A LPV approach to control and real-time scheduling codesign: application to a robot-arm control

Olivier Sename, Daniel Simon and Mongi Ben Gaid

***Abstract*— This paper deals with real-time control under computational constraints. A robust control approach to control/real-time scheduling co-design is proposed using the H_∞ framework for Linear Parameter Varying (LPV) polytopic systems. The originality consists in a new resource sharing between control tasks according to the controlled plant performances. Here the varying parameters are images of the control performance w.r.t. the sampling frequencies. Then a LPV based feedback scheduler is designed to adapt the control tasks periods according to the plant behavior and to the availability of computing resources. The approach is illustrated with a robot-arm controller design, whose feasibility is assessed in simulation.**

Keywords: control/computing co-design, robust control, LPV systems, resource management

I. INTRODUCTION

Optimisation of computing resources in computer-controlled systems is a challenging problem. Current solutions consist in on-line changing the algorithm or adapting the sampling period in order to increase the flexibility by adaptation of the processor utilisation.

This recent research field has received few attention in the past. In [1] some sampling period dependent PID controller is developed and a feedback scheduler based on a LQ optimisation of the control tasks periods is proposed to change on-line the sampling periods of the controller according to the resource availability. In [2] a processor load regulation is proposed and applied for real-time control of a robot arm. In [3], some results are given using the lifting technique for output-feedback synthesis for LPV sampled-data systems, where the sampling period may also be parameter varying. Finally methods to design sampling period dependent controllers have been proposed in [4] for RST ones and in [5], [6] using the H_∞ control approach LPV polytopic systems.

However in all these studies the variation of computing resources is linked to the real-time system performance only and not to the plant expected performances. In this paper we will provide a methodology to design a feedback scheduling controller which will make the resource utilization vary on line according to the resource availability and to the plant trajectory. The design of the real-time control scheme is done in the context of robust control for LPV systems.

Thanks to recent works on robust linear control [7], [8]see and references therein, Linear Matrix Inequalities and optimization tools (see e.g. [9], [10]) and references therein, robust controller synthesis theory is now well established and widely used in control applications. This approach makes possible to tackle the non-linearities of a system by considering them as parameter uncertainties and to build robust controllers w.r.t. these uncertainties. LPV theory allows to model the non-linearities or to make the controller performances varying through the linear introduction of parameters. Hence, since a decade, LPV modelling is being increasingly used and allows to extend classical linear robust control methodology to a larger class of systems, keeping the usage of linear tools.

Following the preliminary results in [2] a LPV/H_∞ feedback scheduler is designed in this paper, which is new in the context of real-time control. The objective is to adapt on line the control task periods (i.e. the varying parameters) not only according to the availability of computing resources but also to the plant performances.

In section II the need for more flexible real-time scheduling is stated and some features on feedback scheduling are given. Section III presents the main result of the paper, i.e an original strategy for control-scheduling co-design which consists in a LPV feedback scheduler where the varying parameter is function of the process to be controlled. This new methodology is illustrated on a robot-arm control problem in section IV.

II. PROBLEM STATEMENT AND STATE OF THE ART

A. The need for more flexible real-time scheduling

Embedded digital control systems use a computer to periodically sample sensors, compute a control law and send control signals to the actuators of a continuous time physical process.

Implementation related issues in digital control are the control latencies and intervals, and also timing uncertainties such as deviations of the sampling period and computing delays, jitter and occasional data loss [11], [12], [13]. However, a shown in [14], a robust closed-loop system is, to some extent, able to tolerate such problems with no loss of stability or integrity.

On the other hand the scheduling policy would give better results when chosen based on application's based requirements instead of traditional policies (such as Rate Monotonic

O.Sename is with GIPSA-lab - Department of Control Systems, ENSE3, BP 46, 38402 Saint Martin d'Hères Cedex, France olivier.sename@gipsa-lab.inpg.fr

D.Simon is with INRIA Rhône-Alpes, NeCS team, Inovallée, Montbonnot, 38334 Saint-Ismier Cedex, France Daniel.Simon@inrialpes.fr

M. Ben Gaid is with Institut Français du Pétrole, IFP, Rueil-Malmaison, France mongi.ben-gaid@ifp.fr

This work was supported by the French National Research Agency (ANR) project Safe-NeCS under grant No. ANR-ARA SSIA-NV-15

for fixed priorities or Earliest Deadline First for dynamic ones), which are not control aware but only computing aware.

Finally off-line schedulability analysis rely on a right estimation of the tasks worst case execution time. However in embedded systems the processors use caches and pipelines to improve the average computing speed while decreasing the timing predictability. Another source of uncertainty may come from some pieces of the control algorithm. For example, the duration of a vision process highly depends on incoming data from a dynamic scene. In a dynamic environment, some control activities can be suspended or resumed and control algorithms with different costs can be scheduled according to various control modes leading to large variations in the computing load.

