Robótica de Manipulação, 1º Relatório: Cinemática

Instituto Superior Técnico

MEMec

Abril 2021

Grupo 18

Catarina Pires, Nº 90230

Ricardo Henriques, Nº 90349

Professores: João Reis, André Carvalho

Versão de *Matlab* usada: *Matlab R2016a*

Índice

1 Introdução

No âmbito da unidade curricular Robótica de Manipulação, este trabalho consiste na análise do robô UR5 da Universal Robots. O foco deste relatório será na análise Cinemática do robô. Nomeadamente, é feita uma pequena inrodução aos conceitos de Cinemática Direta, matriz Jacobiana Geométrica e o efeito de singularidades na mesma, Cinemática Inevrsa, e, finalmente, é apresentada uma solução *Closed-form* para a mesma. Em cada uma desta secções é também apresentado o modelo de simulação obtido, em *Simulink*, e a sua validação com dados fornecidos no enunciado.

2 Ficheiros de *Matlab*

Para a implementação dos modelos utilizados neste relatório recorreu-se, como ponto de partida, à *toolbox Robotics_Symbolic_Matlab_Toolbox* fornecida pelo corpo docente, a qual é constituída pelas seguintes funções de *Matlab*:

- ***RobotX*:** Esta função devolve a tabela de parâmetros do robô, de acordo com a convenção Denavit-Hartenberg (nome mudado para *Robot_G18* na realização deste projeto);
- ***DHTransf*:** Devolve a matriz de transformção da articulação Denavit-Hartenberg (nome mudado para *DHTransf_G18* na realização deste projeto);
- ***DKin*:** Devolve a matriz de transformação do robô para o modelo de Cinemática Direta, tranformação da *frame 0* para a última *frame*. (nome mudado para *DKin_G18* na realização deste projeto)
- ***vrrobot*:** Cria uma animação do robô, representando os seus eixos num espaço 3D. (nome mudado para *vrrobot_G18* na realização deste projeto);
- ***Kinematics_G18*:** Servindo-se das unções acima descritas, define a tabela Denavit-Hartenberg do manipulador, a partir da mesma define a matriz de rotação do *end-effector* no referencial 0, *R06*, a posição final do *end-effector* no referencial 0, *p6*, e a matriz Jacobiana Geométrica em função dos ângulos de rotação nas juntas;
- ***GeoJacobian*:** Esta função calcula matriz Jacobiana Geométrica do robô.

3 Modelo Cinemático

Primeiramente, obteve-se o modelo cinemático do braço robótico em estudo de acordo com a configuração Denavit-Hartenberg (DH). Esta convenção permite definir a posição relativa e a orientação de dois links consecutivos, de forma a calcular a equação cinemática direta para um manipulador de cadeia aberta. Para obter esta equação são necessários certos parâmetros característicos desta convenção que são apresentados de seguida:

- a_i: distância entre O_i e O_i'.
- d_i: coordenada de O_i' no eixo z_{i-1}.
- α_i: ângulo entre os eixos z_{i-1} e z_i sobre o eixo x_i sendo que este ângulo é positivo se a rotação for no sentido anti horário.
- θ_i: ângulo entre os eixos x_{i-1} e x_i sobre o eixo z_i sendo que este ângulo é positivo se a rotação for no sentido anti horário.

Nesta convenção existe um conjunto de regras que permitem não só a atribuição dos referenciais locais, como o cálculo das matrizes que permitem obter a equação cinemática. Estas regras podem ser escritas na forma de um algoritmo:

- **1:** Encontrar e numerar de forma consecutiva os eixos articulados e colocar a direcção dos eixos: z_0,...,z_n.
- **2:** Escolher a frame 0 localizando a origem no eixo z_0. Definir x_0 e y_0 de acordo com a regra da mão direita.
- **NOTA:** Os passos 3 a 5 são executados de forma recursiva para i = 1,...,n-1:
- **3:** Localizar a origem O_i na intersecção de z_i com a normal comum aos eixos z_{i-1} e z_i. Se z_{i-1} e z_i são paralelos e se a junta i for de rotação, O_i localiza-se de forma a que d_i=0. Se a junta i for prismática, O_i localiza-se num ponto de referência do alcance da junta.
- **4:** Escolher o eixo x_i ao longo da normal comum aos eixos z_{i-1} e z_i com a direcção da junta i para a junta i+1.
- **5:** Escolher y_i de acordo com a regra da mão direita.
- **6:** Escolher a frame n. Se a junta for de rotação, alinha-se z_n e z_{n-1}. Se a junta for prismática, escolhe-se z_n de forma arbitrária.
- **7:** Para i=1,..,n escreve-se a tabela de parâmetros.
- **8:** Escrever as matrizes de transformação homogéneas ${A_i}^{i-1}(q_i)$ para i=1,...,n.
- **9:**Escrever a matriz final de transformação homogénea ${T_n}^0(\mathrm{q}) = {A_0}^1 ... {A_n}^{n-1}$
- **10:** Escrever a função da cinemática direta ${T_e}^b(\mathrm{q}) = {T_0}^b \ {T_n}^0(\mathrm{q}) \ {T_e}^n$

Através da análise da figura 1 pode-se observar que o robô em estudo apresenta seis juntas, todas de rotação, apresentando, assim, seis graus de liberdade, uma rotação por cada junta. O sentido positivo das rotações encontra-se também representado na figura 1 e na figura 2.

Após a identificação destas características desenhou-se o diagrama de juntas e definiram-se os referenciais para cada *frame* de acordo com o algoritmo apresentado anteriormente:

- Optou-se por colocar a origem do referencial 0 na base do robô, com o eixo z_0 a apontar para cima, e o eixo x_0 alinhado com a saliência na base, que se pode observar no alçado principal na figura 1. O eixo y_0 foi obtido pela regra da mão direita;
- Em seguida, definiu-se z_1, com sentido contrário a x_0, e, sendo este perpendicular a z_0, definiu-se x_1 no ponto de intercepção entre z_0 e z_1, com sentido contrário a y_0. O eixo y_1 foi obtido pela regra da mão direita;
- Para os referenciais 2 e 3, uma vez que z_1, z_2 e z_3 são todos paralelos entre si, há uma infinidade de posições para x_2 e x_3 que se podem escolher. Por simplicidade, mais uma vez, escolheu-se a configuração que obriga d_2 e d_3 serem zero, alinhando x_2 e x_3 com z_0. Os eixos y_2 e y_3 foram obtidos pela regra da mão direita para os respectivos referenciais;
- O eixo z_4 foi definido apontando para cima, e, sendo este perpendicular a z_3, definiu-se x_4 no ponto de intercepção entre z_3 e z_4, com sentido contrário a y_3. O eixo y_4 foi obtido pela regra da mão direita;

- Para o referencial 5, visto que z_5 é perpendicular a z_4, x_5 é definido de forma semelhante ao que foi feito previamente para x_4, sendo estes dois eixos paralelos e com o mesmo sentido. Finalmente, definiu-se y_5 pela regra da mão direita e o referencial 6 é semelhante ao referencial 5.

O diagrama juntas encontra-se representado na figura 2:

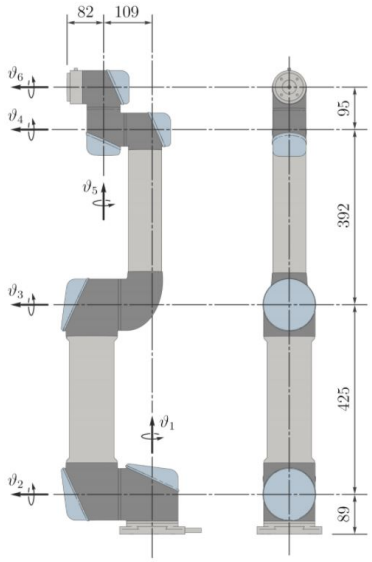


Figura 1: Desenho técnico do robô

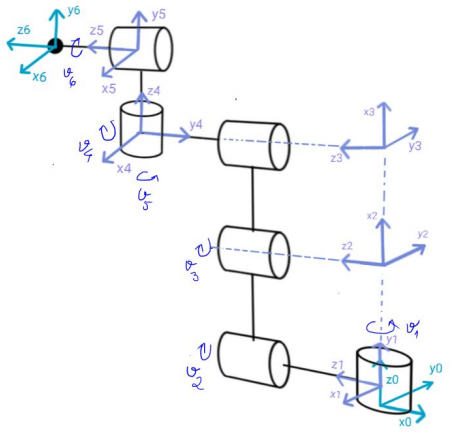


Figura 2: Diagrama de articulações e *frames* do robô

Atendendo ao modelo encontrado e às dimensões do robô, expressas em milímetros na figura 1, preencheu-se a tabela de parâmetros, tabela 1, com os valores de d_i, θ_i, a_i e α_i.

