Rhythmic Tasks: A New Task Model with Continually Varying Periods for Cyber-Physical Systems

Junsung Kim, Karthik Lakshmanan, Ragunathan (Raj) Rajkumar
Electrical and Computer Engineering Department
Carnegie Mellon University, Pittsburgh, PA 15213
{junsungk, klakshma, raj}@ece.cmu.edu

***Abstract*—Traditional mechanical subsystems in automobiles are being replaced by electronically controlled systems, often with no mechanical backup. This trend towards "drive-by-wire" systems is becoming increasingly popular. In these cyber-physical systems, a critical task not meeting its timing deadline can lead to a safety violation and damage to life and/or property. Classical real-time scheduling techniques such as RMS and EDF can be used to guarantee the schedulability of periodic tasks. However, certain critical tasks like the engine control task are activated by engine events such as pulses generated by sensors at the engine crankshaft. The periods of these engine tasks vary continually and even dramatically depending on the engine speed. The conventional periodic task model is inadequate for handling such tasks in cyber-physical systems due to its pessimism when combined with common schedulability analyses. In this paper, we define a new task model called *Rhythmic Tasks* for tasks having periods that vary due to external physical events. To the best of our knowledge, this is the first model that considers continually varying periods for fixed-priority scheduling in dynamic operating environments. We formally define the rhythmic task model and study its scheduling properties. In the context of rhythmic engine control tasks, we offer schedulability tests for determining the maximum possible utilization under the steady state, which is related to the physical engine speed. We also investigate the range of possible engine acceleration and deceleration rates. We show that excessive acceleration and deceleration can make the system unschedulable. We provide algorithms to find the appropriate ranges for acceleration and deceleration rates. We use a specific case study of engine control to illustrate our analysis.**

***Keywords*-Real-time; Rhythmic-Task; Engine; Cyber-Physical; Scheduling; Embedded-System;**

I. Introduction

Cyber-Physical Systems (CPS) embed computing and communication capabilities in all types of physical objects. Embedded and real-time systems are now essential to control the physical environment, to monitor the timing of dynamic processes taking place in it, to efficiently coordinate CPS operations and most importantly to ensure safety. This trend will only continue and in fact is expected to accelerate [1].

Among many applications of CPS such as aerospace systems, building and industrial infrastructure control, medical devices, robotic systems and transportation vehicles, automotive sub-systems such as engine control and chassis control must operate in real-time. The embedded *control* system in a car is also safety-critical and requires a high level of confidence in system correctness. In such systems, a critical task not meeting its timing deadline can lead to system failure.

Specifically, the engine and transmission in a car together form the car's powertrain, which is controlled by Powertrain Control Module (PCM) software using closed-loop control. At *periodic* intervals, software calculates the engine speed and position, determines the next time to fire a spark signal, and based on speed-change commands from the driver, adjusts settings for fuel flow. The software then senses the exhaust system to determine the effectiveness of the combustion process as depicted in Figure 1. As the engine runs faster, the fuel intake cycle gets shorter, and the frequency of calculating the injected fuel volume goes up. Incorrect fuel volumes or mistimed fuel injection can even damage the engine. Therefore, control algorithms of engine events require significant signal conditioning, and place stringent response-time requirements. In order to meet these requirements, a real-time operating system such as OSEK [3] and AUTOSAR [4] is used.

Although real-time operating systems are widely used in cars and PCM applications contain many periodic tasks, the engine events activating the fuel injection task come from reference pulses generated by sensors at the engine crankshaft. That is, the periods of these tasks *vary* depending on the speed of the crankshaft. As an analog variable, speed is continuous and hence the period of the task also can change both rapidly and continuously. Also, the execution time of these tasks vary and the worst-case execution time (WCET) arises when the engine speed increases to its maximum [5]. It is known in the automotive community that the engine control performance deteriorates with under-sampling, i.e., tasks having a longer period than the required minimum period for a given speed. In particular, the controller task period is known to have a greater impact on control performance than execution time.

In this paper, we will focus on Rate-Monotonic Scheduling (RMS) [6], the optimal fixed-priority preemptive scheduling policy, which automotive OS standards such as

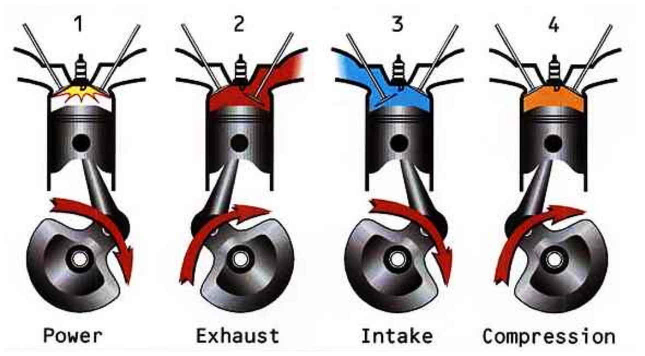


Figure 1. Four-stroke cycle in gasoline engines [2]

OSEK and AUTOSAR support and other general-purpose OSes like Linux also do. Under RMS, the shorter the period of a task, the higher is its priority. A common assumption of using RMS is that the task periods do *not* change during run-time. A *Utilization Bound (UB) test* is often used to check if the given tasks are *schedulable*, which means that each of the given tasks meets its timing deadlines.

Conventional task models such as periodic tasks or aperiodic tasks are not adequate to handle engine tasks with varying periods. Consider a periodic task with $60ms$ execution time and a $140ms$ period, along with an engine task that has a period varying from $10ms$ to $120ms$, which is depicted in Figure 2. The UB test [6] gives $4ms$ as the maximum computation time of the engine task to guarantee the schedulability of the given two tasks, which is 46% utilization in the worst case. However, the engine task at lower speeds can have up to $60ms$ as its computation time, yielding 93% utilization. This 47% difference comes from the worst-case assumption for using a single offline UB test, where the shortest period and no period change are taken into account. In case we have more than one periodic task, this analysis could require the addition of more hardware, which is undesirable in a mass production industry such as car manufacturing.

Contributions: In this paper, we define a new task model called *Rhythmic Tasks* for characterizing and analyzing tasks that have continually varying periods depending on external physical events. We provide response-time analyses for rhythmic tasks under three cases: constant engine speed, accelerating engine speed and decelerating engine speed. We provide guidelines to evaluate schedulable utilization levels for the rhythmic task model by introducing *harmonic points* and *flexion points*. An integrated rhythmic task analysis framework with periodic tasks is also provided. We finally provide a case study of the rhythmic task model for PCM to show the applicability of the rhythmic task model to a CPS. To the best of our knowledge, this is the first model considering both continually varying periods and WCET for cyber-physical systems.

Organization: The rest of this paper is organized as follows. We summarize prior research related to our work in Section II. In Section III, we define the rhythmic task model. In Section IV, we provide an analysis of one rhythmic task and one periodic task. In Section V, we provide an integrated analysis framework for one rhythmic task with multiple periodic tasks. In Section VI, a case study on an automotive PCM is presented. Finally, in Section VII, we provide our concluding remarks and discuss future work.

II. Related Work

The periodic task model [6] has been extensively studied in the real-time systems literature. Extensions to this model such as constrained-deadline sporadic tasks [7] and arbitrary deadline tasks [8] have been explored in the past. Although these task models represent tasks having different relationships between their periods and deadlines, the task parameters themselves are static and/or worst-case in nature. In this work, we explore a model where certain tasks have dynamically changing parameters, which are determined by external or *physical* system attributes such as the engine speed in a PCM.

The importance of task periods on the quality of engine control has been demonstrated in [5]. As the engine speed varies, the system must continuously change the engine control task periods. Given vehicle dynamics [9], maintaining a close relationship between the control task and the engine speed is key for achieving high efficiency. Worst-case execution time analysis of engine tasks was carried out in [10]. At higher speeds, system designers tend to adaptively reduce the task computation times to counteract the shrinking task periods, and try to maintain approximately constant system utilization. In this work, we develop the rhythmic task model in detail to represent such types of engine control tasks and study the resulting properties.

