

# The Self Triggered Task Model for Real-Time Control Systems

Manel Velasco and Josep M. Fuertes  
*Automatic Control Department*  
*Universitat Politècnica de Catalunya*  
*Pau Gargallo 5, 08028 Barcelona, Spain*  
*{manel.velasco,josep.m.fuertes}@upc.es*

Pau Marti  
*Computer Science Department*  
*University of California, Santa Cruz*  
*1156 High St., Santa Cruz, CA95064, US*  
*pmarti@cs.ucsc.edu*

## Abstract

*In this paper we present an extended state-space representation for closed-loop systems that allows each control task to trigger itself optimizing computing resources and control performance. Using this model, the executing instance informs the scheduler when the next instance should be executed. The next instance execution point in time is dynamically obtained as a function of the utilization factor and control performance. Preliminary results show that control activities are able to define self-execution patterns that dynamically balance optimal levels of control performance and resource utilization.*

## 1. Introduction

In real-time control systems, the objective of control activities (to control processes) and the objective of scheduling policies (to meet deadlines) are accomplished separately. This may derive sub-optimal designs in terms of both control performance and resource utilization.

On one hand, control activities optimize control performance regardless of the computational demands of other tasks. Traditionally, controllers are designed with a constant sampling period. In terms of task execution, this means that at run-time the controller will demand a constant processing capacity. That is, the controller design does not allow increasing the execution rate of the control task to exploit available resources that may have been released by other tasks.

On the other hand, scheduling techniques optimize the use of resources regardless of the dynamics of the control application. For instance, a periodic control tasks may not require the designed execution rate (processing capacity) if the controlled plant is in equilibrium. When a plant is in equilibrium, the contribution of each control task instance execution can be considered useless. In such situation, the processing capacity wasted for those instances could be used by other tasks with higher processing demands.

To overcome these problems, we present an extended state-space model for closed-loop systems in which computing resources and control performance are jointly

considered. The model allows each control task to trigger itself: each executing instance informs the scheduler when the next instance should be executed, thus adjusting its timing constraints at run time. The next instance execution point in time is dynamically obtained as a function of the utilization factor (global parameter) and control performance (local parameter)<sup>1</sup>. Therefore, each control task acts as a co-scheduler, helping the scheduler at the scheduling decisions. Preliminary results show that control activities, at run-time, are able to define self-execution patterns that dynamically balance optimal levels of control performance and resource utilization.

## 2. State of the art

The model we present resembles the framework presented in [3]. They propose to use feedback information from the controlled plants to take scheduling decisions. Specifically, all control tasks periods are proportionally enlarged or shorted at a given time instant as a function of the utilization factor. Such mechanism does not allow the exchange of processing capacity among control tasks if the control application requires higher execution rates for specific tasks, as we do.

The later can be achieved using the elastic model [2]. In such model, the elastic coefficient of each task allows the scheduler change the task execution rate within specified ranges. The elastic coefficients are regarded as fixed parameters to be specified before run-time. The model we propose matches the elastic model if the elastic coefficient of each control task could be treated as a dynamic parameter, being a function of the resource utilization and control performance.

Some similarities may be identified between our model and event-based systems. However, in event-based systems the sampling period takes random values. Using the model we present, the variation of the sampling period is dynamically *controlled* by each control task.

---

<sup>1</sup> The utilization factor of the system is obtained taken into account all tasks in the system. Therefore, it is a global parameter and affects all tasks. The control performance is obtained by each task from the corresponding controlled plant. Therefore, it is a local parameter and affects each task.

### 3. Problem formulation

For control tasks, the task period is given by the sampling period,  $h$ , that has to balance the desired control performance and the feasible computational demand [6]. The sampling period can be selected from a range of values. Short periods allow quick reactions when the controlled system is affected by perturbations (which is positive from a control point of view), but increases the processor's load (which is negative from a resource utilization point of view). Long periods decrease the processor's load but may give poor control performance.

This situation calls for models that can dynamically accommodate different values for the control task period according to the system load and control performance. The model we present allows control tasks to vary their period. The exact value for the period (at each control task instance execution) is dynamically adjusted depending on the controlled system status and the CPU load.

