LUND
UNIVERSITY

Department of
AUTOMATIC CONTROL

Real-Time Systems

Exam June 1, 2016, 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/

Solutions to the exam in Real-Time Systems 160601

These solutions are available on WWW:
http://www.control.lth.se/course/FRTN01/

1. Consider the continuous-time system

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

Sample the system using ZOH and sampling period $h = 2$. (1 p)

Solution

$$\Phi(h) = e^{Ah} = \sum_{k\geq 0} \frac{1}{k!} A^k h^k = I + Ah = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix}$$
$$\Gamma(h) = \int_0^h e^{At} B\, dt = \int_0^h \begin{bmatrix} t \\ 1 \end{bmatrix} dt = \begin{bmatrix} t^2/2 \\ t \end{bmatrix}\Big|_{t=0}^{h} = \begin{bmatrix} h^2/2 \\ h \end{bmatrix}$$
$$x(k+1) = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} x(k) + \begin{bmatrix} h^2/2 \\ h \end{bmatrix} u(k).$$

The system then becomes

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

2. Consider the two SISO controller structures in Figure 1 and Figure 2.

a. How do you motivate a change from the structure in Figure 1 to the structure in Figure 2? (1 p)

b. What is the transfer function from r to y in Figure 2? (1 p)

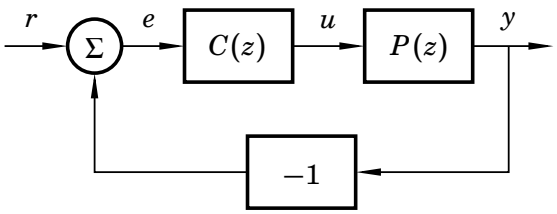


Figure 1 The old structure in Problem 2.

Solution

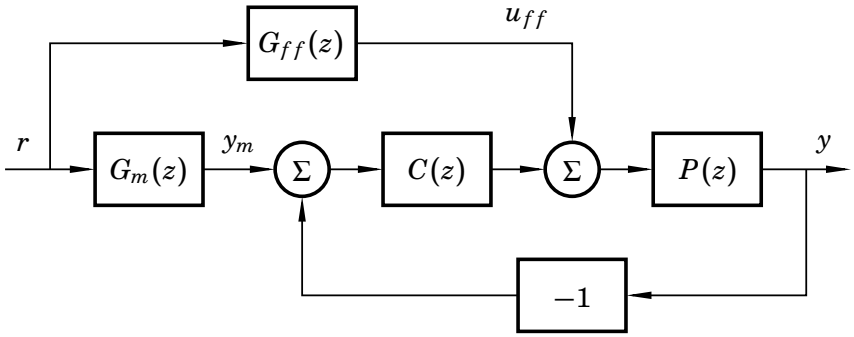


Figure 2 The new structure in Problem 2.

a. In the original structure, $C(z)$ is responsible for handling both set-point changes and disturbances. This means that there is a trade off in the performance. In the new structure, the problems have been separated. $C(z)$ now handles disturbances and $G_m(z)$ and $G_{ff}(z)$ handles set-point changes. This gives one more degree of freedom and therefore often better performance.

b.
$$Y(z) = \frac{P(G_{ff}(z) + C(z)G_m(z))}{1 + P(z)C(z)} R(z).$$

3. Given the following continuous-time system:

$$\dot{x}(t) = \begin{pmatrix} -6 & 1 \\ -9 & 0 \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(t)$$
$$y(t) = \begin{pmatrix} 1 & 0 \end{pmatrix} x(t).$$

a. Compute the continuous-time transfer-function and determine if the system is stable. (1 p)

b. Approximate the continuous-time transfer-function with a discrete time pulse-transfer function using forward difference and $h = 1$. Determine if the discrete-time system is stable. (1 p)

c. Now approximate the continuous-time system using Tustin instead. Determine the poles of the discrete-time system and whether it is stable or not. (1 p)

Solution

a. The cont. transfer function is given by

$$G(s) = C(sI - A)^{-1} B = \frac{1}{s^2 + 6s + 9} = \frac{1}{(s+3)^2}.$$

So the poles are located in $s = -3$ and the system is stable.

