Frequency domain evaluation of real-time systems in control applications

C. Aciti [#†1], R. Cayssials [*2], E. Ferro, J. Urriza

[#] *Universidad Nacional del Centro de la Provincia de Buenos Aires*

Pinto 399 – Tandil – Argentina

[†] *Universidad Nacional de Tres de Febrero*

Valentin Gomez 4752 – Caseros - Argentina

[1] claudio.aciti@gmail.com

[*] *Universidad Tecnológica Nacional –Facultad Regional Bahía Blanca*
11 de Abril 461 - Bahía Blanca - Argentina

[2] rcayssials@frbb.utn.edu.ar

***Abstract*— Real-time systems are utilized in control applications. However, real-time systems may cause undesirable effects on the control application. Stability, overshoot and settling time may be affected when an inadequate real-time system is used.**

Frequency domain techniques are widely applied in control theory for designing control strategies as well as analyzing and measuring the performance of control mechanisms.

In this paper, frequency domain analysis is proposed to measure the harmonic distortion that a real-time system produces on the control application. Harmonic distortion is defined as a criterion to evaluate the control performance of real-time systems. This criterion is used to compare the control performance of the two most important priority disciplines in real-time: Fixed Priority (FP) and Earliest Deadline First (EDF). Experiences show that the control performance of both priority disciplines is similar for low utilization factors and FP performs better than EDF for not only higher utilization factors but also over-saturated conditions.

***Resumen*— Los sistemas de tiempo-real son utilizados en aplicaciones de control. Sin embargo, los sistemas de tiempo-real pueden causar efectos indeseados en las aplicaciones de control. Estabilidad, sobre picos y el tiempo de establecimiento pueden ser afectados cuando se usa un sistema de tiempo-real inadecuado.**

Diversas técnicas en el dominio de la frecuencia son ampliamente aplicadas en la teoría de control para el diseño de estrategias de control, así como el análisis y la medición del desempeño de los mecanismos de control .

En este trabajo se propone el análisis frecuencial para medir la distorsión armónica que un sistema de tiempo real produce en las aplicaciones de control. La distorsión armónica se define como un criterio para evaluar el rendimiento de control de sistemas de tiempo-real. Este criterio es utilizado para comparar el rendimiento de las dos disciplinas, basadas en prioridades, más importantes en sistemas de tiempo-real: Prioridad Fija (PF) y Primero el Más Urgente (EDF). La experiencia muestra que el rendimiento de control de ambas disciplinas prioritarias es similar para los factores de carga bajos y que FP se comporta mejor que EDF para factores de carga altos y en condiciones de sobrecarga .

I. Introduction

Digital control utilizes real-time systems to implement the control functions that the controller must achieve. Real-time tasks perform each one of these control functions. These real-time tasks need to be executed periodically with exact timing constraints to guarantee that the system stays under control, meeting the design specifications.

Frequency domain analysis is widely utilized in control theory. The behavior of a control application may be described according to the response it produces to each input frequency. A typical control system is shown in Fig 1. The controller should be designed to accomplish the adequate attenuation and amplification of the different input frequencies to produce the desired response of the application [1]. Hence, from a frequency domain point of view, a controller is a filter designed to generate the correct close-loop response.

When temporal constraints are not met, then the controller does not behave as a filter anymore and produces undesirable output frequencies. These frequencies are transmitted to the application through the actuators and may produce undesirable effects such as: vibration, heat, instability and noise.

A typical real-time system consists of several control functions in addition to other non-control functions (like user interaction, system maintenance, data communication, etc.). A scheduler mechanism is needed to share the processor among the different system tasks. Usually, the scheduler is implemented as a task of a Real-Time Operating System (RTOS). The scheduler implements a priority discipline that defines the next task that grants the processor for execution.

Several priority disciplines have been proposed in real-time systems. EDF and FP are two of the most analyzed ones because of their real-time features. In [2], the FP and EDF scheduling algorithms are proposed and analyzed to be used in real-time systems. In [3], the real-time features of EDF and FP are compared but no analysis of these disciplines in control application is performed.

