Homework Assignment 3

Dynamics Of Non Linear Robotic Systems

Ilia Sevostianov

November 10, 2019

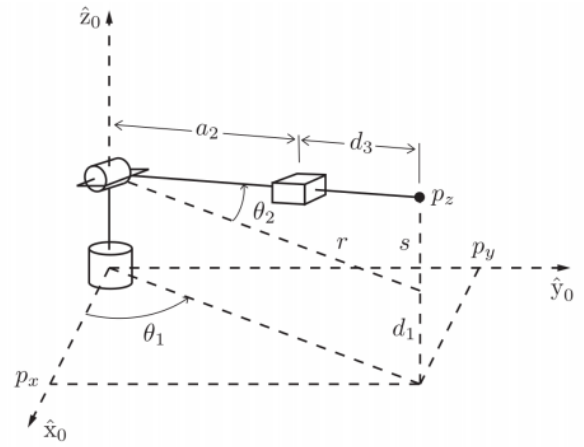

Figure 1: RRP robot.

Task 1:

Derive FK equations for the robot depicted in figure 1. Use θ_1, θ_2, d_3 as joint space variables, p_x, p_y, p_z as operational space variables. Parameters d_1, a_2 are known (assign them some positive values for succeeding tasks).

Solution

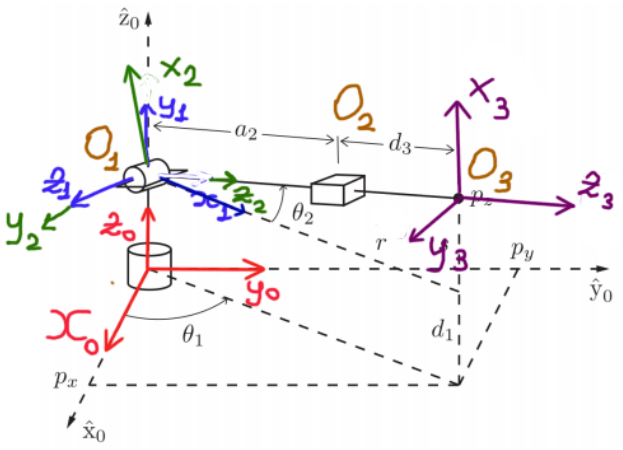

Figure 2: RRP robot: joint axes

Let's derive FK equations for the robot:

$$\boldsymbol{O_1} = \begin{bmatrix} 0 \\ 0 \\ d_1 \end{bmatrix} \tag{1}$$

$$\boldsymbol{O_2} = \begin{bmatrix} a_2 c_2 c_1 \\ a_2 c_2 s_1 \\ d_1 + a_2 s_2 \end{bmatrix} \tag{2}$$

$$\boldsymbol{O_3} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} a_2 c_2 c_1 + d_3 c_2 c_1 \\ a_2 c_2 s_1 + d_3 c_2 s_1 \\ d_1 + a_2 s_2 + d_3 s_2 \end{bmatrix} = \begin{bmatrix} c_2 c_1 (a_2 + d_3) \\ c_2 s_1 (a_2 + d_3) \\ d_1 + s_2 (a_2 + d_3) \end{bmatrix} \tag{3}$$

Task 2

Derive IK equations.

Solution

In IK we assume that we know coordinates of the tool:

$$\boldsymbol{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{4}$$

Let's start from angle θ_1 (see figure 2):

$$\theta_1 = atan2(p_y, p_x) \tag{5}$$

$$s = p_z - d_1, r = \sqrt{p_x^2 + p_y^2} \tag{6}$$

So, we can calculate θ_2:

$$\theta_2 = \frac{\pi}{2} + atan2(s, r) \tag{7}$$

To calculate d_3 we must write the next equation for the triangle:

$$r^2 + s^2 = (a_2 + d_3)^2 \tag{8}$$

$$r^2 + s^2 = a_2^2 + 2a_2 d_3 + d_3^2 \tag{9}$$

$$d_3^2 + 2a_2 d_3 + a_2^2 - r^2 - s^2 = 0 \tag{10}$$

Let's find d_3:

$$D = 4(a_2^2 - a_2^2 + r^2 + s^2) = 4(r^2 + s^2) \tag{11}$$

$$d_3 = \frac{-2a_2 \pm \sqrt{D}}{2} = \frac{-2a_2 \pm 2\sqrt{r^2 + s^2}}{2} = -a_2 \pm \sqrt{r^2 + s^2} \tag{12}$$

