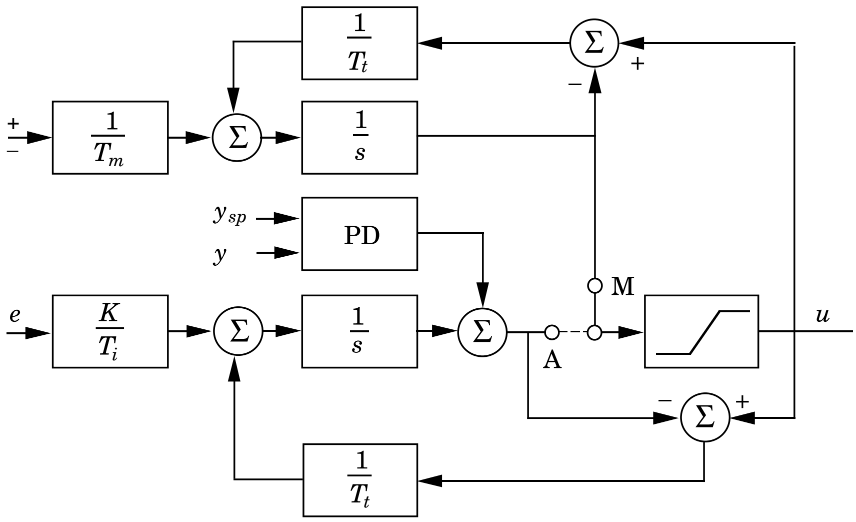
Solutions to the exam in Real-Time Systems, Dec 12, 2011

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

1. Priority inheritance solves the problem with priority inversion. Priority inversions occurs when a lower priority process blocks a higher priority process without any shared resources directly involved between the lower priority process and the higher priority process. The typical case is when we have a low priority process and a high priority process that communicate using a shared resource, e.g., a monitor, wnd when there in addition is a medium priority process which does not access the monitor. The situation occurs when the monitor is held by the low priority process and the high process wants to access it. If there is a medium priority process that is readý for execution it will prevent the low priority process from finishing its work inside the monitor and, hence, indirectly block the high priority process.

With priority inheritance the priority of the process that holds a monitor will be raised to the priority of the process that wants to enter the monitor, and later reset again when it leaves the monitor.



2.

3. G_1 has a pole in zero, which means that it should exhibit deadbeat behavior. G_2 on the other hand is an integrator, which means that its pulse response should be a constant. G_3 has a pole on the negative real axis, and its response is on the form $x(n) = -0.5x(n-1) = 0.25x(n-2) = \cdots = (-0.5)^n x(0)$. G_4 on the other hand has a pole in 0.5, which corresponds to the response $x(n) = 0.5^n x(0)$. Therefore the matching is:

$$
\begin{aligned}
G_1 &\to 1\\
G_2 &\to 4\\
G_3 &\to 3\\
G_4 &\to 2
\end{aligned}
$$

4.

a.

$$U(z) = K(1 + T_d \frac{z-1}{h})E(z)$$

b.

$$U(z) = K(1 + \frac{2T_d}{h}\frac{z-1}{z+1})E(z)$$

c. Using forward difference the controller becomes non-causal and, hence, cannot be implemented.

5.

a. Since the A-matrix is nilpotent, $\Phi = \exp(Ah)$ can be computed by series expansion.

$$A = \begin{pmatrix} 0 & 1 & -1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

$$A^2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$A^3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\Phi = \exp(Ah) = I + Ah + A^2h^2/2 = \begin{pmatrix} 1 & h & h^2/2 - h \\ 0 & 1 & h \\ 0 & 0 & 1 \end{pmatrix}$$

$$\Gamma = \int_0^h \exp(As)dsB = \int_0^h \begin{pmatrix} s^2/2 - s \\ s \\ 1 \end{pmatrix} ds = \begin{pmatrix} h^3/6 - h^2/2 \\ h^2/2 \\ h \end{pmatrix}$$

The discrete time system is

$$x(t+h) = \begin{pmatrix} 1 & h & h^2/2 - h \\ 0 & 1 & h \\ 0 & 0 & 1 \end{pmatrix} x(t) + \begin{pmatrix} h^3/6 - h^2/2 \\ h^2/2 \\ h \end{pmatrix} u(t)$$

$$y(t) = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} x(t)$$

b. For small values of h, the elements of Γ and the off-diagonal elements of Φ will be very small, and the diagonal elements of Φ are equal to 1. This means that at each time step, the new value of the state will be equal to the old value of the state plus a small increment. If this increment is too small, it will be rounded to zero.

6.

a.

