Solutions to the exam in Real-Time Systems 150507

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

1.

a. The pulse-transfer function is given by:

$$
\begin{aligned}
H(z) &= C(zI-\Phi)^{-1}\Gamma + D \\
\Rightarrow H(z) &= \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} z-1/2 & 2 \\ 0 & z \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ -1 \end{pmatrix} = \\
&= \frac{1}{z(z-1/2)} \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} z & -2 \\ 0 & z-1/2 \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ -1 \end{pmatrix} = \\
&= \frac{2}{z(z-1/2)}
\end{aligned}
$$

So there are no zeroes and the poles are located in $z_1 = 0$ and $z_2 = 0.5$ which are both within the unit circle, and thus the system is asymptotically stable.

b.

$$
\begin{aligned}
Y(z) = \frac{2}{z^2-0.5z}U(z) &\Longleftrightarrow \\
z^2Y(z) - 1/2zY(z) &= 2U(z) \\
\Rightarrow y(k+2) - 0.5y(k+1) &= 2u(k)
\end{aligned}
$$

2.

a. The system without the delay L is equivalent to the system:

$$
\begin{aligned}
\dot{x} &= -\frac{1}{T}x + \frac{K}{T}u \\
y &= x
\end{aligned}
$$

It is now straightforward to use zoh-sampling which gives:

$$
\begin{aligned}
x(k+1) &= e^{-h/T}x(k) + K(1-e^{-h/T})u(k) \\
y(k) &= x(k)
\end{aligned}
$$

The discrete-time transfer function $\hat{H}(z)$ of this system is given by:

$$
\hat{H}(z) = \frac{K(1-e^{-h/T})}{z-e^{-h/T}}
$$

Adding a 3 sample delay z^{-3} to $\hat{H}(z)$ gives the discrete-time transfer function $H(z)$ we are looking for:

$$
H(z) = \frac{K(1-e^{-h/T})}{z^3(z-e^{-h/T})}
$$

b. In order to write the state space of the a system with 3 sample delay we have to introduce the 3 states $x_2(k) = u(k-1), x_3(k) = u(k-2), u_4(k) = u(k-3)$. This gives:

$$x(k+1) = \Phi x(k) + \Gamma u(k)$$
$$y(k) = Cx(k)$$

Where $x(k) = [x_1(k),\ x_2(k),\ x_3(k),\ x_4(k)]^T$,

$$\Phi = \begin{pmatrix} e^{-h/T} & 0 & 0 & K(1-e^{-h/T}) \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix},\ \Gamma = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix},\ C = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}^T.$$

.

3.

a. According to Theorem 8.1 in Real-Time Control Systems all deadlines will be met if

$$U = \sum_{i=1}^{i=n} \frac{C_i}{T_i} \le 1$$
$$\Rightarrow U = \frac{2}{10} + \frac{2}{6} + \frac{2}{8} = 0.7833 \le 1$$

So the task set is schedulable using EDF.

b. With RMS the priorities will be assigned according to the period time:

Task	Priority
A	low
B	high
C	medium

Using Theorem 8.3 the task set is schedulable if:

$$\sum_{i=1}^{i=n} \frac{C_i}{T_i} \le n\left(2^{1/n} - 1\right)$$
$$\Longleftrightarrow 0.7833 \le 3\left(2^{1/3} - 1\right) = 0.7798$$

which is a contradiction. However, if one uses the alternative hyperbolic bound test one has that

$$(1 + \frac{2}{10})(1 + \frac{2}{6})(1 + \frac{2}{8}) = 1.9999 \le 2$$

and, hence, the task set is schedulable. Alternatively one can use the exact response time analysis:

$$\begin{aligned}
&R_B^0 = 0, \qquad R_B^1 = C_B = 2, \qquad R_B^2 = C_B = 2\\
&R_C^0 = 0, \qquad R_C^1 = C_C = 2,\\
&R_C^2 = C_C + \left\lceil \frac{R_C^1}{T_B} \right\rceil C_B = 2 + \left\lceil \frac{2}{6} \right\rceil 2 = 2 + 2 = 4\\
&R_C^3 = 2 + \left\lceil \frac{4}{6} \right\rceil 2 = 2 + 2 = 4\\
&R_A^0 = 0, \qquad R_A^1 = C_A = 2,\\
&R_A^2 = C_A + \left\lceil \frac{R_A^1}{T_B} \right\rceil C_B + \left\lceil \frac{R_A^1}{T_C} \right\rceil C_C = 2 + \left\lceil \frac{2}{6} \right\rceil 2 + \left\lceil \frac{2}{8} \right\rceil 2 = 2 + 2 + 2 = 6\\
&R_A^3 = 2 + \left\lceil \frac{6}{6} \right\rceil 2 + \left\lceil \frac{6}{8} \right\rceil 2 = 2 + 2 + 2 = 6
\end{aligned}$$

