Schedulability Tests for Tasks with Variable Rate-Dependent Behaviour under Fixed Priority Scheduling

Robert I. Davis
Real-Time Systems Research Group,
University of York
rob.davis@cs.york.ac.uk

Timo Feld, Victor Pollex, Frank Slomka
Institute of Embedded Systems / Real-Time Systems
Ulm University, Germany
{timo.feld, victor.pollex, frank.slomka}@uni-ulm.de

Abstract*—Automotive embedded real-time systems such as Engine Management utilise cyclic tasks that are activated periodically based on angular rotation rather than time. As well as having variable inter-arrival times, these tasks also have deadlines and worst-case execution times that are dependent on angular velocity i.e. engine speed or rpm. Such tasks exhibit Variable Rate-dependent Behaviour (VRB). In this paper, we introduce response time analysis for systems comprising VRB and sporadic tasks under fixed priority scheduling. Sufficient schedulability tests are introduced; from simple linear upper bounds on interference, to a more complex analysis using information about the physical limitations of the system to provide constraints for an ILP formulation of the problem.

Keywords: real-time scheduling; schedulability analysis; fixed priority; variable rate; variable deadline; variable execution time; mode changes; automotive;

I. Introduction

In automotive embedded real-time systems, some tasks in the Engine Management ECU (Electronic Control Unit) are activated according to interrupts generated by a sensor reading the crankshaft position. These tasks execute with a variable inter-arrival time or period reflecting the angular velocity of the crankshaft (i.e. engine speed or rpm). The purpose of these tasks includes determining parameters controlling ignition timing, fuel injection, inlet and exhaust valve timing etc. The deadline of each job of these tasks is also determined by the engine speed and relates to a specific angular position of the crankshaft or camshafts.

In a typical four cylinder, 4-stroke engine, a cylinder fires every 180 degrees of crankshaft rotation; hence at an idle speed of 600rpm, the tasks computing ignition timing and fuel injection parameters have a period that equates to 50ms, whereas at 6000rpm, this period reduces to 5ms with a corresponding reduction in the tasks' deadlines. The worst-case execution time of the tasks is also dependent on engine speed. The fuel injection system for a typical petrol engine uses three injection pulses per cycle at low rpm and one pulse at high rpm. By contrast, a typical diesel engine uses seven injection pulses at low rpm, and three pulses at high rpm. This is due to the fact that there is simply not enough time for seven pulses at high rpm. Further, at high engine speeds, the input data (e.g. accelerator pedal position) does not change so much per cycle, as the elapsed time is shorter, and so it is sufficient to compute the amount of fuel that should be injected every two invocations of the task.

In general, at lower engine speeds, typical of normal driving, complex functionality is executed minimising fuel consumption and emissions, and ensuring that the engine runs as smoothly as possible; however, if this functionality was also executed at high rpm, then processor utilisation would become prohibitive and the system unschedulable. Instead, some functionality is shed at high engine speeds.

In his keynote talk [12] at ECRTS 2012, Buttle highlighted the problem of tasks with *Variable Rate-dependent Behaviour* (VRB), formulating it as a specific schedulability analysis challenge.

```
TASK(Variant_execution)(){
    f1();
    if(rpm < 3000) {
        f2();
    }
    f3();
}
```

Figure 1: Task with execution time dependent on its period.

Figure 1 adapted from slide 35 of [12] illustrates the pseudo code for a task that sheds some functionality at high rpm, and thus has a worst-case execution time (WCET) that correlates with its inter-arrival time. This task effectively has two execution modes corresponding to two distinct ranges for its arrival rate or inter-arrival time.

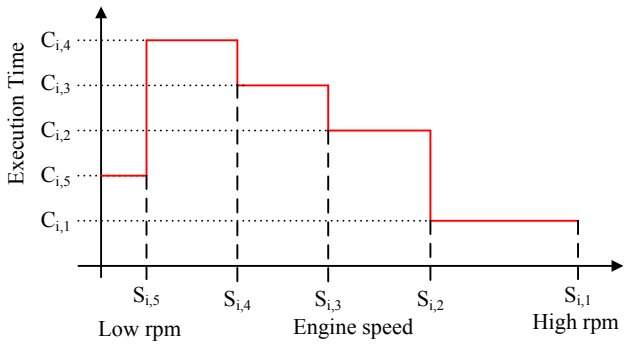

Figure 2: Worst-case execution time of a task as a function of engine speed.

In general, VRB tasks may be modelled as having a number of execution modes, each related to a fixed range of inter-arrival times or periods, as illustrated in Figure 2. A VRB task is modelled as changing from one execution mode to another at a particular value of angular velocity and hence inter-arrival time. Figure 2 shows five different execution modes for a single task, and how they relate to engine rpm.

As an alternative solution to VRB tasks, some systems use a periodic task with a fixed inter-arrival time, combined with interrupt handlers that make inputs and outputs triggered by crankshaft rotation. However, this approach suffers from the classic problems of polling delays and jitter, while providing only limited capability to accommodate functionality that is dependent on engine speed. This is due to the need to continually run the periodic task at a high rate to support high rpm operation, something that is unnecessary with VRB tasks and fully event-driven operation.

In this paper, we address the challenge described by Buttle, introducing sufficient schedulability tests for VRB and sporadic tasks executing on a single processor, under a fixed priority pre-emptive scheduler (such as the OSEK or AUTOSAR RTOS used in automotive applications).

A. Related Work

Schedulability analysis has been developed for a variety of different task models, including: periodic tasks [19], sporadic tasks [22], multi-frame tasks [34], [23], generalised multi-frame (GMF) tasks [5], non-cyclic GMF tasks [24], recurring real-time tasks [6], non-cyclic recurring real-time tasks [7], and the digraph task model [28], extended with constraints in [29]. Currently, the most general is the digraph task model which describes each task via a directed graph. Here, each node represents a type of job the task can release and is labelled with its deadline and execution time. An edge is labelled with the minimum time between activations of the jobs it connects. Constraints are added to this in [29] to specify a minimum time between nodes.

While the problem we address could potentially be mapped onto the non-cyclic GMF or digraph task models, recent work has shown that exact analysis of the GMF task model and the more general digraph task model are intractable [30] assuming fixed priority scheduling, while the complexity of exact analysis for the non-cyclic GMF task model is to the best of our knowledge unknown. This contrasts with EDF scheduling where pseudo-polynomial time exact schedulability tests exist and are known for the digraph task model [28]. Some progress has however been made on sufficient schedulability tests for the non-cyclic GMF model [9] for fixed priority scheduling, while exact tests have recently been developed for the digraph task model which although having exponential complexity in the worst-case are in practice of similar efficiency to the pseudo-polynomial time tests for EDF [31].

The problem of scheduling tasks with Variable Rate-dependent Behaviour has some similarities to the classical problem of system-wide mode changes in hard real-time systems [32], [27]. In the classical case, on a mode change, some tasks change their parameters (e.g. execution time, deadline, and period) and all tasks must be schedulable in the old mode, in the new mode, and also across the mode change transition. Typically, no further mode changes are permitted until the new mode is fully established (i.e. all tasks have switched to their new mode parameters and the processor has since become idle). VRB tasks differ from this classical description of a mode change in that different VRB tasks may change their execution mode according to different thresholds (inter-arrival times), and multiple changes of each task's execution mode may take place over consecutive jobs of the task. Further, in the general case with VRB tasks driven from different angular sources (e.g. engine speed, wheel speed, etc.), different VRB tasks may progress through their different execution modes independent of each other. Thus the concept of execution modes in VRB tasks is distinct from that of system-wide operating modes.

Some preliminary steps have previously been taken to analyse VRB tasks under fixed priority scheduling: In 2013 [26] Pollex et al. considered systems where the rotational speed is arbitrary, but fixed; however, this simple first step does not account for the important effect of transitions between different execution modes. Pollex et al. [25] subsequently provided a simple analysis for systems with angular acceleration, but considering only the maximum execution time and the minimum inter-arrival time that could be obtained in the analysis interval, starting from different engine speeds. Kim et al. [21] also studied VRB tasks (referred to as rhythmic tasks), but only accounted for a single VRB task with the highest priority among a set of periodic tasks. In this paper, we provide analysis for the more general and practical case of multiple VRB tasks at arbitrary priorities among sporadic tasks, and fully account for the dynamic behaviour of the system.

Initial work on analysing VRB tasks under EDF scheduling has been carried out by Buttazzo et al. [11], providing a simple utilisation-based test for implicit deadline tasks, which also accounts for dynamic behaviour.

B. Organisation

The remainder of the paper is organised as follows. Section II describes the system model, terminology and notation used. Section III presents sufficient schedulability tests for VRB tasks assuming that any arbitrary sequence of permitted inter-arrival times and hence execution modes is possible. Section IV uses information about the physical limitations of the system (i.e. maximum rate of engine acceleration and deceleration) to provide more precise schedulability analysis. Section V provides an experimental evaluation, comparing the effectiveness of the various schedulability tests. Finally, section VI concludes with directions for future work.

II. System model, Terminology and Notation

In this paper, we consider the fixed priority pre-emptive scheduling of a set of n tasks on a single processor. Each task τ_i is assumed to have a unique index i from 1 to n representing its priority. We assume a discrete time model, so all task parameters are integers.

