TRABALHO 1

Relatório do Trabalho 1 do Curso de Robótica
Universidade Federal do Rio de Janeiro
Campus Centro de Tecnologia

Yure Freitas Pereira de Souza - 122042647

Rio de Janeiro
27 de dezembro de 2025

Sumário

Lista de Figuras

Lista de Tabelas

1 Denavit-Hartenberg do Braço Robótico KR30

Para determinar os parâmetros de Denavit-Hartenberb do braço KR30, primeiro precisamos saber a disposição do eixo de cada manipulador. Para isso, será utilizado os parâmetros de URDF para desenvolver uma representação do manipulador afim de ter uma visalização clara e simples do eixo de cada junta e em seguida, dispor os sistemas de coordenadas seguindo os critérios de DH.

Na tabela abaixo se encontra os parâmetros de URDF obtidos dos arquivos *kr30.urdf* e *torch_fronius_kr30.urdf*:

Junta	**Pai**	**Filho**	**xyz**	**rpy**	**Tipo**	**Eixo**
0	world	0	0, 0, 0	0, 0, 0	fixo	-
1	0	1	0, 0, 0.575	0, 0, 0	R	0, 0, -1
2	1	2	0.175, 0, 0	0, 0, 0	R	0, 1, 0
3	2	3	0.89, 0, 0	0, 0, 0	R	0, 1, 0
4	3	4	0, 0, 0.05	0, 0, 0	R	-1, 0, 0
5	4	5	1.035, 0, 0	0, 0, 0	R	0, 1, 0
6	5	6	0.185, 0, 0	0, $\frac{\pi}{2}$, 0	R	0, 0, -1
torch	6	t	0.000361, 0, 0.461877	0, 33.86º, 0	fixo	0, 0, -1

Tabela 1: URDF do Braço Robótico KR30

Com base nesses dados montamos uma representação visual do manipulador usando figuras simples, assumindo que o sistema de coordenadas de cada junra se encotra extamente no seu centro.

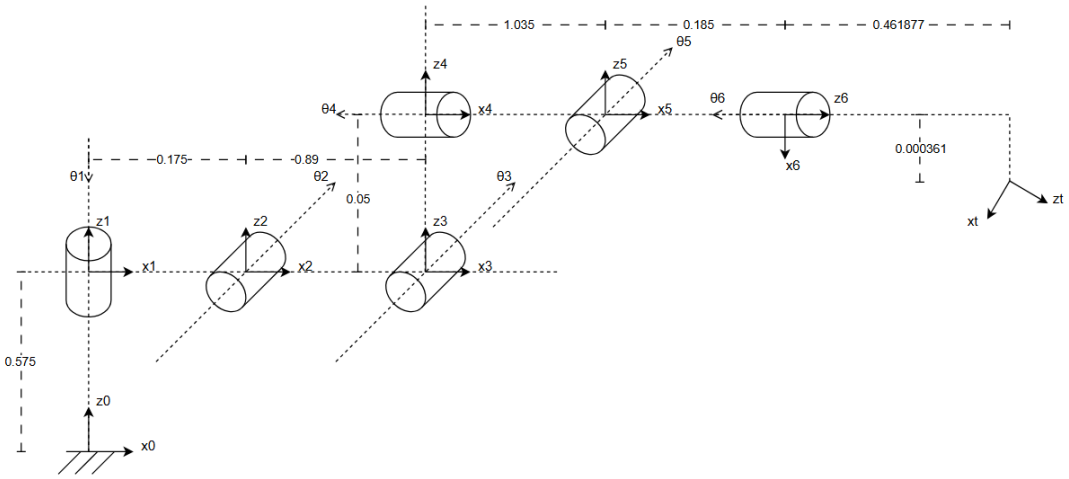


Figura 1: Representação do Braço Robótico KR30 através do URDF

A partir dessa representação e com base nos criterios de DH Standart para os sistemas de coordenadas, organização os sistemas do manipulador indicados na figura a seguir.

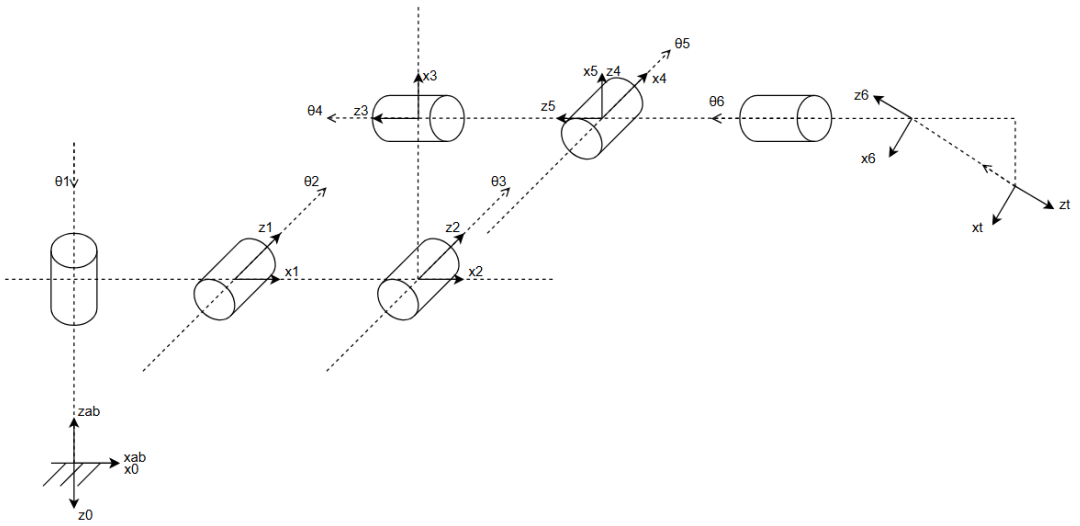


Figura 2: Representação do Braço Robótico KR30 através do DH

Onde, através dela recolhemos os parâmetros de Denavit-Hartenberb para o braço KR30 e as transformações homogêneas do elo 0 em relação à base (T_0^{ab}) e do efetuador em relação ao elo 6 (T_t^6).

Elo	θ	**d**	**a**	α	**Tipo**	**Offset**
1	θ_1	-0.575	0.175	$\frac{\pi}{2}$	R	0
2	θ_2	0	0.89	0	R	0
3	θ_3	0	0.05	$\frac{\pi}{2}$	R	$-\frac{\pi}{2}$
4	θ_4	-1.035	0	$-\frac{\pi}{2}$	R	0
5	θ_5	0	0	$\frac{\pi}{2}$	R	0
6	θ_6	d_{6DH}	0	-33.86º	R	-$\frac{\pi}{2}$

Tabela 2: DH do Braço Robótico KR30

$$T_0^{ab} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_t^6 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & z_{6t} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Sendo:

$$d_{6Dh} = \frac{0.000361}{tg(33.86°)} - 0.461877 - 0.185 \tag{1}$$

$$z_{6t} = -\frac{0.000361}{sen(33.86°)} \tag{2}$$

2 Denavit-Hartenberg da Mesa Posicionadora DKP400

O mesmo processo foi feito para a mesa posicionadora usado os dados recolhidos do arquivo *dkp400.urdf*, que podem ser vistos na tabela abaixo.

Junta	**Pai**	**Filho**	**xyz**	**rpy**	**Tipo**	**Eixo**
0	world	0	0, 0, 0	0, 0, 0	fixo	-
1	0	1	0, 0, 0.51	0, 0, π	R	-1, 0, 0
2	1	2	0, 0, 0.347	0, 0, 0	R	0, 0, 1
tool	2	tool	0, 0, 0	0, 0, 0	fixo	-

Tabela 3: URDF da Mesa Posicionadora DKP400

Com esses dados temos a representação dos sistemas de coordenadas.

