Akhmetzhanov Ravil, RO-21

Dynamics of Non-Linear Robotics System

HA 3. Trajectory planning

This report accompanies code, that was made in a form of Jupyter Notebook (.ipynb).

Task 1: Calculating and plotting position, velocity, and acceleration trajectories of driving robot model from configuration q_0 to configuration q_f in joint space.

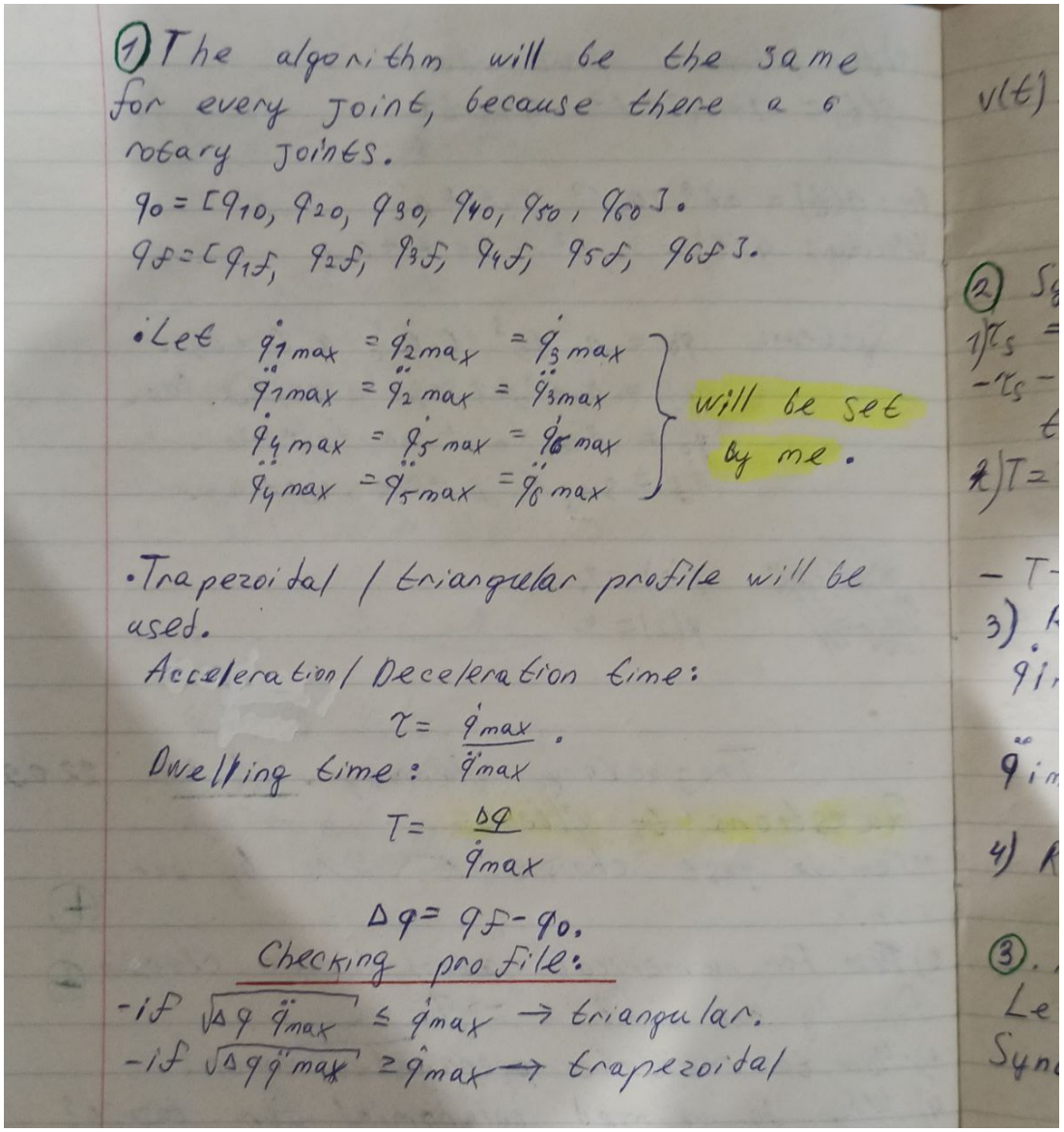

Figure 1. Solution for Task_1

We have 6 joints, and we need to synchronize them later. Written code can execute both trapezoidal and triangular velocity profiles.