We assume that each VRB task τ_i may give rise to a potentially unbounded sequence of invocations (or *jobs*). We assume that task τ_i has M_i (≥ 1) unique execution modes corresponding to a distinct set of inter-arrival time intervals

$[T_{i,1},T_{i,2})$, $[T_{i,2},T_{i,3})$, … $[T_{i,M_i-1},T_{i,M_i})$, $[T_{i,M_i},\infty]$ where $T_{i,1} < T_{i,2} < T_{i,3} < ... < T_{i,M_i}$. Each execution mode m of task τ_i is thus characterised by a triplet $(C_{i,m},D_{i,m},T_{i,m})$ representing the worst-case execution time (WCET) $C_{i,m}$, minimum relative deadline $D_{i,m}$, and minimum inter-arrival time or period $T_{i,m}$ for a job executing in that mode. Note the minimum inter-arrival time $T_{i,m}$ for a mode corresponds directly to the maximum angular velocity (e.g. engine speed) $S_{i,m}$ for that mode, under steady state conditions (i.e. constant engine speed). The mode m of each job of task τ_i is determined at runtime by the task's inter-arrival time.

We assume that each execution mode of each VRB task has a *constrained deadline*, and is not trivially unschedulable, hence $\forall i,m\ C_{i,m} \le D_{i,m} \le T_{i,m}$, thus each task meets the *frame separation* constraint, whereby each job of the task must complete before the next job is released. We place no other restrictions on the relative values of these parameters across different execution modes. We note that both the release and absolute deadline of a job of a VRB task typically correspond to angular positions, and thus the task's period and its relative deadline are variable, dependent on engine speed, but related by some constant $D_i = \lambda T_i$. Our model is however more general than this and copes with the situation were, for example the deadline is some fraction of the task's period plus a fixed time.

We use $C_i^{\max}$, $D_i^{\max}$, $T_i^{\max}$ to mean the maximum execution time, relative deadline, and period of any job of task τ_i in any execution mode, and similarly, $C_i^{\min}$, $D_i^{\min}$, and $T_i^{\min}$ to mean the minimum of such values. The maximum processor utilisation of task τ_i in mode m is given by $U_{i,m} = C_{i,m} / T_{i,m}$. The maximum utilisation of the task in any mode is denoted by $U_i^{\max}$.

Simple sporadic tasks are also accommodated in the model. They have a single execution mode, with no dependency on engine speed.

We assume that tasks may access shared resources according to the Stack Resource Policy [4], and so a job of task τ_i which executes in mode m may be blocked for at most $B_{i,m}$ during which the processor is occupied by a lower priority task accessing a resource that is shared with the mode m execution of task τ_i or a higher priority task. We assume that tasks are otherwise independent and do not have any precedence constraints.

The *worst-case response time* $R_{i,m}$ of a job of task τ_i which executes in mode m is given by the longest possible time from release of such a job until it completes execution. Thus task τ_i is schedulable if and only if for every execution mode m of the task, $R_{i,m} \le D_{i,m}$, and a taskset is schedulable if and only if all of its tasks are schedulable.

The VRB task model is a general one; it covers tasks that are driven from different angular sources (e.g. engine speed, wheel speed etc.) and hence have inter-arrival times (and execution modes) that are independent of one another. It also covers tasks that are driven from the same angular source, but have different thresholds (angular velocities) denoting their transitions from one mode to the next, as well as tasks that use the same thresholds and effectively transition through their execution modes in lock-step. Further, it also covers tasks that only actually execute in a subset of their execution modes (e.g. at high engine speeds). For clarity, this latter case is not explicitly considered in the analysis; however, it is easily catered for by setting the worst-case execution time of the task to zero for any modes in which it does not execute. To ease consideration of inter-arrival times, and the overall interference on other tasks, empty jobs (with a worst-case execution time of zero) should still be regarded as arriving. (We note that schedulability analysis is not required, or valid, for execution modes with a worst-case execution time of zero).

III. SCHEDULABILITY ANALYSIS FOR VRB TASKS

In this section, we derive sufficient schedulability tests for VRB tasks. First, we briefly recapitulate on Response Time Analysis [3] used to provide an exact schedulability test for sporadic tasks with constrained deadlines. We then discuss what is required for schedulability analysis of VRB tasks, giving a Theorem that helps in deriving this analysis. We then provide schedulability tests for VRB tasks for the general case of tasks with multiple execution modes, using (i) an ILP formulation and (ii) a simple linear upper bound.

The schedulability tests given in this section make no assumptions about the relationships between the inter-arrival times or execution modes of different VRB tasks. Hence, the tests are applicable to tasks driven from angular sources with different behaviours (e.g. engine speed, wheel speed etc.).

A. Recapitulation of Schedulability Analysis for FPPS

Under fixed priority pre-emptive scheduling, the worst-case response time R_i of a constrained-deadline, sporadic task τ_i corresponds to the length of the longest priority level-*i busy period*, which starts at a critical instant. The busy period comprises three components, the blocking time B_i , the execution time C_i of the task itself, and so called *interference*, equal to the time for which task τ_i is prevented from executing by higher priority tasks. The length of the busy period w_i , can be computed using the following fixed point iteration [3], with the summation term giving the interference due to the set of higher priority tasks *hp*(*i*).

$$w_i^{q+1} = B_i + C_i + \sum_{\forall j \in hp(i)} I_j(w_i^q) \qquad (1)$$

where:

$$I_j(w) = \left\lceil w / T_j \right\rceil C_j \qquad (2)$$

Iteration starts with an initial value w_i^0 , typically $w_i^0 = C_i$, and ends when either $w_i^{q+1} = w_i^q$ in which case the worst-case response time R_i , is given by w_i^{q+1} , or when $w_i^{q+1} > D_i$ in which case the task is unschedulable. We note that convergence can be speeded up by starting with a suitable lower bound on the response time [13].

We note that baseline schedulability analysis for VRB tasks can be obtained by pessimistically converting each VRB task τ_i into a sporadic task such that $C_i = C_i^{\max}$,

$D_i = D_i^{\min}$, $T_i = T_i^{\min}$; however, such a simple approach can potentially be grossly pessimistic.

B. Maximum interference and the mode change problem

We now consider schedulability analysis for VRB tasks. Given the frame separation constraint and fixed priority pre-emptive scheduling, then there can be no push-through interference from one job of a task to the next. Hence to prove the schedulability of a task τ_i, we need only show that it is schedulable in each execution mode *m*, assuming a priority level-*i* busy period starting at the release of the task in that mode. The difficulty arises in determining the worst-case interference from each higher priority VRB task in that busy period. One might naively assume that it is sufficient to compute the interference from each higher priority task in each of its execution modes and take the largest value; however, this is not sufficient as we now show.

Consider a system with two tasks; a VRB task τ_1, and a sporadic task τ_2, with τ_1 having a higher priority than τ_2. Assume that task τ_1 has a period corresponding to 360° of crankshaft rotation, and a deadline corresponding to 180°. Further, below 3000rpm, (i.e. 20ms period, 10ms deadline), it has a WCET of 5ms, and above 3000rpm, but below the maximum engine speed of 6666 rpm (9ms period, 4.5ms deadline) it has a WCET of 2ms. Task τ_2 has a fixed period of 50ms, a deadline of 35ms, and a WCET of 25ms.

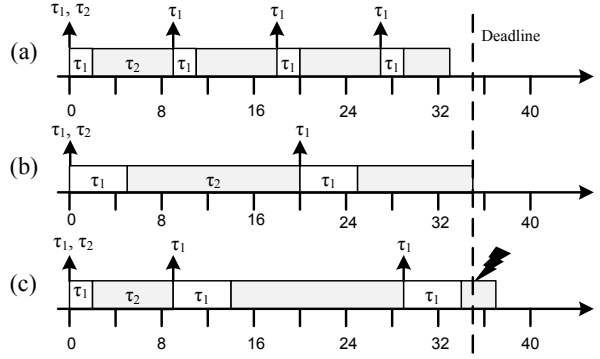

Figure 3: Mode transitions may provide the worst-case interference.

In this example, task τ_1 would be trivially unschedulable if it did not shed some functionality, as its low rpm execution time of 5ms is greater than its minimum high rpm deadline of 4.5ms. However, with its variable rate-dependent behaviour task τ_1 is schedulable, and we would like to know if this is also the case for task τ_2.

Using conventional response time analysis, if the system operates continuously at high rpm (6666rpm), then the response time of task τ_2 is 33ms, see Figure 3(a). Similarly, if it operates continuously at just under 3000rpm, then the response time of task τ_2 is 35ms, see Figure 3(b). However, if after a cycle of high rpm operation, we get two cycles of low rpm operation for τ_1, then the total interference from task τ_1 will be 12ms and task τ_2 will have a response time of 37ms and so miss its deadline, see Figure 3(c). This is an example of the classic mode change problem [32].

C. Sequences maximising interference

To obtain schedulability tests for VRB tasks, we need to consider the maximum amount of interference $I_j(w)$ due to a higher priority VRB task τ_j that can be released in a window of length *w*. The example in Figure 3 showed that in general it is necessary to consider all possible combinations of execution modes in deriving this worst-case interference.

Theorem 1: There is a sequence *Y* of jobs of task τ_j, that releases the maximum interference $I_j(w)$ in a window [0,*w*), where (i) the offset, from the start of the window, of the first job of τ_j is zero, (ii) each of the jobs of τ_j released in the window has the minimum period commensurate with its particular execution mode, and (iii) the last job has the largest WCET for any execution mode. (Note, sequence *Y* may, without restriction other than (iii), contain jobs of a number of different execution modes).

