Fast and Tight Analysis for AUTOSAR Schedule Tables

Pierre-Emmanuel Hladik

Fast and Tight Analysis for AUTOSAR Schedule Tables

—
Pierre-Emmanuel Hladik (pehladik@laas.fr)
—
CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France
—
The main parts of this report were produced in 2010,
(re)implemented in 2014 and published the
February 12, 2016
—

Abstract

AUTOSAR is the name of a consortium that defines a set of industrial standards for the design of in-vehicle embedded systems. The AUTOSAR consortium promotes a model-based approach, supported by tools that are expected to automatically transform a functional design into a real-time multitask software. Before to consider the design of such tools, one must be able to verify the real-time behavior of a multitask software. This report presents an extension of a previous work on a mathematical modeling framework for AUTOSAR applications and a worst-case response time analysis [7, 8, 9] published with Anne-Marie Déplanche, Sébastien Faucou and Yvon Trinquet. Three new algorithms are proposed, implemented and evaluated to improve performance of the analysis.

Acknowledgements

Special thanks goes to Oussama El Fatayri, who help me to implement and to experiment.

Contents

1 Introduction

AUTOSAR (AUTomotive Open System ARchitecture) is a consortium composed of the main actors of the automotive domain. It publishes a family of standards specifying a reference architecture for modern in-vehicle embedded systems. It promotes a tool-assisted model-based approach to system design. One of the tool is expected to transform a set of interconnected software components into a real-time multitask software. To achieve its duty, this tools must be able to verify the real-time behavior of a multitask software built on top of an AUTOSAR OS compliant real-time kernel.

At first sight, AUTOSAR OS is just another fixed priority preemptive real-time kernel. Thus, one can predict the real-time behavior of an AUTOSAR software by using state-of-the-art algorithms such as presented in the reference book of Giorgio Buttazo [5]. Actually, AUTOSAR OS allows to design rather complex multitask softwares. To the best of our knowledge, no algorithm has been published that allows to compute the response times of jobs in such a software without performing pessimistic over-approximations. In this paper, we build such an algorithm, and we prove its correctness. We focus on the possibility to take into account most of the features of AUTOSAR OS so as to minimize the pessimism of the results.

This report extends previous work [7, 8, 9] published with Anne-Marie Déplanche, Sébastien Faucou and Yvon Trinquet.

The paper is organized as follow: in section 2, the related works is presented; in Section 3, the scheduling services of AUTOSAR OS are described; in Section 4, the modeling framework is introduced; the Section 5 exposes how to compute upper bounds on the response time of the tasks of a system; Sections 6, 7 and 8 propose some improvements to speed up the analysis; in Section 9, the method is expremiented; in Section 10, the paper is concluded.

2 Related works

This report addresses the following (pragmatic) problem: how to compute reasonable bounds on the response times of jobs spawned by a complex multitask software executing on top of an AUTOSAR OS compliant kernel. Other authors did study similar problems, for instance [12], [4] and [8] expose how to analyze OSEK/VDX OS applications. OSEK/VDX OS is the ancestor of AUTOSAR OS.

Thus, the existing algorithms has to be updated to take into account the features introduced in AUTOSAR OS. The main novelty is the introduction of schedule tables, which leads to a major update. Let us underline that in [4], the overhead of kernel activities are taken into account and this dimension is not explored here.

The RTA (Response Time Analysis) approach is followed. This approach was developed in the mid 80's by Joseph and Pandya [10] and improved by Audsley [1]. In the following years and up to now, many extensions and improvements have been proposed. Concerning our work, the most important extensions are the concept of asynchronous transaction introduced in [18]; the generalization of the concept of preemption threshold proposed in [19]; the study of systems scheduled FPP/FIFO (Fixed Priority Preemptive as a primary criterion / First In First Out policy as a secondary criterion) [4].

The main difficulty of this problem comes from the complexity of the models. Two dimensions are affected. First, it is difficult to express and prove an algorithm that takes into account all the features of the target model, even if an algorithm is known for each feature on its own. Second, the complexity of the model impacts the time complexity of the algorithm. Many solutions have been proposed to alleviate this problem, based on the elimination of useless computations [6] and/or the trading of time complexity for pessimism.

In this paper, we put the pieces together in order to take into account the AUTOSAR scheduling policy for complex systems.

This theoretical problem was studied because of a very pragmatic concern: the AUTOSAR OS standards includes these FPP/FIFO and asynchronous transactions. Our original goal is to provide the AUTOSAR OS users with a modeling framework to capture real-time multitask software architectures and a set of algorithms to analyze these architectures. This same goal has been followed and reported in previous publications. Concerning AUTOSAR OS, early results have been reported in [9]. However, they do not offer a satisfactory support for Schedule Tables (the name of the AUTOSAR OS concept related to asynchronous transaction).

Our contribution. The main contribution of this document is the availability of a modeling framework and analysis method for AUTOSAR OS based real-time multitask software. We compile and mix the existing results and show that we can have an scheduling analysis for FPP with asynchronous transactions. Comparatively to previous work this report is focused on new implementations of the schedulability analysis to improve computation performances.

3 Scheduling policy of AUTOSAR OS

The AUTOSAR OS standard [2] describes the interface and behavior of the operating system component of an AUTOSAR ECU. It is a real-time operating system providing a limited amount of services : scheduling, synchronization, IRQ handling, signaling, message passing, alarms handling and partitioning . Each service can be manipulated through a set of compile-time primitives and a set of run-time primitives. Given the aim of our work, the following of the presentation focuses on the subset of the standard that must be introduced to understand the modeling framework exposed in section 4. To be as clear as possible, this presentation uses some vocabulary from the object oriented modeling domain. Let us underline that AUTOSAR OS is not an objet oriented operating system: its interface is defined in C language.

Active objects AUTOSAR OS supports two abstract classes of active objects: tasks and OsIsr (Operating system Interrupt service routine).

The task class has two specialization: BasicTask (BT) and ExtendedTask (ET). Once created, a BT instance is either in the Ready state or in the Running state (it is allocated to the CPU) until it terminates. A compile-time primitive allows to bound the number of instances of a BT existing at the same time (subsequent creation requests are dropped). An ET instance can be either in the Ready state, in the Running state, or in the Waiting state (waiting for an event to be signaled). There can be at most one instance of an ET in the system at a given date.

The OsIsr class has also two specialization: OsIsr Category 1 (OsIsr1) and OsIsr Category 2 (OsIsr2). OsIsr1 are out of control for the operating system, whereas OsIsr2 are not. Therefore, we focus on system without OsIsr1. From the point-of-view of the scheduling service, OsIsr2 instances are similar to BT instances. The difference between the two lies mainly in the source of the instance creation requests.

We introduce now the concept of job. Although it is not explicitly defined in the standard, it allows us to offer a unified view of ET, BT and OsIsr2 instances (notice that our definition is consistent with the behavior of the timing protection service described in the standard). A job is a non-blocking activity that is performed by the CPU. The scheduling service arbitrates the distribution of the CPU between concurrent jobs. Jobs are generated by ET, BT and OsIsr2 instances according to the following rules:

- Each instance of a BT or OsIsr2 generates exactly one job. The job terminates with the instance.
- An instance of an ET generates a job when it is created and each time it leaves the Waiting state. The job terminates either with the instance, or the next time it enters the Waiting state.

Each job is characterized by two scheduling parameters: its base priority (constant, configured at compile-time, attached to the task or OsIsr2 generating the job) and its current priority (dynamic, attached to the job). For the sake of simplicity, we will name from now the base priority “priority", and the current priority “preemption threshold”. When a job is created, its preemption threshold equals its priority. The range of priority used for OsIsr2 is above the range of priority used for tasks.

Resources AUTOSAR OS provides a Resource (abstract) class. Its instances (named resource from now) are created at compile-time and are used to control the access to mutually exclusive critical sections in a multitask software. Competitors for a resource can be any job generated by an ET, an BT or even an OsIsr2 instance. Each resource has a constant ceiling priority configured through a compile-time primitive. The synchronization protocol used by AUTOSAR OS is similar to Baker's SRP [3] in fixed priority context, with the restriction that the capacity of each resource is 1. When a job takes a resource, its preemption threshold is raised to the ceiling priority of the resource. When it releases a resource, its preemption threshold recovers its previous value. The ceiling priority of the resource must be greater than or equal to the preemption threshold of any job likely to take the resource. It must also be as small as possible to avoid undesirable priority inversions.

The Resource class has two realizations: standard resource, the instances of which are manipulated explicitly by the competing jobs through run-time primitives; internal resource, the instances of which are manipulated implicitly: when the job starts its execution (enter the Running state for the first time), it takes the resource and does not release it before it terminates.

A specific resource, named Res_Scheduler is defined by default. Its ceiling priority is the greatest priority used for tasks. Tasks can manipulate Res_Scheduler either as a standard or as an internal resource.

Scheduling policy AUTOSAR OS standards enforce the following scheduling policy: each time the scheduler is invoked, it selects the job with the greatest preemption threshold for running. If two or more jobs have the same preemption threshold, the oldest one is chosen. The scheduler is invoked when: a job is created; the current job terminates; the preemption threshold of the current job decreases (when it releases a standard resource).

Counters, alarms and schedule tables: AUTOSAR OS standards provides the concept of Counter to record recurring events (including real-time clock ticks) and the concept of Alarm to link counters and tasks. An alarm is programmed to expire when the counter reaches a specific value. When it expires, it releases a job (either by activating the task, or signaling an event to it). It is possible to program an alarm to expire one time or to have a cyclic behavior.

The concept of Schedule Table extends the concept of Alarm. A schedule table is programmed to expire when the counter reaches a specific value. This expiry date is used as a reference to measure a set of time offsets , where each offset corresponds to the release of a job. The jobs released by a schedule table can potentially originate from different tasks. It is possible to program a schedule table to expire one time or to have a cyclic behavior.

4 Modeling AUTOSAR OS multitask software

4.1 Modeling framework

A mathematical modeling framework suitable for a wide variety of multitask applications built on top of an AUTOSAR OS compliant kernel is now introduced. So as to be consistent with the terms used in the real-time scheduling literature, we will use from now the term task in its usual sense: an entity that generates a potentially infinite sequence of jobs. We have exposed above how to map this concept onto the concepts of extended task, basic task and OsIsr2 supported by AUTOSAR OS.

We consider a system $\langle T, \Lambda, src \rangle$ composed of a set of n tasks $T = \{\tau_i\}_{1 \leq i \leq n}$, a set of m activation sources $\Lambda = \{\lambda_j\}_{1 \leq j \leq m}$ and a mapping from task indexes to activation sources indexes $src : [1..n] \rightarrow [1..m]$.

