

# Stability-Aware Analysis and Design of Embedded Control Systems

Amir Aminifar<sup>1</sup>, Petru Eles<sup>1</sup>, Zebo Peng<sup>1</sup>, Anton Cervin<sup>2</sup>

<sup>1</sup>Department of Computer and Information Science, Linköping University, Sweden

<sup>2</sup>Department of Automatic Control, Lund University, Sweden

**Abstract**—Many embedded systems comprise several controllers sharing available resources. It is well known that such resource sharing leads to complex timing behavior that can jeopardize stability of control applications, if it is not properly taken into account in the design process, e.g., mapping and scheduling. As opposed to hard real-time systems where meeting the deadline is a critical requirement, control applications do not enforce hard deadlines. Therefore, the traditional real-time analysis approaches are not readily applicable to control applications. Rather, in the context of control applications, stability is often the main requirement to be guaranteed, and can be expressed as the amount of delay and jitter a control application can tolerate. The nominal delay and response-time jitter can be regarded as the two main factors which relate the real-time aspects of a system to control performance and stability. Therefore, it is important to analyze the impact of variations in scheduling parameters, i.e., period and priority, on the nominal delay and response-time jitter and, ultimately, on stability. Based on such an analysis, we address, in this paper, priority assignment and sensitivity analysis problems for control applications considering stability as the main requirement.

## I. INTRODUCTION

Many embedded systems, e.g., automotive systems, comprise several control applications. The design of such systems requires special attention due to the fundamental difference between such control systems and what we classically understand by hard real-time systems. While in the real-time system area most of the analysis algorithms assume that applications have hard deadlines, control applications do not primarily enforce hard deadlines. As opposed to classical hard real-time systems, in the control area, stability is the fundamental requirement considered. The stability of a control application is directly related to the amount of delay and jitter it can tolerate. Therefore, in the context of embedded control systems, not only the nominal delay, but also the response-time jitter is an important factor [25]. Ignoring this fact can potentially lead to suboptimal and/or unstable design solutions.

In this paper, we consider the nominal delay and worst-case response-time jitter to link the real-time and control areas. Considering the two metrics, our goal is to analyze the effect of variation in scheduling parameters, i.e., priorities and periods, on the stability of control applications. This is done in two consecutive steps. The first step is to investigate the impact of this variation on the nominal delay and worst-case response-time jitter. The second step is to interpret these changes in the nominal delay and worst-case response-time jitter in terms of stability of the control application, which is facilitated by the Jitter Margin toolbox [13], [9], [8].

We show that the worst-case response-time jitter does not have the monotonicity property, and hence, the existing optimal priority assignment, e.g., [4], and sensitivity analysis, e.g., [21], cannot be applied immediately. The problem is addressed by considering the linear bounds on the exact results. To

this end, we have developed a lower bound on the best-case response time and used the existing upper bound on the worst-case response time developed in [6].

The first main contribution of this paper is an optimal priority assignment similar to Audsley’s algorithm [4] but dealing with worst-case control quality and stability instead of worst-case response time and deadline. Thus, the novelty of our priority assignment is in finding stable design solutions. The priority assignment algorithm is optimal in the sense that if there exists any priority assignment policy that can find a priority order for the considered set of applications such that all controllers are stable, so can our priority assignment. The second main contribution of the paper is to perform sensitivity analysis for sampling frequencies with respect to stability of control applications. The sensitivity analysis identifies the shortest distance of an operating point to the border of feasibility region, i.e., the region within which all control applications are guaranteed to be stable. This can be regarded as a metric for quantifying the robustness of different operating points.

## II. SYSTEM MODEL

### A. Plant Model

Let us consider a given set of plants  $\mathbf{P}$ . Each plant  $P_i$  is modeled by a continuous-time system of equations [3]

$$\begin{aligned}\dot{\mathbf{x}}_i &= \mathbf{A}_i \mathbf{x}_i + \mathbf{B}_i \mathbf{u}_i, \\ \mathbf{y}_i &= \mathbf{C}_i \mathbf{x}_i,\end{aligned}\tag{1}$$

where  $\mathbf{x}_i$  and  $\mathbf{u}_i$  are the plant state and control signal, respectively. The control signal is updated at some point in each sampling period and is held constant between updates. The plant output, denoted by  $\mathbf{y}_i$ , is sampled with the constant interval  $h_i$ .

### B. Platform and Application Model

The platform considered in this paper is a uniprocessor. Fixed-priority preemptive scheduling is considered throughout this paper assuming a set of independent tasks  $\mathbf{T}$ . Each task  $\tau_i \in \mathbf{T}$  is identified by four parameters,

- *unique priority* denoted by  $\rho_i$ ,
- *worst-case execution-time* denoted by  $c_i^w$ ,
- *best-case execution-time* denoted by  $c_i^b$ ,
- *period* denoted by  $h_i$  ( $f_i = \frac{1}{h_i}$ ).

Therefore, task  $\tau_i$  can be identified by a tuple,  $\tau_i = (\rho_i, c_i^w, c_i^b, h_i)$ . Each control application  $\Lambda_i \in \mathbf{\Lambda}$  has a corresponding task  $\tau_i$ . The set of all applications is captured by  $\mathbf{\Lambda}$ .

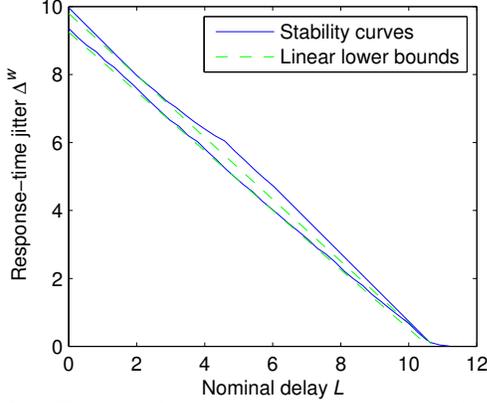


Figure 1. The stability curves generated by the Jitter Margin toolbox and their linear lower bounds (the area below the curves is the stable area)

### III. STABILITY ANALYSIS

In order to quantify the amount of delay and jitter a control application can tolerate before instability, we use the Jitter Margin toolbox [13], [9], [8]. The Jitter Margin toolbox provides sufficient stability conditions for a closed-loop system with a linear continuous-time plant and a linear discrete-time controller.

The Jitter Margin toolbox provides the stability curve that determines the maximum tolerable response-time jitter based on the nominal delay. The solid curves in Figure 1 are examples of the stability curves generated by the Jitter Margin toolbox for two different sampling periods. Observe that the area below each solid curve is the stable area. The graph is generated for the plant with transfer function  $P = \frac{1000}{s^2 + s}$  and a discrete-time Linear-Quadratic-Gaussian (LQG) controller. The upper and lower solid curves correspond to sampling periods 6 ms and 12 ms, respectively.

For a given sampling period, the stability curve can be *safely* approximated by a linear function of the nominal delay and worst-case response-time jitter. The linear *stability condition* for control application  $\Lambda_i$  is of the form  $L_i + \alpha_i \Delta_i^w \leq \beta_i$ , where  $\alpha_i \geq 1, \beta_i \geq 0$ . The nominal delay, denoted by  $L_i$ , identifies the constant part of the delay that the control application  $\Lambda_i$  experiences, whereas the worst-case response-time jitter, denoted by  $\Delta_i^w$ , captures the varying part of the delay (see Figure 2, where  $R^b$  and  $R^w$  represent the best-case and worst-case response times, respectively). The linear lower bounds, depicted by the dashed lines, on the original curves generated by the Jitter Margin toolbox are shown in Figure 1. Observe that the linear lower bounds can efficiently capture the stable area identified by the Jitter Margin toolbox.

### IV. DELAY AND JITTER ANALYSES

In order to apply the stability analysis discussed in the previous section, the values of the nominal delay ( $L_i$ ) and worst-case response-time jitter ( $\Delta_i^w$ ) of control application  $\Lambda_i$  should be computed. The two metrics are defined based on the worst-case and best-case response times as follows,

$$\begin{aligned} L_i &= R_i^b, \\ \Delta_i^w &= R_i^w - R_i^b, \end{aligned} \quad (2)$$

where  $R_i^w$  and  $R_i^b$  denote the worst-case and best-case response times, respectively.

