Homework assignment 2 - EL2450

Adam Lang (861110-3956) & Gabriel Andersson Santiago (910706-4538)

February 21, 2016

1 Rate Monotonic

1.1

Rate Monotonic scheduling is a scheduling method that will predetermine the priority of each task proportional to the tasks activation frequency. The priority is determined at the task creation and will remain unchanged during the whole application.

1.2

A set of tasks $J = \{J_1, J_2, ..., J_n\}$ is schedulable with RM if

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

where C_i is the computation time, T_i is the period, U is the utilization factor and n is the number of computations. When we have a sampling time of $T = \{20, 29, 30\}ms$ and a computation time of 6 ms each we can see that,

$$\frac{6}{20} + \frac{6}{29} + \frac{6}{35} = 0.678 \tag{2}$$

and with a utilization factor of $U = 0.780$ we can see that the set of tasks J should be schedulable with RM.

1.3

The pendulums are indeed stabilized. There is however a slight difference in the control performance. From what we can see is the settling time longer the shorter the pendulum gets. Which is correct according to the lab introduction.

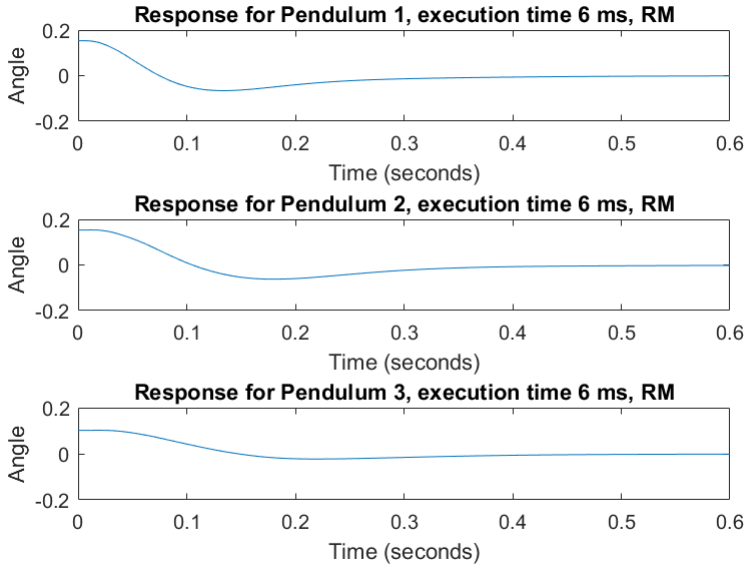

Figure 1: Pendulum angles for the three different pendulums

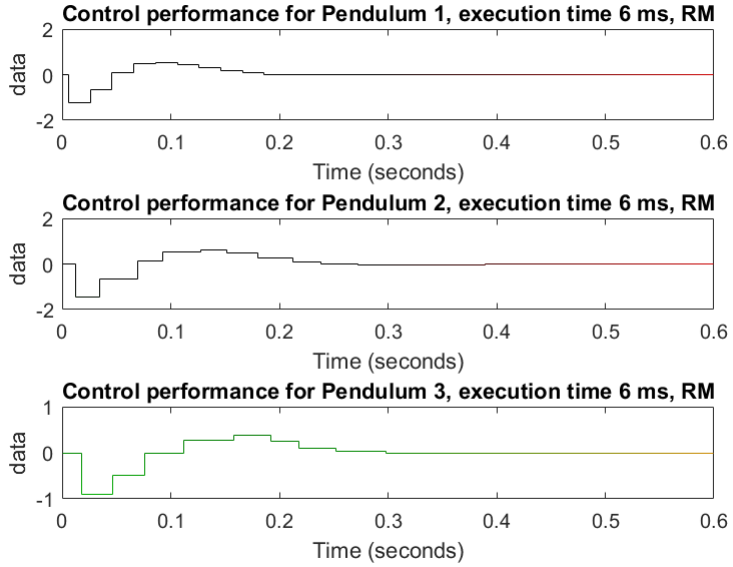

Figure 2: Control signal for pendulum

1.4

The schedule plot from Simulink corresponds to our own schedule. This can be seen in figure 3 and 4. Be aware that our schedule is ordered from shortest-longest pendulum and the one from Simulink is longest-shortest pendulum.

