
Engenharia de Controle e Automação

**Trabalho Prático de Manipuladores Robooóticos**

Lucas Merhi Faria
Rafael Araújo Medeiros
Vincent Pernarh

Prof. Gustavo Medeiros Freitas

Belo Horizonte, Novembro/2023

# 1 Introdução

A Indústria 4.0, também conhecida como a quarta revolução industrial, é caracterizada pela automatização de processos industriais e pela integração de diversas tecnologias, tanto em ambientes físicos quanto digitais. A engenharia de controle e automação desempenha um papel abrangente em várias áreas da indústria, sendo a robótica um componente crucial. A presença crescente de robôs inteligentes é evidente em diversos processos industriais.

Os robôs industriais disponíveis atualmente são projetados para tarefas específicas, variando suas configurações de acordo com os graus de liberdade exigidos pela atividade. Em geral, para operar em um espaço tridimensional, um robô terá 6 graus de liberdade, o que implica em 6 juntas. Um exemplo desse tipo de robô é o Comau SmartSix, ilustrado na Figura 1.

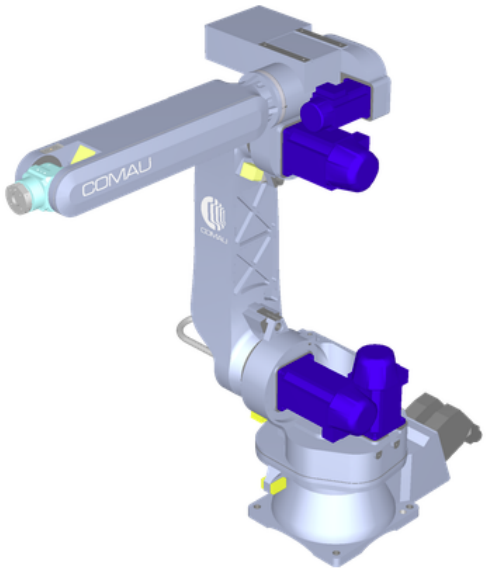


Figura 1: Robô Comau SmartSix [1]

Para empregar o robô, é imprescindível programá-lo e controlá-lo por meio de software. Dada a variedade de tarefas, serão executados movimentos e trajetórias distintas, tornando o estudo do controle cinemático não apenas necessário, mas fundamental.

# 2 Revisão de Conceitos

## 2.1 Cinemática diferencial

Conforme a abordado em [4] e [3] bem como em [2], a cinemática diferencial, no contexto de manipuladores robóticos, investiga a relação entre as velocidades das juntas de um robô e a velocidade de seu efetuador. A matriz Jacobiana $J$ estabelece essa relação, viabilizando a análise do controle de movimento do robô e de seu efetuador, conforme discutido em [4]. Por meio da Jacobiana, torna-se possível realizar o planejamento e a execução de trajetórias diversas, identificar singularidades, calcular as equações dinâmicas de movimento, entre outros aspectos cruciais para o controle cinemático.

Expressa pela equação (1), a relação entre as velocidades linear e angular do efetuador, representadas por $\vec{p}$ e $\vec{w}$, respectivamente, está vinculada à velocidade das juntas do robô $\dot{\theta}$ por meio da Jacobiana $J \in \Re^{nxm}$. Nessa equação, $n$ refere-se às velocidades angular e linear nos eixos $x, y, z$, enquanto $m$ representa o número de juntas do robô.

$$\begin{pmatrix} \vec{p} \\ \vec{w} \end{pmatrix} = J(\theta)\dot{\theta} \tag{1}$$

Considerando que a Jacobiana $J \in R^{nxn}$ é não singular, podemos realizar a cinemática inversa para determinar os valores das velocidades das juntas a partir das velocidades no efetuadores. Desta forma, a equação (2) é dada por

$$u = J^{-1}v, \tag{2}$$

onde v é dado por $v = \dot{x_d} + K_p(x_d - x)$. A equação (2) é utilizada para realizar a malha de controle cinemático, que será explicado mais a frente.

## 2.2 Malhas de controle

Conforme destacado por [4], o controle cinemático pode ser subdividido em duas perspectivas distintas: controle no espaço de trabalho e controle no espaço das juntas.

Na primeira abordagem, o controlador recebe como entrada a trajetória desejada e aplica ao robô as velocidades das juntas. Este, por sua vez, utiliza essas velocidades para a movimentação desejada. Notavelmente, o controlador, ilustrado na Figura 2, emprega uma ação feedforward (controle antecipatório) para atender às restrições temporais da trajetória.

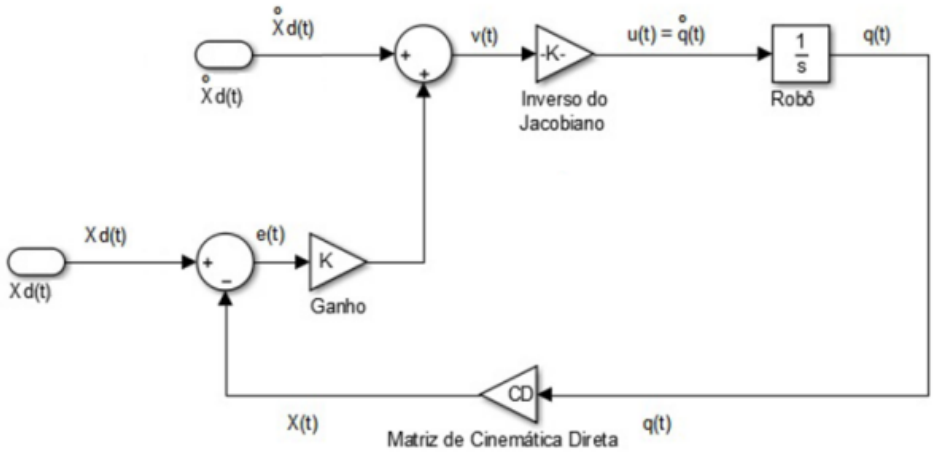


Figura 2: Malha de controle no espaço de trabalho com ação feedforward

Na segunda abordagem, o controlador é concebido para ajustar a trajetória das juntas, assegurando o seguimento da trajetória do efetuador. Entretanto, essa aplicação apresenta desvantagens em comparação com a primeira, uma vez que nem sempre existem soluções viáveis para as equações de cinemática inversa, essenciais para sua implementação. Para cada junta, uma malha de controle individual é estabelecida, como exemplificado na Figura 3.

