Solutions to the exam in Real-Time Systems 2016-05-12

1 a. Since the delay $\tau = 30$ is an integer multiple of the sampling interval $h = 1$, it will translate into a backward shift of $\tau/h = 30$ samples in the discrete-time system.

Sampling the non-delayed part of the system using Table 3 in IFAC PB, we obtain

$$H'(z) = \frac{1.1(1-e^{-0.1})z + e^{-0.1}(e^{-0.1}-0.9)}{(z-e^{-0.1})^2} = \frac{0.004679z + 0.004377}{z^2 - 1.81z + 0.8187}$$

Including the delay we have

$$H(z) = \frac{0.004679z + 0.004377}{(z^2 - 1.81z + 0.8187)z^{30}}$$

b. The system order is given by the order of the denominator polynomial, which is 32.

Stability can be assessed in two different ways: 1) Since the continuous-time plant is stable (two poles in the left half plane plus a time delay that does not affect stability), the ZOH-sampled plant must also be stable. 2) The discrete-time system has two poles in $e^{-0.1} \approx 0.904$ and 30 poles in 0, which are all inside the unit circle, so the system is stable.

2. The problem is that there is a circular wait-chain. If all processes execute the first Wait-statement, they all have acquired a resource. In the next statement, they all try to acquire another resource which is already held by one of the other processes. Deadlock will occur. This may be solved by using hierarchical resource allocation and rewriting P3 to:

```
Wait(R1);
Wait(R3);
// Using R1 and R3
Signal(R3);
Signal(R1);
```

3 a. The sorting logic works as follows:

- Animals 2.0 years or younger, regardless of gender, are released.
- Male cattle 12.0 years or younger are released, while male cattle older than 12.0 years are sent to the abattoir.
- Female cattle older than 2.0 years are milked for 6 minutes and then released.

b. Change the condition `AGE>2 & FEMALE` to `2<AGE<10 & FEMALE` and the condition `AGE>12 & !FEMALE` to `(AGE>12 & !FEMALE) | (AGE>10 & FEMALE)` to achieve the desired result.

4. Both implementations are correct from a mutual exclusion point of view. However, Implementation 2 may cause unnecessary blocking. Since a single monitor is used to hold all the shared variables, it is, for example, possible that the `Regul` thread will be blocked when trying to extract the controller parameters using `getParameters`, by the `Refgen` thread which is changing the reference value by executing `setRef`. Similarly, the `Regul` thread can be blocked when calling `getRef` by the Swing thread in `Opcom` calling `setParameters`.

5 a. With $L = [l_1, l_2]$, the poles of the closed loop system are given by the characteristic polynomial

$$\begin{aligned}\det(zI - (\Phi - \Gamma L)) &= \det\left(\begin{bmatrix} z & 0 \\ 0 & z \end{bmatrix} - \begin{bmatrix} 0.5 & 0 \\ 1 & 0.9 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ l_1 & l_2 \end{bmatrix}\right) = \\ & (z - 0.5)(z + l_2 - 0.9)\end{aligned}$$

from which it is evident that one pole is already in 0.5. The position of the second pole depends on the value of l_2 and to place it in 0.5 simply means selecting

$$l_2 - 0.9 = -0.5 \rightarrow l_2 = 0.4$$

while the value of l_1 is irrelevant.

Finally, in order to have a static gain of 1

$$l_r = \frac{1}{C(I - \Phi + \Gamma L)^{-1}\Gamma} = l_2 + 0.1 = 0.5$$

b. An observer is given by

$$\begin{aligned}\hat{x}(k+1) = \Phi\hat{x}(k) + \Gamma u(k) + K(y(k) - C\hat{x}(k)) \rightarrow \\ \hat{x}(k+1) = (\Phi - KC)\hat{x}(k) + \Gamma u(k) + Ky(k)\end{aligned}$$

where $K = [k_1, k_2]^T$. The poles of the observer are determined by the eigenvalues of $\Phi - KC$, which are computed as

$$\begin{aligned}\det(zI - (\Phi - KC)) &= \det\left(\begin{bmatrix} z & 0 \\ 0 & z \end{bmatrix} - \begin{bmatrix} 0.5 & 0 \\ 1 & 0.9 \end{bmatrix} + \begin{bmatrix} 0 & k_1 \\ 0 & k_2 \end{bmatrix}\right) = \\ & z^2 + (k_2 - 1.4)z + (0.45 - 0.5k_2 + k_1)\end{aligned}$$

and imposing that the poles are in 0.25 means imposing that the characteristic polynomial has the form $(z - 0.25)(z - 0.25)$ which is $z - 0.5z + 0.0625$.

$$z^2 + (k_2 - 1.4)z + (0.45 - 0.5k_2 + k_1) = z - 0.5z + 0.0625$$

gives us

$$k_2 - 1.4 = -0.5 \rightarrow k_2 = 0.9$$

and consequently

$$0.45 - 0.5 \cdot 0.9 + k_1 = 0.0625 \rightarrow k_1 = 0.0625$$

6 a. According to the RMS scheme, the task with the shortest period is assigned the highest priority. Thus we have: Task A: medium priority, Task B: low priority and Task C: high priority.

The task set is not schedulable using the approximate schedulability condition since

$$\sum_{i=1}^{i=3} \frac{C_i}{T_i} = \frac{1}{3} + \frac{2}{5} + \frac{0.5}{2} = 0.9833 > 3(2^{1/3} - 1) = 0.7798.$$

b. According to Theorem 8.3 in the course book, applicable to RMS analysis assuming fixed priority assignment, all tasks, n to the number, will meet their deadlines if and only if

$$\forall i, R_i \leq D_i$$

where

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

where $hp(i)$ is the set of tasks of higher priority than task i.