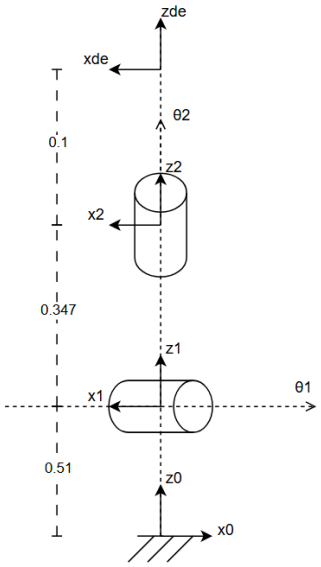


Figura 3: Representação da Mesa Posicionadora através do URDF

E a partir deles fizemos a distribuição dos sistemas de cooredenadas seguindo os critérios de DH Standard.

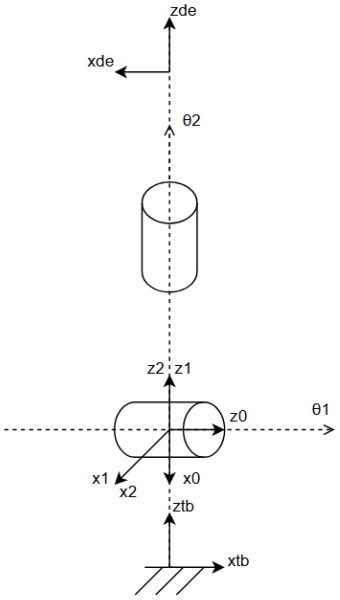


Figura 4: Representação da Mesa Posicionadora DKP400 através do DH

Com eles é possivel recolher os parâmetros de DH e as trasnformações homogêneas do elo 0 em relação à base (T_0^{tb}) e da superficie da mesa em relação ao elo 2 (T_{de}^2), como pode ser observado na tabela a seguir.

Elo	θ	**d**	**a**	α	**Tipo**	**Offset**
1	θ_1	0	0	$\frac{\pi}{2}$	R	$-\frac{\pi}{2}$
2	θ_2	0	0	0	R	0

Tabela 4: DH da Mesa Posicionadora DKP400

$$T_0^{tb} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0.51 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_{de}^2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.447 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3 Denavit-Hartenberg do Sistema KR30-DKP400

Para os parâmetros de Denavit-Hartenberg do sistema braço-mesa KR30-DKP400, utilziamos as representações de cada sistema individual e a transformação homogênea da base da mesa em relação a base do braço para desenvolvermos uma representação do sistema conjunto.

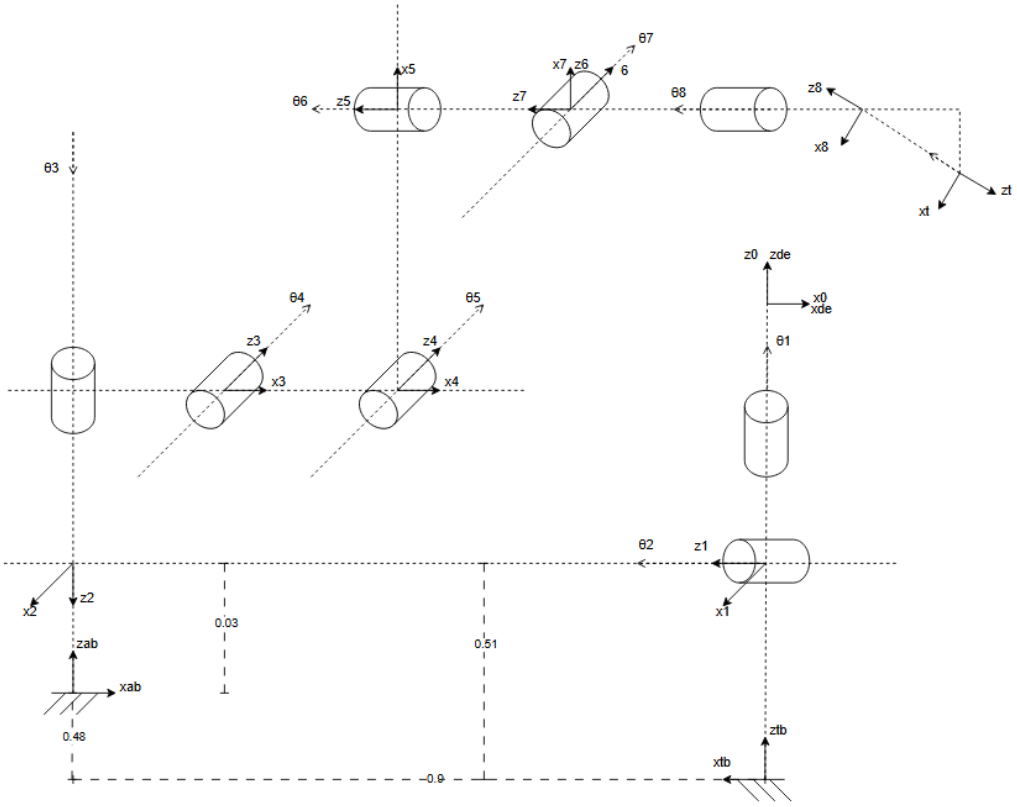


Figura 5: Representação do Sistema KR30-DKP400

Com base nessa representação, retiramos os parâmetros de DH do sistema de coordenadas do efetuador do braço em relação ao efetuador da mesa, seguindo os criterios de DH Standard para a alocação dos sistemas de coordenadas, resultando na tabela abaixo.

Elo	θ	**d**	**a**	α	**Tipo**	**Offset**
1	θ_1	-0.447	0	$\frac{\pi}{2}$	R	$-\frac{\pi}{2}$
2	θ_2	0.9	0	$\frac{\pi}{2}$	R	0
3	θ_3	0.545	0.175	$\frac{\pi}{2}$	R	$-\frac{\pi}{2}$
4	θ_4	0	0.89	0	R	0
5	θ_5	0	0.05	$\frac{\pi}{2}$	R	$-\frac{\pi}{2}$
6	θ_6	-1.035	0	$-\frac{\pi}{2}$	R	0
7	θ_7	0	0	$\frac{\pi}{2}$	R	0
8	θ_8	d_{6DH}	0	-33.86º	R	$-\frac{\pi}{2}$

Tabela 5: DH do Sistema KR30-DKP400

$$T_0^{de} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_t^8 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & z_{8t} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Sendo $z_{8t} = z_{6t}$ da equação 2.

4 Modelos para os Parametros de DH

4.1 Braço Robótico KR30

```
clear; clc; close all;

% Definição dos elos (DH Standard)
L(1) = Link('d', -0.575, 'a', 0.175, 'alpha', pi/2, 'offset', 0);
L(2) = Link('d', 0, 'a', 0.89, 'alpha', 0, 'offset', 0);
L(3) = Link('d', 0, 'a', 0.05, 'alpha', pi/2, 'offset', -pi/2);
L(4) = Link('d', -1.035, 'a', 0, 'alpha', -pi/2, 'offset', 0);
L(5) = Link('d', 0, 'a', 0, 'alpha', pi/2, 'offset', 0);
L(6) = Link('d', -0.646339, 'a', 0, 'alpha', -deg2rad(33.89), 'offset', -pi/2);

% Criação do robô
robot = SerialLink(L, 'name', 'KR30');

% Transformação Base -> Frame 0
robot.base = transl(0.0, 0.0, 0.0) * trotx(pi);

% Transformação Elo final -> Efetuador
robot.tool = transl(0, 0, -0.000648) * (trotx(-pi)*trotz(pi/2));

% Configuração das juntas
q = [0 0 0 0 0 0];

T = robot.fkine(q);

% Plot
robot.plot(q);
robot.teach;
```