Após a escolha entre os dois tipos de controle mencionados anteriormente, o problema de controle em robótica pode ser subdividido em dois aspectos: o problema de regulação ou caminho e o problema de trajetória.

O problema de caminho é considerado o mais simples entre os dois. Nesse

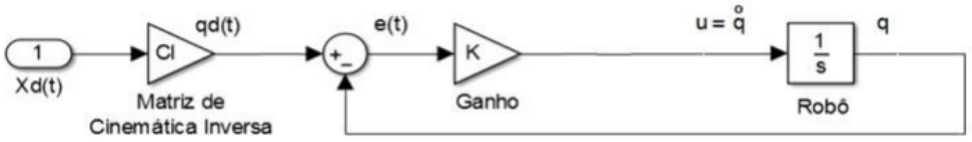


Figura 3: Malha de controle no espaço das juntas

contexto, o objetivo do controle cinemático é levar o robô a uma posição inicial de referência, sem restrição temporal. Dessa forma, o controlador busca reduzir o erro de posição para zero, mesmo que o tempo necessário para isso seja teoricamente infinito.

Por outro lado, o problema de trajetória introduz uma restrição temporal para cada posição que o robô deve atingir. Assim, surge uma trajetória composta (caminho + restrição temporal) que o robô deve percorrer.

## 2.3 Prova de estabilidade: problema de regulação

Considerando um efetuador padrão, cujas velocidades das juntas são descritas por $\dot{x} = J(\theta)\dot{q}$, podemos definir o erro de pose como $\tilde{x} \stackrel{\Delta}{=} x_{desejada} - x_{atual}$. Se a pose desejada for constante, ao derivar o erro em relação ao tempo, a equação torna-se $\dot{\tilde{x}} = -\dot{x}$.

O modelo de transferência de um robô comum pode ser representado por um modelo de integrador simples, onde $u \stackrel{\Delta}{=} \dot{q}$.

Ao empregar um controlador puramente proporcional, o sinal de controle pode ser expresso pela equação (3),

$$u = J^{-1}\lambda\tilde{x}, \tag{3}$$

onde $\lambda$ representa o ganho do controlador.

Ao aplicar o controlador ao robô, obtemos $\dot{\tilde{x}} = -JJ^{-1}\lambda\tilde{x}$, resultando em $\tilde{x}(t) = \tilde{x}(0)e^{-\lambda t}$. Para um valor positivo de $\lambda$, essa equação que descreve o erro converge para zero conforme $t$ se aproxima do infinito.

### 2.4 Prova de estabilidade: seguimento de trajetória

A análise de estabilidade para o problema de seguir uma trajetória segue a mesma linha de raciocínio utilizada para o problema de regulação. No entanto, neste contexto, não é possível assumir que a pose desejada seja constante, pois ela varia ao longo do tempo. Portanto, o erro é definido como $\dot{\tilde{x}} = \dot{x}_{desejada} - \dot{x}$. Será utilizado um controlador com ação feedforward, conforme explicado anteriormente, e a ação de controle será descrita pela equação (4).

$$u = J^{-1}(\lambda\tilde{x} + \dot{x}_{dsejado}) \tag{4}$$

Ao aplicar o controlador ao robô, é obtido $\dot{\tilde{x}} = \dot{x}_{desejado} - JJ^{-1}(\lambda\tilde{x} + \dot{x}_{desejado})$. Assim, a expressão $\tilde{x}(t) = \tilde{x}(0)e^{-\lambda t}$ é novamente alcançada, na qual, para um $\lambda$ positivo, o erro converge para zero à medida que $t$ se aproxima do infinito.

## 3 Objetivos do Trabalho

Este projeto tem como metas:

a. Modelar o robô Comau SmartSix no Robotics Toolbox, seguindo a convenção de Denavit-Hartenberg.

b. Posicionar o robô na configuração inicial especificada pelas posições das juntas $q = [0°; 0°; -90°; 0°; -90°; 0°]$ e gerar uma representação gráfica para validar o modelo.

c. A partir da configuração inicial, movimentar o robô para uma sequência de posições constantes em milímetros, definidas por:

- $P0 = [700; 0; 700]$
- $P1 = [1000; -600; 300]$
- $P2 = [1000; -600; 1000]$
- $P3 = [1000; 600; 1000]$
- $P4 = [1000; 600; 300]$

Mantendo uma orientação constante, com o eixo $Z_e$ do efetuador alinhado ao eixo $X_b$ da base e o eixo $X_e$ do efetuador na direção oposta ao eixo $Z_b$ da base.

d. Partindo da posição P4, comandar o robô para realizar dois círculos no plano YZ, no sentido anti-horário, com centro em Pcentro = [1000; 0; 650] e raio de 350 mm. O primeiro ponto do círculo utilizado como referência deve ser $P_{circle}(4) = [1000; 600; 300]$, mantendo a orientação constante de referência Rd indicada em b.
Metodologia

# 4 Modelagem do Robô

Para iniciar a modelagem do robô, a matriz de parâmetros de Denavit-Hatenberg do manipulador COMAU SmartSix foi definida, utilizando os parâmetros previamente calculados conforme os eixos indicados na Figura 4. Os parâmetros resultantes são apresentados na Tabela 1.

