COSMOS: Making Robots and Making Robots Intelligent
Lecture 7: A guided task: the inverted pendulum

Jorge Cortés and William B. Dunbar

October 27, 2005

Abstract

In this lecture, we use feedback control into a more complex system: the pendulum. As we explained in the previous lecture, feedback control can be used to shape the equilibria of the system, and also to stabilize them. That's what we will do here. This practice will prepare us for the `Robobrain` system.

Contents

1 A guided task: the inverted pendulum

Consider the pendulum in Figure 1. For simplicity, assume that there is no friction with the air. Then, a model of the motion of the pendulum based on Newton's second law is

$$\frac{1}{\Delta^2}(\theta_{k+1} - 2\theta_k + \theta_{k-1}) = -\frac{g}{l}\sin\theta_k,$$

where $g = 9.8$ corresponds to gravity, and $l = 9.8$ is the pendulum length chosen such that the ratio $g/l = 1$. So, from now on, we will replace g/l with 1.

Let us express this in the form of a discrete-time dynamical system, as we know. Define $\mathbf{x} = (z, y)$ as a 2-dimensional vector and at time instant k, let $\mathbf{x}_k = (z_k, y_k) = (\theta_k, \theta_{k-1})$. Then, we have the 2-dimensional iterated map

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k), \tag{1}$$

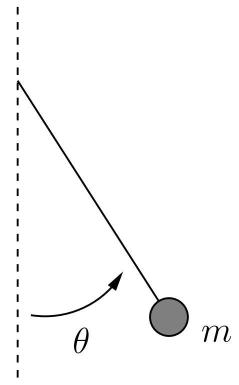

Figure 1: Pendulum example.

where

$$f(\mathbf{x}_k) = f(\theta_k, \theta_{k-1}) = (2\theta_k - \theta_{k-1} - \Delta^2 \sin\theta_k,\ \theta_k),$$

or

$$f(\mathbf{x}_k) = f(z_k, y_k) = (2z_k - y_k - \Delta^2 \sin z_k,\ z_k),$$

It should be clear that the map f is now 2-dimensional, instead of 1-dimensional. All the notions we have introduced so far about equilibrium points, stability, etc, are valid in this context. We will check stability through simulations, although there exist formal methods to study it – but they require linear algebra, that we do not know yet. As in the predator-prey system, we can write the two equations in the more familiar form

$$\begin{cases} z_{k+1} = 2z_k - y_k - \Delta^2 \sin z_k \\ y_{k+1} = z_k \end{cases} \tag{2}$$

where $k = 1, 2, 3, \ldots$. Also, as before, given a starting state $\mathbf{x}_1 = (z_1, y_1) = (\theta_1, \theta_0)$, we can compute the orbit, that is, the trajectory or path of angles θ, for the pendulum, for some number of iterations.

Task 1.1 Why do we start with $k = 1$? What are the equilibrium points of the pendulum? Use equation (1) or (2) to find them. Use MATLAB® to determine the stability of each one (requires writing a program like the predator-prey program...).

1.1 Shaping the equilibrium

Now, assume we can push the pendulum. Let's introduce this force in the model of its motion,

$$\frac{1}{\Delta^2}(\theta_{k+1} - 2\theta_k + \theta_{k-1}) = -\sin\theta_k + u_k.$$

By means of the input u_k, we can affect the dynamics of the pendulum. Using the map f above, this can be rewritten as

$$\begin{cases} z_{k+1} = 2z_k - y_k - \Delta^2 \sin z_k + \Delta^2 u_k \\ y_{k+1} = z_k \end{cases} \tag{3}$$

The first thing we are going to do is to select u_{eq} such that $\theta_{\text{eq}} = \frac{\pi}{2}$ is an equilibrium (funny position, eh!).

Task 1.2 Find u_{eq} such that $\mathbf{x}_{\text{eq}} = (\frac{\pi}{2}, \frac{\pi}{2})$ is an equilibrium of the system (3).