4.2 Mesa Posicionadora DKP400

```
clear; clc; close all;

% Definição dos elos (DH Standard)
L(1) = Link('d', 0, 'a', 0, 'alpha', pi/2, 'offset', -pi/2);
```

```
L(2) = Link('d', 0, 'a', 0, 'alpha', 0, 'offset', 0);

% Criação do robô
robot = SerialLink(L, 'name', 'DKP400');

% Transformação Base -> Frame 0
robot.base = transl(0.0, 0.0, 0.51) * troty(pi/2);

% Transformação Elo final -> Efetuador
robot.tool = transl(0, 0, 0.447) * trotz(-pi/2);

% Configuração das juntas
q = [0 0];

T = robot.fkine(q);

% Plot
robot.plot(q);
robot.teach;
```

4.3 Sistema KR30-DKP400

```
clear; clc; close all;

% Definição dos elos (DH Standard)
% Elos da mesa
L(1) = Link('d', -0.447, 'a', 0, 'alpha', pi/2, 'offset', -pi/2);
L(2) = Link('d', 0.9, 'a', 0, 'alpha', pi/2, 'offset', 0);
% Elos do braço
L(3) = Link('d', -0.545, 'a', 0.175, 'alpha', pi/2, 'offset', -pi/2);
L(4) = Link('d', 0, 'a', 0.89, 'alpha', 0, 'offset', 0);
L(5) = Link('d', 0, 'a', 0.05, 'alpha', pi/2, 'offset', -pi/2);
L(6) = Link('d', -1.035, 'a', 0, 'alpha', -pi/2, 'offset', 0);
L(7) = Link('d', 0, 'a', 0, 'alpha', pi/2, 'offset', 0);
L(8) = Link('d', -0.646339, 'a', 0, 'alpha', -deg2rad(33.89), 'offset', -pi/2);

% Criação do robô
```

```
16 robot = SerialLink(L, 'name', 'KR30-DKP400');
17
18 % Transformação Base -> Frame 0
19 robot.base = transl(0.0, 0.0, 0.0) * trotx(0);
20
21 % Transformação Elo final -> Efetuador
22 robot.tool = transl(0, 0, -0.000648) * (trotx(-pi)*trotz(pi/2));
23
24 % Configuração das juntas
25 q = [0 0 0 0 0 0 0 0];
26
27 T = robot.fkine(q);
28
29 % Plot
30 robot.plot(q);
31 robot.teach;
```

5 Validação dos Parâmetros de Denavit-Hartenberg

Para validar os moddelos de DH que desenvolvemos no anteriormente, iremos comparar os valores de posição e orientação de cada modelo com a sua respectiva transformação homogênea, feita a partir dos parâmetros de URDF.

Então geremos quatro valores aleatórios para cada junta dos dois sistemas, dentro do limete estabelecido nos arquivos *kr30.urdf* e *dkp400.urdf*, tem-se as seguintes inclinações das juntas, em radiano:

$$\theta^T_{braço} = \begin{bmatrix} -1.2764 & 0.6857 & -1.7073 & 0.3506 & 0.7602 & 1.3329 \\ -0.2073 & -2.9612 & -2.0336 & -2.1091 & -1.3828 & -3.7221 \\ -0.4974 & -1.8194 & 0.5113 & -1.3379 & 1.5663 & -1.6302 \\ -2.4928 & 0.1397 & 2.1512 & 0.1646 & -0.4733 & 3.7511 \end{bmatrix}$$

$$\theta^T_{mesa} = \begin{bmatrix} -0.4612 & -1.4962 \\ -1.3348 & -1.3143 \\ 1.3745 & -1.2494 \\ 1.479 & 0.3826 \end{bmatrix}$$

Usando esses valores nos modelos gereados pelos parâmetros de DH resulta nos seguintes valores de posição (em metros) e orientação (em graus) [x y z ϕ θ ψ]^T, do efetuador em relação a base:

$$\text{Braço: } \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} 0.716 \\ 1.834 \\ 1.102 \\ -13.1 \\ 63.2 \\ -62.8 \end{bmatrix}, \begin{bmatrix} -0.518 \\ -0.666 \\ -0.452 \\ 130.7 \\ 39.6 \\ 83.4 \end{bmatrix}, \begin{bmatrix} -0.018 \\ 0.706 \\ 2.414 \\ -155.2 \\ -47.3 \\ -79.2 \end{bmatrix}, \begin{bmatrix} -0.229 \\ 0.113 \\ -0.985 \\ 174.9 \\ -25.0 \\ 164.9 \end{bmatrix}$$

$$\text{Mesa: } \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} 0.000 \\ 0.199 \\ 0.910 \\ 94.3 \\ 0.0 \\ -26.4 \end{bmatrix}, \begin{bmatrix} 0.000 \\ 0.435 \\ 0.615 \\ 104.7 \\ 0.0 \\ -76.5 \end{bmatrix}, \begin{bmatrix} 0.000 \\ -0.438 \\ 0.597 \\ 108.4 \\ 0.0 \\ 78.8 \end{bmatrix}, \begin{bmatrix} 0.000 \\ -0.445 \\ 0.551 \\ -158.1 \\ 0.0 \\ 84.7 \end{bmatrix}$$

$$\text{Braço-Mesa: } \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} -0.579 \\ -0.753 \\ 1.462 \\ -49.4 \\ 46.2 \\ 53.2 \end{bmatrix}, \begin{bmatrix} -0.042 \\ 1.448 \\ -1.214 \\ -136.5 \\ -24.9 \\ -142.4 \end{bmatrix}, \begin{bmatrix} 1.821 \\ -1.298 \\ 0.976 \\ -67.3 \\ -13.4 \\ 46.3 \end{bmatrix}, \begin{bmatrix} 0.170 \\ -1.149 \\ -1.430 \\ 107.9 \\ 30.4 \\ 147.0 \end{bmatrix}$$

5.1 Validação

Através dos parâmetros de URDF, obtevemos a transformação homogênea entre cada uma das juntas, da base até o efetuador, tanto do sistema do braço quanto do sistema da mesa. Com elas tem-se a T_{0e}, a transformação homogênea do efetuador em relação a base.

Aplicando os valores de $\theta_{braço}$ e θ_{mesa} em suas respectivas transformações:

$$\text{Braço: } \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} 0.7165 \\ 1.8342 \\ 1.1016 \\ -13.0847 \\ 63.2236 \\ -62.7993 \end{bmatrix}, \begin{bmatrix} -0.5175 \\ -0.6663 \\ -0.4521 \\ 130.6580 \\ -39.5718 \\ 83.3866 \end{bmatrix}, \begin{bmatrix} -0.0182 \\ 0.7064 \\ 2.4143 \\ -155.1732 \\ -47.3303 \\ -79.2455 \end{bmatrix}, \begin{bmatrix} -0.2290 \\ 0.1127 \\ -0.9846 \\ 174.9123 \\ -24.9605 \\ 164.8666 \end{bmatrix}$$

$$\text{Mesa: } \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} 0.0000 \\ 0.1989 \\ 0.9103 \\ 94.2741 \\ 0.0000 \\ -26.4248 \end{bmatrix}, \begin{bmatrix} 0.0000 \\ 0.4346 \\ 0.6145 \\ 104.6962 \\ 0.0000 \\ -76.4784 \end{bmatrix}, \begin{bmatrix} 0.0000 \\ -0.4384 \\ 0.5972 \\ 108.4147 \\ 0.0000 \\ 78.7530 \end{bmatrix}, \begin{bmatrix} 0.0000 \\ -0.4451 \\ 0.5510 \\ -158.0786 \\ 0.0000 \\ 84.7405 \end{bmatrix}$$

Comparando os dois resultados para cada um dos sistemas, nota-se que eles são iguais, considerando que as pequenas diferenças decorrem do arredondamento feito pelo modelo de DH. Com isso, concluimos que o modelo de DH é valido.

