Tank example

F. A. Ramponi
Revision 0.2, 2019-10-17
Example courtesy of Simone Garatti (PoliMI) and Marco C. Campi (UniBS)

0.1 Model of a tank

Suppose that a tank with constant section is continuously filled with a flow of water $u(t)$, adjustable by means of a valve. Let $x(t)$ be the level of water in the tank. At the bottom of the tank there is a pipe that lets water flow out of the system; for the sake of simplicity we ignore energy dissipation, nonlinear effects etc., and let the flow be proportional to the water level.

Here's the tank model:

$$\dot{x}(t) = -\bar{\alpha}x(t) + u(t) \tag{1}$$

where $\bar{\alpha} > 0$ is the pipe's output flow rate. The differential equation (1) comes naturally as a continuous-time dynamical system in state space form; since $-\bar{\alpha} < 0$, such system is obviously asymptotically stable. Here is the solution of (1), where $x(t_0) = x_0$ is the initial condition:

$$x(t) = e^{-\bar{\alpha}(t-t_0)}x_0 + \int_{t_0}^{t} e^{-\bar{\alpha}(t-\tau)}u(\tau)\ d\tau; \tag{2}$$

indeed $x(t)$ in (2) satisfies $x(t_0) = x_0$, and

$$\begin{aligned}
\dot{x}(t) &= \frac{d}{dt}e^{-\bar{\alpha}(t-t_0)}x_0 + \frac{d}{dt}\int_{t_0}^{t} e^{-\bar{\alpha}(t-\tau)}u(\tau)\ d\tau \\
&= -\bar{\alpha}\left(e^{-\bar{\alpha}(t-t_0)}x_0\right) + \int_{t_0}^{t} \frac{d}{dt}e^{-\bar{\alpha}(t-\tau)}u(\tau)\ d\tau + e^{-\bar{\alpha}(t-t)}u(t) \\
&= -\bar{\alpha}\left(e^{-\bar{\alpha}(t-t_0)}x_0 + \int_{t_0}^{t} e^{-\bar{\alpha}(t-\tau)}u(\tau)\ d\tau\right) + u(t) \\
&= -\bar{\alpha}x(t) + u(t).
\end{aligned}$$

To obtain a model suitable for adaptive control in discrete-time, we proceed to discretize (1) with sampling time $\Delta = 1$. Suppose that $u(t)$ is kept constant over intervals $[t, t+1)$; then

$$\begin{aligned}
x(t+1) &= e^{-\bar{\alpha}(t+1-t)}x(t) + \int_{t}^{t+1} e^{-\bar{\alpha}(t+1-\tau)}u(\tau)\ d\tau \\
&= e^{-\bar{\alpha}}x(t) + \left(\int_{t}^{t+1} e^{-\bar{\alpha}(t+1-\tau)}d\tau\right)u(t) \qquad (\text{let } s = t+1-\tau, ds = -d\tau) \\
&= e^{-\bar{\alpha}}x(t) + \left(\int_{1}^{0} e^{-\bar{\alpha}s}(-d\tau)\right)u(t) \\
&= e^{-\bar{\alpha}}x(t) + \left[\frac{-1}{\bar{\alpha}}e^{-\bar{\alpha}s}\right]_0^1 u(t) \\
&= e^{-\bar{\alpha}}x(t) + \frac{(1-e^{-\bar{\alpha}})}{\bar{\alpha}}u(t).
\end{aligned}$$

So here's the discretized model/system:

$$x(t+1) = e^{-\bar{\alpha}}x(t) + \frac{(1-e^{-\bar{\alpha}})}{\bar{\alpha}}u(t).$$

To simplify ideas and computations, we suppose from now on that $\bar{\alpha} = 1$ (but the take-home message would remain intact with a different $\bar{\alpha}$):

$$x(t+1) = \frac{1}{e}x(t) + \frac{e-1}{e}u(t). \tag{3}$$

0.2 Tracking with known parameter

Suppose that we want $x(t)$ to follow a reference signal $r(t)$ (typically a constant water level, $r(t) \equiv \bar{r}$). We assume that at least a future sample of the reference is always available (at time t, $r(t+1)$ is known). One may think that "knowing future samples" is the trademark of non-causality, and in control theory with non-causality we feel often uncomfortable; but please note that the reference signal $r(t)$ here is a *goal*, and it is perfectly legitimate to assume the knowledge of [some samples of] a future *goal*.