Calculation of the response times, R, for the tasks gives

$$\begin{aligned}
R_{A0} &= 0 \\
R_{A1} &= 1 \\
R_{A2} &= 1 + \left\lceil \frac{1}{2} \right\rceil 0.5 = 1.5 \\
R_{A3} &= 1 + \left\lceil \frac{1.5}{2} \right\rceil 0.5 = 1.5 \\
R_{B0} &= 0 \\
R_{B1} &= 2 \\
R_{B2} &= 2 + \left\lceil \frac{2}{3} \right\rceil 1 + \left\lceil \frac{2}{2} \right\rceil 0.5 = 3.5 \\
R_{B3} &= 2 + \left\lceil \frac{3.5}{3} \right\rceil 1 + \left\lceil \frac{3.5}{2} \right\rceil 0.5 = 5 \\
R_{B4} &= 2 + \left\lceil \frac{5}{3} \right\rceil 1 + \left\lceil \frac{5}{2} \right\rceil 0.5 = 5.5 \\
R_{B5} &= 2 + \left\lceil \frac{5.5}{3} \right\rceil 1 + \left\lceil \frac{5.5}{2} \right\rceil 0.5 = 5.5 \\
R_{C0} &= 0 \\
R_{C1} &= 0.5
\end{aligned} \tag{1}$$

We have that $R_A = 1.5 \leq D_A$, $R_B = 5.5 > D_B$ and $R_C = 1 \leq D_C$ and thus the task set is not schedulable using RMS priority assignment.

c. The schedule is shown in Figure 1.

d. The utilization of the system is calculated from the expression

$$U = \sum_{i=3}^{i=1} \frac{C_i}{T_i} = 0.9833 < 1.$$

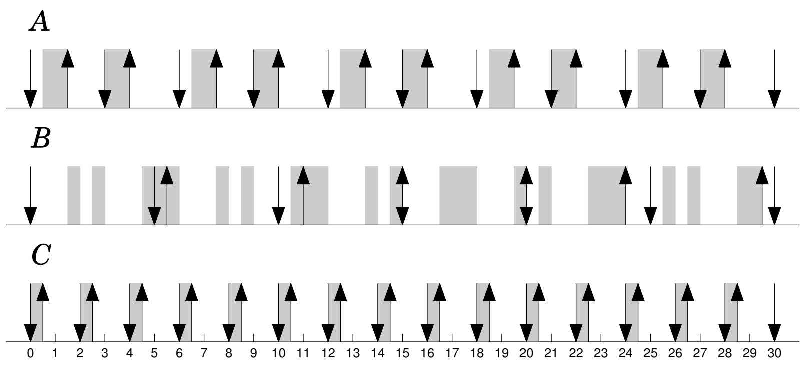

Figure 1 The execution trace resulting from rate monotonic scheduling.

Since the utilization of the system is less than 1, the system is schedulable using EDF.

7 a. Differentiating the equation for γ, we get

$$p\gamma(t) = \beta(y(t))$$

Setting $p = \frac{q-1}{h}$ gives

$$\frac{q-1}{h}\gamma(t) = \beta(y(t))$$

or equivalently

$$\gamma(t+h) = \gamma(t) + h\beta(y(t))$$

The equation for v becomes

$$v(t) = \frac{s_1\frac{q-1}{h} + s_0}{(\frac{q-1}{h})^2 + r_1\frac{q-1}{h} + r_0}y(t) =$$
$$= \frac{qs_1/h + s_0 - s_1/h}{q^2/h^2 + q(r_1/h - 2/h^2) + r_0 - 1/h^2 - r_1/h}y(t)$$

or equivalently

$$v(t+2h) = -(r_1h-2)v(t+h)-(r_0h^2-1-r_1h)v(t)+s_1hy(t+h)+(s_0h^2-s_1h)y(t)$$

(The equation for u is purely algebraic and does not have to be discretized.)

b. Since the original system is stable, Tustin's method is guaranteed to give a stable controller.

8. The smallest coefficient is $0.0625 = 1/16 = 1/2^4$. Using four fractional bits, we can represent all coefficients without round-off errors. We should not use a larger number of fractional bits since that will increase the risk of overflow in the intermediate calculations.

In fixed-point representation, the system matrices are given by

$$F = 2^4\Phi = \begin{pmatrix} 12 & 0 \\ 4 & 16 \end{pmatrix}, \quad G = 2^4\Gamma = \begin{pmatrix} 4 \\ 1 \end{pmatrix}$$

In the calculations, the magnitude of the results should never exceed $2^{15} = 32768$. To avoid overflow, we should ensure that

$$|12x_1 + 4u| < 32768, \quad |4x_1 + 16x_2 + u| < 32768$$

Knowing that $|u| \leq 256$ and assuming the same magnitude limit l for both x_1 and x_2, the second inequality will dominate, yielding

$$20l + 256 < 32768 \quad \Rightarrow \quad l < 1625.6$$

A safe limit is hence 1625. Reasoning only about the number of bits needed, another option is to choose the limit as $2^{10} - 1 = 1023$.

The C code becomes something like:

```
int16_t x1, x2, u, x1unsat, x2unsat;

u = read_input();

x1unsat = (12 * x1 + 4 * u) >> 4;
x2unsat = (4 * x1 + 16 * x2 + u) >> 4

if (x1unsat > 1625)
   x1 = 1625;
else if (x1unsat < -1625)
   x1 = -1625;
else
   x1 = x1unsat;

if (x2unsat > 1625)
   x2 = 1625;
else if (x2unsat < -1625)
   x2 = -1625;
else
   x2 = x2unsat;
```