Proof: We assume that there exists some arbitrary sequence *X* of jobs of task τ_j, that releases the maximum amount of interference $I_j(w)$ in the window [0,*w*). Note, sequence *X* makes no restrictions on the inter-arrival times of the jobs of task τ_j, only that they are valid (i.e. $\geq T_j^{\min}$), hence the different jobs may have different execution modes (and execution times) commensurate with their inter-arrival times. We prove the three aspects of the theorem by transforming sequence *X* into sequence *Y* without reducing the interference $I_j(w)$. (i) We move the release of every job in *X* earlier by the offset of the first job in *X*. As all of the jobs continue to be released within the window, the amount of interference cannot decrease. (ii) We reduce the time intervals between releases to the minimum for the corresponding execution mode. Again, as all of the jobs continue to be released within the window, the overall interference cannot decrease. (iii) We change the execution mode of the last job to the mode that has the maximum execution time $C_j^{\max}$. As this job is by definition the last to be released in the window, then any increase in its period cannot cause a reduction in interference due to later release of the following job. Further, setting the execution mode of the last job in this way cannot decrease the amount of execution time released in the window, as it now has the largest WCET of any job of the task. □

D. Schedulability analysis for VRB tasks

We now make use of Theorem 1 to derive an upper bound on the interference $I_j(w)$ from a VRB task τ_j released in a window of arbitrary length *w*. Let $k_{j,x} \geq 0$ be the integer number of jobs of execution mode *x* released by task τ_j within a window of length *w*. From Theorem 1, finding the maximum interference is equivalent to maximising:

$$I_j(w) = \sum_{\forall x} k_{j,x} C_{j,x} \tag{3}$$

Subject to the constraints that:

$$k_{j,x} \geq 0 \quad \forall x \neq y$$
$$k_{j,x} \geq 1 \quad x = y$$
$$\sum_{\forall x} k_{j,x} T_{j,x} \leq w + T_{j,y} - 1 \tag{4}$$

where y is the mode with the longest execution time (arbitrarily chosen in the case of ties). Note that the last job must be released strictly before the end of the window, hence the '-1' in the final inequality in (4) as all values are integers.

The above Integer Linear Programming (ILP) problem in the variables $k_{j,x}$ is a combinatorial optimisation problem which can be solved in a reasonable time frame (see Section V.D for runtime information) for the small numbers of execution modes characteristic of real systems.

Assuming that $I_j(w)$ can be found, then an upper bound on the worst-case response time $R_{i,m}$ of any job of task τ_i executing in mode m can be computed as follows:

$$w_{i,m}^{q+1} = B_{i,m} + C_{i,m} + \sum_{\forall j \in hp(i)} I_j(w_{i,m}^q) \quad (5)$$

Iteration starts with an initial value $w_{i,m}^0$, typically $w_{i,m}^0 = C_{i,m}$, and ends when either $w_{i,m}^{q+1} = w_{i,m}^q$ in which case the worst-case response time $R_{i,m}$, is given by $w_{i,m}^{q+1}$, or when $w_{i,m}^{q+1} > D_{i,m}$ in which case the task is unschedulable in that execution mode. (The task is schedulable if all of its execution modes are schedulable).

Consider the set of tasks defined in Table I, where τ_A has the highest priority. Table II illustrates how the fixed point iteration of (5) progresses from an initial value of $C_B = 270$, showing the total number of mode *x* and mode *y* jobs of task τ_A that are included in the interference term.

TABLE I: TASK PARAMETERS

Task	Mode	$C_{i,m}$	$T_{i,m}$	$D_{i,m}$
τ_A	x	20	90	45
	y	50	200	100
τ_B		270	500	400

TABLE II: FIXED POINT ITERATION

Iteration (q)	w^q	$k_{A,x}$	$k_{A,y}$	$I_A(w^q)$	w^{q+1}
0	270	0	2	100	370
1	370	4	1	130	400
2	400	2	2	140	410
3	410	0	3	150	420
4	420	0	3	150	420

TABLE III: RESPONSE TIMES VERSUS PATTERNS OF EXECUTION MODES

Pattern of task τ_A jobs by mode with response time R_B			
y, y	370	x, y, x	360
y, x, y	390	x, x, y	360
y, x, x	360	x, x, x, y	380
x, y, y	390	x, x, x, x	350

This example serves to show that the response time computed via (3), (4) and (5) is an upper bound, rather than an exact value. This is because the combination of jobs that give the maximum interference for a specific window length w^q may be different from the combination required to give the maximum interference for the subsequent window of length w^{q+1}.

Table III gives the response time of task τ_B for all of the distinct scenarios in terms of the sequence of mode *x* and mode *y* jobs of task τ_A that can occur up to the completion of τ_B. (Note we do not include sub-sequences such as *x,x* which omit a further job that could execute within the response time). The exact worst-case response time is 390 rather than 420 as computed by the sufficient test. The pessimism in the test can be seen in the progression from iterations 0 to 1 in Table II where two distinct combinations of jobs are required to maximise interference with an invalid transition between them. By an invalid transition, we mean that the scenario cannot progress from having some number of jobs of a given mode to subsequently having fewer jobs of that mode, for example two mode *y* jobs, and then only one. We note that obtaining the exact worst-case response time involves examining sequences for different combinations of jobs in different execution modes. With multiple higher priority VRB tasks, these combinations extend to the jobs of all higher priority tasks, making the problem intractable.

E. Linear upper bounds on the interference

In this section, we provide a sufficient schedulability test for VRB tasks using linear bounds, similar to the ones derived in [14]. These tests are potentially less precise than the analysis given earlier, but require much less computation.

A simple upper bound on the maximum amount of interference due to a VRB task τ_j that could be released in an interval of length *w* can be derived from Theorem 1. Here, we assume that the interval is filled by an integer number of periods of jobs with the maximum utilisation $U_j^{\max}$ and then a job with maximum execution time $C_j^{\max}$ is released at the end of the interval, thus:

$$I_j^{UB}(w) = wU_j^{\max} + C_j^{\max} \quad (6)$$

This upper bound can be improved upon using the techniques described in [14] as follows: The solid line in Figure 4 depicts the processing time that could be used by task τ_j against time for some sequence of jobs of task τ_j that result in the maximum amount of execution time strictly *within* an interval of length *w*, assuming that task τ_j is the only task in the system. Let $P1(t1, y1)$ be the last minima on this line such that $t1 < w$. As $P1$ is the last minima, then it follows that the maximum processing time in an interval of length w is achieved when a job of task τ_j with the maximum execution time is released at time $t1$. Hence we define: $t2 = t1 + C_j^{\max}$ and $y2 = y1 + C_j^{\max}$. Further, $t1$ is the sum of an integer multiple (e.g. $k_{j,1}$, $k_{j,2}$ etc.) of each of the minimum inter-arrival times for the different execution modes of task τ_j (i.e. $t1 = k_{j,1}T_{j,1} + k_{j,2}T_{j,2} + \ldots$). Similarly, $y1$ is the sum of the same set of integer multiples of each of the execution times for the different execution modes of task τ_j (i.e. $y1 = k_{j,1}C_{j,1} + k_{j,2}C_{j,2} + \ldots$), and therefore $y1 \le U_j^{\max} t1$. A valid upper bound $I_j^{UB}(w)$ on the execution time of task τ_j in an interval of length w (depicted by the dashed line in Figure 4) is thus given by:

$$I_j^{UB}(w) = wU_j^{\max} + C_j^{\max}(1 - U_j^{\max}) \quad (7)$$

Since this upper bound is guaranteed to be no smaller than the actual amount of execution of τ_j in the interval, and the actual amount is a value in discrete time units, then we can convert the upper bound to discrete time units using the floor function:

$$I_j^{UB*}(w) = \lfloor I_j^{UB}(w) \rfloor \quad (8)$$

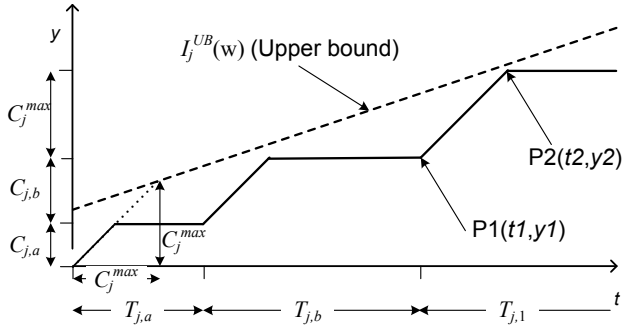

Figure 4: Upper bound on interference within an interval.

We note that this upper bound covers the amount of execution of task τ_j that could actually take place *within* an interval of length *w*, rather than the amount of execution that could be *released* in such an interval. Such a bound can however be used in the fixed point iteration (5). This follows from the fact that (8) is a monotonically non-decreasing function of *w*, and the worst-case response time for any mode *m* job of task τ_i is no larger than the smallest value of *w* that is large enough to accommodate the execution time of that job, and all the execution of higher priority tasks that could possibly occur within the same interval. We note that using (7) & (8) may in some circumstances cause the fixed point iteration to converge relatively slowly.

