Stability Properties of Adaptive Real-Time Feedback Scheduling: A Statistical Approach [1]

Tolga Ayav [‡] Giancarlo Ferrari-Trecate [†] Sinan Yılmaz [⊛]

[‡] İzmir Institute of Technology, Department of Computer Engineering
35430, Urla İzmir, Türkiye
e-mail: `TolgaAyav@iyte.edu.tr`

[†] INRIA, Domaine de Voluceau
Rocquencourt - B.P.105, 78153, Le Chesnay Cedex, France
e-mail: `Giancarlo.Ferrari-Trecate@inria.fr`

[⊛] Ege University, Department of Computer Engineering
35100, Bornova İzmir, Türkiye
e-mail: `Yilmaz@staff.ege.edu.tr`

Abstract

This paper focuses on the statistical analysis of an adaptive real-time feedback scheduling technique based on imprecise computation. We consider two-version tasks made of a mandatory and an optional part to be scheduled according to a feedback control rate-monotonic algorithm. A Proportional-Integral-Derivative (PID) control action provides the feedback strategy for deciding about the execution or rejection of the optional sub-tasks. By modelling the task execution times as random variables, we compute the probability density of the CPU utilization and derive conditions on PID parameters guaranteeing the stability of the overall system around a desired level of CPU utilization. This allows us to highlight the tasks statistics and the scheduling parameters that affect critically stability. The analysis is developed by first exploiting a number of simplifying assumptions that are progressively removed. The main results are also demonstrated through monte-carlo simulations of the scheduling algorithm.

[1] This work has been financially supported by the cooperation project between Izmir Institute of Technology and Institute Aéronautique et Spatial.

Contents

Keywords

Hard real-time systems, imprecise computation, adaptive scheduling, feedback control.

1. Introduction

Real-Time (RT) scheduling theory has traditionally focused on the development of algorithms for analyzing feasibility and for producing schedules at run-time. This traditional perspective of RT scheduling theory has well served the safety-critical embedded systems community. On the other hand, as a consequence of this behavior, RT systems are designed to have spare capacity under normal operation. There are several reasons why such a restricted perspective is proving to be increasingly inadequate for many application demands. Most real-time scheduling results assume that the workload and the real-time environment are relatively static and completely characterizable at the design stage. Run-time scheduling algorithms are typically designed to be correct only on feasible systems (i.e. systems for which it is indeed possible to always meet all deadlines); on infeasible systems, the performance of such algorithms may be unacceptably poor. A good example is the earliest-deadline-first algorithm, which is optimal on uniprocessors under non-overload conditions, but has been observed to perform miserably upon overload. According to [3], [4], [5], [13], the next generation of real-time systems will be more complex and capable of adaptivity as well as of meeting time constraints for mission and safety critical functions. A challenge of the current research is to make use of this spare capacity, in order to satisfy requirements for adaptivity in the system.

Adaptivity can be implemented by means of optional computations, which can be guaranteed at run-time by the use of flexible scheduling. Adaptive scheduling techniques, most of which are based on imprecise computation techniques, such as feedback control scheduling, have therefore been announced for systems with unpredictable characteristic [3], [4], [12], [13]. Although feedback control scheduling has been considered in many works so far, many questions are still open. The most important one is how to systematically tune the controller parameters in order to guarantee the stability of the overall system. In this work we consider a feedback control rate-monotonic scheduling (FC-RMS) system and present a stability analysis that highlights the main factors influencing the tuning of the controller parameters.

In section 2.1, imprecise computation models are briefly summarized. In particular we focus on two-version tasks made by a mandatory and an optional part. In section 2.2 and 2.3, the workload model and the FC-RMS are introduced. In FC-RMS, we consider the use of PID controllers for two reasons. First, their implementation is simple enough to guarantee acceptable overhead introduced by the computation of the control action. Second, as already remarked in [4], [5] and [13], a PID control structure is flexible enough to stabilize the system. On the other hand, in order to account for the unpredictability of task durations, we model them as random variables over which the scheduler has no control. The FC-RMS scheduler can only decide how many optional sub-tasks will be executed in a control period and this process will be modelled by the proper use of quantizer functions.

In section 3, we show how to characterize the statistical properties of all the signals in the control loop. In particular, the application of the Central Limit Theorem allows to model the CPU utilization as a gaussian random variable. In section 4, by neglecting the effect of the quantizer, (i.e. when the quantizer function is replaced by a linear one which corresponds assuming that fractions of optional sub-tasks can be executed) we show rigorously the relations linking task statistics to the choice of PID parameters guaranteeing stability. This is done by resorting first to the state-space representation of the closed-loop system and then by analyzing stability of the mean state. Finally, in section 5 we discuss the effect of the quantizer on the dynamics of the mean state. We highlight that, depending on the task statistics, two extreme behaviors can be noticed: pure quantization effect and no quantization effect. In such cases, by considering an integral control action we derive tight upper bounds to the mean variation, in terms of the sampling time, the quantization step and the task statistics. Once more, these relations provide useful insight for the tuning of the controller gain.

All the theoretical results presented in the paper are demonstrated and complemented by monte-carlo simulations of the FC-RMS scheme.

2. Feedback Control Rate-Monotonic Scheduling Architecture

In this section, we present Feedback Control RM scheduling, which integrates feedback control and rate-monotonic scheduling. This is done in two steps. First, we review imprecise computation models underlying feedback control scheduling. Then, the task model and workload model used in the experiments will be illustrated and finally the FC-RMS scheme will be introduced comprehensively.

2.1. Imprecise Computations Revisited

Undesirable effects of timing faults are tolerable as long as all the important tasks are completed in time. Therefore, rather than letting the operating system treat all tasks equally, the programmer may identify some tasks as mandatory, meaning that they must meet their deadlines, and some less important tasks as optional, meaning that these tasks can be skipped to prevent possible timing faults. The result produced by a task when it is entirely completed is the optimal result. If the task is terminated before completion, the intermediate result will be called imprecise. For instance, in radar signal processing, computation of a new estimate of the noise level in the received signal can be skipped and an old estimate can be used for one or a few sampling period(s) during overload. Imprecise computations include a number of implementation techniques such as milestone, sieve, and multiple-version that are presented thoroughly in [1] and [2]. In this work, a two-version implementation is considered. This means that each task is composed of a mandatory and an optional sub-task.

