VT16 – EL2450 – Assignment II

Alexandros Filotheou
871108-5590
alefil@kth.se

Roberto Sanchez-Rey
840616-9139
rosr@kth.se

1 Part One: Scheduling

Throughout this part, the pendulum with length $l_1 = 0.1$ m will be referred to as P_1, $l_2 = 0.2$ m as P_2 and $l_3 = 0.3$ m as P_3. Plots in blue will refer to P_1, in red to P_2 and in orange to P_3.

1.1 Rate Monotonic scheduling

1.1.1 Question 1

Rate Monotonic is an scheduling method that assigns fixed priorities to tasks, proportional to its activation frequency. That means that for any given tasks J_a, J_b with periods $T_a < T_b$, J_a is assigned a higher priority than J_b.

1.1.2 Question 2

A set of periodic tasks $\{J_i\}$ is schedulable with Rate Monotonic scheduling if

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

In the case where $T_1 = 20, T_2 = 29, T_3 = 35$ ms and $C_i = 6$ ms, $i = \{1, 2, 3\}$, $U = 0.678$ and $n(2^{1/n} - 1) = 0.78$. Hence tasks J_1, J_2, J_3 are schedulable with RM. A portion of the calculated schedule can be found in figure 3.

1.1.3 Question 3

All pendula are stable. We observe that the higher the natural frequency of a pentulum, the quicker the response is in its rise time, although with magnified overshoot. This makes sense since the higher the natural frequency of a pendulum, the lower its length and the more difficult it is to stabilize, hence the control must be swift. Figure 1 shows the angular displacement of each pendulum.

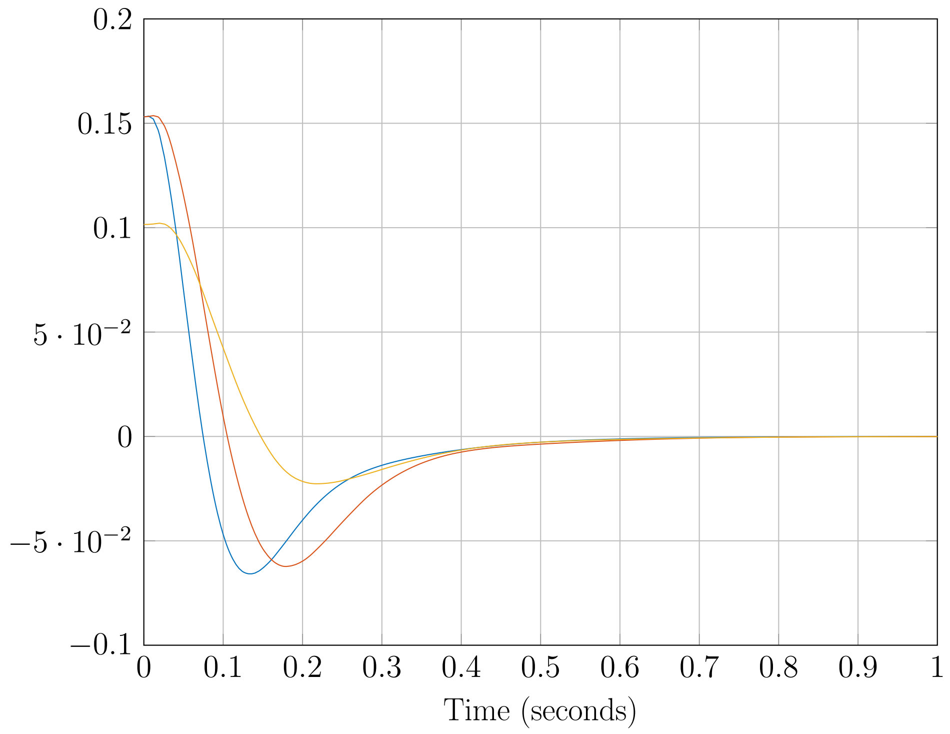

Figure 1: The angular displacement of each pendulum as a function of time. `Blue`: P_1, `Red`: P_2, `Orange`: P_3

1.1.4 Question 4

Figure 2 shows the executed schedule magnified over the period of the first 60 ms. As per the response to question 2, figure 4 illustrates that the schedule is indeed feasible by plotting the overall usage of the CPU over the aforementioned timespan.

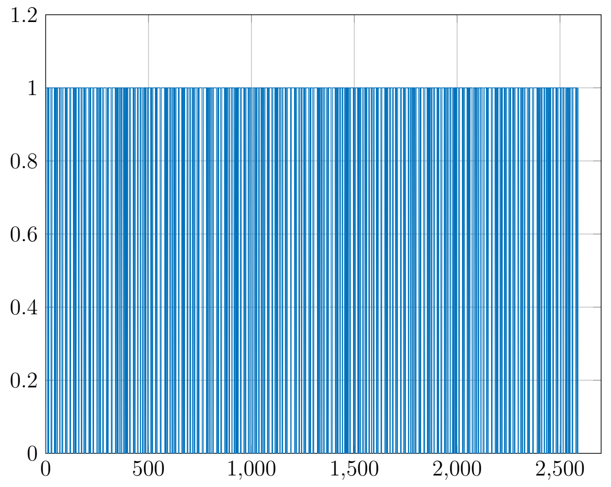

Figure 4: The overall processing usage. Notice that it is at most at 100%. $C_i = 6$ ms.

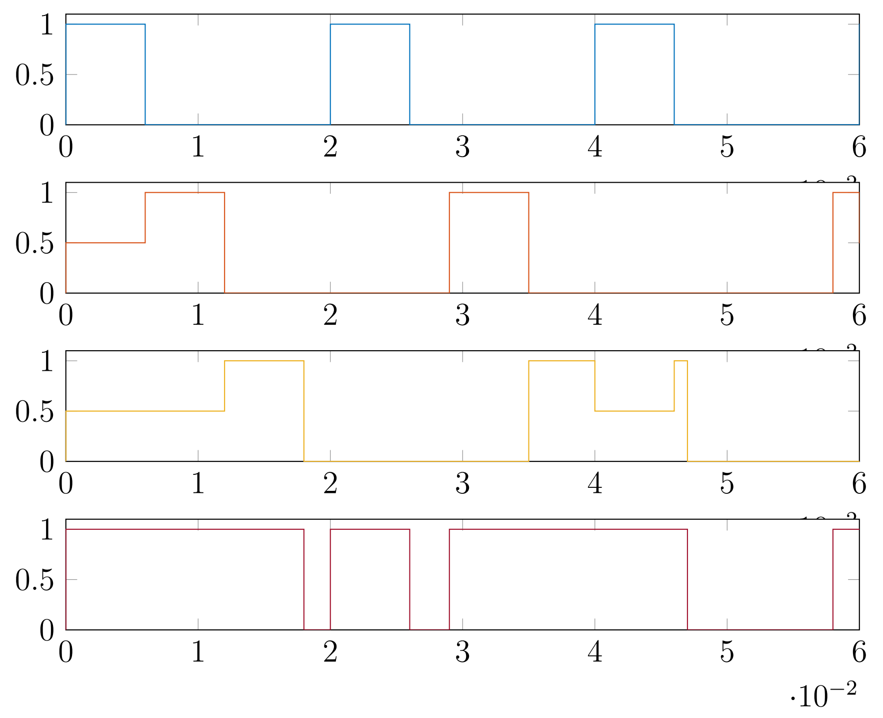

Figure 2: The executed schedule for the three pendula restricted to the first 60ms. `Blue`: P_1, `Red`: P_2, `Orange`: P_3. $C_i = 6$ ms. The last figure shows the overall processor usage for verification purposes.

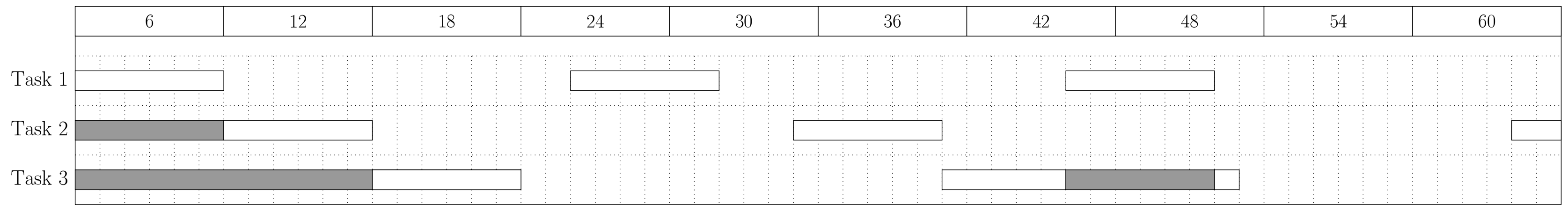

Figure 3: A portion of the calculated RM schedule σ for tasks J_1, J_2, J_3. Shaded areas denote the waiting time.

1.1.5 Question 5

In the case where $T_1 = 20, T_2 = 29, T_3 = 35$ ms and $C_i = 10$ ms, $i = \{1, 2, 3\}$, $U = 1.131 > 1$. Hence tasks J_1, J_2, J_3 are not schedulable under any scheduling scheme. A portion of the calculated schedule can be found in figure 10.

Pendulum P_3 is left to its own devices and tends slowly but surely towards instability. This makes sense since no control signal pertaining to P_3 is assigned execution time in the processor. Figure 6 shows the angular displacement of pendulum P_3 as a function of time.

For pendula P_1 and P_2, however, due to the increased execution time, delays are introduced, and the control input is not as swift as before, hence the increased magnitude of the overshoot and the larger rise and settling times. Figure 5 shows the angular displacement of each stable pendulum as a function of time.

Figures 7 and 8 show the angular displacement of pendula P_1 and P_2 respectively, as a function of time, for the two different cases of execution time.

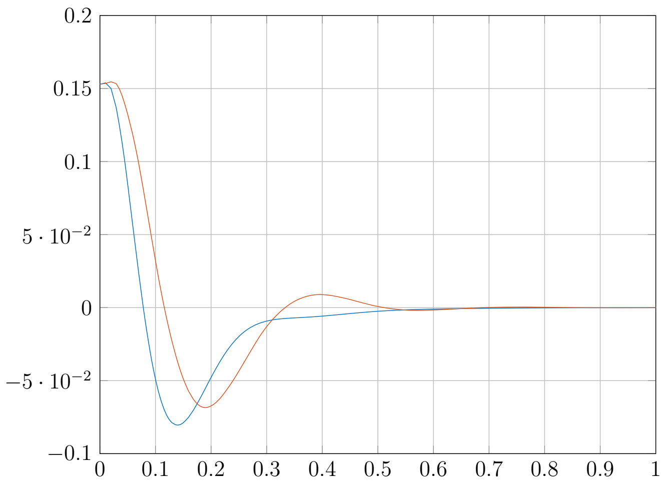

Figure 5: The angular displacement of the stable pendula P_1 and P_2 as a function of time. `Blue`: P_1, `Red`: P_2. $C_i = 10$ ms.

