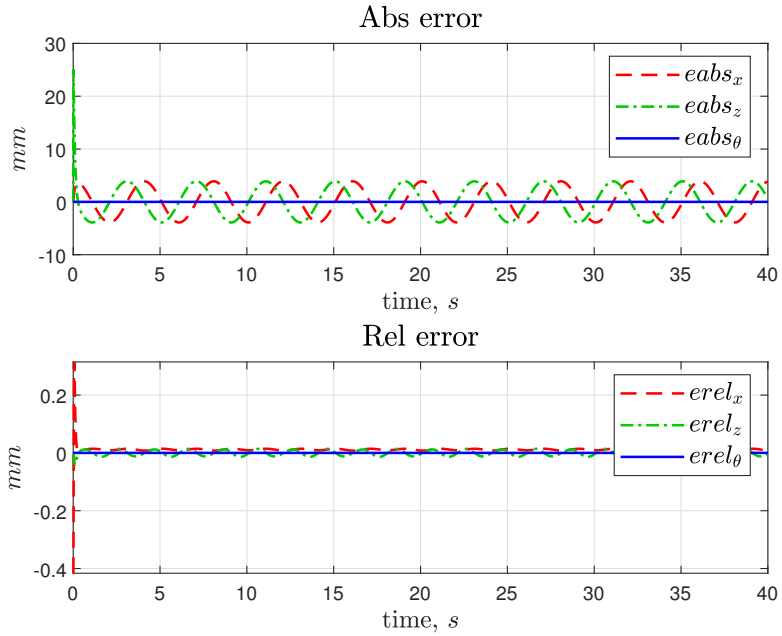

Figura 22: Gráfico da coordenada x das posições absoluta e relativa para Euler avançado + P

Análise: Na figura 22 acima, é possível observar que a erro da pose absoluta é estável. Contudo, como foi discutido anteriormente, existe um erro de regime permanente, devido a d_{xad}. Por outro lado, o erro da pose relativa tende a zero no regime permanente, pois só um controlador proporcional é necessário para controlá-la.

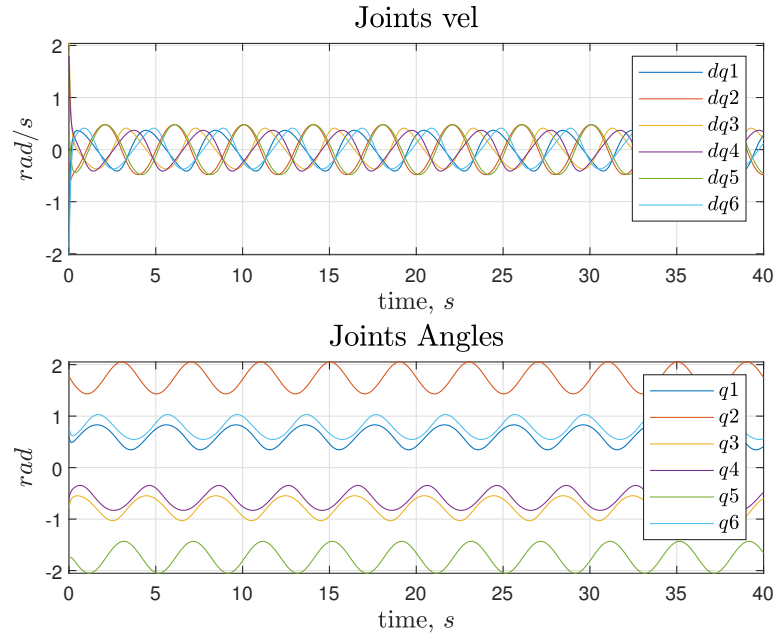

Figura 23: Gráfico dos ângulos das juntas e das velocidades das juntas para Euler avançado + P

Análise: Na figura 23, observa-se que as velocidades das juntas estão dentro do valor desejado, que era de no máximo $2rad/s$. Com isso, esse controlador não faz com que haja saturação dos motores. Também percebe-se que os ângulos das juntas estão estáveis e não alcançam valores

absurdos.

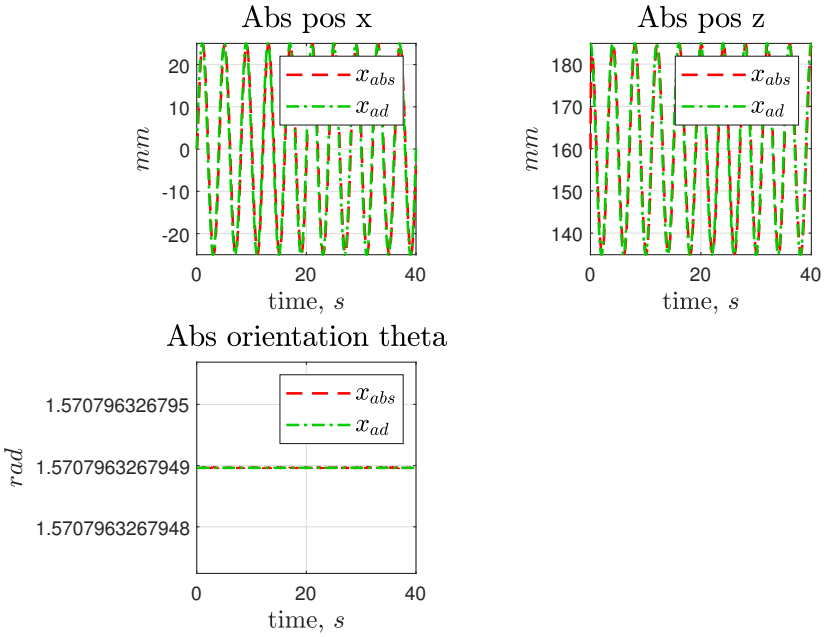

Figura 24: Gráfico da pose absoluta para Euler avançado + P

Análise: A posição absoluta, como visto na figura 24, apresenta o formato de onda esperado, que é de um seno e um cosseno, como visto na equação (116). Contudo, é possível observar que o sistema não segue fielmente a trajetória desejada, pois as amplitudes dos gráficos não são exatamente iguais. Entretanto, a orientação fica no valor esperado, calculado na equação (115).

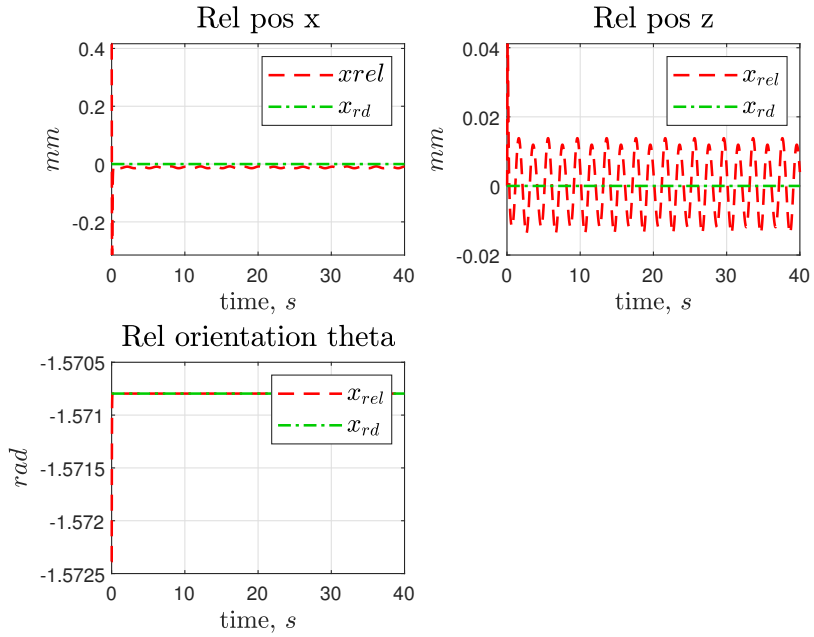

Figura 25: Gráfico da pose relativa para Euler avançado + P

Análise: A posição relativa, como vista na figura 25, segue o comportamento esperado, apesar do pequeno ruído em relação a coordenada z. A orientação relativa também tende a um valor esperado, calculado na expressão (120).

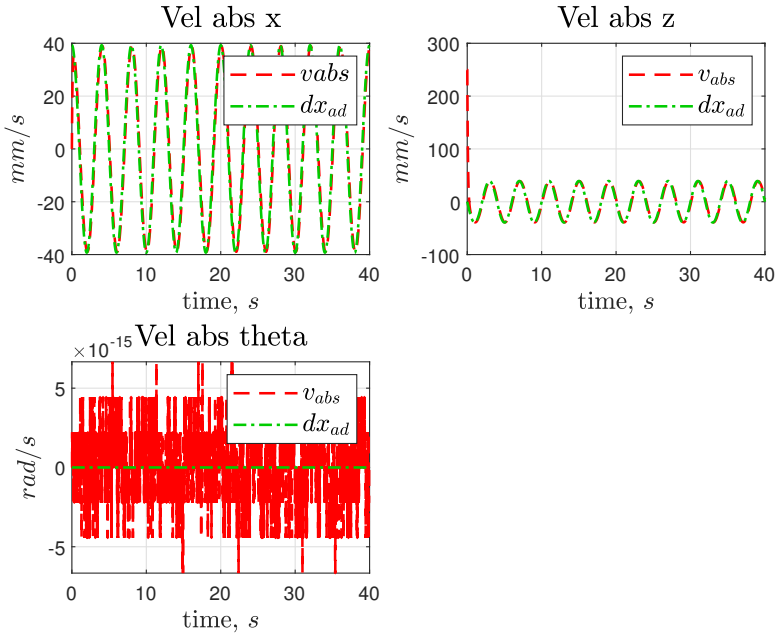

Figura 26: Gráfico da velocidade absoluta para Euler avançado + P

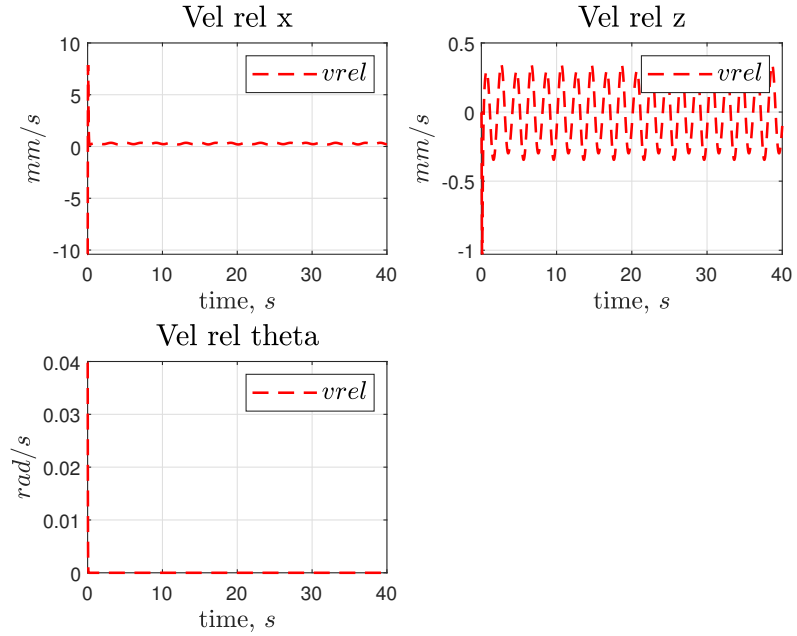

Figura 27: Gráfico da velocidade relativa para Euler avançado + P

Análise: A velocidade absoluta, como visto na figura 26, tem o gráfico que era espero, como foi visto na equação (117), e apesar do ruído na velocidade da orientação absoluta, ela também tende a um valor esperado, que é zero. A velocidade relativa, como a pose relativa deve tender a zero no regime permanente. Desse modo, pelos gráficos, percebe-se que essas velocidades tendem a esse valor desejado, apesar do ruído. Portanto, as velocidades relativas e absolutas tendem ao valor desejado.

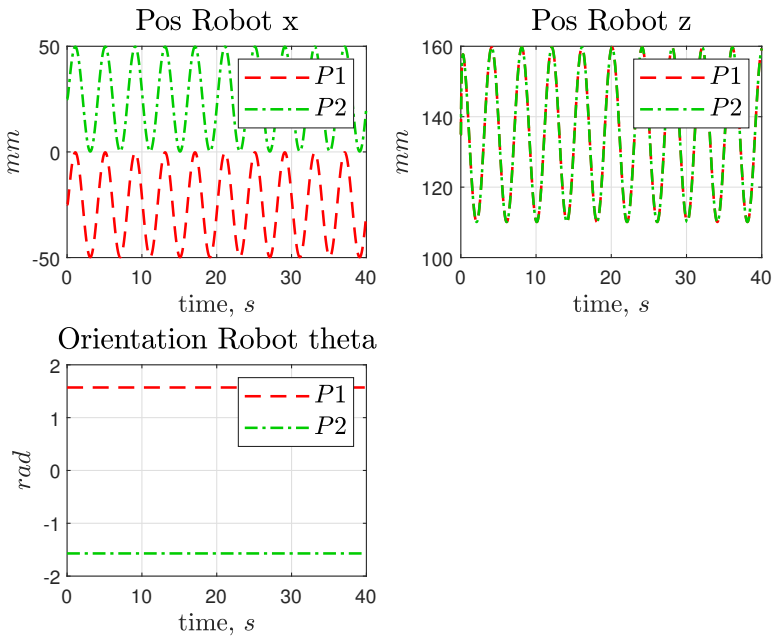

Figura 28: Gráfico da pose dos efetuadores para Euler avançado + P

Análise: Como o raio da trajetória circular é de $25cm$, a distância que os efetuadores percorrem nas coordenadas x e z, é o valor do diâmetro da trajetória, como visto na figura 28. As orientações dos *end-effectors* eram esperadas, como foi visto nas equações (95) e (96).

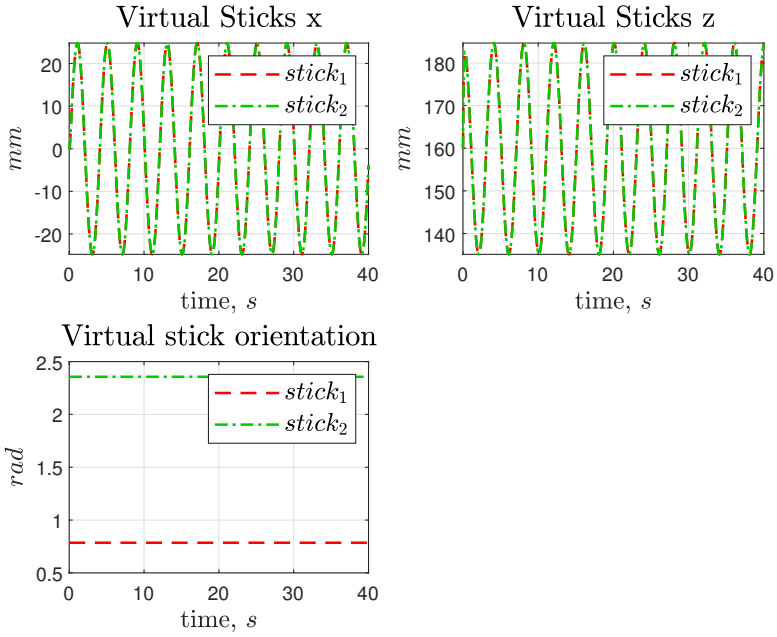

Figura 29: Gráfico da pose dos *Virtual Sticks* para Euler avançado + P

Analise: Como as posições dos *End-Effectors*, as distâncias percorridas pelos *Virtual Sticks* devem ser iguais ao diâmetro da trajetória, que é de $50mm$ como pode ser visto na figura 29. A coordenada x, varia entre $-25mm$ e $25mm$, pois o centro de movimento para essa coordenada é o valor zero. Seguindo esse raciocínio, a coordenada z varia de aproximadamente $135mm$ a 185, pois o centro de movimento é em $159.86mm$. As orientações dos *virtual sticks* seguem os valores

que eram esperados. Esses valores foram calculados nas expressões (101) e (102).

