Sampling period dependent RST controller used in control/scheduling co-design
David Robert, Olivier Sename, Daniel Simon

To cite this version:
David Robert, Olivier Sename, Daniel Simon. Sampling period dependent RST controller used in control/scheduling co-design. 16th IFAC 2005 World Conference, IFAC, Jul 2005, Prague. inria-00000735

HAL Id: inria-00000735
https://inria.hal.science/inria-00000735
Submitted on 15 Nov 2005

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Abstract: The paper presents a sampling period dependent RST controller used in the context of control/scheduling co-design. This plant controller is implemented on a real-time operating system as a control task. A scheduling controller adapts control task periods to regulate the processor load. The link established between scheduling and plant control provides more flexibility and robustness w.r.t. the variation of the processor load. Parameterized RST and scheduling controller synthesis are presented and a co-design example illustrates the approach flexibility.

Keywords: Computer controlled systems, digital control, RST controller, feedback scheduling

1. INTRODUCTION

Digital control systems are often implemented as a set of tasks running on top off a real-time operating system. Control tasks are generally viewed by the scheduling community as hard real-time tasks with fixed sampling periods and known "worst case execution time" (WCET). On the other hand control community supposes that periods are constants and can not be changed on-line. Theses assumptions are often too restrictive. For instance many control laws can tolerate small variations on the sampling period without leading to instability. On the other hand WCET based scheduling of control laws with varying execution time leads to an under utilization of the processor unit. Moreover, practical estimation of the WCET is a difficult and time consuming work. Finally embedded applications with complex control laws require flexibility to allocate computational resources on the fly.

In this paper a feedback scheduler regulates the processor utilization to avoid overload. It acts on the task periods and measures task loads. Plant control laws, computed by the tasks, should fit the sampling period variations, therefore a sampling period dependant RST controller is proposed. The link established between scheduling and plant control provides more flexibility and robustness w.r.t. the variation of the processor load.

Control and scheduling co-design is a recent interest in both the computer and control communities. A first approach uses off-line co-design. In (Seto et al., 1996) optimal periods which maximize the control law performances w.r.t. the resource constraints are obtained by solving an optimization problem. In (Ryu and Hong, 1998)
control law performances depend on the periods as well as on input/output latencies and an heuristic algorithm computes the optimal periods. In (Palopoli et al., 2002) an optimization problem is solved to find optimal task periods and feedback gains. The previous approaches do not bring flexibility therefore on-line adaptation was studied. In (Cervin and Eker, 2000) a feedback controller with a sampling period dependant PID controller is used. In (Cervin et al., 2002; Eker et al., 2000) rescaling factors obtained by off-line optimization preserve the optimality of a set of control task periods. In (Caccamo et al., 2000; Sha et al., 2000) a modified Constant Bandwidth Server adapts task periods to locally handle an overrun when the execution time varies. In (Cervin and Eker, 2003) a framework, based on a Constant Bandwidth Server, is used to enhance the determinism of the co-design. In (Sename et al., 2003) the authors have developed a LQ controller to regulate the plant control tasks, the latter should fit the utilization and inputs are control task frequencies.

As far as the adaptation of the control tasks is concerned, the load of the other tasks is seen as an output disturbance.

The scheduling is here limited to periodic tasks. In this case the processor load induced by a task is defined by \( U = \frac{c}{h} \) where \( c \) and \( h \) are the execution time and period of the task. It can be rewritten as \( U = c f \), where \( f \) is the task frequency, which is linear when the execution time is constant.

Based on this equation, processor load induced by a task is modelled in a way similar to Cervin et al. (2002). Thus, for each period \( h_i \) of the scheduling controller, the processor load of one plant control task is estimated as:

\[
\hat{U}_{kh_i} = \lambda \hat{U}_{(k-1)h_i} + (1 - \lambda) \overline{\tau} h_i f_{(k-1)h_i} \quad (1)
\]

where \( f \) is the sampling frequency currently assigned to the plant control task and \( \overline{\tau} \) is the mean of its measured job execution-time. \( \lambda \) is a forgetting factor used to smooth the measure.

