Universidade Estadual de Campinas

Faculdade de Engenharia Mecânica

TRABALHO - ROBÔ IRB 120

ES 827 – Robótica Industrial

Arnaud Bosquillon de Jarcy 203079

Gabriel de Freitas Leite 216180

Igor Barros Teixeira 217947

Matheus Santos Sano 222370

CAMPINAS, junho de 2022

1. Esquemático do manipulador

O esquemático do manipulador IRB 120 é apresentado na especificação do produto e pode ser visto a seguir:

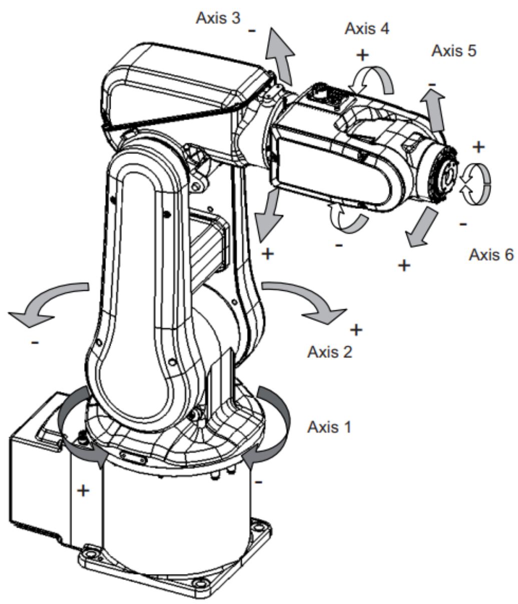
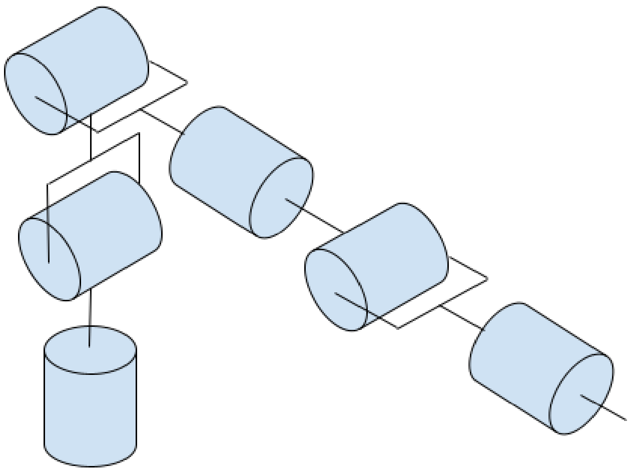
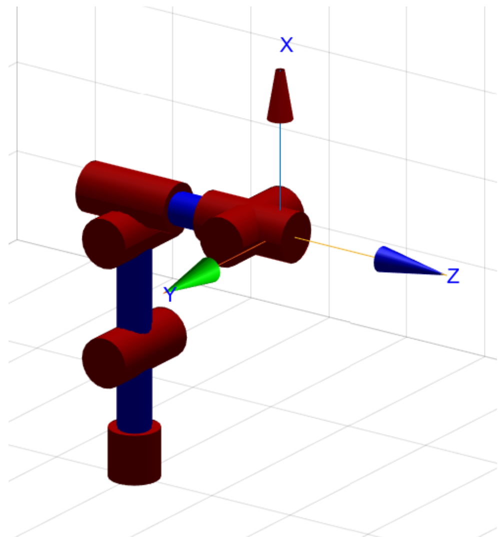


Figura 01: Esquemático do manipulador IRB 120.

Para facilitar a representação do robô e para obter os parâmetros de Denavit-Hartenberg o esquemático das juntas rotacionais foi apresentado a seguir:





Figuras 2a e 2b: Esquemáticos das juntas do IRB 120 em desenho e simulação.

Como pode ser visto na Figura 02, o robô trabalhado possui 6 juntas e todas são rotacionais (6 GDL RRRRRR).

2. Parâmetros DH

Para obter os parâmetros DH é necessário, primeiramente, colocar as coordenadas nos sistemas:

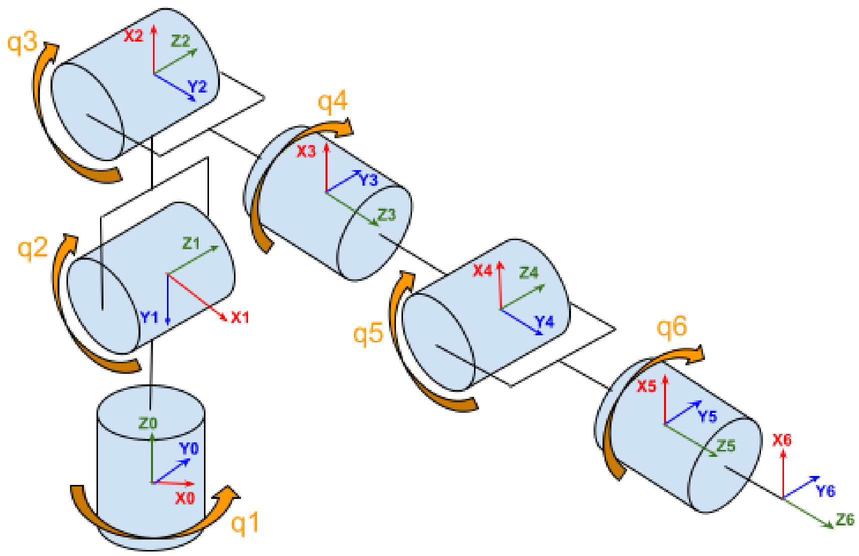


Figura 03: Esquemático do manipulador com as coordenadas em cada junta rotacional.

