EL2450: Hybrid and Embedded Control Systems: Homework 2

Introduction

This homework consists of two parts, the first part is about scheduling and the second about networked control systems. In the first part, control performance for three control tasks executed on the same CPU will be evaluated for different scheduling algorithms. In the second part, networked control systems with delay and packet losses will be modelled and analyzed.

Part One: Scheduling [25p]

This part of the homework is on analysis, simulation and implementation of a realtime control system. The main focus is on scheduling of the processor executing the control algorithms. The processes that should be controlled are three inverted pendulums and the control algorithms are all running on the same processor. To be able to simulate such a system, a Simulink toolbox called Truetime is used.

Exercises

Download the Matlab files package from the homepage. The processes to be controlled are illustrated in Figure 1.

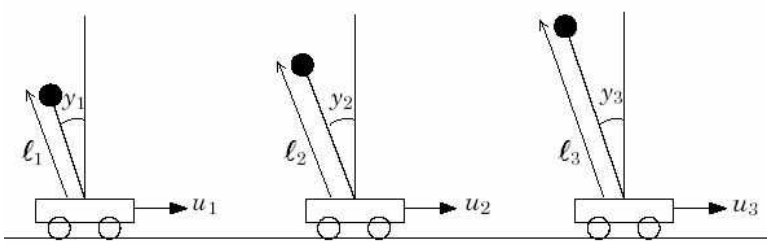

Figure 1: Three pendulums.

A linearized model of a pendulum is given by the transfer function:

$$G_i(s) = \frac{\omega_i^2}{s^2 - \omega_i^2} \ , \quad i = 1, 2, 3$$

where ω_i is the natural frequency of the pendulum. In this example the pendulums are of different length, $\ell_i = \{0.1, 0.2, 0.3\}$ m, which corresponds

to the frequencies $\omega_i = \{9.9, 7.0, 5.7\}$ rad/s. Each pendulum is controlled by a controller derived with discrete LQG. Because of the different lengths the sampling times for the controllers are not equal. The sampling times T_i are chosen to be 20, 29 and 35 ms (a shorter pendulum is harder to stabilize and needs a shorter sampling period). The discrete controller for the shortest pendulum is:

$$C(z) = \frac{-9.243z^2 + 7.278z}{z^2 - 0.3768z + 0.1392}.$$

Rate Monotonic scheduling

1. Explain what Rate Monotonic scheduling means. **[1p]**

2. Assume the execution time for all three tasks are equally $6ms$. Are the tasks schedulable with this policy? Please motive your answer and construct part of the corresponding schedule manually. **[2p]**

3. Install Truetime and Jitterbug by following the Appendix. Once it is done, open the simulink model by typing
   `> inv_pend_three`
   Simulate the system. Plot and save the pendulum angles. Are the pendulums stabilized? What are the differences in control performance between the different pendulums? **[2p]**

4. Compare the schedule plot with yours. Does the result agree with the analysis, *i.e.*, does the task get time to execute before their sampling period is over? **[2p]**

5. Now assume the execution time for all three tasks are equally $10ms$. Then resolve Question 2, 3 and 4, with the new execution time. What are the differences with respect to the control performance and the schedule? **[3p]**

   (*Hint*: Open `init_pend_three.m` and change the execution time to $10ms$.)

Earliest Deadline First (EDF)

Next we will examine how EDF scheduling affects the performance of the system.

1. Explain how Earliest Deadline First scheduling works. What are the advantages and the disadvantages of using the EDF compared to RM? **[1p]**

2. Assume the execution time is changed back to $6ms$. Are the tasks schedulable with the EDF? Please motive your answer and construct part of the corresponding schedule manually. **[2p]**

3. Open `init_pend_three.m` and change the scheduling policy to 'prioEDF'. Save the script. Rerun `inv_pend_three.mdl`. Plot and save the angles of three pendulums. Are the pendulums stabilized? What are the differences in control performance between different pendulums? **[2p]**

4. Compare the schedule plot with yours. Does the result agree with the analysis, *i.e.*, does the task get time to execute before its sampling periods is over? **[2p]**

5. Resolve Question 2, 3 and 4 with the new execution time $T = 10ms$. What are the differences with respect to the control performance and the schedule? **[4p]**

6. Do the controllers perform better than under RM-scheduling? Please motive your answer using the simulation results. **[4p]**

Part Two: Networked Control Systems [10p]

In the second part of the homework, you will analyze networked control systems (NCS).

