EL2450 Hybrid and Embedded Systems
Exercise Notes

Sofie Ahlberg

February 6, 2020

This pdf contains my notes (as TA) for exercise 5-8.
There may be errors!

Exercise 5 - Implementation Aspects (Delays, Jitter, Quantization)

Problems considered in this exercise: 5.1, 5.2, 5.3

Brief Theory

- Delay - Process time and/or transmission time leads to the system not excuting everything instantly.
- Jitter - Jitter is the maximum deviation of the time delay ($\delta\tau = \tau_{max} - \tau_{min}$).
- Packet loss - Due to errors or overload in the network some packet of data doesn't reach the receiver.
- Quantization - Discretization of a continous signal to specific levels, e.g. rounding up to closest level.

This will effect the system (stability and performance).

Problems and Solutions

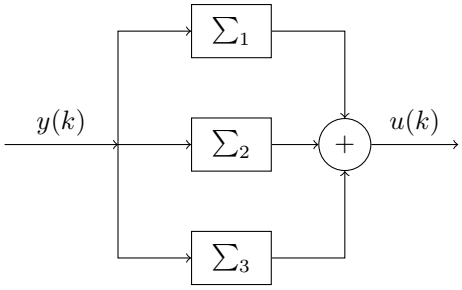
5.1)

$$H(z) = \frac{1}{(z-1)(z-0.5)(z^2+0.5z+0.25)} \tag{1}$$

Consider the discrete-time controller characterized by the pulse-transfer function $H(z)$. Implement the controller in paralllel form.

Solution: **What is parallel form?**

Parallel form: $u(k) = \sum_1 y(k) + \sum_2 y(k) + \sum_3 y(k)$ i.e.



We need to partition $H(z)$, what should the new denominators be? Why? How do we pick the corresponding numertors?

To rewrite $H(z)$ on parallel form we need to partition it such that

$$H(z) = \frac{A}{z-1} + \frac{B}{z-0.5} + \frac{Cz+D}{z^2+0.5z+0.25} \tag{2}$$

Why these denominators? Why not divide the third into two first order systems?
→ Check roots of the second order polynom (i.e. poles)!

$$z^2 + 0.5z + 0.25 = 0 \rightarrow z = -0.25 \pm \sqrt{0.0625 - 0.25} = -0.25 \pm i0.433 \quad (3)$$

The second order polynom has complex poles, and hence can't be divided into two real one order polynoms!

How do we determine A, B, C and D?

Finding A, B, C and D:

$$H(z) = \frac{A}{z-1} + \frac{B}{z-0.5} + \frac{Cz+D}{z^2+0.5z+0.25} = \frac{1}{(z-1)(z-0.5)(z^2+0.5z+0.25)} \quad (4)$$

$$\rightarrow A(z-0.5)(z^2+0.5z+0.25) + B(z-1)(z^2+0.5z+0.25) + (Cz+D)(z-1)(z-0.5) = 1 \quad (5)$$

Rearranging we get: $(A+B+C)z^3 + (-0.5B - 1.5C + D)z^2 + (-0.25B + 0.5C - 1.5D)z + (-0.125A - 0.25B + 0.5D) = 1$ or

$$A + B + C = 0 \quad (6)$$
$$-0.5B - 1.5C + D = 0 \quad (7)$$
$$-0.25B + 0.5C - 1.5D = 0 \quad (8)$$
$$-0.125A - 0.25B + 0.5D = 1 \quad (9)$$

How can we solve the equation system? There are many options, make sure you know one!

The equation system can be solved by Gaussian elimination:

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & -0.5 & -1.5 & 1 \\ 0 & -0.25 & 0.5 & -1.5 \\ -0.125 & -0.25 & 0 & 0.5 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 3 & -2 \\ 0 & 1 & -2 & 6 \\ 1 & 2 & 0 & -4 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -8 \end{bmatrix} \begin{matrix} r_1 \\ r_2 \cdot -2 \\ r_3 \cdot -4 \\ r_4 \cdot -8 \end{matrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & -5 & 8 \\ 0 & 1 & -1 & -4 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -8 \end{bmatrix} \begin{matrix} r_1 \\ r_2 \\ r_3 - r_2 \\ r_4 - r_1 \end{matrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & 1 & -8/5 \\ 0 & 0 & -4 & -2 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -8 \end{bmatrix} \begin{matrix} r_1 \\ r_2 \\ r_3 \cdot -1/5 \\ r_4 - r_2 \end{matrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & 1 & -8/5 \\ 0 & 0 & 1 & 0.5 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 2 \end{bmatrix} \begin{matrix} r_1 \\ r_2 \\ r_3 \\ r_4 \cdot -1/4 \end{matrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & 1 & -8/5 \\ 0 & 0 & 0 & 2.1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 2 \end{bmatrix} \begin{matrix} r_1 \\ r_2 \\ r_3 \\ r_4 - r_3 \end{matrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & 1 & -8/5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 2/2.1 \end{bmatrix} \begin{matrix} r_1 \\ r_2 \\ r_3 \\ r_4 \cdot 1/2.1 \end{matrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 3 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} 0 \\ 4/2.1 \\ 3.2/2.1 \\ 2/2.1 \end{bmatrix} \begin{matrix} r_1 \\ r_2 + 2r_4 \\ r_3 + 8/5 r_4 \\ r_4 \end{matrix}$$

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} -3.2/2.1 \\ -5.6/2.1 \\ 3.2/2.1 \\ 2/2.1 \end{bmatrix} \begin{matrix} r_1 - r_3 \\ r_2 - 3r_3 \\ r_3 \\ r_4 \end{matrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} 2.4/2.1 \\ -15.2/2.1 \\ 3.2/2.1 \\ 2/2.1 \end{bmatrix} \begin{matrix} r_1 - r_2 \\ r_2 \\ r_3 \\ r_4 \end{matrix}$$

$A = 2.4/2.1 \approx 1.14$, $B = -5.6/2.1 \approx -2.67$, $C = 3.2/2.1 \approx 1.52$, $D = 2/21 \approx$

0.95. The same result can be found by solving $X\backslash y$ in Matlab, where $X = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & -0.5 & -1.5 & 1 \\ 0 & -0.25 & 0.5 & -1.5 \\ -0.125 & -0.25 & 0 & 0.5 \end{bmatrix}$ and $y^T = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$.

To implement the controller on paralllel form we now have to rewrite the resulting $H(z)$ as 3 discrete-time controllers. The first two controllers are first order and hence

$$x_1(k+1) = a_i x_1(k) + y(k) \tag{10}$$
$$u_i(k) = b_i x_1(k) \tag{11}$$

where $u(z)_i = \frac{b_i}{z-a_i} y(z)$ i.e.

$$x_1(k+1) = -x_1(k) + y(k) \tag{12}$$
$$u_1(k) = 1.14 x_1(k) \tag{13}$$

and

$$x_1(k+1) = -0.5 x_1(k) + y(k) \tag{14}$$
$$u_2(k) = -2.67 x_1(k) \tag{15}$$

The third controller is second order:

$$x_1(k+1) = x_2(k) \tag{16}$$
$$x_2(k+1) = -b_i x_1(k) - a_i x_2(k) + y(k) \tag{17}$$
$$u_3(k) = c_i x_1(k) + d_i x_2(k) \tag{18}$$

where $u_i(z) = \frac{c_i + d_i z}{z^2 + a_i z + b_i} y(z)$, and hence in our case:

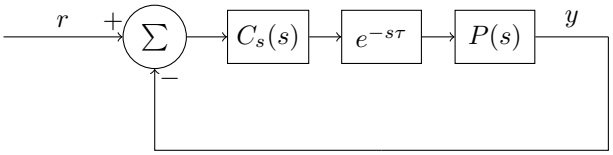
$$x_1(k+1) = x_2(k) \tag{19}$$
$$x_2(k+1) = -0.25 x_1(k) - 0.5 x_2(k) + y(k) \tag{20}$$
$$u_3(k) = 1.52 x_1(k) + 0.95 x_2(k) \tag{21}$$

Why can we translate from $u(z) = g(z)y(z)$ to a discrete-time system like this?
$\rightarrow$ Comparable to translation from transfer function to state space representation in control courses.
First order: $u(z) = \frac{b_i}{z-a_i} y(z)$ use $u = Bx \rightarrow (z - a_i)Bx = By \rightarrow x(k+1) - a_i x(k) = y(k) \rightarrow x(k+1) = a_i x(k) + y(k)$.
Second order: $u(z) = \frac{c_i + d_i z}{z^2 + a_i z + b_i} y(z)$, use $u = c_i x_1 + d_i x_2$ where $x_2(k) = x_1(k+1)$ $\rightarrow (z^2 + a_i z + b_i)(c_i + d_i z)x_1 = (c_i + d_i z)y \rightarrow x_1(k+2) + a_i x_1(k+1) + b_i x_1(k) = y(k) \rightarrow x_2(k+1) + a_i x_2(k) + b_i x_1(k) = y(k) \rightarrow x_2(k+1) = -b_i x_1(k) - a_i x_2(k) + y(k)$.

We then have $u(k) = u_1(k) + u_2(k) + u_3(k)$.

Note! As for continuous transfer functions to state space, there are multiple ways to express the same discrete transfer function on state space form! This is one of them, there are several correct answers and depending on your method you might get a different result.

5.2)



a) Find $C_s(s)$ s.t. the closed-loop transfer function from r to y becomes H_{cl}

$$H_{cl}(s) = \frac{C(s)P(s)}{1 + C(s)P(s)} e^{-s\tau} \tag{22}$$

b) Let $P(s) = \frac{1}{s+1}$ and $H_{cl}(s) = \frac{8}{s^2+4s+8} e^{-s\tau}$. Find the expression for the Smith predictor $C_s(s)$!

Solution:

We want to find another expression for H_{cl} dependent on C_s and compare to the first. How do we find the transfer function of $H_{cl}(C_s)$?

