Uniprocessor EDF Scheduling of AVR Task Systems*

Zhishan Guo
University of North Carolina at Chapel Hill
201 S Columbia Street
Chapel Hill, North Carolina 27599
zsguo@cs.unc.edu

Sanjoy K. Baruah
University of North Carolina at Chapel Hill
201 S Columbia Street
Chapel Hill, North Carolina 27599
baruah@cs.unc.edu

ABSTRACT

The adaptive varying-rate (AVR) task model has been proposed as a means of modeling certain physically-derived constraints in CPS's in a manner that is more accurate (less pessimistic) than is possible using prior task models from real-time scheduling theory. Existing work on schedulability analysis of systems of AVR tasks is primarily restricted to fixed-priority scheduling; this paper establishes schedulability analysis results for systems of AVR and sporadic tasks under Earliest Deadline First (EDF) scheduling. The proposed analysis techniques are evaluated both theoretically via the speedup factor metric, and experimentally via schedulability experiments on randomly-generated task systems.

Categories and Subject Descriptors

D.4.1 [**Operating systems**]: process management—*scheduling*; D.4.7 [**Operating systems**]: organization and design—*real-time systems and embedded systems*

1. INTRODUCTION AND MOTIVATION

The research described in this paper is part of the effort that seeks to define and better understand the role that real-time scheduling theory should play within the emergent discipline of cyber-physical systems.

Rigorous model-based design (MBD) approaches are widely used in the development of complex cyber-physical systems. In such approaches, one typically *specifies a model* of the system under development using some abstract modeling formalism such as Timed Automata [1] (usually within the context of a tool such as UPPAAL [7]) or Synchronous Reactive models [8, 14, 9], then *demonstrates the correctness of the model* by proving that the model possesses certain safety and progress properties that together imply correctness, and finally *implements the model* upon an implementation platform in a manner that is consistent with the assumptions made in the modeling formalism. In this approach, real-time scheduling theory plays a role in the implementation step: it is used to help obtain resource-efficient implementations, upon actual implementation platforms, of formal models that have been rigorously proved to be correct. To accomplish this step, the elements of the abstract model used during the MBD process are mapped on to the recurrent task models such as periodic and sporadic tasks [18, 19] that are used in real-time scheduling theory, and results from real-time scheduling theory then applied to obtain resource-efficient implementations of the resulting recurrent task systems. Unfortunately, there is not always a perfect fit between the models of MBD and the lower-level ones of scheduling theory, and consequently the mapping is achieved by introducing additional *pessimism*. (For example, a recurrent physical phenomenon that does not occur periodically, represented by a Timed Automaton in the MBD process, may need to be modeled by a periodic task that is assigned a period parameter equal to the smallest duration between successive occurrences of the physical phenomenon.) Such pessimism in the mapping process results in resource under-utilization during run-time: not all the resources that are needed to guarantee execution of the more pessimistic lower-level model are required by the higher-level model, and the additional assigned resources remain unused during run-time.

In an attempt to reduce such implementation inefficiency and thereby be able to obtain more resource-efficient implementations of cyber-physical systems, real-time scheduling theory has recently begun looking at developing, from first principles, new low-level *task models that are inspired by requirements of actual physical systems*. Systems represented using these new models cannot always be analyzed using prior results from real-time scheduling theory; rather, new analysis methodologies (that may be based upon prior insights and techniques from scheduling theory) need to be developed. Ensuring that such new models are both (i) useful to the developer of CPS's, and (ii) amenable to efficient analysis and implementation, seems to require much closer interaction between the engineers developing a CPS and the scheduling-theory experts seeking to devise the model and formal techniques for its analysis — they would ideally work hand-in-hand from the early stages of system development to concurrently develop the appropriate models and the anal-

*Work supported by NSF grants CNS 1115284, CNS 1218693, CPS 1239135, CNS 1409175, and CPS 1446631, AFOSR grant FA9550-14-1-0161, ARO grant W911NF-14-1-0499, and a grant from General Motors Corp.

ICCPS '15, April 14 - 16, 2015, Seattle, WA, USA

http://dx.doi.org/10.1145/2735960.2735976

```
task sample_task {
   rpm = read_speed();
   f1();
   if (rpm < 4000) {
     f2();
   }
   if (rpm < 2000) {
     f3();
   }
}
```

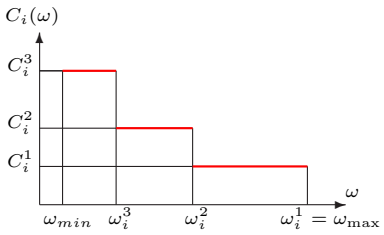


Figure 1: An example AVR task with WCET dependent to the engine rotation speed.

ysis methodologies.

The research that we describe in this paper can be viewed as an instantiation of this approach to CPS design and analysis. Specifically, it deals with the modeling of recurrent processes in cyber-physical systems for which each activation of the recurrent process is triggered by the values of variables describing the state of the physical system. Such processes abound in cyber-physical systems: for example, height detection in avionic systems is activated more frequently at lower altitudes; sensor acquisition in mobile robots often depends on the robot location (e.g., whether it is approaching edge areas); while fuel injection in the Engine Control Unit (ECU) of an automobile is dependent upon the position of each piston (which is a function of the crankshaft angular position).

Consider the ECU, a typical hard real-time system as an example, on which some tasks execute at a varying rate depending upon engine speed. These tasks calculate the quantity of fuel to be injected, and the precise instants at which the injected fuel is to combusted, in order to achieve optimal engine performance (maximize the thrust obtained, avoid engine knocking, etc.). A key challenge in the schedulability analysis of collections containing one or more such tasks is that not only the periods and deadlines, but also the execution requirements, are determined by specific angular positions of the crankshaft, which relate to the engine speed [12]. Since engine behavior is generally more stable when running at higher speeds, some functions (that must execute at lower speeds) do not need to execute at higher speeds. Figure 1 adapted (from [12]) a typical piece of pseudocode for a task with worst-case execution time (WCET) dependent on crankshaft rotation frequency, and further illustrates such relationship under a WCET-speed function (to be explained in detail later on).

The challenge with this type of task is that the time between successive activations is neither constant nor arbitrary; rather, it depends on the engine rotation speed, which varies within a specified range with specified maximum acceleration and deceleration. If we were to use classical real-time scheduling analysis (for example, the sporadic model [18, 19]), the relationship between activation period and WCET can only be modeled by introducing unnecessary and excessive pessimism.

