Abstract

Digital control uses real-time systems to implement its control strategies. However, an inadequate scheduling strategy, even schedulable from a real-time point of view, may produce undesirable control consequences on the application. In this paper, we propose a scheduling mechanism based on a Dual-Priority scheme ([6]) to reduce the adverse control effects that jitter produces in an automatic digital control system. On the other hand, with the use of Dual-Priority scheme, the response time of non-control tasks in real-time systems is improved. The mechanism proposed promotes the control tasks of the system according to its current level of jitter sensitiveness. We show that with the mechanism proposed the performance of both control and non-control task is improved.

Keywords: Control Application, Real-Time, Dual-Priority.
1. Introduction

Digital control uses computers to implement control systems. A controlled system consists generally of a plant and a controller. The plant is the system wanted to control and the basic operation of a control task is to read information from multiple sensors located in the plant, calculates the output to be performed to the plant to achieve some desired control performance and send the results to actuators.

A typical control system is made up of many of these control activities in addition to other non-control operations (such as user interaction, health monitoring, among others). These operations need to be performed periodically and within strict timing constraints in order to guarantee that the system stays under control within the required performance requirements.

Modeling techniques are applied in order to get a mathematical expression of the control features of the application. From the control model of the application a controller can be design to get a controlled behavior of the application.

The real behavior of the application can differ from the expected one due to practical implementation issues such as sensor accuracy, actuators sensitiveness and range, inputs and outputs clamping (bounded), mechanical and computation precision, environment conditions, and so on. Control strategies should be designed to be robust enough in order to avoid these side effects.

A control system is a real-time system. Multiple concurrent activities need to be performed within well-defined deadlines, however there are important discrepancies between the assumptions and the computational models used in both disciplines. The most obvious one is the interpretation of period and deadline. In real-time systems the notion of a periodic task implies a computation that is started at $kT$ units, where $T$ is the period and $k=0,1,2,...$, and has to be completed before $D$ units after it has been started (in most cases $D$ is considered to be equal to the period). In control theory, “periodicity” means a computation that is performed with a separation of exactly $T$ units. The real-time model does not consider the fact that the starting time is essential from the control theory perspective. In the ideal case, it is assumed that sensing, computation and actuation take no time and therefore the control task is executed at precisely $kT$ time instants. Another example is the fact that control systems are robust and can tolerate occasional misses of deadlines, however such concept of missed deadlines does not map well in the real-time domain.

The time a control task has to wait to be executed depends on the computation time required by higher priority tasks ready to be executed. Because the pattern of releases is not fixed, this results in a variable interference and therefore a variable response time of the task. These time variations, named jitter, are not modeled by the control theory and consequently they may produce undesirable effects in the control application.

The trivial solution to reduce the jitter is to assign the highest priorities to the control tasks. However, as the real-time time system should execute more than one control task as well as other more important tasks (such as a safety guard task) it is not feasible. Just the highest priority task will get a low jitter whilst the lowest priority tasks will not get an adequate performance.

Several papers have analyzed the effects that the real-time scheduling police produce on a control application. In [1], it is proposed a priority assignment algorithm that takes into account the control features of the application to maximize the control performance. The Control Server is presented in [5] for the implementation of control tasks in flexible real-time systems. In [9], it is proposed a mechanism that modifies the control strategy during runtime in order compensate, but not eliminate the effects of the jitter. In [3], it is proposed a control task model that splits a control task into three subtasks with different real-time features. In [7], it is presented a feedback scheduling strategy for multiple control tasks that optimizes the overall control performance, subject to constraints on the total systems workload. All these papers propose different task models to reduce the jitter but none of them try to improve the response time of the non-control task.

In this paper we propose a dynamic priority mechanism that reduces the adverse effects of the jitter in control systems. Because the consequences of the jitter are not the same during the lifetime of the system, the proposed mechanism, Jitter-Aware Scheduler, reacts to the changes on the workload conditions according to how sensitive is the control application to the jitter under its current circumstances. We analyze the range of both input sensors and output actuators in order to set when a task is more sensitive to the jitter. In Section 4, it is described how to detect when tasks are more sensitive to jitter. In this way, we can assign the processor to the task most sensitive to jitter. The performance of Jitter-Aware Scheduler is compared to the performance of Fixed Priority (FP) and Earliest Deadline First (EDF).