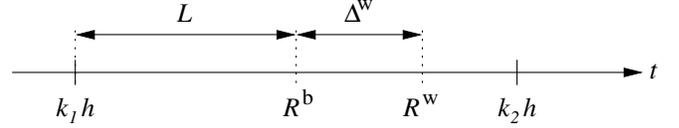


Figure 2. Graphical interpretation of the nominal delay and worst-case response-time jitter

In the following, we give a brief overview on computing the worst-case, and best-case response times. Further, we introduce linear bounds on worst-case and best-case response times and these bounds are used to address the monotonicity requirements in the analyses.

#### A. Worst-Case Response Time Analysis

Under fixed-priority preemptive scheduling, assuming deadline  $D_i \leq h_i$  and an independent task set, the exact worst-case response time of a task  $\tau_{ij}$  can be computed by the following equation [12],

$$R_i^w = c_i^w + \sum_{\tau_j \in hp(\tau_i)} \left\lceil \frac{R_i^w}{h_j} \right\rceil c_j^w, \quad (3)$$

where  $hp(\tau_i)$  denotes the set of higher priority tasks for task  $\tau_i$ . Equation 3 is solved by fixed-point iteration starting with, e.g.,  $R_i^w = c_i^w$ .

The worst-case response time for independent task sets with arbitrary deadlines is given by [14] and [24],

$$\begin{aligned} w_i(q) &= (q+1)c_i^w + \sum_{\tau_j \in hp(\tau_i)} \left\lceil \frac{w_i(q)}{h_j} \right\rceil c_j^w, \\ R_i^w &= \max_q \{w_i(q) - qh_i\}. \end{aligned} \quad (4)$$

Under the assumption of arbitrary deadlines, all instances in the busy period must be considered in order to obtain the worst-case response time.

An upper bound for computing the worst-case response time for a task  $\tau_i$  in fixed-priority scheduling suggested by [6] is as follows,

$$\bar{R}_i^w = \frac{c_i^w + \sum_{\tau_j \in hp(\tau_i)} c_j^w (1 - u_j^w)}{1 - \sum_{\tau_j \in hp(\tau_i)} u_j^w}, \quad (5)$$

where  $u_i^w = \frac{c_i^w}{h_i}$  is the worst-case utilization of the task  $\tau_i$ .

#### B. Best-Case Response Time Analysis

Under fixed-priority preemptive scheduling, assuming  $D_i \leq h_i$  and an independent task set, the exact best-case response time of a task  $\tau_i$  is given by the following equation [22],

$$R_i^b = c_i^b + \sum_{\tau_j \in hp(\tau_i)} \left\lceil \frac{R_i^b}{h_j} - 1 \right\rceil c_j^b. \quad (6)$$

Similar to worst-case response-time analysis, Equation 6 also has to be solved by iteration, but starting from an initial value of, e.g.,  $R_i^b = R_i^w$ . The above equation is also a valid lower bound for task sets with arbitrary deadlines.

Using techniques similar to [6], we provide a lower bound for the best-case response time as follows,

$$\underline{R}_i^b = \max \left\{ \frac{c_i^b - \sum_{\tau_j \in hp(\tau_i)} c_j^b (1 - u_j^b)}{1 - \sum_{\tau_j \in hp(\tau_i)} u_j^b}, c_i^b \right\}, \quad (7)$$

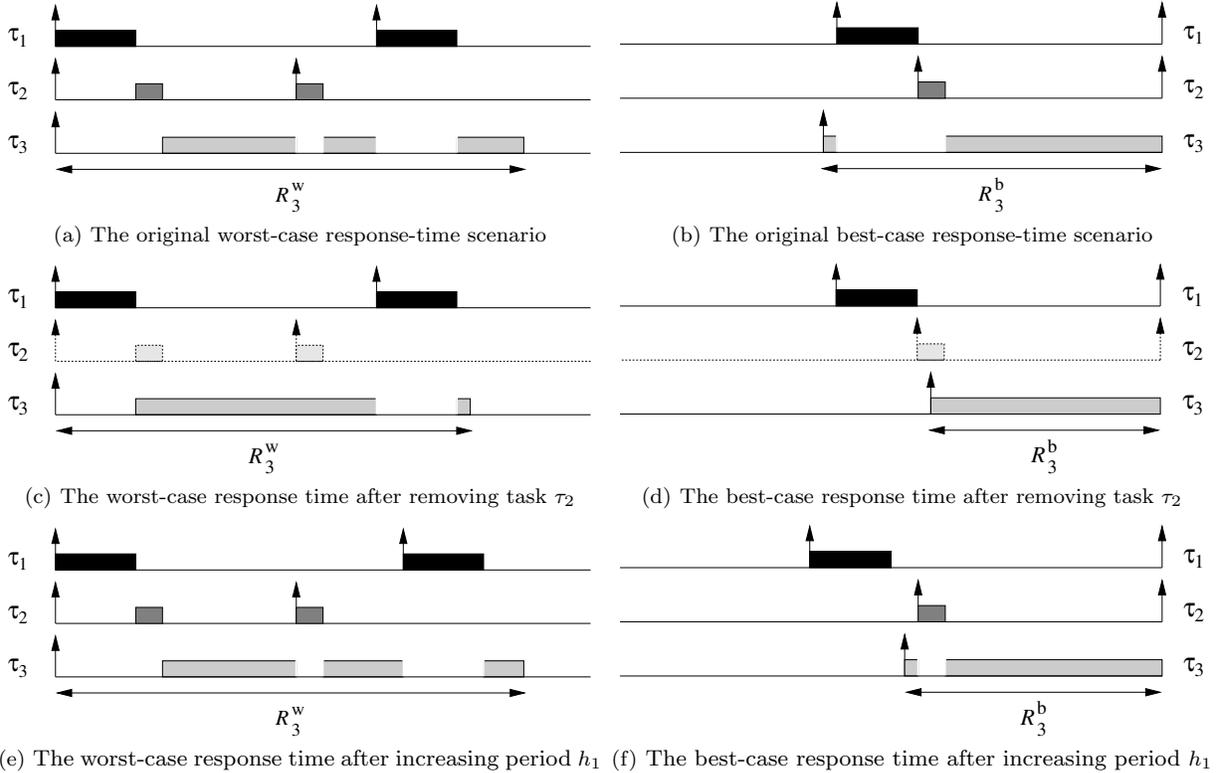


Figure 3. Non-monotonicity of response-time jitter with respect to priorities and periods

where  $u_i^b = \frac{c_i^b}{h_i}$  denotes the best-case utilization of task  $\tau_i$  (see Appendix A for the proof).

## V. PROPERTIES OF EXACT DELAY AND JITTER

In this section, we investigate the effect of changing scheduling parameters, i.e., priorities and periods, on the delay and jitter a control application experiences. In particular, it will be shown that the monotonicity property required for performing priority assignment and sensitivity analysis does not hold for the response-time jitter obtained based on Equations 3, 4, and 6.

### A. Analysis with Respect to Priorities

The nominal delay and worst-case response-time jitter, defined in Equation 2, only depend on the best-case and worst-case response times. Further, the best-case and worst-case response times only depend on the *set* of higher priority tasks. Therefore, the nominal delay and response-time jitter remain unchanged as long as the set of higher priority tasks remains the same and the relative priority order of the higher priority tasks (and lower priority tasks) is irrelevant.

In addition to the above, we shall consider the effect of removing a higher priority task from the set of high priority tasks. It is clear from Equation 6 that removing a higher priority task results in less or equal nominal delay. As opposed to the nominal delay, the response-time jitter, however, does not monotonically change with removing high priority tasks due to the jumps in the worst-case and best-case response times as results of the ceiling functions in Equations 3, 4, and 6. This will be illustrated using a small example.

It should be reminded that the worst-case response time occurs at the critical instant, i.e., when the task under analysis

is released at the same time as all other high priority tasks [12]. The best-case response time occurs when the task under analysis is released such that it finishes executing simultaneously with the release of all its high priority tasks, i.e., at the favorable instant [22].