Related Work. This problem, formalized as the Adaptive Varying-Rate (AVR) tasks scheduling problem, was first introduced to the real-time system community in a keynote address at ECRTS [12] in 2012. Some early work on understanding the scheduling of AVR tasks was done by [17] on a simplified model, where only a *single* task, that is assigned highest priority, is analyzed. Pollex et al. [22] later considered systems with multiple tasks, but constant engine speed. This work was later extended to the case with angular acceleration in [21] — by considering the maximum execution time and the minimum inter-arrival time starting from different speeds, sufficient schedulability conditions were derived.For the preemptive uniprocessor fixed-priority scheduling of AVR tasks, Davis et al. [13] present an Integer Linear Programming (ILP) based sufficient dynamic schedulability test; Biondi et al. [10] have recently proposed a search tree based method based on a Brute-Force approach with pruning rules.

To the best of our knowledge, [11] is the only prior work to study the preemptive uniprocessor dynamic-priority scheduling of systems of AVR tasks (which is the subject of this paper). [11] derives a sufficient, but not necessary, schedulability condition under the added simplifying (but not necessarily valid) assumption that when an AVR task switches to a new mode due to a positive acceleration, the next job will run with the computation time associated with the new mode.

Within the context of "traditional" real-time scheduling, schedulability analysis has also been developed for a variety of different task and/or system models, from the traditional periodic and sporadic task models to rate-based abstractions [15, 16], multi-frame models [20, 5], digraph models [23, 24], and the varying-speed platform model [2, 3]. Unfortunately, AVR tasks cannot be expressed under even the most general existing models without introducing considerable additional pessimism, since these tasks may change their execution modes asynchronously (under different thresholds), and multiple changes of execution mode may take place by a single task during consecutive execution of its jobs.

Contribution and Organization. The remainder of this paper is organized as follows. Sec. 2 formally describes the system model and provides additional definitions for AVR tasks. Sec. 3 proposes an efficient sufficient schedulability test for a kind of AVR task system called *implicit mixed* AVR task systems, and performs speedup factor analysis under practical assumptions on system parameters. A further improvement that yields necessary and sufficient schedulability analysis is also discussed. We consider AVR task systems with *constrained deadlines* in Sec. 4, and derive approximate schedulability analysis via transformation to the digraph-based task model [23, 24]. Sec. 5 reports schedulability experiments that compare the proposed methods to the current state of the art.

2. SYSTEM MODEL

We consider the EDF scheduling of a set of n tasks on a preemptive uniprocessor. Each task τ_i generates an infinite sequence of jobs $J_{i,1}, J_{i,2}, ...$, and can either be a *regular sporadic* task, characterized by a (fixed) WCET c_i, period T_i, and relative deadline D_i, or an *adaptive varying-rate (AVR)*

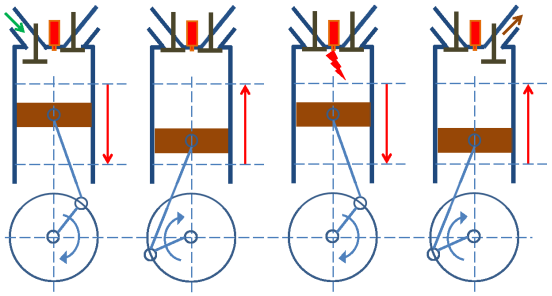


Figure 2: The four strokes (intake, compression, power, and exhaust) of a typical car engine.

task, where all three parameters[1] are variables. We use the notation $\tilde{\tau}_i$ to denote an AVR task. The activation pattern and functionality of an AVR task $\tilde{\tau}_i$ are determined by the physical evolution of the engine. Specifically, each subsequent job $J_{i,k}$ is released when the crankshaft reaches a predefined angular position within each *revolution*, and the inter-arrival time between two consecutive jobs $J_{i,k}$ and $J_{i,k+1}$ is denoted as the activation period $T_{i,k}$, which is a function of the crankshaft rotation speed ω of the engine.

We point out that there are various kinds of car engines — they may be operated with different types of fuel (either petrol or diesel), and more importantly the number of cycles per crankshaft revolution may vary. Figure 2 sketches the basic cycles (intake, compression, power, and exhaust) of a four-stroke engine, where the crankshaft rotates for two cycles (with a total angle of 4π radians) each revolution. Two-stroke engines also exist and are sometimes used in aircraft and marine engines, as well as some trucks and motorcycles, due to a more compact size, a lighter weight, and greater efficiency. However, due to exhaust pollution, two-stroke engines are losing out to, and being replaced by, four-stroke engines in many applications, especially in modern cars (which is our focus). In this work we therefore associate AVR tasks with crankshaft rotations for a cumulative angle of 4π rather than 2π as in some previous works; however the essence of our results remain unchanged if 2π (or an altogether different activation angle) is used instead.

The angular speed at time t is denoted as $\omega(t)$, which is assumed bounded within a specified interval $[\omega_{\min}, \omega_{\max})$. The angular speed may vary during run time, and the angular acceleration $\alpha(t)$ is also assumed to be bounded within a specified interval: $\alpha \in [\alpha^-, \alpha^+]$.

Unless otherwise specified, the **measurement units** for time, angular speed, and angular acceleration (deceleration) in this work are seconds (sec), radian per second (rad/sec), and radian per second per second (rad/sec^2), respectively. Note that revolutions per minute (rpm) and rpm per second (rpm/sec) are widely used for engine rotation measurements in industry; the relationship $1\ rpm = 0.10472\ rad/sec$ translates between this convention and our chosen units.

[1]In keeping with convention, we will refer to the minimum duration that must elapse between the release time of $J_{i,k}$ and $J_{i,k+1}$ as the *period* of $J_{i,k}$ (see Definition 1), even though this quantity is not a true "period" since the jobs are not necessarily invoked periodically.

As depicted in Figure 1, an AVR task $\tilde{\tau}_i$ is composed of M_i *modes*: $\mathcal{M}_i = \{(C_i^m, \omega_i^m), m = 1, 2, ..., M_i\}$, where $\omega_i^{M_i+1} = \omega_{\min}$, and $\omega_i^1 = \omega_{\max}$. As engine speed falls in the range $[\omega_i^m, \omega_i^{m+1})$, mode $\mathcal{M}_i^m$ is triggered when job activation occurs, and the AVR job's WCET is C_i^m. Figure 3(a) depicts how the varying angular speed may lead a task into different modes, and result in jobs of different WCETs (J_{k-1} and J_k). It further shows how periods of jobs (released by an AVR sporadic task) depend on the system states (e.g., the varying angular speed). Given the same angular speed of ω_k at job J_k's activation instant (i.e., the very beginning of its cycle), three possible periods are shown, where two threshold ones are dashed (and represented by different colors). A constant angular speed of ω may lead to a fixed period of $4\pi/\omega$, while acceleration (or deceleration) may cause the shrinking (or extension) to the period of the released job.

When a job is activated, its WCET is determined by the angular speed of the system at its release, while its period may vary due to dynamic system behaviors. For the convenience of later discussions, we formally define the period of an AVR task as the following.