Since $R_i \leq D_i$ for all i the task set is schedulable using RMS.

c. With DMS the priorities will be according to the deadline:

Task	Priority
A	medium
B	high
C	low

Using Theorem 8.5 the task set is schedulable if:

$$\sum_{i=1}^{i=n} \frac{C_i}{D_i} \leq n\left(2^{1/n} - 1\right)$$

$$\Longleftrightarrow \frac{2}{6} + \frac{2}{4} + \frac{2}{7} = 1.1190 \leq 3\left(2^{1/3} - 1\right) = 0.7798$$

Which is a contradiction and we will have to use exact analysis:

$$\begin{aligned}
&R_B = 2\\
&R_A^0 = 0, \qquad R_A^1 = 2\\
&R_A^2 = C_A + \left\lceil \frac{R_A^1}{T_B} \right\rceil C_A = 2 + \left\lceil \frac{2}{10} \right\rceil 2 = 2 + 2 = 4\\
&R_A^3 = C_A + \left\lceil \frac{R_A^2}{T_A} \right\rceil C_A = 2 + \left\lceil \frac{4}{10} \right\rceil 2 = 2 + 2 = 4\\
&R_C^0 = 0, \qquad R_C^1 = 2\\
&R_C^2 = C_C + \left\lceil \frac{R_C^1}{T_A} \right\rceil C_A + \left\lceil \frac{R_C^1}{T_B} \right\rceil C_B = 2 + \left\lceil \frac{2}{10} \right\rceil 2 + \left\lceil \frac{2}{6} \right\rceil 2 = 6\\
&R_C^3 = C_C + \left\lceil \frac{R_C^2}{T_A} \right\rceil C_A + \left\lceil \frac{R_C^2}{T_B} \right\rceil C_B = 2 + \left\lceil \frac{6}{10} \right\rceil 2 + \left\lceil \frac{6}{6} \right\rceil 2 = 6
\end{aligned}$$

Since the worst-case response time is less than the deadline for all the tasks the task set is schedulable using DMS.

The continuous time transfer functions for the first four systems are

$$
\begin{aligned}
G_1 &= 1/s \\
G_2 &= 1/(s+1) \\
G_3 &= 10/(s+2) \\
G_4 &= 1/(s^2+0.3s+1)
\end{aligned}
$$

where all but G_4 are first-order systems. G_4 exhibits an oscillating step response, which no first order system can do.

$H_5(z)$ contains a pole on the negative real axis. Thus, $H_5(z)$ can not be a ZoH-sampled version of a first order system ($\mathrm{Re}(e^{ph}) \geq 0 \ \forall p$ which means no poles in the left half plane).

$H_6(z)$ contains **two** poles in the **left** half plane which means $H_6(z)$ is of order two, further, the step response is oscillating.

Conclusion: Systems $H_1(z), ..., H_3(z)$ can and are ZoH-sampled versions of first order, continuous systems, which $H_4(z), ..., H_6(z)$ can **not** be.

5.

a. Studying the formula

$$f = |(f_1 + f_N) \bmod (f_s) - f_N|$$

and inserting $f = 20\ Hz,\ f_s = 50\ Hz,\ f_N = 25\ Hz$, gives the equation

$$20 = |(f_1 + 25) \bmod (50) - 25|$$

this gives the two possible solutions $(f_1 + 25) \bmod (50) = 5$ and $(f_1 + 25) \bmod (50) = 45$. In the range 25 to 500 Hz the first solution generates the possible frequencies
$f_1 = \{30, 80, 130, 180, 230, 280, 330, 380, 430, 480\}$ and the second solution generates the possible frequencies $f_2 = \{70, 120, 170, 220, 270, 320, 370, 420, 470\}$. So the complete answer is

$$\{30, 70, 80, 120, 130, 170, 180, 220, 230, 270, 280, 320, 330, 370, 380, 420, 430, 470, 480\}.$$

They could also be found by subtracting and adding 20 to multiples of the sampling frequency.

b. A second order deadbeat filter has no low-pass filtering properties.

