Solutions to the exam in Real-Time Systems, Jan 10, 2012

These solutions are available on WWW:
http://www.control.lth.se/Education/EngineeringProgram/FRTN01.html

1.

a. Since A is nilpotent ($A^2 = 0$), the Taylor series expansion method is straightforward.

$$\Phi = e^{Ah} = I + Ah + \frac{(Ah)^2}{2} + ... = \begin{pmatrix} 1 & 0 \\ 3h & 1 \end{pmatrix}$$
$$\Gamma = \begin{pmatrix} 0 \\ h \end{pmatrix}$$

b. Here the Laplace transform method is the most convenient.

$$\Phi = e^{Ah} = \mathcal{L}^{-1}(sI - A)^{-1} = \begin{pmatrix} 1 & \frac{1}{4}(1 - e^{-4h}) \\ 0 & e^{-4h} \end{pmatrix}$$
$$\Gamma = \frac{1}{4}\begin{pmatrix} h + \frac{1}{4}e^{-4h} - \frac{1}{4} \\ 1 - e^{-4h} \end{pmatrix}$$

2.

a.

$$H_1(z) = \frac{z+2}{z^2 - 0.25} = \frac{z+2}{(z+0.5)(z-0.5)}$$
$$H_2(z) = \frac{z-1}{z^2 - 3.5z + 1.5} = \frac{z-1}{(z-0.5)(z-3)}$$

b. The zeros and poles are

	H_1	H_2
Zeros	-2	1
Poles	0.5 and -0.5	0.5 and 3

c. For the first system the static gain is

$$H_1(1) = \frac{1+2}{1-0.25} = 4$$

The second system has a pole outside the unit circle, and hence infinite static gain.

3.

a. The CPU utilization is

$$U = \sum_i \frac{C_i}{T_i} = \frac{1}{40} + \frac{1}{4} + \frac{1}{50} = 0.295$$

b. The normal schedulabiliy test for EDF, i.e. $U \leq 1$ only holds when $D_i = T_i$ and that is not the case here. Although more elaborate schedulability tests exist they are not part of the course requirements. Hence, the only possibility left is to draw the schedule for one hyperperiod (= 200) and check manually that all the deadlines are met. This quite tedious work leads to the answer that all deadlines will be met for the task set.

Accidentally this subproblem turned out more difficult than anticipated. This will be taken into account in the grading.

c. Yes. A sufficient condition for schedulability with DM scheduling is

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

In this example we have

$$\sum_{i=1}^{n} \frac{C_i}{D_i} = \frac{1}{40} + \frac{1}{4} + \frac{1}{2} = 0.775$$

$$n(2^{1/n} - 1) = 3(2^{1/3} - 1) \approx 0.7798$$

d. The sufficient conditions for rate monotonic schedulability are only valid when $D_i = T_i$, so we need to do an exact analysis. The priorities are in descending order B, A, C. The worst case response times R_i for the different tasks are

$$R_B = C_B = 1 \leq D_B = 4$$

$$\begin{aligned}
R_A^1 &= C_A = 1 \\
R_A^2 &= C_A + \left\lceil \frac{R_A^1}{T_B} \right\rceil C_B = 1 + \left\lceil \frac{1}{4} \right\rceil 1 = 2 \\
R_A^3 &= C_A + \left\lceil \frac{R_A^2}{T_B} \right\rceil C_B = 1 + \left\lceil \frac{2}{4} \right\rceil 1 = 2 \leq D_A = 40
\end{aligned}$$

$$\begin{aligned}
R_C^1 &= C_C = 1 \\
R_C^2 &= C_C + \left\lceil \frac{R_C^1}{T_A} \right\rceil C_A + \left\lceil \frac{R_C^1}{T_B} \right\rceil C_B = 1 + \left\lceil \frac{1}{40} \right\rceil 1 + \left\lceil \frac{1}{4} \right\rceil 1 = 3 > D_C = 2
\end{aligned}$$

Hence the deadline for task C can not be guaranteed to be met.

4.

a. The largest value is 14, requiring 4 integer bits, leaving $n = 16-1-4 = 11$ fractional bits.

b. The fixed-point representation of a coefficient a can be calculated as $A = \text{round}(a \cdot 2^n)$. This results in

$$x(k+1) = \begin{pmatrix} 16998 & 12288 \\ 6554 & 28672 \end{pmatrix} x(k) + \begin{pmatrix} 0 \\ 512 \end{pmatrix} u(k)$$

5. The three main improvements are:

1. There is no synchronization between calculateOutput and updateState. This makes it possible for the GUI to update the controller parameters between the call to calculateOuput and the call to updateState.
2. The input output latency is not minimized.
3. The sleep does not take into account the execution time of the algorithm.