Definition 1 (Activation Period). *Given an activation angular speed ω, the* activation period $T_i(\omega)$ *of (a job of) an AVR task $\tilde{\tau}_i$ is defined as the minimum possible duration that must elapse between this release and the release of the succeeding job of the task. Using standard results from physics concerning the relationships between distance, speed, and acceleration, it is easily shown that*

$$T_i(\omega) = \begin{cases} \frac{\sqrt{\omega(t)^2 + 8\pi\alpha^+} - \omega(t)}{\alpha^+}, & \text{if } \omega(t) \le \Omega(4\pi), \\ \frac{8\pi + (\omega_{\max} - \omega(t))^2/\alpha^+}{2\omega_{\max}}, & \text{if } \omega(t) > \Omega(4\pi); \end{cases} \quad (1)$$

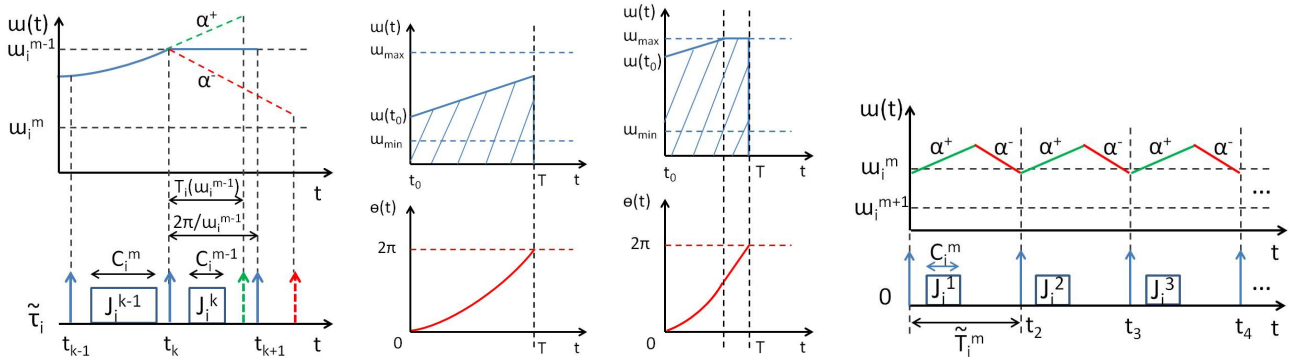
where α^+ is the maximum engine acceleration, $\omega_{\max}$ is the upper bound of engine rotation speed, and threshold $\Omega(\cdot)$ is given by the following equation to determine whether the initial speed $\omega(t_0)$ is fast enough for the system to possibly reach the upper bound of its angular speed ($\omega_{\max}$) when rotated for a certain angle θ:

$$\Omega(\theta) = \sqrt{\max\{\omega_{\max}^2 - 2\theta\alpha^+, \omega_{\min}^2\}}. \quad (2)$$

Note that although the job is released at time t_0 with an angular speed of $\omega(t_0)$, its period may be shorter than $4\pi/\omega(t_0)$ due to possible engine acceleration[2]. With an initial speed $\omega(t_0) = \Omega$ (at the threshold), the crankshaft should be able to reach $\omega_{\max}$ right at the end of one cycle of rotation; i.e., $(\omega_{\max} + \Omega)(\frac{\omega_{\max} - \Omega}{\alpha^+})/2 = \theta$, which leads to Equation (2) above.

Since activation period is defined as the minimum possible separation, we need to consider the "worst case" angular speed changes during runtime when estimating the period of a task. Figure 3 illustrates two possible scenarios under accelerated engine behaviors to determine current job activation period T. Subfigure (b) shows the case when $\omega(t)$

[2]Recall that it is assumed that all engine-triggered tasks are generated at *each cycle*; i.e., when the crankshaft rotates for a cumulative angle of 4π.



(a) Periods vs. Engine Speeds (b) Act. Period I (c) Act. Period II (d) A Release Pattern

Figure 3: Activation period analysis for AVR tasks.

cannot possibly reach $\omega_{\max}$ within one cycle, while Subfigure (c) shows the other case, and $\theta(t)$ denotes the rotated angle since current job activation (at time t_0).

Given any angular speed $\omega(t_0) \in [\omega_{\min}, \omega_{\max})$, the two possible cases can be treated separately with the help of the threshold Ω. The corresponding activation periods T for the task can be derived by assigning 4π to the rotated angles $\theta(T)$ of the crankshaft during such period (also reflected by the shaded areas of Figures 3(b) and 3(c)), which is equivalent to (1).

For *implicit* tasks with job deadlines at the end of their cycles, their relative deadlines are the same as the activation periods; i.e., $\forall i, D_i(\omega) = T_i(\omega)$. For *constrained* tasks with (relative) deadlines specified as a certain angle $\Delta_{\theta_i} < 4\pi$ after activation, $D_i(\omega)$ may be calculated by (1) by replacing π with $\Delta_{\theta_i}/4$.

3. IMPLICIT SYSTEMS

In this section, we start out considering system only composed of *implicit* tasks, i.e., the ones with relative deadlines equal to the periods for both regular and AVR tasks. For implicit AVR tasks, no timing constraint is violated so long as each job finishes its execution before its successor is released. Given current angular speed $\omega(t_0)$, the activation period $T_i(\omega(t_0))$ given by Eqn (1) may serve as a *safe* relative deadline for each AVR task in an implicit task system, since no successor job may be triggered prior to then according to Definition 1.

3.1 A Utilization Based Test

DEFINITION 2 (UTILIZATION). *Similar to a regular task, for AVR task $\tilde{\tau}_i$, we define* utilization in mode m *as the ratio of WCET and the possible minimum activation period of the mode:*

$$U_i^m = C_i^m / T_i^m, \tag{3}$$

where $T_i^m = T_i(\omega_i^m)$; and further refer to the maximum utilization over all modes as the utilization *of the task:*

$$U_i = \max_m \{U_i^m\}. \tag{4}$$

With this definition, we are able to provide the first schedulability test to the system composed of regular tasks and AVR ones:

$$\sum_{i=1}^{n} U_i \leq 1. \tag{5}$$

THEOREM 1. *Schedulability Test (5) is a sufficient EDF-schedulability test for an implicit system.*

Proof: Since we only focus on implicit systems, where U_i also serves as the *density* for regular tasks, all we need to show is that the *execution requirement* by any AVR task $\tilde{\tau}_i$ within a time demand of length t does not exceed $U_i t$. Here execution requirement is defined as the total WCETs for jobs with both release time and deadlines lying inside the interval.