This said, substituting in (3) the control law

$$u(t) = \frac{e}{e-1}\left(-\frac{1}{e}x(t) + r(t+1)\right), \tag{4}$$

in closed loop we obtain $x(t+1) = r(t+1)$ for all $t \geq 0$, that is, *perfect tracking*.

0.3 Unknown parameter

The control law (4) in Section 0.2 is linear and very simple, yet it attains a suspiciously good (from an engineer's standpoint) performance. This happens for at least two reasons: first, because the model was oversimplified, for didactic purposes, neglecting all sorts of details that you will indeed face in your engineering career: assuming linearity for free, ignoring delays in the loop, assuming that the sampling time $\Delta = 1$ was good enough, assuming that $x(t)$ is measurable without errors, and so on.

The second reason, which happens to be the starting point of this course, is that *we have assumed perfect knowledge of the system parameters.*

Indeed, the system at hand has the form

$$x(t+1) = \bar{a}x(t) + \bar{b}u(t), \tag{5}$$

where $\bar{a} = e^{-\bar{\alpha}} = \frac{1}{e}$ and $\bar{b} = \frac{(1-e^{-\bar{\alpha}})}{\bar{\alpha}} = \frac{e-1}{e}$. The perfect knowledge of $\bar{a}$ and $\bar{b}$ allows to attain, at least in principle, a perfect performance; this is OK from the mathematical standpoint, but we are *engineers*: and when on earth did we happen to know *exactly* the "true" parameters of a physical system?

The control of a DC engine requires the knowledge of the moment of inertia of the rotor: were you ever able to know with 100% certainty, or compute exactly, or measure without errors, the moment of inertia of an *actual* complex object along a given axis? I bet that you did not; the same holds for the tank parameters and for the parameters of any *mathematical model* (as opposed to the "true" parameters of a physical system), and that's the main point of this course.

In engineering and applied sciences perfect knowledge is *never* the case. Even if we have a good guess of a nominal parameter $\theta = (a, b)$, we must accept that such nominal parameter can be, and in general is, different from the actual, "true" parameter $\bar{\theta}$. And therefore, we are forced to deal not with a single parameter $\bar{\theta}$, but with an entire set $\Theta = \{\theta\} = \{(a, b)\}$ of parameters (imposing at least the obvious hypothesis that $\bar{\theta} \in \Theta$). In other words, we must live not with a single model, but with an entire *model class* $\{x(t+1) = ax(t) + bu(t) \ : \ (a, b) \in \Theta\}$. The goal is now to design a single control law that will work for all $\theta \in \Theta$, so that it will work, in particular, also for the "true" $\bar{\theta}$. This is called *robustness* with respect to uncertainty in the parameter.

Note that, besides the uncertainty around the one and only "true" parameter $\bar{\theta}$, there could be another reason why attaining robustness is a good practice: it could happen (read: it normally happens) that the one and only "true" parameter does not actually exist. *The system may be time-varying*, so that the parameter $\bar{\theta}$ may actually be a $\bar{\theta}_t$: recall the Jumbo Jet travelling from Frankfurt to New York (fuel consumption $\to$ weight change $\to$ gain scheduling). That systems will change their behavior in time is a normal fact of life, even when a control designer believes that they will not: think at the consumption of mechanical devices, at the deterioration of electrical components, at unforeseen changes in operating conditions of any kind. The hope (and the confidence) is that changes will be *slow*: if $\bar{\theta}_t$ is not constant, but changes slowly in comparison to the dynamics that it imposes (i.e. the dynamics of the time-varying system $x(t+1) = \bar{a}_t x(t) + \bar{b}_t u(t)$), a robust controller will typically do a good job in following $\bar{\theta}_t$ and counteracting the effects of its drift.

Anyway: for sure you have already encountered robustness in your basic control courses. That was the name of the game when the instructor introduced *stability margins*: they attained some sort of robustness with respect to uncertainty in [the parameters of] the transfer function, delays in the closed loop, etc., and they were quantities readily computable from the nominal transfer function.

0.4 A self-tuning regulator

Here we will follow a different strategy to attain robustness. We will split the controller in two components: one (*the tuning unit*) computing a better and better *estimate* $\hat{\theta}_t$ of $\bar{\theta}$ based on the input and output samples of the plant, and the other (*control unit*) computing the actual control action, with the same linear design scheme as in (4), but taking "the numbers" from the estimate $\hat{\theta}_t$ instead of the unknown $\bar{\theta}$.