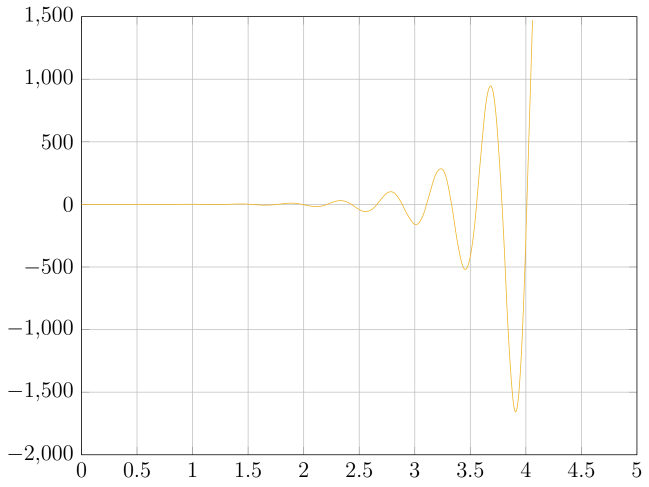

Figure 6: The angular displacement of the unstable pendulum P_3 as a function of time. $C_i = 10$ ms.

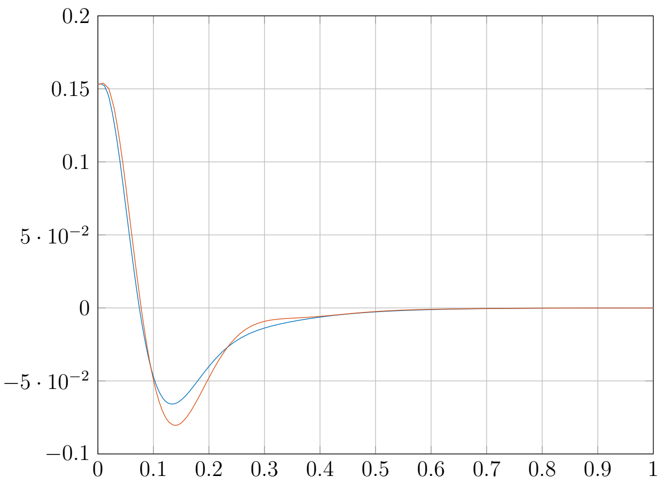

Figure 7: The angular displacement of pendulum P_1 as a function of time. `Blue`: $C_1 = 6$ ms, `Red`: $C_1 = 10$ ms

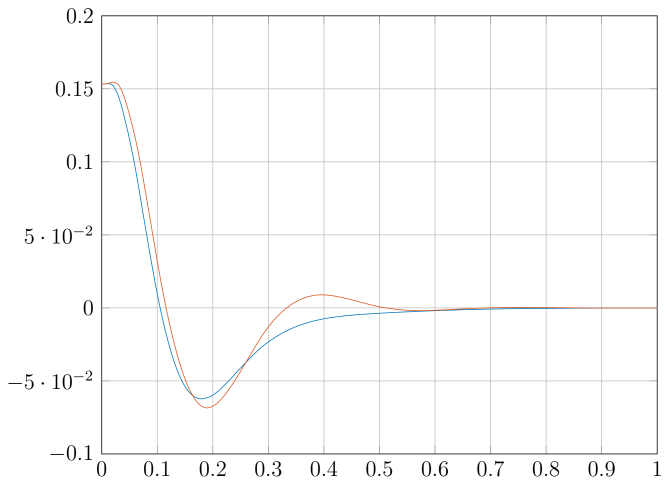

Figure 8: The angular displacement of pendulum P_2 as a function of time. `Blue`: $C_2 = 6$ ms, `Red`: $C_2 = 10$ ms

Figure 9 shows the former schedule magnified over the period of the first 60 ms. Figure 11 illustrates that the schedule is not feasible by ploting the overall usage of the CPU over the length of a schedule period, which is at all times 100%, indicative of the excessive processing load demanded.

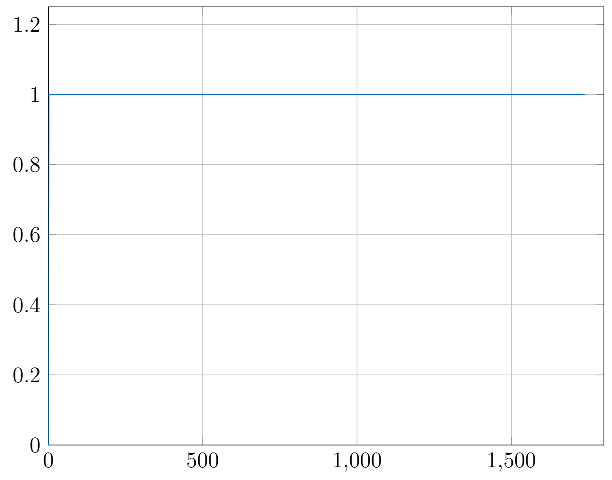

Figure 11: The overall processing usage. Notice that it is always at 100%. $C_i = 10$ ms.

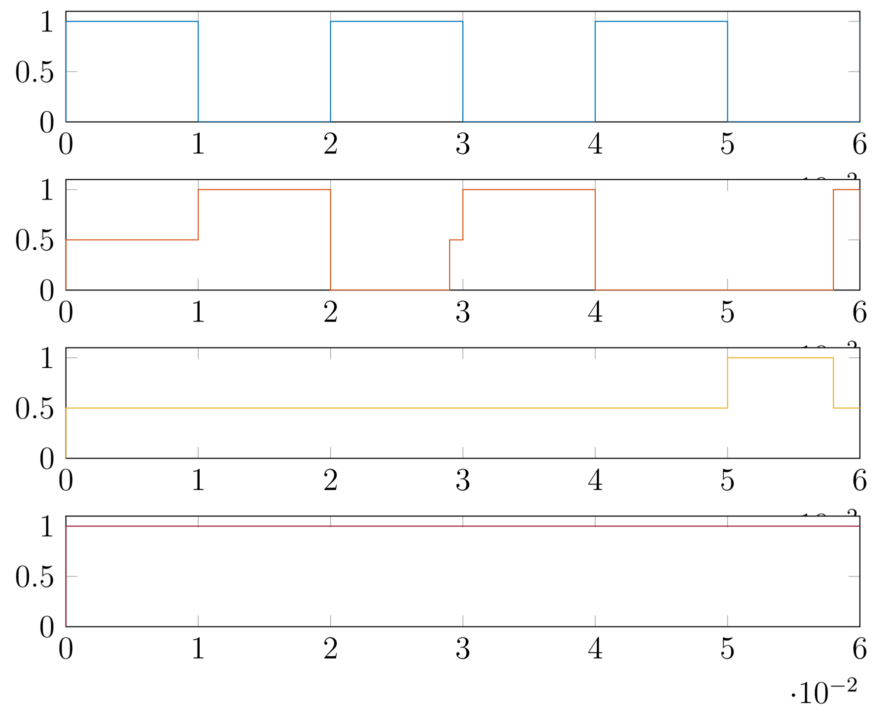

Figure 9: The executed schedule for the three pendula restricted to the first 60 ms. `Blue`: P_1, `Red`: P_2, `Orange`: P_3. $C_i = 10$ ms. The last figure shows the overall processor usage for verification purposes.

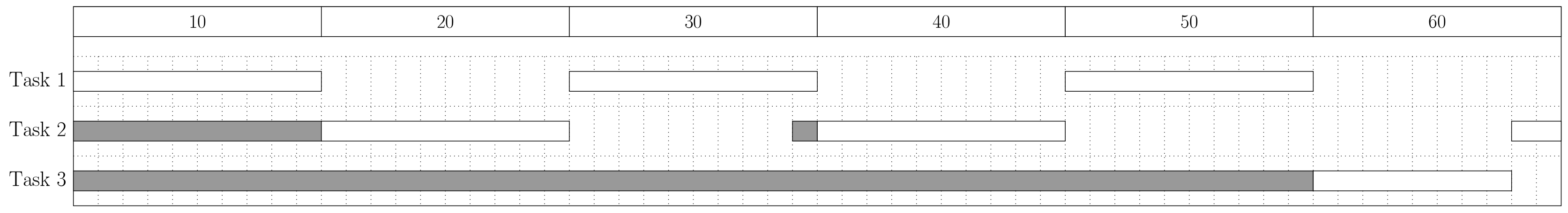

Figure 10: A portion of the calculated RM schedule σ for tasks J_1, J_2, J_3 for $C_i = 10$ ms. Shaded areas denote the waiting time. Notice that J_3 misses its deadlines consecutively, indicative of the inability of schedulability.

1.2 Earliest Deadline First scheduling

1.2.1 Question 1

In contrast to Rate Monitoring scheduling, Earliest Deadline First scheduling assigns dynamic priorities to tasks, introducing a degree of flexibility. The task whose deadline is closest to the current timestep is given the highest priority and is executed for one time unit.

EDF is more lenient than RM, and this can be seen in the condition that identifies it:

$$U \leq 1 \Leftrightarrow \sigma \text{ is feasible}$$

whereas in RM

$$U \leq n(2^{1/n} - 1) \ (< 1 \text{ for } n \geq 2) \Rightarrow \sigma \text{ is feasible}$$

This means that if $U \leq 1$ for a certain collection of tasks $\{J_i\}$, there is always a feasible schedule for $\{J_i\}$, where the processor is utilized to the fullest it can be, whereas the former certainty does not hold with RM. However, if certain tasks are indeed more important than others and there are extra requirements per their execution, it is possible that EDF introduces delays between their release and start times due to the indiscrimination it shows to absolute task priorities.

1.2.2 Question 2

As stated above, a set of periodic tasks $\{J_i\}$ is schedulable under Earliest Deadline First scheduling if and only if

$$U = \sum_i \frac{C_i}{T_i} \leq 1$$

In the case where $T_1 = 20, T_2 = 29, T_3 = 35$ ms and $C_i = 6$ ms, $i = \{1, 2, 3\}$, $U = 0.678 \leq 1$. Hence tasks J_1, J_2, J_3 are schedulable with EDF. A portion of the calculated schedule can be found in figure 14.

