LUND INSTITUTE
OF TECHNOLOGY
Lund University

Department of
AUTOMATIC CONTROL

Real-Time Systems

Exam January 7, 2014, hours: 14.00–19.00

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. Consider the following continuous-time triple integrator process with output y and input u

$$\dddot{y}(t) = u(t)$$

a. Introduce suitable continuous-time state variables and ZOH-sample the process with $h = 1$. (1 p)

b. Compute the pulse transfer function for the discrete-time system derived in subproblem a and determine its poles and zeros. (Hint: The general expression for the inverse of an upper-triangular 3x3 matrix can be found on the last page of the exam.) (1 p)

2. Consider a discrete-time PI controller without any setpoint weighting ($\beta = 1$).

a. What is the pulse transfer function for the PI controller when the I part is discretized using a forward difference approximation? What are the poles and zeros? (1 p)

b. Implement the PI-controller using the following pseudo-code skeleton. In order to get full points the code should be written so that the input-output latency is minimized. Insert code at the places indicated by ..... Make sure that all variables are properly declared. The code need not contain any anti-windup mechanism. (1 p)

```
private double u = 0, y = 0, r = 0;
...

while (1) {
  y = getY();
  r = getReference();
  ....
  outputU(u);
  ....
  sleep()
}
```

3. A moving object with external force u and position y is described by the discrete-time state-space system

$$x(k+1) = \begin{pmatrix} 0 & -0.8 \\ 1 & 1 \end{pmatrix} x(k) + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u(k)$$
$$y(k) = \begin{pmatrix} 0 & 0.5 \end{pmatrix} x(k)$$

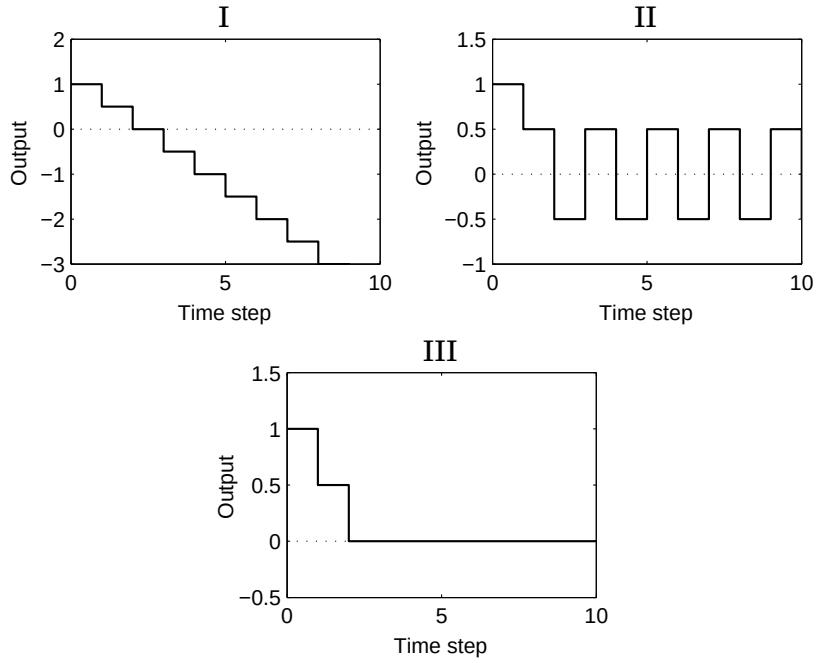
Three different state feedback controllers

$$u(k) = -Lx(k)$$

have been designed using pole placement. Match the corresponding feedback vectors L_1–L_3, pole placements λ_A–λ_C and initial condition responses I–III. All answers should be carefully motivated. (2 p)

$$L_1 = \begin{pmatrix} -1 & -0.8 \end{pmatrix} \quad L_2 = \begin{pmatrix} 2 & 1.2 \end{pmatrix} \quad L_3 = \begin{pmatrix} 1 & 0.2 \end{pmatrix}$$

$$\lambda_A = \begin{pmatrix} -1 & 0 \end{pmatrix} \quad \lambda_B = \begin{pmatrix} 0 & 0 \end{pmatrix} \quad \lambda_C = \begin{pmatrix} 1 & 1 \end{pmatrix}$$



4. A control system is running with a sampling time of $h = 0.02$ s. A measurement disturbance enters the control loop at the sensor, causing disturbances in the control signal.

What will be the frequency/frequencies of the disturbances in the control signal (before the D/A converter) if

a. the measurement disturbance consists of two sinusoidal components with the frequencies 20 Hz and 550 Hz? (1 p)

b. the measurement disturbance contains frequencies between 20 Hz and 30 Hz? (1 p)

5. A development team, UNI-X, at an embedded systems company was developing an application that contained three independent tasks: A, B, and C. The application was going to be hosted on a processor which used fixed priority scheduling. On the same processor a number of other applications were executing, each of them containing a number of independent periodic tasks with deadlines equal to the task periods.