Let us consider three tasks  $\tau_1 = (\rho_1 = 3, c_1^w = 3, c_1^b = 3, h_1 = 12)$ ,  $\tau_2 = (2, 1, 1, 9)$ ,  $\tau_3 = (1, 9.5, 8.5, 100)$ . The worst-case and best-case scenarios for task  $\tau_3$ , under analysis, are shown in Figures 3(a) and 3(b). The set of higher priority tasks is  $hp(\tau_3) = \{\tau_1, \tau_2\}$ . Let us further consider implicit deadlines for all three tasks. The worst-case and best-case response times are  $R_3^w = 9.5 + 2 \times 3 + 2 \times 1 = 17.5$  and  $R_3^b = 8.5 + 1 \times 3 + 1 \times 1 = 12.5$ , respectively. The worst-case jitter in the response time of task  $\tau_3$  is  $\Delta_3^w = 5$ . Let us remove task  $\tau_2$  from the set of higher priority tasks of task  $\tau_3$ , i.e.,  $hp(\tau_3) = \{\tau_1\}$ . Figures 3(c) and 3(d) show the new worst-case and best-case response times. The dotted lines show the execution of task  $\tau_2$  according to the previous scenario. The worst-case and best-case response times decrease to  $R_3^w = 9.5 + 2 \times 3 + 0 \times 1 = 15.5$  and  $R_3^b = 8.5 + 0 \times 3 + 0 \times 1 = 8.5$ . The worst-case jitter in the response time of task  $\tau_3$  is, however, increased to  $\Delta_3^w = 7$  by removing high priority task  $\tau_2$ .

### B. Analysis with Respect to Periods

The nominal delay, as defined in Equation 2, monotonically increases with decrease in the period of high priority tasks. Increasing the periods of higher priority tasks results in less or equal interference at the favorable instant. In other words, increasing the periods of higher priority tasks may cause some instances of these tasks to fall outside the interference scenario.

While the nominal delay monotonically decreases with increasing the periods of the higher priority tasks, the worst-case response-time jitter does not monotonically change with periods. This can be illustrated by an example as follows. Let us consider the same task set as in the previous example, i.e.,  $\tau_1 = (3, 3, 3, 12)$ ,  $\tau_2 = (2, 1, 1, 9)$ ,  $\tau_3 = (1, 9.5, 8.5, 100)$ . The deadlines are considered to be equal to the periods. The worst-case and best-case instants are shown in Figures 3(a) and 3(b) and, as shown before, we have  $R_3^w = 17.5$ ,  $R_3^b = 12.5$ , and  $\Delta_3^w = 5$ . Now, let us increase the period of task  $\tau_1$  to  $h_1 = 13$ . Figures 3(e) and 3(f) show the new worst-case and best-case scenarios. While the worst-case response time remains the same, the best-case response time of task  $\tau_3$  is decreased to  $R_3^b = 8.5 + 0 \times 3 + 1 \times 1 = 9.5$ , leading to an increase in the worst-case response-time jitter experienced by task  $\tau_3$  ( $\Delta_3^w = 8$ ).

The section can be summarized in the following Remark.

**Remark 1.** The nominal delay and worst-case response-time jitter defined in Equation 2 have the following properties,

- 1) Priorities:
  - The nominal delay and worst-case response-time jitter experienced by a task are independent of the priority order of other tasks as long as the set of higher priority tasks remains the same.
  - Increasing the priority level of a task leads to a less or equal nominal delay for that task.
  - Increasing the priority level of a task does *not* necessarily lead to a less or equal worst-case response-time jitter for that task.
- 2) Periods:
  - Shorter period for a higher priority task leads to a greater or equal nominal delay for the task under analysis.
  - Shorter period for a higher priority task does *not* necessarily lead to a greater or equal worst-case response-time jitter for the task under analysis.

## VI. PROPERTIES OF BOUNDED DELAY AND JITTER

As discussed in the previous section, the worst-case response-time jitter does not monotonically change with priorities and periods. To address this monotonicity problem, utilizing simpler, but safe, bounds instead of the exact values is inevitable. Therefore, we redefine the nominal delay and worst-case response-time jitter as follows,

$$\begin{aligned} \underline{L}_i &= \begin{cases} R_i^b & \text{for priority assignment} \\ R_i^b & \text{for sensitivity analysis} \end{cases} \\ \overline{\Delta}_i^w &= \overline{R}_i^w - \underline{R}_i^b, \end{aligned} \quad (8)$$

where  $\overline{R}_i^w$  and  $\underline{R}_i^b$  denote the linear bounds on worst-case and best-case response times obtained according to Equations 5 and 7, respectively. Note that, Cervin et. al., in an earlier paper [9], showed that it is always safe, from the stability perspective, to over-approximate the worst-case response time and under-approximate the best-case response time.

While using the linear bounds leads to a more pessimistic analysis, it provides monotonicity which facilitates the process of analysis. These results can be summarized in the following remark.

**Remark 2.** The nominal delay and worst-case response-time jitter as defined in Equation 8 have the following properties,

- 1) Priorities:
  - *Commutativity property*: the nominal delay and worst-case response-time jitter experienced by a task are independent of the priority order of other tasks as long as the set of higher priority tasks remains the same.
  - *Monotonicity property*: increasing the priority level of a task leads to less or equal nominal delay and worst-case response-time jitter for that task.
- 2) Periods:
  - *Monotonicity property*: shorter period for a higher priority task leads to greater or equal nominal delay and worst-case response-time jitter for the task under analysis.

The proof of the above is given in Appendix B. Note that the linear lower bound for the best-case response-time, given by Equation 7, does not monotonically change with priorities. Therefore, the nominal delay in Equation 8 is define, for the priority assignment, based on the best-case response time defined in Equation 6.

**Remark 3.** The commutativity and monotonicity properties discussed in Remark 2 also hold for  $\underline{L}_i + \alpha_i \overline{\Delta}_i^w$ , since  $\alpha_i$  is (constant and) nonnegative and  $\underline{L}_i$  and  $\overline{\Delta}_i^w$  have commutativity and monotonicity properties (it should be noted that the nominal delay and worst-case response-time jitter are both either non-increasing or non-decreasing).

Observe that Remark 3 finally bridges the gap between the control stability and the real-time related notions of delay and jitter and provides us with the possibility of analyzing how stability of a control application depends on scheduling parameters, i.e., priorities and periods. Further, Remark 3 is considered to be the basis of the proposed methods in the next sections.

In the following sections, two main problems are addressed. Section VII presents an optimal priority assignment algorithm where the priorities are assigned such that all control applications are guaranteed to be stable (if there does exist a possible priority assignment). In Section VIII, a sensitivity analysis approach is proposed for the space of sampling frequencies.

## VII. OPTIMAL PRIORITY ASSIGNMENT

The problem of priority assignment has previously been addressed in the context of real-time applications with hard deadlines. The priority assignment problem for the hard real-time application focuses on assigning priorities such that all tasks are schedulable. Optimality of rate monotonic priority assignment for independent synchronous task sets with implicit deadlines is shown by Serlin [23] and Liu and Layland [16]. In the case of constrained deadlines and synchronous task sets, it is shown that deadline monotonic priority assignment is the optimal policy [15]. Audsley [4] proposed an optimal priority assignment algorithm for independent asynchronous task sets. The algorithm is also applicable to task sets with arbitrary deadlines. Davis and Burns [11] proposed a robust priority assignment algorithm based on Audsley's priority assignment. Note that *optimality* is defined with regard to the respective schedulability test, i.e., a priority assignment policy is referred to as optimal if, considering the given

schedulability test, there are no task sets that are schedulable by another priority assignment policy, that are not schedulable by the optimal priority assignment [11]. Recently, Mancuso et. al. [17] proposed an optimal priority assignment for control applications considering a linearization of the original control cost function.

While optimal priority assignment for hard real-time applications has been discussed to a great extent, it has gained less attention in the context of control applications. For control applications, the priority assignment problem can be defined to find a priority order for which all control applications are stable. The difficulty in approaching such a problem is first in capturing control stability in terms of real-time metrics. The second difficulty arises since stability depends on, as opposed to hard real-time applications, not only the response-time delay, but also the response-time jitter [25] which does not monotonically change with priorities (see Section V).

In order to overcome the problem discussed, the nominal delay and response-time jitter are defined based on the bounds in Equation 8 for which both monotonicity and commutativity properties hold (Remark 2). Moreover, the function  $\underline{L}_i + \alpha_i \overline{\Delta}_i^w$  preserves these properties as discussed in Remark 3. Having the required properties, Algorithm 1 is proposed which is adapted from the priority assignment algorithms proposed in [4], [11], and [1]. Observe that the algorithm has quadratic time complexity which clearly states its scalability.