During engine spinning, any AVR task may switch modes within the given M_i modes, where it releases a job with WCET of C_i^m in Mode m for a period of at least T_i^m. Since U_i serves as an upper bound of the ratios $C_i/T_i^m, m = 1, 2, ..., M_i$, the execution requirement per time unit for the task will not exceed U_i in any mode. Thus total time demand for any interval with length t will not exceed $U_i t$. □

3.2 Speedup Factor

We now seek to quantify, via the *speedup factor* metric, the "distance" of this schedulability test from optimality; to our knowledge, this is the first such analysis of schedulabilty tests for AVR task systems.

DEFINITION 3 (SPEEDUP FACTOR). *A schedulability test has speedup factor $s, s \geq 1$, if any mixed task set that is schedulable by any algorithm on a unit-speed processer, will be deemed schedulable by this test upon a processor that is s times as fast.*

Consider the release pattern of an AVR task and its corresponding engine behavior shown in Figure 3(d).

We restrict that all AVR tasks share the same mode and angular speed ω^m when their utilizations $U_i = \max_m\{C_i^m/T_i^m\}$ are maximized. Under such case, we further define an *adjusted period* $\tilde{T}_i^m$ as the minimum length for the system to finish at an angular speed same to the initial one after a cycle. To reach such adjusted period, the system needs to speed up at the highest acceleration, and then decelerate as much as possible after a certain while (shown in Figure 3(d)).

Given α^+, α^-, $\omega_{\min}$, and $\omega_{\max}$, we know that if initial angular speed ω^m reaches a certain value $\tilde{\Omega}$, the system may reach $\omega_{\max}$ within one cycle and drop back to ω^m. Similar to the derivation of Ω in Sec. 2, when setting the highest angular speed within this cycle (the value of the upper points of inflection in Figure 3(d)) as $\omega_{\max}$, and denoting the corresponding activation period as Γ, we know that:

$$\begin{cases} \tilde{\Omega}\Gamma + (\omega_{\max} - \tilde{\Omega})\Gamma/2 = 4\pi; \\ (\omega_{\max} - \tilde{\Omega})/\alpha^+ + (\tilde{\Omega} - \omega_{\max})/\alpha^- = \Gamma. \end{cases} \tag{6}$$

Getting rid of Γ from (6) results in the following Equation (7), where the threshold $\tilde{\Omega}$ can be used to determine whether the initial speed ω^m is fast enough for the system to possibly reach its upper bound angular speed limit $\omega_{\max}$:

$$\tilde{\Omega} = \sqrt{\max\{\omega_{\max}^2 - 8\pi/\Delta_\alpha, \omega_{\min}^2\}}, \tag{7}$$

where

$$\Delta_\alpha = 1/\alpha^+ - 1/\alpha^-. \tag{8}$$

Furthermore, the length of the *adjusted period* $\tilde{T}_i^m$ can be derived:

$$\tilde{T}_i^m = \begin{cases} \Delta_\alpha(\sqrt{(\omega^m)^2 + 8\pi/\Delta_\alpha} - \omega^m), & \text{if } \omega^m \le \tilde{\Omega}, \\ \big(8\pi + (\omega_{\max} - \omega^m)^2\Delta_\alpha\big)/(2\omega_{\max}), & \text{if } \omega^m > \tilde{\Omega}. \end{cases} \tag{9}$$

When considering the time interval $[0, \tilde{T}_i^m)$, the total time demand of the system within this interval may reach $\sum_{i|\tau_i} U_i\tilde{T}_i^m + \sum_{i|\tilde{\tau}_i} C_i^m$ [3]. Thus the following condition is necessary for this system to be schedulable:

$$\sum_i \tilde{U}_i \le 1, \tag{10}$$

where $\tilde{U}_i = U_i$ if τ_i is regular, and $\tilde{U}_i = C_i^m/\tilde{T}_i^m$ for m that maximizes the ratio C_i^m/T_i^m if $\tilde{\tau}_i$ is AVR. Denote β as the AVR task utilization ratio; i.e.,

$$\beta = (\sum_{i|\tilde{\tau}_i} U_i)/(\sum_{j|\tau_j} U_j + \sum_{i|\tilde{\tau}_i} U_i) \in [0, 1]; \tag{11}$$

[3]We use $i|\tilde{\tau}_i$ to denote AVR tasks, and $i|\tau_i$ as other tasks (with fixed period).

and

$$\eta(\omega^m) = \max_i\{\tilde{T}_i^m/T_i^m\} \ge 1. \tag{12}$$

THEOREM 2. *Schedulability Test (5) has a speedup factor of no larger than* $1/\big(1 - \beta + \beta/\eta(\omega)\big)$.

Proof: According to the discussion above, the speedup factor of the sufficient condition (5) satisfies:

$$\begin{aligned} s &\le \frac{\sum_j U_j}{\sum_i \tilde{U}_i} \\ &= \frac{\sum_{j|\tau_j} U_j + \sum_{i|\tilde{\tau}_i} U_i}{\sum_{j|\tau_j} U_j + \sum_{i|\tilde{\tau}_i} C_i/\tilde{T}_i} \\ &= 1/\Big(\frac{\sum_{j|\tau_j} U_j + \sum_{i|\tilde{\tau}_i} \frac{C_i}{T_i}/\frac{\tilde{T}_i}{T_i}}{\sum_{j|\tau_j} U_j + \sum_{i|\tilde{\tau}_i} U_i}\Big) \\ &\le 1/\Big(\frac{\sum_{j|\tau_j} U_j + (\sum_{i|\tilde{\tau}_i} \frac{C_i}{T_i})/\max_i \frac{\tilde{T}_i}{T_i}}{\sum_{j|\tau_j} U_j + \sum_{i|\tilde{\tau}_i} U_i}\Big) \\ &= 1/\big(1 - \beta + \beta/\eta(\omega)\big). \end{aligned} \tag{13}$$

□

Note that this approximation ratio may not be *tight* since our comparison is based upon a specific given scenario which may not occur. However, (13) does provide an upper bound upon the speedup factor. We now show that this upper bound is already quite promising for the analysis to petrol car engines nowadays.

From [25] we know that current vehicles have an acceleration α^+ no greater than 13000 *rpm/sec* (this occurs when the transmission is out-of-gear and the engine is only accelerating based upon its own internal masses). Since the acceleration is too fast for an analogue tachometer to track accurately, precise α^+ value is provided neither in industrial reports nor in research papers. Moreover, engine normally decelerates at a much larger absolute rate, which means $1/\alpha^+ < \Delta_\alpha \le 2/\alpha^+$ holds for vehicle engines (please refer to (8) for Δ_α's definition).

Figure 4 depicts the derived upper bound of the approximation ratio s of the given schedulability test over a range of different angular speeds ω (since η is a function of ω). Some typical values of acceleration parameters α^+ and Δ_α are shown using different lines. Here we choose $\omega_{\min} = 800$ *rpm*, $\omega_{\max} = 8000$ *rpm*, and the utilization ratio of AVR tasks is (pessimistically) assumed to be half; i.e., $\beta = 0.5$.