5.1.1 Código usado para simulação dos valores das juntas

- **Braço KR30**:

```
clear; clc; close all;

q = [ ];

% Transformação junta 0 -> junta 1
T01 = transl(0.0, 0.0, 0.575) * trotz(-q(1));
% Transformação junta 1 -> junta 2
T12 = transl(0.175, 0.0, 0.0) * troty(q(2));
% Transformação junta 2 -> junta 3
T23 = transl(0.89, 0.0, 0.0) * troty(q(3));
% Transformação junta 3 -> junta 4
T34 = transl(0.0, 0.0, 0.05) * trotx(-q(4));
% Transformação junta 4 -> junta 5
T45 = transl(1.035, 0.0, 0.0) * troty(q(5));
% Transformação junta 5 -> junta 6
T56 = transl(0.185, 0.0, 0.0) * (troty(pi/2)*trotz(-q(6)));
% Transformação junta 6 -> efetuador
T6t = transl(0.000361, 0.0, 0.461877) * troty(deg2rad(33.89));

T0t = T01 * T12 * T23 * T34 * T45 * T56 * T6t;

% XYZ
x = T0t(1,4);
y = T0t(2,4);
z = T0t(3,4);

% Extrai matriz de rotação
R = T0t(1:3, 1:3);

% Cálculo RPY
pitch = asin(R(1,3));
yaw   = atan2(-R(2,3), R(3,3));
roll    = atan2(-R(1,2), R(1,1));
```

- **Mesa DKP400**:

```
clear; clc; close all;

q = [ ];

% Transformação junta 0 -> junta 1
T01 = transl(0.0, 0.0, 0.51) * (trotz(pi)*trotx(-q(1)));
% Transformação junta 1 -> junta 2
T12 = transl(0.0, 0.0, 0.347) * trotz(q(2));
% Transformação junta 2 -> efetuador

T2de = transl(0.0, 0.0, 0.1);

T0de = T01 * T12 * T2de;

% XYZ
x = T0de(1,4);
y = T0de(2,4);
z = T0de(3,4);

% Extrai matriz de rotação
R = T0de(1:3, 1:3);

% Cálculo RPY
pitch = asin(R(1,3));
yaw  = atan2(-R(2,3), R(3,3));
roll   = atan2(-R(1,2), R(1,1));
```

6 Cinemática Inversa

Para o cálculo de cinemática inversa usaremos a função *ikine()* do *robot toolbox*, onde trabalharemos com os seguintes parametros:

- Td: transformação homogenea desejada, do efetuador do braço em relação ao efetuador da mesa;
- q_0: valor inicial de inclinação de cada uma das juntas do sistema
- mask: máscara, define a importancia de cada valor de posição e orientação [x y z ϕ θ ψ]^T para o resultado desejado;
- tol: tolerancia para o erro;
- lambda: amortecimento numérico.

Para este caso, deseja-se seguir uma trajetorio em zigzag, mantendo uma orientação fixa de π em roll e $\frac{\pi}{2}$ em yaw:

Waypoint	**Posição** p_{dt}
1	[0.04, 0, 0]^T
2	[-0.04, 0, 0]^T
3	[-0.04, 0.01, 0]^T
4	[0.04, 0.01, 0]^T
5	[0.04, 0.02, 0]^T
6	[-0.04, 0.02, 0]^T
7	[-0.04, 0.03, 0]^T
8	[0.04, 0.03, 0]^T
9	[0.04, 0.04, 0]^T
10	[-0.04, 0.04, 0]^T

Tabela 6: Waponts da Trajetoria

Inicialmente precisamos definir o modelo do manipulador, que seria o modelo do sistema braço-mesa KR30-DKP400, apresentado na seção 4.3. Entretanto, temos que as inclinações juntas da mesa são constantes, ou seja, não precisamos nos precupar em com a cinemática inversa dessas juntas.

Com isso, podemos usar o modelo do braço KR30, apresentado na seção 4.1, e definir a transformação *.base* como sendo a transformação do elo 0 do braço em relação ao efetuador da mesa, usando o modelo da mesa apresentado na seção 4.2, com angulos das juntas fixos, a transformação do elo 0 do braço em relação a sua base e a transformação da base da mesa em relação a base do braço.

Com essas configurações chegamos aos seguintes angulos (em radianos) das juntas do braço:

Waypoint	θ_1	θ_2	θ_3	θ_4	θ_5	θ_5
1	-0.7911	1.1134	-1.7997	1.9726	2.2941	-0.8118
2	-0.8355	1.1611	-1.8709	2.0332	2.3152	-0.7640
3	-0.8392	1.1623	-1.8633	2.0294	2.3216	-0.7730
4	-0.7948	1.1145	-1.7923	1.9689	2.3001	-0.8209
5	-0.7985	1.1155	-1.7847	1.9651	2.3061	-0.8301
6	-0.8428	1.1634	-1.8556	2.0255	2.3280	-0.7822
7	-0.8464	1.1643	-1.8478	2.0215	2.3344	-0.7913
8	-0.8021	1.1164	-1.7771	1.9611	2.3120	-0.8393
9	-0.8057	1.1173	-1.7693	1.9570	2.3179	-0.8486
10	-0.8500	1.1652	-1.8399	2.0173	2.3407	-0.8006

Tabela 7: θ para cada waypoint

Para essas configurações de θ teremos os seguintes valores para posição (em metros) e orientação (em graus):

Waypoint	x	y	z	ϕ	θ	ψ
1	0.0400	0.0000	0.0000	180.0000	0.0000	90.0000
2	-0.0400	0.0000	0.0000	180.0000	0.0000	90.0000
3	-0.0400	0.0100	0.0000	180.0000	0.0000	90.0000
4	0.0400	0.0100	0.0000	180.0000	0.0000	90.0000
5	0.0400	0.0200	0.0000	180.0000	0.0000	90.0000
6	-0.0400	0.0200	0.0000	180.0000	0.0000	90.0000
7	-0.0400	0.0300	0.0000	180.0000	0.0000	90.0000
8	0.0400	0.0300	0.0000	180.0000	0.0000	90.0000
9	0.0400	0.0400	0.0000	180.0000	0.0000	90.0000
10	-0.0400	0.0400	0.0000	180.0000	0.0000	90.0000

Tabela 8: Posicação e orientação para cada waypoint

Possuindo um erro ($\Delta valor = valor_{desejado} - valor_{obtido}$) de:

Waypoint	Δx	Δy	Δz	$\Delta\phi$	$\Delta\theta$	$\Delta\psi$
1	-0.0008	-0.0009	0.0006	0.0252	0.0042	0.0275
2	-0.0010	-0.0012	0.0010	0.0388	0.0042	0.0295
3	-0.0015	-0.0016	0.0010	0.0425	0.0084	0.0434
4	0.0005	0.0000	0.0012	0.0492	-0.0081	-0.0101
6	0.0044	0.0031	0.0039	0.1459	-0.0406	-0.1285
7	-0.0013	-0.0015	0.0012	0.0501	0.0057	0.0370
8	0.0000	-0.0005	0.0019	0.0719	-0.0058	0.0024
9	0.0252	0.0190	0.0173	0.6422	-0.2156	-0.7170
10	0.0003	0.0003	0.0000	0.0009	-0.0030	-0.0095
1	0.0042	0.0030	0.0061	0.2154	-0.0388	-0.1277

Tabela 9: Erro para posicação e orientação para cada waypoint ($\cdot 10^{-5}$)