Important notion: During synchronizing, if at least 1 joint will have a trapezoidal velocity profile, others should be converted to trapezoidal too.

$$q(t)=\begin{cases} q_0+\frac{1}{2}\ddot{q}_{max}t^2 & ,0<t\le t_b \\ q_0-\frac{1}{2}\ddot{q}_{max}t_b^2+\dot{q}_{max}\left(t-t_b\right) & ,t_b<t\le t_f-t_b \\ q_f-\frac{1}{2}\ddot{q}_{max}(t-t_f)^2 & ,t_f-t_b<t\le t_f \end{cases}$$

Figure 2. Trajectory for trapezoidal primitive

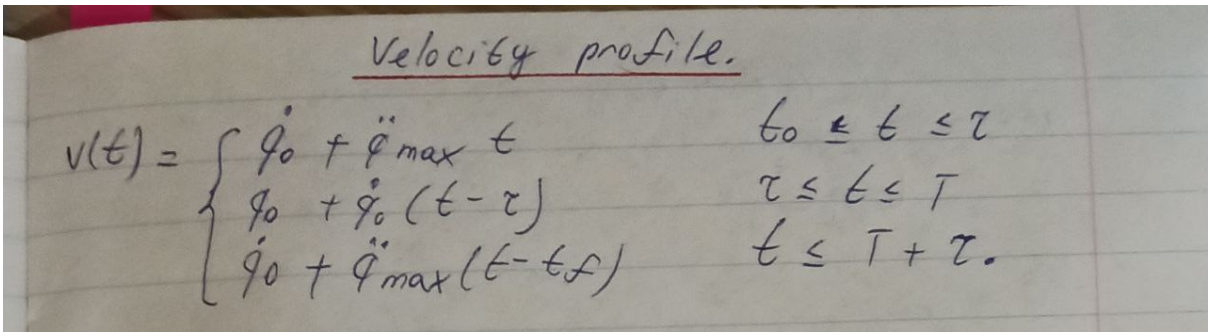

Figure 3. Velocity for trapezoidal primitive

In Figure 5 you can see unsynchronized movements of joints 1 -3. You can find the full result of task 1 (movements of joints 1 - 6) in notebook section 1.3 “Plotting initial trajectories”.

```
# q0, qf, v_max, a_max

j1 = [0, 90, 7, 4]
j2 = [0, 75, 7, 4]
j3 = [0, 60, 7, 4]
j4 = [0, 45, 3.5, 2]
j5 = [0, 30, 3.5, 2]
j6 = [0, 15, 3.5, 2]
```

