Modeling

Last update: 2021-04-21

Gen3 lite Cursus

Table of content

KINOVA

Objectives

During this lab, our objective will be to build a dynamic model of the Gen3 lite manipulator. We will do so step by step, starting with a characterization of the electrical motors, then moving on to a static model and finally the full dynamic model. Then, we will use MATLAB to send trajectories to the robot to validate the output of our model.

About Gen3 lite

During the lab, we will need to know the gear ratio of the gearboxes that are in Gen3 lite's actuators. These values are summarized in Table 1.

Joint #	Gear ratio
1	60
2	90
3	60
4	30
5	30
6	30

TABLE 1: Gear ratio of the gearbox located in each of Gen3 lite's joint

We will also need the inertial information of each link. These are available in the User Guide on pages 127 and following.

Reminder: Gen3 lite's API assumes that all lengths are in meters (m), angles are in degrees (°) and torques are in Newton-meter (Nm).

Take note that Gen3 lite does not have torque sensors. All torque readings returned by the robot are estimated based on the motor current using the model described in the next section.

Theory

Current-based torque estimation

Electrical motors are often modeled with a linear equation.

$$\tau = K_t I - F$$

Where

- τ is the output torque
- I is the motor current
- K_t is a constant obtained experimentally (conveniently called motor constant)
- F is a term for external loss

In robot actuators, the raw torque output from electrical motors is often too low, while the rotation speed on the other hand is too high. To fix this problem, actuators generally include gearboxes with high gear ratios (R) which mechanically convert speed into torque. In addition to this, actuator designs usually incorporate features such as grease and ball bearings to reduce the external losses as much as possible. Therefore, for a full actuator, assuming an ideal gearbox and perfectly frictionless motion, we obtain the following model:

$$\tau = RK_t I$$

This is the model used by Gen3 lite to return torque estimation with its feedback.

Dynamic Modeling

The trajectory followed by a robot applying torque to it's joints is given by:

$$\tau = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q)$$

Where

- τ is a vector containing the torque applied on each joint
- q is a vector containing the position of each joint
- $M(q)$ is a matrix representing the inertial terms (like m in F=ma)
- $C(q, \dot{q})$ is a matrix that takes into account Coriolis-type effects
- $g(q)$ is a vector containing the torque required to counter the effects of gravity

We can see that when the robot is not moving, we obtain $\tau = g(q)$, which is a static model.

Manipulations

Part 1: Characterization of the electrical motor

In this section, we will take measurements with the robots to determine the motor constant in its actuators. You can take your measurements directly from the Web App, or modify and use one of the scripts we made in the previous lab sessions. If you need a refresher on how to use the MATLAB API, the documentation is here.

1.1 Take a few synchronized measures of torques and motor current to determine the motor constant (K_t) of the motor in actuators 2, 3 and 5. Explain your method.

Part 2: Implementation of the dynamic model

In this section, we will implement in MATLAB a model capable of evaluating the torques required to compensate for the force applied by gravity on the robot. Then, we will complete our model with the inertia and Coriolis matrices which we will use to evaluate the torque necessary to follow a trajectory.

2.1 Write a function called **gravity**(q) which takes as an input a joint configuration (**q**) and outputs the torques necessary for the robot to support its own weight.

2.2 Write a function called **inertiaMatrix(**q) which takes as an input a joint configuration and outputs the inertia matrix of the robot.

2.3 Write a function called **coriolis**(q, dq) that takes as inputs a joint configuration and a vector of joint velocities (**dq**) to return the corresponding Coriolis matrix.

2.4 Write a function called **precomputed_torque**(q, dq, ddq) which takes as inputs a trajectory defined by joint positions, velocities and accelerations and returns a *n x m* matrix, where *n* is the number of joints (here 6) and *m* is the number of points in the trajectory.

Part 3: Validation with the robot

In this section, we will run trajectories on the robot and record the estimated applied torque returned in the robot feedback to compare it to our algorithms.

3.1 Write a script using the API function **PlayPreComputedTrajectory**() to execute the trajectory written in the file *trajectory.csv*. You can easily load *.csv* files in MATLAB using **csvread**(). Then, use RefreshFeedback() to obtain the torque applied by all the actuators during the trajectory. Plot the curves obtained for every actuator. **Note that the .csv file is structure so that each line represents a time step of 1ms, the first column contains time, the following 6 contain the position of each joint, then the 6 velocities and finally 6 acceleration.** You may also use a trajectory you generated during our lab on *Trajectory Generation*.

3.2 Run your **precomputed_torque**() function on the same trajectory you tested in **3.1**. Plot the curves you obtain and compare them to the experimental data.

3.3 Name two plausible reasons why your experimental and theoretical results are not identical.

Acknowledgements

This document was produced in collaboration with the following establishments

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Including direct implication from the following people:

- Prof. David Saussié, Eng., M.A.Sc., Ph.D., Department of Electrical Engineering.
- Alexis Tang, M.Eng., Department of Electrical Engineering
- Prof. Jérôme Le Ny, Eng., M.A.Sc., Ph.D., Department of Electrical Engineering.
- Prof. Richard Gourdeau , B.A.Sc., M.A.Sc., Ph.D., Department of Electrical Engineering.
