Control of industrial robots
Prof. Paolo Rocco

Inverse dynamics control of a manipulator

Consider the two d.o.f. planar manipulator sketched in the picture. Assume the following values for geometrical and physical parameters of the two links:

- lengths: \( a_1 = a_2 = 1 \, \text{m} \)
- distances of centers of mass from joint axes: \( h_1 = h_2 = 0.5 \, \text{m} \)
- masses: \( m_1 = m_2 = 50 \, \text{kg} \)
- moments of inertia around axes parallel to \( z_0 \) and passing through the centers of mass: \( I_{1z} = I_{2z} = 10 \, \text{kg m}^2 \)

Assume that the motion is actuated through two motors connected to the links by means of rigid joints. Use the following numerical values:

- moment of inertia of the motors around their own axes: \( J_{m1} = 5 \times 10^{-3} \, \text{kg m}^2 \), \( J_{m2} = 2 \times 10^{-3} \, \text{kg m}^2 \)
- reduction ratios: \( n_1 = n_2 = 100 \)

Consider the robot initially at steady-state with the end effector located in the point \((0.2, 0)\), in a lower elbow posture. We want to design an inverse dynamics control system, that allows to move the robot along the \( x_0 \) axis for a 1.6 m displacement in 0.5 s, with a maximum speed of 5 m/s.

How to proceed

1. Define the manipulator with the command `SerialLink`, taking care of defining all the geometrical and physical parameters, including the moments of inertia of the motors and the reduction ratios;

2. Compose in Simulink the diagram that performs the computation of the inverse dynamics. The block takes as inputs joint positions and velocities as well as a third vector, that will be formed by the outputs of PD controllers. This vector plays the role of acceleration vector in the computation of the inverse dynamics. You will use the Newton-Euler algorithm (rne) available in the Toolbox. The Matlab function to be indicated in the related block will then be:

   \[
   r2.rne(u(1:2)', u(3:4)', u(5:6)')
   \]

   where \( r2 \) is the name of the previously defined robot object.

3. Compose now the diagram that implements the PD controllers. It takes as inputs joint positions and velocities, both actual and desired ones, computes the related errors and multiplies them by the gains KP and KD (identical for the two joints):
4. Using the blocks prepared in the previous lab sessions, assemble the Simulink model reported in the picture:

![Simulink Model](image)

For both the joints compute the gains KP and KD in such a way to assign to the closed loop poles of the system, made linear and decoupled by the application of the inverse dynamics method, a natural frequency of 100 rad/s and a damping factor of 0.7.

5. Simulate the system, taking care of indicating in the mask of the block that simulates the robot, within the “Robot object” field, the string `r2.perturb(0.1)`, which modifies the dynamic parameters of the robot by 10% with respect to the nominal values. This is to provide a more realistic simulation scenario.

6. Assume now that the joints of the manipulator are affected by **torsional flexibility**. Adopt the following numerical values:
   - stiffness constants of the transmissions: \( \text{Ke}_1 = \text{Ke}_2 = 70 \text{ Nm/rad} \)
   - viscous friction coefficients of the transmissions \( \text{De}_1 = \text{De}_2 = 0.05 \text{ Nms/rad} \)

Set to zero the moments of inertia of the motors inside the model of the robot, and compose, using the blocks prepared in the previous lab sessions, the following Simulink diagram:

![Simulink Model](image)

Keeping the same gains of the PD controllers previously determined, simulate the system and compare the performance to the case of manipulator with rigid joints.

7. Try out also the “PD plus gravity compensation” solution:
   - remove the computation of the derivative of the reference signal
   - change the Matlab function inside the “Inverse dynamics” block
   - simulate both the tracking of the time-varying reference and the stabilization at a given constant reference