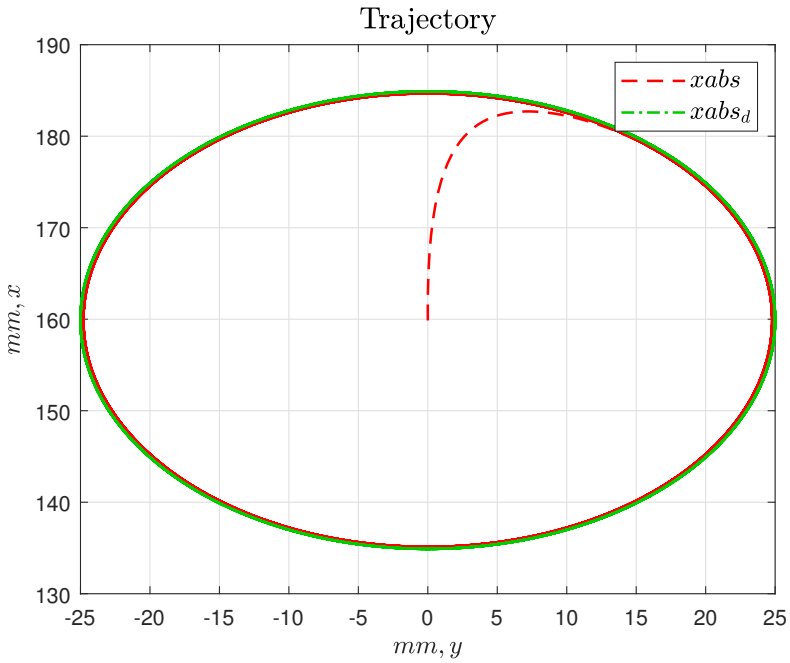

Figura 30: Gráfico da trajetória para Euler avançado + P

Análise: Apesar do sistema cooperativo seguir uma trajetória circular, ele apresenta um pequeno erro se comparado a trajetória desejada, como visto na figura 30. Esse comportamento era esperado, como foi visto nas análises anteriores.

2. **Tustin com controlador proporcional**

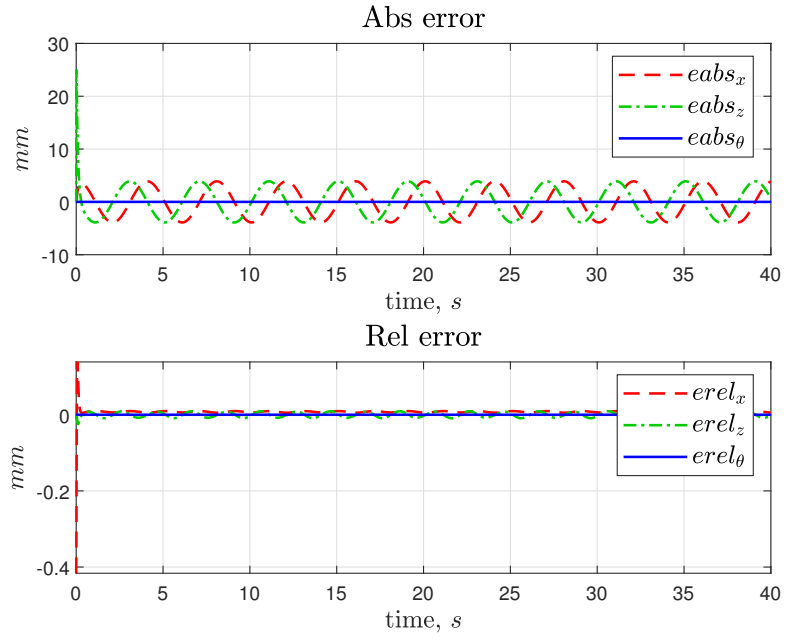

Figura 31: Gráfico do erro para Tustin + P

Análise: Observando a figura 31, é possível notar que não há grandes mudanças se for comparado ao método de Euler. Contudo, percebe-se que o *overshoot* nos dois gráficos é menor, se for

comparado ao método de integração anterior. Essa vantagem é notória no gráfico do erro relativo, já que para Euler, ele chegava a valores de $0.2mm$ e no Tustin ele não ultrapassa o $0mm$.

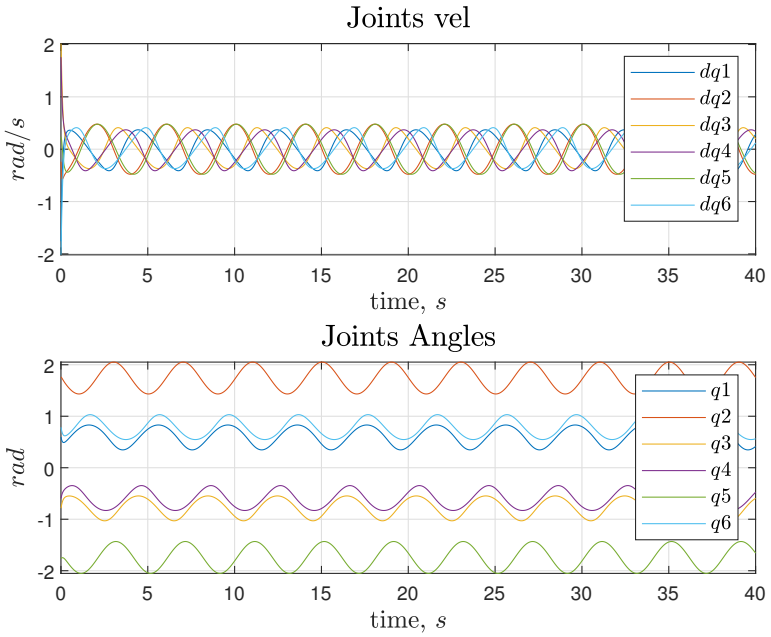

Figura 32: Gráfico dos ângulos das juntas e das velocidades das juntas para Tustin + P

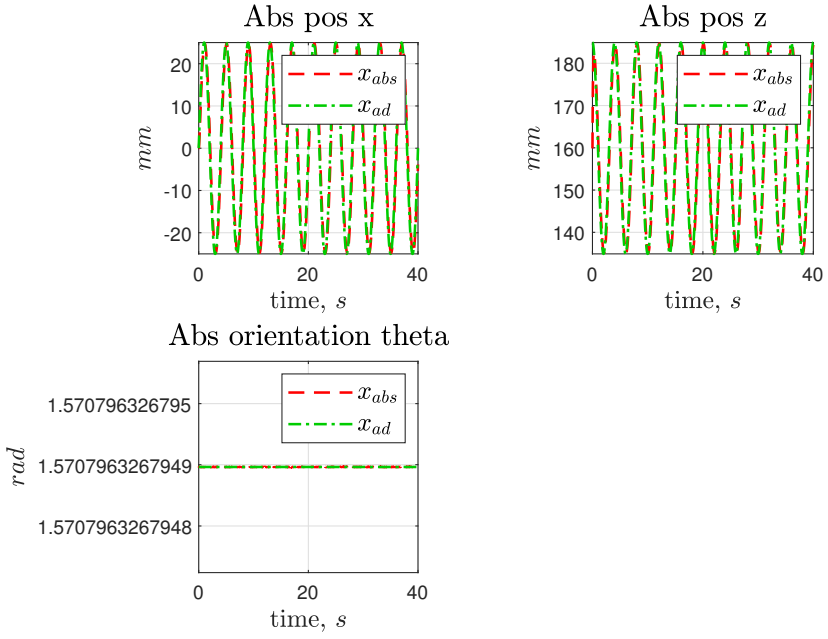

Figura 33: Gráfico da pose absoluta para Tustin + P

Análise: Como visto na figura 32, não há nenhuma diferença notória para os ângulos das juntas e para as velocidades das juntas se for feita a comparação entre os dois métodos. Também não há diferença entre a pose absoluta do sistema robótico, se for comparado os métodos, observando a figura 33. Desse modo, é possível concluir que para essas variáveis, os métodos de Tustin e de Euler não provocam grandes mudanças. Desse modo, tanto nos gráficos da figura 33, tanto os gráficos da figura 24, seguem o comportamento previsto nas equações (115) e (116).

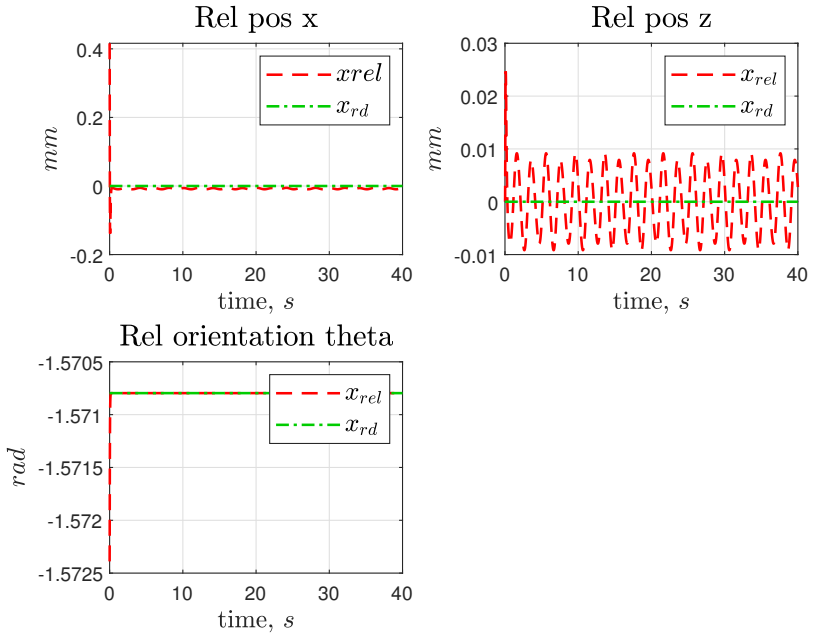

Figura 34: Gráfico da pose relativa para Tustin + P

Análise: Com a implementação do método de Tustin, percebe-se que teve uma atenuação do *overshoot* e do *undershoot* da pose relativa, como visto na figura 34. A diminuição do *overshoot*, fica clara no gráfico da coordenada z, onde no método de Euler, o valor dessa coordenada chega a $0.04mm$ e para Tustin, o z alcança valores entre $0.02mm$ e $0.03mm$. A atenuação do *undershoot*, pode ser vista na coordenada x, onde para o método de Euler, o valor de x chega a $0.02mm$ e para o método de Tustin, essa coordenada chega a um valor menor, como pode se visualizado no gráfico.

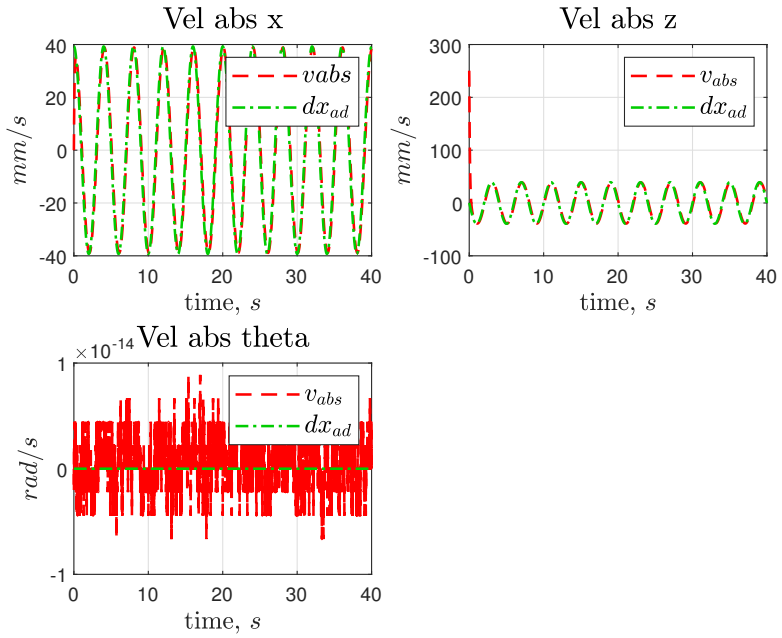

Figura 35: Gráfico da velocidade absoluta para Tustin + P

Análise: Não há alterações significativas na velocidade absoluta, como visto no gráfico 35, se for

comparado os dois métodos, como ocorre para a pose absoluta. A única grande mudança vista, é no aumento do ruído na orientação. É possível visualizar que para o método de Euler, o ruído está na ordem de 10^{-15} e para o método de Tustin, o ruído está na ordem de 10^{-14}.

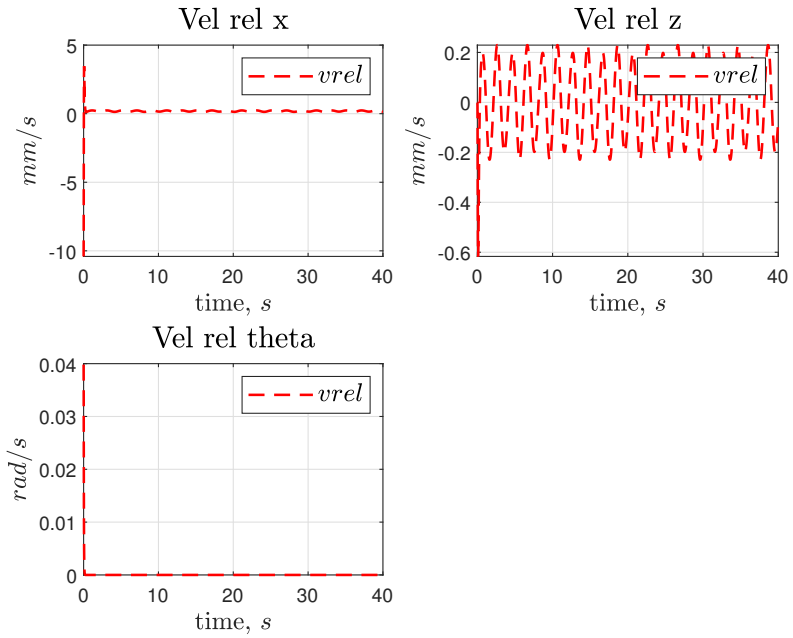

Figura 36: Gráfico da velocidade relativa para Tustin + P

Análise: Para a velocidade relativa, o método de Tustin apresentou uma melhora, se for comparado o método de Euler, observando a figura 36. É possível perceber isso, visualizando os gráficos das coordenadas x e z, onde as amplitudes dos gráficos diminuíram, junto com o *Overshoot*e o *Undershoot*.

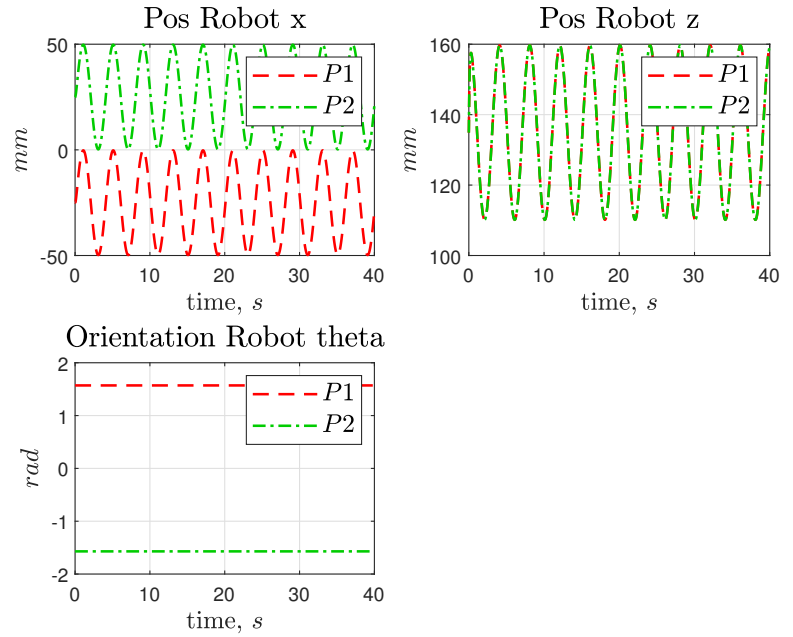

Figura 37: Gráfico da pose dos efetuadores para Tustin + P

Análise: Não houveram alterações significativas nos gráficos das figuras 37 e 38 se for feita a

comparação entre os métodos. Desse modo, os dois métodos de integração fazem com que os gráficos das poses dos *end-effectors* e dos *Virtual sticks* sigam o comportamento esperado, visto nas equações acima.