6.1 Código Usado para Cálcular a Cinemática Inversa

```
clear; clc; close all;

% Definição dos elos da mesa (DH Standard)
Lt(1) = Link('d', 0, 'a', 0, 'alpha', pi/2, 'offset', -pi/2);
Lt(2) = Link('d', 0, 'a', 0, 'alpha', 0, 'offset', 0);

% Criação da mesa
DKP400 = SerialLink(Lt, 'name', 'DKP400');

% Transformação Base -> Frame 0
DKP400.base = transl(0.0, 0.0, 0.51) * troty(pi/2);

% Transformação Elo final -> Efetuador
DKP400.tool = transl(0, 0, 0.447) * trotz(-pi/2);

% Configuração das juntas da mesa
theta_t = [pi/4 0];

Ttb_de = DKP400.fkine(theta_t);

% Transformação Base do Braço -> Base da Mesa
Tab_tb = transl(0.9, 0, -0.48) * trotz(pi);
Tab_tb = SE3(Tat_tb);
```

```
24
25  % Transformação Base do Braço -> Efetuador da Mesa
26  Tab_de = Tab_tb * Ttb_de;
27
28  % Transformação Efetuador da Mesa -> Base do Braço
29  Rab_de = Tab_de.R;
30  pab_de = Tab_de.t;
31
32  Tab_de_inv = [Rab_de.' -Rab_de.'*pab_de;
33                0 0 0  1];
34
35  % Definição dos elos do braço (DH Standard)
36  La(1) = Link('d', -0.575, 'a', 0.175, 'alpha', pi/2, 'offset', 0);
37  La(2) = Link('d', 0, 'a', 0.89, 'alpha', 0, 'offset', 0);
38  La(3) = Link('d', 0, 'a', 0.05, 'alpha', pi/2, 'offset', -pi/2);
39  La(4) = Link('d', -1.035, 'a', 0, 'alpha', -pi/2, 'offset', 0);
40  La(5) = Link('d', 0, 'a', 0, 'alpha', pi/2, 'offset', 0);
41  La(6) = Link('d', -0.646339, 'a', 0, 'alpha', -deg2rad(33.89), 'offset', -pi/2);
42
43  % Criação do braço
44  KR30 = SerialLink(La, 'name', 'KR30');
45
46  % Transformação Efetuador da Mesa -> Frame 0
47  KR30.base = Tade_inv * (transl(0.0, 0.0, 0.0) * trotx(pi));
48
49  % Transformação Elo final -> Efetuador
50  KR30.tool = transl(0, 0, -0.000648) * (trotx(-pi)*trotz(pi/2));
51
52  %=====================================================================
53  % Cinemática inversa
54  %=====================================================================
55
56  p = [ 0.04 0 0;
57       -0.04 0 0;
58       -0.04 0.01 0;
59        0.04 0.01 0;
60        0.04 0.02 0;
```

```
61        -0.04 0.02 0;
62        -0.04 0.03 0;
63         0.04 0.03 0;
64         0.04 0.04 0;
65        -0.04 0.04 0 ];
66
67  R = trotz(pi/2) * trotx(pi);
68  Td = zeros(4, 4, size(p,1));
69
70  for i = 1:size(p,1)
71      Td(:,:,i) = transl(p(i,:)) * R;
72  end
73
74  mask = [1 1 1 1 1 1];
75
76  tol = 1e-6;
77
78  lambda = 1;
79
80  q_sol = zeros(size(p,1), 6);
81
82  % chute inicial
83  q0 = zeros(1,6);
84
85  for i = 1:size(p,1)
86      q = KR30.ikine(Td(:,:,i), q0, 'mask', mask, 'tol', tol, 'lambda', lambda);
87
88      q_sol(i,:) = q;
89
90      q0 = q;
91  end
92
93  xyz = zeros(size(q_sol,1), 3);
94  rpy = zeros(size(q_sol,1), 3);
95
96  for i = 1:size(q_sol,1)
97      T = KR30.fkine(q_sol(i,:));
```

```
98      xyz(i,:) = T.t';
99      rpy(i,:) = rad2deg(tr2rpy(T));
100 end
```

7 Jacobiano

Para sabermos o jacobiano do efetuador do braço em relação ao efeduador da mesa (J_t^{de}), usaremos a relação das velocidades, onde a velocidade relativa é iqual à diferença entre a velocidade do efetudador braço e a velocidade do efetudador da mesa, ou seja:

$$V_t^{de} = (V_t^{de})_{braço} - (V_{de}^{de})_{mesa} \tag{3}$$

Onde, sabendo os jacobianos dos efetuadores em relação às suas bases (J_{de}^{tb} e J_t^{ab}), usaramos:

$$(V_t^{de})_{braço} = Ad_{T_{ab}^{de}} J_t^{ab} \dot{\theta}_{braço} \tag{4}$$

$$(V_{de}^{de})_{mesa} = Ad_{T_{tb}^{de}} J_{de}^{tb} \dot{\theta}_{mesa} \tag{5}$$

Como ja sabemos T_{de}^{tb}, fica mais fácil encontrar $(V_{de}^{de})_{mesa}$ pois, $Ad_{T_{tb}^{de}} = Ad_{(T_{de}^{tb})^{-1}}$, ou seja.

$$(V_{de}^{de})_{mesa} = Ad_{(T_{de}^{tb})^{-1}} J_{de}^{tb} \dot{\theta}_{mesa} \tag{6}$$

Porém, para sabermos o $(V_t^{de})_{braço}$, primeiro precisamos encontrar a transformação da base do braço em relação ao efetuador da mesa (T_{ab}^{de}), para isso usamos a transformações da base do braço em relação à base da mesa (T_{ab}^{tb}) e da base da mesa em relação ao efetudador da mesa (T_{tb}^{de}) da seguinte forma:

$$T_{ab}^{de} = T_{tb}^{de} T_{ab}^{tb} \tag{7}$$

Onde, $T_{tb}^{de} = (T_{de}^{tb})^{-1}$ e $T_{ab}^{tb} = (T_{tb}^{ab})^{-1}$, ou seja:

$$T_{ab}^{de} = (T_{de}^{tb})^{-1} (T_{tb}^{ab})^{-1} = (T_{tb}^{ab} T_{de}^{tb})^{-1} \tag{8}$$

Desta forma conseguiremos calcular o adjunta da transformação da base do braço em relação ao efetuador da mesa $Ad_{T_{ab}^{de}}$ sabendo que:

$$Ad_T = \begin{bmatrix} R & 0 \\ \hat{p}R & R \end{bmatrix} \tag{9}$$

Desta forma, usando 8 para calcular o adjunto de T_{ab}^{de} e aplica-lo na equação 3 junto de

6, teremos:

$$V_t^{de} = Ad_{(T_{tb}^{ab}T_{de}^{tb})^{-1}} J_t^{ab}\dot{\theta}_{braço} - Ad_{(T_{de}^{tb})^{-1}} J_{de}^{tb}\dot{\theta}_{mesa} \tag{10}$$

Que pode ser escrito como:

$$V_t^{de} = \begin{bmatrix} Ad_{(T_{tb}^{ab}T_{de}^{tb})^{-1}} J_t^{ab} & -Ad_{(T_{de}^{tb})^{-1}} J_{de}^{tb} \end{bmatrix} \begin{bmatrix} \dot{\theta}_{braço} \\ \dot{\theta}_{mesa} \end{bmatrix} \tag{11}$$

Sendo assim, temos que:

$$J_t^{de} = \begin{bmatrix} Ad_{(T_{tb}^{ab}T_{de}^{tb})^{-1}} J_t^{ab} & -Ad_{(T_{de}^{tb})^{-1}} J_{de}^{tb} \end{bmatrix} \tag{12}$$