b. With forward difference the approximation is given by $H(z) = G(s')$ with $s' = z - 1$.

$$H(z) = \frac{1}{(s'+3)^2} = \frac{1}{(z-1+3)^2} = \frac{1}{(z+2)^2}.$$

So the poles are located in $z = -2$, which is outside the unit circle, and the discrete-time system is unstable. (Sample time $h = 1$)

c. With Tustin approximation we have that $s' = \frac{2}{h} \cdot \frac{z-1}{z+1}$, with $h = 1$ we get:

$$\begin{aligned} H(z) = G\left(s'\right) &= \frac{1}{\left(2\frac{z-1}{z+1} + 3\right)^2} = \frac{1}{\left(2\frac{z-1}{z+1}\right)^2 + 6 \cdot 2\frac{z-1}{z+1} + 9} \\ &= \frac{(z+1)^2}{4(z-1)^2 + 12(z-1)(z+1) + 9(z+1)^2} \\ &= \frac{(z+1)^2}{4z^2 - 8z + 4 + 12z^2 - 12 + 9z^2 + 18z + 9} \\ &= \frac{(z+1)^2}{25z^2 + 10z + 1} = \frac{1}{25} \cdot \frac{(z+1)^2}{z^2 + \frac{10}{25}z + \frac{1}{25}} \\ &= \frac{0.4(z+1)^2}{z^2 + 0.4z + 0.04} = \frac{0.4(z+1)^2}{(z+0.2)^2}. \end{aligned}$$

So there is a zero at $z = -1$ and two poles at $z = -0.2$, which is within the unit circle and hence the system is stable.

4. Consider the following discrete-time system

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

a. Design a state feedback controller $u(k) = -Lx(k)$. Choose the closed loop characteristic polynomial in such a way that the sampled dynamics with $h = 0.1$ s corresponds to a continuous-time double pole in $s = -1$. (1 p)

b. Design an observer in the form

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

such that the observer dynamics is twice as fast as the closed loop dynamics, i.e., corresponds to a continuous-time double pole in $s = -2$. (1 p)

c. Design a model and feedforward generator in such a way that

1. The states of the model are compatible with the process states.

2. The discrete-time model dynamics corresponds to a continuous-time double pole in $s = -2$.
3. The steady state gain from the command signal u_c to y should be 1.

(2 p)

d. Write pseudo-code for the calculations that should be implemented in the controller, i.e., the calculations needed to implement the state feedback, the observer, and the model and feedforward generator. 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. 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 statement. Use $\hat{x}$ to denote the estimated state vector and x_m to denote the model state vector. (2 p)

Solution

a. A continuous-time double pole in $s = -1$ corresponds to a discrete-time double pole in $e^{-h} = e^{-0.1}$, i.e., the desired closed loop characteristic polynomial should be

$$(z - e^{-0.1})^2 = z^2 - 2e^{-0.1}z + e^{-0.2} = z^2 + p_1 z + p_2.$$

With the linear feedback

$$u(k) = -l_1 x_1(k) - l_2 x_2(k)$$

the closed-loop system becomes

$$x(k+1) = \begin{pmatrix} 1 - l_1 h^2/2 & h - l_2 h^2/2 \\ -l_1 h & 1 - l_2 h \end{pmatrix} x(k).$$

The characteristic polynomial of the closed-loop system is

$$z^2 + \left(\frac{l_1 h^2}{2} + l_2 h - 2\right) z + \left(\frac{l_1 h^2}{2} - l_2 h + 1\right).$$

Comparing this with the desired characteristic polynomial leads to the following linear equations for l_1 and l_2

$$\begin{aligned} \frac{h^2 l_1}{2} + h l_2 - 2 &= p_1 \\ \frac{h^2 l_1}{2} - h l_2 + 1 &= p_2 \end{aligned}$$

with the solution

$$\begin{aligned} l_1 &= \frac{1}{h^2}(1 + p_1 + p_2) = 0.9056 \\ l_2 &= \frac{1}{2h}(3 + p_1 - p_2) = 1.8580 \end{aligned}.$$

b. The characteristic polynomial of the observer is given by

$$\begin{aligned}\det(zI - \Phi + KC) &= \det\begin{pmatrix} z-1+k_1 & -0.1 \\ k_2 & z-1 \end{pmatrix} \\ &= z^2 + (k_1 - 2)z + 1 - k_1 + 0.1k_2\end{aligned}.$$