Some task models with dynamically changing parameters have been studied in the past. For instance, the elastic task model [11] treats tasks as springs with given elastic coefficients. More recently, the gravitational task model [12] was introduced by representing tasks as bobs hanging on a pendulum with the objective of preferably executing at a target set point. Although these task models have dynamically changing parameters, their usage is often motivated by the need to provide *quality of service* or to maximize *system utility*. Also, due to the fact that the elastic task model uses dynamic-priority scheduling and the gravitational task model is based on non-preemptive jobs, the previous work is not appropriate for fixed-priority preemptive scheduling. In this work, we consider a model where the changes in task parameters are resulting from the physical nature of the system, and changes in the operating environment drive task requirements.

From a schedulability analysis perspective, the analyses of minimum task periods and maximum worst-case execution time are well-known results for the periodic task model [13, 14]. The acyclic task model [15] uses a task model where a task comprises successive invocations but with no constraints between the periods of successive invocations. The utilization bound for acyclic tasks was also derived. Our rhythmic task model is more restricted, is motivated by cyber-physical requirements and should yield better utilization. We provide some bounds and guidelines to find schedulable regions for the generic rhythmic task model. These results are also helpful to understand the utilization bounds when only the task periods are given. We study the properties of acceleration and deceleration, which correspond to the maximum rate at which task periods can be decreased and increased respectively. In this regard, the closest work to ours is that of the mode change protocol [16]. However, we are interested in understanding the effect of a series of *continuous* mode changes on the schedulability of lower-priority tasks, as opposed to one single independent system-level mode change.

Tasks with relationships between task periods and phys-

Table I
NOTATION FOR THE RHYTHMIC TASK MODEL

τ^*	Rhythmic task
v_s	State vector that represents physical environmental attributes
$C(v_s)$	Varying worst-case execution time of τ^*
$T(v_s)$	Continually varying period of τ^*
$D(v_s)$	Relative deadline of τ^*
$u(v_s)$	Utilization of τ^*
α	Acceleration rate of τ^*
n_α	Maximum acceleration duration of τ^*
n_β	Maximum deceleration duration of τ^*

ical attributes can be also found in other cyber-physical subsystems besides the engine control task. For example, in the context of autonomous driving [17], the sensor processing tasks need to execute at a higher rate when the vehicle is moving at a higher speed, since the vehicle would cover a longer distance in a shorter time. Another good example is building energy management [18] where fine-grained management depending on varying environmental parameters will save more energy. Also, most CPS with control algorithms can likely obtain benefits from the rhythmic task model because the quality of control is affected significantly by sampling rates.

III. The Rhythmic Task Model

A. Definitions

A *rhythmic task* is a task with a (potentially) continually varying period and varying WCET. The change in the period value of a rhythmic task can depend on the current physical attributes of the system. The physical attributes of the given system are represented by a state vector, $v_s \in \mathbb{R}^k$, where k is the number of dimensions that capture the current system status. The WCET, period and deadline of a rhythmic task are a function of v_s and are denoted as $C(v_s)$, $T(v_s)$ and $D(v_s)$ respectively. Hence, the utilization of a rhythmic task is also a function of v_s and it is represented as $U(v_s)$.

Let J_i denote the i^{th} job of the rhythmic task and $T(v_s, J_i)$ denote the period of J_i. We define the acceleration $\alpha(v_s)$ of the rhythmic task as $1-\frac{T(v_s,J_{i+1})}{T(v_s,J_i)}$. If $T(v_s, J_{i+1}) < T(v_s, J_i)$, acceleration is positive and the engine speed is increasing. The duration of acceleration is limited by $n_\alpha(v_s)$ in terms of the number of job releases. In other words, the rhythmic task can be positively accelerated by a factor of $\alpha(v_s)$ for $n_\alpha(v_s)$ job releases. When $T(v_s, J_{i+1}) > T(v_s, J_i)$, $\alpha(v_s)$ becomes negative and represent the deceleration of the rhythmic task. To avoid ambiguity, we use $n_\beta(v_s)$ when $\alpha(v_s)$ is negative. For ease of readability, we denote $C(v_s), T(v_s), D(v_s), U(v_s), \alpha(v_s), n_\alpha(v_s)$ and $n_\beta(v_s)$ as $C^*, T^*, D^*, u^*, \alpha, n_\alpha$, and n_β respectively. A summary of the notation is given in Table I.

B. System Assumptions

We consider a set of hard real-time tasks $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$, where n is the number of tasks. The tasks in Γ are classified into two subsets: *Periodic Task Set*, Γ^P, and *Rhythmic Task Set*, Γ^R. We assume that Γ^R consists of m tasks ($m \leq n$). In other words, $\Gamma = \Gamma^P \cup \Gamma^R$ and $\Gamma^P \cap \Gamma^R = \emptyset$. For the sake of convenience, if a task τ_i is in Γ^R, the task may be denoted as τ_i^*. If a task τ_i is in Γ^P, the task will be represented as τ_i without the asterisk symbol (*).

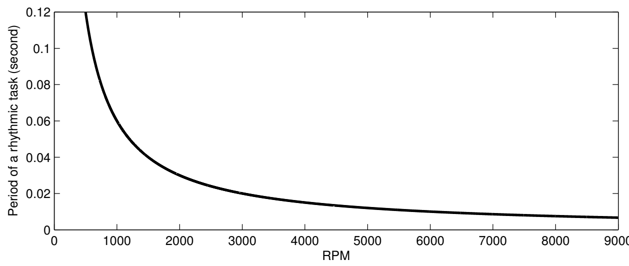


Figure 2. The variation of period according to engine RPM

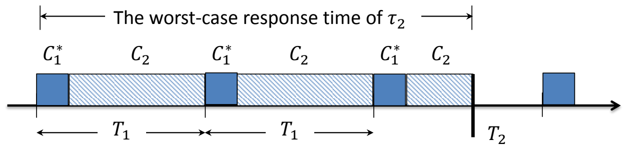
A periodic task τ_i is specified as (C_i, T_i, D_i), where C_i is its WCET, T_i is its period, and D_i is the deadline of each of the task's jobs relative to the release time of each job. A rhythmic task τ_i^* is denoted by (C_i^*, T_i^*, D_i^*), where the WCET, period, and deadline are functions of v_s. In addition, u_i is the utilization of τ_i, defined as $\frac{C_i}{T_i}$. In this paper, we assume that $T_i = D_i$ and $T_i^* = D_i^*$.

A rhythmic task τ_i^* is classified into three categories according to how C_i^* varies. A rhythmic task in the first category has a constant value of C_i^*. We refer to a rhythmic task with constant C_i^* as a *Constant Computation Rhythmic Task* (CCRT). In the second category, the utilization of a rhythmic task is maintained constant, and C_i^* varies accordingly. We name a rhythmic task with a constant utilization u_i^* as a *Constant Utilization Rhythmic Task* (CURT). In the third category, the WCET C_i^* of a rhythmic task is defined by a function $C_i^* = f_i(v_s)$, where f represents a general behavior of rhythmic tasks. One example of rhythmic tasks in this category is a task having a step function for C_i^* to maintain approximately constant utilization with discrete steps. We refer to a task in this category as a *General Computation-time Rhythmic Task* (GCRT).

