1.4 State-Space representations

State-space representations are at the basis of modern control theory, and many control concepts such as stability, controllability and observability are defined using state-space representations. As we will see, the power of these representations lies in their ability to describe a wide range of systems, including nonlinear systems and systems with multiple inputs and multiple outputs (similarly to the examples of the beginning of this chapter). State-space representations are also very useful when it comes to demonstrating important dynamical properties, mostly coming from the relative simple way state-space representations are defined.

1.4.1 The State-Vector and the State-Space

Introduce first a time-dependent vector $\mathbf{x}(t)$ taking values in $\mathbb{R}^n$ (mathematically, write $\mathbf{x}(t) \in \mathbb{R}^n$). We call this vector the *state vector* or simply state, while the space $\mathbb{R}^n$ is called the *State-Space*. In state-space representations, ie centered around the state vector and the state-space concepts, a particularity is that the knowledge of the state vector $\mathbf{x}$ of a considered system at instant t_0 is enough to know the future evolution of this system for all $t \geq t_0$.

Example: A simple discrete-time system

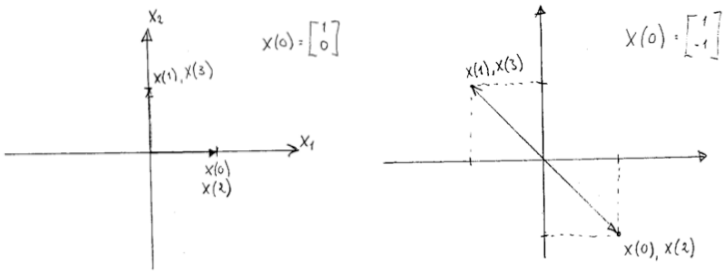
Consider a system described by the following set of difference equations:

$$\begin{cases} x_1(t+1) = x_2(t), & x_1(0) = 1 \\ x_2(t+1) = x_1(t), & x_2(0) = 0 \end{cases} \tag{1.15}$$

In the above system, the state vector is the vector of dimension 2 defined by

$$\mathbf{x}(t) := \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \tag{1.16}$$

So that the state space is $\mathbb{R}^2$. Let us now draw a simple 2-dimensional graphical representation of the evolution of this state at time $t = 0, 1, 2, 3$. For initial conditions set as above (ie $\mathbf{x}(0) = [1, 0]^T$), this evolution can be seen in figure 1.18a:



(a) State evolution for system (1.15).

(b) Initial conditions changed to $\mathbf{x}(0) = [1, -1]^T$

Figure 1.18: Graphical representation of the evolution of state vector $\mathbf{x}(t)$

If we now modify only the initial conditions of system (1.15) to $\mathbf{x}(0) = [1, -1]^T$, it is easy to obtain the new values of the state for different subsequent times, as represented in figure 1.18b.
This means that knowing $\mathbf{x}$ at time $t = t_0 = 0$ is enough to know the evolution of $\mathbf{x}(t)$ for all $t \geq 0$. Hence the vector/variable we have chosen for system (1.15) is indeed a valid state vector. □

Generally speaking, state-space representations are described by a system of *first-order* difference or differential equations with the state-vector as the delayed or differentiated variable. This is obviously the case in the previous example.
Let us see now another example.

Example: Fibonacci sequence

Consider the system described by the difference equation:

$$F(t+2) = F(t+1) + F(t) \tag{1.17}$$

The order of this system is 2, so we need 2 initial conditions which are set to

$$F(1) = 1, \qquad F(0) = 0.$$

The first observation one can make is that since system (1.17) is of order 2, then it is *not* a state-space representation.
Let us now consider the variable $F(t)$. Could this be a valid state vector? If that were the case, then this would mean that knowning $F(0)$ would be sufficient to obtain F for any future t, which is not the case since we also need $F(1)$! Hence dynamic variable $F(t)$ is not a state vector. □

1.4.2 State-Space representations of Linear Time Invariant (LTI) Systems

We would like to extend the above ideas to have a linear representation containing a dynamic part responsible for the evolution of the state, and also allows for the addition of external control signals.
Hence we have the two following linear state-space representations given below, one for continuous-time systems, the other for discrete-time ones.

Continuous-time: Linear State-Space Representation

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \qquad \mathbf{x}(0) = \mathbf{x}_0 \tag{1.18}$$

where the vectors and matrices have the dimensions

$$\mathbf{x} \in \mathbb{R}^n, \qquad \mathbf{A} \in \mathbb{R}^{n \times n}, \qquad \mathbf{u} \in \mathbb{R}^m, \qquad \mathbf{B} \in \mathbb{R}^{n \times m}$$

and where $\mathbf{u}(t)$ is called the *control input vector*.

Discrete-time: Linear State-Space Representation

$$\mathbf{x}(t+1) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \qquad \mathbf{x}(0) = \mathbf{x}_0 \tag{1.19}$$

again with

$$\mathbf{x} \in \mathbb{R}^n, \qquad \mathbf{A} \in \mathbb{R}^{n\times n}, \qquad \mathbf{u} \in \mathbb{R}^m, \qquad \mathbf{B} \in \mathbb{R}^{n\times m}$$

Some systems of differential equations are directly or almost directly put into a state-space representation. Let us see that with the example below.