The desired characteristic polynomial is

$$(z - e^{-0.2})^2 = z^2 - 2e^{-0.2}z + e^{-0.4}$$

Equating the coefficients we get

$$\begin{cases} k_1 - 2 = -2e^{-0.2} \\ 1 - k_1 + 0.1k_2 = e^{-0.4} \end{cases} \Rightarrow \quad K = \begin{pmatrix} 0.3625 & 0.3286 \end{pmatrix}^T.$$

c. The model and feedforward generator design is based is performed as follows. In order to make sure that the states of the model are compatible with the states of the process the following approach can be used. To begin with the model can be chosen identical to the process, i.e.,

$$\begin{aligned} x_m(k+1) &= \Phi x_m(k) + \Gamma u_{ff}(k) \\ y_m(k) &= C x_m(k) \end{aligned}. \tag{1}$$

The dynamics of the model can then be modified by the linear control law

$$u_{ff}(k) = -L_m x_m(k) + l_r u_c(k). \tag{2}$$

The model dynamics is then given by

$$\begin{aligned} x_m(k+1) &= (\Phi - \Gamma L_m) x_m(k) + \Gamma l_r u_c(k) \\ y_m(k) &= C x_m(k) \end{aligned}. \tag{3}$$

Here, L_m is chosen to give the model the desired eigenvalues and l_r is chosen to give the model a static gain of 1. The feedforward control signal $u_{ff}(k)$ is generated in such a way that it will give the desired behavior when used as an input to the process.

The desired characteristic polynomial of the model is given by

$$(z - e^{-2h})^2 = (z - e^{-0.2})^2 = z^2 - 2e^{-0.2}z + e^{-0.4}$$

From the first sub-problem it follows that the coefficients of L_m should be chosen as

$$\begin{aligned} l_1 &= \frac{1}{h^2}(1 + p_1 + p_2) = 3.2859 \\ l_2 &= \frac{1}{2h}(3 + p_1 - p_2) = 3.4611 \end{aligned}$$

Finally, l_r is chosen to get the static gain 1. This is obtained by setting

$$l_r = \frac{1}{C(I - \Phi + \Gamma L_m)^{-1}\Gamma} = 3.2$$

d. Both the observer and the process model should be updated in UpdateState. Also in UpdateState pre-calculations can be done both for u and for u_{ff}. This leads to the following pseudo-code:

CalculateOutput:
Sample y and obtain u_c
$u_{ff} = u_{ff} + l_r u_c$
$u = u + u_{ff}$
Output u
Update State:
$\hat{x} = \Phi\hat{x} + \Gamma u + K(y - C\hat{x})$
$x_m = \Phi x_m + \Gamma u_{ff}$
$u_{ff} = -L_m x_m$
$u = L(x_m - \hat{x})$

5. The operation

$$x = a \cdot b$$

should be performed using a fixed-point implementation. In the operation, a is a constant with value 1.35. 8-bits signed variables should be used to represent x, a and b.

a. Choose an appropriate number of fractional bits for a and convert a into the corresponding fixed-point representation. (1 p)

b. Zero fractional bits are used to represent x and b. Complete the code skeleton below to compute x. In case of overflow, the result should be saturated. 16-bits variables may be used for intermediate results. (1 p)

```
#include <inttypes.h>
// Insert in the next two lines the
// results from the first subproblem
#define n ... // number of fractional bits
#define a ... // fixed-point representation of a

int8_t x, b;
// define more variables if needed
...

// assume b is initialized to a value (you don't need to writ
// the code for the initialization of b)
// write the code to compute x
...
x = ...
```

Solution

a. Since $1 \leq a < 2$, only one bit is needed to represent the integer part and $8-1-1 = 6$ bits are left for the fractional part. The fixed point representation of a is $\text{round}(1.35 \cdot 2^6) = 86$.