Tabela 1: Tabela de parâmetros segundo a configuração Denavit-Hartenberg

link	$d_i \quad [m]$	$\theta_i \quad [rad]$	$a_i \quad [m]$	$\alpha_i \quad [rad]$	Offset $[rad]$
1	0.089	q1	0	$\frac{\pi}{2}$	$-\frac{\pi}{2}$
2	0	q2	0.425	0	$\frac{\pi}{2}$
3	0	q3	0.392	0	0
4	0.109	q4	0	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$
5	0.095	q5	0	$\frac{\pi}{2}$	0
6	0.082	q6	0	0	0

4 Modelo de *Simulink* para Cinemática Direta

Após a obtenção da tabela com os parâmetros da convenção DH procedeu-se à implementação do problema da cinemática direta em *Simulink*, como se pode observar na figura 3.

É de salientar que para todas as implementações, à excepção da verificação da matriz Jacobiana, foi utilizada uma animação do braço robótico obtida pela função *vrrobot_G18*. Esta animação recebe os valores de q_i e representa o robô UR5 na respectiva configuração.

Primeiro é necessário definir a matriz DH e, para tal, chama-se a função *Robot_G18*. Em seguida, através da função *DHKin_G18* obtém-se a matriz de rotação R_6^0 e o vetor de posição p_6^0 do

referencial local 6 no referencial 0. Recorreu-se à função do *Matlab*, *matlabFunctionBlock* para criar um bloco *MATLAB Function* no *Simulink*, que tem como argumentos de entrada os ângulos q_i, correspondentes a θ_i na matriz DH, com i=1,...,6, e como argumentos de saída as matrizes referidas anteriormente. Por motivos de organização nos modelos seguintes, a Jacobiana geométrica, mencionada na secção 5, também é um argumento de saída deste bloco, mas esta encontra-se ligada a um bloco *Terminator* nesta implementação. De notar que a matriz DH implementada no *Matlab* possui os *offsets* a 0.

No modelo em *Simulink* foram pré-definidas as configurações correspondentes às posições de verificação, e, em adição, é possível colocar valores arbitrários de q manualmente durante a simulação, observando o robô a mover-se na animação, em tempo real.

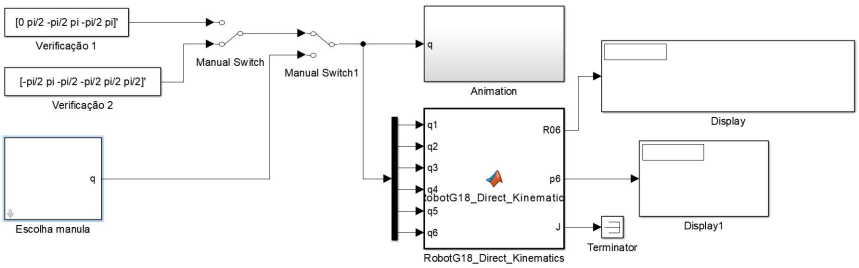


Figura 3: Modelo de *Simulink* para Cinemática Direta

4.1 Validação do Modelo

De forma a averiguar se a implementação está correta, procedeu-se à sua verificação colocando o robô nas posições fornecidas no apêndice A do enunciado do projeto, figura 5. As configurações q foram obtidas através do ângulo entre os eixos x_{i-1} e x_i.

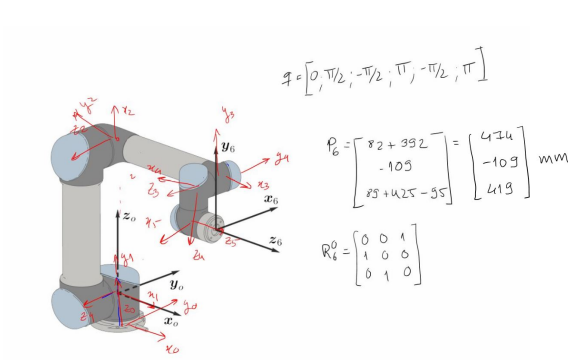


Figura 4: Configuração 1 do robô para verificação do modelo

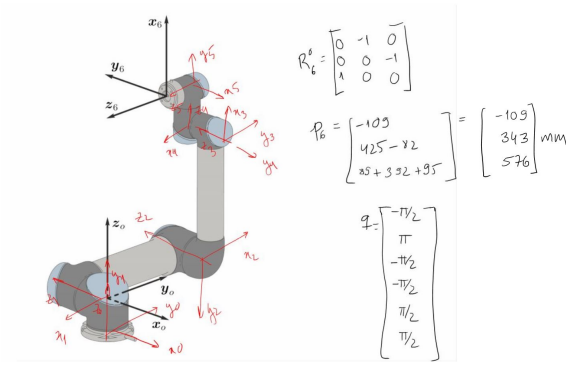


Figura 5: Configuração 2 do robô para verificação do modelo

- **Posição 1**
  -Vetor de Entrada: $q = [0° \quad 90° \quad -90° \quad 180° \quad -90° \quad 180°]^T$
  -Vetor de Posição do *end-effector*: $p_6^0 = [0.474 \quad -0.109 \quad 0.419]^T$ (m)
  -Matriz de Rotação: $R_6^0 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$

- **Posição 2**
  -Vetor de Entrada: $q = [-90° \quad 180° \quad -90° \quad -90° \quad 90° \quad 90°]^T$
  -Vetor de Posição do *end-effector*: $p_6^0 = [-0.109 \quad 0.343 \quad 0.576]^T$ (m)
  -Matriz de Rotação: $R_6^0 = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix}$

Nas imagens seguintes é possível observar a animação, matriz de rotação e vetor de posição obtidos no *Simulink* para as entradas referentes a cada uma das posições:

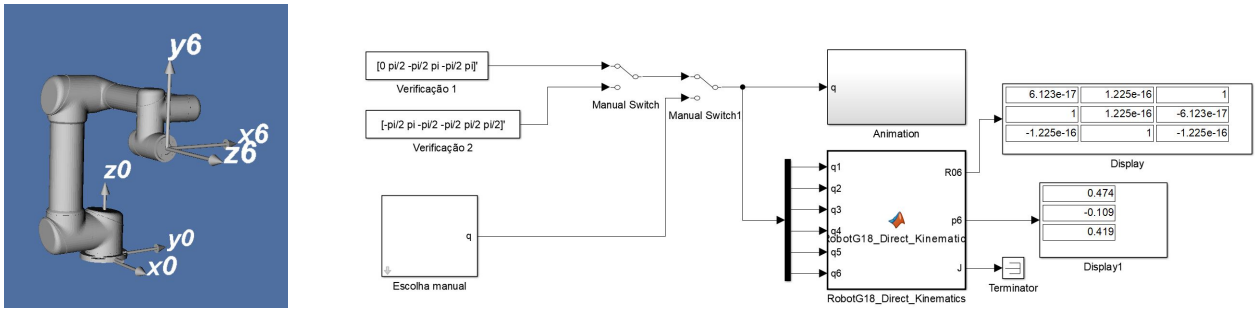


Figura 6: Validação do modelo na primeira posição

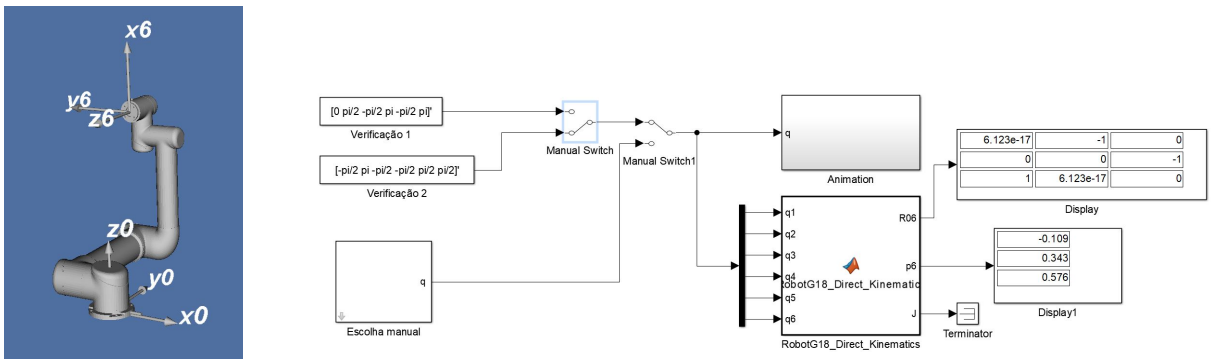


Figura 7: Validação do modelo na segunda posição

Conclui-se assim que a cinemática direta foi bem implementada.