Example: The Compartment model as a linear state-space representation

Let take again the system represented by

$$\begin{cases} V_1\dot{c}_1(t) = q\left(c_2(t) - c_1(t)\right) - q_0 c_1(t) + c_0 u(t) \\ V_2\dot{c}_2(t) = q\left(c_1(t) - c_2(t)\right) \end{cases} \tag{1.20}$$

If we rewrite this dynamical system as

$$\begin{cases} \dot{c}_1(t) = \dfrac{q}{V_1}\left(c_2(t) - c_1(t)\right) - \dfrac{q_0}{V_1}c_1(t) + \dfrac{c_0}{V_1}u(t) \\ \dot{c}_2(t) = \dfrac{q}{V_2}\left(c_1(t) - c_2(t)\right) \end{cases} \tag{1.21}$$

then we are not very far from a state-space representation (not far at all actually). Indeed, defining the state as

$$\mathbf{x}(t) := \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} c_1(t) \\ c_2(t) \end{bmatrix}, \tag{1.22}$$

replacing the c_i's by the corresponding x_i's, and putting everything in vectorial form, we get

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} -\dfrac{q}{V_1} - \dfrac{q_0}{V_1} & \dfrac{q}{V_1} \\ \dfrac{q}{V_2} & -\dfrac{q}{V_2} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} \dfrac{c_0}{V_1} \\ 0 \end{bmatrix} u(t) \tag{1.23}$$

so that we have state-space representation (1.18) with

$$\mathbf{A} = \begin{bmatrix} -\dfrac{q}{V_1} - \dfrac{q_0}{V_1} & \dfrac{q}{V_1} \\ \dfrac{q}{V_2} & -\dfrac{q}{V_2} \end{bmatrix} \text{ and } \mathbf{B} = \begin{bmatrix} \dfrac{c_0}{V_1} \\ 0 \end{bmatrix}. \tag{1.24}$$

□

Coming back to general state-space representations, equations (1.18) and (1.19) are often called dynamic equations in a state-space representation context. In order to represent outputs of interest, for example parts of the state that are measured by sensors, it is common to add to the dynamic equation a so-called *output equation* so that we obtain:

$$\begin{cases} \dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), & \mathbf{x}(0) = \mathbf{x}_0 \\ \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \end{cases} \tag{1.25}$$

where

$$\mathbf{y} \in \mathbb{R}^p, \qquad \mathbf{C} \in \mathbb{R}^{p\times n}, \qquad \mathbf{D} \in \mathbb{R}^{p\times m}$$

Example: Compartment continued

Coming back to our previous example, we assume now we measure only $c_2(t)$ or equivalently $x_2(t)$, which means that we have $y(t) = c_2(t) = x_2(t)$[1]. In vectorial form, this gives

$$y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} u(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}u(t) \tag{1.26}$$

□

1.4.3 Nonlinear state-space representations

In the nonlinear case, matrices of the linear state-space representations are replaced by a *vector field* or multi-dimensional function.
Thus, if the system is without control input, a state-space representation will have the following form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)), \qquad \mathbf{x}(0) = \mathbf{x}_0 \qquad \text{(continous-time)} \tag{1.27}$$

or, for the discrete-time version,

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t)), \qquad \mathbf{x}(0) = \mathbf{x}_0 \qquad \text{(discrete-time)} \tag{1.28}$$

where $\mathbf{f}$ is a vector field taking values in $\mathbb{R}^n$, ie we have

$$\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n \tag{1.29}$$

Example: The Lorenz System

This system is already described in state-space form since all the differential equations are of first order:

$$\begin{cases} \dot{x}_1(t) = -px_1(t) + px_2(t) \\ \dot{x}_2(t) = -x_1(t)x_3(t) - x_2(t) \\ \dot{x}_3(t) = x_1(t)x_2(t) - x_3(t) \end{cases} \tag{1.30}$$

where p is a constant. Let us then define the state as

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} \tag{1.31}$$

so that $\mathbf{f} : \mathbb{R}^3 \to \mathbb{R}^3$:

$$\mathbf{f}(\mathbf{x}(t)) = \begin{bmatrix} -px_1(t) + px_2(t) \\ -x_1(t)x_3(t) - x_2(t) \\ x_1(t)x_2(t) - x_3(t) \end{bmatrix} = \begin{bmatrix} f_1(x_1(t), x_2(t), x_3(t)) \\ f_2(x_1(t), x_2(t), x_3(t)) \\ f_3(x_1(t), x_2(t), x_3(t)) \end{bmatrix} \tag{1.32}$$

When inputs are included into the representation, the vector field $\mathbf{f}$ also depends on $\mathbf{u}(t)$ and we have

$$\dot{\mathbf{x}}(t) = \mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t)\right) \tag{1.33}$$

[1]The output is here not in bold font because the output vector is of dimension one, ie there is only one output. Same for the input.

and similarly for discrete-time systems (ie we have $\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$). Similarly to the linear case, an output equation, also nonlinear, can be added:

$$\mathbf{y}(t) = \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t)) \tag{1.34}$$

where $\mathbf{h}$ is another vector field. □

1.5 Bridges

Mathematical bridges or transformations between systems that are described mathematically can be very useful, not only for finding interesting theoretical results, but also for more practical-related matters.
The first one which we will see below concerns the actual implmentation of our continuous-time systems on digital computers, ie how to discretize a system by a simple program written in C, Python or pure Matlab (ie no Simulink).

1.5.1 From continuous to discrete-time state-space representations: discretization using the Euler Method

The Euler method is probably amongst the crudest and simplest methods of discretizing an ODE. However, it is still very widely used because it can quite often be enough to give satisfactory results.