Thus real-time control design based on worst case execution time, maximum expected delay and strict deadlines inevitably leads to a low average usage of the computing resource and to a poor adaptivity w.r.t. a complex execution environment. All these drawbacks call for a better integration of control goals and computing capabilities through a co-design approach.

B. About feedback scheduling

Some preliminary works have been done in the last decade in view of control/scheduling co-design, for instance in [15] for off-line iterative optimisation of scheduling parameters.

Concerning co-design for on-line implementation, recent results deal with varying sampling rates in control loops in the framework of linear systems: for example [16] show that, while switching between two stable controllers, too frequent control period switches may lead to unstability. Moreover most real-life systems are non-linear and the extrapolation of timing assignment through linearisation often gives rough estimations of allowable periods and latencies or even can be meaningless. In fact, the knowledge of the plant's behaviour w.r.t varying sampling is necessary to get an efficient control/scheduling co-design: the on-line combination the control performance and implementation constraints lead to a feedback scheduling approach.

This approach has been initiated from both the real-time computing side [17] and the control side [1]. The idea consists in adding to the process controller an outer sampled feedback loop ("scheduling regulator") to control the scheduling parameters as a function of a QoC (Quality of Control) measure. Indeed the objective is to increase the control performance (via efficient resource sharing) and robustness w.r.t timing uncertainties.

Figure 1 gives an overview of a feed-back scheduler architecture where an outer loop (the *scheduling controller*) adapts in real-time the scheduling parameters from measurements taken on the computer's activity, e.g. the computing load. Besides this controller working periodically (at a rate larger than the sampling periods of the plant control tasks), the system's structure may evolve along a discrete time scale upon occurrence of events, e.g. for new tasks admission or exception handling. These decisional processes may be handled by another real-time task, the *scheduling manager*,

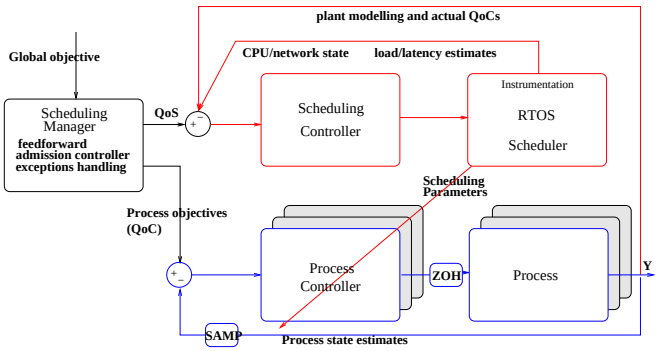

Fig. 1. Hierarchical control structure

which is not further detailed in this paper. Notice that such a manager may give a reference to the controller resource utilisation.

The design problem can be stated as control performance optimisation under constraint of available computing resources. Early results come from [18] where a problem of optimal control under computation load constraints is theoretically solved by a feedback scheduler, but leads to a solution too complex to be implemented in real-time. Then [14] shows that this optimal control problem can be often simply implemented by computing the new tasks periods by the rescaling:

$$h_i^{k+1} = h_i^k \frac{U}{U_{sp}}$$

where U_{sp} is the utilisation set-point and U the estimated CPU load. The feedback scheduler then controls the processor utilisation by assigning task periods that optimise the overall control performance. This approach is well suited for a "quasi-continuous" variation of the sampling periods of real-time tasks under control of a preemptive real-time operating system.

Another approach has been used in the framework of the so-called *(m,k)-firm* schedulability policy, where the scheduling strategy ensures the successful execution of at least *m* instances of a given task (or message sending) for each time window of length *k* slots. Hence a selective data drop policy (as in [19]) or a computing power allocation to selected tasks (as in [20]) can be used to perform optimal control of a plant under constraint of computing or communication limitations. This latter approach is well suited for non-preemptive scheduling of control tasks and for networked control systems subject to messages loss : the tasks or messages are scheduled to jointly perform congestion avoidance and optimal control.

Finally the authors have used in [21] a LQG approach to design the feedback scheduling controller while in [2] an H_∞ approach is proposed for a multi-task control systems, assuming an a priori distribution of the computing resources between the control tasks.

However the distribution is fixed and the feedback scheduling strategy only ensures the use of computing resources as a function of the resource constraints (availability). No

plant information is used to make the resource distribution optimal w.r.t the closed-loop performances of the plant to be controlled by the real-time computer.

In the following section the approach in [2] is extended, allowing to make the distribution of the computing resources to vary according to the plant trajectory, which is new in the context of real-time feedback scheduling.

The next section concerns the main paper result, i.e. the design of a new methodology for real-time control. It consists in designing a LPV feedback scheduler where the varying parameter is function of the process to be controlled.

