Teoria Sterowania w Robotyce

Piotr Kicki

June 1, 2020

1 Introduction

Throughout the rest of the course, we will be considering a single dynamic object – 2 Degrees of Freedom (2DoF) Planar Manipulator. As you can see in Figure 1 it consists of 2 links, 2 joints, and an object attached at the tip.

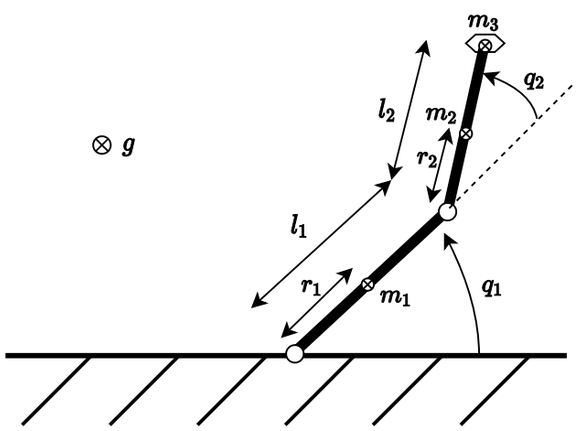

Figure 1: Scheme of the 2DoF manipulator

2 Derivation of dynamics of 2 DoF planar manipulator

In order to derive the dynamics of 2 DoF planar manipulator we will use a Lagrangian dynamic formulation – an energy based approach to dynamics. Namely, we will obtain dynamics from the formula

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}}\right) - \frac{\partial \mathcal{L}}{\partial q} = \tau, \tag{1}$$

where q is the robot state (configuration), τ are the actuator torques and

$$\mathcal{L} = T - V, \tag{2}$$

where T is a kinetic energy of the manipulator and V is its potential energy. For the sake of this considerations we assume that manipulator work space lies in a plane such that vector of the gravity is orthogonal to it, thus $V = 0$.

Lets consider kinetic energy T of the 2 DoF planar manipulator as a sum of the energies of its links (for simplicity, we are assuming that there is no object at the tip of the second link, thus $m_3 = 0$ and $I_{z3} = 0$)

$$T = T_1 + T_2, \tag{3}$$

where

$$T_i = \frac{1}{2} m_i v_i^2 + \frac{1}{2} I_{zi} \left(\sum_{k=1}^{i} \omega_k\right)^2, \tag{4}$$

which can be also reformulated to the form

$$T_i = \frac{1}{2} m_i \left(\dot{x}_i^2 + \dot{y}_i^2 \right) + \frac{1}{2} I_{zi} \left(\sum_{k=1}^{i} \omega_k \right)^2 , \tag{5}$$

where x_i and y_i are the coordinates of the center of mass, ω_i is the angular velocity, I_{zi} is the inertia around z axis and m_i is the mass of the i-th link.

At the very end we expect to obtain a dynamics in the following form

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} = \tau, \tag{6}$$

where $M(q)$ is the mass matrix, $C(q, \dot{q})$ is the matrix of Coriolis and centrifugal forces and τ is the vector of actuator torques. To do so, we have to express the $\mathcal{L}$ and thus T in terms of robot state q elements and its derivatives.

Let first define the coordinates of the links centers of masses in the global coordinate system. For first link they will be following

$$\begin{cases} x_1 = r_1 c_1 \\ y_1 = r_1 s_1 \end{cases} , \tag{7}$$

where $s_1 = \sin q_1$ and $c_1 = \cos q_1$ for brevity. Similarly, for second link we obtain

$$\begin{cases} x_2 = l_1 c_1 + r_2 c_{12} \\ y_2 = l_1 s_1 + r_2 s_{12} \end{cases} , \tag{8}$$

where $s_{12} = \sin(q_1 + q_2)$.