b. In the following code, the result of the multiplication is saved in a temporary variable, which has type `int16_t`. The value is then saturated to handle overflows and underflows and finally casted into its corresponding `int8_t` value.

```
#include <inttypes.h>
// Insert in the next two lines the
// results from the first subproblem
#define n 6
#define a 86

int8_t x, b;
// define more variables if needed
int16_t temporary;

// assume b is initialized to a value (you don't need to write
// the code for the initialization of b)
// write the code to compute x
temporary = ((int16_t) a*b) >> n;
if (temporary > 127)
  temporary = 127;
else if (temporary < -128)
  temporary = -128;

x = (int8_t) temporary;
```

6. An engineer has implemented a real-time control system using three threads. The first one is the control thread, which must execute at a sampling rate of 4 milliseconds. The second one is a reference generator, which should update the reference value every 12 milliseconds. The third one is a user interface thread, which should update the plotters 50 times per second. The engineer has timed precisely the execution of the reference generator, 4 milliseconds, and of the user interaction thread, 9 milliseconds. However, she does not know exactly how much time the control thread takes to execute. Assume an ideal kernel.

a. What is the maximum controller execution time allowed if the task set is to be schedulable using Earliest Deadline First scheduling? (1 p)

b. Assume that the deadline D_i for each thread is equal to the period T_i. Assume that the controller execution time is 0.5 milliseconds and that all blocking due to inter-process communication can be ignored. Will the task set be schedulable using Earliest Deadline First scheduling? And using rate monotonic priority assignments? (3 p)

Solution

a. Denoting the control task execution time with x it is possible to compute the utilization as

$$U = \frac{x}{4} + \frac{4}{12} + \frac{9}{20}$$

where 20 is the period of the user interaction tasks in milliseconds obtained as 1000/50. For schedulability the utilization should be less than 1 and imposing the conditions above gives $x < 0.867$ milliseconds.

b. As stated above, the task set is schedulable using EDF. For monotonic priority assignments, a sufficient schedulability criterion is if the utilization satisfies the condition

$$\sum_{i=1}^{i=n} \leq n(2^{1/n} - 1).$$

But the utilization for the given task set is

$$\sum_{i=1}^{i=n} = 0.5/4 + 4/12 + 9/20 = 0.9083 \geq 3(2^{1/3} - 1) \approx 0.78$$

and we can not conclude that the tasks are schedulable. We instead turn to the exact analysis, where the response time for process i can be calculated from iteration of the equation

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j.$$

The highest priority task is the control task, which will trivially have a response time which we denote $R_1 = C_1 = x = 0.5$ ms. For the medium priority reference generator we get, with initial guess $R_2^0 = C_2 = 4$:

$$R_2^1 = C_2 + \left\lceil \frac{R_2^0}{T_1} \right\rceil C_1 = 4 + \left\lceil \frac{4}{4} \right\rceil 0.5 = 4.5$$
$$R_2^2 = C_2 + \left\lceil \frac{R_2^1}{T_1} \right\rceil C_1 = 4 + \left\lceil \frac{4.5}{4} \right\rceil 0.5 = 5.0$$
$$R_2^3 = C_2 + \left\lceil \frac{R_2^2}{T_1} \right\rceil C_1 = 4 + \left\lceil \frac{5.0}{4} \right\rceil 0.5 = 5.0$$

which gives $R_2 = 5.0$ ms. For the user I/O task, with initial guess $R_3^0 = C_3 = 9$ we get