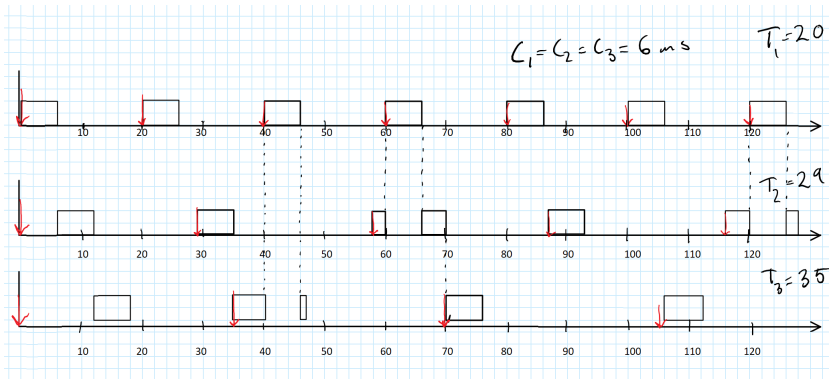

Figure 3: Our schedule for 6 ms computation time

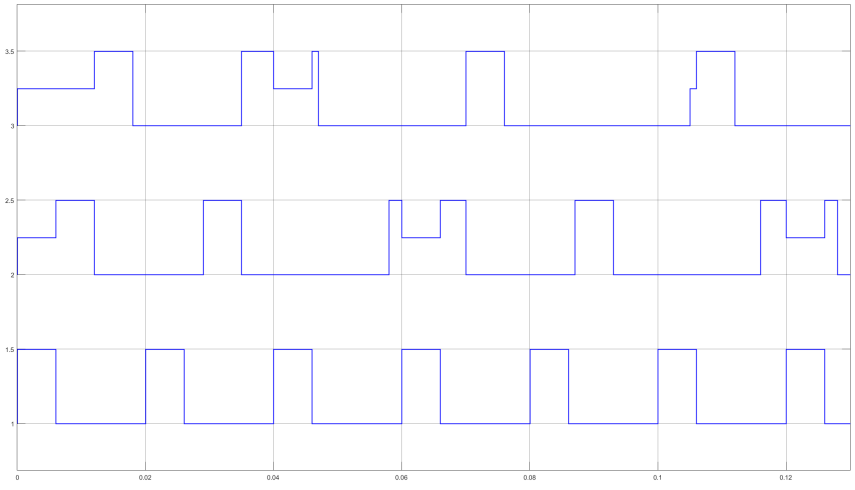

Figure 4: Schedule from Simulink for 6 ms

1.5

The longest pendulum becomes unstable after a few seconds due to missed deadlines.

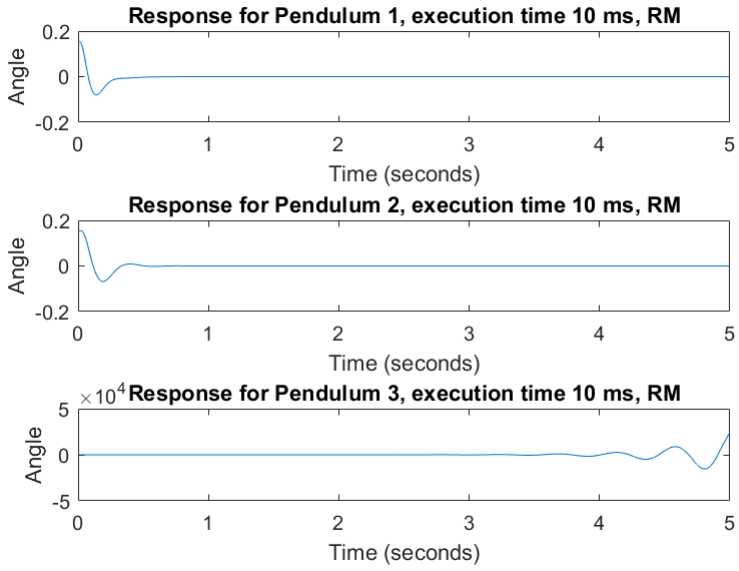

Figure 5: Pendulum angles for the three different pendulums

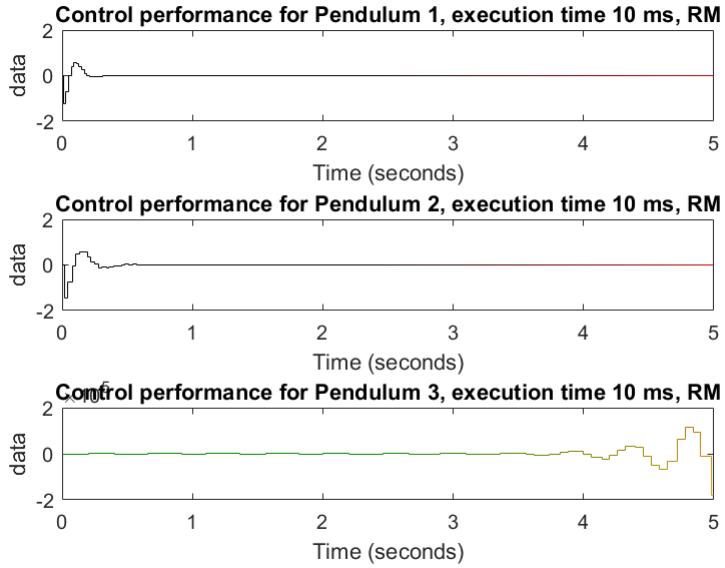

Figure 6: Control signal for pendulum

Deadlines are missed for the long pendulum which means that the tasks are not schedulable. This is verified by calculating the equation

$$U \leq n(2^{(1/n)} - 1 = 3 * (2^{(1/3)} - 1) = 0.779 \tag{3}$$

where

$$U = \sum_{i=1}^{n} \frac{C_i}{D_i} = \frac{10}{20} + \frac{10}{29} + \frac{10}{35} = 1.13 \tag{4}$$