The closed-loop transfer function is defined s.t. $Y(s) = H_{cl}(s)R(s)$. From the figure we get:

$$Y(s) = P(s)e^{-s\tau}C_s(s)(R(s) - Y(s)) \tag{23}$$

$$\rightarrow Y(s) = \frac{P(s)C_s(s)}{1 + P(s)C_s(s)e^{-s\tau}} e^{-s\tau} R(s) \tag{24}$$

To achieve the desired closed-loop transfer function we then need:

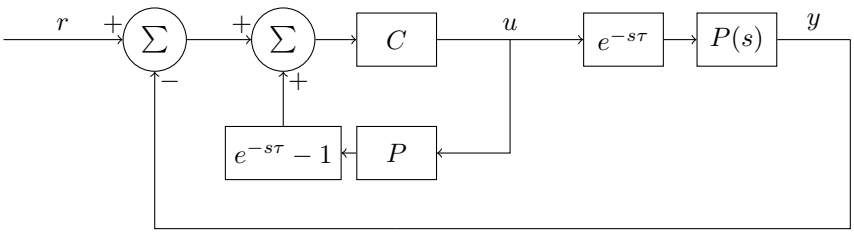
$$\frac{P(s)C_s(s)}{1 + P(s)C_s(s)e^{-s\tau}} e^{-s\tau} = \frac{C(s)P(s)}{1 + C(s)P(s)} e^{-s\tau} \tag{25}$$

$$\rightarrow C_s(s)(1 + C(s)P(s)) = C(s)(1 + P(s)C_s(s)e^{-s\tau}) \tag{26}$$

$$\rightarrow C_s(s) = \frac{C(s)}{1 + C(s)P(s)(1 - e^{-s\tau})} \tag{27}$$

Is this a reasonable answer?

A Smith predictor for known time delays is illustrated by the block diagram below. Here we can note that the inner loop is $C_s(s)$. Hence we get directly $C_s(s) = \frac{C(s)}{1+C(s)P(s)(1-e^{-s\tau})}$. We then conclude that our calculated answer is the expected one!



What do we need to determine C_s? How can we do it?

We have an expression for C_s given P and C. To find the desired expression we hence have to find C first. From the given transfer function of H_{cl} we have:

$$H_{cl}(1+CP) = CPe^{-s\tau} \tag{28}$$

$$\rightarrow C = \frac{H_{cl}}{P(e^{-s\tau} - H_{cl})} = ... = \frac{8(s+1)}{s(s+4)} \tag{29}$$

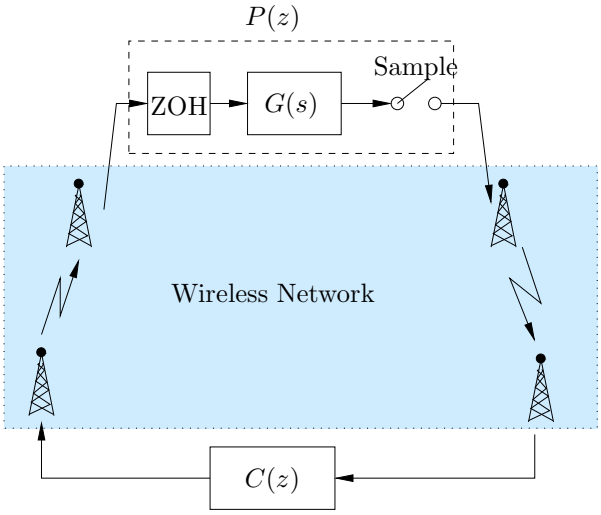
where the transfers functions P and H_{cl} has been inserted in the last equality. Next we can find C_s from the formula calculated in (a).

$$C_s(s) = \frac{C(s)}{1 + C(s)P(s)(1-e^{-s\tau})} = ... = \frac{8(s+1)}{s(s+4) + 8(1-e^{-s\tau})} \tag{30}$$

An alternative is to find a formula for C_s dependent on P and H_{cl}. From $H_{cl}(s) = \frac{C(s)P(s)}{1+C(s)P(s)}e^{-s\tau}$ we get $C_s = \frac{H_{cl}}{Pe^{-s\tau}(1-H_{cl})}$, and the answer is found by inserting the expressions of P and H_{cl}.

5.3)

$P(z) = \frac{z}{z-0.5}$ and $C(z) = K_p + K_i\frac{z}{z-1}$, $K_p = 0.2$, $K_i = 0.1$. Due to retrans-



missions of dropped packets, the network introduces time-varying delays. How large can the maximum delay be so that the closed-loop system is stable?

Solution:

What problems are we trying to handle here? Packet loss or delay? Delay! The packet loss is already handled by retransmissions. This approach is the reason for the delays.

What conditions do we know for time delays corresponding to stability?

$$\tau < \phi_m/\omega_c \qquad \text{For CTS with fixed delays} \tag{31}$$

$$|H_{cl}(i\omega)| < 1/\tau_{max}\omega,\ \forall\omega \in [0,\infty) \qquad \text{For CTS with time-varying delays} \tag{32}$$

The second condition is sufficient if the non-delayed system is stable.

Can we use one of them? Not directly! The second is close since we have a time-varying system, but we have a DTS!

Can we find a condition for a DTS using our knowledge of the CTS bounds?

The closed-loop system is stable if

$$\left|\frac{P(e^{i\omega})C(e^{i\omega})}{1+P(e^{i\omega})C(e^{i\omega})}\right| < \frac{1}{N|e^{i\omega}-1|} \tag{33}$$

for a discrete-time system with time-varying delay, if the non-delayed system is stable!

From lectures and reading material we have the condition $|H_{cl}(i\omega)| < \frac{1}{\tau_{max}\omega}$ for continuous-time systems with time-varying delays. To find the condition for discrete-time we need to go back to the small gain theorem (which was used to find the continous-time condition):

$$\gamma(S_i) = \sup_{e_i \in L_2} \frac{\|S_i(e_i)\|_2}{|e_i\|_2} \quad \text{stability if } \gamma(S_1)\gamma(S_2) < 1 \tag{34}$$

For the continous case we had $S_1(s) = \frac{sPC}{1+PC}$ which can be translated to $S_1(z) = \frac{z-1}{z}\frac{PC}{1+PC}$, and $S_2(s) = \frac{\Delta-1}{s}$ which can be translated into $S_2(z) = \frac{(\Delta-1)z}{z-1}$.
For $S_1(z)$ we get $\gamma(S_1) = \left|\frac{e^{i\omega}-1}{e^{i\omega}}\frac{P(e^{i\omega})C(e^{i\omega})}{1+P(e^{i\omega})C(e^{i\omega})}\right| = |(1-e^{-i\omega})H_{cl}(e^{i\omega})|$. For $S_2(z)$ it can be proven (paper 3 in the PhD thesis [L] from the reading material) that $\gamma(S_2) \leq N$. It then follows that the stability criterion of the small gain theorem is fullfilled if

$$|N(1-e^{-i\omega})H_{cl}(e^{i\omega})| < 1 \tag{35}$$

or

$$|H_{cl}(e^{i\omega}))| < \frac{1}{N|1-e^{-i\omega}|} \tag{36}$$

We need to check if the non-delayed system is stable and the condition. How do we check stability?

Let's check the poles! The non-delayed closed-loop system is given by:

$$H_{cl}(z) = \frac{P(z)C(z)}{1+P(z)C(z)} = \frac{z(K_p(z-1)+K_iz)}{(z-1)(z-0.5)+z(K_p(z-1)+K_iz)} \tag{37}$$

and hence the poles are: $z = 0.65 \pm 0.206$, i.e. $z_1 = 0.86$, $z_2 = 0.44$. It is clear that z_i are within the unit circle and hence the system is stable.

How can we check the condition?

We now know that the condition is valid. It is however difficult to perform explicit calculations to determine if it is satisfied. To check the condition we therefore plot $|H_{cl}(e^{i\omega})|$ and $\frac{1}{N|1-e^{-i\omega}|}$ for different values of N to check when the condition holds. The resulting plot is found below. From the plot we make the conlusion that the condition is satisfied for $N \leq 3$. That is, the maximum delay allowed to keep stability is N steps. (OBS! There is an error in the solution manual for this task. The error occur only in the figure and does not effect the answer. The error consists of them plotting the function $H_{cl}(e^{i\omega})$ with a mistake in the nominator, adding an extra 1. That is they plot $\frac{z(K_p(z-1)+K_i z)+1}{(z-1)(z-0.5)+z(K_p(z-1)+K_i z)}$)

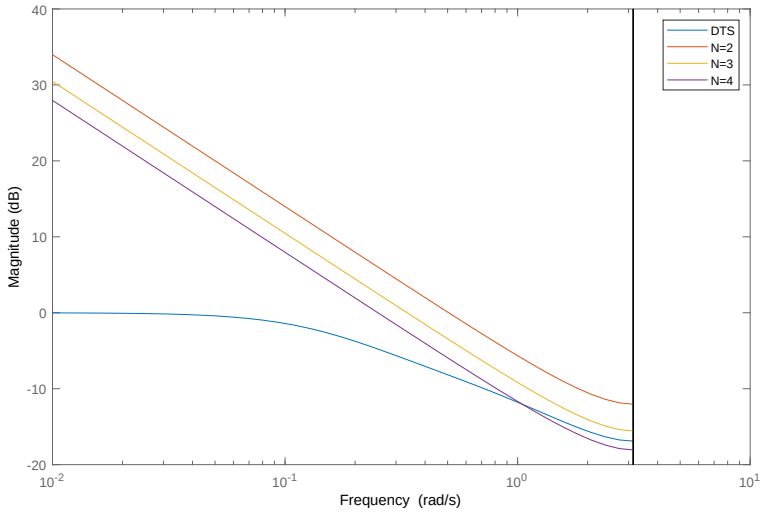


Figure 1: $|H_{cl}|$, and $\frac{1}{N|1-e^{-i\omega}|}$ for $N \in [2, 3, 4]$

Exercise 6 - Eventbased Control and Real-Time Systems

Problems considered in this exercise: 6.1, 6.7, 6.8

Brief Theory

- Lyapunov theory - can be used to show if a system converges to the origin
- Sampling (periodic and aperiodic) - measuring the signals of the system at specific times instead of continuously. Periodic means that we measure at fixed times and aperiodic means that we measure at varying times.
- Event-triggered control - The control input change dependent on events instead of time, e.g. if a distance is too far.

Problems and Solutions

6.1

Consider the following first-order system:

$$\dot{x} = x + u$$

where $x, u \in \mathbb{R}$ are the state and control input, respectively. Moreover, $x = 5$ when $t = 0$. The feedback control law is given by:

$$u = -2\,x$$

(a) Show that under this control law, the closed-loop system asymptotically converges to the origin, using a Lyapunov function.