Após o sistema de coordenadas ser colocado em cada junta, foram obtidos os parâmetros DH:

elo	θ_i	d_i	a_i	α_i
1	q_1^*	d_1	0	$-90°$
2	q_2^*	0	a_2	$0°$
3	q_3^*	0	a_3	$-90°$
4	q_4^*	d_4	0	$90°$
5	q_5^*	0	0	$-90°$
6	q_6^*	d_6	0	$0°$

Tabela 01: Parâmetros DH de cada elo.

Os parâmetros Denavit-Hartenberg são apresentados na tabela 01, sendo q_1 , q_2 , q_3 , q_4 , q_5 e q_6 os ângulos de rotação de cada junta (variáveis). Os valores das constantes foram obtidas na especificação do produto:

$$d_1 = 0,29$$
$$a_2 = 0,27$$
$$a_3 = 0,07$$
$$d_4 = 0,302$$
$$d_6 = 0,072$$

3. Cinemática direta

A partir dos parâmetros DH, é realizada a cinemática direta do manipulador, obtendo as matrizes de transformação:

$$A_i = \begin{bmatrix} C_i & -S_iC(\alpha) & S_iS(\alpha) & a_iC_i \\ S_i & C_iC(\alpha) & -C_iS(\alpha) & a_iS_i \\ 0 & S(\alpha) & C(\alpha) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & 0{,}29 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_2 = \begin{bmatrix} C_2 & -S_2 & 0 & 0{,}27C_2 \\ S_2 & C_2 & 0 & 0{,}27S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} C_3 & 0 & -S_3 & 0{,}07C_3 \\ S_3 & 0 & C_3 & 0{,}07S_3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_4 = \begin{bmatrix} C_4 & 0 & S_4 & 0 \\ S_4 & 0 & -C_4 & 0 \\ 0 & 1 & 0 & 0{,}302 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_5 = \begin{bmatrix} C_5 & 0 & -S_5 & 0 \\ S_5 & 0 & C_5 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_6 = \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & 0{,}072 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A partir de tais matrizes, foram calculadas as matrizes de transformação por meio do software MATLAB. As matrizes de transformação são apresentadas a seguir:

$$T_1^0 = A_1 = \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & 0{,}29 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^0 = A_1 \cdot A_2 = \begin{bmatrix} C_1 C_2 & -C_1 S_2 & -S_1 & 0{,}27 C_1 C_2 \\ C_2 S_1 & -S_1 S_2 & C_1 & 0{,}27 S_1 C_2 \\ -S_2 & -C_2 & 0 & 0{,}29 - 0{,}27 S_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^0 = A_1 A_2 A_3$$

$$T_3^0 = \begin{bmatrix} C_1 C_{23} & S_1 & -C_1 S_{23} & 0{,}27 C_1 C_2 + 0{,}07 C_1 C_{23} \\ S_1 C_{23} & -C_1 & -S_1 S_{23} & 0{,}27 S_1 C_2 + 0{,}07 . S_1 C_{23} \\ -S_{23} & 0 & -C_{23} & 0{,}29 - 0{,}27 S_2 - 0{,}07 S_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^0 = A_1 A_2 A_3 A_4$$

$$T_4^0 = \begin{bmatrix} S_1 S_4 + C_1 C_4 C_{23} & -C_1 S_{23} & C_1 S_4 C_{23} - S_1 C_4 & 0{,}27 C_1 C_2 + 0{,}07 C_1 C_{23} - 0{,}302 C_1 S_{23} \\ -C_1 S_4 + S_1 C_4 C_{23} & -S_1 S_{23} & C_1 C_4 + S_1 S_4 C_{23} & 0{,}27 . S_1 C_2 + 0{,}07 S_1 C_{23} - 0{,}302 . S_1 S_{23} \\ -C_4 S_{23} & -C_{23} & -S_4 S_{23} & 0{,}29 - 0{,}27 . S_2 - 0{,}07 . S_{23} - 0{,}302 . C_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_5^0 = A_1 A_2 A_3 A_4 A_5$$

$$T_5^0 = \begin{bmatrix} C_5 (S_1 S_4 + C_1 C_4 C_{23}) - C_1 S_5 C_{23} & S_1 C_4 - C_1 S_4 C_{23} & -S_5 (S_1 S_4 + C_1 C_4 C_{23}) - C_1 C_5 S_{23} & 0{,}27 C_1 C_2 + 0{,}07 C_1 C_{23} - 0{,}302 C_1 S_{23} \\ -C_5 (C_1 S_4 - S_1 C_4 C_{23}) - S_1 S_5 S_{23} & -S_1 S_4 C_{23} - C_1 C_4 & S_5 (C_1 S_4 - C_1 S_1 C_{23}) - S_1 S_5 S_{23} & 0{,}27 S_1 C_2 + 0{,}07 S_1 C_{23} - 0{,}302 S_1 S_{23} \\ -S_5 C_{23} - C_4 C_5 S_{23} & S_4 S_{23} & C_4 S_5 S_{23} - C_5 C_{23} & 0{,}29 - 0{,}27 S_2 - 0{,}07 S_{23} - 0{,}302 C_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_6^0 = A_1 A_2 A_3 A_4 A_5 A_6$$