This means that it is not schedulable due to

$$1.13 > 0.779 \tag{5}$$

Our schedule for a computation time of 10 ms can be seen in figure NUMBER as well can the schedule from Simulink be seen in figure NUMBER. Our schedule is equal to the one from Simulink.

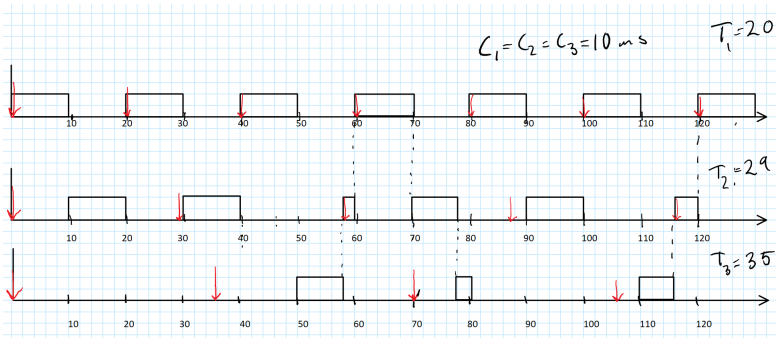

Figure 7: Our schedule for 10 ms computation time

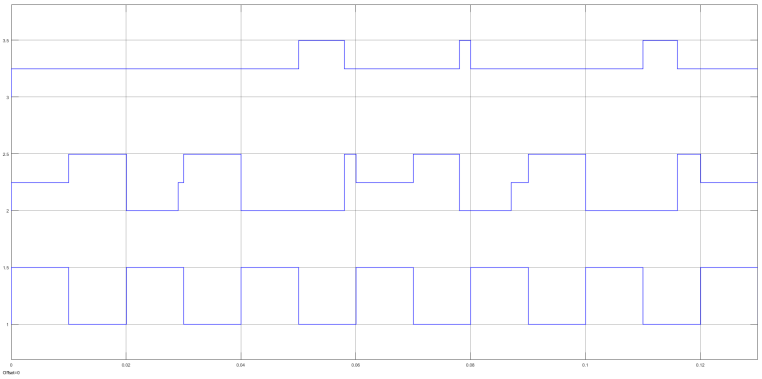

Figure 8: Simulink schedule for 10 ms computation time

2 Earliest Deadline First

2.1

Earliest deadline first scheduling executes the task with the shortest time left of its deadline d_k. This means that the schedule dynamically changes depending on the different task's deadlines. The advantages of EDF compared to RM is that EDF fully uses the processor computational power while RM is limited to $U = n(2^{1/n} - 1)$. Disadvantage is

2.2

Tasks are schedulable with $EarliestDeadlineFirst$ if

$$U <= 1 \tag{6}$$

which in this case is

$$U = \frac{6}{20} + \frac{6}{29} + \frac{6}{35} = 0.678 < 1 \tag{7}$$

This means that the tasks are schedulable which we verified with our schedule that is shown in 2.4.

2.3

The pendulums have stabilized. The shortest one is the hardest to control just as with Rate monotonic scheduling. There is not a huge difference in the control signal but you can see the sampling time differences. Since pendulum 3 is sampled the slowest it also takes longest time for it to be completely stable.