Figure 4. Q - angles and parameters of the joints

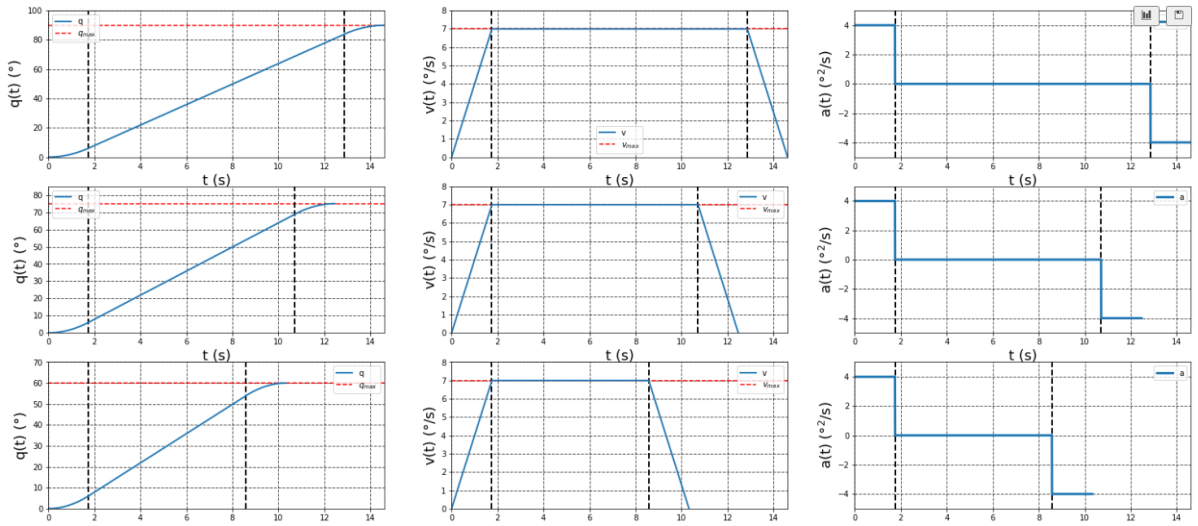

Figure 5. Unsynchronized movements of joints 1 -3

Task 2: Synchronizing 6 joints to start and end motion at the same time.