$$T_6^0 = \begin{bmatrix} S_6 (S_1 C_4 - C_1 S_4 C_{23}) + C_6 [C_5 (S_1 S_4 + C_1 C_4 C_{23}) - C_1 S_5 S_{23}] & C_6 (C_4 S_1 - C_1 S_4 C_{23}) - S_6 [C_5 (S_1 S_4 + C_1 C_4 C_{23}) - C_1 S_5 S_{23}] & -S_5 (S_1 S_4 + C_1 C_4 C_{23}) - C_1 C_5 S_{23} & 0{,}27 C_1 C_2 - 0{,}072 . [S_5 . (S_1 S_4 + C_1 C_4 C_{23}) + C_5 . C_1 S_{23}] + 0{,}07 C_1 C_{23} - 0{,}302 C_1 S_{23} \\ -S_6 (C_1 C_4 + S_1 S_4 C_{23}) - C_6 [C_5 (C_1 S_4 - C_4 S_1 C_{23}) + S_1 S_5 S_{23}] & -C_6 (C_1 C_4 + S_1 S_4 C_{23}) + S_6 [C_5 (C_1 S_4 - S_1 C_4 C_{23}) + S_1 S_5 S_{23}] & S_5 (C_1 S_4 - S_1 C_4 C_{23}) - S_1 C_5 S_{23} & 0{,}27 S_1 C_2 + 0{,}072 . [S_5 . (C_1 S_4 - C_1 C_4 C_{23}) - C_5 . C_1 S_{23}] + 0{,}07 S_1 C_{23} - 0{,}302 S_1 S_{23} \\ S_4 S_6 S_{23} - C_6 (S_5 C_{23} + C_4 S_5 S_{23}) & C_6 S_4 S_{23} + S_6 (S_5 C_{23} + C_4 C_5 S_{23}) & S_5 C_4 S_{23} - C_5 C_{23} & 0{,}29 - 0{,}27 S_2 + 0{,}072 . [C_4 S_5 S_{23} - C_5 C_{23}] - 0{,}07 S_{23} - 0{,}302 C_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4. Jacobiano

A partir das matrizes de transformação, é possível calcular o Jacobiano. Para calculá-lo foi implementado o seguinte código no MATLAB:

```
Jr1=[0; 0; 1];
Jr2=A1(1:3,3);
Jr3=T2(1:3,3);
Jr4=T3(1:3,3);
Jr5=T4(1:3,3);
Jr6=T5(1:3,3);

Jr=[Jr1 Jr2 Jr3 Jr4 Jr5 Jr6];

O0=[0; 0; 0];
O1=[T1(1,4); T1(2,4); T1(3,4)];
O2=[T2(1,4); T2(2,4); T2(3,4)];
O3=[T3(1,4); T3(2,4); T3(3,4)];
O4=[T4(1,4); T4(2,4); T4(3,4)];
O5=[T5(1,4); T5(2,4); T5(3,4)];
O6=[T6(1,4); T6(2,4); T6(3,4)];

J1=cross(Jr1,O6-O0);
J2=cross(Jr2,O6-O1);
J3=cross(Jr3,O6-O2);
J4=cross(Jr4,O6-O3);
J5=cross(Jr5,O6-O4);
J6=cross(Jr6,O6-O5);

Jv=[J1 J2 J3 J4 J5 J6];

Jg= [Jv; Jr];
```

Figura 04: Código em MATLAB para o cálculo do Jacobiano, sendo Jv o Jacobiano de velocidade e Jr o Jacobiano de rotação.

Como a matriz Jacobiana J_g é uma matriz 6 x 6, ela é muito grande, não sendo possível colocá-la neste relatório. Porém, para uma posição igual a `q=[0 -pi/2 0 0 0 0]`, a matriz jacobiana é apresentada a seguir:

```
Jg =

        0    0.3400    0.0700         0         0         0
   0.3740         0         0   -0.0000         0   -0.0000
        0   -0.3740   -0.3740         0   -0.0720         0
        0         0         0    1.0000         0    1.0000
        0    1.0000    1.0000         0    1.0000         0
   1.0000         0         0   -0.0000         0   -0.0000
```

5. Cinemática direta vs Cinemática inversa

A partir dos parâmetros DH e dada a posição: `q=[0 -pi/2 0 0 0 0]` foi obtida a matriz de transformação homogênea por meio da cinemática direta implementada pela função "fkine" do MATLAB. Esta função retornou a seguinte matriz de transformação homogênea:

```
T =
         0         0         1     0.374
         0        -1         0         0
         1         0         0      0.63
         0         0         0         1
```

Figura 05: Matriz de transformação homogênea obtida por meio da função "fkine" do MATLAB.

Ao substituir os valores da posição `q` na matriz de transformação T_6^0 obtida no **tópico 3**, é calculada a seguinte matriz de transformação:

```
T6 =

    0.0000         0    1.0000    0.3740
         0   -1.0000         0         0
    1.0000         0   -0.0000    0.6300
         0         0         0    1.0000
```

Figura 06: Matriz de transformação homogênea obtida por meio dos cálculos apresentados no tópico 3.

Como pode ser observado na Figura 06, a matriz de transformação homogênea calculada no tópico 3 é igual à matriz obtida por meio da função "fkine" do MATLAB (Figura 05) para uma mesma posição `q`. Portanto, o resultado obtido no tópico 3 está correto.

Para verificar o resultado obtido pela cinemática direta foi implementada a cinemática inversa a partir da matriz de transformação (Figura 06) por meio da função "ikine" do MATLAB. A posição retornada pela função foi:

```
q =

         0   -1.5708         0         0         0         0
```

Figura 07: Ângulos vindos da função "ikine" a partir da matriz de transformação T6.

Como pode ser observado na Figura 07, a posição obtida pela cinemática inversa é igual à posição setada na cinemática direta. Portanto, a matriz de transformação homogênea e a implementação das cinemáticas direta e inversa estão corretas.

6. Trajetória da elipse

Para elaborar a trajetória de uma elipse no chão, foi necessário primeiramente traçar esta elipse no eixo de coordenadas x e y. Dessa forma, foi possível obter todos os pontos da elipse no chão. Em seguida, foi obtida a matriz transformação para cada ponto da elipse no chão e cada uma delas foi concatenada dentro de uma matriz T. Dessa forma, esta matriz T continha as matrizes transformação para todos os pontos da elipse no chão.

Em seguida, a partir de cada ponto da elipse, foi realizada a cinemática inversa para obter todas as posições do robô, e concatená-las em uma matriz Q. Sendo assim, a matriz Q possui todas as posições do robô, desde a inicial, para ele traçar o movimento de uma elipse no chão e depois retornar à posição inicial.