7.1 Implementação em Código

Acrescentando as linhas de código a seguir ao código apresentado na seção 6.1

```
% Transformação Efetuador da Mesa -> Base da mesa
Rtb_de = Ttb_de.R;
ptb_de = Ttb_de.t;

Ttb_de_inv = [Rtb_de.' -Rtb_de.'*ptb_de;
              0 0 0  1];

% Adjunto da transformação Efetuador da Mesa -> Base do Braço
Rab_de_inv = SE3(Tab_de_inv).R;
pab_de_inv = SE3(Tab_de_inv).t;

Ad_de_ab = [Rab_de_inv zeros(3,3);
            skew(ptb_de)*Rab_de_inv Rab_de_inv];

% Adjunto da transformação Efetuador da Mesa -> Base da Mesa
Rtb_de_inv = SE3(Ttb_de_inv).R;
ptb_de_inv = SE3(Ttb_de_inv).t;

Ad_de_tb = [Rtb_de_inv zeros(3,3);
            skew(ptb_de)*Rtb_de_inv Rtb_de_inv];

```

```
% Volta a base de KR30 para a base do braço
KR30.base = (transl(0.0, 0.0, 0.0) * trotx(pi));

Jab_t = KR30.jacob0(theta_braco);
Jtb_de = DKP400.jacob0(theta_mesa);

Jde_t = [Ad_de_ab*Jab_t -Ad_de_tb*Jtb_de];
```

Usanda $\theta_{braço} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$ e $\theta_{mesa} = \begin{bmatrix} 0 & 0 \end{bmatrix}$. Teremos o seguinte jacobiano:

$$J_t^{de} = \begin{bmatrix} 0.3161 & -0.1609 & 0.4005 & -0.3335 & -0.4693 & -0.0004 & -0.0000 & 0 \\ -0.6647 & 0.3874 & 0.5110 & -0.2922 & 0.3697 & 0.0000 & -0.4470 & 0 \\ -0.6647 & -0.6176 & 0.0618 & 0.1960 & -0.2480 & -0.0000 & -0.0000 & 0 \\ 0.7592 & -0.8359 & -1.1528 & -0.3645 & 0.4612 & -0.0000 & 1.3693 & 0 \\ 0.9682 & 0.3642 & 0.8280 & -0.2163 & 0.2146 & 0.5573 & -0.0000 & 0 \\ -0.6072 & 0.4463 & 0.6237 & -0.9425 & -0.5529 & 0.8300 & -0.0000 & -1.0000 \end{bmatrix}$$

8 Validação do Jacobiano

Para validar o Jacobiano iremos comparar o resultado ao aplicarmos uma velocidade $\dot{\theta} = 0.01 rad/s$ arbitraria com a velocidade calculada a partir das transformação homogênea, vinda do modelo apresentado na seção 6.1, ao aplicarmos θ_i e θ_f, sendo $\theta_i = \theta_{waypoint1}$ e $\theta_f = \theta_{waypoint1} + \Delta t \cdot \dot{\theta}$, para $\Delta t = 0.01s$.

Os resultados obtidos das velocidades lineares (em m/s) e angulares (em rad/s), para os metodos do jacobiano e das transformação, e o erro entro elas ($Erro = Transformações - Jacobiano$), foram:

	Jacobiano	**Transformações**	**Erro**
$\dot{x}$	-0.0025	-0.0025	0.0
$\dot{y}$	0.0031	0.0031	0.0
$\dot{z}$	-0.0127	-0.0127	0.0
$\dot{\phi}$	-0.0113	-0.0128	-0.0015
$\dot{\theta}$	0.0272	0.0297	0.0026
$\dot{\psi}$	-0.0020	-0.0011	0.0009

Tabela 10: Velocidadas pelo metodo do Jacobiano e pelo metodo das Transformações

8.1 Código usando para cálculo das velocidades pelo metodo das Transformações

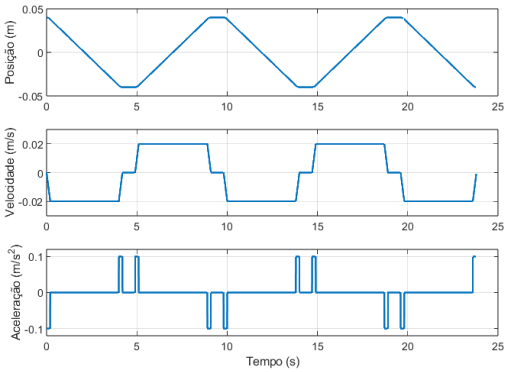
Para calcularmos as velocidades pelo metodo das transformações, acrescentamos as linhas de código a seguir ao código da seção 6.1.

```
% Volta a base de KR30 para o Efetuador da Mesa
KR30.base = Tab_de_inv * (transl(0.0, 0.0, 0.0) * trotx(pi));

T1 = KR30.fkine(q1);
T2 = KR30.fkine(q1 + 0.001*dq(1:6));

delta = tr2delta(T1, T2); % Retorna [dx dy dz wx wy wz] já escalonado
d_real = delta / 0.001;

% Adjunto da transformação Efetuador da Mesa -> Base da Mesa
R = T1.R;
```

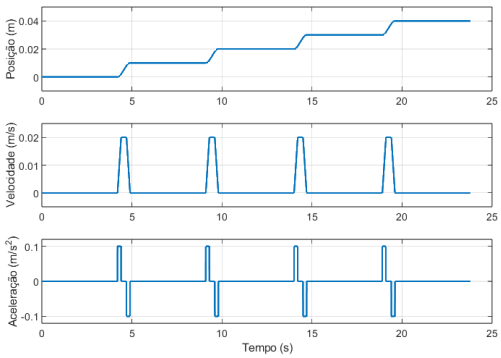
```
12 p = T1.t;
13
14 Ad_T1 = [R zeros(3,3);
15         skew(p)*R R];
16
17 d_real_spatial = Ad_T1 * d_real;
```

9 Controle Cinemático

Para o controle cinemático, geremos a tragetória que passa por cada um dos waypoints estabelecidos na tabela 6 seguindo um perfil trapezoidal, com: $\dot{xc} = 0.02m/s$, $\ddot{xc} = 0.1m/s^2$ e $tc = 0.2s$. Após isso, convertemos essa tragetoria para o espaço das juntas.