5 Modelo de *Simulink* para a Jacobiana Geométrica

Cada coluna da matriz Jacobina Geométrica corresponde a uma junta no manipulador, sendo as primeiras 3 linhas relacionadas com a velocidade linear do *end-effector* e as últimas 3 relacionaas com a velocidade angular. O cálculo da Jacobiana geométrica é dividido, então, em 2 parcelas:

$$J = \begin{bmatrix} J_{P_1} & \dots & J_{P_n} \\ J_{O_1} & \dots & J_{O_n} \end{bmatrix} \tag{1}$$

Estas parcelas relacionam-se com a velocidade linear e com a velocidade angular da seguinte forma:

$$\dot{p}_e = \sum_{i=1}^{n} J_{P_i} \dot{q}_i \tag{2}$$

$$\omega_e = \sum_{i=1}^{n} J_{O_i} \dot{q}_i \tag{3}$$

- J_p **de acordo com o tipo de junta**
  Para uma junta prismática é calculado da seguinte forma:

$$J_P i = \overrightarrow{z}_{i-1} \tag{4}$$

  Para uma junta de rotação a fórmula é a seguinte:

$$J_P i = \overrightarrow{z}_{i-1} \times (\overrightarrow{p}_e - \overrightarrow{p}_{i-1}) \tag{5}$$

- J_o **de acordo com o tipo de junta**
  Para uma junta prismática é calculado da seguinte forma:

$$J_O i = 0^T \tag{6}$$

  Para uma junta de rotação a fórmula é a seguinte:

$$J_O i = \overrightarrow{z}_{i-1} \tag{7}$$

Os termos nas equações acima são dados por:

$$\overrightarrow{z}_{i-1} = R_1^0(q_1)...R_{i-1}^{i-2}(q_{i-1})z_0 \tag{8}$$

$$\tilde{p}_e = A_1^0(q_1)...A_n^{n-1}(q_n)\tilde{p}_0 \tag{9}$$

$$\tilde{p}_{i-1} = A_1^0(q_1)...A_{i-1}^{i-2}(q_n)\tilde{p}_0 \tag{10}$$

$$z_0 = [0;0;1] \tag{11}$$

$$\tilde{p}_0 = [0;0;0;1] \tag{12}$$

Para calcular a Jacobiana criou-se a função *GeoJacobian.m*, que recebe a matriz DH, e o tipo de cada junta do manipulador através de uma *string*, 'P' para juntas prismáticas e 'R' para juntas de revolução. A função começa por definir os vetores z_0 e p_0, que, juntamente com a utilização da função *DKin_G18.m*, são usados para calcular as matrizes T_i, A_i na função, a partir das quais se obtêm as matrizes de rotação, os vetores de posição, e os eixos z, implementando as equações 8 a 10. Um vez calculados e guardados todos os valores necessários de p_i e z_i, calcula-se-se para cada junta, atendendo ao tipo de cada uma, J_P e J_O, utilizando as equações 4 a 7, para se obter a Jacobiana. Esta é uma matriz 6x6, uma vez que o robô tem seis juntas. Finalmente, utilizou-se a função de *Matlab*, *matlabFunctionBlock*, para criar um bloco de *Simulink* que recebe as configurações das das juntas e devolve a Jacobiana. Como referido na secção anterior, este bloco também tem R_6^0 e p_6^0 como argumentos de saída, para simplificar os modelos.

5.1 Validação do Modelo

De forma a verificar o cálculo da Jacobiana, implementou-se o modelo *Simulink* apresentado na figura 8, baseado nas equações 2 e 3.

Neste modelo definiu-se a velocidade nas juntas como uma sinusoide, a qual se integrou para obter os q, que são os argumentos de entrada no bloco da cinemática direta e Jacobiana. Os valores

de J_P e de J_O foram verificados independentemente um do outro. Para o primeiro calculou-se a diferença entre $\dot{p}_e$ e $\sum_{i=1}^{n} J_{P_i}\dot{q}_i$, e para o segundo calculou-se a diferença entre ω_e e $\sum_{i=1}^{n} J_{O_i}\dot{q}_i$, sendo que $\omega_e = [S_{32} \quad S_{13} \quad S_{21}]$, onde $S = \dot{R}R^T$.

Para confirmar se o modelo está bem implementado é preciso verificar que os valores nos diferentes *displays*,que apresentam o erro, é 0. Uma vez que os valores se encontram muito próximos de 0 pode-se concluir que a Jacobiana foi bem calculada.

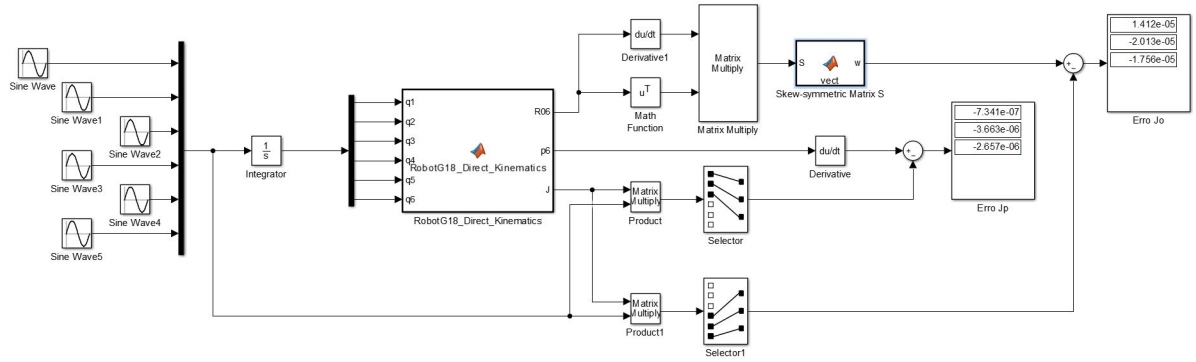


Figura 8: Verificação da Jacobiana Geométrica

5.2 Singularidades

A matriz Jacobiana é função dos ângulos θ_i, q_i na implementação, com i=1,...,n. O robô UR5 apresenta 6 graus de liberdade, portanto, se o manipulador não for redundante, a característica da Jacobiana deverá ser igual a 6. Contudo, certas configurações afectam a característica da matriz, isto é, reduzem a mobilidade da estrutura. Essas configurações denominam-se singularidades. Quando uma estrutura está numa singularidade pode haver infinitas soluções para o problema cinemático. Existem 2 tipos de singularidades:

- **Singularidades de fronteira:**
  Ocorrem quando o robô é levado ao seu limite, isto é, quando está totalmente esticado, figura 9 e 10, ou recolhido. Estas singularidades podem ser evitadas se o manipulador não for levado ao limite do seu espaço operacional.

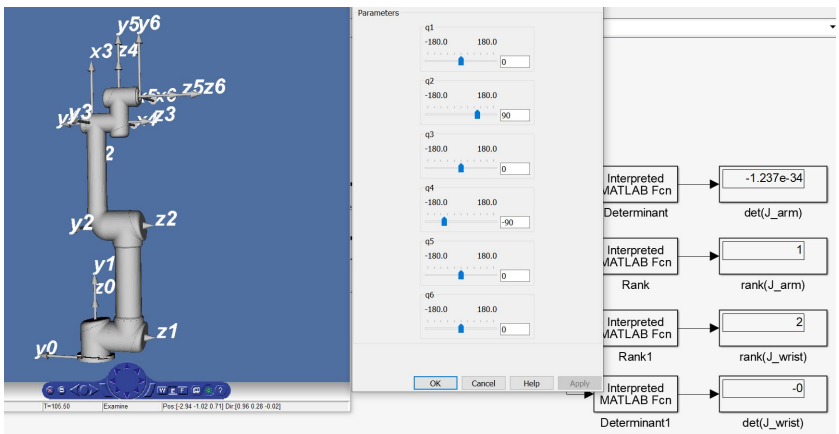


Figura 9: Exemplo 1 de singularidades de fronteira

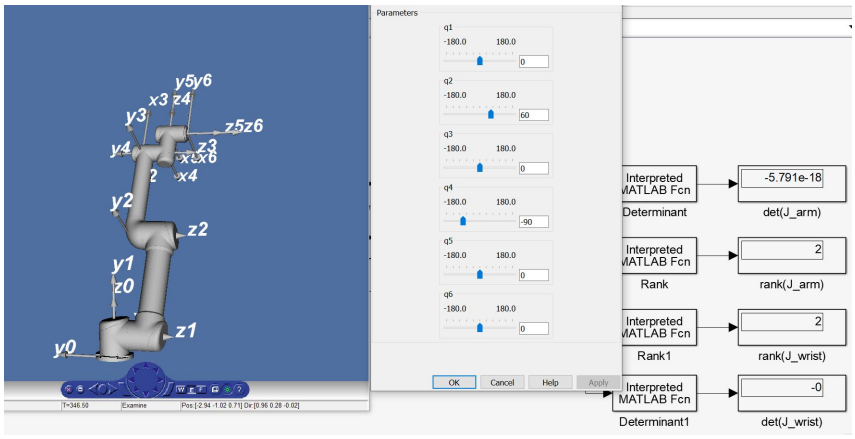


Figura 10: Exemplo 2 de singularidades de fronteira

- **Singularidades internas:**
  Este tipo de singularidade ocorre dentro do espaço operacional, figuras 11 e 12, quando existe o alinhamento de dois ou mais eixos de movimento. Este é o tipo de singularidade mais difícil de evitar.