### 4. Self-triggered task model

The model we propose is an extended state-space representation, which includes the controlled plant as well as information about the resource utilization. State-space models allow us to describe the future response of a system, given the present state (characterized by the state variables), the excitation inputs and the equations describing its dynamics. The extension we suggest is to incorporate the task period as a new state variable. Therefore, we will be mixing the control behavior (already represented in the original state space model) with the execution rate of the task (and the CPU load).

In the following, we develop the model using an example. The closed loop system we consider is formed by a ball and beam, which is the plant to be controlled, and a control task that has to be executed on a processor and has to control plant. The ball and beam system has a motor that balances a beam in order to keep the ball (that can rotate freely along the beam) in the desired beam position. The objective of the controller is to actuate on the motor to locate the ball in the desired position. To do so, at each sampling time, the controller takes the value of the position of the ball and the angle of the beam and generates the new angle for the beam that derives in the corresponding actuation on the motor.

#### 4.1. Original model

A linear discrete-time invariant state-space model [1] of the ball and beam is given by equation (1), where  $x_k$  and  $y_k$  (which are the state variables) represent the position of the ball and the beam angle at the  $k$  sampling instant. The first matrix ( $2 \times 2$  dimension), called system

matrix, describes the dynamics of the ball and beam. The second matrix ( $2 \times 1$  dimensions), called input coefficient matrix, links the inputs  $U$  (provided by the controller if closed-loop operation) with the system dynamics. In both matrices,  $h$  is the sampling period.  $U$  is the available vector of inputs; in our case it is the tension ( $1 \times 1$  dimensions) that we provide to the motor. The input can adopt positive and negative values, allowing the motor of the beam to rotate to both sides.

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_k \\ y_k \end{bmatrix} + \begin{bmatrix} \frac{h^2}{2} \\ 2 \cdot h \end{bmatrix} U \quad (1)$$

Note that in the state space representation of the ball and beam, at each sampling instant  $x_k$  and  $y_k$  vary according to the system dynamics and the input. However,  $h$ , the sampling period, which appears on the matrices as a result of the discretization process of a continuous-time model, has a constant value that has been chosen at the controller design stage. Therefore,  $h$  has nothing to do with the system state, although it influences its dynamics.

Recall that the state of the system can be directly related to control performance. For instance, a simple rule could be *the smaller the norm of the state vector, the better the controlled system performance*. In terms of the ball and beam: the smaller is the deviation of the beam with respect to the horizontal position and the smaller is the distance of the ball with respect to the desired location, the better the performance.

#### 4.2. First model modification

We want a model able to accommodate different values for the sampling period (i.e., the task period). To do so, we extend the state representation of the system with a new state variable, the task period,  $h_k$ , as represented in (2)<sup>2</sup>. Note that for the system in equation (2) a new control law giving the appropriate sequence of values for the input  $U$  is needed.

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ h_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & h_{k+1} & 0 \\ 0 & 1 & 0 \\ \alpha & \beta & \omega \end{bmatrix} \cdot \begin{bmatrix} x_k \\ y_k \\ h_k \end{bmatrix} + \begin{bmatrix} \frac{h_{k+1}^2}{2} \\ 2h_{k+1} \\ 0 \end{bmatrix} \cdot U \quad (2)$$

In (2), at each task instance execution, the task period will be changed according to the state of the system given

<sup>2</sup> Note that  $h_{k+1}$  (and not  $h_k$ ) appears inside of the system and input matrices. This is due to the solution of the system equations. In the non-extended model, the  $h$  of the system and input matrices has no index ( $k+1$  or  $k$ ) because it is constant ( $h$  at the  $k$  instant and  $h$  at the  $k+1$  instant have the same value). In the present model, since  $h_k$  is a system variable that varies from instance execution to instance execution, it is necessary to distinguish which is the appropriate  $k$  index.

by  $x_k$ ,  $y_k$  and the new state variable  $h_k$ . The dependency of this new variable with the others system variables is given by parameters  $\alpha$ ,  $\beta$  and  $\omega$ . Let's discuss some properties of the extended model, depending on values of  $\alpha$ ,  $\beta$  and  $\omega$ :

