Real-Time Control Systems

Exam 2009-03-11

8:30 – 12:30: Halls at “Väg och vatten”

Course code: SSY190

Teacher: Knut Åkesson, phone 0701-749525

The teacher will visit examination halls twice to answer questions. This will be done approximately one hour after the examination started and one hour before it ends.

The exam comprises 30 credits. For the grades 3, 4, and 5, is respectively required 15, 20 and 25 credits.

Solutions and answers should be complete, written in English and be unambiguously and well motivated. In the case of ambiguously formulated exam questions, the suggested solution with possible assumptions must be motivated. The examiner retains the right to accept or decline the rationality of assumptions and motivations.

Solutions will be announced on the course web-page on the first week-day after the exam date. *Exam results* will be announced on the department notice board on the latest 2009-03-25 at 12:30. *The results* are open for review 2009-03-25, 12:30-13:30 at the department.

Allowed aids:

- An A4 sheet with handwritten notes on one page. You should hand-in your notes together with your solutions.
- A pocket calculator with erased memory.
- Dictionary (paper and electronic) between English and the students native language.

1

a) Describe why "race condition" is a problem in concurrent programming.

(1p)

b) Describe the priority inversion problem and present a solution for how to handle it.

(2p)

c) Describe two low-level ways, i.e. not using semaphores or monitors, to implement a critical region.

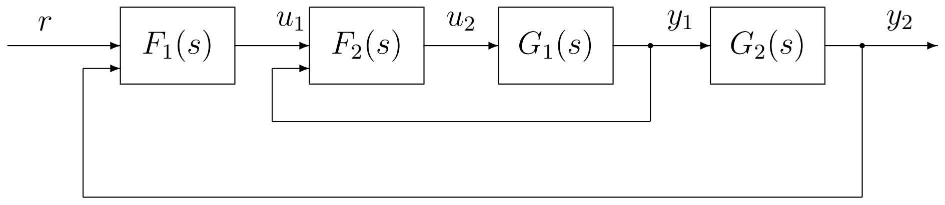
(1p)

d) Describe two reasons for jitter between the AD-conversion of the inputs and DA-conversion of the outputs in a controller, i.e. during the calculation of an updated control signal.

(2p)

2

Consider the cascade setup below with the two controllers F_1 and F_2 and the two plants G_1 and G_2.



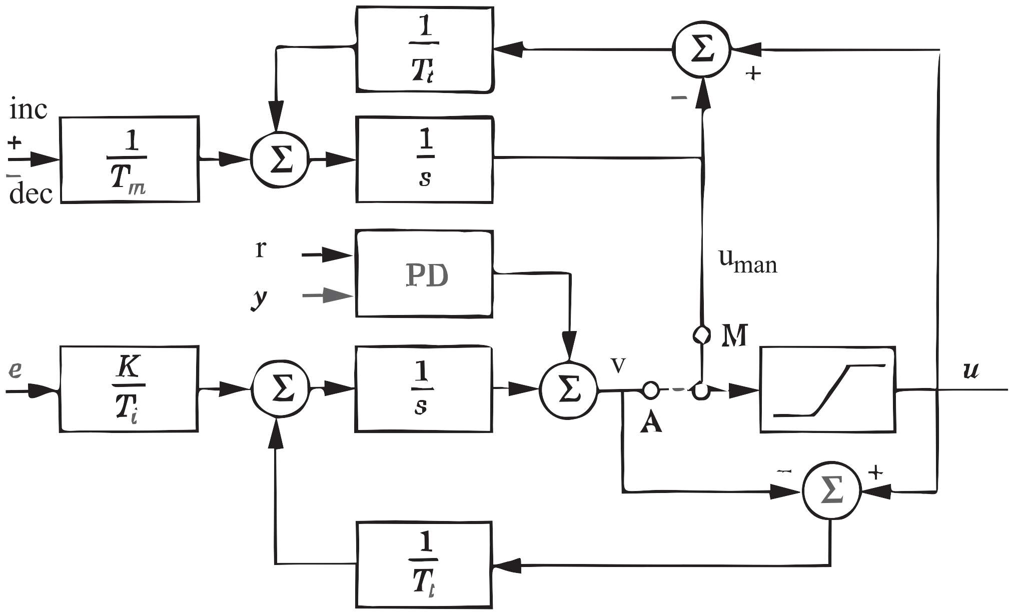
The two controllers will have different sampling periods, h_1 and h_2. Each controller will be implemented in its own thread. The controller F_1 writes to u_1 which is a shared variable that might then be read by F_2. You will write pseudo-code for implementing this setup, on the last page you will find the operations that the real-time operating system provides to you.

Write pseudo-code for implementing F_1 and F_2. You can assume that the control algorithms do exists (i.e., you do not have to provide an implementation for `calculateOutput1`, `updateState1`, `calculateOutput2`, `updateState2`). However, your code should make sure that control output are updated periodically with minimum IO-latency and jitter. Protect concurrent reading and writing to u_1 by treating the reading and writing to u_1 as a critical region. Use semaphores to implement the critical region for u_1.

