ME 446 Laboratory #3
Inverse Dynamics Joint Control

Weeks 3/30/2020, 4/6/2020

Report Due Friday April 17th at 5pm.

Task1:

Implement the Inverse Dynamics Control Law on Joints One, Two and Three.

Inverse Dynamics

This document does not go into detail on Inverse Dynamics control. Please see Section 8.2 of Robot Modeling and Control or section 11.3 of "Robot Dynamics and Control" second edition.

In this task, you need to modify your feed forward control from lab 2 to implement Inverse Dynamics control for joint 1, joint 2 and joint 3 to track the cubic polynomial in Lab2.

So for joints one, two and three implement the inverse dynamics control

$$\tau = D(\theta)a_\theta + C(\theta,\dot{\theta})\dot{\theta} + g(\theta)$$

to track the cubic polynomial reference trajectory from part 4 of lab 2. This control equation is called the inner loop because it cancels the nonlinearities leaving the equation $a_\theta = \ddot{\theta}$. (This of course assumes we know the systems perfectly). Then an outer loop control is needed to control this linear set of equations. For the outer loop control use PD plus feed forward control and the same cubic trajectory from lab 2

$$a_{\theta_1} = \ddot{\theta}_1^d + K_{P1} * \left(\theta_1^d - \theta_1\right) + K_{D1} * \left(\dot{\theta}_1^d - \dot{\theta}_1\right)$$

$$a_{\theta_2} = \ddot{\theta}_2^d + K_{P2} * \left(\theta_2^d - \theta_2\right) + K_{D2} * \left(\dot{\theta}_2^d - \dot{\theta}_2\right)$$

$$a_{\theta_3} = \ddot{\theta}_3^d + K_{P3} * \left(\theta_3^d - \theta_3\right) + K_{D3} * (\dot{\theta}_3^d - \dot{\theta}_3).$$

Note that your Kp and Kd gains will be different from the PD plus feed forward values from lab 2.

So given these equations, the flow of your code should be:

1) Calculate the desired trajectory
2) Given measured thetas, calculate actual states, error, error_dot.
   (Note: You do not need to calculate theta_dot here because you have access to it directly in the simulation.)
3) Calculate the outer loop control to come up values fora_{θ_1}, a_{θ_2} and a_{θ_3}.
4) Calculate the inner loop control to find control effort to apply to joint 1, 2 and 3.

Tune your Kp and Kd gains to minimize error as the joints following the trajectory.

After you have tuned your Kp and Kd gains for the "two second" trajectory you designed in Lab 2, create a faster moving trajectory to allow you to better tune your Kp and Kd gains and also to better see the advantages of the Inverse Dynamics control law. Have the trajectory start, t=0.0, at 0 radians (from the "zero" position, (theta_1,theta_1,theta_3)=(0,-pi/2,pi/2)). Have it follow a cubic path for 0.33 seconds to 0.5 radians from its starting position of (0,-pi/2,pi/2). Then have the joints stay at 0.5 radians from the starting position until 1 second has elapsed. From 1 second to 1.33 seconds, follow a cubic trajectory back to 0. Have time repeat every 2 seconds so that there is a pause between each cubic path to the new joint angle.

With this faster trajectory tune your Kp and Kd gains to minimize error. Produce trajectory response plots along with error plots for your report.

Task 2:

Compare Inverse Dynamics Controller to PD controller.

Compare PD plus feedforward to Inverse Dynamics

Below you will run a few comparisons between your PD plus feedforward control from Lab 2 and the inverse dynamics control you just designed. Hopefully these comparisons will demonstrate that the inverse dynamics control does a slightly better job controlling the linkage. Each step is listed.

1. Using your PD plus feedforward control from lab 2, control the robot using this control law to follow the new faster cubic trajectory. For this comparison, remove the integral control. With the PD plus feedforward implemented, tune its Kp and Kd gains to achieve similar error responses as your inverse dynamics control. For theta two and theta three, inverse dynamics should have zero steady state error where the PD will not be able to achieve zero steady state error when the links are in a position that gravity is pulling the links away from zero steady state error. We could use integral control to fix this but in order to see the advantages of inverse dynamics control do not implement integral control here. So you will not be able to tune the PD plus feedforward control to have the exact same error responses but try to get them pretty close to your inverse dynamics control.
2. Now that you have both your inverse dynamics controller and the PD controller working quite similar, we are going to add a 0.5kg point mass at the end effector of the robot. Then to compare, you will leave your PD plus feedforward controller gains unchanged and check the error responses with this mass added. For the inverse dynamics control, you will modify the parameters of the system that take into account this extra mass. With the new parameters applied, run your inverse dynamics controller and see if there is a noticeable improvement over the PD plus feedforward controller. Compare both the error peaks and the steady state error. The new parameters of the system with mass are given in the simulation, please have a look of fcn/fcn_params.m. Think about how the parallel access theorem is used to add in the additional mass. (Note: with this comparison we are being a bit unfair to the PD plus feedforward controller. When we add the mass to the end of the third link we are using new parameters for the inverse dynamics controller, but

for the PD plus feedforward controller we are not changing the J3 value or adjusting gains. I wanted the differences to be more obvious so you can hopefully see the advantage of the inverse dynamics control.)

3. By comparing these two controllers, does inverse dynamics controller perform better? For example, does the response improve during quick movements of the arm when the nonlinearities are prevalent? Are the effects of gravity on the response improved with the inverse dynamics controller? Any other differences you can note. What do you think would happen if we did not know the parameters of the robot exactly like we are using in the simulation?

Report: (Minimal Requirements)

1. Include the final version of your Matlab code. (MAIN.m, fcn_gen_traj.m and fcn_controller.m)
2. In your own words, explain the inverse dynamics control algorithm.
3. Answer the questions found in the lab.
4. Trajectory response plots and error plots.
5. Your observations about the two controllers used in this lab.