III. A LPV FEEDBACK SCHEDULER IN VIEW OF PLANT CLOSED-LOOP PERFORMANCES

Feedback scheduling is a dynamic approach allowing a better usage of the computing resources, in particular when the workload changes (e.g. due to the activation of an admitted new task). Here the CPU activity will be controlled according to the resource availability by adjusting scheduling parameters (i.e. period) of the plant control tasks. However the use of computing resources should also be linked to the dynamical behavior of the plant(s) to be controlled. Indeed while controlling different subsystems in a single computer it is natural to ensure the resource availability when large transient behaviors occur. The main result given in this section consists in deriving a new feedback scheduling controller which will depend on the plant trajectory in view of an "optimal" resource sharing. It is designed in the LPV/H_∞ framework for polytopic systems.

Following previous authors' results in [2], the feedback scheduling is illustrated in Fig 2 as a dynamical system between control task frequencies and processor utilisation. As far as the adaptation of the control tasks is concerned, the load of the other tasks is seen as an output disturbance.

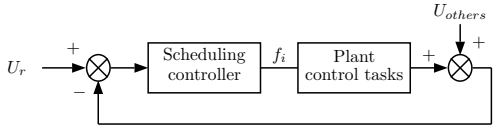

Fig. 2. Feedback scheduling bloc diagram

We assume that the CPU utilization is measured or estimated. Let us first recall that 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. Hence processor load induced by a task is estimated, in a similar to way [1], for each period h_s of the scheduling controller, as:

$$\hat{U}_{kh_s} = \lambda\, \hat{U}_{(k-1)h_s} + (1-\lambda)\, \frac{\overline{c}_{kh_s}}{h_{(k-1)h_s}} \tag{1}$$

where h is the sampling frequency currently assigned to the plant control task (i.e. at each sampling instant kh_s) and $\overline{c}$ is the mean of its measured job execution-time. λ is a forgetting factor used to smooth the measure (here $\lambda = 0.3$).

Now for a n-multi-tasks control system, one should note that, as shown in [22], if the execution times are constant, then the relation, $U = \sum_{i=1}^{n} C_i f_i$ (where $f_i = 1/h_i$ is the frequency of the task) is a linear function (while it would not be the case if expressed as a function of the task periods). Therefore, using (1), the estimated CPU load is given as:

$$\hat{U}(kh_S) = \frac{(1-\lambda)}{z-\lambda} \sum_{i=1}^{n} \overline{c}_i(kh_S) f_i(kh_S) \tag{2}$$

However in practice, the execution-time of the control tasks may vary according to the run-time environment (e.g. processor speed). As proposed in [2], a "normalized" linear model of the task i (i.e independent on the execution time), G_i', is used for the scheduling controller synthesis where $\overline{c}$ is omitted and will be compensated by on-line gain-scheduling $(1/\overline{c})$ as shown below.

$$G_i'(z) = \frac{\hat{U}(z)}{f_i(z)} = \frac{1-\lambda}{z-\lambda}, \quad i = 1, \ldots, n \tag{3}$$

Also, as explained above the use of computing resources is chosen to depend on the plant trajectory. Hence the control scheme of computing resource control is illustrated in figure 3 for a 2-tasks control systems for simplicity.

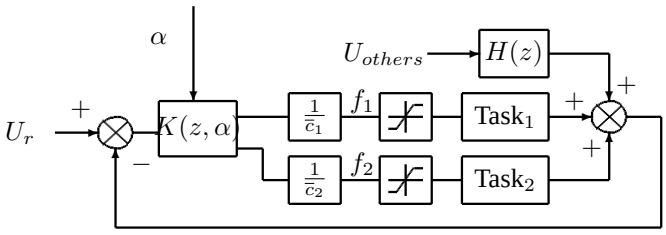

Fig. 3. Control scheme for CPU resources

In figure 3 the interval of frequencies is limited by the "saturation" block, α represents a set of real parameters $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ dedicated to the set of control tasks $\{U_1, U_2, \ldots, U_n\}$. These parameters will be used to make the resource sharing vary according to the plant trajectory. For instance, in a 2 control-tasks system, where $U = U_1 + U_2$, we will require that :

$$U_1 = \alpha U \tag{4}$$
$$U_2 = (1-\alpha) U \tag{5}$$

α being a varying parameter.

This makes the control scheme flexible enough to distribute on-line the use of computing resources to the different control tasks. The choice of the value of the time-varying parameters $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ can be realized by many different ways, from on-line computation of optimal cost functions, to a dependency on the control effort. It will be illustrated in details in section IV for the robot-arm control example.

Here the design of the controller $K(\alpha)$ is done using the H_∞ control approach for LPV systems. The H_∞ control scheme to synthesize the controller $K(\alpha)$ is given in figure 4.