In [4], True-Time and Jitterbug are proposed as simulation tools for real-time systems in control applications. True-time proposes a stability region which bounds the jitter that a control application may support. This boundary is based on an energy analysis and consequently it produces a conservative conclusion about stability but none about performance nor perturbation. Jitterbug is a set of simulation models for Simulink/Matlab. It allows to simulate the behavior of a compound real-time/control application. However, as simulation is based on particular case studies, we cannot perform a general analysis of the features of the priority disciplines. Both tools are intended for simulation of case studies but no analysis is done about the FP and EDF disciplines in control applications.

In this paper, we analyze the properties of the FP and EDF disciplines in control applications proposing an innovative technique. We measure the distortion that each one of these disciplines produces in the output of the real-time system. This distortion can be used as a figure of the perturbation that the real-time system produces on the application. We show, through experiences, that the performance of both disciplines for control applications is similar.

The paper is organized as follows: Section II introduces the main concepts on state-space model and discrete-time control. Section III describes the typical task model applied to real-time theory. Section IV explains the main concepts of the frequency domain techniques utilized in classical control theory. Section V describes the mechanisms to implement control application in real-time systems. The management of overload is detailed in Section VI. Experiences are described in Section VII. Results are analyzed in section VIII. Conclusions are drawn in section IX.

II. Control Model

A controlled system consists essentially of a plant and a controller. The plant is the system to be controlled and the controller reads information from the plant and computes the actions required to achieve some control performance.

Modeling techniques are developed to express the behavior of both the plant and the controller. A model of a system is a simplified, abstracted construct used to predict the dynamic of the system. It is often possible to obtain an analytical system model using the laws that determine its behavior.

The state-space modeling is a classical control technique to model a system. For a linear, time-invariant, continuous-time system, a state-space description consists of a first-order differential equation vector, named state-space equation, for x(t), named state variables,

$$x(t) = A \cdot x(t) + B \cdot u(t)\,, \tag{1}$$

$$y(t) = c \cdot x(t) + d \cdot u(t)\,, \tag{2}$$

where A is the system matrix, B is the input matrix, c is the output matrix, d is the feedthrough matrix, $u(t)$ is the input variable and $y(t)$ is the output variable .

The prototype control system encountered in classical control theory is shown in Fig 1. In this figure, the measured feedback signal is directly the system output $y(t)$. The input $r(t)$ is the reference input and $e(t)$ is the error signal.

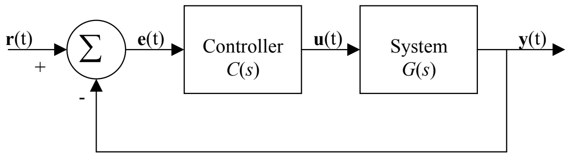

Fig. 1. Continuous-time Control System.

The design aims to specify the transfer function of the controller, $C(s)$, to give to the closed-loop the desired control characteristics. These characteristics include stability and possibly some specification on the step response such as overshooting, settling time, and steady-state error.

A. Discrete State-Space Models

When a controller is implemented on a computer system, a discrete-time model should be utilized. Fig 2 shows a diagram of a discrete-time control of a continuous-time plant.

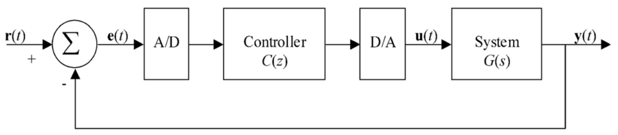

Fig. 2. Discrete-Time Control System.

$C(z)$ is implemented as software and it is executed by a processor. The state-space model of a discrete-time system is expressed by difference equations:

$$x(k+1) = \Phi \cdot x(k) + \Gamma \cdot u(k)\,, \tag{3}$$

$$y(k) = c \cdot x(k) + d \cdot u(k)\,, \tag{4}$$

In [1], it is proved that, given a state-space description of a plant characterized by (A, B, c, d) in equations (1) and (2), and a sampling interval T, the equivalent discrete-time system is given by Φ, Γ, c, d, where

$$\Phi = e^{AT}, \Gamma = \int_0^T e^{AT} \cdot B \cdot \partial\tau\,, \tag{5}$$

The discrete-time model derived from this transformation is not an approximation but an exact description of the behavior of the plant at sampling instants.

The computational execution of the discrete model takes some time to complete. If the computational delay is very small compared with the dynamic of the system, it can be neglected. If the delay introduced by the computation is constant, then it may be considered in the transfer function of the controller.

