University of Toronto

Department of Electrical and Computer Engineering

ECE411S – Real-Time Computer Control

Lab ~~4~~ 5

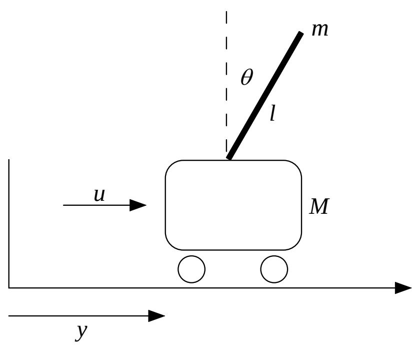
Real-Time Scheduling of Three Inverted Pendulums

1 Purpose

This lab will introduce you to TrueTime, a MATLAB and Simulink based simulation program. Using TrueTime, you will simulate the responses and task schedules of three inverted pendulums controlled by one processor. Scheduling algorithms and priority assignments will be investigated as well.

2 Introduction

Consider the cart-pendulum:



The linearized state equation is

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{mg}{M} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{(M+m)g}{Ml} & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ -\frac{1}{Ml} \end{bmatrix} u.$$

The problem is to control three pendulums and carts by a single processor. They have the following physical parameters:

Name	$M(kg)$	$m(kg)$	$l(m)$
Pendulum A	0.45	0.2	0.3
Pendulum B	0.45	0.26	0.35
Pendulum C	0.45	0.3	0.4

Of course, $g = 9.81m/s^2$.
Let the design specifications be the following:

(a) The inverted pendulums are balanced in the upright position.

(b) The carts track a unit step.

(c) Satisfactory transient response.

The state is assumed to be available for feedback control.

You have to learn how to use TrueTime from the manual. You can download the TrueTime version 2.0 Reference Manual from

http://www.control.lth.se/truetime

You should read the MATLAB-relevant parts of Sections 3-7. It is recommended that you also download the TrueTime version 2.0 software from the same website, even if you do not have MATLAB 7 or higher to run the software. By unzipping the truetime-2.0.zip file, you will find the kernel directory and the examples directory. The TrueTime kernel is started as a Simulink block by truetime.mdl in the kernel directory. The threeservos example in the examples directory is useful as a guide to building the Simulink diagram for your experiment.

In class we assumed that when a task misses its deadline, it gets terminated and the newly released task gets executed. However, if you initialize the TrueTime kernel simply based on the instructions in Sections 3-7 of the TrueTime manual, this will not be the case; i.e., a task that misses its deadline will continue to run and the newly released task will be queued.

So to terminate tasks that miss their deadline, you need to add a deadline overrun handler to your code. Simply add the following two lines to your TrueTime kernel init file, after “ttCreatePeriodicTask('name_of_pend_task_x',0,...)”:

```
ttCreateHandler('name_of_DL_handler_x', 1, 'hdlcode');

ttAttachDLHandler('name_of_pend_task_x', 'name_of_DL_handler_x');
```

and download the file hdlcode.m from the course website into the same directory. The hdlcode.m file can also be found in the overrun sub-directory in the examples directory. For more details, read Section 11.4 of the manual.

3 Preparation

1. Determine the A and B matrices for the 3 pendulums, A_1 corresponding to the A matrix for pendulum A, etc.

2. Consider pendulum A. The tracking specification can be converted to a simple specification in the following way. Let $r(t)$ denote the reference step input. Then $\dot{r} = 0$, with $r(0) = 1$. Define $\tilde{x} = \begin{bmatrix} x_1 - r & x_2 & x_3 & x_4 \end{bmatrix}^T$. Write down the differential equation that $\tilde{x}$ satisfies. Express the design specifications in terms of the desired value of $\tilde{x}$.

3. Since the full state is available, the control law can be taken to be

$$u = F\tilde{x}$$

Determine an feedback gain, call it F_1, such that the design specifications are met. You may use pole placement, or lqr if you have already learned lqr design from previous courses. Build a Simulink diagram for controlling pendulum A in continuous time, with the unit step as the reference input. Record the simulation on a Simulink scope and print the trajectories of the pendulum states to show that the design specifications are met.