Dessa forma, o robô foi capaz de realizar os seguintes movimentos: sair da posição inicial e ir até o chão, traçar uma elipse e retornar à posição inicial. Para que o robô trace uma elipse na parede, a implementação é a mesma, porém os seus pontos estão localizados no eixo de coordenadas x e z. A implementação de todo o movimento do robô está no arquivo presente ".m" junto a este relatório.

7. Cálculo do torque

Para calcular o torque do sistema foi necessário definir as massas e as dimensões de cada junta do robô. Ademais, os centros de massa das juntas foram definidos e localizados no centro de cada junta. Vale ressaltar que a junta 5 e a junta 6 do robô estão localizadas juntas. Foi considerado também que o todas as juntas são delgadas, logo, o momento de inércia em torno do próprio eixo é igual a zero.

Quando a posição inicial é igual a: `q=[0 -pi/2 0 0 0 0]`, o manipulador está em uma configuração singular. Portanto, não seria possível obter a inversa da jacobiana e calcular o torque em alguns pontos nessa região. Sendo assim, foi considerada uma nova posição inicial na qual não há singularidades. A posição inicial é, então: `q=[0 -pi/6 -pi/6 0 pi/3 0]`.

Assim, ao longo da trajetória do robô, são obtidos a posição, a velocidade e a aceleração das juntas para cada ponto da trajetória. Para isso foram obtidas as posições do ponto atual e do próximo e, a partir delas, foi calculado o Jacobiano do ponto atual e do próximo. Para calcular a velocidade no ponto atual e no próximo, foram multiplicadas as inversas das matrizes jacobianas pela velocidade linear e rotacional do sistema (dX).

A partir das posições e das velocidades de ambos os pontos, foram obtidas a posição final, a velocidade final e a aceleração final em cada ponto, e foram concatenadas, respectivamente, nas matrizes Qfinal, QDfinal e QDDfinal.

Dessa forma, foi possível calcular o torque por meio da função “RNE” do MATLAB:

```
torque = bot.rne(Qfinal,QDfinal,QDDfinal);
```

Assim, o torque de cada junta em função do tempo é apresentado a seguir:

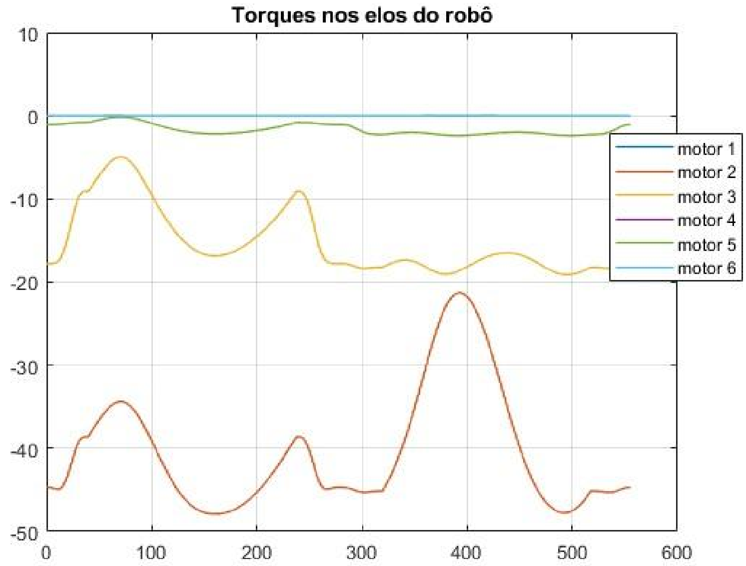


Figura 08: Torque em função do tempo de cada junta do manipulador.

Na Figura 08, pode-se observar que as juntas 2 e 3 são as que exigem mais torque. Isso ocorre, pois estas são as juntas que são responsáveis por sustentar o restante do manipulador. Em contrapartida, a junta 1 exige menos torque, pois ela apenas gira o conjunto sem sustentar o restante do robô. A junta 1 só precisará de maiores torques quando o sistema for submetido a maiores acelerações, pois aí existe o efeito Coriolis e a inércia do sistema. Durante a trajetória das elipses o punho não faz grandes movimentos, então apenas o motor 5 que também realiza sustentação vertical sofre certo torque. Na posição em que o robô se encontra os elos 2, 3 e 5 precisam tentar ser girados no sentido anti-horário para que o robô se mantenha na posição, isso justifica o torque ser negativo para as juntas de sustentação.

8. Controle do torque

Para controlar o torque das juntas do manipulador, foi utilizado um compensador PD via Simulink. É possível utilizar o próprio objeto *SerialLink* criado no Matlab como um bloco no Simulink. A malha utilizada no controle do robô é apresentada a seguir:

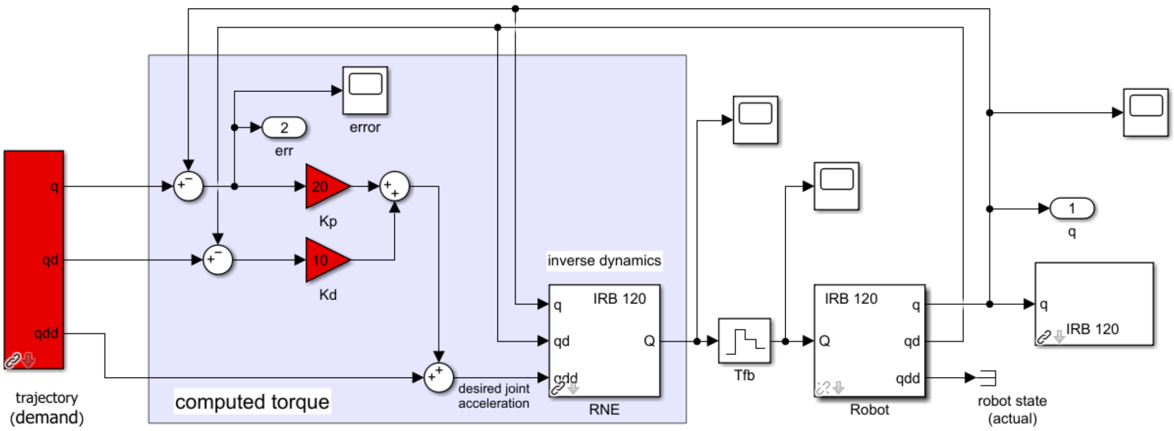


Figura 09: Malha de controle do robô IRB 120 com compensador PD.