In the following, Algorithm 1 will be explained in details. In the first step, Algorithm 1 identifies all control applications which can be assigned the lowest priority and still are stable. This group of controllers is captured by  $\mathbf{G}_1$ . Then, the controllers in group  $\mathbf{G}_1$  are removed from the set of all control applications and the algorithm proceeds with the same procedure for the remaining controllers  $\mathbf{S}$  to obtain group  $\mathbf{G}_2$ . The algorithm terminates when either the set of remaining controllers is empty ( $\mathbf{S} = \emptyset$ ), or group  $\mathbf{G}_i$  is empty ( $\mathbf{G}_i = \emptyset$ ). While the former case is the normal termination, the latter indicates that there does not exist any priority assignment for the given application set which can guarantee the stability of all control applications. The stability of a control application  $\Lambda_i$  is investigated by assessing the stability condition  $\underline{L}_i + \alpha_i \overline{\Delta}_i^w \leq \beta_i$  (see Section III). The algorithm produces the priority groups  $\mathbf{G}_i$  and guarantees stability as long as the relative priority order among the groups is preserved (i.e., all applications in group  $\mathbf{G}_i$  have higher priority than all applications in group  $\mathbf{G}_j$ , for all  $i$  and  $j$ , where  $i > j$ ).

The essential properties to prove the validity and optimality of Algorithm 1 are the commutativity and monotonicity as discussed in Remark 3 [11]. To prove the validity of the priority order produced, we note that all applications in group  $\mathbf{G}_i$  are stable as long as the high priority task set is *exactly equal* to  $\Lambda \setminus \mathbf{G}$  due to the commutativity property ( $\mathbf{G} = \bigcup_{j=1}^{i-1} \mathbf{G}_j$ ). Considering the monotonicity property this can be extended further, i.e., all applications in group  $\mathbf{G}_i$  are stable as long as the high priority task set is a *subset* of  $\Lambda \setminus \mathbf{G}$ . Since Algorithm 1 ensures that the high priority task set for all control applications in any group  $\mathbf{G}_i$  is a subset of  $\Lambda \setminus \mathbf{G}$ , the resulting priority assignment guarantees the stability of all control applications.

As the proof sketch of optimality, it is sufficient to show that there does not exist any feasible priority assignment if group  $\mathbf{G}_i$  is empty (only in such a case our algorithm fails to

---

### Algorithm 1 Optimal Priority Assignment

---

```

1: %  $\mathbf{S}$ : remaining applications set;
2: %  $\mathbf{G}_i$ : the  $i$ -th group;
3: Compute  $\alpha_i$  and  $\beta_i, \forall \Lambda_i$ ;
4: Initialize set  $\mathbf{S} = \Lambda$ ;
5: for  $n = 1$  to  $|\Lambda|$  do
6:    $\mathbf{G}_n = \emptyset$ ;
7:   for  $\Lambda_i \in \mathbf{S}$  do
8:     • Delay and jitter analyses  $\underline{L}_i$  and  $\overline{\Delta}_i^w$ ,
       considering  $hp(\Lambda_i) = \mathbf{S} \setminus \{\Lambda_i\}$ ;
9:     if  $\underline{L}_i + \alpha_i \overline{\Delta}_i^w \leq \beta_i$  then
10:       $\mathbf{G}_n = \mathbf{G}_n \cup \{\Lambda_i\}$ ;
11:     end if
12:   end for
13:   if  $\mathbf{S} == \emptyset$  then
14:     % Terminate!
15:     return  $\langle \mathbf{G}_1, \mathbf{G}_2, \dots, \mathbf{G}_n \rangle$ ;
16:   else if  $\mathbf{G}_n == \emptyset$  then
17:     % No possible solution!
18:     return  $\langle \rangle$ ;
19:   else
20:      $\mathbf{S} = \mathbf{S} \setminus \mathbf{G}_n$ ;
21:   end if
22: end for

```

---

find a feasible priority assignment). Due to the monotonicity property, considering the applications  $\mathbf{G} = \bigcup_{j=1}^{i-1} \mathbf{G}_j$  as low priority applications only improves the stability margin for the remaining applications  $\Lambda \setminus \mathbf{G}$  (Remark 3), and therefore, it is safe to ignore all applications in  $\mathbf{G}$ . Moreover, in a priority order for a set of applications, inevitably, one of the applications will be assigned the lowest priority. Therefore, among the remaining applications  $\Lambda \setminus \mathbf{G}$ , to have a feasible priority order, there must exist at least one application which is stable even if it is assigned the lowest priority (i.e.,  $\mathbf{G}_i \neq \emptyset$ ). Note that in this step, the relative priority order of the higher priority applications is not important due to the commutativity property as discussed in Remark 3.

An intrinsic property of Algorithm 1 is that the priorities inside each group can be assigned arbitrarily and this can be used for further optimization, e.g., with respect to expected control quality or energy consumption. This property can simply be clarified by, first, observing that the priority order of tasks in a group has no impact on the stability of the tasks in other groups due to the commutativity property. Second, each control task can be assigned the lowest priority in its group and increasing the priority level of each task inside its group can never lead to a worse stability margin (i.e., delay and jitter) due to the monotonicity property. Hence, the priorities within each group can be assigned arbitrarily.

## VIII. SAMPLING FREQUENCY SENSITIVITY ANALYSIS

Sensitivity analysis provides the designer with useful information regarding the feasibility slack at the current operating point, which determines the distance to the border of the feasibility region. Sensitivity analysis is often restricted to the one-dimensional case, where only a single property of one application is considered to be subject to change, due to the complexity of the multi-dimensional case. Moreover, the early work on sensitivity analysis, in the area of real-time systems, regarded the periods to be fixed since they are related to the environment and focused mainly on metrics related to the worst-case execution-time [19]. Cottet and Babau [10]

proposed a graphical approach to adjust task periods considering the deadline to be the acceptance criterion. Racu et. al. [21] developed a framework for one and multi-dimensional sensitivity analysis of complex embedded real-time systems. The framework is based on the monotonicity property and binary search algorithm. However, performing a feasibility test in each iteration of the binary search leads to a computationally complex process, in particular, for the multi-dimensional sensitivity analysis which has been remedied by a stochastic algorithm based on evolutionary search techniques.<sup>1</sup>

As opposed to the previous work where periods are regarded to be fixed, in the context of control applications, the sampling periods of controllers can often be set to any value in an interval obtained based on common rules of thumb [3]. Palopoli et. al. [18] proposed an approach to find the stability radius for control applications considering a time-triggered model of computation and translating the stability into deadline, but at a price: such approaches are restricted to a time-triggered model of computation which can potentially lead to under-utilization or poor control performance [2]. Bini et. al. [5] considered a simpler task model compared to [20] and proposed a new type of sensitivity analysis that also applies to the domain of task periods for control systems which perform rate adaptation to avoid overload conditions. However, the proposed sensitivity analysis approach is still based on the concept of deadlines, while control applications do not enforce hard deadlines. Therefore, for embedded control systems, a new sensitivity analysis approach is needed based on both the nominal delay and response-time jitter, capturing the stability of control applications.

The basic goal of sensitivity analysis is to determine the feasibility slack in any possible direction at a given point. The feasibility slack is defined as the distance from the current operating point to the border of the feasibility region, i.e., the region within which all applications satisfy their requirements. In this paper, given an operating point in the space of task frequencies, the objective is to find out how robust the operating point is. Moreover, it is possible to identify the most robust – least sensitive – sampling frequency assignment inside a subregion in the search space.

**Definition:** Robustness is defined as the shortest distance to the border of the feasibility region, i.e., the region within which all control applications are guaranteed to be stable.

According to the above definition, to quantify robustness, the idea is to find the largest inscribed ball (also can be extended to maximum volume inscribed ellipsoid) around each operating point. The robustness of different operating points can then be compared based on the volumes of the inscribed balls. Such a definition identifies the maximum distance from the operating point in any direction which still satisfies the requirements. Therefore, the larger the inscribed ball, the more robust the operating point.

For each control application  $\Lambda_i$ , the stability condition can be formulated using a simple inequality of the form (see

<sup>1</sup>Note that the framework proposed in [21] also supports sensitivity analysis of response-time jitter with respect to periods, but the authors do not consider the stability issues. The monotonicity problem has been pointed out in [20] and is addressed by setting the best-case response times of the tasks to their best-case execution times.

