Project Report

Jinrui Zhang

December 10, 2023

Contents

1 Overview of Robot simulation

The following pseucode code only demonstrate how the three main functions work.

1.1 Pseudocode of Overview

```
clear;clc;
load variables;
load length_of_Links;

T06 = Initial() % Initialize the robot

Visual() % Visualize specific robot configuration

Q = Inverse(T06) % Run inverse kinematic computation

Workspace()% Draw workspace
```

2 Support Functions

2.1 Rotation Functions

This function are used to output the transformation matrix after rotation by X axis.

Algorithm 1: Rotation by X axis

```
  Input: Rotation Degree $D$ in radian, *Option*
  Output: Transformation Matrix $R_x$
1 if number of Input lager than 1 and Option equals to d then
2     $D \leftarrow D \times \frac{\pi}{180}$;
3 end
4 $R_x \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(D) & -\sin(D) & 0 \\ 0 & \sin(D) & \cos(D) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$;
```

The other two rotation functions, rotation by the Y and Z axis, work similarly, do not need to paraphrase again.

2.2 Translation Function

This function is used to output the transformation matrix after translation.

Algorithm 2: Translation

```
  Input: Translation distance align $X, Y, Z$ axis
  Output: Transformation Matrix $T$
1 $T \leftarrow \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$;
```

2.3 Transformation Matrix from DH Parameters

There are two ways to derive the transformation matrix: John J. Craig's way and Mark W. Spong's way. We need John J. Craig's way because it is very helpful during the computation of inverse kinematics.

And the transformation matrix is derived from a multiple of four matrix instead of being directly defined. Because π in MATLAB is not an irrational number but a 15-digit decimal. If we directly define the transformation matrix, the output of $\sin(\pi)$ is very close to 0 but not equal to 0. To avoid it from happening, the product of four matrixes will provide a precise outcome.

2.3.1 Transformation Matrix in way of John J. Craig

```
Algorithm 3: Transformation Matrix with DH Parameters -John J. Craig
  Input: $theta, d, a, alpha$
  Output: Transformation Matrix $T$
1 $T \leftarrow rot_x(alpha) \times trans(a,0,0) \times rot_z(theta) \times trans(0,0,d)$
```

2.3.2 Transformation Matrix in way of Mark W. Spong

```
Algorithm 4: Transformation Matrix with DH Parameters -Mark W. Spong
  Input: $theta, d, a, alpha$
  Output: Transformation Matrix $T$
1 $T \leftarrow rot_z(theta) \times trans(0,0,d) \times rot_z(alpha) \times trans(a,0,0)$
```

2.4 Auxiliary functions

2.4.1 Round matrixes

Matrix simplification function, which ensures that the final outcome of the matrix is numerical and rational. The algorithm is shown as Algorithm 5.

```
Algorithm 5: Round matrixes
  Input: Matrix $A$
  Output: Rounded matrix $T$
1 $row, col \leftarrow size(A)$;
2 foreach $row,col$ do
3   $element(row, col) \leftarrow$ retain three decimal places of $element(row, col)$
4 end
```

2.4.2 Auxiliary Sine

This function aims to output the precise outcome of the sine and cosine functions. Because π in MATLAB is a 15-digit decimal, it could not output $\sin(\pi)$ as 0. This function can make the inverse kinematic computation procedure much more accurate. The algorithm is shown as Algorithm 6.

There is also an auxiliary cosine function, which has the same purpose. Do not need to paraphrase again.

3 Main Functions

There are three main functions: Forward, Visual, Inverse. These functions implement forward kinematics, inverse kinematics, and visualization functionality, respectively.