Principle: Approximate a first-order derivative by a difference in time.
Thus, discretizing

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{1.35}$$

gives, after discretization,

$$\frac{\mathbf{x}(t+1) - \mathbf{x}(t)}{T_s} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{1.36}$$

where we recall that, in the above, time variable t is now discrete. The constant T_s is the period at which the signal is discretized or sampled. Finally, isolating the terms in $(t+1)$, we have the following *discrete-time* state-space representation

$$\mathbf{x}(t+1) = \mathbf{x}(t) + T_s\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) = \mathbf{f}'(\mathbf{x}(t), \mathbf{u}(t)) \tag{1.37}$$

Example: Discretizing the Lorenz system

Recall a description of the Lorenz chaotic system

$$\begin{cases} \dot{x}_1(t) = -px_1(t) + px_2(t) \\ \dot{x}_2(t) = -x_1(t)x_3(t) - x_2(t) \\ \dot{x}_3(t) = x_1(t)x_2(t) - x_3(t) \end{cases} \tag{1.38}$$

Discretizing this system using the Euler method gives

$$\begin{cases} \dfrac{x_1(t+1)-x_1(t)}{T_s} = -px_1(t) + px_2(t) \\ \dfrac{x_2(t+1)-x_2(t)}{T_s} = -x_1(t)x_3(t) - x_2(t) \\ \dfrac{x_3(t+1)-x_3(t)}{T_s} = x_1(t)x_2(t) - x_3(t) \end{cases} \tag{1.39}$$

from which we can now obtain the following discrete-time state-space representation

$$\begin{cases} x_1(t+1) = x_1(t) + T_s\left[-px_1(t) + px_2(t)\right] \\ x_2(t+1) = x_2(t) + T_s\left[-x_1(t)x_3(t) - x_2(t)\right] \\ x_3(t+1) = x_3(t) + T_s\left[x_1(t)x_2(t) - x_3(t)\right] \end{cases} \tag{1.40}$$

□

It is quite important to note that choosing the sampling period obviously depends on the system dynamics, otherwise the first-order approximation might be too crude. A fast system will generally need a small T_s while a slow one will do with a larger sampling period.

1.5.2 From ODEs to state-space representations

Obviously, ODEs can be of any order, while state-space representations are exclusively first-order forms. While state-space representations are widely used in control systems, ODEs of higher-order come naturally when modeling many physical systems (especially mechanical systems, ie many of second order).

If we are to control them, we should find a way to obtain a state-space representation out of systems modelled by ODEs. We give below a few tips on how to do that (note: in this section, we consider only systems with 1 input and 1 output).

Consider the following differential equation

$$y^{(n)}(t) + a_{n-1}y^{(n-1)}(t) + ... + a_1\dot{y}(t) + a_0y(t) = bu(t) \tag{1.41}$$

which is a linear equation of order n, where the coefficients a_i are constant.
Main idea: Define a state vector $x(t)$ that contains $y(t)$ and the derivatives of $y(t)$, and whose dimension is the same as the order of the above system, ie n. This gives

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ \vdots \\ x_n(t) \end{bmatrix} := \begin{bmatrix} y(t) \\ \dot{y}(t) \\ \ddot{y}(t) \\ \vdots \\ y^{(n-1)}(t) \end{bmatrix} \tag{1.42}$$

Since a state-space representation relates $\dot{\mathbf{x}}(t)$ to $\mathbf{x}(t)$, we first compute the

derivative of $\mathbf{x}(t)$, and obtain

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{y}(t) \\ \ddot{y}(t) \\ \vdots \\ y^{(n)}(t) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ x_3(t) \\ \vdots \\ y^{(n)}(t) = \dot{x}_n(t) \end{bmatrix} \tag{1.43}$$

Then, by isolating the highest-order derivative (1.41), i.e. by computing

$$y^{(n)}(t) = -a_0 y(t) - a_1 \dot{y}(t) - ... - a_{n-1} y^{(n-1)}(t) + b.u(t) \tag{1.44}$$

we get the set of first-order differential equations, put in component form

$$\begin{cases} \dot{x}_1(t) = x_2(t) \\ \dot{x}_2(t) = x_3(t) \\ \vdots \\ \dot{x}_n(t) = -a_0 x_1(t) - a_1 x_2(t) - ... - a_{n-1} x_n(t) + b.u(t) \end{cases} \tag{1.45}$$

which can then be put under a vectorial/matrix form

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \\ \vdots \\ \dot{x}_{n-1}(t) \\ \dot{x}_n(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & ... & 0 \\ 0 & 0 & 1 & 0 & ... & 0 \\ 0 & 0 & 0 & 1 & ... & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & ... & 1 \\ -a_0 & -a_1 & -a_2 & -a_3 & ... & -a_{n-1} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ \vdots \\ x_{n-1}(t) \\ x_n(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ b \end{bmatrix} u(t) \tag{1.46}$$

so that we have the following $\mathbf{A}$ and $\mathbf{B}$ matrices

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & ... & 0 \\ 0 & 0 & 1 & 0 & ... & 0 \\ 0 & 0 & 0 & 1 & ... & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & ... & 1 \\ -a_0 & -a_1 & -a_2 & -a_3 & ... & -a_{n-1} \end{bmatrix} \text{ and } \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ b \end{bmatrix} \tag{1.47}$$

As for the output equation, note that the definition of the state (1.42) also implies that

$$y(t) = x_1(t) \tag{1.48}$$

which simply translates in having the matrix $\mathbf{C}$ written as

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix} \tag{1.49}$$

while we obviously have $\mathbf{D} = 0$.[2]

[2]The above form of a state-space representation, with matrices $\mathbf{A}$ and $\mathbf{B}$ having this special structure shown in (1.47), in particular for matrix $\mathbf{A}$, with a diagonal of ones and all the a_i coefficients on the last line of the matrix, is often referred to as *Controllability Canonical Form*.

Example: Mass-Spring-Damper system

Consider the following mass-spring-damper system

$$m\ddot{y}(t) + d\dot{y}(t) + ky(t) = u(t) \tag{1.50}$$

We would like to find a state-space representation for this system.