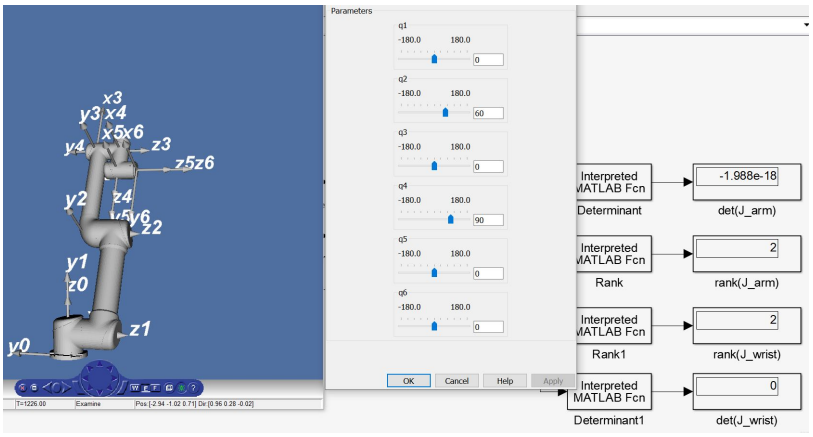


Figura 11: Exemplo 1 de singularidade interna

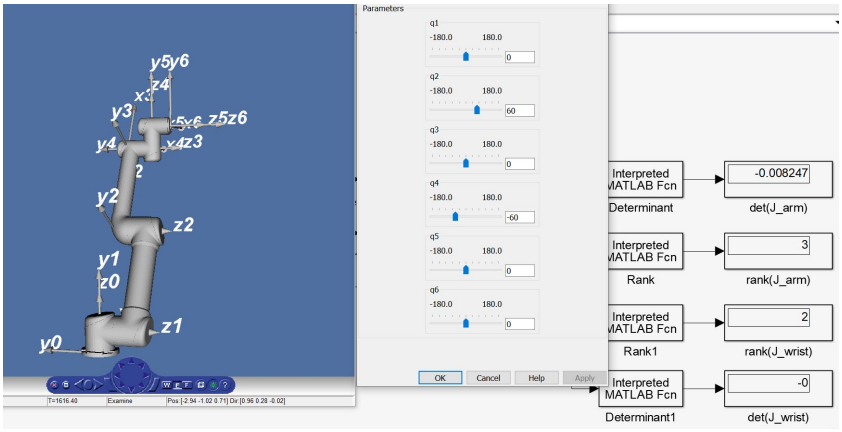


Figura 12: Exempo de 2 singularidade interna

De forma a resolver este problema começou-ses por fazer uma divisão entre as singularidades do braço e do pulso. As singularidades do braço estão associadas às 3 primeiras juntas sendo que as do

pulso estão associadas às 3 últimas. Desta forma, é possível dividir a Jacobiana da seguinte forma:

$$J = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} \tag{13}$$

Sendo J_{ij} uma matriz 3x3.

Pode-se observar as singularidades do braço analisando J_{11}, e as do pulso analisando J_{22}. Existem duas abordagens para ver se o manipulador se encontra numa singularidade ou perto dela:

- **Colunas da Jacobiana linearmente independentes:**
  Através da característica da matriz pode-se encontrar o número de colunas linearmente independentes. Se a característica for menor que 3, no caso de J_{11} para o braço, e J_{22} para o pulso, o robô encontra-se numa singularidade.
- **Determinante da Jacobiana:** Através do cálculo do determinante da matriz é possível perceber se o manipulador se está a aproximar de uma singularidade ou se está numa singularidade. Se o valor do determinante for igual a 0 o robô encontra-se numa singularidade. É possível saber se se trata de uma singularidade do braço ou do pulso se se calcular o determinante de J_{11} e de J_{22}, para o braço e pulso respectivamente. Se o valor do determinante estiver próximo de 0 então o robô encontra-se na vizinhança de uma singularidade. Esta abordagem vai ser importante na implementação da cinemática inversa.

Para verificar quais são as singularidades existentes no robô em estudo utilizou-se o seguinte modelo de *Simulink*:

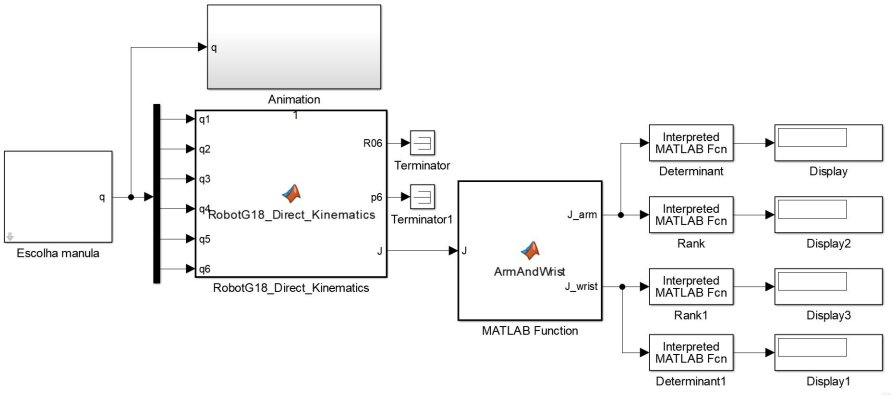


Figura 13: Verificação das singularidades

Em que o bloco que possui a função *ArmAndWrist* retira da matriz Jacobiana as matrizes J_{11} e J_{22} para serem analisadas independentemente.

Verificou-se que existem 3 singularidades no pulso, quando o ângulo θ_5 é igual a 0,-π, ou π. Estes valores correspondem ao alinhamento do eixo z_5 com o eixo z_3 como se pode observar na figura 12.

No caso do braço, este apresenta singularidades quando está completamente esticado, e os eixos z_3, z_2 e z_1 e z_4 se encontram no mesmo plano, figuras 9 e 11.

Outra singularidade que limita o movimento do robô que é representada por um cilindro em torno do eixo z_0 e com raio igual a 0.109m,como se pode observar na figura 14. Na figura 9 a característica de J_{11} é 1 por que não só o braço está esticado, como o manpulador se encontra nesta singularidade.

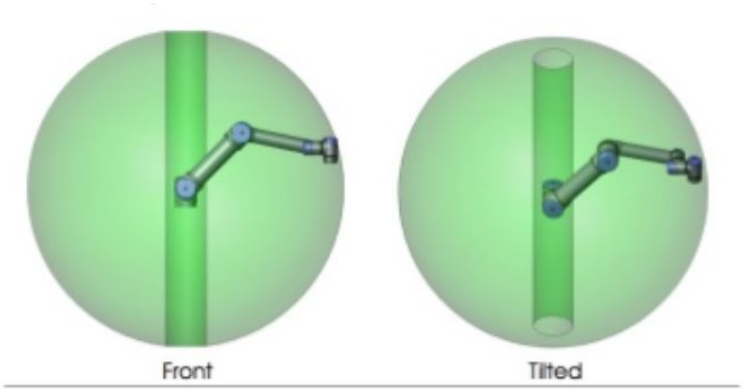


Figura 14: Espaço Operacional do Robô[3]

6 Cinemática Inversa em Anel Fechado

No método da cinemática inversa introduz-se a posição do *end-effector*, p_0^6, e rotação, R_0^6, desejadas e o sistema devolve os ângulos de rotação θ_i, q_i nas implementações, para cada junta. O problema de cinemática inversa é mais complexo que o da cinemática direta, visto que podem existir múltiplas soluções para as mesmas matrizes de posição e rotação. O robô UR5 em específico possui oito soluções possíveis.