We observe that a larger angular speed upper bound $\omega_{\max}$ may cause a slight increase of the speedup factor, while faster deceleration results in a significant decrease of the speedup. Nevertheless, for the given range of angular speed $\omega \in [800, 8000]$, the speedup remains at a relatively low value (< 1.14), which indicates that the proposed schedulability test (5) is "wasting" no more than about $(1 - 1/1.14)$ or ≈ 13 percent of the processor capacity.

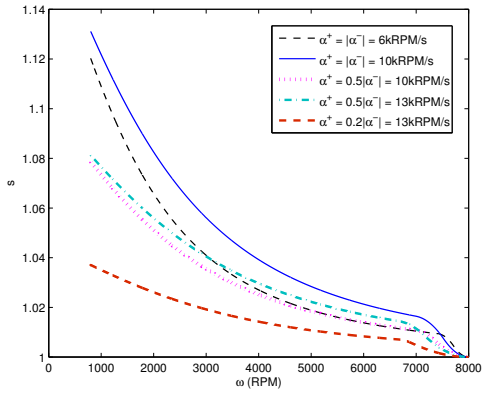


Figure 4: Speedup factor of Schedulability Test (5) under various typical parameter choices.

REMARK 1. *Theoretically speaking, in an extreme case when $\alpha^- \to -\infty$, we have $\Delta_\alpha \to 1/\alpha^+$, $\eta \to 1$, and thus a speedup factor of $s \to 1$. This indicates that the schedulability test based on Equation (5) is asymptotically optimal upon engines where deceleration can occur at a very great rate (equivalently, the crankshaft rotation may stop almost instantaneously).*

3.3 A Pragmatic Improvement

Remark 1 in Sec. 3.2 indicates that the proposed schedulability test (5) is *tight* when deceleration of the system is sufficiently large. A natural question arises: what if one could provide a bound $\alpha^* > 0$ for the absolute value of acceleration and deceleration (which is, of course more likely in practice)?

In this subsection, we will derive a *tighter* schedulability test when some restrictions hold for α^*, where $\alpha^* = \max_\omega\{\alpha^+(\omega), |\alpha^-(\omega)|\}$. The intuition behind the (to-be-shown) improvement is that in order to generate a job within an interval of duration $T(\omega^m)$ (for a given maximum utilization mode m), the engine has to accelerate throughout the interval, which causes it to end up in some different mode instead of the current one. Thus when considering demand based analysis starting at time t_0 when the engine initially start to rotate (right after the ignition), it is impossible for the system to reach or approach a demand density of $\sum_i c_i/T_i$ in any possible cases of the future.

The number of modes is normally quite small in engine control designs, and each mode corresponds to a relatively wide range of angular speed. Denote $\Delta^j_{\omega,i} = \omega^j_i - \omega^{j+1}_i$ as the angular speed difference of two neighboring modes of a given AVR task $\tilde{\tau}_i$; for example, $\Delta_\omega = [3000, 2000, 1200]^T$ for the sample task shown in Figures 1 and 1 (assuming $\omega_{\min} = 800$).

Given any initial angular speed ω^{j+1}_i and a maximum acceleration of α^*, we know that the least time T_{two} for the crankshaft to rotate two revolutions (4 cycles with a total angle of 8π) satisfies $(\omega^{j+1}_i + \omega^{j+1}_i + \alpha^* T_{two})T_{two}/2 = 8\pi$. Thus we simply have $T_{two} = (\sqrt{(\omega^{j+1}_i)^2 + 16\pi\alpha^*} - \omega^{j+1}_i)/\alpha^*$.

We would like to restrict the the highest absolute value (denoted as α^*) for acceleration and deceleration[4] such that *the system will always stay in the current mode or its neighboring modes within two revolutions*; i.e., $\omega^{j+1}_i + \alpha^* T_{two} \le \omega^j_i$ holds for all $i = 1, ..., n$, and $j = 1, ..., m-1$, which implies:

$$\forall i, j,\ \alpha^* \le \Delta^j_{\omega,i}(2\omega^j_i + \Delta^j_{\omega,i})/(16\pi), \tag{14}$$

Inequality (14) will serve as the additional constraint for the systems (considered in this subsection).

Table I shows the relationship of Δ_ω, ω and α^* in (14), by depicting some typical values. Although it is not safe to claim that Inequality (14) always holds for car engines, there are nevertheless many cases where the modes of engine-triggered tasks are widely separated from each other, or the maximum acceleration is small enough for the analysis in this subsection to be applicable.

Table 1: Relationship between acceleration upper bounds and mode angular speed ranges.

$\alpha^*(rpm/sec)$	$\Delta_\omega(rpm)$	$\omega(rpm)$
2604.1	500	1000
6250	1000	1000
10000	1408.3	1000
12500	1000	2500
10000	294.6	8000

THEOREM 3. *Schedulability Test (10) is sufficient when Condition (14) holds.*

Proof: The only difference between schedulability tests (10) and (5) is the utilization calculation for AVR tasks. It is thus sufficient to show that the demand for an engine-triggered task within period $[t_0, t)$ cannot exceed $\max_m\{c^m_i/\tilde{T}^m_i\}\cdot(t - t_0)$.

The intuition is that whenever there is a revolution with density greater than $\tilde{U}_i = \max_m\{c^m_i/\tilde{T}^m_i\}$, an accelerating mode change is unavoidable, which results in a second revolution with a relatively small enough demand density, such that average density always remains below U_i.

More formally, given any possible release pattern (according to engine rotation speed changes) during the period of interest $[t_0, t)$, Inequality (14) guarantees that accelerating mode-changes do not occur *successively*. As a result, the corresponding job generation series (except perhaps the last one) can be partitioned into *sub-periods* that each

1. contains only one job release without accelerating mode change, or
2. contains two successive job releases with an accelerating mode change before the second release;

[4]Deceleration is symmetric to acceleration in this analysis, with the same restrictions to its absolute value.

where each sub-period begins when some job releases, and ends right before another job releases.

Given a sub-period of Case 1) where a job is released with WCET of c^m at the beginning, we know that the sub-period length is at least $\tilde{T}_i^m$ (or else there must be a mode change during the period), and thus the demand density within such sub-period does not exceed $\tilde{U}_i = \max_m\{c_i^m/T_i^m\}$.