- If  $\alpha, \beta = 0$  and  $\omega = 1$ , then, for each  $k$ ,  $h_{k+1} = h_k$ , and the system may be considered as the original (1). The control law that will give the sequence of inputs  $U$  can be obtained by classic controller design methods.
- If  $\alpha, \beta = 0$  and  $0 < \omega < 1$ , the sampling period will be decreased at each instance execution, tending to 0, thus leading to a system that can not be implemented.
- If  $\alpha, \beta = 0$  and  $\omega \geq 1$ , the sampling period will be increased at each instance execution, tending to  $\infty$ , thus violating the Shannon's sampling theorem.
- If  $\alpha, \beta \neq 0$  and zero and  $0 < \omega < 1$ , we have a system with a variable period, each one depending on the previous system state. In particular, each  $h_{k+1}$  value for the next task period depends on the previous one ( $\omega h_k$ ), with smooth transitions in period variations. In addition, the state space model becomes nonlinear. Therefore, finding the adequate control law giving the appropriate sequence of inputs  $U$  will be a more difficult task (see for example [4]), if feasible.
- If  $\alpha, \beta \neq 0$  and  $\omega = 0$ , we get a variable period system depending only on the original state variables. Consequently, the more quickly these variables move (angle and position), the faster the period changes, thus losing the smooth transitions found in the previous case. This may result in values for the sampling periods out of the permissible ranges.
- If  $\alpha, \beta \neq 0$  and  $\omega \geq 1$ , the evolution of the system will depend on the specific chosen values for  $\alpha$ ,  $\beta$  and  $\omega$ , which require a deeper analysis, out of the scope of this paper.

From the model given by (2), three elements should be highlighted. First, the system is nonlinear. Second, it would be possible to obtain negative values for the task period. Considering only a theoretical view, this possibility means that the system should return to the past in order to modify already taken decisions. But this is clearly non-programmable. We could solve this problem by using the absolute value for the  $h$  in (2). This will guarantee that  $h$  will be always positive. Finally, it must be stressed that the system matrix in (2) includes an  $h_{k+1}$  at the  $k$  instant, which is an inconsistency. This can be solved by substituting the  $h_{k+1}$  value for the expression  $h_{k+1} = \alpha x_k + \beta y_k + \omega h_k$ , which is known at the  $k$  instant.

### 4.3. Second model modification

Looking at the final model obtained in the previous section, two problems, apart from having a nonlinear

model, can be identified. First, the absolute value would increase the model complexity, because it implies using two symmetric models, one for positive values of  $h$  and the other for negative values. Second, the  $h$  may take unbounded values, due to the linear relation between  $h$  and the original state variables. For example, if the state variables take huge values,  $h$  will quickly increase.

To solve the previous problems, we suggest limiting the possible  $h$  values by introducing an appropriate function of the state variables, called *h-function* (instead of having a simple linear relation). In addition, taking advantage of the  $h$ -function, we incorporate the utilization factor in the model. Up to now, the model only related the varying period of the task with the original state variables (as a measure of control performance). In this new extension given by (3), we will relate the period variation to the control performance and to the processing capacity.

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ h_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + y_k \cdot f(x_k, y_k, h_k, \zeta) \\ y_k \\ f(x_k, y_k, h_k, \zeta) \end{bmatrix} + \begin{bmatrix} \frac{f(x_k, y_k, h_k, \zeta)^2}{2} \\ 2f(x_k, y_k, h_k, \zeta) \\ 0 \end{bmatrix} \cdot U \quad (3)$$

In (3),  $\zeta$  represents the utilization factor of the processor at the  $k$  instant and the  $h$ -function is given by  $f(\cdot)$ . In (3),  $h_{k+1}$  is obtained by an appropriate function of the state of the controlled system and the CPU load. Note that the goal of the  $h$ -function is to allow the task period to take values from a bounded range. The  $h$ -function determines how  $h$  changes at each moment. For instance, the same system with two different  $h$ -functions may result in very different behaviors in terms of CPU load and control performance. Note that choosing a specific  $h$ -function could facilitate system schedulability as well as improve control performance.

It is important to point out that for the state space model given by (3), the analysis and design of a control law can be a complex task. However, it is possible to design control laws that guarantee the complete stability of the system around a desired working point. These techniques range from the system linearization [4] to the complex techniques of feedback linearization [7].