6.

a. By approximating the s with $\frac{(z-1)}{z}$ we get the pulse transfer function

$$H_c(z) = K\left(1 + \frac{z}{T_1(z-1)}\right).$$

This can then be converted to the difference equation

$$u(k+1) - u(k) = \frac{K(T_i+1)}{T_i} e(k+1) - Ke(k).$$

b. In the difference equation in **a)** the two coefficients we need to convert to fixed point is $\frac{K(T_i+1)}{T_i}$ and K. Since the largest coefficient is in the interval $[4,8]$ we need $m = 3$ integer bits and are left with $n = 16 - m - 1 = 12$ fractional bits. The conversion to fixed point is done using round $\left(\frac{K(T_i+1)}{T_i} \cdot 2^{11}\right) = 19548$ and round $\left(K \cdot 2^{11}\right) = 17613$.

```
int16_t coeff1 = 19548;
int16_t coeff2 = 17613;
int n = 12;

int16_t do_control(int16_t e) {
  // Controller states
  static int16_t old_u, old_e;

  int16_t u = old_u
    + (int16_t) (((int32_t)coeff1*e) >> n)
    - (int16_t) (((int32_t)coeff2*old_e) >> n);
  old_u = u;
  old_e = e;
  return u;
}
```

7. The poles in the system are in 0.9 and 1.2. Since $|1.2| > 1$ it is unstable. The controllability matrix is $\begin{bmatrix} 0 & 0 \\ 1.1 & 1.32 \end{bmatrix}$, so the system is uncontrollable. However, even though it is uncontrollable it is possible to make it asymptotically stable since the uncontrollable state is already asymptotically stable. For example, the state-feedback vector $L = [\,0 \quad 1\,]$ makes the system asymptotically stable.

This can also be realized by studying the characteristic polynomial with the state feedback law $u = -Lx$, $L = [l_1, l_2]$.

$$\det(\lambda I - A + BL) = (\lambda - 0.9)(\lambda - 1.2 + 1.1 l_2)$$

here the unstable root can be "stabilized" with the parameter l_2.

8.

- The `synchronized`-block in the `put`-method must span the entire method block. After exiting the block we might have a race condition which may cause loss of data.
- One might think that having just an `if`-statement around the `wait` in the `get`-method should be enough since we use `notify` when we `put` data. But since both consumers waiting and producers waiting uses the same monitor, the `notifyAll` in `get` will also wake those threads that are waiting to get data. The `put`-method must call `notifyAll` instead of `notify` and make the `if`-statement a `while`-loop.
- Another reason why the `if`-statement in `get` must be made a `while`-loop is to protect against spurious wakeups.

9.

a. The simplest way to show this is to find a simple counter-example, e.g., $G_1(s) = G_2(s) = 1/s$. In this case we know, e.g. from Table 3 that $H_{G_1}(z) = h/(z-1)$. However, from the same table we have that $H_{G_1G_2}(z) = h^2(z+1)/2(z-1)^2$ which is not the same as $H_{G_1}(z)H_{G_2}(z) = h^2/(z-1)^2$.

b. For zero-order hold sampling the input should be a piece-wise constant signal. When two systems are connected in series this holds for the input to the first system, i.e., the system that is connected to the ZOH hold circuit, but it does not hold for the input to the second system.

c. By studying the matrices it is easily shown that the system consists of a double integrator (states x_1 and x_2) and of a first order system with a continuous-time pole in -1 (state x_3). Hence, the pulse transfer function is the sum of the pulse transfer function of the double integrator and the pulse transfer function of a first order system, which again according to Table 3 is

$$\begin{aligned} H(z) &= \frac{h^2(z+1)}{2(z-1)^2} + \frac{1-e^{-h}}{z-e^{-h}} \\ &= \frac{(h^2/2)(z+1)(z-e^{-h}) + (1-e^{-h})(z-1)^2}{(z-1)^2(z-e^{-h})} \end{aligned}$$

10.

a. The problem is that the run method is spelled with a capital R. Since Java is case-sensitive `run()` and `Run()` are two separate unrelated methods. The built-in Thread class contains a default `run()` method that is supposed to be overridden. This method does not do anything. In this case when the thread is started the default `run()` method will be started and nothing will happen.

b. The problem is still the same. However, the Java compiler requires that each class that implements an interface also provides implementations for all the methods defined in the interface. The `Runnable` interface contains the `run()` method. Since the code above does not provide any implementation for this method a compilation error is generated.