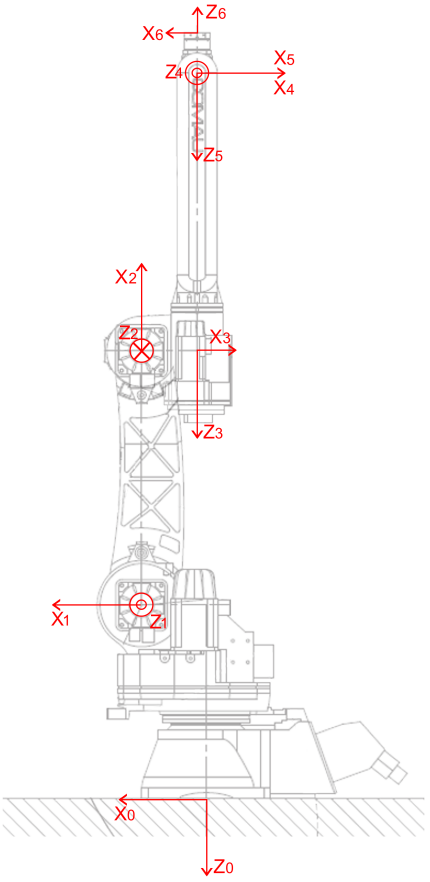


Figura 4: Modelo do robô com eixos de acordo com Denavit-Hatenberg

| Junta | $\theta$ | $d$ | $a$ | $\alpha$ |
|---|---|---|---|---|
| 1 | 0 | -0.45 | 0.15 | $\pi/2$ |
| 2 | 0 | 0 | 0.59 | $\pi$ |
| 3 | 0 | 0 | 0.13 | $-\pi/2$ |
| 4 | 0 | -0.6471 | 0 | $-\pi/2$ |
| 5 | 0 | 0 | 0 | $\pi/2$ |
| 6 | 0 | -0.095 | 0 | $\pi$ |

Tabela 1: Parâmetros de DH obtidos.

Com os parâmetros, o robô foi modelado utilizando o toolbox Robotics ToolBox do Matlab. O robô foi posicionado na configuração inicial $q_{inicial} = [0; 0; -90°; 0; -90°; 0]$. A Figura 5 ilustra o robô modelado na configuração $q_{inicial}$.

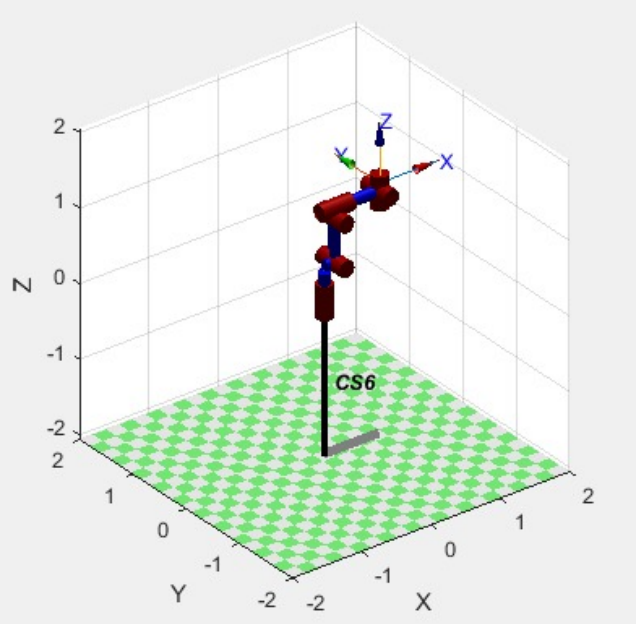


Figura 5: Robô Comau SmartSix modelado no Matlab.

# 5 Malhas de Controle

## 5.1 Controle de Regulação

A próxima etapa foi definir a malha de controle de regulação para rastrear pontos específicos. Os pontos [0.7;0;0.7], [1;-0.6;1], [1;-0.6;0.3], [1;0.6;0.3], [1;0.6;1] formam um quadro, sendo o último ponto o início da próxima trajetória. Referente ao codígo, o controle de regulação foi implementado com dois *loops*, o primeiro percorrendo os pontos desejados $P[0]$ a $P[4]$.

No segundo *loop*, a jacobiana geométrica $J$, o erro $E$ (diferença entre a pose do robô e a pose desejada), e o ganho $K = 2$ foram utilizados para calcular o sinal de controle por meio da relação:

$$u = J^{-1} \cdot K \cdot e \tag{5}$$

A inversa $J^{-1}$ da jacobiana geométrica foi empregada. O sinal de controle foi multiplicado por 0.1 (Regra do Trapézio), e somado ao valor da posição atual até que o erro fosse menor que a margem de erro de $\epsilon = e^{-}3$.

## 5.2 Controle de Trajetória

A terceira parte envolve colocar o robô em uma trajetória circular, repetindo-a duas vezes. Foi implementada uma malha de controle de trajetória. Uma matriz de rotação de referência $R_d$ foi definida como:

$$R_d = \begin{vmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{vmatrix}$$

Para seguir as especificações, foi realizada uma rotação de $\pi/2$ em relação ao eixo $y$ do sistema de coordenadas original. Após definir $R_d$, a geometria do círculo foi declarada, com centro em [1;0;0.650] e raio $r = 0.350$.

Parâmetros de simulação foram definidos como $t_{amostragem} = 0.1$, $t_{simulação}$ e $\omega_n = 2\pi \cdot \frac{1}{t_{amostragem}}$. A trajetória foi dividida em um conjunto de momentos equivalentes às posições para as três voltas que o robô deve dar em um intervalo de 10 segundos cada, respeitando o tempo de amostragem definido.

trajetoria(1) = center(1);
trajetoria(2) = center(2) -radius*sin($\omega_n * n$)
$trajetoria(3) = center(3) - radius * cos(\omega_n * n)$
$pd = trajetoria$

(6)

Derivadas das mesmas :

Dtrajetoria(1) = 0
Dtrajetoria(2) = -radius*wn*cos($\omega_n * n$)
$Dtrajetoria(3) = radius * wn * sin(\omega_n * n)$

(7)

As equações 6 e 7 do círculo foram utilizadas para calcular os valores de $x$, $y$, e $z$ para cada ponto e suas derivadas. Com a jacobiana geométrica do robô para a rotação das juntas e a pose do efetuador calculadas via cinemática direta, a função *rotm2axang* foi utilizada para converter uma matriz de rotação em forma ortonormal para sua representação correspondente em eixo-ângulo.