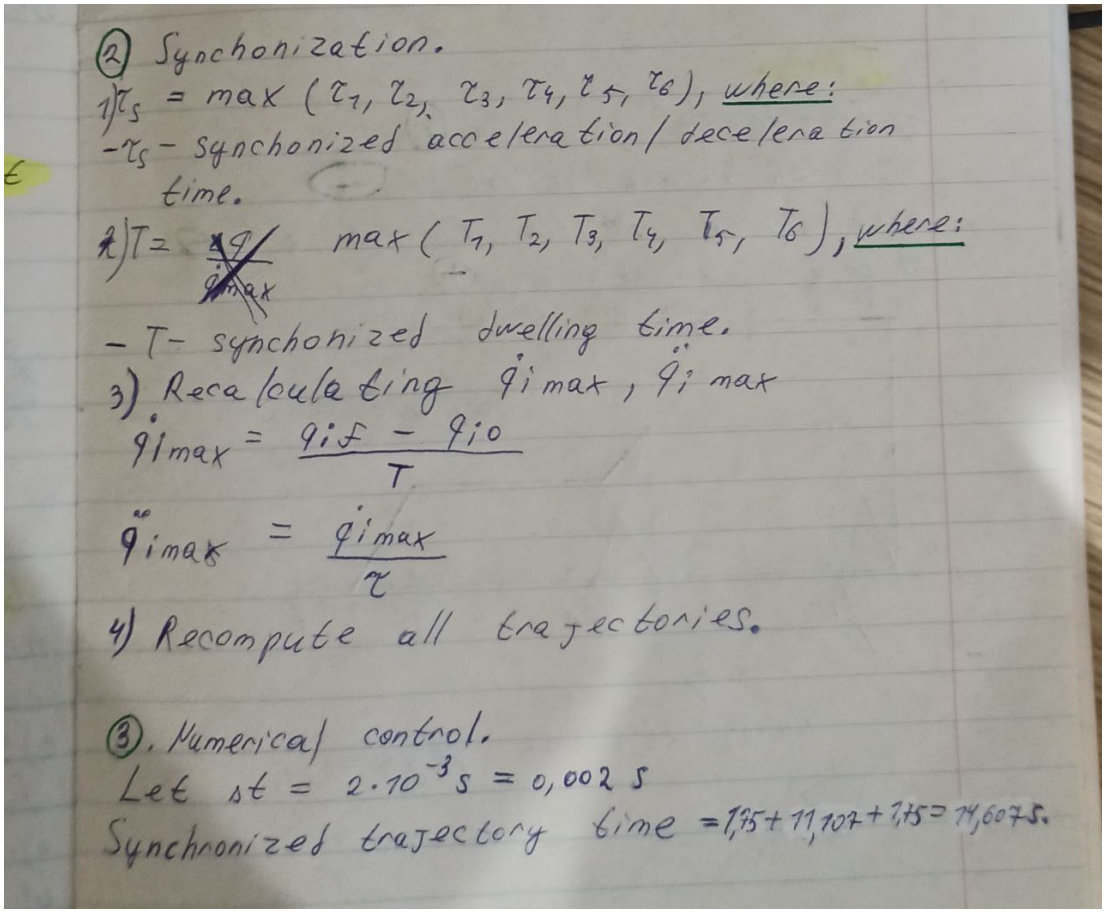

Figure 6. Steps for joint synchronization

```
J1 rise time: 1.75s, J2 rise time: 1.75s, J3 rise time: 1.75sJ4 rise time: 1.75s, J5 rise time: 1.75s, J6 rise time: 1.75ssynchronized rise time: 1.75

J1 dwell time: 11.107142857142858s, J2 dwell time: 8.964285714285714s, J3 dwell time: 6.821428571428571sJ4 dwell time: 11.107142857142858s, J5 dwell time: 6.821428571428571s, J6 dwell time:
2.5357142857142856ssynchronized dwell time: 11.107142857142858

Synchronized trajectory time = 1.75 + 11.107142857142858 + 1.75 = 14.607142857142858s

t_params: [0, 1.75, 12.857142857142858, 14.607142857142858]
Joint 1 velocity modified from 7 to 7.0 and acceleration from 4 to 4.0
Joint 2 velocity modified from 7 to 5.833333333333333 and acceleration from 4 to 3.3333333333333333
Joint 3 velocity modified from 7 to 4.666666666666666 and acceleration from 4 to 2.6666666666666665
Joint 4 velocity modified from 3.5 to 3.5 and acceleration from 2 to 2.0
Joint 5 velocity modified from 3.5 to 2.333333333333333 and acceleration from 2 to 1.3333333333333333
Joint 6 velocity modified from 3.5 to 1.1666666666666665 and acceleration from 2 to 0.6666666666666666
```

Figure 7. New time parameters after synchronization

In Figure 8 you can see unsynchronized movements of joints 1 -3. You can find the full result of task 1 (movements of joints 1 - 6) in notebook section 2.3 “Plotting each trajectory”.

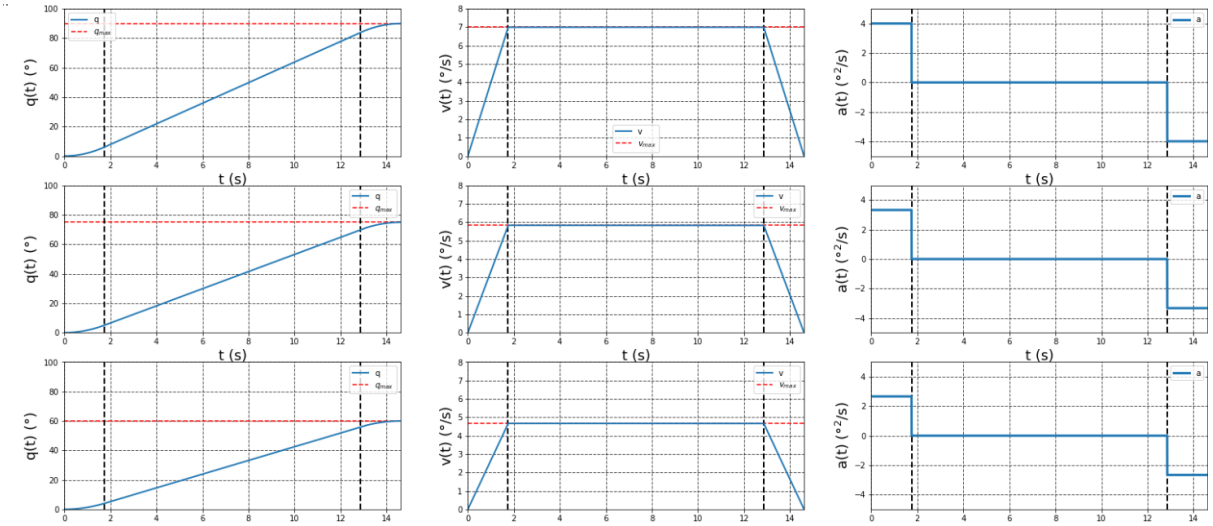

Figure 8. Synchronized movements of joints 1 -3

Task 3: Numerical control

Sample time of 2ms was chosen for performing numerical control.

