TP - Introduction to Digital Control

Introduction

In the previous exercises, all controllers were designed to control a simulated system. The objective of this TP is to get familiar with the implementation of a digital RST controller, and control the position of the red disk of a QUBE Servo 2, shown in Fig. 1. The implementation of the controller will be done using the `C` programming language. No extensive knowledge of this programming language is required, only the basics (array and for loops).

In the following sections, the sampled version of a signal $y(t)$ is denoted $y_k = y(t = kh)$, where $h = 5$ ms the sampling period. The backward shift operator is denoted q^{-1}, and $q^{-1} \cdot y_k = y_{k-1}$.

Figure 1: DC motor - QUBE Servo 2

RST controller (recap)

RST is a 2-degree of freedom controller. The block diagram using an RST controller can be found in Fig 2. The R, S, T transfer functions can be obtained using the (discrete-time) methods seen during the *Automatique et Commande Numerique* course, or by discretizing a continuous controller. In this TP, the RST controller will be designed using the pole-placement technique.

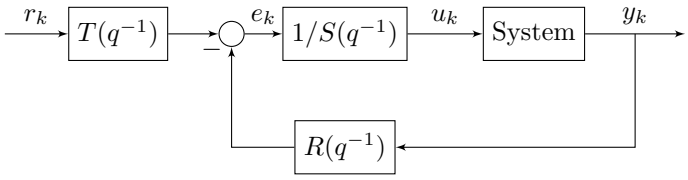

Figure 2: RST block diagram

From an implementation point of view, the controller's objective is to compute the output u_k, given the measurements and references. The algebraic relation between the output u_k and the inputs (of the controller) $\{r_k, y_k\}$ is given by:

$$\frac{T(q^{-1})r_k - R(q^{-1})y_k}{S(q^{-1})} = u_k$$

or equivalently

$$T(q^{-1})r_k - R(q^{-1})y_k = S(q^{-1})u_k. \tag{1}$$

Denote R_i, S_i, T_i the i^{th} coefficient of the polynomials $R(q^{-1})$, $S(q^{-1})$, $T(q^{-1})$. It is assumed each polynomial has order N, and therefore the highest power is q^{-N}. Using the shift-backwards property of q^{-1}, (3) can be transformed into the following difference equation:

$$\sum_{n=0}^{N} T_n r_{k-n} - \sum_{n=0}^{N} R_n y_{k-n} = \sum_{n=0}^{N} S_n u_{k-n}$$

If $S(q^{-1})$ is monic ($S_0 = 1$), then solving for u_k results in:

$$\begin{aligned} u_k &= \sum_{n=0}^{N} (T_n r_{k-n} - R_n y_{k-n}) - \sum_{n=1}^{N} S_n u_{k-n} \\ &= T_0 r_k - R_0 y_k + \sum_{n=1}^{N} (T_n r_{k-n} - R_n y_{k-n} - S_n u_{k-n}) \end{aligned} \tag{2}$$

This last equation is the implementation of the RST controller shown in Fig. 2, and you will implement it in this TP. It should be noted that computing u_k requires the current measurement y_k and reference r_k, along with $N - 1$ previous outputs, measurements, and references.

TP instructions

1) RST controller design (Pen & Paper)

The model of the system shown in Fig. 1 is given by

$$G(q^{-1}) = \frac{0.05522q^{-1} + 0.1877q^{-2}}{1 - 1.961q^{-1} + 0.9609q^{-2}}.$$

Design an RST controller using the pole-placement technique with the following specifications:

- Place the dominant pole at $q = 0.8$ and all auxiliary poles at $q = 0.4$,
- Force an integrator in the controller using the fixed term $H_s = [1, -1]$,
- Use $T(q^{-1}) = R(1) + 0q^{-1} + 0q^{-2}$

Note: Do the matrix inversion with your favorite numerical software package.
Note: If you did not succeed after the first hour of the PC, ask a TA for the solutions.