Then, take time derivative of (7) and (8) and obtain

$$\begin{cases} \dot{x}_1 = -r_1 s_1 \dot{q}_1 \\ \dot{y}_1 = r_1 c_1 \dot{q}_1 \end{cases} \tag{9}$$

and

$$\begin{cases} \dot{x}_2 = -l_1 s_1 \dot{q}_1 - r_2 s_{12}(\dot{q}_1 + \dot{q}_2) \\ \dot{y}_2 = l_1 c_1 \dot{q}_1 + r_2 c_{12}(\dot{q}_1 + \dot{q}_2) \end{cases} . \tag{10}$$

Next, substitute (5) with the time derivatives from (9) for first link

$$T_1 \overset{(5)}{=} \frac{1}{2} m_1 \left(\dot{x}_1^2 + \dot{y}_1^2 \right) + \frac{1}{2} I_{z1} \dot{q}_1^2 \overset{(9)}{=} \frac{1}{2} \left(m_1 r_1^2 + I_{z1} \right) \dot{q}_1^2 \tag{11}$$

and the second link

$$\begin{aligned} T_2 \overset{(5)}{=} & \frac{1}{2} m_2 \left(\dot{x}_2^2 + \dot{y}_2^2 \right) + \frac{1}{2} I_{z2} (\dot{q}_1 + \dot{q}_2)^2 \overset{(10)}{=} \\ \overset{(10)}{=} & \frac{1}{2} m_2 \left(\left(l_1 \dot{q}_1 s_1 + r_2 (\dot{q}_1 + \dot{q}_2) s_{12} \right)^2 + \left(l_1 \dot{q}_1 c_1 + r_2 (\dot{q}_1 + \dot{q}_2) c_{12} \right)^2 \right) \\ & + \frac{1}{2} I_{z2} (\dot{q}_1 + \dot{q}_2)^2 = \\ = & \frac{1}{2} m_2 \left(l_1^2 \dot{q}_1^2 s_1^2 + 2 l_1 \dot{q}_1 s_1 r_2 (\dot{q}_1 + \dot{q}_2) s_{12} + r_2^2 (\dot{q}_1 + \dot{q}_2)^2 s_{12}^2 \right) + \\ & + \frac{1}{2} m_2 \left(l_1^2 \dot{q}_1^2 c_1^2 + 2 l_1 \dot{q}_1 c_1 r_2 (\dot{q}_1 + \dot{q}_2) c_{12} + r_2^2 (\dot{q}_1 + \dot{q}_2)^2 c_{12}^2 \right) \\ & + \frac{1}{2} I_{z2} (\dot{q}_1 + \dot{q}_2)^2 = \\ & \frac{1}{2} m_2 \left(l_1^2 \dot{q}_1^2 + r_2^2 (\dot{q}_1 + \dot{q}_2)^2 + 2 l_1 (\dot{q}_1 + \dot{q}_2) r_2 \dot{q}_1 (s_1 s_{12} + c_1 c_{12}) \right) + \\ & + \frac{1}{2} I_{z2} (\dot{q}_1 + \dot{q}_2)^2 = \\ = & \frac{1}{2} m_2 \left(l_1^2 \dot{q}_1^2 + r_2^2 (\dot{q}_1 + \dot{q}_2)^2 + 2 l_1 (\dot{q}_1 + \dot{q}_2) r_2 \dot{q}_1 c_2 \right) + \frac{1}{2} I_{z2} (\dot{q}_1 + \dot{q}_2)^2 \end{aligned} \tag{12}$$

Then, lets find the derivatives of $\mathcal{L}$ with respect to $\dot{q}$ as they are necessary to obtain (1)

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \dot{q}_1} =& \left(m_1 r_1^2 + I_{z1} + m_2(l_1^2 + r_2^2) + I_{z2} + 2m_2 l_1 r_2 c_2\right)\dot{q}_1 \\ &+ \left(m_2 l_1 r_2 c_2 + m_2 r_2^2 + I_{z2}\right)\dot{q}_2 = \\ =& \alpha\dot{q}_1 + \gamma\dot{q}_2 + \beta c_2(2\dot{q}_1 + \dot{q}_2)\end{aligned}, \tag{13}$$

where for simplicity we used following substitiution

$$\begin{cases}\alpha = m_1 r_1^2 + I_{z1} + m_2(l_1^2 + r_2^2) + I_{z2} \\ \beta = m_2 l_1 r_2 \\ \gamma = m_2 r_2^2 + I_{z2}\end{cases}. \tag{14}$$

Similarly, we get

$$\frac{\partial \mathcal{L}}{\partial \dot{q}_2} = \gamma(\dot{q}_1 + \dot{q}_2) + \beta c_2 \dot{q}_1, \tag{15}$$

and subsequently derivatives with respect to q

$$\frac{\partial \mathcal{L}}{\partial q_1} = 0, \tag{16}$$

and

$$\frac{\partial \mathcal{L}}{\partial q_2} = -\beta s_2(\dot{q}_1^2 + \dot{q}_1\dot{q}_2). \tag{17}$$