Start by defining the state as

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} := \begin{bmatrix} y(t) \\ \dot{y}(t) \end{bmatrix} \Rightarrow \dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{y}(t) \\ \ddot{y}(t) \end{bmatrix} = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} \tag{1.51}$$

and isolate the highest-order derivative

$$\ddot{y}(t) = -\frac{d}{m}\dot{y}(t) - \frac{k}{m}y(t) + \frac{1}{m}u(t)\,, \tag{1.52}$$

which, using definition (1.51), gives

$$\dot{x}_1(t) = x_2(t) \tag{1.53}$$

and

$$\dot{x}_2(t) = -\frac{d}{m}x_2(t) - \frac{k}{m}x_1(t) + \frac{1}{m}u(t)\,, \tag{1.54}$$

so that we have the state-space representation

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{d}{m} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u(t) \tag{1.55}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \tag{1.56}$$

□

Important remark: At this point, it is important to insist on the fact that the definition of the state is something which is *decided upon*.
Hence one can have many different state-space representations corresponding to a single system, as there are many possible ways to define the state. Let us see that in the next example.

Example: Second-order system

Consider the system described below

$$\ddot{y}(t) + a_1\dot{y}(t) + a_0 y(t) = bu(t) \tag{1.57}$$

and let this time the state vector be defined by

$$x_1(t) := \frac{1}{b}y(t), \qquad x_2(t) := \frac{d}{dt}\left(\frac{1}{b}y(t)\right) = \frac{1}{b}\dot{y}(t) \tag{1.58}$$

Hence, using (1.58) we obtain the derivative of the state

$$\dot{x}_1(t) = x_2(t), \qquad \dot{x}_2(t) = \frac{1}{b}\ddot{y}(t) \tag{1.59}$$

and putting (1.59) into (1.57), we get

$$\frac{1}{b}\ddot{y}(t) = -\frac{a_1}{b}\dot{y}(t) - \frac{a_0}{b}y(t) + u(t) \tag{1.60}$$

so that replacing with our state definition, we get

$$\dot{x}_2(t) = -a_1 x_2(t) - a_0 x_1(t) + u(t). \tag{1.61}$$

We then have the following state-space representation:

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{bmatrix} = \begin{bmatrix}0 & 1\\ -a_0 & -a_1\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix} + \begin{bmatrix}0\\ 1\end{bmatrix}u(t) \tag{1.62}$$

$$y(t) = \begin{bmatrix}b & 0\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix}$$

which is different from a state-space representation obtained by our usual definition of the state, ie we would have

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{bmatrix} = \begin{bmatrix}0 & 1\\ -a_0 & -a_1\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix} + \begin{bmatrix}0\\ b\end{bmatrix}u(t) \tag{1.63}$$

$$y(t) = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix}$$

Thus, we can see that two different state-space representations can be obtained starting with the same dynamical system. □

Derivatives of the input

State-space representations can also be obtained for linear equations containing derivatives of the input signal, ie starting with

$$\begin{aligned} &y^{(n)}(t) + a_{n-1}y^{(n-1)}(t) + ... + a_1\dot{y}(t) + a_0 y(t)\\ &= b_{n-1}u^{(n-1)}(t) + b_{n-2}u^{(n-2)}(t) + \cdots + b_1\dot{u}(t) + b_0 u(t), \end{aligned} \tag{1.64}$$

a corresponding state-space representation reads

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\\ \dot{x}_3(t)\\ \vdots\\ \dot{x}_{n-1}(t)\\ \dot{x}_n(t)\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0 & ... & 0\\ 0 & 0 & 1 & 0 & ... & 0\\ 0 & 0 & 0 & 1 & ... & 0\\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & 0 & ... & 1\\ -a_0 & -a_1 & -a_2 & -a_3 & ... & -a_{n-1}\end{bmatrix}\mathbf{x}(t) + \begin{bmatrix}0\\ 0\\ 0\\ \vdots\\ 0\\ 1\end{bmatrix}u(t) \tag{1.65}$$

and

$$y(t) = \begin{bmatrix}b_0 & b_1 & b_2 & \cdots & b_{n-1}\end{bmatrix}\mathbf{x}(t) \tag{1.66}$$

for the output equation. Note that, if this state-space representation is very similar to the one we obtained without derivatives in the input, the definition of the state is generally a bit more involved than the simple $\mathbf{x}(t) := [y(t), \dot{y}(t), \cdots, y^{(n-1)}(t)]^T$.
However, seeing how the above state-space representation is obtained is not so complicated if one uses a simple trick. Let us see that on a second-order example.

Example: Second-order system with input derivative

Start with

$$\ddot{y}(t) + a_1\dot{y}(t) + a_0 y(t) = b_0 u(t) + b_1\dot{u}(t). \tag{1.67}$$

In order to find the state-space representation of (1.67), let us first consider a modified version of (1.67), but without the input derivatives, ie we have

$$\ddot{y}_m(t) + a_1\dot{y}_m(t) + a_0 y_m(t) = u(t) \tag{1.68}$$

which also implies that

$$\dddot{y}_m(t) + a_1\ddot{y}_m(t) + a_0\dot{y}_m(t) = \dot{u}(t). \tag{1.69}$$

A state-space representation can be found similarly as we have done at the beginning of this section so that we have

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{bmatrix} = \begin{bmatrix}0 & 1\\ -a_0 & -a_1\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix} + \begin{bmatrix}0\\ 1\end{bmatrix} u(t) \tag{1.70}$$

with

$$y_m(t) = x_1(t) \text{ and } \dot{y}_m(t) = \dot{x}_1(t) = x_2(t). \tag{1.71}$$