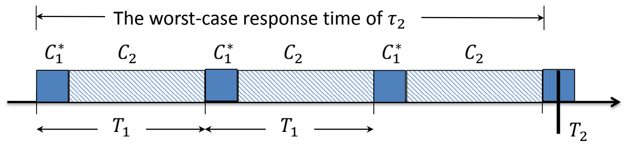
We assume the use of fixed-priority scheduling, specifically RMS on a uniprocessor. Therefore, for RMS, the priority of a rhythmic task τ^* at time *t* will be determined based on its period T^* depending on the instantaneous system state v_s at time *t*. In this paper, we assume that $m = 1$ and that the rhythmic task has the highest priority among all the n tasks. Therefore, T_1^* and C_1^* represent the period and the WCET of the rhythmic task τ_1^*. Hence, the inequality, $\forall v_s, C_1(v_s) \leq T_1(v_s) \leq T_2$, also holds. In other words, the only rhythmic task in the system always has the shortest period and, by RMS, is assigned the highest priority of all tasks. This paper defines the rhythmic task model and studies the single rhythmic task scenario with many periodic tasks. This work in itself is useful for general CPS applications. However, there could be a need for multiple rhythmic tasks, and analyzing such systems is key future work.

C. Application of Rhythmic-Task Definition to Engine Control

The engine shown in Figure 1 has two revolutions every engine cycle, and the speed of revolutions is affected by four primary parameters ($k = 4$): RPM, number of active cylinders, the amount of fuel injected into cylinders, and gear ratio. Hence, $v_s :=$ <RPM, Number of active cylinders, Fuel



(a) Case I: The worst-case response time of τ_2 is less than or equal to its deadline.



(b) Case II: The $\left\lceil \frac{T_2}{T_1^*} \right\rceil$-th instance of τ_1^* is executed at T_2.

Figure 3. Two different cases to consider to prove *Lemma* 1

amount, Gear ratio>. The period of the rhythmic task driven by the engine cycle is directly related to the duration of each revolution. If the rhythmic task is triggered every revolution, its period varies as illustrated in Figure 2. As can be seen, the period of the rhythmic task can vary over a wide range. Using the parameters of v_s, the acceleration rate α and the maximum acceleration duration n_α can be determined. Most modern cars are equipped with a *rev limiter* to prevent engines from being *redlined*. We can treat this redline as the maximum RPM for calculating α. By accelerating the engine at a particular gear level till the engine hits the redline from the minimum engine RPM, we can measure the acceleration duration at that gear level. Using the acceleration duration and the RPM changes, the acceleration rate α is determined. The measured duration is converted to n_α. Similarly, the maximum deceleration duration n_β also can be determined.

D. Problem Formulation

Our objective is to determine whether a given taskset with a rhythmic task τ_1^* running at the highest priority is schedulable under (a) steady-state conditions (b) positive acceleration, and (c) deceleration. We will provide a schedulability test for each of these cases. For steady-state conditions, we will propose an algorithm to determine schedulability given the current (C_1^*, T_1^*) values along with the periodic tasks. For accelerating and decelerating conditions, we will provide a range of period change ratios for which schedulability holds. These outcomes can be used by CPS developers to determine when C_1^* needs to be decreased in order to maintain schedulability.

IV. One Rhythmic Task and One Periodic Task

We first consider a simple taskset Γ with one rhythmic task τ_1^* and one periodic task τ_2.

A. Steady-State Analysis

Lemma 1. *Given one rhythmic task, represented by (C_1^*, T_1^*), and one periodic task, represented by (C_2, T_2), both tasks are schedulable if the following inequality is satisfied.*

$$C_1^* \leq max\left(\frac{T_2 - C_2}{\left\lceil \frac{T_2}{T_1^*} \right\rceil}, T_1^* - \frac{C_2}{\left\lfloor \frac{T_2}{T_1^*} \right\rfloor}\right) \tag{1}$$

Proof: In order to prove this statement, we should find the maximum value of C_1^*, which does not cause τ_2 to miss its deadline. By assumption (see Section III-B), τ_1^* has higher priority than τ_2. Hence, the two tasks are schedulable if τ_2 is schedulable. In order to obtain the maximum schedulable value of C_1^*, we should consider two different cases. The first case is when the response time of τ_2 is less than or equal to its relative deadline, which is illustrated in Figure 3a. In this case, $C_1^* \left\lceil \frac{T_2}{T_1^*} \right\rceil + C_2 \leq T_2$ should be satisfied. Then, in this case, the maximum value of C_1^* is given by

$$C_1^* = \frac{T_2 - C_2}{\left\lceil \frac{T_2}{T_1^*} \right\rceil} \tag{2}$$

The second case is depicted in Figure 3b. Since τ_1^* can preempt τ_2, the $\left\lceil \frac{T_2}{T_1^*} \right\rceil$-th instance of τ_1^* can overlap the period of task τ_2. Therefore, $C_1^* \left\lfloor \frac{T_2}{T_1^*} \right\rfloor + C_2 \leq T_1^* \left\lfloor \frac{T_2}{T_1^*} \right\rfloor$ should hold. Hence, in this case, the maximum value of C_1^* is given by

$$C_1^* = T_1^* - \frac{C_2}{\left\lfloor \frac{T_2}{T_1^*} \right\rfloor} \tag{3}$$

The maximum value of Equation (2) and Equation (3) will provide the bound of C_1^*, given τ_2. Therefore, if Inequality (1) is satisfied, both tasks are schedulable. □

Inequality (1) allows us to visualize the schedulability of one rhythmic task and one periodic task. Given τ_2: $(6, 14)$, a well-known worst-case task for the least-upper bound on schedulable utilization as an example [6], Figure 4 plots the maximum value of C_1^* as T_1^* varies from 0 to 14. If (C_1^*, T_1^*) lies under the curve in this figure, the taskset with τ_2: $(6, 14)$ is schedulable.

Accordingly, we can see the utilization change of the taskset. Figure 5 shows the variation in the total utilization as T_1^* changes. In this figure, we observe two types of interesting points: local maxima and local minima. We call local maxima as *harmonic points*, since the task periods are "compatible" at these points, and local minima as *flexion points*, since the slope changes from negative to positive here. Let $U(\Gamma)$ denote the total utilization and $U_{lub}(\Gamma)$ denote the least-upper bound on schedulable utilization of the given taskset Γ having one rhythmic task and one periodic task.

Lemma 2. *At $T_1^* = \frac{T_2}{i}$, where $i \in \mathbb{Z}^+$, harmonic points occur, where $U(\Gamma)$ is 1.*

Proof: Substituting $\frac{T_2}{i}$ for T_1^* in the right-hand side of Equation (2) returns $\frac{T_2 - C_2}{i}$ since $\frac{T_2}{T_1^*}$ becomes an integer i. We also obtain the same value from the right-hand side of Equation (3). Then, Inequality (1) is equivalent to $C_1^* \leq \frac{T_2 - C_2}{i}$. The utilization of two tasks is given by $U(\Gamma) = \frac{C_1^*}{T_1^*} + \frac{C_2}{T_2}$. By substituting $\frac{T_2 - C_2}{i}$ and $\frac{T_2}{i}$ for C_1^* and T_1^* respectively, we obtain 1 as the total utilization. □

Lemma 3. *Flexion points, local minima of $U(\Gamma)$, happen at $T_1^* = \frac{C_2}{i(i-1)} + \frac{T_2}{i}$, where $i \geq 2$ and $i \in \mathbb{Z}^+$. $C_1^* = \frac{T_2 - C_2}{i}$*

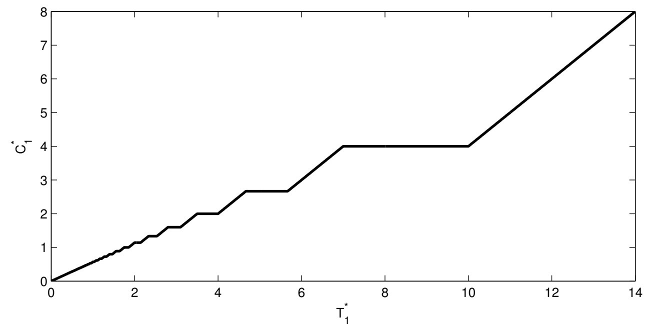


Figure 4. Schedulable region of the taskset including τ_1^* and τ_2 as (6,14)

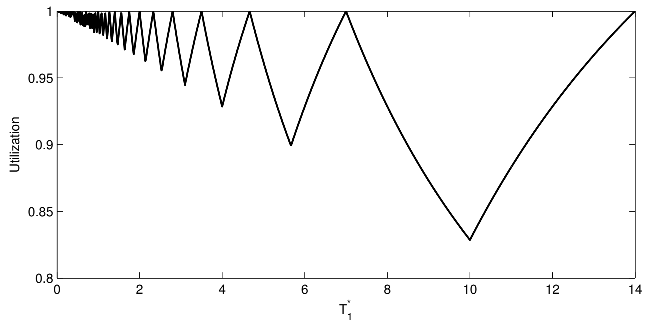


Figure 5. Corresponding utilization value where there are one rhythmic task and one periodic task, (6,14)

also holds.

