Lab 5 part 2
Preliminary: Matlab installed with Simulink, Robotics Toolbox(RTB file) from Peter Corke's website is installed (double click from inside Matlab).

Goal: model the motion of a manipulator using 3 approaches. Use the same model as in Lab 5 part.
1st approach – joint positions PID gains only -- proportional gains are applied only to joint angle positions, q.
2nd approach - joint positions PID plus only gravity compensation (variable ArmModel) -- feed-forward error is added to joint angle position error after the PID regulator in 1st approach. Set the inertia matrix elements are ZERO.
3rd approach -- joint positions PID plus inverse dynamic model (variable Arm) plus joint velocity PID – feed-forward error is added to joint angle position error after the above PID regulator. Inertia elements are not zero. Joint velocity error (after its own PID) is added to the joint position error.

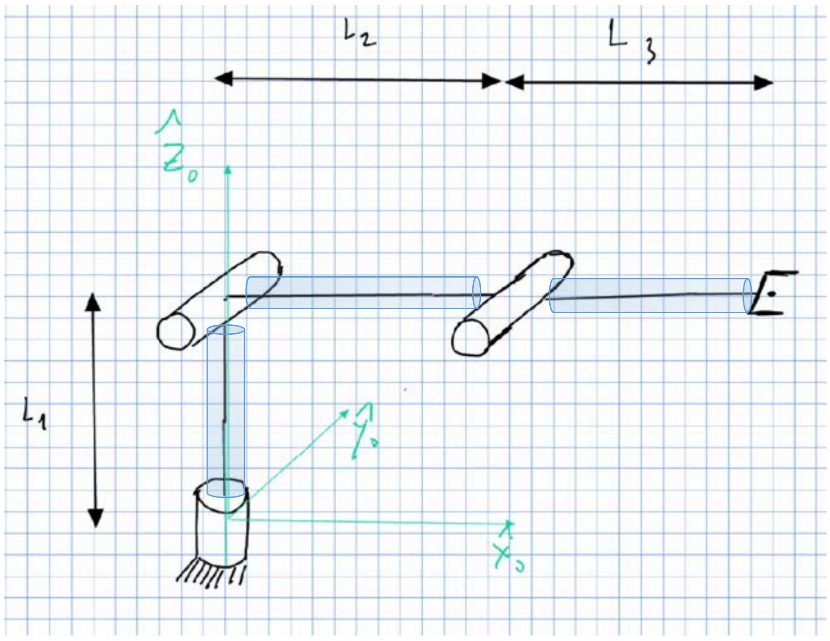
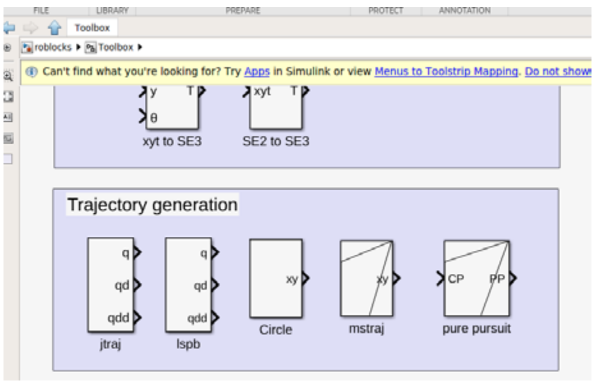