A trajetória foi escolhida de modo a colocar esforço na maioria das juntas. Assim, os pontos inicial e final foram os seguintes:

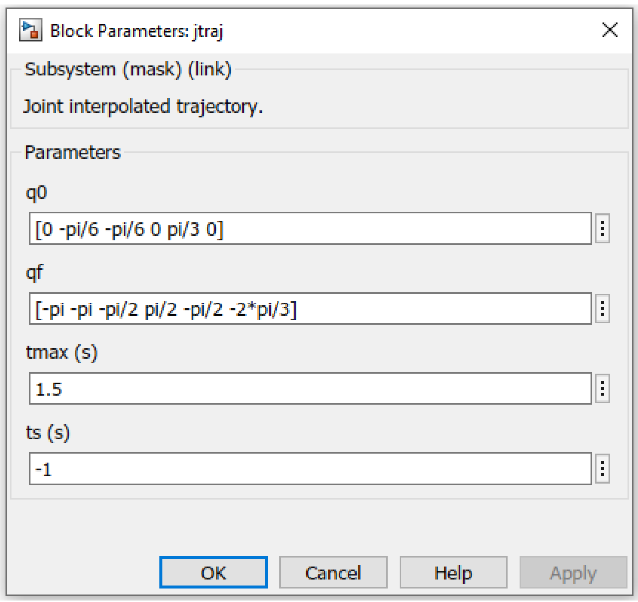


Figura 10: Pontos inicial e final para a malha de controle

Foram alterados os valores da massa do objeto no efetuador a ser transportado e dos ganhos Kp e Kd para analisar as alterações do torque. A primeira massa utilizada é um valor dentro do máximo que a ABB indica ser possível transportar, depois esse valor será extrapolado para entender seu efeito nos esforços dos motores.

O torque das juntas para uma massa de 3 kg, Kp = 20 e Kd = 5 é apresentado a seguir:

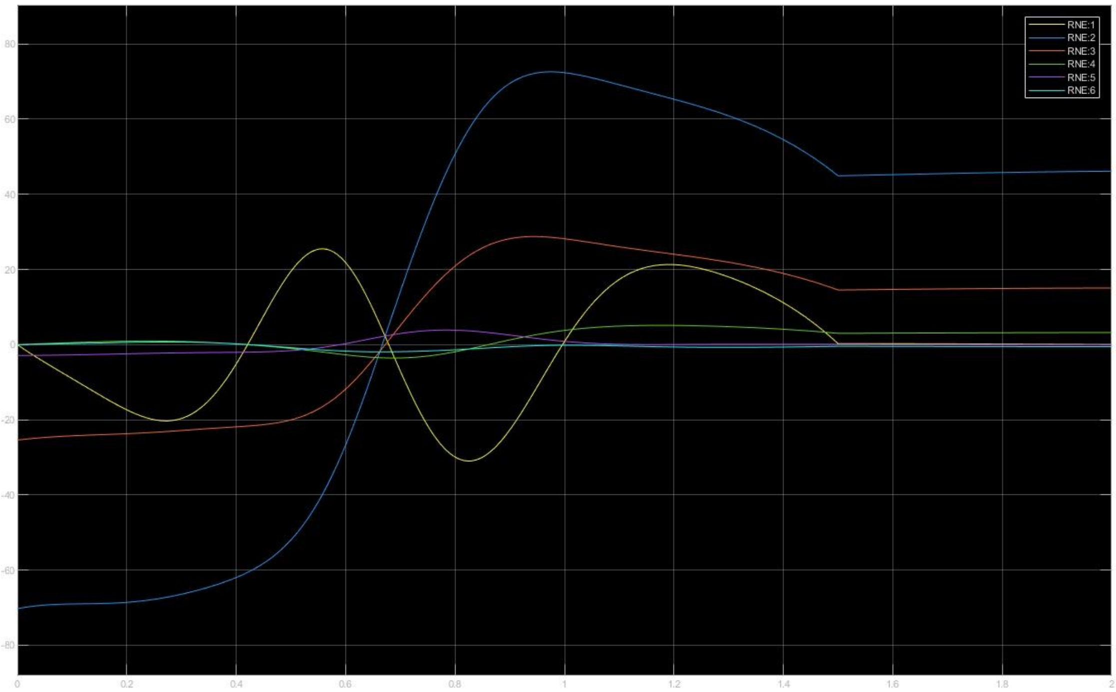


Figura 11: Torque das juntas para m = 3kg, Kp = 20 e Kd = 5.

Neste caso o torque máximo foi da junta 2 (azul) que ficou entre -75 Nm e 75 Nm. Os outros motores ficaram com torques menores.