In order to ensure that imprecise computation works properly, all the mandatory sub-tasks should have bounded resource and limited processing time requirements. Moreover, mandatory sub-tasks must be allocated sufficient processor time so as to complete by their deadlines. Then, the system can use leftover processor time to complete as many optional sub-tasks as possible. For guaranteed performance, a conservative scheduling policy can be used to schedule mandatory sub-tasks. The scheduling of optional sub-tasks for optimal processor use calls for a more dynamic policy. This work proposes a rate-monotonic scheduler for mandatory sub-tasks and a feedback control scheduling technique for optional sub-tasks.

2.2. Task Model and Workload

In order to characterize imprecise computations, classical task model of real-time systems can be used with a few minor additions. We consider a set of tasks $Tasks = \{\tau_1, \tau_2, ..., \tau_q\}$ where each task is characterized by some parameters such as deadline d_i (this is considered as period T_i if the task is periodic), processing time C_i and priority p_i. Each task is logically decomposed into two sub-tasks, the mandatory part M_i and the optional part O_i. The processing times of M_i and O_i are $C_{i,m}$ and $C_{i,o}$, respectively. Thus, $C_{i,m} + C_{i,o} = C_i\,,\ i \in 1, \ldots, q$. The classical hard real-time model is a special case of imprecise computation model where $C_{i,o} \equiv 0$. Similarly, the soft real-time model is a special case where all tasks are completely optional, which means that $C_{i,m} \equiv 0$. We assume that all tasks are periodic and the execution times $C_{i,m}$ and $C_{i,o}$ are random variables with finite average and variance.

In all the experiments, we used the task parameters given in table 1.

q	15 (All periodic tasks)
$C_{i,m}$ and $C_{i,o}$	$U[150\mu s, 250\mu s]$
$T_{i\in\{1,\ldots,q\}}$	$\{15, 2, 14, 3, 13, 4, 12, 5, 11, 6, 10, 7, 9, 8, 8\}\ ms$
T	100 ms
K_p, K_i, K_d, IW, y_{ref}	0.1, 2, 0.1, 100, 0.7

Table 1: Workload model and system parameters used in the experiments. The uniform ditribution is denoted with U

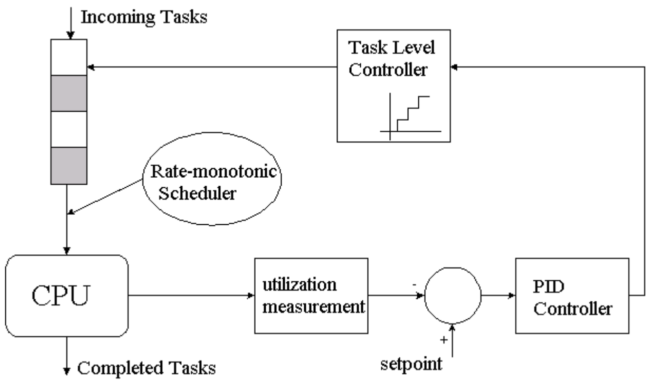

Figure 1: Feedback Control Rate-Monotonic Scheduling

2.3. Feedback Control Rate-Monotonic Scheduling

Rate-monotonic is a static priority driven preemptive scheduling technique that works as follows: Each task is assigned with a priority that reflects its required frequency. The scheduler selects the task having the highest priority among the ready tasks, and gives the CPU control to it. Rate-monotonic is well established technique and guarantees that all the mandatory sub-tasks will meet their deadlines as long as the processor utilization fulfills the following bound

$$\sum_{i=1}^{q} C_{i,m} f_i \leq q(2^{1/q} - 1)$$

where $f_i = 1/T_i$ for periodic tasks (see [10] for further details).

On the other hand, making decision on which optional tasks or how many optional tasks will be executed is a much more complex problem. Feedback control scheduling is a method for solving it. It is based on classical control theory and brings the scheduling area a new approach: closed-loop scheduling. Algorithms like rate monotonic and earliest-deadline-first are all “open-loop” where “open-loop” refers to the fact that once schedules are created they are not adjusted based on continuous feedback measurements. While open-loop scheduling algorithms can perform well in static or dynamic systems in which

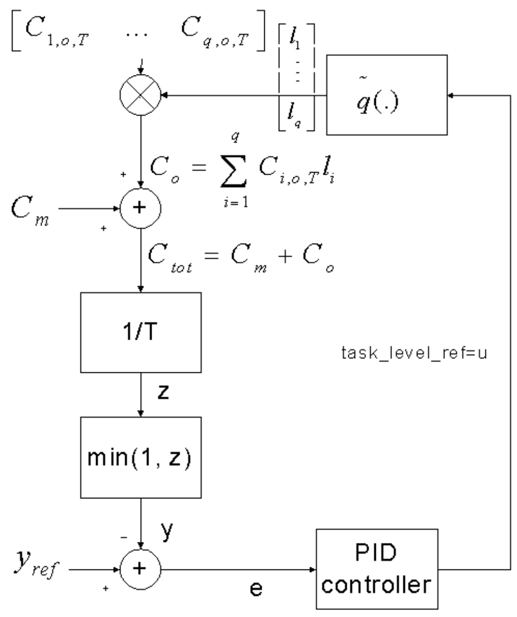
$\begin{bmatrix} C_{1,o,T} & \cdots & C_{q,o,T} \end{bmatrix}$
$\begin{bmatrix} l_1 \\ \vdots \\ l_q \end{bmatrix}$
$\tilde{q}(.)$
$C_o = \sum_{i=1}^{q} C_{i,o,T} l_i$
C_m
$C_{tot} = C_m + C_o$
1/T
task_level_ref=u
z
min(1, z)
y
y_{ref}
e
PID controller

