LUND
UNIVERSITY

Department of
AUTOMATIC CONTROL

Real-Time Systems

Exam June 4, 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 discrete time system

$$x(k+1) = \begin{pmatrix} 0.8 & 0.4 \\ 0.3 & 1.1 \end{pmatrix} x(k) + \begin{pmatrix} 0.2 \\ 0.3 \end{pmatrix} u(k)$$
$$y(k) = \begin{pmatrix} 1 & 0 \end{pmatrix} x(k)$$

a. Design a state-feedback controller so that the state goes to origin in exactly two time steps given an impulse disturbance $x(0) = x_o$. (1 p)

b. Modify the above controller so that the state goes to $x_{ref} = [0.5 \quad 0.2]^T$ in exactly two time steps given an impulse disturbance $x(0) = x_o$. (1 p)

2. Consider the continuous-time system

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

Sample this using ZOH with sampling period h for the two cases below.

a.

$$A = \begin{pmatrix} 0 & 0 \\ 3 & 0 \end{pmatrix}$$

(1 p)

b.

$$A = \begin{pmatrix} 0 & 1 \\ 0 & -4 \end{pmatrix}$$

(1 p)

3. You are controlling a system that can be modeled as a double integrator and where one of the control objectives is to be able to follow step reference changes without any stationary errors:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{1}{s^2}. \tag{1}$$

You are discussing with a colleague whether you should implement a state feedback controller or a PD controller.

a. Why aren't you considering integral action in the controller? (1 p)

b. What do you think in the end will be the difference between the state feedback controller $U(s) = L \cdot X(s)$ and the PD controller $U(s) = K_p \cdot E(s) + K_d \cdot sY(s)$? (1 p)

4. The table below describes three processes with their respective periods, relative deadlines and worst case execution times.

a. What is the CPU utilization? (1 p)

Task	T_i	D_i	C_i
A	7	5	3
B	3	2	1
C	15	8	3

b. Will all deadlines be met if rate monotonic scheduling is used? (2 p)

5. Nicolas, a new engineer who had taken the Real-Time Systems course in France had not learned that it was advantageous to discretize the three terms in the PID controller separately. His continuous-time PID representation was as follows

$$U(s) = (k_p + \frac{k_i}{s} + k_d s)E(s)$$

a. First Nicolas tried to discretize the above transfer function as a *single transfer function* using the Forward Euler approximation. However, when he tried to implement the controller based upon this discretization then he ran into a problem. Do the discretization and explain what the problem is. (1.5 p)

b. In order to avoid this problem he instead tried to discretize the above transfer function as a *single transfer function* using the Backward Euler approximation. Do this and write the resulting control code split up into one CalculateOutput and one UpdateState part using the scheme below. Anti-windup is not necessary. In order to get full points the code must be written so that the execution time is minimized. (1.5 p)

```
  y = getInput();
  r = getReference;
// CalculateOutput
  setOutput(u);
// updateState
```

6. In the Stork real-time kernel a context switch is initiated by a call to the `Schedule` procedure. Inside `Schedule` a check is performed to see if a change has been made that affects the first process (thread) in `ReadyQueue`. If that is the case a context switch is performed. Explain under which conditions a context switch will be performed in the following three cases (the following three calls to `Schedule`). (3 p)

a.
```
PROCEDURE Wait(sem: Semaphore);
VAR
  oldDisable : InterruptMask;

BEGIN
  oldDisable := Disable();
  WITH sem^ DO
    IF counter > 0 THEN
      DEC(counter);
    ELSE
      MovePriority(Running,waiting);
      Schedule; (* Case a *)
    END;
```

```
  END;
  Reenable(oldDisable);
END Wait;
```

b.
```
PROCEDURE Signal(sem: Semaphore);
VAR
  oldDisable : InterruptMask;

BEGIN
  oldDisable := Disable();
  WITH sem^ DO
    IF NOT isEmpty(waiting) THEN
      MovePriority(waiting^.succ, ReadyQueue);
      Schedule;  (* Case b *)
    ELSE
      INC(counter);
    END;
  END;
  Reenable(oldDisable);
END Signal;
```

c.
```
PROCEDURE Clock;

VAR P: ProcessRef;

BEGIN
  IncTime(Now,Tick); (* Now := Now + Tick *)
  LOOP
    P := TimeQueue^.succ;
    IF CompareTime(P^.head.nextTime,Now) <= 0 THEN
      MovePriority(P,ReadyQueue);
    ELSE
      EXIT;
    END;
  END;
  DEC(Running^.timer); (* Round-robin time slicing *)
  IF Running^.timer <= 0 THEN
    MovePriority(Running,ReadyQueue);
  END;
  Schedule;  (* Case c *)
END Clock;
```

7. The following code for multiplying two $Q4.3$ fixed-point numbers X and Y and storing the result in Z (also $Q4.3$) contains three errors. Which are the errors and what will their consequences be? (3 p)

```
#include <inttypes.h>
#define n 3
int8_t X, Y, Z;
int16_t temp;
...
temp = X * Y;
temp = temp + (1 << n);
temp = temp >> n;
Z = temp;
if (Z > INT8_MAX)
   Z = INT8_MAX;
```

```
else if (Z < INT8_MIN)
   Z = INT8_MIN;
```

