LUND INSTITUTE
OF TECHNOLOGY
Lund University

Department of
AUTOMATIC CONTROL

Real-Time Systems

Exam January 13, 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 dynamic system

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

a. Verify that the system is observable. (0.5 p)

b. Design the deadbeat observer (predictor form) for the system. (1 p)

2. Given the following piece of code

```
x = 12.4;
y = 0.4;
z = x/y;      // exact value of z = 31.0
```

Compute the value of z with a fixed-point implementation with $N = 8$ bits, and a $Q5.2$-format. (1 p)

3. For simulation purposes the continuous-time system

$$\dot{y} = y + u.$$

is to be discretized with the backward Euler method. Why is this not a good idea for the sampling period $h = 3$? (1 p)

4. Match the following transfer functions to their pulse responses in Figure 1. Also determine the unknown, *positive* constant p. Do not forget to properly motivate your answer. (2 p)

$$G_A = \frac{z}{z+p} \qquad G_B = \frac{1}{z^2+p}$$
$$G_C = \frac{1}{z} \qquad G_D = \frac{1}{z+p}$$
$$G_E = \frac{z}{z^2-p} \qquad G_F = \frac{1}{z-p}$$

5. Consider the following task set.

Task	T	D	C
A	4	4	2
B	3	3	1
C	6	6	0.5

a. Using fixed-priority scheduling and rate-monotonic priority assignment, verify whether the the task set is schedulable or not. (1 p)

b. Using EDF scheduling, verify whether the task set is schedulable or not. (0.5 p)

c. Again assuming EDF scheduling, draw the schedule for the three tasks over one hyperperiod for the case when the task execution times equal their worst-case values. Assume that the tasks are all released simultanelously at $t = 0$ (the critical instant). The EDF scheduler is implemented in such a way that when there is a deadline tie, priority will be given according to the task rate, i.e., a task with a short period will have priority over a task with a long period.

(1 p)

d. What is the worst-case response times for the three tasks with the same scheduling as in subproblem c? (0.5 p)

6. Given the system:

$$y(k+2) + 0.4y(k+1) = u(k)$$

a. For which positive values of K in the proportional controller

$$u(k) = K(r(k) - y(k))$$

is the closed loop system stable? (1 p)

b. Determine the stationary error $r - y$ when r is a unit step and $K = 0.5$. (1 p)

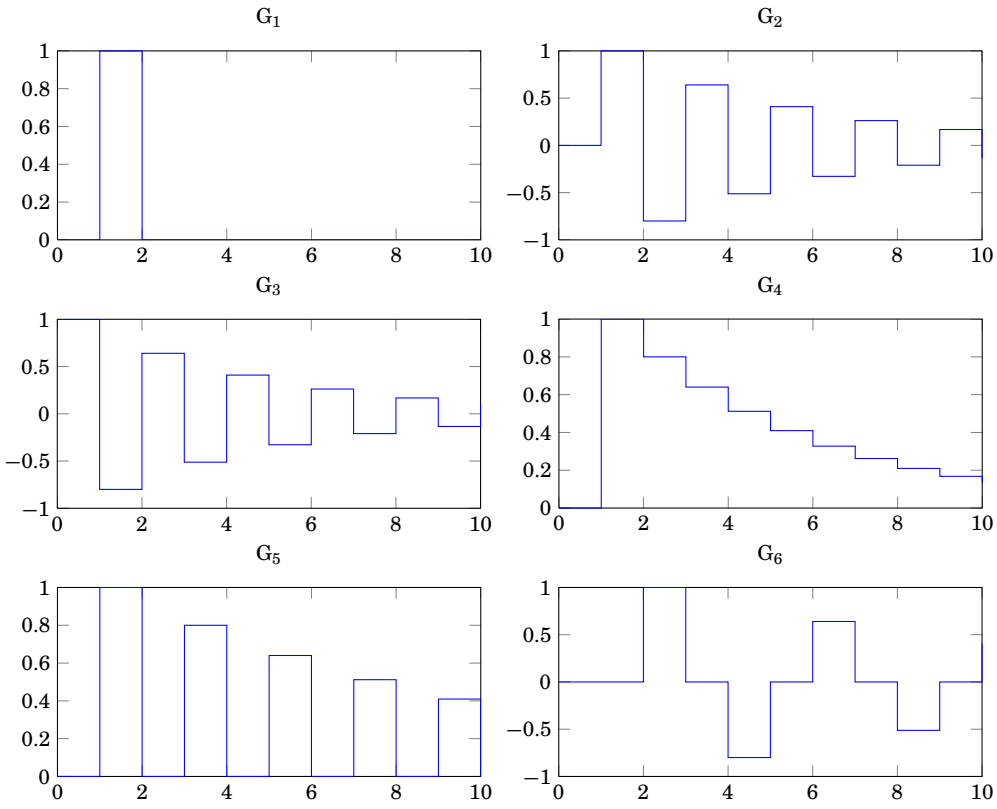

Figure 1 Pulse responses for Problem 4

.

c. In order to get rid of stationary errors a PI controller could be used, find the pulse transfer function of the PI controller when discretizing the I part using forward Euler. Show that the static gain from r to y with a PI controller is 1. ($K = 0.1,\ T_i = 1,\ h = 1$ will give a stable closed loop). (2 p)

7. Inertial Measurement Units (IMU) often contain both gyroscopes and accelerometers for positioning purposes. For the purpose of pose estimation, the gyroscope can be assumed to measure angular velocities and the accelerometer to measure an angle.