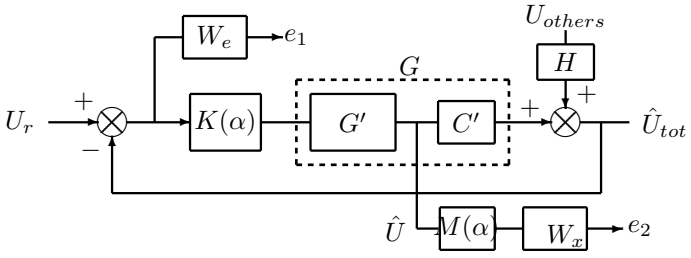

Fig. 4. A LPV Hinf controller for CPU resources

In figure 4, G' is the model of the scheduler, the output of which is the vector of all task loads. To get the sum of all task loads as in (3), we use $C' = [1 \ \ldots 1]$. The H transfer function represents the sensor dynamical behaviour which measures the load of the other tasks. It may be a first order filter. The template W_e specifies the performances on the load tracking error. It is chosen in the continuous-time domain as :

$$W_e(s) = \frac{s/M_s + \omega_b}{s + \omega_s \epsilon} \tag{6}$$

with $M_s = 2$, $\omega_s = 10 rad/s$, $\epsilon = 0.01$ to obtain a closed-loop settling time of $300\,ms$, a static error less than $1\,\%$ and a good robustness margin.

The resource distribution is realized through the $M(\alpha)$ matrix defined below. Note that for a n-multi-tasks system:

$$U = U_1 + U_2 + \ldots + U_n \tag{7}$$
$$U = (\alpha_1 + \alpha_2 + \ldots + \alpha_n)U \tag{8}$$

where $\alpha_1 + \alpha_2 + \ldots + \alpha_n = 1$. Then:

$$U_1 = \alpha_1 U \tag{9}$$
$$U_2 = \alpha_2 U = \frac{\alpha_2}{\alpha_1} U_1 \tag{10}$$
$$U_3 = \alpha_3 U = \frac{\alpha_3}{\alpha_2} U_2 \tag{11}$$
$$\vdots \quad \vdots \tag{12}$$
$$U_n = \alpha_n U = \frac{\alpha_n}{\alpha_{n-1}} U_{n-1} \tag{13}$$

Then to ensure the on-line distribution of the computing resources M is chosen as follows:

$$M = \begin{bmatrix} -\alpha_2 & \alpha_1 & 0 & \ldots & \ldots & 0 \\ 0 & \alpha_3 & \alpha_2 & 0 & \ldots & 0 \\ \vdots & \ldots & \ldots & \ldots & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & -\alpha_n & \alpha_{n-1} \end{bmatrix} \tag{14}$$
$$= \alpha_1 M_1 + \alpha_2 M_2 + \ldots + \alpha_n M_n \tag{15}$$

Using [7] the LPV controller $K(\alpha)$ is obtained through the solution of the H_∞ control problem for polytopic systems, and consists in solving 2 LMIs. Then the design of $K(\alpha)$ can be done directly in the discrete-time domain or in the continuous-time one and then discretized. In this paper $K(\alpha)$ has been synthesized in the continuous-time domain using the H_∞ control approach for polytopic systems, as described in the Appendix.

By solving the H_∞ problem for the LPV system using the Yalmip interface and Sedumi solver [9], [10], one obtains $\gamma_{opt} = 1.8885$, and a controller of order 7.

IV. Application to a robot-arm control

We consider here a seven degrees of freedom Mitsubishi PA10 robot arm that has been previously modelled and calibrated [2].

The problem under consideration is to track a desired trajectory for the position of the end-effector. Using the Lagrange formalism the following model can be obtained:

$$\Gamma = M(q)\ddot{q} + Gra(q) + C(q, \dot{q}) \tag{16}$$

where q stands for the positions of the joints, M is the inertia matrix, Gra is the gravity forces vector and C gathers Coriolis, centrifugal and friction forces.

The structure of the (ideal) linearising controller includes a compensation of the Gravity, Coriolis/centrifugal effect and Inertia variations as well as a Proportional-Derivative (PD) controller for the tracking and stabilisation problem, of the form:

$$\Gamma = Gra(q) + C(q, \dot{q}) + K_p(q_d - q) + K_d(\dot{q_d} - \dot{q}), \tag{17}$$

leading to the linear closed-loop system $M(q)\ddot{q} = K_p(q_d - q) + K_d(\dot{q_d} - \dot{q})$,
where q_d and $\dot{q_d}$ stand for the reference trajectory positions and velocities.

The controller is split in five tasks, i.e. a specific task is considered for the PD control, the trajectory generation and for the Gravity, Inertia and Coriolis compensations, in order to use a multi-rate controller. In this feedback scheduling scheme, only the periods of the compensation tasks will be adapted, as they are time consuming compared with the PD task while being less critical for the stability.

A. Performance evaluation of the control tasks in view of optimal resource distribution

In order to associate the use of computing resources with the robot trajectory, the contribution of each of the 3 control tasks to the closed-loop system performances has been evaluated as a function of its execution period.