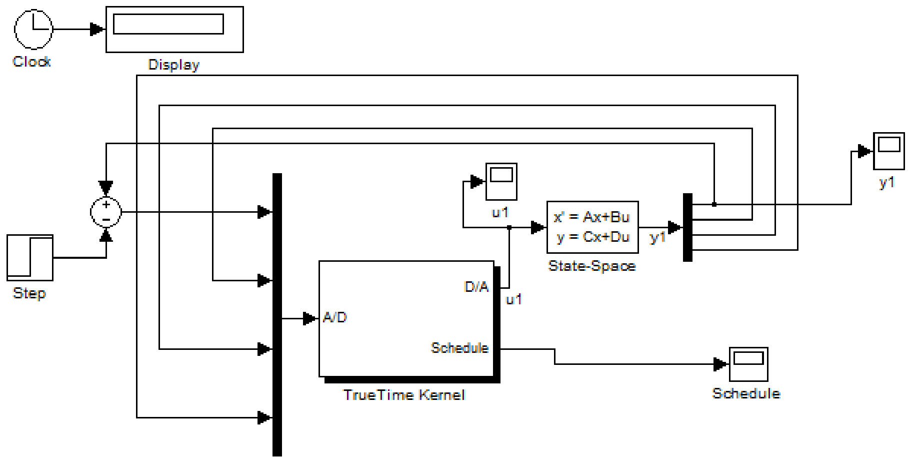
4 Experiment

4.1 Using TrueTime

1. Using your control design for pendulum A in your preparation as a guide, design state feedback gains for pendulums B and C to meet design specifications. Note that since the control law is static, there is no further **c2d** operation needed to implement the controller.

2. Record the poles you used and the resulting F matrices, for each system.

3. Using TrueTime, build a Simulink model that incorporates the TrueTime kernel block for controlling pendulum A. The Simulink model should look like this:



The total number of inputs into the TrueTime kernel block A/D should be 4, and the total number of outputs from the D/A should be 1. Thus, the TrueTime kernel is controlling only 1 pendulum. Assume execution times (C_i's) of 10, 14 and 17ms for pendulums A, B and C, respectively. Choose appropriate periods for each system (to yield a utilization factor U of about 70% for each system), and set all initial conditions to zero. Record your choices of periods and the corresponding utilization factors for the 3 pendulums.

4. Incorporate all the data for real-time control of pendulum A, i.e., the feedback gains, execution time, and period into the initialization script file for the TrueTime kernel block. Simulate the response of pendulum A, record it on a Simulink scope, and print the response. Is the response satisfactory? Sketch the task schedule of pendulum A for several periods, using the scope connected to the Schedule port. Is the schedule consistent with the period and execution time?

5. Repeat the above for pendulums B and C.

In the experimental work that follows, you will compare the responses of the scheduled pendulums to the simulations generated above.

4.2 Fixed-Priority Scheduling

1. Build the system with all three pendulums being controlled by one TrueTime kernel. Now, the total number of inputs into the TrueTime kernel block A/D should be 12, and the total number of outputs from the D/A should be 3. Use the rate-monotonic priority assignment. Set the execution times and periods equal to those used in Part 1 (do not redesign the T_i's from Section 4.1). Set the priority accordingly. Why are the performances not satisfactory?

2. Keep the same execution times, but adjust the periods so that the utilization factor is 75%: Allow each system to account for 25% of U. Are the responses satisfactory? You can judge by comparing the graphs to those obtained in Section 4.1. If they are not, explain why.

3. Now adjust the periods (by trial and error) so that the responses of the systems are satisfactory, and all tasks meet their deadlines. What are the new periods, and the corresponding utilization factor? Print the response of System C, and the task schedule.

4. Use the deadline-monotonic priority assignment ('prioDM'). What is the performance of System A? Now reverse the priorities: $P_C = 3, P_B = 2, P_A = 1$ (or $P_C = 3, P_B = 1, P_A = 2$). How does this affect the performance of System A? Does System A meet its deadlines? What does this indicate about the fixed-priority scheduling?

4.3 Earliest-Deadline-First Scheduling

1. Change the scheduling algorithm to EDF. By adjusting the T_i's, see how high you can push U before the performances deteriorate significantly. What are the new periods, and the corresponding U?

2. Print and label the step responses of the three systems.

4.4 Discussion

1. Briefly discuss what you believe are the advantages and disadvantages of the fixed-priority and earliest-deadline first scheduling algorithms.