Figure 2: System Model of Feedback Control RM Scheduling

the workloads can be accurately modelled, they can perform poorly in unpredictable dynamic environments. The idea of feedback control scheduling comes from a typical feedback control system which is composed of a controller, a plant to be controlled, actuators, and sensors. In order to apply control theory to a scheduler, we have to choose the components of the scheduler corresponding to those of a feedback control system. Some possible controlled variables can be miss ratio (which is defined as the number of deadlines misses divided by total number of completed tasks) or CPU utilization (which is defined as the percentage of CPU busy time in a sampling period). We have chosen CPU utilization as the controlled variable, since in hard real-time systems the miss ratio should be always kept zero. In order to keep the utilization at a given setpoint, the manipulated variable is the percentage or number of the optional sub-tasks to be executed. As seen in figure 1, controller receives as input the error, $Utilization\,Setpoint - Measured\,CPU\,Utilization$; processes the error and produces a value. This value is used by the task level controller in order to decide how many optional sub-tasks will be executed. In order to keep the CPU overhead for computing the control action to an acceptable level, the controller output is updated at a frequency $1/T$, where T denotes the sampling period (that is equal to 100 ms in our experiments, see table 1). In order to characterize the evaluation of the CPU utilization in a period of time T, the model given in figure 2 can be used to represent the system. Let $t \in \mathbb{N}$ denote the discrete time index, i.e. the actual time $\tilde{t}$ can be computed as $\tilde{t} = Tt$. Let N denote the total number of the tasks at time t, i.e.

$$N = \sum_{i=1}^{q} \frac{T}{T_i} \tag{1}$$

Note that, in general, the number of tasks is not an integer. However, by choosing $T \gg \max T_i$ the following approximation can be used

$$N = \sum_{i=1}^{q} N_i = \sum_{i=1}^{q} \left\lceil \frac{T}{T_i} \right\rceil \tag{2}$$

where $N_i = \lceil T/T_i \rceil$ is the number of instances of task τ_i in the period $[tT,\ (t+1)T]$ and $\lceil \cdot \rceil$ is the ceil function returning the lowest integer upper bound to the argument. As a rule of thumb, approximation (2) is sensible if T is at least an order of magnitude bigger than $\max T_i$.

Let the execution times of mandatory and optional sub-tasks be $C_{j,m}$ and $C_{j,o}$ respectively. The total execution times of mandatory and optional parts of one instance within a sampling period T are given by

$$C_{i,m,T}(t) = \sum_{j=1}^{N_i} C_{j,m}(t)\,,\ \ C_{i,o,T}(t) = \sum_{j=1}^{N_i} C_{j,o}(t) \tag{3}$$

The total execution time of all mandatory parts within a sampling period is

$$C_m(t) = \sum_{i=1}^{q} C_{i,m,T}(t) \tag{4}$$

The total execution time of all optional parts within a sampling period depends on the scheduling policy. If optional sub-tasks are scheduled according to Sieve method, each one of them is entirely executed or rejected. We can model this fact through the use of vectors $l^{(k)} \in \{0,1\}^q$, defined as $l^{(k)} = [1,1,...,1,0,0,...,0]^T$ and having k “1”s corresponding to the optional sub-tasks to be executed. Assume that the control action $u(t)$ is such that the optional sub-tasks of tasks $\tau_1, \tau_2, ..., \tau_k$ will be executed in the starting period (this corresponds to set $l(t) = l^{(k)}$ where the vector $l^{(k)}$ is obtained by quantizing $u(t)$ as explained next). Then, equation (5) gives the total execution time of all optional parts within one sampling period, under the assumption that $\tau_{1,o}, \tau_{2,o}, ...\tau_{q,o}$ are ordered according to the priorities $p_1 > p_2 > ... > p_q$.

$$\begin{aligned} C_o(t) &= \begin{pmatrix} C_{1,o,T}(t) & ... & C_{q,o,T}(t) \end{pmatrix} . \begin{pmatrix} l_1(t) \\ l_2(t) \\ \vdots \\ l_q(t) \end{pmatrix} \\ &= \sum_{i=1}^{q} C_{i,o,T}(t).l_i(t) = \sum_{i=1}^{k} C_{i,o,T}(t) \end{aligned} \tag{5}$$

where clearly, in the last inequality k is a time-varying quantity. The signal $z(t)$ and the CPU utilization $y(t)$ represented in figure 2 are given by

$$z(t) = \frac{C_m(t) + C_o(t)}{T}, \qquad y(t) = min\{1,\, z(t)\} \tag{6}$$

The error signal is the difference between requested utilization and measured utilization.

$$e(t) = y_{ref} - y(t) \tag{7}$$

The controller takes the error, processes it and produces a value in the range of $[0,\, 1]$. If the controller has a PID structure, then it computes $u(t)$ according to the equation

$$u(t) = K_p e(t) + K_i \sum_{i=t-IW}^{t} e(i) + K_d(e(t) - e(t-1)) \tag{8}$$

where IW is the time window over which to sum the errors. In other words, only errors in the last IW sampling periods are considered in the integral term. The output of PID controller is then quantized by the block $\widetilde{q}(.)$ seen in figure 2. The quantizer function first determines k according to the rule

$$k = \begin{cases} 0 & \text{if } -\frac{1}{2q} \le u < \frac{1}{2q} \\ 1 & \text{if } \frac{1}{2q} \le u < \frac{3}{2q} \\ & \vdots \\ q & \text{if } \frac{2q-1}{2q} \le u < \frac{2q+1}{2q} \end{cases} \tag{9}$$

and then generates the vector $l^{(k)}$ which determines the number of optional sub-tasks to be executed. For instance, if k is 3 then $l^{(3)} = [1, 1, 1, 0, ..., 0]^T$ and consequently three optional sub-tasks having the highest priorities p_1, p_2 and p_3 will be executed in the next sampling interval.