The total demand in sub-periods of Case 2) is $c^j + c^{j+1}$, where the system switches from Mode j to $j+1$ during the first revolution of the sub-period. The minimum length of such a sub-period is $T^m + (\omega^{j+1} - \sqrt{\omega^2 - 4\alpha^*})/\alpha^* \leq \tilde{T}^j + \tilde{T}^{j+1}$ (when the engine is rotating at an angular speed of $\omega(t) = \omega^j$, and accelerates at the highest possible rate thereafter). As a result, the demand density of such sub-period is no greater than $(c^j + c^{j+1})/(\tilde{T}^j + \tilde{T}^{j+1}) = \tilde{U}_i$.

Here the last revolution may be an exception during which an accelerating mode change may occur (without a "density-neutralizing" successor revolution). To deal with this possibly special revolution, the initial revolution (with angular speed starting at $\omega_{\min}$, and no mode change) may be combined with it to form a special revolution, and the length of the first period dominates the combined sub-period than any other Case 2) one (due to slower angular speed).

Summing up all sub-periods with demand densities of at most $\tilde{U}_i$ results in a total demand $\leq \tilde{U}_i(t-t_0)$ for any period of interest $[t_0, t)$. □

Remark 2. *Theorem 3 yields a* tight *schedulability test for current engine scheduler design and verification under the case that Inequality (14) holds. By tight, we mean that Test (10) is necessary and sufficient (i.e., speedup-*1*). Theorem 3 shows its sufficiency, while its necessity can be shown by reconsidering the possible release pattern in Figure 3(d) — for a long enough period, the engine may generate a dominating number of tasks in such a pattern with a total utilization of* $\max_m\{c_i^m/\tilde{T}_i^m\}$*, such that any violation to Condition (10) results in an unschedulable set.*

Remark 3. *Although it was previously mentioned that the acceleration may reach as high as* 13000 *rpm/sec, it is only when the engine is out-of-gear. Hence in cases of interest, acceleration during run time may be significantly smaller and Condition (14) may hold. Note that this condition plays an important role in the proof of Theorem 3 for Case 2).*

4. CONSTRAINED DEADLINES

Recall that a task system is assumed to include both "normal" sporadic tasks [18, 19] and engine triggered (AVR) ones. Traditional sporadic task systems with constrained deadlines have been well-studied, and a pseudo-polynomial time feasibility test has been derived [4] for systems with utilization capped (strictly) below 1.

Constrained deadlines frequently arise in engine triggered tasks as well. (For example, a task that reads system variables and calculates the amount of gas to be injected during each revolution may be released at the beginning of the intake cycle, and must finish before the compression cycle begins – i.e., with a period of 2 revolutions, and a deadline of 0.5 revolutions.) As a result, although we have derived (in Sec. 3) a necessary and sufficient schedulability test for implicit deadline systems, there is an emerging need to study the schedulability conditions of AVR task set with constrained deadlines. Some of our initial thoughts and observations concerning the scheduling of systems including such tasks will be reported in this section.

A naive pessimistic approach would simply check the total density (rather than utilization in Condition (10)) of the system; i.e.

$$\sum_{i|\tau_i} \frac{c_i}{D_i} + \sum_{i|\tilde{\tau}_i} \max_m\{\frac{c_i^m}{D_i(\omega_i^m)}\} \leq 1 \qquad (15)$$

However if Schedulability Test (15) fails for a given task set, in order to provide a more precise test instead of declaring failure, demand based analysis may be done via transformation to the *digraph-based model* [23]. Under such a model, release structures of jobs in terms of order and timing can be expressed via a directed graph. It is recognized as the most expressive task model for which pseudo-polynomial time EDF-schedulability testing algorithms are known.

We first use a simple (but typical) example to show how the digraph model may be used to represent and analyze the schedulability intuitively, and then describe the transformation formally.

Example 1. *Consider the sample AVR task shown in Figures 1 and 1. Figure 5 shows one of its possible digraph-based representations. (Note that we choose a simple transformation here to have each vertex representing an entire single mode, which is certainly not necessary — there are infinitely many different digraph representations of the same AVR task.) An execution of the task corresponds to a path*

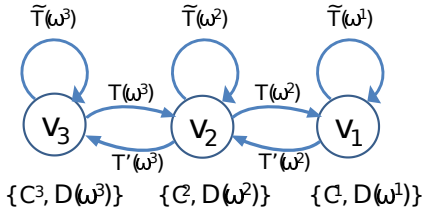


Figure 5: Expression of the sample AVR task under the digraph-based model.

in this digraph. In this example, each vertex represents a different engine mode. Each time a vertex is visited, a job with corresponding mode (with specified WCET and deadline parameters) will be released. The values labeling each edge represent the minimum time between releases of the corresponding jobs. Take the edge $v_2 \rightarrow v_1$ *as an example, when a job with mode* 2 *is released at an angular speed of* ω^2*, and the engine accelerates aggressively, the next job (under mode* 1*) may be released after a period of* $\tilde{T}(\omega^2)$*. We will formally show that such a task model transformation is* safe *— whenever the digraph-based model passes schedulability test, the original AVR task set is schedulable.*

In the cases where maximum acceleration and deceleration share the same absolute value, T' and T are the same. Otherwise $T'(\omega^m)$ can be calculated similar to (1), by considering the worst case that the processor decelerates at the highest value and reaches ω^m just within a revolution (i.e., replacing α^+ with $|\alpha^-|$).

For this example, when acceleration (or deceleration) is large enough, it is possible for the system to "jump" from Mode 1 into Mode 3 (or vice versa) within a revolution, which results in additional edges connecting non-neighborhood vertices (yet not shown in Figure 5). □

Formally speaking, the representation of an AVR task in the digraph model can be done by the following steps:

(i) Determine N_v, the number of adjacent ranges $\omega(j)$ for vertices one by one, $j = N_v, N_v - 1, ..., 2$.

(ii) Each vertex V_j represents all jobs releasing within an angular range of $[\omega(j+1), \omega(j))$, with execution of C^m, where $m = \min_i \omega^i \geq \omega(j)$, and a deadline of $D_i(\omega(j))$.

(iii) Add a self loop edge to each vertex with period of $\tilde{T}(\omega(j))$.

(iv) For each pair of vertices $\{V_i, V_j\}, i > j$, add an edge from V_i to V_j (or from V_j to V_i) with period of $T(\omega^i)$ (or $T'(\omega^i)$) if the acceleration (or deceleration) is large enough for such mode change to occur within a revolution.

