LUND UNIVERSITY

Department of
AUTOMATIC CONTROL

Real-Time Systems

Exam May 7, 2015, hours: 8.00–13.00

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 and authorized "Real-Time Systems 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. Given the following discrete-time system:

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

a. Calculate the pulse-transfer function, determine the poles and zeros, and determine whether the system is stable or not. (1 p)

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

2. The following system should be sampled with zero order hold with $h = L/3$:

$$G(s) = \frac{Ke^{-sL}}{sT+1}$$

a. Find the transfer function $H(z)$ of the sampled discrete-time system. (1 p)

b. Find the state space representation (Φ, Γ, C) of the sampled system. (1 p)

3. Given the following task set:

Task Name	T	D	C
A	10	10	2
B	6	6	2
C	8	8	2

a. Is the task set schedulable under Earliest Deadline First (EDF) scheduling? (1 p)

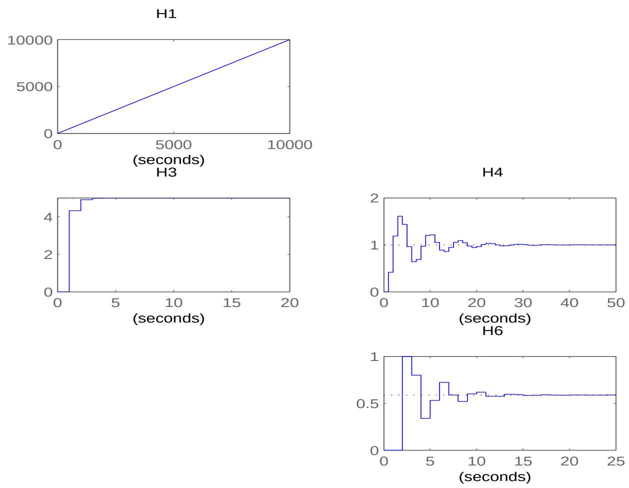
b. Is the task set schedulable under fixed priority scheduling using Rate Monotonic priority assignment? (1 p)

c. Assume now that the deadlines instead are $D_A = 6, D_B = 4$ and $D_C = 7$. Is the task set schedulable under fixed priority scheduling using Deadline Monotonic priority assignment? (2 p)

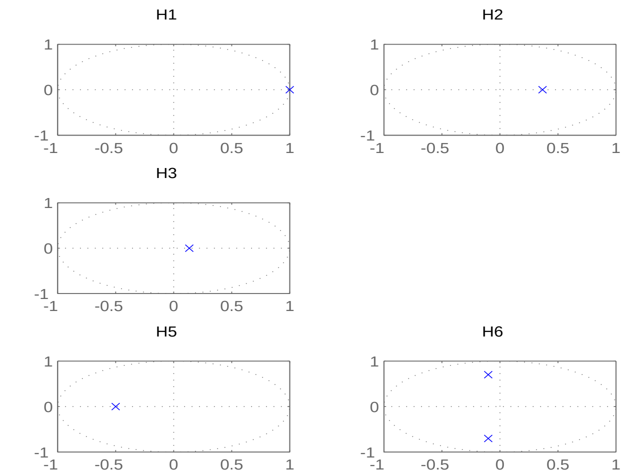
4. Figures 1a and 1b contain step responses and pole-zero maps of six different discrete-time systems (with some missing entries). Which of the systems $H_1(z), ..., H_6(z)$ could be a Zero-order-Hold sampled representation of a continuous, first-order system? Do not forget to motivate your answer! (2 p)

5.

a. A new engineer working in process industry got as her first task to implement a computer-based control system for a heat exchanger plant. The sampling rate was 50 Hz. When she studied the sampled signals she saw a disturbance signal with frequency 20 Hz. Since she had taken the Real-Time Systems course at LTH she immediately realized that this could be



(a) Step responses for problem 4



(b) Pole-Zero maps for problem 4

an aliasing effect. So she digged up an old analog oscilloscope and went out to search for the original disturbance signal. Which frequencies should she search for, i.e., which frequencies have 20 Hz as their fundamental aliasing frequency when sampled with 50 Hz? You may restrict the frequency range to 25 Hz - 500 Hz. (1 p)

b) The engineer does find a high-frequency signal that is aliased to 20 Hz and decides to implement a digital second-order deadbeat filter in order to remove the 20 Hz disturbance frequency, what is the problem with this approach? (1 p)