The case study is based on the classical inverted pendulum, using TrueTime ([8]) as simulation tool. We consider in the simulations all the practical effects of the real implementation (sensor accuracy,
input and output jitter, sampling effect, sensor and actuator precision, output clamping, conversion quantization etc.) in such a way that the simulations perform in the same way that the real implemented pendulums.

The inverted pendulum is used in control research because it illustrates ideas in linear control such as stabilization of unstable systems ([2], [11]). Control strategies designed using the inverted pendulum can be applied to applications that present similar control features found in the most diverse areas such as hydraulics, mechanics, robotics, thermodynamics and electronics. Therefore, results obtained considering the inverted pendulum are applied to a great deal of control areas.

This paper is organized as follows: Section 2 describes brief concepts of the theory of control with the aim of stating the temporal requirements imposed by the control theory. The main concepts of real-time systems are described in Section 3. Jitter-Aware Scheduler, the scheduling mechanism proposed in this paper, is detailed in Section 4. In Section 5, we present the inverted pendulum used in this paper as the case study. In Section 6, we evaluate the performance of Jitter-Aware Scheduler through simulations. Analyses of results are presented in Section 7. Conclusions are drawn in Section 8.

2. Brief Control Concepts

In this Section, we describe briefly the main concepts of control modeling. From this description, a closer relationship will be established with the temporal features of a real-time system. We describe the theoretical and practical issues to take into account when a control application is implemented.

When a controller is implemented on a computer system, a discrete-time model should be used. Figure 1 shows a diagram of a discrete time control of a continuous-time plant. In this figure, the measured feedback signal is simply the system output $y(t)$. The input $r(t)$ is the reference input and $e(t)$ is the error signal.

![Figure 1. Discrete-time control system.](image)

$C(z)$ is implemented as software and it is executed by a processor. It is defined according to the control features desired for the closed-loop such as stability, step response, percent overshoot, settling time, steady-state error and so on.

The state-space model of a discrete-time system is expressed by difference equations:

$$x[k+1] = \Phi \cdot x[k] + \Gamma \cdot u[k]$$  

(1)

$$y[k] = c \cdot x[k] + d \cdot u[k]$$  

(2)

with

$$\Phi = e^{A\tau}, \Gamma = \int_0^\tau e^{A\tau} \cdot B \cdot d\tau$$  

(3)

where $A$ is the system matrix, $B$ is the input matrix, $c$ is the output matrix, $d$ is the feedthrough matrix, $u[k]$ is the input variable, $y[k]$ is the output variable and $T$ is the sampling interval.

Varying the sampling interval along the time, the discrete-time model is not valid anymore, and consequently the same actions will produce different effects. A variable delay during outputs updates will produce an effect that may not be modeled and consequently it could lead to an undesirable behavior.

The selection of the sampling period is important. If the sampling period is chosen too long, the continuous-time signal will be not 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 ([12]).

2.1. Robustness and Perturbation Sensitiveness
When control systems are implemented, their behavior may differ from the one expected from the model because of several practical issues. Some of these factors may be:

- Variations of systems parameters for different implementations. Mechanical tolerances as well as quality of manufacturing of the product may lead to different features of the final implementation. It also has to take into account that mechanical properties as well as dynamic features of the system may change during its lifetime.

- Precision and hysteresis of sensors and actuators. Real sensors and actuators cannot respond to small variations in its inputs because of precision, sensitiveness or hysteresis properties. Usually, better precision and sensitiveness means more expensive sensors and actuators.

- Quantization errors of converters. A/D as well D/A converters have quantization errors because of the finite length of their binary representation.

- Finite dynamic range of sensors and actuators. The response of either a sensor or an actuator is bounded in a certain range. Larger dynamic range devices could lead to unsafe work conditions (for instance, whether intrinsically safe procedure has to be satisfied) as well as more expensive implementations.

