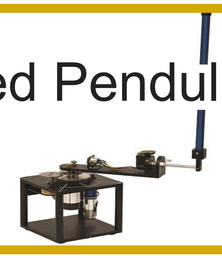
Design of a Balance and Swing-up Control System for a Rotary Inverted Pendulum

Bryan Clum Guner*, Christopher Hingston Tenev*
*Department of Electrical Engineering
The College of New Jersey, Ewing, NJ



Abstract

The purpose of this experiment was to design and implement a state-feedback control system to control a rotary inverted pendulum system. This exercise served to introduce students to the process of linearizing nonlinear equations of motion in order to obtain the linear state-space representation of the pendulum system, and to demonstrate the effectiveness of single-loop feedback control in electromechanical application. A state-space model of the Quanser SRV02 Rotary Pendulum System was developed, and controls for directing the system's balance and vertical orientation were designed.

Introduction

In these experiments, a hardware system was modelled in MATLAB and controlled in Simulink using a single-loop feedback system.

The hardware system that was modelled and controlled consists of an actuated rotary arm, connected to a free-swinging, invertible pendulum arm; the positions of both arms are determined in real time by encoders. To provide a reliable and consistent mathematical model of the hardware system, time-invariant state-space representation was employed, which formed the basis of all the implemented control.

The system is required to balance the pendulum arm vertically, and to swing the pendulum up from the downward-hanging position. Control of the pendulum system was implemented in Simulink, by applying negative feedback to the motor driver; the feedback gain required to control the system was calculated in real time using the previously determined state-space model and the principles of pole placement, according to the desired control type.

Methods

Modelling

- State-space representation of the hardware components allows for description of the system with time-invariant differential equations.
- Time-invariant model allows for precise control, regardless of component position, speed, or acceleration.
- State-space model is used to continuously determine the control gain of the control voltage which drives the system's motor.

ROTPEN module

- State-space system model is implemented in a Simulink block to allow integration into larger systems within Simulink.
- Module reads encoder values from rotary and pendulum arms, and outputs rotary arm angle, θ, "raw" pendulum angle, α_r, and pendulum angle with respect to vertical, α.
- Negative inverse of control voltage drives motor to counteract falling of pendulum.

Balance control

- Signal generator provides vector state which instructs pendulum to remain within 20° of vertical.
- Multiport switch enables balance control when pendulum is within 20° of vertical; input voltage is zero when pendulum is further than 20° from vertical, and equals system error otherwise.
- ROTPEN module counteracts error by applying negative inverse of system error to the motor.

Swing-up control

- To allow for swing-up control, input voltage must be switched between error and energy-based control.
- Swing-up control must constantly move the pendulum with enough force to counteract gravity without overshooting the desired position.
- Friction must remain constant to avoid introduction of position- and velocity-dependent damping.

$$\begin{bmatrix}\dot{\theta}\\ \dot{\alpha}\\ \ddot{\theta}\\ \ddot{\alpha}\end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{0.25L_p^2L_rgm_p^2}{J_rJ_p+0.25m_pJ_rL_p^2+J_pm_pL_r^2} & -\frac{B_r\left(J_p+0.25m_pL_p^2\right)}{J_rJ_p+0.25m_pJ_rL_p^2+J_pm_pL_r^2} & -\frac{0.5B_pL_pL_rm_p}{J_rJ_p+0.25m_pJ_rL_p^2+J_pm_pL_r^2} \\ 0 & \frac{0.5L_pm_pg\left(J_r+m_pL_r^2\right)}{J_rJ_p+0.25m_pJ_rL_p^2+J_pm_pL_r^2} & -\frac{0.5B_pL_pL_rm_p}{J_rJ_p+0.25m_pJ_rL_p^2+J_pm_pL_r^2} & -\frac{-\left(J_r+m_pL_r^2\right)B_p}{J_rJ_p+0.25m_pJ_rL_p^2+J_pm_pL_r^2} \end{bmatrix} \begin{bmatrix}\theta\\ \alpha\\ \dot{\theta}\\ \dot{\alpha}\end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{0.25m_pL_p^2+J_p}{J_rJ_p+0.25m_pJ_rL_p^2+J_pm_pL_r^2} \\ \frac{L_pL_rm_p}{J_rJ_p+0.25m_pJ_rL_p^2+J_pm_pL_r^2} \end{bmatrix} u(\tau)$$

Figure 1: State-space model of inverted rotary pendulum system. Theta values refer to the pendulum, and alpha values refer to the rotary arm.