Once we have found u_{eq}, we select $u_k = u_{\text{eq}} + \tilde{u}_k$, and substitute it in equation (3) to get

$$\begin{cases} z_{k+1} = 2z_k - y_k - \Delta^2 \sin z_k + \Delta^2(u_{\text{eq}} + \tilde{u}_k) \\ y_{k+1} = z_k \end{cases} \tag{4}$$

The unforced system looks now like

$$\begin{cases} z_{k+1} = 2z_k - y_k - \Delta^2 \sin z_k + \Delta^2 u_{\text{eq}} \\ y_{k+1} = z_k \end{cases}$$

and, at equilibrium, we have

$$\begin{cases} z_k = 2z_k - z_k - \Delta^2 \sin z_k + \Delta^2 u_{\text{eq}} \\ y_k = z_k \end{cases}, \quad \text{for all } k = 1, 2, 3, \dots$$

If u_{eq} was chosen properly, then the only solution is $\mathbf{x}_{\text{eq}} = (\frac{\pi}{2}, \frac{\pi}{2})$.

Task 1.3 Is the equilibrium $\mathbf{x}_{\text{eq}} = (\frac{\pi}{2}, \frac{\pi}{2})$ of the unforced system $\mathbf{x}_{k+1} = \tilde{f}(\mathbf{x}_k, 0)$ stable or unstable? To figure this out, you can plot various orbits of the unforced dynamics starting close to $\mathbf{x}_{\text{eq}} = (\frac{\pi}{2}, \frac{\pi}{2})$. To do this, use in MATLAB® the program `pendulum.m`. First download it from the usual website and place it in your working directory. Type "help pendulum" in the command window to understand the syntax of the function. For instance, `pendulum`([pi/2 + .1, pi/2 + .1], pi/2, .5, 50, 0, 0) should give a plot similar to Figure 2.

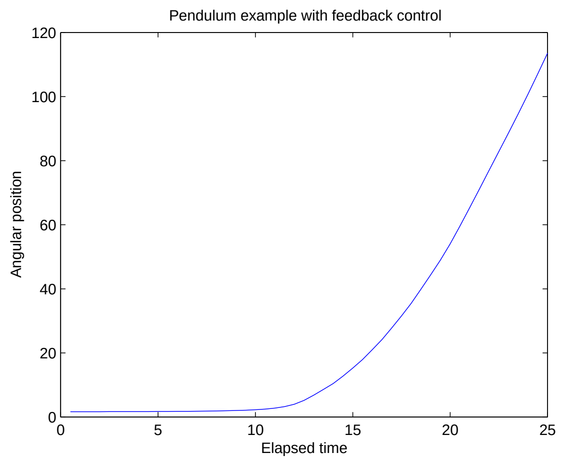

Figure 2: Orbit starting from $\mathbf{x}_0 = (\frac{\pi}{2} + .1, \frac{\pi}{2} + .1)$ of the unforced system $\mathbf{x}_{k+1} = \tilde{f}(\mathbf{x}_k, 0)$.

1.2 Making the equilibrium stable

After having done Task 1.3, we have understood that our desired equilibrium is not a stable equilibrium of the unforced system. Let's use feedback control to stabilize it. The idea is to choose $\tilde{u}_k$ above so that $\mathbf{x}_{\mathrm{eq}}$ is stable. To do that, we use a term proportional to the error,

$$\tilde{u}_k = K_1(\theta_{\mathrm{eq}} - z_k) + K_2(\theta_{\mathrm{eq}} - y_k)$$

The examples we have seen so far were only one-dimensional. Instead, this pendulum example is two-dimensional. That's the reason why we have to include both states in the design of our feedback controller. Substituting into equation (4), we get

$$\begin{cases} z_{k+1} = 2z_k - y_k - \Delta^2 \sin z_k + \Delta^2 \left[u_{\mathrm{eq}} + K_1(\theta_{\mathrm{eq}} - z_k) + K_2(\theta_{\mathrm{eq}} - y_k)\right] \\ y_{k+1} = z_k \end{cases} \tag{5}$$