On the other hand, if the computation produces a non-constant delay then the discrete-time model is not valid anymore, and consequently the same actions may produce different effects, leading to an undesirable behavior [1].

B. Specification the Sampling Period

The selection of the sampling period of the system is important. If the sampling period is chosen too long, the continuous-time signal will not be able to be reconstructed. On the other hand, if it is chosen too small, the workload on the computer will increase and possibly the output update will not take place on time.

There exist several rules of thumb to determine the range to select the sampling frequency, defined as $ws=2.\pi/T$ rad/sec. Most of these rules are based on relationships between the sampling frequency and the close-loop bandwidth of the system, denoted by w_B.

A higher sampling rate could turn the system very sensitive to the precision of the parameters and to round-off errors.

III. Real-Time System Scheduling

The discrete-time control model needs the periodical computation of the controller strategies. These strategies are performed by control tasks that are executed concurrently by a processor. A scheduler mechanism is needed to share the processor among the system tasks.

Real-time systems theory allows analyzing the temporal properties of a set of concurrent tasks. The general process model of a real-time system consists of a set Π of N periodic and non-periodic tasks. Each task, τ_i is characterized by either its period in case of periodic tasks or minimum interarrival time for non-periodic ones, T_i, deadline, D_i, worst-case execution time, C_i, offset, O_i and priority, P_i.

$$\Pi = \left\{ \tau_i = \left(T_i, D_i, C_i, O_i, P_i \right), 1 \le i \le N \right\}, \qquad (6)$$

Each time that a task requires the processor to be executed, it is said that the task is invoked. The ready task with the highest priority P_i is the next task selected to be executed. The notion of jitter is important for our discussion. Sampling jitter is the maximum difference between the exact sampling period of two consecutive invocations of the same task. Similarly, the output jitter is the maximum difference between the exact output instants of two consecutive invocations of the same task.

Exact, necessary and sufficient scheduling analysis conditions exist to guarantee that the temporal requirements will be satisfied in a real-time system. The most advanced technique computes the worst case response time of any task to guarantee that completes before its deadline.

The time that a task has to wait to be executed depends on the computation time required by higher priority tasks that are ready to be executed. Because the pattern of releases is not fixed, and execution times of a task may vary from invocation to invocation, this results in a variable interference and therefore a variable response time of the task. These time variations produce a jitter on both input sampling and output updating that may cause undesirable effects on the control system [5] [6] [7].

IV. Digital Control and Frequency domain Analysis

The basic operation of a digital control system (Fig. 2) is to read information from multiple sensors, calculate the output and send the results to actuators. *C(z)* is implemented as software and it is executed by a processor. The input data to *C(z)* is a sequence of numbers obtained from an *A/D* converter and the output is a sequence of numbers that is converted to a piecewise control signal by the D/A converter. Both *A/D* and *D/A* converters are sampled at regular intervals and transform the continuous-time transfer function *G(s)* into a discrete-time model of the control application.

The controller should produce an output control signal adequate to control the application. When a classical control system is considered (linear and time-invariant), then the frequency of the signal at controller´s output should be equal to the frequency of the signal at its input. This output control signal can be transformed to a summation of sine and cosine signals according to the Fourier transform. When there exist jitter on the execution of the control task, then undesired frequencies appears on the output signal that are transmitted to the application through the actuators and may produce unwanted effects such as: vibration, heat, instability, noise. Consequently, the harmonic distortion may be used as an indicator of the perturbation that a control task produces when it is executed with jitter.

In this paper, we proposed the harmonic distortion as a measure to evaluate the performance of a real-time priority discipline in control applications.

V. Scheduling Control Tasks

A real-time system includes others types of tasks different from the control ones. There exist safety as well as secondary functions (e.g. man-machine-interface and communication tasks) that should be executed by the real-time system.

A control task can be divided into three subtasks with different real-time requirements:

- Sampling Subtask: this subtask has to be periodically executed to read the inputs of the system. Its execution has to be strictly periodic ($D << T$) to avoid non-linearities and time-variances that can lead to uncontrollable dynamics. The period of the sampling subtask is defined by the sampling period of the discrete-time control model and may vary in narrow range.
- Calculation Subtask: this subtask is executed after its corresponding sampling subtask has completed and it computes the control strategy. Its execution time depends on the complexity of the control strategy implemented.
- Actuation Subtask: this subtask may be executed either: (1) when the calculation subtask is completed or (2) after an fixed offset from the invocation of the sampling subtask. It should be executed as soon as it is invoked.