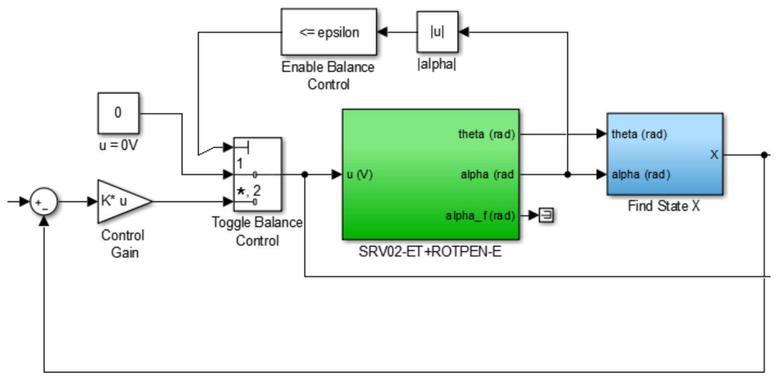


Figure 2: Balance control block diagram. The input to the rotary pendulum module (shown in green) is the error of the system; the rotary pendulum module takes the negative inverse of this input signal and applies it to the rotary arm motor to counteract the error. The switch enables negative feedback and control gain when the pendulum is within 20° of vertical.

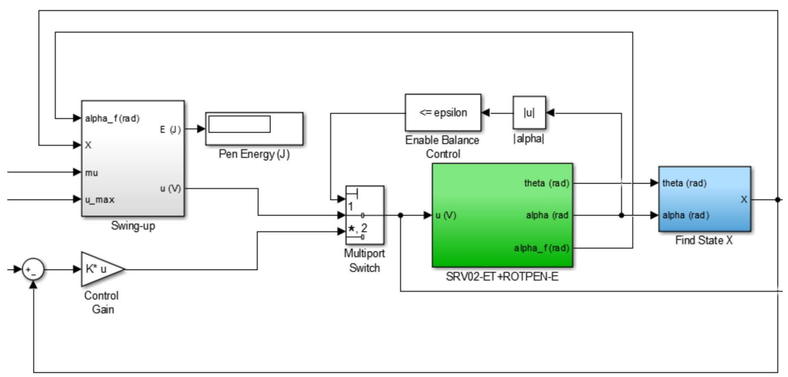


Figure 3: Block diagram showing implementation of rotary pendulum model with both balance and swing-up control. By replacing the 0 V input block in Figure 2 *with the constant-energy swing-up control module shown here, the pendulum arm is swung upwards when it is further than 20° from vertical. When the pendulum arm is within 20° of vertical, the multiport switch enables the balance control which keeps the pendulum vertical.*

Results

- State-space model of the rotary inverted pendulum system was devised, and the result was confirmed algebraically (see *Figure 1*).
- Simulink models were successfully implemented, and appropriate communication with hardware system was achieved.
- Some level of balance control was achieved, but the control gain appeared too high; motor arm over-compensated for the tilting of the pendulum arm.
- Swing-up control was attempted, but again, gain was too high.
- Manual gain adjustment was attempted, but the complexity of adjusting a vector quantity prevented full balance control from being achieved.

Conclusions

Despite setbacks and a lack of fully optimized balance and swing-up controls, the design successfully demonstrated the modelling of a hardware system in Simulink and MATLAB using state-space representation of the system and its output, as well as the control of such a model within the MATLAB/Simulink environment. State-space representation provided a model that could reliably describe the system regardless of the physical state of the hardware, and balance and swing-up controls were achieved through the implementation of simple feedback systems, the gain of which was determined in real time by the state-space system model and the physical status of the system.

It is believed that the error in implementing the controls was due to incorrect implementation of the state-space system model in MATLAB. The derivation of the state-space matrices was checked and re-checked, and the matrices are believed to be accurate. The error is thought to lie in the computation of the system's closed-loop poles, which are used by MATLAB to continuously calculate the system's control gain. Unfortunately, troubleshooting and optimization were not executed quickly enough to achieve the desired results within the time allotted for the experiments. Given more time to troubleshoot the MATLAB operations, it is likely that optimal performance would be achieved.

State-space modelling of hardware systems, in conjunction with single-loop feedback control systems implemented in Simulink, provide simple, yet powerful, tools for observing and controlling hardware systems mathematically. As any system may be represented by a single-loop feedback system, these methods may be used to model and control virtually any control system. The use of alternative control gain calculation methods, along with alternative system modelling techniques, such as with solid modelling CAD software, further expand the functionality and versatility of these design techniques.

References

[1] Nise, Norman S. *Control Systems Engineering.* 7th ed. New York: John Wiley, 2000. Print.