Control of industrial robots
Prof. Paolo Rocco

Motion control of a manipulator with elastic joints

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: \( l_1 = l_2 = 0.5 \text{ m} \)
- masses: \( m_1 = m_2 = 50 \text{ kg} \)
- moments of inertia around axes parallel to \( z_0 \) and passing through the centers of mass: \( I_1 = I_2 = 10 \text{ kg m}^2 \)

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

- moment of inertia of the motors around their own axes: \( J_{m_1} = 5 \times 10^{-3} \text{ kg m}^2 \), \( J_{m_2} = 2 \times 10^{-3} \text{ kg m}^2 \)
- reduction ratios: \( n_1 = n_2 = 100 \)
- stiffness constants of the transmissions: \( K_{el_1} = K_{el_2} = 70 \text{ Nm/rad} \)
- viscous fiction coefficients of the transmissions: \( D_{el_1} = D_{el_2} = 0.05 \text{ Nms/rad} \)

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 independent joint control system (made by P/PI controllers with speed feed-forward), 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, using a trapezoidal velocity profile.

How to proceed
1. Define the manipulator with the command \texttt{SerialLink}, taking care of defining all the geometrical and physical parameters, but setting to zero the moments of inertia of the motors (which will be simulated with a separate model);
2. Compose a Simulink model, as the one shown here, which simulates the motors and the elastic transmissions:

![Simulink model](image)

3. Using the blocks assembled in the previous lab session, and the module of motors/transmission worked out in the previous step, compose the Simulink diagram reported in the picture:
4. For the design of the speed and position controllers for the two joints, you can make reference to values for the load side inertias equal to $J_l = 60 \text{ kg m}^2$, $J_r = 22.5 \text{ kg m}^2$. For each joint, compute the values $\mu$, $\omega_z$, $\zeta_z$, $\omega_p$, $\zeta_p$ and build the transfer function $G_{vm}$ from motor torque to motor speed:

$$G_{vm} = \text{tf}(\mu*[1/(\omega_z^2) 2*\zeta_z/\omega_z 1], [1/(\omega_p^2) 2*\zeta_p/\omega_p 1\ 0]);$$

5. Enter the interactive environment rltool and import the system previously determined ($P=G_{vm}$). Insert a pole in the origin for the controller ("compensator") as well as a zero in a suitably low frequency range (for example at a frequency $\omega_z/10$). Select then the gain of the controller in such a way as to maximize the damping of the closed loop poles. Compute the parameter $\tilde{\omega}_c$ (ratio between the nominal crossover frequency\(^1\) and the antiresonance frequency $\omega_z$) corresponding to this tuning.

6. For the purpose of the design of the position controller, we need to express the speed closed loop transfer function. Back to the Matlab prompt, the transfer function of the speed controller is assigned as:

$$R_v = \text{tf}(K_{pv}*[1\ 0\ T_{iv}], [1\ 0])$$

where $K_{pv}$ is the proportional gain of the controller and $T_{iv}$ is the reset time. The loop transfer function will be:

$$L_v = G_{vm}*R_v$$

and the closed loop one:

$$F_v = \text{feedback}(L_v, 1)$$

7. The transfer function "seen" by the position controller ($K_{pp}$) is $F_p$ in series connection with the integrator from speed to position:

$$G_{pm} = F_v*\text{tf}(1, [1\ 0]).$$

Import this transfer function in the rltool environment. Select the gain of the controller in such a way, for example, to achieve a damping of the poles equal to 0.7.

8. Compute the initial steady state, where the motor and link speeds are all zero, the motor positions are identical to those initially prescribed by the trajectory planner (suitably reported to the motor axes) while the link positions are such that the torques transmitted through the elastic elements balance the gravitational load:

$$N*K_{el}*(qm_0'-N*q_0') - r_2.\text{gravload}(q_0)' = g(q)$$

You can obtain $\tilde{q}$ in Matlab using function fsolve for the solution of the nonlinear implicit system:

$$q_0 = \text{fsolve}(\theta(q_0) \ N*K_{el}*(qm_0'-N*q_0')-r_2.\text{gravload}(q_0)', q_{rif0});$$

where $q_{rif0}$ is an initial guess set equal to the value initially prescribed by the trajectory planner.

9. Simulate the “brake release”, too, by including in the simulator at the controller output an additional torque term equal to the gravitational load (referred to the motor axes). At a certain instant, previous to the start of the trajectory, this term has to be set to zero, so as to approximately simulate the transient related to the brake release.

---

\(^1\) Obtained as $\omega_c = K_{pv} \frac{\mu}{K_{pv}(J_m+J_b)}$