ECE 276C Assignment 1 Report

Mingwei Xu A53270271

Fall 2019

1 Question 1

1.1 Forward Kinematics

To get the forward model of this 2-DOF arm, we can use D-H parameters.
Given the 2-DOF arm, the D-H parameters are listed below:

Table 1: D-H Parameters

α_{i-1}	a_{i-1}	θ_i	d_i
0	0	q_0	0
0	$L_0 = 0.1$	q_1	0
0	$L_1 = 0.11$	0	0

Then we can calculate the D-H matrices:

$$ {}^{i}_{i-1}H = \begin{bmatrix} \cos\theta_i & \text{-} \sin\theta_i & 0 & \mathrm{a}_{i-1} \\ \sin\theta_i \cos\alpha_{i-1} & \cos\theta_i \cos\alpha_{i-1} & \text{-} \sin\alpha_{i-1} & \text{-} \sin\alpha_{i-1} d_i \\ \sin\theta_i \sin\alpha_{i-1} & \cos\theta_i \sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The forward model can be obtained by multiplying the D-H matrices:

$$ {}^{3}_{0}H = {}^{1}_{0}H {}^{2}_{1}H {}^{3}_{2}H $$

With the help of MATLAB, we get the foward model:

$$ {}^{3}_{0}H = \begin{pmatrix} \cos(q_0)\cos(q_1) - \sin(q_0)\sin(q_1) & -\cos(q_0)\sin(q_1) - \cos(q_1)\sin(q_0) & 0 & L_1\,(\cos(q_0)\cos(q_1) - \sin(q_0)\sin(q_1)) + L_0\cos(q_0) \\ \cos(q_0)\sin(q_1) + \cos(q_1)\sin(q_0) & \cos(q_0)\cos(q_1) - \sin(q_0)\sin(q_1) & 0 & L_1\,(\cos(q_0)\sin(q_1) + \cos(q_1)\sin(q_0)) + L_0\sin(q_0) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

1.2 Jacobian

Using the forward model, we can obtain:

$$ f_1(q) = L_1(\cos(q_0)\sin(q_1) + \cos(q_1)\sin(q_0)) + L_0\sin(q_0) $$

$$ f_2(q) = L_1(\cos(q_0)\sin(q_1) + \cos(q_1)\sin(q_0)) + L + 0\sin(q_0) $$

$$f_3(q) = q_1 + q_2$$

The Jacobian is calculated by:

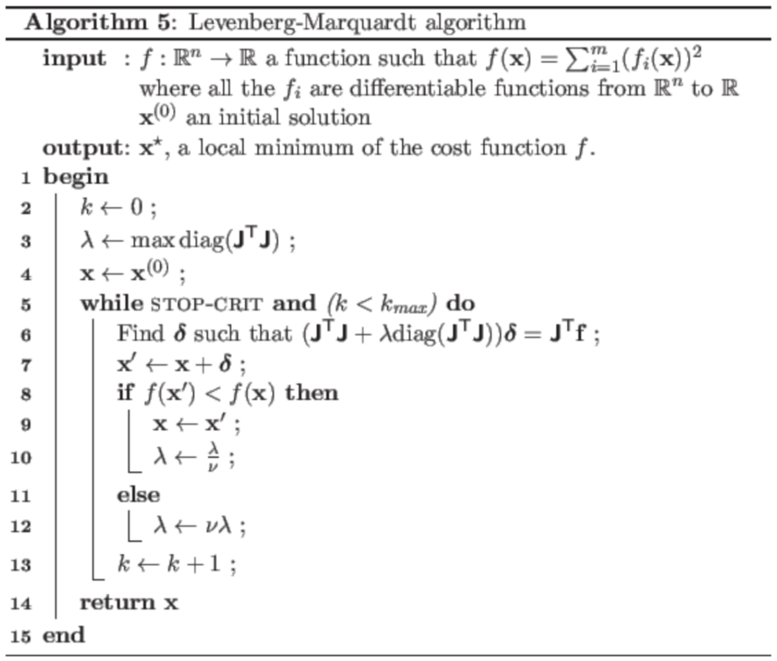
$$J(q) = \begin{bmatrix} \frac{\partial f_1(q)}{\partial q_1} & \frac{\partial f_1(q)}{\partial q_2} \\ \frac{\partial f_2(q)}{\partial q_1} & \frac{\partial f_2(q)}{\partial q_2} \\ \frac{\partial f_3(q)}{\partial q_1} & \frac{\partial f_3(q)}{\partial q_2} \end{bmatrix}$$