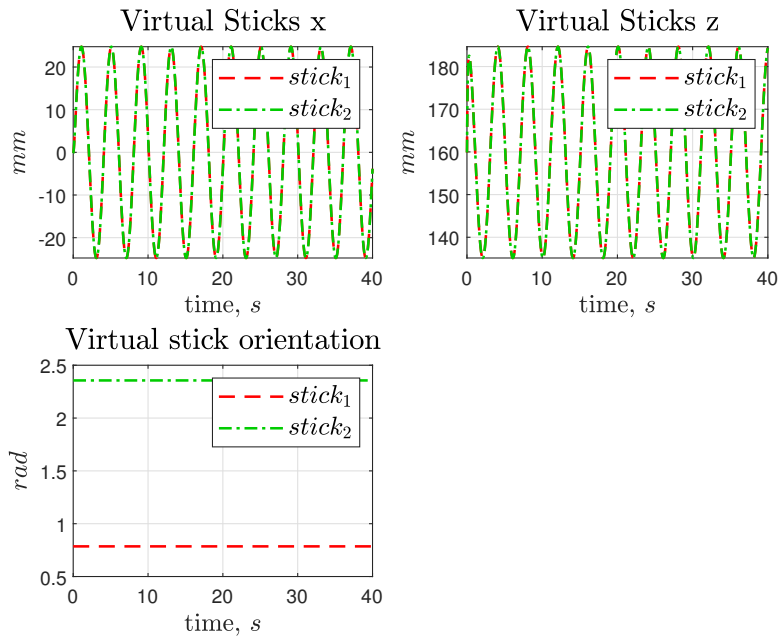

Figura 38: Gráficos da pose dos *Virtual Sticks*

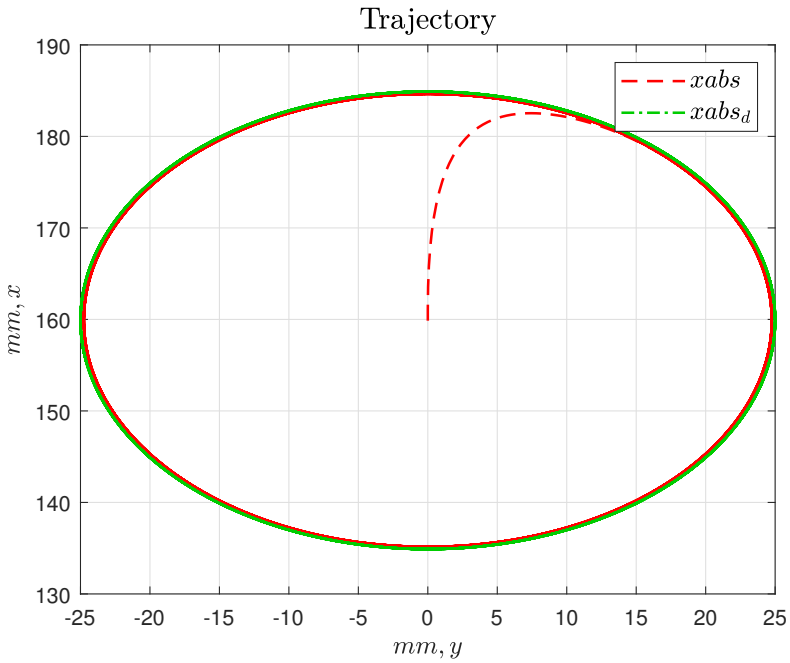

Figura 39: Gráfico da trajetória para Tustin + P

Análise: Não houveram grandes alterações nos gráficos da trajetória percorrida entre os dois métodos. Contudo, visualizando as figuras 30 e 39, percebe-se que no método de Tustin, o círculo realizado pelo sistema robótico, possui um raio levemente menor e apresenta uma suavização no movimento, se for feita a comparação com o método de Euler e Tustin. Com isso, através de todas as análises feitas, relacionando os dois métodos, o de Tustin geralmente serve para

atenuar problemas de *Overshoot* e *Undershoot* e também diminuir a amplitude do sinal de algumas variáveis.

Após realizar as análises dos métodos de integração para o controlador P, agora serão feitas as análises para o controlador PI.

3. **Euler avançado com controlador PI**

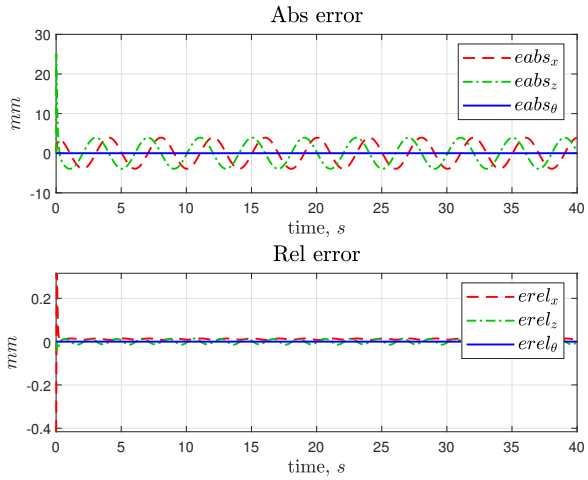

Figura 40: Gráfico do erro para Euler + PI

Análise: Se for comparado o gráfico de erro visto para o controlador P com o visto para o controlador PI não são notadas muitas diferenças, como pode ser observado na figura 40. Contudo, se for verificado o gráfico da figura 48, da trajetória, percebe-se que o controlador PI teve um melhor desempenho.

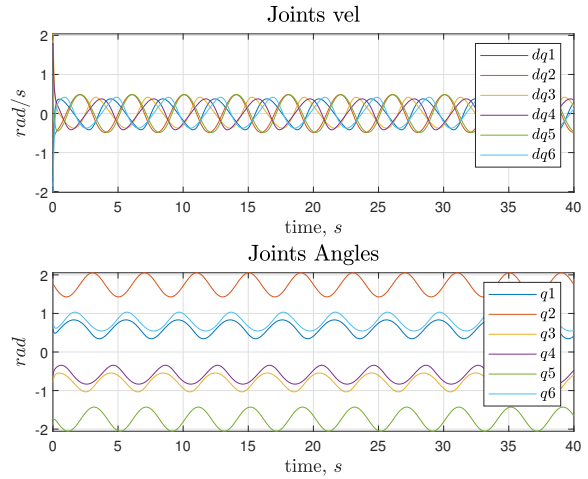

Figura 41: Gráfico dos ângulos das juntas e das velocidades das juntas para Euler + PI

Análise: As velocidades das juntas estão dentro do valor desejado, como visto na figura 41, que é de no máximo $2rad/s$, igual o controlador P. Desse modo, não há saturação do motor, devido ao controlador integral. Os ângulos das juntas também não atingem valores muito grandes, e possuem gráficos estáveis, significando que o manipulador não tornou-se incontrolável ou instável.

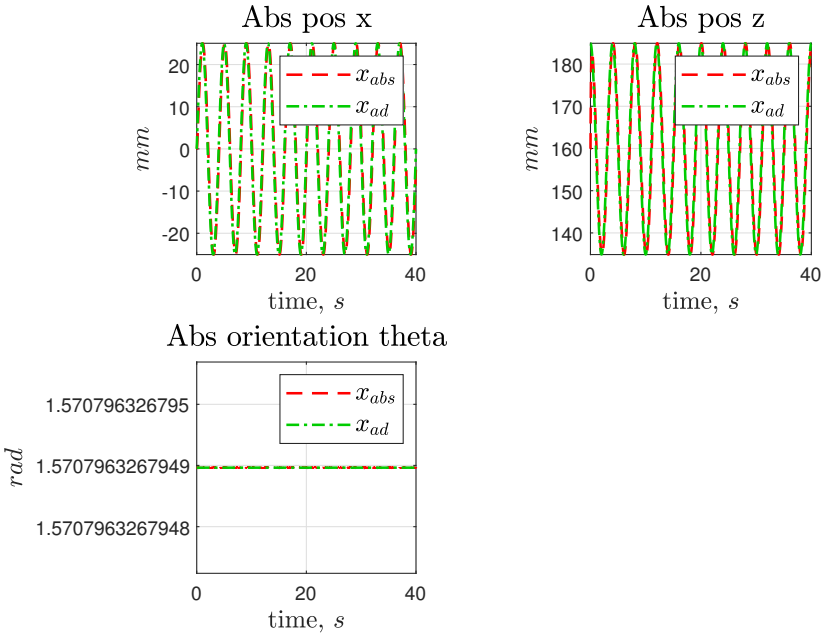

Figura 42: Gráfico da pose absoluta para Euler + PI

Análise: Não há diferença visual entre os gráficos visualizados para o controlador P e para o controlador PI, como visto na figura 42. Isso é devido ao controlador integral realizar um ajuste fino para ajudar o robô a seguir mais fielmente a trajetória desejada. Desse modo, não é possível notar diferença visual, porém, pelo gráfico da figura 48, da trajetória , percebe-se que o controlador PI obteve um resultado melhor.

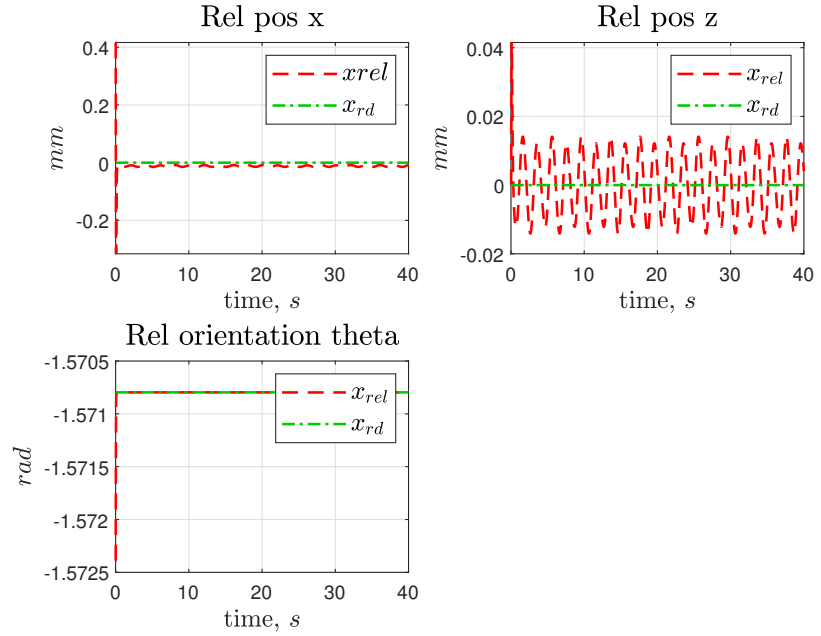

Figura 43: Gráfico da pose relativa para Euler + PI

Análise: Já que a única alteração feita, foi no controlador para a pose absoluta, o comportamento visto na figura 43, para a pose relativa, se mantém, igual ao da figura 38. Desse modo, uma forma de melhorar o resultado, como foi visto nas análise de Tustin+P, é utilizado o método de integração

de Tustin. Contudo, os resultados são esperados como foi visto nas equações.

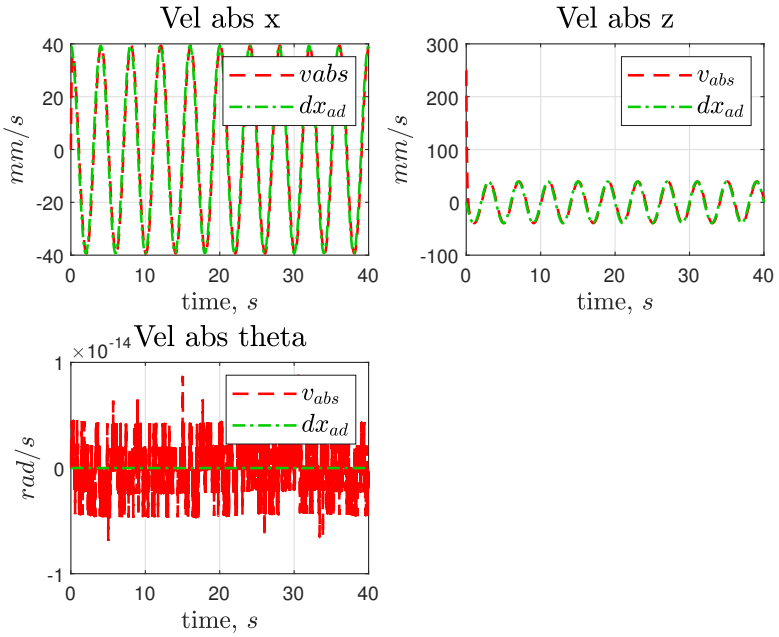

Figura 44: Gráfico da velocidade absoluta para Euler + PI

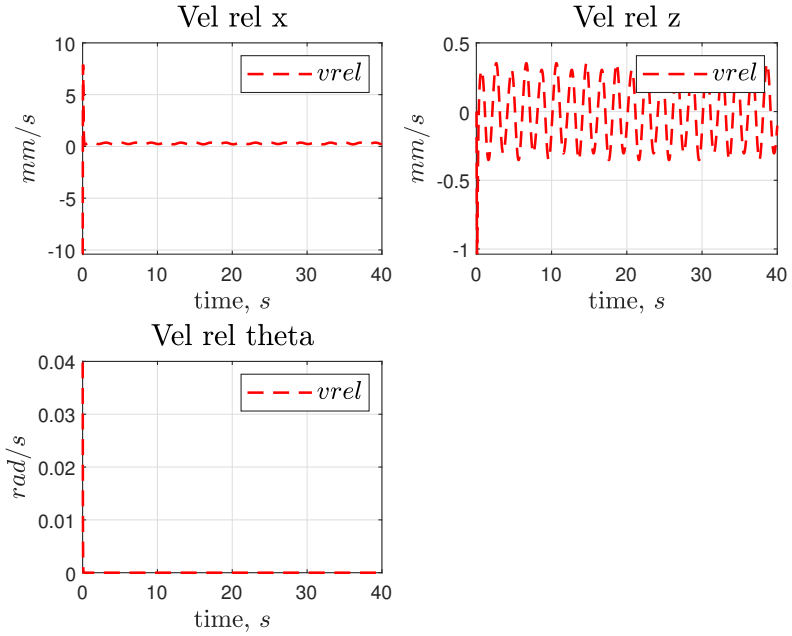

Figura 45: Gráfico da velocidade relativa para Euler + PI

Análise: Não houveram grandes alterações visuais nas velocidades relativas e absolutas, se forem comparados os dois controladores, como visto na figura 46. A única diferença visível, foi no aumento do ruído para a velocidade absoluta em relação a variável θ, que passou a ter uma ordem de 10^{-14}. Contudo, como foi visto nas outras análises, teve pequenas mudanças nas variáveis absolutas, pois o sistema obteve melhor desempenho ao seguir a trajetória. Contudo, o controlador integral não conseguiu compensar dx_{ad} de forma perfeita, e por isso há um erro em regime estacionário. Também, como foi discutido anteriormente, já que só houve mudança no controlador da pose absoluta, as velocidades relativas não sofreram nenhum impacto nos resultados.

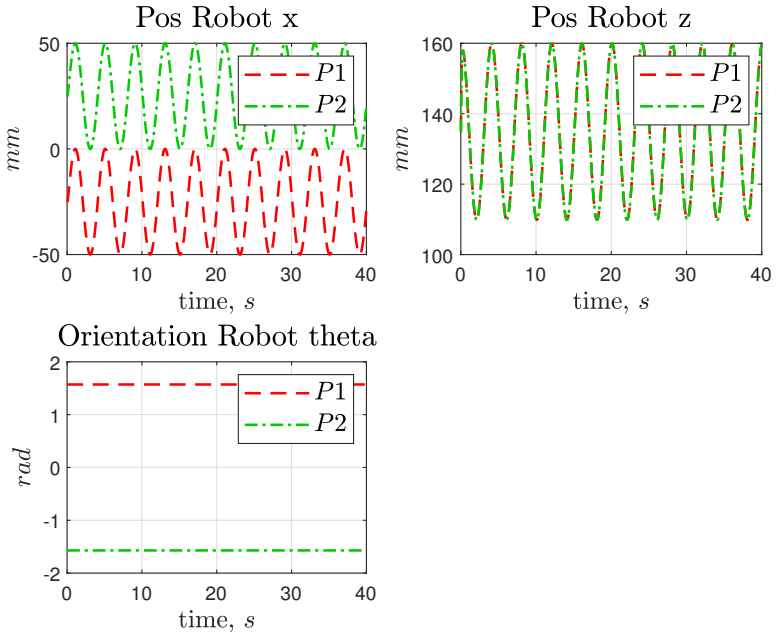

Figura 46: Gráfico da pose dos efetuadores para Euler + PI

Análise: Não houveram alterações significativas nos resultados da posição e da orientação do efetuador, como se pode observar na figura 47. Desse modo, percebe-se que os controladores implementados até agora, possuem resultados razoáveis para realizar a tarefa desejada. Eles também foram capazes de manter os manipuladores estáveis.