We exclude negative solution because it's impossible for out robot's configuration. So,

$$d_3 = -a_2 + \sqrt{r^2 + s^2} \tag{13}$$

Also, here is the second valid solution for θ_1 and θ_2:

$$\theta_1 = \pi + atan2(p_y, p_x) \tag{14}$$

$$\theta_2 = \frac{3\pi}{2} - atan2(s, r) \tag{15}$$

To conclude:

$$\boldsymbol{q} = \begin{bmatrix} \theta_1 \\ \theta_2 + \frac{\pi}{2} \\ d_3 \end{bmatrix} = \begin{bmatrix} atan2(p_y, p_x) \\ \frac{\pi}{2} + atan2(s, r) \\ -a_2 + \sqrt{r^2 + s^2} \end{bmatrix} \tag{16}$$

And the second solution:

$$\boldsymbol{q} = \begin{bmatrix} \theta_1 \\ \theta_2 + \frac{\pi}{2} \\ d_3 \end{bmatrix} = \begin{bmatrix} \pi + atan2(p_y, p_x) \\ \frac{3\pi}{2} - atan2(s, r) \\ -a_2 + \sqrt{r^2 + s^2} \end{bmatrix} \tag{17}$$

Task 3:

Compute the manipulator Jacobian for representation of linear and angular velocity of point **p**.

- Use classical approach (partial derivatives).
- Use geometric approach (cross products).

Solution

Jacobian is the matrix that relates the linear and angular velocity of the end-effector to the vector of joint velocities $\dot{q}(t)$:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{y} \\ w_x \\ w_y \\ w_z \end{bmatrix} = \boldsymbol{J}\dot{\boldsymbol{q}} \tag{18}$$

where J is Jacobian.

Classical Approach

In this approach the formula in figure 3 is used.

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial \mathbf{f}}{\partial x_1} & \cdots & \dfrac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_m}{\partial x_1} & \cdots & \dfrac{\partial f_m}{\partial x_n} \end{bmatrix}.$$

Figure 3: Formula for the Jacobian calculation

So, I made the matlab script to calculate the Jacobian of my sistem. The idea is that we obtained the coordinate O_i in FK. And to calculate Jacobian we must calculate partial derivatives of this matrixes by **q**.

Let me show the part of the matlab code, that calculates Jacobian, in figure 4(I assumed **q** as [theta1 theta2 d3]):

```
T = Calculate_T(3); % function for calculating homo. matrix T30.
O30 = T(1:3,4); % It's coordinates of the tool (O3)
dO30 = [diff(O30, theta1), diff(O30, theta2), diff(O30, d3)] ;  % Get Jacobian
Jacobian_classic = dO30;
```

Figure 4: MATLAB code to calculate Jacobian

After implementation this, I obtained the result shown in figure 5.

```
Jacobian_classic =

[ -sin(theta1)*sin(theta2)*(a2 + d3), cos(theta1)*cos(theta2)*(a2 + d3), cos(theta1)*sin(theta2)]
[  cos(theta1)*sin(theta2)*(a2 + d3), cos(theta2)*sin(theta1)*(a2 + d3), sin(theta1)*sin(theta2)]
[                                  0,             sin(theta2)*(a2 + d3),           -cos(theta2)]
```

Figure 5: Jacobian by Classical Approach

Geometrical Approach

In this approach the formulas in the table 1 is used.