A cinemática inversa apenas apresenta soluções *closed-form* para manipuladores que têm uma estrutura cinemática simples. Estas limitações devem-se ao facto da relação entre as variáveis de junta e as variáveis operacionais ser extremamente não-linear. Por outro lado, a cinemática diferencial permite um mapeamento linear entre a velocidade nas juntas e a velocidade operacional, embora esta seja dependente da configuração das juntas, equação 14.

$$v_e = \begin{bmatrix} \dot{p_e} \\ \omega_e \end{bmatrix} = J(q)\dot{q} \tag{14}$$

Onde v_e, $\dot{p_e}$, e ω_e correspondem aos o vetores de velocidade, velocidade linear, e velocidade angular do *end-effector*, $\dot{q}$ é o vetor de velocidades nas juntas, e J é a matriz Jacobiana que depende das configurações das juntas.

Se J tiver característica igual ao número de graus de liberdade do manipulador, as velocidades nas juntas podem ser obtidas pela expressão 15:

$$\dot{q} = J^{-1}(q)v_e \tag{15}$$

Admitindo que as condições iniciais são conhecidas, as configurações das juntas podem ser obtidas por integração da velocidade, $\dot{q}$, ao longo do tempo. Este método de cinemática inversa é independente da solvabilidade da estrutura cinemática, contudo a Jacobiana tem de ser quadrada e todas as suas colunas devem ser linearmente independentes, para que esta seja invertível. Caso contrário, como visto anteriormente, o manipulador é redundante e ocorre uma singularidade.

Na implementação numérica, a velocidade nas juntas é obtida por,

$$q_{(t_{k+1})} = q_{(t_k)} + J^{-1}(q_{(t_k)})v_{e(t_k)}\Delta t \tag{16}$$

Contudo, os valores obtidos por esta computação não satisfazem a equação 15. Para corrigir isto, implementa-se então um algoritmo de cinemática inversa que tem em conta o erro da posição e o erro da rotação. Este método trata-se de cinemática inversa em anel fechado.

O erro da posição é definido por,

$$e_p = p_d - p_e(q) \tag{17}$$

Onde p_d denota o vetor de posição desejada do *end-effector* no referencial 0, e p_e denota a posição instâtanea do *end-effector* para uma dada configuração das juntas obtida por cinemática inversa.

A expressão do erro da orientação depende de uma representação particular da orientação do *end-effector*, nomeadamente de ângulos de Euler, ângulo e eixo, e do quaternião unitário. Nesta implementação utilizou-se um algoritmo de cinemática inversa baseado no quaternião unitário. Definindo $\mathscr{Q}_d = \{\eta_d, \varepsilon_d\}$ e $\mathscr{Q}_e = \{\eta_e, \varepsilon_e\}$ como os quaterniões associados à orientação desejada, R_d, e à orientação obtida por cinemática inversa, R_e, o erro associado à orientação pode então ser descrito pela matriz $R_d R_e^T$, e, em termos do quaternião, pode-se escrever $\Delta\mathscr{Q} = \{\Delta\eta, \Delta\varepsilon\}$, em que,

$$\Delta\mathscr{Q} = \mathscr{Q}_d * \mathscr{Q}_e^{-1} \tag{18}$$

Assim, o erro associado à orientação é dado por:

$$e_O = \Delta\varepsilon = \eta_e(q)\varepsilon_d - \eta_d\varepsilon_e(q) - S(\varepsilon_d)\varepsilon_e(q) \tag{19}$$

Em que $S()$ é um operador antissimétrico. Definindo a matiz instantânea de orientação obtida por cinemática inversa, R_e,

$$R_e = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Pode-se escrever η_e e ε_e em função de R_e como,

$$\eta_e = 0.5\sqrt{r_{11} + r_{22} + r_{33} + 1} \tag{20}$$

$$\varepsilon_e = 0.5 \begin{bmatrix} sign(r_{32} - r_{23})\sqrt{r_{11} - r_{22} - r_{33} + 1} \\ sign(r_{13} - r_{31})\sqrt{r_{22} - r_{11} - r_{33} + 1} \\ sign(r_{21} - r_{12})\sqrt{r_{33} - r_{11} - r_{22} + 1} \end{bmatrix} \quad , \quad \forall \eta_e \neq 0 \tag{21}$$

Onde a função $sign(x)$ é igual a 1 se $x \geq 0$, -1 se $x \leq 0$ e 0 se $x = 0$.

O vetor ε_e, expresso acima na equação 21, é posteriormente utilizado na implementação da cinemática inversa em anel fechado. Assumindo que o manipulador não passa por nenhuma singularidade, a solução para a velocidade nas juntas pode ser obtida, para um manipulador não redundante, através da inversa da Jacobiana por:

$$\dot{q} = J^{-1}(q) \begin{bmatrix} \dot{p}_d + K_P e_P \\ \dot{\phi}_d + K_O e_O \end{bmatrix} \tag{22}$$

Onde K_P e K_O são matrizes de ganhos positivas definidas.

Desta forma, é possível representar o algoritmo da cinemática inversa em anel fechado pelo diagrama de blocos apresentado na figura 15.

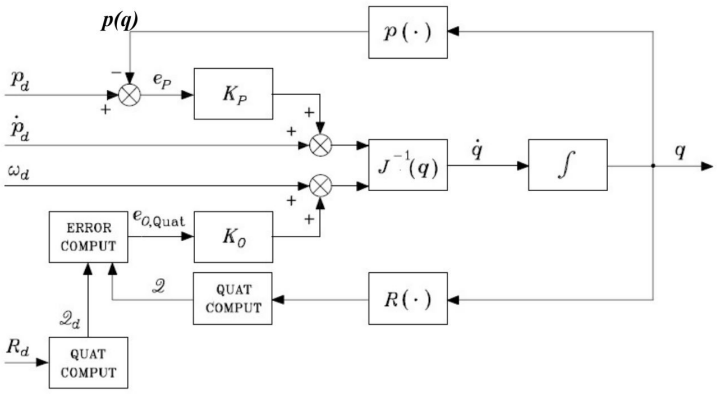


Figura 15: Diagrama de Blocos do algoritmo de Cinemática Inversa

É de salientar que, na proximidade de singularidades, pode-se substituir a inversa da matriz Jacobiana pela sua transposta, uma vez que esta deixa de ser invertível se o manipulador se encontrar numa singularidade.

6.1 Modelo de *Simulink*

Na implementação em *Simulink* da cinemática inversa, figuras 16 e 17, criou-se um subsistema denominado *CLIK* (*Closed-Loop Inverse Kinematics*) onde se reproduziu o diagrama de blocos ilustrado na figura 15. Este subsistema tem como entradas a posição e orientação do *end-effector* desejadas, as velocidades linear e angular do *end-effector* desejadas (que nesta implementação são 0 visto que não se pretende que o *end-effector* siga uma trajectória), as configurações iniciais, e as matrizes de ganhos K_P e K_O. Como saída,s apresenta o erro da orientação, o erro da posição, a velocidade nas juntas (que não foi analisada neste trabalho), e as configurações das juntas que servem de entrada à animação do braço robótico.

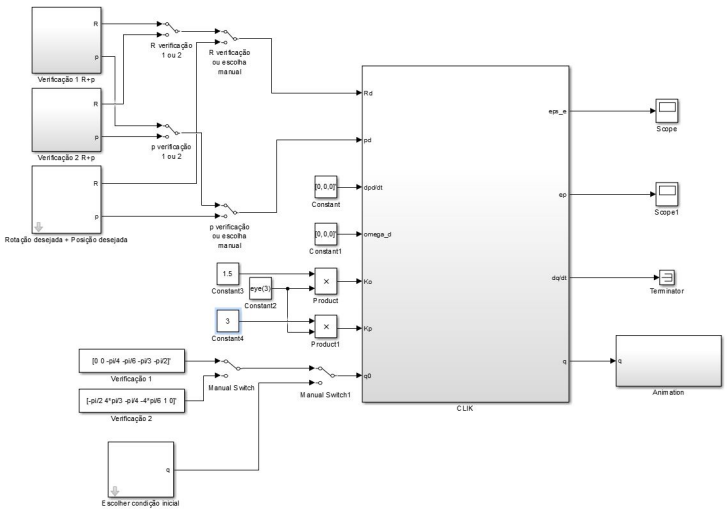


Figura 16: Entradas e saídas do subsistema da cinemática inversa em anel fechado

No bloco que permite escolher a posição e a orientação desejadas deve-se colocar as coordenadas desejadas do *end-effector* em relação ao referencial 0, em metros, e os ângulos de Euler desejados, em graus. Dentro deste bloco os ângulos são transformados na matriz de rotação desejada pela matriz de rotação ZYZ, equação 23. O valores de posição e rotação de validação foram

pré-definidos no modelo para a sua rápida utilização,tal como as correspondentes condições iniciais escolhidas.

$$R(\Phi) = R_z(\phi)R_y(\theta)R_z(\psi) = \begin{bmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi + c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix} \tag{23}$$

Onde a se usou as abreviaturas: $cos(\phi) = c_\phi$, $cos(\theta) = c_\theta$ e $cos(\psi) = c_\psi$, e igualmente para os senos.