The CPU resources were divided so that 39% of the CPU utilization was reserved for the tasks in the other applications. The parameters of UNI-X's tasks are the following.

Task name	T_i	D_i	C_i
A	6	6	0.5
B	x	x	1
C	12	12	1

One design goal of UNI-X is to minimize the period, x, of task B.

a. What is the minimum task period for task B in order to guarantee that the entire task set on the processor is schedulable under rate-monotonic priority assignment and under the assumption that the kernel is ideal. (1 p)

b. The achievable task period for task B was, however not enough and after a lot of political maneuvers UNI-X were allowed to implement their tasks on a separate processor. One of the members of the team had heard about the fantastic EDF scheduling method. What is the minimum task period for task B that can be achieved with EDF scheduling? (1 p)

c. However, the team soon found out that EDF scheduling is not so well supported in commercial real-time kernels. Therefore it was decided that the fixed priority scheduling must be used after all. It was decided to choose the task period of B so that the utilization of the CPU was 5/6 to have some safety margin. What task period does this correspond to? (1 p)

d. Verify that the task set is schedulable with the task periods from subproblem c. (1 p)

6. Consider the following first-order continuous-time system

$$\frac{dx(t)}{dt} = -ax(t) + au(t-\tau)$$
$$y(t) = x(t)$$

a. Calculate the pulse transfer function for the ZOH-sampled system when $\tau = 0$. (0.5 p)

b. Calculate the pulse transfer function for the ZOH-sampled system when $\tau = h$. (0.5 p)

c. Calculate the pulse transfer function for the ZOH-sampled system when $0 < \tau < h$. (0.5 p)

7. In Fig. 1 you can see a state machine. Convert the state machine into a Grafcet diagram. (2 p)

8. In Laboratory 1 some students suspected that there was an error in the GUI which sometimes caused incorrect PID parameters to be set. Thus, to be safe, the `PID` class was modified so that requested parameter changes must be approved by the user before they were applied:

```
// PID.java
public PID(...) {
  // ...
  setParameters(p);
}
public synchronized double calculateOutput(double y, double r) {
  // ...
}
```

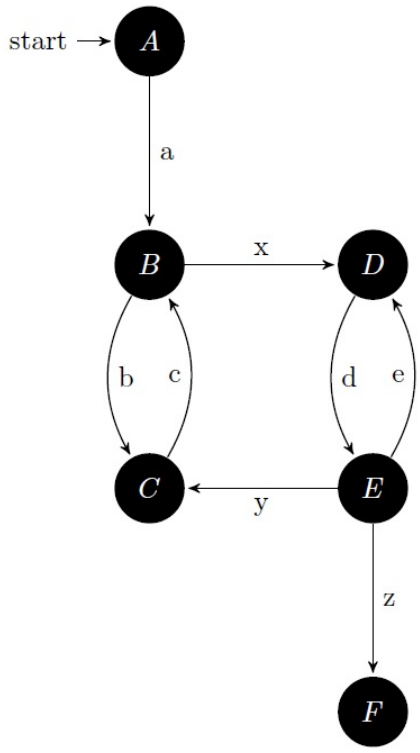


Figure 1 State machine

```
public synchronized void updateState(double u) {
  // ...
}
public synchronized void setParameters(PIDParameters p) {
  System.out.println("Got new parameters:");
  printParameters();
  System.out.println("Apply these parameters? (Y/N)");
  if (getUserConfirmation()) {
    this.p = (PIDParameters)p.clone();
    // ... update ad and bd
  }
}
```

a. An unexpected side effect of the new implementation is that the controller thread (Regul) sometimes does not work. When and why? Modify the code to fix this issue. (1.5 p)

b. Mention at least two additional issues introduced by the new `setParameters()` (possibly related to the fix in the previous problem) and discuss possible solutions. (1 p)

9. We are writing C code using fixed point variables. The variable `fixX` contains the value of x with `nx` fractional bits, the variable `fixY` contains the value of y with no fractional bits and the variable `fixZ` contains the value of z with `nz` fractional bits. We know that `nx >= nz` is true.

We want to calculate the value of $x \cdot y + z$ with zero fractional bits. For each of the following snippets of C code, say if it will calculate the value correctly

or not. If not, motivate your answer.

A. `(fixX * fixY + (fixZ << (nx - nz)) >> nx`

B. `(fixX >> nx) * fixY) + (fixZ >> nz)`

C. `(((fixX * fixY) >> (nx - nz)) + fixZ) >> nz`

D. `(((fixX * fixY) >> nx) + fixZ) >> nz`

For a calculation to be considered correct we want to keep all the precision, but we are fine with always rounding the final value down. You may assume that there is always enough bits for intermediate values and final results.