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

where

$$\begin{aligned} \Phi &= e^{-h} = 0.6065 \\ \Gamma_1 &= e^{-(h-\tau)} 2(1-e^{-\tau}) = 2e^{-(h-\tau)} - 2e^{-h} = 0.5966 \\ \Gamma_0 &= 2(1-e^{-(h-\tau)}) = 0.1903 \end{aligned}$$

b. The augmented state space system with the state vector $z(kh) = [x(kh)\ u(kh-h)]^T$ is

$$z(kh+h) = \Phi_a z(kh) + \Gamma_a u(kh)$$

where

$$\begin{aligned} \Phi_a &= \begin{pmatrix} \Phi & \Gamma_1 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 0.6065 & 0.5966 \\ 0 & 0 \end{pmatrix} \\ \Gamma_a &= \begin{pmatrix} \Gamma_0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0.1903 \\ 1 \end{pmatrix} \end{aligned}$$

c. The desired characteristic polynomial is

$$z(z - e^{-1}) = z^2 - 0.3679z$$

With $L = (l_1\ l_2)$ the characteristic polynomial is

$$\begin{aligned} \det(zI - (\Phi_a - \Gamma_a L)) &= \det \begin{pmatrix} z - (0.6065 - 0.1903l_1) & 0.1903l_2 - 0.5966 \\ l_1 & z + l_2 \end{pmatrix} \\ &= z^2 + (l_2 + 0.1903l_1 - 0.6065)z + 0.5966l_1 - 0.6065l_2 \end{aligned}$$

Setting the corresponding coefficients equal leads to the following two equations

$$\begin{aligned} l_2 + 0.1903 l_1 - 0.6065 &= -0.3679 \\ 0.5966 l_1 - 0.6065 l_2 &= 0 \end{aligned}$$

with the solution

$$\begin{aligned} l_1 &= 0.2032 \\ l_2 &= 0.1999 \end{aligned}$$

For unit gain l_r should be

$$l_r = \frac{1}{C(I - \Phi_a + \Gamma_a L)^{-1}\Gamma_a} = \frac{1}{1.245} \approx 0.8032$$

d.
```
while(1) {
y = getY();
r = getRef();

// CalculateOutput code
u = lr*r - l1*y - temp;
setOutput(u);
// UpdateState code
uold = u;
temp = l2*uold;
// Sleep code
}
```

7.

a. The sufficient Liu-Layland test gives that the smallest value of the period is $T_{min} = 4.77$. The hyperbolic schedulability test gives that $T_{min} = 4.5$. However, by response time analysis arguments or simply by drawing the schedule it can easily be seen that $T_{min} = 4$.

b. The sufficient Liu-Layland test gives that the largest value of the execution time is $C_{max} = 3.1421$. The hyperbolic schedulability test gives that $C_{max} = 3.33$. However, by response time analysis arguments or simply by drawing the schedule it can easily be seen that $C_{max} = 4$.

c. The response time analysis formulae, i.e.,

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

consists of two terms. The first one is the execution time for the task itself and the second is the amount of time that the task is preempted by higher priority task. We can use the same approach to calculate the maximum starting time, however, in the analysis we should pretend that the execution time for the task is very small, i.e., equal to ϵ, and then we let ϵ approach zero.

Hence. the formulae for the latest possible starting time can be formulated as

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

If we apply this to our task set we will get the following result:

$$\begin{aligned}
S_A &= \epsilon \\
S_B^0 = 0, S_B^1 &= \epsilon, S_B^2 = \epsilon + \lceil \frac{\epsilon}{1} \rceil 0.2 = \epsilon + 0.2 \\
S_B^3 &= \epsilon + \lceil \frac{\epsilon + 0.2}{1} \rceil 0.2 = \epsilon + 0.2 \\
S_C^0 = 0, S_C^1 &= \epsilon, S_C^2 = \epsilon + \lceil \frac{\epsilon}{1} \rceil 0.2 + \lceil \frac{\epsilon}{4} \rceil 1.2 = \epsilon + 1.4 \\
S_C^3 &= \epsilon + \lceil \frac{\epsilon + 1.4}{1} \rceil 0.2 + \lceil \frac{\epsilon + 1.4}{4} \rceil 1.2 = \epsilon + 1.6 \\
S_C^4 &= \epsilon + \lceil \frac{\epsilon + 1.6}{1} \rceil 0.2 + \lceil \frac{\epsilon + 1.6}{4} \rceil 1.2 = \epsilon + 1.6
\end{aligned}$$

If we now let ϵ go to 0 we will have that $S_A = 0, S_B = 0.2, S_C = 1.6$. The same result can also be derived by simply drawing the schedule, starting from the critical instant.

8.

a. The following queues are involved:

- the ReadyQueue
- the TimeQueue
- the monitor queue associated with the monitor `mon`
- the waiting queue associated with the event (condition variable) `nonFull`
- the waiting queue associated with the event (condition variable) `nonEmpty`

b. There may be a scenario where none of the processes is blocked, i.e., the Readyqueue may contain maximum 7 processes (8 if we also count the Idle process).

There may be a scenario in which all the processes are sleeping, i.e., the TimeQueue may contain maximum 7 processes.

There may be a scenario in which there is one process executing inside the monitor and all the others are waiting for access, i.e., the monitor queue may contain maximum 6 processes.

There may be a scenario in which all the producer processes wants to enter a data element into a full buffer, i.e., the waiting queue associated with `nonFull` may contain maximum 3 processes.

There may be a scenario in which all the consumer processes wants to extract a data element from an empty buffer, i.e., the waiting queue associated with `nonEmpty` may contain maximum 4 processes.

9.
```
int16_t multiply(int16_t X, int16_t Y, int16_t n)
{
  int32_t Z;

  Z = ((int32_t)X*Y) >> n;
  if (Z > 32767)
    Z = 32767 ;
  if (Z < -32768)
    Z = -32768;

  return (int16_t)Z;
}
```