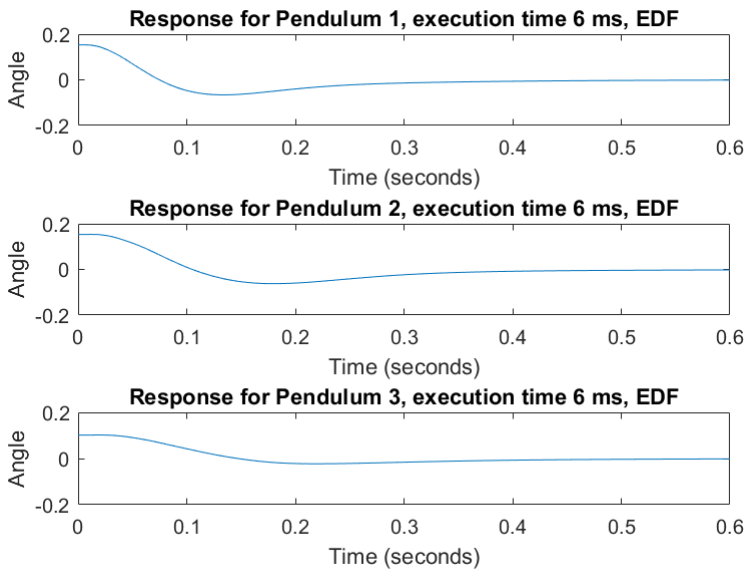

Figure 9: Pendulum angles for the three different pendulums

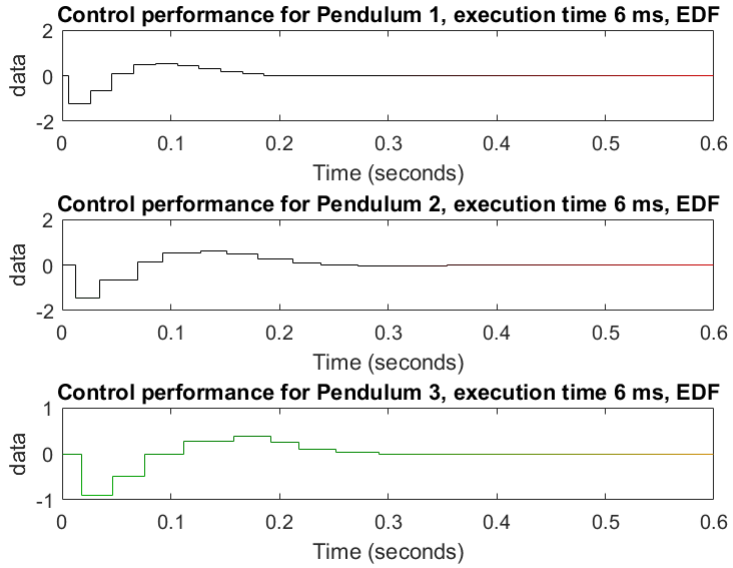

Figure 10: Control signal for the three different pendulums

2.4

Our schedule for EDF with 6 ms can be seen in figure NUMBER. The schedule from Simulink can be seen in figure NUMBER. The result from Simulink agrees with our schedule; all tasks are carried out before the next sampling time.

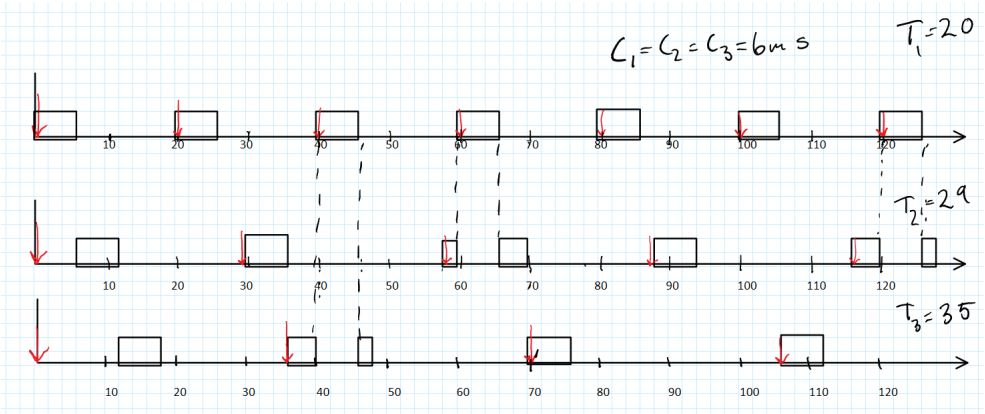

Figure 11: Our schedule for 6 ms computation time, EDF

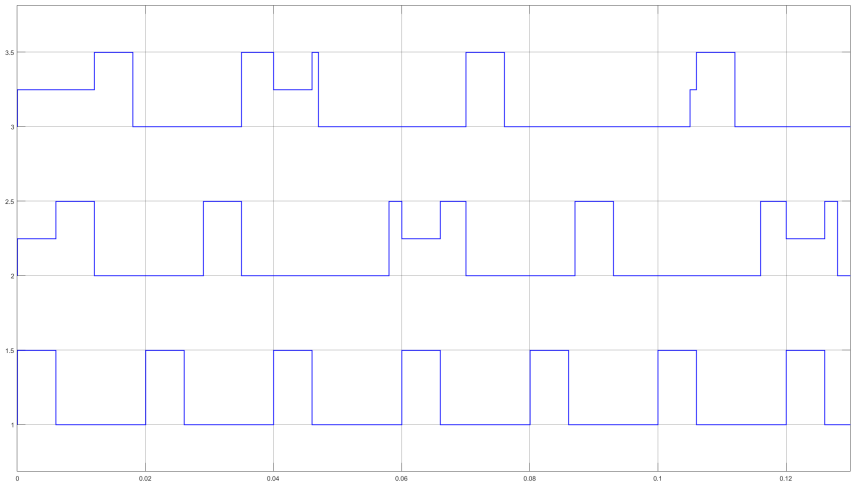

Figure 12: Simulink schedule for 6 ms computation time, EDF

2.5

Even though deadlines are missed with EDF the pendulums still stabilizes.