3. Statistical Characterization of the Signals

In order to account for the unpredictability of task durations, we model them as *independent* random variables with finite average and variance over which the scheduler has no control. In fact, as it will be shown next our results will be independent of the probability distributions describing task durations.

In the experiments, we assume that the execution times of mandatory and optional sub-tasks are sampled from the following uniform distributions

$$C_{j,m}(t) \sim U[a_{j,m},\, b_{j,m}]\,,\; C_{j,o}(t) \sim U[a_{j,o},\, b_{j,o}] \tag{10}$$

Then, mean and variance of the random variables $C_{j,m}$ and $C_{j,o}$ are given by

$$\mu_{C_{j,m}} = \frac{a_{j,m}+b_{j,m}}{2} \quad , \quad \sigma^2_{C_{j,m}} = \frac{(b_{j,m}-a_{j,m})^2}{12} \tag{11}$$

$$\mu_{C_{j,o}} = \frac{a_{j,o}+b_{j,o}}{2} \quad , \quad \sigma^2_{C_{j,o}} = \frac{(b_{j,o}-a_{j,o})^2}{12} \tag{12}$$

If there is a sufficient number of mandatory sub-tasks and a sufficient number of optional sub-tasks to be executed within a sampling period the Central Limit Theorem (CLT) can be used [7] in order to approximate the distributions of C_m and C_o. In our setting, the total number of mandatory sub-tasks N_m is equal to N given by equation (2) and the total number of optional sub-tasks to be executed is $N_o = \sum_{i=1}^{k} N_i$. We point out that despite the fact that CLT is an asymptotic theorem, it provides good approximations even if N_m and N_o are low (see [7]). In our experiments, we have $N_m = 238$ and $N_o = 179$ as can be calculated from table 1. An important fact is that CLT holds even if the task durations are not uniformly distributed. Therefore, all our results hold for general distributions associated to $C_{i,m}$ and $C_{i,o}$.

For a generic signal ξ, we denote with $f_\xi(\xi|l)$ its probability density function (PDF) for the choice of $l = l^{(k)}$ at time t. Let $g(\mu, \sigma^2)$ be the gaussian distribution with mean μ and variance σ^2. Then, for a fixed vector l, CLT states that C_m and C_o can be represented by

$$C_m \sim f_{C_m}(C_m|l) = g(\mu_m, \sigma^2_m),\; \mu_m = \sum_{i=1}^{q}\sum_{j=1}^{N_i} \mu_{C_{j,m}},\; \sigma^2_m = \sum_{i=1}^{q}\sum_{j=1}^{N_i} \sigma^2_{C_{j,m}} \tag{13}$$

$$C_o \sim f_{C_o}(C_o|l) = g(\mu_o, \sigma^2_o),\; \mu_o = \sum_{i=1}^{k}\sum_{j=1}^{N_i} \mu_{C_{j,o}},\; \sigma^2_o = \sum_{i=1}^{k}\sum_{j=1}^{N_i} \sigma^2_{C_{j,o}} \tag{14}$$

According to the scheme reported in figure 2, C_{tot} is the sum of C_m and C_o. Thus, in view of the statistical independence of C_m and C_o, we have

$$C_{tot} \sim f_{C_{tot}}(C_{tot}|l) = g(\mu_{tot}, \sigma^2_{tot}) = g(\mu_m + \mu_o, \sigma^2_m + \sigma^2_o) \tag{15}$$

The distribution function of z is given by

$$z \sim f_z(z|l) = g(\mu_z, \sigma_z^2) = g(\frac{\mu_m + \mu_o}{T}, \frac{\sigma_m^2 + \sigma_o^2}{T^2}) \tag{16}$$

For the distribution of y and e, we have

$$y \sim f_y(y|l) = \begin{cases} f_z(y|l) & y < 1 \\ (1 - \int\limits_{-\infty}^{1} f_z(z|l)dz)\delta(y-1) & y = 1 \\ 0 & y > 1 \end{cases} \tag{17}$$

$$e \sim f_e(e|l) = \begin{cases} f_z(y|l) & y < y_{ref} - 1 \\ (1 - \int\limits_{-\infty}^{1} f_z(z|l)dz)\delta(y - y_{ref} + 1) & y = y_{ref} - 1 \\ 0 & y > y_{ref} - 1 \end{cases} \tag{18}$$

Note that the block $min(1, z)$ in figure 2 produces a Dirac delta function into the distributions of y and e (see [7]). In fact, (17) and (18) represent a truncated gaussian distribution and the coefficient $\beta = 1 - \int\limits_{-\infty}^{1} f_z(z|l)dz$ multiplying the δ function is significantly different from zero only if the average CPU utilization y is close to one (β represents the probability of CPU overload for a given l). This situation is clearly seen in the left panel of figure 3 where the empirical PDF of e has been obtained through monte-carlo simulations of the system in figure 2. However, when the setpoint is chosen sufficiently away from 1 and a stabilizing controller is used, the dirac delta function disappears as shown in the right panel of figure 3. This corresponds to remove the block $min(1, z)$ in figure 2. We highlight that this approximation is realistic since the setpoint should be chosen pretty far away from 1 in order to prevent undesirable saturation effects [4], [11]. In fact, a setpoint that is close to one may cause overload and lead to deadline misses of mandatory sub-tasks, which is not allowed in hard real-time systems. Hence, formula (17) and (18) can be replaced by

$$y \sim f_y(y|l) = g(\mu_y, \sigma_y^2) = g(\frac{\mu_m + \mu_o}{T}, \frac{\sigma_m^2 + \sigma_o^2}{T^2}) \tag{19}$$

$$e \sim f_e(e|l) = g(\mu_e, \sigma_e^2) = g(y_{ref} - \frac{\mu_m + \mu_o}{T}, \frac{\sigma_m^2 + \sigma_o^2}{T^2}) \tag{20}$$

By using the PID formula (8), the probability distribution of the PID controller output is the gaussian function given by

$$u \sim f_u(u|l) = g(\mu_u, \sigma_u^2) = g((K_p + K_i IW)\mu_e,\ (K_p^2 + K_i^2 IW + 2K_d^2)\sigma_e^2) \tag{21}$$

