Solutions to the exam in Real-Time Systems 170103

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

1. Since the task set does not use rate-monotonic or deadline monotonic priority assignment the simple schedulability tests are not applicable.

Using response-time analysis the following is obtained.

$$R_C^0 = 0,\ R_C^1 = C_C = 1,\ R_C^2 = C_C = 1 < 6$$

$$R_B^0 = 0,\ R_B^1 = C_B = 1,$$

$$R_B^2 = C_B + \left\lceil \frac{1}{T_C} \right\rceil C_C = 2,$$

$$R_B^3 = \ldots = 2 < 4$$

$$R_A^0 = 0,\ R_A^1 = C_A = 3,$$

$$R_A^2 = C_A + \left\lceil \frac{3}{T_B} \right\rceil C_B + \left\lceil \frac{3}{T_C} \right\rceil C_C = C_A + C_B + C_C = 5$$

$$R_A^3 = C_A \left\lceil \frac{5}{T_B} \right\rceil C_B + \left\lceil \frac{5}{T_C} \right\rceil C_C = C_A + 2C_B + C_C = 6$$

$$R_A^4 = \ldots = 6 < 10$$

$R_i \leq D_i \Rightarrow$ The task set is schedulable.

2.

a. The system is unreachable when the reachability matrix $W_C = [\Gamma\ \Phi\Gamma]$ does not have full rank. In this case

$$W_C = \begin{pmatrix} d & da + be \\ e & ce \end{pmatrix}.$$

The condition for this matrix to not have full rank is that the determinant is equal to zero. In this case $\det W_C = e(dc - da - be)$, i.e., the condition is that $e = 0$ or $dc - da - be = 0$.

b. In the first case there is no way that $u(k)$ can effect $x_2(k)$. In the second case there is no way that $u(k)$ can effect $x_1(k)$. Finally in the third case $u(k)$ will effect $x_1(k)$ and $x_2(k)$ in exactly the same way, i.e., we cannot control the two state variables independently of each other.

3. Since the time-delay τ for this system is longer than the sampling period h, one can write the time-delay as $\tau = (d-1)h + \tau'$ with $d = 3$ and $\tau' = 0.6$. The sampled system can then be expressed by

$$x(kh + h) = \Phi x(kh) + \Gamma_0 u(kh - (d-1)h) + \Gamma_1 u(kh - dh)$$

where

$$\Phi = e^{-1} = 0.3679$$
$$\Gamma_0 = \int_0^{0.4} e^{-s} ds = 1 - e^{-0.4} = 0.3297$$
$$\Gamma_1 = e^{-0.4} \int_0^{0.6} e^{-s} ds = e^{-0.4} - e^{-1} = 0.3024$$

Introducing $x_1 = x$, $x_2 = u(k-3)$, $x_3 = u(k-2)$ and $x_4 = u(k-1)$ gives

$$x[k+1] = \begin{bmatrix} 0.3679 & 0.3297 & 0.3024 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} x[k] + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u[k]$$

4.

a. We replace s with $z - 1$.

$$U(z) = K\left(1 + \frac{1}{(z-1)^2} + b(z-1)\right) E(z)$$

$$(z-1)^2 U(z) = K\left((z-1)^2 + 1 + b(z-1)^3\right) E(z)$$

b. The resulting controller is not causal, and cannot be implemented.

c. With a low-pass filter on the derivate part the continuous-time controller structure would look like

$$U(s) = K(1 + \frac{1}{s^2} + \frac{bs}{(s+c)})$$

If we again replace s with $z - 1$ we will get

$$U(z) = K\left(1 + \frac{1}{(z-1)^2} + \frac{b(z-1)}{z-1+c}\right) E(z)$$

$$(z-1)^2(z-1+c)U(z) = K\left((z-1)^2(z-1+c) + (z-1+c) + b(z-1)^2\right) E(z)$$

This controller is causal and can be implemented.

5.

a. The control law for the state feedback controller is

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

where $L = [l_1\ l_2]$. The poles of the closed loop system are given by the characteristic polynomial that follows.

$$\begin{aligned} det(zI-\Phi+\Gamma L) &= det\left(\begin{bmatrix} z & 0 \\ 0 & z \end{bmatrix} - \begin{bmatrix} 0.2 & 0 \\ -0.8 & 0.4 \end{bmatrix} + \begin{bmatrix} 0.1l_1 & 0.1l_2 \\ 0 & 0 \end{bmatrix}\right) = \\ &= z^2 + (0.1l_1 - 0.6)z + (-0.08l_2 - 0.04l_1 + 0.08) \end{aligned}$$