### 4.4. Selection of the h-function

A natural way for selecting the  $h$ -function is to translate into a mathematical function the following desired rule: as the controlled system gets closer to the desired working point (equilibrium), the period should be as large as possible, within the Shannon limit (recall discussion of section III). If a perturbation affects the system, bringing it away from the equilibrium point, the period should be decreased (to improve control

performance) taking into account the available processing capacity. Mathematically, this can be accomplished by the h-function given by (4)

$$h_{k+1} = f(x_k, y_k, h_k, \zeta_k) = e^{-(x_k^2 + y_k^2)} g(\zeta_k) \quad (4)$$

Note that (4) has a negative exponential shape. The first part of (4) is the exponential function, which depends on the original state variables. It allows each value to smoothly vary from an upper limit to a lower limit, if the exponent of the exponential function is kept positive. That's why we suggest putting the square exponents over  $x_k$  and  $y_k$ . Note that the original state variables already are a measure of control performance. Otherwise, the exponent should include the operation needed to measure the controlled system performance. The second part of (4), the function  $g(\zeta_k)$ , allows correcting the next value of  $h$  taking into account the processor's utilization factor.

## 5. Simulation results

Using the extended model, at each control task instance execution, the period selection is in consonance with all the elements that are involved in the control of the plant and in the scheduling of the task set, thus facilitating the optimization of the whole system in terms of both control performance and resource utilization. The main goal of the presented model is that if several tasks are driven according to this model, the processing capacity can be dynamically balanced among them according to the measured controlled performance, as shown next.

In Figure 1 we show the execution pattern of two *ball and beam* tasks. The control laws implemented in the two tasks have been calculated by means of linearization techniques. In Figure 1 we draw four lines. The two upper ones correspond to the sequences of task periods (low levels mean shorter task periods) for each control task, and the lower ones correspond to the dynamics of each controlled plant. The utilization factor is injected as a simulation variable.

At the beginning (left side of the figure), both systems are stable, so both tasks have the same value for the period. When a perturbation affects system1, it brings away the plant from the desired working point. This fact causes an immediate decrease of the task period controlling system 1 (to improve control performance) and an increase of the task period controlling 2 (to optimize resource utilization). Therefore, the exchange of the processing capacity among the two control tasks has taken place. Once system 1 is in equilibrium (before the perturbation arrival over system 2), task1 and task2 have again the same period. A similar processing capacity exchange occurs when system2 suffers a perturbation, but in inverse direction.

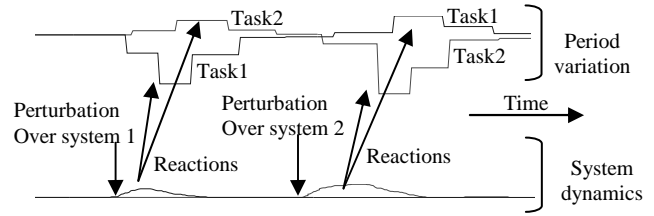


Figure 1. Task periods adjustment according to the system dynamics

## 6. Conclusions

In this paper we have presented the self-triggered task model that drives control task executions according to controlled system performance and available processing capacity. The model, which extends the original state space representation of a controlled plant with the control task period, allows control task to adjust their execution rate, acting as a co-scheduler. The main research issues that derive from this work is the analysis and design of the controller as well as the scheduling techniques that can support such type of self-triggering control tasks.

## Acknowledgments

This work has received support from the Spanish Ministerio de Ciencia y Tecnología Project ref. DPI2002-01621.

## References

- [1] K.J. Åström and B. Wittenmark, *Computer Controlled Systems. Third Ed.*, Prentice Hall, 1997.
- [2] G. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni, "Elastic Scheduling for Flexible Workload Management". *IEEE Trans. Computers*, 51:3, 2002
- [3] A. Cervin, J. Eker, B. Bernhardsson, K.-E. Årzén "Feedback-Feedforward Scheduling of Control Tasks", *Real-Time Systems*, 23:1, 2002.
- [4] A. Isidori, *Nonlinear Control Systems*. Springer Verlag, New York, 1989.
- [5] D.J. Leith, R.N. Shorten, W.E. Leithead, O. Mason, and P. Curran "Issues in the design of switched linear control system: A benchmark study". *Int. Journal of Adaptive Control and Signal Processing*. 2003; 17:103-108
- [6] P. Marti, G. Fohler, K. Ramamritham, and J.M. Fustes, "Improving Quality-of-Control using Flexible Timing Constraints: Metric and Scheduling Issues." *Real-Time Systems Symposium*, Dec. 2002.
- [7] H. Nijmeier, *Nonlinear Dynamical Control Systems*, Springer-Verlag, 1990.