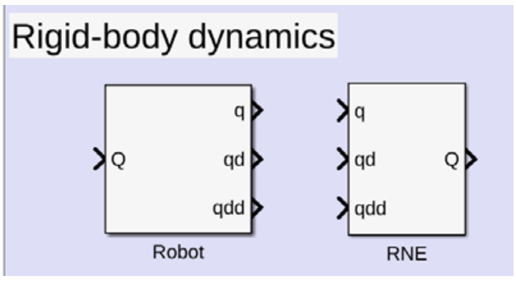
Fig. 1 – robot model for Lab 5

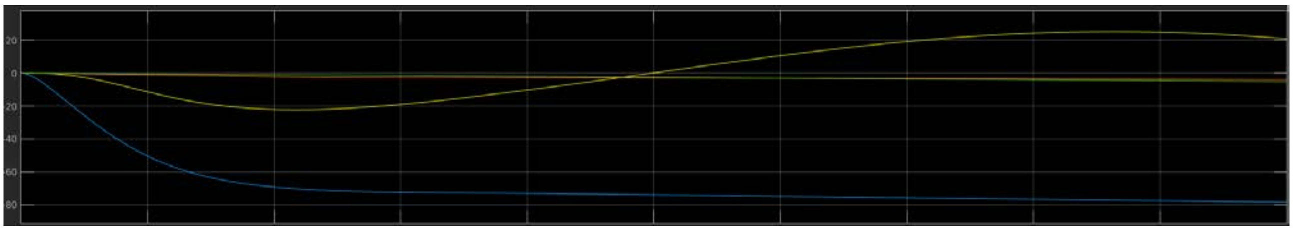
Step 7: Prepare the Simulink model by using the blocks provided in the file rvctools/simulink/roblocks.mdl (you need to know where the toolbox was installed to open this Simulink file) and the matlab model you prepared just before (the manipulator you created is represented as a data structure in the matlab workspace).

Step 8: Control the robot joints by providing a smooth trajectory implemented with the block "**jtraj**" (included in the file roblocks.mdl) and by using a simple MIMO proportional controller (based on position error). Initial position **θ_{init}=[0 0 0],** final position **θ_{final}=[90 -45 -45]** degrees**,** duration of the movement **t=2s**. 80 points.



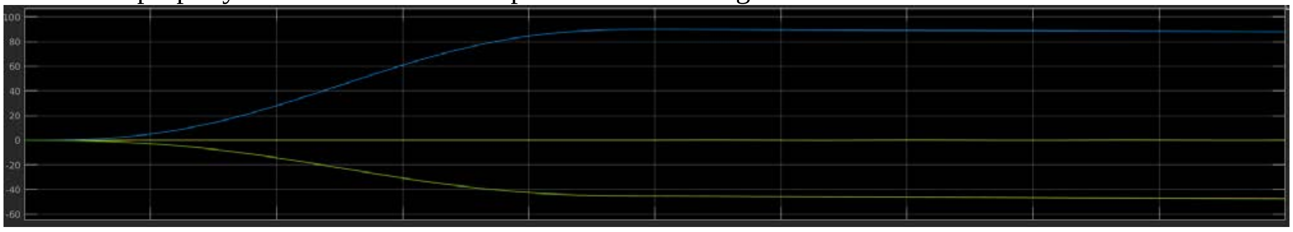
Hint: For building a robot you need these blocks below, and specify Robot object as you named your robot in Matlab. To understand how these blocks work, please, try rvctools/simulink/sl_ztorque.slx with Puma560_Model.m





Not tuned position error in proportional controller

Tune them properly to reach to the desired position as in the figure below.