The control subtasks can be arranged in four main configurations:

A. Case A

The sampling subtask is executed periodically. The calculation subtask is executed when the sampling subtask is completed and then the actuation subtask. Fig 3 shows a real-time system that consists of two tasks: $Task_0$ and $Task_1$. While $Task_0$ is a non-control tasks, $Task_1$ is a control task constituted by its three corresponding subtasks: sampling subtask, S_i, computing subtask, C_i and actuation subtask, A_i, where i is the invocation index. In Fig 3, $Task_0$ is assigned with a higher priority than $Task_1$ to show the interference from higher priority tasks.

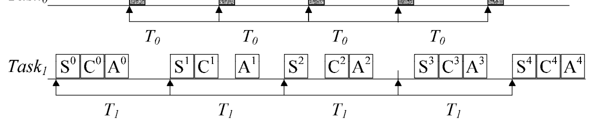

Fig. 3: Case A - control $Task_1$ is executed at a low priority level. The task suffers jitter due to higher priority tasks.

In this case, the arrival of a higher priority task may delay either the release of the task or the final actuation resulting in both input and output jitter. In Fig 3, sampling jitter on the $Task_1$ takes place because of, for instance, the interval between S^2 and S^3 is greater than T_1 while interval between S^3 and S^4 is less than T_1. Actuation jitters can be noted because of the interval between A^0 and A^1 is greater than T_1 whilst the interval between A_3 and A_4 is less than T_1.

B. *Case B*

The above problem of input jitter can be alleviated if the task is split into two computational entities (threads), the sampling subtask and the rest and where each part is executed at a different priority. By running all sampling subtasks at a higher priority, the input jitter is reduced dramatically. Fig 4 shows the execution of the three subtasks of $Task_1$ in two different priority levels.

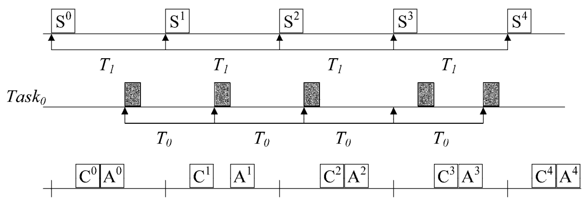

Fig.4: Case B - the sampling subtask is assigned to a higher priority to avoid sampling jitter.

Both subtasks should have the same period, this ensures that although the computation and actuation thread is released at the same time, it will never be executed before the sampling subtask because of the priority ordering. Another alternative implementation is for the sampling subtask to release explicitly the computation and the actuation subtasks.

C. *Case C*

The previous configuration solves the problem of input jitter, but not output jitter. A simple extension to overcome the problem is to run the actuation subtask at a higher priority together with the sampling task.
In this case, the sampling subtask is executed periodically. The actuation subtask is executed just after the sampling subtask is completed, however it outputs the result of the calculation of the previous invocation. All computation subtasks run as before at lower priorities. Because this is a fixed delay of precisely T units, its impact can be modeled precisely in the discrete-time control model. In this situation we have no output jitter but only a longer delay in the actuation. Fig 5 shows the execution of the three subtasks of the $Task_1$.

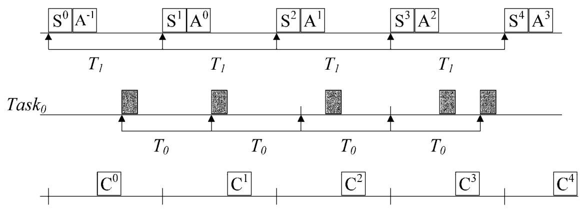

Fig.5: Case C - Both sampling and actuation subtasks are assigned to a higher priority to avoid both sampling and actuation jitter. Actuation delay is introduced.

D. *Case D*

The previous case is a particular case of the most general problem of having the subtasks at different priorities. In this case, the sampling subtask is executed periodically. The actuation subtask is executed with a fixed offset from sampling period. Response time analysis techniques can be used to compute the smallest offset that ensures that the computation subtask will always have completed before the actuation subtask is released. In this case, there is no output jitter and a smaller delay than in the previous case.

Fig 6 shows the execution of the three subtasks of $Task_1$ in two different priority levels. When the priority of the actuation subtask is just below the sampling subtask and $O_1=T_1$, this case reduces to case C.

