ME 221, Fall 2021 HW 4
University of California, Riverside Due: 11/22/2021
Department of Mechanical Engineering

1. Consider the (double pendulum) planar robot in Fig. 1, where $\mathcal{Q}_i$ represent the torque at the i^{th} joint.

   a. Find the equations of motion using Lagrangian mechanics.

   b. Specify symbolic expressions for the following:

      i. The mass matrix $M(\boldsymbol{q})$.

      ii. The Coriolis matrix $C(\boldsymbol{q}, \dot{\boldsymbol{q}})$.

      iii. The gravity vector $g(\boldsymbol{q})$.

   c. Simulate the equations of motion with $l_1 = l_2 = m_1 = m_2 = 1$ and $\mathcal{Q}_1 = \mathcal{Q}_2 = 0$ using `ode45` for $T = 10$ seconds with a time step $dt = 1/1000$ with the following initial conditions. Make an animation (`hw4-1.mp4`)(you can skip some frames so it's not so slow, e.g., use `for i=1:10:numel(t)`) of one of the simulations and make a plot with the x and y coordinate of m_2 for all time comparing the trajectories of both sets of initial conditions.

      i. IC_1: $\theta_1(0) = \frac{3\pi}{4}$, $\theta_2(0) = 0$, $\dot{\theta}_1(0) = 0$, $\dot{\theta}_2(0) = 0$.

      ii. IC_2: $\theta_1(0) = \frac{2.99\pi}{4}$, $\theta_2(0) = 0$, $\dot{\theta}_1(0) = 0$, $\dot{\theta}_2(0) = 0$.

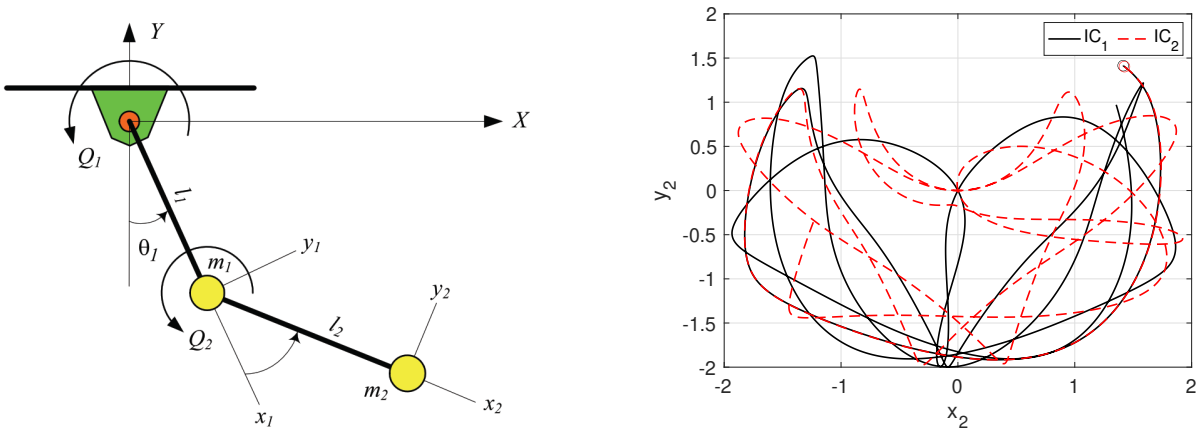

Figure 1: Schematic and results plot for Problem 1.

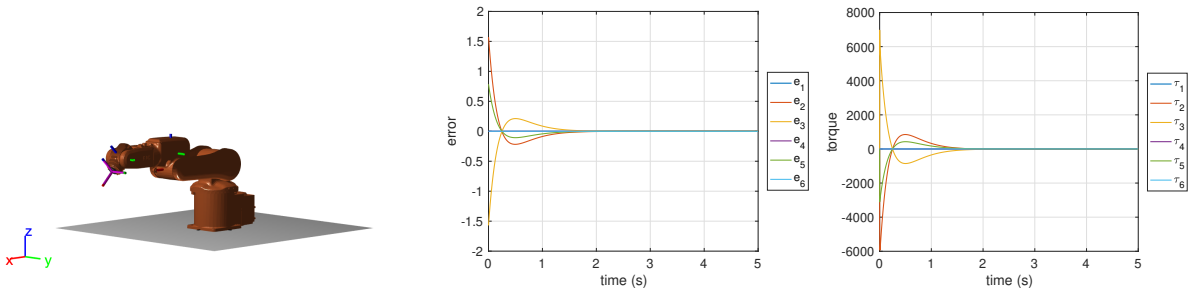

Figure 2: Results plot for Problem 2 d..