Stability of NCS with Network-induced Delay

Consider the Networked Control Systems in Figure 2. The system consists of a continuous plant

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t)$$

and a discrete controller

$$u(kh) = -Kx(kh), \qquad k = 0, 1, 2, \ldots$$

where $A \in \mathbb{R}$, $B \in \mathbb{R}$, $C \in \mathbb{R}$.

Consider a simple integrator model, *i.e.*, $A = 0$, $B = I$. Further assume that the delay is less than one sampling period, *i.e.*, $\tau = \tau_{sc} + \tau_{ca} \leq h$.

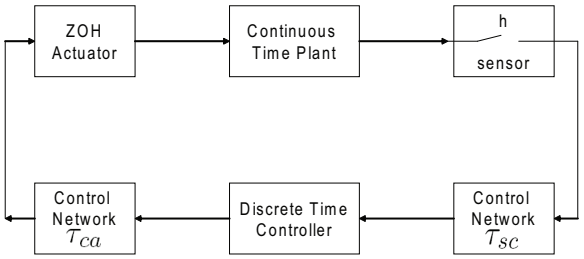

Figure 2: Networked Control System with communication delay.

1. Calculate analytically the closed-loop equations of the system. **[2p]**

2. Calculate analytically the region of stability of the system as a relation between Kh and $\frac{\tau}{h}$. Derive a plot of the stability region, where the unstable and stable parts are illustrated. Comment on interesting values of h. **[4p]**
   (*Hint*: Use the stability triangle for second order systems)

Now we will simulate an example of NCS with network-induced delay to show how it affects the control performance. Make sure you have the file `inv_pend_delay.mdl` and open the simulink model by typing
`> inv_pend_delay`
Simulate the system. Check by simulations that for which value of the communication delay (in the *Transport Delay* block) the system becomes unstable. Please support your answer by simulation results. **[4p]**

Part Three: Discrete Event System [15p]

Consider the following manufacturing process which involves two machines ($\mathbf{M}_1$ and $\mathbf{M}_2$) and a buffer ($\mathbf{B}$) in between. $\mathbf{M}_1$ and $\mathbf{M}_2$ are modeled in the same way, as described in the following. Each machine has three states: *Idle* (the initial state), *Processing*, and *Down*; and four transitions:

- from *Idle* to *Processing* by event `START`,
- from *Processing* to *Idle* by event `END`,
- from *Processing* to *Down* by event `BREAKDOWN`,
- from *Down* to *Idle* by event `REPAIR`.

Moreover, there is an infinite supply of parts to $\mathbf{M}_1$. When a part is processed at $\mathbf{M}_1$, it is placed in $\mathbf{B}$, which has a capacity of one part only. The part is subsequently processed by $\mathbf{M}_2$, which leads to the final product. The buffer has two states: *Empty* (the initial state) and *Full*; and two transitions:

- from *Empty* to *Full* by event `PLACED`,
- from *Full* to *Empty* by event `TAKEN`.

Answer the following questions:

(1) Construct and draw the discrete event system (DES) that models $\mathbf{M}_1$, $\mathbf{M}_2$ and $\mathbf{B}$.**[4p]**

(2) Construct, draw and describe the complete DES that models all behaviors of the manufacturing process that fulfills the descriptions above. **[4p]**

(3) Now consider that the system needs to be restricted to satisfy the following rules: (i) $\mathbf{M}_1$ can only begin processing if the buffer is empty; (ii) $\mathbf{M}_2$ can only begin processing if the buffer is full; (iii) $\mathbf{M}_1$ cannot begin processing if $\mathbf{M}_2$ is down; (iv) if both machines are down, then $\mathbf{M}_2$ gets repaired first. Among the admissible behaviors from (2), find the admissible ones satisfying these new rules. **[3p]**