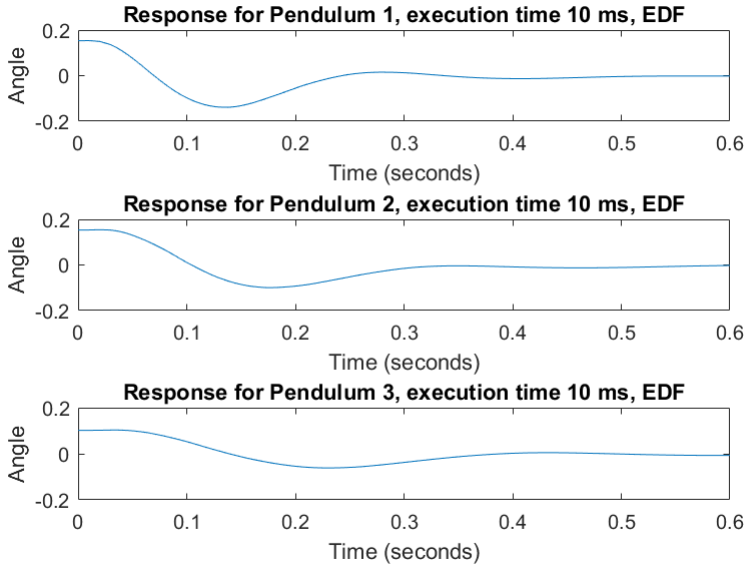

Figure 13: Pendulum angles for the three different pendulums, EDF

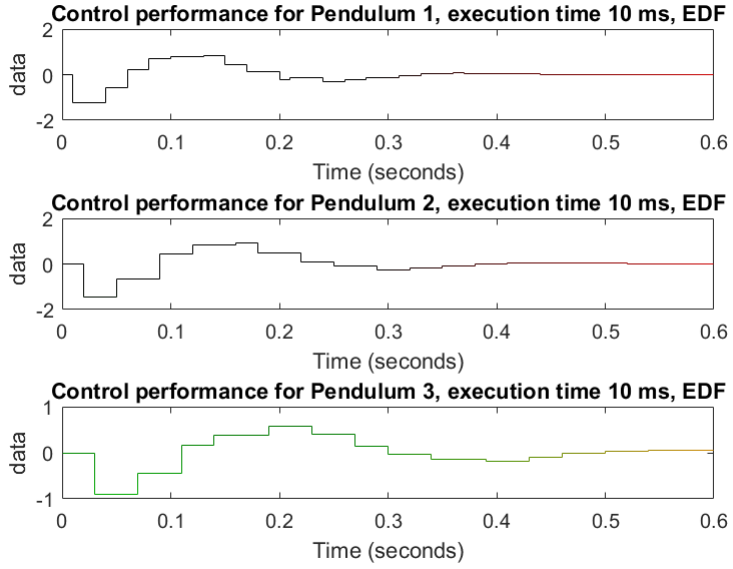

Figure 14: Control signal for the three different pendulums, EDF

The tasks are not schedulable with a computation time of 10 ms due to

$$U = \sum_{i=1}^{n} \frac{C_i}{D_i} = \frac{10}{20} + \frac{10}{29} + \frac{10}{35} = 1.13 > 1 \tag{8}$$

which we also verified when we created our schedule. Our schedule for a computation time of 10 ms with EDF is shown in figure NUMBER. Simulink schedule for a computation time of 10 ms is shown in figure NUMBER. Our schedule matches the one from Simulink.