O sistema com m = 3 kg, Kp = 20 e Kd = 5 será usado como modelo para analisar a variação do torque quando os ganhos e a massa são alterados. Caso o ganho proporcional aumente em 2 vezes, o sistema parte com mais velocidade, pois o erro de posição inicial. Dessa forma, a junta 2 tem um estresse maior e inicia o movimento com um torque de -100 Nm. A junta 3 também sofre deste problema. Já a junta 1 teve seu torque máximo reduzido, passando pouco de -20 Nm. Abaixo o gráfico mostrando a evolução do torque no tempo:

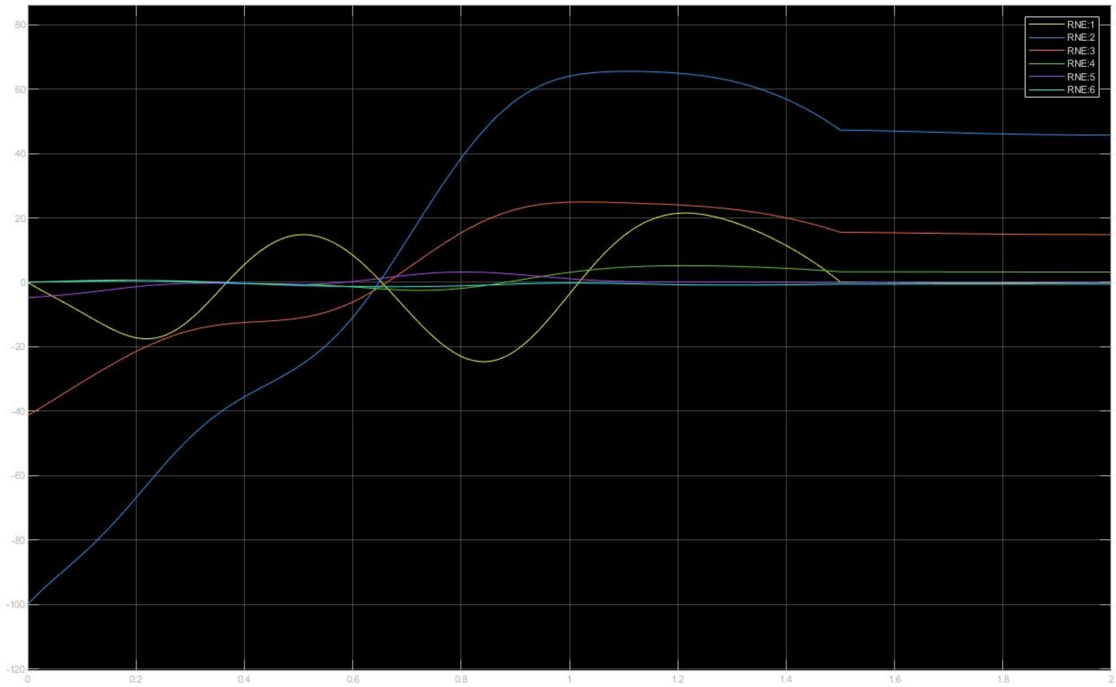


Figura 12: Torque das juntas para m = 3 kg, Kp = 40 e Kd = 5.

Caso apenas o ganho derivativo aumente em 2 vezes, as influências nos torques não serão tão significativas quanto no caso onde o Kp foi alterado. Para a situação do Kd maior houve um pequeno deslocamento no tempo das curvas mas os valores máximos se mantiveram bem próximos. Como segue:

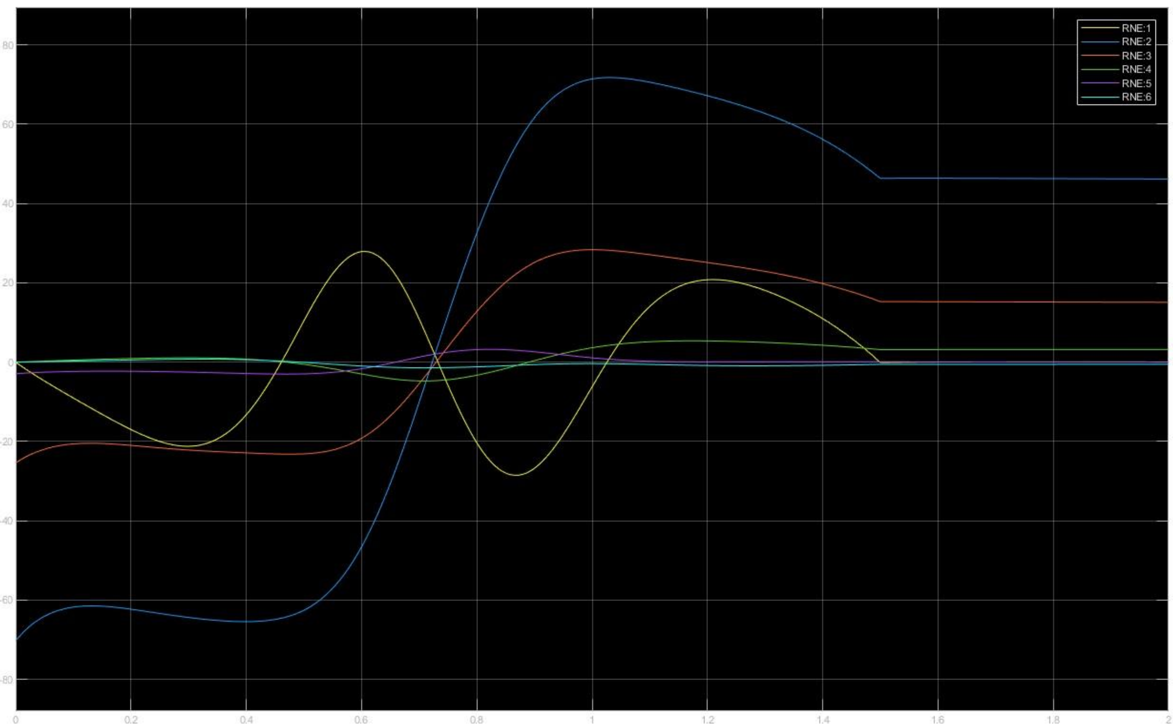


Figura 13: Torque das juntas para m = 3kg, Kp = 20 e Kd = 10.

Vale ressaltar que nos 3 testes anteriores o tempo de estabilização não teve alterações perceptíveis, o torque entrava em condição constante sempre em um tempo próximo a 1,5 segundos.