Proof: From Lemma 2, the harmonic points happen when $T_1^* = \frac{T_2}{i}$, where $i \in \mathbb{Z}^+$. Let's assume that the flexion points occur when $T_1^* = \frac{T_2}{i} + x$, where x is the value we want to find. The flexion points correspond to the intersections of Equation (2) and Equation (3) as shown in [6]. We can substitute $\frac{T_2}{i} + x$ for T_1^* in both equations. Suppose $i \geq 2$. Then, because $\frac{T_2}{i} \leq \frac{T_2}{i} + x \leq \frac{T_2}{i-1}$, $\left\lceil \frac{T_2}{T_1^*} \right\rceil = \left\lceil \frac{T_2}{\frac{T_2}{i}+x} \right\rceil = i-1$ and $\left\lfloor \frac{T_2}{T_1^*} \right\rfloor = \left\lfloor \frac{T_2}{\frac{T_2}{i}+x} \right\rfloor = i$. We substitute these in both equations, and we obtain $C_1^* = \frac{T_2 - C_2}{i-1} = x + \frac{T_2}{i} - \frac{C_2}{i}$. Solving for x, we get $x = \frac{T_2 - C_2}{(i-1)i}$. When $i \geq 2$ and $i \in \mathbb{Z}^+$, $T_1^* = \frac{T_2}{i} + x = \frac{T_2}{i} + \frac{T_2 - C_2}{(i-1)i} = \frac{C_2}{i(i-1)} + \frac{T_2}{i}$ □

Lemma 4. *The minimum flexion point, U_{lub}, occurs at $i = 2$.*

Proof: Substituting the results from *Lemma* 3 gives us $U(\Gamma) = \frac{T_2 - C_2}{\frac{C_2}{i-1} + T_2} + \frac{C_2}{T_2}$. Because $T_2 - C_2 \geq 0$, U_{lub} can be found when i has the smallest allowable value, which is 2. □

We can see that the results from Lemma 3 and Lemma 4 are consistent with the curve shown in Figure 5. One interesting observation of Lemma 3 is that only the parameters of the periodic task affect the locations of the flexion points. A similar property will be shown in Theorem 3.

Lemma 5. *The flexion points of one rhythmic task and one periodic task occur at $C_2 = T_2\left(\sqrt{i(i-1)} - (i-1)\right)$, where $i \in \mathbb{Z}^+$ and $i \geq 2$.*

Proof: From the proof of Lemma 4, $U(\Gamma) = \frac{T_2 - C_2}{\frac{C_2}{i-1} + T_2} + \frac{C_2}{T_2}$. Differentiating with respect to C_2, we obtain $\frac{\partial U(\Gamma)}{\partial C_2} = \frac{1}{T_2} + \frac{i(i-1)T_2}{((i-1)T_2 + C_2)^2}$. We solve the equation $\frac{\partial U(\Gamma)}{\partial C_2} = 0$, and the solution is given by $C_2 = T_2\left(\sqrt{i(i-1)} - (i-1)\right)$, where the flexion points happen. □

B. Acceleration Analysis

The positive acceleration of an engine is a significant event from a schedulability perspective. Car manufacturers always provide two types of information on engine specifications, horse power and torque. The horse power of an engine is related to the maximum speed analogous to the steady-state discussed in Section IV-A, and the torque of an engine has a strong relationship with the acceleration of a vehicle. The acceleration of a car is immediately followed by the change of task periods controlling the engine. As shown in [5], the quality of engine control decreases significantly if the periods of engine tasks decrease. In this subsection, we will discuss how much the engine can accelerate by finding the maximum rate of period changes under the given taskset Γ.

As shown in Figure 2, the duration for one revolution becomes shorter as positive acceleration occurs. Suppose that the i^{th} revolution occurs at time t. Then, as the engine accelerates, the $(i+1)^{th}$ revolution will have a shorter period. Let α denote the rate of period change, where $0 \leq \alpha \leq 1$ and $\alpha \in \mathbb{R}$. Let n_α denote the maximum positive acceleration duration in terms of the number of job releases of the rhythmic task. Suppose that $T^{*,i}$ is the period of the i^{th} revolution of the rhythmic task τ^*. Then, if the period of the rhythmic task reduces after the first job release, we can express the period of the $(i+1)^{th}$ revolution as $T^{*,i+1} = T^{*,i}(1-\alpha)$ by using α, when $i \leq n_\alpha$. Otherwise, $T^{*,i+1} = T^{*,i}$. Hence, we can define $T^{*,i}$ as $T^{*,i} = T_1^*(1-\alpha)^{\min(i,n_\alpha)}$ for a non-negative integer i. We will find if the given taskset is schedulable under the given α and n_α.

Suppose that there are two tasks, one rhythmic task τ_1^* and one periodic task τ_2. Under the given (C_1^*, T_1^*) at a certain time, let n_p^a denote the number of preemptions which τ_2 experiences while T_1^* is decreasing. Then, if T_1^* starts decreasing at the first job release, n_p^a is defined as $n_p^a = \max\{n | \sum_{i=0}^{n-1} T_1^{*,i} \leq T_2 \text{ and } n \in \mathbb{Z}^+\}$.

Given the definition of n_p^a, the following inequality should be satisfied.

$$\sum_{i=0}^{n_p^a - 1} T_1^{*,i} \leq T_2 \tag{4}$$

Let f_C^* denote the function of $T_1^{*,i}$ which returns the computation time of the rhythmic task, where f_C^* has a different type of function depending on which category among CCRT, CURT and GCRT the rhythmic task is classified into. Then, n_p^a and α should meet one of the following two inequalities:

$$\sum_{i=0}^{n_p^a - 1} \left\{ f_C^*\left(T_1^{*,i}\right) \right\} + C_2 \leq T_2 \tag{5}$$

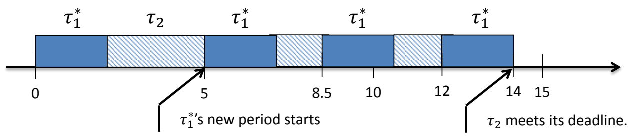


Figure 6. The example scenario when a rhythmic task accelerates.

$$\sum_{i=0}^{n_p^a-2} \left\{ f_C^* \left(T_1^{*,i} \right) \right\} + C_2 \leq \sum_{i=0}^{n_p^a-2} T_1^{*,i} \tag{6}$$

where Inequality (5) refers to the case when the n_p^a-th instance of τ_1^* completes before T_2 under the assumption that the acceleration is started at the first instance. Inequality (6) represents the case when the n_p^a-th instance of τ_1^* overlaps T_2. These two cases are also explained in the proof of Lemma 1. In order to decide if α is possible, it should be substituted in Inequality (4). According to the α value, we determine n_p^a, which should be substituted in Inequality (5) and Inequality (6) to check if one of those inequalities is satisfied.