Formulas for Jacobian's parts calculation		
	Prismatic	Revolute
Linear	$J_v = {R_{i-1}}^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	$J_v = {R_{i-1}}^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} ({d_n}^0 - {d_{i-1}}^0)$
Rotational	$J_w = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$	$J_w = {R_{i-1}}^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

Table 1: Formulas for Jacobian's parts calculation

So, I made the matlab script to calculate the Jacobian of my sistem. The idea is that we calculate T matrix for each joint and obtain R and d.

$$T = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} \tag{19}$$

Let me show the part of the matlab code, that calculates Jacobian, in figure 6.

```
%% Geometrical Approach for Jacobian

k = [0
    0
    1];

T10 = Calculate_T(1);
R10 = T10(1:3,1:3);
O10 = T10(1:3,4);
T20 = Calculate_T(2);
R20 = T20(1:3,1:3);
O20 = T20(1:3,4);
Jacobian = [cross(k, O30), cross(R10*k, O30-O10), R20*k
            k, R10*k, 0*k];
Jacobian_geom = simplify(Jacobian)
```

Figure 6: Jacobian calculation by Geometrical Approach

And the result you can see in figure 7.

```
Jacobian_geom =

[ -sin(theta1)*sin(theta2)*(a2 + d3), cos(theta1)*cos(theta2)*(a2 + d3), cos(theta1)*sin(theta2)]
[  cos(theta1)*sin(theta2)*(a2 + d3), cos(theta2)*sin(theta1)*(a2 + d3), sin(theta1)*sin(theta2)]
[                                  0,             sin(theta2)*(a2 + d3),           -cos(theta2)]
[                                  0,                       sin(theta1),                      0]
[                                  0,                      -cos(theta1),                      0]
[                                  1,                                 0,                      0]
```

Figure 7: Jacobian by Geometrical Approach

As you can see, it's the same as we got in classical approach, despite the difference that in this approach we have calculated J_w part also.

Task 4:

Analyze the Jacobian for singularities. Characterize each singular configuration if any.

Solution

There are a number of methods that can be used to determine the singularities of the Jacobian. Here, we will exploit the fact that a square matrix is singular when its determinant is equal to zero. But in general, it is difficult to solve the nonlinear equation det $J(q) = 0$.

The idea is that we partition the Jacobian J into 3 x 3 blocks as:

$$J = \begin{bmatrix} J_v \\ J_w \end{bmatrix} \tag{20}$$

And then find the dets of the parts to determine and analize the singularities.

So, I made the MATLAB code to determine the determinats. You can see it in the figure8.

```
%% Jacobian singularities
clc;clear;
[Jacobian_classic,Jacobian_geom] = JacobMonachSym();
J1 = Jacobian_geom(1:3, 1:3);

J2 = Jacobian_geom(4:6,1:3);

detJ1 = simplify(det(J1))

detJ2 = simplify(det(J2))
```

Figure 8: Code to calculate Jacobian determinants

The result is the derminant which we must set equal to zero to determine

singularities:

$$detJ(q)_v = sin(\theta_2 + \frac{\pi}{2})(a_2 + d_3)^2 \tag{21}$$

As we can see, determinant is equal to zero only when:

- $d_3 = -a_2$. In our configuration, I think, it's impossible,because of no offsets.
- $\theta_2 = \pi k - \frac{\pi}{2}, k = \overline{1, n}$. It's the place, when our manipulator looks up or down. Like in the figure 9. Here are singularities because this positions have infinity amount of solutions.

(a) "Up" Configuration (b) "Down" Configuration

Figure 9: Singularities of the robot

Task 5:

Compute the velocity of the tool frame when joint variables are changing with time as follows:
$\theta_1(t) = sin(t)$, $\theta_2(t) = cos(2t)$, $d_3(t) = sin(3t)$.

Add some fancy graphs showing evolution of all variables

Solution

For now, I have Jacobian of the robot in the symbolic view. Hence, I can substitute into the Jacobian Laws for θ_1, θ_2 and d_3 and then use the formula 18 to obtain tool frame velocities.

I also made MATLAB code, which algorithm is:

1. Input movement laws for the joint variables.
2. Calculate differentials of the joint variables.
3. Substitute into the Jacobian Laws for θ_1, θ_2 and d_3.