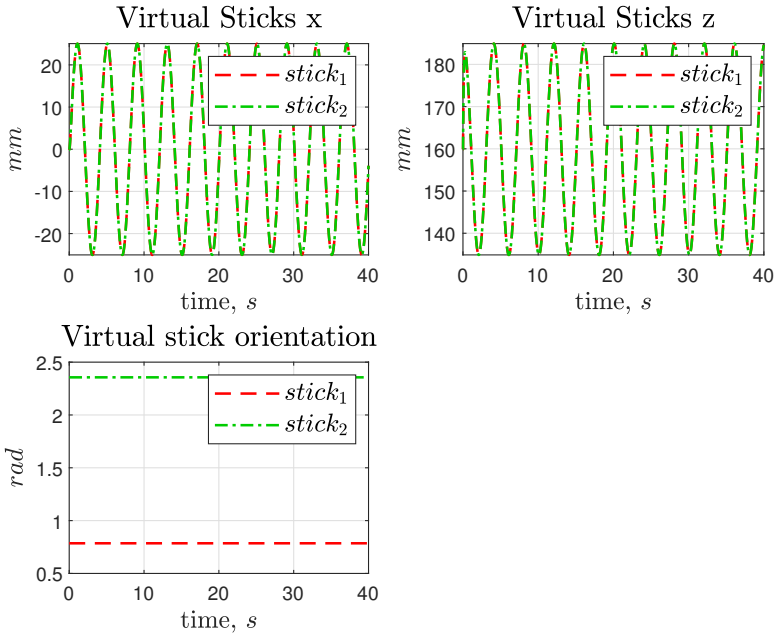

Figura 47: Gráfico da pose dos *Virtual Sticks* para Euler + PI

Análise: Não houveram alterações significativas nos gráficos dos dois controladores, se for feita a comparação entre eles. Desse modo, tanto o controlador P, tanto o controlador PI, fazem com que o sistema robótico cooperativo consiga manter a pose desejada dos *Virtual sticks*, como é percebido na figura 47. Portanto, como foi visto para os resultados das poses dos *end-effectors*, a

pose dos *sticks* seguem o comportamento esperado, visto no desenvolvimento das equações.

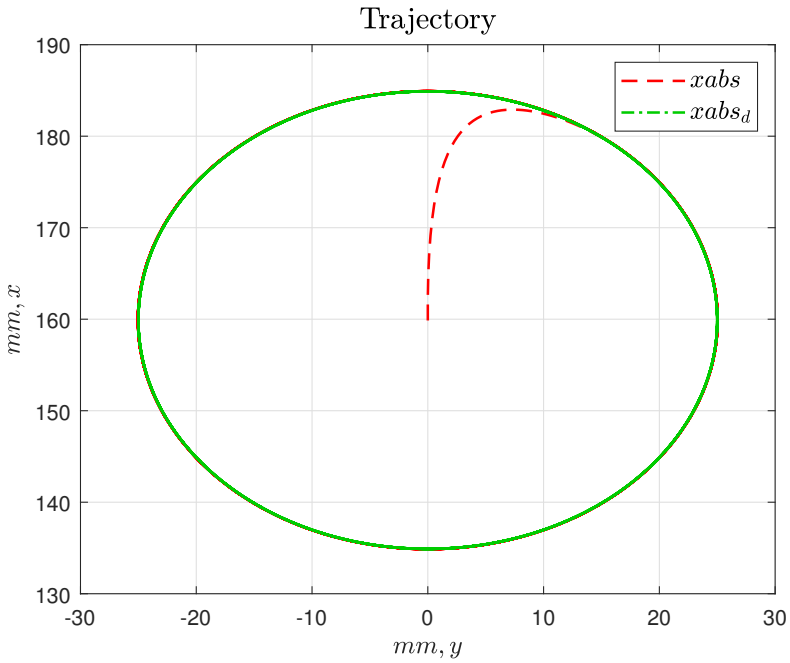

Figura 48: Gráfico da trajetória para Euler + PI

Análise: Olhando a figura 48, da trajetória, percebe-se que o controlador PI apresentou melhores resultados, se for comparado com os resultados do controlador P. Desse modo, ele consegue com que os manipuladores sigam a trajetória desejada mais fielmente.

4. **Tustin com controlador PI**

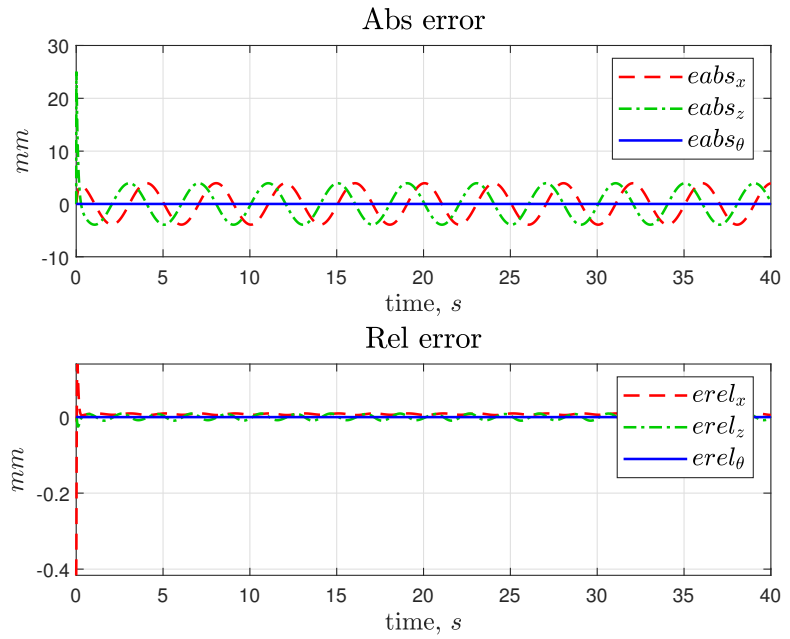

Figura 49: Gráfico do erro para Tustin + PI

Análise: Como ocorreu no controlador P, para o controlador PI, o método de Tustin diminuiu a amplitude do *Overshoot* do sistema, como visto na figura 49. É possível perceber essa melhora,

através do erro relativo, pois para o método de Euler, o pico de *overshoot* chegava a mais de $0.2mm$ e no método de Tustin ele passa um pouco de $0mm$.

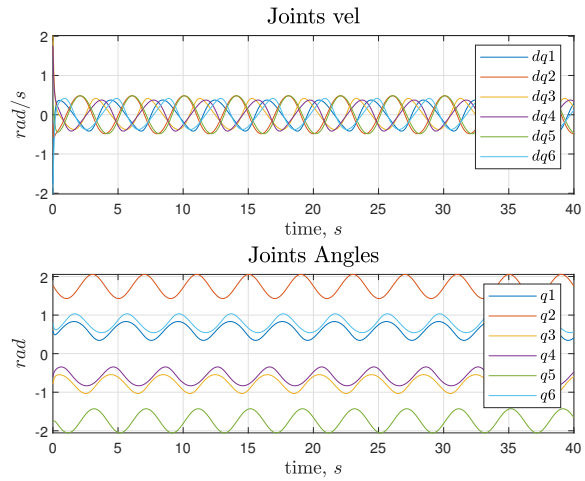

Figura 50: Gráfico dos ângulos das juntas e das velocidades das juntas para Tustin + PI

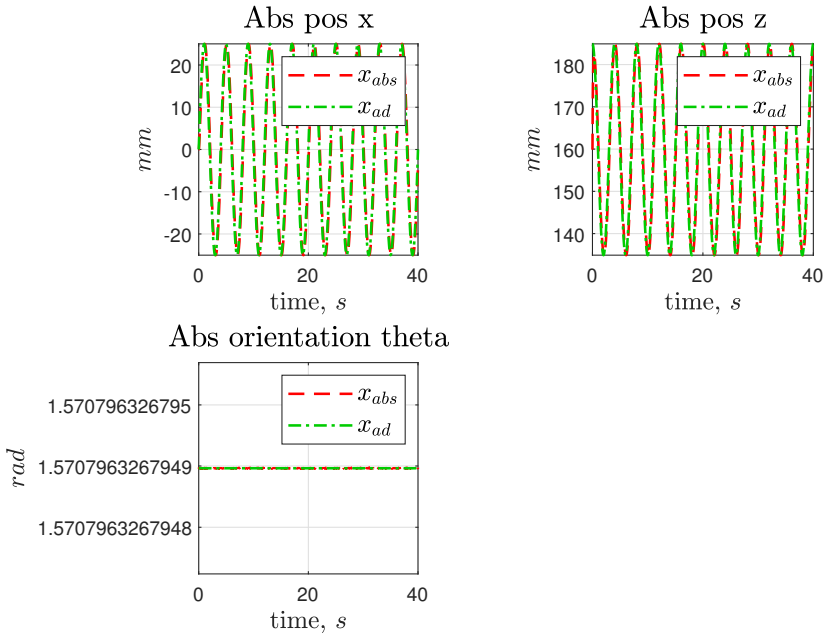

Figura 51: Gráfico da pose absoluta para Tustin + PI

Análise: Não há nenhuma diferença notória nos ângulos das juntas, e também nas velocidades das juntas, se for feita a comparação entre os dois métodos de integração. Também não é possível notar muita diferença na pose absoluta do sistema robótico, se fizer a comparação entre os métodos de Euler de Tustin, como é visto na. Portanto, concluí-se que essa variáveis não sofrem uma grande influência de nenhum dos dois métodos. Também nota-se que pelo método de Tustin, visualizando as imagens 50 e 51, que com o uso do controlador PI, o comportamento dessas variáveis seguem o que foi previsto nas equações (115) e (116).

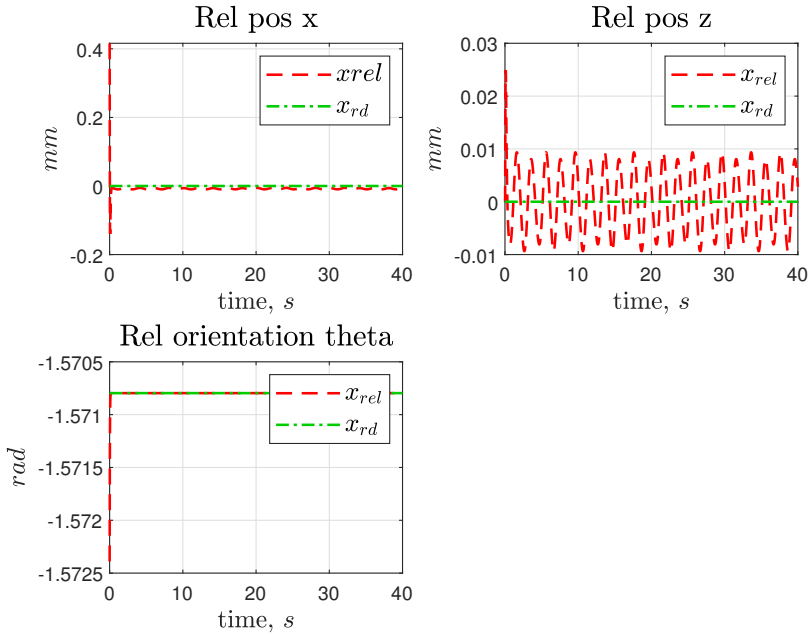

Figura 52: Gráfico da pose relativa para Tustin + PI

Análise: Como ocorreu com o erro relativo, o *overshoot* dos gráficos da pose relativa diminuiu, como é visto na figura 52. É possível perceber essa mudança no gráfico da coordenada z, onde para o método de Euler, a amplitude do *overshoot* chega em $0.04mm$ e para Tustin, essa amplitude não passa de $0.03mm$. Também percebe-se que houve atenuação da amplitude desse mesmo resultado no regime permanente, onde antes esse valor era próximo de $0.02mm$ e agora é próximo de $0.01mm$.

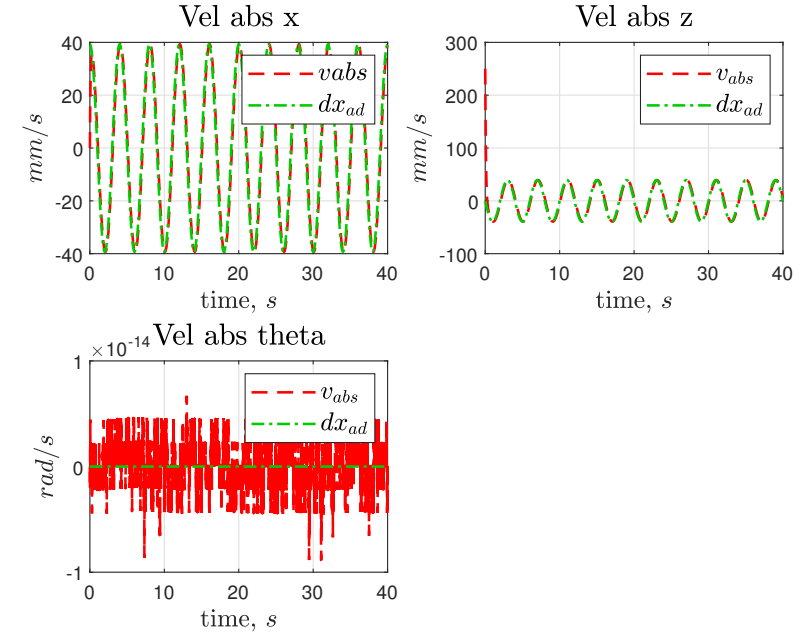

Figura 53: Gráfico da velocidade absoluta para Tustin + PI

Análise: Não há diferença visível nos resultados das velocidades absolutas apresentados pelos dois métodos, como pode ser visto nas imagens 53 e 44. Desse modo, percebe-se que o método

de integração não há grande influencia no comportamento dessa variável.

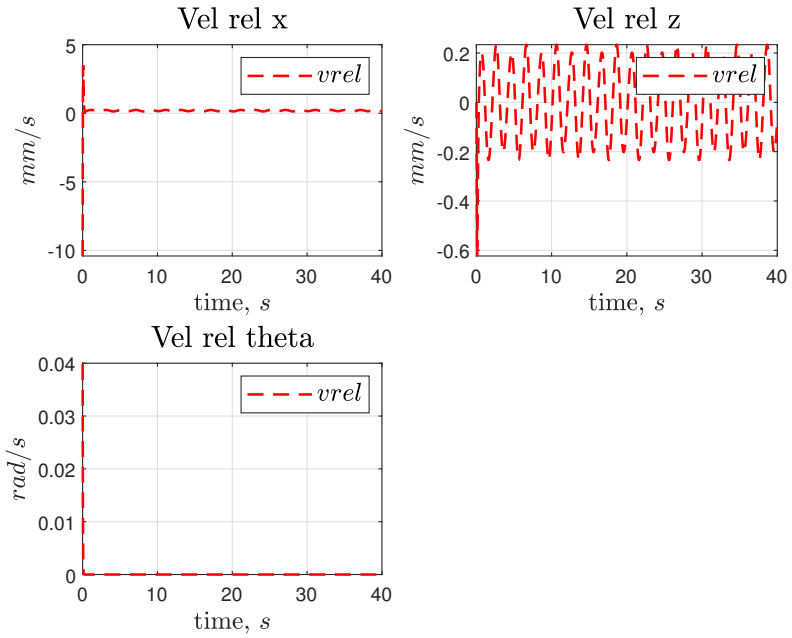

Figura 54: Gráfico da velocidade relativa para Tustin + PI

Análise: Vendo a figura 54, percebe-se que ocorreu o mesmo comportamento se comparado com as outras análises para as variáveis relativas, o método de Tustin, diminuiu o *overshoot* do resultado, se for comparado com o método de Euler. Percebe-se isso, quando é feita uma análise das velocidades em x e z, visto que elas possuíam valores maiores anteriormente. Também ocorreu o mesmo comportamento que tinha ocorrido para a posição relativa na coordenada z, que é a atenuação da amplitude no regime permanente. Contudo, além de diminuir o efeito de *overshoot*, também houve uma melhora em relação ao *undershoot*. É possível notar essa melhora, através do gráfico da velocidade relativa em z, onde antes a amplitude de *undershoot* era de $-'mm/s$ e agora é de $-0.6mm/s$.