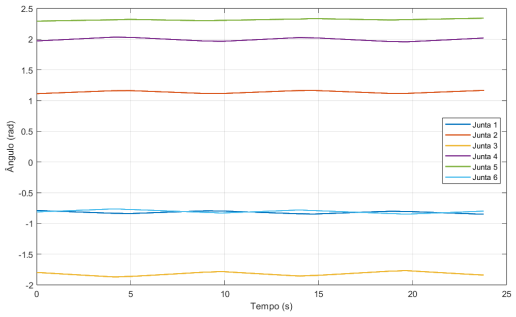


((a)) Tragetória no Eixo X

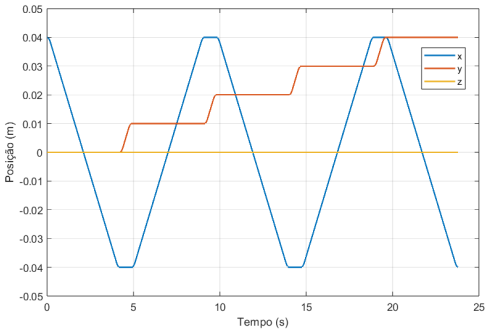


((b)) Tragetória no Eixo Y

Figura 6: Tragetórias no Espaço Operacional



((a)) Espaço das Juntas

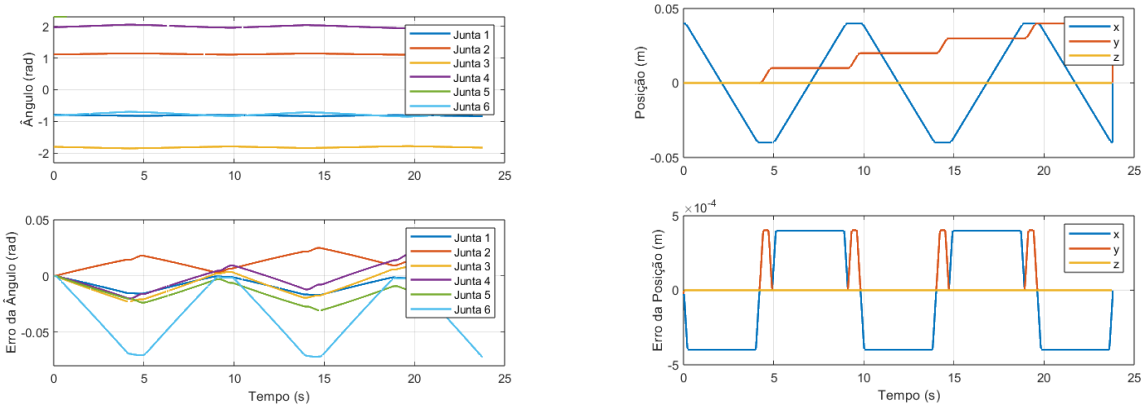


((b)) Espaço Operacional

Figura 7: Tragetórias

9.1 Controlador sem FeedForward

Para o caso do primeiro controlador, temos que $u = J(\theta)^{-1}K(x_d - x)$. Após algumas simulações para diferentes valores de K, optamos por adotar $K = 50$, pois mantem o erro dentro da escala de $10^{-4}m$ e mantem a velocidade das juntas abaixo de $0.1rad/s$.



((a)) Espaço das Juntas ((b)) Espaço Operacional

Figura 8: Resultado para Controlador sem FeedForward

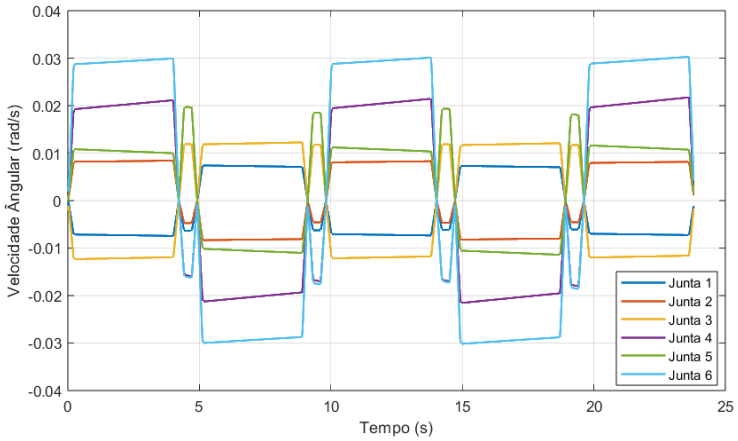
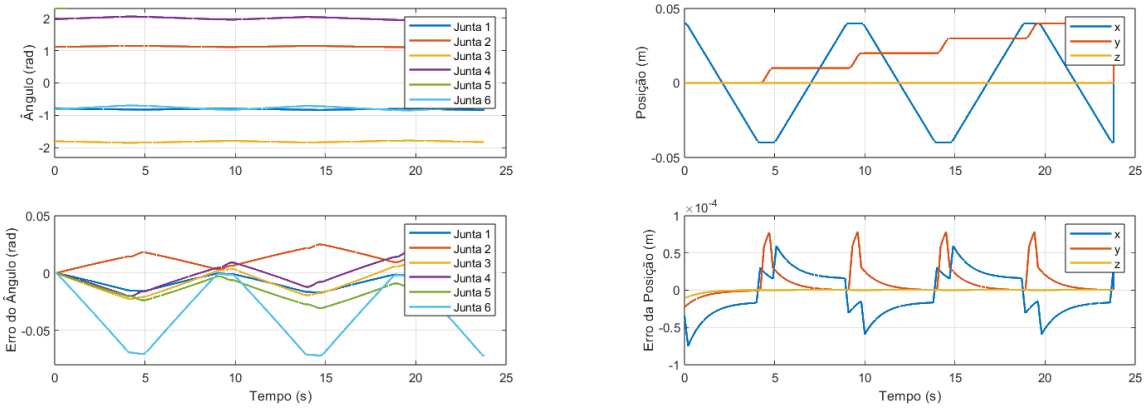


Figura 9: Velocidade das Juntas sem FeedForward

9.2 Controlador com FeedForward

Para o caso do segundo controlador, temos que $u = J(\theta)^{-1}[\dot{x_d} + K(x_d - x)]$. Após algumas simulações para diferentes valores de K, optamos por adotar $K = 1$, pois mantem o erro dentro da escala de $10^{-4}m$ e mantem a velocidade das juntas abaixo de $0.1rad/s$.



((a)) Espaço das Juntas ((b)) Espaço Operacional

Figura 10: Resultado para Controlador com FeedForward

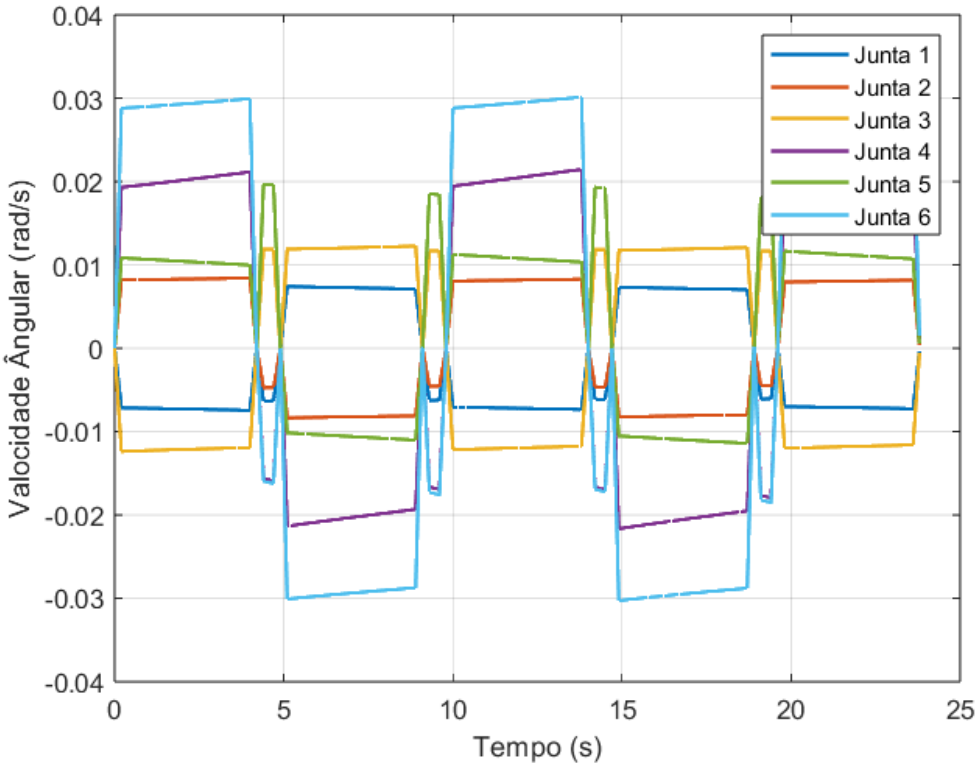


Figura 11: Velocidade das Juntas com FeedForward

9.3 Conclusão

Após a analise dos resultados obtidos anteriormente, notamos que, os dois controladores se mostraram capazes de estabilizar o movimento do manipulador, mesmo para um ganho pequeno no controlador sem feedforward. Porém, em uma comparação entre os dois controladores, o controlador que faz uso do feedforward se mostrou ter um desempenho melhor, visto que, para valores semelhantes de erro, o controlador com feedforward precisou de um ganho cerca de cinquenta vezes menor do que o controlador sem feedforward.