The idea of computing the control action with a standard linear design scheme but employing $\hat{\theta}_t$ *as if it was* the "true" $\bar{\theta}$ is called *certainty equivalence principle*; the complex of the tuning unit and the control unit, estimating and acting together, is called a *self-tuning regulator*; and since in this game the ideal goal (and often the reasonable hope) is that $\hat{\theta}_t$ converges to $\bar{\theta}$ and the control unit converges to the linear controller that would be designed with the exact knowledge of $\bar{\theta}$, it is customary to say that the tuning unit gradually *adapts* the control unit to the initially unknown system, so that a self-tuning regulator is an *adaptive controller*.

Thus, now we work under the hypothesis that the "true" parameter of (3) is not completely known. To simplify ideas and computations, assume that uncertainty affects only the dynamic coefficient $\bar{a}$ of equation (5) but the "true" $\bar{b} = \frac{e-1}{e}$ is actually known to the designer. We let the unknown parameter be $\theta = a$, and the model class becomes

$$x(t+1) = \theta \cdot x(t) + \frac{e-1}{e} u(t). \tag{6}$$

The new ingredient that we need is a rule to estimate θ: the most classical one adopted by self-tuning regulators is Recursive Least Squares (RLS). Take for instance $\hat{\theta}_0 = \frac{1}{e}$ (nominal parameter, or initial guess), and for $t > 0$ update the estimation with the recursion:

$$\hat{\theta}_{t+1} = \hat{\theta}_t + \frac{x(t)}{\sum_{\tau=0}^{t} x(\tau)^2} \left(x(t+1) - \hat{\theta}_t \cdot x(t) - \frac{e-1}{e} u(t) \right). \tag{7}$$

For now, you can take this recursion as a "magic trick", whose structure and behavior will become clear in the second part of the course. Note, however, that if $x(t) = 0$ for $t = 0, 1, 2, \ldots$ ("insufficient excitation") (7) asks for repeated divisions by zero and does not actually make any sense. The obvious remedy is *not to update* $\hat{\theta}_t$ in this case; but to avoid unnecessary sophistication we replace (7) with the following recursion, called *regularized* RLS:

$$\hat{\theta}_{t+1} = \hat{\theta}_t + \frac{x(t)}{1 + \sum_{\tau=0}^{t} x(\tau)^2} \left(x(t+1) - \hat{\theta}_t \cdot x(t) - \frac{e-1}{e} u(t) \right). \tag{8}$$

Equation (8) is the dynamic equation of the tuning unit. It goes without saying that $\hat{\theta}_t$ is a state variable of such dynamic equation, and that $x(t+1)$ is one of its inputs; but it is easy to recognize that any meaningful software implementation of (8) should have also *another* state variable, namely the sum $S_t = \sum_{\tau=0}^{t} x(\tau)^2$ in one form or another. The real RLS algorithm has $P_t = (1 + \sum_{\tau=0}^{t} x(\tau)^2)^{-1}$ as the other state variable: how to update it in a smart way is a subject for future discussion.

Now we can plug the estimate $\hat{\theta}_t$ in place of the dynamic coefficient $\frac{1}{e}$ that was known in (4), and obtain the control action:

$$u(t) = \frac{e}{e-1}\left(-\hat{\theta}_t \cdot x(t) + r(t+1)\right) \tag{9}$$

Equations (8) and (9) together form a *self-tuning regulator.*

Exercise: simulate the self-tuning regulator with these parameters for the plant: $\theta = \frac{1.2}{e}$, $\theta = \frac{0.8}{e}$. Note: here is how the update step of the simulation should proceed[1]:

- at time t,
  - the plant is in the state $x(t)$,
  - the tuning unit is in the state $\hat{\theta}_t$, $S_t = \sum_{\tau=0}^{t} x(\tau)^2$, and keeps a copy of $x(t)$ as a further state,
  - the control unit has read $\hat{\theta}_t$ from the tuning unit, $x(t)$ from the plant and $r(t+1)$ from outside, and has computed the control action $u(t)$;
- now comes time $t+1$:
  - the plant reads $u(t)$ from the control unit and updates its state to $x(t+1)$,
  - the tuning unit reads $x(t+1)$ from the plant and updates $\hat{\theta}_{t+1}$, S_{t+1}, and the copy of $x(t+1)$,
  - the control unit reads $\hat{\theta}_{t+1}$ from the tuning unit, $x(t+1)$ from the plant and $r(t+2)$ from outside, and computes the control action $u(t+1)$.