1.2.3 Question 3

All pendula are stable for execution time $C_i = 6$ ms. We observe that the higher the natural frequency of a pentulum, the quicker the response is in its rise time, although with magnified overshoot. This makes sense since the higher the natural frequency of a pendulum, the lower its length and the more difficult it is to stabilize, hence the control must be swift. Figure 12 shows the angular displacement of each pendulum.

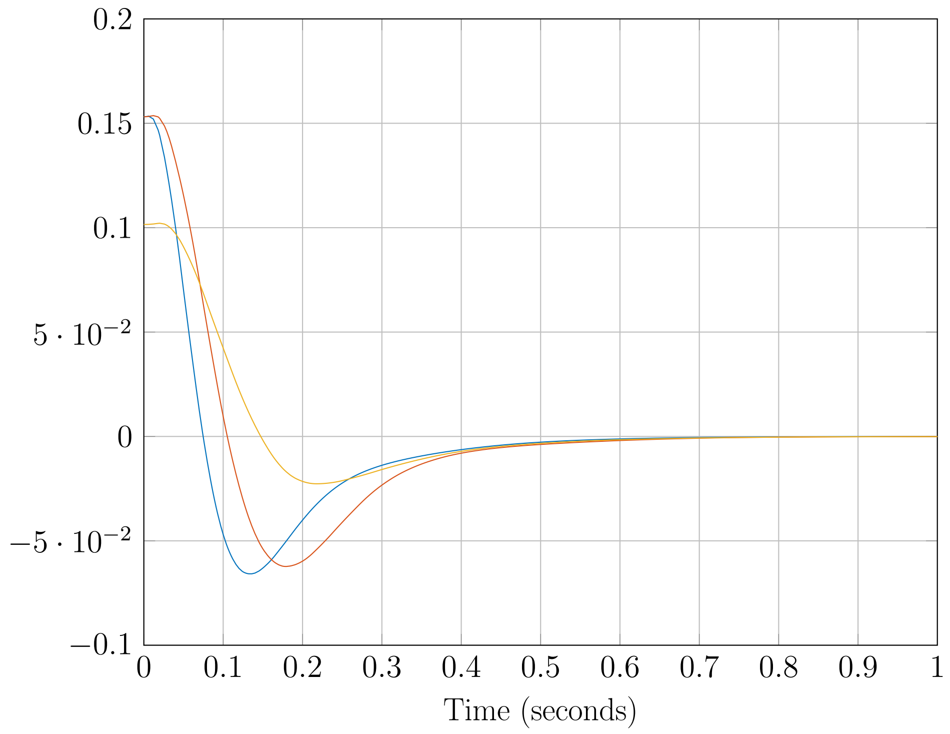

Figure 12: The angular displacement of each pendulum as a function of time. `Blue`: P_1, `Red`: P_2, `Orange`: P_3

1.2.4 Question 4

Figure 13 shows the former schedule magnified over the period of the first 60 ms. As per the response to question 2, figure 15 illustrates that the schedule is indeed feasible by plotting the overall usage of the CPU over the aforementioned timespan.

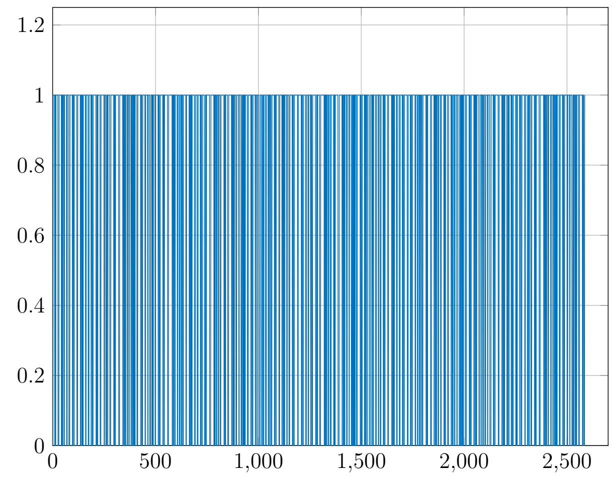

Figure 15: The overall processing usage. Notice that it is at most at 100%. $C_i = 6$ ms.

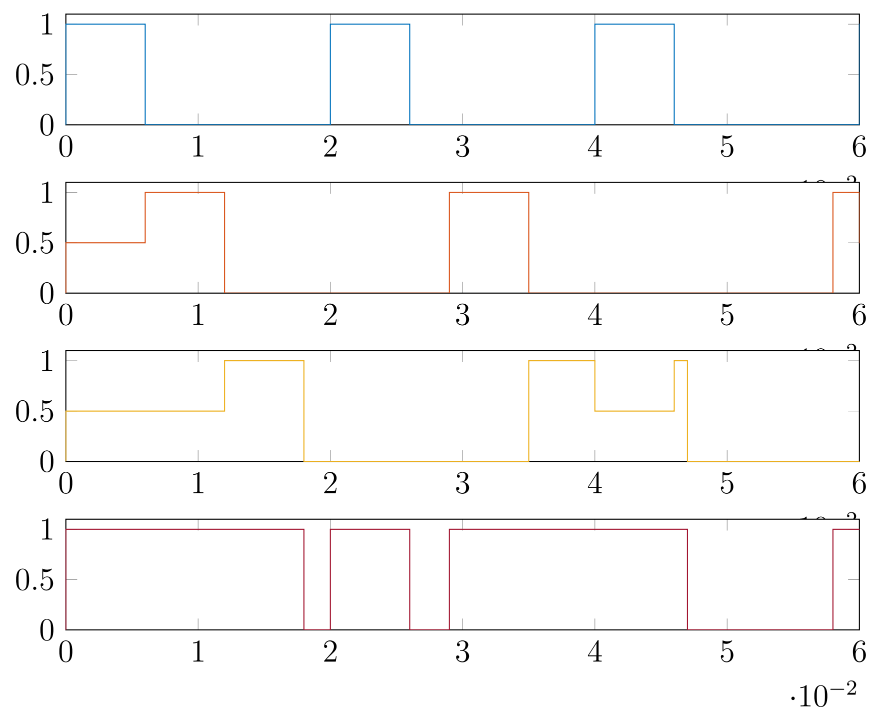

Figure 13: The executed schedule for the three pendula restricted to the first 60 ms. `Blue`: P_1, `Red`: P_2, `Orange`: P_3. $C_i = 6$ ms. The last figure shows the overall processor usage for verification purposes.

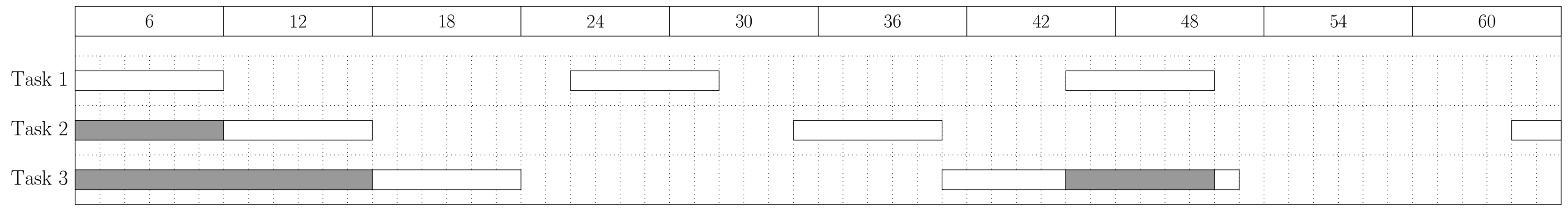

Figure 14: A portion of the calculated EDF schedule σ for tasks J_1, J_2, J_3. Shaded areas denote the waiting time.

1.2.5 Question 5

In the case where $T_1 = 20, T_2 = 29, T_3 = 35$ ms and $C_i = 10$ ms, $i = \{1, 2, 3\}$, $U = 1.131 > 1$. Hence tasks J_1, J_2, J_3 are not schedulable under any scheduling scheme. A portion of the calculated schedule can be found in figure 14.

Here we note that the lack of feasibility of a schedule is connected to the notion and fact of deadlines not being met. This, however, does not mean that there can be no control over a system. All pendula are stable as opposed to the case of RM scheduling with the same execution time. This is because all control tasks can be preemped, hence every one of them is allowed execution in the processor, which results to all of them being executed. Hence, despite the fact that deadlines are missed, it is possible to control all three pendula, although it may be with worse control performance than in the case where $C_i = 6$ ms.

Due to the missing of deadlines, and, ultimately, due to the increased execution time, the control input is not as swift as before, hence the increased magnitude of the overshoot and the larger rise and settling times. Figure 16 shows the angular displacement of each pendulum as a function of time. Figures 17, 18 and 19 show the angular displacement of pendula P_1, P_2 and P_3 respectively, as a function of time, for the two different cases of execution time.

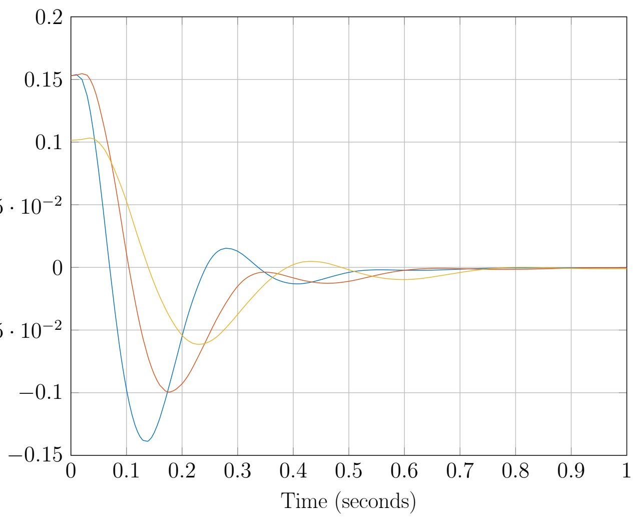

Figure 16: The angular displacement of each pendulum as a function of time. `Blue`: P_1, `Red`: P_2, `Orange`: P_3. $C_i = 10$ ms.