As an example of a type of f_C^*, we consider CCRT. As C_1^* does not change, Inequalities (5) and (6) become $n_p^a C_1^* + C_2 \leq T_2$ and $(n_p^a - 1)C_1^* + C_2 \leq \sum_{i=0}^{n_p^a-2} T_1^{*,i}$, respectively.

Acceleration Example: Suppose that there is one CCRT rhythmic task τ_1^* and one periodic task τ_2: $(6, 14)$. Given (2,5) as (C_1^*, T_1^*), acceleration is possible when α is 0.3 and n_α is 1. As shown in Figure 6, the period of the rhythmic task will become 3.5 at time 5, and τ_2 meets its deadline at time 14. However, any α greater than 0.3 will make the taskset unschedulable. In this case, therefore, we can say that the maximum possible α is 0.3 when $n_\alpha = 1$.

C. Deceleration Analysis

Engine deceleration happens generally when the amount of fuel injected into the cylinders decreases. Engine deceleration is also significant since it is related to shifting of gears. This occurs very frequently in urban areas, and careless system design could affect the system schedulability whenever the gear shifting occurs. When the engine decelerates, the engine task periods increase.

For the deceleration analysis, α becomes a negative value to represent the rate of period increase. Under the given (C_1^*, T_1^*) at a certain time, let n_p^d denote the number of preemptions which τ_2 experiences while T_1^* is increasing. Then, if T_1^* starts increasing at the first job release, n_p^d is defined as $n_p^d = \max\{n | \sum_{i=0}^{n-1} T_1^{*,i} \leq T_2 \text{ and } n \in \mathbb{Z}^+\}$.

Given the definition of n_p^d, the following inequality should be satisfied, and the period does not increase after the n_β-th job of the rhythmic task if $n_p^d > n_\beta$.

$$\sum_{i=0}^{n_p^d-1} T_1^{*,i} \leq T_2 \tag{7}$$

Then, n_p^d should meet one of the following two inequalities:

$$\sum_{i=0}^{n_p^d-1} \left\{ f_C^*(T_1^{*,i}) \right\} + C_2 \leq T_2 \tag{8}$$

Algorithm 1 Rhythm-Max-C(Γ)

Input: Γ: a given taskset including a rhythmic task
Output: The WCET of the rhythmic task

```
1: for $i = 1$ to $n$ do
2:     ▷ Build a set of points to check
3:     $S_i = \{kT_j | j = 1...i, k$ is an integer satisfying
4:         $kT_j \leq T_i\}$
5:     ▷ From Inequalities (11) and (12)
6:     for For each element $s_i^m \in S_i$, do
7:         Calculate $C_{1,i}^m = \frac{s_i^m - \sum_{j=2}^{i} \left\lceil \frac{s_i^m}{T_j} \right\rceil C_j}{\left\lceil \frac{s_i^m}{T_1} \right\rceil}$
8:     Maintain the largest value of $C_{1,i}^m$ as $C_{1,i}$
9: return $min\{C_{1,i}, 1 \leq i \leq n\}$
```

$$\sum_{i=0}^{n_p^d-2} \left\{ f_C^*(T_1^{*,i}) \right\} + C_2 \leq \sum_{i=0}^{n_p^d-2} T_1^{*,i} \tag{9}$$

The reason behind these two inequalities is already mentioned in Section IV-B. Based on α, we can find the value of n_p^d which satisfies Inequalities (8) and (9) to check if one of those inequalities is satisfied. This process applies to both CURTs and GCRTs. For rhythmic tasks having fixed C_1^* (CCRT defined in Section III-B), they will be always schedulable because periods are *sustainable* as they are increased [19].

V. One Rhythmic Task and Many Periodic Tasks

In this section, we consider a taskset Γ with one rhythmic task τ_1^* and $n-1$ periodic tasks $\tau_2, ..., \tau_n$. In Section IV, we analyzed the case of having one rhythmic task and one periodic task. The results from the previous section will be extended to support several periodic tasks for constant speed, positive acceleration, and deceleration cases. A real-world example will be analyzed using this model in Section VI.

A. Steady-state Analysis

In order to determine the schedulability of Γ, we find the maximum value of C_1^* which does not make any periodic task miss its deadline. Let $f_{C_{max}}^*(T_1^*)$ denote the function which returns the maximum possible value of WCET for C_1^* which makes Γ schedulable when received T_1^* as an input.

Theorem 1. *$f_{C_{max}}^*(T_1^*)$ is given by*

$$\min_{\forall \tau_i \in \Gamma} \left\{ \max \left(T_1^* - \frac{\sum_{j=2}^{i} \left\lceil \frac{T_i}{T_j} \right\rceil C_j}{\left\lfloor \frac{T_i}{T_1^*} \right\rfloor}, \frac{T_i - \sum_{j=2}^{i} \left\lceil \frac{T_i}{T_j} \right\rceil C_j}{\left\lceil \frac{T_i}{T_1^*} \right\rceil} \right) \right\} \tag{10}$$

Proof: In order to check if Γ is schedulable, the worst-case response time of each periodic task τ_i should not exceed its deadline. Under the assumption of critical instant from [6], we should compare the worst-case response time of each periodic task to its deadline [20]. Then, as described in Lemma 1, two different cases should be considered.

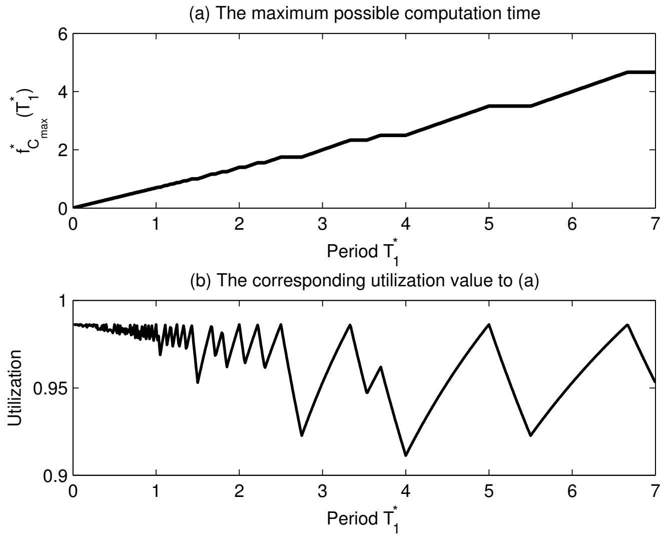


Figure 7. An example for a rhythmic task with 3 periodic tasks: τ_2: $(1,7)$, τ_3: $(1,10)$ and τ_4: $(1,23)$

The first case is that the execution time of the $\left\lceil \frac{T_i}{T_1^*} \right\rceil$-th instance of the rhythmic task is long enough to overlap T_i, where T_i is the deadline of the i^{th} periodic task τ_i.

$$\left\lfloor \frac{T_i}{T_1^*} \right\rfloor C_1^* + \sum_{j=2}^{i} \left\lceil \frac{T_i}{T_j} \right\rceil C_j \leq \left\lfloor \frac{T_i}{T_1} \right\rfloor T_1$$

As long as the following inequality is satisfied,

$$C_1^* \leq T_1^* - \frac{\sum_{j=2}^{i} \left\lceil \frac{T_i}{T_j} \right\rceil C_j}{\left\lfloor \frac{T_i}{T_1^*} \right\rfloor} \tag{11}$$

the given taskset is schedulable.

