Solutions to the exam in Real-Time Systems 120410

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

1 a. From the zero-order hold sampling table we immediately get

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

b. The backward difference is found via the substitution $s' = \frac{z-1}{zh}$. This gives

$$H(z) = \frac{1}{1 + \frac{z-1}{zh}T} = \frac{zh}{zh + (z-1)T} = \frac{zh}{z(h+T) - T}$$

c. Zero-order hold sampling is used to sample a plant for the purpose of designing a discrete-time controller. Discretization using e.g. backward differences can be used to find a discrete-time approximation of a controller that has been designed in continuous time.

2. The sampled system is given by

$$\begin{aligned} x(k+1) &= \Phi x(k) + \Gamma_0 u(k) + \Gamma_1 u(k-1) \\ y(k) &= Cx(k) \end{aligned}$$

where

$$\Phi = e^{Ah} = I + Ah = \begin{pmatrix} 1 & 0 \\ h & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0.1 & 1 \end{pmatrix}$$

$$\Gamma_0 = \int_0^{h-\tau} e^{As} B ds = \int_0^{h-\tau} \begin{pmatrix} 10 \\ 10s \end{pmatrix} ds = \begin{pmatrix} 10(h-\tau) \\ 5(h-\tau)^2 \end{pmatrix} = \begin{pmatrix} 0.8 \\ 0.032 \end{pmatrix}$$

$$\Gamma_1 = e^{A(h-\tau)} \int_0^{\tau} e^{As} B ds = \begin{pmatrix} 1 & 0 \\ h-\tau & 1 \end{pmatrix} \begin{pmatrix} 10\tau \\ 5\tau^2 \end{pmatrix} = \begin{pmatrix} 10\tau \\ 10\tau(h-\tau) + 5\tau^2 \end{pmatrix} = \begin{pmatrix} 0.2 \\ 0.018 \end{pmatrix}$$

3.

a. From the figures it follows that the task parameters are

Task name	T_i	D_i	C_i
A	4	4	1
B	6	6	2
C	8	8	3

and that the CPU utilization is

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} = \frac{1}{4} + \frac{2}{6} + \frac{3}{8} = 0.9583$$

b. The scheduling policy in the first figure is fixed priority scheduling The priority assignment is rate-monotonic. Task A has the shortest period and hence the highest priority and is therefore not preempted by any other tasks. Task B with medium priority is only preempted by Task A, etc.

The scheduling policy in the second figure is EDF scheduling. The task with the earliest absolute deadline has always the highest priority.

c. In the first case. since $U = 0.9582 > 3(2^{1/3} - 1) = 0.7798$ we cannot use the sufficient schedulability condition. The best-case respone time analysis gives the following:

$$R_A^0 = 0, R_A^1 = C_A = 1 \leftarrow R_A = 1 < D_A$$

$$R_B^0 = 0, R_B^1 = C_B = 2$$

$$R_B^2 = 2 + \left\lceil \frac{1}{4} \right\rceil 1 = 3$$

$$R_B^3 = 2 + \left\lceil \frac{3}{4} \right\rceil 1 = 3 < D_B$$

$$R_C^0 = 0, R_C^1 = C_C = 3$$

$$R_C^2 = 3 + \left\lceil \frac{3}{4} \right\rceil 1 + \left\lceil \frac{3}{6} \right\rceil 2 = 6$$

$$R_C^3 = 3 + \left\lceil \frac{6}{4} \right\rceil 1 + \left\lceil \frac{6}{6} \right\rceil 2 = 7$$

$$R_C^3 = 3 + \left\lceil \frac{7}{4} \right\rceil 1 + \left\lceil \frac{7}{6} \right\rceil 2 = 9 > D_C$$

Hence the task set is not schedulable under fixed priority scheduling using rate-monotonic priority assignment.

For EDF the necessary and sufficient schedulability condition is $U \leq 1$ which is fulfilled here. Hence, the task set is schedulable under EDF scheduling.