The idea to make $\mathbf{x}_{\mathrm{eq}}$ stable is the same as in previous examples: choose K_1 and K_2 in the appropriate way. Now, in order to do that mathematically, you would need to know about linear algebra, matrices and eigenvalues. Since that is advanced math stuff, we are not going to go into it. Instead, we will choose K_1 and K_2 "by hand", and check in MATLAB® that our selection is correct.

Task 1.4 Choose your preferred pendulum configuration θ_{des} (e.g., you might choose $\theta_{\mathrm{des}} = \pi$ for the inverted pendulum position, or $\theta_{\mathrm{des}} = \pi/2$ for the funny horizontal configuration). Find values for K_1 and K_2 that make the desired equilibrium stable. You can rely on the MATLAB® program `pendulum.m` to do the check. With appropriate choices of K_1 and K_2 you should get plots that look like Figure 3.

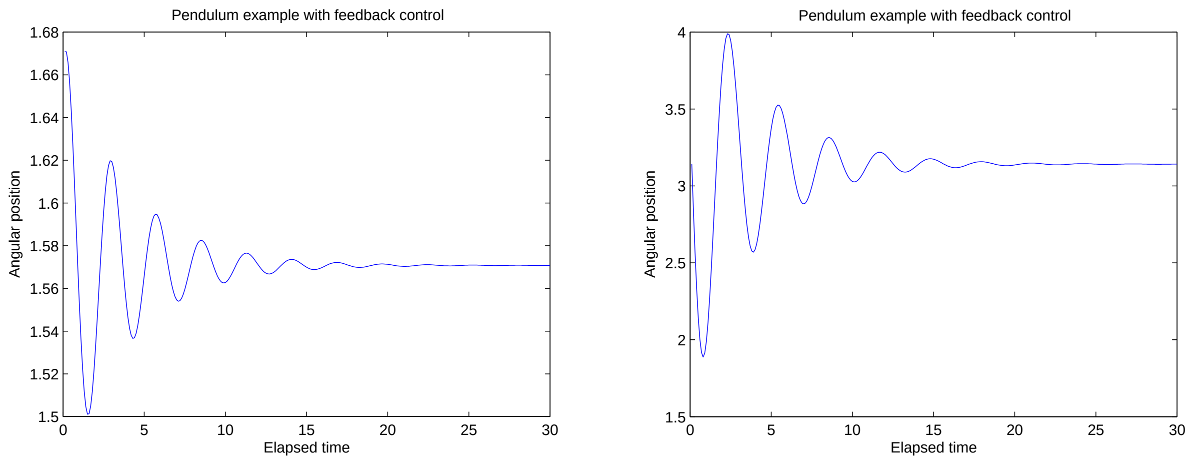

Figure 3: Orbits starting from $\mathbf{x}_0 = (\frac{\pi}{2} + .1, \frac{\pi}{2} + .1)$ (left) and $\mathbf{x}_0 = (\pi - .3, \pi - .25)$ (right) of the system $\mathbf{x}_{k+1} = \tilde{f}(\mathbf{x}_k, \tilde{u}_k)$, with $\tilde{u}_k = K_1(\theta_{\mathrm{eq}} - z_k) + K_2(\theta_{\mathrm{eq}} - y_k)$, where $\theta_{\mathrm{eq}} = \pi/2$ (left) and $\theta_{\mathrm{eq}} = \pi$ (right), respectively.

2 Influence of model errors: PI controllers

If there are model errors or disturbances the actual equilibrium may deviate from its desired value.