Os erros de orientação e posição foram calculados, inseridos em uma matriz $\epsilon$, e o *feedforward* para a trajetória desejada foi obtido. A partir disso, o sinal de controle foi calculado como:

$$u = J^{-1} * (K * \epsilon + \text{feedforward}) \tag{8}$$

Esse sinal de controle, adicionado à posição atual, descreve a próxima posição do robô.

# 6 Resultados

A partir de toda a modelagem e montagem do código, foi feita a execução do programa tanto no ambiente do Matlab quanto no ambiente do CoppeliaSim para visualização dos resultados.

## 6.1 Simulações no Matlab

De acordo com o plano delineado na seção de metodologia, inicialmente foram conduzidos os procedimentos para o controle de regulação, visando atingir os pontos especificados.

Na Figura 6, é possível visualizar o robô em sua posição final, com os pontos desejados destacados, juntamente com as orientações de cada eixo. Um vídeo dinâmico do comportamento do modelo está disponível para visualização **https://youtu.be/oXc2TCGsTjI**

Em seguida, a Figura 7 apresenta o sinal de controle gerado para alcançar as posições desejadas. Os resultados alinham-se com as expectativas, revelando uma variação mais pronunciada nos pontos onde o erro inicial era mais elevado, demandando uma ação de controle mais intensa. Em geral, os movimentos causados pelo sinal de controle exibiram comportamento semelhante, exceto nos estágios iniciais, onde o erro entre a posição final e o primeiro ponto desejado era notavelmente alto.

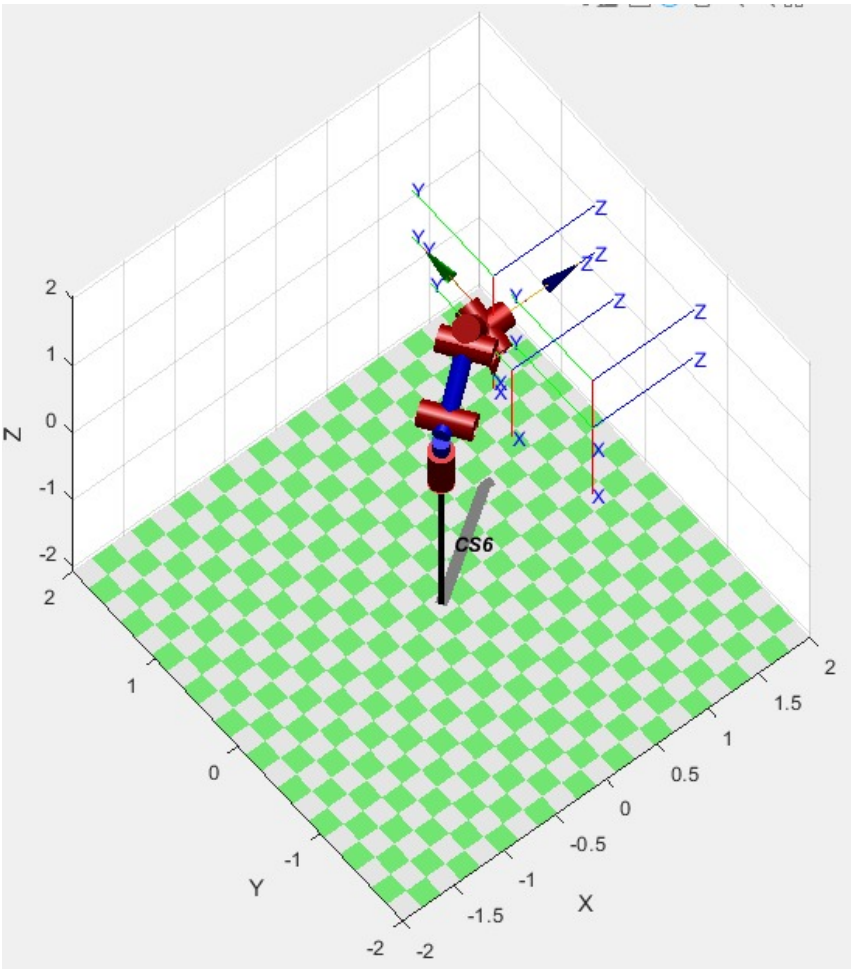


Figura 6: Posição final do robô e pontos demarcados
Imagem ajustada para melhor visualização