Each task is a tuple: $\tau_i = \langle \pi_i, o_i, d_i, E_i \rangle$ where:

- $\pi_i \in \mathbb{N}^+$ is the priority of the task.
- $o_i \in \mathbb{N}$ is the offset of the task, relative to an occurrence of its activation source.
- $d_i \in \mathbb{N}^+$ is the deadline of the task, relative to its activation date.
- $E_i = \{b_{ik} = \langle e_{i,k}, \gamma_{i,k} \rangle\}_{1 \leq k \leq l}$ is the set of l execution blocks composing the tasks, where each block is characterized by its worst-case execution time (WCET), $e_{i,k} \in \mathbb{N}^+$, and its preemption threshold, $\gamma_{i,k} \in [\pi_i, P]$ with P is the greatest priority in the system. The first block is associated to the whole task. Therefore $e_{i,0}$ is the WCET of the task and $\gamma_{i,0}$ is either the priority of the task, or the ceiling priority of an internal resource. Each other block is associated to a critical section in the code of the task protected by a standard resource. Notice that blocks can be only nested or sequential.

Each activation source is a couple: $\lambda_i = \langle p_i, \omega_i \rangle$ where:

- $p_i \in \mathbb{N}^+$ is the minimal inter-occurrence time of the activation source.
- $\omega_i \in \mathbb{N} \cup \{\perp\}$ is the offset of the activation source, relative to the activation date of the system ($\perp$ is used when this offset is unknown at compile-time).

The mapping from tasks to activation sources described by $src(i) = k$ means that τ_i releases a job o_i unit of time after an occurrence of λ_k.

Let call $S(T, \Lambda, src) = \{s_1, s_2, \ldots\}$ the (potentially infinite) set of execution scenario of a system. In each scenario s_k, each task τ_i generates a (potentially infinite) sequence of jobs $\tau_{i,0}^k, \tau_{i,1}^k, \ldots$, where each job $\tau_{i,q}^k$ is characterized by the following values:

- $a_{i,q}(s_k) \in \mathbb{N}$ is the arrival date of $\tau_{i,q}^k$.
- $s_{i,q}(s_k) \in \mathbb{N}$ is the starting date of $\tau_{i,q}^k$ (date where it enters the Running state for the first time).
- $f_{i,q}(s_k) \in \mathbb{N}^*$ is the finishing date of $\tau_{i,q}^k$.

One of the contribution of this work is the support of schedule tables. A schedule table is similar to a transaction as defined by Tindell in [18]. We define $\theta_k = \{i \in 1..n |\, src(i) = k\}$ the set of tasks attached to a same activation source. If $|\theta_k| = 1$ then λ_k is an alarm. If $|\theta_k| > 1$ then λ_k is a schedule table.

4.2 Mapping to AUTOSAR

The proposed framework can model any AUTOSAR OS multitask software meeting the following constraints:

- It is possible to compute the WCET (or an upper bound) of each task.
- It is possible to compute the minimal duration (or a lower bound) between the arrival of two jobs from the same task.
- There is no precedence relationship between tasks (*i.e.* the arrival date of a job is independent from the state of the other jobs).

- Each task has exactly one activation source[1].

5 A response-time analysis algorithm for AUTOSAR OS

In this section, we propose an algorithm to compute an upper bound on the worst-case response time (WCRT) that a task can experience in a system conform to the structure exposed in Section 4. Intuitively, the WCRT of a task is the longest time interval delimited by the arrival date and the finishing date of a job among all jobs generated by a task among all scenarios. Our proposal follows the "Response Time Analysis" (RTA) approach originating in the works by Joseph and Pandya [10] and Audsley [1]. It is built upon theoretical extensions of these works, especially: response times greater than period [11]; tasks sharing resources [17, 3]; asynchronous transaction based systems [18]; preemption threshold mechanism [19]; FPP/FIFO scheduling policy [4].

We introduce the analysis in two steps. In the first step we define an algorithm that computes an upper bound on the worst-case response time of a task in a given busy-period. In a second step we derive the set of busy-periods that have to be explored in order to find the greatest upper bound.

> We only focus on the case where priorities are all different, *i.e.* $\forall (i,j) \in T, i \neq j, \pi_i \neq \pi_j$.

5.1 Definitions

Definition 1. *[11] A π-busy-period is a maximal time interval where the processing units continuously processes jobs having a preemption threshold no lower than π.*

Definition 2. *The phasing vector $\mathbf{\Phi}$ of a π-busy-period starting at t_0 is a set of values $\langle \phi_1, \ldots, \phi_m \rangle$ such that ϕ_k is the greatest date of occurrence of the activation source λ_k lower or equal than t_0.*

Remark that t_0 and $\mathbf{\Phi}$ completely define the sequence of jobs' execution during a π-busy-period. Therefore, $\langle t_0, \mathbf{\Phi} \rangle$ will be used to denote a π-busy-period.

Definition 3. *In a π_i-busy-period $\langle t_0, \mathbf{\Phi} \rangle$, the phasing of task τ_i belonging to the activation source λ_k is:*

$$\varphi_i(\phi_k) : \mathbb{R}^+ \rightarrow [0, p_k) = (o_i - \phi_k) \mod p_k \tag{1}$$

with $x \mod y = x - y\lfloor x/y \rfloor$. It represents the delay between t_0 and the first activation of the task τ_i at or after t_0.

The phasing is a periodic function with a period equal to p_k, *i.e.* $\varphi_i(\phi_k) = \varphi_i(\phi_k + p_k)$, so its study can be limited to a period, here the interval $(0, p_k]$ is choosen. From now, we restrict the domain of $\varphi_i(\phi_k)$ to $(0, p_k] : \varphi_i(\phi_k) : (0, p_k] \rightarrow [0, p_k)$.

Definition 4. *In a π_i-busy-period $\langle t_0, \mathbf{\Phi} \rangle$, the delay between t_0 and the qth activation of τ_i belonging to the activation source λ_k after t_0 is:*

$$a_{i,q}(\phi_k) = \varphi_i(\phi_k) + (q-1)p_k \tag{2}$$

The figure 1 illustrates the notion of phasing for a task τ_j and its activation source $\lambda_{src(j)}$ relative to a π_i-busy-period which starts at t_0.

Definition 5. *For a task τ_i the set of:*

[1] This constraint can be relaxed by differentiating jobs of a tasks for each activation sources into new tasks.

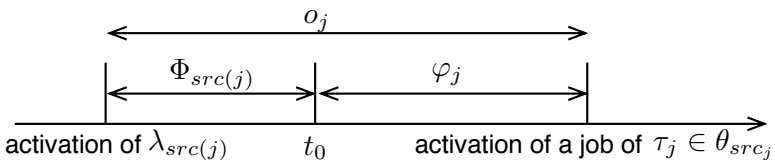

Figure 1: Phasing of task τ_j relative to t_0

- *higher priority tasks is denoted* $hp(i) = \{j | \pi_j > \pi_i\}$,
- *higher threshold priority tasks is denoted* $ht(i) = \{j | \pi_j > \gamma_{i,0}\}$,
- *higher tasks (including* τ_i*) is denoted* $hp^*(i) = hp(i) \cup \{i\}$.

5.2 Maximum response time of a task in a busy-period

In this section, an expression of the maximum response time of a task in a busy-period is produced. In a first time, different workload functions are introduced.

Definition 6. *For a task* τ_i *belonging to the activation source* λ_k*, the right-closed-bounded-workload function* $\underline{w}_i(\phi_k, t) : (0, p_k] \times \mathbb{R} \to \mathbb{R}$ *is:*

$$\underline{w}_i(\phi_k, t) = \left(1 + \left\lfloor \frac{t - \varphi_i(\phi_k)}{p_k} \right\rfloor\right) e_{i0}$$

and the right-open-bounded-workload function $\overline{w}_i(\phi_k, t) : (0, p_k] \times \mathbb{R} \to \mathbb{R}$ *is:*

$$\overline{w}_i(\phi_k, t) = \left(\left\lceil \frac{t - \varphi_i(\phi_k)}{p_k} \right\rceil\right) e_{i0}$$

Lemma 1. *For a fixed t, the argument of the maximum for the functions* $\underline{w}_i$ *and* $\overline{w}_i$ *is:*

$$\phi_k^* = \operatorname*{argmax}_{\phi_k} \underline{w}_i(\phi_k, t) = \operatorname*{argmax}_{\phi_k} \overline{w}_i(\phi_k, t) = \begin{cases} o_i \mod p_k & \text{if } o_i \text{ is not a multiple of } p_k \\ p_k & \text{otherwise} \end{cases}$$

and the functions are monotonic increasing for the intervals $(0, \phi_k^*]$ *and* $(\phi_k^*, p_k]$.

Proof. Figure 2 shows the graph of the φ_i function. It is clear that φ_i is minimal for ϕ_k^* and decreases on intervals $[0, \phi_k^*]$ and $(\phi_k^*, p_{src(j)})$. Hence, by definition of the ceil and floor functions, for a given t, $\underline{w}_i$ and $\overline{w}_i$ are maximal for ϕ_k^* and increase on the intervals $(0, \phi_k^*]$ and $(\phi_k^*, p_k]$. □

Definition 7. *The workload of a task* τ_i *in a* π*-busy-period is the time needed to execute the task* τ_i *during this* π*-busy-period.*

Lemma 2. *For a* π*-busy-period* $\langle t_0, \mathbf{\Phi} \rangle$*, the workload generated in* $[t_0, t_0 + t]$ *by the task* τ_i *belonging to the activation source* λ_k *is bounded by* $\underline{w}_i(\phi_k, t)$.

Proof. By definition (see def. 3), the first job of τ_i arrived in the π-busy-period $\langle t_0, \mathbf{\Phi} \rangle$ at $t_0 + \varphi_i(\phi_k)$ and the next job earliest arrived p_k later. Thus, the number of jobs of τ_i activated in $[t_0, t_0 + t]$ is bounded by $1 + \lfloor (t - \varphi_i(\phi_k))/p_k \rfloor$. Moreover, the workload of a job is maximal when it consumes all its WCET, *i.e.* e_{j0}. □

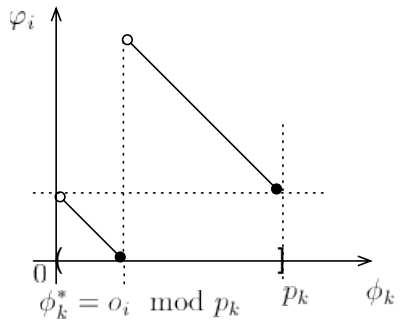

Figure 2: Graph of the function $\varphi_i(\phi_k, t)$ for a fixed t

Notice that the bound is tight when τ_i produces jobs at its maximal rate.

Lemma 3. *For a π-busy-period $\langle t_0, \mathbf{\Phi}\rangle$, the workload generated in $[t_0, t_0 + t)$ by the task τ_i belonging to the activation source λ_k is bounded by $\overline{w}_i(\phi_k, t)$.*

Proof. Same reasoning as lemma 2, except that an activation happens at t is not taken into account. □

Once again, the bound is tight when τ_i produces jobs at its maximal rate.