As \( \overline{\tau} \) depends on the runtime environment (e.g. processor speed) a “normalized” linear model of the task \( i \) (2) is used for the scheduling controller synthesis where \( \overline{\tau} \) is omitted and will be compensated by on-line gain-scheduling (1/\( \overline{\tau} \)), as illustrated by the figure (4) in section 5.

\[
G_i(z) = \frac{\hat{U}_i(z)}{f_i(z)} = \frac{1 - \lambda_i}{z - \lambda_i} \quad (2)
\]

Finally, the scheduling of \( n \) plant control tasks is modelled by the discrete-time state-space representation:

\[
\begin{align*}
x(k+1) &= Ax(k) + B f(k) \\
\hat{u}(k) &= C x(k)
\end{align*} \quad (3)
\]

with \( A = \text{diag}\{\lambda_{1}, \ldots, \lambda_{n}\} \), \( B = \text{diag}\{1 - \lambda_{1}, \ldots, 1 - \lambda_{n}\} \) and \( C = [1 \ldots 1] \). \( f \) and \( x \) are respectively the vectors of task frequencies and task loads whereas \( \hat{u} \) is the load of all plant control tasks.

Based on this state-space representation standard control methodology can be used to design a controller. Here a discrete \( H_{\infty} \) synthesis is proposed and illustrated in section 5.

2. SCHEDULING CONTROLLER

Plant control task periods are on-line adjusted according to the processor load variations. This work is done by a specific task, the scheduling controller, which has the highest priority and a period much larger than the others tasks.

Figure (1) presents the bloc diagram of the feedback scheduling. The scheduling is viewed as a dynamical system which output is the processor utilization and inputs are control task frequencies. As far as the adaptation of the control tasks is concerned, the load of the other tasks is seen as an output disturbance.

![Feedback scheduling bloc diagram](image)

Fig. 1. Feedback scheduling bloc diagram
time controller. The desired closed-loop performances are specified by model matching, as done in (Åström and Wittenmark, 1997).

\[ G(s) = \frac{B(s)}{A(s)} \text{ and } G_m(s, h) = \frac{B_m(s, h)}{A_m(s, h)} \]  

(4)

A formal discretization, exact or approximative, with \( h \) as parameter, leads to both discrete-time transfer functions \( G(z, h) \) and \( G_m(z, h) \) as below, with \( i \leq j \):

\[ G(z, h) = \frac{B(z, h)}{A(z, h)} = \frac{b_i(h)z^i + ... + b_0(h)}{a_j(h)z^j + ... + a_0(h)} \]  

(5)

For easier reading the dependence in the \( z \) and \( h \) variables will sometimes be missed out. Then, the closed-loop controlled system of figure (2) can be expressed by:

\[ G_{cl}(z, h) = \frac{B(z, h)T(z, h)}{A(z, h)R(z, h) + B(z, h)S(z, h)} \]  

(6)

The synthesis objective is to find \( R(z, h), S(z, h) \) and \( T(z, h) \) such that \( G_{cl}(z, h) \) matches \( G_m(z, h) \). In a way similar to Åström and Wittenmark (1997) the following factorizations (7) are used, where \( A \) and \( B \) do not have any common factors, \( A^+ \) and \( B^+ \) are monic and contained stable poles and zeros which can be cancelled, \( A^- \) and \( B^- \) contained unstable poles and zeros, \( R_d \) and \( S_d \) specify given factors of \( R \) and \( S \) (e.g. integral terms).

\[ A = A^+A^- \quad S = A^+S_dS' \]
\[ B = B^+B^- \quad R = B^+R_dR' \]
\[ B_m = B^-B'_m \quad T = A^+T' \]  

(7)

The matching problem (8) needs to solve the Diophantine equation (9). \( R, S \) and \( T \) are obtained with (7). \( A_o \) is the observer polynomial which will be defined, see (Åström and Wittenmark, 1997).

\[ \frac{B^+A^+B^-T'}{B^+A^+(A^-R_dR' + B^-S_dS')} = \frac{B^-B'_m}{A_m} \]  

(8)

\[ (A^-R_dR' + B^-S_dS') = A_mA_o \]  

(9)

\[ T' = B'_mA_o \]  

(10)

Necessary and sufficient conditions to obtain an unique solution and a causal controller, when computation time is disregarded, are given on polynomial degrees in \( z \), by: for all \( h \in \mathbb{R}^+ \),

\[ d^pA_m - d^pB_m = d^pA - d^pB \]  