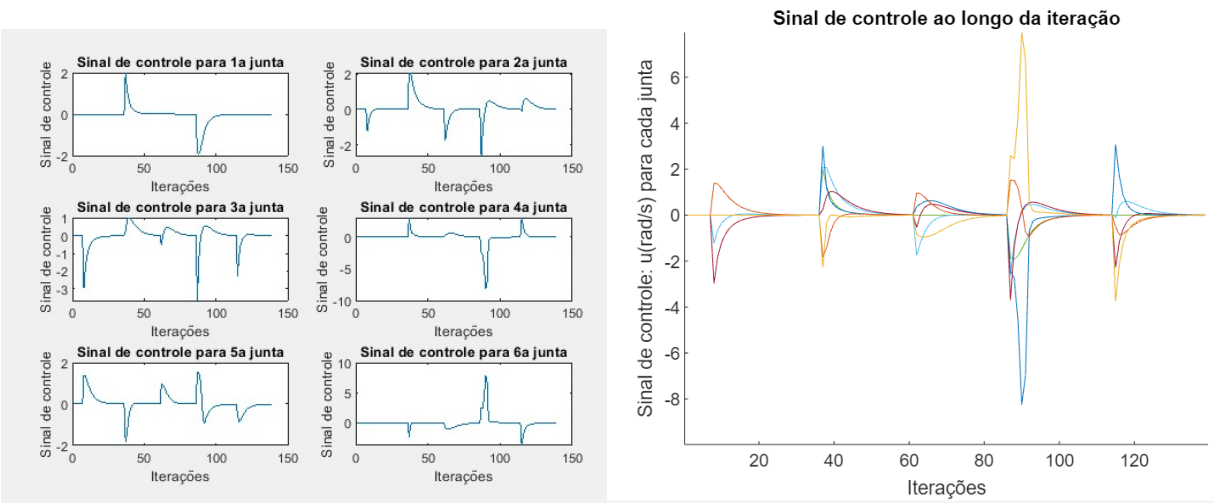


Figura 7: Sinal de controle para rastreamento de posição por juntas

Na Figura 7, foi gerado um gráfico que exibe os sinais de controle de cada junta ao longo da simulação para atingir os pontos desejados. O gráfico ilustra as modificações realizadas em cada junta para alcançar as posições solicitadas, evidenciando as adaptações efetuadas ao longo do processo de controle.

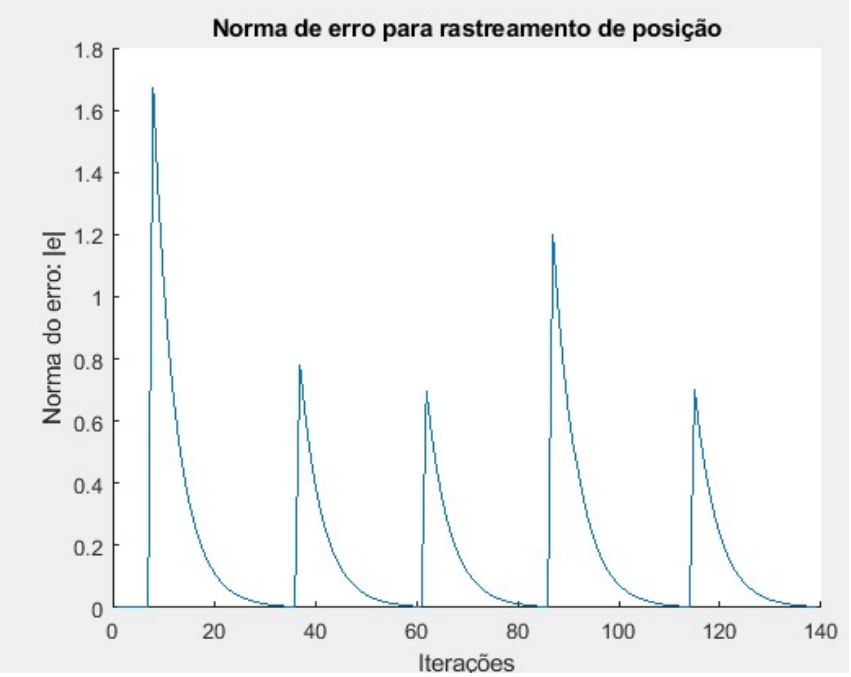


Figura 8: Norma de erro para rastreamento de posição

O fenômeno mencionado anteriormente para as situações iniciais é claramente evidenciado na Figura 7, em que a norma do erro foi plotada na Figura 8, ilustrando sua variação ao longo das iterações. Também é notável acentuadamente nos pontos em que a referência é alterada, resultando em um aumento abrupto do erro.

Além disso, os resultados também retratam a variação do erro de orientação ao longo de cada eixo e em relação à configuração no espaço, como mostrado no gráfico tridimensional da Figura 9

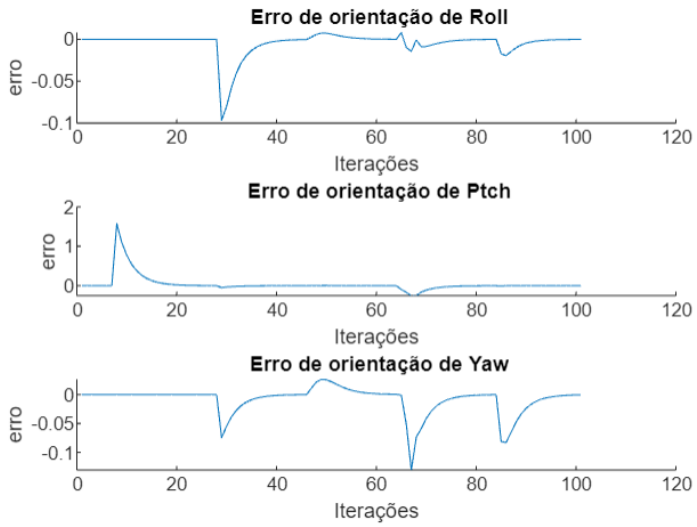


Figura 9: Erro de orientação

Por fim, a trajetória completa do efetuador ao longo do espaço XYZ é visualizada no gráfico tridimensional da Figura 10. Todos os pontos são alcançados com sucesso. Ao final do percurso, o ponto inicial do círculo é atingido, preparando-se para a próxima sequência de movimentos, agora em um controle de seguimento de trajetória.

Num segundo momento, os resultados do controle de seguimento de trajetória foram avaliados de forma separada, focando especificamente no movimento do círculo a ser percorrido pelo efetuador, conforme as especificações de tempo já explicitadas na seção anterior. A metodologia de análise utilizada foi a mesma apresentada anteriormente no controle de regulação. Assim, o primeiro ponto a ser avaliado foi o sinal de controle fornecido ao atuador, conforme descrito na Figura 11

Neste caso, é importante destacar que, como são realizados 2 círculos a partir da mesma posição, os sinais de controle são bastante similares ao longo das iterações. Ao observar as especificações do programa, fica evidente que esse era exatamente o comportamento desejado.