(3p)

3

Consider the block diagram of a PID-controller. The figure shows a PID controller with anti-reset windup based on tracking with internal control signal limitation, support for manual control mode with increment/decrement action, and bumpless mode changes. e is the control error, r is the set-point value, y is the output from the process.



Write pseudo-code for implementing the PID-controller above. The derivation should act on the measured signal y only. Motivate any assumptions.

(4p)

Hint: By using backward difference approximation the discretization of an integral action (assuming sampling interval h),

$$I(s) = \frac{K}{T_i s}$$

is given by

$$I(t_{k+1}) = I(t_k) + \frac{Kh}{T_i} e(t_{k+1})$$

When there is no derivate action on the set-point signal the transfer function is given by

$$D(s) = -\frac{sT_d}{1 + sT_d/N} Y(s)$$

by using backward difference approximation the discrete approximation is given by

$$D(t_k) = \frac{T_d}{T_d + Nh} D(t_{k-1}) - \frac{KT_dN}{T_d + Nh}(y(t_k) - y(t_{k-1})).$$

4

A system consists of three processes, P_1, P_2, and P_3. Each process executes the following sequences of semaphore operations. The system has three sempahores, all initialized to 1. Intermediate code that depends on the semaphore is represented by the function call "useXY" where "XY" denotes that semaphores X and Y must be taken when the code is executed.

P_1	P_2	P_3
wait(C)	wait(B)	wait(C)
wait(A)	useB()	use(C)
useAC()	signal(B)	wait(A)
signal(C)	wait(A)	wait(B)
wait(B)	useA()	useABC()
useAB()	wait(B)	signal(C)
signal(A)	useAB()	useAB()
signal(B)	signal(A)	signal(A)
	signal(B)	signal(B)

May the above system deadlock? If the system contains a deadlock situation describe it, if it does not explain why.

(2p)

5

You are about the implement a state feedback controller for a plant where not all states are observable. To deal handle the situation you have designed an observer.

$$\begin{aligned} \frac{d\hat{x}(t)}{dt} &= (A - BL - KC)\hat{x}(t) + Ky(t) + BL_r r(t) \\ u(t) &= -L\hat{x}(t) + L_r r(t) \end{aligned}$$

where y is the measurement signal, u is the control signal, r is the reference signal, and $\hat{x}$ is the estimated state.

Discretize the controller using the backward difference. Let the sampling interval be h. For simplicity you can assume that all variables are scalar, i.e. there is only one state to be estimated. Simplify the answer as far as possible.

(3p)

6

When scheduling among controller processes it is often the case that the relative deadline (D) is less than the period (T). Standard rate monotonic scheduling assume that the relative deadline is equal to the period. In deadline montonic is is the relative deadline of the task instead of the period of the task that determine the priority. In deadline monotonic scheduling the shorter relative deadline the higher priority of the task. Consider the following task set. C is th worst case execution time.

Task name	T (ms)	D (ms)	C (ms)
A	30	30	8
B	40	20	8
C	60	55	20

a) Determine the utilization.

(1p)

b) Determine if it is possible to meet all deadlines with deadline monotonic scheduling for the task set above.

(4p)

c) For task C the relative deadline is 55 ms. Explain how many degrees that the phase-margin should be increased in the control design phase to handle this latency. Assume that the cross-over frequency $\omega_c = 3$ rad/s.

(2p)

7

Two communication protocols were discussed in the course, CAN and Ether-Cat.

a) Consider three nodes A, B and C that are sending the frames given below on a CAN-bus. Assume that the three nodes want to send the frames simultaneously.

Frame	Message Identifier
A	1011 0101 0010
B	1101 0101 1100
C	1011 1101 1100

Describe the arbitration phase, i.e. what bits the nodes are sending and what bits that are read back from the bus.

(2p)

b) Describe how bitstuffing is used in the CAN-bus protocol.

(1p)

c) Describe why standard TCP/IP Ethernet is not suitable for communication between a controller and sensors and actuators. Describe how the EtherCat protocol handle these problems.

(2p)

Good Luck!

Real-Time Operating System API

When writing your pseudo code you can assume that the Real-Time Operating System provide an implementation for the following operations.

```
Sempahore
=========
initialize(sem, value)
wait(sem)
signal(sem)

Monitor
=======
acquire(mutex)
release(mutex)
await(condition)
cause(condition)

Time
====
currentTime(t)
incTime(t, h)
wait(h)
waitUntil(t)

AD/DA conversion
================
y = AD_read("y")
DA_write("u", u)
```

1/

a/ A race condition occurr when the result of executing concurrent threads depends on the timing of context switches.

A typical example is when two threads updates a shared variable that is not protected by a critical region.

See section 4.1 in the book.

b/ A high priority process will be delayed by both a low priority process that is holding a lock on a resource that the high priority process tries to access, and possibly a set of unrelated medium priority processes.

