VT16 – EL2450 – Assignment II

Alexandros Filotheou
871108-5590
alefil@kth.se

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

1 Part One: Scheduling

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.

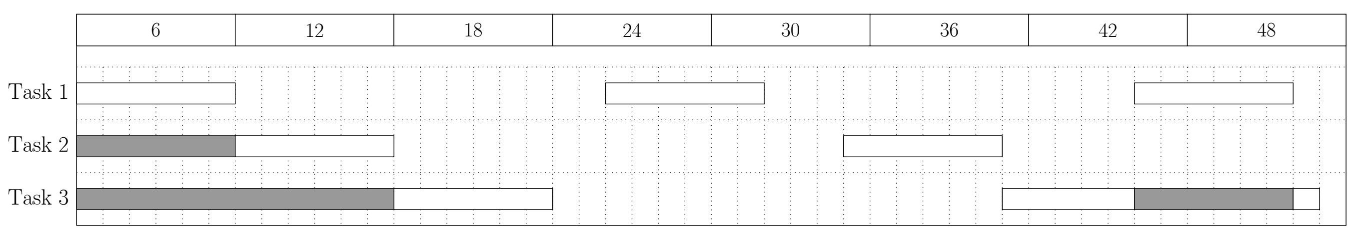

Figure 1: A portion of the RM schedule σ for tasks J_1, J_2, J_3. Shaded areas denote the waiting time.

1.1.3 Question 3

All penduli 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 2 shows the angular displacement of each pendulum.

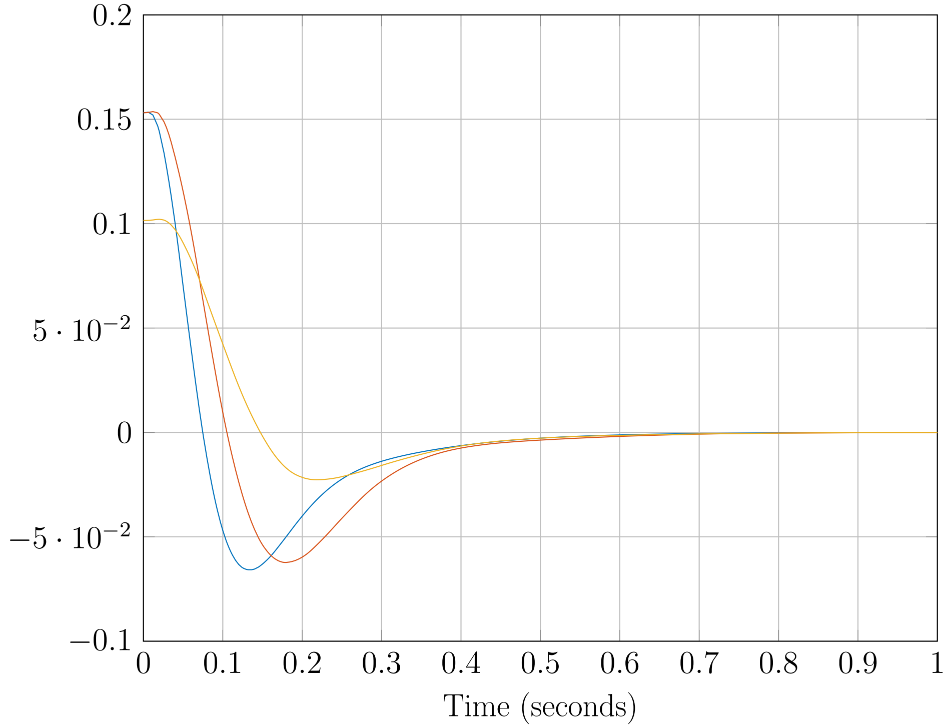

Figure 2: The angular displacement of each pendulum as a function of time. `Blue`: P_1, `Red`: P_2, `Orange`: P_3

1.1.4 Question 4

Figure 3 shows the actual execution schedule for each pendulum over a timespan of $lcm(20, 29, 35) = 4060$ ms: exactly one period of the schedule. 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.