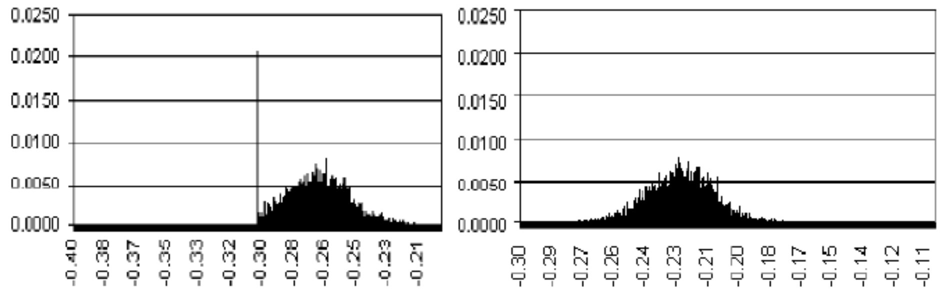

Figure 3: Empirical PDF of the error. Left panel: case of CPU overload. Right panel: absence of CPU overload.

Finally, we can derive the density function of k for a fixed vector $l^{(k)}(t)$,

$$\begin{aligned} f_k(k|l) &= \delta(k) \int_{-\infty}^{1/2q} f_u(u|l)du + \delta(k-1) \int_{1/2q}^{3/2q} f_u(u|l)du + \dots \\ &\quad + \delta(k-q) \int_{(2q-1)/2q}^{\infty} f_u(u|l)du \end{aligned} \qquad (22)$$

4. Design of a Stabilizing PID Controller

In this section, a stability analysis is presented by neglecting the effect of the quantizer. The assumption that there is no quantizer corresponds to the fact that the scheduler executes fractions of optional sub-tasks. Although the final goal is to consider the sieve implementation method for optional sub-tasks (which means that the sub-tasks are either executed or rejected entirely, i.e. the quantizer is used) for sake of clarity, we first focus on a system without quantization. Then, in section 5 we generalize our results to the case of quantization and highlight how quantization affects stability.

Note that, in absence of quantization, the total execution time of all optional sub-tasks within a sampling period is given by

$$C_o(t) = C_{o,tot}(t) \cdot u(t) = \sum_{i=1}^{q} C_{i,o,T}(t) \cdot u(t) \qquad (23)$$

In other words, formula (23) replaces (5).

The analysis is first done for a purely integral control action (i.e. $K_p = K_d = 0$ and $K_i \neq 0$ in (8)), and then generalized to the case of a complete

PID controller. The reason of considering a pure integral controller is that the integral action is sufficient to stabilize the system. On the other hand, the proportional and derivative actions may improve the tracking performance during the transients [8].

4.1. Stability with Integral Controller

In order to find a a stabilizing controller we first derive the state space form of the closed-loop system. Assume that $x \in \mathbb{R}$ is the state of the integral controller. Hence, controller equations are,

$$x(t+1) = x(t) + e(t)$$

$$u(t) = K_i x(t)$$

and the system equations are

$$y(t) = \frac{1}{T}(C_m(t) + C_{o,tot}(t)u(t))$$

$$e(t) = -y(t) + y_{ref}(t).$$

The closed-loop system is therefore described by

$$x(t+1) = x(t) + y_{ref}(t) - \frac{1}{T}C_m(t) - \frac{K_i}{T}x(t)C_{o,tot}(t) \qquad (24)$$

By assuming that $C_{o,tot}$, C_m and y_{ref} are stationary signals, the mean state obeys to the law

$$E[x(t+1)] = E[x(t)] + (y_{ref} - \frac{1}{T}\mu_m) - \frac{K_i}{T}E[x(t)]\mu_{o,tot} \qquad (25)$$

Let $\mu_x(t) = E[x(t)]$. Hence, we have

$$\mu_x(t+1) = (1 - \frac{K_i}{T}\mu_{o,tot})\mu_x(t) + (y_{ref} - \frac{1}{T}\mu_m) \qquad (26)$$

From this equation, the equilibrium for μ_x can be found as

$$\bar{\mu}_x = \frac{T}{K_i \mu_{o,tot}}(y_{ref} - \frac{1}{T}\mu_m) \qquad (27)$$

A classical stability criterion for discrete time linear systems [8] guarantees that $\mu_x(t)$ is asymptotically stable around $\bar{\mu}_x$, if the following condition is fulfilled:

$$|1 - \frac{K_i}{T}\mu_{o,tot}| < 1 \qquad (28)$$

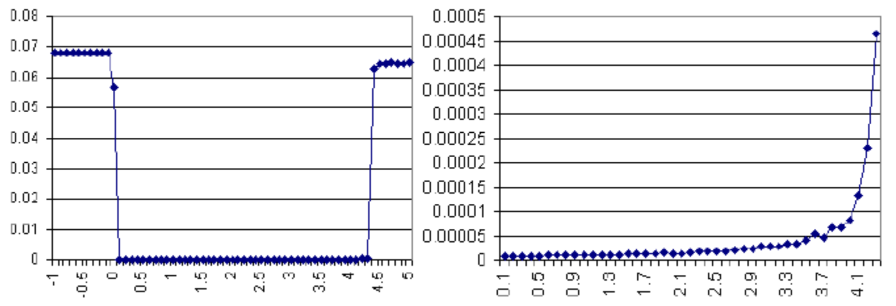

Figure 4: $E[e^2(t)]$ vs. K_i with **non-quantized** $u(t)$. Left panel: plot for $K_i \in [-1, 5]$, right panel: zoom for $K_i \in [0.1, 4.3]$.