9.4 Códigos para o Controle Cinemático

9.4.1 Cálculo das Tragetórias

```
clear; clc; close all;

%=========================================================================
% Configuração do Manipulador
%=========================================================================
% Definição dos elos da mesa (DH Standard)
Lt(1) = Link('d', 0, 'a', 0, 'alpha', pi/2, 'offset', -pi/2);
Lt(2) = Link('d', 0, 'a', 0, 'alpha', 0, 'offset', 0);

% Criação da mesa
DKP400 = SerialLink(Lt, 'name', 'DKP400');

% Transformação Base -> Frame 0
DKP400.base = transl(0.0, 0.0, 0.51) * troty(pi/2);

% Transformação Elo final -> Efetuador
DKP400.tool = transl(0, 0, 0.447) * trotz(-pi/2);

% Configuração das juntas da mesa
theta_t = [pi/4 0];

Ttb_de = DKP400.fkine(theta_t);

% Transformação Base do Braço -> Base da Mesa
```

```
25 Tat = transl(0.9, 0, -0.48) * trotz(pi);
26 Tat = SE3(Tat);
27
28 % Transformação Base do Braço -> Efetuador da Mesa
29 Tab_de = Tat * Ttb_de;
30
31 % Transformação Efetuador da Mesa -> Base do Braço
32 Rab_de = Tab_de.R;
33 pab_de = Tab_de.t;
34
35 Tab_de_inv = [Rab_de.'  -Rab_de.'*pab_de;
36               0 0 0  1];
37
38 % Definição dos elos do braco (DH clássico)
39 La(1) = Link('d', -0.575, 'a', 0.175, 'alpha', pi/2, 'offset', 0);
40 La(2) = Link('d', 0, 'a', 0.89, 'alpha', 0, 'offset', 0);
41 La(3) = Link('d', 0, 'a', 0.05, 'alpha', pi/2, 'offset', -pi/2);
42 La(4) = Link('d', -1.035, 'a', 0, 'alpha', -pi/2, 'offset', 0);
43 La(5) = Link('d', 0, 'a', 0, 'alpha', pi/2, 'offset', 0);
44 La(6) = Link('d', -0.646339, 'a', 0, 'alpha', -deg2rad(33.89), 'offset', -pi/2);
45
46 % Criação do braço
47 KR30 = SerialLink(La, 'name', 'KR30');
48
49 % Transformação Efetuador da Mesa -> Frame 0
50 KR30.base = Tab_de_inv * (transl(0.0, 0.0, 0.0) * trotx(pi));
51
52 % Transformação Elo final -> Efetuador
53 KR30.tool = transl(0, 0, -0.000648) * (trotx(-pi)*trotz(pi/2));
54
55 %=====================================================================
56 % Geração da Trajetoria no Espaço Operacional
57 %=====================================================================
58 p = [ 0.04 0 0;
59      -0.04 0 0;
60      -0.04 0.01 0;
61       0.04 0.01 0;
```

```
62          0.04 0.02 0;
63         -0.04 0.02 0;
64         -0.04 0.03 0;
65          0.04 0.03 0;
66          0.04 0.04 0;
67         -0.04 0.04 0 ];
68
69  dxc = 0.02; ddxc = 0.1; tc = 0.2;
70
71  pps = 200;
72  tt = 0;
73
74  xd = [0 0 0];
75  dxd = [0 0 0];
76  ddxd = [0 0 0];
77
78  for i = 1:size(p,1)-1
79      % Parâmetros
80      q0 = p(i,:);
81      qf = p(i+1,:);
82
83      dq = norm(qf - q0);
84      tf = tc + dq/dxc;
85
86      t = linspace(0, tf, tf*pps);
87
88      q   = zeros(length(t), length(q0));
89      qd  = zeros(length(t), length(q0));
90      qdd = zeros(length(t), length(q0));
91
92      for k = 1:length(t)
93          if t(k) < tc
94              % Fase 1: aceleração
95              q(k,:)   = q0 + sign(qf - q0) * 0.5*ddxc*t(k)^2;
96              qd(k,:)  = sign(qf - q0) * ddxc*t(k);
97              qdd(k,:) = sign(qf - q0) * ddxc;
98
```

```
99          elseif t(k) < tf-tc
100             % Fase 2: velocidade constante
101             q(k,:)   = q0 + sign(qf - q0) * (0.5*ddxc*tc^2 + dxc*(t(k)-tc));
102             qd(k,:)  = sign(qf - q0) * dxc;
103             qdd(k,:) = 0;
104
105         else
106             % Fase 3: desaceleração
107             q(k,:)   = qf - sign(qf - q0) * 0.5*ddxc*(tf - t(k))^2;
108             qd(k,:)  = sign(qf - q0) * ddxc*(tf - t(k));
109             qdd(k,:) = -sign(qf - q0) * ddxc;
110         end
111     end
112
113     tt = tt + tf;
114     xd = [xd; q];
115     dxd = [dxd; qd];
116     ddxd = [ddxd; qdd];
117
118 end
119
120 xd = xd(2:end-1, :);
121 dxd = dxd(2:end-1, :);
122 ddxd = ddxd(2:end-1, :);
123
124 tn = linspace(0, tt, tt*pps);
125
126 %=====================================================================
127 % Trajetoria no Espaço das Juntas
128 %=====================================================================
129 R = trotz(pi/2) * trotx(pi);
130 Td = zeros(4, 4, length(tn));
131
132 for i = 1:length(tn)
133     Td(:,:,i) = transl(xd(i,:)) * R;
134 end
135
```

```
mask = [1 1 1 1 1 1];

tol = 1e-6;

lambda = 1;

thetad_sol = zeros(length(tn), 6);

% posição inicial
thetad0 = [-0.7911 1.1134 -1.7997 1.9726 2.2941 -0.8118];

for i = 1:length(tn)
    thetad = KR30.ikine(Td(:,:,i), thetad0, 'mask', mask, 'tol', tol, 'lambda', lambda);

    thetad_sol(i,:) = thetad;

    thetad0 = thetad;
end
```

9.4.2 Controlador

```
%=====================================================================
% Controle Cinemático
%=====================================================================
% Parâmetros
theta0 = [-0.7911 1.1134 -1.7997 1.9726 2.2941 -0.8118];
theta = zeros(length(tn), length(theta0));
dtheta = zeros(length(tn), length(theta0));
x = zeros(length(tn), 3);
e = zeros(length(tn), 3);
Ksf = 50;
Kcf = 1;
dt = tn(2) - tn(1);

for i = 1:length(tn)-1
    if i == 1
        theta(i,:) = theta0;
```

```
17      end
18
19      % Coletanto os valores de x(t)
20      % Volta a base de KR30 para o Efetuador da Mesa
21      KR30.base = Tab_de_inv * (transl(0.0, 0.0, 0.0) * trotx(pi));
22      T = KR30.fkine(theta(i,:));
23      x(i,:) = T.t';
24
25      % Calculando o Jacobiano do Efetuador do Braço em relação ao Efetuador da Mesa
26      % Volta a base de KR30 para a base do braço
27      KR30.base = (transl(0.0, 0.0, 0.0) * trotx(pi));
28
29      Jab_t = KR30.jacob0(theta(i,:));
30
31      J = Ad_de_ab*Jab_t;
32
33      % Controle
34      e(i,:) = xd(i,:) - x(i,:);
35      % Caso 1: sem feedfoward
36      usf = J \ (Ksf * [e(i,:) zeros(1,3)].');
37      % Caso 1: com feedfoward
38      ucf = J \ ([dxd(i,:) zeros(1,3)].' + Kcf * [e(i,:) zeros(1,3)].');
39
40      dtheta(i,:) = ucf.';
41      theta(i+1,:) = theta(i,:) + dtheta(i,:) * dt;
42  end
43
44  dtheta(end,:) = dtheta(end-1,:);
```