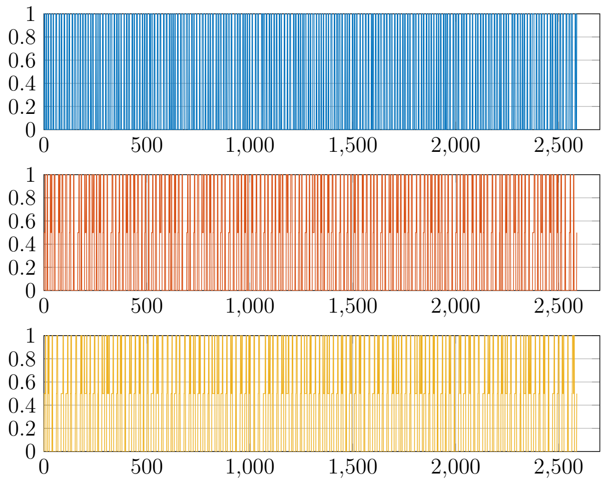

Figure 3: The calculated schedule for the three penduli. `Blue`: P_1, `Red`: P_2, `Orange`: P_3. $C_i = 6$ ms.

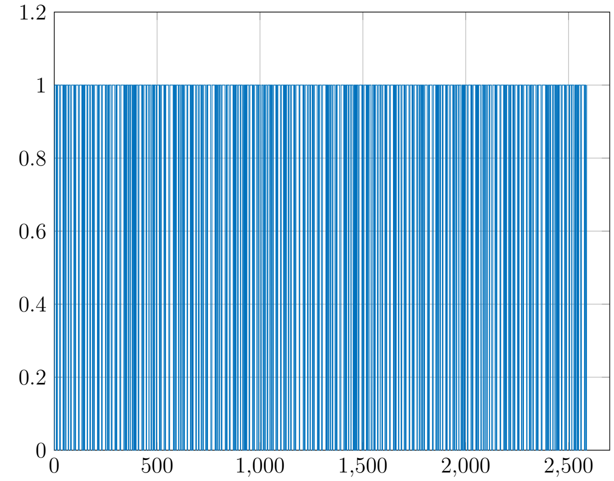

Figure 4: The overall processing usage. Notice that it is at most at 100%. $C_i = 6$ ms.

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.

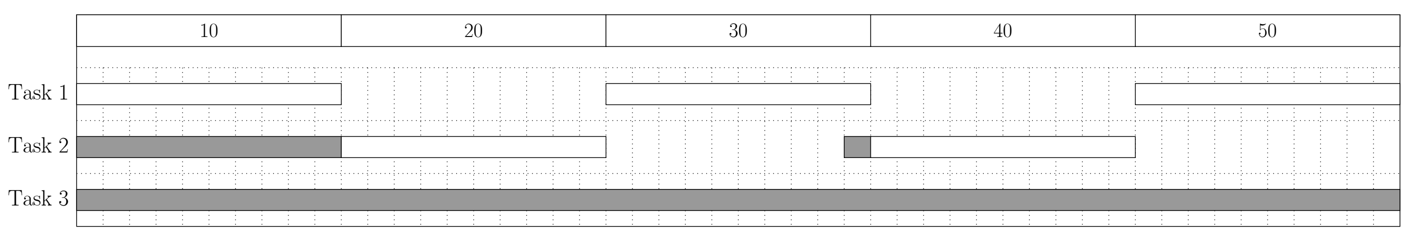

Figure 5: A portion of the RM schedule σ for tasks J_1, J_2, J_3 for $C_i = 10$ ms. Shaded areas denote the waiting time. Notice that J_3 misses its deadlines consecutively, indicative of the inability of schedulability.