In the second case, the execution time of the $\left\lceil \frac{T_i}{T_1^*} \right\rceil$-th instance of the rhythmic task does not overlap T_i. Hence, the following inequality must be satisfied:

$$\left\lceil \frac{T_i}{T_1^*} \right\rceil C_1^* + \sum_{j=2}^{i} \left\lceil \frac{T_i}{T_j} \right\rceil C_j \leq T_i$$

Solving for C_1^* returns the following inequality.

$$C_1^* \leq \frac{T_i - \sum_{j=2}^{i} \left\lceil \frac{T_i}{T_j} \right\rceil C_j}{\left\lceil \frac{T_i}{T_1^*} \right\rceil} \tag{12}$$

Use the maximum value from Inequalities (11) and (12) as we are looking for the maximum allowable C_1^*. At this point, we have $n-1$ candidates for C_1^*. Since no periodic task must miss its deadline, we use the minimum value among those candidates. Then, $f^*_{C_{max}}(T_1^*)$ is given by Equation (10). □

Theorem 2. *The slope of $f^*_{C_{max}}(T_1^*)$ is either 1 or 0.*

Proof: We consider two different cases as the proof of Theorem 1. In the first case, Inequality (11) indicates that the slope of $f^*_{C_{max}}(T_1^*)$ is 1. In the second case, due to the fact that we consider the execution time of the $\left\lceil \frac{T_i}{T_1^*} \right\rceil$-th instance of the rhythmic task, the right hand side of Inequality (12) does not change as T_1^* changes. Hence, the slope of $f^*_{C_{max}}(T_1^*)$ is 0 with respect to T_1^*. □

Based on Theorems 1 and 2, we have designed an algorithm for finding the maximum possible value of WCET in Algorithm 1. Figure 7 shows an outcome of Algorithm 1 for a rhythmic task with 3 periodic tasks: τ_2: $(1,7)$, τ_3: $(1,10)$ and τ_4: $(1,23)$. As shown, the slope of the curve in Figure 7a is either 1 or 0, where the taskset is schedulable if (C_1^*, T_1^*) lies under the curve. Also, the minimum flexion point in Figure 7b is at 4 which satisfies Theorem 3 when $T_1^* = \frac{T_2}{2} + \frac{C_2}{2}$.

Corollary 1. *If a taskset is schedulable with (C_1^*, T_1^*), the taskset is schedulable with $\left(\frac{C_1^*}{k}, \frac{T_1^*}{k}\right)$, where k is a positive integer.*

Proof: Compute $f^*_{C_{max}}$. The parameters of (C_1^*, T_1^*) make the system schedulable. Then, $\left(\frac{C_1^*}{k}, \frac{T_1^*}{k}\right)$ will always be below $f^*_{C_{max}}$ from Theorem 2. □

Theorem 3. *The minimum flexion point lies in the range, $\frac{T_2}{2} + \frac{C_2}{2} \leq T_1^* \leq T_2$.*

Proof: The worst-case utilization happens when $1 \leq \frac{T_2}{T_1^*} \leq 2$ from [6]. Hence, $\frac{T_2}{2} \leq T_1^* \leq T_2$. From [6], since $\frac{T_2 - C_2}{2} \geq C_1^*$ holds good, $T_1^* + \frac{T_2 - C_2}{2} \geq T_2$ is satisfied. Then, by solving for T_1^*, $\frac{T_2}{2} + \frac{C_2}{2} \leq T_1^* \leq T_2$. □

Theorem 3 is critical to find the least-upper bound utilization of the given taskset regardless of the other tasks except τ_2, the second highest priority task. This theorem could also be a hint to find the least-upper bound utilization when only the task periods are given, a problem that appears to be unsolved yet.

B. Acceleration Analysis

Let α denote the rate of period decrease of τ_1^* and n_α denote the maximum positive acceleration duration in terms of the number of job releases of the rhythmic task. Then, the following theorem is satisfied.

Theorem 4. *Given a taskset Γ with one rhythmic task τ_1^* and $n-1$ periodic tasks, Γ is schedulable when the rhythmic task is accelerating if the following inequality is satisfied when $n_\alpha = 1$.*

$$\alpha \leq 1 - \frac{T_1^*}{C_1^*}\left(UB(n) - \sum_{i=2}^{n} \frac{C_i}{T_i}\right)$$

where $UB(n)$ returns the utilization bound [6] of n tasks.

Proof: While the acceleration of a rhythmic task continues, the period of the i^{th} instance affects the WCET of $(i+1)^{th}$ instance. Hence, $\frac{C_1^*}{T_1^*(1-\alpha)} + \sum_{i=2}^{n} \frac{C_i}{T_i} \leq UB(n)$ is satisfied from [6] when n_α is 1. Solving for α, $\alpha \leq 1 - \frac{T_1^*}{C_1^*}\left(UB(n) - \sum_{i=2}^{n} \frac{C_i}{T_i}\right)$ holds. □

The bound of Theorem 4 is not tight because it uses the utilization bound [6]. A tighter bound can be found by extending the results from Section IV. We need to extend the definition of n_p^a first. Let $n_{p,i}^a(t)$ denote the number of

Algorithm 2 Rhythmic-Acc-$\alpha(\Gamma, \alpha, n_\alpha)$

```
Input: Γ: a taskset including a rhythmic task, α: the acceleration ratio and n_α: the duration of rhythmic task acceleration in terms of the number of job releases
Output: Schedulability of Γ
 1: for $i = 2$ to $n$ do
 2:     ▷ Calculate the initial condition for each task $\tau_i$
 3:     $W_i^0 = C_1^* + \sum_{j=2}^{i} C_j$ and $W_i^1 = 0$
 4:     $k = 0$
 5:     while $W_i^{k+1} \neq W_i^k$ do
 6:         ▷ Pick the maximum number of preemptions for each iteration
 7:         $n_{p,i}^a(W_i^k)$ = Num-Preemptions$(T_1^*, \alpha, n_\alpha, W_i^k)$
 8:         $E_1^*$ = Execution-Time$(n_{p,i}^a(W_i^k), C_1^*, \alpha, n_\alpha)$
 9:         $W_i^{k+1} = C_i + E_1^* + \sum_{h=2}^{i-1} \left\lceil \frac{W_i^{k+1}}{T_h} \right\rceil C_h$
10:         Update necessary parameters
11:     if $W_i^k \leq D_i$ then
12:         Mark $\tau_i$ schedulable
13: if all tasks schedulable then
14:     return TRUE
15: else
16:     return FALSE
```

Algorithm 3 Num-Preemptions$(T_1^*, \alpha, n_\alpha, W_i^k)$

```
1: ▷ The time duration of rhythmic task acceleration
2: $d_{acc} = \sum_{j=0}^{n_\alpha - 1} \{T_1^*(1-\alpha)^j\}$
3: if $d_{acc} > W_i^k$ then
4:     $n_{p,i}^a(W_i^k) = \max\{l | \sum_{j=0}^{l-1} \{T_1^*(1-\alpha)^j\} \leq W_i^k$,
5:     where $l \in \mathbb{Z}^+\}$
6: else
7:     $n_{p,i}^a(W_i^k) = n_\alpha + \left\lceil \frac{W_i^k - d_{acc}}{T_1^*(1-\alpha)^{n_\alpha - 1}} \right\rceil$
8: return $n_{p,i}^a(W_i^k)$
```

Algorithm 4 Execution-Time$(n_{p,i}^a(W_i^k), C_1^*, \alpha, n_\alpha)$

```
1: if $n_{p,i}^a(W_i^k) < n_\alpha$ then
2:     $E_1^* = \sum_{j=0}^{n_{p,i}^a(W_i^k)} \{C_1^*(1-\alpha)^j\}$
3: else
4:     $E_1^* = \sum_{j=0}^{n_\alpha - 1} \{C_1^*(1-\alpha)^j\} + (n_{p,i}^a(W_i^k) - n_\alpha) \times C_1^*(1-\alpha)^{n_\alpha - 1}$
5: return $E_1^*$
```

preemptions of the periodic task τ_i caused by the rhythmic task τ_1^* during t units of time. Then, $n_{p,i}^a(t)$ is defined as $n_{p,i}^a(t) = \max\{n | \sum_{j=0}^{n-1} T_1^{*,j} \leq t, n \in \mathbb{Z}^+\}$. Therefore, the following inequality should be satisfied.

$$\forall i, \sum_{j=0}^{n_{p,i}^a(T_i)-1} T_1^{*,j} \leq T_i, i \in \{k | k \in \mathbb{Z}^+ \text{ and } k \geq 2\} \quad (13)$$

Then, by using the value found above, the response-time test [21] has to be modified as

$$W_i^{k+1} = C_i + C_1^* + \sum_{j=0}^{n_{p,i}^a(W_i^k)} f_C^*(T_1^{*,j}) + \sum_{h=2}^{i-1} \left\lceil \frac{W_i^{k+1}}{T_h} \right\rceil C_h \quad (14)$$

where $W_i^0 = C_1^* + \sum_{j=2}^{i} C_j$ and the test terminates when $W_i^{k+1} = W_i^k$.