Gyroscopes exhibit desirable high frequency properties, but suffer from low frequency drift. Accelerometers on the other hand, often suffer from high frequency noise, but exhibit no drift. A common method to fuse measurements from the two sensors is a complementary filter. The idea behind the filter is to obtain an estimate of the angle by combining a low-pass filtered value of the accelerometer signal with a high-pass filtered value of the integrated gyroscope signal. The result typically obtained can be seen in Figure 2.

The simple complementary filter is characterized by two transfer functions, $G_g(s)$ and $G_a(s)$, and the relations

$$1 = G_g(s) + G_a(s) \tag{1}$$

$$Y_f(s) = G_g(s)Y_g(s) + G_a(s)Y_a(s) \tag{2}$$

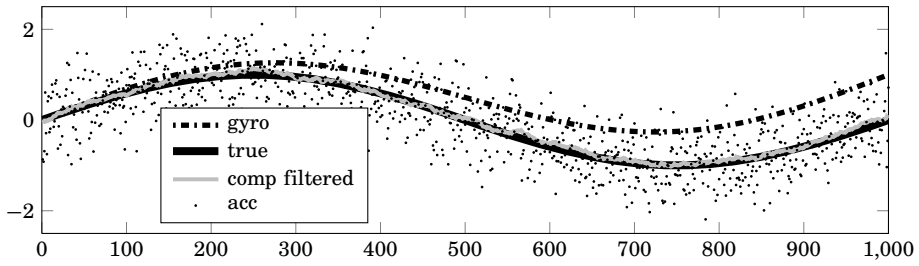

Figure 2 Illustration of how the measurements from an accelerometer and a gyroscope are fused using a complementary filter.

It has been determined that a suitable low-pass filter for the accelerometer measurements is given by

$$G_a(s) = \frac{1}{s+1}$$

Your job is to

a. Design the complementary high-pass filter for the gyroscope measurements, $G_g(s)$, such that Eq.(1) is satisfied. Assume that the signal from the gyroscope is pre-integrated, such that the gyroscope returns an angle measurement. (0.5 p)

b. Sample the two filters using zero order hold. The filters are to be implemented with the sample time h. (1 p)

c. Implement the discrete time filters obtained in **b.** in the pseudo code skeleton below. The output `yf` is the filtered signal, the inputs `ya` and `yg` are the measurements from the accelerometer and the (pre-integrated) gyroscope respectively. A variable declared `static` will keep its value between two consecutive calls to the function.

```
double compFilt(double ya, double yg, double h){
        /* Declare and initialize variables */
        static yf = 0; // Filtered signal
        ... Your code here ...

        /* Perform calculation */
        ... Your code here ...

        return yf;
}
```

(2.5 p)

8.

a. The signals f_1 and f_2

$$f_1(t) = a_1 \sin(\pi t),$$

$$f_2(t) = a_2 \sin(3\pi/2t)$$

are sampled. Which frequencies are there at the output if the sampling period is $h = 1$? (1 p)

b. A signal with frequency content between 0 and 250 Hz is going to be sampled. However the signal is corrupted by a disturbance sinusoid at 510 Hz. The solutions I and II to extract the original signal without aliasing are suggested:

I) Sample the signal at 1000 Hz and then filter the signal using a digital filter with cut-off frequency at 400 Hz.

II) Use an analog filter with cutoff frequency at 400 and then sample the signal at 250 Hz.

Explain which one of the solutions is best and which one is wrong and why. (1 p)

9. Consider the block diagram given by Figure 30 (pg 70) in IFAC Professional Brief where the model and feedforward generator block is given by Figure 31.

a. Assume that the observer is chosen on predictor form. Write pseudo-code for the calculations that should be implemented in the controller, i.e. the calculations needed to implement the feedback, the observer, and the model and feedforward generator block. Write the code so that the input-output latency is minimized, i.e. split up the code in two parts: CalculateOutput and UpdateState, where the amount of code in CalculateOutput is minimized, and do all possible pre-calculations in UpdateState. Your pseudo-code may contain matrix expressions involving both scalar and vector variables, e.g., it is OK to write $u = -Lx$ as a single statement.