The methodology is the following. Assuming a nominal sampling period for each task of $1ms$, the period of each compensation control task is changed, and new simulations are performed during which the following cost is computed:

$$J = \int_{t_i}^{t_f} (P_{h_i,h_g,h_c}(t) - P_{ref}(t))^2 - \int_{t_i}^{t_f} (P_c(t) - P_{ref}(t))^2 \tag{18}$$

where P_{ref} is the desired position in the operational space of the end tip, computed from q_d using the geometric model, P_c is the position obtained when all the control tasks act with the minimal sampling period of 1 ms. Finally P_{h_i,h_g,h_c} is the position obtained when the sampling period of one of the compensation tasks is increased from 1 to 30 ms.

Simulations are performed for a particular robot trajectory, defined by the reference vector $(q_d, \dot{q_d})$ for all the robot

joints. Here q_d goes from $\pi/2$ to $-\pi/2$. We get the following results presented in figures 5, 6 and 7, where the evolution of the cost function J is shown for the three compensation control task.

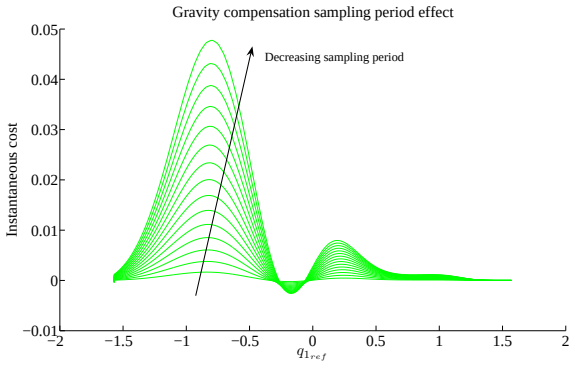

Fig. 5. Cost variation due to varying sampling for gravity compensation task

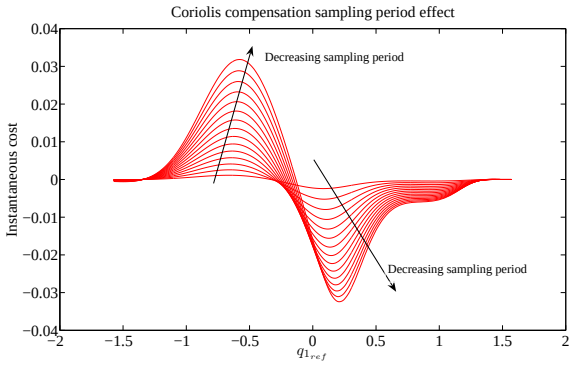

Fig. 6. Cost variation due to varying sampling for Coriolis compensation task

While it is difficult to infer the relations between the compensation tasks execution period and the trajectory tracking performance, a natural interpretation is as follows. First the Gravity compensation effect is very sensitive to the increase of the sampling period at the end of the trajectory, as the cost increases in the second part of the trajectory (first part of the graph as the trajectory goes from $\pi/2$ to $-\pi/2$). We will require to ensure the availability of CPU resources to this task, in a linear way with the trajectory position. Then the situation is almost reverse for the Inertia effect. Finally, even if some variations can be observed, we will ask for a constant use of CPU resources of the Coriolis compensation task, all along the trajectory.

We have then chosen that the distribution of control task periods should be:

$$U_I = \alpha_I U, \quad U_G = \alpha_G U, \quad U_C = \alpha_C U \tag{19}$$

where $\alpha_C = 0.25$, $\alpha_I = 1 - \alpha_G$, and α_G is linked to the plan trajectory by:

$$\alpha_G = \alpha_{min} + (\alpha_{Max} - \alpha_{min}) \times \frac{q_d - q_{end}}{q_{ini} - q_{end}} \tag{20}$$

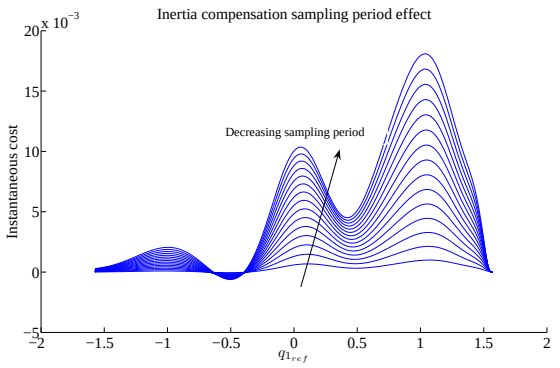

Fig. 7. Cost variation due to varying sampling for Inertia compensation task

where $[\alpha_{min};\ \alpha_{Max}] = [0.1;\ \ 0.65]$, q_{ini} is the initial position and q_{end} the final trajectory position.

B. Simulation with TrueTime

TrueTime (`www.control.lth.se/truetime/`) [14] is a MATLAB/Simulink-based simulator for real-time control systems that eases simulation of the temporal behaviour of a multi-tasking real-time system executing controller tasks. The tasks are controlling processes that are modelled as ordinary continuous-time Simulink blocks.