Last step to perform is to take the time derivative of (13)

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_1}\right) = (\alpha + 2\beta c_2)\ddot{q}_1 + (\gamma + \beta c_2)\ddot{q}_2 - \beta s_2 \dot{q}_2(\dot{q}_1 + \dot{q}_2), \tag{18}$$

and (15)

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_2}\right) = (\gamma + \beta c_2)\ddot{q}_1 + \gamma\ddot{q}_2 + \beta s_2 \dot{q}_1^2. \tag{19}$$

Then, we can form a matrix form of (1) with the formulas derived in (16), (17), (18) and (19), obtaining

$$\begin{bmatrix}\alpha + 2\beta c_2 & \gamma + \beta c_2 \\ \gamma + \beta c_2 & \gamma\end{bmatrix}\begin{bmatrix}\ddot{q}_1 \\ \ddot{q}_2\end{bmatrix} + \begin{bmatrix}-\beta s_2 \dot{q}_2 & -\beta s_2(\dot{q}_1 + \dot{q}_2) \\ \beta s_2 \dot{q}_1 & 0\end{bmatrix}\begin{bmatrix}\dot{q}_1 \\ \dot{q}_2\end{bmatrix} = \begin{bmatrix}\tau_1 \\ \tau_2\end{bmatrix}. \tag{20}$$

Notice: The form of the $C(q, \dot{q})$ matrix is determined with the use of Christoffel symbols. However, for our purposes it is not obligatory.

2.1 Exercise

Derive dynamics of 2 DoF planar manipulator with a point mass m_3 at the tip of the second link.

3 Feedback linearization

As you can see, the system described with (20) is nonlinear. There are some ways of dealing with this class of systems. The one which we will cover here – feedback linearization is one of those.

Form engineering studies you already familiar with linear dynamical systems of the form

$$\dot{x} = Ax + Bu, \tag{21}$$

however, the real world is usually nonlinear and only sometimes can be approximated well with the linear model.

Lets consider in general nonlinear system of the form

$$\dot{x} = f(x, \cdot) + g(x, \cdot)u, \tag{22}$$

where $f(x, \cdot)$ and $g(x, \cdot)$ are in general nonlinear functions of the state x and some other parameters. If we assume that $g(x, \cdot) \neq 0$ for all $t > 0$. We can propose a feedback

$$u(x, \cdot) = \frac{1}{g(x, \cdot)} \left(v - f(x, \cdot) \right), \tag{23}$$

where v is the new desired input. Then, system (22) simplifies to the following form

$$\dot{x} = f(x, \cdot) + g(x, \cdot)u \overset{(23)}{=} f(x, \cdot) + g(x, \cdot) \left(\frac{1}{g(x, \cdot)} \left(v - f(x, \cdot) \right) \right) = v. \tag{24}$$

In effect, we got a linear system, where $A = 0$ and $B = \mathbb{I}$, which is called an *integrator*.

An attentive student can spot that it is also possible to linearize (6) in that manner. We can propose following control law

$$\tau = M(q)v + C(q, \dot{q})\dot{q}, \tag{25}$$

which applied to (6) results in following dynamics

$$\ddot{q} = v. \tag{26}$$

Such a system is called the *double integrator* and can be controlled by the designed $v = \ddot{q}_r$.

So in fact, having the ideal model of the manipulator one is able to control it using the desired acceleration signal $\ddot{q}_r$. By doing so, we obtain an *open loop* feed forward controller! You should be a bit scared now - there is no feedback! If we make a mistake and wait, then we will end up with error growing and growing. Let's fight it! Define

$$v = \ddot{q}_r + K_d(\dot{q} - \dot{q}_r) + K_p(q - q_r), \tag{27}$$

thus we got a combined the feed forward part $\ddot{q}_r$ with the PD feedback $K_d(\dot{q} - \dot{q}_r) + K_p(q - q_r)$. By doing so we can observe, that even some parametric uncertainty exists, controller is able to overcome this. Under some mild conditions there could be even a proof given that system (6) with control (25), (27) is stable even for not ideal model.