Most of these factors may be reduced by increasing the hardware used (and consequently the cost of the system) but always should be taken into account on the robustness of the system, even when the performance could not be the optimal one.

The influence of these factors on the control performance should be considered altogether: there is not point on improving one of them without taking care on the others.

In this paper, we analyze the real-time system and the control application altogether. For instance, it has no sense to execute a control task with the highest priority if we can foresee that the output of the controller will be bounded according to the characteristics of the actuator used. In that case, it could be suitable to assign the highest priority to a control task more sensitive to the jitter.

3. Basic Concepts on Real-Time System Scheduling

The general process model of a real-time system consists of a set $\Pi$ of $N$ periodic and non-periodic tasks, $\Pi = \{ \tau_i = \{ T_i, D_i, C_i \} \}_{1 \leq i \leq N}$. Each task, $\tau_i$, is characterized by either its period in case of periodic tasks or minimum interarrival time for non-periodic ones, $T_i$, deadline, $D_i$ and worst-case execution time, $C_i$. Every time that a task requires the processor to be executed, it is said that the task is invoked.

The notion of jitter in real-time is important for our discussion. Sampling jitter is the maximum difference between the exact sampling periods 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 ([4]).

The time a task has to wait to be executed depends on the computation time required by higher priority tasks which are ready to be executed. Because the pattern of releases is not fixed, this results in a variable interference that may produce undesirable effects on the control system.

A real-time system includes several types of tasks different from the control ones. There exists safety as well as secondary functions (e.g. man-machine-interface and communication tasks) that should be executed by the real-time system. For instance, a safety task could check whether the arm of a pendulum is stuck or an unknown object is inside its working area in order to shutdown the power supplied and put the arm into a low energy state.

Scheduling mechanisms intending to reduce the jitter of the control tasks can worsen either the schedulability of the real-time system or the performance of the other tasks of the system.

In the next Section, we describe the main concepts of the dual-priority scheduler proposed in [6] to improve the performance of the non-real-time tasks in real-time systems. In [10], the dual-priority scheduling algorithm was used to improve the power consumption on a hard real-time system. We use the dual-priority scheduler in real-time control applications.
3.1. Dual-Priority Scheduler

The dual-priority scheduler was proposed in [6] to improve the response time of non-real-time tasks in real-time systems. It is based on three priority bands: Upper, Middle and Lower. Tasks assigned to the same band are schedule according a certain priority discipline.

When no task is ready at the upper priority band, then the highest priority task ready for execution at the middle band is executed. If there is not any ready task at both upper and middle priority bands, then the highest priority task ready for execution in the lower band is executed.

In a dual-priority mechanism, a task can be promoted from its priority band to a higher priority band during runtime in order to meet its real-time requirements.

Davis et. al. proposed the following assigned criteria: non-real-time tasks are assigned to the middle band while the real-time tasks are initially assigned to the lower band and they are promoted to the upper band when exists a risk to miss a deadline.

Tasks in the upper band are schedule under a fixed priority discipline. The priority disciplines used in both middle and lower bands can be chosen to improve any arbitrary performance criteria.

4. Jitter-Aware Real-Time Scheduler

In this Section we describe the scheduling mechanism proposed to improve the performance of the control tasks of a real-time system at the same time that the response time of non-real-time task is improved as well.

A control task performs a certain function of the application. It implements a control strategy in order to get a controlled behavior of the application. Under a change in the inputs of the controller it calculates the control function in order to update the outputs to the actuator.

The control strategy should be executed periodically. When a system is in a stable state, the changes in the inputs are almost neglected and consequently the outputs remain in the same value. In this case, reducing the jitter of the control task will not produce any effect on the control performance. On the contrary, when the inputs of the system change, the outputs should be applied as soon as possible with the strict time constraint that the control modeling techniques specify.

Control strategies are linear functions whose outputs are applied to actuators with finite dynamic range. Consequently, we can assure there is not point to reduce the jitter of a control task whose output is oversaturated.