Relativamente aos ganhos K_P e K_O, verificou-se que o aumento destes ganhos torna o sistema mais oscilatório, aproximando-o mais de singularidades. O aumento de K_O em particular tem um grande influência no comportamento do manipulador, sendo que se verificou que se podia aumentar mais o K_P, sem que o manipulador passasse por uma singularidade, se se mantivesse o valor de K_O baixo. Isto salienta a necessidade do manipulador ter de seguir uma trajectória. Também se verificou que, se o manipulador acertar a posição antes de finalizar a correção da orientação, são evitadas singularidade mais facilmente. Assim, é conveniente utilizar um valor de K_O menor que K_P, definindo-se então $K_P = 3$ e $K_O = 1.5$.

Para tornar a simulação robusta a situações em que o robô passa na vizinhança de singularidades, a cada iteração é calculado o determinante da matriz Jacobiana, e se este for maior que 0.005 efectua-se a inversão da matriz, caso contrário, utiliza-se a sua transposta.

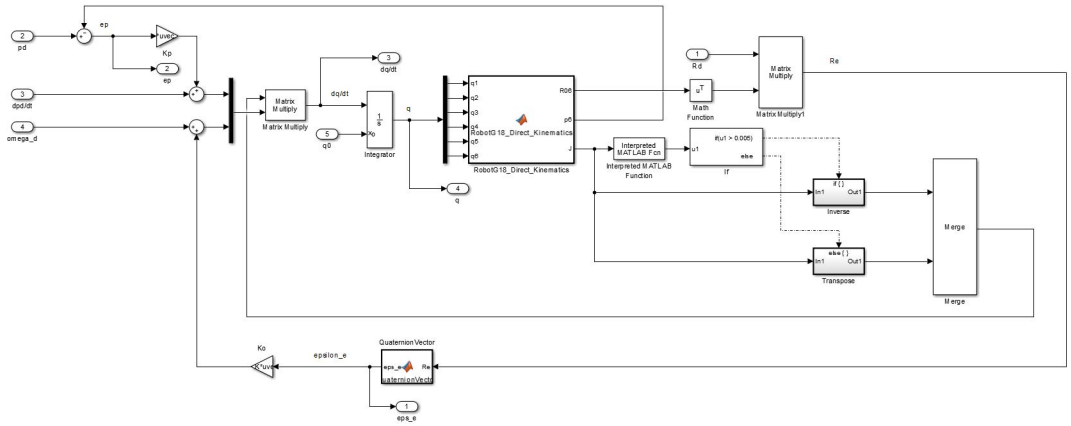


Figura 17: Implementação em *Simulink* da cinemática inversa em anel fechado

6.1.1 Validação do Modelo

Para a validação do modelo obtido, recorreu-se, mais uma vez, às configurações ilustradas na figura 5, cujos vetores q já foram previamente apresentados na secção 4. As posições iniciais foram escolhidas atendendo ao facto que o robô UR5 tem oito soluções possíveis para uma dada posição e orientação (combinações do braço estar à esquerda ou à direita, o cotovelo estar para cima ou para baixo, e o pulso estar para cima ou para baixo), e que se deve evitar ao máximo que o manipulador passe por uma singularidade.

- **Posição 1**
  -Vetor de Posição do *end-effector*: $p_6^0 = [0.474 \quad -0.109 \quad 0.419]^T$ (m)

-Ângulos de Euler: $\phi = 0°$, $\theta = 90°$, $\psi = 90°$
-Posição Inicial: $[0 \quad 0 \quad -\pi/4 \quad -\pi/6 \quad -\pi/3 \quad -\pi/2]^T$

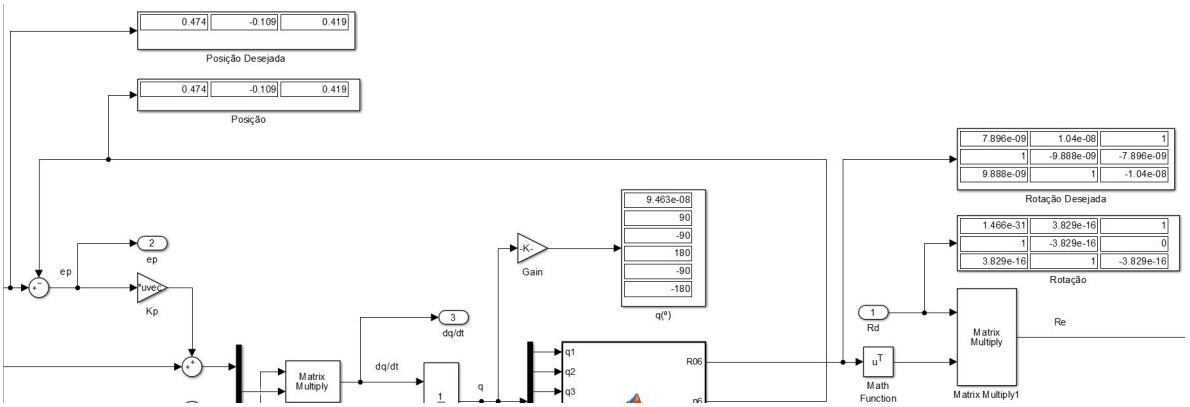


Figura 18: Resultados da primeira posição do robô para verificação do modelo

- **Posição 2**
-Vetor de Posição do *end-effector*: $p_6^0 = [-0.109 \quad 0.343 \quad 0.576]^T$ (m)
-Ângulos de Euler: $\phi = 90°$, $\theta = -90°$, $\psi = 0°$
-Posição Inicial: $[-\pi/2 \quad 4\pi/3 \quad -\pi/4 \quad -4\pi/6 \quad 1 \quad 0]^T$

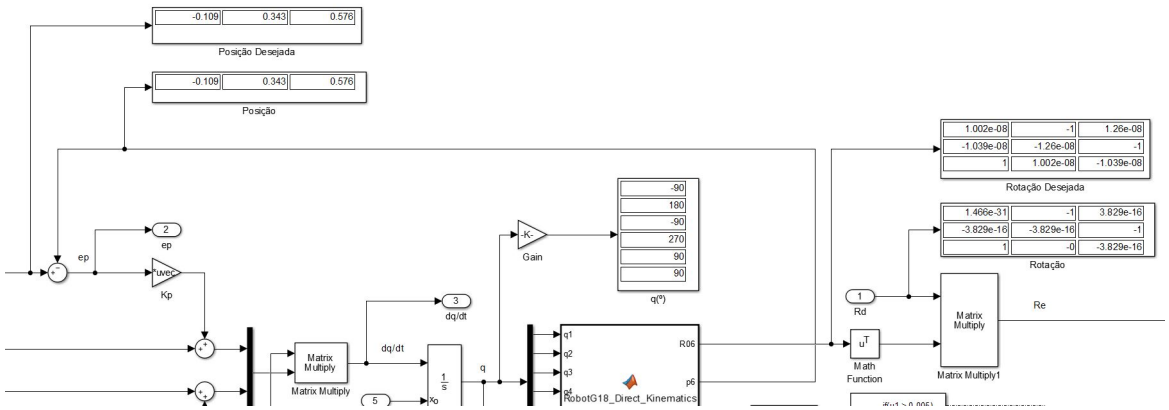


Figura 19: Resultados da segunda posição do robô para verificação do modelo

Dos dados experimentais observou-se que na verificação 1 o ângulo q_6 obtido foi $-180°$ em vez de 180°, e na verificação 2 o ângulo q_4 obtido foi 270° em vez de $-90°$, como definidos anteriormente. Estas disparidades não foram consideradas significantes, uma vez que o manipulador se encontra, efectivamente, na configuração desejada. Assim, pode-se verificar que a implementação da cinemática inversa está a funcionar como era previsto.

7 Solução *Closed-form* para a Cinemática Inversa

Como abordado anteriormente, através da cinemática inversa pode-se obter os valores das variáveis de junta do manipulador através da posição e orientação desejadas do *end-effector*, isto é, obter os valores de θ_i para i=1,...,n através de T_n^0, em que:

$$T_n^0 = \begin{bmatrix} R_n^0 & p_n^0 \\ 0 & 1 \end{bmatrix} \tag{24}$$

$$R_n^0 = \begin{bmatrix} x_x & y_x & z_x \\ x_y & y_Y & z_y \\ x_z & y_z & z_z \end{bmatrix} \tag{25}$$

$$p_n^0 = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{26}$$

De forma a resolver o problema em causa utilizou-se a abordagem do desacoplamento.