We now briefly return to the example used in the previous section – see Table I. Substituting the values for task τ_A into (7) & (8), we have $I_A^{UB*}(w) = \lfloor 0.25w + 37.5 \rfloor$ which results in an upper bound for the response time of task τ_B of $R_B = 409$. We note that in this case, this linear upper bound on interference gives a less pessimistic response time than using the ILP schedulability test. As it is trivial to construct tasksets that are deemed schedulable using the ILP test, but not when using the linear upper bound, then these two schedulability tests are *incomparable*.

Using the simple linear upper bound given by (6) & (8), we have $I_A^{UB*}(w) = \lfloor 0.25w + 50 \rfloor$ which results in an upper bound response time of $R_B = 426$. The schedulability test based on the upper bound given in (7) *dominates* the test using the simpler upper bound given in (6), this is because (7) always provides a value that is no larger than that given by (6).

IV. Improved Schedulability Analysis

In this section, we revisit the physical system that motivates our scheduling problem. We make use of limitations on the maximum rate of acceleration and deceleration of the engine to constrain the possible transitions between execution modes, and hence provide a refined analysis. We note that this refined analysis is applicable to VRB tasks that are driven from the same angular source (e.g. engine speed). Further, we also deal with a problem caused by lag in the measurement of engine speed and hence the selection of execution mode.

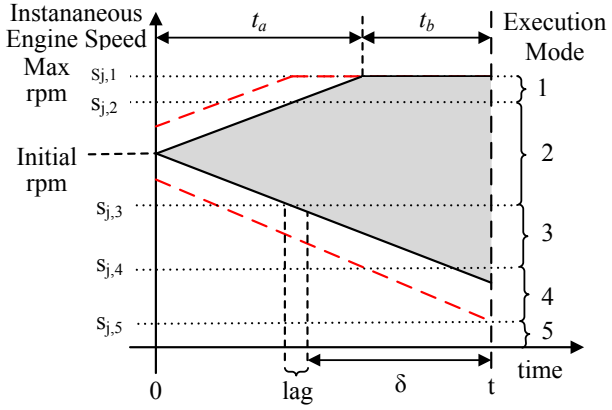

Figure 5: Engine speed envelope.

For production car engines, the maximum rate of acceleration in engine speed[1] is around 10,000rpm / sec. Note this is when the transmission is decoupled (out-of-gear) and the engine is only accelerating its own internal masses e.g. during a gear change, or 'blipping' the throttle. Hence in time intervals of interest, which are typically in the range of 10ms to 100ms, the change in rpm is limited to approx. ± 100rpm to ± 1000rpm which is significantly less than the full operating range of the engine (for a petrol engine this is typically around 700rpm to 7000rpm). Figure 5 illustrates this, showing the possible progression in engine speed against time, starting at some arbitrary initial rpm. Only the rpm envelope (shaded area) is feasible within a time *t*. (Figure 5 is used to illustrate a number of aspects of the analysis developed in this section. The annotations 'lag', t_a, t_b, δ and the red dashed lines are explained in the later subsections that refer to them)..

A. Engine speed measurement and lag

Recall that VRB tasks are typically released and have deadlines corresponding to specific angular positions of the engine. Further, the execution mode of a job is determined by its inter-arrival time. In practice, this is given by the average engine speed (effectively the time interval) between the previous release and the current one. This means that under acceleration, the speed measured by taking the elapsed time between releases lags behind the instantaneous speed, and this needs to be accounted for. This issue is illustrated in Figure 6. Here, job *J* executes in mode *m* in the second time interval, since the average engine speed between the previous and the current release corresponds to that mode. However, due to acceleration, the interval that job *J* actually

[1] The V10 engine in the Lexus LFA is able to go from idle to its redline (9000rpm) in 0.6 sec which is claimed to be too fast for an analogue tachometer to track accurately [30]. This equates to acceleration in engine speed of around 13,000rpm/sec.

executes in, and hence the time to the next release, can be shorter than the minimum period $T_{j,m}$ associated with mode m. In this case the subsequent job does not execute in mode m.

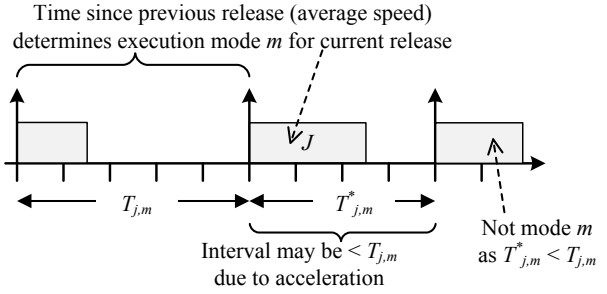

Figure 6: Lag in determining the execution mode

Let $S_{j,m}$ be the engine speed corresponding to the shortest period $T_{j,m}$ for execution mode m. We assume a direct mapping from $S_{j,m}$ to $T_{j,m}$ of the form $T_{j,m} = \beta_j / S_{j,m}$ where β_j is a scaling factor reflecting the task period in terms of angle of rotation divided by 360°, hence β_j = 1/2 if the task period is 180°. For convenience, we assume that S is measured in revolutions per second, T in seconds, and engine acceleration in revolutions per second per second and thus avoid the need for any unit conversions.

Consider the scenario where the engine is accelerating at its maximum rate over two intervals corresponding to mode m execution, as depicted in Figure 7. The average speed in the first interval is $S_{j,m}$ (as the interval is of length $T_{j,m}$) and so the execution mode in the second interval is m. However, the average speed in the second interval is some higher value $S^*_{j,m} > S_{j,m}$ (equivalent to an interval of length $T^*_{j,m}$).

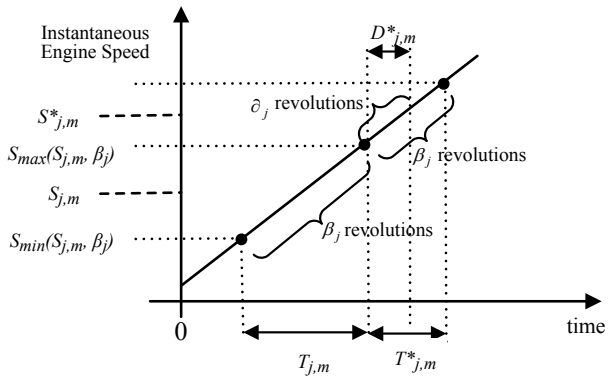

Figure 7: Changes in engine speed over two invocations.

Let $S_{\max}(S_{j,m},\beta_j)$ be the maximum instantaneous speed in an interval of β_j revolutions which has an average speed of $S_{j,m}$ (the maximum for mode m execution), and similarly, $S_{\min}(S_{j,m},\beta_j)$ be the minimum speed in such an interval. Applying the simple physics equation linking (angular) acceleration, speed, distance, and time, the time period $T_{j,m}$ for β_j revolutions under maximum acceleration α, starting at speed $S_{\min}(S_{j,m},\beta_j)$ is given by the solution to:

$$\beta_j = \alpha T_{j,m}^2/2 + S_{\min}(S_{j,m},\beta_j).T_{j,m} \quad (9)$$

hence:

$$T_{j,m} = \frac{-S_{\min}(S_{j,m},\beta_j) + \sqrt{(S_{\min}(S_{j,m},\beta_j))^2 + 2\alpha\beta_j}}{\alpha} \quad (10)$$

Re-arranging (9) to make $S_{\min}(S_{j,m},\beta_j)$ the subject of the equation, and using $S_{j,m} = \beta_j / T_{j,m}$ as the average speed, we have:

$$S_{\min}(S_{j,m},\beta_j) = \frac{\beta_j}{T_{j,m}} - \frac{\alpha T_{j,m}}{2} = S_{j,m} - \frac{\alpha\beta_j}{2S_{j,m}} \quad (11)$$

Noting that $S_{\max}(S_{j,m},\beta_j) = S_{\min}(S_{j,m},\beta_j) + \alpha T_{j,m}$:

$$S_{\max}(S_{j,m},\beta_j) = \frac{\beta_j}{T_{j,m}} + \frac{\alpha T_{j,m}}{2} = S_{j,m} + \frac{\alpha\beta_j}{2S_{j,m}} \quad (12)$$

From (11) and (12) we observe that $(S_{\max}(S_{j,m},\beta_j))^2 - (S_{\min}(S_{j,m},\beta_j))^2 = 2\alpha\beta$.

As the maximum (final) speed for one interval is also the minimum (initial) speed for the next interval, we may compute $T^*_{j,m}$ from $S_{\max}(S_{j,m},\beta_j)$ by substituting the latter into (10) in place of $S_{\min}(S_{j,m},\beta_j)$, thus obtaining:

$$T^*_{j,m} = \frac{-S_{\max}(S_{j,m},\beta_j) + \sqrt{(S_{\max}(S_{j,m},\beta_j))^2 + 2\alpha\beta_j}}{\alpha} \quad (13)$$

If the deadline is implicit (equal to the angular period), then $T^*_{j,m}$ determines the minimum effective deadline for mode m execution which is $D^*_{i,m} = T^*_{i,m}$. If the deadline is a smaller angular interval ∂_j than the angular period, then (13) may be used to compute $D^*_{i,m}$ by substituting ∂_j in place of β_j in the $2\alpha\beta_j$ term – i.e. computing the deadline assuming maximum acceleration for an angular rotation of ∂_j starting from an engine speed of $S_{\max}(S_{j,m},\beta_j)$.