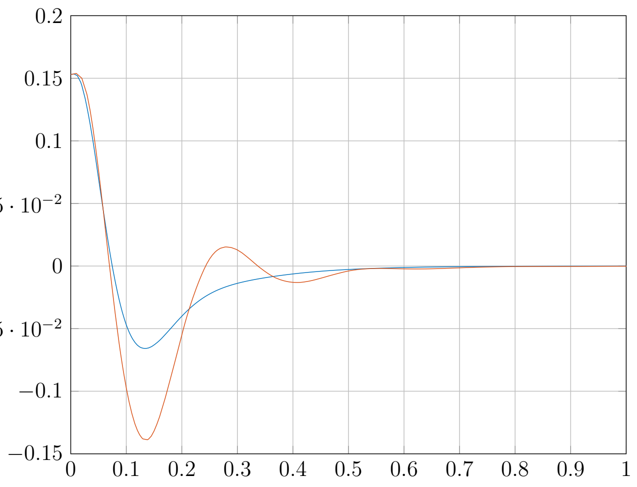

Figure 17: The angular displacement of pendulum P_1 as a function of time. `Blue`: $C_1 = 6$ ms, `Red`: $C_1 = 10$ ms

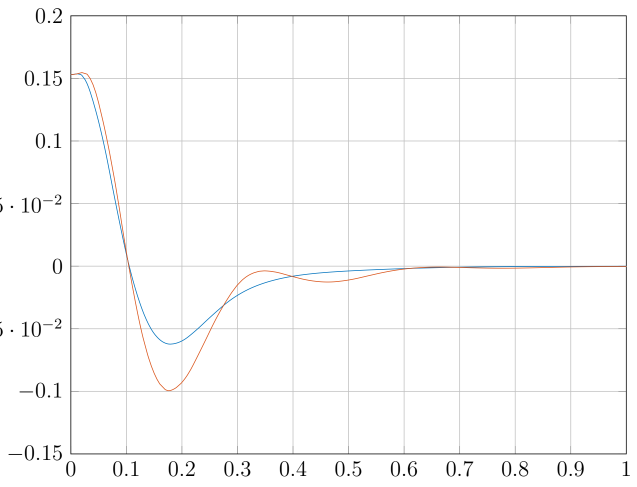

Figure 18: The angular displacement of pendulum P_2 as a function of time. `Blue`: $C_2 = 6$ ms, `Red`: $C_2 = 10$ ms

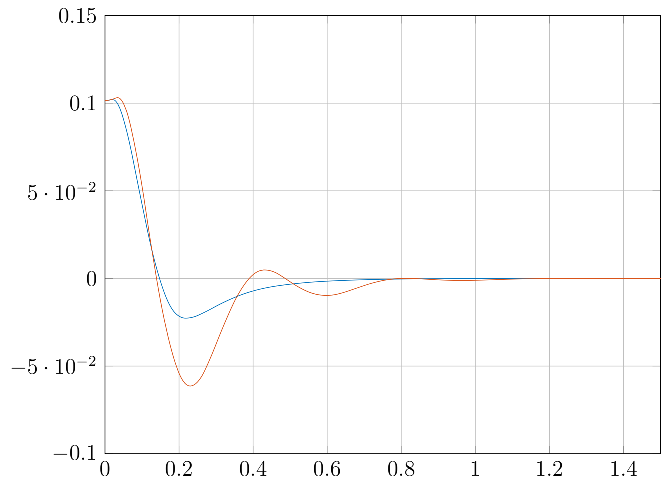

Figure 19: The angular displacement of pendulum P_3 as a function of time. `Blue`: $C_3 = 6$ ms, `Red`: $C_3 = 10$ ms

Figure 20 shows the former schedule magnified over the period of the first 120 ms. Figure 22 illustrates that the schedule is not feasible by ploting the overall usage of the CPU over the length of a schedule period, which is at all times 100%, indicative of the excessive processing load demanded.

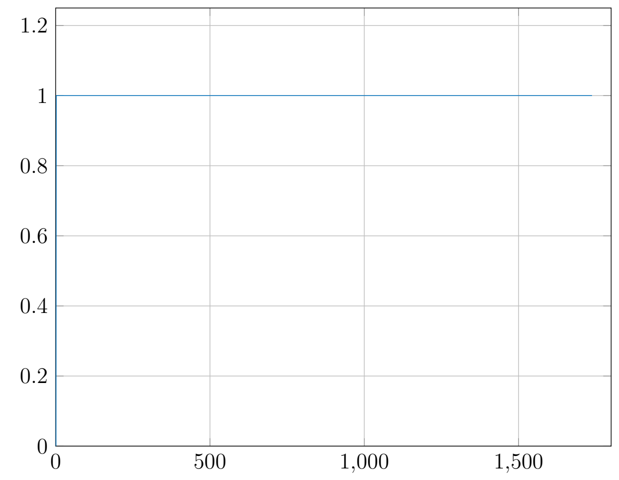

Figure 22: The overall processing usage. Notice that it is always at 100%. $C_i = 10$ ms.

1.2.6 Question 6

Figures 23, 25, 27, 29, and 31 feature the comparison of the course of the angular displacement of every stable pendulum, between the two considered scheduling policies. Figures 24, 26, 28, 30 and 32 graphically illustrate the evolution of the difference in angular displacement between the two considered scheduling policies for the two considered execution times for all stable pendula.

With execution time $C_i = 6$ ms the behaviour of all three pendula is nearly identical with respect to the two different scheduling policies. Scheduling under EDF makes the system have a

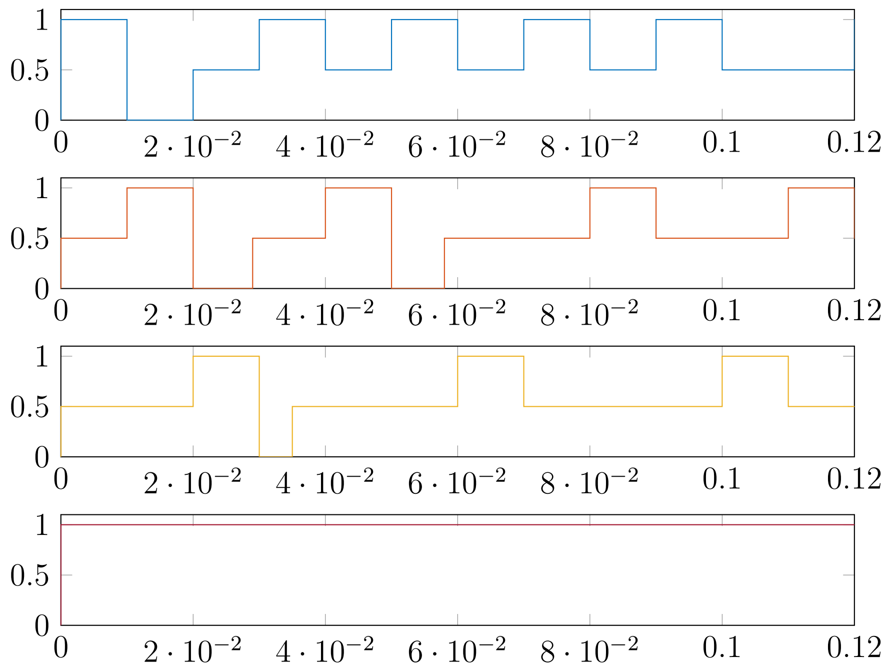

Figure 20: The executed schedule for the three pendula restricted to the first 120 ms. Notice how J_3 misses its deadlines. `Blue`: P_1, `Red`: P_2, `Orange`: P_3. $C_i = 10$ ms. The last figure shows the overall processor usage for verification purposes.

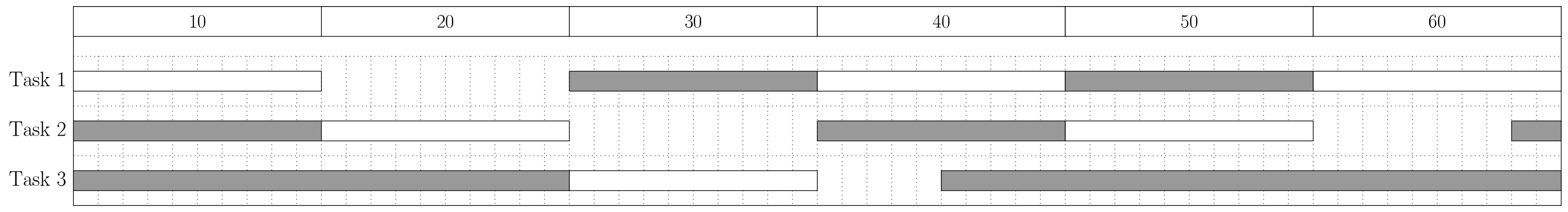

Figure 21: A portion of the calculated EDF schedule σ for tasks J_1, J_2, J_3 for $C_i = 10$ ms. Shaded areas denote the waiting time. Notice that J_3 misses its deadlines consecutively (as is apparent in the interval $[100, 110]$ ms), indicative of the inability of schedulability.

higher overshoot but its peak difference from that of RM's is by an order of magnitude of 10^{-3} degrees. Hence their difference is negligible.

With execution time $C_i = 10$ ms there are two things to consider. First that pendulum P_3 is not controllable under RM but it is under EDF. Hence, if the purpose at hand is to control all three pendula, EDF is conclusively the only choice there is. On the other hand, though, EDF makes the system behave oscillatory in a degree larger than RM, with higher overshoot, rise and settling times.

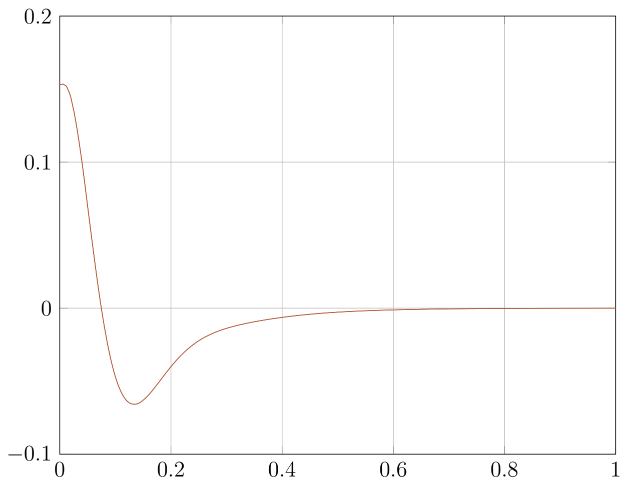

Figure 23: Evolution of the angular displacement of pendulum P_1 for $C_1 = 6$ ms. **Blue**: RM scheduling, **Red**: EDF scheduling

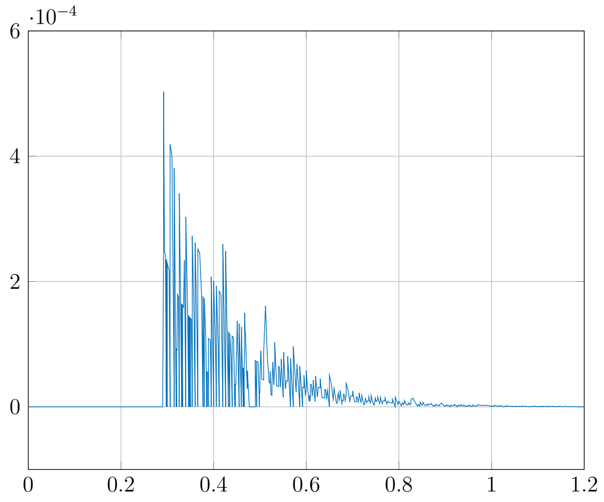

Figure 24: The evolution of the difference in angular displacement between RM and EDF of pendulum P_1 for execution time $C_1 = 6$ ms.