4 Polynomial trajectory

Polynomials are widely used in engineering, as a representation of paths and trajectories. They are simple and guarantee some desired level of smoothness. In our laboratory we will use following parametrization of 3rd degree polynomials

$$q_r(t) = \sum_{i=0}^{3} a_i t^i (1 - t)^{3-i} \quad \text{for} \quad t \in [0; 1], \tag{28}$$

where a_i are the parameters of the polynomials, which has to be determined using initial and desired states q_0 and q_d, and velocities $\dot{q}_0$ and $\dot{q}_d$.

To determine parameters a_i one need to create a system of 4 equations

$$\begin{cases} q_r(0) = q_0 \\ \dot{q}_r(0) = \dot{q}_0 \\ q_r(1) = q_d \\ \dot{q}_r(1) = \dot{q}_d \end{cases}. \tag{29}$$

To do so, we have to compute derivatives of (28) with respect to the parameter t, namely

$$\dot{q}_r = \frac{dq_r}{dt}(t) = \frac{d}{dt}\left(\sum_{i=0}^{3} a_i t^i (1-t)^{3-i}\right). \tag{30}$$

Next, we have to evaluate (28) and (30) at 0 and 1 and plug in into (29), and solve for all parameters a_i.

Notice In the laboratory we will be assuming that $\dot{q}_{r0} = \dot{q}_d 0 = 0$.

Finally, having the polynomial parameters one can calculate the second derivative

$$\ddot{q}_r = \frac{d^2 q_r}{dt^2}(t) = \frac{d^2}{dt^2}\left(\sum_{i=0}^{3} a_i t^i (1-t)^{3-i}\right), \tag{31}$$

which will be our new control.

5 Tasks - Laboratory 1

1. Implement model of the 2DoF manipulator, using the parameters from the manipulators/planar_2dof.py (remember to set self.m_3 to 0) and equation (20).
2. Implement Feedback Linearization Controller, based on the previously implemented model and (25) (assume $v = \ddot{q}_r$).
3. Change the mass m_3 in the manipulator object to nonzero value and observe how the performance of the controller degrades (structural uncertainty).
4. Use the derivation from exercise 2.1 to implement model which takes mass m_3 into account.
5. Observe how the parametric uncertainty affects the performance of the controller.
6. Implement the polynomial trajectory generator, according to point 4.
7. Check out how the polynomial trajectory generator works together with the controller.
8. Add a feedback to v, according (27), and observe how the parametric uncertainty affects the performance of the controller.

6 Adaptive Control

In this section we will consider an adaptive control. In our last considerations, as well as throughout the engineering studies, you were analyzing stationary processes. There was some manipulator, with constant properties (such as mass, links lengths etc.), however in real-life we meet a lot of processes where that is not a case! For example: rocket, which burns the fuel during the flight and modifies its mass during the flight, or manipulator, which picks and moves different objects. In both cases, simple stationary controller, which do not adapt to the changes in the model, can be not enough to control the system. However, in such cases, we can use Adaptive Control. It is a vast branch of control theory, which considers the situations, when it is necessary to adapt the behavior of the controller to the changing dynamics of the process being controlled.

In this class we will focus on the method called Multi-model Adaptive Control, which is the topic of the next Section, and another technique called Active Disturbance Rejection Control.

7 Multi-model Adaptive Control

Multi-model Adaptive Control is a term which unifies many different approaches to model-based adaptive control. General scheme of this approach is depicted in Figure 2. Basically the idea of MMAC is the following: based on the output of the object y, and the output of the models, models selector decide, which model approximates the object dynamics the best at given moment and use that model to produce the control signal. Simply, try to predict the future witch some bunch of models, observe how the reality works, and choose the most suitable one for control.