Then, let the actual output of system (1.67) be defined by

$$y(t) = b_0 y_m(t) + b_1\dot{y}_m(t). \tag{1.72}$$

Then, since system (1.69) is a second-order system, differentiate expression (1.72) twice to obtain

$$\ddot{y}(t) = b_0\ddot{y}_m(t) + b_1\dddot{y}_m(t). \tag{1.73}$$

The above expression, using (1.68) and (1.69) can be rewritten and progressively re-arranged so that we finally get

$$\begin{aligned}\ddot{y} &= b_0\left(-a_1\ddot{y}_m(t) - a_0\dot{y}_m(t) + \dot{u}(t)\right) + b_1\left(-a_1\ddot{y}_m(t) - a_0\dot{y}_m(t) + \dot{u}(t)\right)\\ &= -a_1\left(b_0\dot{y}_m(t) + b_1\ddot{y}_m(t)\right) - a_0\left(b_0 y_m(t) + b_1\dot{y}_m(t)\right) + b_0 u(t) + b_1\dot{u}(t)\\ &= -a_1\dot{y}(t) - a_0 y(t) + b_0 u(t) + b_1\dot{u}(t)\end{aligned}$$

Hence, system (1.67) corresponds to state-space representation (1.70) with output equation (using (1.72))

$$y(t) = b_0 x_1(t) + b_1 x_2(t) = \begin{bmatrix}b_0 & b_1\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix} \tag{1.74}$$

□

Discrete-time systems

The reasoning is almost the same for discrete-time systems. Indeed, in this case, consider instead the difference equation of order n

$$y(t+n) + a_{n-1}y(t+n-1) + ... + a_1 y(t+1) + a_0 y(t) = bu(t) \tag{1.75}$$

Define then the state vector as

$$\mathbf{x}(t) := \begin{bmatrix} y(t) \\ y(t+1) \\ y(t+2) \\ \vdots \\ y(t+(n-1)) \end{bmatrix}$$

and calculate the state vector at time $t+1$, ie

$$\mathbf{x}(t+1) = \begin{bmatrix} y(t+1) \\ y(t+2) \\ y(t+3) \\ \vdots \\ y(t+n) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ x_3(t) \\ x_4(t) \\ \vdots \\ -a_0x_1(t) - a_1x_2(t) ... - a_{n-1}x_n(t) \end{bmatrix} \tag{1.76}$$

so that we have

$$\mathbf{x}(t+1) = \begin{bmatrix} 0 & 1 & 0 & 0 & ... & 0 \\ 0 & 0 & 1 & 0 & ... & 0 \\ 0 & 0 & 0 & 1 & ... & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & ... & 1 \\ -a_0 & -a_1 & -a_2 & -a_3 & ... & -a_{n-1} \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ b \end{bmatrix} u(t) \tag{1.77}$$

with the output equation

$$y = \begin{bmatrix} 1 & 0 & ... & 0 \end{bmatrix} \mathbf{x}(t) \tag{1.78}$$

Example: The Fibonacci Sequence

Let us go back to the Fibonacci sequence described by

$$y(t+2) = y(t+1) + y(t) \tag{1.79}$$

with the 2 inital conditions

$$y(1) = 1, \qquad y(0) = 0$$

To find a state-space representation, start with the definition of a state vector

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} := \begin{bmatrix} y(t) \\ y(t+1) \end{bmatrix} \Rightarrow \mathbf{x}(t+1) = \begin{bmatrix} y(t+1) \\ y(t+2) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ x_1(t) + x_2(t) \end{bmatrix} \tag{1.80}$$

which gives

$$\mathbf{x}(t+1) = \begin{bmatrix} x_1(t+1) \\ x_2(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \mathbf{x}(t) \tag{1.81}$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}, \qquad \mathbf{x}(0) = \begin{bmatrix} y(0) \\ y(1) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{1.82}$$

□

Nonlinear Systems

Generally, not all nonlinear ODEs can be converted to state-space representations, and this is a topic of current research.
For example, the nonlinear system

$$\ddot{y}(t) = y(t)\dot{u}^2(t) \tag{1.83}$$

cannot be transformed into a state-space representation. This system is then said to be *nonrealizable*.

However, in many cases, it is possible to obtain a state-space representation of a nonlinear system by using the same reasoning as for linear systems. Let us show this through the following example.

Example: state-space representation for the Pendulum/Robot elbow

Take again the pendulum equation

$$ml^2\ddot{\theta}(t) + mgl\sin\theta(t) = u(t), \qquad y(t) = \theta(t) \tag{1.84}$$

Isolating the highest-order equation

$$\ddot{\theta}(t) = -\frac{g}{l}\sin\theta(t) + \frac{1}{ml^2}u(t), \qquad y(t) = \theta(t) \tag{1.85}$$

and defining the state vector as

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} := \begin{bmatrix} \theta(t) \\ \dot{\theta}(t) \end{bmatrix} \tag{1.86}$$

we obtain the following state-space representation

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), u(t)) \text{ where } \mathbf{f}(\mathbf{x}(t), u(t)) = \begin{bmatrix} x_2(t) \\ -\frac{g}{l}\sin x_1(t) + \frac{1}{ml^2}u(t) \end{bmatrix} \tag{1.87}$$

□

1.5.3 From nonlinear systems to their linear approximation

As we will see, quite a few techniques used in autonomous systems (the Kalman Filter, PID, LQR, MPC) were initially defined in a linear setting, with good reason: linear systems are usually easier to tackle than nonlinear ones.
Interestingly, applying linear techniques to systems which are nonlinear is quite often possible. The basic idea consists in choosing a point of interest where one wishes to consider the nonlinear system and compute a linear approximation around this point of interest.

Equilibrium Points

In order to make the notion of "point of interest" a bit more precise, we will use the concept of equilibrium point. Let us start by giving a definition of an equilibrium point in the state-space context.