(b) Suppose $x(t)$ is sampled periodically and the controller is followed by a Zero Order Hold, namely

$$u(t) = -2\,x(kh), \qquad t \in [kh, kh + h),$$

where $k \in \mathbb{N}$, $h > 0$ is the sampling period. What is the maximal value $h_{\max}$ of h before the closed-loop system becomes unstable?

(c) Now $x(t)$ is sampled aperiodically at the sequence $\{t_k\}$, $k \in \mathbb{N}$ and the controller is followed by a Zero Order Hold, namely

$$u(t) = -2\,x(t_k),\ t \in [t_k, t_{k+1}).$$

how to design the sequence $\{t_k\}$, $k \in \mathbb{N}$ such that the closed-loop system still converges to the origin? what is the maximal sampling interval $\max_k(t_{k+1} - t_k)$ in this case?

(d) Describe how the event-triggered controller could be implemented on a digital platform and compare it with periodic controller in (b).

Solution:

(a) **What do we need to check?**

Lyapunov theory tells us that x asymptotically converges to the origin if $\dot{x} = f(x)$, and there exists a function $V : \mathcal{R}^n \to \mathcal{R}$ such that

1. $V(0) = 0$
2. $V(x) > 0$ for $x \neq 0$
3. $\dot{V}(x) = \frac{\partial V}{\partial x}\dot{x} < 0$ for $x \neq 0$

With the given controller we have

$$\dot{x} = x + u = x - 2x = -x$$

We need to find a V that satisfies the conditions above! We test $V = x^2$.

1. $V(0) = 0^2 = 0$ - ok!
2. $V(x) = x^2 > 0$ for $x \neq 0$ - ok!
3. $\dot{V}(x) = 2x\dot{x} = -2x^2 < 0$ for $x \neq 0$ - ok!

It then follows from Lyapunov theory that the closed-loop system asymptotically converts to the origin!

(b) **How does the closed loop system change? Is there a formula for the sampled data system?**

For a system $\dot{x} = Ax + Bu$ with periodic sampling $h > 0$ the sampled data system is:

$$x(kh + h) = e^{Ah}x(kh) + \int_0^h e^{As}dsBu(kh) \tag{38}$$

In our case this corresponds to $A = B = 1$ and hence we have

$$x(kh + h) = e^h x(kh) + \int_0^h e^s dsu(kh) = e^h x(kh) + (e^h - 1)u(kh) \tag{39}$$

With $u(kh) = -2x(kh)$ we then get

$$x(kh + h) = e^h x(kh) - (e^h - 1)2x(kh) = (2 - e^h)x(kh) \tag{40}$$

Can we find some pattern from this?

We then have for $k = 0, 1, ...$

$$x(h) = (2 - e^h)x(0) \tag{41}$$
$$x(2h) = (2 - e^h)x(h) = (2 - e^h)^2 x(0) \tag{42}$$
$$\vdots \tag{43}$$
$$x(kh) = (2 - e^h)^k x(0) \tag{44}$$

How can we use this to guarantee stability?

To avoid instability, $x(kh) \to \infty$, we need $|2 - e^h| < 1$. That is,

$$-1 < 2 - e^h < 1 \tag{45}$$
$$1 < e^h < 3 \tag{46}$$
$$0 < h < \ln 3 \tag{47}$$

and hence, $h_{max} = \ln 3$

(c) **How does the closed-loop system change now?**

We introduce $e(t) = x(t_k) - x(t)$ for $t \in [t_k, t_{k+1})$, as the error. It follows that

$$u(t) = -2(x(t) + e(t)),\ t \in [t_k, t_{k+1}) \tag{48}$$

We then get

$$\dot{x} = x(t) + u(t) = x(t) - 2x(t) - 2e(t) = -x(t) - 2e(t) \tag{49}$$

How can we determine if we have convergance?

To study convergance we use the same technique as in (a) - Lyapunov. We try with the same choice of $V = x^2$. We already know that the first 2 criteria hold so we go directly for the third.

$$\dot{V} = 2x(-x - 2e) = -2x^2 - 4xe = -2|x|^2 - 4xe \tag{50}$$

Using the inequality $-ab \leq |a| \cdot |b|$ we get $-xe \leq |x| \cdot |e|$ and hence

$$\dot{V} = -2|x|^2 - 4xe \leq -2|x|^2 + 4|x| \cdot |e| = -2|x|(|x| - 2|e|) \tag{51}$$

It follows that we have asymptotic conversion if $|x| - 2|e| > 0$ or $|e| < \frac{|x|}{2}$.

How can we express the sampled system? Formula?

For a aperiodic sampling on $\dot{x} = Ax + Bu$ we have:

$$x(t) = e^{A(t-t_k)}x(t_k) + \int_{t_k}^{t} e^{A(t-s)}Bu(s)ds \tag{52}$$

It then holds for $t \in [t_k, t_{k+1})$:

$$\begin{aligned}
x(t) &= e^{(t-t_k)}x(t_k) + \int_{t_k}^{t} e^{(t-s)}u(s)ds = && (53)\\
&= e^{(t-t_k)}x(t_k) + \int_{t_k}^{t} e^{(t-s)}dsu(t_k) = && (54)\\
&= e^{(t-t_k)}x(t_k) - (e^{t-t} - e^{t-t_k})(-2x(t_k)) = && (55)\\
&= e^{(t-t_k)}x(t_k) - (1 - e^{t-t_k})(-2x(t_k)) = && (56)\\
&= e^{(t-t_k)}x(t_k) + 2(1 - e^{t-t_k})x(t_k) = && (57)\\
&= (2 - e^{(t-t_k)})x(t_k) && (58)
\end{aligned}$$

which yields the error

$$e(t) = x(t_k) - x(t) = \left(\frac{1}{2 - e^{t-t_k}} - 1\right)x(t) = \left(\frac{e^{t-t_k} - 1}{2 - e^{t-t_k}}\right)x(t) \tag{59}$$

What requirement can we find to have convergence then?

To satisfy the condition from earlier and get convergence we need $|e| < \frac{|x|}{2}$ i.e.

$$\frac{|e|}{|x|} = \left|\frac{e^{t-t_k}-1}{2-e^{t-t_k}}\right| < \frac{1}{2} \tag{60}$$

or

$$e^{t-t_k} - 2 < 2(e^{t-t_k}-1) < 2 - e^{t-t_k} \tag{61}$$

$$0 < e^{t-t_k} < 4/3 \tag{62}$$

$$t - t_k < \ln 4/3 \tag{63}$$

for this to hold for $t \in [t_k, t_{k+1})$ we need $t_{k+1} - t_k < \ln 4/3$ which is the maximum sampling interval.

(d) Both the event-based and the periodic controller are implemented using the same 4 steps, but the steps are implemented differently.

What are the steps? Can you describe the steps for periodic? Describe it for event-triggered?

	Periodic	Event-based
1. Monitor	Monitor the *time* t and check if $t = kh$ If it is, go to 2!	Monitor the *state* $x(t)$ and check if $\|e\| < 0.5\|x\|$. If not go to 2!
2. Sample	Sample the $x(t)$ to obtain the $x(kh)$, go to 3!	Sample $x(t)$ to obtain $x(t_k)$, go to 3!
3. Update	Update $u(t) = -2x(kh)$ with the new value from 2, go to 4!	Update $u(t) = -2x(t_k)$ with the new value from 2, go to 4!
4. ZOH	Apply ZOH on u, go back to 1!	Apply ZOH on u, go back to 1!

How do the two control implementation differs?

Note that the difference occur mainly in step 1 where we monitor the state instead of the time.

6.7

Consider the following linear second-order system

$$(S): \begin{cases} \dot{x}_1(t) = 3x_1(t) + x_2(t) + u(t) \\ \dot{x}_2(t) = 5x_1(t) - 2x_2(t) + u(t) \end{cases}$$

with $[x_1, x_2]^\top = x \in \mathbb{R}^2, u \in \mathbb{R}, t \geq 0$ and initial conditions $x_1(0) = x_2(0) = 1$.

(a) Show that the system is unstable.

(b) Determine a linear state-feedback controller $u(t) = Kx(t)$ with $K = [K_1\ K_2]$, $K_1, K_2 \in \mathbb{R}$ such that the poles of the closed loop system are placed in -2 and -4.

(c) In order to implement the controller on a digital platform, the state of the system is sampled aperiodically at a sequence of time instants $\{t_k\}, k \in \mathbb{N}$, and the control signal is now given by

$$u(t) = Kx(t_k), t \in [t_k, t_{k+1}).$$

Find the closed loop equation of the system in terms of the state $x(t)$ and the state error $e(t)$, where

$$e(t) = x(t_k) - x(t), t \in [t_k, t_{k+1}).$$

(d) By using the positive definite quadratic Lyapunov function

$$V(x) = \frac{1}{2}\, x^\top \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} x$$

find a relation between the error $e(t)$ and the state $x(t)$ such that the system is still asymptotically stable.

Solution:

(a) **How do we check stability?**

To check stability we can write the system on matrix form and check the eigenvalues of A (which corresponds to the poles).

$$\dot{x} = Ax + Bu = \begin{pmatrix} 3 & 1 \\ 5 & -2 \end{pmatrix} x + \begin{pmatrix} 1 \\ 1 \end{pmatrix} u \tag{64}$$

The poles are then found by solving $\det(\lambda I - A) = 0$. This yields $\lambda^2 - \lambda - 11 = 0$ or poles in $\lambda = 0.5 \pm 3.35$. It is clear that one of the poles is a positive real number, i.e. in the RHP, and hence the system is unstable!

(b) **What is the closed-loop system? How do we get the closed-loop poles?**

With the given controller the closed-loop system becomes

$$\dot{x} = (A + BK)x \tag{65}$$

The closed-loop poles are then given as the eigenvalues of $A + BK$. Using the same method as in (a), we get

$$\lambda^2 - (1 + K_1 + K_2)\lambda - 11 - 3K_1 - 2K_2 \tag{66}$$

How can we place the poles where we want? How should the equation look like?

To place the poles in -2 and -4 we need

$$0 = (\lambda + 2)(\lambda + 4) = \lambda^2 + 6\lambda + 8 \tag{67}$$

Comparing the two characteristic equations (the one we have and the one we want) we get

$$6 = -(1 + K_1 + K_2) \quad 8 = -(11 + 3K_1 + 2K_2) \tag{68}$$

or $K_1 = -5$ and $K_2 = -2$.