Lemma 4. *In a π_i-busy-period $\langle t_0, \mathbf{\Phi}\rangle$, an upper bound on the starting date of the qth job of τ_i belonging to the activation source λ_k activated at or after t_0 is the smallest solution of the following equation:*

$$s_{i,q}(\mathbf{\Phi}) = \underbrace{B_i}_{(1)} + \underbrace{(q-1)e_{i0}}_{(2)} + \underbrace{\sum_{j \in hp(i)} \underline{w}_j(\phi_{src(j)}, s_{i,q}(\mathbf{\Phi}))}_{(3)} \tag{3}$$

with

$$B_i = \max\{e_{jk} | \pi_j < \pi_i \leq \gamma_{jk}\} \tag{4}$$

Proof. The qth job of τ_i can start its execution early after:

(1) the jobs with priority lower than π_i finish to execute a block which has a preemption threshold greater than π_i. The priority manager ensures that there is at most one such job, and that this job will be preempted as soon as it recovers its base priority. Therefore, B_i bounds the interference caused by this job on τ_i (see [17] for explanation of maximum blocking time for priority ceiling protocol).

(2) the previous jobs of τ_i activated in the busy-period finish their execution. The maximum interference from these jobs is $(q-1)e_{i0}$.

(3) the jobs with priority greater than π_i, *i.e.* belonging to $hp(i)$, activated in $[t_0, t_0 + s_{i,q}(\mathbf{\Phi})]$ finish their execution. The maximum interference from these jobs is given by $\underline{w}_j(\phi_{src(j)}, t)$ (see lemma 2).

There is no other source that can delay the start of the job in the π_i-busy-period. □

Lemma 5. *In a π_i-busy-period $\langle t_0, \mathbf{\Phi}\rangle$, an upper bound on the finishing date of the qth job of τ_i relative to t_0 is given by the smallest solution of the following equation:*

$$f_{i,q}(\mathbf{\Phi}) = B_i + qe_{i0} + \sum_{j \in ht(i)} \overline{w}_j\left(\phi_{src(j)}, f_{i,q}(\mathbf{\Phi})\right) + \sum_{j \in hp(i) \setminus ht(i)} \underline{w}_j(\phi_{src(j)}, s_{i,q}(\mathbf{\Phi}))$$

Proof. Once started and before to finish, the qth job of τ_i is only delayed by its WCET and by workloads of jobs having a priority greater than γ_{i0}, *i.e.* belonging to $ht(i)$, and activated in $(t_0 + s_{i,q}(\mathbf{\Phi}), t_0 + f_{i,q}(\mathbf{\Phi}))$, so:

$$f_{i,q}(\mathbf{\Phi}) = s_{i,q}(\mathbf{\Phi}) + e_{i0} + \sum_{j \in ht(i)} \overline{w}_j \left(\phi_{src(j)}, f_{i,q}(\mathbf{\Phi})\right) - \sum_{j \in ht(i)} \underline{w}_j \left(\phi_{src(j)}, s_{i,q}(\mathbf{\Phi})\right)$$

Equation (5) is obtained by joining equation (3) to previous one and by using $ht(i) \subseteq hp(i)$. □

Definition 8. *For a π_i-busy-period $\langle t_0, \mathbf{\Phi}\rangle$, the response time of the qth job of τ_i is:*

$$r_{i,q}(\mathbf{\Phi}) = f_{i,q}(\mathbf{\Phi}) - a_{i,q}(\Phi_k) \tag{5}$$

Lemma 6. *The length of a π_i-busy-period $\langle t_0, \mathbf{\Phi}\rangle$ is bounded by the smallest solution of the following equation:*

$$L_i(\mathbf{\Phi}) = B_i + \sum_{j \in hp^*(i)} \overline{w}_j(\phi_{src(j)}, L_i(\mathbf{\Phi})) \tag{6}$$

Proof. The π_i-busy-period finish when the worst-case blocking time from a lower priority tasks, *i.e.* B_i, and all tasks of $hp^*(i)$ activated in $[t_0, t_0 + L_i(\mathbf{\Phi}))$ are finished. □

Lemma 7. *The maximum number of activations of a task τ_i in a π_i-busy-period $\langle t_0, \mathbf{\Phi}\rangle$ is*

$$Q_i(\mathbf{\Phi}) = \left\lceil \frac{L_i(\mathbf{\Phi}) - \varphi_i(\phi_k)}{p_k} \right\rceil \tag{7}$$

Proof. It is immediate from Lemma 6. □

Definition 9. *For a π_i-busy-period $\langle t_0, \mathbf{\Phi}\rangle$, the maximum response time of task τ_i belonging to the activation source λ_k is:*

$$R_i(\mathbf{\Phi}) = \max_{q \in Q_i(\mathbf{\Phi})} r_{i,q}(\mathbf{\Phi}) \tag{8}$$

where

$$Q_i(\mathbf{\Phi}) = \left\lceil \frac{L_i(\mathbf{\Phi}) - \varphi_i(\phi_k)}{p_k} \right\rceil \tag{9}$$

5.3 General results on iterated functions

In this part, some general results on iterated functions are given.

5.3.1 Hypothesis on iterated functions

Consider a set of n functions $f_i : X_i \times \mathbb{R} \to \mathbb{R}$ where $X_i = (a_i, b_i]$ is left-open and right-closed interval, and all f_i respect the following properties:

- P1 f_i is a monotonic increasing function for t : $\forall x \in X_i, \forall (t_1, t_2) \in \mathbb{R}^2$ such that $t_1 \leq t_2$ one has $f_i(x, t_1) \leq f_i(x, t_2)$,
- P2 $\forall x \in X_i, f_i(x, t) \leq \alpha_i + \beta_i t$ with $\alpha_i > 0$, $\beta_i > 0$ and $\sum_{i \in [1..n]} \beta_i < 1$,
- P3 $\forall x \in X_i, f_i(x, 0) \geq 0$,
- P4 f_i is a monotonic increasing function for x: $\forall t \in \mathbb{R}, \forall (x_1, x_2) \in X_i^2$ such that $x_1 \leq x_2$ one has $f_i(x_1, t) \leq f_i(x_2, t)$.

5.3.2 Results on the summation of functions

We define $\kappa(\mathbf{x},t) : \prod_{i\in[1..n]} X_i \times \mathbb{R} \to \mathbb{R} = \sum_i f_i(x_i,t)$ with $\mathbf{x} = (x_1,...,x_n)$.

Lemma 8. *For an interval $X_i = (a_i, b_i]$, for a given $t \in \mathbb{R}$, the maximum of f_i is $f_i(b,t)$:*

$$\forall t \in \mathbb{R}, \max_{x\in(a_i,b_i]} f_i(x,t) = f_i(b_i,t)$$

Proof. The result is immediate by considering the proposition P4. □

Lemma 9. *For a given $t \in \mathbb{R}$, the maximum of $\kappa(\mathbf{x},t)$ is $\kappa(\bar{\mathbf{x}},t)$ where $\bar{\mathbf{x}} = (b_1,...,b_n)$:*

$$\forall t \in \mathbb{R}, \max_{\mathbf{x}\in\prod_{i\in[1..n]} X_i} \kappa(\mathbf{x},t) = \kappa(\bar{\mathbf{x}},t)$$

Proof. The functions f_i are independent, so, for a given t we have

$$\max_{\mathbf{x}\in\prod_{i\in[1..n]} X_i} \sum_{i\in[1..n]} f_i(x_i,t) = \sum_{i\in[1..n]} \max_{x_i\in X_i} f_i(x_i,t)$$

Lemma 8 gives immediatly $\max_{\mathbf{x}\in\prod_{i\in[1..n]} X_i} \sum_{i\in[1..n]} f_i(x_i,t) = \sum_{i\in[1..n]} f_i(b_i,t) = \kappa(\bar{\mathbf{x}},t)$. □

5.3.3 Results on the fixed point

Consider the iterated function $u_\kappa(\mathbf{x})$:

$$\left\{ \begin{array}{lcl} u^0_\kappa(\mathbf{x}) & = & \kappa(\mathbf{x},0) \\ u^{n+1}_\kappa(\mathbf{x}) & = & \kappa(\mathbf{x}, u^n_\kappa(\mathbf{x})) \end{array} \right.$$

and denote $\mathrm{fix}_\kappa(\mathbf{x})$ the fixed point of the iterative sequence $u_\kappa(\mathbf{x})$. Properties P1, P2 and P3 assure that this fixed point exists and is equal to the least fixed point.

Lemma 10. *The function $\mathrm{fix}_\kappa(\mathbf{x})$ is a monotone increasing function for each variable x_i.*

Proof. From lemma 9, $\forall \mathbf{x} \in \prod_{i\in[1..n]} X_i$ with $x_i + \delta_i < b_i, \delta_i \geq 0$, we have $\kappa(\mathbf{x},0) \leq \kappa(\mathbf{x}+\boldsymbol{\delta},0)$ with $\boldsymbol{\delta} = (\delta_1,...,\delta_n)$ and so $u^0_\kappa(\mathbf{x}) \leq u^0_\kappa(\mathbf{x}+\boldsymbol{\delta})$. Now, suppose that for a rank n, $\forall \mathbf{x} \in \prod_{i\in[1..n]} X_i, u^n_\kappa(\mathbf{x}) \leq u^n_\kappa(\mathbf{x}+\boldsymbol{\delta})$, because of property P1 (f_i increases with t) $u^{n+1}_\kappa(\mathbf{x}) \leq u^{n+1}_\kappa(\mathbf{x}+\boldsymbol{\delta})$. Thus, $\mathrm{fix}_\kappa(\mathbf{x}) \leq \mathrm{fix}_\kappa(\mathbf{x}+\boldsymbol{\delta})$. □

Lemma 11.

$$\max_{\mathbf{x}\in\prod_{i\in[1..n]} X_i} \mathrm{fix}_\kappa(\mathbf{x}) = \mathrm{fix}_\kappa(\bar{\mathbf{x}})$$

Proof. From lemma 9, $\forall \mathbf{x} \in \prod_{i\in[1..n]} X_i, \kappa(\mathbf{x},0) \leq \kappa(\bar{\mathbf{x}},0)$ and so $u^0_\kappa(\mathbf{x}) \leq u^0_\kappa(\bar{\mathbf{x}})$. Now, suppose that for a rank n, $\forall \mathbf{x} \in \prod_{i\in[1..n]} X_i, u^n_\kappa(\mathbf{x}) \leq u^n_\kappa(\bar{\mathbf{x}})$, because of property P1 (f_i increases with t) $u^{n+1}_\kappa(\mathbf{x}) \leq u^{n+1}_\kappa(\bar{\mathbf{x}})$. Thus, $\mathrm{fix}_\kappa(\mathbf{x}) \leq \mathrm{fix}_\kappa(\bar{\mathbf{x}})$. □