Section III),

$$\underline{L}_i + \alpha_i \overline{\Delta}_i^w \leq \beta_i. \quad (9)$$

#### A. One-Dimensional Sensitivity Analysis

Let us first address the one-dimensional sensitivity analysis problem where control application  $\Lambda_i$  is considered to be the application under analysis. Therefore, the sensitivity analysis is performed with respect to the sampling frequency of application  $\Lambda_i$ , i.e.,  $f_i$ . The frequencies of all other applications are considered to be fixed as one-dimensional sensitivity analysis is considered. The dependencies of coefficients  $\alpha_i$  and  $\beta_i$  on frequency  $f_i$  are captured by linear functions:  $\alpha_i = \alpha_{i1} f_i + \alpha_{i2}$  and  $\beta_i = \beta_{i1} f_i + \beta_{i2}$  (see Section III).

Given an operating point  $\mathbf{f}^0 = (f_1^0, f_2^0, \dots, f_{|\Lambda|}^0)$ , the objective is to find the largest symmetric interval for frequency  $f_i$ , around the operating point  $\mathbf{f}^0$ , within which all control applications are stable. Observe that the maximum sampling frequency of application  $\Lambda_i$  is bounded above by the utilization of the processor,

$$f_i \leq \frac{1 - \sum_{\tau_j \in \Lambda \setminus \{\Lambda_i\}} u_j^w}{c_i^w}. \quad (10)$$

The impact of varying the sampling frequency of application  $\Lambda_i$  can be considered in two cases: (1) on its own stability, and (2) on the stability of other applications. Let us first formulate the stability constraint for the application under analysis,  $\Lambda_i$ . Since one-dimensional sensitivity analysis is considered, only  $\alpha_i$  and  $\beta_i$  depend on  $f_i$  in the stability condition 9, and therefore the constraint is linear with respect to sampling frequency  $f_i$ ,

$$k_i^{(1)} f_i + k_i^{(2)} \leq 0, \quad (11)$$

where  $k_i^{(j)}$  are constant coefficients.

Having considered the stability constraint for the application under analysis, we proceed with formulating the stability constraints for the low priority applications, denoted by  $lp(\Lambda_i)$ , since the variation in the sampling frequency of application  $\Lambda_i$  has no impact on the stability of its higher priority applications. For control application  $\Lambda_j \in lp(\Lambda_i)$ , the stability condition can be written according to the stability condition 9. However, in this inequality  $\alpha_j$  and  $\beta_j$  have constant values as the sampling frequency of application  $\Lambda_j$  remains unchanged, and therefore, the inequality has the following form (we skip the elementary algebra),

$$\min \left\{ k_j^{(1)} f_i^2 + k_j^{(2)} f_i + k_j^{(3)}, k_j^{(4)} f_i^2 + k_j^{(5)} f_i + k_j^{(6)} \right\} \leq 0, \quad (12)$$

where the minimum of two single variable quadratic functions should be less than or equal to zero.

Observe that the distance to the border of the stability region given by the constraints in inequalities 10, 11, and 12 can be found efficiently. Thus, let us capture by  $\mathcal{D}(\Lambda_i, \Lambda_j, \mathbf{f}^0)$ , the shortest distance of the operating point  $\mathbf{f}^0$  to the border of the stability region of application  $\Lambda_j \in lp(\Lambda_i) \cup \{\Lambda_i\}$  when the frequency of application  $\Lambda_i$  is subject to change. The minimum distance to the stability border is then given by,

$$r = \min_{\Lambda_j \in lp(\Lambda_i) \cup \{\Lambda_i\}} \{ \mathcal{D}(\Lambda_i, \Lambda_j, \mathbf{f}^0) \}.$$

### B. Towards Multi-Dimensional Sensitivity Analysis

Given an operating point  $\mathbf{f}^0 = (f_1^0, f_2^0, \dots, f_{|\Lambda|}^0)$ , the objective is to find the largest inscribed ball in the space of frequencies, centered at the operating point  $\mathbf{f}^0$ , within which all control applications remain stable. As opposed to the one-dimensional case, considering the dependency of both  $\alpha_i$  and  $\beta_i$  on the sampling frequency  $f_i$  in the stability condition 9 leads to a nonlinear problem, which in general is hard to solve optimally. Therefore, we proceed with the constant, but safe, values of  $\alpha_i$  and  $\beta_i$ .<sup>2</sup> In addition, the vector  $\mathbf{c}$  is the execution-time vector of all tasks, i.e.,  $\mathbf{c} = (c_1, c_2, \dots, c_{|\Lambda|})$ , where  $c_i$  captures the execution-time of task  $\tau_i$ . The stability constraint 9 can be considered with slight modification of the nominal delay and response-time jitter and can be written as follows (Thanks to the simplifications and optimization approach used, the monotonicity property is not required in this subsection),

$$\frac{c_i + (2\alpha_i - 1) \sum_{\tau_j \in hp(\Lambda_i)} c_j (1 - u_j)}{1 - \sum_{\tau_j \in hp(\Lambda_i)} u_j} \leq \beta_i. \quad (13)$$

The above inequality can be reformulated as follows by multiplying both sides by  $(1 - \sum_{\tau_j \in hp(\Lambda_i)} u_j) > 0$  and taking into account that the utilization for control application  $\Lambda_i$  is defined as  $u_i = c_i f_i$ ,

$$\begin{aligned} \sum_{\tau_j \in hp(\Lambda_i)} (\beta_i - (2\alpha_i - 1)c_j) c_j f_j &\leq \\ \beta_i - (2\alpha_i - 1) \sum_{\tau_j \in hp(\Lambda_i)} c_j - c_i. \end{aligned} \quad (14)$$

Let us reformulate the stability constraint for control application  $\Lambda_i$  (inequality 14), in the multi-dimensional case, as follows,

$$\mathbf{a}_i \cdot \mathbf{f} \leq b_i, \quad (15)$$

where  $\mathbf{f} = (f_1, f_2, \dots, f_{|\Lambda|})$  is the vector of frequencies and  $\mathbf{a}_i$  and  $b_i$  are a constant vector and a constant scalar, respectively.

Having discussed the stability constraints, we can proceed with formulating the multi-dimensional sensitivity analysis as follows (Chebyshev center of a polyhedron [7]),

$$\begin{aligned} \max_{r, \mathbf{f}} \quad & r \\ \text{s.t.} \quad & \mathbf{a}_i \cdot \mathbf{f} + r \|\mathbf{a}_i\| \leq b_i, \quad \forall \Lambda_i \in \Lambda \\ & \mathbf{c} \cdot \mathbf{f} + r \|\mathbf{c}\| \leq 1, \\ & \mathbf{f} = \mathbf{f}^0, \end{aligned} \quad (16)$$

where the first constraint is the stability constraint, while the second constraint ensures that the utilization is less than one. The value of  $r$  captures the radius of the largest inscribed ball, centered at  $\mathbf{f}^0$ , and can be considered as a measure of robustness (or equivalently sensitivity). The larger the radius  $r$ , the less sensitive the operating point  $\mathbf{f}^0$ . Since the only actual variable in the above problem formulation is radius  $r$ , the problem boils down to the following simple equation,

$$r = \min \left\{ \min_{\Lambda_i} \left\{ \frac{b_i - \mathbf{a}_i \cdot \mathbf{f}^0}{\|\mathbf{a}_i\|} \right\}, \frac{1 - \mathbf{c} \cdot \mathbf{f}^0}{\|\mathbf{c}\|} \right\}.$$

<sup>2</sup>Observe that this may not be a severe restriction for the sensitivity analysis as it is often performed locally and  $\alpha_i$  and  $\beta_i$  coefficients are subject to minor changes.