Example 2.1 (Cruise control) In the cruise control example of Lecture 1, given a desired speed v_{des}, we used a proportional control

$$u_{eng,k} = K(v_{\text{des}} - v_k) \tag{6}$$

in Bob's model:

$$v_{k+1} = v_k + \frac{\Delta}{m}[-bv_k + u_{eng,k} + u_{hill}]. \tag{7}$$

When we computed the steady-state equilibrium v_{ss}, we got as a result

$$v_{\text{ss}} = \frac{K}{b+K}v_{\text{des}} + \frac{1}{b+K}u_{\text{hill}}$$

Therefore, the perturbation caused by the slope of the hill causes the steady-state of the system v_{ss} to deviate from the desired value v_{des}.

This can be overcome by means of a PI (P for proportional and I for integral) controller. This name comes from the fact that the control signal is the sum of two terms, one proportional to the error and the other proportional to the discrete integral of the error. Let's see how this works in the cruise controller example.

Instead of (6), consider the following controller:

$$u_{eng,k} = K(v_{\text{des}} - v_k) + \sum_{i=1}^{k} \Delta(v_{\text{des}} - v_i) \tag{8}$$

The term we have added to the controller (6) is an approximation to the integral of the error across time. You probably don't know anything about integrals yet, but you soon will! For now, let's just say that there is a reason behind our choice.

Let's substitute (8) into Bob's model (7) to get

$$v_{k+1} = v_k + \frac{\Delta}{m}[-bv_k + K(v_{\text{des}} - v_k) + \sum_{i=1}^{k} \Delta(v_{\text{des}} - v_i) + u_{hill}].$$

Let us now compute the value of the steady-state velocity. Assume the system is in steady state v_{ss}, that is, the orbit starting at v_{ss} is simply $(v_{\text{ss}}, v_{\text{ss}}, v_{\text{ss}}, \dots)$. Then the previous equation looks like

$$v_{\text{ss}} = v_{\text{ss}} + \frac{\Delta}{m}[-bv_{\text{ss}} + K(v_{\text{des}} - v_{\text{ss}}) + \sum_{i=1}^{k} \Delta(v_{\text{des}} - v_{\text{ss}}) + u_{hill}].$$

Simplifying, we get

$$0 = -bv_{\text{ss}} + K(v_{\text{des}} - v_{\text{ss}}) + k\,\Delta(v_{\text{des}} - v_{\text{ss}}) + u_{hill},$$

which can also be rewritten

$$(b+K)v_{\mathrm{ss}} - Kv_{\mathrm{des}} - u_{hill} = k\,\Delta(v_{\mathrm{des}} - v_{\mathrm{ss}}), \quad k = 0, 1, 2, 3, \ldots$$

The right-hand side of this equation has to be necessarily zero.

Task 2.2 Why?

The right-hand side of the equation being zero, that means that $v_{\mathrm{ss}} = v_{\mathrm{des}}$, which is what we were looking for. Therefore, the idea of the PI controller is to drive the steady state error to be zero by applying an increasingly large input when $v_{\mathrm{ss}} \neq v_{\mathrm{des}}$ (through the integral term).

Task 2.3 Let's check the usefulness of PI controllers. Take the MATLAB® program that you designed in Lecture 6 to simulate the cruise control example on a flat road (i.e., $u_{hill} = 0$). We found that $u_{eng,des} = bv_{\mathrm{des}}$ makes v_{des} an equilibrium. Do the following

(i) Choose $\tilde{u}_{eng,k}$ in $u_{eng,k} = u_{eng,des} + \tilde{u}_{eng,k}$ that makes $v_{\mathrm{des}} = 55$ an stable equilibrium point on a flat road.

(ii) Plot the orbit of the system when we are climbing a hill with the car of slope 1 ($u_{hill} = 1$). Are we going at our desired speed? Why?

(iii) Design a PI controller that will make the cruise controller robust to perturbations like the hill slope. Plot the orbit of the system when we are climbing a hill with the car of slope 1 ($u_{hill} = 1$). Are we going at our desired speed now? Why?