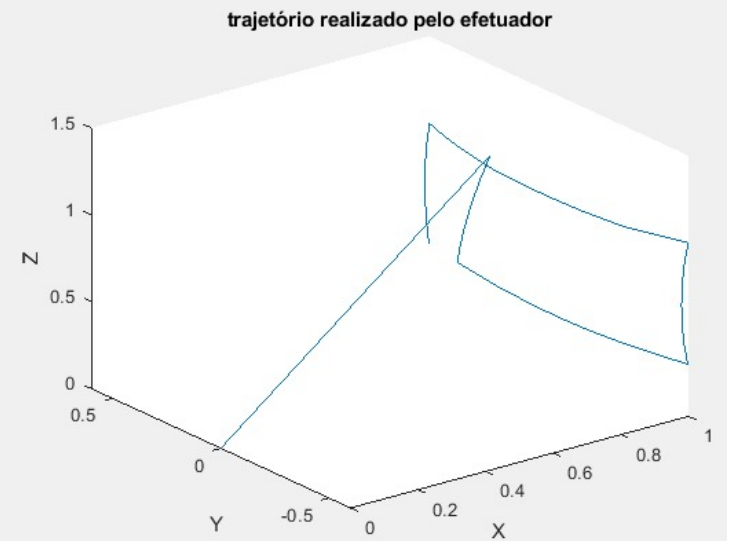


Figura 10: Trajetório realizado pelo o efetuador

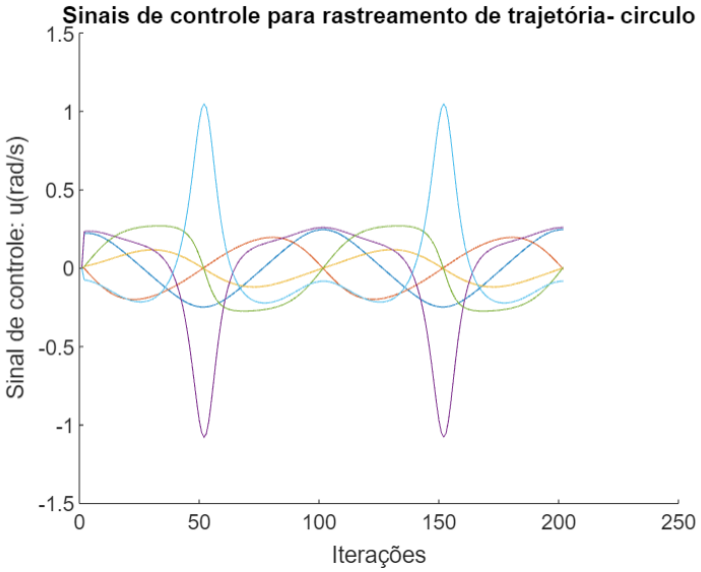


Figura 11: Sinal de controle

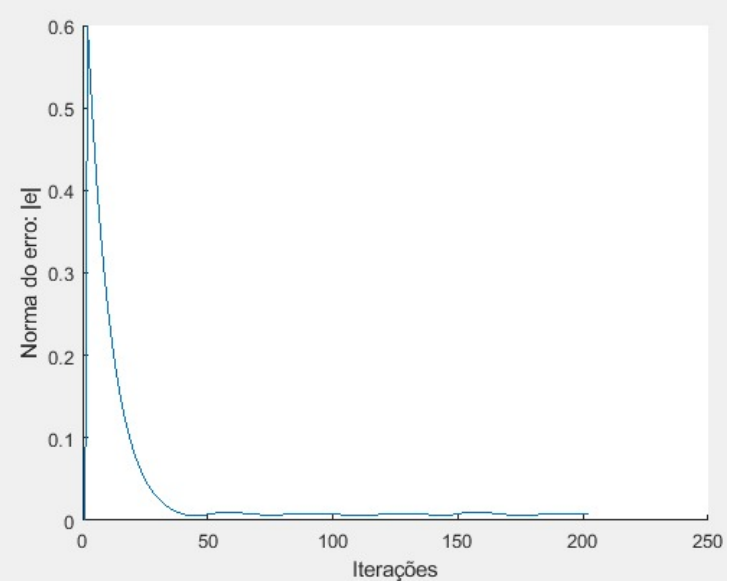


Figura 12: Norma de erro

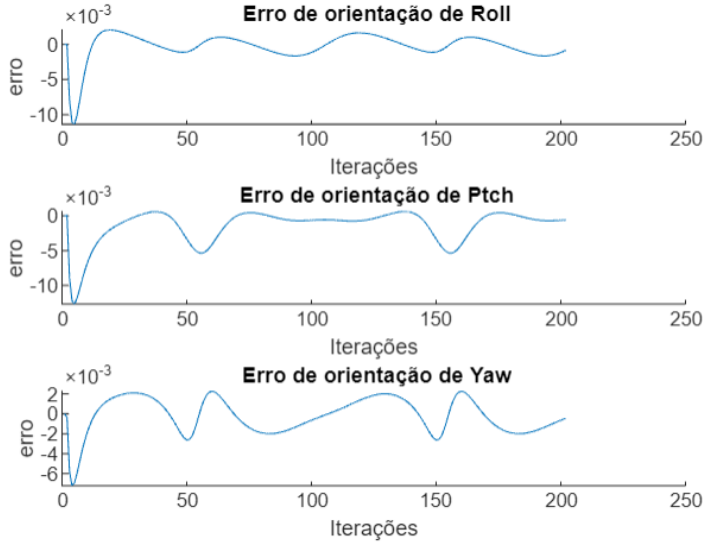


Figura 13: Erro de orientação no trajetório

Em seguida, a análise da norma do erro na Figura 12 fornece informações complementares para explicar a situação descrita acima. Da mesma maneira como o caso de controle de regulação explicitou, há um erro inicial elevado que se estabiliza dentro do limite desejado ao longo do controle. Neste caso, a situação é ainda mais evidente, já que os dois círculos são realizados em torno do mesmo centro.