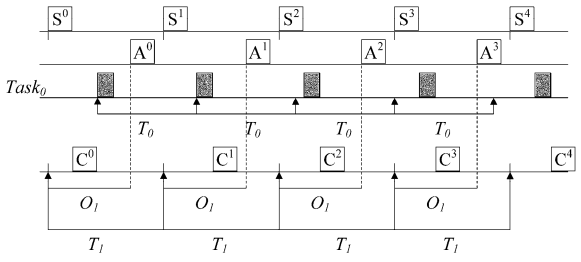

Fig.6: Case D - The actuation subtask is invoked at a determined offset. Sampling and actuation jitter are eliminated.

Complexity and overhead are the disadvantages of this approach. This configuration requires almost as much resources of the systems as if they were three independent tasks. Besides, an optimal priority assignment is not trivial.

Real-time tasks that implement control functions can be scheduled according to any of the cases described above. However, when two or more control functions have to be performed simultaneously, the real-time scheduling mechanism will produce perturbations on the control application independently of the case selected.

VI. Overload Management and Scheduler Preemption

Hard real-time theory considers that no deadline can be missed. However, designing a real-time system under this premise may be very conservative because the system is designed to be schedulable for a worst case that may rarely occur. Thus, if the system is designed considering a more feasible scenario instead of considering the worst case of load that may ever happen, then some deadline may be missed when the utilization factor overcome a certain bound. Depending on the application model, the notion of "missed deadline" can be associated to [8]:

- Delayed completion: the task invocation runs until completion even though it finishes after the deadline.
- Abortion: the task invocation is terminated before finishing its computation either because it will not end its computation before its deadline or because a higher priority task needs the resource.

This categorization is important because it identifies one of the potential applications of resource adaptation. A scheduler may determine the abortion of some task invocations with the aim of reducing the processor load and meet the deadline of critical task during a overloaded situation. If a delayed completion mechanism is implemented, then new invocations are skipped when the previous invocation of the task does not finish before the release of the new invocation.

Real-time schedulers may implement non-preemptive as well as preemptive algorithms to make it suitable to the application. Non-preemptive algorithms are designed so that once a task switches to the execution state, it is not removed from the processor until it has completed. In non-preemptive algorithms, the context switching is called only when the task completes or blocks. On the other hand, preemptive algorithms are driven by the notion of prioritized computation. The task with the highest priority should always be the one currently assigned to the processor. If a task is being executed and a new task with a higher priority releases, the task on the processor should be removed and returned to the ready state until it is once again the highest-priority ready task in the system.

Non-preemptive algorithms introduce a limited scheduling overhead because of the reduced number of context switches. On the other side, executing a non-preemptive scheduling algorithm might lead to limited schedulability performances due to the large blocking imposed on tasks with smaller deadlines [9][10][11]. Preemptive schedulers improve the response time of the highest priority tasks.

In this paper, we evaluate the control performance of EDF and FP priority disciplines under non-preemptive and preemptive schedulers with delayed completion and abortion.

VII. Experimental set-up

In this section we propose a set of randomly generated real-time systems to compare the performance of FP and EDF disciplines in control applications.

We generated 1200 real-time systems of 10 tasks each. The period of each task was randomly generated between $400\mu s$ and $7000\mu s$. Deadlines were set equal to periods. The total utilization factor of these systems ranged from 0.6 to 1.6.

Each system was simulated applying both an EDF and a FP disciplines. The simulation time was set equal to 12 seconds. Each task of the system was programmed to perform the following activities: (1) read a value from a table, (2) wait some time in order to produce the required execution time and (3) write the data into a memory variable. The reading is associated with the sampling action of the control task while the writing is associated with the output updating action. When a task is invoked before the completion of the previous invocation, then the new invocation is ignored.

The table stores 32 values of a sine wave. Each time that the task is invoked, it reads a value and increments a pointer to the next value of the table for the next invocation. Therefore, as reading and writing are performed periodically, they produce an input and an output of a sine wave, respectively. Each time that a reading or writing event takes places, it is logged in a file for further frequency domain analysis.

Frequency domain analysis was applied to the data obtained to show the perturbations that each one of the priority disciplines produce on the control tasks. When a task is executed with no jitter, then the spectrum analysis shows a very sharp component at frequency f_s (Fig. 7) equal to:

f_s=1 / (period of the task * length of the table).