2. In this exercise you will explore simple feedback controllers with the MATLAB simulation we have been developing. Make sure you can render the IRB 120 robot in the virtual environment. The goal of this problem is to develop a PID (proportional-integral-derivative) controller to hold desired postures (e.g., fixed joint positions) under external perturbations.

   a. The equations of motion of the robot can be described as:

   $$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + g(\boldsymbol{q}) = \boldsymbol{\tau} - J(\boldsymbol{q})^T \boldsymbol{F}_{ext},$$

   which can be rearranged as:

   $$\ddot{\boldsymbol{q}} = M(\boldsymbol{q})^{-1} \left(\boldsymbol{\tau} - J(\boldsymbol{q})^T \boldsymbol{F}_{ext} - C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - g(\boldsymbol{q}) \right) \tag{1}$$

   Use a new state variable

   $$\bar{\boldsymbol{q}} = \begin{bmatrix} q_1 & \cdots & q_n & \dot{q}_1 & \cdots & \dot{q}_n \end{bmatrix}^\intercal$$

   to write Eq. 1 as a system of first order differential equations, e.g., of the form:

   $$\dot{\bar{\boldsymbol{q}}} = f(\bar{\boldsymbol{q}}, \boldsymbol{\tau}, \boldsymbol{F}_{ext})$$

   b. The Matlab function `forwardDynamics` directly computes Eq. 1. Using this function, create a new (anonymous) function (e.g., `dynamics =@(t,q) ...`) that computes the system of first order ODE's developed in the Part a. Use `ode23tb` to simulate ($dt = 1/1000$, $T = 5$, $\boldsymbol{q}_0 = 0$) the robot without any external forces or joint torques ($\boldsymbol{\tau} = 0$, and $\boldsymbol{F}_{ext} = 0$). Create and submit an animation (`hw4-2b.mp4`) of the simulation.

   c. Create a new (anonymous) function to implement a PID controller, e.g.,

   $$\boldsymbol{\tau} = k_p(\boldsymbol{q}_d - \boldsymbol{q}) + k_i \int_0^t (\boldsymbol{q}_d - \boldsymbol{q})dt + k_d(-\dot{\boldsymbol{q}}) \tag{2}$$

   The (anonymous) function should be of the form `tau =@(t,q,qd,kp,kd,ki)`. Note that Eq.2 can be rewritten as

   $$\boldsymbol{\tau} = k_p(\boldsymbol{q}_d - \boldsymbol{q}) + k_i\bar{\boldsymbol{e}} + k_d(-\dot{\boldsymbol{q}}), \quad \text{with } \bar{\boldsymbol{e}} = \int_0^t (\boldsymbol{q}_d - \boldsymbol{q})dt$$

   Therefore

   $$\dot{\bar{\boldsymbol{e}}} = \boldsymbol{q}_d - \boldsymbol{q}$$

   To properly simulate the PID controller, augment your previous function `dynamics` to include the error state variable $\bar{\boldsymbol{e}}$. Your new system of ODE's should be of the form:

   $$\dot{\bar{\boldsymbol{z}}} = f(\bar{\boldsymbol{z}}, \boldsymbol{\tau}, \boldsymbol{F}_{ext}), \quad \text{with } \bar{\boldsymbol{z}} = \begin{bmatrix} \bar{\boldsymbol{q}} & \bar{\boldsymbol{e}} \end{bmatrix}^\intercal$$

   Tune your controller gains (k_p, k_i, k_d) so that the robot can remain fixed in the home position (e.g., $\boldsymbol{q}_d = \boldsymbol{q}_0 = 0$). Test your controller's ability to move to a new desired posture

   $$\boldsymbol{q}_d = \begin{bmatrix} 0 & \pi/2 & -\pi/2 & 0 & \pi/4 & 0 \end{bmatrix}^\intercal.$$

   Retune your gains until the robot reduces the error to near zero within 2 seconds. Make and submit an animation (`hw4-2c.mp4`) of the robot moving to the new posture. On separate plots, show the error and torque of each joint during the simulation as in Fig. 2.

   d. Create an external wrench on the tool frame of the form:

   $$\boldsymbol{F}_{ext}(t) = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \\ F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -Au(t-a) \\ 0 \\ Au(t-a) \end{bmatrix}$$

   with $A = 1000$, $a = 2$, and u the unit step function. The Matlab function `externalForce` will create the external wrench. Simulate the robot with the external perturbation and submit an animation (`hw4-2d.mp4`) and plots of the errors and torques for each joint.