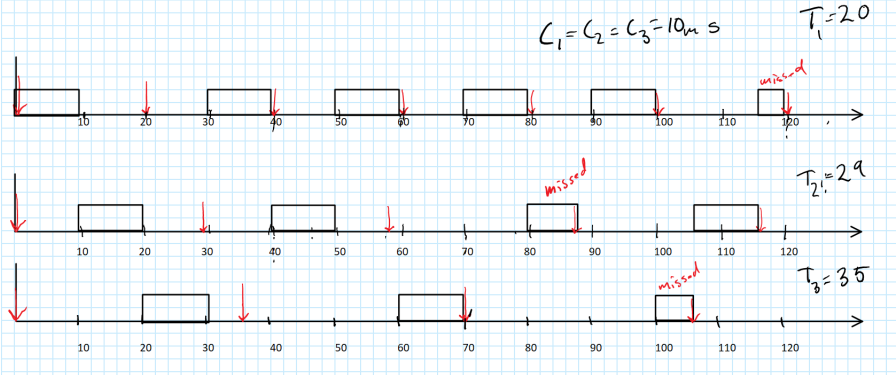

Figure 15: Our schedule for 10 ms computation time, EDF

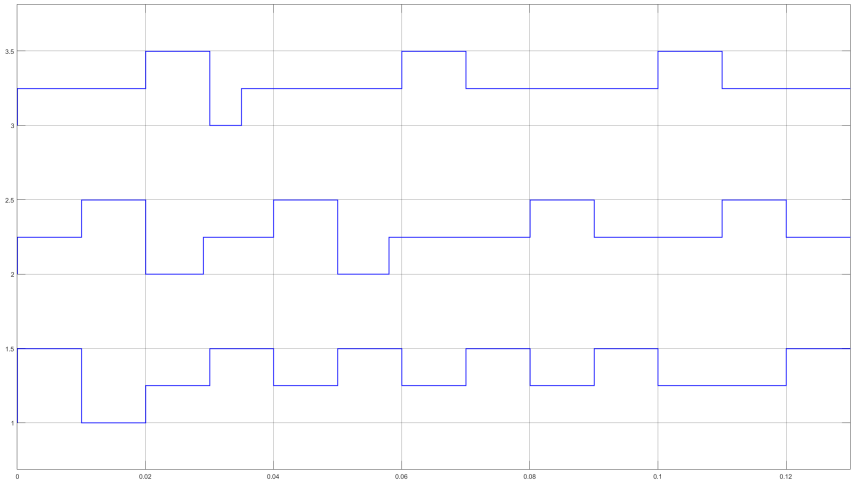

Figure 16: Simulink schedule for 10 ms computation time, EDF

2.6

The controller performs better with Earliest Deadline First due to every pendulum stabilizes which pendulum 3 does not with Rate Monotonic.

3 Network Control System

3.1

We have the following system

$$\dot{x}(t) = Ax(t) + Bu(t)y(t) = Cx(t) \tag{9}$$

where

$$A = 0$$
$$B = I$$

and the discrete controller

$$u(kh) = -Kx(kh), k = 0, 1, 2, ... \tag{10}$$

discretizing the system in equation 9 will give that the plant can be writen

$$x((k+1)h) = \Phi x(kh) + \Gamma_0(\tau_)u(kh) + \Gamma_1(\tau)u((k-1)h) \tag{11}$$

Given that $A = 0, B = I$ and that $\tau = \tau_{sc} + \tau_{ca} \leq h$ we can calculate the matrices,

$$\Phi = e^{Ah} = 1$$
$$\Gamma_0(\tau) = \int_0^{h-\tau} e^{As}Bds = I(h-\tau)$$
$$\Gamma_1(\tau) = \int_{h-\tau}^{h} e^{As}Bds = I\tau$$

This, together with equation 10 and the system in equation 11 gives,

$$x(kh+\tau) = x(kh) + \begin{bmatrix} -IK(h-\tau) & I\tau \\ -K & 0 \end{bmatrix} \begin{bmatrix} x(kh) \\ u(kh+\tau) \end{bmatrix} \tag{12}$$

3.1.1

Considering the systems denominator of the transfer function on the form,

$$d(z) = z^2 + a_1 z + a_2, \tag{13}$$

the system is BIBO stable if the poles,

$$p_1, p_2 = -\frac{a_2}{2} \pm \sqrt{\frac{a_1^2 - 4a_2}{4}}. \tag{14}$$

This gives us that

$$a_1 = -(p_1 + p_2) \tag{15}$$

$$a_2 = p_1 p_2. \tag{16}$$

For the system to be stable, the poles needs to lie inside the unit circle, meaning that,

$$p_1, p_2 < 1. \tag{17}$$

With equation 15 and 16 we can now form the criteria for stabillity,

$$|a_2| = |p_1 p_2| = |p_1||p_2| < 1 \tag{18}$$

and for a_1

$$|a_1| < 1 + a_2 \tag{19}$$

Calculating the eigenvalues of 11 will give the poles of the system, which together with equation 16 and 15 give,

$$a_2 = KI\tau \tag{20}$$

and

$$a_1 = IKh - IK\tau - 1 \tag{21}$$

Combining equation 21 and 20 with 18 and 19 and solving for τ/h will give us that the relation,

$$max\left\{\frac{1}{2} - \frac{1}{KIh}, 0\right\} < \frac{\tau}{h} < min\left\{\frac{1}{KIh}, 1\right\}. \tag{22}$$

must be satisfied for stability.

3.2

The system will become unstable when a delay of 0.4 seconds is used. The settling time for 0.39 is quite long but the system does stabilize. Figure 17 shows different time delay settings in Simulink. It clearly shows that a delay of 0.3 seconds still makes the system stabilize quickly while a delay of 0.4 makes it unstable.