6. A linearized model of the single tank process is given by the transfer function

$$G_p(s) = \frac{\rho\tau_1}{1 + s\tau_1} \tag{1}$$

where ρ and τ_1 are process parameters which can be identified with two simple experiments. A PI controller in continuous time is given by

$$G_c(s) = K\left(1 + \frac{1}{sT_i}\right). \tag{2}$$

In the second lab in the Basic Control Course the identification experiment and pole placement in continuous time was performed. Since we assume that you already know how to do this, we have done it already and found the controller parameters $K = 4.3, T_i = 9.1$.

a. Approximate the controller given in (2) using a backward difference approximation (backward Euler). Assume that the sampling period $h = 1$. Answer with a *difference equation* relating the control signal u to the error signal e. (1 p)

b. We now want to implement the controller from subproblem **a)** using fixed point arithmetic in a 16 bit embedded CPU. Use as many fractional bits as possible. The controller should be written in C using the following code skeleton. Make sure all multiplications are done in 32 bits and that the error signal, control signal and all controller states are stored as 16 bit integers.

```
int16_t coeff1 = ...;
int16_t coeff2 = ...;
int n = ...;

int16_t do_control(int16_t e) {
  // Controller states
  static int16_t ...;

  // Control signal
  int16_t u = ...;

  // Update states
  ...

  return u;
}
```

(2 p)

7. Is the system

$$x(k+1) = \begin{bmatrix} 0.9 & 0 \\ 0.1 & 1.2 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1.1 \end{bmatrix} u(k)$$

$$y(k) = \begin{bmatrix} 1.0 & 0.5 \end{bmatrix} x(k)$$

unstable, stable or asymptotically stable? Is the system controllable? If it is not asymptotically stable, is it possible to make it asymptotically stable using state feedback, $u(k) = -L\,x(k)$? (2 p)

8. The following code is supposed to be a buffer in a producer/consumer system.

```java
public class BufferError {
    // This class is broken
    private double data;
    private boolean hasData = false;

    public void put(double d) throws InterruptedException {
        synchronized (this) {
            while (hasData) {
                wait();
            }
        }

        hasData = true;
        data = d;
        notify();
    }

    public double get() throws InterruptedException {
        synchronized (this) {
            if (!hasData) {
                wait();
            }

            hasData = false;
            notifyAll();
            return data;
        }
    }
}
```

One requirement is that it has to support several producer threads (calling put) and several consumer threads (calling get).

The code contains at least one concurrency error. Explain the error(s) and how it/they should be remedied. (2 p)

9.

a. Show that for zero-order hold sampling the following equality does not hold

$$H_{G_1G_2}(z) = H_{G_1}(z)H_{G_2}(z)$$

where $H_{G(s)}(z)$ is the pulse transfer function of $G(s)$. (1 p)

b. Give an intutitive explanation why the above is does not hold. (1 p)

c. However, for zero-order hold sampling the following equation does hold $H_{G_1+G_2}(z) = H_{G_1}(z) + H_{G_2}(z)$, i.e., when the two subsystems are connected in parallel then we can simply sum the individual pulse transfer functions. Use this to calculate the zero-order hold pulse transfer function for the following continuous-time state-space system (Hint: Study the structure of the matrices).

$$\frac{dx}{dt} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} u$$
$$y = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix} x$$

(2 p)

10. When doing the Buttons exercise in Computer exercise 2 some students presented the following solution to the Regul thread (minor details omitted). The program compiled OK but when they they started the application the controller did not appear to be executing.

```
public class Regul extends Thread {

  public void Run() {
    final long h = 100; // period (ms)
    try {
      while (!Thread.interrupted()) {
        // Get inputs
        double y = yIn.get();
        double r = refMon.getRef();
        double K = paramMon.getK();

        // Compute control signal
        double u = 0.0;
        if (isOn()) {
          u = K * (r - y); // P controller
        }

        // Set outputs
        uOut.set(u);
        rOut.set(r);

        Thread.sleep(h);
        }
      } catch (InterruptedException e) {
      // Requested to stop
      }
      System.out.println("Regul stopped.");
    }
}
```

a. Explain what the problem was and why nothing happened. The error could be related to concurrency and/or to other issues with the code. (1 p)

b. After a while the student modified the code as follows, which helped them to find the error. Explain what happened in this case? (1 p)

```
public class Regul implements Runnable {

  public void Run() {
    final long h = 100; // period (ms)
    try {
      while (!Thread.interrupted()) {
        // Get inputs
        double y = yIn.get();
        double r = refMon.getRef();
        double K = paramMon.getK();

        // Compute control signal
        double u = 0.0;
        if (isOn()) {
          u = K * (r - y); // P controller
        }

        // Set outputs
        uOut.set(u);
        rOut.set(r);

        Thread.sleep(h);
        }
      } catch (InterruptedException e) {
      // Requested to stop
      }
      System.out.println("Regul stopped.");
    }
}
```