3.1 Initial

This function serves the dual purpose of initializing the robot and performing forward kinematics. The syntax of the function is:

```
Algorithm 6: Auxiliary Sine
  Input: A degree $D$ in radian
  Output: A precise degree $S$ in radian
1 if $\sin(D) < 0.001$ then
2 |  $S \leftarrow 0$
3 end
4 if $\sin(D) > 0.001$ then
5 |  $S \leftarrow D$
6 end
```

```
function T06 = Initial(q1,q2,q3,q4,q5,q6,length_list)
        pass
end
```

It requires seven parameters for operation.The first six parameters correspond to the variables of each joint individually. The seventh parameter is a vector encapsulating the length of each link. If no entries are provided, the function will calculate the forward kinematics symbolically. In the case of six entries, the function will numerically compute the robot's outcome with each link length set to 5. With seven entries, the function will generate the transformation matrix based on user's specific input.

```
Algorithm 7: Initialize and Forward Kinematic
   Input: joint variables:$q_1, q_2, q_3, q_4, q_5, q_6$; Length of links: $Length$
   Output: Transformation matrix ${}^0T_6$
 1 if number of entries is not 0,6 or 7 then
 2 |  print error
 3 end
 4 if number of entries is 6 then
 5 |  foreach $i$ do
 6 |  |  $theta_i \leftarrow q_i$;
 7 |  |  $l_i \leftarrow 5$
 8 |  end
 9 end
10 if number of entries is 7 then
11 |  foreach $i$ do
12 |  |  $l_i \leftarrow Length(i)$
13 |  end
14 end
15 foreach $Link_i$ do
16 |  $Link_i = [theta_i, d_i, a_i, alpha_i]$;
17 |  ${}^{i-1}T_i \leftarrow Algorithm4(Link_i(1), Link_i(2), Link_i(3), Link_i(4))$
18 end
19 ${}^0T_6 \leftarrow {}^0T_1 \times {}^1T_2 \times {}^2T_3 \times {}^3T_4 \times {}^4T_5 \times {}^5T_6$
```

3.2 Visual

This function is designed to visualize the robot at a specific configuration, necessitating seven parameters for its operation. The syntax of the function is:

```
function [X,Y,Z] = Visual(q1,q2,q3,q4,q5,q6,length_list)
        pass
end
```

The initial six parameters are dedicated to the variables of each joint individually, while the seventh parameter is a vector encapsulating the length of each link. If no entries are provided, the function will display the robot in a homogeneous configuration. In the case of six entries, signifying the provision of variables for each joint, the function will depict the specific configuration with each link length set to

5. When seven entries are provided, the function will plot the configuration exactly as specified by the client.The output of this function is the coordinate matrix for each end of the link.

Algorithm 8: Visualize the robot

```
   Input: joint variables:$q_1, q_2, q_3, q_4, q_5, q_6$; Length of links: $Length$
   Output: coordinate matrix $[X, Y, Z]$
1  if number of entries is not 0,6 or 7 then
2      print error
3  end
4  if number of entries is 6 then
5      foreach $i$ do
6          $theta_i \leftarrow q_i$;
7          $l_i \leftarrow 5$
8      end
9  end
10 if number of entries is 7 then
11     foreach $i$ do
12         $l_i \leftarrow Length(i)$
13     end
14 end
15 foreach $Link_i$ do
16     $Link_i = [theta_i, d_i, a_i, alpha_i]$;
17     ${}^{i-1}T_i \leftarrow Algorithm4(Link_i(1), Link_i(2), Link_i(3), Link_i(4))$
18 end
19 foreach ${}^0T_i$ do
20     ${}^0T_i \leftarrow {}^0T_{i-1} \times {}^{i-1}T_i$
21 end
22 $X \leftarrow [0, {}^0T_1(1,4), {}^0T_2(1,4), {}^0T_3(1,4), {}^0T_4(1,4), {}^0T_5(1,4), {}^0T_6(1,4)]$;
23 $Y \leftarrow [0, {}^0T_1(2,4), {}^0T_2(2,4), {}^0T_3(2,4), {}^0T_4(2,4), {}^0T_5(2,4), {}^0T_6(2,4)]$;
24 $Z \leftarrow [0, {}^0T_1(3,4), {}^0T_2(3,4), {}^0T_3(3,4), {}^0T_4(3,4), {}^0T_5(3,4), {}^0T_6(3,4)]$;
25 plot diagram where coordinate of each axis is X,Y,Z
```

3.3 Inverse

This function is designed to determine inverse kinematic computation using the transformation matrix derived from John J. Craig's DH table. This approach is chosen for its potential ease in deriving inverse kinematics.