If prior information about the total execution time of all optional sub-tasks is available in the form $\mu_{o,tot}(t) \in [\mu_{o,tot,min},\, \mu_{o,tot,max}]$, then, for guaranteing the stability, the parameter K_i of the integral controller must verify the inequality

$$0 < K_i < \frac{2T}{\mu_{o,tot,max}} \tag{29}$$

Inequality (29) takes into account the worst case for stability. This result is demonstrated in figure 4 that shows the diagram of $E[e^2(t)]$ vs. K_i obtained by simulations. According to (28) and the workload parameters given in table 1, the system is stable for $0 < K_i < 4.17$. This predicts quite accurately the experimental results (see figure 4) showing the achievement of stability for $K_i \in [0, 4.3]$. The main value of formula (29) is to highlight that stability properties depend only on $\mu_{o,tot}$ and T, while other parameters are irrelevant.

Remark 1 *The previous analysis provides some useful hints for the choice of the sampling period T. From the control side, T should be chosen as small as possible in order to guarantee a more effective control action. However, as remarked in Section 2.3, the computation of the control action introduces a CPU overhead that increases as T decreases. Then, a good choice of T results from the trade-off between these requirements. Moreover, periodicity of the system causes an anomaly such that if all task periods are not considerably smaller than the sampling period (i.e. $T < \max T_i$), different number of tasks appears in different sampling periods and utilization becomes jittery. This phenomenon can be avoided by imposing a lower bound to the sampling period [12].*

4.2. Stability with PID Controller

When the controller has the complete PID structure, the state equations of the controller become

$$\begin{aligned}
x_1(t+1) &= x_1(t) + e(t) = x_1(t) - \frac{C_m}{T} - \frac{C_{o,tot}}{T}u(t) + y_{ref} \\
x_2(t+1) &= e(t) = -\frac{C_m}{T} - \frac{C_{o,tot}}{T}u(t) + y_{ref} \\
x_3(t+1) &= x_2(t) \\
u(t) &= K_i x_1(t) + K_p x_2(t) + K_d(x_2(t) - x_3(t))
\end{aligned}$$

and the closed-loop system equations are

$$\begin{aligned}
\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \\ x_3(t+1) \end{bmatrix} &= \begin{bmatrix} 1 - \frac{C_{o,tot}K_i}{T} & -\frac{C_{o,tot}}{T}(K_p + K_d) & \frac{K_d C_{o,tot}}{T} \\ -\frac{C_{o,tot}K_i}{T} & -\frac{C_{o,tot}}{T}(K_p + K_d) & \frac{K_d C_{o,tot}}{T} \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} \\
&+ \begin{bmatrix} y_{ref} - \frac{C_m}{T} \\ y_{ref} - \frac{C_m}{T} \\ 0 \end{bmatrix}
\end{aligned}$$

Let $\mu_i(t) = E[x_i(t)]$ for $i \in \{1, 2, 3\}$. Then, we have $\mu(t+1) = A \cdot \mu(t) + B$ where

$$\begin{aligned}
\mu(t) &= \begin{bmatrix} \mu_1(t) \\ \mu_2(t) \\ \mu_3(t) \end{bmatrix}, A = \begin{bmatrix} 1 - \frac{\mu_{o,tot}K_i}{T} & -\frac{\mu_{o,tot}}{T}(K_p + K_d) & \frac{K_d \mu_{o,tot}}{T} \\ -\frac{\mu_{o,tot}K_i}{T} & -\frac{\mu_{o,tot}}{T}(K_p + K_d) & \frac{K_d \mu_{o,tot}}{T} \\ 0 & 1 & 0 \end{bmatrix}, \\
B &= \begin{bmatrix} y_{ref} - \frac{\mu_m}{T} \\ y_{ref} - \frac{\mu_m}{T} \\ 0 \end{bmatrix}
\end{aligned}$$

For stability, the following condition must be satisfied [8]:

$$\max |eig(A)| < 1 \tag{30}$$

where $eig(A)$ denotes the set of eigenvalues of A. An explicit computation shows that the eigenvalues of A must verify the equation

$$\lambda^3 + (\frac{\mu_{o,tot}}{T}(K_p + K_i + K_d) + 1)\lambda^2 + (\frac{\mu_{o,tot}}{T}K_p)\lambda - \frac{\mu_{o,tot}}{T}K_d = 0 \tag{31}$$

The stability conditions on K_p, K_i and K_d are now more involved than in the case of a pure integral action. However, formula (30) and (31) show that they still depend on the sampling period T and the total average of the execution times of optional sub-tasks $\mu_{o,tot}$.

5. The Effect of Quantization on the Stability

The effect of quantization on the stability will now be discussed. For convenience, we just consider an integral control action. As seen from figure 2, the block $\tilde{q}(u(t))$ quantizes the output of the controller and produces a value which determines the number of optional sub-tasks to be executed. The function implemented by the quantizer is given by $\tilde{q}(u) = k$ where k is defined in equation (9). When the quantizer enters the system, equation (24) becomes

$$x(t+1) = x(t) + (y_{ref}(t) - \frac{1}{T}C_m(t)) - \frac{1}{T}\tilde{q}(K_i x(t))C_{o,tot}(t) \qquad (32)$$

Taking the means, we have

$$\mu_x(t+1) = \mu_x(t) + (y_{ref} - \frac{1}{T}\mu_m) - \frac{1}{T}\, E_x[\tilde{q}(K_i x(t))]\mu_{o,tot} \qquad (33)$$

By using the definition of average, the mean of the quantizer output can be expressed as a function of μ_x:

$$E[\hat{q}(x)](\mu_x) = \int_{-\infty}^{\infty} \tilde{q}(K_i x)\frac{1}{\sqrt{2\pi\sigma_x^2}} e^{-\frac{|x-\mu_x|^2}{2\sigma_x^2}} dx \qquad (34)$$

Let $\hat{q}(x) = \tilde{q}(K_i x)$. Thus, from equation (9), we have

$$\hat{q}(x) = \begin{cases} 0 & \text{if } -\frac{1}{2qK_i} \leq x < \frac{1}{2qK_i} \\ 1 & \text{if } \frac{1}{2qK_i} \leq x < \frac{3}{2qK_i} \\ & \vdots \\ q & \text{if } \frac{2q-1}{2qK_i} \leq x < \frac{2q+1}{2qK_i} \end{cases} \qquad (35)$$