$$R_3^1 = C_3 + \left\lceil \frac{R_3^0}{T_1} \right\rceil C_1 + \left\lceil \frac{R_3^0}{T_2} \right\rceil C_2 = 9 + \left\lceil \frac{9}{4} \right\rceil 0.5 + \left\lceil \frac{9}{12} \right\rceil 4 = 14.5$$
$$R_3^2 = C_3 + \left\lceil \frac{R_3^1}{T_1} \right\rceil C_1 + \left\lceil \frac{R_3^1}{T_2} \right\rceil C_2 = 9 + \left\lceil \frac{14.5}{4} \right\rceil 0.5 + \left\lceil \frac{14.5}{12} \right\rceil 4 = 19$$
$$R_3^3 = C_3 + \left\lceil \frac{R_3^2}{T_1} \right\rceil C_1 + \left\lceil \frac{R_3^2}{T_2} \right\rceil C_2 = 9 + \left\lceil \frac{19}{4} \right\rceil 0.5 + \left\lceil \frac{19}{12} \right\rceil 4 = 19.5$$
$$R_3^4 = C_3 + \left\lceil \frac{R_3^3}{T_1} \right\rceil C_1 + \left\lceil \frac{R_3^3}{T_2} \right\rceil C_2 = 9 + \left\lceil \frac{19.5}{4} \right\rceil 0.5 + \left\lceil \frac{19.5}{12} \right\rceil 4 = 19.5$$

which gives $R_3 = 19.5$ ms. All tasks will therefore meet their deadlines.

7. The Traffic Control Agency (T.C.A) are currently developing a new structure for the traffic light in a crossing. They have just heard about Petri nets, and the ability to represent real-time systems with it.

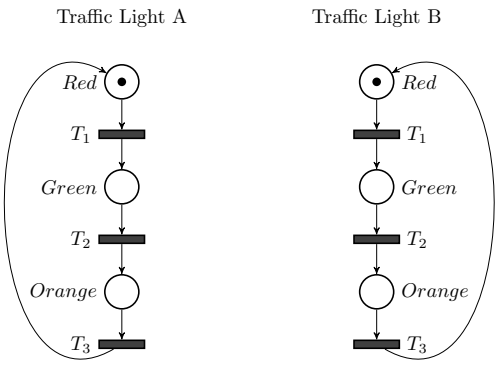


Figure 3 Petri Net over two traffic lights, developed by the Traffic Control Agency. The goal is to control the two traffic lights and assure that the green lights are mutually exclusive.

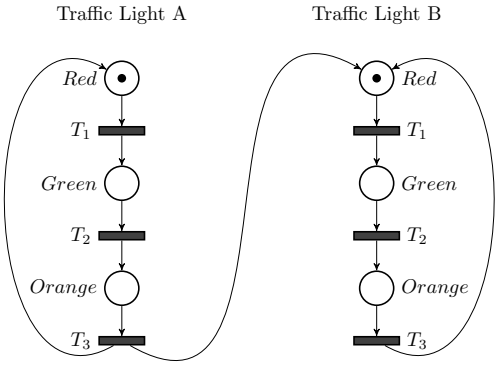


Figure 4 K.T.H's solution to the problem of the Traffic Control Agency.

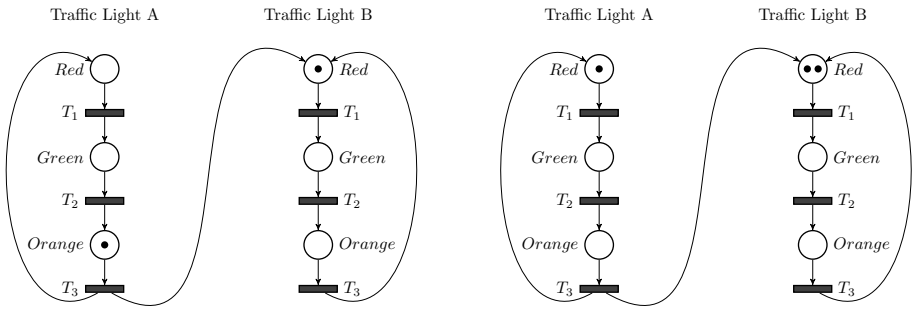
So far they have developed two Petri nets (see Fig. 3), one for each traffic light, that they wish to use. However, they wish to connect the two traffic lights with each other and have now turned to the Automatic Control community for help. The only specification they have so far is:

- The green lights should be mutually exclusive (only one can shine at every given moment).
- The original Petri net developed by the T.C.A must be part of the solution.

a. The first person T.C.A reached out to for help was Klas Theodor Hoppsan (K.T.H), who studied at an unknown university in Sweden. He came up with the solution shown in Fig. 4. Explain and illustrate whether or not his solution is:

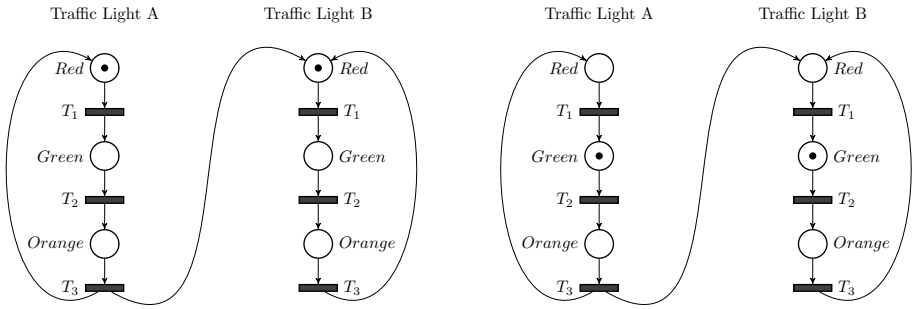
- unbounded
- mutually exclusive (w.r.t. the green lights)

(2 p)



(a) Before A.T3 firing (b) After A.T3 firing

Figure 5 Illustration of the unboundedness of the suggested solution.



(a) Before A.T1 and B.T1 firing (b) After A.T1 and B.T1 firing

Figure 6 Illustration of the lack of mutual exclusion of the suggested solution.

b. Disappointed with what K.T.H produced they now turn to you for help. Using the Petri net in Fig. 3 as a starting point, extend it and develop a new solution that is:

- mutually exclusive (the green lights)
- unbounded

(2 p)

Solution

a. The suggested solution is indeed unbounded. Every time traffic light A fire transition T3 a new token will appear in B.Red, even though it might already have one there. See Fig. 5.

The suggested solution is clearly not mutually exclusive since there is nothing preventing it from firing both A.T1 and B.T1 simultaneously, see Fig 6.

b. To solve all problems all one needs to do is to add a semaphore that is shared between the two traffic lights. The extended solution can be seen in Fig. 7.

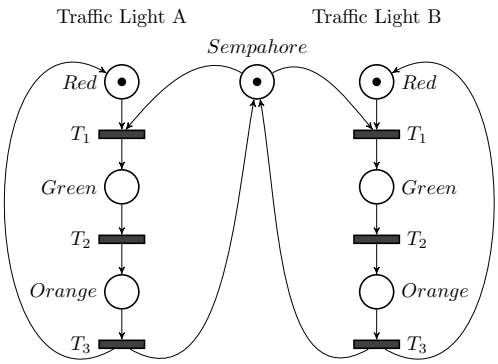


Figure 7 Solution that ensures both boundedness and mutual exclusion between the green lights.

8. Consider the following Java implementation of a PI-controller which is structured into one PI class and one Regul class.

```
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();
            y = yChan.get();
            synchronized(pi) { // To avoid parameter changes inbetween
                v = pi.calculateOutput(y, yref);
                u = limit(v,uMin,uMax);
                uChan.set(u);
                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, the application also contains other threads of lower priority than Regul.

a. When starting the application it was detected that something was wrong. However, the controller appeared to do some type of control. What type of controller does the code realize? (1 p)

b. When first testing the controller, the native thread model in Java was used. After a while this was changed to the green thread model. Then, a different timing behavior was obtained. Describe the timing behavior using the two thread models and the reasons for the behavior. (2 p)

Solution

a. The sampling interval, H, in the code will always be 0. This means that the PI controller will execute as a P-Controller. The Regul thread will never sleep. This means that the sampling interval of the P-controller will be very short and will depend on the speed of the computer.

b. In the green thread model, the JVM will follow the thread priorities strictly. This means that Regul will take all the CPU time and no other threads will run, i.e. starvation. In the native thread model, the execution of the Java threads is mapped down to the underlying OS threads, e.g., Linux threads. These threads normally execute at the same priority and employ time-sharing. The effect of this is that the threads with lower priority than Regul will still execute, albeit possibly with a slower speed.