In this section, simulations have been performed using TrueTime. In this application, the period of the feedback scheduler has been fixed to $30ms$ to be larger than the robot control tasks periods, which limits have been set from $1ms$ to $30ms$.

In the experiment depicted in figure 10 the desired CPU usage is initially set to 50% of the maximum usage. The upper plots show the tasks periods and CPU usage. The PD loop period is fixed at $1ms$ and the trajectory generator at $5ms$.

As seen in figure 10, the load of the compensation tasks (Gravity, Coriolis and Inertia) vary on line as expected according to the parameter α_I (see figure 12). The corresponding evolution of the task periods is shown in figure 11. Moreover, in figure 13, the adaptive LPV case (α varying) is compared with the constant case ($\alpha = 0.375$). It can be seen that the LPV case leads to a smaller cost function which emphasizes the real interest of the provided approach.

V. Conclusions

In this paper a new approach for integrated control/real-time scheduling has been proposed in the framework of the robust theory for LPV systems. It consists in a LPV/H_∞ feedback control of computing resources, where the varying parameters can be linked to the controlled plant performances. The given control structure is open enough to allow for many plant performance criteria to be used for the on-line variation of the parameters. Here it is illustrated through the real-time control of a robot-arm, and the varying parameters are in this case linked to the robot reference trajectory, leading to a better use of computing resources

