LUND
UNIVERSITY

Department of
AUTOMATIC CONTROL

Real-Time Systems

Exam April 11th, 2018, hours: 08:00–13:00

Points and grades

All answers must include a clear motivation and a well-formulated answer. Answers may be given in **English or Swedish**. The total number of points is 25. The maximum number of points is specified for each subproblem.

Accepted aid

The textbooks Real-Time Control Systems and Computer Control: An Overview - Educational Version. Standard mathematical tables, authorized "Real-Time Systems Formula Sheet", and authorized "Reglerteknik AK Formula Sheet". Pocket calculator.

Results

The result of the exam will become accessible through LADOK. The solutions will be available on WWW:
http://www.control.lth.se/course/FRTN01/

1. Consider the following discrete-time system:

$$x(k+1) = \begin{pmatrix} -2 & 3 \\ 0 & 0.5 \end{pmatrix} x(k) + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u(k)$$
$$y(k) = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

a. Determine if the system is reachable or not. (0.5 p)

b. If possible, decide the values of l_1, l_2, and l_r in the state feedback law

$$u(k) = l_r y_{ref}(k) - l_1 x_1(k) - l_2 x_2(k)$$

so that the closed loop system has both its poles in 0.5 and the static gain 1. (1.5 p)

2. Consider the following task set that is executing under earliest-deadline-first scheduling on a uniprocessor, using the standard notation.

$Task_i$	T_i	D_i	C_i
A	4	4	2
B	3	2	1

a. What is the CPU utilization? (0.5 p)

b. Is the task set schedulable or not? (1 p)

3. Consider a continuous time system on state space form:

$$\dot{x} = \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix} x + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u$$
$$y = \begin{bmatrix} 0 & 1 \end{bmatrix} x$$

The system is ZOH-sampled with $h = 1$.

a. Determine the poles and zeros of the sampled system. Is the sampled system asymptotically stable? **Hint:** There are many ways to determine $H(z)$. (2 p)

b. State a difference equation that relates the input and output signals. (1 p)

4. Consider the following continuous time notch band-stop filter $G(s)$, specifically designed to block a disturbance $\omega_d = 10$ rad/s.

$$G(s) = \frac{s^2 + 2s + 100}{s^2 + 20s + 100}$$

a. Discretize the filter using forward difference. For what values of $h > 0$ is the discretized system asymptotically stable? **Hint:** The asymptotic stability conditions for a second-order discrete-time system with the characteristic polynomial $A(z) = z^2 + a_1 z + a_2$ are given by $a_2 < 1$, $a_2 > -1 + a_1$, $a_2 > -1 - a_1$. (1.5 p)

b. Assume that you instead approximate the filter using Tustin's method with sampling interval $h = 0.2$ s. What is the frequency ω_1 that is blocked by the filter? (1 p)

5. An incorrect implementation of a Boolean semaphore called `MutexSem` is given below. What is wrong with the code? Modify the code to get rid of the error. (2 p)

```
public final class MutexSem implements Semaphore {

  private Object lock;
  private boolean access;
  private Thread owner;

  public MutexSem {
    lock = new Object();
    access = true;
  }

  public void give() {
    synchronized (lock) {
      if (Thread.currentThread() != owner) {
        throw new SemViolation("MutexSem.give: Not owner!");
      };
      access = true;
      owner = null;
      lock.notify();
    }
  }

  public void take() {
    try {
      synchronized (lock) {
        while (!access) lock.wait();
      }
      owner = Thread.currentThread();
      access = false;
    } catch (InterruptedException exc) {
        throw new SemInterrupted("MutexSem.take interrupted: " + exc);
      }
  }
}
```

6. The NastyTM company has designed a software to control an industrial manipulator. The robot has three tasks to do, with the following characteristics. You have been asked to design a static cyclic scheduling policy for the tasks.

$Task_i$	T_i	D_i	C_i
A	3	3	1
B	6	6	3
C	9	9	2

a. How long should your schedule be – i.e., at what time does the schedule repeat itself? (0.5 p)

b. Design a static **non-preemptive** schedule, showing which task executes at what time. Compute the worst case response time with the given schedule, for each of the tasks. (1.5 p)

c. Write the pseudo-code for the dispatcher of the three tasks. The task dispatcher should give the correct timing also when the execution times are less than their worst-case values. (1 p)

7. Assume that you should implement the following control algorithm using a small 8-bit processor with no hardware support for floating point arithmetic.

$$x(k+1) = 0.8967x(k) + 0.2332y(k)$$
$$u(k) = -3.262x(k) - 0.8484y(k)$$

All variables (y, u, x) and all coefficients should be represented using 8-bit signed integers (`int8_t`). Intermediate results however use 16 bits (`int16_t`).

a. Select the number of fractional bits that gives the best possible resolution and convert the coefficients to fixed-point numbers. (Use the same number of fractional bits for all coefficients.) (1 p)

b. Write pseudo-C code, showing how the calculations in the control algorithm are implemented. Declare the word-length (size) of each variable used. Make sure that the x and u do not overflow - saturate them instead! In the code you may use `int8_t readInput()` and `writeOutput(int8_t u)`. You may assume that the value returned by `readInput()` and the argument to `writeOutput` use the same Q-format as you are using for your coefficients.