Synchronizes time before numerical control: 1.75 + 11.107142857142858 + 1.75 = 14.607142857142858s

Synchronizes time after numerical control: 1.75 + 11.108 + 1.75 = 14.608s

```
delta_t = 0.002

if t1 % delta_t != 0:
    num_t1 = (t1 // delta_t) * delta_t + delta_t

if T % delta_t != 0:
    T = (T // delta_t) * delta_t + delta_t

if tf % delta_t != 0:
    tf = (tf // delta_t) * delta_t + delta_t

t_params = [t0, t1, T, tf]
print("Synchronized trajectory time = {} + {} + {} = {}s\n".format(t1, T - t1, t1, tf))
```

Figure 9. Code for numerical control

```
Synchronized trajectory time = 14.608 + -1.7479999999999993 + 14.608 = 14.610000000000001s

q_0, q_f, v_max, a_max
[0, 90, 6.998444790046656, 0.4790830223197327]
 [0, 75, 5.83203732503888, 0.3992358519331106]
 [0, 60, 4.665629860031104, 0.3193886815464885]
 [0, 45, 3.499222395023328, 0.23954151115986635]
 [0, 30, 2.332814930015552, 0.15969434077324424]
 [0, 15, 1.166407465007776, 0.07984717038662212]
```

Figure 10. New time parameters, angles, velocities, and accelerations after numerical control.

Task 4: Calculating propagated error

To calculate error we needed:

1) Recompute t1, T, tf, v_max, a_max after numerical control
2) Recompute angles
3) Solve FK for original angles
4) Solve FK for new angles
5) Get robot positions with original angles and new angles
6) Subtract position vectors to find error

There is no error in our set of joint time parameters and sample time (shown in figure 11). But in other cases an error might occur.

```
diff = pos_new - pos_orig
print(f'Propagated error after numerical control is: {diff}')
```

✓ 0.6s

```
[[ 0.79989459  0.16982733  0.57561039  0.17663304]
 [ 0.2803479   0.74230659 -0.60859344  0.02284126]
 [-0.53063519  0.64818177  0.54615629  0.54357944]
 [ 0.          0.          0.          1.        ]]

[[ 0.79989459  0.16982733  0.57561039  0.17663304]
 [ 0.2803479   0.74230659 -0.60859344  0.02284126]
 [-0.53063519  0.64818177  0.54615629  0.54357944]
 [ 0.          0.          0.          1.        ]]

Propagated error after numerical control is: [0. 0. 0.]
```

Figure 11. Results of calculating propagated error

Task 5: Trajectory between 3 consequent points using polynomial

For solving this task cubic and quintic polynomyal were used.
This task was done only for 1 joint.
We will drive our joint through these points:

q0 = 0 degree
q1 = 90 degrees.
q2 = 120 degrees.

In this time:

t0 = 0
t1 = tf = 14,608 s
t2 = 2 * t1

For cubic polynome we will use position and velocity constraints.
For quintic polynome we will use position, velocity and acceleration constraints. These constraints are shown in Figure 12:

```
t0 = 0
t1 = 14.608
t2 = 2*t1

q_0 = 0
q_1 = 90
q_2 = 120

v_0 = 0
v_1 = 2
v_2 = 0

a_0 = 0
a_1 = 1
a_2 = 0
```

Figure 12. Constraints for polynomial

- System of equations:

$$q_0 = a_3 t_0^3 + a_2 t_0^2 + a_1 t_0 + a_0$$
$$q_f = a_3 t_f^3 + a_2 t_f^2 + a_1 t_f + a_0$$
$$\dot{q}_0 = 3a_3 t_0^2 + 2a_2 t_0 + a_1$$
$$\dot{q}_f = 3a_3 t_f^2 + 2a_2 t_f + a_1$$

$$\begin{bmatrix} t_0^3 & t_0^2 & t_0 & 1 \\ t_f^3 & t_f^2 & t_f & 1 \\ 3t_0^2 & 2t_0 & 1 & 0 \\ 3t_f^2 & 2t_f & 1 & 0 \end{bmatrix} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} q_0 \\ q_f \\ \dot{q}_0 \\ \dot{q}_f \end{bmatrix}$$