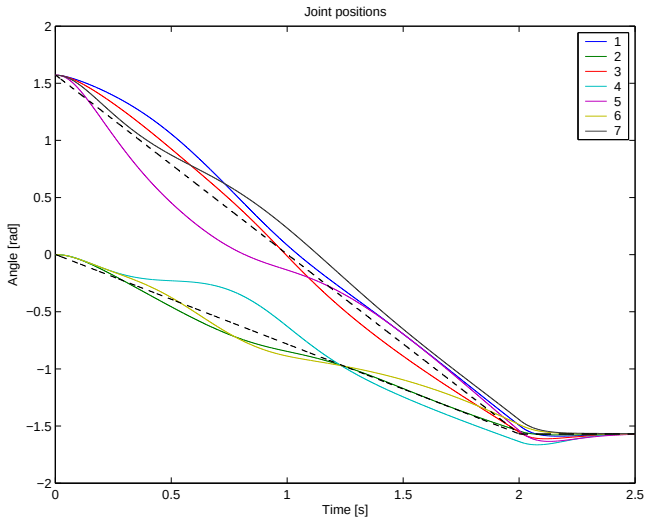

Fig. 8. Positions

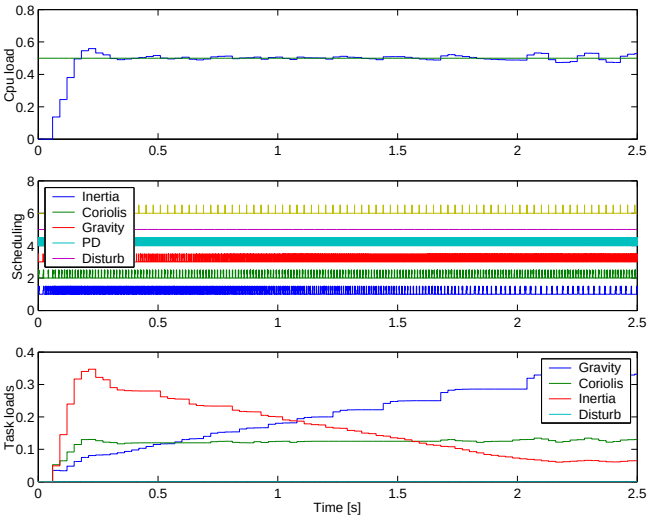

Fig. 10. TrueTime real-time parameters: loads, periods

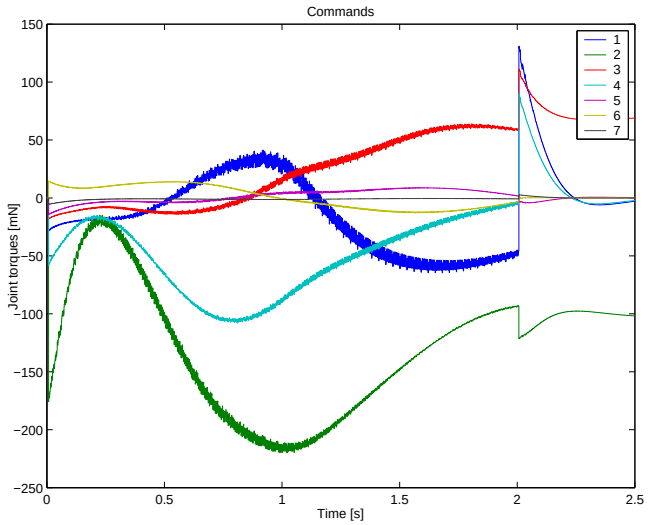

Fig. 9. Control torques

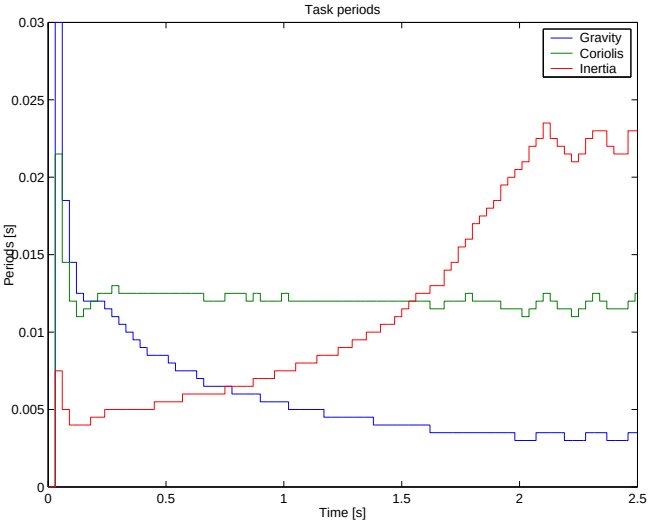

Fig. 11. Task periods

and an increase of closed-loop performances. The provided simulation results using TrueTime emphasize the interest and efficiency of the proposed methodology, which highlights the benefits of the robust control approach.

Note that, as explained in [2] and depicted in Figure 14, the scheduling feedback loop can be easily implemented on top of an off-the-shelf real-time operating system (e.g. Posix) under the form of an additional real-time periodic task, i.e. a control module which function is specified and encoded by the control designer. The inputs are the measured execution times of the control tasks. The set point is a desired global computing load. Outputs are the sampling intervals of the Gravity, Coriolis and Inertia control tasks which are triggered by programmable timers provided by the operating system.

Thanks to the use of a hierarchical control structure, the given results may also be integrated with existing methods for the design of varying sampled controllers, as in [3], [6]. This makes this integrated approach easier and generic.

REFERENCES

[1] A. Cervin, J. Eker, B. Bernhardsson, and K. Årzén, "Feedback-feedforward scheduling of control tasks," *Real-Time Systems*, vol. 23, no. 1–2, pp. 25–53, Jul. 2002.

[2] D. Simon, D. Robert, and O. Sename, "Robust control / scheduling co-design: application to robot control," in *Proceedings of the 11th IEEE Real-Time and Embedded Technology and Applications Symposium*, San Francisco, USA, March 2005.

[3] K. Tan, K.-M. Grigoriadis, and F. Wu, "Output-feedback control of LPV sampled-data systems," *Int. Journal of Control*, vol. 75, no. 4, pp. 252–264, 2002.

[4] D. Robert, O. Sename, and D. Simon, "Sampling period dependent RST controller used in control/scheduling co-design," in *Proceedings of the 16th IFAC World Congress*, Prague, Czech Republic, Jul. 2005.

[5] ——, "Synthesis of a sampling period dependent controller using LPV approach," in *5th IFAC Symposium on Robust Control Design ROCOND'06*, Toulouse, France, july 2006.

[6] ——, "A reduced polytopic LPV synthesis for a sampling varying controller : experimentation with a T inverted pendulum," in *European Control Conference ECC'07*, Kos, Greece, 2007.

[7] P. Apkarian and P. Gahinet, "A convex characterization of gain scheduled $\mathcal{H}_\infty$ controllers," *IEEE Transaction on Automatic Control*, vol. 40, no. 5, pp. 853–864, may 1995.