Agora a massa na ponta do manipulador foi alterada para 15 kg, valor bem acima do especificado pelo fabricante.

Primeiramente, com ganhos Kp = 20 e Kd = 5 para 15 kg de carga. As formas das curvas ficaram bem semelhantes com o teste de 3 kg para os mesmos ganhos, indicando que o controlando está se mostrando estável com alterações de carga. Agora a diferença ficou por conta das magnitudes. A junta com mais carga ainda é a junta 2, desta vez ela chegou na marca de quase 200 Nm. Segue gráfico:

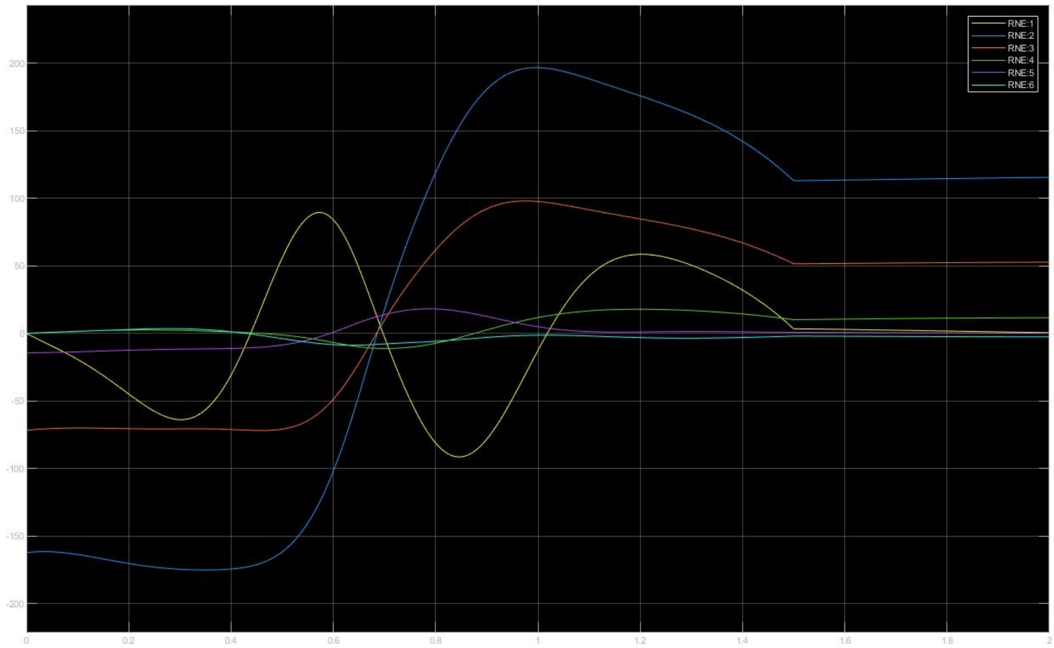


Figura 14: Torque das juntas para m = 15 kg, Kp = 20 e Kd = 5.

Alterando o ganho Kp de 20 para 40 o cenário também é semelhante ao caso de 3kg, a maior mudança ocorre no início do movimento onde o elo 2 sofre um esforço de quase -250 Nm e o elo 3 em algo próximo a -125 Nm. O elo 1 teve uma leve redução nos valores máximos de torque. Segue o gráfico:

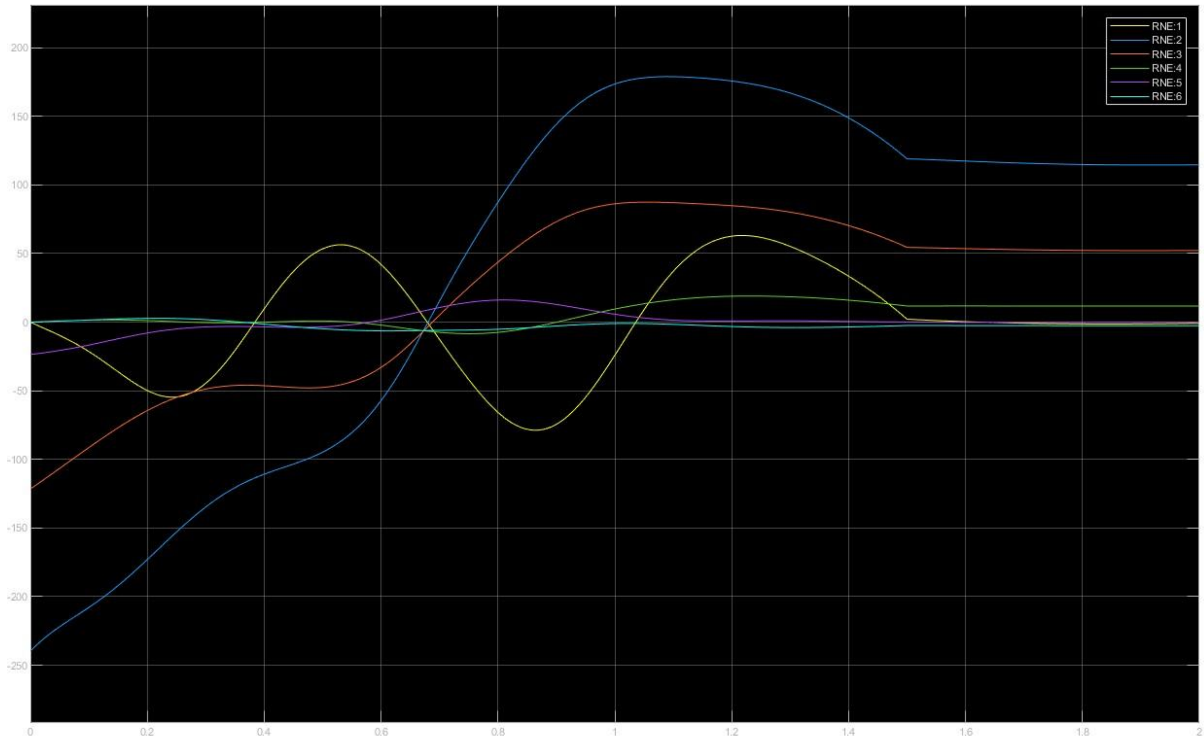


Figura 15: Torque das juntas para m = 15 kg, Kp = 40 e Kd = 5.