(4) Suppose that the events `START` and `REPAIR` of each machine can be enabled or disabled by a controller. Describe how they can be controlled based on the DES from (2), to ensure that all rules specified in (3) are satisfied. **[4p]**

Appendix: Installation of Truetime and Jitterbug

To be able to simulate the system in Part one, a Simulink toolbox called Truetime is used. In Truetime, the control algorithms are executed on a simulated real-time operating system. By doing this, the performance of the controller with respect to latency in the system can be evaluated.

Truetime Installation

32-bit Windows OS and Matlab 2011 (or higher) are recommended for this homework. Both softwares are free of charge from KTH program distributions like `MSDN` and `KTH ProgDist`. Otherwise please follow Section 2 of this Truetime Manual carefully before you continue.

Follow the instructions below to install Truetime within Matlab:

1. Unzip Truetime from the file `truetime-2.0.zip` and save it to any folder such as "`C:\TEMP\truetime-2.0`".

2. Execute the following commands at the Matlab command prompt:
```
>setenv('TTKERNEL','C:\TEMP\truetime-2.0\kernel')
>addpath([getenv('TTKERNEL')])
>init_truetime
>truetime
```

3. After executing the last line, check that you get the same window as in Figure 3.

Truetime Introduction

TrueTime is a Matlab/Simulink-based simulator for real-time control systems. It consists of seven blocks:

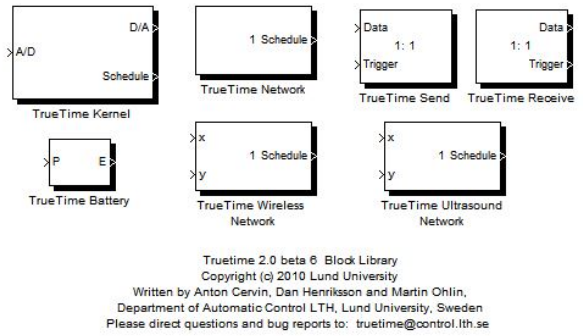

Figure 3: Truetime's Simulink blocks.

The first one, the one that this homework focuses on, is a computer block that simulates a computer running a real time operating system. The others are network blocks for simulation of networked systems, and will not be used in this homework.

The computer block executes user-defined task, i.e. control algorithms, and it is able to simulate the system using arbitrary scheduling policies. In Figure 3 the computer block (`TrueTime kernel`) is shown.

The ports used throughout this homework are the A/D, D/A and the schedule port. The input to the computer is connected to the A/D port of the block and the output to the D/A port. The schedule port shows how the different tasks are scheduled in the processor throughout the simulation.

The initialization script (`init_pend_three.m`) sets up the scheduling algorithm and creates the controller tasks. The data structure `data` contains measurement and control signals and values for the control algorithm. Open `init_pend_three.m` and try to understand the basics of the code. The scheduling policy and execution time can be modified there. Remember to save the script once you have changed something.

(*Note*: Deadline-monotonic scheduling (`'prioDM'`) is used in the code, which is the same as the Rate-monotonic scheduling in this example. Earliest Deadline First scheduling can be chosen by setting the `schedulingPolicy` to `'prioEDF'`).

The function (`pend_reg_three_code.m`) contains the code executed by different tasks. A code function is built up by a number of code segments executed in order by the kernel. Each segment has an execution time and next segment will not start until the time associated with the previous segment has elapsed in the simulation. Open `pend_reg_three_code.m` and try to understand the basics of the code.

Jitterbug

In order to run this simulation, you also need to install Jitterbug.

1. Unzip Jitterbug from the file `jitterbug-1.23.zip` and save it to any folder such as "`C:\TEMP\jitterbug-1.23`".
2. Add path (with subfolders) to the matlab path.
   `>addpath(genpath('C:\TEMP\jitterbug-1.23'))`

Now, you should be able to simulate the system in Part one by yourself!