To get full points the code should be written so that the input-output delay of the controller is minimized and rounding should be used rather than truncation. (2 p)

8. For the coming summer you are in charge of a swimming pool with space for 5 persons. Your job is to ensure that there are never more than 5 persons in this pool at the same time. Seeing that you are an expert on Petri-net you think it would be a good idea to automate this job, allowing you to watch the world cup in soccer instead.

You figure that you should draw a Petri-net that models this system and prevents that more than 5 persons are in the pool at the same time. To do this, you think a bit and realize that you probably need the following places:

- person ready to enter the queue,
- people in queue (waiting to enter the pool),
- people in the pool,
- space in the pool,

and the following transitions:

- new person in queue,

- new person enters the pool,
- person leaving the pool.

Draw a Petri-net that captures this system using the places and transitions given above. You should initialize the system with no person being in the pool and no person being in the queue. (2 p)

9. Consider the feedback loop shown in Figure 1.

a. We have tried two different controllers

$$H_{C1}(z) = \frac{z - 0.75}{z - 0.15}, \qquad H_{C2}(z) = \frac{10 \cdot (z - 0.75)^2}{(z - 1) \cdot (z - 0.15)}$$

in order to control the plant

$$H_p(z) = \frac{0.01 \cdot (z + 0.95)}{(z - 0.95)^2}.$$

For each controller we show a step response of the closed-loop transfer function in Figure 2. Your task is to figure out (and motivate) which of the two step responses (A) and (B) correspond to which controllers ($H_{C1}(z)$) and ($H_{C1}(z)$).

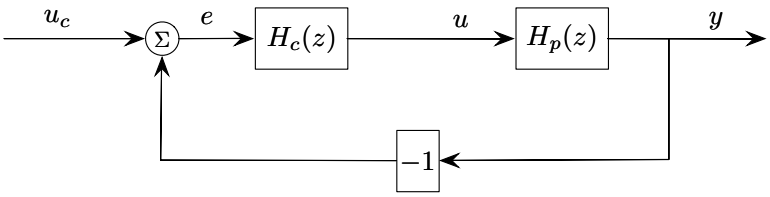

Figure 1

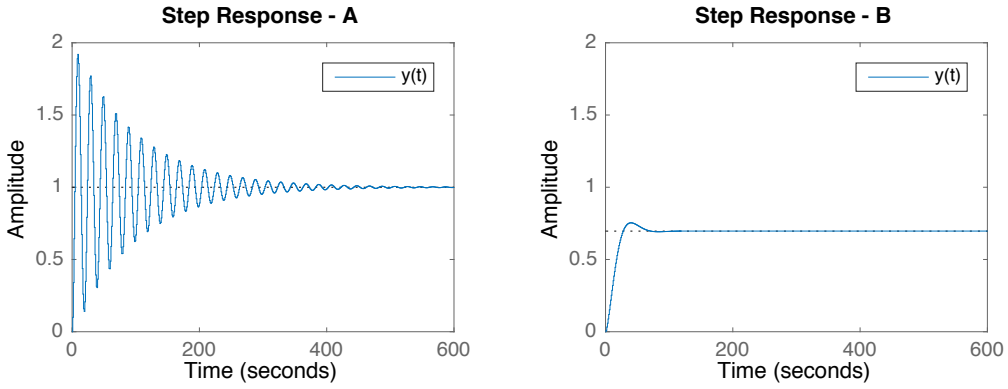

Figure 2 Step response for the two controllers, which step response corresponds to which controller?

(1.5 p)

b. To improve the system and make it faster one can use a 2-DOF controller, as shown in Figure 3. The plant for this system is the same as in **a**, i.e.,

$$H_p(z) = \frac{0.01 \cdot (z + 0.95)}{(z - 0.95)^2}.$$

When doing this, we have tried two different models

$$H_{m1}(z) = \frac{0.1 \cdot (z + 0.6)}{(z - 0.6)^2}, \qquad H_{m2}(z) = \frac{0.821 \cdot (z + 0.95)}{(z - 0.6)^2}.$$

For both models the feedforward block was chosen as $H_{\text{ff}}(z) = H_m(z)/H_p(z)$.

The controller for both cases was the controller corresponding to step-response A in the previous problem (although, it is not necessary to have solved **a** in order to solve this problem).

Your task is to figure out (and motivate) which of the two models $H_{m1}(z)$ and $H_{m2}(z)$ correspond to which step response (C) and (D) shown in Figure 3. Note that for each model we show both the step response for the output signal $y(t)$ (top plot) and the control signal $u(t)$ (bottom plot).