When the task is executed either with jitter or with a different period (as it could happen with an oversaturated EDF discipline), then different harmonic frequencies appear in the spectrum analysis (Fig 8).

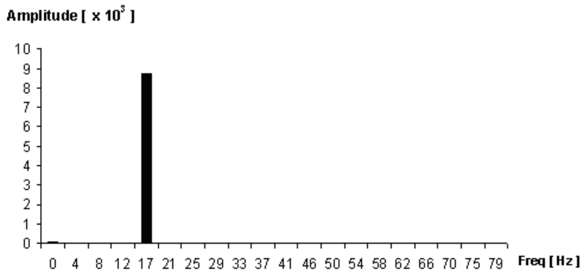

Fig.7: Frequential spectrum of a low harmonic distorted signal.

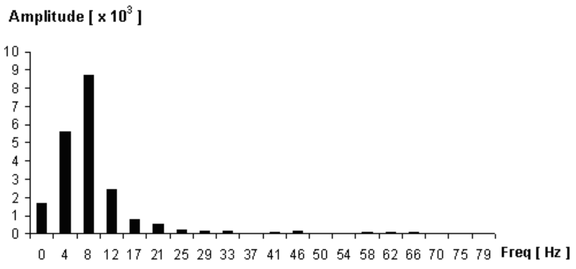

Fig.8: Frequential spectrum of a high harmonic distorted signal.

The Total Harmonic Distortion (THD) is calculated to get a figure of the perturbation that the real-time system produces on each control task. We define THD for a signal with a fundamental frequency f_s, as:

$$THD = \frac{\sqrt{\sum_{j \neq s} \left(V_j\right)^2}}{V_S}$$

where V_j is the magnitude of the component at the frequency f_j in the spectrum of the signal.

Lower values of THD means lower perturbation. On the other hand, when the signal presents high harmonic components, then the value of THD increases.

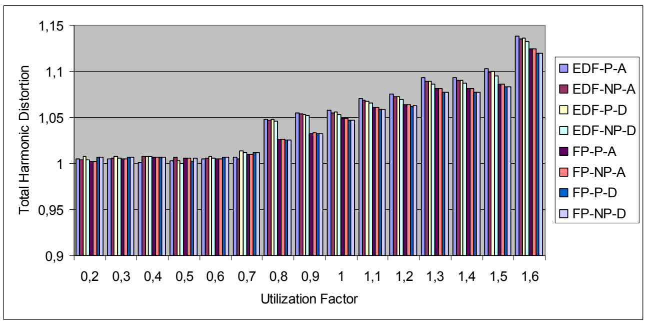

Fig.9: Simulation results.

Fig 9 shows the harmonic distortion of both EDF and FP disciplines. Legends are noted with the following features:

- P: for preemptive scheduling,
- NP: for non-preemptive scheduling,
- A: for abortion management,
- D: for delayed execution.

when utilization factor is lower than 0.7, the harmonic distortion is very low for both priority disciplines. In this case, the perturbation depends on how tasks are invoked instead of how they are scheduled. However, when utilization factor increases, the pattern of execution produced by a Fixed Priority discipline produces a lower harmonic distortion of the task than a EDF discipline.

VIII. Discussion

From the simulation results, we can observe that the performance of Fixed Priority is better than the performance of an EDF discipline when utilization factor is greater then 0.8. This feature can be explained by analyzing how the execution pattern is generated by each priority discipline.

In a Fixed Priority discipline, highest priority tasks are executed prior to lowest priority tasks. In this way, the time that a task has to wait for execution depends on the subset of higher priority tasks that are ready for execution. On the other hand, in an EDF discipline, the priority of a task changes during runtime. From experiences, it could be noted that the response time of a task i have a greater variation in EDF than in Fixed Priority because the subset of higher priority tasks changes from invocation to invocation. Thus, while in a Fixed Priority discipline the response time of a task is bounded, in an EDF discipline the response time of a task may vary from invocation to invocation and consequently the harmonic distortion introduced in a control application increases.