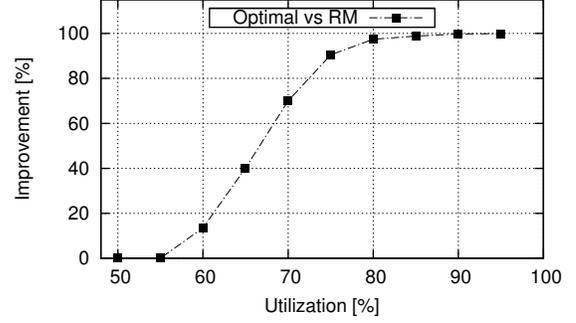


Figure 4. Percentage of benchmarks for which the solution found by the rate monotonic priority assignment is not guaranteed to be stable (100% are stable with the proposed approach)

In addition to sensitivity analysis, it is also possible to solve the following problem: given a subregion in the space of task frequencies, the objective is to identify the operating point which is farthest from the exterior of the search space, i.e., the most robust (least sensitive) operating point. The formulation is similar to the problem formulated in the previous section (problem formulation 16), but the operating point is not set to be fixed. This can be obtained by removing the equality constraint in the previous problem formulation (problem formulation 16). Observe that the problem of finding the most robust operating point is a linear programming problem and can be solved efficiently using the existing toolboxes [7].

The multi-dimensional sensitivity analysis problem inherently suffers from exponential computational complexity as the number of dimensions grows since, in general, the feasibility frontier is not known a priori. However, in our formulation, the feasibility region is given by a set of linear and convex inequalities and increasing the dimension corresponds to adding a finite number of inequalities which does not increase the complexity of the problem exponentially. Therefore, scalability is not an issue for the problems addressed in this section.

## IX. EXPERIMENTAL RESULTS

In this section, the priority assignment algorithm and the sensitivity analysis approach are evaluated.

### A. Priority Assignment

To investigate the efficiency of our proposed approach, we have compared our optimal priority assignment algorithm against the rate monotonic priority assignment (RM), i.e., the higher the sampling rate, the higher the priority. For a set of 1000 benchmarks with 8 control applications and plants taken from [3] and [9], the experiments are repeated for different values of processor utilization. The results are shown in Figure 4 where the percentage of benchmarks for which the rate monotonic priority assignment ends up with an unstable design solution is given as a function of processor utilization. It should be noted that our algorithm could find, at least, a stable design solution for all the benchmarks. It can be seen that the percentage of cases in which the rate monotonic priority assignment fails increases drastically with utilization of the processor, while it is zero for low utilization.

### B. Sampling Frequency Sensitivity Analysis

In this section, we consider a small example comprising two control applications  $\Lambda = \{\Lambda_1, \Lambda_2\}$ , modeled by tasks  $\tau_1 = (\rho_1 = 2, c_1^w = 11, c_1^b = 11)$  and  $\tau_2 = (1, 20, 20)$ . The periods are removed from the list of tasks parameters since they are subject to change for different design solutions. The coefficients  $\alpha_i$  and  $\beta_i$  for each control application  $\Lambda_i$  are bounded by constant values. Similar to the previous example, we have  $\Lambda_1 = (\alpha_1 = 1.18, \beta_1 = 72)$  and  $\Lambda_2 = (1.22, 143)$ . We are given two operating points in the feasible region and the objective is to identify the one that is more desirable from the robustness perspective.

As for the first operating point, consider the periods to be  $h_1 = 13.2$  and  $h_2 = 150$  ( $f_1 = 7575$  and  $f_2 = 667$ ). The sensitivity analysis result is shown in Figure 5(a). The solid lines (borders) in Figure 5(a) identify the maximum and minimum frequencies allowed. The dashed line is the stability line, while the dash-dot line depicts the utilization criterion. The shaded area captures the stable region. The radius of the ball around this point is equal to 92 and the limiting factor is the stability bound. It should be noted that the stability line restricts the maximum frequency of the higher priority task  $\tau_1$  to guarantee the stability of the lower priority task  $\tau_2$  by limiting the amount of delay and jitter task  $\tau_2$  experiences.

Let us consider another operating point, where the periods are  $h_1 = 13.5$  and  $h_2 = 140$  ( $f_1 = 7407$  and  $f_2 = 714$ ). The result of analysis is shown in Figure 5(b). The radius of the sensitivity ball is 185 which is twice as large as the first operating point. This clearly states the robustness of the second operating point compared to the first. Note that for this operating point the limiting factor is the utilization bound.

Given a subregion in the search space, the most robust operating point can also be identified according to the optimization problem formulated in 16. In this example, we consider the whole search space to be the region inside which we wish to find the most robust operating point. The most robust point is obtained for periods  $h_1 = 13.6$  and  $h_2 = 180$  ( $f_1 = 7306$  and  $f_2 = 563$ ) using the optimization toolbox in MATLAB. The optimization result is shown in Figure 5(c) and the radius of the ball is 361.

While the approach can be applied to larger systems, we have illustrated our approach with a small example of two control applications since this is easy to illustrate graphically.

## X. CONCLUSIONS

As opposed to hard real-time applications where meeting the deadlines is the basis of most analyses, control applications often do not enforce hard deadlines. In the case of control applications, stability should be regarded as the acceptance criterion, and not the hard deadlines. It turns out that stability depends on not only the response-time delay, but also the response-time jitter experienced by a control application. This fundamental difference between control and hard real-time applications requires new design and analysis approaches. We consider the two main metrics, i.e., the nominal delay and response-time jitter, to bridge the existing gap between the real-time and control areas. The properties of the two metrics are discussed and the metrics are redefined to have the monotonicity property. Finally, we have addressed the priority

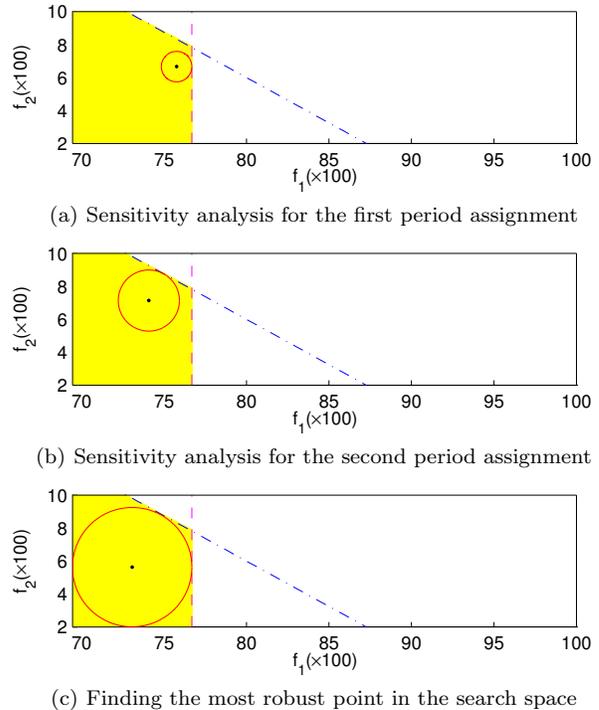


Figure 5. An example for sensitivity analysis assignment and sensitivity analysis problems with respect to stability of control applications.

## REFERENCES

- [1] A. Aminifar, S. Samii, P. Eles, Z. Peng, and A. Cervin. Designing high-quality embedded control systems with guaranteed stability. In *Proceedings of the 33<sup>th</sup> IEEE Real-Time Systems Symposium*, pages 283–292, 2012.
- [2] K. E. Årzén and A. Cervin. Control and embedded computing: Survey of research directions. In *Proceedings of the 16<sup>th</sup> IFAC World Congress*, 2005.
- [3] K. J. Åström and B. Wittenmark. *Computer-Controlled Systems*. Prentice Hall, 3 edition, 1997.
- [4] N. C. Audsley. Optimal priority assignment and feasibility of static priority tasks with arbitrary start times. Technical Report YCS 164, Department of Computer Science, University of York, December 1991.
- [5] E. Bini, M. Di Natale, and G. Buttazzo. Sensitivity analysis for fixed-priority real-time systems. *Real-Time System*, 39(1-3):5–30, 2008.
- [6] E. Bini, T. Huyen Châu Nguyen, P. Richard, and S. K. Baruah. A response-time bound in fixed-priority scheduling with arbitrary deadlines. *IEEE Transactions on Computer*, 58(2):279–286, 2009.
- [7] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.
- [8] A. Cervin. Stability and worst-case performance analysis of sampled-data control systems with input and output jitter. In *Proceedings of the 2012 American Control Conference (ACC)*, 2012.
- [9] A. Cervin, B. Lincoln, J. Eker, K. E. Årzén, and G. Buttazzo. The jitter margin and its application in the design of real-time control systems. In *Proceedings of the 10<sup>th</sup> International Conference on Real-Time and Embedded Computing Systems and Applications*, 2004.