Figure 13. Equations for cubic polynomial

- System of equations:

$$q_0 = a_5 t_0^5 + a_4 t_0^4 + a_3 t_0^3 + a_2 t_0^2 + a_1 t_0 + a_0$$
$$q_f = a_5 t_f^5 + a_4 t_f^4 + a_3 t_f^3 + a_2 t_f^2 + a_1 t_f + a_0$$
$$\dot{q}_0 = 5a_5 t_0^4 + 4a_4 t_0^3 + 3a_3 t_0^2 + 2a_2 t_0 + a_1$$
$$\dot{q}_f = 5a_5 t_f^4 + 4a_4 t_f^3 + 3a_3 t_f^2 + 2a_2 t_f + a_1$$
$$\ddot{q}_0 = 20a_5 t_0^3 + 12a_4 t_0^2 + 6a_3 t_0 + 2a_2$$
$$\ddot{q}_f = 20a_5 t_f^3 + 12a_4 t_f^2 + 6a_3 t_f + 2a_2$$

$$\begin{bmatrix} t_0^5 & t_0^4 & t_0^3 & t_0^2 & t_0 & 1 \\ t_f^5 & t_f^4 & t_f^3 & t_f^2 & t_f & 1 \\ 5t_0^4 & 4t_0^3 & 3t_0^2 & 2t_0 & 1 & 0 \\ 5t_f^4 & 4t_f^3 & 3t_f^2 & 2t_f & 1 & 0 \\ 20t_0^3 & 12t_0^2 & 6t_0 & 2 & 0 & 0 \\ 20t_f^3 & 12t_f^2 & 6t_f & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} q_0 \\ q_f \\ \dot{q}_0 \\ \dot{q}_f \\ \ddot{q}_0 \\ \ddot{q}_f \end{bmatrix}$$