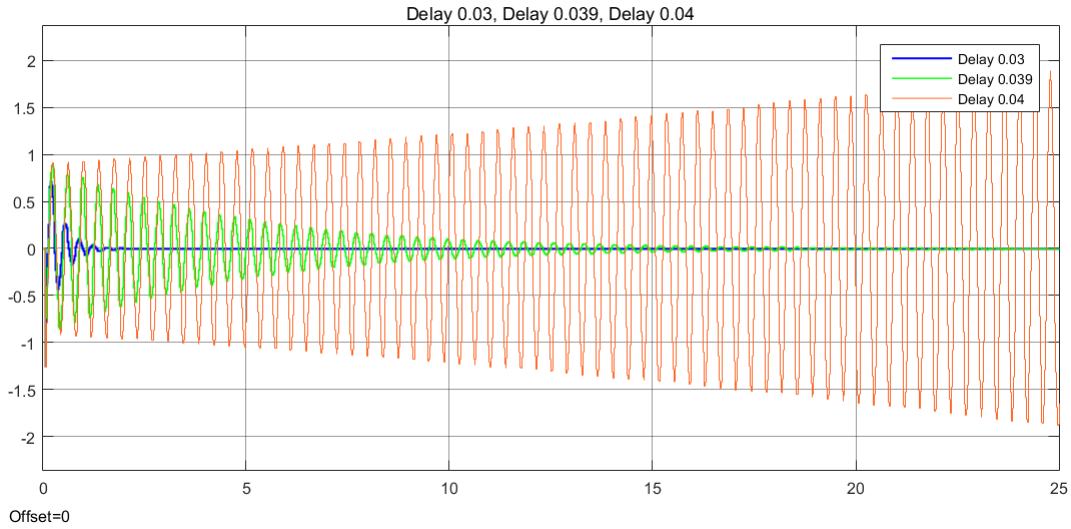

Figure 17: Control signal for different delay settings

4 Discrete Event System

4.1

Since the state machines M1 and M2 have the same states they are represented as the same state machine. This can be seen in Figure 17.

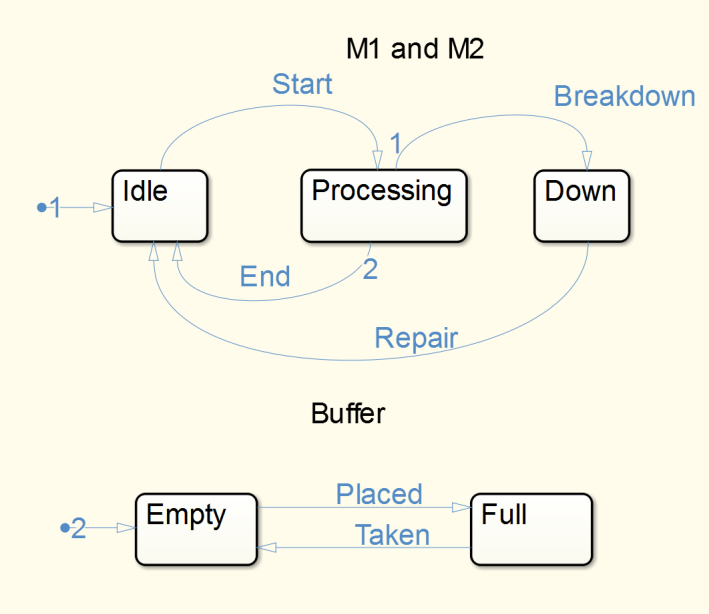

Figure 18: State machine for M1, M2 and the buffer

4.2

Each state in Figure 18 represents the current state in M1, M2 and the buffer. There are 18 different states available

$$3^2 * 2 = 18 \tag{23}$$

since both machines has 3 states and the buffer has 2. The form x, y, z is used where $x = M1, y = Buffer, z = M2$. For example

$$IEI = Idle, Empty, Idle$$

The events of each transition are not shown in Figure 18 due to too many transitions and the events are obvious.