See section 4.4 in the book.

c/ i) Turn of interrupts

ii) Use test-and-set instruction provided by the cpu

iii) Use Dekkers algorithm.

d/ i) Higher priority process could preempt the current process

ii) Variations in code execution time

iii) Blocking due to accessing shared resources

iv) Nondeterministic computing platform

2/ initialize (mutex, 1)

Task F1

```
currentTime(t1)
while(1)
{
    r = AD_read("r")
    y2 = AD_read("y2")
    wait(mutex)
        u1 = calculateOutput1(r, y2)
    signal(mutex)
    updateState1()
    incTime(t1, h1)
    waitUntil(t1)
}
```

Task F2

```
currentTime(t2)
while (1)
{
    y1 = AD_read("y1")
    wait(mutex)
    u2 = calculateOutput2(u1, y1)
    signal(mutex)
    AD_write("u2", u2)
    updateState2()
    IncTime(t2, h2)
    waitUntil(t2)
}
```

3/

```
currentTime(t)
ad = Td/(Td + N·h)
bd = K·Td·N/(Td + N·h)
I = 0
while (1)
{
    y = AD_read("y")
    r = AD_read("r")
    e = r - y
    D = ad · D - bd * (y - yold)
    v = K * e + I + D
    if (mode = auto)
        u = sat(v, umax, umin)
    else
        u = sat(uman, umax, umin)
    AD_write("u", u)
    I = I + (K·h/Ti)·e + (h/Tt)·(u-v)
    if inc
        uinc = 1
    elseif dec
        uinc = -1
    else
        uinc = 0
    uman = uman + (h/Tm)·uinc + (h/Tt)·(u-uman)
    yold = y
    incTime(t,h)
    waitUntil(t)
}
```

4/ Circular wait are not possible.

C is always allocated before A, which in turn is allocated before B.

5/ Given:

a/

$$\begin{cases} \frac{d\hat{x}}{dt} = (A-BL-KC)\hat{x} + Ky + BL_r \cdot r \\ u = -L\hat{x} + L_r \cdot r \end{cases}$$

Approximate using backward difference $\left(\frac{dx(t)}{dt} \approx \frac{x(t)-x(t-h)}{h}\right)$

$$\frac{\hat{x}(t) - \hat{x}(t-h)}{h} = (A-BL-KC)\hat{x}(t) + Ky(t) + BL_r \cdot r(t)$$

$\Rightarrow$

$$(I-h(A-BL-KC))\hat{x}(t) = \hat{x}(t-h) + hKy(t) + hBL_r \cdot r(t)$$

Assuming scalars, introduce $D = I - h(A-BL-KC)$

$$\hat{x}(t) = \frac{1}{D}\hat{x}(t-h) + \frac{hBL_r}{D} r(t) + \frac{hK}{D} y(t)$$

6) a) $u = \frac{8}{30} + \frac{8}{40} + \frac{20}{60} = 0.8$

b) Priorities according to deadline monotonic:

B highest priority

A medium priority

C lowest priority

$hp(A) = \{B\}$

$hp(B) = \{\}$

$hp(C) = \{A, B\}$

Standard analysis can now be used to analyze behavior.

Iteration 0

$R_A^0 = 8 + 8 = 16$

$R_B^0 = 8$ (largest responsetime for B)

$R_C^0 = 20 + 8 + 8 = 36$

Iteration 1

$R_A^1 = 8 + \left\lceil \frac{16}{40} \right\rceil \cdot 8 = 16$ (largest response time for A)

$R_B^1 = 8$

$R_C^1 = 20 + \left\lceil \frac{36}{30} \right\rceil \cdot 8 + \left\lceil \frac{36}{40} \right\rceil \cdot 8 = 44$

Iteration 2

$R_C^2 = 20 + \left\lceil \frac{44}{30} \right\rceil \cdot 8 + \left\lceil \frac{44}{40} \right\rceil \cdot 8 = 52$

Iteration 3

$R_C^3 = 20 + \left\lceil \frac{52}{30} \right\rceil \cdot 8 + \left\lceil \frac{52}{40} \right\rceil \cdot 8 = 52$ (largest response time for C)

$R_A < D_A$, $R_B < D_B$, $R_C < D_C$ $\Rightarrow$ All deadlines can be met!

c) The phase shift is $-\omega_c \cdot D \cdot \frac{180}{\pi} = -3 \cdot 55 \cdot 10^{-3} \cdot \frac{180}{\pi} \approx 10°$

7/

a/

	SOF ↓	
bit on bus	0	1011 0101 0010
Node A	0	1011 0101 0010
B	0	11 ← stop sending
C	0	10111 ← stop sending

b/ Bit stuffing is used for clock synchronization. After five bits with the same value a bit of the opposite value is inserted.

c/ One Ethernet frame has to be sent to each sensor and actuator. Slow and takes a lot of bandwidth. Multiple sensors and actuators are sent using the same frame in EtherCAT. No collisions are possible since on each cable there is a single sender and a single receiver. See lecture notes.