From (34), it is obvious that $E[\hat{q}(x)]$ is the convolution between $\hat{q}(x)$ and a gaussian function. As σ_x^2 decreases, the gaussian converges to $\delta(x - \mu_x)$ and then $E[\hat{q}(x)](\mu_x)$ converges to $\hat{q}(\mu_x)$. Figure 5 shows the mean when $\sigma_x << \frac{1}{qK_i}$, $\sigma_x \simeq \frac{1}{qK_i}$ and $\sigma_x >> \frac{1}{qK_i}$, respectively. First, the case for $\sigma_x << \frac{1}{qK_i}$ will be examined. In this case, $E[\hat{q}(x)](\mu_x)$ is well approximated by $\hat{q}(x)$. Therefore, equation (33) becomes

$$\mu_x(t+1) = \mu_x(t) + (y_{ref} - \frac{1}{T}\mu_m) - \frac{1}{T}\, \hat{q}(\mu_x(t))\mu_{o,tot} \qquad (36)$$

In order to investigate the stability of the system described by equation (36), we first highlight that, because of the quantized input, the best we can expect is that μ_x asymptotically lies in a bounded interval D around $\bar{\mu}_x$ (where $\bar{\mu}_x$ is computed as in (27) for a fixed value of y_{ref}). The objective of the controller is to shrink D as much as possible. A precise result can be obtained by applying proposition 2.3 of [9], that specialized to our context, results in the next statement.

Proposition 1 *Let γ be such that the inequality $|1 - \frac{K_i\mu_{o,tot}}{T}| < \gamma < 1$ is verified and consider the interval $D = \left\{\mu_x \in \mathbb{R} \ : \ \|\mu_x - \bar{\mu}_x\| \leq \frac{\mu_{o,tot}}{qT(1-\gamma)}\right\}$*

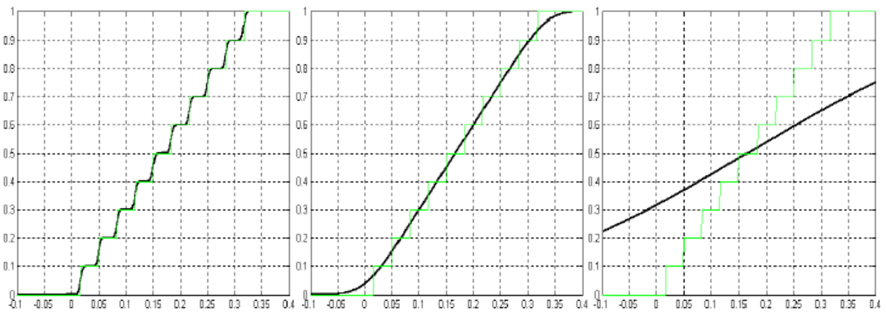

Figure 5: Computation of $E[\tilde{q}(K_i x)](\mu_x)$ according to (34) when $\sigma_x(t) = \frac{1}{10}\frac{1}{qK_i}$, $\sigma_x(t) = \frac{1}{qK_i}$ and $\sigma_x(t) = \frac{10}{qK_i}$ respectively ($q = 10$, $K_i = 3$).

where $\bar{\mu}_x$ is computed as in (27) for a given value of y_{ref}. Then, for all $\mu_0 = \mu_x(0)$ there exists $N > 0$ such that $\mu_x(t) \in D$, $\forall t \geq N$. Moreover, if $\mu_0 \in D$, then $\mu_x(t) \in D$, $\forall t \geq 0$.

Let $\delta = \frac{\mu_{o,tot}}{qT(1-\gamma)}$. When μ_x remains in D, the output of the controller u lies within $[K_i(\bar{\mu}_x - \delta),\ K_i(\bar{\mu}_x + \delta)]$. From the equilibrium equation (27), it can also be found that CPU utilization verifies $y \in [y_{ref} - \frac{1}{T}\mu_{o,tot}\delta,\ y_{ref} + \frac{1}{T}\mu_{o,tot}\delta]$. Let us say that $\delta \leq L\Delta$ where $\Delta = \frac{1}{qK_i}$ is the quantizer's sensitivity and $L \in \mathbb{Z}^+$. Obviously, it is desirable to choose K_i such that $\delta \leq L\Delta$ is verified for $L = 1$. In order to keep μ_x within $[\bar{\mu}_x - L\Delta,\ \bar{\mu}_x + L\Delta]$, the following condition must be satisfied

$$0 < K_i < \frac{2L}{1+L}\frac{T}{\mu_{o,tot}} \tag{37}$$

If we choose $L = 1$, then, to keep μ_x within the range of $[\bar{\mu}_x - \Delta,\ \bar{\mu}_x + \Delta]$, the parameter K_i should be chosen as

$$0 < K_i < \frac{T}{\mu_{o,tot}} \tag{38}$$

In our experiments, according to table 1, this corresponds having $0 < K_i < 2.09$. When K_i exceeds 2.09, μ_x becomes within the range of $[\bar{\mu}_x - 2\Delta,\ \bar{\mu}_x + 2\Delta]$ and from condition (37), for $K_i > 2.78$, the range for μ_x will be $[\bar{\mu}_x - 3\Delta,\ \bar{\mu}_x + 3\Delta]$. Note that as K_i increases, also $E[e^2(t)]$ increases. This can be seen from the simulations reported in figure 6.