8. Consider a surveillance system of cameras that are connected to a central network manager.

The network manager allocates a certain amount of bandwidth to the cameras in the network. The cameras use this allocated bandwidth to send images (frames) captured by them. Once 30 frames are sent by the cameras, the program ends. Write a Grafcet solution of this problem for the case when there is only one camera and one network manager. The Grafcet solution may consist of two Grafcet function charts that communicate with each other using shared variables. The special boolean variable <step-name>.x may also be used as a shared variable. (3 p)

The network manager performs the following actions:

1. Initialize
2. Set the bandwidth allocated to the camera (= `alloc_bw`) to, e.g., 50 and wait for the camera to send the image.
3. If the number of sent frames is smaller than 30, then go to Step 2.
4. If the number of sent frames is greater than or equal to 30 then terminate.

The camera performs the following actions

1. Initialize and wait for the network manager to allocate bandwidth.
2. Calculate the frame size for the current frame using the formula `framesize = quality * max_size` (You may assume that these variables already are available).
3. If the frame size is greater than the allocated bandwidth (`alloc_bw`) then do not send the frame.
4. If the frame size is smaller or equal to the allocated bandwidth then send the frame (using the Grafcet action `S sendFrame()` and increment the number of frames sent (use the variable `Frames` to count the number of frames sent).
5. Wait for the network manager to allocate bandwidth again.
6. Go to Step 2.

9. Kalle, a new engineer who did not take the Real-Time Systems course, was given the task to implement a PI-controller. He came up with the following solution. It is structured into one PI class and one Regul class in the common way.

```
public class PI {
    private double y, yref, v;
    private double I = 0.0;
    private double K = 0.0;
    private double Ti = 0.0;
    private double Tr = 0.0;
    private double H = 0.0;
    private double Beta = 0.0;
```

```
    public PI(Reference ref) {
        setParameters(1.0,10.0,10.0,1.0);
    }

    public synchronized double calculateOutput(double y, double yref) {
        this.y = y;
        this.yref = yref;
        v = K*(Beta*yref - y) + I;
        return v;
    }

    public synchronized void updateState(double u) {
        I = I + K*H/Ti*(yref - y) + (H/Tr)*(u - v);
    }

    public synchronized long getHMillis() {
        return (long)(H*1000.0); //Sampling interval in milliseconds
    }

    public synchronized void setParameters(double K, double Ti,
                                           double Tr, double Beta) {
        this.K = K;
        this.Ti = Ti;
        this.Tr = Tr;
        this.Beta = Beta;
    }
}

// ---------------------------------------------------------------------------
// ---------------------------------------------------------------------------


public class Regul extends Thread {
    private Reference ref;
    private PI pi = new PI();
    private AnalogIn yChan;
    private AnalogOut uChan;
    private long h;
    private double y,yref,v,u;
    private double uMax = 10.0;
    private double uMin = -10.0;


    public Regul(Reference ref) {
      this.ref = ref;
      try {
        yChan = new AnalogIn(1);
        uChan = new AnalogOut(1);
      } catch (Exception e) {
          System.out.println(e);
      }
    }

    private double limit(double v, double min, double max) {
        if (v < min) {
```

```
            v = min;
        } else {
            if (v > max) {
                v = max;
            }
        }
        return v;
    }

    public void run() {
        setPriority(7);
        long duration;
        long t=System.currentTimeMillis();
        while (true) {
            yref = ref.getReference();
            try {
              y = yChan.get();
            } catch (Exception e) {
              System.out.println(e);
            }
            synchronized(pi) { // To avoid parameter changes inbetween
                v = pi.calculateOutput(y, yref);
                u = limit(v,uMin,uMax);
                try {
                  uChan.set(u);
                } catch (Exception e) {
                  System.out.println(e);
                }
                pi.updateState(u);
            }
            t = t + pi.getHMillis();
            duration = t - System.currentTimeMillis();
            if (duration > 0) {
                try {
                    sleep(duration);
                } catch (Exception x) {
                }
            }
        }
    }
}
```

In addition to the Regul thread, Kalle's application also contained other threads of lower priority than Regul.

a. When Kalle started the application he soon detected that something was wrong. However, the controller appeared to do some type of control. What type of controller was it that Kalle in fact had implemented? (Hint: Subproblem b might point you in the right direction.) (1 p)

b. Kalle detected that something was strange when he looked upon the control performance plots (step responses). The plant that he was controlling was a continuous-time first order system with the pole strictly in the left complex half plane. What was it that Kalle found strange? (1 p)

c. When Kalle started the testing he used a real-time Java Virtual Machine (JVM) implementation in which the underlying native threads had the same

priority as the Java threads. After a while he changed to an ordinary JVM and then he got a different timing behaviour. Describe the timing behaviour that Kalle got using the two JVMs and the reasons for the behaviour. (2 p)