The management of deadlines has influence on the harmonic distortion when system is oversaturated (utilization factor greater than 1). Deadlines are missed when utilization factor is greater than 1. If a task is aborted when its deadline is missed, the computation time used to execute the task is missed too and the new invocation has to make a fresh start. If the scheduler performed a delayed completion of the task, then the next invocation of the task is skipped, freeing an execution time similar to the one used until the task missed its deadline and consequently the response time of the tasks is improved and therefore the harmonic distortion reduced.

From simulations, it can be noted that non-preemptive mechanism produces a lower harmonic distortion than a preempted one when system is oversaturated. This is explained because the utilization factor available is fairly shared among all the tasks of the system under a non-preemptive mechanism. When preemptions are allowed, the response time of lowest priority tasks is worsen as well as the harmonic distortion produced for such tasks. A non-preemptive mechanism forces completion of the lowest priority tasks when they grant the processor for execution. When system is oversaturated, the utilization factor obtained from missing deadlines and skipping invocations is fairly distributed among all the tasks of the systems when a non-preempted mechanism is implemented.

IX. Conclusions and future work

Real-time systems are used to implement controllers in control applications but there is not a straightforward relationship between real-time systems and control applications. Real-time theory deals with periodic tasks, missed deadlines and scheduling disciplines. Classical control theory applies frequency domain techniques to design and evaluate the performance of the controllers. Controllers are designed to produce a linear, time-invariant response. An inadequate implementation of the controller may cause undesirable and unpredictable consequences on the control application. However, real-time scheduling analysis cannot be utilized to evaluate the control performance of the controller implemented as a real-time task.

In this paper, we propose the utilization of frequency domain analysis to evaluate the control performance of a real-time system. It is based on measuring the harmonic distortion that the real-time tasks produce under different priority disciplines. This mechanism allows evaluating a real-time system with control parameters.

From experiences, it can be noted that priority discipline does not affect the control performance when utilization factor is lower than 0.8. However, a Fixed Priority discipline produces a lower harmonic distortion with higher utilization factors.

The harmonic distortion evaluation proposed allows evaluating the real-time performance in control application. It could be analyzed that an EDF priority discipline, optimal from a real-time point of view, is not more adequate than a Fixed Priority one when the real-time system is about or above saturation. It could be shown the differences among the ways that the available utilization factor is used when system is oversaturated with preemptive and non-preemptive mechanisms and delayed completion and aborting missed deadlines management. The harmonic distortion mechanism proposed in this paper may be utilized to evaluate the control performance of real-time systems.

References

[1] Vaccaro, Ed., *Digital Control: A State-Space Approach*. McGraw-Hill College, 1995, 480 pp. ISBN-13: 978-0070667815.

[2] C. L. Liu and J. W. Layland, *"Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment,"* Journal of the ACM, vol. 20, pp. 46-61, 1973.

[3] G. C. Buttazzo, *"Rate Monotonic vs. EDF: Judgment Day"* in 3rd International Conference on Embedded Software, Filadelfia, PA, EUA, 2003, pp. 67-83.

[4] A. Cervin, *"Integrated Control and Real-Time Scheduling,"* Department of Automatic Control, Lund Institute of Technology, 2003.

[5] G. C. Buttazzo, M. Bertogna and G. Yao, *"Limited Preemptive Scheduling for Real-Time Systems. A Survey,"* in IEEE Transactions on Industrial Informatics, vol. 9, no. 1, pp. 3-15, Feb. 2013.

[6] E. Bini and G. C. Buttazzo, *"Schedulability analysis of periodic fixed priority systems"*. IEEE Trans. Comput., vol. 53, no. 11, pp. 1462–1473.(2004)

[7] Astrom, K. J. and B. Wittenmark. *"Computer Controlled Systems"*. Prentice Hall. 1997

[8] G. Bernat, A. Burns and A. Llamosí. *"Weakly Hard Real-Time Systems"*. IEEE Transactions on Computers. Vol 50. Nro 4. pp: 308-321. April 2001

[9] Buttazzo GC, Cervin A, *"Comparative assessment and evaluation of jitter control methods"*, in International Conference on Real-Time and Network Systems, pp 163–172. (2007)

[10] Bini E, Cervin A, *"Delay-aware period assignment in control systems"*, in IEEE Real-Time Systems Symposium (RTSS), pp 291–300. (2008)

[11] Y. Wu and M. Bertogna, *"Improving task responsiveness with limited preemptions,"* in Emerging Technologies Factory Automation, 2009, pp. 1-8.