Because of the precision, sensitiveness and hysteresis of both, sensors and actuators, it has no sense to reduce the jitter of the control tasks if its performance is not improved in an equivalent magnitude. For example, if the precision of the actuator is +/- .05V then it is not worth it to increase the timing precision of the system to get at most a precision of .0005V.

In this paper we proposed a dynamic scheduling mechanism based on a dual-priority scheme ([6]) that changes the priority of the control tasks according to its sensitiveness to the jitter. The control tasks can be assigned either to the lower band as well as the upper band. In the lower band, theses are schedule using a RMA priority assignment. When the scheduler detects that a control task is sensitive to the jitter, then it promotes that task from the lower band to the upper band. When the control task is not sensitive to the jitter anymore then it is assigned to the lower band again. Safety tasks are assigned to the upper band in order to assure a safety behavior of the application. The middle band of the scheduler is left to execute the non-control tasks to maximize its response time.

The criterion the scheduler applies to promote the priority of control tasks is:

- The variation of the input in the last invocation. A threshold is defined for the variation of the input between two consecutive invocations to determine whether the control task is sensitive to the jitter or not. If the input varies in a neglected quantity with respect to the precision of the sensor, the control task is not sensitive to the jitter and consequently it is not promoted.

- The variation of the output in the last invocation. A threshold is defined for the variation of the output from invocation to invocation to determine whether the control task is sensitive to jitter or not. If the output of the control strategy varies in a neglected amount with respect to the precision of the actuator, the control task is not sensitive to the jitter and consequently it is not promoted.
• The value of both the inputs and the outputs in the last invocation. When either inputs or outputs values are out of the range of the sensor or actuators, then the effect of the jitter will not be noticeable. For instance, the Figure 2 shows the output calculated by the control strategy describes in the case study in the next Section, and the voltage applied to the actuator. It can be noted that when the control strategy produces outputs out of the range of the actuator, the task will not be affected by the jitter, because the output remains in the same value. In this case, we say that the task is not sensitive to the jitter and consequently it is not promoted. It should be remarked that robust control allows control designers to consider the clamping of the outputs into the control strategy ([12]).

• In the other cases we consider that the control task is sensitive to the jitter and therefore it is promoted.

We can note that the scheduler uses the data obtained from the last invocation of the control task to set the priority of the next invocation. Consequently, the reaction of the scheduler could be delayed at most a sampling interval. However, since the control strategy is executed with a frequency higher than the bandwidth of the control application, this constant delay can be either modeled or neglected ([12]).

A more strict analysis of the control robustness, require a deep knowledge of the sensitiveness of the controller to variation in its parameters.

From the generic state-space model we can express the state-space equation of both the input and output controller considering all the parameters such as temperature, humidity, viscosity, noise, friction and so on as:

\[ u[k] = f(x[k], p_1, p_2, \ldots, p_n) \]
\[ y[k] = g(x[k], p_1, p_2, \ldots, p_n) \]  

(4)

The sensibility of these functions with respect to each parameter is given by:

\[ S_{u_i}^n = \frac{\partial u[k]}{\partial p_i} \frac{p_i}{u[k]}, \ldots, S_{u_o}^n = \frac{\partial u[k]}{\partial p_o} \frac{p_o}{u[k]} \]
\[ S_{y_i}^n = \frac{\partial y[k]}{\partial p_i} \frac{p_i}{y[k]}, \ldots, S_{y_o}^n = \frac{\partial y[k]}{\partial p_o} \frac{p_o}{y[k]} \]  

(5)

On the other hand, we can express the variation of the input in the last invocation as \( \Delta u = u_k - u_{k-1} \) and the variation of the output in the last invocation as \( \Delta y = y_k - y_{k-1} \). We define the threshold for the \( \Delta u \) as the maximum sensibility of the sensor, named \( \hat{S}_u \) and the threshold for the \( \Delta y \) as the maximum sensibility of the actuator, named \( \hat{S}_y \).