Based on Inequality (13) and Equation (14), the generalized algorithm is given in Algorithm 2 which checks if a taskset is schedulable under the given α and n_α. Algorithm 3 is used for obtaining the number of preemptions caused by the rhythmic task, and Algorithm 4 calculates the preemption duration during the execution for CURT. The maximum value can be found by using this function for the range of α.

For analyzing engine deceleration, the definition of n_p^d also needs to be extended, and a similar modified response-time test can be used.

C. Guidelines for CPS Application Developers

In this subsection, we will provide some guidelines that help CPS application developers to apply the rhythmic task analysis results and guarantee the schedulability of the system. The developers should ensure that:

1) The application is categorized into one of the three categories: CCRT, CURT and GCRT.
2) The computation time of a rhythmic task lies under the schedulable region as depicted in Figures 4 and 7.
    - The minimum flexion point of the total utilization can be used. By not exceeding this bound, the system schedulability is guaranteed. However, it should be noted that this bound could be pessimistic.
    - Algorithm 1 and Theorem 1 can be used together to find the exact schedulable region.
3) The application can be in the form of modules for different speeds. The application can have all the modules or a subset of the modules depending upon the execution time to meet the system schedulability.
4) The difference between the maximum allowable worst-case execution time and the actual computation time should be enough to tolerate the acceleration for the values α, n_α and n_β. Theses values are computed by using Algorithm 2.

Once the parameters of the schedulable rhythmic task (C^*, T^*) are found, this information can be used for finding other schedulable regions. For CCRT, the rhythmic task with a longer period will be schedulable. For CURT, the rhythmic task with $(\frac{C^*}{k}, \frac{T^*}{k})$, where k is a positive integer, will be also schedulable.

VI. Case Study of the Rhythmic Task Model

In this section, we provide a case study to show how to apply the rhythmic task model to an existing CPS. Our model is applicable to a generic CPS having tasks with

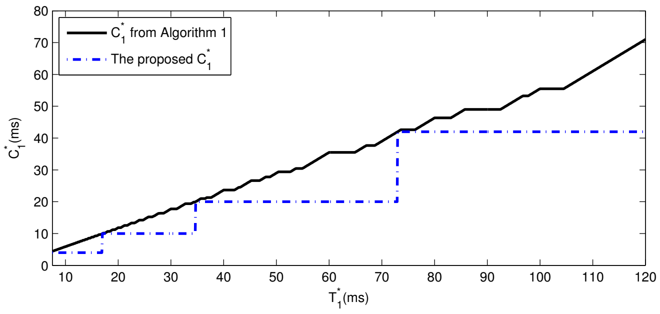


Figure 8. The maximum possible computation time for a rhythmic task that has varying period from 7.5ms to 120ms with 9 periodic tasks

varying periods. In this paper, we investigate the automotive PCM from Section I.

Figure 9 illustrates a process for injecting and delivering fuel to each cylinder at every revolution [22]. Since the depicted task is triggered by timing signals from engine events, increasing/decreasing the duration of revolution will change the period of the given task. Suppose the RPM varies from 500 to 9000, so the period of the corresponding task varies from $7.5ms$ to $120ms$. The operations illustrated in Figure 9 are also executed. The execution path is composed of a service routine, sensor reads, air calculation, fuel calculation and fuel delivery. We model this task as a rhythmic task. Each block has its own WCET: Service Routine and Fuel Delivery: $4ms$, Sensor Reads: $6ms$, Air Calculation: $10ms$ and Fuel Calculation: $22ms$. In addition, a typical Engine Control Module (ECM) has other features [23] such as monitoring the processor that runs the control algorithms, reporting the current status to a diagnosis module, and managing sensors which measure the amount of fuel injected. We picked nine tasks to show the typical behaviors of the ECM representing the periodic engine operations [23] and study the impact of having a rhythmic task. Specifically, we use the following periodic tasks: τ_2:$(5ms, 120ms)$, τ_3:$(20ms, 120ms)$, τ_4:$(5ms, 180ms)$, τ_5:$(6ms, 200ms)$, τ_6:$(8ms, 240ms)$, τ_7:$(10ms, 240ms)$, τ_8:$(3ms, 300ms)$, τ_9:$(1ms, 360ms)$ and τ_{10}:$(7ms, 400ms)$.

The solid line in Figure 8 shows the maximum possible computation time of the rhythmic task using Algorithm 1. As mentioned earlier, if the value of (C_1^*, T_1^*) is below the given solid curve, the taskset is schedulable. This offers design-time guidelines to determine the feasible WCET of the engine control task. The dotted line is a recommendation for when a particular software block from Figure 9 can be executed. The Service Routine and Fuel Delivery functions will be executed by every job of the rhythmic task regardless of the value of T_1^*. The Sensor Reads function will be executed in addition to Service Routine and Fuel Delivery if T_1^* is greater than $17ms$. Since the WCET of Sensor

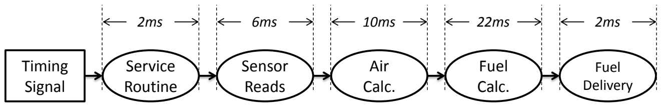


Figure 9. Flow diagram for the start of injection in PCM software

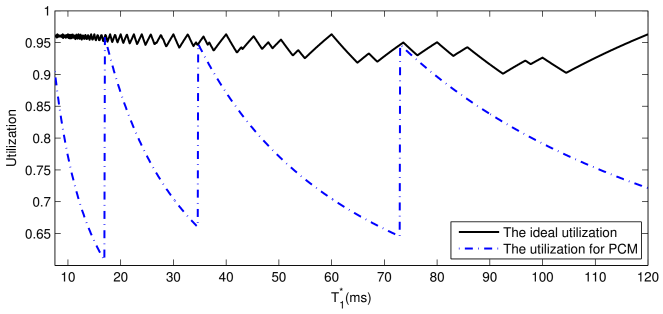


Figure 10. The corresponding utilization value to Figure 8 for a rhythmic task that has varying period from $7.5ms$ to $120ms$ with 9 periodic tasks

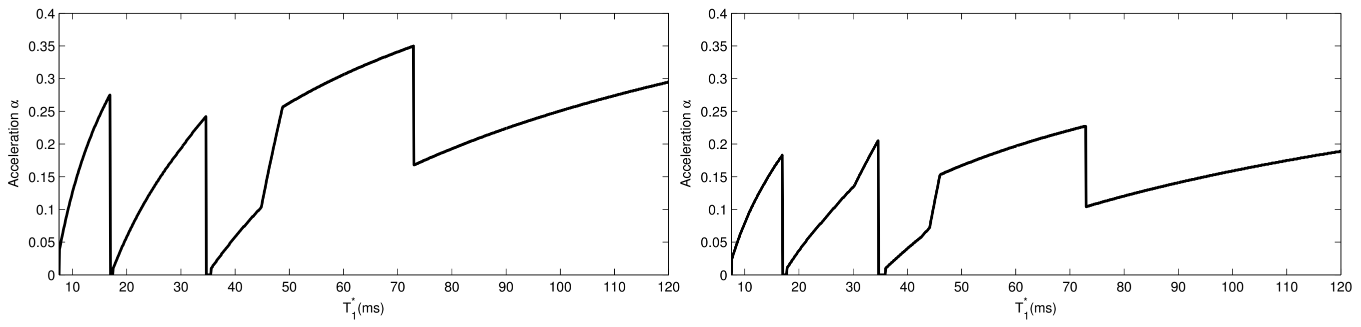
Reads is $6ms$, it can make the system unschedulable if T_1^* is smaller than $17ms$. In this case, however, instead of not running the whole instructions of Sensor Reads, the previous sensor values can be used for the operation. Similarly, the function Air Calculation will be executed if T_1^* is greater than $34.7ms$; Fuel Calculation will be executed if T_1^* is greater than $73ms$. These recommendations correspond to making the rhythmic task a GCRT with a discrete step function.