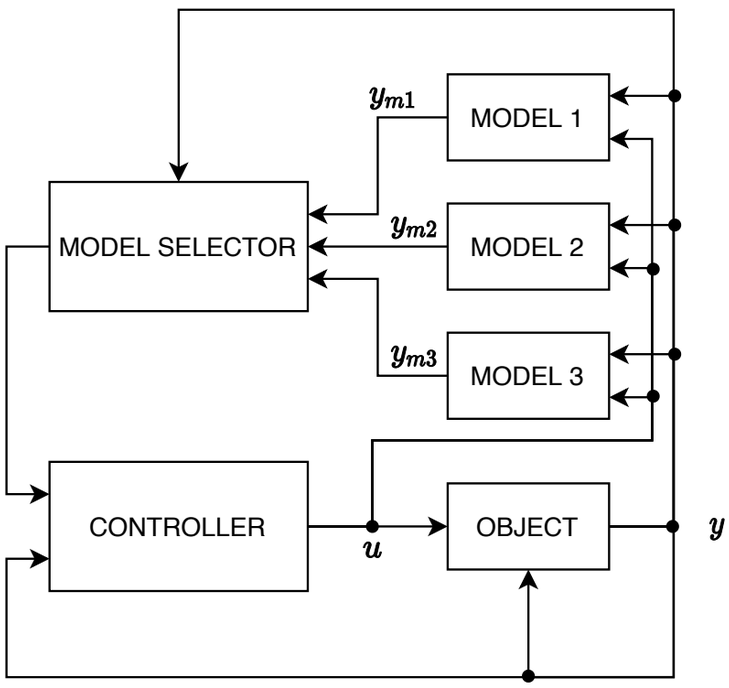

Figure 2: General scheme of MMAC.

Seems straightforward and easy, but actually it works. However, as every method this one has also its drawbacks. Can you name one?

It is important to notice how big impact has the switching behavior! For example: it can destabilize the system, even each of individual controllers is stable (see Figure 3), or it can decrease the control signal quality by making in discontinuous.

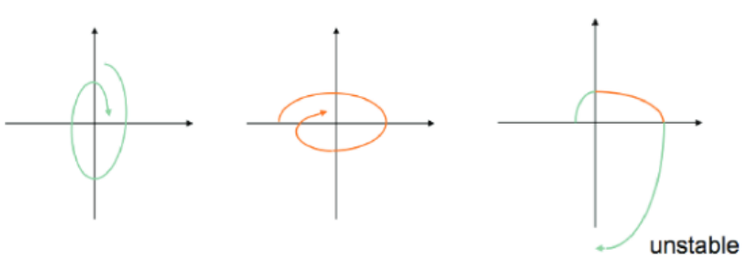

Figure 3: MMAC can be unstable, even each of controllers alone is stable.

There is a lot of research in the area od Adaptive Control and MMAC, some basic view on the field can be found in `http://www.gipsa-lab.fr/~ioandore.landau/adaptivecontrol/Transparents/Courses/AdaptiveCourse5GRK.pdf`.

In our classes we consider only very basic version of Multi-model Adaptive Control (MMAC), which can be described with the scheme in Figure 4.

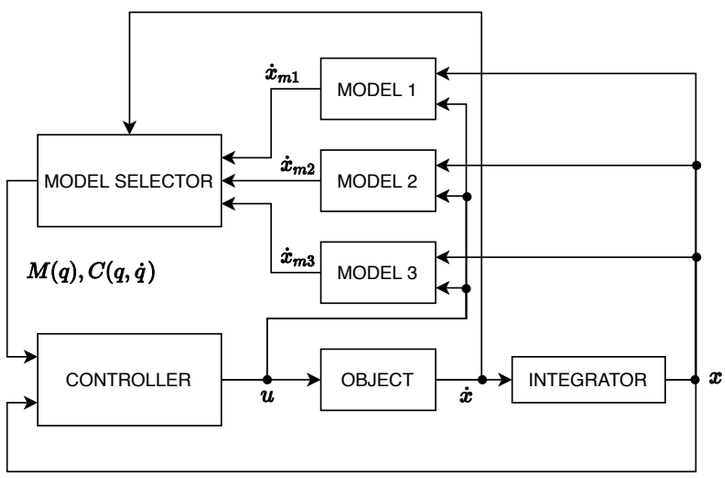

Figure 4: General scheme of the MMAC version we will implement in the lab.

We will use the Feedback Linearization Controller (FLC) you implemented previously, and we will only modify its model by choosing the best one out of three model candidates. Model selection will be made, based on the comparison of the $x\dot{x}$ obtained from the object, and predicted $\dot{x}_{mi}$ by the models. The idea is to choose the model with the lowest error

$$\arg\min_i e_i = \arg\min_i (\dot{x} - \dot{x}_{mi}). \tag{32}$$

And then, to use this model to produce the matrices $M(q)$ and $C(q, \dot{q})$, which will be used by the FLC.