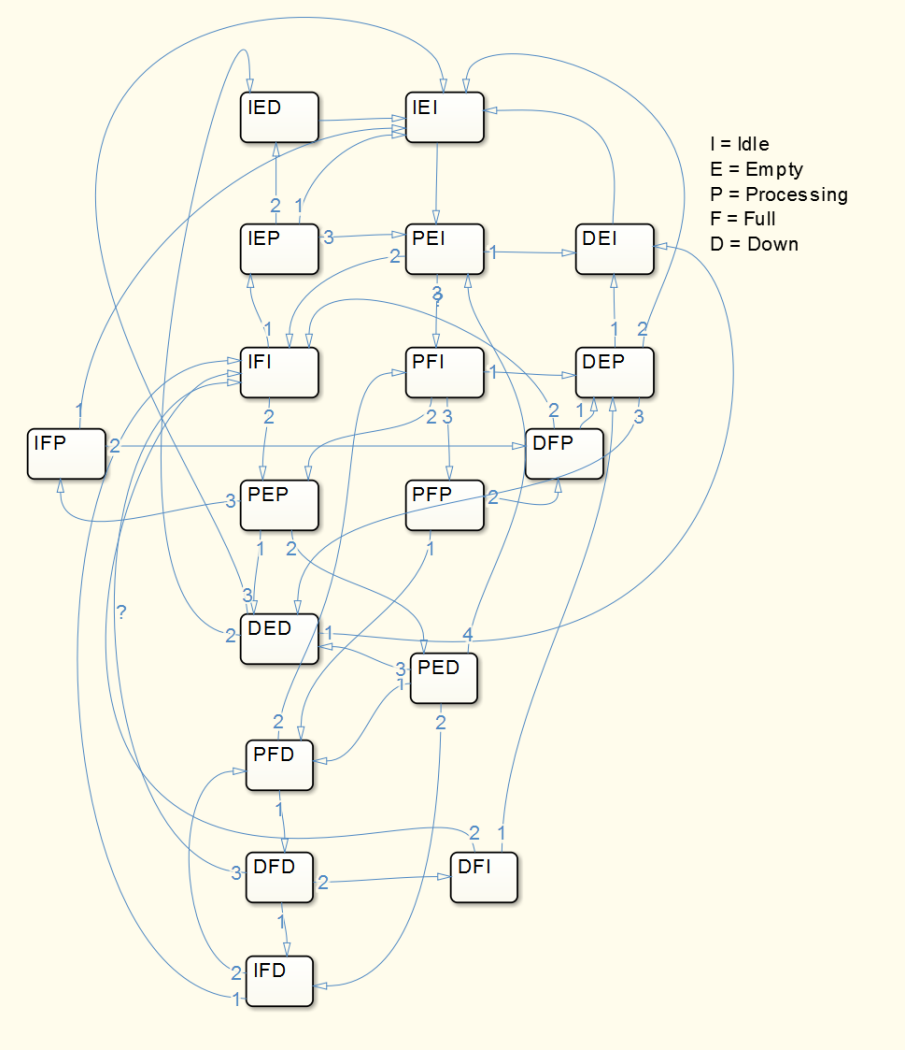

Figure 19: State machine for M1, M2 and the buffer

4.3

With the requirements given some states could be disregarded due to our interpretation of the functionality of the state machine. We assume that when a process has finished a product and put it in the buffer it goes to idle due to a new product can't begin processing if the buffer is full even though it still could be in the processing state.

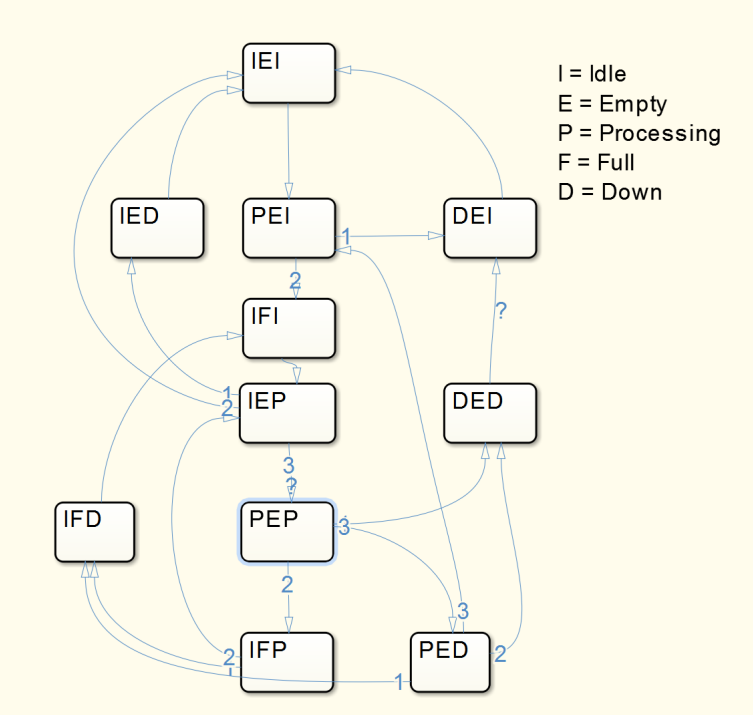

Figure 20: State machine for M1, M2 and the buffer

4.4

A controller to make sure the requirements are fulfilled when using the complete Discrete Event System from 4.2 needs to have information of which state each machine is in. Then by using that information *Guards* can be used on each transition along with the event to ensure it goes to the right state. A *Guard* is a form of If-statement used in state machines to set requirements for transitions.