(c) **How does the closed-loop system change? What information do we have now?**

We then have

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{69}$$
$$u(t) = Kx(t_k) \quad t \in [t_k, t_{k+1}) \tag{70}$$
$$e(t) = x(t_k) - x(t) \quad t \in [t_k, t_{k+1}) \tag{71}$$

We can then rewrite $u(t) = K(e(t) + x(t))$ and the closed-loop system becomes

$$\dot{x}(t) = Ax(t) + BK(e(t) + x(t)) = (A + BK)x(t) + BKe(t) \tag{72}$$

or with the values

$$\dot{x} = \begin{pmatrix} -2 & -1 \\ 0 & -4 \end{pmatrix} x + \begin{pmatrix} -5 & -2 \\ -5 & -2 \end{pmatrix} e \tag{73}$$

(d) **What do we need to check?**

Let's first check if V satisfies the first two criteria!

$$V(0) = 0 \tag{74}$$
$$V(x) > 0 \tag{75}$$

are satisfied! The second hold since $\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$ is positive definite (eigenvalues are positive).

Finally, let's check the last criteria to find the relation! We have

$$V = \frac{1}{2} x^T \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} x = \frac{1}{2}(2x_1^2 + x_2^2)$$

$$\dot{V} = \frac{\partial V}{\partial x_1}\dot{x}_1 + \frac{\partial V}{\partial x_2}\dot{x}_2 = 2x_1\dot{x}_1 + x_2\dot{x}_2 = \tag{76}$$
$$= 2x_1(-2x_1 - x_2 - 5e_1 - 2e_2) + x_2(-4x_2 - 5e_1 - 2e_2) = \tag{77}$$
$$= -4(x_1^2 + x_2^2) - (5e_1 + 2e_2)(2x_1 + x_2) - 2x_1x_2 = \tag{78}$$
$$= -4\|x\|^2 - (5e_1 + 2e_2)(2x_1 + x_2) - 2x_1x_2 \tag{79}$$

Can we use any inequalities to get the realtion between $\|x\|$ and $\|e\|$ from this?

We use the inequalities $-2ab \leq a^2 + b^2$ and $-ab \leq |a| \cdot |b|$ and conclude that

$$\dot{V} \leq -4|x\|^2 + (5|e_1| + 2|e_2|)(2|x_1| + |x_2|) + x_1^2 + x_2^2 \tag{80}$$

Finally, we use that $v_1^2 + v_2^2 = v^2$ and $|v_i| \leq \|v\|$ for a signal $v = \begin{pmatrix} v_1 & v_2 \end{pmatrix}$ and apply it to e_1, e_2, x_1 and x_2.

$$\dot{V} \leq -4\|x\|^2 + 7\|e\|3\|x\| + \|x\|^2 = \|x\|(-3\|x\| + 21\|e\|) \tag{81}$$

From this we see that $\dot{V} \leq 0$ if $-3\|x\| + 21\|e\| < 0$ whcih yields the relation

$$\|e\| < \frac{\|x\|}{7} \tag{82}$$

OBS! Depending on what inequlities we use we can reach different limits. Alternative limits are not wrong but may be more or less conservative. For isntance we could have used only the inequality $-ab \leq |a||b|$ in the first step. Doing so the result is $\|e\| < \frac{\|x\|}{10.5}$, which is more conservative then what we got above.

6.8

Consider three robots R_1, R_2 and R_3 that want to meet at the same place. Each robot is controlled as a simple integrator; i.e., if we denote as $p_i(t) \in \mathbb{R}^2$ the position of robot R_i, then the motion of the robot is described by

$$\dot{p}_i(t) = u_i(t).$$

In order to meet at the same place, R_1 follows R_2, R_2 follows R_3, and R_3 follows R_1, as described by the following equations:

$$\begin{aligned} u_1(t) &= p_2(t) - p_1(t), \\ u_2(t) &= p_3(t) - p_2(t), \\ u_3(t) &= p_1(t) - p_3(t). \end{aligned}$$

Consider the state variables $x_1(t) = p_2(t) - p_1(t)$ and $x_2(t) = p_3(t) - p_2(t)$. Note that the robots reach their goal if and only if $x_1 = x_2 = 0$.

(a) Find the state space representation

$$\dot{x}(t) = Bu(t),$$

where $x(t) = [x_1(t), x_2(t)]^\top$ and $u(t) = [u_1(t), u_2(t), u_3(t)]^\top$.

(b) Find the matrix K such that $u(t) = Kx(t)$.

(c) Write the closed-loop system as

$$\dot{x}(t) = BKx(t).$$

(d) Use the Lyapunov function

$$V(x) = \frac{1}{2}x_1^2 + \frac{1}{2}x_2^2$$

to show that the robots asymptotically meet at the same place.

Now suppose that the robots measure each other's positions only on the aperiodic sampling times t_k, with $k \in \mathbb{N}$. Therefore, the control inputs become

$$\begin{aligned} u_1(t) &= p_2(t_k) - p_1(t_k), \\ u_2(t) &= p_3(t_k) - p_2(t_k), \\ u_3(t) &= p_1(t_k) - p_3(t_k), \end{aligned}$$

for $t \in [t_k, t_{k+1})$.

(e) Let $e(t) = x(t) - x(t_k)$, and write the closed-loop system for $t \in [t_k, t_{k+1})$ as a function of $x(t)$ and $e(t)$.

(f) Using the same Lyapunov function as in (d), to find a condition in the form $\|e(t)\| \leq \alpha \|x(t)\|$, with $\alpha > 0$, that guarantees that the robot asymptotically meet at the same place. Choose α as large as possible.
Hint: for a matrix $M \in \mathbb{R}^{n \times m}$, we have $||M|| = \sqrt{\lambda_{\max}(M^T M)}$, where $\lambda_{\max}(\cdot)$ denotes the largest eigenvalue.

Solution:

(a) **What do we know of how x depends on u? can we adapt it to $\dot{x}$?**

As x_1 and x_2 are defined we have

$$\dot{x}_1 = \dot{p}_2 - \dot{p}_1 = u_2 - u_1 \tag{83}$$
$$\dot{x}_2 = \dot{p}_3 - \dot{p}_2 = u_3 - u_2 \tag{84}$$

We can then write the system as

$$\dot{x} = Bu = \begin{pmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix} u \tag{85}$$

(b) **What do we know of how u depends on x?**

We have

$$u_1 = p_2 - p_1 = x_1 \tag{86}$$
$$u_2 = p_3 - p_2 = x_2 \tag{87}$$
$$u_3 = p_1 - p_3 = -(p_2 - p_1) - (p_3 - p_2) = -x_1 - x_2 \tag{88}$$

We can then write the system as

$$u = Kx = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -1 & -1 \end{pmatrix} x \tag{89}$$

(c) **How do we solve it?**

With B and K from (a) and (b) we get

$$\dot{x} = BKx = \begin{pmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -1 & -1 \end{pmatrix} x = \begin{pmatrix} -1 & 1 \\ -1 & -2 \end{pmatrix} x \tag{90}$$

(d) **What does it mean mathematically that they meet?**

They will meet at the same place if x converges to the origin!

How can we determine that?

Hence we need to show that V satisfies the 3 criteria.

We note directly that $V(0) = 0$ and $V(x) > 0$ for $x \neq 0$. Finally we check:

$$\dot{V} = x_1\dot{x}_1 + x_2\dot{x}_2 = x_1(-x_1 + x_2) + x_2(-x_1 - 2x_2) = \tag{91}$$

$$= -x_1^2 + x_1x_2 - x_1x_2 - 2x_2^2 = -x_1^2 - 2x_2^2 \tag{92}$$

it is then clear that $\dot{V} \leq 0$ and we have convergence!

(e) **How is the closed-loop system change? What is u?**

The new $u(t)$ is then

$$u(t) = Kx(t_k) = K(x(t) - e(t)) \tag{93}$$

and hence we get

$$\dot{x}(t) = BK(x(t) - e(t)), \quad t \in [t_k, t_{k+1}) \tag{94}$$

(f) We have already checked the first two criteria. Let's check the third!

$$\dot{V} = x_1\dot{x}_1 + x_2\dot{x}_2 = x_1(-x_1 + x_2 + e_1 - e_2) + x_2(-x_1 - 2x_2 + e_1 + 2e_2) = \tag{95}$$

$$= -x_1^2 - 2x_2^2 + x_1(e_1 - e_2) + x_2(e_1 + 2e_2) \leq -\|x\|^2 + x^T \begin{pmatrix} 1 & -1 \\ 1 & 2 \end{pmatrix} e \leq \tag{96}$$

$$\leq -\|x\| \left(\|x\| - \left\| \begin{pmatrix} 1 & -1 \\ 1 & 2 \end{pmatrix} \right\| \|e\| \right) \tag{97}$$

Why do the inequalities above hold? What condition do we get for convergence?

Hence $\dot{V} \leq 0$ if $\left(\|x\| - \left\| \begin{pmatrix} 1 & -1 \\ 1 & 2 \end{pmatrix} \right\| \|e\| \right) \geq 0$ which gives the relation

$$\|\|e\| \leq \frac{\|x\|}{\left\| \begin{pmatrix} 1 & -1 \\ 1 & 2 \end{pmatrix} \right\|} \tag{98}$$

We then identify $\alpha = \frac{1}{\left\| \begin{pmatrix} 1 & -1 \\ 1 & 2 \end{pmatrix} \right\|}$.

How can we determine α?

Using the hint we have

$$\left\| \begin{pmatrix} 1 & -1 \\ 1 & 2 \end{pmatrix} \right\| = \|M\| = \sqrt{\lambda_{max}(M^TM)} = \tag{99}$$

$$= \sqrt{\lambda_{max}\left(\begin{pmatrix} 2 & 1 \\ 1 & 5 \end{pmatrix} \right)} \approx \sqrt{5.3} \approx 2.3 \tag{100}$$

and hence $\alpha \approx 1/2.3 \approx 0.43$.

Exercise 7 - Real Time Scheduling

Problems considered in this exercise: 7.3, 7.4, 7.9, 7.10

Brief Theory

- Scheduling algorithms (fixed and dynamic) - deciding what task sould be performed when, planning such that deadlines are met. Fixed priority such as RM determines which task has priority before planning starts, this never change. Dynamic priority such as EDF determines a law for priority before planning, priority is then updated dynamically during the scheduling.
- Utilization factor - Quantative metric on how much capacity the set of tasks requires to schedule.
- Periodic and aperiodic tasks - tasks that are released with a fixed period and tasks that are released aperiodically. The scheduling of these tasks are a bit different.