Consider now the case when σ_x is "large", i.e. it verifies $\sigma_x \geq \frac{1}{qK_i}$. From figure 5, we have that $E[\tilde{q}(K_i x)](\mu_x)$ can be approximated with the function

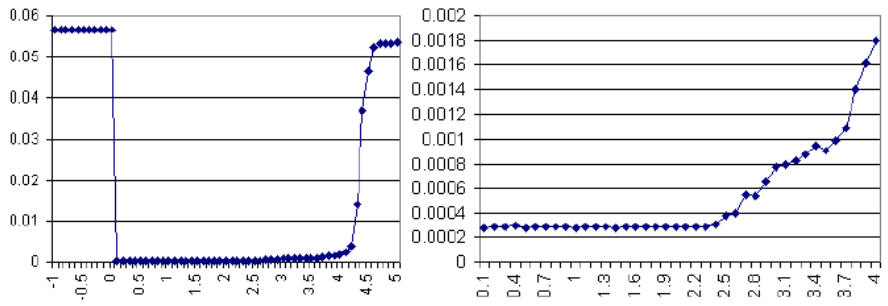

Figure 6: $E[e^2(t)]$ vs. K_i with **quantized** $u(t)$. Left panel: plot for $K_i \in [-1, 5]$, right panel: plot for $K_i \in [0.1, 4]$.

$\eta\mu_x + \xi$ on the interval $[-\frac{1}{2qK_i}, \frac{2q+1}{2qK_i}]$. In this case, the quantization effect disappears. Therefore, the mean equation becomes

$$\mu_x(t+1) = \mu_x(t) + (y_{ref} - \frac{1}{T}\mu_m) - \frac{1}{T}\psi(K_i\mu_x(t))\mu_{o,tot} \qquad (39)$$

where $\psi(K_i\mu_x(t)) \approx \eta\mu_x + \xi\,, \quad \forall\,[K_i\mu_x(t)] \in [-\frac{1}{2qK_i}, \frac{2q+1}{2qK_i}]$. Then, the mean equation can be written as

$$\mu_x(t+1) = \mu_x(t) + (y_{ref} - \frac{1}{T}\mu_m) - \frac{1}{T}(\eta\mu_x(t) + \xi)\mu_{o,tot} \qquad (40)$$

and the stability condition reads as

$$0 < K_i < \frac{1}{\eta}\frac{2T}{\mu_{o,tot,max}} \qquad (41)$$

Note that the previous analysis highlight that, in the quantized case the state variance $\sigma_x(t)$ influences the stability properties and the choice of the parameter K_i. This suggests the use of an adaptive strategy for tuning $K_i(t)$ on-line based on the empirical estimate of $\sigma_x(t)$. This topic will be considered in future research.

6. Conclusions

In this paper, we presented a feedback control rate-monotonic scheduling system. By assuming that all tasks are periodic and implemented with the two-version method, we derived stability conditions for the proposed system when PID controllers are used. The reason of choosing PID as a control action is that it is a well-established technique in automatic control and it has been proved to stabilize the feedback control scheduling system in aforementioned works. On the other hand, the choice of more effective control schemes is still an important research issue and future work will focus on alternatives to pure PID such as adaptive PID or optimal control. This work can also be extended to the case where the system processes are both periodic and aperiodic tasks and there is also task dependency. Finally, optimizing system performance by tuning simultaneously all the design parameters, including K_p, K_i, K_d and T, also requires additional research that will hinge on the statistical framework developed in the paper.

7. References

[1] J.W.S. Liu, K.J. Lin, W.K. Shih, A.C.S. Yu, J.Y. Chung, W. Zhao, "Algorithms for scheduling imprecise computations", IEEE Computer, May (1991) $58 - 68$. (5) (1987).

[2] Jane W.S. Liu, Wei-Kuan Shih, Kwei-Jay Lin, Riccardo Bettati and Jen-Yao Chung, "Imprecise Computations", Proceedings of the IEEE, Vol. 82, No.1, January 1994.

[3] Chenyang Lu, John A. Stankovic, Gang Tao, Sang H. Son, "Design and Evaluation of a Feedback Control EDF Scheduling Algorithm". 20^{th} IEEE Real-Time Systems Symposium, 1999, pg. 56-67.

[4] Chenyang Lu, John A. Stankovic, Gang Tao, Sang H. Son, "Feedback Control Real-Time Scheduling: Framework, Modeling, and Algorithms", Real-Time Systems Journal Special Issue on Control Theoretical Approach to Real-Time Computing 23(1/2):85-126 September, 2002.

[5] Chenyang Lu, John A. Stankovic, Tarek F. Abdelzaher, "Performance Specifications and Metrics for Adaptive Real-Time Systems", IEEE Real-Time System Symposium (RTS2000), December 2000.

[6] D.A.Lawrence, J.Guan, S.Mehta, L.R. Welch, "Adaptive Scheduling via Feedback Control for Dynamic Real-Time Systems", 20^{th} International Performance, Computing and Communications Conference, April 2001.

[7] Athanasios Papoulis, 1987, Probability, Random Variables and Stochastic Processes, McGraw-Hill International Editions.

[8] Gene F. Franklin, J. David Powell, Abbas Emami-Naeini, 2002, Feedback Control of Dynamic Systems, Prentice Hall.

[9] David F. Delchamps, "Stabilizing a Linear System with a Quantized State Feedback", IEEE Transactions on Automatic Control, Vol.35, No. 8, August 1990.

[10] Liu, C.L., Layland, J.W., 1973. "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment". JACM. 20(1): 40-61.

[11] Tolga Ayav, Sinan Yilmaz, "Neuro-Fuzzy Controller in Real-Time Feedback Schedulers", In Proceedings of the 10^{th} Workshop on Nonlinear Dynamics of Electronic Systems, pp. 3.37-3.40, Izmir, Turkey, 2002.

[12] J. Eker, P. Hagander, K. Arzen, “A Feedback Scheduler for Real-Time Controller Tasks”, In IFAC Control Engineering Practice, 2000.

[13] John A. Stankovic, Chenyang Lu, Sang H. Son and Gang Tao, “The Case for Feedback Control Real-Time Scheduling”, In Proceedings of the 11^{th} Euromicro Conference on Real-Time Systems, pp. 11-20, York, UK, 1999.

[14] Giorgio Buttazzo and Luca Abeni, “Adaptive rate control through elastic scheduling”, Proceedings of the 39^{th} IEEE Conference on Decision and Control, Sydney, Australia, Dec. 2000.