Pendulum P_3 is left to its own devices and tends towards instability slowly but surely. This makes sense since no control signal pertaining to P_3 is assigned execution time in the processor. Figure 7 shows the angular displacement of pendulum P_3 as a function of time.

For penduli 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 6 shows the angular displacement of each stable pendulum as a function of time.

Figures 8 and 9 show the angular displacement of penduli P_1 and P_2 respectively, as a function of time, for the two different cases of execution time.

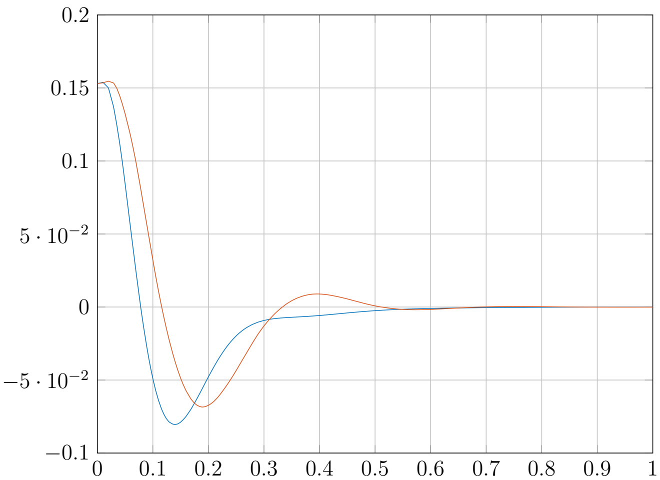

Figure 6: The angular displacement of the stable penduli P_1 and P_2 as a function of time. `Blue`: P_1, `Red`: P_2. $C_i = 10$ ms.

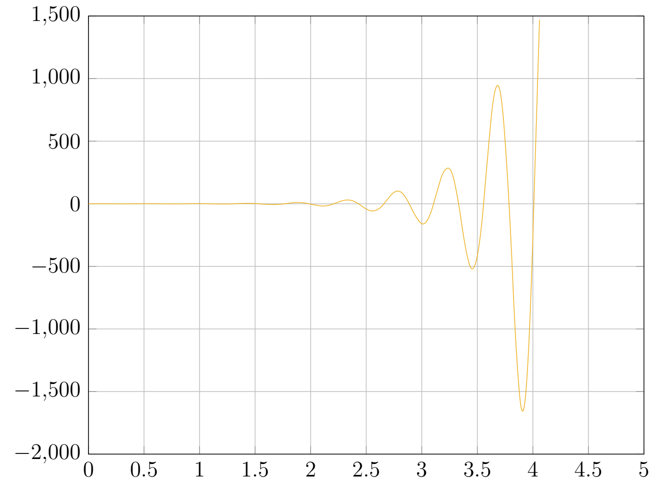

Figure 7: The angular displacement of the unstable pendulum P_3 as a function of time. $C_i = 10$ ms.

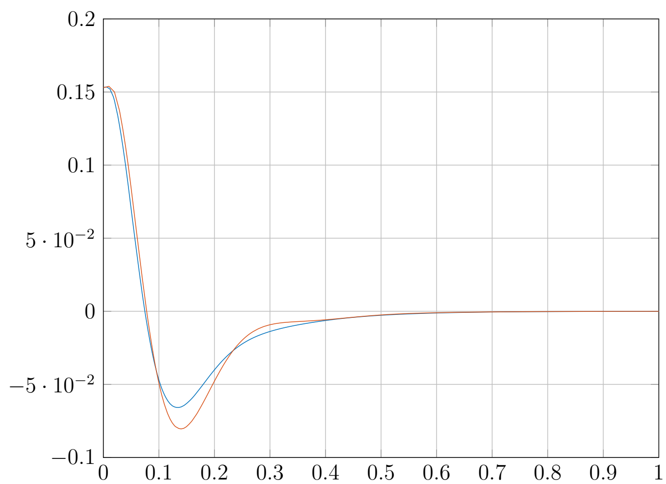

Figure 8: The angular displacement of pendulum P_1 as a function of time. `Blue`: $C_1 = 6$ ms, `Red`: $C_1 = 10$ ms