Lemma 12. *Consider two functions $f_1(\mathbf{x},t)$ and $f_2(\mathbf{x},t)$ with properties P1 such $\forall(\mathbf{x},t)$, $f_1(\mathbf{x},t) \leq f_2(\mathbf{x},t)$, then $\mathrm{fix}_{f_1}(\mathbf{x}) \leq \mathrm{fix}_{f_2}(\mathbf{x})$.*

Proof. By definition with have $u^0_{f_1}(\mathbf{x}) \leq u^0_{f_2}(\mathbf{x})$. Suppose that $u^n_{f_1}(\mathbf{x}) \leq u^n_{f_2}(\mathbf{x})$ is true for the rank n, then by using P1 $u^{n+1}_{f_1}(\mathbf{x}) = f_1(\mathbf{x}, u^n_{f_1}(\mathbf{x})) \leq f_1(\mathbf{x}, u^n_{f_2}(\mathbf{x})) \leq f_2(\mathbf{x}, u^n_{f_2}(\mathbf{x})) = u^{n+1}_{f_2}(\mathbf{x})$. □

5.3.4 Results on the fixed point of a composed function

Consider a set of functions $F = \{f_i(x,t) : X_i \times \mathbb{R} \to \mathbb{R}\}$ and a function $g(\mathbf{x}) : \prod_i X_i \to \mathbb{R}$ such all functions f_i respect properties P1, P2, P3 and P4 and g increases for all variables, *i.e.* $\forall(\mathbf{x}_1, \mathbf{x}_2) \in (\prod_i X_i)^2$ with $\mathbf{x}_{1,i} \leq \mathbf{x}_{2,i}$ we have $g(\mathbf{x}_1) \leq g(\mathbf{x}_2)$.

The function λ is defined as:

$$\lambda(\mathbf{x}, t) : \prod_{i\in[1..n]} X_i \times \mathbb{R} \to \mathbb{R} = g(\mathbf{x}) + \sum_{i\in[1..n]} f_i(x_i, t)$$

the iterated function $u_\lambda(\mathbf{x})$:

$$\begin{cases} u_\lambda^0(\mathbf{x}) & = & \lambda(\mathbf{x}, 0) \\ u_\lambda^{n+1}(\mathbf{x}) & = & \lambda(\mathbf{x}, u_\lambda^n(\mathbf{x})) \end{cases}$$

and $\mathrm{fix}_\lambda(\mathbf{x})$ the fixed point of the iterative sequence $u_\lambda(\mathbf{x})$.

Lemma 13.

$$\max_{\mathbf{x}\in\prod_{i\in[1..n]} X_i} \mathrm{fix}_\lambda(\mathbf{x}) = \mathrm{fix}_\lambda(\bar{\mathbf{x}})$$

Proof. By definition $\forall \mathbf{x} \in \prod_i X_i, g(\mathbf{x}) \leq g(\bar{\mathbf{x}})$. Thus, from lemma 9, $\forall \mathbf{x} \in \prod_{i\in[1..n]} X_i$, $\lambda(\mathbf{x}, 0) \leq \lambda(\bar{\mathbf{x}}, 0)$ and so $u_\lambda^0(\mathbf{x}) \leq u_\lambda^0(\bar{\mathbf{x}})$. Now, suppose that for a rank n, $\forall \mathbf{x} \in \prod_{i\in[1..n]} X_i, u_\lambda^n(\mathbf{x}) \leq u_\lambda^n(\bar{\mathbf{x}})$, because of property P1 (f_i increases with t) $u_\lambda^{n+1}(\mathbf{x}) \leq u_\lambda^{n+1}(\bar{\mathbf{x}})$. Thus, $\mathrm{fix}_\lambda(\mathbf{x}) \leq \mathrm{fix}_\lambda(\bar{\mathbf{x}})$. □

Now, $\bar{X}_i = \{x_{i,j} | j \geq 0\}, x_{i,0} = 0$ denotes an totally ordered set of positive values such that on each interval $X_{i,j} = (x_{i,j}, x_{i,j+1}]$, f_i is defined and respects properties P1, P2, P3 and P4.

Lemma 14. *The maxima of* $\mathrm{fix}_\kappa(\mathbf{x})$ *and* $\mathrm{fix}_\lambda(\mathbf{x})$ *are found in* $\prod_{i\in[1..n]} \bar{X}_i$*:*

$$\max_{\mathbf{x}\in\prod_{i\in[1..n]} \bigcup_j X_{i,j}} \mathrm{fix}_\kappa(\mathbf{x}) = \max_{\mathbf{x}\in\prod_{i\in[1..n]} \bar{X}_i} \mathrm{fix}_\kappa(\mathbf{x})$$
$$\max_{\mathbf{x}\in\prod_{i\in[1..n]} \bigcup_j X_{i,j}} \mathrm{fix}_\lambda(\mathbf{x}) = \max_{\mathbf{x}\in\prod_{i\in[1..n]} \bar{X}_i} \mathrm{fix}_\lambda(\mathbf{x})$$

Proof. The proof is immediate by considering the lemma 11 and 13. □

5.4 Reducing set of phasing vectors

To find the worst-case response time of a task, it is necessary to compute the maximum response time of this task for all busy-periods for all execution of (T, Λ, src). From previous sections, we can remark that the response time of a task in a busy-period depends only on $\mathbf{\Phi}$. Thus, if Γ denotes the set of all possible values of $\mathbf{\Phi}$ for all busy-periods, we have :

$$R_i = \max_{\mathbf{\Phi}\in\Gamma} R_i(\mathbf{\Phi}) \tag{10}$$

In this section we prove that it is not necessary to explore all Γ, but only π_i-busy-periods $\langle t_0, \mathbf{\Phi}\rangle$ such that for each activation sources, at least one task of priority greater or equal to π_i generates a job at t_0.

5.4.1 Rewriting functions

The end and start of a task τ_i in a π_i-busy-period were defined by

$$\begin{aligned} s_{i,q}(\mathbf{\Phi}) &= \min\{s_{i,q}(\mathbf{\Phi},t) = t | t > 0\} \\ f_{i,q}(\mathbf{\Phi}) &= \min\{f_{i,q}(\mathbf{\Phi},t) = t | t > 0\} \end{aligned}$$

with

$$s_{i,q}(\mathbf{\Phi},t) = B_i + (q-1)e_{i0} + \sum_{j \in hp(i)} \underline{w}_j(\phi_{src(j)}, t) \tag{11}$$

$$f_{i,q}(\mathbf{\Phi},t) = B_i + qe_{i0} + \sum_{j \in ht(i)} \overline{w}_j\left(\phi_{src(j)}, t\right) + \sum_{j \in hp(i) \setminus ht(i)} \underline{w}_j\left(\phi_{src(j)}, s_{i,q}(\mathbf{\Phi})\right) \tag{12}$$

These can be solve by the computation of fixed points of

$$\begin{cases} u_s^0(\mathbf{\Phi}) &= s_{i,q}(\mathbf{\Phi}, 0) \\ u_s^{n+1}(\mathbf{\Phi}) &= s_{i,q}(\mathbf{\Phi}, u_s^n(\mathbf{\Phi})) \end{cases}$$

$$\begin{cases} u_f^0(\mathbf{\Phi}) &= f_{i,q}(\mathbf{\Phi}, 0) \\ u_f^{n+1}(\mathbf{\Phi}) &= f_{i,q}(\mathbf{\Phi}, u_f^n(\mathbf{\Phi})) \end{cases}$$

Let $\mathcal{F}_{k,i}$, $\mathcal{G}_{k,i}$ and $\mathcal{H}_{i,q}$ the sums of workload from an activation source on a π_i-busy-period at t defined as:

$$\mathcal{F}_{k,i}(x,t) : \quad (0, p_k] \times \mathbb{R} \to \mathbb{R} = \sum_{j \in hp(i) \cap \theta_k} \underline{w}_j(x,t) \tag{13}$$

$$\mathcal{G}_{k,i}(x,t) : \quad (0, p_k] \times \mathbb{R} \to \mathbb{R} = \sum_{j \in ht(i) \cap \theta_k} \overline{w}_j(x,t) \tag{14}$$

$$\mathcal{H}_{i,q}(\mathbf{\Phi}) : \quad \Gamma \to \mathbb{R} = \sum_{j \in hp(i) \setminus ht(i)} \underline{w}_j(\phi_{src(j)}, s_{i,q}(\mathbf{\Phi})) \tag{15}$$

By definition $hp(i) = \bigcup_{k \in [1..m]} (hp(i) \cap \theta_k)$ and $ht(i) = \bigcup_{k \in [1..m]} (ht(i) \cap \theta_k)$, thus:

$$s_{i,q}(\mathbf{\Phi},t) = B_i + (q-1)e_{i0} + \sum_{k \in [1..m]} \mathcal{F}_{k,i}(\phi_k, t) \tag{16}$$

$$f_{i,q}(\mathbf{\Phi},t) = B_i + qe_{i0} + \mathcal{H}_{i,q}(\mathbf{\Phi}) + \sum_{k \in [1..m]} \mathcal{G}_{k,i}(\phi_k, t) \tag{17}$$

5.4.2 Set of phasing vectors

Let $\bar{X}_{k,i}$ denotes for an activation source λ_k and a task τ_i the ordered set of phasing values ϕ_k such $\bar{X}_{k,i} = \{0\} \cup \{x | x = o_j \mod p_k, \forall j \in hp(i) \cap \theta_k\} \cup \{p_k\}$. This set is the set of phase values such that for each activation source a task of $hp(i)$ is synchronous with the start of the busy-period.

Lemma 15. *The functions $\mathcal{F}_{k,i}$ and $\mathcal{G}_{k,i}$ respect properties P1, P2, P3 and P4 on each interval $(x_1, x_2]$ where x_1 and x_2 are two consecutive values of $\bar{X}_{k,i}$.*

Proof. Consider x_1 and x_2 two consecutive values of $\bar{X}_{k,i}$, from the lemma 1 :

$$\forall x \in (x_1, x_2], \forall t \in \mathbb{R}, \forall j \in \theta_k, \underline{w}_j(x_2, t) \geq \underline{w}_j(x, t) \implies \mathcal{F}_{k,i}(x_2, t) \geq \mathcal{F}_{k,i}(x, t)$$
$$\forall x \in (x_1, x_2], \forall t \in \mathbb{R}, \forall j \in \theta_k, \overline{w}_j(x_2, t) \geq \overline{w}_j(x, t) \implies \mathcal{G}_{k,i}(x_2, t) \geq \mathcal{G}_{k,i}(x, t)$$

□

Lemma 16. *The function $\mathcal{H}_{i,q}$ is monotone increasing on each interval $\prod_{k\in[1..m]}(x_{k,1}, x_{k,2}]$ where $x_{k,1}$ and $x_{k,2}$ are two consecutive values of $\bar{X}_{k,i}$.*

Proof. The functions $\mathcal{F}_{k,i}$ respect properties P1, P2, P3 and P4, so, from lemma 10, the function $s_{i,q}(\mathbf{\Phi})$ (that is the fixed point of the sum of $\mathcal{F}_{k,i}$) is a monotonic increasing function for each ϕ_k.