To place the poles in 0.2 and -0.2, the characteristic polynomial should be equal to $(z+0.2)(z-0.2) = z^2 - 0.04$. Imposing

$$z^2 + (0.1l_1 - 0.6)z + (-0.08l_2 - 0.04l_1 + 0.08) = z^2 - 0.04$$

gives the following system of equations:

$$\begin{aligned} 0.1l_1 - 0.6 &= 0 \\ -0.08l_2 - 0.04l_1 + 0.08 &= -0.04 \end{aligned}$$

From the first one, one obtains $l_1 = 0.6/0.1 = 6$. Substituting $l_1 = 6$ in the second equation it is possible to obtain $l_2 = -1.5$.
Finally, in order to have a static gain of 1

$$l_r = \frac{1}{C(I-\Phi+\Gamma L)^{-1}\Gamma} = \frac{-24 - 3l_1 + 4l_2}{4} = -12$$

and the equation of the controller becomes

$$u(k) = -12r(k) - [6 \ \ -1.5]x(k)$$

b. The observer is given by

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

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

$$\begin{aligned} \det(zI - (\Phi - KC)) &= \det\left(\begin{bmatrix} z & 0 \\ 0 & z \end{bmatrix} - \begin{bmatrix} 0.2 & 0 \\ -0.8 & 0.4 \end{bmatrix} + \begin{bmatrix} 0 & k_1 \\ 0 & k_2 \end{bmatrix}\right) = \\ & z^2 + (k_2 - 0.6)z + (0.08 - 0.8k_1 - 0.2k_2) \end{aligned}$$

and imposing that the poles are in 0.1 means imposing that the characteristic polynomial has the form $(z-0.1)(z-0.1)$ which is $z^2 - 0.2z + 0.01$.

$$z^2 + (k_2 - 0.6)z + (0.08 - 0.8k_1 - 0.2k_2) = z^2 - 0.2z + 0.01$$

gives us

$$k_2 - 0.6 = -0.2 \rightarrow k_2 = 0.4$$

and consequently

$$0.08 - 0.8k_1 - 0.2 \cdot 0.4 = 0.01 \rightarrow k_1 = -0.0125$$

The observer is therefore given by the equation

$$\hat{x}(k+1) = \begin{bmatrix} 0.2 & 0.0125 \\ -0.8 & 0.0 \end{bmatrix}\hat{x}(k) + \begin{bmatrix} 0.1 \\ 0 \end{bmatrix}u(k) + \begin{bmatrix} -0.0125 \\ 0.4 \end{bmatrix}(y(k) - [0 \quad 1]\,\hat{x}(k)) \ .$$

6.

a. Depending on the order of execution of the tasks, task B and C may be in a deadlock (and they may be also blocking task A). In fact, if task B acquires the lock for s4 and task C acquires the lock for s2, none of the attempts to lock these two semaphores will work afterwards. The problem can be solved using hierarchical locks and ensuring that the tasks lock semaphores in lexicographic order. The code for the tasks becomes the following (the only change is in the code for Task B).

Task A
```
...
lock(s1);
lock(s3);
...
unlock(s1);
lock(s5);
...
unlock(s3);
unlock(s5);
...
```

Task B
```
...
lock(s2);
lock(s3);
lock(s4);
...
...
unlock(s4);
unlock(s3);
unlock(s2);
...
```

Task C
```
...
lock(s2);
lock(s4);
...
...
...
...
unlock(s4);
unlock(s2);
...
```

b. Task A locks the semaphores in the set { s1, s3, s5 }. Task B locks the semaphores in the set { s2, s3, s4 }. Task C locks the semaphores in the set { s2, s4 }. The ceiling of a semaphore is the priority of the highest priority task that can lock that semaphore. This implies that $ceil(s1) = 5$, $ceil(s2) = \max\{9, 10\} = 10$, $ceil(s3) = \max\{5, 9\} = 9$, $ceil(s4) = \max\{9, 10\} = 10$ and $ceil(s5) = 5$.

7.

a. If you need a resolution of at least 0.1, you need to use 4 bits for the fractional part. In fact, $2^{-3} = 0.1250$ and $2^{-4} = 0.0625$, which means that 3 fractional bits is not enough and, thus, 4 fractional bits are needed. Given that 1 bit is used for the sign, only 3 bits remain for the integer part. The numbers are therefore represented using a $Q3.4$ representation.

b. With the given format, it is not possible to correctly represent the number 14.28. The maximum number that one can represent with a $Q3.4$ representation is 7.9375. To verify this, the number 14.28 would be converted into the decimal number $X = round(14.28 \cdot 2^4)$, which corresponds to 228. Converting the number into its binary representation one obtains 11100100. Looking at the first digit, it is possible to see that this is the representation of a negative number, therefore the amount of bits available was not enough to correctly encode the given number. In case you have not solved the previous point, a $Q2.5$ notation has even more restrictive ranges, therefore the same reasoning applies.

c. The decimal representation of the constant 5.5 using the $Q3.4$ notation is given by $round(5.5 \cdot 2^4) = 88$. The corresponding binary representation using 8 bits is 01011000. In case you have not solved the first subproblem, the constant 5.5 cannot be encoded with a $Q2.5$ notation. In fact, $round(5.5 \cdot 2^5) = 176$, whose binary representation is 10110000, where the sign bit is needed to

represent the decimal part of the number. Therefore with 2 bits for the integer part and 5 for the decimal one, it is not possible to represent the given constant.