(11)

\[ d^pR = d^pS = d^pT \]  

(12)

\[ d^pA_o = d^pA + d^pS_d + d^pR_d \]
\[ + d^pA^- - d^pB^+ - 1 - d^pA_m \]  

(13)

The internal observer should be faster than the closed-loop, therefore the roots of \( A_o \) are chosen in accordance with \( A_m \) and (13). The observer dynamic can be defined as a function of the closed-loop dynamic (e.g. five times faster) or arbitrary defined by a continuous-time model discretized as for the closed-loop model.

Remark 1. Due to the dependence in \( h \), equation (9) has to be solved analytically, which can be done with any symbolic computation software.

Remark 2. \( h \)-dependent plant model of equation (5) could also be the result of an interpolation of identifications issue at different sampling periods.

Remark 3. Conditions (11-13) should be generically satisfy, i.e. for almost all values of \( h \). In practice the designer can select a set of \( h \) for which none of polynomials in (11-13) loses a degree in \( z \).

4. CONSTANT VS VARIABLE PLANT CONTROLLER PERFORMANCES

As introduced in section 3 the plant controller can be designed with constant or sampling period dependent performance specifications. Both approaches are here compared in an example.

4.1 Plant controller synthesis

The plant is a stable pendulum. The continuous model (14), linearized at the equilibrium, expressed the angular position in function of the applied torque with \( \omega_0 = 3.77 \text{ rad/s} \), \( \xi = 0.2 \) and \( K = \omega_0^2/9.81 \), see (Eker et al., 2000).

\[ G(s) = \frac{K}{s^2 + 2\xi\omega_0s + \omega_0^2} \]  

(14)

Closed-loop performances are defined in continuous-time form by (15) with \( \xi_m = 0.7 \) and \( K_m = \omega_m^2 \) for a unity static gain. Internal observer \( A_o \) is chosen five times faster than the closed-loop model therefore \( \omega_0ks = 5 \omega_m \) in (16). A zero steady state error to a step disturbance is required.
\[ G_m(s,h) = \frac{K_m(h)}{s^2 + 2\xi_m \omega_m(h)s + \omega_m^2(h)} \quad (15) \]
\[ A_p(s,h) = s^2 + 2\xi_m \omega_{obs}(h)s + \omega_{obs}^2(h) \quad (16) \]

Both cases are considered:

- Constant closed-loop performances
- Variable closed-loop performances

According to the rule of thumb of Åström and Wittenmark (1997), \( \omega_{obs} h \approx 0.2 \ldots 0.6 \)

By defining \( B^- = (z+1)^2 \), which appears in both \( G \) and \( G_m \) numerators, \( B^- = 1, A^+ = 1, S_0 = 1 \) and \( R_d = (z - 1) \) for the integral action, equation (9) and (10) are solved. Using conditions (12) and (13) we obtain \( degT = degS = degR = 2, degR^2 = 1 \) and \( degA_0 = 2 \). Solving the Diophantine equation (9) leads to the three polynomials (18) where each parameters are expressed by a fourth order rational function in \( h \).

\[
\begin{align*}
R(z,h) &= (z-1)(r_1(h) z + r_0(h)) \quad (18) \\
S(z,h) &= s_2(h) z^2 + s_1(h) z + s_0(h) \quad (19) \\
T(z,h) &= t_2(h) z^2 + t_1(h) z + t_0(h) \quad (20)
\end{align*}
\]

### 4.2 Simulation

The two plant control methods designed in the previous section are now compared for the sampling periods 2, 8, 15, 25, and 50 ms using a non-linear model of the pendulum.

Figure (3.a) presents the step response when closed-loop performance specifications are constant. The system becomes unstable for sampling periods greater than 25 ms. According to the rule of thumb \( \omega_{obs} h \approx 0.2 \ldots 0.6 \), a good sampling period for this case is \( h \in [4; 12] \) ms. It appears that this rule is here more restrictive than necessary which is due to a minimum phase margin guaranteed by the rule.

Figure (3.b) presents the step response when closed-loop performances depend on the sampling period. As it was predictable, the system is never unstable and the performances (i.e. \( \omega_m \)) decrease when the sampling period increases.