4. Calculate the tool frame velocities using formula 18.

5. Plot the result.

The fancy graphs you can see in the figure 10.

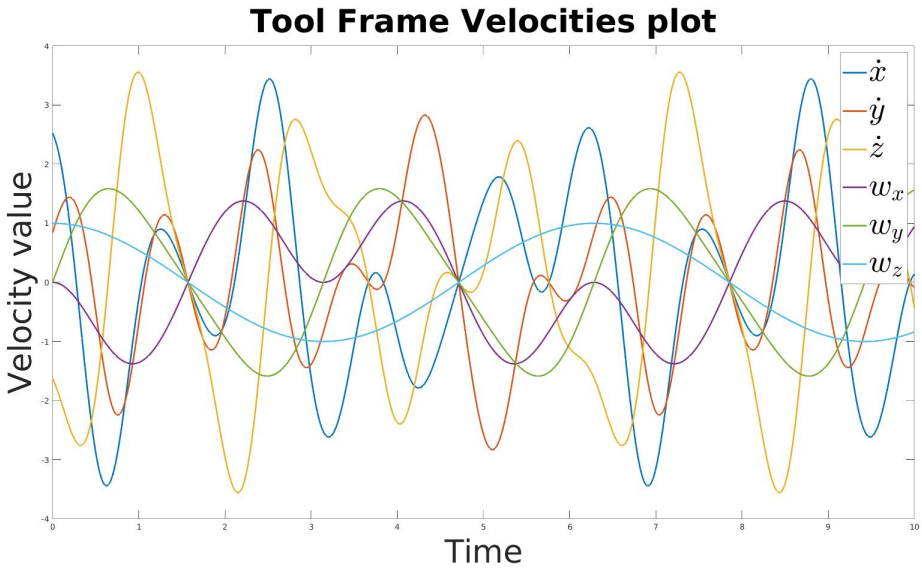

Figure 10: Tool Frame Velocities plot

Formulas for Tool Frame Velocities you can see below:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{y} \\ w_x \\ w_y \\ w_z \end{bmatrix} = \begin{pmatrix} 3\cos(3\,t)\cos(\sin(t))\sin(\cos(2\,t)) - 2\sin(2\,t)\cos(\sin(t))\cos(\cos(2\,t))(\sin(3\,t)+1) - \sin(\sin(t))\sin(\cos(2\,t))\cos(t)(\sin(3\,t)+1) \\ 3\cos(3\,t)\sin(\sin(t))\sin(\cos(2\,t)) - 2\sin(2\,t)\sin(\sin(t))\cos(\cos(2\,t))(\sin(3\,t)+1) + \cos(\sin(t))\sin(\cos(2\,t))\cos(t)(\sin(3\,t)+1) \\ -3\cos(3\,t)\cos(\cos(2\,t)) - 2\sin(2\,t)\sin(\cos(2\,t))(\sin(3\,t)+1) \\ -2\sin(2\,t)\sin(\sin(t)) \\ 2\sin(2\,t)\cos(\sin(t)) \\ \cos(t) \end{pmatrix} \tag{22}$$

Task 6:

Let tool coordinates changing with time as follows:
$p_x(t) = 2a_2 sin(t)$, $p_y(t) = 2a_2 cos(2t)$, $p_z(t) = d_1 sin(3t)$
Determine a feasible joint trajectory for this tool trajectory.

- Use IK solution.

- Use inverse differential kinematics approach. Consider only linear velocity part of Jacobian.

Solution

Using IK solution for joint trajectories

We know the tool coordinates changing. Also, I have solved the IK problem before in the section Task 2.

Hence, I made MATLAB code for IK solving and put tool coordinates changing in it to obtain joints' movements.

The results you can see below in figures 11-12.