We note that to account for the lag in determining the actual execution mode, the analysis given in Section III can simply utilise $D^*_{i,m}$ in place of $D_{i,m}$. We do not; however, need to substitute $T^*_{j,m}$ in place of $T_{j,m}$ as the period for jobs in execution mode m, since the next job that follows an interval of length $T^*_{j,m}$ must necessarily be in a different execution mode (e.g. mode $m-1$).

B. Constraints from the maximum rate of acceleration

For a given initial instantaneous engine speed S we can use the maximum rate of acceleration to calculate a set of constraints on the maximum number of jobs of each execution mode of task τ_j that can feasibly occur in a time interval of length t.

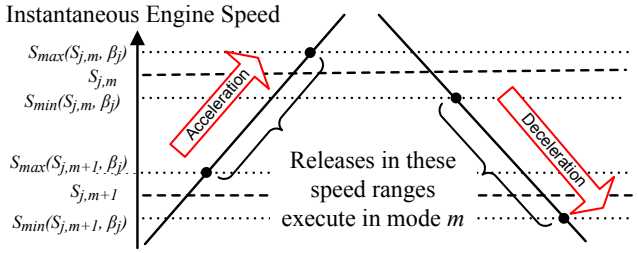

Figure 8: Engine speed range for mode m execution

Recall that a job of task τ_j executes in mode m provided that the average engine speed between the previous and the current release (β_j revolutions) is in the range $(S_{j,m+1}, S_{j,m}]$. Thus the instantaneous engine speed S at the release of a job can be in the range $(S_{\min}(S_{j,m+1},\beta_j), S_{\max}(S_{j,m},\beta_j)]$ and still correspond to mode m execution, since the average speed over the previous β_j revolutions can be in the appropriate range. For values of S outside of this range, then the engine speed needs to either accelerate or decelerate into the range for mode m execution first, and so under maximum acceleration, must reach $S_{\max}(S_{j,m+1},\beta_j)$ from below or $S_{\min}(S_{j,m},\beta_j)$ from above before mode m execution can begin (see Figure 8). Hence, the maximum time $\delta_{j,m}(S,t)$ spent in mode m, in an interval of length t, starting at an instantaneous engine speed S, with a maximum rate of acceleration or deceleration of α is given by:

$$\delta_{j,m}(S,t) = \begin{cases} t & S_{\min}(S_{j,m+1},\beta_j) < S \le S_{\max}(S_{j,m},\beta_j) \\ \max\left(0, t - \dfrac{S - S_{\min}(S_{j,m},\beta_j)}{\alpha}\right) & S > S_{\max}(S_{j,m},\beta_j) \\ \max\left(0, t - \dfrac{S_{\max}(S_{j,m+1},\beta_j) - S}{\alpha}\right) & S \le S_{\min}(S_{j,m+1},\beta_j) \end{cases} \quad (14)$$

This is illustrated in Figure 5, starting at the initial speed S, and considering an interval of length t, no time can be spent in mode 5, at most time δ can be spent in mode 3 (notice the small lag in actually entering mode 3), and at most time t in mode 2.

From (14), we obtain the following simple constraint (upper bound) on the maximum number of jobs of mode m that can be released in an interval of length t starting at engine speed S.

$$A_{j,m}(S,t) = \left\lceil \frac{\delta_{j,m}(S,t)}{T_{j,m}} \right\rceil \quad (15)$$

In the case that $S \le S_{\max}(S_{j,m},\beta_j)$ then a more precise bound is possible via a consideration of the maximum number of engine revolutions in time $\delta_{j,m}(S,t)$. We upper bound the number of revolutions by considering a time interval assuming the maximum acceleration from the entry speed S^e into mode m where:

$$S^e = \begin{cases} S_{\max}(S_{j,m+1},\beta_j) & S \le S_{\min}(S_{j,m+1},\beta_j) \\ S & S > S_{\min}(S_{j,m+1},\beta_j) \end{cases} \quad (16)$$

up to speed $S_{\max}(S_{j,m},\beta_j)$, which is the maximum instantaneous speed for a release in mode m; with any remaining time spent at the maximum average speed $S_{j,m}$ for execution in mode m.

$$\rho_{j,m}(S,t) = \frac{\alpha t_a^2}{2} + S^e.t_a + S_{j,m}.t_b \quad (17)$$

where $t_a = \min(\delta_{j,m}(S,t), (S_{\max}(S_{j,m},\beta_j) - S^e)/\alpha)$ is the time spent accelerating up to speed $S_{\max}(S_{j,m},\beta_j)$ from the entry speed S^e, and $t_b = \delta_{j,m}(S,t) - t_a$ is the remaining time spent at the maximum average speed $S_{j,m}$.

An upper bound on the number of mode m jobs released in $\rho_{j,m}(S,t)$ revolutions of the engine is then given by:

$$A_{j,m}(S,t) = \left\lceil \frac{\rho_{j,m}(S,t)}{\beta_j} \right\rceil \quad (18)$$

We may also obtain an additional constraint $A_j^{\max}(S,t)$ on the total number of jobs of task τ_j of all execution modes released in an interval of length t, starting at an instantaneous speed S, based on the maximum number of engine revolutions $\rho(S,t)$ in that interval.

$$\mathrm{A}_{\mathrm{j}}^{\max}(S,t) = \left\lceil \frac{\rho(S,t)}{\beta_j} \right\rceil \quad (19)$$

where

$$\rho(S,t) = \frac{\alpha t_a^2}{2} + S.t_a + V^{\max}.t_b \quad (20)$$

Here, $t_a = \min(t, (V^{\max} - S)/\alpha)$ is the part of the interval t during which the engine speed is accelerating at its maximum rate from speed S to the maximum permitted engine speed $V^{\max}$, and $t_b = t - t_a$ is the remaining time during which speed $V^{\max}$ is sustained. (Note $V^{\max}$ corresponds to the maximum engine speed permitted by the rev limiter used to prevent damage to the engine).

Finally, we note that it is typically not possible for jobs to be released in non-adjacent modes without some time being spent in the intervening mode. For example, in Figure 5 it is not possible for jobs to be released in both mode 2 and mode 4, without crossing the range of engine speeds corresponding to mode 3.

When crossing the range of mode m speeds under maximum acceleration, job releases at instantaneous speeds exceeding $x = S_{\max}(S_{j,m+1},\beta_j)$ must necessarily be in mode m, until an instantaneous speed of $y = S_{\max}(S_{j,m},\beta_j)$ is reached. The minimum time taken between these two speeds is $t_c = (y - x)/\alpha$ and hence the minimum number of engine revolutions $\rho_{j,m}$ required is given by:

$$\rho_{j,m} = \frac{\alpha t_c^2}{2} + x.t_c = \frac{(y^2 - x^2)}{2\alpha} \quad (21)$$

Alternatively, crossing the range of mode m speeds under deceleration, job releases at instantaneous speeds below $y = S_{\min}(S_{j,m},\beta_j)$ must necessarily be in mode m, until an instantaneous speed of $x = S_{\min}(S_{j,m+1},\beta_j)$ is reached. The minimum number of engine revolutions $\rho_{j,m}$ required between these two speeds is given by substituting these values for x and y into (21). We note that the minimum number of engine revolutions obtained for acceleration and deceleration are the same, since $(S_{\max}(S_{j,m+1},\beta_j))^2 - (S_{\min}(S_{j,m+1},\beta_j))^2 = (S_{\max}(S_{j,m},\beta_j))^2 - (S_{\min}(S_{j,m},\beta_j))^2 = 2\alpha\beta$.

Recall that $k_{j,m}$ is used to denote the number of jobs of task τ_j. Hence if there are jobs released in modes $m-1$,

and $m+1$, then there must also be a minimum number of jobs released in mode *m*:

$$(k_{j,m-1} > 0) \wedge (k_{j,m+1} > 0) \Rightarrow k_{j,m} \geq \left\lfloor \frac{\rho_{j,m}}{\beta_j} \right\rfloor \quad (22)$$

This minimum may be zero if the speed range of the mode is sufficiently small to be skipped over entirely between job releases.

We can limit the maximum interference $I_j(S,t)$, from jobs of task τ_j, released in an interval of length *t*, starting at speed *S* by constraining (i) the maximum number of jobs of mode *m* released in the interval via (15) and (18), (ii) the total number of jobs of any mode released in the interval via (19), and (iii) the minimum number of jobs of a mode where there are jobs released in adjacent modes via (22). We use these constraints in an ILP formulation.

ILP Problem: Maximise:

$$I_j(S,t) = \sum_{\forall x} k_{j,x} C_{j,x} \quad (23)$$

Subject to the constraints that:

$$\forall x \quad 0 \leq k_{j,x} \leq A_{j,x}(S,t)$$

$$\sum_{\forall x} k_{j,x} \leq \mathrm{A}_{\mathrm{j}}^{\max}(S,t)$$

$$1 < x < M_j \quad (k_{j,x-1} \leq 0) \vee (k_{j,x+1} \leq 0) \vee \left(k_{j,x} \geq \left\lfloor \frac{\rho_{j,x}}{\beta_j} \right\rfloor \right)$$

$$\sum_{\forall x} k_{j,x} T_{j,x} \leq t + T_{j,Z} - 1 \quad (24)$$

Where *Z* is the mode with the largest execution time of any mode with $k_{j,x} > 0$, and $T_{j,Z}$ is the minimum period for execution in that mode *Z*.