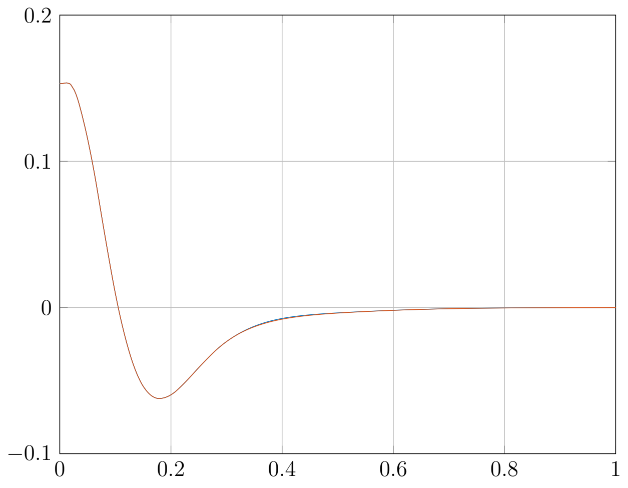

Figure 25: Evolution of the angular displacement of pendulum P_2. $C_2 = 6$ ms. **Blue**: RM scheduling, **Red**: EDF scheduling

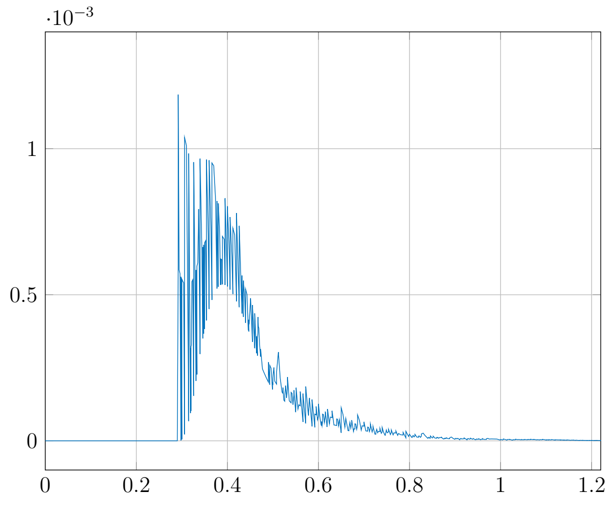

Figure 26: The evolution of the difference in angular displacement between RM and EDF of pendulum P_2 for execution time $C_2 = 6$ ms.

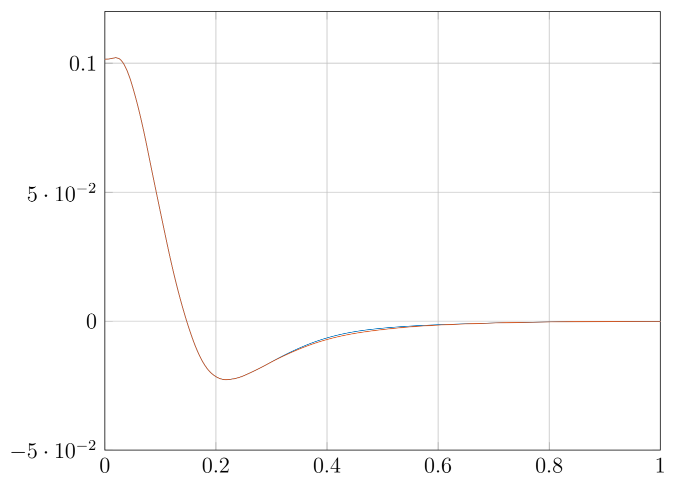

Figure 27: Evolution of the angular displacement of pendulum P_3. $C_3 = 6$ ms. `Blue`: RM scheduling, `Red`: EDF scheduling

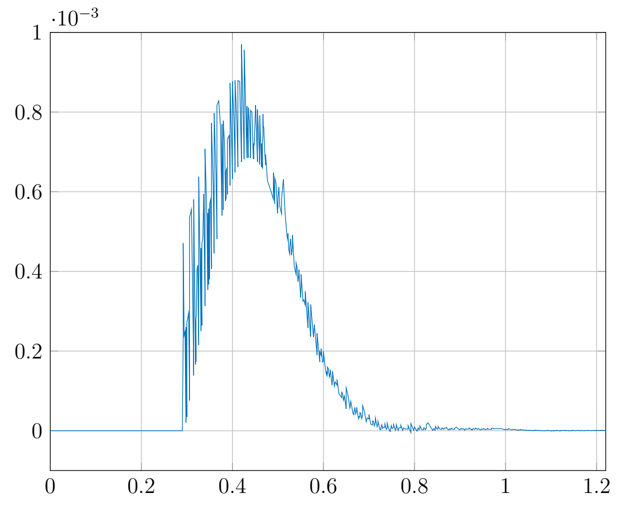

Figure 28: The evolution of the difference in angular displacement between RM and EDF of pendulum P_3 for execution time $C_3 = 6$ ms.

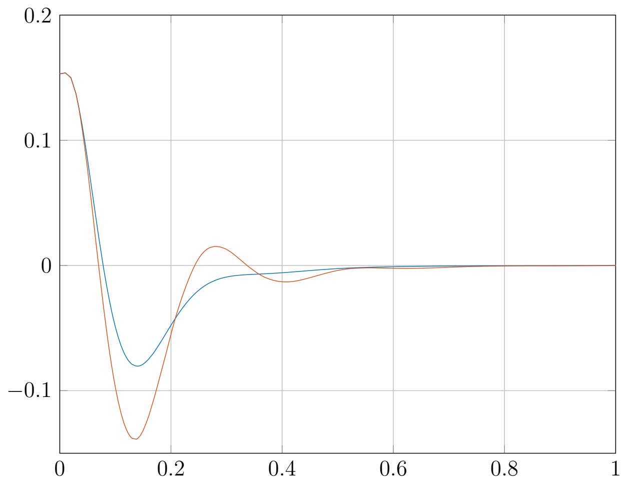

Figure 29: Evolution of the angular displacement of pendulum P_1 for $C_1 = 10$ ms. `Blue`: RM scheduling, `Red`: EDF scheduling

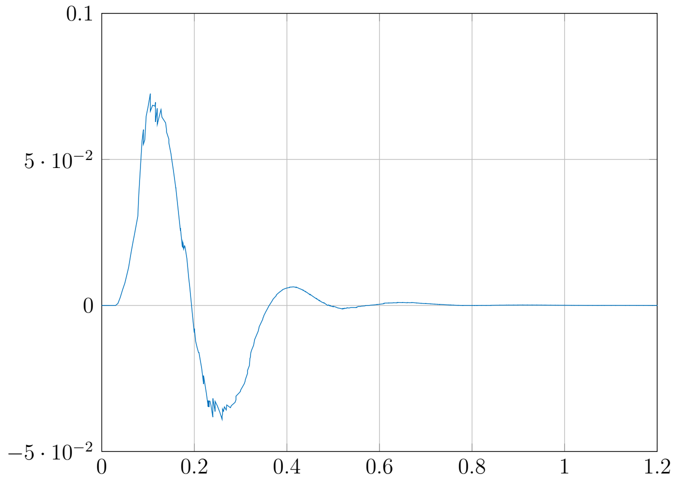

Figure 30: The evolution of the difference in angular displacement between RM and EDF of pendulum P_1 for execution time $C_1 = 10$ ms.

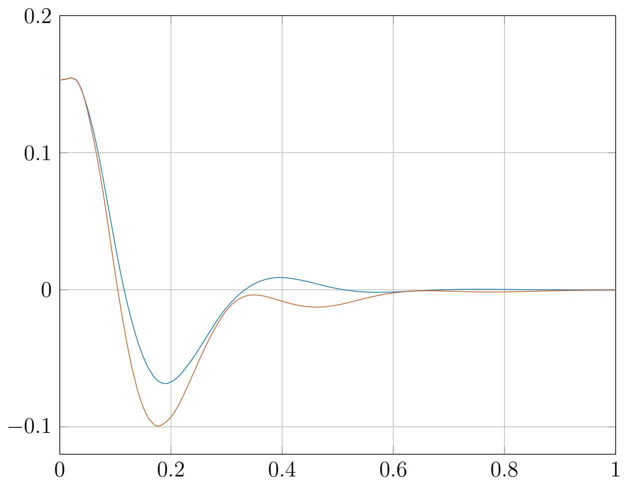

Figure 31: Evolution of the angular displacement of pendulum P_2. $C_2 = 10$ ms. `Blue`: RM scheduling, `Red`: EDF scheduling

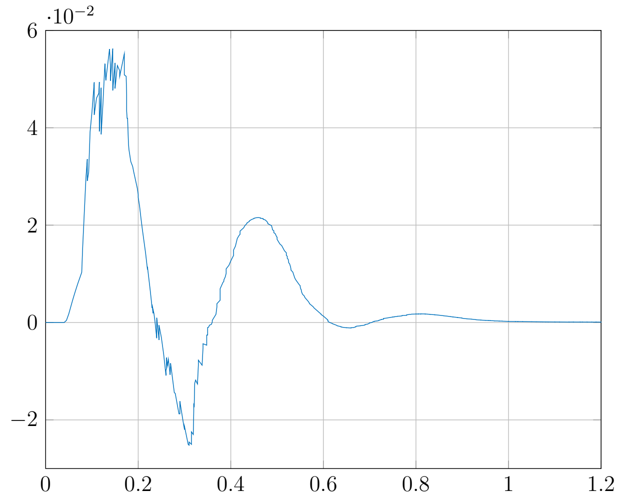

Figure 32: The evolution of the difference in angular displacement between RM and EDF of pendulum P_2 for execution time $C_2 = 10$ ms.