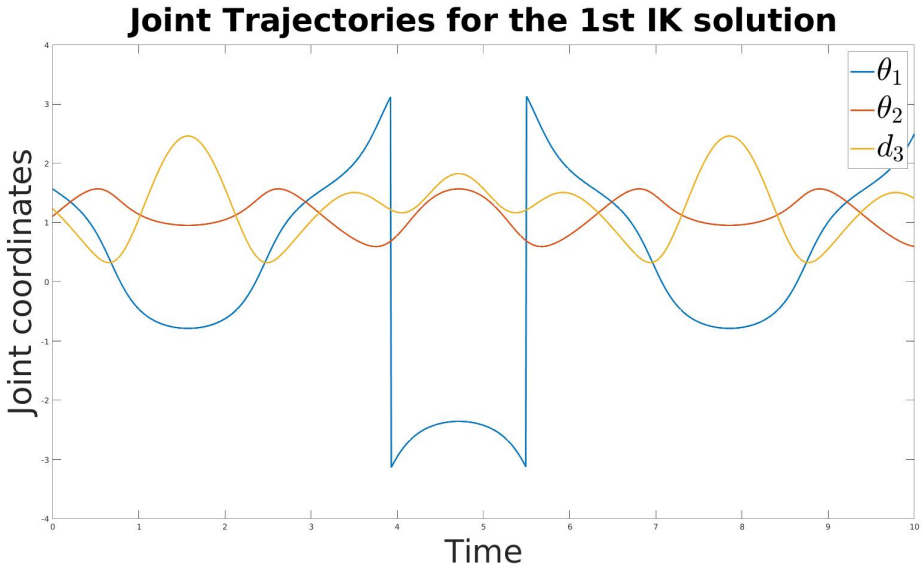

Figure 11: Joint Trajectories for the 1st IK solution

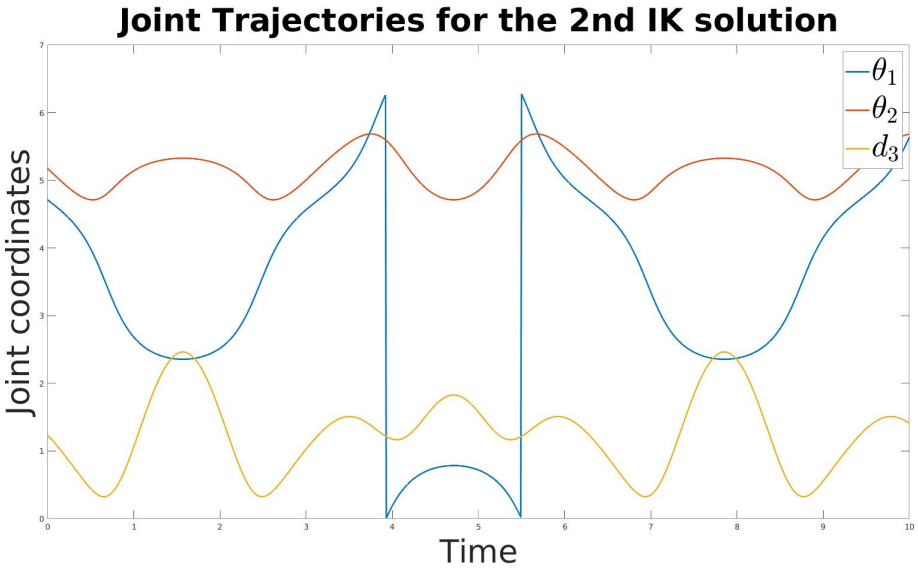

Figure 12: Joint Trajectories for the 2nd IK solution

The video of plotting the robot and it's desired (red) and actual (blue) paths (part of it) you can find here.

The whole code for HA you can find here.

Using inverse differential kinematics approach for joint trajectories

The idea is that we have 18 and we want to calculate $\dot{\boldsymbol{q}}$:

$$\dot{\boldsymbol{q}} = J^{-1} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{y} \\ w_x \\ w_y \\ w_z \end{bmatrix} \tag{23}$$

After that, we solve the system of differential equations to obtain $\boldsymbol{q}$.

Also, as usual, I made MATLAB code to solve this problem and plot the results.

The video of plotting the robot and it's desired (red) and actual (blue) paths (part of it) you can find here.

As I concluded, the both methods give pretty the same results and the error is quite miserable.