- **Cálculo de θ_1:**

  Para encontrar o ângulo θ_1 é preciso de encontrar o vetor de posição da *frame* 5 em relação à *frame* 0. Tal é conseguido fazendo a translação de distância d_6 na direção negativa de z_6, como se pode observar na figura seguinte:

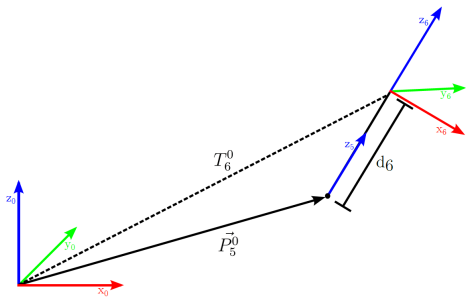


Figura 20: Esquema geométrico para encontrar a origem do *frame* 5[3]

$$\overrightarrow{p}_5^0 = T_6^0 \begin{bmatrix} 0 \\ 0 \\ -d_6 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{27}$$

Com a localização da *frame* 5 pode-se desenhar a seguinte vista aérea do robô:

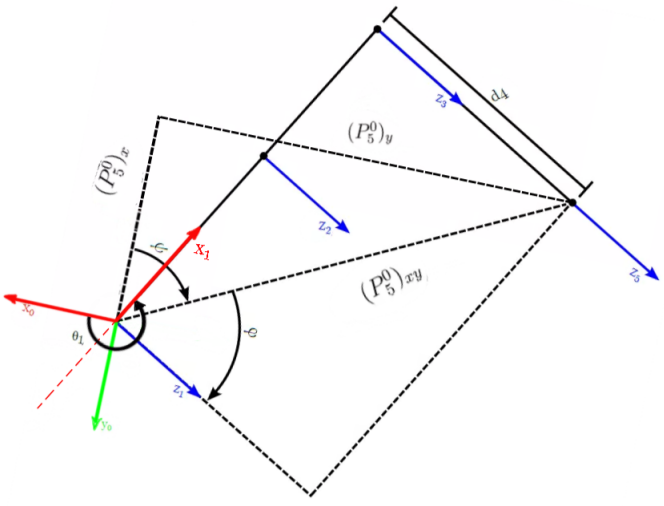


Figura 21: Esquema geométrico para encontrar θ_1

Da figura 21 pode-se observar que $\theta_1 = \phi + \psi + \pi/2$ onde:

$$\psi = atan2\left((p_5^0)_y, (p_5^0)_x\right) \tag{28}$$

$$\phi = \pm arccos\left(\frac{d_4}{(p_5^0)_{xy}}\right) \tag{29}$$

Através da equação 29 pode-se observar que existem duas posições para o ombro: para a esquerda ou para a direita. Na implementação deste método, visando resolver esta situação, analisam-se as coordenadas x e y da posição desejada do *end-effector*, determinando a posição do ombro de acordo com o quadrante em que se encontra a projeção de O_6 no plano $O_0x_0y_0$. Verifica-se também que esta equação só tem solução caso $d_4 > (p_5^0)_{xy}$. Esta particularidade forma um cilindro inacessível, que foi abordado na subsecção 5.2.

- **Cálculo de θ_5:**

  É possível calcular-se θ_5 uma vez que θ_1 já é conhecido. Recorrendo à vista aérea anterior mas considerando a localização da *frame* 6 em relação à *frame* 1:

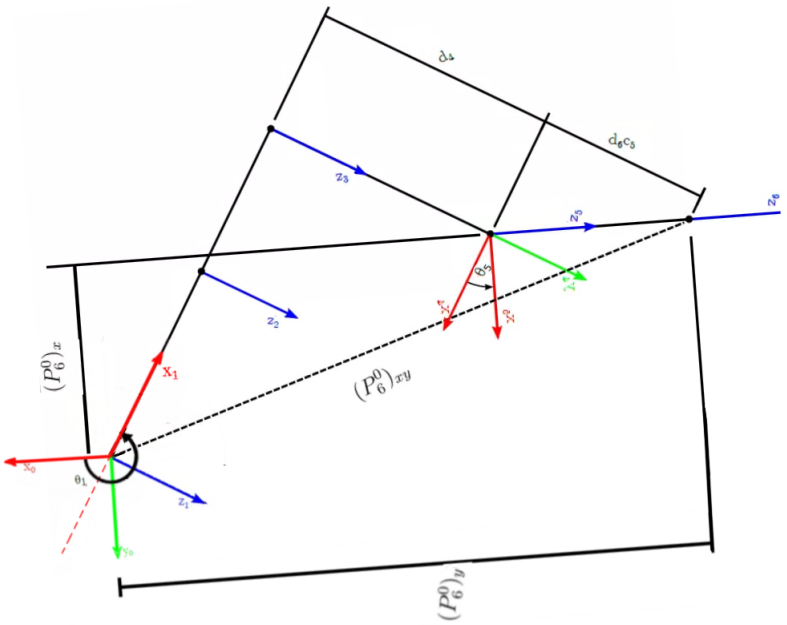


Figura 22: Esquema geométrico para encontrar θ_5

Observando que:

$$(p_6^1)_z = d_6 cos(\theta_5) + d_4 = (p_6^0)_x sin(\theta_1) - (p_6^0)_y cos(\theta_1) \tag{30}$$

Desta forma, é possível resolver a equação anterior em ordem a θ_5:

$$\theta_5 = \pm arccos\left(\frac{(p_6^1)_z - d_4}{d_6}\right) \tag{31}$$

Através da equação 31 verifica-se que existem duas posições para o pulso: para baixo ou para cima. Na implementação deste método, para solucionar este problema analisou-se a primeira coluna da matriz de rotação desejada, mais especificamente a última entrada que representa a posição de x_6 em relação a z_0. Verificando se este valor é positivo ou negativo, é possível averiguar em que direção se encontra o pulso.

- **Cálculo de θ_6:**

  Em seguida calculou-se θ_6 a partir de θ_5. Começa-se por encontrar a transformação da *frame* 6 para a *frame* 1:

$$T_1^6 = ((T_1^0)^{-1} T_6^0)^{-1} \tag{32}$$

  Através da matriz de transformação homogénea T_1^6 chega-se às seguintes igualdades:

$$sin(\theta_6)sin(\theta_5) = z_y \tag{33}$$

$$cos(\theta_6)sin(\theta_5) = z_x \tag{34}$$

  Desta forma, pode-se resolver a equação anterior em relação a θ_6:

$$\theta_6 = atan2(z_y, z_x) \tag{35}$$

- **NOTA:** As 3 juntas que faltam podem ser calculadas considerando que estas formam um manipulador com 3 juntas de rotação.

- **Cálculo de θ_3:**

  Em primeiro lugar determina-se a posição da *frame* 3 em relação à *frame* 1:

$$T_4^1 = T_6^1 T_4^6 = T_6^1 (T_5^4 T_6^5)^{-1} \tag{36}$$

$$\overrightarrow{p}_3^1 = T_4^1 \begin{bmatrix} 0 \\ d_4 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{37}$$

  Pode-se então desenhar o plano que contém as *frames* 1 a 3 como se pode observar na seguinte figura:

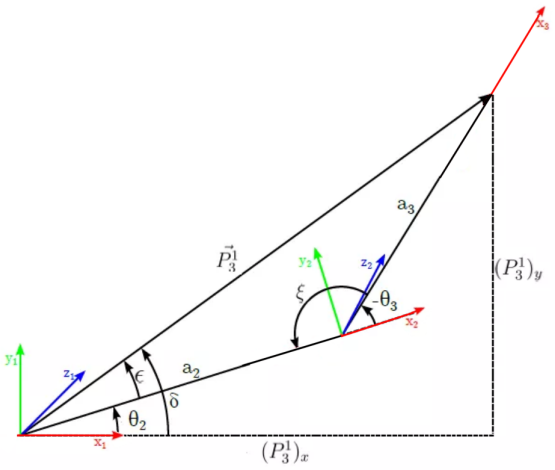


Figura 23: Esquema geométrico para encontrar $\theta_3 e \theta_2$

Pela lei dos cossenos, $a^2+b^2-2abcos(\beta)=c^2$ em que β é o ângulo entre os lados a e b do triângulo,tem-se que:

$$cos(\varepsilon)=\frac{||\overrightarrow{p}_3^1||^2-a_2^2-a_3^2}{2a_2a_3} \tag{38}$$

E através das propriedades dos cossenos:

$$cos(\varepsilon)=-cos(\pi-\varepsilon)=-cos(-\theta_3)=cos(\theta_3) \tag{39}$$

Resolvendo as equações 38 e 39 em ordem a θ_3:

$$\theta_3=\pm arcos\left(\frac{||\overrightarrow{p}_3^1||^2-a_2^2-a_3^2}{2a_2a_3}\right) \tag{40}$$

Esta equação tem solução desde que o arcos $\in$ [-1,1]. Valores elevados da norma do vetor $\overrightarrow{p}_3^1$ podem levar a que o valor exceda 1. Fisicamente este fenómeno traduz-se na direção máxima que o robô atinge em todas as direções, criando um espaço operacional esférico abordado na subsecção 5.2.