2 Part Two: Networked Control Systems

2.1 Question 1

From the continuous plant

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t)\\ y(t) &= Cx(t)\end{aligned}$$

and the facts that $A = 0, B = 1, \tau \in [0, h]$ we can derive the equation for the discretized state

$$\begin{aligned}x[(k+1)h] &= \Phi x[kh] + \Gamma_0 u[kh] + \Gamma_1 u[(k-1)h]\\ x[(k+1)h] &= x[kh] + (h-\tau)u[kh] + \tau u[(k-1)h]\end{aligned}$$

where

$$\begin{aligned}\Phi &= e^{Ah} = 1\\ \Gamma_0 &= \int_0^{h-\tau} e^{As} ds B = h - \tau\\ \Gamma_1 &= e^{A(h-t)} \int_0^{\tau} e^{As} ds B = \tau\end{aligned}$$

Due to the additional delayed input term $u[(k-1)h]$, the system's state is augmented and is now considered as the vector

$$s = \begin{bmatrix} x[kh] \\ u[(k-1)h] \end{bmatrix}$$

Considering the fact that $u[kh] = -Kx[kh]$, the equations of the closed-loop system thus become

$$\begin{bmatrix} x[(k+1)h] \\ u[kh] \end{bmatrix} = \begin{bmatrix} \Phi - \Gamma_0 K & \Gamma_1 \\ -K & 0 \end{bmatrix} \begin{bmatrix} x[kh] \\ u[(k-1)h] \end{bmatrix}$$

or

$$\begin{bmatrix} x[(k+1)h] \\ u[kh] \end{bmatrix} = \begin{bmatrix} 1 - (h-\tau)K & \tau \\ -K & 0 \end{bmatrix} \begin{bmatrix} x[kh] \\ u[(k-1)h] \end{bmatrix}$$

2.2 Question 2

In order for the system to be stable, the eigenvalues of matrix

$$\begin{bmatrix} 1-(h-\tau)K & \tau \\ -K & 0 \end{bmatrix}$$

should lie inside the unit circle.
These eigenvalues λ_i can be found from

$$\begin{aligned} \left| \lambda I - \begin{bmatrix} 1-(h-\tau)K & \tau \\ -K & 0 \end{bmatrix} \right| &= 0 \\ \left| \begin{bmatrix} \lambda - 1 + (h-\tau)K & -\tau \\ K & \lambda \end{bmatrix} \right| &= 0 \\ \lambda^2 + \lambda(K(h-\tau)-1) + K\tau &= 0 \end{aligned}$$

Where $\prod \lambda_i = K\tau$ and $\sum \lambda_i = -(K(h-\tau)-1)$.
The stability triangle dictates that the system is stable if the following inequalities hold:

$$\left.\begin{aligned} &K\tau < 1 \\ &K\tau < K(h-\tau) - 1 - 1 \\ &K\tau > -K(h-\tau) + 1 - 1 \end{aligned}\right\} \Leftrightarrow$$

$$\begin{aligned} \frac{Kh-2}{2} < K\tau < 1 \Leftrightarrow \\ \frac{1}{2} - \frac{1}{Kh} < \frac{\tau}{h} < \frac{1}{Kh} \end{aligned}$$

And, since $K > 0$ and $0 \leq \tau \leq h$:

$$\max\left(0, \frac{1}{2} - \frac{1}{Kh}\right) < \frac{\tau}{h} < \min\left(1, \frac{1}{Kh}\right)$$

The stability region is illustrated in figure 33. The system is stable if the combination of the Kh and τ/h quantities lies inside the area bounded by the curves and the vertical axis.

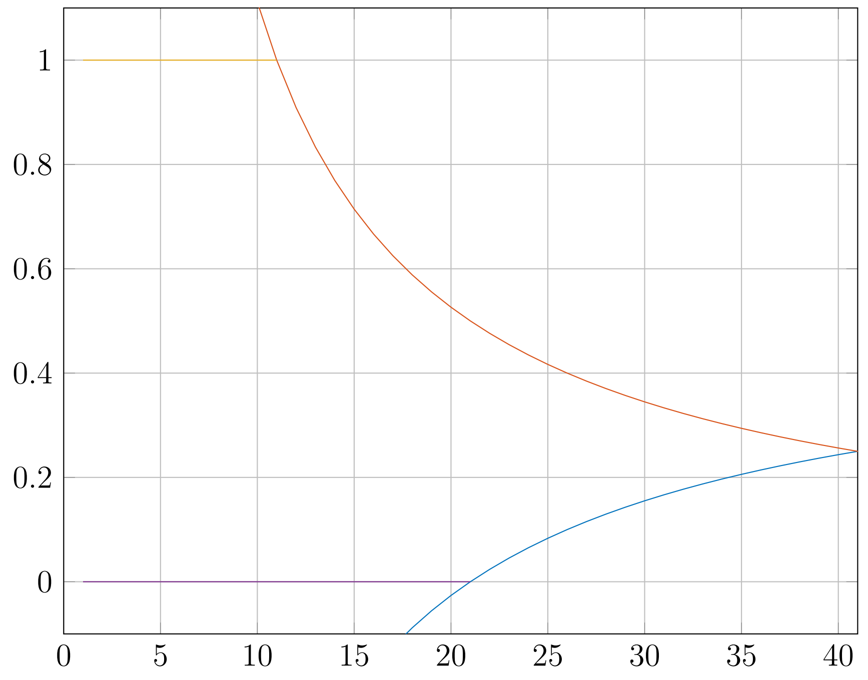

Figure 33: The stability region is defined here as the area in the x, y plane that is bounded by the curves: $x = 0$, $y = 1$, $y = 0$, $y = 0.5 - 1/x$ and $y = 1/x$. The x axis represents the product Kh. The y axis represents the magnitude of the fraction τ/h.

2.3 Question 3

The minimum value for the communication delay that causes the system to be unstable is approximately $\tau = 0.04$ sec. Figures 34 and 35 show the response of the system for $\tau = 0.039$ and $\tau = 0.04$ sec respectively.

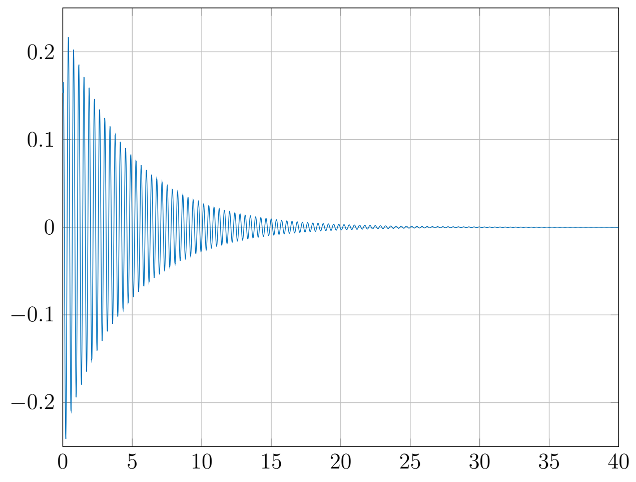

Figure 34: Angular displacement of a pendulum with a communication delay of $\tau = 0.039$ sec.

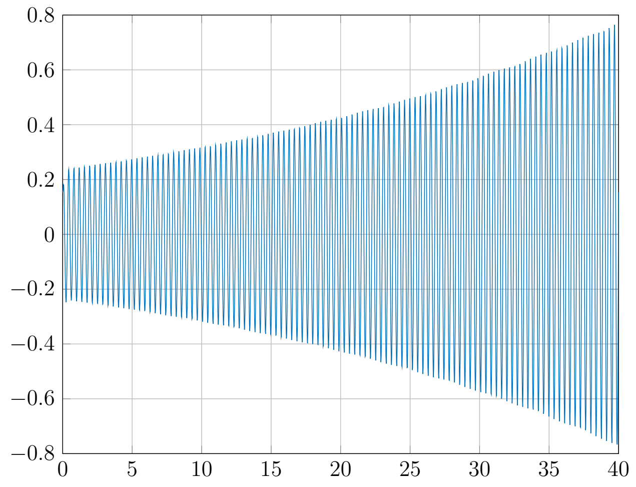

Figure 35: Angular displacement of a pendulum with a communication delay of $\tau = 0.040$ sec.

3 Part Three: Discrete Event System

3.1 Question 1

The discrete event systems that model a machine M and a buffer B are illustrated in figures 36 and 37 respectively.

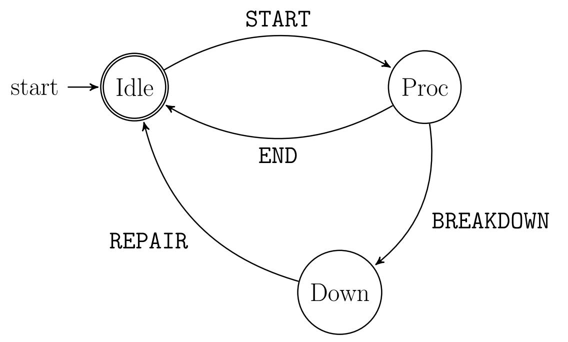

Figure 36: The DES that models a machines of type M.

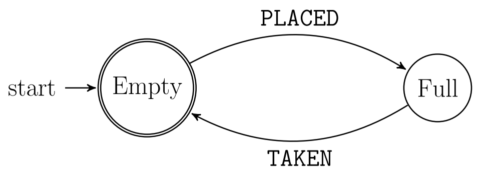

Figure 37: The DES that models buffer B.

4 Question 2

The set of states of the discrete event system that models the complete manufacturing process Q is the set of combination of all states of the DES of machines M_1, M_2 and buffer B:

$$Q \equiv Q_{M_1} \times Q_{M_2} \times Q_B$$

State set Q is defined in table 1.

The set of events of the overall DES is the union of the sets of the three separate DES:

$$\begin{aligned} E = E_{M_1} \cup E_{M_2} \cup E_B = \\ \{\texttt{START(1), END(1), BREAKDOWN(1), END(1),} \\ \texttt{START(2), END(2), BREAKDOWN(2), END(2),} \\ \texttt{PLACED, TAKEN}\} \end{aligned}$$

where the number X inside parentheses denotes an event pertaining to machine M_X.