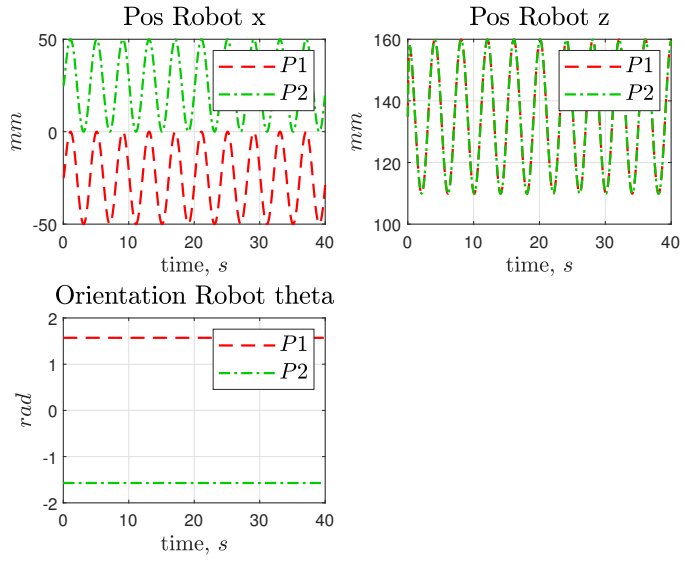

Figura 55: Gráfico da pose dos efetuadores para Tustin + PI

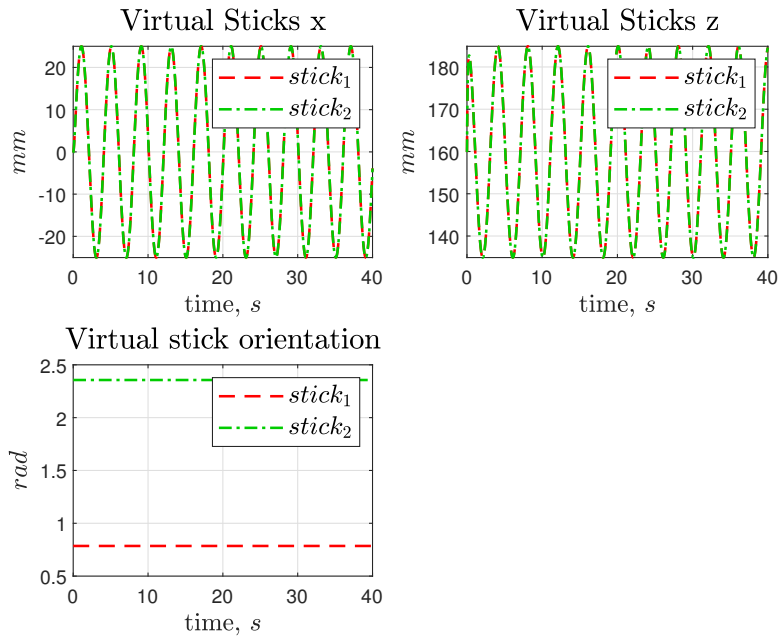

Figura 56: Gráfico da pose dos *Virtual Sticks* para Tustin + PI

Análise: Também não houve alterações nas poses dos efetuadores e dos *virtual sticks*, como é visto na figura 56. Dessa maneira, os dois métodos e os dois controladores, cumpriram bem os seus objetivos, que é fazer com que o sistema robótico siga uma trajetória circular, sem alterar as orientações dos robôs.

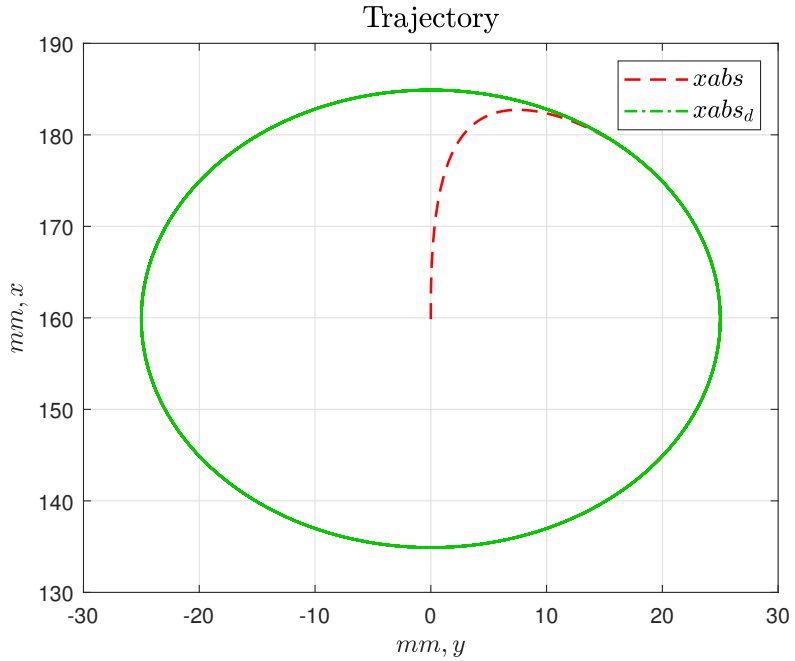

Figura 57: Gráfico da trajetória para Tustin + PI

Análise: Os dois métodos fizeram com que o sistema cooperativo tivesse uma boa performance ao seguir a trajetória desejada, como visto na figura 57 . Contudo, analisando bem os dois gráficos, percebe-se um detalhe que também ocorreu com o controlador P para o método de Tustin, que á a realização da tarefa de maneira mais suave. É possível perceber, que para o método de Euler,

algumas vezes o sistema tende a sair um pouco da trajetória, ou seja, o raio do círculo percorrido, torna-se um pouco maior que o raio do círculo da trajetória desejada. Entretanto, para o método de Tustin esse problema não ocorre.

5. **Euler avançado com controlador Feedforward + P**

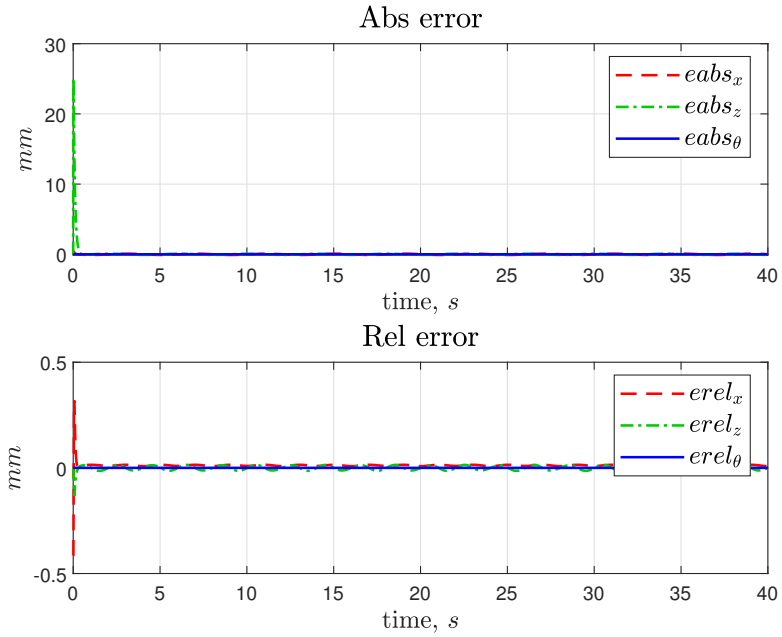

Figura 58: Gráfico do erro para Euler + Feedforward

Análise: Percebe-se uma grande melhora no comportamento do erro como é observado na figura 58. Se for for feita a comparação com os outros controladores. Com o controlador feedforward + P, foi possível eliminar o erro de regime permanente, que existia na variável absoluta nos outros algoritmos de controle.

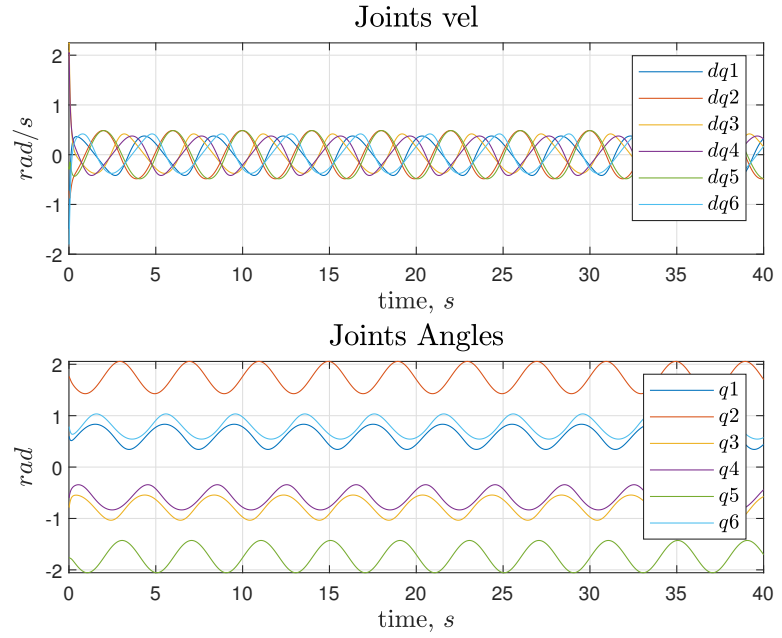

Figura 59: Gráfico dos ângulos das juntas e das velocidades das juntas para Euler + Feedforward

Análise: Para as variáveis das juntas, todos os controladores implementados tiveram uma boa performance, que era realizar a tarefa dentro dos valores aceitáveis para as velocidades das juntas e para os ângulos das juntas. Esse controlador manteve o bom resultado, como visto na figura 59.

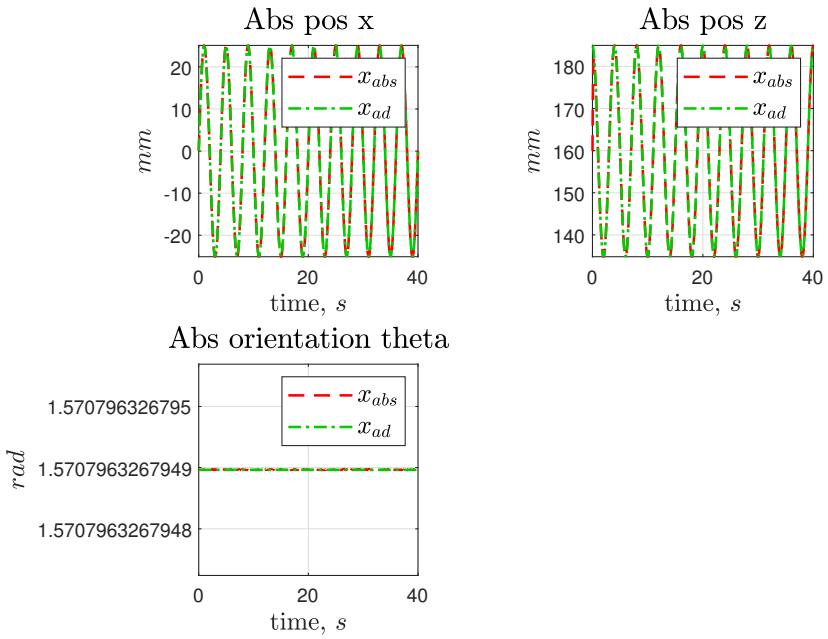

Figura 60: Gráfico da pose absoluta para Euler + Feedforward

Análise: Para a pose absoluta, nota-se que no começo da execução da tarefa, o controlador feed-forwatd + P, como observado na figura 60, e o controlador PI, possuem o mesmo comportamento. Contudo, quando a variável chega em regime estacionário, o gráfico da posição absoluta, torna-se muito próximo ao gráfico da posição desejada. Esse detalhe não ocorreu de uma forma muito boa nos controladores P e PI.

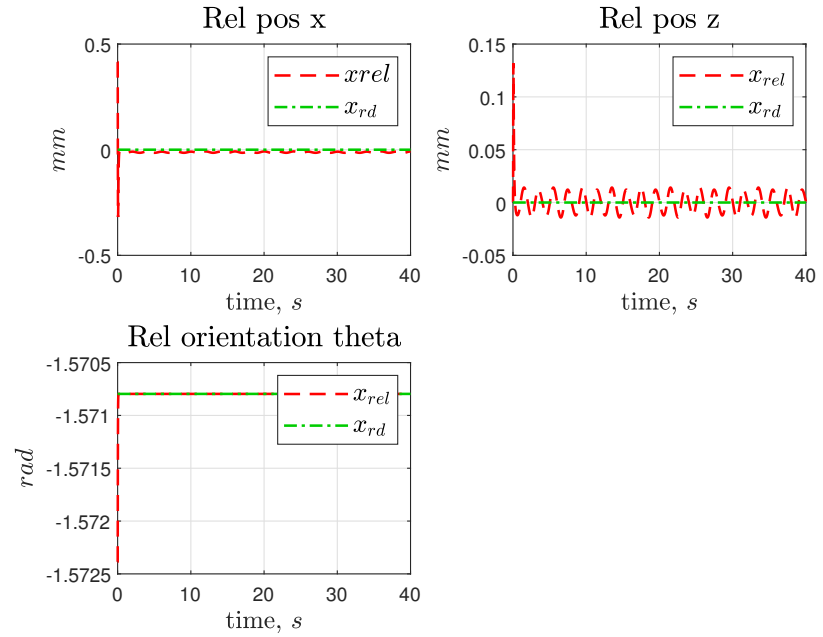

Figura 61: Gráfico da pose relativa para Euler + Feedforward

Análise: Apesar da mudança do controlador ser somente na variável absoluta, o erro na variável relativa também apresenta mudança, como observa-se na figura 61. Essa mudança pode ser vista nas coordenadas x e z, onde a amplitude de *overshoot* piorou. No controlador PI com método de Euler, a coordenada x apresenta um valor de *overshoot* máximo próximo de $0.4mm$, porém para o controlador feedforward + P, esse valor é próximo de $0.5mm$. Esse detalhe é ainda pior para a coordenada z, onde antes o *overshoot* era de $0.04mm$ e agora aproxima-se de $0.15mm$.

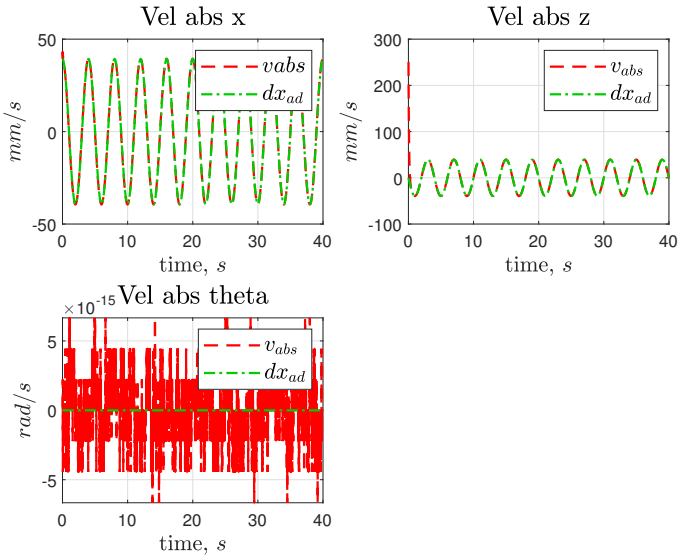

Figura 62: Gráfico da velocidade absoluta para Euler + Feedforward

Análise: É possível perceber mudanças nas amplitudes das coordenadas da velocidade absoluta, como visto na figura 62. Esse detalhe é visto melhor no gráfico da coordenada x, onde na figura 53, para PI + Euler, a amplitude dessa variável chega no máximo de $40mm/s$ e agora, na figura 62, chega próximo de $50mm/s$. Essa mudança ocorre, devido ao controlador compensar dx_{ad}, para eliminar o erro de regime estacionário

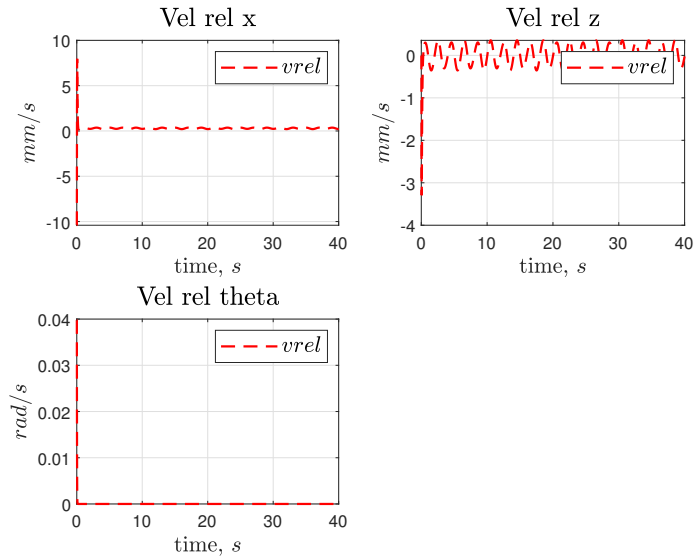

Figura 63: Gráfico da velocidade relativa para Euler + Feedforward

Análise: Com o aumento do *overshoot* na pose relativa, também teve um aumento desse parâme-

tro na velocidade relativa, devido ao controlador proporcional, como se pode ver, na figura 63. Com isso, a amplitude de *overshoot* na coordenada x, para o controlador feedforward, chega próximo de $10mm/s$ e para o controlador PI, esse valor era menor que $5mm/s$. Um detalhe que não era possível ser notado, na pose relativa, devido a escala do gráfico, é que a amplitude de *undershoot* também piorou. Para o controlador PI, o *undershoot* chega no valor máximo de $-0.6mm/s$ e agora, com esse controlador, ele ultrapassa $-3mm/s$.