REMARK 4. *Traditional tasks can easily be modeled as digraph ones, with a single node of WCET c_i, deadline D_i, and a self-pointing edge of period T_i.*

Discussion on choosing the number of vertices. Actually, each vertex is pessimistically representing a range of angular speeds. To make such a transformation precise, the range needs to be small enough which may lead to infinite number of vertices (each representing a different exact rotation speed). Adding more vertices helps in increasing the precision of the transformation (and further analysis) by reducing the pessimism, but makes the digraph more complicated, since both the number of vertices and the number of edges connecting non-neighboring vertices, become larger. One would prefer dealing with the case that *the system always stay in the current vertex or its neighborhood vertex within a revolution.* Under such a restriction, the constructed digraph will be a chain without edges connecting non-neighborhood vertices. Similar to Restriction (14), when the following inequality (which is much weaker and very likely to hold in practice) holds, we can always form such chain-shaped digraphs according to the steps described above:

$$\forall i, j,\ \alpha^* \leq \Delta^j_{\omega,i}(2\omega^j_i + \Delta^j_{\omega,i})/(8\pi). \tag{16}$$

Moreover, when the angular speed range of some mode is too large, we could certainly partition it and use more than one vertices to represent this mode.

5. EXPERIMENTAL ANALYSIS

In this section, we evaluate the effectiveness of our proposed method with existing schedulability tests for task systems comprised of a mix of traditional sporadic and AVR tasks. Comparisons are made with the following existing schedulability tests:

- RTA-SP: Use the maximum WCET, minimum period, and minimum deadline to represent all modes to model each AVR task as a traditional sporadic task.
- Exact-CON: Compute the maximum interference with additional constraints from the physical system via the exact analysis in [10] instead of Integer Linear Programming [13], which is the current state of the art analysis technique forAVR task systems under fixed priority scheduling.

Task Set Generation. To ensure a fair comparison with existing work, we adopt exactly the workload generation methodology described in [13]. Here, a task system is generated according to the following steps:

(i) Generate utilizations (U_i) and periods T_i with the UUniFast algorithm and log-uniform distribution, which together determine the WCET (C_i).

(ii) Constrained deadlines (D_i) are set equal to periods (T_i) for implicit systems, and are uniformly distributed in the range $[C_i + x(T_i - C_i), T_i]$ for constrained ones.

(iii) A fraction p of these tasks are then converted to AVR tasks, with pre-determined mode ranges (under scaling factor f). WCETs of modes are then adjusted under parameter e.

(iv) Apply some quick feasibility check and maintenance of acceleration bounds.

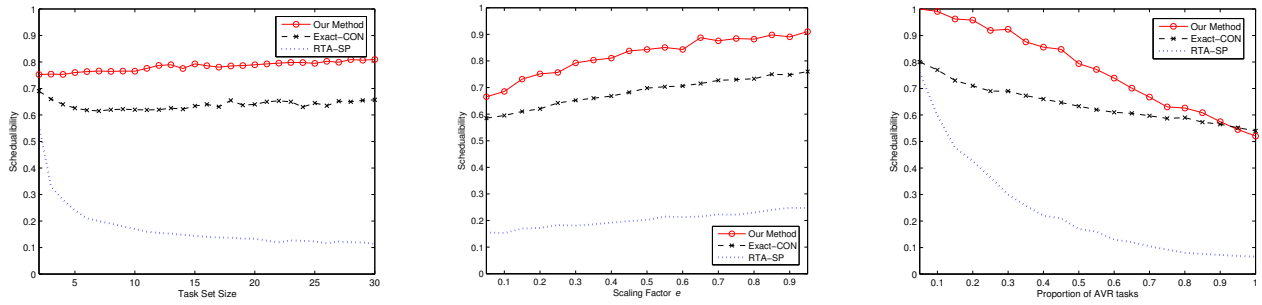
Detailed description and default parameter settings can be found in Sec. V-A of [13].

Comparison for Implicit Systems. Figure 6 shows the *success ratios* under different utilizations, which represents the proportion of sets that are deemed schedulable. 1000 randomly generated task sets are used for each utilization value.

As expected, we observe that our method outperforms existing ones, especially when the systems are heavily loaded (i.e., have large utilization). This is quite similar to the schedulability comparison between EDF and fixed priority scheduling for traditional sporadic task systems.

Comparison for Constrained Systems. Given a mixed task set τ, feasibility only needs to checked (based upon demand bound functions $dbf_T(\cdot)$) until an upper bound $D(\tau)$, which (according to Theorem V.4 in [23]) can be decided by:

$$D(\tau) = \frac{\sum_{i|\tau_i} c_i + \sum_{i|\tilde{\tau}_i} \sum_{v \in G(T)} c_i(v)}{1 - U(\tau)}; \tag{17}$$



(a) Different task size n (b) Different scaling factor e (c) Different AVR ratio f

Figure 7: Weighted schedulability comparison for constrained deadline systems.

where $G(T)$ is the corresponding digraph of task T, and $U(\tau)$ is given by:

$$U(\tau) = \sum_{i|\tau_i} U_i + \sum_{i|\tilde{\tau}_i} \max_{\pi}\{U(\pi)|\pi \text{ is a cycle in } G_i\}$$
$$= \sum_{i|\tau_i} U_i + \sum_{i|\tilde{\tau}_i} \tilde{U}_i. \quad (18)$$

In the case that each mode is assigned a single edge in the digraph, $\sum_{v\in G(T)} c_i(v)$ can be replaced by $\sum_{j=1}^{m} c_i^m$.

Demand bound functions $dbf_T(\cdot)$ can be calculated in pseudopolynomial time when all time periods are integers, which is unlikely to be true given the complicated definition of periods in fraction forms, e.g., (1) (9). One way to deal with this is to pessimistically round all values into an acceptable precise level (say $0.01ms$). Here by saying pessimistically, we should use ceiling values for WCETs, and floor values for periods, deadlines, and so on. The problem here is that we are losing preciseness of the analysis again while rounding the numbers. The other way is to explore most possible paths of each graph with a total period less than the given upper bound, while pruning some branches with the rules described in Sec. 5 of [10], and use them to derive all $dbf(\cdot)$ values (see Figure 3 of [23]). Although this approach has exponential time complexity, it is adopted for the simulations in this paper since task sets are small and the number of modes is fixed to a small constant.

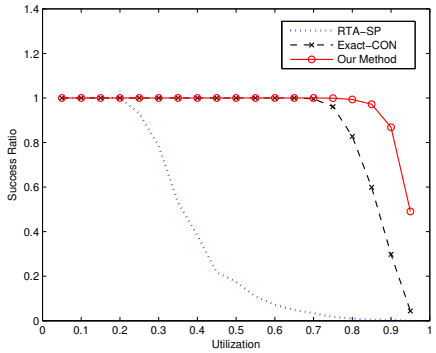


Figure 6: Schedulability ratio comparison for implicit systems.

Parameter Studies. We now compare the performance of the listed schedulability tests with respect to changes to one of the specified parameters via *weighted schedulability* [6] measurements. In these experiments, 100 (instead of 1000) random sets are used per utilization level that varies from 0.05 to 0.95 in steps of 0.05.