Figure 14. Equations for quintic polynomial

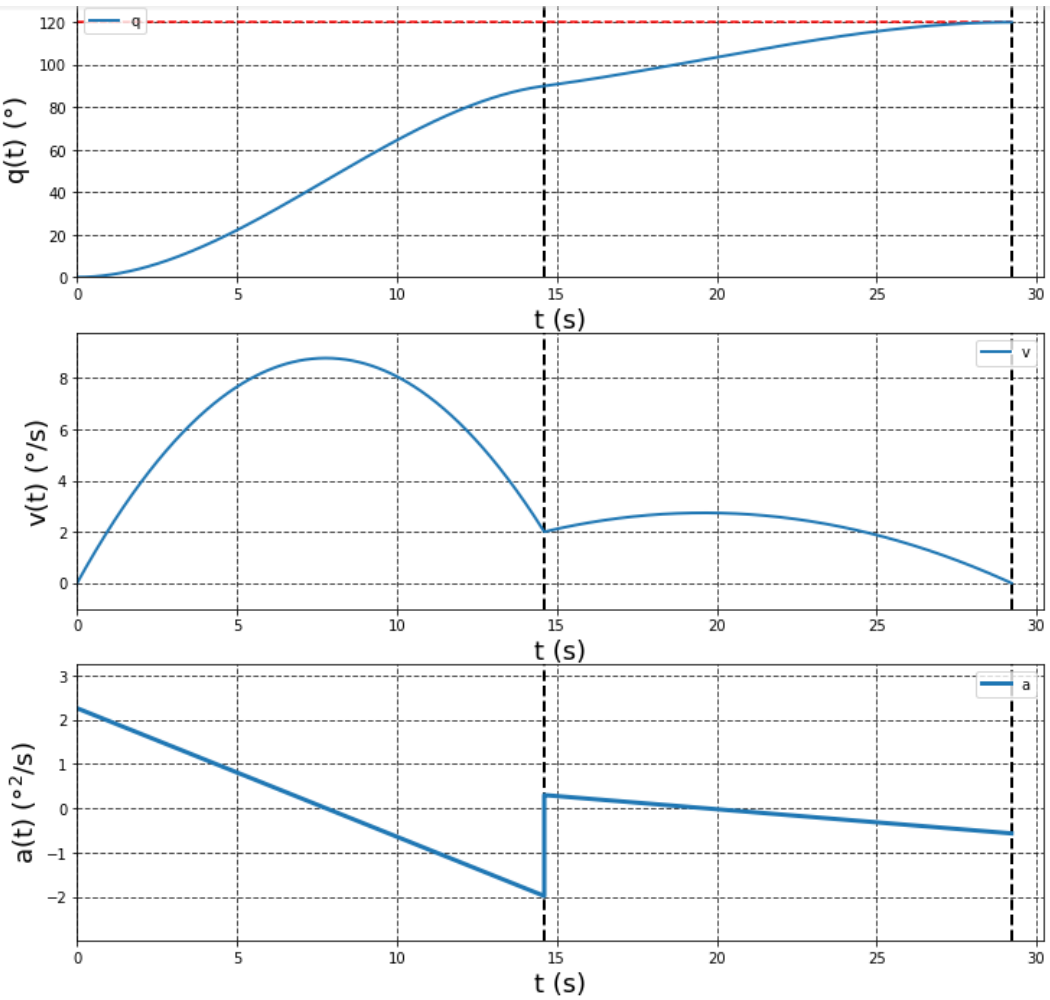

Figure 15. Trajectory of a joint using cubic polynome aproximation

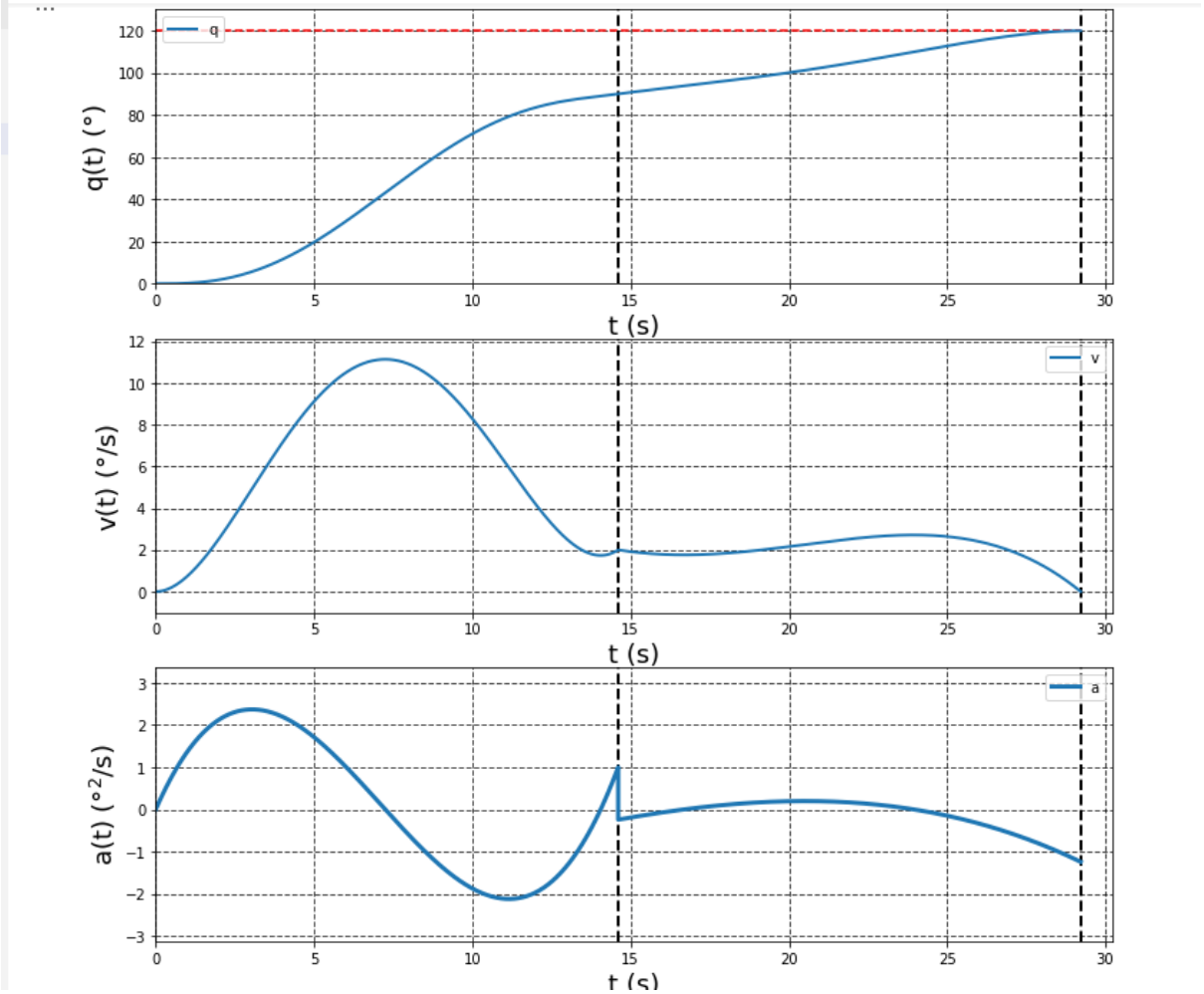

Figure 16. Trajectory of a joint using quintic polynome aproximation

Task 6: Trajectory junction