- **Cálculo de θ_2:**

  Através da figura 23 verifica-se que:

$$\theta_2=\delta-\in \tag{41}$$

$$\delta=atan2\left((p_3^1)_y,(p_3^1)_x\right) \tag{42}$$

$$\frac{sin(\varepsilon)}{||\overrightarrow{p}_3^1||^2}=\frac{sin(\in)}{a_3} \tag{43}$$

  Desta forma, obtem-se o ângulo θ_2:

$$\theta_2=atan2\left((p_3^1)_y,(p_3^1)_x\right)-arcsin\left(\frac{a_3sin(\varepsilon)}{||\overrightarrow{p}_3^1||^2}\right) \tag{44}$$

  Existem duas soluções para θ_2 e θ_3, que são cotovelo para cima ou cotovelo para baixo.

- **Cálculo de θ_4:**

  Por fim, falta apenas o cálculo de θ_4. Em primeiro lugar calcula-se T_4^3:

$$T_4^3=T_1^3T_4^1=(T_2^1T_3^2)^{-1}T_4^1 \tag{45}$$

  Através da primeira coluna da matriz de transformação homogénea T_4^3 pode-se calcular:

$$\theta_4=atan2(x_y,x_x) \tag{46}$$

Após o cálculo dos seis ângulos é fácil ver que existem, de facto, oito soluções possíveis.

7.1 Modelo de *Simulink*

De forma a perceber se a solução apresentada anteriormente está correta implementou-se o seguinte modelo apresentado na figura 24 no *Simulink*. Este modelo consiste num bloco *MATLAB Function* onde estão definidas as equações abordadas anteriormente para o cálculo dos ângulos θ_i com i=1,..,6, q_i na implementação. As entradas para este bloco são a matriz de rotação R_6^0 e o vetor de posição p_6^0 desejados, e a saída são os ângulos q_i. Mais uma vez, as posições e rotações de verificação foram pré-definidas, e para ajuda à validação de resultados também se utilizou a animação do manipulador, e *displays* para as configurações das juntas em graus, e para as matrizes de posição e rotação obtidas pela cinematica direta, a partir das configurações provenientes da cinemática inversa.

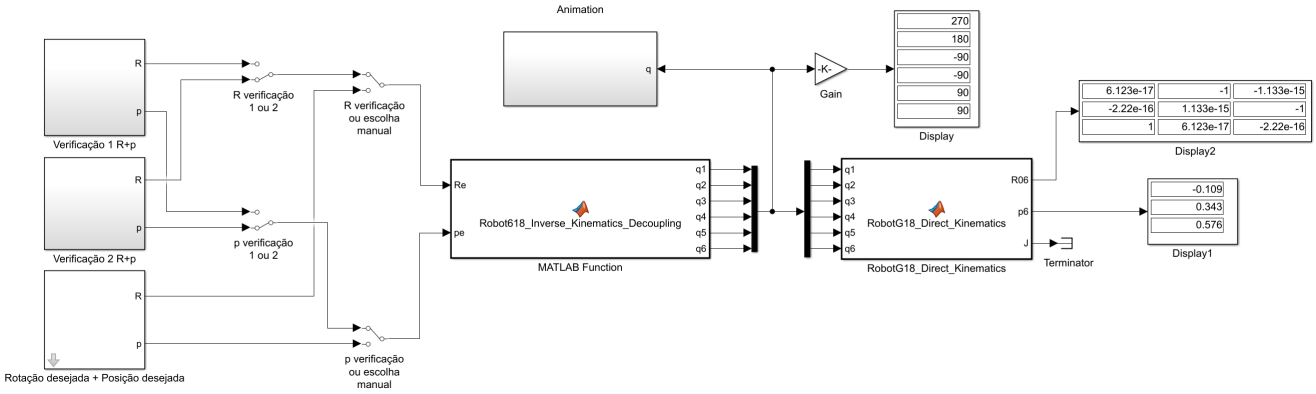


Figura 24: Modelo Simulink para a Cinemática Inversa

7.1.1 Validação do Modelo

Para verificar a implementação foram usadas, mais uma vez, as configurações da figura 5 dadas no enunciado. Os resultados encontram-se apresentados nas figuras seguintes:

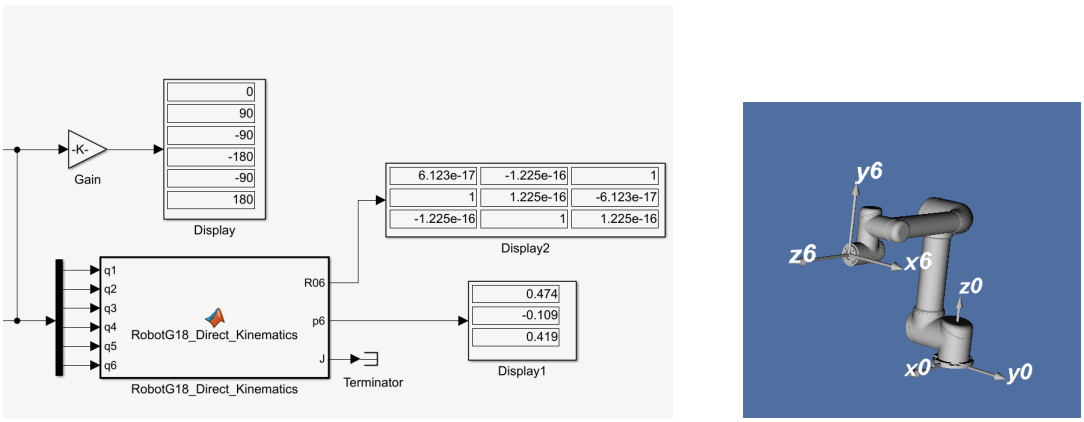
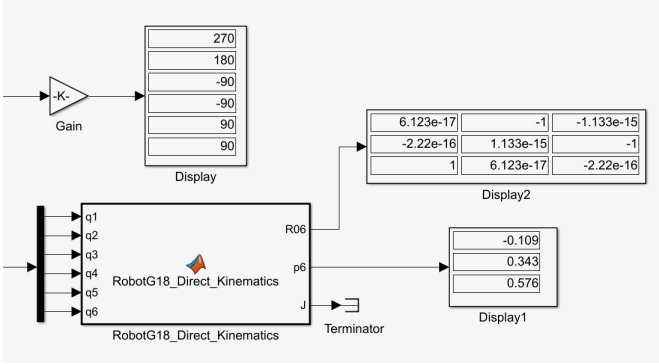


Figura 25: Resultados e configuração obtida para a verificação 1



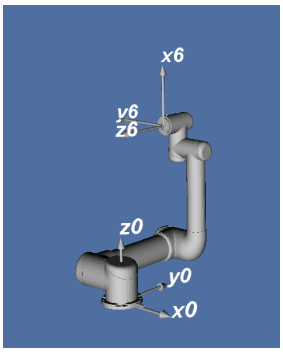


Figura 26: Resultados e configuração obtida para a verificação 2

Para a segunda configuração o ângulo q_1, ou θ_1 na notação teórca, obtido é 270º e não -90º, como inicialmente definido, contudo a posição é exactamente a mesma e, visto que o manipulador não tem de seguir nenhuma trajectória, este erro não foi considerado significante. Um fenómeno semelhante ocorre na primeira configuração em q_4, ou θ_4, em que se obteve o resultado de -180º em vez de 180º. Conclui-se, assim, que os ângulos e a configuração obtidas para ambos os exemplos estão corretos, e, portanto, a solução apresentada para a cinemática inversa foi bem implementada.

8 Conclusões

Após o estudo da cinemática do robô UR5 conseguiu-se concluir que a implementação dos diferentes conceitos e métodos foi bem sucedida, uma vez que os resultados obtidos foram os pretendidos. Ao implementar estes métodos foi possível entender melhor o funcionamento de um robô manipulador e os conceitos ligados à sua cinemática, assim como as limitações do seu movimento, nomeadamente o conceito de singularidade e a forma como estas podem ser ultrapassadas.

Referências

[1] *Robotics - Modelling, Planning and Control*, B. Siciliano, L. Sciavicco, L. Villani and G. Oriolo 2009 Springer-Verlag

[2] Apontamentos da disciplina Robótica de Manipulação: Jorge Martins 2009 IST

[3] Ryan Keating, *UR5 Inverse Kinematics*, M.E. 530.646, Johns Hopkins University, Baltimore, Maryland, EUA, 2016