The mathematical reasoning process is then applied. (For convenient purposes, during derivation s_1 indicates $\sin(\theta_1)$, c_1 indicates $\cos(\theta_1)$, s_{12} indicates $\sin(\theta_1 + \theta_2)$.

$$ {}^0T_6 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ x_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^0T_1(\theta_1) \times {}^1T_2(\theta_2) \times {}^2T_3(\theta_3) \times {}^3T_4(\theta_4) \times {}^4T_5(\theta_5) \times {}^5T_6(\theta_6) \tag{1} $$

1. θ_1

$$ {}^0T_1^{-1}(\theta_1){}^0T_6 = {}^1T_2(\theta_2){}^2T_3(\theta_3){}^3T_4(\theta_4){}^4T_5(\theta_5){}^5T_6(\theta_6) $$

$$ \begin{bmatrix} c_1 & s_1 & 0 & 0 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ x_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^1T_6 \tag{2} $$

Multiple the matrixes and get the (2,4) element of the product.

$$ -s_1 p_x + c_1 p_y = d_2 \tag{3} $$

assign $\rho = \sqrt{p_x^2 + P_y^2}$; $\phi = atan2(p_y, p_x)$

$$\sin(\phi - \theta_1) = \frac{d_2}{\rho}$$

$$\cos(\phi - \theta_1) = \pm\sqrt{1 - (\frac{d_2}{\rho})^2}$$

$$\therefore \phi - \theta_1 = atan2(\frac{d_2}{\rho}, \pm\sqrt{1 - (\frac{d_2}{\rho})^2})$$

$$\therefore \theta_1 = atan2(p_x, p_y) - atan2(d_2, \pm\sqrt{p_x^2 + p_y^2 - d_2^2})$$

2. θ_3

   After figure out θ_1, We pay attention to the (1,4) and (3,4) element of equation (2)

$$c_1p_x + s_1p_y = a_3c_{23} - d_4s_{23} + a_2c_2 \tag{4}$$
$$-p_x = a_3s_{23} + d_4c_{23} + a_2s_2 \tag{5}$$

   square equations (3), (4), and (5), and sum them up, we have

$$a_3c_3 - d_4s_3 = K$$

   where $K = \frac{p_x^2+p_y^2+p_z^2-a_2^2-a_3^2-d_2^2-d_4^2}{2a_2}$

$$\therefore \theta_3 = atan2(a_3, d_4) - atan2(K, \pm\sqrt{a_3^2 + d_4^2 - K^2})$$

3. θ_2

   For θ_2, we need to multiple ${}^0T_3^{-1}$ to both side of equation(1)

$$\begin{bmatrix} c_1c_{23} & s_1c_{23} & s_{23}p_z & -a_2c_3 \\ -c_1s_{23} & -s_1s_{23} & -s_{23} & -a_2c_3 \\ -s_17c_1 & 0 & -a_2s_3 & \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ x_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} =^3 T_6 \tag{6}$$

   Multiple the matrixes and get the element(1,4) and (2,4) of the product. We get

$$c_1c_{23}p_x + s_1c_{23}p_y - s_{23}p_z - a_2c_3 = a_3 \tag{7}$$
$$-c_1s_{23} - s_1s_{23}p_y - c_{23}p_z + a_2s_3 = d_4 \tag{8}$$

   Solve the system of equations (6) and (7), we get

$$s_{23} = \frac{(-a_3 - a_2c_3)p_z + (c_1p_x + s_1p_y)(a_2s_3 - d_4)}{p_z^2 + (c_1p_x + s_1p_y)^2}$$

$$c_{23} = \frac{(-d_4 + a_2s_3)p_z - (c_1p_x + s_1p_y)(-a_2c_3 - a_4)}{p_z^2 + (c_1p_x + s_1p_y)^2}$$

   Since s_{23} and c_{23} have the same denominator and they are both positive.