With the help of MATLAB, we get the Jacobian:

$$J(q) = \begin{pmatrix} -L_1\ (\cos(q_0)\ \sin(q_1) + \cos(q_1)\ \sin(q_0)) - L_0\ \sin(q_0) & -L_1\ (\cos(q_0)\ \sin(q_1) + \cos(q_1)\ \sin(q_0)) \\ L_1\ (\cos(q_0)\ \cos(q_1) - \sin(q_0)\ \sin(q_1)) + L_0\ \cos(q_0) & L_1\ (\cos(q_0)\ \cos(q_1) - \sin(q_0)\ \sin(q_1)) \\ 1 & 1 \end{pmatrix}$$

1.3 Inverse Kinematics

To get inverse kinematics of the robot, we use the Levenberg Marquardt method:



```
Algorithm 5: Levenberg-Marquardt algorithm
input  : $f : \mathbb{R}^n \to \mathbb{R}$ a function such that $f(\mathbf{x}) = \sum_{i=1}^{m} (f_i(\mathbf{x}))^2$
         where all the $f_i$ are differentiable functions from $\mathbb{R}^n$ to $\mathbb{R}$
         $\mathbf{x}^{(0)}$ an initial solution
output: $\mathbf{x}^\star$, a local minimum of the cost function $f$.
1  begin
2      $k \leftarrow 0$ ;
3      $\lambda \leftarrow \max \mathrm{diag}(\mathbf{J}^\mathsf{T}\mathbf{J})$ ;
4      $\mathbf{x} \leftarrow \mathbf{x}^{(0)}$ ;
5      while STOP-CRIT and $(k < k_{max})$ do
6          Find $\boldsymbol{\delta}$ such that $(\mathbf{J}^\mathsf{T}\mathbf{J} + \lambda \mathrm{diag}(\mathbf{J}^\mathsf{T}\mathbf{J}))\boldsymbol{\delta} = \mathbf{J}^\mathsf{T}\mathbf{f}$ ;
7          $\mathbf{x}' \leftarrow \mathbf{x} + \boldsymbol{\delta}$ ;
8          if $f(\mathbf{x}') < f(\mathbf{x})$ then
9              $\mathbf{x} \leftarrow \mathbf{x}'$ ;
10             $\lambda \leftarrow \frac{\lambda}{\nu}$ ;
11         else
12             $\lambda \leftarrow \nu\lambda$ ;
13         $k \leftarrow k + 1$ ;
14     return x
15 end
```

In this case, we define the objective function:

$$f = \frac{1}{2}[x_{err}, y_{err}]^T$$

The break condition inside the iterative loop is $||\mathbf{q_{k-1}} - \mathbf{q_k}||_2 <=$ threshold

Then, given the current end effector position $\mathbf{x} = [x, y]^T$ and inital joint angle guess $\mathbf{q_{guess}} = [q_{1guess}, q_{2guess}]^T$, we can obtain the joint angle using IK.

1.4 PD controller on end-effector position

In this problem, we use gain:

$$K_p = \begin{bmatrix} 200 & 0 \\ 0 & 200 \end{bmatrix}$$

$$K_d = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$$

The result is shown in "Fig. 1":

14 End Effector Position PD Control.png 14 End Effector Position PD Control.png

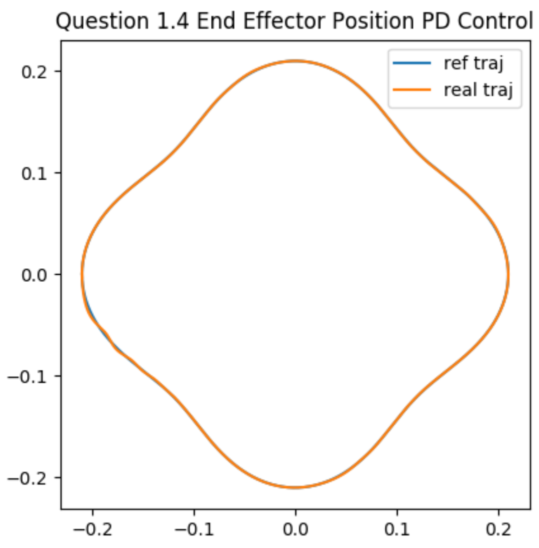


Figure 1: End Effector PD Control

From the result we can see that the end-effector position PD control works very well, as the reference trajectory and real trajectory are pretty much the same.