Definition: An equilibrium point or equilibrium, noted $\mathbf{x}^*$, is a point in the state-space, for which, if the state $\mathbf{x}(t)$ reaches $\mathbf{x}^*$, it will stay there forever (in a state of equilibrium). □

From the above definition, note, importantly, that while it is possible that the origin of the state-space is an equilibrium, it does not have to be so (see for example the illustration in figure 1.19 below). Second, and as we will see on an example, the definition implies that there can be *several* points, not just one.

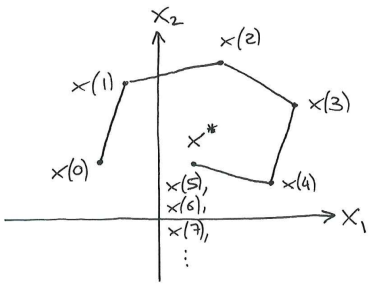


Figure 1.19: Trajectory and equilibrium point for a two-dimensional discrete-time system

It now remains to be able to determine/compute equilibrium point(s) for a given system. As it turns out, the above definition also helps us with that. Indeed, for continuous-time systems, an important implication of the definition can be written as

$$\frac{d}{dt}\mathbf{x}^* = 0 \tag{1.88}$$

which, if the considered system is described by $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$, gives

$$0 = \mathbf{f}(\mathbf{x}^*) \tag{1.89}$$

Hence, *a mathematical way to obtain the set of equilibrium points of a continuous-time system described by the nonlinear state-space representation* $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ *consists in finding all the solutions* $\mathbf{x}^*$ *of equation (1.89).*

Let us see that through the following example.

Example: Equilibrium points for the pendulum

Recall that we have

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} x_2(t) \\ -\dfrac{g}{l}\sin x_1(t) \end{bmatrix} \tag{1.90}$$

A first simple way to find the equilibrium points of this system is through using a simple physical reasoning. The system is in equilibrium means that it does not move anymore :-). Hence we have that the second component of $\mathbf{x}^*$, ie angular velocity x_2^* is zero. This corresponds to 2 equilibrium positions (modulo 2π), pendulum up and pendulum down. Thus we can easily conjecture 2 equilibrium points:

$$\mathbf{x}^* = \begin{bmatrix} x_1^* \\ x_2^* \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{x}^* = \begin{bmatrix} x_1^* \\ x_2^* \end{bmatrix} = \begin{bmatrix} \pi \\ 0 \end{bmatrix} \tag{1.91}$$

Let us now confirm that mathematically. Indeed, having equation (1.89) means that we need to solve

$$\begin{bmatrix} x_2^* \\ -\dfrac{g}{l}\sin x_1^* \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{1.92}$$

which indeed leads exactly to the 2 equilibrium points obtained in (1.91). □

At this point, it is worth mentioning that the dynamic behavior around (but not on) each equilibrium point can be very different whether the pendulum is up or down. This is indeed the case and we will discuss that when we look at stability considerations. For now, let us continue our discussion on equilibrium points.

For discrete-time systems like $\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$, the concept of derivative with respect to the time-variable does not apply. However, saying that x^* "will stay there forever" can easily be translated mathematically by the expression

$$\mathbf{x}^* = \mathbf{f}(\mathbf{x}^*) \tag{1.93}$$

which we need to solve in order to find the set of equilibrium points.

The systems we have considered now did not have any input signal... Let us first see what happens when a torque input is added to our pendulum example.

Example: Pendulum with control input

Consider now our pendulum with a torque applied to it. The state-space model is now

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} x_2(t) \\ -\dfrac{g}{l}\sin x_1(t) + \dfrac{1}{ml^2}u(t) \end{bmatrix} \tag{1.94}$$

where the control input signal $u(t)$ is the torque applied to the pendulum. Let us now start with a basic consideration given by the physics of the system. Since a specific constant torque can maintain the pendulum in a particular fixed angle/position, then the input signal plays an important role in the definition of equilibrium points.
Let us see that mathematically. Setting the vector field to zero leads to expression

$$\begin{bmatrix} x_2^* \\ -\dfrac{g}{l}\sin x_1^* + \dfrac{1}{ml^2}u^* \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{1.95}$$

where u^* stands for the control input *associated* to the equilibrium $\mathbf{x}^*$, ie the value of the control input for which we have an equilibrium. While the first line

of (1.95) means that we have $x_2^* = 0$, rewriting the second line of this expression gives

$$0 = -\frac{g}{l}\sin x_1^* + \frac{1}{ml^2}u^* \tag{1.96}$$

where we see that we can choose *any* value of x_1^* as long as we find the corresponding u^*!
For example, assume that we would like to maintain the pendulum on position $x_1^* = \pi/4$. Then we have the associated control input

$$u^* = mgl\sin\frac{\pi}{4} \tag{1.97}$$

Finally, for system (1.94), we have the set of equilibrium points given by

$$\mathbf{x}^* = \begin{bmatrix} x_1^* \\ 0 \end{bmatrix} \tag{1.98}$$

and its associated control input

$$u^* = mgl\sin x_1^* \tag{1.99}$$

□

Hence, for systems with a control input, the set of equilibrium points have to be considered in conjunction with their associated control input vector $\mathbf{u}^*$, ie in continuous-time and for systems of the form $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$, we need to find the solutions $\mathbf{x}^*$, $\mathbf{u}^*$ of the algebraic equation

$$0 = \mathbf{f}(\mathbf{x}^*, \mathbf{u}^*)\,, \tag{1.100}$$

while for a discrete-time system with input vector $\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$, we need to solve

$$\mathbf{x}^* = \mathbf{f}(\mathbf{x}^*, \mathbf{u}^*)\,. \tag{1.101}$$

In the simpler case where the considered state-space representation is linear, ie as exemplified by the continuous-time case $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$, we simply have the *linear* algebraic equation

$$0 = \mathbf{A}\mathbf{x}^* + \mathbf{B}\mathbf{u}^*. \tag{1.102}$$

Let us use our now infamous Mass-Spring-Damper example to illustrate this last point.