Note IBM CPLEX, the ILP solver we used, handles the logical OR operations in (24).

C. Schedulability analysis

We now make use of (23) and (24) to determine the schedulability of a set of VRB tasks. We do this by effectively checking schedulability for all possible values of the initial engine speed and the subsequent envelope of feasible engine speed trajectories over time.

Our overall approach is summarised by the pseudo code in Figure 9 which provides a schedulability test for all *m* execution modes of task τ_i. The worst-case response time $R_{i,m}(S)$ of a job of task τ_i in execution mode *m* for a starting engine speed *S*, can be computed via the following fixed point iteration. On each iteration, (23) and (24) are used to maximise the interference from higher priority tasks.

$$w_{i,m}^{q+1}(S) = B_{i,m} + C_{i,m} + \sum_{\forall j \in hp(i)} I_j(S, w_{i,m}^{q}) \quad (25)$$

Iteration starts with an initial value $w_{i,m}^{0}$, typically $w_{i,m}^{0} = C_{i,m}$, and ends when either $w_{i,m}^{q+1}(S) = w_{i,m}^{q}(S)$ in which case $R_{i,m}(S)$, is given by $w_{i,m}^{q+1}(S)$, or when $w_{i,m}^{q+1}(S) > D_{i,m}^{*}$ in which case the task is unschedulable. Recall that $D_{i,m}^{*}$ corresponds to the shortest possible deadline for execution in mode *m*.

```
for all modes of task τ_i initialise R_{i,m} = 0

for each instantaneous engine speed S {
    for each mode m corresponding to S {
        Compute the worst-case response time
        R_{i,m}(S) of a mode m job of task τ_i starting
        at an initial engine speed S.
        R_{i,m} = max(R_{i,m}(S), R_{i,m})
        if( R_{i,m} > D*_{i,m} ) {
            return unschedulable
        }
    }
}
return schedulable
```

Figure 9: Response time calculation for a single task

As engine speed is a continuous variable we need to address the issue of a potentially infinite number of initial engine speeds. This is achieved by considering quantised values of *S* (e.g. every Q = 100rpm) taken to represent a range of values from $S^- = S - Q/2$ to $S^+ = S + Q/2$. To accommodate this approximation, we use both S^+ and S^- to determine the constraints (in (15), (18) and (19)) and then utilise the more relaxed constraint of each pair. Further, we consider a speed *S* to corresponds to a particular mode *m* if a job could be released in the mode within the speed range $[S^-, S^+]$, i.e. if $S_{\min}(S_{j,m+1}, \beta_j) < S^+$ and $S_{\max}(S_{j,m}, \beta_j) \geq S^-$. In this way, the envelope of possible modes is increased as shown by the dashed lines in Figure 5.

V. Experimental Evaluation

In this section, we evaluate the performance of five sufficient schedulability tests for VRB tasks:

- RTA-SP: obtained by reducing each VRB task to the sporadic task model by assuming the maximum execution time $C_i^{\max}$, minimum period $T_i^{\min}$ and deadline $D_i^{\min}$ across all modes.
- VRB-L1 and VRB-L2: using the linear upper bounds given by (6) & (8) and (7) & (8) respectively.
- VRB-ILP: using the ILP formulation to compute the maximum interference via (3) and (4) when arbitrary sequences of execution modes are permitted.
- VRB-ILP-CON: using the ILP formulation with additional constraints from the physical system to compute the maximum interference via (23) and (24).

In addition, we also evaluate two *necessary* upper bounds on taskset schedulability.

- UB-N: forms an upper bound on exact schedulability when arbitrary sequences of execution modes are permitted. It considers the interference from each higher priority VRB task as being entirely due to one mode or another, whichever mode results in the most interference when modelled as a simple periodic behaviour.

- UB-NX: forms an upper bound on exact schedulability taking account of constraints from the physical system. It is computed in a similar way to UB-N. UB-NX considers the interference from each higher priority VRB task as being entirely due to one mode or another, whichever mode results in the most interference when modelled as a simple periodic behaviour. However, the number of jobs that may be released in a specific execution mode is constrained using (15) and (18).

Both UB-N and UB-NX are necessary, but not sufficient test for schedulability (assuming no dependency between the modes of different VRB tasks). UB-N upper bounds schedulability according to RTA-SP, VRB-L1, VRB-L2, and VRB-ILP, while UB-NX upper bounds schedulability according to VRB-ILP-CON.

A. *Parameter generation*

The parameters for the sporadic tasks used in our experiments were randomly generated as follows:

- The UUniFast algorithm [10] was used to generate a set of n utilisation values U_i, with a total utilisation of U.
- Task periods were generated according to a log-uniform distribution[2]. Here the ratio between the maximum and the minimum permissible task period was given by 10^r. By default, this range was 100, i.e. $r = 2$.
- Task execution times were set based on the utilisation and period selected: $C_i = U_i T_i$.
- Task deadlines were either *implicit*, and so equal to their periods, or *constrained* and chosen at random according to a uniform distribution in the range $[C_i + x(T_i - C_i), T_i]$, with $x = 0.5$ as the default.
- The default taskset cardinality was 10.

A fixed proportion p (default p = 50%) of the sporadic tasks were then converted to VRB tasks as follows:

- There were 5 execution modes.
- The existing sporadic task triplet (C_i, T_i, D_i) was assigned as the mode 1 parameters.
- A scaling factor (default f=1.5) was used to determine the parameters of the other modes, via: $C_{i,m+1} = fC_{i,m}$, $T_{i,m+1} = fT_{i,m}$,. (Note, with 5 modes, f=1.5 equates to a ratio of 7.6 between the largest period and the smallest period for jobs of the same mode; equivalent to a range of engine speeds from say 1000rpm to 7600rpm).
- A mode was randomly chosen to have the largest utilisation. The execution times of the remaining modes were adjusted by multiplying them by uniform random values in the range [1-e, 1], default $e = 0.25$.
- The deadline for each mode was based on $T^*_{i,m}$. In the rare cases that $T^*_{i,m} < C_{i,m}$ the taskset was discarded as invalid, otherwise the deadline was given by $D^*_{i,m} = T^*_{i,m}$ (implicit deadline), or chosen at random in the range $[C_{i,m} + x(T^*_{i,m} - C_{i,m}), T^*_{i,m}]$ (constrained deadline).
- The factor β_j was chosen to be $T_{i,1}/T_{j,1}$ where $T_{j,1}$ is the longest period of the first mode of any VRB task. This ensures that all VRB tasks share the same maximum engine speed.
- The maximum acceleration α was chosen so that 36 revolutions were required when crossing the whole range of possible speeds from 0 to $V_{\max} = \beta_i / T_{i,1}$. $\alpha = (V^2_{\max} - 0^2)/(2*36)$. (Note, an engine with maximum acceleration of 10,000 rpm per second does approx. 36 revolutions in going from 1000 rpm to 6700 rpm. The Lexus LFA V10 engine does 50 revolutions when going from idle, around 1000 rpm, to its redline, 9000 rpm, in 0.6 sec [33]).

For schedulability test VRB-ILP-CON the analysis was performed for a series of 10 instantaneous speeds.

In each experiment, the taskset utilisation was varied from 0.05 to 0.95 in steps of 0.05. For each utilisation value, synthetic tasksets were generated and their schedulability determined according to the various schedulability tests. Priority assignment was according to Audsley's Optimal Priority Assignment (OPA) algorithm [1], [2], since the schedulability tests given for VRB tasks in Section III (VRB-L1, VRB-L2, VRB-ILP) comply with the three conditions required for OPA-compatibility [15], [16], as does the constrained ILP schedulability test given in Section IV (VRB-ILP-CON) when blocking is not considered.

B. *Success Ratio*

In our first experiment, we compared the performance of the schedulability tests via a metric referred to as the *success ratio*; the proportion of randomly generated tasksets that are schedulable in each case. In this experiment, 1000 taskset were used for each utilisation level.

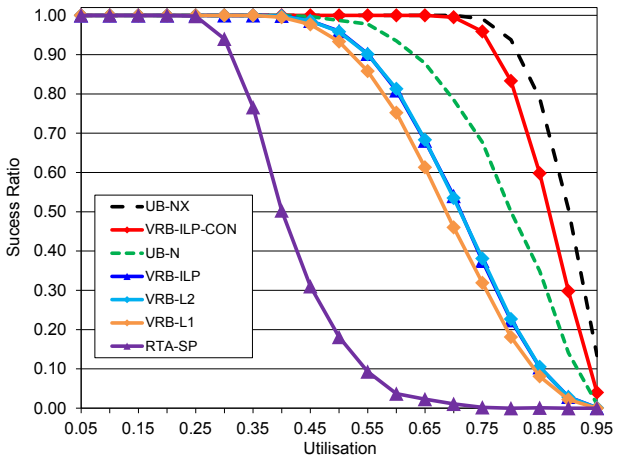

Figure 10: Success ratio for n = 10, D = T

Figure 10 shows that the linear upper bounds (VRB-L1 & VRB-L2) and the ILP formulation (VRB-ILP) introduced in Section III significantly improve upon the default approach (RTA-SP) of treating VRB tasks as if they were sporadic tasks and assuming worst-case parameters. Further, using the physical constraints of the system to limit the interference considered (VRB-ILP-CON) results in

[2] The log-uniform distribution of a variable x is such that $ln(x)$ has a uniform distribution.

substantial further improvements in schedulability, obtaining results close to the upper bound (UB-NX).