To conclude this section, adapting closed-loop performance specifications according to the sampling period can increase robust stability w.r.t. sampling period changes. The next section presents the benefits of adapting closed-loop performances in the context of control/scheduling co-design.

### 5. CO-DESIGN EXAMPLE

In this example, two independent stable pendulums are controlled by a computer. Four tasks share the same processor unit on top of a real-time operating system with priority based scheduling. Two tasks control the both pendulums, one task implements the scheduling controller and the last is a disruptive task. Tables (1) and (2) summarize scheduling and pendulum properties respectively. Pendulum control laws are designed as explained in the previous section.

<table>
<thead>
<tr>
<th>Task</th>
<th>Priority</th>
<th>Period (ms)</th>
<th>Execution time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>schedctrl</td>
<td>1</td>
<td>500</td>
<td>1</td>
</tr>
<tr>
<td>pend1</td>
<td>2</td>
<td>4 to 400</td>
<td>2</td>
</tr>
<tr>
<td>disturb</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>pend2</td>
<td>4</td>
<td>4 to 400</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pendulum</th>
<th>Task</th>
<th>( \xi )</th>
<th>( \omega_m )</th>
<th>( \omega_m )</th>
<th>( \omega_m )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>pend1</td>
<td>0.2</td>
<td>3.77</td>
<td>0.7</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>pend2</td>
<td>0.2</td>
<td>4.08</td>
<td>0.7</td>
<td>10</td>
</tr>
</tbody>
</table>

### 5.1 Scheduler control design

The scheduling controller has to adjust the two plant control task periods when the desired processor utilization varies, in response to a reference change or a disruptive task.

Figure (4) presents the bloc diagram of this example which includes saturations and on-line gain-scheduling used to compensate the variations of the job execution time, see section 2.
5.3 Example 1, constant closed-loop performances

In this example, the two control laws have constant closed-loop performance specifications summarized in the table 2. A good sampling period for these performances is \( h \in [4; 12] \) ms, see section 4.

5.2 Co-simulation

This co-simulation is done using a non-linear model of the pendulum and the TrueTime Matlab Toolbox, see (Cervin et al., 2003). A specific Simulink bloc simulates a real-time kernel with tasks, scheduler and input/output capabilities used to connect it with classical Simulink dynamic system blocs.

Two examples are presented, the first with two pendulums with constant closed-loop performances and the second with variable ones.

The scheduling scenario is the same for both examples. Processor utilization starts with a reference of 80%. At time \( t = 5 \) s reference decreases to 60%, representing a decrease of available resources. At time \( t = 12 \) s a new (disruptive) task appears which needs 50% of the processor resources. The priorities and the load allocation ratio \( \alpha = 2 \) are chosen to favor the first pendulum. The two pendulums are excited by square wave reference signal of period 6 s.
In figure (7), both pendulums remain stable after the load reference change. The pendulum two becomes unstable when the disruptive task appears because the sampling period becomes too high.

5.4 Example 2, varying closed-loop performances

In this example, pendulum 1 (high priority) has constant performances and pendulum 2 (low priority) has varying performances, see section 4. The scheduling behavior is the same as the first example, therefore it is not shown here.

Fig. 8. Example 2, pendulum positions

In figure (8) both pendulums stay stable even if the sampling periods rise sharply at $t = 12$ s. The stability of the pendulum 2 is preserved by decreasing the performances. This example emphasizes the interest for adapting closed-loop performances w.r.t. sampling period.

6. CONCLUSION AND FUTURE WORK

In this paper a processor load regulation has been presented based on a simple scheduling model and $H_{\infty}$ synthesis. The synthesis of a sampling period dependent RST controller for the plant has been exposed. Then it has been shown that a co-design, with these two controllers and varying plant performances, can improve robustness in stability and flexibility w.r.t. processor load variations.

However this work is only at its first stage. The scheduling model use a measure of execution time which may be difficult to get in practice, especially with off-the-self operating systems. In this case an observer should be design to estimate the execution time. On the other hand the RST synthesis procedure is complex to used with higher order plant due to the high order, in $h$, of the R, S and T polynomial parameters. The next studies will use parameter dependent controller synthesis such as LPV.

REFERENCES