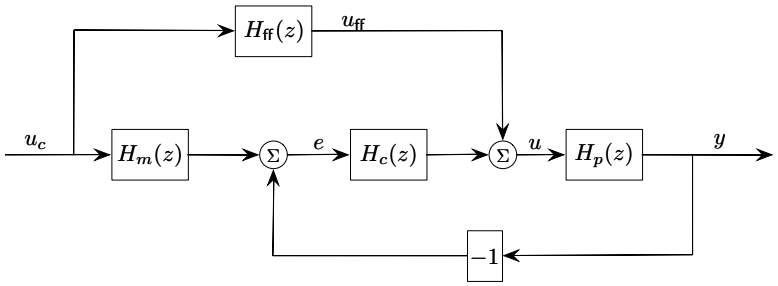

Figure 3 Model-based feedforward structure used in problem b.

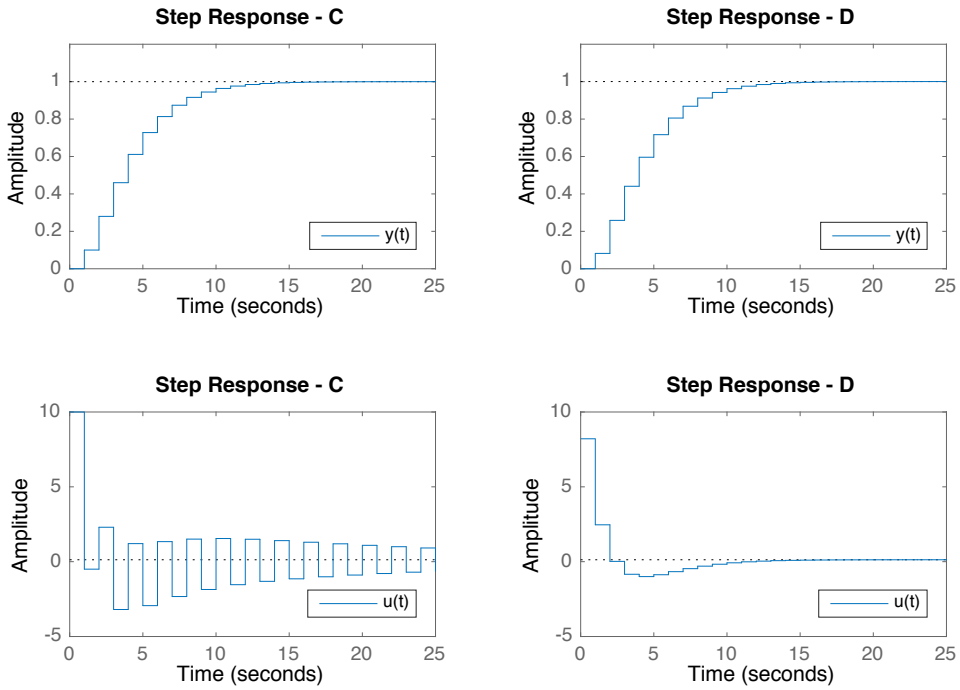

Figure 4 Step response for the two models. The up-most step response correspond to the output signal and the lower-most to the control signal.

(1.5 p)

10. You are a new employee at ReelThyme, a company creating pillows with nice scents ("doftkuddar" in Swedish). They are currently trying to control and stabilize two plants using a single thread. Unfortunately, it is not going very well for them and they are only able to stabilize one out of the two plants..

They have asked you to take a look at their controller implementation and explain to them why it doesn't work. They also want you to suggest a change to their code and/or their controller parameters so that it is possible to control the two plants using the single thread. The plants are given below:

$$P_1(s) = \frac{1}{s+1}, \qquad P_s(s) = \frac{10}{s+10}.$$

The run-method of the controller implementation is shown below in Listing 1.

a. Show why they are unable to control and stabilize both plants. (1.5 p)

b. Suggest a change to the code and/or the controller parameters (without changing the type of controller used) so that they are able to stabilize the two plants using the single thread. (1.5 p)

Listing 1 Controller code used by ReelThyme

```
public void run() {
  // assume all different variables
  // are initialized in a correct way

  h = 50; // 0.05 seconds
  i = 0;

  t = System.currentTimeMillis();
  while(WeShouldRun) {

    y1 = readInputFromPlant1();
    u1 = -10*y1; // compute control signal
    sendControlSignalToPlant1(u1); /* ZOH actuation */
    i = i+1;

    if (i==2) {
      y2 = readInputFromPlant2();
      u2 = -20*y2; // compute control signal
      sendControlSignalToPlant2(u2); /* ZOH actuation */
      i = 0;
    }

    t = t + h;
    duration = t - System.currentTimeMillis();
    if (duration > 0) {
      try {
        sleep(duration);
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }
  }
}
```