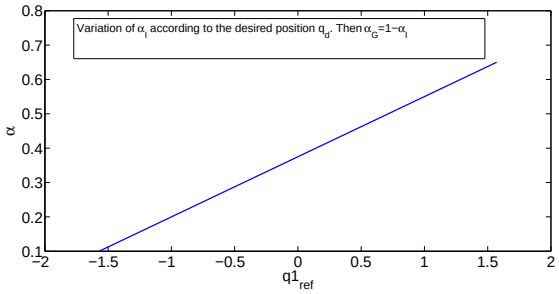

Fig. 12. Variation of α_I according to the desired position

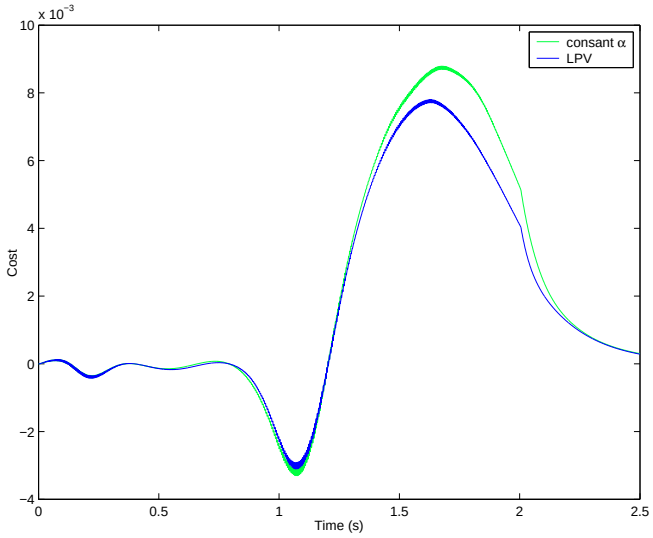

Fig. 13. Comparison of the LPV approach with the constant one: total cost

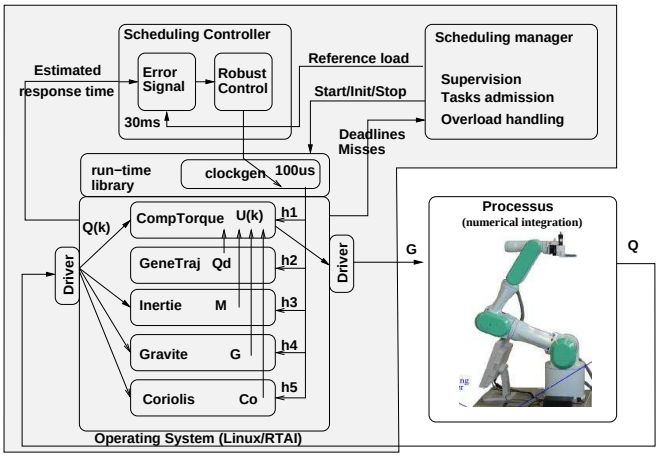

Fig. 14. Feed-back scheduling experiment

[8] P. Gahinet, P. Apkarian, and M. Chilali, "Affine parameter-dependent lyapunov functions and real parametric uncertainty," *IEEE Transaction on Automatic Control*, vol. 41, no. 3, pp. 436–442, march 1996.

[9] J. F. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optim. Methods Softw.*, vol. 11/12, no. 1-4, pp. 625–653, 1999, interior point methods.

[10] J. Lofberg, "Yalmip : A toolbox for modeling and optimization in matlab," in *Proceedings of the CACSD Conference*, 2004.

[11] K. Åström and B. Wittenmark, *Computer-Controlled Systems*, 3rd ed. Prentice Hall, 1997.

[12] M. Törngren, "Fundamentals of implementing real-time control applications in distributed computer systems," *Real Time Systems*, vol. 14, no. 3, pp. 219–250, 1998.

[13] D. Simon, E. Castillo, and P. Freedman, "Design and analysis of synchronization for real-time closed-loop control in robotics," *IEEE Trans. on Control Systems Technology*, vol. 6, no. 4, pp. 445–461, july 1998.

[14] A. Cervin, "Integrated control and real-time scheduling," Ph.D. dissertation, Department of Automatic Control, Lund Institute of Technology, Sweden, Apr. 2003.

[15] M. Ryu, S. Hong, and M. Saksena, "Streamlining real-time controller design: from performance specifications to end-to-end timing constraints," in *IEEE Real Time Systems Symposium*, San Francisco, USA, 1997.

[16] M. Schinkel, W.-H. Chen, and A. Rantzer, "Optimal control for systems with varying sampling rate," in *Proceedings of American Control Conference*, Anchorage, USA, May 2002.

[17] C. Lu, J.-A. Stankovic, G. Tao, and S.-H. Son, "Feedback control real-time scheduling: Framework, modeling, and algorithms," *Real Time Systems*, vol. 23, no. 1, pp. 85–126, 2002.

[18] J. Eker, P. Hagander, and K.-E. rzn, "A feedback scheduler for real-time controller tasks," *Control Engineering Practice*, vol. 8, no. 12, pp. pp 1369–1378, 2000.

[19] N. Jia, Y.-Q. Song, and F. Simonot-Lion, "Graceful degradation of the quality of control through data drop policy," in *Proceedings of the European Control Conference*, Kos, Greece, July 2007.

[20] M.-M. Ben Gaid, A. Çela, Y. Hamam, and C. Ionete, "Optimal scheduling of control tasks with state feedback resource allocation," in *Proceedings of the 2006 American Control Conference*, Minneapolis, USA, June 2006.

[21] O. Sename, D. Simon, and D. Robert, "Feedback scheduling for real-time control of systems with communication delays," in *ETFA'03 9th IEEE International Conference on Emerging Technologies and Factory Automation*, Lisbonne, september 2003.

[22] D. Simon, O. Sename, D. Robert, and O. Testa, "Real-time and delay-dependent control co-design through feedback scheduling," in *CERTS'03 Workshop on Co-design in Embedded Real-time Systems*, Porto, Portugal, july 2003.