Thus, if all ϕ_k increases in each $(x_{k,1}, x_{k,2}]$ then the functions $\underline{w}_j(\phi_{src(j)}, s_{i,q}(\mathbf{\Phi}))$ increase, and so $\mathcal{H}_{i,q}(\mathbf{\Phi})$ is an monotone increasing fonction. □

Lemma 17. *The maximum of $f_{i,q}(\mathbf{\Phi})$ is fond for a phasing vectors $\mathbf{\Phi}$ in the set $\prod_{k\in[1..m]} \bar{X}_{k,i}$.*

Proof. The value $s_{i,q}(\Phi)$ is the fixed point of an iterative function which is the sum of functions $\mathcal{F}_{k,i}$ that respect properties P1, P2, P3 and P4 so $s_{i,q}(\mathbf{\Phi})$ is a monotonic increasing function for each ϕ_k (Lemma 14).

From Lemma 14 and the rewriting of $f_{i,q}(\mathbf{\Phi})$, we alse show that

$$\max_{\mathbf{\Phi}\in\Gamma} f_{i,q}(\mathbf{\Phi}) = \max_{\mathbf{\Phi}\in\prod_{k\in[1..m]} \bar{X}_{k,i}} f_{i,q}(\mathbf{\Phi})$$

Q.E.D. □

Lemma 18. *The maximum of $r_{i,q}(\mathbf{\Phi})$ is fond for a phasing vectors $\mathbf{\Phi}$ in the set $\prod_{k\in[1..m]} \bar{X}_{k,i}$.*

Proof. By definition (eq. 2) $a_{i,q}(\phi_k) = \varphi_i(\phi_k) + (q-1)p_k$, so for x_1 and x_2 two consecutive values of $\bar{X}_{k,i}$, $\forall x \in (x_1, x_2], a_{i,q}(x) > a_{i,q}(x_2)$ (φ_i is decreasing in $(x_1, x_2]$, see fig. 2). Moreover, by definition (eq. 5) $r_{i,q}(\mathbf{\Phi}) = f_{i,q}(\mathbf{\Phi}) - a_{i,q}(\Phi_k)$ and lemma 17, $r_{i,q}$ is maximal for a value in $\prod_{k\in[1..m]} \bar{X}_{k,i}$. □

5.4.3 Discussion about the complexity

The complexity to compute the worst-case response time comes from the size of $\prod_k |\bar{X}_{k,i}|$. This set represents all possible combination of candidates for the beginning of the busy-period among all schedule table in the system. This could be become computationally intractable even fo small task sets.

In the literature, two approaches exit to solve this problem. The first one consists to exhibit some approximate functions with a lower computational complexity. However, these approximations increases the pessimism of the analysis. In [18], Tindell provided an approximate function that still gives good results but uses a single approximation function for each table. Palencia Gutierrez *et al.* [16] formalized and generalized Tindell's work. In [14], an improvement of Tindell and Palencia *et al.* is presented to reduce the over-estimation of the worst-case response time, *i.e* to reduce the pessimism.

The second approach consists to optimize the algorithm to increase time efficiency. In [13], a static representation of cyclic functions is pre-computed and used to speed up the computation. The advantage is to reduce drastically the number of computation during the solution search. However, sometimes, the pre-computation could be heavy time costly .

In [14], Mäki *et al.* done a synthesis of the approximate approach with static function representation. For the moment, we do not study this approach.

All this works [18, 16, 14, 13] do not use the same model than AUTOSAR one. They use the usual Response Time Analysis (RTA) model of preemptive transactions, *i.e* schedule tables, under fixed-priority scheduler. They do not consider mixed preemption. Next Sections expose how to extend these works to the AUTOSAR model.

6 Approximate computation

The computational complexity of the AUTOSAR worst-case analysis comes from every possible combination of phases among all schedule tables in the system. An identical problem appears for RTA of real-time transactions [18]. A transaction is exactly the same object than a schedule table. The main difference with the present model is the scheduling policy. In [18], the fixed-priority scheduler is only take into account.

Tindell provided an approximate RTA that still gives good results. To reduce the number of combination, an approximation function of the interference is used for each transaction and not for each task. A formalization and generalization of Tindell's work is done by Palencia Gutierrez *et al.* in [16].

In this section, we extend the Tindell's approach to our model.

6.1 Approximate functions of starting date

We have:

$$\begin{aligned} s_{i,q}(\mathbf{\Phi}) &= \min\{s_{i,q}(\mathbf{\Phi},t) = t | t > 0\} = \mathrm{fix}_{s_{i,q}}(\mathbf{\Phi}) \\ s_{i,q}(\mathbf{\Phi},t) &= B_i + (q-1)e_{i0} + \sum_{k\in[1..m]} \mathcal{F}_{k,i}(\phi_k,t) \end{aligned}$$

with $\mathcal{F}_{k,i}(\phi_k,t) = \sum_{j\in hp(i)\cap\theta_k} \underline{w}_j(\phi_k,t)$

We introduce now:

$$\hat{\mathcal{F}}_{k,i}(t) : \mathbb{R} \to \mathbb{R} = \max_{\phi_k\in\bar{X}_{k,i}} \sum_{j\in hp(i)\cap\theta_k} \underline{w}_j(\phi_k,t) = \max_{x\in\bar{X}_{k,i}} \mathcal{F}_{k,i}(\phi_k,t)$$

and we define a new approximation function:

$$\hat{s}_{i,q}(\mathbf{\Phi},t) = B_i + (q-1)e_{i0} + \mathcal{F}_{src(i),i}(\phi_{src(i)},t) + \sum_{k\neq src(i)} \hat{\mathcal{F}}_{k,i}(t)$$

Lemma 19.

$$\forall \mathbf{\Phi} \in \textstyle\prod_{k\in[1..m]} \bar{X}_{k,i}, \mathrm{fix}_{s_{i,q}}(\mathbf{\Phi}) \leq \mathrm{fix}_{\hat{s}_{i,q}}(\mathbf{\Phi})$$

Proof. By definition we have $\forall x \in \bar{X}_{k,i}, \forall t > 0, 0 \leq \mathcal{F}_{k,i} \leq \hat{\mathcal{F}}_{k,i}$ and so $\forall(\mathbf{\Phi},t), s_{i,q}(\mathbf{\Phi},t) \leq \hat{s}_{i,q}(\mathbf{\Phi},t)$. From Lemma 12 we have $\mathrm{fix}_{s_{i,q}}(\mathbf{\Phi}) \leq \mathrm{fix}_{\hat{s}_{i,q}}(\mathbf{\Phi})$. □

Definition 10. *We denote the maximum of $\hat{s}(\mathbf{\Phi})$ as*

$$\hat{s}_{i,q} = \max_{\mathbf{\Phi}\in\prod_{k\in[1..m]} \bar{X}_{k,i}} \mathrm{fix}_{\hat{s}_{i,q}}(\mathbf{\Phi}) = \max_{\phi_{src(i)}\in\bar{X}_{i,i}} \mathrm{fix}_{\hat{s}_{i,q}}(0,...,\phi_{src(i)},...,0)$$

(remark that $\hat{s}_{i,q}(\mathbf{\Phi})$ do not depend to $\phi_k, k \neq src(i)$).

Lemma 20.

$$\forall \mathbf{\Phi} \in \Gamma, s_{i,q}(\mathbf{\Phi}) \leq \hat{s}_{i,q}$$

Proof. From Lemma 17, $\forall \mathbf{\Phi} \in \Gamma, s_{i,q}(\mathbf{\Phi}) \leq \max_{\mathbf{\Phi}\in\prod_{k\in[1..m]} \bar{X}_{k,i}} s_{i,q}(\mathbf{\Phi})$ Q.E.D. □

6.2 Approximate functions of finishing date

We have

$$\begin{aligned} f_{i,q}(\mathbf{\Phi}) &= \min\{f_{i,q}(\mathbf{\Phi},t) = t | t > 0\} \\ f_{i,q}(\mathbf{\Phi},t) &= B_i + qe_{i0} + \sum_{k\in[1..m]} \mathcal{G}_{k,i}(\phi_k,t) + \sum_{j\in hp(i)\setminus ht(i)} \underline{w}_j(\phi_{src(j)}, s_{i,q}(\mathbf{\Phi})) \end{aligned}$$

and we introduce

$$\mathcal{I}_{k,i}(x,t) : (0,p_k]\times\mathbb{R}\to\mathbb{R} = \mathcal{G}_{k,i}(x,t) + \sum_{j\in hp(i)\setminus ht(i)\cap\theta_k} \underline{w}_j(x,\hat{s}_{i,q})$$

$$\hat{\mathcal{I}}_{k,i}(t) : \mathbb{R}\to\mathbb{R} = \max_{\phi_k\in\bar{X}_{k,i}} \mathcal{I}_{k,i}(\phi_k,t)$$

and we define a new approximation function:

$$\hat{f}_{i,q}(\mathbf{\Phi},t) = B_i + qe_{i0} + \mathcal{I}_{src(i),i}(\phi_{src(i)},t) + \sum_{k\neq src(i)} \hat{\mathcal{I}}_{k,i}(t)$$

Lemma 21.

$$\forall\mathbf{\Phi}\in\Gamma, f_{i,q}(\mathbf{\Phi}) = \mathrm{fix}_f(\mathbf{\Phi}) \leq \mathrm{fix}_{\hat{f}}(\mathbf{\Phi}) = \hat{f}_{i,q}(\mathbf{\Phi})$$

Proof. From the definition of $\hat{\mathcal{I}}_{k,i}$, we have $\forall(\mathbf{\Phi},t)\in(\prod_{k\in[1..m]}\bar{X}_{k,i},\mathbb{R}), f_{i,q}(\mathbf{\Phi},t)\leq\hat{f}_{i,q}(\mathbf{\Phi},t)$ and so from Lemma 12 we have $\mathrm{fix}_f(\mathbf{\Phi})\leq\mathrm{fix}_{\hat{f}}(\mathbf{\Phi})$. Q.E.D. □

6.3 An approximation of worst-case response time

We define a first over-approximation of the response time of the qth job in an π_i-busy period as:

$$\hat{r}_{i,q}(\mathbf{\Phi}) = \hat{f}_{i,q}(\mathbf{\Phi}) - a_{i,q}(\phi_{src(i)})$$