We define the threshold for the \( \Delta u \) as the maximum sensibility of the sensor, named \( \hat{S}_u \) and the threshold for the \( \Delta y \) as the maximum sensibility of the actuator, named \( \hat{S}_y \).

\[ \hat{S}_u = \max_{j=1 \ldots n} \{ S_{u_j}^n \} \]
\[ \hat{S}_y = \max_{j=1 \ldots n} \{ S_{y_j}^n \} \]

From the manufacturer, we can obtain various parameters of both sensors and actuators. Two of the most important ones are the precision and the operating range of sensor and actuators, denoted \( \varepsilon_s \) and \( \varepsilon_a \) (precision of sensor and actuator respectively) and \( \eta_s \) and \( \eta_a \) (operating range of sensor and actuator respectively).

The scheduler does not promote a control task if at least one of these equations holds:
\[ \Delta u \leq \lambda_1 e \hat{s}_1, \quad \Delta y \leq \lambda_o e \hat{s}_o, \]
\[ -\eta_1 \leq \Delta u \leq \eta_1, \quad -\eta_o \leq \Delta y \leq \eta_o \]

where \( \lambda_1 \) and \( \lambda_o \) are proportionality constants. We can conclude that if some of these conditions hold, the control task is not promoted to the upper. Otherwise the control task is promoted to the upper band or is maintained executing in this band.

5. A Case Study: The Inverted Pendulum

In this Section, we describe the features of the inverted pendulums we used to evaluate the Jitter-Aware Scheduler. As it was mentioned in the introduction, the inverted pendulum is used in control since present similar control features were found in areas such as hydraulics, mechanics, robotics, thermodynamics and electronics.

For sake of simplicity, we just describe one of the 5 pendulums we used. The inverted pendulum 1 consists of a wood arm of 50 cm length with a mass of 20.67 g. In one end there is an extra mass of 50.65 g and it is attached to the shaft of a DC motor by the other end. The motor has a torque of \( 4.369 \times 10^{-6} \) N. The voltage applied to the motor ranges from -5 to +5 DC Volts. The torque of the motor is proportional to the voltage. When the voltage is negative, the torque is counterclockwise whilst it is clockwise when the voltage is positive.

Because of the torque of the motor is not enough to take the arm from the down position to the upright position at once, a lifting control strategy should swing the arm in order to increase its energy in each oscillation. Moreover, the lifting control strategy should leave the arm in the upright control zone with angular velocity near null, because otherwise the torque of the motor could be not enough to stop it and consequently the arm will turn round.

The lifting controller for the inverted pendulum 1 changes the motor voltage according to the following PD action:

\[ y_1 = -55 \cdot \alpha_1 - 11 \cdot \frac{d\alpha_1}{dt} \]  

where \( \alpha_1 \) is the angle from the upright position of the inverted pendulum 1, and \( d\alpha_1/dt \) is its angular velocity.

An upright control strategy holds the arm in the upright position and takes over when the arm is inside the angular zone between -0.14 radians and +0.14 radians (i.e. 8° approximately) from the upright position.

Equation 7 can be expressed in a state-space form using the state-space variables \( x_1 \) and \( x_2 \):

\[ x_1 = \alpha_1, \quad x_2 = \frac{d\alpha_1}{dt} \]

Therefore, the PD control strategy for the inverted pendulum 1 is given by the following state-space equation:

\[ y_1 = -55 \cdot x_1 - 11 \cdot x_2 \]  

Because of the dynamic range of the motor voltage is bounded, the voltage given by both equation is clamped between -5 Volts and +5 Volts.

The angle of the arm is measured with a slotted encoder with a resolution of \( \pi/2200 \) radians. The angular velocity is indirectly obtained by dividing the difference between two consecutive angle sampling by the sampling period. Consequently, the discrete-time form of Equation 9 turns to:

\[ y_1[k] = -55 \cdot x_1[k] - 11 \cdot (x_1[k] - x_1[k-1])/T_i \]

where \( T_i \) is the control task period of the inverted pendulum 1.

The PD control strategies for the other inverted pendulum are obtained in a similar way:
The coefficients of the PD controllers were designed applying the root locus method in such a way that the settling time is minimized.

These PD control strategies were implemented in Section 6 in a TrueTime/Simulink simulation environment in order to evaluate the behavior of the pendulums under different real-time scheduling conditions.