4.

a. Signed variables means 7 bits for actual value (i.e. 127). The largest coefficient is 1.912. It is enough with one integer bit to represent this, leaving 6 bits for the fractional part. The value of a coefficient C for 6 number of fractional bits are round$(2^6 * C)$, i.e. $0.405 = 26$, $0.3539 = 23$, $1.912 = 122$, $1.583 = 101$, $0.6703 = 43$.

b.

```
int8_t filter(int8_t u) {
    static int8_t b1 = ..., b0 = ..., ... ;
    static int8_t yold1 = 0, yold2 = 0, yold3 = 0;
    static int8_t uold1 = 0 uold2 = 0, uold3 = 0;
    int16_t y16;
    int8_t y;
    int N = 6;

    y = ((int16_t)-a2*yold1 - (int16_t)a1*yold2 -
```

```
            (int16_t)a0*yold3 + (int16_t)b1*uold2 +
            (int16_t)b0*uold3)>>N;


        /* Perform suitable limitation of the return value */
        if (y16 > (2<<N-1))
            y = 2<<N-1;
        else if (y16 < -(2>>N))
            y = -2>>N;
        else
            y = y16;

        yold3 = yold2; yold2 = yold1; yold1 = y;
        uold3 = uold2; uold2 = uold1; uold1 = u;

        return y;
    }
```

5.

a.

$$\begin{aligned}
U(s) &= K(\beta Y_{ref}(s) - Y(s) + \frac{1}{sT_I}E(s) + sT_D(\gamma Y_{ref}(s) - Y(s)) \\
&= K(E(s) + \frac{1}{sT_I}E(s) + sT_D E(s)) + K(\beta - 1)Y_{ref}(s) + sT_D K(\gamma - 1)Y_{ref}(s) \\
&= K(1 + \frac{1}{sT_I} + sT_D)E(s) + (K(\beta - 1) + sT_D K(\gamma - 1))Y_{ref}(s) \\
&= FB(s)E(s) + FF(s)Y_{ref}(s)
\end{aligned}$$

b. With the parameter values inserted $FB(s)$ can be written as

$$\begin{aligned}
FB(s) &= K(\frac{T_I s + 1 + T_I T_D s^2}{T_I s}) \\
&= \frac{5s + 1 + s^2}{5s}
\end{aligned}$$

The transfer function has a pole in $s = 0$ and two zeros in $s = -4.791$ and $s = -0.209$.

c. Replacing s with $2(z-1)/h(z+1)$ in $FB(s)$ gives poles in $z = -1$ and $z = 1$.

6.

a.

```
public class DataStore {
    private Cloneable data;

    public synchronized void putData(Cloneable newData) {
```

```
            data = newData.clone();
            notifyAll();
        }

        public synchronized Cloneable getData() {
            return data.clone();
        }

        public synchronized void waitForData() {
            wait();
        }
    }
```

b. The phenomenon is called priority inversion and comes from that the lowest level thread can, while accessing the data object, be preempted by the middle level threads and hence prevent the highest level thread from executing. Ways to solve this include the Priority Inheritance protocol, Priority Ceiling protocol or the Immediate Inheritance protocol.

7.

a. The only restriction we have is that the total computation time in each procedure chain has to be less than the base period time 10 [ms]. This is achieved with e.g. the assignments $P_1^1 := \tau_1$, $P_2^1 := \tau_2$, $P_2^2 := \tau_3$, $P_4^1 := \tau_6$, $P_4^2 := \tau_4$, $P_4^3 := \tau_7$, $P_4^4 := \tau_5$.

b. The tasks τ_6, τ_4, τ_7 and τ_5 always start at a clock pulse, and the jitter for these tasks are thus zero. The start times (relative to a clock pulse) of τ_2 are calculated by merging the best and worst case start times of the two possible combinations involving execution of τ_2. The resulting interval is [3.5 5.0] and the jitter for τ_2 is thus 1.5. The starting times for τ_3 are in the interval [1.5 2], which gives the jitter 0.5. By merging the best and worst case starting times of all four combinations the starting time interval [6 8] is obtained for τ_1, and the jitter for this process is thus 2.