Definition 11. *We denote the maximum of $\hat{r}_{i,q}(\mathbf{\Phi})$ as*

$$\hat{r}_{i,q} = \max_{\mathbf{\Phi}\in\prod_{k\in[1..m]}\bar{X}_{k,i}} (\hat{f}_{i,q}(\mathbf{\Phi}) - a_{i,q}(\phi_{src(i)})) = \max_{\phi_{src(i)}\in\bar{X}_{i,i}} (\hat{f}_{i,q}(\mathbf{\Phi}) - a_{i,q}(\phi_{src(i)}))$$

Lemma 22.

$$\forall\mathbf{\Phi}\in\Gamma, r_{i,q}(\mathbf{\Phi}) \leq \hat{r}_{i,q}$$

Proof. We have $\forall\mathbf{\Phi}\in\Gamma, r_{i,q}(\mathbf{\Phi}) = f_{i,q}(\mathbf{\Phi}) - a_{i,q}(\phi_{src(i)}) \leq \hat{f}_{i,q}(\mathbf{\Phi}) - a_{i,q}(\phi_{src(i)}) = \hat{r}_{i,q}(\mathbf{\Phi})$. Moreover $\forall\mathbf{\Phi}\in\Gamma, r_{i,q}(\mathbf{\Phi}) \leq \max_{\mathbf{\Phi}\in\prod_{k\in[1..m]}\bar{X}_{k,i}} r_{i,q}(\mathbf{\Phi})$. Q.E.D by definition of $\hat{r}_{i,q}$. □

Lemma 23. *The worst-case response time R_i of a task τ_i is over-approximated by $\hat{R}_i$ computed by*

$$\hat{R}_i = \max_{q\in\hat{Q}_i} \hat{r}_{i,q}$$

with

$$\hat{Q}_i = \left\lceil \frac{\hat{L}_i}{p_{src(i)}} \right\rceil \text{ and } \hat{L}_i = \min\{B_i + \textstyle\sum_{j\in hp^*(i)} \overline{w}_j(0,t) = t \mid t > 0\}$$

Proof. We have $R_i = \max_{\mathbf{\Phi}\in\Gamma} \max_{q\in[1..Q_i(\mathbf{\Phi})]} r_{i,q}(\mathbf{\Phi})$ and by definition $\forall \mathbf{\Phi} \in \Gamma, \hat{Q}_i > Q_i(\mathbf{\Phi})$, so $R_i \leq \max_{\mathbf{\Phi}\in\Gamma} \max_{q\in[1..\hat{Q}_i]} r_{i,q}(\mathbf{\Phi}) = \max_{q\in[1..\hat{Q}_i]} \max_{\mathbf{\Phi}\in\Gamma} r_{i,q}(\mathbf{\Phi})$. By Lemma 22, we have $R_i \leq \max_{q\in[1..\hat{Q}_i]} \hat{r}_{i,q} = \hat{R}_i$. □

6.4 Discussion about the complexity

It is necessar to evaluate $\hat{s}_{i,q}$ and $\hat{f}_{i,q}$ with a fixed point iterative function for each value of $\bar{X}_{i,i}$ to compute $\hat{R}_i$. Moreover for each iteration, the maxima of $\hat{\mathcal{F}}_{k,i}$, $\hat{\mathcal{G}}_{k,i}$ and $\hat{\mathcal{H}}_{k,i}$ have to be found in the set $\bar{X}_{k,i}$. These can be done in $O(|\bar{X}_{k,i}|)$.

Remark that the number of iterations can be bound by γ (see appendix) and thus the complexity if $O(\gamma|\bar{X}_{i,i}| \sum_{k\neq src(i)} |\bar{X}_{k,i}|)$.

7 Fast response-time

In this Section, the fast response-time analysis of Mäki-Turja *et al.* [13] is extended to the AUTOSAR model. This method is based on a pre-computed table to make a static representation of $\hat{\mathcal{F}}$ and $\hat{\mathcal{I}}$ and thus speed the computation of $\hat{r}_{i,q}$. The key to done this representation is to recognize a cyclic pattern that repeats itself every p_k.

7.1 Preliminaries

Lemma 24.

$$\forall \phi_k, \forall t = a.p_k + t', a \in \mathbb{N}, \underline{w}_i(\phi_k, t) = a.e_{i0} + \underline{w}_i(\phi_k, t')$$
$$\forall \phi_k, \forall t = a.p_k + t', a \in \mathbb{N}, \overline{w}_i(\phi_k, t) = a.e_{i0} + \overline{w}_i(\phi_k, t')$$

Proof.

$$\begin{aligned}
\underline{w}_i(\phi_k, t) &= \left(1 + \left\lfloor \frac{t - \varphi_i(\phi_k)}{p_k} \right\rfloor\right) e_{i0} = \left(1 + \left\lfloor \frac{a.p_k + t' - \varphi_i(\phi_k)}{p_k} \right\rfloor\right) e_{i0} \\
&= \left(1 + \left\lfloor \frac{a.p_k}{p_k} + \frac{t' - \varphi_i(\phi_k)}{p_k} \right\rfloor\right) e_{i0} = \left(1 + a + \left\lfloor \frac{t' - \varphi_i(\phi_k)}{p_k} \right\rfloor\right) e_{i0} \\
&= a.e_{i0} + \left(1 + \left\lfloor \frac{t' - \varphi_i(\phi_k)}{p_k} \right\rfloor\right) e_{i0} = a.e_{i0} + \underline{w}_i(\phi_k, t')
\end{aligned}$$

The reasoning is exactly the same for $\overline{w}_i(\phi_k, t)$. □

Lemma 25.

$$\forall k \in [1..m], \forall t = a.p_k + t', a \in \mathbb{N}, \hat{\mathcal{F}}_{k,i}(t) = a.F_{k,i} + \hat{\mathcal{F}}_{k,i}(t')$$
$$\forall k \in [1..m], \forall t = a.p_k + t', a \in \mathbb{N}, \hat{\mathcal{I}}_{k,i}(t) = a.I_{k,i} + \hat{\mathcal{I}}_{k,i}(t')$$

with $F_{k,i} = \sum_{j\in hp(i)\cap\theta_k} e_{j,0}$ *and* $I_{k,i} = \sum_{j\in hp(i)\cap\theta_k} e_{j,0}$.

Proof. It is immediate from Lemma 24. □

Previous lemmata show that $\hat{\mathcal{F}}_{k,i}$ and $\hat{\mathcal{I}}_{k,i}$ have a periodic behavior, and so a statically pre-computed representation of these functions in a period interval can be used to speed up the response-time calculation.

7.2 Static representation

To pre-compute $\hat{\mathcal{F}}_{k,i}$ and $\hat{\mathcal{G}}_{k,i}$ in the interval [0,p_k), their convex corners (the function lies bellow the straight line segment connecting two consecutive points) is used. For instance, these corners are represented in Figure 3 with crosses for $\underline{w}_i(\phi_k, t)$ and $\overline{w}_i(\phi_k, t)$ functions.

The representation of $\hat{\mathcal{F}}_{k,i}$, respectively $\hat{\mathcal{I}}_{k,i}$, uses an array $A_{k,i}$, resp. $B_{k,i}$ and $C_{k,i}$, of points that represent their convex corners and $F_{k,i}$, resp. $I_{k,i}$.

A point is an object which has an x (representing time) and a y (representing workload) coordinate. We denote $X[j].x$, respectively $X[j].y$, the x, resp. y, value of the jth point in X. Using these objects, we have:

$$\begin{aligned}
\hat{\mathcal{F}}_{k,i}(t) &= a.F_{k,i} + A_{k,i}[k_A].y \\
\hat{\mathcal{I}}_{k,i}(t) &= a.I_{k,i} + B_{k,i}[k_B].y + C_{k,i}[k_C].y \\
a &= t \text{ div } p_k \\
t' &= t \text{ rem } p_k \\
k_A &= \max\{l : A_{k,i}[l].x \leq t'\} \\
k_B &= \max\{l : B_{k,i}[l].x < t'\} \\
k_C &= \max\{l : C_{k,i}[l].x \leq t'\}
\end{aligned}$$

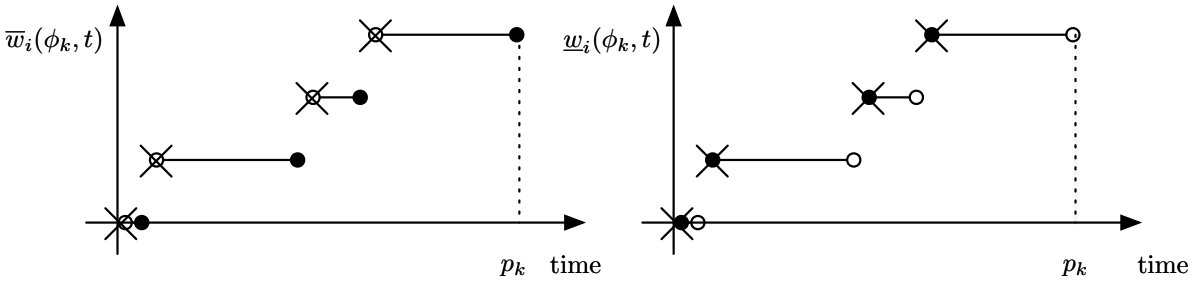

Figure 3: Representaion of the convex cornes for $\underline{w}_i(\phi_k, t)$ and $\overline{w}_i(\phi_k, t)$ for a given ϕ_k.

The set of points $A_{k,i}$ is computed by evaluating $\underline{w}_j(\phi_k, t)$ for each $j \in hp(i) \cap \theta_k$ and for each $\phi_k \in \bar{X}_{k,i}$ and then by extracting the maximum.

For each value ϕ_α of $\bar{X}_{k,i}$, a set C_α of $|hp(i) \cap \theta_k|$ 2-uplets $< \varphi, e >$ is computed, such that $\{< C_\alpha[l].\varphi = \varphi_j(\phi), C_\alpha[l].e = e_{j,0} >: j \in hp(i) \cap \theta_k\}$. We assume that the 2-uplets of C_α are ordered according to increasing values of φ, *i.e.* for $l_1 < l_2$ then $C_\alpha[l_1].\varphi \leq C_\alpha[l_2].\varphi$. The set C_α represents for a given phase $\phi_\alpha \in \bar{X}_{k,i}$, the activation dates of all tasks $\tau_j : j \in hp(i) \cap \theta_k$.

The convex corner of the workload function induced by the set of tasks $\{\tau_j : j \in hp(i) \cap \theta_k\}$ for a given phase ϕ_α is simply represented by

$$\begin{array}{ll}
 & \left\{ \begin{array}{l} W_\alpha[1].x = C_\alpha[1].\varphi \\ W_\alpha[1].y = C_\alpha[1].e \end{array} \right. \\
l \in 2...|hp(i) \cap \theta_k| & \left\{ \begin{array}{l} W_\alpha[l].x = C_\alpha[l].\varphi \\ W_\alpha[l].y = W_\alpha[l-1].y + C_\alpha[l].e \end{array} \right.
\end{array}$$

The W_α represents the convex corner of $\mathcal{F}_{k,i}(\phi_\alpha, t)$. To compute the convex corner of $\hat{\mathcal{F}}_{k,i}(t)$ we need to compare all $W_\alpha : \alpha \in \bar{X}_{k,i}$ for all t and to keep only the maximum value.Then, the set $A_{k,i}$ is computed by keeping all points $p \in \bigcup_{\alpha \in \bar{X}_{k,i}} W_\alpha$ such $\forall p' \neq p \in A$ if $p'.x \leq p.x$ then $p'.y \leq p.y$.

