LUND UNIVERSITY

Department of
AUTOMATIC CONTROL

Real-Time Systems

Exam May 3, 2019, 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" plus the formula sheet from Reglerteknik AK. 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

$$x(k+1) = ax(k) + u(k)$$

a. For which values of a can the above system have been obtained from a ZOH-sampling of a first-order continuous-time system? (1 p)

b. Assume that $a = 0$. What is the name for a system with this type of dynamics? (1 p)

2. A fixed-point representation of the first-order digital filter

$$\begin{aligned} x(k+1) &= 0.78x(k) - 1.23y(k) \\ u(k) &= 0.91x(k) \end{aligned}$$

with word length $N = 8$ and 6 fractional bits is shown below:

```
int8_t x = 0, y, u;
y = readInput();
u = (int8_t)(((int16_t)58*x) >> 6);
writeOutput(u);
x = (int8_t)(((int16_t)50*x - (int16_t)79*y) >> 6);
```

The variables y, x, and u are all represented as 8-bit signed integers (i.e. 0 fractional bits).

a. What would the consequence be if instead the following implementation was used?

```
int8_t x = 0, y, u;
y = readInput();
u = (58*x) >> 6;
writeOutput(u);
x = (50*x - 79*y) >> 6;
```

(1 p)

b. What would the consequence be if instead the following implementation was used?

```
int8_t x = 0, y, u;
y = readInput();
u = (int8_t)(((int16_t)58*x) >> 5);
writeOutput(u);
x = (int8_t)(((int16_t)50*x - (int16_t)79*y) >> 6);
```

(1 p)

3.

a. Consider the following task set where the standard notation is used. Assign the priorities *High*, *Medium* and *Low* to the three tasks when fixed-priority scheduling with deadline-monotonic priority assignment is used.

Task	T	C	D
A	6	1	5
B	4	2	4
C	3	1	2

(0.5 p)

b. Decide if the task set is schedulable or not using fixed-priority scheduling with deadline-monotonic priority assignment. (1.5 p)

c. Now we instead assume that Earliest Deadline First scheduling should be used. Decide if the task set is schedulable or not. In order to get any points on the problem you may only use methods and results that are part of this course to derive the result. (2 p)

4. When we teach PID control we propose to discretize the three terms (P, I and D) separately. The main reason for this is that it increases the transparency of the algorithm. However, this is not the only way to do it.

a. Consider the following PI controller:

$$U(s) = (K + \frac{K}{T_I s})E(s)$$

Discretize the transfer function above *as a single transfer function* using the Backward Euler approximation, i.e., the two terms should not be discretized individually. Write the result as a difference equation involving the control signal u and the error e and with $u(k)$ only, on the left hand side of the equality sign. (Hint: Start by writing the transfer function with a single fraction bar.) (1 p)

b. Assume now that the actuator used is limited between u_{max} and u_{min}. What unwanted behavior might occur if the above discretized controller is used? (1 p)

c. Modify the controller so that the problem in **b.** is solved. (1 p)

5. You are part of a film crew doing a documentary on old propeller planes. At one time, you set up your camera (recording 60 frames per second) and film a plane starting its engine. You watch as the propeller starts to turn with increasing speed, until it reaches a constant angular rate.

Curious, you ask the pilot how fast the propeller was turning when the angular rate was constant, but he only remembers it was somewhere between 2000 and 3000 revolutions per minute (rpm).

a. You watch the film and count the number of revolutions during a short time period. The estimate you get is 1200 rpm. Assuming that the pilot is correct, what is the likely cause for this mismatch? What could have been done differently to ensure that the film would show the correct angular rate of the propeller? (1 p)

b. Assuming that the pilot is correct, what was the exact angular rate of the propeller? (1 p)

6. Consider the following linear system

$$x(k+1) = \begin{pmatrix} 0.5 & 1 \\ 0.5 & 0.7 \end{pmatrix} x(k) + \begin{pmatrix} 0.2 \\ 0.1 \end{pmatrix} u(k).$$

a. Assume state-feedback and design a dead-beat controller for this system. (1 p)

b. Assume that $x(0) = \begin{pmatrix} a & b \end{pmatrix}^\intercal$. For what values of of a and b will the magnitude of $u(0)$ using the dead-beat controller from **a.** be less than or equal to 10? Draw a figure where the values of a and b that satisfies the inequality are clearly marked out. (1 p)

7. For a linear system of the form

$$\begin{aligned} x(k+1) &= \Phi x(k) + \Gamma u(k) \\ y(k) &= Cx(k) \end{aligned}$$

we can design an observer with direct term given by the following system of difference equations:

$$\hat{x}(k) = \Phi\hat{x}(k-1) + \Gamma u(k-1) + K[y(k) - C(\Phi\hat{x}(k-1) + \Gamma u(k-1))]$$