The initial state q_0 shall be state $q_0 = (I, I, E)$, that is, both M_1 and M_2 are idle, and buffer B is empty. Furthermore, the marked state shall be the initial state $Q_m \equiv \{q_0\}$.

Since there are 18 states, the maximum number of transitions between them is 18^2. However, it is assumed that transitions between states are caused by single events $e \in E$, i.e. two events cannot happen at the same time. This fact decreases the number of allowed transitions from 18^2 to 66.

The transition function $\delta()$ is shown in table 2. The state machine diagram is illustrated in figure 38 of the appendix. From this and the size of the number of transitions we can discern that

state notation of the complete DES	M1	M2	B
(I,I,E)	Idle	Idle	Empty
(I,I,F)	Idle	Idle	Full
(I,P,E)	Idle	Processing	Empty
(I,P,F)	Idle	Processing	Full
(I,D,E)	Idle	Down	Empty
(I,D,F)	Idle	Down	Full
(P,I,E)	Processing	Idle	Empty
(P,I,F)	Processing	Idle	Full
(P,P,E)	Processing	Processing	Empty
(P,P,F)	Processing	Processing	Full
(P,D,E)	Processing	Down	Empty
(P,D,F)	Processing	Down	Full
(D,I,E)	Down	Idle	Empty
(D,I,F)	Down	Idle	Full
(D,P,E)	Down	Processing	Empty
(D,P,F)	Down	Processing	Full
(D,D,E)	Down	Down	Empty
(D,D,F)	Down	Down	Full

Table 1: The complete set of all state combinations between machines M_1 and M_2 and buffer B.

synthesis of separate processes into one, even simple ones like these in this exercise, can lead to a DES whose complexity is higher than that of the separate DES.

5 Question 3

Translated into the language of the transition function, the four requirements demand the removal of the transitions of the following forms:

$$\begin{aligned}
\delta((I,*,F),\texttt{START(1)}) &= (P,*,*)\\
\delta((*,I,E),\texttt{START(2)}) &= (*,P,*)\\
\delta((I,D,*),\texttt{START(1)}) &= (P,*,*)\\
\delta((D,D,*),\texttt{REPAIR(1)}) &= (I,*,*)
\end{aligned}$$

Nince transitions are removed, bringing the total number of transitions between states of the complete DES to 57. The transition function $\delta()$ is shown in table 3 and the new state machine diagram is illustrated in figure 39 of the appendix.

$\delta((I,I,E), \texttt{PLACED}) = (I,I,F)$	$\delta((P,P,F), \texttt{END(1)}) = (I,P,F)$
$\delta((I,I,E), \texttt{START(2)}) = (I,P,E)$	$\delta((P,P,F), \texttt{END(2)}) = (P,I,F)$
$\delta((I,I,E), \texttt{START(1)}) = (P,I,E)$	$\delta((P,P,F), \texttt{TAKEN}) = (P,P,E)$
$\delta((I,I,F), \texttt{TAKEN}) = (I,I,E)$	$\delta((P,P,F), \texttt{BREAKDOWN(2)}) = (P,D,F)$
$\delta((I,I,F), \texttt{START(2)}) = (I,P,F)$	$\delta((P,P,F), \texttt{BREAKDOWN(1)}) = (D,P,F)$
$\delta((I,I,F), \texttt{START(1)}) = (P,I,F)$	$\delta((P,D,E), \texttt{END(1)}) = (I,D,E)$
$\delta((I,P,E), \texttt{END(2)}) = (I,I,E)$	$\delta((P,D,E), \texttt{REPAIR(2)}) = (P,I,E)$
$\delta((I,P,E), \texttt{PLACED}) = (I,P,F)$	$\delta((P,D,E), \texttt{PLACED}) = (P,D,F)$
$\delta((I,P,E), \texttt{BREAKDOWN(2)}) = (I,D,E)$	$\delta((P,D,E), \texttt{BREAKDOWN(1)}) = (D,D,E)$
$\delta((I,P,E), \texttt{START(1)}) = (P,P,E)$	$\delta((P,D,F), \texttt{END(1)}) = (I,D,F)$
$\delta((I,P,F), \texttt{END(2)}) = (I,I,F)$	$\delta((P,D,F), \texttt{REPAIR(2)}) = (P,I,F)$
$\delta((I,P,F), \texttt{TAKEN}) = (I,P,E)$	$\delta((P,D,F), \texttt{TAKEN}) = (P,D,E)$
$\delta((I,P,F), \texttt{BREAKDOWN(2)}) = (I,D,F)$	$\delta((P,D,F), \texttt{BREAKDOWN(1)}) = (D,D,F)$
$\delta((I,P,F), \texttt{START(1)}) = (P,P,F)$	$\delta((D,I,E), \texttt{REPAIR(1)}) = (I,I,E)$
$\delta((I,D,E), \texttt{REPAIR(1)}) = (I,I,E)$	$\delta((D,I,E), \texttt{PLACED}) = (D,I,F)$
$\delta((I,D,E), \texttt{PLACED}) = (I,D,F)$	$\delta((D,I,E), \texttt{START(2)}) = (D,P,E)$
$\delta((I,D,E), \texttt{START(1)}) = (P,D,E)$	$\delta((D,I,F), \texttt{REPAIR(1)}) = (I,I,F)$
$\delta((I,D,F), \texttt{REPAIR(2)}) = (I,I,F)$	$\delta((D,I,F), \texttt{TAKEN}) = (D,I,E)$
$\delta((I,D,F), \texttt{TAKEN}) = (I,D,E)$	$\delta((D,I,F), \texttt{START(2)}) = (D,P,F)$
$\delta((I,D,F), \texttt{START(1)}) = (P,D,F)$	$\delta((D,P,E), \texttt{REPAIR(1)}) = (I,P,E)$
$\delta((P,I,E), \texttt{END(1)}) = (I,I,E)$	$\delta((D,P,E), \texttt{END(2)}) = (D,I,E)$
$\delta((P,I,E), \texttt{PLACED}) = (P,I,F)$	$\delta((D,P,E), \texttt{PLACED}) = (D,P,F)$
$\delta((P,I,E), \texttt{START(2)}) = (P,P,E)$	$\delta((D,P,E), \texttt{BREAKDOWN(2)}) = (D,D,E)$
$\delta((P,I,E), \texttt{BREAKDOWN(1)}) = (D,I,E)$	$\delta((D,P,F), \texttt{REPAIR(1)}) = (I,P,F)$
$\delta((P,I,F), \texttt{END(1)}) = (I,I,F)$	$\delta((D,P,F), \texttt{END(2)}) = (D,I,F)$
$\delta((P,I,F), \texttt{TAKEN}) = (P,I,E)$	$\delta((D,P,F), \texttt{TAKEN}) = (D,P,E)$
$\delta((P,I,F), \texttt{END(2)}) = (P,P,F)$	$\delta((D,P,F), \texttt{BREAKDOWN(2)}) = (D,D,F)$
$\delta((P,I,F), \texttt{BREAKDOWN(1)}) = (D,I,F)$	$\delta((D,D,E), \texttt{REPAIR(1)}) = (I,D,E)$
$\delta((P,P,E), \texttt{END(1)}) = (I,P,E)$	$\delta((D,D,E), \texttt{REPAIR(2)}) = (D,I,E)$
$\delta((P,P,E), \texttt{END(2)}) = (P,I,E)$	$\delta((D,D,E), \texttt{PLACED}) = (D,D,F)$
$\delta((P,P,E), \texttt{PLACED}) = (P,P,F)$	$\delta((D,D,F), \texttt{REPAIR(1)}) = (I,D,F)$
$\delta((P,P,E), \texttt{BREAKDOWN(2)}) = (P,D,E)$	$\delta((D,D,F), \texttt{REPAIR(2)}) = (D,I,F)$
$\delta((P,P,E), \texttt{BREAKDOWN(1)}) = (D,P,E)$	$\delta((D,D,F), \texttt{TAKEN}) = (D,D,E)$

Table 2: Allowed transitions between states.

δ((I,I,E), `PLACED`) = (I,I,F)
~~δ((I,I,E), `START(2)`) = (I,P,E)~~
δ((I,I,E), `START(1)`) = (P,I,E)
δ((I,I,F), `TAKEN`) = (I,I,E)
δ((I,I,F), `START(2)`) = (I,P,F)
~~δ((I,I,F), `START(1)`) = (P,I,F)~~
δ((I,P,E), `END(2)`) = (I,I,E)
δ((I,P,E), `PLACED`) = (I,P,F)
δ((I,P,E), `BREAKDOWN(2)`) = (I,D,E)
δ((I,P,E), `START(1)`) = (P,P,E)
δ((I,P,F), `END(2)`) = (I,I,F)
δ((I,P,F), `TAKEN`) = (I,P,E)
δ((I,P,F), `BREAKDOWN(2)`) = (I,D,F)
~~δ((I,P,F), `START(1)`) = (P,P,F)~~
δ((I,D,E), `REPAIR(1)`) = (I,I,E)
δ((I,D,E), `PLACED`) = (I,D,F)
~~δ((I,D,E), `START(1)`) = (P,D,E)~~
δ((I,D,F), `REPAIR(2)`) = (I,I,F)
δ((I,D,F), `TAKEN`) = (I,D,E)
~~δ((I,D,F), `START(1)`) = (P,D,F)~~
δ((P,I,E), `END(1)`) = (I,I,E)
δ((P,I,E), `PLACED`) = (P,I,F)
~~δ((P,I,E), `START(2)`) = (P,P,E)~~
δ((P,I,E), `BREAKDOWN(1)`) = (D,I,E)
δ((P,I,F), `END(1)`) = (I,I,F)
δ((P,I,F), `TAKEN`) = (P,I,E)
δ((P,I,F), `END(2)`) = (P,P,F)
δ((P,I,F), `BREAKDOWN(1)`) = (D,I,F)
δ((P,P,E), `END(1)`) = (I,P,E)
δ((P,P,E), `END(2)`) = (P,I,E)
δ((P,P,E), `PLACED`) = (P,P,F)
δ((P,P,E), `BREAKDOWN(2)`) = (P,D,E)
δ((P,P,E), `BREAKDOWN(1)`) = (D,P,E)
δ((P,P,F), `END(1)`) = (I,P,F)
δ((P,P,F), `END(2)`) = (P,I,F)
δ((P,P,F), `TAKEN`) = (P,P,E)
δ((P,P,F), `BREAKDOWN(2)`) = (P,D,F)
δ((P,P,F), `BREAKDOWN(1)`) = (D,P,F)
δ((P,D,E), `END(1)`) = (I,D,E)
δ((P,D,E), `REPAIR(2)`) = (P,I,E)
δ((P,D,E), `PLACED`) = (P,D,F)
δ((P,D,E), `BREAKDOWN(1)`) = (D,D,E)
δ((P,D,F), `END(1)`) = (I,D,F)
δ((P,D,F), `REPAIR(2)`) = (P,I,F)
δ((P,D,F), `TAKEN`) = (P,D,E)
δ((P,D,F), `BREAKDOWN(1)`) = (D,D,F)
δ((D,I,E), `REPAIR(1)`) = (I,I,E)
δ((D,I,E), `PLACED`) = (D,I,F)
~~δ((D,I,E), `START(2)`) = (D,P,E)~~
δ((D,I,F), `REPAIR(1)`) = (I,I,F)
δ((D,I,F), `TAKEN`) = (D,I,E)
δ((D,I,F), `START(2)`) = (D,P,F)
δ((D,P,E), `REPAIR(1)`) = (I,P,E)
δ((D,P,E), `END(2)`) = (D,I,E)
δ((D,P,E), `PLACED`) = (D,P,F)
δ((D,P,E), `BREAKDOWN(2)`) = (D,D,E)
δ((D,P,F), `REPAIR(1)`) = (I,P,F)
δ((D,P,F), `END(2)`) = (D,I,F)
δ((D,P,F), `TAKEN`) = (D,P,E)
δ((D,P,F), `BREAKDOWN(2)`) = (D,D,F)
~~δ((D,D,E), `REPAIR(1)`) = (I,D,E)~~
δ((D,D,E), `REPAIR(2)`) = (D,I,E)
δ((D,D,E), `PLACED`) = (D,D,F)
~~δ((D,D,F), `REPAIR(1)`) = (I,D,F)~~
δ((D,D,F), `REPAIR(2)`) = (D,I,F)
δ((D,D,F), `TAKEN`) = (D,D,E)