C. Weighted Schedulability Measure

In our second set of experiments we compared how the overall performance of each of the schedulability tests varies with respect to changes in various parameters via the *weighted schedulability measure* [8].

We examined four different parameters: (i) the taskset size, (ii) the scaling factor *f* between VRB modes, (iii) the variability in the utilisation of execution time modes *e*, and (iv) the proportion of VRB tasks. Due to constraints on space, these results are presented in the appendix of [17].

D. Runtime

The experiments were run on a compute server with two Intel Xeon E5649 CPUs (each with 6 cores and 2x hyper-threading[3]) max clock speed 2.53Ghz. The elapsed times for the complete experiments (all schedulability tests) are given in Table IV. The UB-NX necessary test and a sufficient test based on VRB-L1 (using information about reachable modes) where used to avoid running the IBM CPLEX ILP solver for tasksets that could be proven schedulable or unschedulable by simpler tests.

TABLE IV: RUNTIME

Experiment	Elapsed time	Tasksets	Time per task set
Success ratio	53s	19000	2.8ms
WS: taskset size	6m58s	38000	11ms
WS: VRB scaling factor	38s	19000	2.0ms
WS: WCET variability	7m14s	188100	2.3ms
WS: proportion VRB tasks	1m13s	19000	3.8ms

We observe that the average elapsed time per taskset for the default configuration was less than 3ms, which increased with larger tasksets (to 11ms) and a higher proportion of VRB tasks (to 3.8ms). These runtimes show that the methods are entirely viable for realistic problems, since typical automotive systems have a relatively small number of periodic tasks and just a few VRB tasks.

VI. SUMMARY AND CONCLUSIONS

In this paper, we addressed an interesting scheduling problem posed by the keynote speaker at ECRTS 2012 [12]. We introduced effective schedulability analysis for tasks that are periodic in relation to engine revolutions rather than time, and thus execute with Variable Rate-dependent Behaviour (VRB). To avoid such tasks overloading the processor, system designers manipulate their functionality so that more complex algorithms are employed at low engine speeds where long WCETs are acceptable, and simpler, faster algorithms are used and functionality is shed at high rpm to avoid missing deadlines.

The major contribution of this paper is the introduction of simple schedulability tests for variable rate tasks scheduled under fixed priorities (as implemented in OSEK and AUTOSAR operating systems), and the refinement of these techniques using information about the physical constraints and limitations on the system, such as maximum rpm and maximum rate of acceleration and deceleration.

Our evaluation shows that even simple linear bounds specifically derived for VRB tasks significantly improve upon the default approach of modelling each VRB task as a sporadic task and taking the pessimistic approach of combining the minimum possible period with the maximum possible WCET. Further, utilising the physical constraints on the system to constrain the amount of interference considered provides a schedulability test that is substantially better still. This work provides industry with an effective means of analysing systems that contain tasks whose behaviour is dependent on angular stimuli such as inputs from crankshaft or camshaft sensors.

In practice, it is important not only to have effective and efficient schedulability tests able to determine if a system is schedulable, but also a means of determining when a system is unschedulable, what the reasons are for that (i.e. the scenario that leads to a potential deadline miss), and also an understanding of what design changes or optimisations will be needed to obtain a schedulable system.

The analyses developed in this paper address all of these requirements. When a system is deemed unschedulable, it is possible to obtain from the analysis, the scenario which lead to a response time that exceeds the relevant deadline (including information about the execution mode of the task under study, the number of interfering jobs of higher priority tasks and their modes, and in the case of the refined analysis (Section IV), the initial engine speed for the scenario. Further, our evaluation of the runtime of the analysis techniques, on tasksets of representative complexity, shows that the techniques are viable for use in a design-time analysis tool. Such a tool could be used to explore the sensitivity of system schedulability to changes in the execution times of the different modes of each task, the inter-arrival times denoting the different modes, and the processor speed.

We framed our analysis in terms of engine speed; however, it applies equally well to any cyber-physical system where VRB tasks are released periodically with respect to angular rotation. The simple forms of analysis presented in Section III are applicable to systems with VRB tasks driven from multiple independent angular sources (e.g. engine speed, wheel speed, gear speed etc.). By contrast, the more sophisticated analysis given in Section IV accounts for physical limitations on the rate of angular acceleration, but assumes that VRB tasks are driven from the same angular source (e.g. engine speed). It remains an open issue, how to adapt such sophisticated analysis to systems with multiple angular sources each driving multiple VRB tasks.

[3] To get the average runtime for analysing a single taskset *on its own* one needs to multiply the average times shown in the last column of Table IV by a factor of approx. 20 to account for the parallel execution in the compute server.

Although the schedulability analysis described in this paper is effective, there is an argument that as the deadline of a VRB task changes, then so should its priority, otherwise the system will necessarily suffer from priority inversion. This issue could be addressed either via the use of EDF scheduling, or by having a different priority for each mode of a VRB task. The latter approach is in keeping with the RTOS support for fixed priority scheduling available in automotive systems, and merits further investigation.

ACKNOWLEDGEMENTS

This work was partially funded by the UK EPSRC funded MCC project (EP/K011626/1). The authors would like to thank Darren Buttle of ETAS GmbH for proposing the open scheduling problem in his keynote talk at ECRTS 2012 that led to this work.

REFERENCES

[1] N.C. Audsley, "Optimal priority assignment and feasibility of static priority tasks with arbitrary start times", Technical Report YCS 164, Dept. Computer Science, University of York, UK, 1991.

[2] N.C. Audsley, "On priority assignment in fixed priority scheduling", Information Processing Letters, 79(1): 39-44, May 2001.

[3] N.C. Audsley, A. Burns., M. Richardson, A.J. Wellings, "Applying new Scheduling Theory to Static Priority Pre-emptive Scheduling". Software Engineering Journal, 8(5) pp. 284-292, 1993.

[4] T.P. Baker, "Stack-based Scheduling of Real-Time Processes." *Real-Time Systems Journal* (3)1, pages 67-100. 1991.

[5] S. Baruah, D. Chen, S. Gorinsky, A. Mok. Generalized multiframe tasks. Real-Time Systems: The International Journal of Time-Critical Computing, 17(1):5–22, July 1999.

[6] S. K. Baruah. "Dynamic- and static-priority scheduling of recurring real-time tasks". Real-Time Systems: The International Journal of Time-Critical Computing, 24(1):99–128, 2003.

[7] S.K. Baruah. The non-cyclic recurring real-time task model. In proceedings of the Real-Time Systems Symposium (RTSS), pp 173-182, 2010.

[8] A. Bastoni, B. Brandenburg, and J. Anderson, "Cache-Related Preemption and Migration Delays: Empirical Approximation and Impact on Schedulability," in *Proceedings of OSPERT*, pp. 33-44, Brussels, Belgium, 2010.

[9] V. Berten, J. Goossens. Sufficient FTP Schedulability Test for the Non-Cyclic Generalized Multiframe Task Model. In proceedings of the RTSS WiP Session. 2011

[10] E. Bini, G. Buttazzo. Measuring the performance of schedulability tests. Real-Time Syst., 30(1-2):129–154, 2005.

[11] G. Buttazzo, E. Bini, D. Buttle, "Rate-Adaptive Tasks: Model, Analysis, and Design Issues", In proceedings of the International Conference on Design, Automation and Test in Europe (DATE 2014), March 2014.

[12] D. Buttle, "Real-Time in the Prime Time" Keynote talk at the Euromicro Conference on Real-Time Systems (ECRTS 2012). Presentation available from http://ecrts.eit.uni-kl.de/index.php?id=69.

[13] R.I. Davis, A. Zabos, A. Burns, "Efficient Exact Schedulability Tests for Fixed Priority Real-Time Systems". *IEEE Transactions on Computers*, (Vol. 57, No. 9) pp. 1261-1276, September 2008.

[14] R.I. Davis, A. Burns, "Response Time Upper Bounds for Fixed Priority Real-Time Systems". In proceedings of the Real-Time Systems Symposium (RTSS), 2008.

[15] R.I. Davis, A. Burns ""Priority Assignment for Global Fixed Priority Pre-emptive Scheduling in Multiprocessor Real-Time Systems". In proceedings of the Real-Time Systems Symposium (RTSS), pp. 398-409, 2009.

[16] R.I. Davis, A. Burns, "Improved Priority Assignment for Global Fixed Priority Pre-emptive Scheduling in Multiprocessor Real-Time Systems". Real-Time Systems, Volume 47, Issue 1, pp. 1-40, 2010.

[17] R.I. Davis, T. Feld, V. Pollex, F. Slomka, "Schedulability Tests for Tasks with Variable Rate-Dependent Behaviour under Fixed Priority Scheduling". University of York, Department of Computer Science Technical Report YCS-2014-488, Jan 2014.

[18] M. Garey, D. Johnson. "Computers and Intractability: A Guide to the Theory of NP-Completeness". W. H. Freeman & Co., NY, 1979.

[19] C.L. Liu, J.W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment", Journal of the ACM, 20(1) pp. 46-61, 1973.

[20] J.Y.-T. Leung, J. Whitehead, "On the complexity of fixed-priority scheduling of periodic real-time tasks". Performance Evaluation, 2(4), pp. 237-250, 1982.

