

Model for simulator and model for controller design

The model (21.24) – (21.27) is the basis of both the simulator (representing the real process) and the controller. However, to make it possible to check if the control system is robust, two sets of model parameters are available in the front panel of the simulator:

- Model parameters $M_{\text{real}}, m_{\text{real}}, L_{\text{real}}, d_{\text{real}}$ used in the simulator.
- Model parameters $M_{\text{model}}, m_{\text{model}}, L_{\text{model}}, d_{\text{model}}$ used in the design of the controller.

By default, these two parameter sets have equal numerical values, but if you want to check for robustness of the controller against variations or inaccurate knowledge about one certain parameter, you must set the values of the corresponding parameters to different values, e.g. set d_{model} to a different value from d_{real} .

Figure 21.6 shows simulated responses for the control system. The positional reference was changed. The cart converges to the reference value, and the pendulum is stabilized upright despite the changes of the the cart position.

[End of Example 21.1]

21.3 LQ controller with integral action

21.3.1 Introduction

The basic LQ controller described in Section 21.2 is based on assumptions that in many applications are unrealistic or idealized. The following is more realistic:

- The reference is non-zero.
- There is a reference for a number of the states variables (not for the whole state vector).
- The process disturbance has non-zero mean value.

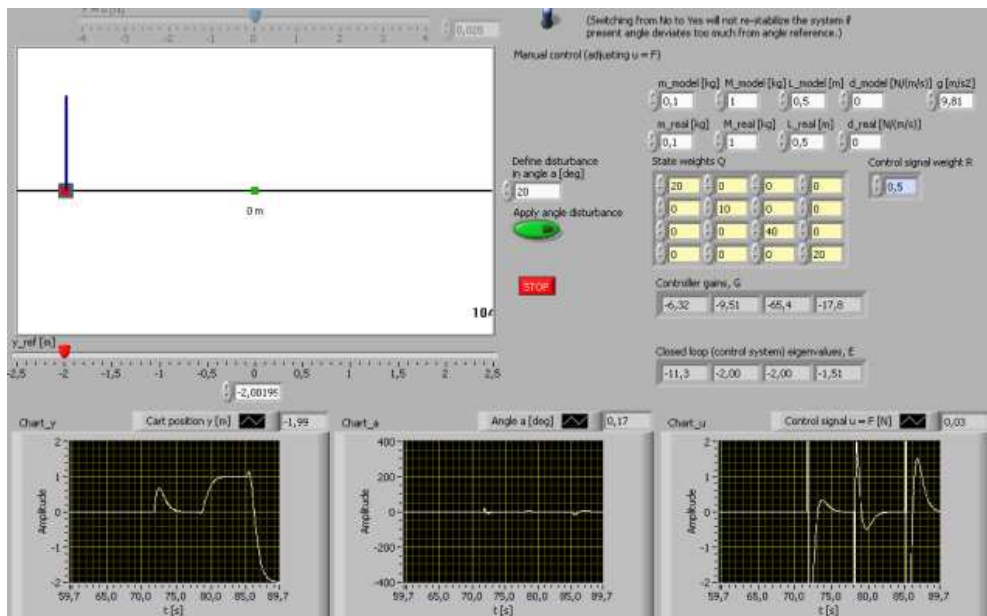


Figure 21.6: Simulated responses for the control system of the pendulum on the cart

The controller given by (21.13) does not ensure zero steady-state control error under these assumptions. What is needed is integral action in the controller!

21.3.2 Including integrators in the controller

Figure 21.7 shows a block diagram of the control system with integrators included in the controller. The integrator block is actually a number of integrators, as many as there are reference variables.

Example 21.2 *LQ controller with integrator*

Figure 21.8 shows the detailed structure of a process with two state variables being controlled by a LQ controller with integrator.

[End of Example 21.2]

The output of the integrators are regarded as augmentative state variables, x_{int} . These state variables are given by the following differential