In fact, the MSE is only 1.1990525772621553e-05.

In addition, this controller is pretty robust even though we start the robot in random joint angle, as the figure shown below:

The result with a random initial joint angle is shown in "Fig. 2":

With MSE = 7.121480023182102e-05

1.5 PD controller on joint angle

In this problem, we use gain:

$$K_p = \begin{bmatrix} 4.2 & 0 \\ 0 & 2.4 \end{bmatrix}$$

$$K_d = \begin{bmatrix} 0.4 & 0 \\ 0 & 0.18 \end{bmatrix}$$

14 End Effector Position PD Control random.png 14 End Effector Position PD Control random.png

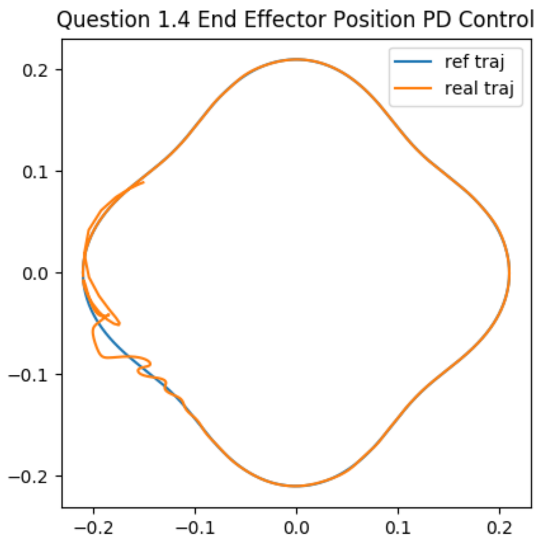


Figure 2: End Effector PD Control (random initial joint angle)

The result is shown in "Fig. 3"

From the figure we can see the result is slightly worse than end effector position PD controller. Note the little oscillation around the corners of the trajectory. Because the joint angle PD controller uses numerical method to find the inverse kinematics solution, sometimes the accuracy and stability is not very good.

By increasing the max iteration time and tightening the convergence condition of the inverse kinematics solver, we can potentially get better result. However the trade-off is longer computing time.

In this result, MSE = 0.00016963091653489742

Anyway, this controller is still robust enough to handle random initial joint angle, as shown in "Fig. 4":

With MSE = 0.0003240437820024025

2 Question 2

In this question, we use two closed-loop proportional controllers on wheel angle and thrust respectively.

2.1 Wheel angle controller

Since the wheel angle has closed-form solution from the vehicle kinematics model:

$$\tan(\gamma + \theta) = \frac{y_d - y}{x_d - x}$$

where γ is wheel angle, θ is vehicle orientation.

We can obtain the wheel angle control signal by:

$$\gamma = K_p[\operatorname{atan2}(\frac{y_d - y}{x_d - x}) - \theta]$$

15 Joint Angle PD Control.png 15 Joint Angle PD Control.png

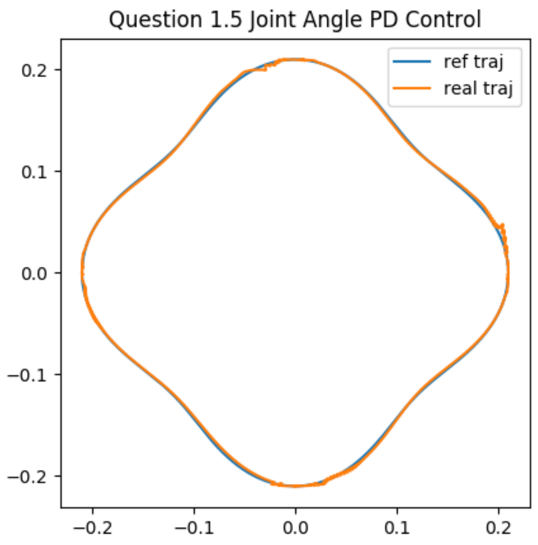


Figure 3: Joint Angle PD Control

15 Joint Angle PD Control random.png 15 Joint Angle PD Control random.png

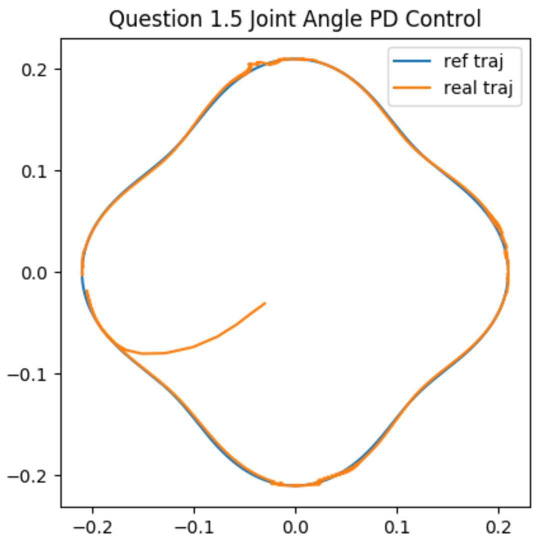


Figure 4: Joint Angle PD Control (random initial joint angle)

2.2 Thrust controller

When controlling the thrust, the control input is the position error in the body frame of vehicle, $\mathbf{x_{derr}}$. To obtain this, we first need to transfer the position error from the world frame to body frame.

A good thing to do this is that we can easily decouple lateral position error x_{derr} and frontal position error y_{derr} of the vehicle in body frame. When there is a large lateral position error, it means the vehicle needs to make a turn, hence we want to decrease speed so we will not slip doing high speed turn. When there is a large frontal position error, it means the vehicle needs to go straight, therefore we want to increase speed to save time.

The thrust controller is:

$$thrust = K_p(y_{derr}^2 - x_{derr}^2)$$

Also, we apply some speed limit to the thrust, so we will not go too fast.

2.3 Result

In “Fig. 5” and “Fig. 6” shows the result running on track ”Circle” and ”FigureEight”

2 Result Track Circle.png 2 Result Track Circle.png

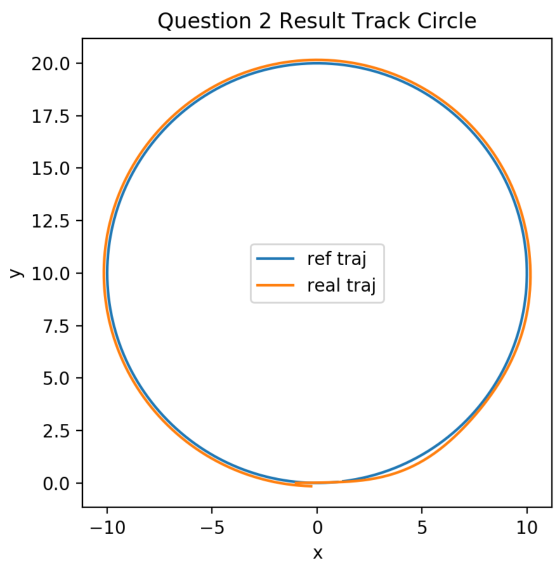


Figure 5: Result on Circle

The result shows that the controller converges. We can finish the track faster by tuning up some parameters like speed limit, however it will result in larger error.

2 Result Track FigureEight.png 2 Result Track FigureEight.png

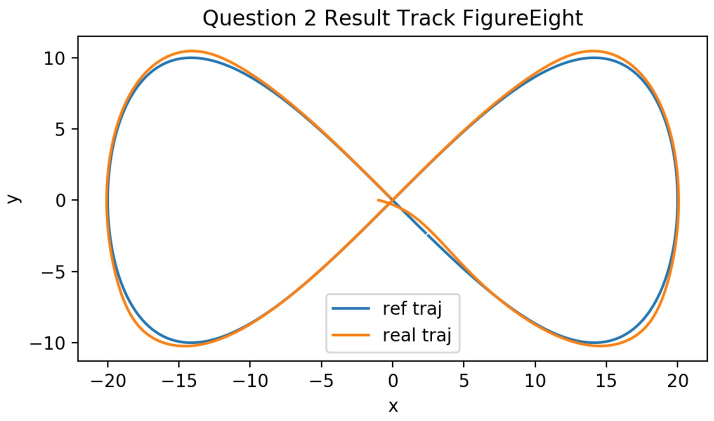


Figure 6: Result on FigureEight