LUNDS TEKNISKA
HÖGSKOLA
Lunds universitet

Institutionen för
REGLERTEKNIK

Real-Time Systems

Exam December 12, 2011, Hours: 08:00-13:00

Points and grades

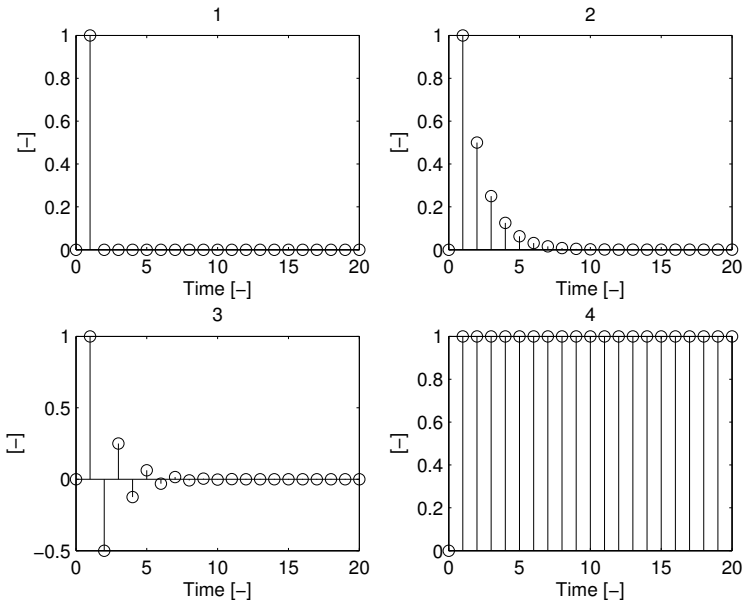
All answers must include a clear motivation and a well-formulated answer. Answers may be given in **English or Swedish**. The total number of points is 25. The maximum number of points is specified for each subproblem.

Accepted aid

Standard mathematical tables and authorized "Real-Time Systems Formula Sheet". Pocket calculator.

Results

The result of the exam will be posted on the notice-board at the Department. The result as well as solutions will be available on WWW:
http://www.control.lth.se/Education/EngineeringProgram/FRTN01.html



Figur 1 Pulse response for Problem 3.

1. What is priority inheritance? Explain the problem that it solves. (1.5 p)

2. Draw the block diagram for a PID controller on parallel form with tracking-based anti-windup, an internal actuator model, and bumpless switching between automatic and manual mode. (2 p)

3. Four pulse responses are shown in Figure 1. Match the pulse responses and the following four transfer functions:

$$G_1 = \frac{1}{z}$$
$$G_2 = \frac{1}{z-1}$$
$$G_3 = \frac{1}{z+0.5}$$
$$G_4 = \frac{1}{z-0.5}$$

(2 p)

4. A continuous-time PD controller is given by (1)

$$U(s) = K(1 + T_d s)E(s) \tag{1}$$

a. Discretize (1) using Forward difference (Euler's method) (0.5 p)

b. Discretize (1) using Tustin's approximation (0.5 p)

c. Is there any problem with either of the two discretizations? (1 p)

5. Consider the continuous time system

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

a. Sample the system using the sampling time h and find the discrete time state space description. (1 p)

b. What do the Φ and Γ matrices look like for very small h. Explain why this may cause numerical problems if a very short sampling time is used. (2 p)

6. Consider the following system:

$$\frac{dx}{dt} = -x(t) + 2u(t-\tau)$$
$$y(t) = x(t)$$

a. ZOH-sample the system with the sampling interval $h = 0.5$ for $\tau = 0.4$. (1 p)

b. Write the system on (augmented) state-space form. (1 p)

c. Design a state feedback controller on the form $u(k) = l_r r(k) - Lx(k)$ for the augmented system in the previous subproblem. Place the pole resulting from the time delay in the origin and the other pole in the discrete-time counterpart of a continuous-time pole in -2. Ensure that the steady state gain from the reference signal, $r(k)$, to the output is 1. (2 p)

d. Write pseudo-code for the controller by extending the following code skeleton. To get full points you must write the code so that the input-output latency is minimized (1.5 p)

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

// CalculateOutput code

.....

setOutput(u);

// UpdateState code

......

// Sleep code

}
```

7.

a. Consider the following task set under fixed priority scheduling where the period and the relative deadline (equal to the period) for the high-priority task A is unknown. The tasks do not suspend themselves, the kernel is ideal, and there is no inter-process communication.

Task name	T_i	D_i	C_i	Priority
A	x	x	3	High
B	10	10	2	Low

What is the smallest possible period of Task A in order for the task set to be schedulable? (1 p)

b. Consider the following task set under fixed priority scheduling where the worst-case execution time for the high-priority task A is unknown. The tasks do not suspend themselves, the kernel is ideal, and there is no inter-process communication.

Task name	T_i	D_i	C_i	Priority
A	5	5	x	High
B	10	10	2	Low

What is the largest possible value for the worst-case execution time for task A in order for the task set to be schedulable? (1 p)

c. For controller tasks it can be useful to have an upper bound on the sampling latency. This can be achieved if one knows the latest possible start time, S_i, for each controller task, i, relative to the task release time, i.e., the largest delay caused by interference from higher-priority tasks before the task really starts to execute. Calculate this value for all of the three tasks in the task set below.

Task name	T_i	D_i	C_i	Priority
A	1	1	0.2	High
B	4	4	1.2	Medium
C	5	5	1.5	Low

In order to get full point you should also derive a general formulae for how to calculate S_i for an arbitrary task set. (Hint: Use response-time analysis arguments for a task that is infinitely short)

(2 p)

8. Consider the following implementation of a bounded buffer using Modula 2 and STORK.

```
TYPE CriticalSectionMonitor = RECORD
       mon        : Monitor;
       nonFull    : Event;
       nonEmpty   : Event;
```

```
      databuffer : buffer;
    END;

VAR R: CriticalSectionMonitor;
```

The pseudo-code for the producer and the consumer processes is shown below.

```
  Producer Process            Consumer Process

...                          ...
WITH R DO                    WITH R DO
  Enter(mon);                 Enter(mon);
  WHILE  "buffer full" DO     WHILE "buffer empty": DO
    Await(nonFull);              Await(nonEmpty)
  END;                        END;
  enter data into buffer;     get data from buffer;
  Cause(nonEmpty);            Cause(nonFull);
  Leave(mon);                 Leave(mon);
END;                          END;
...                           ...
```

Assume that we have 3 periodic Producer processes and 4 periodic Consumer processes that all use the same buffer `R`. Assume further that the buffer only may contain 2 data elements. The periodic processes use the `WaitTime` primitive to implement periodic execution.

a. Which are the process queues involved in the execution of the above processes under STORK? (1 p)

b. What are the maximum number of processes that each queue may contain in some execution scenario? Describe each of execution scenarios. (2 p)

9. Complete the function skeleton below. The function takes two values X and Y represented with n fractional bits in `int16_t` variables. The function should return the product of X and Y using the same data type and the same number of fractional bits as the inputs. In case of overflow, the value should be saturated. (2 p)

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

  // Insert your fixed-point code here

  return (int16_t)Z;
}
```