Finalmente, os ganhos foram configurados em Kp = 20 e Kd = 10 com a carga de 15 kg. Semelhante ao caso de 3 kg, pouca alteração ocorreu nos valores máximos dos torques, apenas um deslocamento no tempo e alteração da forma no início do movimento. Segue o gráfico:

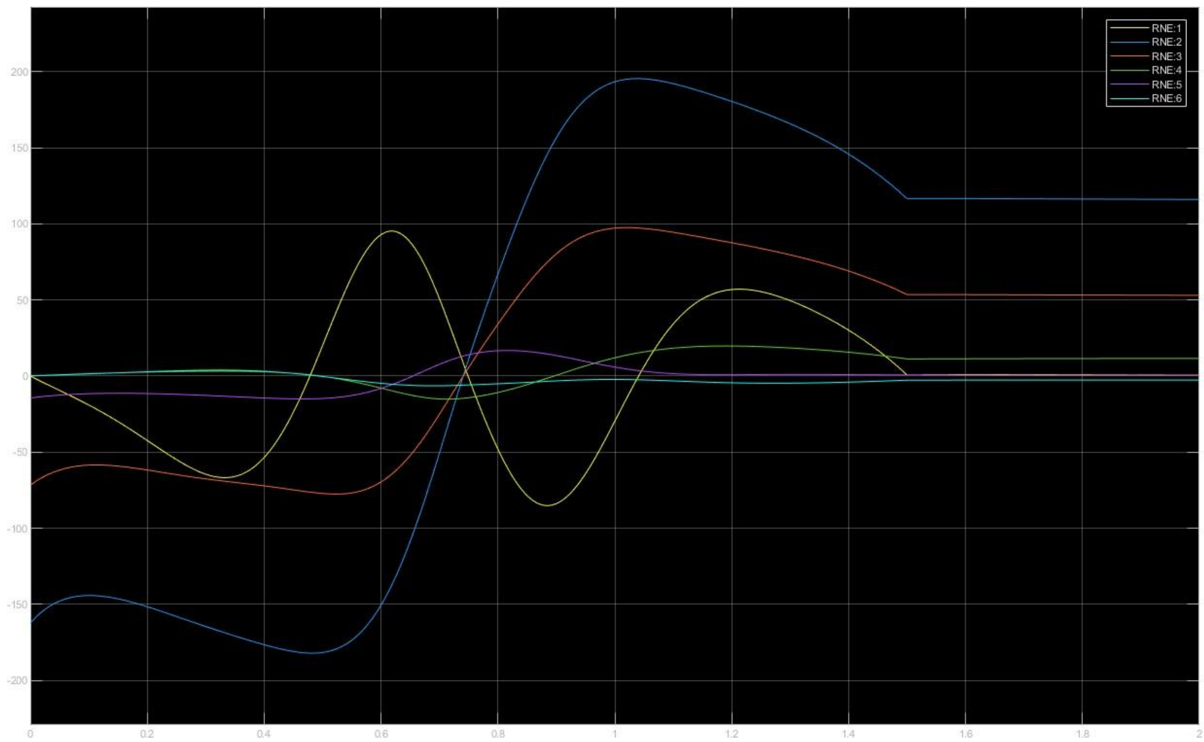


Figura 16: Torque das juntas para m = 15 kg, Kp = 20 e Kd = 10.

9. Dificuldades

Uma das principais dificuldades encontradas neste projeto foi a realização da cinemática inversa. Apesar de já existir uma função pronta, o entendimento do seu funcionamento e a sua aplicação foram bastante trabalhosos.

A principal dificuldade foi a realização da trajetória do robô. Durante a implementação da trajetória, houve muita dificuldade em fazer o manipulador se mover de forma correta e com uma velocidade aceitável. Inicialmente, o robô andava ponto a ponto da trajetória com uma velocidade muito pequena, quase imperceptível. Após muitos esforços, foi possível traçar a trajetória do manipulador de forma correta.

Após a trajetória estar funcionando, a dificuldade foi obter os valores de torque nos motores. Primeiro foi necessário incluir os dados dinâmicos do robô em sua montagem via *SerialLink* (massa dos elos, centro de massa, inércia, gravidade). Foi considerado que os elos são delgados e o centro de massa está no meio do elo, pois não foi possível obter dados do fabricante com relação a centros de massa pela distribuição física dos links. Para que a função RNE calcule os torques, é necessário ter valores de posição, velocidade e aceleração durante a trajetória. Quem faz isso é a função JTRAJ. Esta função precisa das posições inicial e final de cada elo e um valor de *step* para cálculo da trajetória usando polinômio de grau 5. Por padrão ela considera que as velocidades inicial e final são nulas, fazendo com que o robô pare em todos os pontos gerados da elipse.

Para resolver esse problema, foi necessário calcular as velocidades dos elos em cada ponto para que o robô não pare ao chegar neles e deixa o movimento contíguo. Foi determinada uma matriz de velocidades lineares e angulares (dX) e multiplicada pelo inverso do jacobiano naquele ponto, dessa forma foi obtida a velocidade de cada elo. Tendo essas velocidades, é possível utilizar a função JTRAJ com velocidades inicial e final definidas, assim o robô não para nos pontos da elipse.

Agora, tendo dados de posição, velocidade e aceleração vindo da JTRAJ foi possível utilizar a RNE e calcular os torques no elos durante a trajetória das elipses e plotá-la.