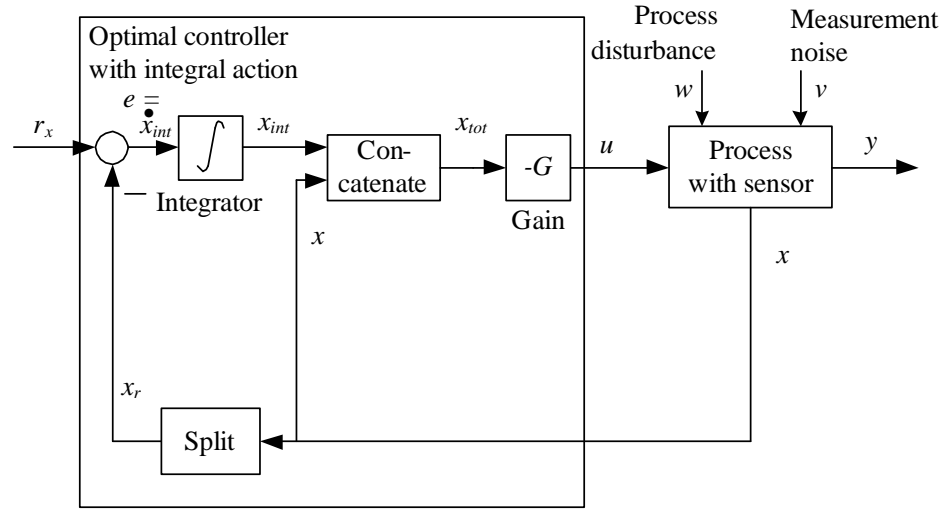


Figure 21.7: Optimal control system with integrators in the controller. e is the control error.

equation(s):

$$\dot{x}_{int} = r_x - x_r \quad (21.41)$$

which corresponds to this integral equation:

$$x_{int}(t) = \int_0^t (r_x - x_r) d\tau \quad (21.42)$$

Here, r_x is the reference vector for the state vector x_r which consists of those state variables among the process state vector x that are to track a reference. (In the above equations it is assumed that x_r is directly available from measurements. If x_r is taken from a state estimator, $x_{r,est}$ is used in stead of x_r , of course.)

The total state vector that is used to design the LQ controller is the state vector consisting of the original process state vector augmented with the integrator state vector:

$$x_{tot} = \begin{bmatrix} x \\ \cdots \\ x_{int} \end{bmatrix} \quad (21.43)$$

The control variable u is given by the control function

$$u = -Gx_{tot} = -G \begin{bmatrix} x \\ \cdots \\ x_{int} \end{bmatrix} \quad (21.44)$$

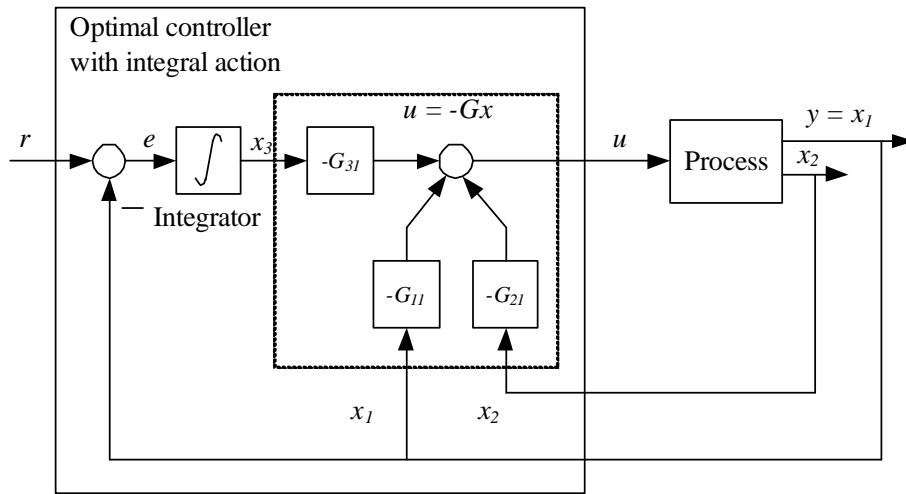


Figure 21.8: Example 21.2: A process with two state variables being controlled by a LQ controller with integrator

Note: When writing up the state-space model that is used for designing the LQ controller, you can disregard the reference r_x , i.e. you set it to zero because it is not taken into account when calculating the controller gain G . But of course it must be included in the implemented controller which is given by (21.44), with x_{int} given by (21.46).

21.3.3 Discrete-time implementation of the LQ controller

In a computer-based implementation of a LQ controller you will probably need to discretize the continuous-time integrator (21.41). This can be done using Backward or Forward discretization. The Backward method is the best with respect to numerical accuracy, and it can be applied without any problem to (21.41) because it is a linear differential equation. Applying Backward discretization on (21.41) gives

$$\dot{x}_{int}(t_k) \approx \frac{x_{int}(t_k) - x_{int}(t_{k-1})}{T_s} = r_x(t_k) - x_r(t_k) \quad (21.45)$$

Solving for $x_{int}(t_k)$ gives the final integrator algorithm ready for being programmed:

$$x_{int}(t_k) = x_{int}(t_{k-1}) + T_s [r_x(t_k) - x_r(t_k)] \quad (21.46)$$

A practical issue of any controller having integral action is *anti windup*, which is a feature to prevent the integrator to “wind up” – or increasing its