5. Tasks - Laboratory 1

Implemented model in `models/manipulator_model.py` file and read data from urdf file

```
def M(self, x):
    q1, q2, q1_dot, q2_dot = x
    #   Without the object at the tip
    alpha   = self.m1*self.d1**2+self.I_1+self.m2*
(self.l1**2+self.d2**2)+self.I_2
    beta    = self.m2*self.l1*self.d2
    gamma   = self.m2*self.d2**2 + self.I_2
    #   With the object at the tip
    alpha = self.I_1 + self.I_2 + self.m1 * self.d1 ** 2 + self.m2 *
(self.l1 ** 2 + self.d2 ** 2) + \
            self.I_3 + self.m3 * (self.l1 ** 2 + self.l2 ** 2)
    beta = self.m2 * self.l1 * self.d2 + self.m3 * self.l1 * self.l2
    gamma = self.I_2 + self.m2 * self.d2 ** 2 + self.I_3 + self.m3 *
self.l2 ** 2
    m11 = alpha + 2*beta*np.cos(q2)
    m12 = gamma + beta*np.cos(q2)
    m21 = gamma + beta*np.cos(q2)
    m22 = gamma
    return np.array([[m11, m12],[m22, m21]])
def C(self, x):
    q1, q2, q1_dot, q2_dot = x
    #   Without the object at the tip
    beta    = self.m2*self.l1*self.d2
    #   With the object at the tip
    beta = self.m2 * self.l1 * self.d2 + self.m3 * self.l1 * self.l2
    c11 = -beta*np.sin(q2)*q2_dot
    c12 = -beta*np.sin(q2)*(q1_dot+q2_dot)
    c21 = beta*np.sin(q1)*q1_dot
    c22 = 0
    return np.array([[c11, c12],[c21, c22]])
```

PROF

Feedback Linearization Controller:

```
def calculate_control(self, x, q_r, q_r_dot, q_r_ddot):
    #--------------------1------------------------
    v = q_r_ddot
    #--------------------2------------------------
    v = self.model.M(x) @ q_r_ddot + self.model.C(x) @ q_r_dot
    #--------------------8------------------------
    q1, q2, q1_dot, q2_dot = x
    q = np.array([q1, q2])
    q_dot = np.array([q1_dot, q2_dot])
    v = v + self.Kd*(q_dot - q_r_dot) + self.Kp*(q - q_r)
    return v
```

Polynomial Generator:

```
class Poly3(TrajectoryGenerator):
    def __init__(self, start_q, desired_q, T):
        self.a_0 = self.q_0
        self.a_1 = - 3 * self.q_0 + start_q
        self.a_2 = 3 * self.q_k  - desired_q
        self.a_3 = self.q_k

    def generate(self, t):
        #--------------------6-------------------------
        t /= self.T
        q = self.a_3 * t**3 + self.a_2 * t**2 * (1 - t) + self.a_1 * t *
(1 - t)**2 + self.a_0 * (1 - t)**3
        q_dot = 3 * self.a_3 * t**2 + self.a_2 * (2 * t - 3 * t**2) + \
                self.a_1 * (3 * t**2 - 4 * t + 1) + 3 * self.a_0 * (1 -
t)**2
        q_ddot = 6 * self.a_3 * t + self.a_2 * (2 - 6 * t) + self.a_1 *
(6 * t - 4) + 6 * self.a_0 * (1 - t)
        return q, q_dot / self.T, q_ddot / self.T**2
```