The algorithm to compute $B_{k,i}$ is exactly the same with the set of tasks $ht(i) \cap \theta_k$ and the set $(hp(i) \setminus ht(i)) \cap \theta_k$ for $C_{k,i}$.

8 Tight esponse-ime

Mäki *et al.* propose in [14] an improvement to the approximation response-time analysis presented by Tindell [18], and Palencia *et al.* [16]. The improvement tightens the analysis, *i.e.* makes it less pessimistic, by removing unnecessary overestimation of the interference a task can impose on other tasks.

The fix-point iteration to compute $s_{i,q}$ and $f_{i,q}$ terminates when the sum of all workload functions meets the line from origin with slope 1 (see figure 4). In [14], the authors replace stepped stairs function with slanted stairs (with slope 1) because they not contribute to earlier fix-point convergence. However, in [14] they just consider ceil functions, as $\overline{w}_i$, to compute the response time. Here, the same thing is done with a new slanted stair function to replace floor function, $\underline{w}_i$, to avoid earlier fix-point convergence (see figure 4 to have an example of convergence problem).

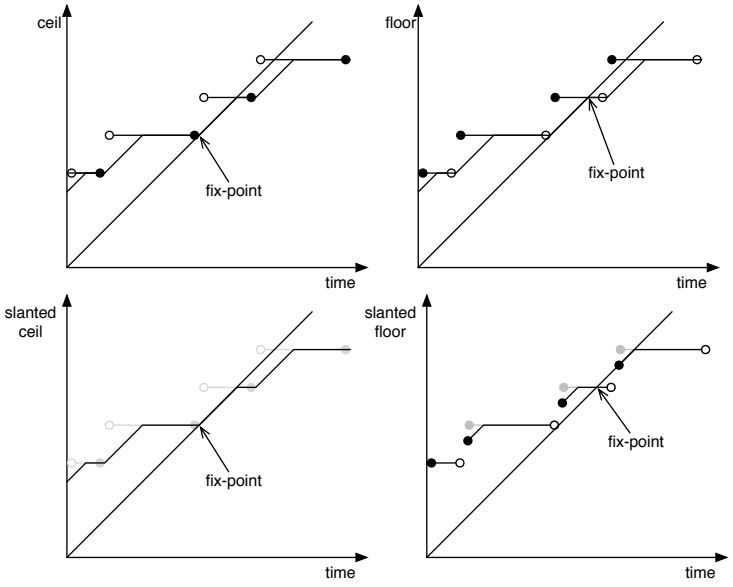

Figure 4: Stepped stairs vs. slanted stairs

We define two slanted stairs functions:

$$\begin{aligned}
\underline{\mathcal{W}}_i(\phi_k,t) &= \left(1+\left\lfloor\frac{t^*}{p_k}\right\rfloor\right)e_{i0}+\underline{x}_i(\phi_k,t)\\
t^* &= t-\varphi_i(\phi_k)\\
\underline{x}_i(\phi_k,t) &= \begin{cases} 0 & t^*<0\\ 0 & t^* \mod p_k \geq e_{i0}-1\\ (t^* \mod p_k)+1-e_{i0} & \text{otherwise}\end{cases}\\
\overline{\mathcal{W}}_i(\phi_k,t) &= \left\lceil\frac{t^*}{p_k}\right\rceil e_{i0}+\overline{x}_i(\phi_k,t)\\
t^* &= t-\varphi_i(\phi_k)\\
\overline{x}_i(\phi_k,t) &= \begin{cases} 0 & t^*\leq 0\\ 0 & t^* \mod p_k = 0\\ 0 & t^* \mod p_k \geq e_{i0}\\ (t^* \mod p_k)-e_{i0} & \text{otherwise}\end{cases}
\end{aligned}$$

We redefine $\widetilde{s}_{i,q}(\mathbf{\Phi})$ and $\widetilde{f}_{i,q}(\mathbf{\Phi})$ as the smallest solution in $\mathbb{N}^+$ of the following fixed-point equation:

$$\widetilde{s}_{i,q}(\mathbf{\Phi},t) = B_i+(q-1)e_{i0}+\sum_{k\in[1..m]}\widetilde{\mathcal{F}}_{k,i}(\phi_k,t)$$

$$\widetilde{f}_{i,q}(\mathbf{\Phi},t) = B_i+qe_{i0}+\widetilde{\mathcal{H}}_{i,q}(\mathbf{\Phi})+\sum_{k\in[1..m]}\widetilde{\mathcal{G}}_{k,i}(\phi_k,t)$$

with

$$\widetilde{\mathcal{F}}_{k,i}(x,t):(0,p_k]\times\mathbb{R}\to\mathbb{R} = \sum_{j\in hp(i)\cap\theta_k}\underline{\mathcal{W}}_j(x,t) \tag{18}$$

$$\widetilde{\mathcal{G}}_{k,i}(x,t):(0,p_k]\times\mathbb{R}\to\mathbb{R} = \sum_{j\in ht(i)\cap\theta_k}\overline{\mathcal{W}}_j(x,t) \tag{19}$$

$$\widetilde{\mathcal{H}}_{i,q}(\mathbf{\Phi}):\Gamma\to\mathbb{R} = \sum_{j\in hp(i)\setminus ht(i)}\underline{\mathcal{W}}_j(\phi_{src(j)},\widetilde{s}_{i,q}(\mathbf{\Phi})) \tag{20}$$

Lemma 26.

$$\forall\mathbf{\Phi}\in\Gamma_i, s_{i,q}(\mathbf{\Phi}) = \widetilde{s}_{i,q}(\mathbf{\Phi})$$
$$\forall\mathbf{\Phi}\in\Gamma_i, f_{i,q}(\mathbf{\Phi}) = \widetilde{f}_{i,q}(\mathbf{\Phi})$$

Proof. See [14]. □

In the same manner than for $s_{i,q}(\mathbf{\Phi})$ and $f_{i,q}(\mathbf{\Phi})$, it is possible to propose an approximation computation of $\widetilde{s}_{i,q}(\mathbf{\Phi})$ and $\widetilde{f}_{i,q}(\mathbf{\Phi})$ that conduct to the definition of the following approximation functions:

$$\breve{s}_{i,q}(\mathbf{\Phi},t) = B_i+(q-1)e_{i0}+\widetilde{\mathcal{F}}_{src(i),i}(\phi_{src(i)},t)+\sum_{k\neq src(i)}\breve{\mathcal{F}}_{k,i}(t)$$

with

$$\breve{\mathcal{F}}_{k,i}(t):\mathbb{R}\to\mathbb{R} = \max_{x\in\bar{X}_{k,i}}\widetilde{\mathcal{F}}_{k,i}(\phi_k,t)$$

and

$$\breve{f}_{i,q}(\mathbf{\Phi}, t) = B_i + qe_{i0} + \widetilde{\mathcal{G}}_{src(i),i}(\phi_{src(i)}, t) + \sum_{k \neq src(i)} (\breve{\mathcal{H}}_{k,i} + \breve{\mathcal{G}}_{k,i}(t))$$

with

$$\breve{\mathcal{G}}_{k,i}(t) : \mathbb{R} \to \mathbb{R} = \max_{x \in \bar{X}_{k,i}} \widetilde{\mathcal{G}}_{k,i}(\phi_k, t)$$

$$\breve{\mathcal{H}}_{k,i} = \max_{\phi_k \in \bar{X}_{k,i}} \sum_{j \in hp(i) \setminus ht(i) \cap \theta_k} \widetilde{W}_j(\phi_k, \widehat{s}_{i,q})$$

9 Experiments

The response-time computation techniques described in the previous sections were implemented in Python in order to evaluate there efficiency. We compared the results of the RTA algorithm described in section 5 to three other RTA methods:

- Approximate Computation: Presented in section 6, this technique guaranty pessimistic results and a slow computation time.
- Fast Response-Time: This technique provide a very fast analysis speed for a much more pessimistic results. It is presented in section 7.
- Tighter Response-Time: Presented in section 8, the aim of this method is the tightness of the results.

Using the implemented techniques and a task generator, we performed a set of simulations in order to support the theory behind the analysis methods.

9.1 Task Generator

We used a task generator in order to generate a set of tasks using the following three elements as parameters: total system load ; number of activation sources ; number of tasks per activation source.

The task generator automatically generates a set of tasks with the following properties:

1. The total system load is proportionally distributed over all the activation sources.
2. Activation sources periods are uniform-randomly distributed in the range 1000 to 1000000.
3. Each offset is uniform-randomly distributed within the associated activation source period.
4. The priority of a task is assigned according to the period of the associated activation source. The lower the period is, the higher the priority will be. If in a task group, more than one task have the same offset, the one with the lower offset will have the higher priority.
5. The execution time is computed by the UUniFast function.
6. The preemption threshold of a task is equal to the task priority.
7. All the tasks are generated with only one execution block.

9.2 Simulation Setup

The simulation setup followed is divided into three operations:

1. Generating 100 task sets using the task generator described in the section above while varying the input parameters (number of activation sources and the number of tasks per activation source). The value of the total system load is fixed to 80%.

2. Calculating the worst case response-time (r_i) of each task (t_i) in the set using the implemented techniques (exact, approximate, fast and tight).

3. Comparing the obtained values and the efficiency of the different techniques.

In order to accurately compare the different techniques, we used the following methods:

1. Evaluating pessimism:

   - %RTA (Response Time approximation): For each task in a task set, we calculated the average increase over the exact computation. For a task t_i, an improvement is defined as $\frac{r_i^{approx}}{r_i^{exact}} - 1$ (same equation applies to the fast and tight techniques).
   - %task (Number of tasks with overestimated response-time) : For each task set, the percentage of tasks in the set with an overestimated response-time compared to the exact computation technique is computed. The results are displayed for the approximate, fast and tight techniques.
   - %WRTA (Response Time approximation for pessimistic results): We calculated the average increase in response-time only the for tasks that meet increase over the exact technique.

2. time (Evaluating time efficiency): For each task set, we estimated the computation time of the three approximation methods (approx, fast and tight).

9.3 Evaluating Pessimism

The pessimism of each computation technique is evaluted by varying the parameters of the task generator. In the first simulation setup (Table 1), the number of tasks per activation source varies from 2 tasks to 13, while maintaining the total system load at 80% and the total number of activation sources at 3.