- [10] F. Cottet and J.-P. Babau. An iterative method of task temporal parameter adjustment in hard real-time systems. In *Engineering of Complex Computer Systems, 1996. Proceedings., Second IEEE International Conference on*, pages 103–106, 1996.
- [11] R. Davis and A. Burns. Robust priority assignment for fixed priority real-time systems. In *Proceedings of the 28<sup>th</sup> IEEE Real-Time Systems Symposium*, pages 3–14, 2007.
- [12] M. Joseph and P. Pandya. Finding response times in a real-time system. *The Computer Journal*, 29(5):390–395, 1986.
- [13] C.-Y. Kao and B. Lincoln. Simple stability criteria for systems with time-varying delays. *Automatica*, 40:1429–1434, 2004.
- [14] J. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *Proceedings of the 11<sup>th</sup> IEEE Real-Time Systems Symposium*, pages 201–209, 1990.
- [15] J. Y. T. Leung and J. Whitehead. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance Evaluation*, 2(4):237–250, 1982.
- [16] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):47–61, 1973.
- [17] M. G. Mancuso, E. Bini, and G. Pannocchia. A framework for optimal priority assignment of control tasks. *Leibniz Transactions on Embedded Systems*, 2013.
- [18] L. Palopoli, C. Pinello, A. Bicchi, and A. Sangiovanni-Vincentelli. Maximizing the stability radius of a set of systems under real-time scheduling constraints. *Automatic Control, IEEE Transactions on*, 50(11):1790–1795, 2005.
- [19] S. Punnekkat, R. Davis, and A. Burns. Sensitivity analysis of real-time task sets. In *Asian Computing Science Conference, LNCS 1345*, pages 72–82, 1997.
- [20] R. Racu. Performance characterization and sensitivity analysis of real-time embedded systems. Technical report, Technical University of Braunschweig, 2008.
- [21] R. Racu, A. Hamann, and R. Ernst. Sensitivity analysis of complex embedded real-time systems. *Real-Time Systems*, 39:31–72, 2008.
- [22] O. Redell and M. Sanfridson. Exact best-case response time analysis of fixed priority scheduled tasks. In *Proceedings of the 14<sup>th</sup> Euromicro Conference on Real-Time Systems*, pages 165–172, 2002.
- [23] O. Serlin. Scheduling of time critical processes. In *Proceedings of AFIPS Spring Computing Conference*, pages 925–932, 1972.
- [24] K. Tindell, A. Burns, and A. J. Wellings. An extendible approach for analyzing fixed priority hard real-time tasks. *Real-Time Systems*, 6(2):133–151, 1994.
- [25] B. Wittenmark, J. Nilsson, and M. Törngren. Timing problems in real-time control systems. In *Proceedings of the American Control Conference*, pages 2000–2004, 1995.

#### APPENDIX A

Let us consider a set of independent tasks, running on a uniprocessor under fixed-priority preemptive scheduling. Under these assumptions, the best-case response time can be computed by Equation 6. The best-case phasing of the task under analysis occurs when it finishes simultaneously with the release of all its higher priority tasks [22] (therefore, Figure 6 is mirrored). Similar to [6], we develop a lower bound for the best-case response time (Equation 6). Note that since

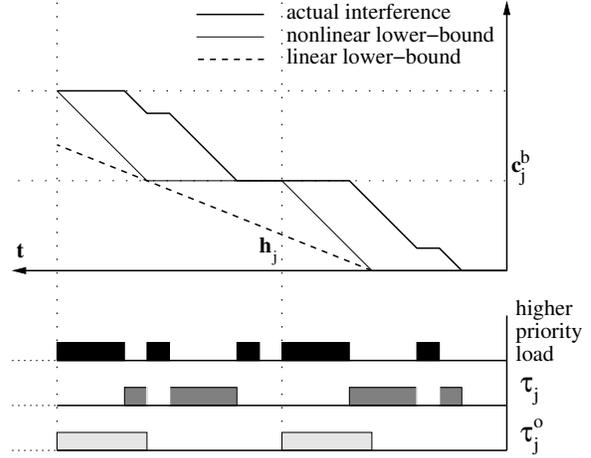


Figure 6. The lower bounds on the actual best-case workload Equation 6 produces a valid lower bound for task sets with arbitrary deadlines, the lower bound developed here is also valid for arbitrary deadlines.

Let us denote the best-case workload of the higher priority tasks for task  $\tau_i$  over an interval of  $t$  by  $B_i(t)$ . Then, we can define the best-case idle time in an interval of  $t$  as follows,

$$H_i(t) = t - B_i(t).$$

Considering these definitions, the best-case response time of task  $\tau_i$  is,

$$R_i^b(c_i^b) = \min_t \{t | H_i(t) \geq c_i^b\}.$$

It is obvious that considering a lower bound for the best-case workload, denoted by  $\underline{B}_i(t)$ , leads to a lower bound for the best-case response time as shown in the following,

$$\begin{aligned} \underline{B}_i(t) &\leq B_i(t), \\ \overline{H}_i(t) &= t - \underline{B}_i(t) \geq t - B_i(t) = H_i(t), \\ \underline{R}_i^b(c_i^b) &= \min_t \{t | \overline{H}_i(t) \geq c_i^b\} \leq \min_t \{t | H_i(t) \geq c_i^b\} = R_i^b(c_i^b), \end{aligned}$$

where the overline and underline indicate upper bound and lower bound, respectively.

The last step is to identify  $\underline{B}_i(t)$ . Let us consider a higher priority task  $\tau_j$ , and denote the best-case amount of its interference in an interval of  $t$  by  $b_j(t)$ . The actual interference experienced by task  $\tau_i$  due to higher priority task  $\tau_j$  is shown in Figure 6. Further, Figure 6 also depicts the nonlinear  $b_j^o(t)$  and linear  $\underline{b}_j(t)$  lower bounds on the actual amount of interference caused by task  $\tau_j$ . It should be mentioned that the nonlinear lower bound  $b_j^o(t)$  occurs when task  $\tau_j$  is the only higher priority task for task  $\tau_i$ . Considering these bounds, we have the following inequalities,

$$b_j(t) \geq b_j^o(t) \geq \underline{b}_j(t) = tu_j^b + c_j^b(u_j^b - 1).$$

Using these relationships, we sum the values of  $\underline{b}_j(t)$  for all higher priority tasks to obtain  $\underline{B}_i(t)$  as follows,

$$\begin{aligned} B_i(t) &= \sum_{\tau_j \in hp(\tau_i)} b_j(t) \geq \sum_{\tau_j \in hp(\tau_i)} b_j^o(t) \\ &\geq \sum_{\tau_j \in hp(\tau_i)} \underline{b}_j(t) \\ &= \sum_{\tau_j \in hp(\tau_i)} tu_j^b + c_j^b(u_j^b - 1) = \underline{B}_i(t). \end{aligned}$$

Since  $\overline{H}_i(t)$  is a one-to-one function, it is safe to require  $\overline{H}_i(t) = t - \underline{B}_i(t) = c_i^b$ . Therefore, by substituting  $\underline{B}_i(t)$  and taking into consideration that  $t = \underline{R}_i^b(c_i^b)$ , we obtain,

$$\underline{R}_i^b(c_i^b) = \frac{c_i^b - \sum_{\tau_j \in hp(\tau_i)} c_j^b (1 - u_j^b)}{1 - \sum_{\tau_j \in hp(\tau_i)} u_j^b}. \quad (17)$$

## APPENDIX B

In this section, we will discuss the issues related to the commutativity and monotonicity properties of Equation 8.

### Commutativity and Monotonicity with Respect to Priorities:

For the commutativity property, observe that all equations related to Equation 8 depend on the *set* of higher priority tasks and therefore the priority order of other tasks is not important as long as the set of high priority tasks remains the same.

Let us now investigate the monotonicity property. The monotonicity of the nominal delay in the case of priority assignment is clear since removing a high priority task,  $\tau_k \in hp(\tau_i)$ , leads to less or equal interference in the best-case response-time scenario of the task under analysis, i.e.,  $\tau_i$ .