Figure 10 illustrates the corresponding utilization based on the WCET given in Figure 8. It shows the maximum allowable utilization of the rhythmic task, where the minimum flexion point is at $T_1^* = 92.5ms$, where $\frac{T_2}{2} + \frac{C_2}{2} \leq T_1^* = 92.5ms \leq T_2$, which is computed using Algorithm 1 and Theorem 3. An alternative way of determining the engine control task behavior is to use this information to ensure that the total utilization does not exceed this bound. The dotted line in Figure 10 is the utilization curve corresponding to our recommendation above.

Figure 11 depicts the bound for Acceleration α considering the maximum allowable rate of period change with different acceleration durations. These curves are generated by using Algorithms 2, 3 and 4, where Algorithm 4 is modified such that it can handle GCRT. The plots use the recommended WCET corresponding to the dotted line from Figure 8. As shown in the figures, the acceleration bound plays a negative role. For example, we cannot accelerate the engine at all when T_1^* is $17ms$ or $34.7ms$ because the processor is fully utilized already. This can be avoided by delaying the timing of changing the execution mode from $17ms$/$34.7ms$ to later values. The effect of the acceleration duration is also shown in Figures 11a and 11b. Figure 11a is when the maximum acceleration duration n_α is 3, and n_α is 5 in Figure 11b. When the acceleration duration is longer, the acceleration bound becomes significantly low. This happens because a longer acceleration duration increases the number of preemptions of periodic tasks.

VII. Conclusion and Future Work

In automotive systems, safety-critical mechanical systems are being replaced by electronically controlled systems. A critical task not meeting its deadline can be catastrophic. In order to meet these stringent requirements, real-time scheduling techniques such as Rate Monotonic Scheduling



(a) Maximum acceleration duration n_α: 3 (b) Maximum acceleration duration n_α: 5

Figure 11. Plots of acceleration values for a rhythmic task that has varying period from $7.5ms$ to $120ms$ with 9 periodic tasks

(RMS) are used to guarantee the schedulability of the periodic tasks. However, the parameters of certain critical control tasks in cyber-physical systems depend on physical attributes of the system such as the speed of the engine in a car. The periods of these engine tasks vary dramatically depending on the engine speed. Conventional periodic task analysis is too conservative for handling such tasks. In this paper, we have defined a new task model called *Rhythmic Tasks* for modeling tasks having continually varying periods depending on external physical events. To the best of our knowledge, this is the first model considering continually varying periods. We provide response-time analysis techniques for rhythmic tasks under constant engine speed, accelerating engine speed and decelerating engine speed along with schedulability tests. We have also provided guidelines to find schedulable utilization levels for the rhythmic task model. We apply our analyses and guidelines to a case-study desired from a real environment. The case study shows how the rhythmic task model is applicable to an existing CPS.

Dealing with multiple rhythmic tasks is an important and needed extension for cyber-physical systems. For example, the periods of planning and perception tasks in an autonomous vehicle [17] are functions of the vehicle speed, and the rates of their period changes depend on various environmental factors.

Future applications of rhythmic tasks include fault-tolerance support and autonomous vehicles. Rhythmic tasks can be replicated for fault-tolerance, and our techniques need to be extended. This rhythmic task model can also be used in autonomous vehicle systems. For example, perception algorithms for vision-based obstacle detection can be analyzed using the rhythmic task model.

Acknowledgment

The authors would like to thank Arvind Kandhalu and Dr. Shinpei Kato for their valuable comments and discussions that improved the paper.

References

[1] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic. Cyber-physical systems: the next computing revolution. In *Proceedings of the 47th Design Automation Conference*, DAC '10, pages 731–736, New York, NY, USA, 2010. ACM.

[2] Four stroke engine. http://www.ustudy.in/node/3268 as of Feb 2012.

[3] ETAS. RTA-OSEK. http://www.etas.com/en/products/rta_osek.php as of Feb 2012, January 2007.

[4] AUTOSAR Administration. AUTOSAR technical overview, 2008.

[5] Z. Gu, S. Wang, J.C. Kim, and K.G. Shin. Integrated modeling and analysis of automotive embedded control systems with Real-Time scheduling. Technical Report 2004-01-0279, SAE International, Warrendale, PA, March 2004.

[6] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20:46–61, 1973.

[7] A. MOK. Fundamental design problems of distributed systems for the real-time environment. *Ph.D. thesis, Dept. of Electrical Engineering and Computer Science, MIT, Cambridge, Mass.*, May 1983.

[8] J. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *Real-Time Systems Symposium, 1990. Proceedings., 11th*, pages 201 –209, dec 1990.

[9] T.D. Gillespie. *Fundamentals of Vehicle Dynamics*. Society of Automotive Engineers, 1992.

[10] C. Li and Z. Jianwu. WCET analysis for gasoline engine control. In *Mechatronics and Automation, 2005 IEEE International Conference*, volume 4, pages 2090–2095 Vol. 4, 2005.

[11] G. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni. Elastic scheduling for flexible workload management. *IEEE Transactions on Computers*, page 289–302, 2002.

[12] R. Guerra and G. Fohler. A gravitational task model for target sensitive real-time applications. In *ECRTS08-20th Euromicro Conference on Real-Time Systems*, 2008.

[13] H. Wei, K. Lin, W. Lu, and W. Shih. Generalized rate monotonic schedulability bounds using relative period ratios. *Information Processing Letters*, 107(5):142 – 148, 2008.

[14] L. George and J. Hermant. A norm approach for the partitioned edf scheduling of sporadic task systems. *Real-Time Systems, Euromicro Conference on*, 0:161–169, 2009.

[15] T.F. Abdelzaher, V. Sharma, and C. Lu. A utilization bound for aperiodic tasks and priority driven scheduling. *Computers, IEEE Transactions on*, 53(3):334 – 350, mar 2004.

[16] L. Sha, R. Rajkumar, J. Lehoczky, and K. Ramamritham. Mode change protocols for priority-driven preemptive scheduling. *Real-Time Systems*, 1(3):243–264, December 1989.

[17] C. Urmson et al. Autonomous driving in urban environments: Boss and the urban challenge. In *The DARPA Urban Challenge*. 2009.

[18] A. Rowe, M. Berges, and R. Rajkumar. Contactless sensing of appliance state transitions through variations in electromagnetic fields. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '10, pages 19–24, New York, NY, USA, 2010. ACM.

[19] A. Burns and S. Baruah. Sustainability in real-time scheduling. *Journal of Computing Science and Engineering*, 2(1):74–97, 2008.

[20] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A.J. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8(5):284–292, 1993.

[21] M. Joseph and P. Pandya. Finding response times in a real-time system. *The Computer Journal*, 29(5):390, 1986.

[22] X. Hu, J.G. D'Ambrosio, B.T. Murray, and D. Tang. Codesign of architectures for automotive powertrain modules. *Micro, IEEE*, 14(4):17–25, 1994.

[23] K. Tindell and A. Burns. Guaranteeing message latencies on control area network (CAN). In *Proceedings of the 1st International CAN Conference*. Citeseer, 1994.