Problems and Solutions

Exercise 7.3

Consider the following set of tasks

	C_i	T_i	D_i
J_1	1	3	3
J_2	2	4	4
J_3	1	7	7

Are the tasks schedulable with rate monotonic algorithm? Are the tasks schedulable with earliest deadline first algorithm?

Solution: **How can we know if it is schedulable for RM/EDF?**

We can use the utilization factor: $U = \sum_{i=1}^{n} \frac{C_i}{T_i}$. We know that for

RM) Schedulable *if* $U < n(2^{1/n} - 1)$

EDF) Schedulable *if and only if* $U \leq 1$

The condition for EDF scehdulability is also the condition for schedulability overall. That is, if a set of tasks are schedulable, they are schedulable with EDF!

Let's check if the conditions hold!

$$U = \frac{1}{3} + \frac{2}{4} + \frac{1}{7} = \frac{41}{42} \approx 0.976 \tag{101}$$

$$n(2^{1/n} - 1) = 3(2^{1/3} - 1) \approx 0.78 \tag{102}$$

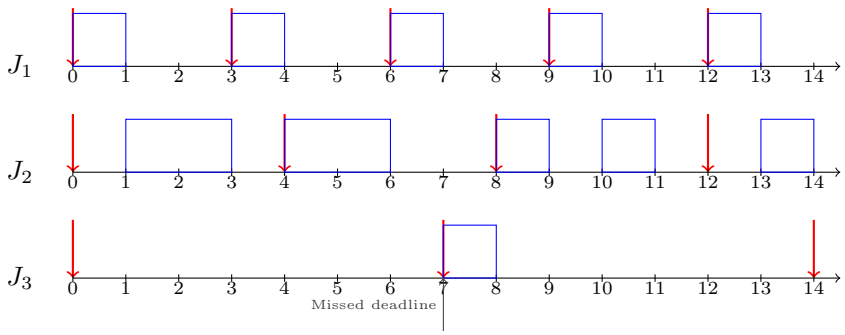
What are our conclusions?

We can conclude that it is schedulable with EDF, and it may be schedulable with RM (the condition is not satisfied but it is only sufficient not necessary).

How can we determine if it is schedulable with RM then?

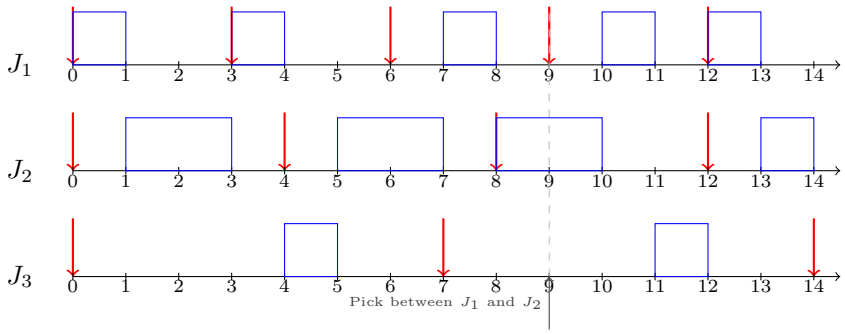
Let's draw the schedule!

1. Some analysis:
   length of schedule: $lcm(T_i) = 3 \cdot 4 \cdot 7 = 84$
   priority: shorter period=higher priority $\rightarrow T_1 < T_2 < T_3 \rightarrow J_1 > J_2 > J_3$
2. Draw! The drawing should include i) task releases, ii) which task is performed at each time-step, and iii) remarks on if deadlines are missed.



Bonus! Draw the schedule for EDF!

Priority: Earliest deadline first, i.e. the task with the next deadline is performed. If multiple deadlines at the same time, we can pick which one we want freely. OBS! If this is the case this should be marked in the schedule, pointing out that we have a choice and which choice you made.



Is this drawing enough to prove that it is EDF schedulable?

Not this one, but a drawing can be enough! However, to show it we need to make the drawing for the entire length, i.e. 84 time units in this case.

Exercise 7.4

Consider the following set of tasks

	C_i	T_i	D_i
J_1	1	4	4
J_2	2	5	5
J_3	3	10	10

Assume that task J_1 is a control task. Every time that a measurement is acquired, task J_1 is released. When executing, it computes an updated control signal and outputs it.

(a) Which scheduling of RM or EDF is preferable if we want to minimize the delay between the acquisition and control output?

(b) Suppose that J_2 is also a control task and that we want its maximum delay between acquisition and control output to be two time steps. Suggest a schedule which guarantees a delay of maximally two time steps, and prove that all tasks will meet their deadlines.

Solution:

What is the goal with respect to the tasks?

We want J_1 to be completed as fast as possible upon release.

How does RM and EDF work?

RM: Fixed priority. The task with shortest release period is given highest priority and hence it is always executed at release time.

EDF: Dynamic priority. The task with the closest deadline is executed at all time. Which task this is will hence vary throughout the schedule.

Which one seem to fit best with our goal?

RM! J_1 has the shortest release period and will hence always be performed at release!

Let's verify! How can we do that?

We can consider worst case response time R_k for fixed priority, as well as draw the schedules for all algorithms.

We have for task J_k that R_k is the smallest positive solution to (OBS! order according to priority):

$$R_i = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \tag{103}$$

We then get

$$R_1 = C_1 = 1 \tag{104}$$

This confirms our conclusion that task J_1 is completed at latest 1 time unit after release when using RM. However, this is only true if the task is schedulable with RM!

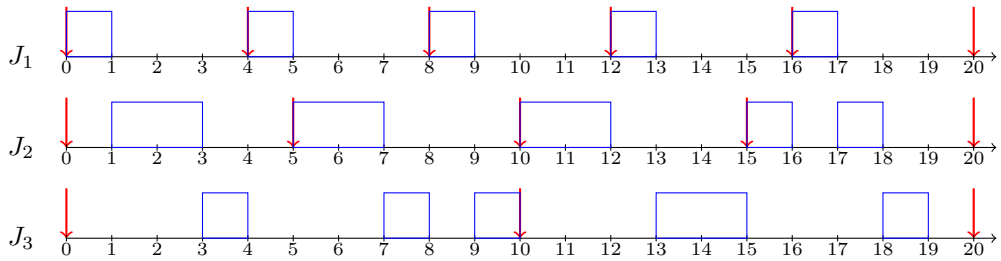
How can we check if it is schedulable with RM?

Utilization factor: $U = 0.95 > n(2^{1/n} - 1) = 0.78$. The condition doesn't hold.

Is it not schedulable with RM then?

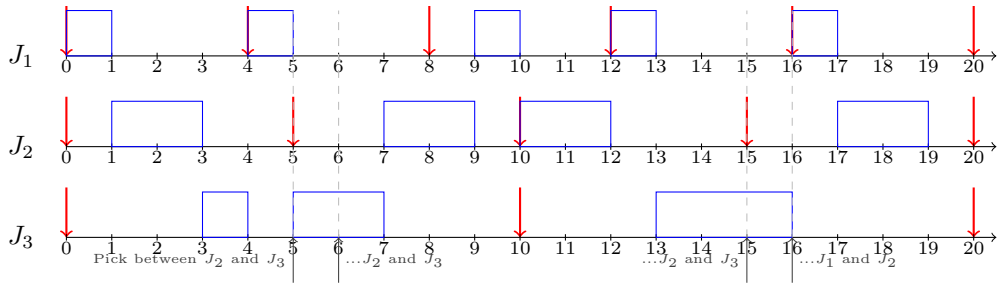
Not necessarily! We can draw the schedule to check!

Length: $lcm(T_i) = lcm(4, 5, 10) = 20$.
Priority: $T1 < T_2 < T_3 \rightarrow J_1 > J_2 > J_3$.



It is schedulable!

Draw for EDF to check worst case response time!



To minimize the time of execution for J_1, we make sure to always pick J_1 when we have the choice.

How can we check worst case response now?

Analyse the schedule! We can see from the figure that $R_1 = 2$. As expected RM was better w.r.t. our goal!

(b)

What is our new goal?

We need $R_2 = 2$ and have $C_2 = 2$, hence J_2 must be executed immediately on release. At the same time we want J_1 to be executed as fast as possible.

What algorithm should we use?

We can use RM and give J_2 highest priority and J_1 second highest priority. As discussed above J_2 will then be executed on release and J_1 as fast as possible after that.

How can we check schedulability?

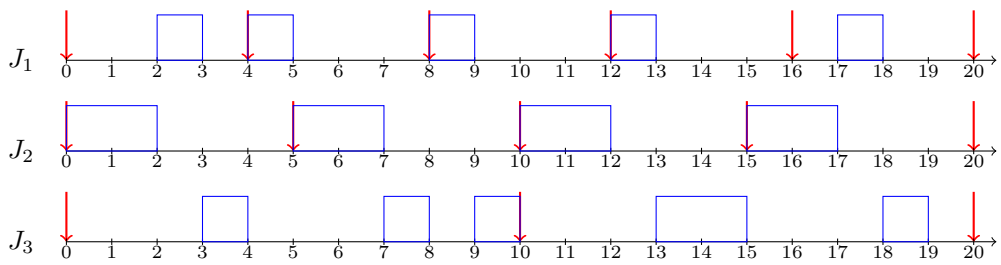
1. Draw schedule and check
2. Check if $R_i \leq D_i \ \forall i$ (in which case it is schedulable)

With the formula from above:

$$R_2 = C_2 = 2 \tag{105}$$
$$. \tag{106}$$
$$R_1^j = C_1 + \left\lceil \frac{R_1^{j-1}}{T_2} \right\rceil C_2 \tag{107}$$
$$R_1^0 = C_1 = 1 \tag{108}$$
$$R_1^1 = 1 + \left\lceil \frac{1}{5} \right\rceil 2 = 3 \tag{109}$$
$$R_1^2 = 1 + \left\lceil \frac{3}{5} \right\rceil 2 = 3 \tag{110}$$
$$\rightarrow R_1 = 3 \tag{111}$$
$$. \tag{112}$$
$$R_3^j = C_3 + \left\lceil \frac{R_3^{j-1}}{T_2} \right\rceil C_2 + \left\lceil \frac{R_3^{j-1}}{T_1} \right\rceil C_1 \tag{113}$$
$$R_3^0 = C_3 = 3 \tag{114}$$
$$R_3^1 = 3 + \left\lceil \frac{3}{5} \right\rceil 2 + \left\lceil \frac{3}{4} \right\rceil = 6 \tag{115}$$
$$R_3^2 = 3 + \left\lceil \frac{6}{5} \right\rceil 2 + \left\lceil \frac{6}{4} \right\rceil = 9 \tag{116}$$
$$R_3^3 = 3 + \left\lceil \frac{9}{5} \right\rceil 2 + \left\lceil \frac{9}{4} \right\rceil = 10 \tag{117}$$
$$R_3^4 = 3 + \left\lceil \frac{10}{5} \right\rceil 2 + \left\lceil \frac{10}{4} \right\rceil = 10 \tag{118}$$
$$\rightarrow R_3 = 10 \tag{119}$$

$R_2 = 2 < D_2 = 5$ (and achieves our goal), $R_1 = 3 < D_1 = 4$ and $R_3 = 10 = D_3$. Hence it is schedulable!