At first sight, one may think that the action (9) computed by the control unit is a time-varying law, since the parameter $\hat{\theta}_t$ likely changes in time. This is not the case: indeed (9) is a non-linear, time-invariant map of the form

$$u = f(\theta, x, r) = \frac{e}{e-1}\left(-\theta \cdot x + r\right),$$

applied to the *three* inputs $(\hat{\theta}_t, x(t), r(t+1))$, coming respectively from the tuning unit, the plant, and the reference signal. Now both the equations (8) = tuning and (9) = control are thought of as the controller: then the first input to (9) is a *state* of the tuning unit, and hence of the controller itself. But then the term $\hat{\theta}_t \cdot x(t)$ is a product of a state and a 'regular' input, so that the whole controller (8) + (9), that is the self-tuning regulator, is time-invariant, but nonlinear by construction.

0.5 Convergence analysis

Here we show that the self-tuning regulator (8) + (9) attains asymptotic reference tracking, that is $\lim_{t\to\infty} x(t) - r(t) = 0$. We must assume that the reference signal $r(t)$ is *bounded*, i.e. there exists a positive constant $R < \infty$ such that $|r(t)| \leq R$ for all t.

Substitute the dynamics of the tank (6) into (8),

$$\hat{\theta}_{t+1} = \hat{\theta}_t + \frac{x(t)}{1 + \sum_{\tau=0}^{t} x(\tau)^2}\left(\theta \cdot x(t) - \hat{\theta}_t x(t)\right),$$

[1]Examine the update step carefully, think about how to implement it in working code, and you'll soon realize one of the reasons why control engineers want at least one delay in any real-world closed loop. How would you implement the self-tuning regulator *and the simulation* if the dynamics of the system was $x(t+1) = ax(t) + bu(\mathbf{t+1})$?

and define the estimation error $\tilde{\theta}_t := \hat{\theta}_t - \theta$; it follows

$$\begin{aligned}\tilde{\theta}_{t+1} &= \hat{\theta}_{t+1} - \theta \\ &= \hat{\theta}_t - \theta + \frac{x(t)}{1+\sum_{\tau=0}^{t} x(\tau)^2}\left(\theta \cdot x(t) - \hat{\theta}_t x(t)\right) \\ &= \tilde{\theta}_t - \frac{x(t)}{1+\sum_{\tau=0}^{t} x(\tau)^2}\left(\tilde{\theta}_t \cdot x(t)\right).\end{aligned}$$

Now multiply both sides by $1+\sum_{\tau=0}^{t} x(\tau)^2$:

$$\left(1+\sum_{\tau=0}^{t} x(\tau)^2\right)\tilde{\theta}_{t+1} = \left(1+\sum_{\tau=0}^{t} x(\tau)^2\right)\tilde{\theta}_t - \tilde{\theta}_t \cdot x(t)^2 = \left(1+\sum_{\tau=0}^{t-1} x(\tau)^2\right)\tilde{\theta}_t.$$

It follows that, for all $t > 0$,

$$\left(1+\sum_{\tau=0}^{t-1} x(\tau)^2\right)\tilde{\theta}_t = C = \text{constant}. \tag{10}$$

The term within parentheses in (10) is a positive, non-decreasing function of t, and as $t \to \infty$ either it tends to $+\infty$, or it converges to some constant $M \geq 1$. In both cases $|\tilde{\theta}_t|$ is a non-increasing sequence bounded from below, and hence it has a limit. We consider the two cases separately.

1. Suppose that $\sum_{\tau=0}^{t} x(\tau)^2 \to \infty$ ($x(t)$ is "persistently exciting" the tuning unit). For how the self-tuning regulator was conceived this is the most interesting case, because it implies that $\tilde{\theta}_t \to 0$, i.e. $\hat{\theta}_t$ converges to the "true" parameter of the system; that is, assuming that the system is time invariant and a "true" parameter exists. This is called *consistency* of the RLS estimator.