Figure 7(a) shows the weighted schedulability measure when size of the task set increases. In general we may conclude that schedulability ratio is unaffected by the cardinality of the set.

Figure 7(b) shows the weighted schedulability measure when the adjusting factor e is increased. This factor $e \in (0, 1)$ is used to shrink WCETs of modes with utilization not equal to the maximum utilization. A smaller e indicates that other modes are having more similar utilizations to the max-utilized mode. As a result, as e increases, the system becomes less affected by other modes, and since the proposed analysis to a single mode is precise, schedulability ratio is increasing.

Finally, Figure 7(c) shows how the proportion of tasks that are AVR ones affects the weighted schedulability. There is a slight decrease of (weighted) schedulability ratio when more tasks within a set are AVR ones. This is as expected since we have not yet obtained schedulability analysis methods for AVR tasks that are as accurate as the ones for traditional sporadic tasks.

6. CONCLUSIONS

This paper seeks to develop efficient schedulability analysis for a novel model of recurrent real-time processes called the AVR task model [12], that was defined through the consideration of physical constraints in a cyber-physical system – an automobile engine. Most previous work on the AVR model had focused attention to fixed-priority scheduling; few results have been shown in earliest deadline first scheduling. However, as the deadline of an AVR task changes, then so should its priority, otherwise the system will necessarily suffer from priority inversion and a resulting under-utilization

of platform computing resources. This issue could be addressed via the use of EDF scheduling, which is the focus in this paper.

A sufficient and fast schedulability test is shown for implicit systems, and its speedup factor (as a function of engine rotation speed) is derived. Under some practical assumptions, this result is further improved to be necessary and sufficient. For constrained systems (with relative deadlines smaller than periods), an attempt for demand based function analysis has been made by transforming into the digraph based task model. Schedulability experiments reported in Sec. 5 confirm that the proposed methods outperform the current state of the art from the perspective of schedulability ratio. Overall performance is further compared in these schedulability experiments with respect to changes in specific parameters, one at a time.

7. REFERENCES

[1] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[2] S. Baruah and Z. Guo. Mixed-criticality scheduling upon varying-speed processors. In *Proceedings of the 34th IEEE Real-Time Systems Symposium (RTSS'13)*, 2013.

[3] S. Baruah and Z. Guo. Scheduling mixed-criticality implicit-deadline sporadic task systems upon a varying-speed processor. In *Proceedings of the 35th IEEE Real-Time Systems Symposium (RTSS'14)*, 2014.

[4] S. Baruah, R. Howell, and L. Rosier. Feasibility problems for recurring tasks on one processor. *Theoretical Computer Science*, 118(1):3–20, 1993.

[5] S. K. Baruah, D. Chen, S. Gorinsky, and A. K. Mok. Generalized multiframe tasks. *Real-Time Systems*, 17(1):5–22, 1999.

[6] A. Bastoni, B. Brandenburg, and J. Anderson. Cache-related preemption and migration delays: Empirical approximation and impact on schedulability. In *Proceedings of the 6th International Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT'10)*, 2010.

[7] J. Bengtsson, K. G. Larsen, F. Larsson, P. Pettersson, and W. Yi. Uppaal — a Tool Suite for Automatic Verification of Real–Time Systems. In *Proceedings of the Workshop on Verification and Control of Hybrid Systems III*, pages 232–243, 1995.

[8] A. Benveniste and G. Berry. The synchronous approach to reactive and real-time systems. *Proceedings of the IEEE*, 79(9):1270–1282, 1991.

[9] A. Benveniste, P. Caspi, S.A. Edwards, N. Halbwachs, P. Le Guernic, and R. de Simone. The synchronous languages 12 years later. *Proceedings of the IEEE*, 91(1):64–83, 2003.

[10] A. Biondi, A. Melani, M. Marinoni, M. Di Natale, and G. Buttazzo. Exact analysis of adaptive variable-rate tasks under fixed-priority scheduling. In *Proceedings of the 25th Euromicro Conference on Real-Time Systems (ECRTS'14)*, 2014.

[11] G. Buttazzo, E. Bini, and D. Buttle. Rate-adaptive tasks: Model, analysis, and design issues. In *Proceedings of the International Conference on Design, Automation and Test in Europe (DATE'14)*, 2014.

[12] D. Buttle. Real-time in the prime-time. In *Keynote speech given at the 24th Euromicro Conference on Real-Time Systems (ECRTS'12)*, 2012.

[13] R. I. Davis, T. Feld, V. Pollex, and F. Slomka. Schedulability tests for tasks with variable rate-dependent behaviour under fixed priority scheduling. In *Proceedings of the 20th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'14)*, 2014.

[14] N. Halbwachs. *Synchronous programming of reactive systems*. Kluwer Academic Publisherss, 1993.

[15] K. Jeffay and D. Bennett. A rate-based execution abstraction for multimedia computing. In *Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, 1995.

[16] K. Jeffay and S. Goddard. A theory of rate-based execution. In *Proceedings of the 20th IEEE Real-Time Systems Symposium (RTSS'99)*, pages 304–314, 1999.

[17] J. Kim, K. Lakshmanan, and R. Rajkumar. Rhythmic tasks: A new task model with continually varying periods for cyber-physical systems. In *Proceedings of the 3rd IEEE/ACM International Conference on Cyber-Physical Systems (ICCPS'12)*, pages 28–38, 2012.

[18] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[19] A. K. Mok. *Fundamental Design Problems of Distributed Systems for The Hard-Real-Time Environment*. PhD thesis, Massachusetts Institute of Technology, 1983.

[20] A. K. Mok and D. Chen. A multiframe model for real-time tasks. *IEEE Transactions on Software Engineering*, 23(10):635–645, 1997.

[21] V. Pollex, T. Feld, F. Slomka, U. Margull, R. Mader, and G. Wirrer. Sufficient real-time analysis for an engine control unit. In *Proceedings of the 21st International Conference on Real-Time and Networked Systems (RTNS'13)*, pages 247–254, 2013.

[22] V. Pollex, T. Feld, F. Slomka, U. Margull, R. Mader, and G. Wirrer. Sufficient real-time analysis for an engine control unit with constant angular velocities. In *Proceedings of the International Conference on Design, Automation and Test in Europe (DATE'13)*, pages 1335–1338, 2013.

[23] M. Stigge, P. Ekberg, N. Guan, and W. Yi. The digraph real-time task model. In *Proceedings of the 17th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'11)*, pages 71–80, 2011.

[24] M. Stigge, P. Ekberg, N. Guan, and W. Yi. On the tractability of digraph-based task models. In *Proceedings of the 22th Euromicro Conference on Real-Time Systems (ECRTS'11)*, pages 162–171, 2011.

[25] D. Vivien. Lexus LFA instruments. *Evo magazine*, page 202, 2013.