Let's consider junction of 2 trapezoidal profiles. To do it we need to perform it like it's shown in Figure 17.

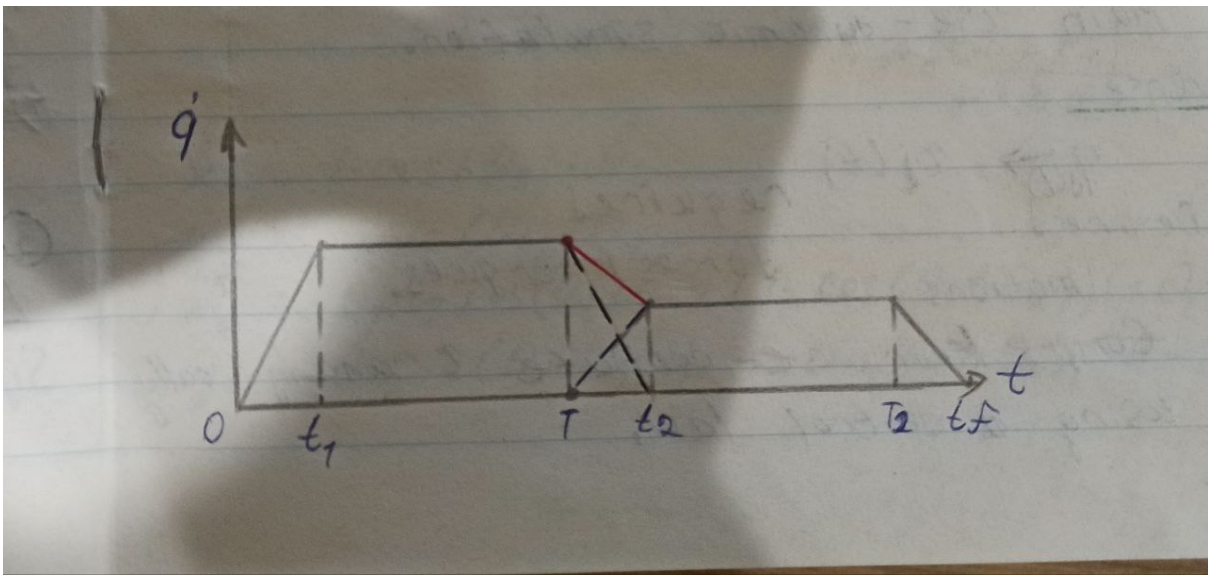

Figure 17. Trajectory junction

To do this task we need to start motion from 2nd to 3rd point earlier, at T. And we need to calculate junction, that is shown as a red line.

Code implementation is not ready yet.