   Substituting the control action (9) into the dynamics of the system (6) we obtain the dynamics of the closed-loop system:

$$\begin{aligned}x(t+1) &= \theta \cdot x(t) + \frac{e-1}{e}u(t) \\ &= \theta \cdot x(t) - \hat{\theta}_t \cdot x(t) + r(t+1) \\ &= -\tilde{\theta}_t \cdot x(t) + r(t+1).\end{aligned}$$

   The consistency $\tilde{\theta}_t \to 0$ implies that this system is externally (BIBO) stable, and hence, since $\{r(t)\}$ is bounded, $\{r(t)\}$ is also bounded. But then

$$\lim_{t\to\infty} x(t+1) - r(t+1) = \lim_{t\to\infty} -\tilde{\theta}_t \cdot x(t) = 0,$$

   and this proves that asymptotic reference tracking is achieved.

2. If $\sum_{\tau=0}^{t} x(\tau)^2$ does not diverge, i.e. if $1+\sum_{\tau=0}^{+\infty} x(\tau)^2 = M < +\infty$, then (10) implies that $\tilde{\theta}_t$ also has a limit $\tilde{\theta}_\infty = \frac{C}{M}$. Yet the convergence of $\sum_{\tau=0}^{+\infty} x(\tau)^2$ implies that $x(t) \to 0$, and hence

$$\lim_{t\to\infty} x(t+1) - r(t+1) = -\tilde{\theta}_\infty \cdot \lim_{t\to\infty} x(t) = 0.$$

   In this case the RLS method is not consistent, i.e. it does not reach asymptotically a correct estimate of the "true" parameter; but reference tracking is achieved anyway.

0.6 What's the logic behind this so-called adaptive control?

Let $\mathcal{L}$ denote the class of linear controllers. Here is a fact that you know from you basic control courses and that has been reminded in Section 0.2:

- any linear system $\bar{\mathbf{s}}$ can be stabilized and attain reference tracking by means of a linear controller $\bar{\mathbf{c}} \in \mathcal{L}$.

Here is a fact of life that has been reminded in Section 0.3:

- we actually don't know the linear system, and hence we have to live with an entire class $\mathcal{S} = \{\mathbf{s}(\theta) \ : \ \theta \in \Theta\}$ of linear systems.

And here is the sad truth:

- no single linear controller $\bar{\mathbf{c}}_{\text{linear}} \in \mathcal{L}$ will do the trick for all the systems $\mathbf{s}(\theta) \in \mathcal{S}$.

But there is hope:

- it is still true that for all $\mathbf{s}(\theta) \in \mathcal{S}$ there exists a $\mathbf{c}(\theta) \in \mathcal{L}$ that will do the trick.

Note that the last two statements, "$\exists \mathbf{c} \in \mathcal{L}$ s.t. $\forall \mathbf{s} \in \mathcal{S}$ tracking is achieved" and "$\forall \mathbf{s} \in \mathcal{S}\ \exists \mathbf{c} \in \mathcal{L}$ s.t. tracking is achieved" are very different, the former being way more demanding (and indeed false), in the same way as the following ones are: $\exists n \in \mathbb{N}$ s.t. $\forall m \in \mathbb{N}\ n > m$ (there exists a maximum integer); $\forall m \in \mathbb{N}\ \exists n \in \mathbb{N}$ s.t. $n > m$ (we can always find a larger integer).

The point is that we don't know the "true" θ. Thus, we have to enlarge the class of controllers, and enter the realm of nonlinearity. Let $\mathcal{N}$ be the class of "all nonlinear controllers".

- Yes, there is a controller $\bar{\mathbf{c}}_{\text{nonlinear}} \in \mathcal{N}$ that will do the trick for all the systems $\mathbf{s}(\theta) \in \mathcal{S}$. Actually, there are infinitely many.

But if we had to search in $\mathcal{N}$ with no guideline, the enterprise would be desperate from the very start, for $\mathcal{N}$ is way too complex to manage. So here is a guideline, a *particular method* to design a nonlinear controller, called *adaptive control*:

- Build a controller $\bar{\mathbf{c}}_{\text{adaptive}} \in \mathcal{N}$ that, examining the behavior of any $\mathbf{s}(\theta) \in \mathcal{S}$, "goes searching" for the correct $\mathbf{c}(\theta) \in \mathcal{L}$. When applied to a particular $\mathbf{s}(\bar{\theta})$ in closed loop, in the long run $\bar{\mathbf{c}}_{\text{a}}$ will behave like $\mathbf{c}(\bar{\theta})$.