2) RST implementation (Remote Access)

Implement the designed RST controller using the `C` programming language. Connect to one of the QUBEs via the the link in under: Control Systems MOOC-TP → Module 8 → Implemenatation → QUBE access.
Use Safari or Chrome as your browser!

Modify the settings as:

- **Measurement:** Position
- **Controller type:** Custom
- **sampling period:** 0.005 s

The "**Custom**" mode allows you to implement your own controller by using the coding window provided below (programming in C). Here you should compute the control action to be applied to the system at each time step according to (2). Note that you are not allowed to define new variables here. However, you have access to an array of parameters that you can define in the **"Params"** tab which you can use for storing your RST controller parameters and **"Globals"** array that can be used to store values between time steps.

Warning: The array that is used to store the past input actions is referred by `pCmd` in the comments inside the coding window. However, you should use `pCMD` instead. Sorry for the typo!

[1] `https://lucris.lub.lu.se/ws/files/6377599/8627775.pdf`

Validation on the Quanser Qube

Once you implement your controller, press the "**Check source**" button to compile your code.If everything is fine you should see the message "Submitted code is valid" in green. Then you can execute your code by pressing the "**Apply source**" button. Now your controller is in action!

The RST conrtoller without anti-windup is very sensitive against disturbances. Thus, it is important to initialize the system by setting the initial error to 0. To do so, you can first set `CMD=0` and execute your controller and also set the reference to 0. Then, by connecting to a new device through the drop-down list on top you can also reset all the initial states to 0. Next, you can define `CMD` as in (2) and apply it to test if the closed-loop system can track references.

Reference tracking

Apply a square waveform as reference. Select appropriate values (e.g., Amplitude 10, 20 or 30 deg and frequency 0.1 Hz). You should see the disk on top moving according to the chosen reference. Try gradually increasing the amplitude until saturation effects result in too much instability. You should observe that without an anti-windup controller the closed-loop is not robust against disturbances and fails tracking a fast changing or large amplitude reference.

Note: The encoder measuring the red disk position is an incremental encoder. No absolute position is available, and the position is initialized to zero at startup.

3) RST with Anti-windup

When the reference jumps by a too-large value, the voltage saturates. Implement an anti-windup scheme for the RST controller as described on page 71 eq. (66) of this document[1], where

$$A_{aw}(q^{-1}) = 1 + 0 \cdot q^{-1} + 0 \cdot q^{-2},$$

and

$$\texttt{sat}(v_k) = \min(5, \max(-5, v_k)).$$

The corresponding block diagram can be found in Fig. 3.

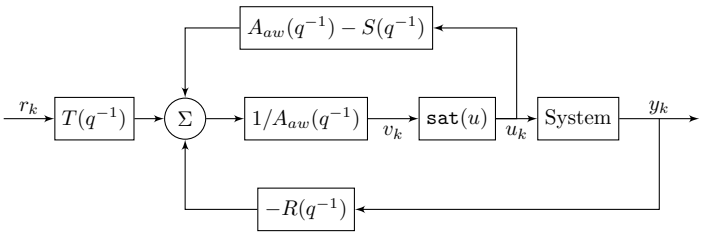

Figure 3: RST block diagram with saturation. A_{aw} is a stable and monic polynomial. Σ indicates a summation of all entering signals.

Remark: The linked PDF uses a different notation than presented in the course: R in the PDF corresponds to S in the course, and S in the PDF corresponds to R in the course. The notation in Fig. 3 is in parallel with the course.

Hint: From Fig. 3 you have:

$$A_{aw}(q^{-1})v_k = T(q^{-1})r_k - R(q^{-1})y_k + (A_{aw} - S(q^{-1}))u_k. \tag{3}$$

By using (3), you can derive the equation to compute v_k the same way we derived (2) from (1). Then, from v_k you can obtain the output of your controller as:

$$u_k = \mathtt{sat}(v_k) = \mathtt{min}(5, \mathtt{max}(-5, v_k)).$$

After implementing the anti-windup scheme, track a square waveform reference with higher amplitude and frequency (e.g., Amplitude 90 or 180 *deg* and frequency 0.8 Hz).