8.

a. The sample rate must be above $10f_0$ to avoid aliasing of the second region of interest.

b. For $f_s > 6f_0$ there will be no aliasing. For $5f_0 < f_s < 6f_0$ the disturbance will be aliased outside the area of interest. For $4f_0 < f_s < 5f_0$ the disturbance will be aliased into the area of interest. For $f_s < 4f_0$ the signal of interest will be aliased. The sample rates that can be used are therefore $f_s > 5f_0$.

9.

a. The task set parameters are as follows.

$Task_i$	C_i	T_i	D_i
T1	3	5	5
T2	1	10	10
T3	4	X	X

With Earliest Deadline First, the sufficient and necessary condition for the taskset to be schedulable is that the utilization should be less or equal to 1. In this case, this means:

$$U = \sum_{i=1}^{3} \frac{C_i}{T_i} = \frac{3}{5} + \frac{1}{10} + \frac{4}{X} \leq 1.$$

and therefore

$$\frac{4}{X} \leq 1 - \frac{7}{10} = \frac{3}{10}$$

which means $X \geq 40/3$ or $X \geq 13.333$. The smallest period and deadline of the communication task is therefore 14 time units.

b. The schedule will repeat after a number of time units equal to the least common multiplier of the periods/deadlines of the tasks. The least common multiplier of 5, 10 and 14 is 70, so after 70 time units the schedule will repeat. In case you have used a period of 12 time units, the least common multiplier is equal to 60 time units.

10.

a. Consider the case when both the buffers are full and when the put-thread and the move-thread are waiting. Assume then that the get-thread notifies a thread. This may very well be the run-thread which then immediately will go to sleep again. This can continue until the second buffer is empty and the get-thread also goes to sleep. Then all threads are sleeping.

A similar problem can happen if both buffers are empty. The move-thread and the get-thread sleeps. The run-thread will run until full, but might if unlucky only wake up the get-thread, which will go to sleep again.

b. Changing `notify()` to `notifyAll()` solves the problem. This is actually not needed for the move method, as the other two threads always can run after the completion of `move()`.

11. The problem is that the two trains share a common resource (the track). One must therefore ensure that the two trains using the track is mutually exclusive. This is done by adding a mutex sempahore in the middle, as shown in Figure 1.

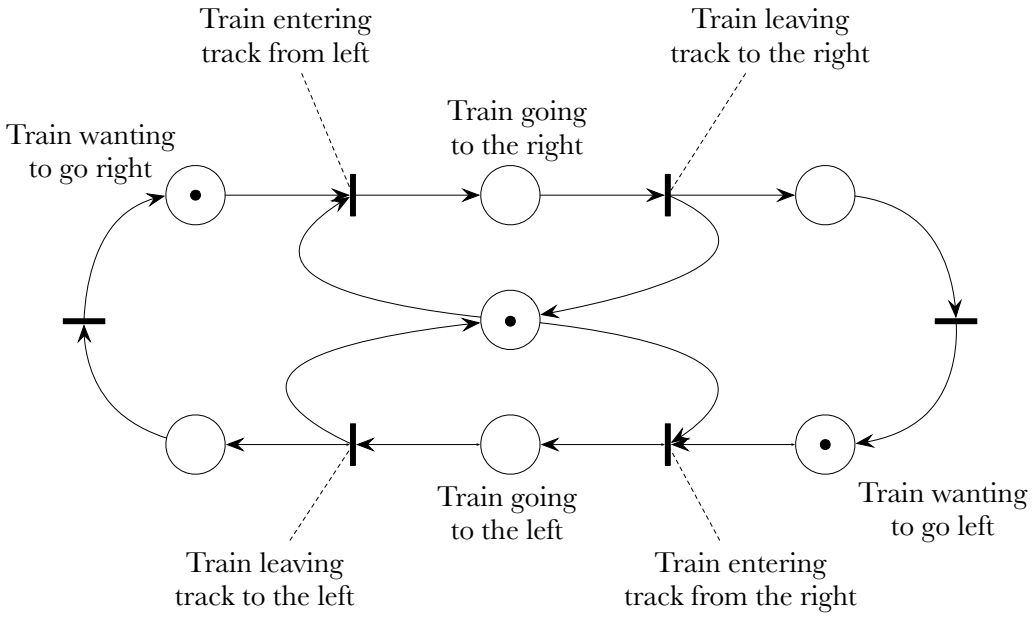

Figure 1 Solution for Problem **??**