In our setup we will assume that there are 3 states, in which mass m_3 and radius R_3 of the object at the tip is different. States are put in the circular queue and the change of states is modeled as the Bernoulli process, where drawing 1 means that we change the model for the next one in the queue.

8 Tasks - Laboratory 2

1. Implement the list of 3 models (you can use the class created in the previous tasks), which parameters will be same as in the object class (*manipulators/mm_planar_2dof.py*)
2. Implement the model selector, which will choose the most suitable model
3. Implement the FLC with the use of selected model
4. Observe how your model is doing

9 Decentralized controllers

Till now, we were considering a group of controllers, which can be called together *centralized controllers*. But what does it actually mean? Centralized or decentralized refers tot the way how we tract the system of the manipulator, namely if we consider it as a whole system or rather ordered set of

subsystems (links with joints associated with them). Till now, we modeled the whole manipulator at once, and used that model for control, but now it will change and we will try to control the system without the model and taking care of each joint separately. General scheme of the decentralized controller is shown in Figure 5.

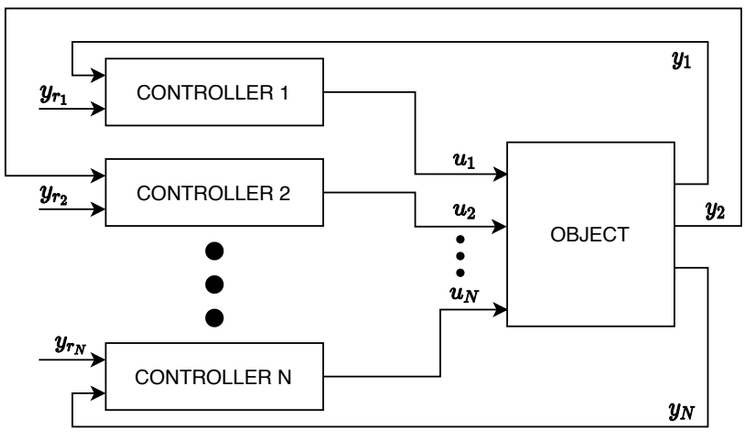

Figure 5: General scheme of the decentralized controller.

10 PD decentralized controller

Before we will dive into the ADRC, we will show how the decentralized controllers works, using the simple PD controller. Firstly, we will assume that both position and velocity of each joint is known, and we will construct a simple PD controller

$$u_i = k_{p_i} e_i + k_{d_i} \dot{e}_i, \tag{33}$$

where

$$e_i = q_{r_i} - q_i, \tag{34}$$

and

$$\dot{e}_i = \dot{q}_{r_i} - \dot{q}_i, \tag{35}$$

and $k_{p_i}, k_{d_i} > 0$ are some positive constants.

The main disadvantage of this approach is that we totally ignore the cross-couplings of joints, which are clearly visible in (20) (however, those couplings are diminished in the real manipulator by the gears). Another one is that we do not use the model of the system, thus we do not use the knowledge about the system, which usually results in the lower control quality. In turn, the main advantage of this approach is its simplicity.

11 ADRC

Knowing what are the decentralized controllers we can jump into one of the most powerful and conceptually simple control techniques - Active Disturbance Rejection Control. The main idea behind the ADRC is the following: given a dynamical system

$$\dot{x} = f(x, \cdot) + bu, \tag{36}$$

where $f(x, \cdot)$ is some unknown nonlinear function (which we call *total disturbance*) and b is a constant, we can using the similar idea to feedback linearization controller choose the

$$u = \frac{v - f(x, \cdot)}{b}, \tag{37}$$

obtaining the integrator. Then, we can use the same control as in (27).

But wait! Can we really get the $f(x,\cdot)$ and b? In real life? Obviously not! So, what we do if we don't know something, but we really want to? Estimate!