Além disso, também é possível analisar os erros de orientação em relação à rotação em cada eixo no gráfico tridimensional da Figura 13.

Por fim, o gráfico da trajetória completa pode ser visualizado na Figura 14. Este gráfico é crucial para verificar o desempenho de acordo com as especificações, onde o robô segue exatamente a trajetória circular desejada no plano YZ.

## 6.2 Testes Realizados no CoppeliaSim

O software CoppeliaSim foi empregado de maneira complementar ao Matlab para visualizar os movimentos do robô projetado.


Figura 14: Trajetória realizado pelo o efetuador

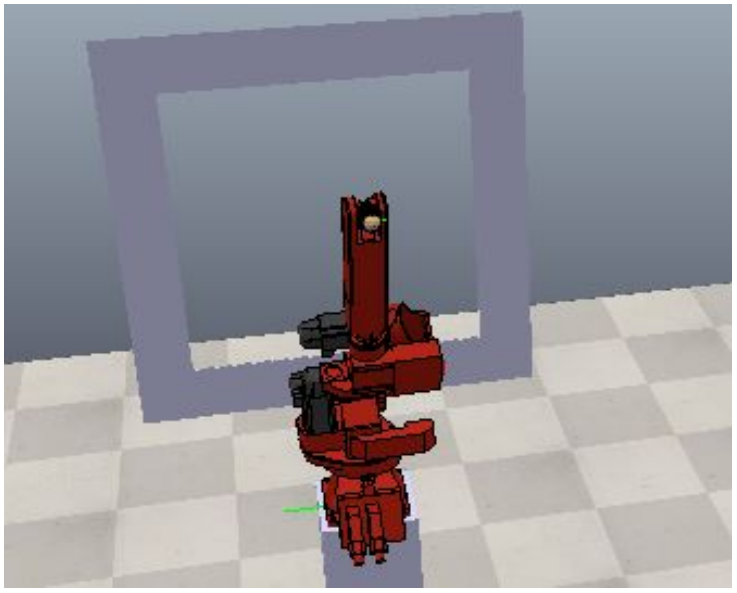

Figura 15: Posição inicial do robô Comau Six

Para possibilitar a integração da simulação, um script foi criado para conectar os dados gerados no Matlab ao CoppeliaSim. Dessa forma, o robô partiu da posição inicial, como mostrado na Figura 15 para executar os movimentos e terminou na posição final, conforme mostrado na Figura 16.

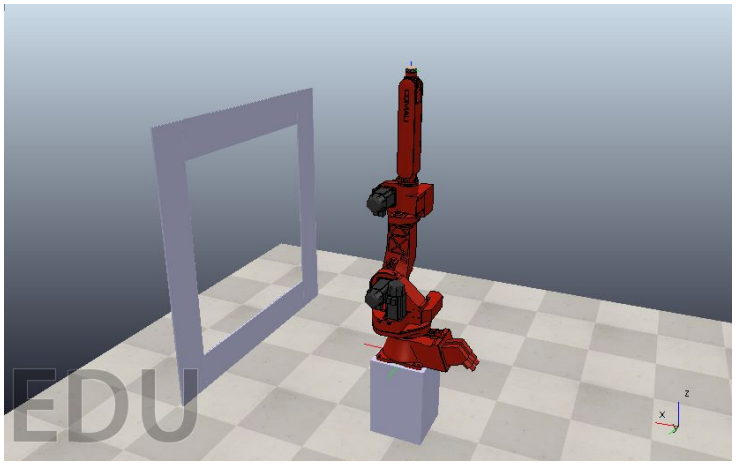


Figura 16: Posição final do robô Comau Six

Ademais, uma simulação em vídeo foi gravada para ilustrar o comportamento durante a dinâmica. Acesse aqui: https://youtu.be/OuS890YHKTg

O vídeo completo da simulação em CoppeliaSim está disponível no link: https://youtu.be/pkOgY581d48

# 7 Conclusão

Este trabalho abordou as bases teóricas necessárias para o projeto de controladores destinados ao seguimento de trajetória e ao rastreamento de referência por regulação. A partir desses conceitos, uma implementação prática foi bem-sucedida, na qual o robô de seis juntas conseguiu rastrear os pontos desejados e seguir a trajetória de uma circunferência. Assim, ambas as metodologias foram validadas.

Posteriormente, uma análise abrangente dos resultados foi conduzida para elucidar o comportamento do robô em ambas as situações. Os gráficos resultantes demonstraram que os pontos foram alcançados conforme as especificações, atendendo aos requisitos de erro e desempenho estabelecidos.

# 8 Bibliografia

## Referências


[1] https://robodk.com/robot/Comau/Smart-SiX-6-1-4.

[2] Professor gustavo. *,Material de aula,desciplina de manipuladores robotica.*

[3] Pedro Sanz. *Robotics: Modeling, planning, and control (siciliano, b. et al; 2009)[on the shelf].* IEEE Robotics Automation Magazine, 16(4):101–101, 2009.