Example: Equilibrium points for the Mass-Spring-Damper system

Recalling the state-space representation of an MSD system given by (1.55), we have the equilibrium relation

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\dfrac{k}{m} & -\dfrac{d}{m} \end{bmatrix} \begin{bmatrix} x_1^* \\ x_2^* \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{1}{m} \end{bmatrix} u^* \tag{1.103}$$

which, after a very simple calculation, gives

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} x_2^* \\ -kx_1^* + u^* \end{bmatrix} \tag{1.104}$$

which means that there is an infinity of equilibrium points corresponding to all positions of the mass with zero velocity, ie $\mathbf{x}^{*T} = [x_1^*, 0]^T$, as long as the associated control input is $u^* = kx_1^*$. A physical interpretation of this result, remembering that u is the force applied to the mass, is that, in order to maintain the mass at a specific position x_1^*, we need to apply to the mass a force related to how strong the spring with stiffness coefficient k is. □

Linear approximation of a nonlinear system

Similarly to a smooth curve being approximated by a straight line in a neighborhood around a point, a nonlinear system can be locally and approximately described by a linear system around an equilibrium point $\mathbf{x}^*$.
Let us see how such a linear approximation can be obtained, first on a system without input. Hence, starting with the nonlinear system dynamics

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}\left(\mathbf{x}(t)\right) \tag{1.105}$$

we would like to find the linear dynamics corresponding to *small deviations around an equilibrium point*, ie we define

$$\delta\mathbf{x}(t) := \mathbf{x}(t) - \mathbf{x}^* \tag{1.106}$$

which makes it possible to rewrite nonlinear system (1.105) as

$$\frac{d}{dt}\left(\mathbf{x}^* + \delta\mathbf{x}(t)\right) = \mathbf{f}\left(\mathbf{x}^* + \delta\mathbf{x}(t)\right) \tag{1.107}$$

Let us now recall a basic mathematical result on the use of Taylor series.

First-order approximation using Taylor series:

Around $\mathbf{x}^*$, an approximation of function $\mathbf{f}$ can be written as

$$\mathbf{f}\left(\mathbf{x}^* + \delta\mathbf{x}(t)\right) \approx \mathbf{f}(\mathbf{x}^*) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x})\right|_{\mathbf{x}=\mathbf{x}^*} .\delta\mathbf{x}(t) \tag{1.108}$$

where, if we have the vector field given by

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(x_1, x_2, ..., x_n) \\ f_2(x_1, x_2, ..., x_n) \\ \vdots \\ f_n(x_1, x_2, ..., x_n) \end{bmatrix}, \tag{1.109}$$

the expression $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ is the so-called *Jacobian matrix* written as

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}) & \frac{\partial f_1}{\partial x_2}(\mathbf{x}) & \dots & \frac{\partial f_1}{\partial x_n}(\mathbf{x}) \\ \frac{\partial f_2}{\partial x_1}(\mathbf{x}) & \frac{\partial f_2}{\partial x_2}(\mathbf{x}) & \dots & \frac{\partial f_2}{\partial x_n}(\mathbf{x}) \\ \vdots & \vdots & \dots & \vdots \\ \frac{\partial f_n}{\partial x_1}(\mathbf{x}) & \frac{\partial f_n}{\partial x_2}(\mathbf{x}) & \dots & \frac{\partial f_n}{\partial x_n}(\mathbf{x}) \end{bmatrix} \tag{1.110}$$

□

Combine now (1.107) and (1.108) to get

$$\frac{d}{dt}\left(\mathbf{x}^* + \delta\mathbf{x}(t)\right) \approx \mathbf{f}(\mathbf{x}^*) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x})\right|_{\mathbf{x}=\mathbf{x}^*} .\delta\mathbf{x}(t) \tag{1.111}$$

Then, since $\mathbf{x}^*$ is an equilibrium point, we have $\frac{d}{dt}\mathbf{x}^* = 0$ and $\mathbf{f}(\mathbf{x}^*) = 0$, which finally gives us the dynamics

$$\frac{d}{dt}\left(\delta\mathbf{x}(t)\right) = \left[\left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x})\right|_{\mathbf{x}=\mathbf{x}^*}\right] \delta\mathbf{x}(t) \tag{1.112}$$

Note that the above system is linear, ie we have the form

$$\delta\dot{\mathbf{x}}(t) = \mathbf{A}\delta\mathbf{x}, \quad \text{with} \quad \mathbf{A} = \left[\left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x})\right|_{\mathbf{x}=\mathbf{x}^*}\right]. \tag{1.113}$$

Example: Linearized pendulum

We would like to have the linear approximation of the pendulum system around both equilibrium points $\mathbf{x}^* = [0, 0]^T$ and $\mathbf{x}^* = [\pi, 0]^T$. Recall that

$$\dot{\mathbf{x}} = \begin{bmatrix} x_2 \\ -\frac{g}{l}\sin x_1 \end{bmatrix} \Rightarrow \mathbf{f}(\mathbf{x}) = \begin{bmatrix} x_2 \\ -\frac{g}{l}\sin x_1 \end{bmatrix} \tag{1.114}$$

which gives the Jacobian

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}) = \begin{bmatrix} 0 & 1 \\ -\frac{g}{l}\cos x_1 & 0 \end{bmatrix} \tag{1.115}$$