6. Experimental Set-Up

In this Section, we evaluate the performance of Jitter-Aware Scheduler. We generate randomly a set of real-time systems. Each system consists of 5 control tasks (in this case the control tasks are the ones developed in the previous section to control the five pendulums), and 5 non-control tasks. Ten systems were generated for each utilization factor ranged from 0.1 to 1 in step of 0.1. The control tasks period was bounded between 50ms and 80ms because it is the interval in which the best control performance were gotten. The non-control tasks period was bounded between 5ms and 200ms.

We simulated each of these systems using TrueTime, taking in account all the practical effects of the real implementation. TrueTime is a set of tools designed by Cevin et al. in [8] to simulate real-time control applications in a Matlab/Simulink environment. TrueTime lets us simulate the behavior of real-time kernels in a control system and analyze the effects of the priority discipline over the control application. Different real-time priority disciplines can be defined in C, Matlab functions or Simulink blocks. Figure 3 shows the scheme implemented with TrueTime.

The TrueTime kernel outputs the 5 control actions through the digital-to-analog (D/A) converters forwarded to each pendulum. The output of each pendulum is feedbacked to the TrueTime kernel through the analog-to-digital converter to calculate the next control action.

Because of their control dynamics ([12]), the pendulum 1 was assigned with the highest priority whilst the pendulum 5 with the lowest priority when they are promoted to the upper band.

Figure 4 show the average of the settling times gotten for each discipline. The settling time was chosen as the figure of comparison because the upright control strategy is very sensitive to the jitter. The upright control strategy swings the arm of the pendulum in order to increment the energy of the arm in each oscillation, ([2]). The energy should be transferred according to the natural oscillation of the pendulum, otherwise the energy would not be accumulated properly and the result could be unpredictable (noise,
vibration, heat). When the upright control strategy is affected by the jitter, the energy could not be transferred to the pendulum adequately.

Figure 4. Settling time of each pendulum for each priority discipline.

7. Analysis of Results

We could note and confirm through simulation, that a jitter greater than a 20% was not tolerable for the upright strategy letting the pendulums in an oscillating state and they never reach the upright position. We could also note that the performance of our mechanism is much better than Fixed Priority and EDF for each one of the pendulums of the application.

Performance of pendulums 1 and 2 is similar for the three scheduling mechanisms. This happens because the control tasks of theses pendulums are assigned with the highest priority most of the time. In both EDF and FP, control task of pendulums 1 and 2 are executed with low jitter when the total utilization factor is low.

We could also observe that control tasks needed to be promoted to the upper band just the 15% of the invocations. In this way, the real-time performance of the non-control tasks is improved. Figure 5 shows the average response time according to the sampling period of the non-control tasks for each utilization factor obtained with each discipline.
We observe that with the Jitter-Aware Scheduler the non-control tasks improve their response time considerably while the utilization factor increases.

We get the settling time of each pendulum, because it is the only control application that the real-time system runs. Results of the settling time for each pendulum are shown on Table 1. The settling time of each pendulum depends on the control features of each of them related mostly to their constructive characteristics.

<table>
<thead>
<tr>
<th>Pendulum</th>
<th>Settling Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>16.91</td>
</tr>
<tr>
<td>4</td>
<td>25.21</td>
</tr>
<tr>
<td>5</td>
<td>25.3</td>
</tr>
</tbody>
</table>

As we observe, the values obtained with Jitter-Aware Scheduler are closer to the values of Table 1.

8. Conclusions

In this paper we proposed a priority mechanism to reduce the adverse effect that real-time scheduling produces on the control applications. The mechanism is based on promoting the control tasks to the upper priority band according to the sensitiveness that such tasks present to jitter during runtime.

We evaluated its performance using TrueTime in a Matlab/Simulink environment and comparing it with the ones obtained considering Fixed Priority and EDF. We show that our mechanism improves the control performance of the application. At the same time, the non-control tasks reduce its response times.

On the other hand, we could check in many cases, EDF and FP are not able to stabilize some pendulums even though the real-time systems were schedulable whilst the Jitter-Aware Scheduler could.

References