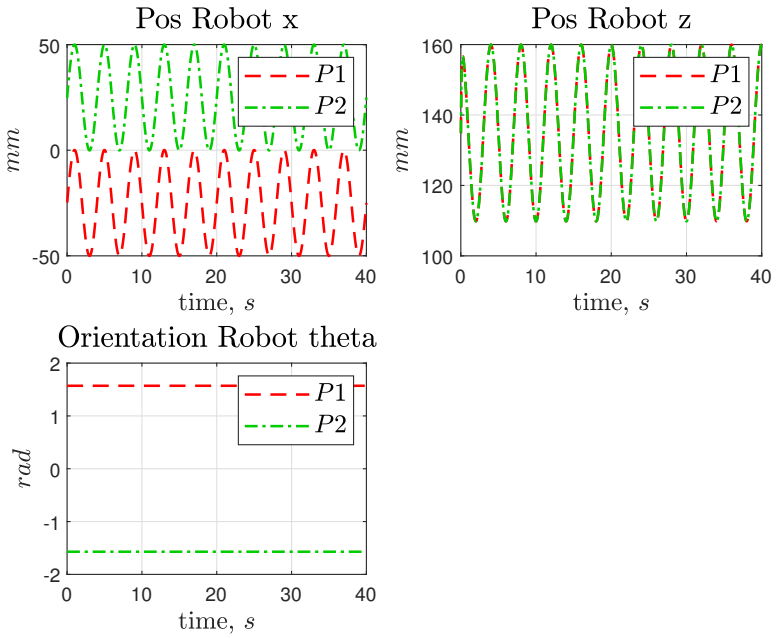

Figura 64: Gráfico da pose do efetuador para Euler + Feedforward

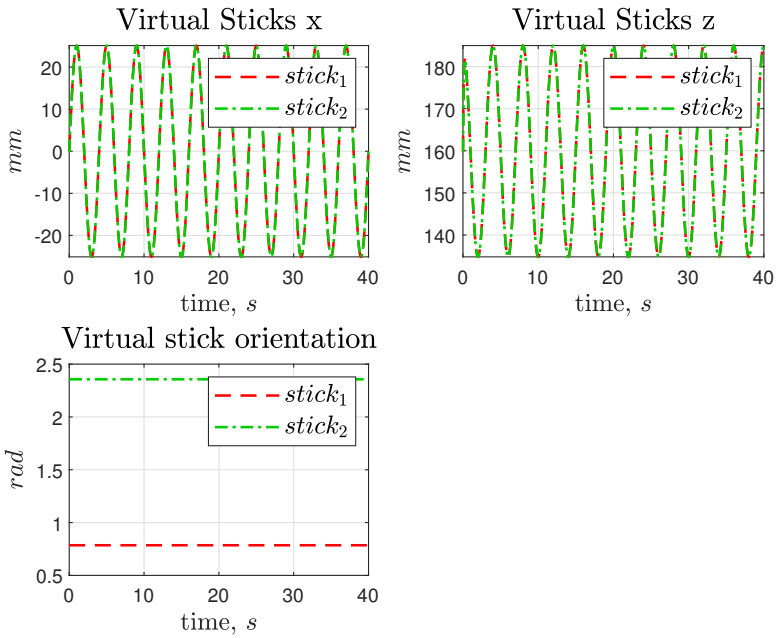

Figura 65: Gráfico da pose do *Virtual Stick* para Euler + Feedforward

Análise: Não foram apresentadas grandes mudanças na pose dos efetuadores e dos *virtual sticks*, se for feita a comparação desse controlador com os anteriores. Desse modo, concluí-se que todos os controladores conseguiram ter resultados razoáveis, ao permitir que o sistema não torne-se

instável e também permitir que os manipuladores sigam uma trajetória circular.

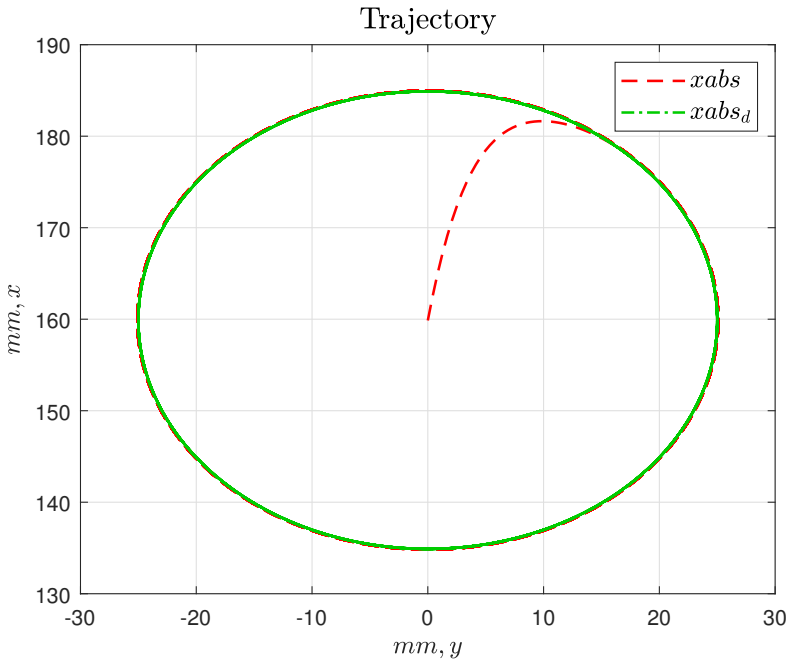

Figura 66: Gráfico da trajetória para Euler + Feedforward

Análise: Olhando a figura 66, concluí-se que o controlador executou uma performance muito boa ao seguir a trajetória desejada. Contudo, nota-se que o raio do círculo feito pelo sistema robótico, é levemente maior que o raio da trajetória desejada.

6. **Tustin com controlador Feedforward + P**

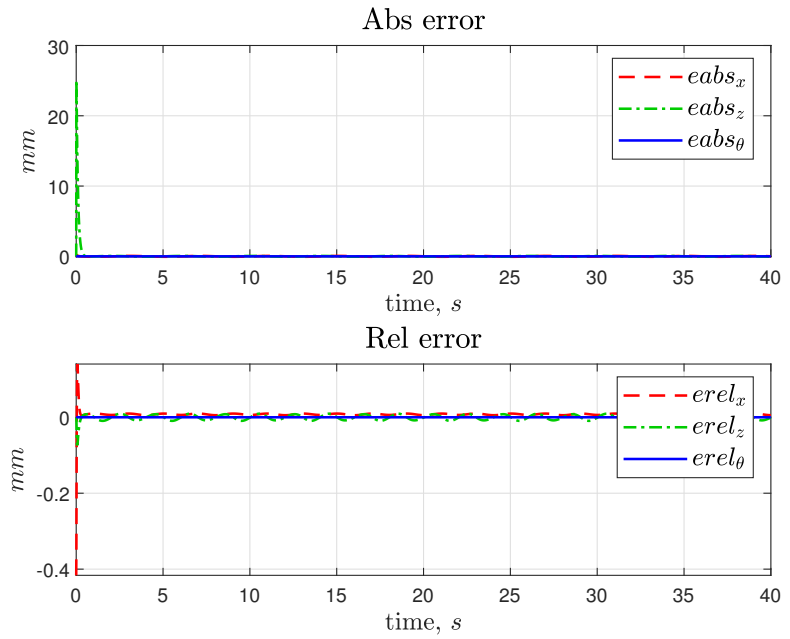

Figura 67: Gráfico do erro para Euler + Feedforward

Análise: Com a mudança de método de integração, não tiveram alterações perceptíveis nos erros absolutos e relativos, como visto na figura 67. Desse modo, concluí-se independente do método,

o controlador feedforward consegue fazer com que os erros tendam a zero no regime permanente.

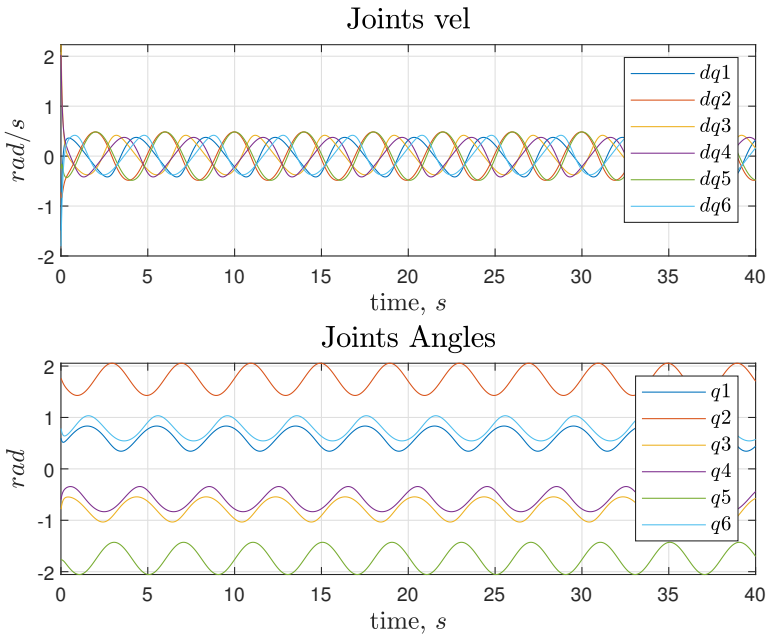

Figura 68: Gráfico dos ângulos das juntas e das velocidades das juntas para Tustin + Feedforward

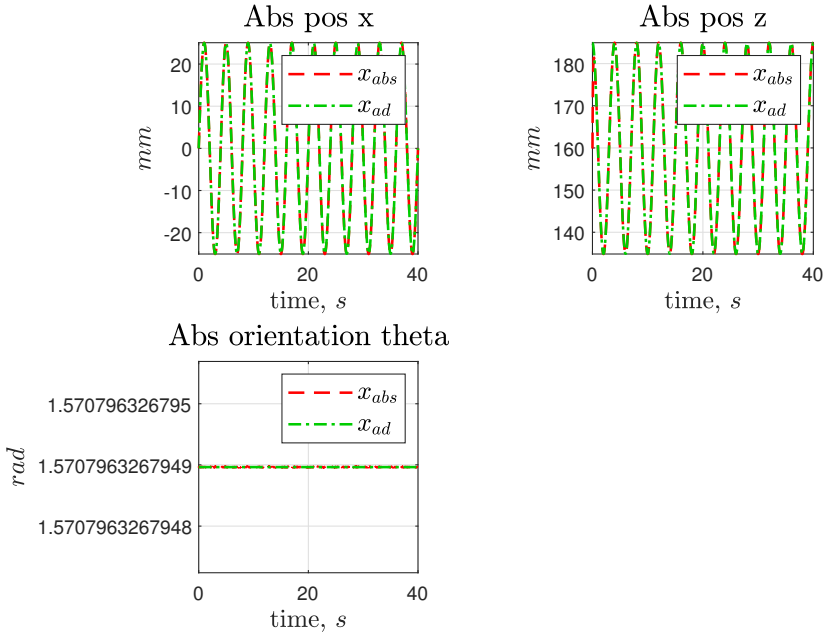

Figura 69: Gráfico da pose absoluta para Tustin + Feedforward

Análise: Os ângulos das juntas e as velocidades das juntas apresentam os mesmos resultados que foram vistos nos outros métodos e outros controladores, como observa-se na figura 68. Com isso, o Tustin com feedforward + P, é capaz de fazer com que o sistema siga a trajetória desejada, sem saturar os atuadores. Conclusão similar pode ser feita para a pose absoluta vista na imagem 69, aonde ela mantém o mesmo comportamento que foi notado para o método de Euler. No regime estacionário, os dois métodos de integração, fazem com que a pose absoluta fique muito próxima da pose desejada.

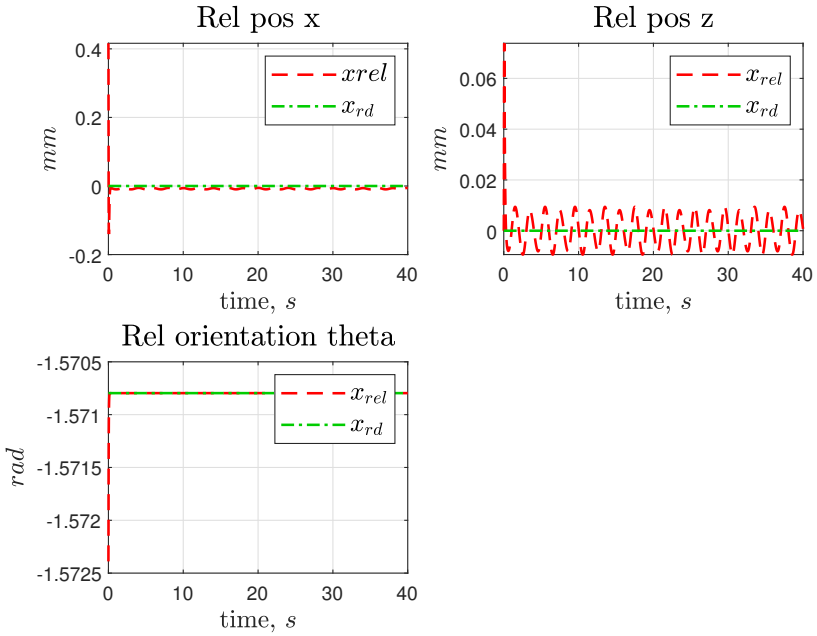

Figura 70: Gráfico da pose relativa para Tustin + Feedforward

Análise: Percebe-se que com a implementação do método de Tustin, houve uma melhora nos resultados da pose relativa. Nota-se isso, visualizando os gráficos da figura 70, aonde o *undershoot* e *overshoot* diminuíram de amplitude. Essa vantagem desse método tinha sido vista anteriormente, para os outros controladores implementados. Desse modo, concluí-se que o método de Tustin possui a característica de suavizar esses parâmetros.

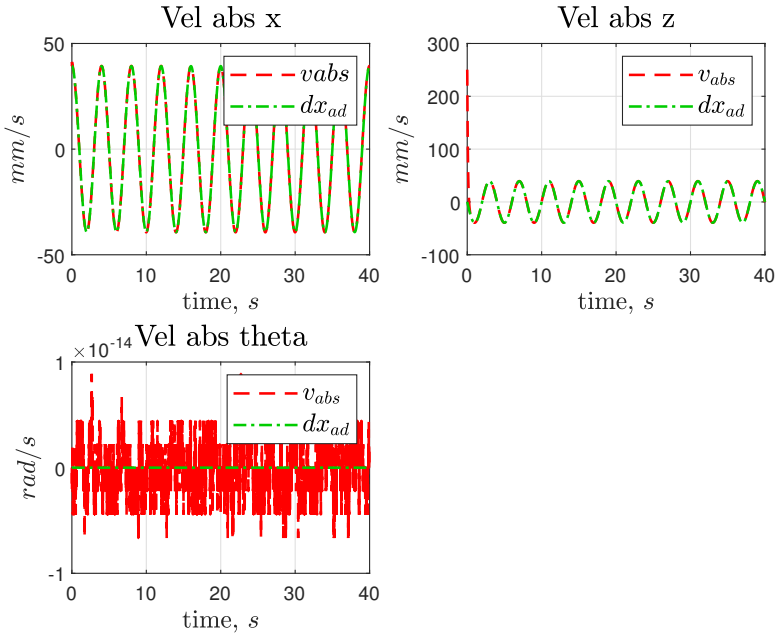

Figura 71: Gráfico da velocidade absoluta para Tustin + Feedforward

Análise: A velocidade absoluta apresenta resultados semelhantes se comparado com o método de Euler. Contudo, uma mudança notável, visualizando a figura 71, é no aumento do ruído na variável θ. Esse detalhe foi percebido também nos outros controladores.