As expected, the results shows that the tight computation technique is the less pessimistic method. The approximation technique is more pessimistic than the tight approach but less than the fast computation. In a matter of fact, the fast approach can reach a high level of overestimation with 60% of tasks with improvement over the exact analysis, and with an average overestimation of 30%. We can also note that the overestimation of all of the three approximate methods increases remarkably when the number of tasks per activation source is increased. For example, if we take a look at the results for the fast computation technique, the average overestimation increases from 6.6% to 36% when the number of tasks vraies from 2 to only 9 tasks per activation source. As for the computation time, the fast computation maintain a steady and fast computation time around 0.1 seconds. The approx and tight computation time increases slowly to reach 11.1 seconds for the approx technique and 24.4 seconds for the tight technique.

In the second simulation setup (Table 2), the number of activation sources varies from 2 activation sources to 13. An 80% total system load and 3 tasks per activation source are fixed.

Unlike the previous results, these numbers indicates that varying the number of activation sources does not affect the approximation methods pessimism in a noteworthy way. In fact, the average overestimation of the approximation approach for 3 tasks per activation sources evolves around 6%. For the fast approach, the average overestimation is steady at a value of approximately 19%. And for the tight approach, the average

		Number of Tasks Per Activation Source							
		2	3	4	5	6	7	8	9
%RTA	Approx	0.4	1.3	1.9	1.9	2.7	2.5	3.22	3
	Fast	2	5.4	8.1	9.4	12.6	15.9	19.9	22.6
	Tight	0.04	0.2	0.6	0.5	0.8	0.7	1.2	1.1
%task	Approx	4.2	13.6	14.8	18.8	24.8	25.1	30.3	32.6
	Fast	15.5	31	35.1	41.4	49.5	54.52	57.7	60.3
	Tight	0.7	2.2	5.7	7.3	10.1	9.1	14.4	15.8
%WRTA	Approx	1.3	4.9	7.6	6.2	9	8.6	9.9	8.1
	Fast	6.6	12	19.7	19.5	22.6	27.1	32.5	36
	Tight	0.1	1.2	3.5	3.1	4.6	4.2	6.6	5.6
time (s)	Approx	0.01	0.06	0.2	0.5	0.9	1.8	3.5	11.1
	Fast	0.01	0.02	0.04	0.05	0.06	0.08	0.1	0.2
	Tight	0.4	0.7	0.8	1.5	2.3	5.8	8.3	24.4

Table 1: Response-Time computation for 80% utilization and 3 activations sources.

overestimation is approximately at 3%. As for the computation time, the most time efficient technique is still the fast approach, followed by the approx approach and then the tight technique. We can also note that this time, the computation time for the approx and tight techniques increases very rapidly to reach 90 seconds for the approx method, and 222 seconds for the tight method. These results are explained by the fact that the number of combination for the phasing vector $\mathbf{\Phi}$ increases exponentially when the number of activation sources increases.

9.4 Evaluating Time Efficiency

In order to evaluate the time efficiency of the three approximation techniques, a third simulation setup (Table 3) is executed. In this setup, the number of tasks per activation source varies while maintaining the total system load at 80% and the number of activation sources at 10.

As we can see in the Table 3, the fast computation method delivers a very fast computation time. As for the tight and approx method, the execution time increases rapidly to reach approximately 100 seconds for the approx technique and 200 seconds for the tight technique.

10 Conclusion

In this paper, we have studied the problem of computing the response times of real-time jobs in multitask softwares built on top of AUTOSAR OS compliant kernels. By doing so, we have obtained a rather complex model of multitask softwares and we have developed, proven and prototyped adequate algorithms.

We have studied the possibility to trade tightness against computation time by using approximation techniques [18, 16, 15]. Without surprise, the different techniques have their advantages and drawbacks. Next study will be on set of tasks with same priority. The same kind of techniques have to be developed and experimented.

		Number of Activation Source							
		2	3	4	5	6	7	8	9
%RTA	Approx	1.4	1.7	1.6	1.5	1.6	1.2	1.3	1.7
	Fast	4.2	8.33	10.1	13	14	14.4	15.6	16
	Tight	0.03	0.3	0.3	0.3	0.3	0.3	0.4	0.5
%task.	Approx	9.8	13.5	15.4	19	20.3	21.1	22.6	28.5
	Fast	18.2	37	50.6	63.9	68.8	74.1	78.9	83.1
	Tight	0.8	3.4	4.7	5.1	7.4	7.2	8.8	10.9
%WRTA	Approx	5.1	6.6	7.3	6.6	6.4	5.1	5.3	5.7
	Fast	12.8	18.8	19.7	20	19.8	19	19.4	20
	Tight	0.1	1.3	2.4	2.3	2.5	2.7	3.3	3.5
time (s)	Approx	0.01	0.02	0.2	0.4	1.36	8.2	38.8	90.6
	Fast	0.01	0.01	0.04	0.02	0.03	0.08	0.1	0.2
	Tight	0.02	0.04	0.7	1	2.8	18.7	79.6	222.1

Table 2: Response-Time computation for 80% utilization and 3 tasks per activations sources.

		Number of Tasks Per Activation Source				
		1	2	3	4	5
time (s)	Approx	0.01	0.2	4	38.4	107.4
	Fast	0.01	0.04	0.1	0.2	0.3
	Tight	0.02	0.5	10.4	87.7	222.7

Table 3: Response-Time computation for 80% utilization and 10 activations sources.

References

[1] N. C. Audsley. Deadline-Monotonic Scheduling. Technical Report YCS 146, Departement of Computer Science, University of York, 1990.

[2] AUTOSAR. Specification of Operating System. Technical Report v3.0.1, AUTOSAR GbR, 2008.

[3] Theodor P. Baker. A stack-based resource allocation policy for realtime processes. In *IEEE Real-Time Systems Symposium (RTSS)*, Lake Buena Vista, FL, USA, December 1990. IEEE Computer Society.

[4] F. Bimbard and L. George. FP/FIFO feasibility conditions with kernel overheads for periodic tasks on an event driven osek system. In *International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC)*, pages 566–574. IEEE Computer Society, 2006.

[5] Giorgio C. Buttazzo. *Hard Real-time Computing Systems: Predictable Scheduling Algorithms And Applications (Real-Time Systems Series), 3rd ed.* Springer, 2011.

[6] Manuel Coutinho, José Rufino, and Carlos Almeida. Response time analysis of asynchronous periodic and sporadic tasks scheduled by a fixed-priority preemptive algorithm. In *Euromicro Conference on Real-Time Systems (ECRTS)*, Prague, Czech Republic, July 2008. IEEE Computer Society.

[7] P. E. Hladik, A. M. Déplanche, S. Faucou, and Y. Trinquet. Schedulability analysis of OSEK/VDX applications. In *International Conference on Real-Time and Network Systems (RTNS)*, pages 131–140, Nancy, France, March 2007.

[8] P.-E. Hladik, A.-M. Déplanche, S. Faucou, and Y. Trinquet. Schedulability analysis of OSEK/VDX applications. In *International Conference on Real-Time and Network Systems (RTNS)*, pages 131–140, Nancy, France, March 2007.

[9] P.-E. Hladik, S. Faucou, A.-M. Déplanche, and Y. Trinquet. Adequacy between AUTOSAR OS specification and real-time scheduling theory. In *IEEE Symposium on Industrial Embedded Systems (SIES)*, pages 225–233, Lisbon, Portugal, July 2007.

[10] M. Joseph and P. Pandya. Finding response times in a real-time system. *The Computer Journal*, 29(5), 1986.

[11] John P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *IEEE Real-Time Systems Symposium (RTSS)*, Lake Buena Vista, FL, USA, December 1990. IEEE Computer Society.

[12] Wang Lei, Zhaohui Wu, and Mingde Zhao. Worst-case response time analysis for OSEK/VDX compliant real-time distributed control systems. In *Annual IEEE International Computer Software and Applications Conference (COMPSAC)*, pages 148–153, Hong Kong, September 2004. IEEE Computer Society.

[13] Jukka Mäki-Turja and Mikael Nolin. Efficient response-time analysis for tasks with offsets. volume 0, page 462, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[14] Jukka Mäki-turja and Mikael Nolin. Tighter response-times for tasks with offsets. In *In Proc. of the 10 th International conference on Real-Time Computing Systems and Applications (RTCSA'04)*, 2004.

[15] Jukka Mäki-Turja and Mikael Nolin. Efficient implementation of tight response-times for tasks with offsets. *Real-Time Systems*, 40(1):77–116, 2008.

[16] J. C. Palencia and M. González Harbour. Schedulability analysis for tasks with static and dynamic offsets. In *IEEE Real-time Systems Symposium (RTSS)*, pages 26–37, Madrid, Spain, December 1998. IEEE Computer Society.

[17] L. Sha, R. Rajkumar, and J.P. Lehoczky. Priority inheritance protocols: An approach to real-time synchronization. *IEEE Transactions on Computers*, 39(9):1175–1185, 1990.

[18] Ken Tindell. Adding time-offsets to scheduling analysis. Technical Report YCS 221, Department of Computer Science, University of York, 1994.

[19] Y. Wang and M. Saksena. Scheduling fixed-priority tasks with preemption threshold. In *IEEE International Conference on Real-Time Computing Systems and Applications (RTCSA)*, pages 328–337, Hong Kong, December 1999. IEEE Computer Society.

Appendix: Complexity

Consider a set $N = [1...n]$ and a function $f(t) = a + \sum_{i \in N} \left\lceil \frac{t}{b_i} \right\rceil c_i$ with $\sum_{i \in N} \frac{c_i}{b_i} < 1$ and $(a, c_i) \in \mathbb{N}^{n+1}$. We have $f(t) < a + \sum_{i \in N} c_i + \sum_{i \in N} \frac{t}{b_i} c_i$. We denote $\alpha = a + \sum_{i \in N} c_i$ and $\beta = \sum_{i \in N} \frac{c_i}{b_i}$, thus $f(t) < \alpha + \beta t$.

The solution $f^* = \min\{t = f(t) \mid t > 0\}$ is the fixed point computed by:

$$\begin{cases} u^0 & = & \alpha \\ u^{n+1} & = & f(u^n) \end{cases}$$

Remark that :

- values of the series u belong to $\mathbb{N}$,
- $u^{n+1} \geq u^n$,
- $f^* \leq \frac{\alpha}{1-\beta}$,

thus the f^* can be found in $[\alpha, \lambda]$ with $\lambda = \frac{\alpha}{1-\beta}$.

Moreover the function f is a step function with a number of values on a interval of size l bounded by $\sum_{i \in N} \left\lceil \frac{l}{b_i} \right\rceil$. So, the number of iterations of f^* is bound by $\gamma = \sum_{i \in N} \left\lceil \frac{\lambda - \alpha}{b_i} \right\rceil = \sum_{i \in N} \left\lceil \frac{\alpha\beta}{(1-\beta)b_i} \right\rceil$.