   $\theta_{23} = \theta_2 + \theta_3 = atan2[-(a_3+a_3c_3)p_z+(c_1p_x+s_1p_y)(a_2s_3-d_4), (-d_4+a_2s_3)p_x+(c_1p_x+s_1p_y)(a_2c_3+a_3)]$

$$\therefore \theta_2 = \theta_{23} - \theta_3$$

4. θ_4

   For now, we will pay attention to the (1,3) and (3,3) element of equation(6), we have

$$a_x c_1 c_{23} + a_y s_1 c_{23} - a_x s_2 3 = -c_4 s_5$$
$$-a_x s_1 + a_y c_1 = s_4 s_5$$

   As long as $\theta_5 \neq 0$

$$\theta_4 = atan2(-a_x s_1 + a_y c_1, -a_x c_1 c_{23} - a_y s_1 c_{23} + a_x s_{23})$$

5. θ_5

$$^0T_4^{-1}(\theta_1, \theta_2, \theta_3, \theta_4)^0T_6 = ^4T_5(\theta_5)^5T_6(\theta_6) \tag{9}$$

   Since we already have derived $\theta_1, \theta_2, \theta_3, \theta_4$, we could derive $^0T_t^{-1}$

$$^0T_4^{-1} = \begin{bmatrix} c_1c_{23}c_4 + s_1s_4 & s_1c_{23}c_4 - c_1s_4 & s_{23}c_4 & -a_2c_3c_4 + d_3s_4 - a_3c_4 \\ -c_1c_{23}s_4 + s_1c_4 & -s_1c_{23}s_4 - c_1c_4 & s_{23}s_4 & a_2c_3s_4 + d_3c_4 + a_3s_4 \\ -c_1s_{23} & s_1s_{23} & -c_{23} & a_2s_3 - d_4 \end{bmatrix}$$

   Now, we pay attention to (1,3) and (3,3) element of equation (9). We have,

$$a_x(c_1c_{23}c_4 + s_1s_4) + a_y(s_1c_{23}c_4 - c_1s_4) - a_z(s_{23}c_4) = -s_5$$
$$a_x(-c_1s_{23}) + a_y(-s_1s_{23}) + a_z(-c_{23}) = c_5$$
$$\therefore \theta_5 = atan2(s_5, c_5)$$

6. θ_6

$$^0T_5^{-1}(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)^0T_6 = ^5T_6(\theta_6) \tag{10}$$

   we pay attention to (3,1) and (1,1) element of equation (10). We have,

$$-n_x(c_1c_{23}s_4 - s_1c_4) - n_y(s_1c_{23}s_4 + c_1c_4) + n_z(s_{23}s_4) = s_6$$
$$n_z[(c_1c_{23}4 + s_1s_4)c_5 - c_1s_{23}s_5] + n_y[(s_1c_23c_4 - c_1s_4)c_5 - s_1s_{23}s_5] = c_6$$
$$\therefore \theta_6 = atan2(s_6, c_6)$$

The syntax of the function is:

```
function Q = Inverse(T)
        pass
end
```

The algorithm is just translating the mathematical derivation process into MATLAB, do not need to paraphrase again here.

3.4 Workspace

This function designed to show the workspace of this robot with Monte Carlo method. The Monte Carlo method, also known as the statistical simulation method, is a numerical technique that uses random sampling (pseudo-random numbers) to solve mathematical problems. This method is easy to implement with graphical display capabilities, offering fast and simple calculations while avoiding the complexity of mathematical derivations and calculations. It is suitable for solving the workspace of any articulated robotic arm, with no restrictions on the range of joint variable changes, and its error is also independent of dimensionality. The syntax of the function is:

```
function Workspace(variables_lim)
        pass
end
```

Variables_lim is a 4 by 2 matrix that includes the minimal and maximum values of each variable.

In *Workspace* function, I call *Transform* function to output the transformation matrix. *Transform* function has exactly the same procedure to compute forward kinematic, but only do some simplification to reduce time complexity. *Transform* function can exponentially reduce the time consumption of *Workspace* function.

Algorithm 9: Draw the workspace

```
Input: joint variables limitation: $var\_lim$
Output: None
n ← 30000;
while $i \leq n$ do
    foreach $j$ do
        $q_j \leftarrow$ a random value from $q_j^{min}$ to $q_j^{max}$
    end
    $T \leftarrow transform(q)$;
    $x(i) \leftarrow T(1,4)$;
    $y(i) \leftarrow T(2,4)$;
    $z(i) \leftarrow T(3,4)$;
end
foreach $point$ do
    plot each point with coordinate ($x(i)$,$y(i)$,$z(i)$)
end
```