Bonus drawing as well!



We confirm that it is schedulable.

Exercise 7.9

Together with the periodic tasks

	C_i	T_i
J_1	1	4
J_2	1	8

we want to schedule the following aperiodic tasks with a polling server having $T_s = 5$ and $C_s = 2$. The aperiodic tasks are

	r_i	C_i
a_1	2	3
a_2	7	2
a_3	9	1
a_3	29	4

Solution:

How do we schedule aperiodic tasks? What is a polling server?

The polling server is scheduled as a periodic task. When the polling server gets execution time we have the possibility to execute an aperiodic task. We will only execute a task if it has been released. If no aperiodic task i available when the polling server gets it chance to execute that call is considered spent. That is, if an aperiodic task is released a time unit after the polling server was called it will have to wait to be executed until the next call of the polling server.

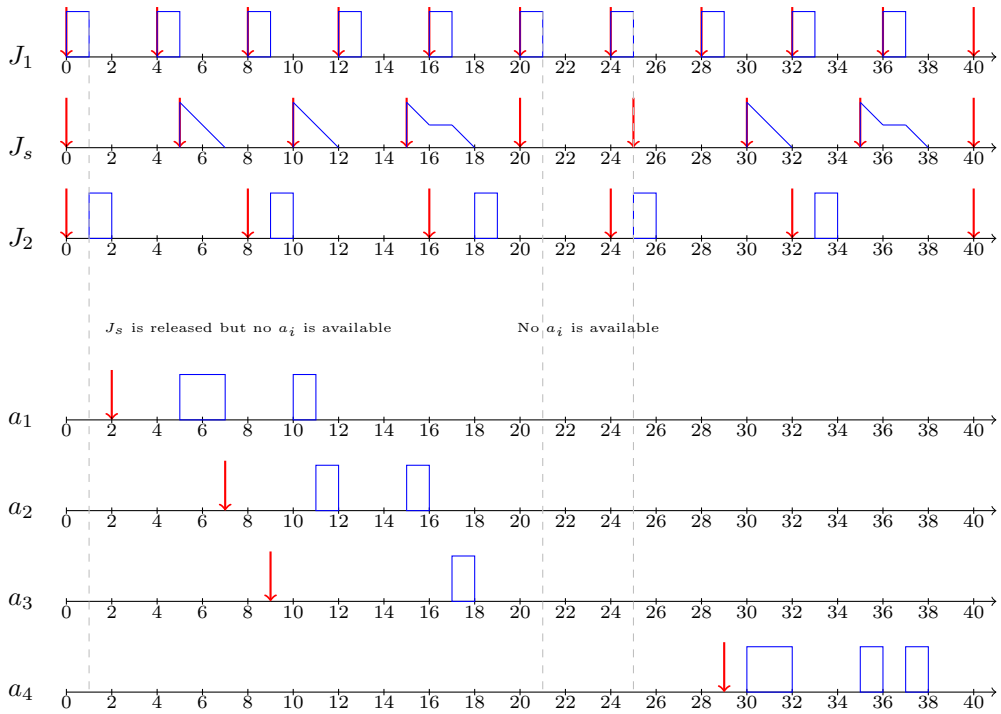
Is it schedulable?

$U = 0.775 < n(2^{1/n} - 1) = 0.78$

and hence we can use both RM or EDF to schedule the tasks!

We'll use RM!

Length: $lcm(4, 8, 5) = 40$
Priority: $J_1 > J_s > J_2$



Comments: At 2 time units a_1 is released, but J_s has laready been called this period. As a result a_1 will have to wait until the next call of J_s. At 7 time units J_s capacity is fully utilized and a_1 has to wait until the next call of J_s. The same thing occur for a_2 at 12 time units, and a_4 at 32 time units.

Exercise 7.10

Consider the set of tasks J_1 and J_2, assuming that an aperiodic task could ask for CPU time. In order to handle the aperiodic task we run a polling server J_s with computation time $C_s = 3$ and period $T_s = 6$. Assume that the aperiodic task has computation time $C_a = 3$ and asks for the CPU at time $t = 3$. Plot the time evolution when a polling server is used together with the two tasks J_1 and J_2 using the rate monotonic algorithm.

	C_i	T_i	D_i
J_1	1	3	3
J_2	1	4	4

Describe the scheduling activity illustrated in the plots.

Solution: **Is it schedulable?**

Check utilization factor: $U = 13/12 > 1$

What does this mean?

This indicates that it is not schedulable with any algorithm!

Is it impossible to solve the problem?

Not necessarily! Since J_s is a polling server for aperiodic tasks it may still be schedulable. However, if all tasks would've been periodic it would not be schedulable!

What is our aim?

Schedule J_1, J_2 and J_s using RM.

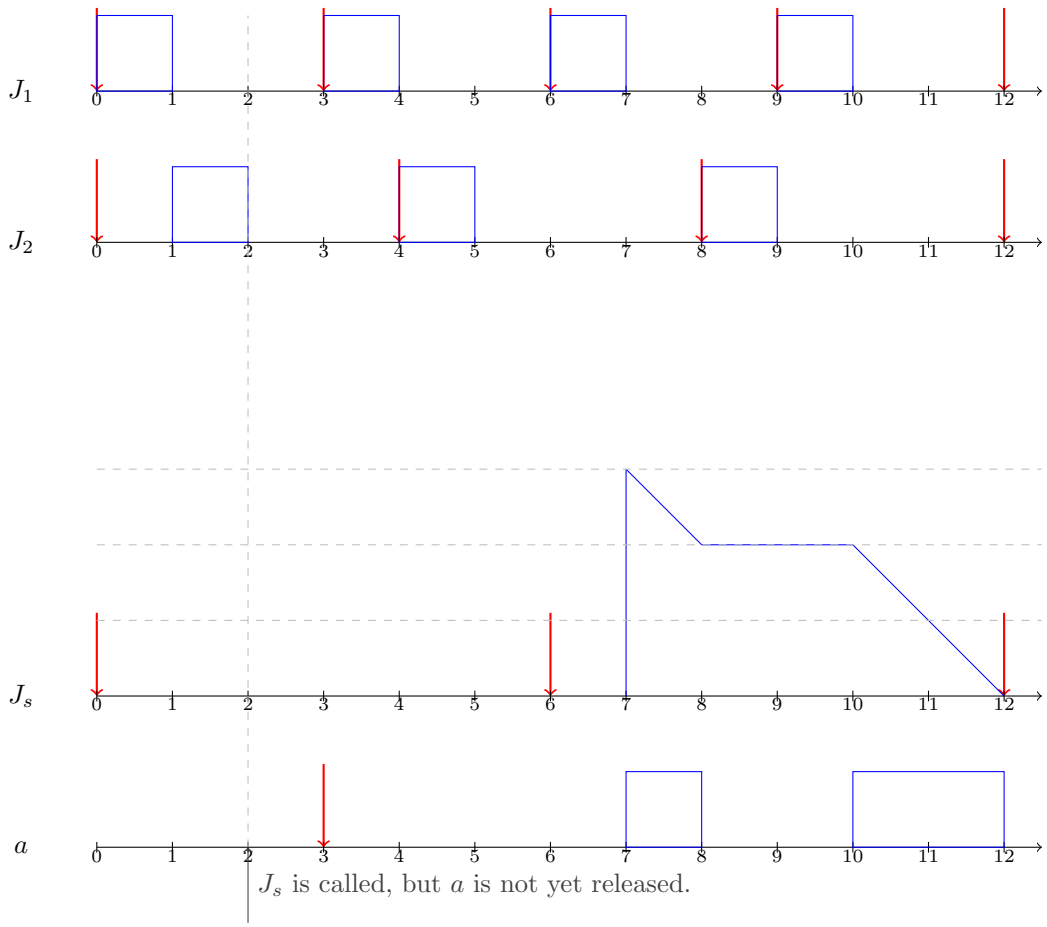
Gather the information!

	C_i	T_i	D_i
J_1	1	3	3
J_2	1	4	4
J_s	3	6	-

	C_i	r_i
a	3	3

Draw a schedule!

Length: $lcm(3, 4, 6) = 12$
Priority: $T_1 < T_2 < T_s \rightarrow J_1 > J_2 > J_s$



Exercise 8 - Models of Computation (Automata, Transition Systems)

Problem considered in this exercise: 8.2, 8.3, 8.6, 8.7, 8.9, 8.11

Brief Theory

- Transition System
- Automaton (Discrete Event System)

Both are tuples used to model/describe some system.

The transition system $T = \{S, \Sigma, \rightarrow, S_0\}$ where S is a set of states, Σ is a set of actions, $\rightarrow$ is a set of transitions, and S_S is a set of initial states. The transition system has no sense of goals.

The automaton $A = \{Q, E, \delta, q_0, Q_m\}$ where Q is a set of states, E is a set of events, δ is a set of transitions, q_0 is a set of initial states, and Q_m is a set of marked/final/accepting states.

Simplified, transition systems are used to model how something works while automata are used to model how something should be done. That is if we have a self-driving car and we want it to avoid crashing into obstacles while driving to our goal, we can model the car with a transitions system and the task (avoiding crashes and reaching the goal) with an automata.

Problems and Solutions

Exercise 8.2

A vending machine dispenses soda for \$0.45. It accepts only dimes (\$0.10) and quarters (\$0.25). It does not give change in return if your money is not correct. The soda is dispensed only if the exact amount of money is inserted. Model the vending machine using a discrete-event system. Is it possible that the machine does not dispense soda? Prove it formally.

Solution: We should use an automaton to model the system.