(2 p)

10. A Petri net is deadlock-free if there is always at least one transition that may be fired. A Petri net is live if no transition can become unfireable. A Petri net is bounded of there is an upper bound on the number of tokens in the net. A Petri net that is live is also deadlock-free. A Petri net that is not deadlock-free is is not live either.

For each of the Petri nets in Fig. 2 determine if it is a syntactically correct Petri net? If it is syntactically correct then also decide if it is deadlock-free, if it is live, and if it is bounded? Motivate your answers briefly.

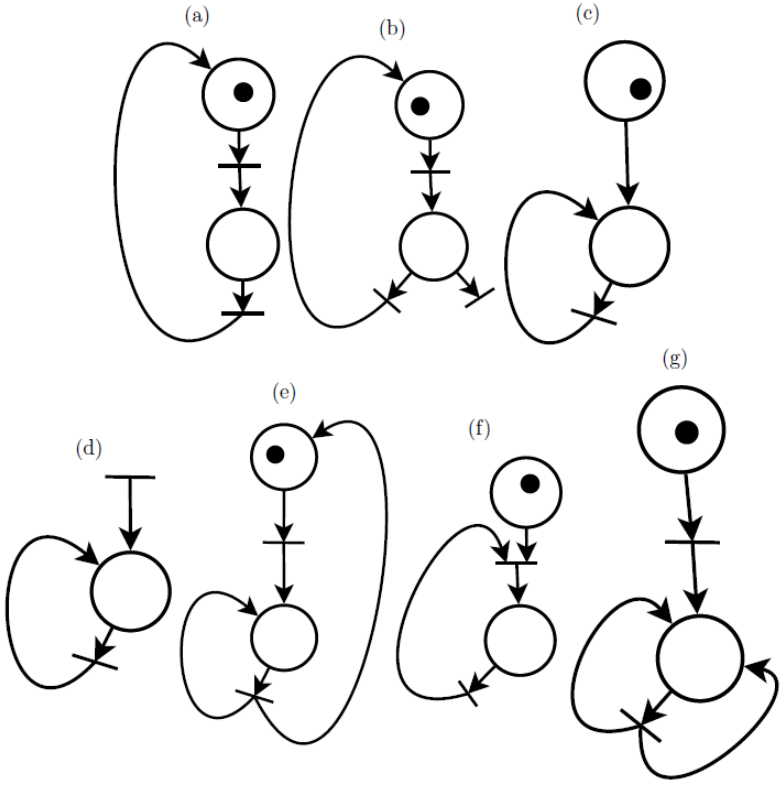


Figure 2 Petri nets

(3 p)

11. In `java.util.concurrent` a number of synchronization mechanisms are available. One of them is the *countdown latch*. A countdown latch is a synchronization mechanism that allows one or more threads to wait until a set of operations being performed in other threads completes.

The countdown latch is implemented by the Java class `CountDownLatch`. A `CountDownLatch` is initialized with a given *count*. The `await()` method causes the calling thread to wait until the current value of count is equal to zero. The value of count is decremented using the method `countDown()`. When the value reaches zero all waiting threads are released. The latch may not be reset, i.e., further attempts to decrement the counter once it has reached zero, will have no effect and calls to `await()` once the latch has triggered will return immediately. The method `getCount()` returns the current value of count.

A `CountDownLatch` can be used for a number of purposes. A `CountDownLatch` initialized with a count of one serves as a simple on/off latch, or gate: all threads invoking `await()` wait at the gate until it is opened by a thread invoking `countDown()`. A `CountDownLatch` initialized to N can, e.g., be used to make one or several threads wait until N threads have completed some action (e.g., N threads that each call `countDown()` once), or some action has been completed N times (e.g., one thread that calls `countDown()` N times) A slightly simplified version of the interface to the `CountDownLatch` class is:

```
public class CountDownLatch {

  public CountDownLatch(int count);

  public void await();

  public void countDown();

  public int getCount();
}
```

Since the countdown latch 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()`).

Implement the class `CountDownLatch` with the interface and semantics described above (You may add modifier keywords to the method signatures). Your implementation should be safe towards spurious wakeups, i.e., if, for some awkward reason, a thread is released although the count is greater than zero it should wait anew. Correct handling of exceptions is not required. (2 p)

The inverse of an upper-triangular 3x3 matrix

If

$$A = \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{bmatrix}$$

then, assuming that the matrix is invertible, the inverse is given by

$$A^{-1} = \frac{1}{adf} \begin{bmatrix} df & -bf & (be - cd) \\ 0 & af & -ae \\ 0 & 0 & ad \end{bmatrix}$$