Evaluating this Jacobian on the two equilibrium points gives

$$\left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x})\right|_{\mathbf{x}=\mathbf{x}^*=[0,0]^T} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{l} & 0 \end{bmatrix} \tag{1.116}$$

for $\mathbf{x}^* = [0, 0]^T$ and

$$\left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x})\right|_{\mathbf{x}=\mathbf{x}^*=[\pi,0]^T} = \begin{bmatrix} 0 & 1 \\ \frac{g}{l} & 0 \end{bmatrix} \tag{1.117}$$

for $\mathbf{x}^* = [\pi, 0]^T$. Hence, for both equilibrium points, we have the linear dynamics of the small variations

$$\delta\dot{\mathbf{x}} = \mathbf{A}\delta\mathbf{x} \tag{1.118}$$

but with a different matrix $\mathbf{A}$! Indeed, for equilibrium $\mathbf{x}^* = [0, 0]^T$ we have

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{l} & 0 \end{bmatrix} \tag{1.119}$$

while for $\mathbf{x}^* = [\pi, 0]^T$ we have

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ \frac{g}{l} & 0 \end{bmatrix} \tag{1.120}$$

□

The previous reasoning can easily be extended to systems with inputs represented by

$$\dot{\mathbf{x}}(t) = \mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t)\right) \tag{1.121}$$

In this case, we want to obtain a linear approximation of the system around $\mathbf{x}^*$ and its associated input $\mathbf{u}^*$. Hence we write

$$\frac{d}{dt}\left(\mathbf{x}^* + \delta\mathbf{x}(t)\right) = \mathbf{f}\left(\mathbf{x}^* + \delta\mathbf{x}(t), \mathbf{u}^* + \delta\mathbf{u}(t)\right) \tag{1.122}$$

so that the first-order approximation of $\mathbf{f}(\mathbf{x}, \mathbf{u})$ is now

$$\begin{aligned} f\left(\mathbf{x}^* + \delta\mathbf{x}(t), \mathbf{u}^* + \delta\mathbf{u}(t)\right) & \\ \approx \mathbf{f}(\mathbf{x}^*, \mathbf{u}^*) + & \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{u})\Big|_{\mathbf{x}=\mathbf{x}^*, \mathbf{u}=\mathbf{u}^*}\right].\delta\mathbf{x}(t) \\ + & \left[\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{x}, \mathbf{u})\Big|_{\mathbf{x}=\mathbf{x}^*, \mathbf{u}=\mathbf{u}^*}\right].\delta\mathbf{u}(t) \end{aligned} \tag{1.123}$$

where $\delta\mathbf{u}(t) := \mathbf{u}(t) - \mathbf{u}^*$. From there, and similarly to what we have done for systems without inputs, we end up with the following linear approximation

$$\frac{d}{dt}\delta\mathbf{x}(t) = \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{u})\Big|_{\mathbf{x}=\mathbf{x}^*, \mathbf{u}=\mathbf{u}^*}\right]\delta\mathbf{x}(t) + \left[\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{x}, \mathbf{u})\Big|_{\mathbf{x}=\mathbf{x}^*, \mathbf{u}=\mathbf{u}^*}\right]\delta\mathbf{u}(t) \tag{1.124}$$

that is we have the following state-space representation

$$\delta\dot{\mathbf{x}} = \mathbf{A}\delta\mathbf{x} + \mathbf{B}\delta\mathbf{u} \tag{1.125}$$

with

$$\mathbf{A} = \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{u})\Big|_{\mathbf{x}=\mathbf{x}^*, \mathbf{u}=\mathbf{u}^*}\right] \qquad \text{and} \qquad \mathbf{B} = \left[\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{x}, \mathbf{u})\Big|_{\mathbf{x}=\mathbf{x}^*, \mathbf{u}=\mathbf{u}^*}\right] \tag{1.126}$$

Example: Linearization of the robot "elbow"

We want to linearize again our controlled pendulum, but this time around the equilibrium points

$$\mathbf{x}^* = \begin{bmatrix} \pi/2 \\ 0 \end{bmatrix} \qquad \text{and} \qquad \mathbf{x}^* = \begin{bmatrix} \pi/4 \\ 0 \end{bmatrix} \tag{1.127}$$

Recalling that the controlled pendulum equations read

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} x_2(t) \\ -\frac{g}{l}\sin x_1(t) + \frac{1}{ml^2}u(t) \end{bmatrix} \tag{1.128}$$

it means that we have the two Jacobians

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}, u) = \begin{bmatrix} 0 & 1 \\ -\dfrac{g}{l}\cos x_1 & 0 \end{bmatrix} \qquad \text{and} \qquad \frac{\partial \mathbf{f}}{\partial u}(\mathbf{x}, u) = \begin{bmatrix} 0 \\ \dfrac{1}{ml^2} \end{bmatrix}. \tag{1.129}$$

Hence, for the first equilibrium point, we have

$$\mathbf{x}^* = \begin{bmatrix} \pi/2 \\ 0 \end{bmatrix} \Rightarrow \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x})\right|_{\mathbf{x}=\mathbf{x}^*, u=u^*} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{1.130}$$

so that the local dynamics around $\mathbf{x}^*$ are

$$\delta\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \delta\mathbf{x} + \begin{bmatrix} 0 \\ \dfrac{1}{ml^2} \end{bmatrix} \delta u \tag{1.131}$$

For the second equilibrium point, we have this time

$$\mathbf{x}^* = \begin{bmatrix} \pi/4 \\ 0 \end{bmatrix} \Rightarrow \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x})\right|_{\mathbf{x}=\mathbf{x}^*, u=u^*} = \begin{bmatrix} 0 & 1 \\ -\dfrac{\sqrt{2}}{2}\dfrac{g}{l} & 0 \end{bmatrix} \tag{1.132}$$

which gives the local linear dynamics

$$\delta\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ -\dfrac{\sqrt{2}}{2}\dfrac{g}{l} & 0 \end{bmatrix} \delta\mathbf{x} + \begin{bmatrix} 0 \\ \dfrac{1}{ml^2} \end{bmatrix} \delta u \tag{1.133}$$

□