The improved, but not perfect, code is shown below (only the run() method):

```
.....

  public void run() {
    double u;
    double y;
    double ref;
    long diff;
    long t = System.currentTimeMillis();

    while (true) {
      y = analogIn.get();
      ref = referenceGenerator.getRef();
      synchronized (controller) {
        u = limit(controller.calculateOutput(y, ref), uMin, uMax);
        analogOut.set(u);
        controller.updateState(u);
      }
      t = t + controller.getSamplingPeriod();
      duration = t - System.currentTimeMillis();
      try {
        sleep(duration);
      } catch (InterruptedException e) {
      }
    }
  }
```

6.

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

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

The desired characteristic polynomial is

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

Equating the coefficients we get

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

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 behaviour when used as an input to the process.

The desired characteristic polynomial of the model is given by

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

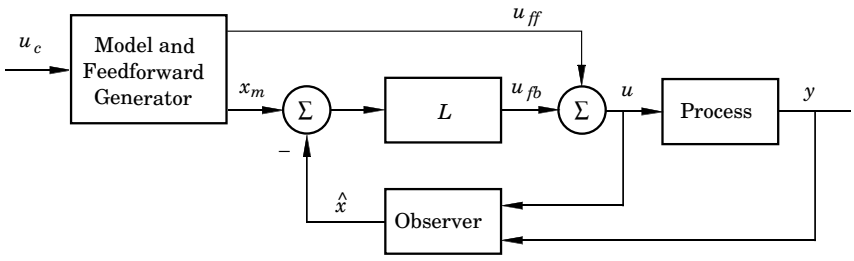
From the first sub-problem it follows that the coefficients of L_m should be chosen as

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

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

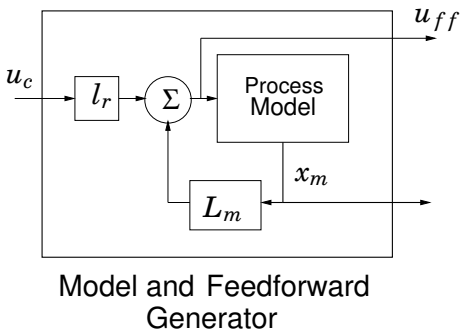
$$l_r = \frac{1}{C(I - \Phi + \Gamma L_m)^{-1}\Gamma} = 3.2$$

d. The block diagram is given by Figure 1 with the internal structure of the



Figur 1 Block Diagram

model and feedforward generator given by Figure 2.

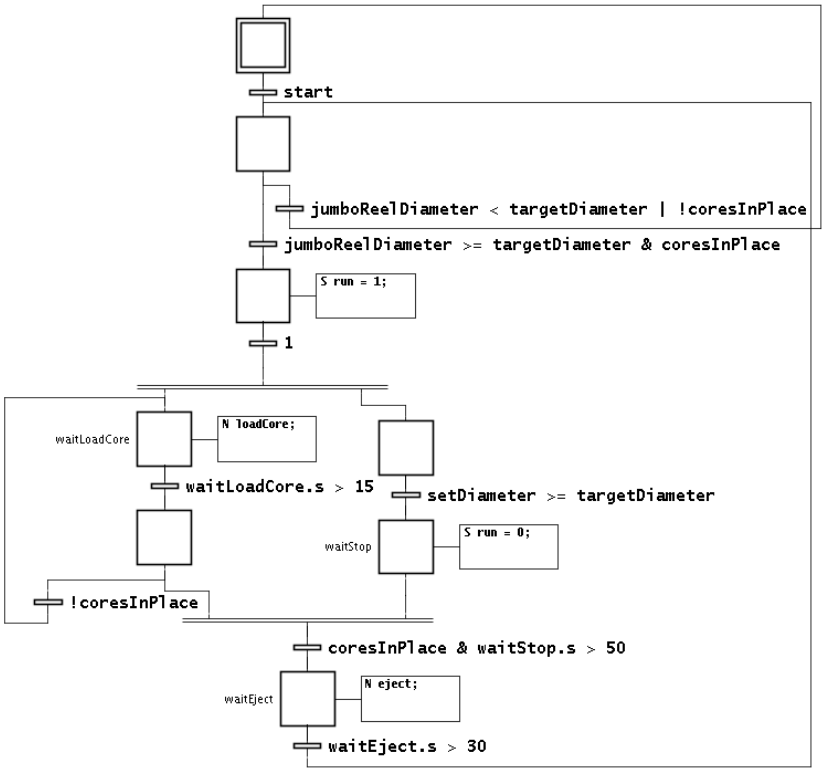


Figur 2 Internal structure of model and feedforward generator

e. 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:
$\hat{x} = \Phi\hat{x} + \Gamma u + K(y - C\hat{x})$
$x_m = \Phi x_m + \Gamma u_{ff}$
$u_{ff} = -L_m x_m$
$u = L(x_m - \hat{x})$

7. The solution is shown in Figure 3.



Figur 3 Winder Grafcet