The process is given by

$$x(k+1) = \Phi x(k) + \Gamma u(k)$$
$$y(k) = Cx(k)$$

(1 p)

b. Do the same thing as in the previous sub-problem, but now for an observer on filter form (with direct term). (2 p)

10. The readers-writers problem where we require that writer processes have exclusive access to the critical section and that there may be at maximum three simultaneous reader processes can be modeled using the generalized Petri net shown in Figure 3.

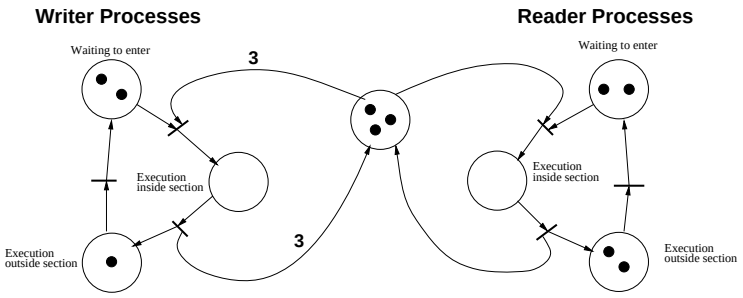

Figure 3 Readers-writers problem modeled with Petri net.

a. Show how readers-writer synchronization can be implemented in Java using a multi-step quantity semaphore (a counting semaphore that permits increments and decrements that are greater than one).

The methods available for a multi-step semaphore are the following:

```
// Constructor that initializes the internal counter to 0
public MultiStepSemaphore();

// Constructor that initializes the internal counter to n
public MultiStepSemaphore(int n);

// Causes the calling thread to block until the counter assumes a value
// greater than n. Before returning the counter is decremented with n.
public void take(int n);

// Equal to take(1);
public void take();

// Increments the counter by n and releases that many blocked threads,
// if any.
public void give(int n);

// Equal to give(1);
public void give();
```

Use the following code structure as the starting point and fill in the missing parts in the `Writer`, `Reader`, and `Main` classes. It is only the syncronization-related code that you need to add.

```
public class Writer extends Thread {

  MultiStepSemaphore sem;

  public Writer(MultiStepSemaphore s) {
    sem = s;
  }

  public void run() {
    while (true) {

    // access critical section

    }
  }
}

public class Reader extends Thread {

  MultiStepSemaphore sem;

  public Reader(MultiStepSemaphore s) {
    sem = s;
  }

  public void run() {
    while (true) {

    // access critical section

    }
  }
}

public class Main {

  public static void main(String[] args) {

  MultiStepSemaphore s;
  Writer w;
  Reader r;
  for (int = 1; i==4; i++) {
    w = new Writer(s);
    w.start();
  }
  for (int i = 1; i==4; i++) {
    r = new Reader(s);
    r.start();
  }
}
```

(1 p)

b. In Figure 4 the solution to an extended readers-writers problem is shown where we also allow two writer processes in the critical section at the same time. Still we may have at most three simultaneous reader processes inside the critical section. Also, we do not allow readers and writers to be in the section at the same time.

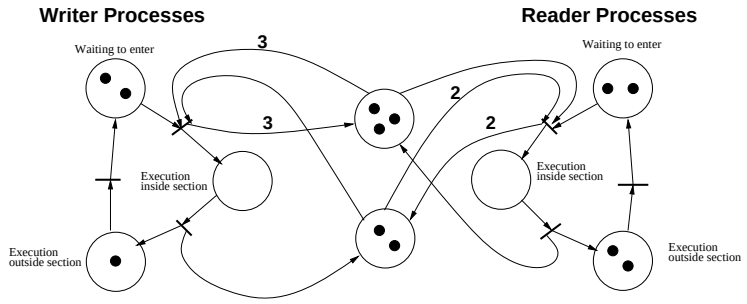

Figure 4 Extended readers-writers problem modeled with Petri net.

Implement the class `ReadersWritersGuard` with the following interface and show using the code selection in the first subproblem how the class should be used. You may exclude all exception handling code (try, catch). You mau not use the `MultiStepSemaphore` class from subproblem a in your solution.

```
// Constructor that takes the maximum number of simultaneous writers
// and readers as arguments
public ReadersWritersGuard(int maxWriters, int maxReaders);

// Equivalent to public ReadersWritersGuard(1,1)
public ReadersWritersGuard();

// Called by a writer process before entering the critical section.
// Blocks if buffer is full of readers or writers
public synchronized void writersTake();

// Called by a reader process before entering the critical section.
// Blocks if buffer is full of readers or writers
public synchronized void readersTake();

// Called by a writer process before exiting the critical section.
// Releases waiting readers and/or writers.
public synchronized void writersGive();

// Called by a reader process before exiting the critical section.
// Releases waiting readers and/or writers.
public synchronized void readersGive();
```

Readers and writers may be considered as equally important when accessing the critical section, i.e., it is not so that writers should have priority over readers or vice versa when both writers and readers are waiting for access to the critical section.

(2.5 p)