What should the elements of the automaton describe?

Q: states, **what can the status be?** Total amount of inserted money

E: events, **what can happen?** Money is inserted, dime or quarter

δ: transitions, **how does the state change given an event?** How the total amount of inserted money change when we insert a dime or a quarter.

q_0: initial states, **where do we start?** Amount of money inserted at start.

Q_m: marked states, **what is our goal?** Amount of money when achieveing our goal.

Determine the specific values of the above!

All combinations of dimes and quarters:

$$Q = \{0, 10, 20, 25, 30, 35, 40, 45, \square\} \tag{120}$$

where $\square$ symbolizes > 45.

Quarter or dime (we can insert oen of them):

$$E = \{10, 25\} \tag{121}$$

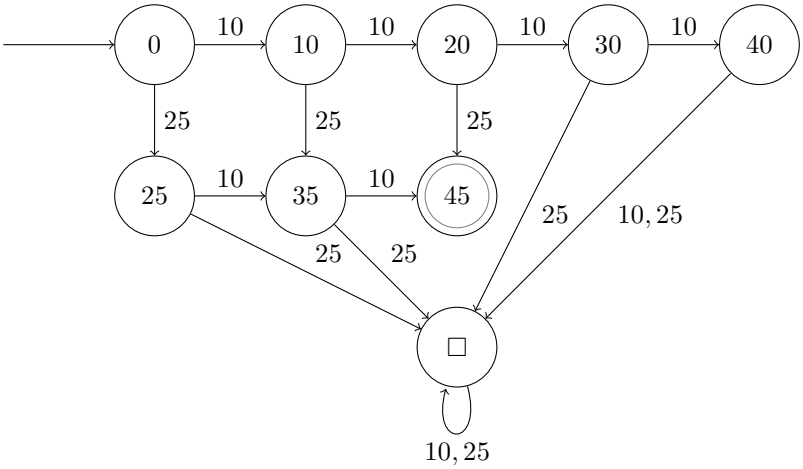
$$\begin{aligned}
\delta(q, e) = q' : \quad & \delta(0, 10) = 10 \quad \delta(0, 25) = 25 && (122)\\
& \delta(10, 10) = 20 \quad \delta(10, 25) = 35 && (123)\\
& \delta(20, 10) = 30 \quad \delta(20, 25) = 45 && (124)\\
& \delta(25, 10) = 35 \quad \delta(25, 25) = \square && (125)\\
& \delta(30, 10) = 40 \quad \delta(30, 25) = \square && (126)\\
& \delta(35, 10) = 45 \quad \delta(35, 25) = \square && (127)\\
& \delta(40, 10) = \square \quad \delta(40, 25) = \square && (128)\\
& \delta(\square, 10) = \square \quad \delta(\square, 25) = \square && (129)
\end{aligned}$$

What happens when $q = 45$? The goal is reached and the soda is dispensed. Either we consider the machine to be one use only, in which case $\delta(45, e) = 45$, $\delta(45, e) = \emptyset$ and $\delta(45, e) = \square$ are all reasonable transitions, or the machine should be reset to $q = 0$ to allow a new try to take place. Here we will use $\delta(45, e) = \emptyset$.

Start value and goal value:

$$q_0 = 0 \quad Q_m = 45 \tag{130}$$

Let's draw the automaton!



What can we conclude from the drawing?

There's a livelock at $\square$, i.e. if we reach $q = \square$ $q = 45$ is not reachable and there is no way of dispencing the soda. $q = 45$ is not reachable from $30, 40$ or $\square$. There is also a deadlock at $q = 45$.

What about language? What is language?

The language of a DES $L(A)$ is the set of all words which can be made with A starting in q_0, where words are combinations of events $e \in E$. The marked language $L_m(A)$ is the subset of $L(A)$ which leads to Q_m.

In our case? $L(A)$ is then any combination of 10 and 25: $L(A) = \{(10^*25^*)^*\}$, here x^* indiactes that we can repeat x any number of times from 0 to ∞.

$L_m(A)$ is all combinations of 10 and 25 that will add up to 45: $L_m(A) = \{10^2\ 25, 10\ 25\ 10, 25\ 10^2\}$. Note that depending on how we define the transitons from 45 the words in L_m includes suffixes of any combination, i.e. $L_m(A) = \{10^2\ 25(10^*25^*)^*, 10\ 25\ 10(10^*25^*)^*, 25\ 10^2(10^*25^*)^*\}$.

Will the soda always be dispensed?

No!

- If we have defined the automata such that 45 has self-loops and consider only infinite words: since $L_m(A) \neq L(A)$ there are words of $L(A)$ which doesn't lead to Q_m, and hence the soda may not be dispensed. OBS! When considering infinite words, a word belongs to the marked language if we will visit a marked state an infinite number of times.
- If we consider finite words, i.e. we stop when we reach a marked state: the soda may be impossible to dispense if there exists at least one word $w \in L(A)$ for which $w \notin L_m(A)$ and w is not a prefix of a word in $L_m(A)$, i.e. there is no word $w' \in L_m(A)$ s.t. $w' = ww''$ where w'' is any suffix.

In the discussion above, and in this exercise overall we have considered finite words, hence the second item is the correct motivation to use here.

Exercise 8.3

Consider the automaton describing some discrete-event system shown in Figure 2. Describe formally the DES. Compute the marked language L_m and the generated language L.

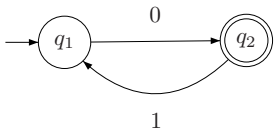


Figure 2: Automaton A.

Solution:

We need to determine Q, E, δ, q_0 and Q_m!

What are they?

$$
\begin{aligned}
Q &= \{q_1, q_2\} \\
E &= \{0, 1\} \\
\delta(q, e): &\ \delta(q_1, 0) = q_2,\ \delta(q_2, 1) = q_1 \\
q_0 &= q_1 \\
Q_m &= \{q_2\}
\end{aligned}
$$

What is the language of A?

The combinatins of 0 and 1 which we can create, starting at q_0:

$$L(A) = \{0, 01, 010, 0101, ...\} = \{(01)^*, 0(10)^*\} \tag{131}$$

What about the marked language?

The subset of $L(A)$ such that we end up in Q_m:

$$L_m(A) = \{0, 010, 01010, ...\} = \{0(10)^*\} \tag{132}$$

Exercise 8.6

Consider the automaton

$$A = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\})$$

be a nondeterministic automaton where

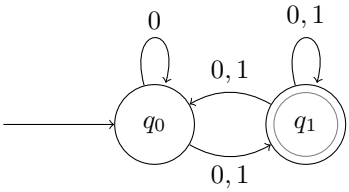
$$\delta(q_0, 0) = \{q_0, q_1\} \quad \delta(q_0, 1) = \{q_1\} \quad \delta(q_1, 0) = \quad \delta(q_1, 1) = \{q_0, q_1\}.$$

Construct an deterministic automaton A' which accept the same L_m.

Solution:

What is the marked language of A? Any word that takes us from q_0 to q_1 is contained by L_m, i.e. any combination of 0 and 1.

Let's draw the automaton!



What conclusions can we make from the figure?

We note from the transitions that both 0 and 1 may lead from q_0 to q_1 and that both 0 and 1 may lead to q_1 from q_1. It then follows that $L_m(A) = L(A) = \{0^*1^*0^*\}$.

What is a non-deterministic automaton/deterministic automaton? Non-deterministic: a specfic event at a specific state leads to several successors! Determenistic: a specific event at a specific state has only one successor!

How can we define a deterministic automaton with the same language? We need to eliminate the non-deterministic transitions!

Introduce a new state $q_2 = [q_0, q_1]$!

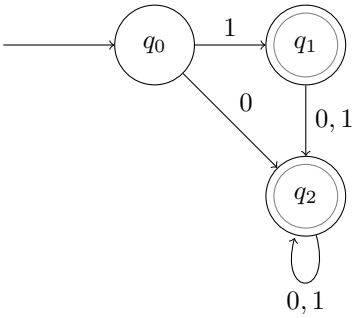
We then get:

$Q = \{q_0, q_1, q_2\}$, $E = \{0, 1\}$ (as before)

$$\delta(q_0, 0) = q_2 \qquad \delta(q_0, 1) = q_1 \tag{133}$$
$$\delta(q_1, 0) = q_2 \qquad \delta(q_1, 1) = q_2 \tag{134}$$
$$\delta(q_2, 0) = q_2 \quad \delta(q_2, 1) = \{q_1, q_2\} = q_2 \tag{135}$$

$q_0 = q_0$ (as before), and $Q_m = \{q_1, q_2\}$

Let's draw the automaton!



What is the language now? Did we do it right?

$L(A') = \{(0^*1^*)^*\}$ (as before any combination of 0 and 1)
$L_m(A') = \{0((0^*1^*)^*), 10((0^*1^*)^*), 11((0^*1^*)^*)\} = \{(0^*1^*)^*\}$ (as before).

Hence, we have succeeded!

Exercise 8.7

Consider the circuit diagram of the sequential circuit with input variable x, output variable y, and register r, cf. Figure 3. The control function for output variable y is given by

$$\lambda_y = \neg(x \oplus r)$$

where $\oplus$ stands for exclusive (XOR, or parity function). The register evaluation changes according to the circuit function

$$\delta_r = x \vee r$$

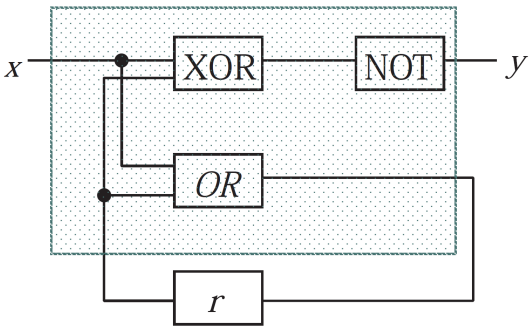


Figure 3: Diagram for a simple hardware circuit.

where $\vee$ stands for disjunction (OR). Initially, the register evaluation is $[r = 0]$.
Model the circuit behavior by a finite transition system.

Solution: **How does the circuit work?** We control x by setting it to 0 or 1. The register r is determined by δ_r, it depends on x and r from the previous time step. The output y is determined by λ_y, it depends on r and x. OR or $\wedge$ is a disjunction operator, $a \wedge b$ will be true (or have value 1) if either a, b or both have value 1. XOR or $\oplus$ is an exlusive disjunction operator, $a \oplus b$ will be true (or have value 1) if a or b has value 1 (not if both have!).