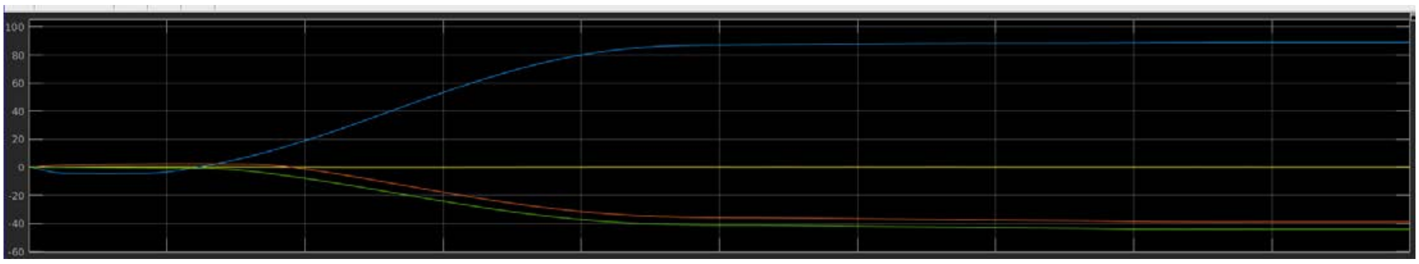


Step 9: In matlab prepare a second model (the one you will be using to test the dynamic compensation control scheme) by duplicating your original model and erasing inertias, and the other parameters you do not want to consider in your control scheme. At the beginning you may consider only the gravity vector (gravity compensation scheme), then you can add the links inertia, and so on, to see if the performances of your controller are improving (error signal, control action signal, joint position, velocity, and acceleration).
Hint: Set the inertia and mass values to 0, and use RNE in Robot-body dynamics. 90 points.

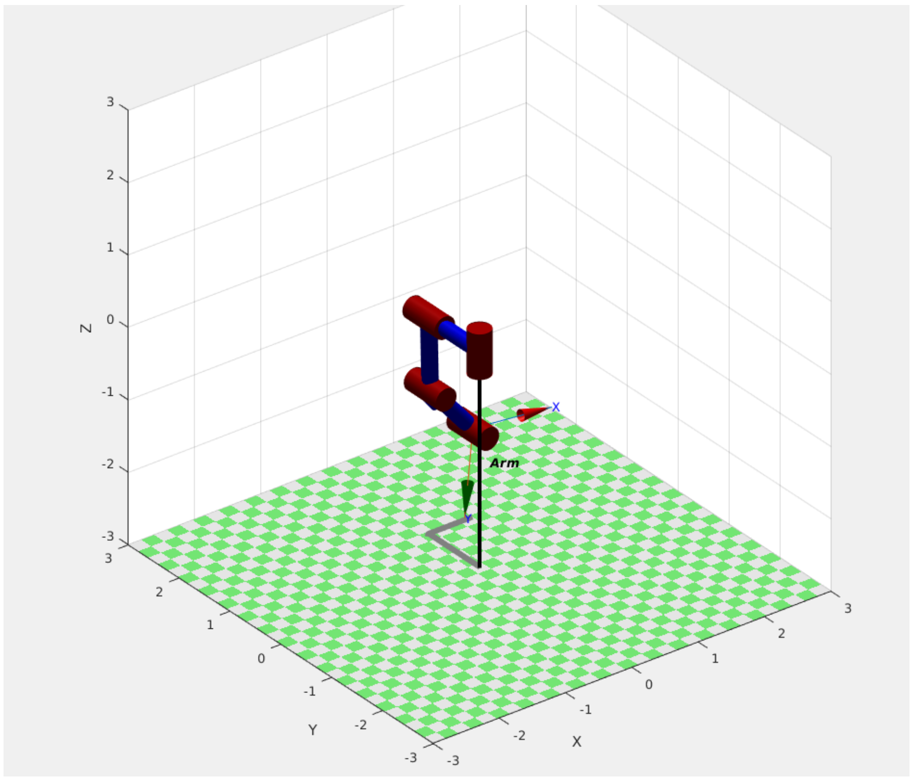
Step 10: Implement a model-based manipulator control system with the model based portion outside the position servo loop (Craig book Fig 10.6 pg. 298), i.e. use the inverse dynamic model (RNE block in the file roblocks.mdl) to compensate for the robot dynamics (use as reference for your model (model of the model ☺) the signals coming from the "**jtraj**" block, the performances will be better in comparison of using the sensors outputs).

Step 11: Improve the controller of **Step 10** by introducing also a control action based on the velocity error. Compare the performances with the previous controllers (prepare different plots). 99 points.

If you completed the tasks correctly, at the end you will get the graph like this:



and your output robot animation:



Step 12: Apply the controller Craig book Fig 10.6 pg. 298 to the 5-link planar robot (ROS labs). 100 points.

Files to submit:

1) Link to Matlab code to Github repository

2) Screen record of the moving robot model (scroll the control joint bar and move the joints) to Youtube or your own drive link.

3) In your text with the link to the github answer: Why the torques are different for different configurations?