
(Spring 2007-08)

Lab Assignment #3

Due Lab #3

In this assignment you will improve your joint motion controller by augmenting your servo loop so that your robot can move to a position while limiting the velocity or following a trajectory. This will allow you to move the robot smoothly without saturating the motors and incurring large position errors. See **Handout #08** and **Chapter 6 in Introduction to Robotics** for references.

Velocity Saturation

1. Implement the user command “**njgoto** $\theta_1 \theta_2 \theta_3 \theta_4 \theta_5 \theta_6$ ”. Use non-dynamic control but include gravity compensation. When invoked the robot should move to the given goal position using a velocity saturating linearized controller. The controller should satisfy the following constraints: $|\dot{\theta}_i| \leq \dot{q}_{max_i}$. Make sure you do not divide by zero when $k_v = 0$.
2. Add safety checking to prevent robot from hitting joint limits. Do not allow a goal position outside of the joint range.
3. Plot θ_3, θ_{3d} vs *time* and $\dot{\theta}_3, \dot{\theta}_{3d}$ vs *time* from position $(0, 0, 0, 0, 0, 0)$ when the command “**njgoto 0 0 90 0 0 0**” is called. Find the error between $\dot{\theta}_3$ and $\dot{\theta}_{3d}$ while in saturation. Discuss your results.
4. Repeat using dynamic control. Implement the user command “**jgoto** $\theta_1 \theta_2 \theta_3 \theta_4 \theta_5 \theta_6$ ”. Compare your results with non-dynamic control.

Cubic Spline Trajectory

1. Implement the user command “**njtrack** $\theta_1 \theta_2 \theta_3 \theta_4 \theta_5 \theta_6$ ”. When invoked this routine should find a cubic spline trajectory in joint space to get from the current position to the given goal position. Use non-dynamic control but include gravity compensation. Remember to handle termination conditions appropriately. The trajectory should satisfy the following constraints: $|\dot{\theta}_i| \leq \dot{q}_{max_i}$ and $|\ddot{\theta}_i| \leq \ddot{q}_{max_i}$.
2. Add safety checking to prevent robot from hitting joint limits. Do not allow a goal position outside of the joint range.
3. Plot θ_3, θ_{3d} vs *time* and $\dot{\theta}_3, \dot{\theta}_{3d}$ vs *time* from position $(0, 0, 0, 0, 0, 0)$ when the command “**njtrack 0 0 90 0 0 0**” is called. Find the error between $\dot{\theta}_3$ and $\dot{\theta}_{3d}$. Make sure your position is cubic, velocity is quadratic, and acceleration is linear.
4. Repeat using dynamic control. Implement the user command “**jtrack** $\theta_1 \theta_2 \theta_3 \theta_4 \theta_5 \theta_6$ ”. Compare with non-dynamics.

Inverse Kinematics

1. Write the function “**bool inv_kin(const PrVector3& pos, const PrMatrix3& rot, int elbow, PrVector& q)**” that computes the joint positions **q** given the Cartesian position **pos** and orientation given by the rotation matrix **rot**. “**inv_kin**” returns false if the given position is outside the robots workspace.

In general, there are 8 ways for the PUMA to reach a position within its workspace. Use the **elbow** argument to determine which one to use:

- If (*elbow*&0x01) is set, rotate joint 1 by $\sim 180^\circ$.
- If (*elbow*&0x02) is set, rotate joint 3 so the elbow is up.
- If (*elbow*&0x04) is set, rotate joint 4 by 180° .

Use the **L1**, **L2**, **L3**, and **L6** measurements in param.h. The easiest approach is to use **L6** and **rot** to find the coordinates of the wrist, and then find the joints in the following order: $\theta_1, \theta_2/\theta_3, \theta_5, \theta_4/\theta_6$.

2. Use “**inv_kin**” to implement the user command “**nxtrack x y z ...**” to move the robot to a given Cartesian position and orientation. Note that “**nxtrack**” needs to call “**inv_kin**” only once to find the joint space goal position. The inverse kinematics will not be computed for the entire trajectory. You can derive **pos** and **rot** from **gv.Td.translation()** and **gv.Td.rotation().matrix()**, respectively. Use non-dynamic control but include gravity compensation.
3. Implement the user command “**xtrack x y z ...**” to move the robot to a given Cartesian position and orientation using dynamic control.
4. Change the robot model from **6-DOF (quaternions)** to **6-DOF (euler angle y)**. Collect data from position (.6, .1, .3, 0, 0, 0, 0) to position (.75, .2, .5, 0, 90, 0, 2). Plot the end-effector trajectory (x,z,theta) and velocity ($\dot{x}, \dot{z}, \dot{\omega}_y$). Compare “**nxtrack**” and “**xtrack**”. Discuss your results.

Make sure you put enough comments so that we can understand what you’re doing!

Submit a short report describing what you learned in lab. Discuss special features in your code. Incorporate material from lectures.