In this problem we will consider the case when Φ, Γ and C are given by

$$\Phi = \begin{pmatrix} 1 & 0.2 \\ 0 & 1 \end{pmatrix}, \quad \Gamma = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & -1 \end{pmatrix}$$

a. Compute the equations governing the reconstruction error $\tilde{x} = x - \hat{x}$ for the case when $K = \begin{pmatrix} 2 & 4 \end{pmatrix}^\intercal$. Will the reconstruction error converge to zero? (1.5 p)

b. Determine K such that the observer poles are placed in $z = 0$ and $z = 0.2$. (1 p)

c. Let $K = \begin{pmatrix} k_1 & k_2 \end{pmatrix}^\intercal$. Find a relation between k_1 and k_2 that ensures that the output y of the system is estimated with zero error at all times. (1.5 p)

8. Match the step responses shown in Figure 1 with the pulse transfer functions below:

$$\begin{aligned} H_1(z) &= \frac{0.3}{z-0.7} \\ H_2(z) &= \frac{1.96}{z^2+0.8z+0.16} \\ H_3(z) &= \frac{0.3}{z^2-0.7z} \\ H_4(z) &= \frac{0.34}{z^2-1.4z+0.74} \\ H_5(z) &= \frac{0.3z}{z-0.7} \\ H_6(z) &= \frac{1.4}{z+0.4} \end{aligned}$$

Each pulse transfer function corresponds to one step response, and it is assumed in all cases that

$$y(-1) = y(-2) = 0, \quad u(k) = \begin{cases} 0, & k < 0, \\ 1, & k \geq 0. \end{cases}$$

Correct motivations for each matched pair are required for full points. (Hint: Some numerical calculations may be needed to solve this.) (3 p)

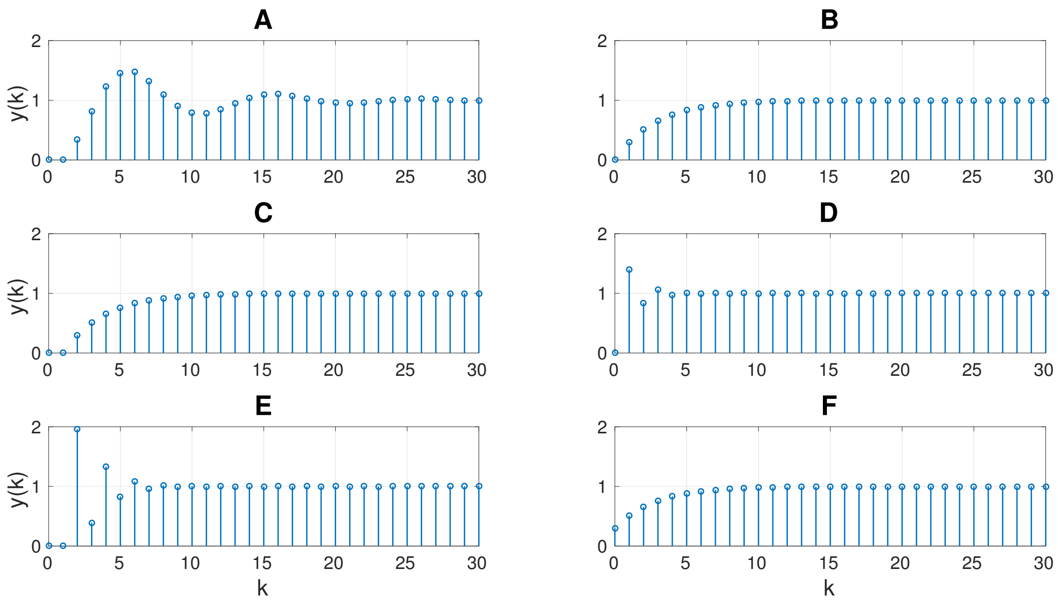

Figure 1 Step responses in Problem 8.

9. Java contains a built-in synchronization mechanism called a *barrier*. Assume that we have a number of threads executing a part of an overall application followed by a point at which the threads must coordinate their results. The barrier is simply a waiting point where all the threads can sync up to either merge results or to safely move on to the next part of the application.

The Java class implementing barriers is called `CyclicBarrier`. The reason it is called *cyclic* is that it can be re-used after the waiting threads have been released.

A slightly simplified version of the interface to `CyclicBarrier` is:

```
public class CyclicBarrier {

public CyclicBarrier(int parties);

public int await();

}
```

The core of the class is the `await()` method. This is called by each thread that needs to wait until the required number of threads are waiting at the barrier. In the constructor the number of threads (parties) using the barrier is specified. This number is used to trigger the barrier; the waiting threads are all released when the number of threads waiting on the barrier is equal to the number of parties. The latter includes the thread that caused the barrier to be triggered.

Each thread that calls the `await()` method gets back a unique return value. This value is related to the arrival order of the thread at the barrier. The first thread that arrives gets a value that is one less than the number of parties and the last thread to arrive will get a value of zero.

The barrier is very simple. All the threads wait until the number of required parties arrive. Upon arrival of the last thread, the waiting threads are released, and the barrier can be re-used. Since the barrier is so simple it is straightforward to implement it using the Java synchronization mechanisms that are part of the course (`synchronized`, `wait()`, `notify()` and `notifyAll()`).

Your task is to implement the class `CyclicBarrier` with the interface and semantics described above. In order to get full points you must ensure that all threads that are released really will be released, also if some other thread has started to re-use the barrier before the released threads have executed. Correct handling of exceptions is not required. You may also disregard any spurious wake-up issues. (3 p)