Table 3: Allowed transitions between states given the four requirements.

6 Question 4

The term for describing the regulation of the logical behavior of a DES in order to fulfill certain specifications is known as "Supervisory Control". The events of interest, `START` and `REPAIR`, are enabled or disabled by the controller in such a way that the controller might not enable, for example, the `START(1)` signal (event) until it has certainty that the actual state is: $(M_1, B, M_2) \equiv$ `(IDLE(1),EMPTY,!DOWN(2))`.

A Appendix

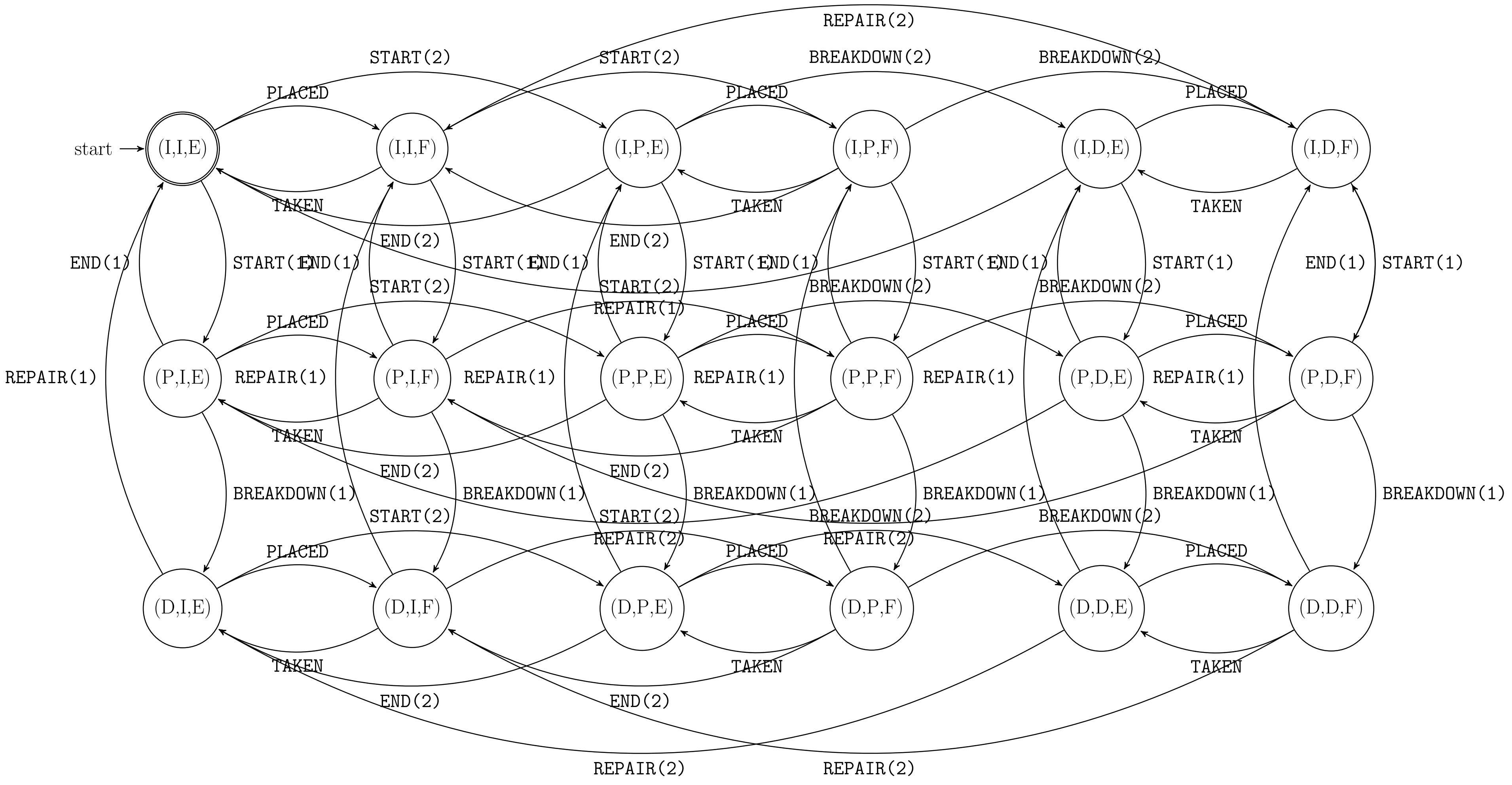

Figure 38: The state machine diagram of the single automaton that models the complete behaviour of the manufacturing process.

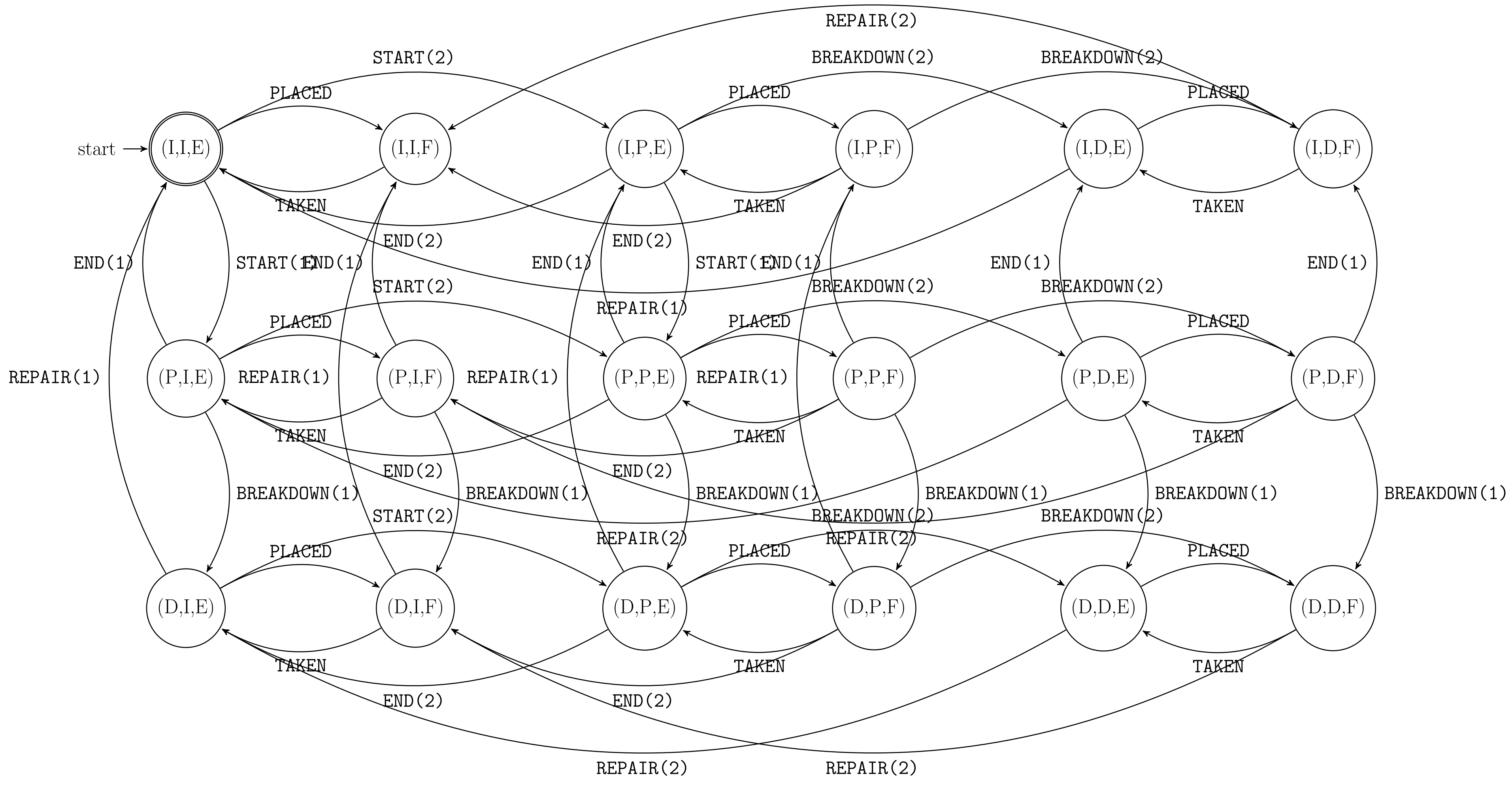

Figure 39: The state machine diagram of the single automaton that models the complete behaviour of the manufacturing process, given the four requirements.