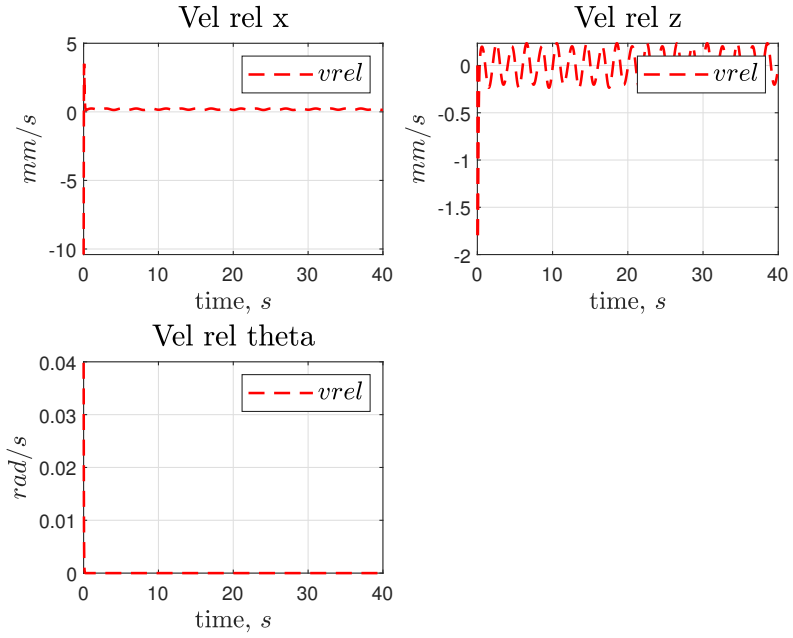

Figura 72: Gráfico da velocidade relativa para Tustin + Feedforward

Análise: Devido a atenuação dos efeitos de *overshoot* e *undershoot* apresentados nos gráficos da pose relativa, os gráficos da velocidade relativa, como visto na imagem 72, apresentam os mesmos comportamentos. Esse resultado ocorre, pois a velocidade relativa e a pose relativa são proporcionais. Com isso, utilizando o método de Euler, a amplitude de *overshoot* chegava próximo a $10mm/s$ para a coordenada x e agora ele não ultrapassa o valor de $5mm/s$. Também é visto uma melhora no *undershoot*, visualizando o gráfico da coordenada z. Percebe-se isso, pois com o método anterior, o pico desse parâmetro chegava a passar de $-3mm/s$ e agora ele não ultrapassa o valor de $-2mm/s$.

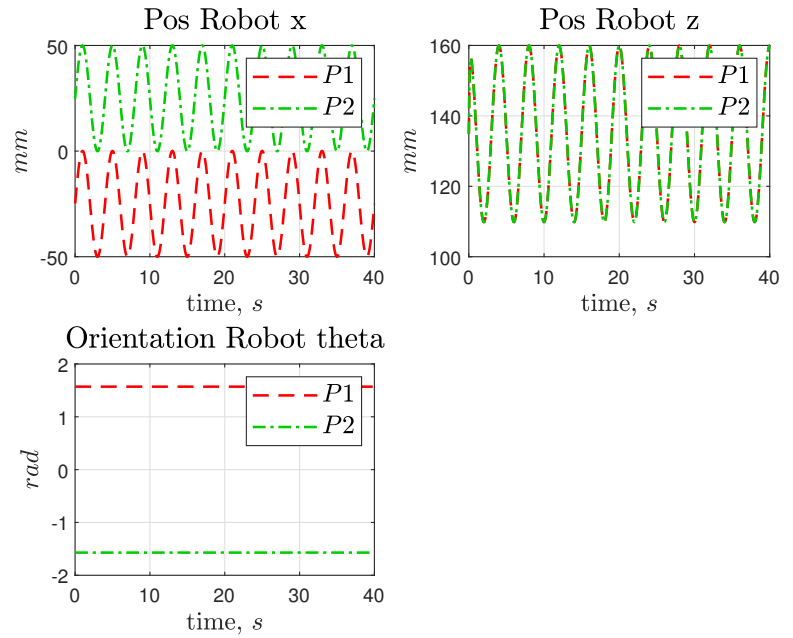

Figura 73: Gráfico da pose dos efetuadores para Tustin + Feedforward

Análise: Não houve nenhuma mudança na pose do robô e do *Virtual Stick*, com a alteração do método de integração, como é possível perceber nas figuras 73 e 74. Desse modo, os dois métodos cumpriram bem o seu papel.

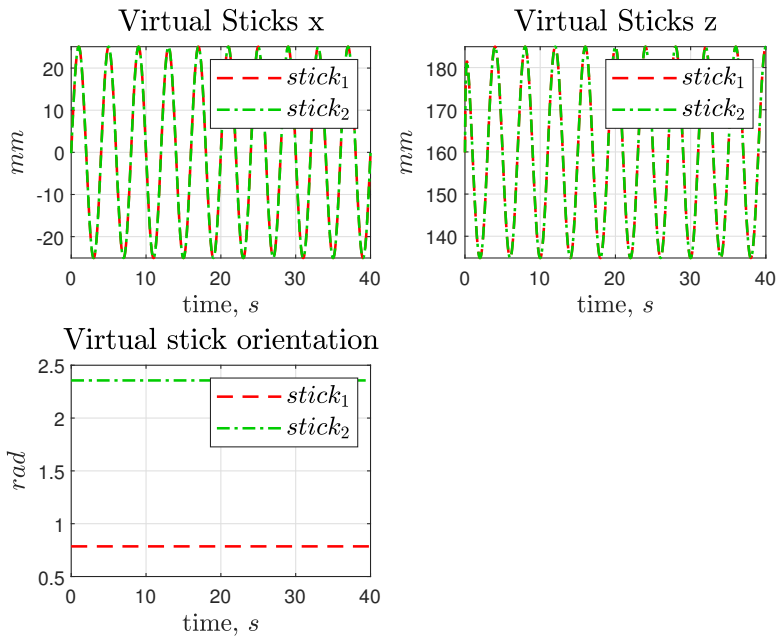

Figura 74: Gráfico da pose dos *Virtual Sticks* para Tustin + Feedforward

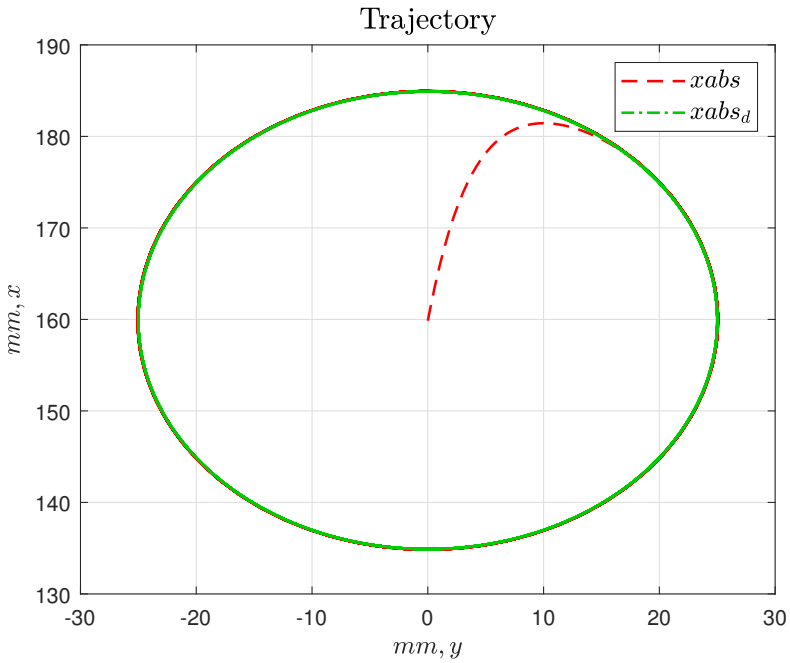

Figura 75: Gráfico da trajetória para Tustin + Feedforward

Análise: Com a utilização do método de Tustin, percebe-se que o sistema robótico possui uma performance melhor, se for comparado com o método de Euler. Desse modo, os raios da trajetória percorrida e desejada são muito próximos, diferente do método anterior, onde o raio da trajetória percorrida era maior. Desse modo, percebe-se que unindo o método de Tustin com o controlador feedforward + P,obtiveram-se melhores resultados.

7 Testes Experimentais

Nessa seção, serão apresentados os resultados obtidos nos testes experimentais, realizados com dois PhantomX Pincher. Desse modo, o algoritmo de controle implementado para realização desses testes, foi o controle feedforward + P com o método de integração de Tustin. Com isso, foi utilizado o ROS para enviar os comandos de entrada para cada motor, e também para realizar as leituras dos *enconders*. Nesse contexto, observando a figura 76 a seguir, é possível ver um diagrama que mostra todos os nós e tópicos que estão sendo utilizados para a realização da tarefa:

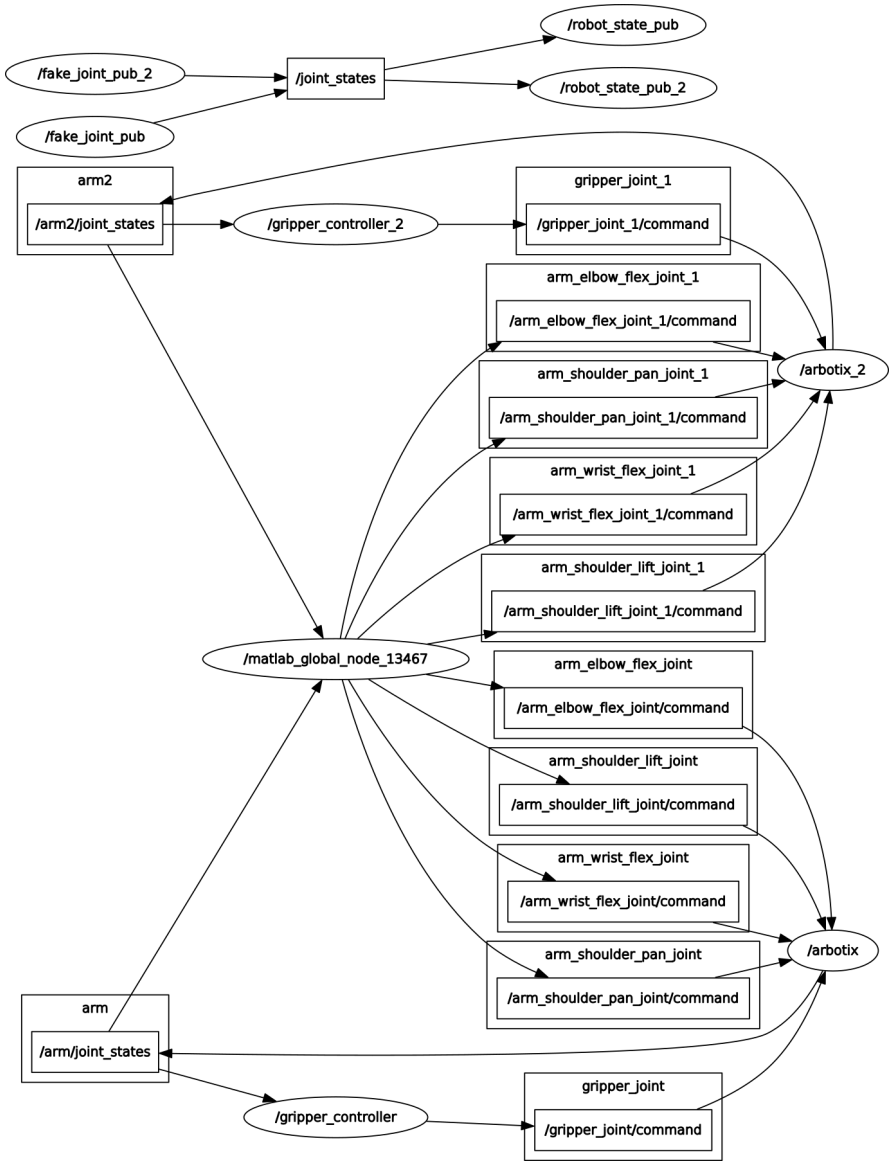

Figura 76: Diagrama de nós e tópicos do ROS

Na figura acima, os nós são as elipses, e os tópicos são os retângulos. Dessa forma, percebe-se que o matlab recebe os ângulos das juntas, realiza o controle e publica as entradas do sistema em cada junta. Após isso, cada tópico que está sendo sub escrito pelo matlab, publica nas placas arbotix de cada manipulador.

Desse modo, a seguir, nas figura 77, são exibidas algumas imagens do sistema robótico coo-

perativo:

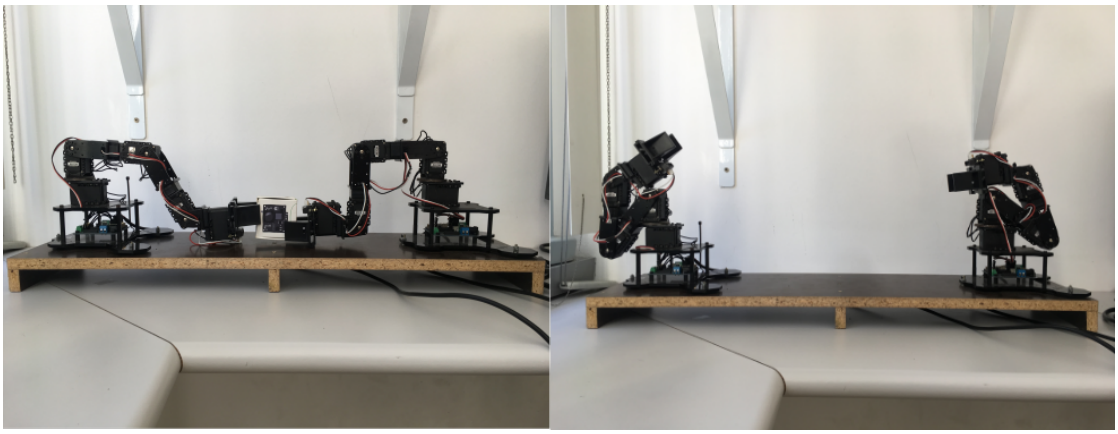

Figura 77: Visão frontal do sistema robótico:

a ROS com PhantomX Pincher

Com o pacote de ROS utilizado, é preciso enviar ângulos das juntas como entrada do sistema, ao invés das velocidades das juntas, como é visto na equação (46). Logo, é necessário utilizar o método de integração de Tustin para obter a nova entrada. Com isso, utilizando a equação (34):

$$\bar{u}(k+1) = \bar{u}(k) + \frac{h}{2}(u(k+1) + u(k)), \tag{123}$$

Aonde, $u(k)$ é obtido da equação (46) e $u(\bar{k})$ é a integral das velocidades das juntas. Desse modo, existirão ângulos das juntas obtidos das leituras dos sensores e os ângulos das juntas obtidos pela integração de dq.

b Resultados Experimentais

A seguir, serão apresentados alguns gráficos dos resultados dos testes experimentais.