Let us now discuss the monotonicity of response-time jitter with respect to priorities. Observe that for the rest of this section we consider  $\underline{R}_i^b$  is defined as in Equation 17, for simplicity of presentation. The claim is that the response-time jitter for task  $\tau_i$  decreases once a higher priority task  $\tau_k \in hp(\tau_i)$  is removed from the set of higher priority tasks  $hp(\tau_i)$  which is shown in the following,

$$\begin{aligned} \overline{\Delta}_{i+}^w &\geq \overline{\Delta}_{i-}^w. \\ \overline{R}_{i+}^w - \max\{\underline{R}_{i+}^b, c_i^b\} &\geq \overline{R}_{i-}^w - \max\{\underline{R}_{i-}^b, c_i^b\}, \\ \overline{R}_{i+}^w - \overline{R}_{i-}^w &\geq \max\{\underline{R}_{i+}^b, c_i^b\} - \max\{\underline{R}_{i-}^b, c_i^b\}, \end{aligned} \quad (18)$$

where the plus sign in the index indicates the metric before removing the higher priority task, and the minus sign indicates the metric after removing the higher task.

Before proceeding with the proofs, we shall define the following (positive) variables,

$$\begin{aligned} X_i^b &= 1 - \sum_{\tau_j \in hp(\tau_i)} u_j^b, & X_i^w &= 1 - \sum_{\tau_j \in hp(\tau_i)} u_j^w, \\ Y_i^b &= \sum_{\tau_j \in hp(\tau_i)} c_j^b (1 - u_j^b), & Y_i^w &= \sum_{\tau_j \in hp(\tau_i)} c_j^w (1 - u_j^w). \end{aligned}$$

Let us investigate the monotonicity property of the response-time jitter in the following four cases,

- $\underline{R}_{i+}^b \geq c_i^b$ ,  $\underline{R}_{i-}^b \geq c_i^b$ : calculating both sides of the last inequality (inequality 18),

$$\begin{aligned} \underline{R}_{i+}^b - \underline{R}_{i-}^b &= -\frac{X_i^b(c_k^b(1 - u_k^b))}{(X_i^b + u_k^b)X_i^b} + \frac{u_k^b c_i^b}{(X_i^b + u_k^b)X_i^b} - \frac{u_k^b Y_i^b}{(X_i^b + u_k^b)X_i^b}, \\ \overline{R}_{i+}^w - \overline{R}_{i-}^w &= \frac{X_i^w(c_k^w(1 - u_k^w))}{(X_i^w + u_k^w)X_i^w} + \frac{u_k^w c_i^w}{(X_i^w + u_k^w)X_i^w} + \frac{u_k^w Y_i^w}{(X_i^w + u_k^w)X_i^w}. \end{aligned}$$

Comparing  $\underline{R}_{i+}^b - \underline{R}_{i-}^b$  and  $\overline{R}_{i+}^w - \overline{R}_{i-}^w$  term by term, it is clear that the monotonicity property holds for response-time jitter in this case (i.e., inequality 18 holds).

- $\underline{R}_{i+}^b \geq c_i^b$ ,  $\underline{R}_{i-}^b \leq c_i^b$ : in this case we have  $\underline{R}_{i+}^b - \underline{R}_{i-}^b \geq \underline{R}_{i+}^b - c_i^b$  and from the first case we know  $\overline{R}_{i+}^w - \overline{R}_{i-}^w \geq \underline{R}_{i+}^b - \underline{R}_{i-}^b$ . Therefore, we have  $\overline{R}_{i+}^w - \overline{R}_{i-}^w \geq \underline{R}_{i+}^b - c_i^b$ .
- $\underline{R}_{i+}^b \leq c_i^b$ ,  $\underline{R}_{i-}^b \leq c_i^b$ : from the previous equation in the first case considered it is clear that  $\overline{R}_{i+}^w - \overline{R}_{i-}^w$  is always positive and therefore it follows  $\overline{R}_{i+}^w - \overline{R}_{i-}^w \geq c_i^b - c_i^b = 0$ .

- $\underline{R}_{i+}^b \leq c_i^b$ ,  $\underline{R}_{i-}^b \geq c_i^b$ : since the value of  $\overline{R}_{i+}^w - \overline{R}_{i-}^w$  is always positive, we have  $\overline{R}_{i+}^w - \overline{R}_{i-}^w \geq 0 \geq c_i^b - \underline{R}_{i-}^b$ .

Therefore, inequality 18 holds.

### Monotonicity with Respect to Periods:

Let us now proceed with proving the monotonicity of the nominal delay with respect to sampling frequency. The proof is by computing the derivative of the nominal delay  $\underline{L}_i$  with respect to sampling frequency (or alternatively the utilization) of higher priority task  $\tau_k \in hp(\tau_i)$ . In this case, we have,

$$\frac{\partial \underline{R}_i^b}{\partial u_k^b} = \frac{c_k^b X_i^b + c_i^b - Y_i^b}{X_i^{b2}}.$$

Observe that under the assumption of  $c_i^b - Y_i^b \geq 0$ , the derivative is always positive. Note that if  $\underline{L}_i = \underline{R}_i^b$ , we know  $\underline{R}_i^b = \frac{c_i^b - Y_i^b}{X_i^b} \geq c_i^b \geq 0$  and  $c_i^b - Y_i^b \geq 0$  and hence the above derivative is positive; otherwise,  $\underline{L}_i = c_i^b$  with derivative equal to zero. Considering the continuity of  $\underline{L}_i$ , the nominal delay, defined as  $\underline{L}_i = \max\{\underline{R}_i^b, c_i^b\}$ , monotonically changes with sampling frequencies.

To prove the monotonicity of the response-time jitter with respect to sampling frequencies we use similar techniques. Two cases need to be considered,

- Having  $\underline{R}_i^b \geq c_i^b$ , the response-time jitter is given by  $\overline{\Delta}_i^w = \overline{R}_i^w - \underline{R}_i^b$  and the proof is as follows,

$$\begin{aligned} \frac{\partial \overline{\Delta}_i^w}{\partial f_k} &= \frac{\partial \overline{R}_i^w}{\partial f_k} - \frac{\partial \underline{R}_i^b}{\partial f_k} \\ &= \frac{-c_k^{w2} X_i^w + c_k^w(c_i^w + Y_i^w)}{X_i^{w2}} - \frac{c_k^{b2} X_i^b + c_k^b(c_i^b - Y_i^b)}{X_i^{b2}} \\ &= \frac{-c_k^{w2} X_i^w + c_k^{w2}(1 - c_k^w f_k)}{X_i^{w2}} + \frac{-c_k^{b2} X_i^b + c_k^{b2}(1 - c_k^b f_k)}{X_i^{b2}} \\ &\quad + \frac{c_k^w Y_i^w - c_k^{w2}(1 - c_k^w f_k)}{X_i^{w2}} + \frac{c_k^b Y_i^b - c_k^{b2}(1 - c_k^b f_k)}{X_i^{b2}} \\ &\quad + \frac{c_k^w c_i^w}{X_i^{w2}} - \frac{c_k^b c_i^b}{X_i^{b2}}. \end{aligned}$$

It is easy to show that the first four terms in the above equation are positive. Also, the sum of the fifth and the sixth terms is positive, and therefore, the total value is positive.

- In the case where we have  $\underline{R}_i^b \leq c_i^b$ , the response-time jitter is given by  $\overline{\Delta}_i^w = \overline{R}_i^w - c_i^b$  and its derivative is obtained as follows,

$$\begin{aligned} \frac{\partial \overline{\Delta}_i^w}{\partial f_k} &= \frac{\partial \overline{R}_i^w}{\partial f_k} - 0 \\ &= \frac{-c_k^{w2} X_i^w + c_k^w(c_i^w + Y_i^w)}{X_i^{w2}} - 0 \\ &= \frac{-c_k^{w2} X_i^w + c_k^{w2}(1 - c_k^w f_k)}{X_i^{w2}} \\ &\quad + \frac{c_k^w Y_i^w - c_k^{w2}(1 - c_k^w f_k)}{X_i^{w2}} \\ &\quad + \frac{c_k^w c_i^w}{X_i^{w2}}. \end{aligned}$$

Observe that all terms in the above equation are positive which indicates the monotonicity of the response-time jitter in this case.

Considering the continuity of  $\overline{\Delta}_i^w$ , the worst-case response-time jitter, defined as  $\overline{\Delta}_i^w = \overline{R}_i^w - \max\{\underline{R}_i^b, c_i^b\}$ , monotonically changes with sampling frequency.