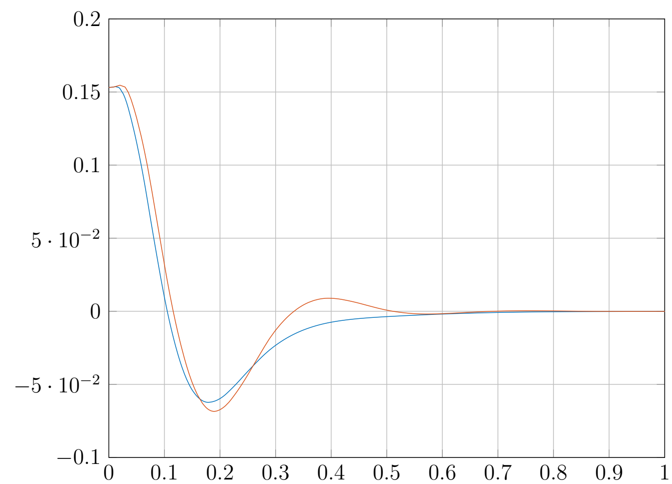

Figure 9: The angular displacement of pendulum P_2 as a function of time. `Blue`: $C_2 = 6$ ms, `Red`: $C_2 = 10$ ms

Figure 10 shows the schedule calculated for each pendulum with all jobs having execution time $C_i = 10$ 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.

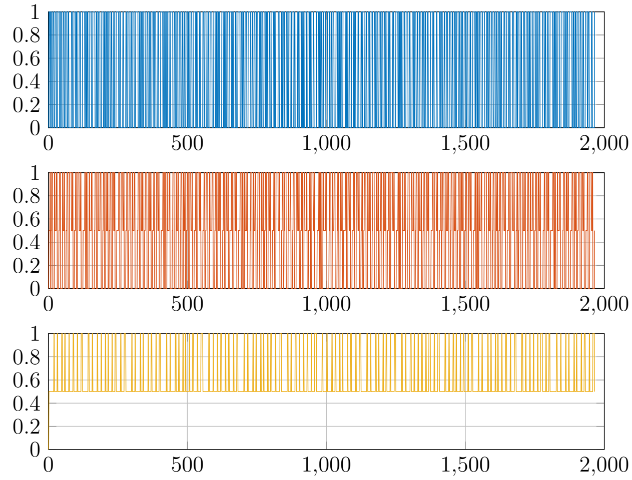

Figure 10: The calculated schedule for the three penduli. `Blue`: P_1, `Red`: P_2, `Orange`: P_3. $C_i = 10$ ms.

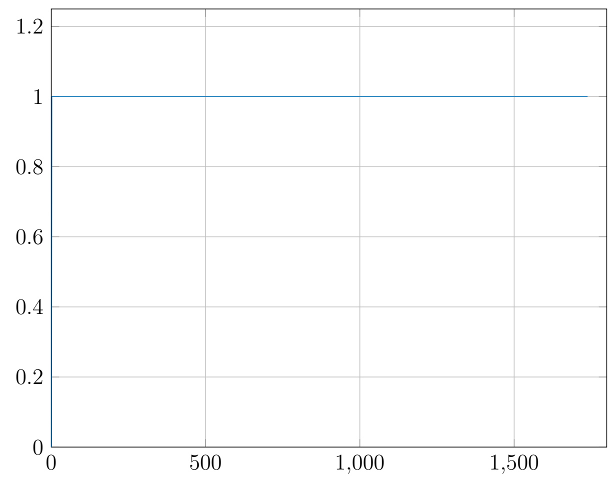

Figure 11: The overall processing usage. Notice that it always at 100%. $C_i = 10$ ms.

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 with 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.