What is a transition system? What do we need to do?

$$T = (S, \Sigma, \rightarrow, S_0) \tag{136}$$

S: states - what is the status?

Σ: actions - what can happen?

$\rightarrow \subset S \times \Sigma \times S$: transition relation - how will the system react to actions?

S_0: initial states - where do we begin?

Identify the elements!

States: register evaluation and input variable (x, r) can be 0 or 1: $\{(0,0),(0,1),(1,0),(1,1)\}$.
Actions: input variable σ can be 0 or 1: $\{0,1\}$
Transition relation: x' takes on the value of σ, at the same time

$$r(k+1) = \begin{cases} 0 & \text{if } x(k) = r(k) = 0 \\ 1 & \text{if } x(k) = 1, r(k) = 0 \text{ or } x(k) = 0, r(k) = 1 \text{ or } x(k) = r(k) = 1 \end{cases}$$

and

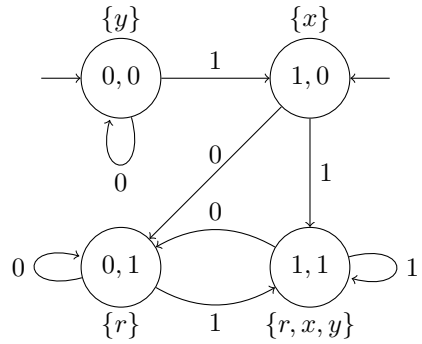
$$y(k+1) = \begin{cases} 1 & \text{if } x(k) = r(k) \\ 0 & \text{if } x(k) \neq r(k) \end{cases}$$

Hence we should have transitions: $(x(k), r(k)) \xrightarrow{\sigma} (\sigma, r(k+1))$

$$(0,0) \xrightarrow{0} (0,0) \quad (0,0) \xrightarrow{1} (1,0) \tag{137}$$
$$(0,1) \xrightarrow{0} (0,1) \quad (0,1) \xrightarrow{1} (1,1) \tag{138}$$
$$(1,0) \xrightarrow{0} (0,1) \quad (1,0) \xrightarrow{1} (1,1) \tag{139}$$
$$(1,1) \xrightarrow{0} (0,1) \quad (1,1) \xrightarrow{1} (1,1) \tag{140}$$
$$\tag{141}$$

Initial states: we're given that $r = 0$ at the start: $\{(0,0),(1,0)\}$

Let's draw!



where the curly bracket indicates the output of each state.

OBS! There are of course multiple ways to model the system! For instance one could use (y, x, r) as states instead of (x, r). The result would then be a larger transition system:

$S' = \{(0,0,0),(1,0,0),(0,0,1),(1,0,1),(0,1,0),(1,1,0),(0,1,1),(1,1,1)\}$,
$\Sigma' = \Sigma$,
$(y(k), x(k), r(k)) \xrightarrow{\sigma} (y(k+1), \sigma, r(k+1))$

$$(0,0,0) \xrightarrow{0} (1,0,0) \quad (0,0,0) \xrightarrow{1} (1,1,0) \tag{142}$$
$$(1,0,0) \xrightarrow{0} (1,0,0) \quad (1,0,0) \xrightarrow{1} (1,1,0) \tag{143}$$
$$(0,0,1) \xrightarrow{0} (0,0,1) \quad (0,0,1) \xrightarrow{1} (0,1,1) \tag{144}$$
$$(1,0,1) \xrightarrow{0} (0,0,1) \quad (1,0,1) \xrightarrow{1} (0,1,1) \tag{145}$$
$$(0,1,0) \xrightarrow{0} (0,0,1) \quad (0,1,0) \xrightarrow{1} (0,1,1) \tag{146}$$
$$(1,1,0) \xrightarrow{0} (0,0,1) \quad (1,1,0) \xrightarrow{1} (0,1,1) \tag{147}$$
$$(0,1,1) \xrightarrow{0} (1,0,1) \quad (0,1,1) \xrightarrow{1} (1,1,1) \tag{148}$$
$$(1,1,1) \xrightarrow{0} (1,0,1) \quad (1,1,1) \xrightarrow{1} (1,1,1) \tag{149}$$

and $S_0 = \{(0,0,0),(0,1,0),(1,0,0),(1,1,0)\}$.

Exercise 8.9

Consider a simple T-intersection of the traffic system, as shown in Fig. 4. There are four types of vehicles:

- $(1, 2)$-type vehicles coming from point 1 and turning right towards 2;
- $(1, 3)$-type vehicles coming from point 1 and turning left towards 3;
- $(2, 3)$-type vehicles going straight from 2 to 3;
- $(3, 2)$-type vehicles going straight from 3 to 2.

The traffic light is set so that it either turns red for $(1, 2)$ and $(1, 3)$ vehicles (green for $(2, 3)$ and $(3, 2)$ vehicles), or it turns green for $(1, 2)$ and $(1, 3)$ vehicles (red for $(2, 3)$ and $(3, 2)$ vehicles). Model the traffic system as a transition system $\mathcal{T} = (S, Act, \rightarrow, I)$ to describe the changes to the number of the vehicles of the individual types waiting at the intersection over time. Model arrivals of new cars and departures of the waiting cars and changes of the traffic light colors. Assume that initially, there is no vehicle at the intersection.

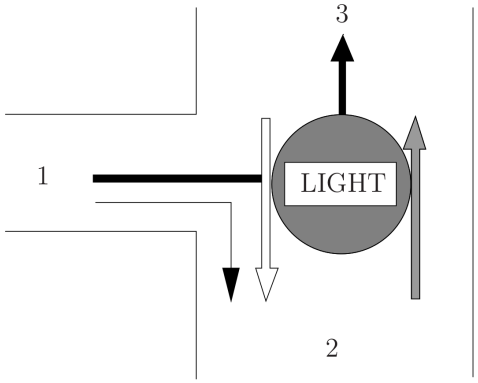


Figure 4: A simple T-intersection with four types of vehicles.

Solution:

What should the elements be?

States: possible status of the intersection= number of vehicles of each type and color of traffic lights. We'll use the notation $c \in \{G, R\}$ for color of traffic lights, where G indicates that it is green for cars of type (1,2) and (1,3) and R indicates that the same cars have a red light. We'll also use x_{ij} to indicate the number of cars of type (i,j):

$$S = \{(x_{12}, x_{13}, x_{23}, x_{32}, c) : x_{ij} \in \mathbb{N}_0, c \in \{G, R\}\} \tag{150}$$

where $\mathbb{N}_0$ is the set of natural numbers including 0, i.e. 0 or any positive integer.

Actions: what can happen = arrival of car of type (i,j), departure of car of type (i,j), light color change. We'll use the notation g to indicate that the light turns green, r to indicate that it turns red, a_{ij} to indicate that a car of type (i,j) arrives, and d_{ij} to indicate that a car of type (i,j) departs:

$$\Sigma = \{a_{12}, a_{13}, a_{23}, a_{32}, d_{12}, d_{13}, d_{23}, d_{32}, g, r\} \tag{151}$$

Initial state: how we start =we should assume that the intersection is empty:

$$S_0 = \{(0,0,0,0,G),(0,0,0,0,R)\} \tag{152}$$

Transition relation: How the system reacts = if light turns green (g) c should change to G in the state, if a car arrives the corresponding car number should increase, and if a car departs the corresponding car number should decrease:

$$Ex: \quad (x_{12},x_{13},x_{23},x_{32},G) \xrightarrow{a_{13}} (x_{12},x_{13}+1,x_{23},x_{32},G) \tag{153}$$

$$(x_{12},x_{13},x_{23},x_{32},G) \xrightarrow{d_{32}} (x_{12},x_{13},x_{23},x_{32}-1,G) \tag{154}$$

$$(x_{12},x_{13},x_{23},x_{32},G) \xrightarrow{r} (x_{12},x_{13},x_{23},x_{32},R) \tag{155}$$

etc

Can we draw the system? Why/why not?

No! Since x_{ij} can take on any positive value there's an infinte number of states.

Exercise 8.11

Consider the transition system $T = \{S, \Sigma, \rightarrow, S_S\}$, where the cardinality of S is finite. The reachability algorithm is

Initialization : $\mathrm{Reach_1} = \emptyset$;
$\mathrm{Reach_0} = \mathrm{S_S}$;
$i = 0$;
Loop : $While \quad \mathrm{Reach_i} \neq \mathrm{Reach_{i-1}} \quad do$
$\mathrm{Reach_{i+1}} = \mathrm{Reach_i} \cup \{\mathrm{s'} \in \mathrm{S} : \exists \ : \mathrm{s} \in \mathrm{Reach_i}, \sigma \in \Sigma, \mathrm{s} \rightarrow^{\sigma} \mathrm{s'} \in \rightarrow\}$;
$i = i + 1$;

Prove formally that

- the reachability algorithm finishes in a finite number of steps;
- upon exiting the algorithm, $Reach_i = Reach_T(S_S)$.

Solution:

Does it make sense? In each loop we add the states s' to the set, where s' is any state whcih can be reached from some state s which is already in the set.

The algorithm stops when the set no longer changes. That is, we stop when the set is invariant (all states which can be reached from the set are in the set).

Can we make a condition for if we'll stop then?

Yes! We will stop if there is a finite number of states which can be added to the set!

Formally: Each loop we have to add at least one state s' to the set $Reach_i$, or the algorithm stops. The cardinality of S is finite, i.e. the number of states are

finite. Let's intoduce X as the cardinality of S (number of states). The maximum number of states which can be added, and hence the maximum number of loops are then $X - 1$ (since at least one state must be included in $S_S = Reach_0$ before the loop begins).

What about the resulting set? We know that $Reach_i$ contains all states which are reachable from $Reach_{i-1}$ in one step. Using induction we can show that $Reach_i$ contains all states which are reachable from $Reach_0$ in i steps. Since $Reach_0 = S_S$ and $Reach_T(S_S)$ si the set of reachable states from S_S in any number of steps it follows that $Reach_i \subset Reach_T(S_S)$ for any value of i. However, from the discussion above we know that all states which are reachable from S_S in any number of steps are in $Reach_i$ when the algorithm stops, since $S_S \in Reach_i$, hence $Reach_T(S_S) \subset Reach_i$ for the final set $Reach_i$. It then follows that $Reach_i = Reach_T(S_S)$ for the final set $Reach_i$.