Let's introduce the estimate $\hat{b}$ of the b parameter, and the estimate $\hat{f}$ of the function f. One can ask, how to calculate those estimates, and here is when the magic begins. Let us construct the extended state of the system

$$z = \begin{bmatrix} x \\ f \end{bmatrix}, \tag{38}$$

since then the system can be written as

$$\dot{z} = \begin{bmatrix} \dot{x} \\ \dot{f} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} z + \begin{bmatrix} b \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ \dot{f} \end{bmatrix} = Az + Bu + \begin{bmatrix} 0 \\ \dot{f} \end{bmatrix}, \tag{39}$$

which is nice and linear, except the $\dot{f}$ part. However, there is some catch - we do not have an access to the f and $\dot{f}$ signals and the value of b. Using our estimates we can rewrite (39) to the following form

$$\dot{\hat{z}} = \begin{bmatrix} \dot{\hat{x}} \\ \dot{\hat{f}} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \hat{z} + \begin{bmatrix} \hat{b} \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ \dot{\hat{f}} \end{bmatrix}. \tag{40}$$

Here, we somewhat have *fixed* already the problems with f and b (maybe not for now, but we will do it in a moment), but the $\dot{\hat{f}}$ remains unknown. This problem will be solved in a different way. Another way of handling unknown signals is to just ignore them :D But not so arbitrarily! We can assume that the changes of $\dot{\hat{f}}$ are very small, but we have a good reason to do that – first, those changes can be indeed very small by the nature of the system, second, if the first is not true, we can pretend that it is by sampling the system with very high frequency.

Now we will focus on the estimation of the $\hat{f}$. To do so, we will use the extended state observer (ESO)

$$\dot{\hat{z}} = A\hat{z} + \hat{B}u + L(y - C\hat{z}), \tag{41}$$

where L is a matrix of the observer gains, y is the measurable output of the system and C is a matrix, which extracts from the vector z the part which is observable. In case when $C = [10]$ and $y = x$

$$\dot{\hat{z}} = A\hat{z} + \hat{B}u + L(x - \hat{x}). \tag{42}$$

Let's check out the observer error

$$e_o = z - \hat{z}, \tag{43}$$

and its dynamics

$$\dot{e}_o = Az + Bu - A\hat{z} + \hat{B}u - LC(z - \hat{z}). \tag{44}$$

Assuming that the $\hat{B} = B$, we can write

$$\dot{e}_o = A(z - \hat{z}) - LC(z - \hat{z}), \tag{45}$$

and then

$$\dot{e}_o = (A - LC)e_o = H_o e_o. \tag{46}$$

This looks really nice, as it is linear and we can manipulate the L matrix in order to obtain the desired dynamics of the error, which is stable and drives the error to 0 asymptotically.

To sum up, by assuming the known $\hat{b}$ and $\dot{f} = 0$, we can propose a control scheme, which estimates the dynamics of the system online and use it to linearize and control the system.

12 ADRC for our manipulator

In case of our manipulator, each of the joints can be described with the state

$$x_i = \begin{bmatrix} q_i \\ \dot{q}_i \end{bmatrix}. \tag{47}$$

Thus the matrix A for the extended state is defined as

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \tag{48}$$

and B is defined as

$$B = \begin{bmatrix} 0 \\ b \\ 0 \end{bmatrix}, \tag{49}$$

whereas

$$C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \tag{50}$$

as we assume that only positions q_i are measurable, and

$$L = \begin{bmatrix} l_{i_1} \\ l_{i_2} \\ l_{i_3} \end{bmatrix}. \tag{51}$$

First, you have to guess a $\hat{b}$ value roughly. Next, use pole placement to determine the value of L, by comparing the eigenvalues of the H_o matrix to the stable characteristic equation such as

$$(\lambda + p)^3, \tag{52}$$

where p is the desired location of the pole. To choose the best value of p, we can run the system in the open loop control, observe the estimation error and then try to choose p to minimize it. Finally, you can choose the PD controller gains, for example using pole placement also.

13 Tasks - Laboratory 3

1. Implement decentralized PD controller
2. Adjust its gains to achieve nice performance
3. Change the mass at the tip m_3 and observe how the performance decreases (the decrease will be greater if you tune the controller very precisely to the default mass)
4. Go back to the default mass
5. Implement Extended State Observer (starting with $\hat{b}$ about 10 is ok)
6. Tune in in the open-loop (p can be quite large - values about 100 are normal)
7. Implement ADRC controller (in PD controller use estimates instead of real $\dot{q}$ value)
8. Tune the controllers in order to obtain satisfactory quality using pole placement
9. Observe how the controller reacts to the changing parameters (for example m_3) - the performance should be quite robust to those changes

14 Bibliography

Richard M. Murray, S. Shankar Sastry, and Li Zexiang. 1994. A Mathematical Introduction to Robotic Manipulation (1st. ed.). CRC Press, Inc., USA.