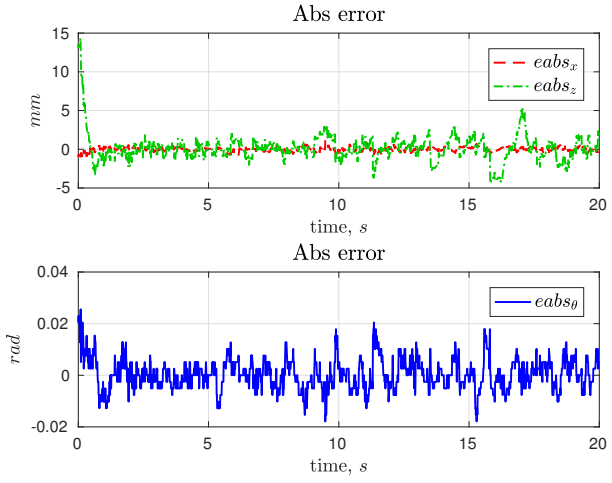

Figura 78: Erro absoluto nos testes experimentais

Análise: Na figura 78 acima, é possível perceber que os erros da pose absoluta tendem a zero ao longo do tempo, como pode ser visto na simulação. Aliado a isso, outro parâmetro que é importante ser analisado, é o *Overshoot* dos gráficos, que está menor, se for feita a comparação entre os testes práticos e a simulação. Entretanto, há muito ruído, devido as leituras dos sensores, como se pode ver na figura acima. Esse ruído é muito mais intenso na orientação absoluta, pois ela depende diretamente dos ângulos lidos nos *encoders*. Entretanto, apesar desse problema, concluí-se que o controlador Feedforward + proporcional, é capaz de realizar o controle das variáveis absolutas do sistema cooperativo na prática, como foi previsto nas simulações.

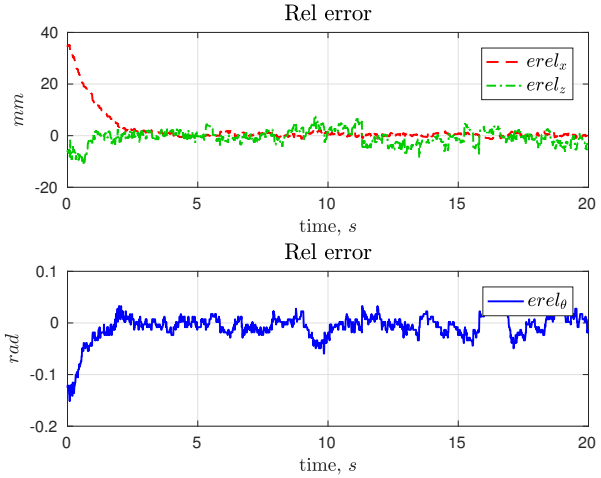

Figura 79: Erro relativo nos testes experimentais

Análise: Na figura 79, observa-se que no regime permanente, os erros da pose relativa tendem a zero, como também pode ser enxergado na simulação. Contudo, o *Overshoot* e o *Undershoot* estão muito maiores, se for feita a comparação com a simulação. Outro problema que se pode perceber, é a existência do ruído no erro relativo também. Entretanto, apesar disso, concluí-se que o controlador proporcional é suficiente para controlar as variáveis relativas na prática também.

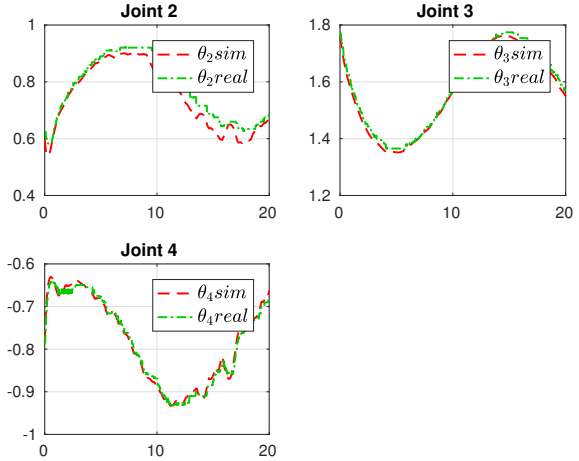

Figura 80: Ângulos das juntas do manipulador esquerdo nos testes experimentais

Análise: Observando a figura 80, percebe-se que para o manipulador esquerdo, os ângulos das juntas calculados pelo método de integração e os ângulos das juntas medidos pelos *enconders*

tendem a um mesmo valor ao longo do tempo. Apesar de não conseguir visualizar muito bem o ruído das leituras, devido a escala do gráfico, quando são utilizadas as equações da cinemática direta, esse ruído é amplificado e aparece nas variáveis relativas e absolutas. Nesse contexto, apesar dos gráficos dos ângulos dos *enconders* e dos ângulos calculados serem muito parecidos, eles apresentam uma leve diferença, devido a falta de calibração dos manipuladores.

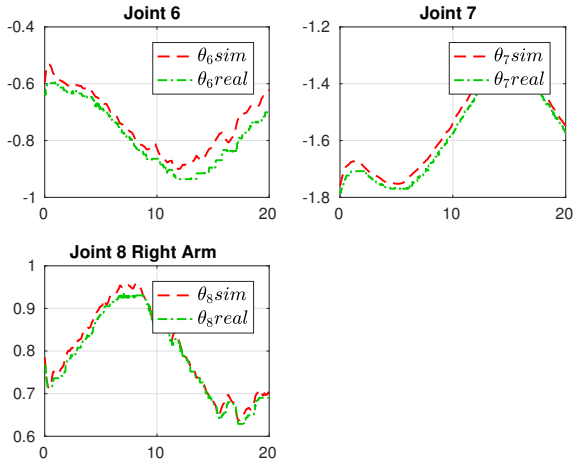

Figura 81: Ângulos das juntas do manipulador direito nos testes experimentais

Análise: As mesmas conclusões encontradas pela análise dos gráficos das juntas do manipulador esquerdo, podem ser utilizadas para o manipulador direito, observando os gráficos da figura 81.

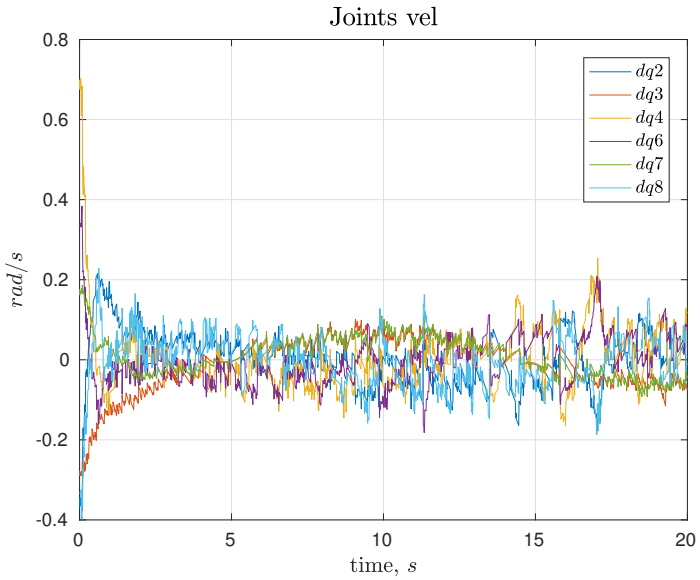

Figura 82: Velocidade das juntas dos manipuladores nos testes experimentais

Análise: Na figura 82, é possível visualizar as velocidades das juntas de ambos os manipuladores. Desse modo, apesar do ruído devido a leitura dos sensores, os motores estão com uma velocidade dentro do limite estabelecido, que é de 2 rad/s. Esse valor se mantém dentro dos limites até no *Overshoot*, ou no *Undershoot* dos gráficos, que é próximo de 0.8 rad/s e -0.4 rad/s.

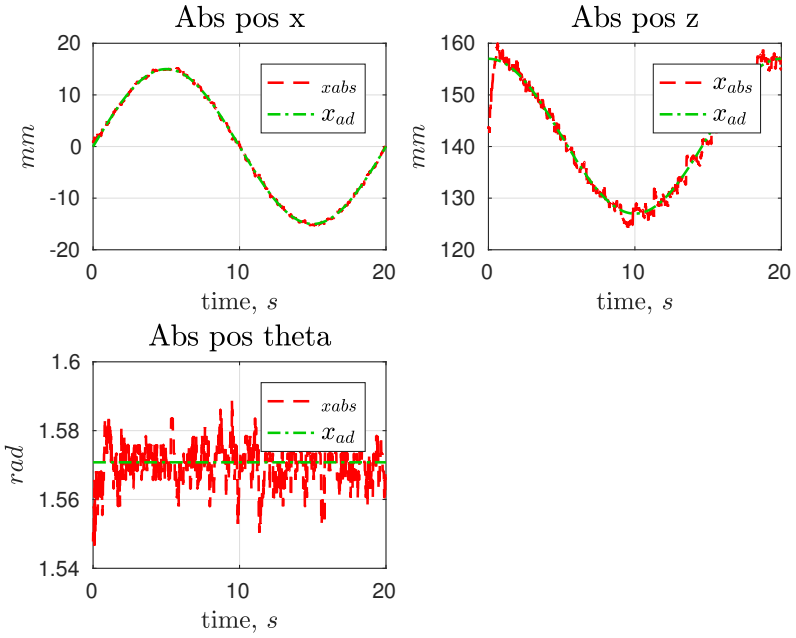

Figura 83: Pose absoluta dos manipuladores nos testes experimentais

Análise: Observando a figura 83, é possível ver a pose absoluta do sistema cooperativo. Com isso, percebe-se, que a variável x, não apresenta grandes mudanças devido aos ruídos, diferente da variável z. Esse problema ocorre, pois a variável z depende do cosseno dos ângulos. Desse modo, quando é feita a cinemática direta, o cosseno do ruído apresenta valores altos, pois o cosseno de um ângulo pequeno, é próximo de 1. Portanto, isso faz com que apareçam senóides de amplitudes maiores que a do sinal original, em frequências maiores, causando o grande ruído visto no gráfico. Também é possível perceber, que há muito ruído na orientação absoluta, visto que essa variável depende diretamente dos ângulos lidos.

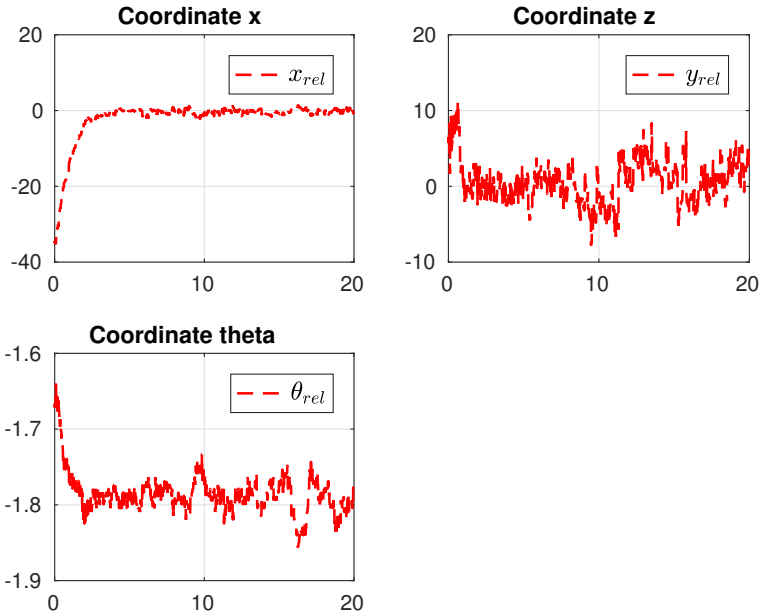

Figura 84: Pose relativa dos manipuladores nos testes experimentais

Análise: Visualizando a figura 84, é possível fazer a mesma análise que foi feita para pose a

absoluta.

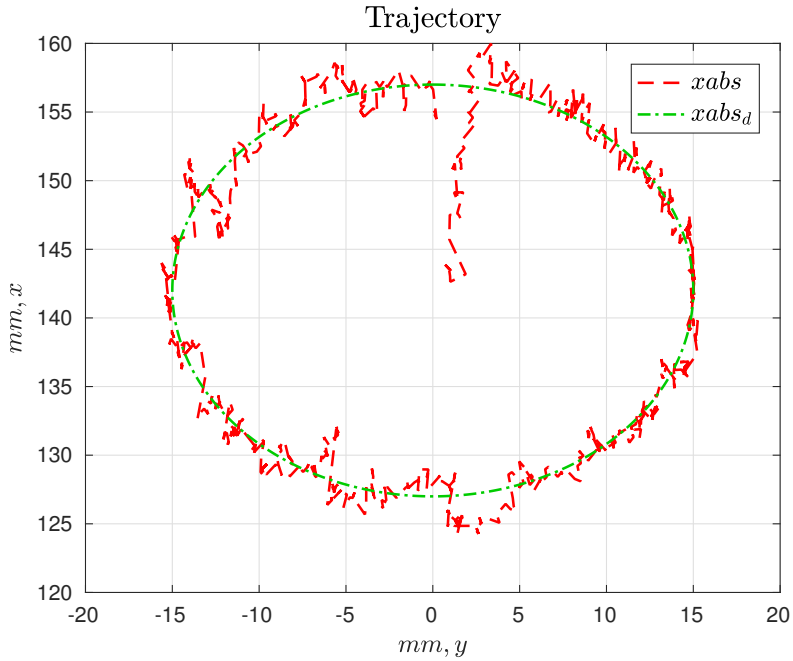

Figura 85: Trajetória percorrida nos testes experimentais

Análise: Visualizando a figura 85, percebe-se que os manipuladores robóticos conseguem seguir uma trajetória circular. Contudo, devido a presença de ruídos nas leituras dos sensores, que depois são transmitidos para a pose dos manipuladores, o sistema segue a trajetória com bastante oscilações.

8 Conclusão

a Conclusão

O objetivo desse projeto, foi implementar um algoritmo de controle para sistemas robóticos cooperativos. Dessa maneira, foi estudado sobre como realizar a modelagem desse tipo de sistema através de uma abordagem denonimada *Symmetric Formulation*. Com isso, os movimentos dos manipuladores foram colocados em função das variáveis absolutas e relativas do sistema. Logo, após a realização da modelagem, foi investigado sobre técnicas de controle cinemático, e com isso, foram implementadas três técnicas: (1) Controle proporcional; (2) Controle Proporcional + Integral; (3) Controle Feedforward + proporcional.

Desse modo, foi feita a simulação dessas três técnicas de controle. Com isso, o controlador proporcional, foi capaz de seguir a trajetória circular, porém, devido a existência de um erro em regime permanente, a tarefa não foi executada com uma performance razoável. Logo, após utilizar o controlador P, foi implementado também, um controlador I. Dessa forma, com o controlador integral, o sistema obteve melhores resultados, porém, não foram suficientes para evitar erros de regime permanente. Entretanto, o controlador feedforward, foi capaz de acabar com esses problemas, visto que, fez com que os erros em regime permanente fossem muito próximos de zero. Além de testar três leis de controle diferentes, também foram testados dois métodos de integração diferentes, que são Euler avançado e Tustin. O método que obteve melhores resultados, foi o Tustin, capaz de minimizar problemas como *overshoot* e *undershoot*.

Nesse contexto, foi utilizado o método de Tustin com um controlador feedForward + proporcional para realizar os testes experimentais, pois foi a combinação que obteve melhores resultados nos testes experimentais. Contudo, na prática, apesar do sistema conseguir seguir uma trajetória circular, os efeitos não foram parecidos com o da simulação, devido ao grande ruído presente na leitura dos sensores.

b Trabalhos Futuros

Para os trabalhos futuros, é necessário realizar as calibrações dos manipuladores ou acrescentar um controlador adaptativo, para acabar com as incertezas paramétricas. Também é necessário a implementação de um bom filtro passa- baixa, para atenuar o ruído. Outra tarefa desejada para o futuro, é fazer com que os robôs sejam capazes de realizar movimentos cooperativos a distância, sem que estejam conectados a um mesmo computador.

Referências

[1] G. M. Freitas, A. C. Leite, and F. Lizarralde, “Kinematic Control of Constrained Robotic System.” *SBA Revista Controle Automação*, vol. 22, no. 6, pp. 559–572, 2011.

[2] Z. JU, C. YANG, and H. MA, “Kinematics Modeling and Experimental Verification of Baxter Robot„” in *33rd Chinese Control Conference*, Nanjing, China, 2014, pp. 8519–8523.

[3] R. de Oliveira Faria, “Hybrid Kinematic Control of Dual-arm Cooperative Robots for Object Manipulation.” Rio de Janeiro, p. 125, ”2016”.

[4] I. Y. Y. Tomasevich, “Passivity-Based Adaptive Bilateral Teleoperation Control for Uncertain Manipulators without Jerk Measurements.” Ph.D. dissertation, UFRJ/COPPE, Rio de Janeiro, ”2017”.

[5] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning annd Control*. Springer-Verlag London Ltda, 2011.