[4] Mark W Spong. *Seth Hutchinson, Mathukumalli Vidyasagar, et al. Robot modeling and control, volume 3. Wiley New York, 2006.*

# 9 Códigos usados para alcançar os resultados



```
clear

% Parametros DH

q = [0 0 (-pi/2) (pi/2) 0 0 pi];

ds = [0 -0.45 0 0 -0.6471 0 -0.095];

as = [0 0.15 0.59 0.13 0 0 0];

alphas = [pi pi/2 pi -pi/2 -pi/2 pi/2 pi];

for i = 1:7

    L(i) = Revolute('offset', q(i), 'd', ds(i), 'a', as(i), 'alpha', alphas(i));

end

levando para a posição inicial

% Levando para a posição inicial

q = [0 0 0 -pi/2 0 -pi/2 0]'; % Posicao incial do robo

robot = SerialLink(L, 'name', 'CS6');

figure(1)

robot.plot(q')



% Controle de Regulação, pontos desejados

pontos = {[0.700 0 0.700], [1 -0.600 0.300], [1 -0.600 1], [1 0.600 1], [1 0.600↙
0.300]};

% O ultimo ponto desejado é o inicio do circulo


Rd_approach = SO3.Ry(pi/2); % Orientação desejada para o approach

Rd_approach = Rd_approach.R(); %realizando a rotação


for c = 1:5

    pd = pontos{c}; % Executa em sequencia o controle de regulação de cada ponto↙
desejado

    Td = SE3(Rd_approach, pd);
```

```
        control_sig(:,i) = u_reg; % Sinal de controle

        juntas_sig(:,i) = q; % Posição Juntas

        erro_orientacao(1:3,i) = nphi_err;
        err(1,i) = norm(erro); % Erro de posicao (escalar)
        caminho(1:3,i) = p_reg; % Caminho Percorrido pelo efetuador
    end
    hold off
end


% Controle Trajetória
Rd = SO3.Ry(pi/2); % Rotacao de referencia
Rd = Rd.R(); % pega matriz de rotacao do efetuador
K = 2; % Define ganho
% Generate circular path
center = [1 0 0.65]; % Center of the circle
radius = 0.35; % Radius of the circle
figure(3);
robot.plot(q'); % Plot robô na configuração inicial
hold on
% Parametros de simulacao escolhidos
t_amostragem = 0.1;
t_simulacao = 10;
wn = pi*2*(1/t_simulacao);
```

```
indice = 1;

for n = 0:t_amostragem:2*t_simulacao

    indice = indice + 1;

    trajetoria(1) = center(1);

    trajetoria(2) = center(2) - radius*sin(wn*n);   % mudando este sinal muda o↙
sentido do circulo.

    trajetoria(3) = center(3) - radius*cos(wn*n); % Eixo z

    pd = trajetoria;

    % Derivadas

    Dtrajetoria(1) = 0;

    Dtrajetoria(2) = -radius*wn*cos(wn*n);  % consequente do sinal, e

    Dtrajetoria(3) = radius*wn*sin(wn*n);

    % Controle

    Joint = robot.jacob0(q); % Jacobiana geométrica

    J = Joint(1:6,2:end);

    T = robot.fkine(q); % Cinemática direta para pegar a pose do efetuador

    p = transl(T); % Translação do efetuador -> posicao atual

    R = SO3(T);

    R = R.R(); % Extrai rotação do efetuador

    p_err = pd - p; % Erro de translação

    nphi = rotm2axang2(Rd*R');

    nphi_err = nphi(1:3)*nphi(4); % Erro de rotação (n*phi)

    erro = [p_err'; nphi_err']; % Vetor de erro

    feedforward = [Dtrajetoria 0 0 0]';

    u = pinv(J)*(K*erro + feedforward); % Lei de controle

    q(2:end) = q(2:end) + t_amostragem*u;

    q_seqC(:,indice) = q(2:end);    % armazonando as posições

    view(3)
```

```
    robot.plot(q');
    control_sig_circulo(:,indice) = u; % Sinal de controle
    juntas_sig_circulo(:,indice) = q; % Posição Juntas
    erro_orientacao_circulo(1:3,indice) = nphi_err; % Erro rotacao
    err_circulo(1,indice) = norm(erro); % Erro de posicao (escalar)
    camimho_circulo(1:3,indice) = p; % Caminho Percorrido pelo efetuador
end
hold off

%Tratamento matriz q_seq (conexão Coppelia)
q_seq = [q_seqP, q_seqC];
q_seq = q_seq*(180/pi); % passar para graus
%% Gráficos
% Controle Regulação
figure(4)
hold on
for b = 1:6
    subplot(3,2,b);
plot(control_sig(b,:))
xlabel('Iterações');
ylabel('Sinal de controle: u(rad/s)');
title("Sinal de controle para "+ b + "a"+ " Junta")

end
hold off



figure(5)
```

```
hold on
plot(err);
xlabel('Iterações')
ylabel('Norma do erro: |e|')
box off
title('Norma de erro para rastreamento de posição')
hold off
a = ["Roll","Ptch", "Yaw"];
figure(6)
hold on
for d = 1:3
    subplot(3,1,d);
    plot(erro_orientacao(d,:));
    xlabel('Iterações');
    ylabel('erro');
    title(' Erro de orientação de '+ a(d))
    box off

end
hold off

figure(7)
view(3)
hold on
plot3(caminho(1,:), caminho(2,:), caminho(3,:));
xlabel('X');
ylabel('Y');
```

```
zlabel('Z');

title('trajetório realizado pelo efetuador')

box off

hold off

Controle de seguimento de trajetório

% Controle Trajetoria

figure(8)

hold on

for l = 1:6

    plot(control_sig_circulo(l,:))

    xlabel('Iterações');

    ylabel('Sinal de controle: u(rad/s)');

    title('Sinais de controle para rastreamento de trajetória- circulo')

end

hold off

figure(9)

hold on

plot(err_circulo);

xlabel('Iterações')

ylabel('Norma do erro: |e|')

box off

hold off

figure(10)

hold on

for s = 1:3

    subplot(3,1,s);

    plot(erro_orientacao_circulo(s,:));

    xlabel('Iterações');
```

```
    ylabel('erro');
    box off
    title(' Erro de orientação de '+ a(s))
    hold off
end
hold off
figure(11)
view(3)
hold on
plot3(camimho_circulo(1,:), camimho_circulo(2,:), camimho_circulo(3,:));
xlabel('X');
ylabel('Y');
zlabel('Z');
box off
title('trajetória realizado pelo efetuador')
hold off
```