[21] J. Kim, K. Lakshmanan, R. Rajkumar. " Rhythmic Tasks: A New Task Model with Continually Varying Periods for Cyber-Physical Systems". In Proceedings of the International Conference on Cyber-Physical Systems (ICCPS), pp. 55-64, 2012.

[22] A. Mok. Fundamental Design Problems of Distributed Systems for the Hard Real-Time Environment. PhD thesis, MIT Laboratory for Computer Science, May 1983.

[23] A.K. Mok, D. Chen D, "A multiframe model for real-time tasks". IEEE Trans Softw Eng 23(10):635–645, 1997.

[24] N. T. Moyo, E. Nicollet, F. Lafaye, and C. Moy. "On schedulability analysis of non-cyclic generalized multiframe tasks". In Proceedings EuroMicro Conference on Real-Time Systems (ECRTS), 2010.

[25] V. Pollex, T. Feld, F. Slomka,U. Margull, R. Mader, G. Wirrer, "Sufficient real-time analysis for an engine control unit " In proccedings of Real-Time Networks and Systems (RTNS), pp.247-254, 16-18 Oct 2013.

[26] V. Pollex, T. Feld, F. Slomka,U. Margull, R. Mader, G. Wirrer, "Sufficient real-time analysis for an engine control unit with constant angular velocities" In proccedings of Design, Automation & Test in Europe (DATE), pp.1335,1338, 18-22 March 2013

[27] J. Palencia, M. G. Harbour, "Schedulability analysis for tasks with static and dynamic offsets", In Proceedings of the Real-Time Systems Symposium, RTSS, pp. 26–37, 1998.

[28] M. Stigge,P. Ekberg, N. Guan; W. Yi; , "The Digraph Real-Time Task Model," Real-Time and Embedded Technology and Applications Symposium (RTAS), 2011 17th IEEE , vol., no., pp.71-80, 11-14 April 2011

[29] M. Stigge, P. Ekberg, N. Guan, W. Yi, "On the Tractability of Digraph-Based Task Models," In proceedings of the Euromicro Conference on Real-Time Systems pp.162,171, 5-8 July 2011

[30] M. Stigge , W.Yi, "Hardness Results for Static Priority Real-Time Scheduling". In proceedings of the Euromicro Conference on Real-Time Systems ECRTS 2012.

[31] M. Stigge, W. Yi, "Combinatorial Abstraction Refinement for Feasibility Analysis", In procedings of the Real-Time Systems Symposium (RTSS) pp 340-349, 2013.

[32] K.W. Tindell, A. Burns, A. Wellings, "Mode Changes in Priority Pre-emptively Scheduled Systems," In proceedings of the Real Time Systems Symposium, pp. 100-109, 1992.

[33] D. Vivien, "Lexus LFA Instruments" in Evo magazine, issue 190, page 202, 2013.

[34] A. Zuhily and A. Burns, Exact Scheduling Analysis of Non-Accumulatively Monotonic Multiframe Tasks, Real-Time Systems Journal, Vol 43, pp119-146, 2009.

APPENDIX: EXPERIMENTAL RESULTS: WEIGHTED SCHEDULABILITY

In this set of experiments we compared how the overall performance of each of the schedulability tests varies with respect to changes in a specific parameter via the *weighted schedulability measure* [8]. In these experiments, we used 100 tasksets per utilisation level, with utilisation levels varied from 0.05 to 0.95 in steps of 0.05, as before.

The first parameter examined was taskset size. Figure 11 shows how the weighted schedulability measure for each of the schedulability tests varies with increasing taskset size. With very small numbers of tasks, there is a significant probability that none of the tasks are VRB tasks or that there is just one VRB task and it has the lowest priority. Thus the schedulability tests all give similar results. As the number of tasks increases, the clear distinctions between schedulability test performance evident in the first experiment (Figure 10) are again apparent, and are largely unaffected by cardinality.

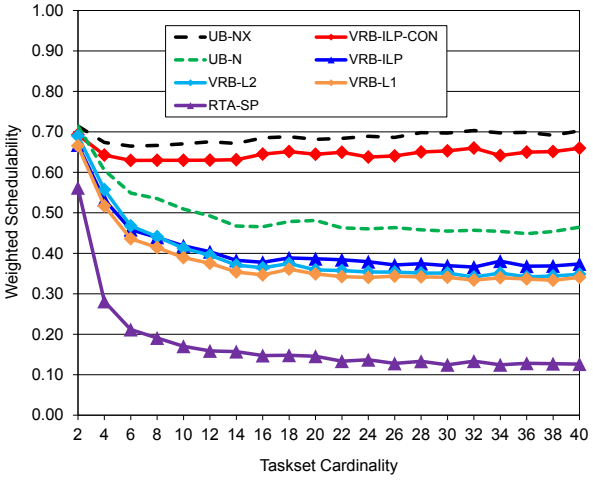

Figure 11: Weighted schedulability v. taskset size, D ≤ T

The second parameter examined was the scaling factor *f* representing the relationship between the parameters of adjacent VRB modes. Figure 12 shows how the weighted schedulability measure for each of the schedulability tests varies as this scaling factor is increased from 1.1 to 2. Here, all of the schedulability tests show a decrease in performance due to the increase priority inversion brought about by an increased range of deadlines for the VRB tasks. In addition, when interference is assumed to be possible from any arbitrary mode, then increases in the scaling factor rapidly increase interference. This is because the interference is at least the sum of the longest WCETs of any mode of each higher priority VRB task. This accounts for the rapid decline in performance of all the schedulability tests with the exception of VRB-ILP-CON.

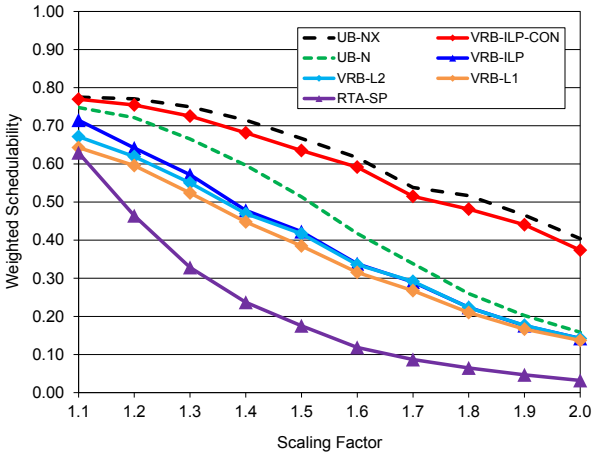

Figure 12: Weighted schedulability v. VRB scaling factor, D ≤ T

The third parameter examined was the variability in the utilisation of the execution modes of each VRB task. Figure 13 shows how the weighted schedulability measure for each of the schedulability tests varies as the value of *e* is varied from 0.05 to 0.95. With increasing values of *e*, WCETs decrease, hence the interference caused by higher priority tasks decreases, improving schedulability.

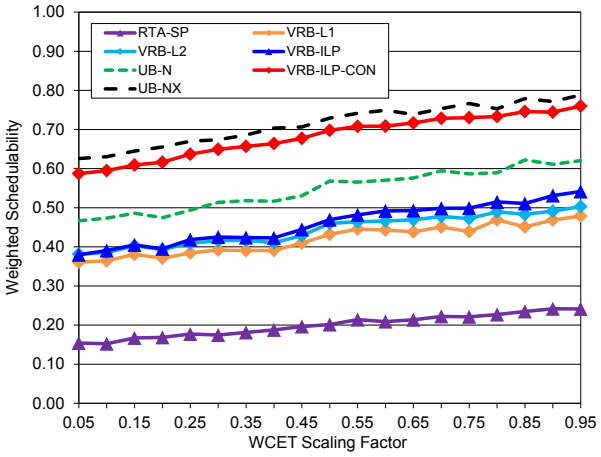

Figure 13: Weighted schedulability v. VRB task WCET variability, D ≤ T

The fourth parameter examined was the proportion of tasks that were VRB tasks. Figure 14 shows how the weighted schedulability measure for each of the schedulability tests varies as this proportion is increased from 10% to 100%. Here, schedulability decreases with an increasing number of VRB tasks. By replacing a sporadic task with a VRB task with several different modes, then assuming interference is possible from any arbitrary mode, then the interference in any given interval cannot decrease. Further a VRB task is itself less likely to be schedulable than the original sporadic task which decreases overall schedulability. It is notable that the more sophisticated VRB-

ILP-CON test is much less affected by the increasing proportion of VRB tasks.

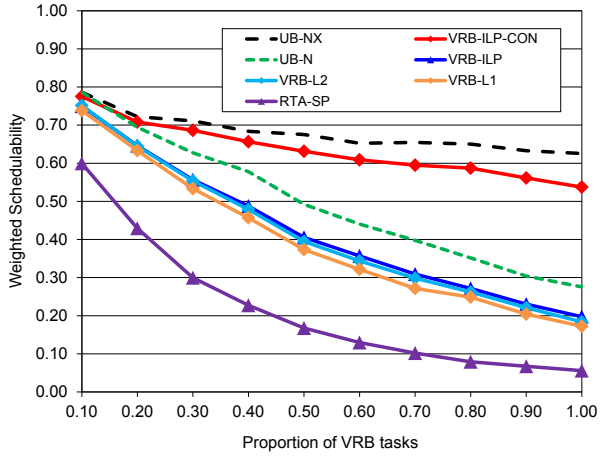

Figure 14: Weighted schedulability v. proportion of VRB tasks, $D \leq T$