Predictive Control of Nonlinear Chemical Processes under Asynchronous Measurements and Controls

P. Varutti * **and R. Findeisen** *

* *P.Varutti and R.Findeisen are with the Institute of Automation Engineering, Otto-von-Guericke University, 39016 Magdeburg, Germany ({paolo.varutti,rolf.findeisen}@ovgu.de).*

Abstract: In many process control problems measurement and control instances might not be available in a periodically-equally-distributed way. Moreover, due to the sensor processing time, actuators/sensors calibration, or computation, inevitable delays can often arise. Also information losses caused, for example, by temporary components failure, or the presence of unreliable communication media, might represent a non-trivial problem. This leads to asynchronous availability of measurement and control inputs, i.e. the controller, sensor, and actuator work in an event-driven, rather than a continuous way. In order to avoid instability and performance loss all these issues must be considered during the control design. In this paper, it is shown that predictive control methods based on continuous time models can be used to stabilize event-based nonlinear systems under variable delays, and limited information losses. It is demonstrated that by using the suggested approach asymptotic convergence is ensured.

Keywords: nonlinear model predictive control, continuous time systems, event-based control, delays, information losses, process control

1. INTRODUCTION

In many cases, continuous time systems are controlled by means of periodically-equally-distributed sampling times, commonly assumed to be known a priori. However, practical control problems are often intrisictly asynchronous, i.e. the dynamics of the system depends on some –maybe exogenous– event. Examples are multi fold: sensors such as chromatographers or laboratory measurements of compositions could need long time due to calibration or limited processing capabilities. The energy of an actuator might be limited, e.g. it must first be "charged" before applying the input. Measurements/actuation might demand human interaction, often unpredictable or inefficient. Moreover, it happens frequently that systems are subject to input/output delays caused, for example, by computational time, communication, and/or sensor/actuator slow dynamics. It might also occur that part of the exchanged information is lost, e.g. due to components failure, or the use of unreliable communication media. If these issues are not taken into account, performance loss or instability of the closed loop can arise. Event-based and asynchronous control is a very active field, Brockett and Liberzon (1998); Heemels et al. (2008). However, most of the work focuses only on linear unconstrained systems, without considering explicitly either time delays or information losses.

In this paper, a solution for the formerly introduced problems, especially suitable to process control applications, is presented. In particular, the suggested solution relies on Predictive Control (PC), which fits well the nature of event-based/asynchronous systems, since the sampling times do not have to neither be equally distant nor to be known a priori (see Fontes (2001); Findeisen (2006)). The compensation capabilities of PC with respect to measurement and computational delays have already been assessed in inter alia Chen et al. (2000); Findeisen and Allgöwer (2004)), where asymptotic stability with respect to such delays has been established. In this work, we focus on the complete asynchronous case, including delays and losses on the actuation and measurement side, which is significantly more challenging. By using smart-sensors and smart-actuators, i.e. components capable of full-duplex communication with the controller, it is possible to achieve closed loop asymptotic convergence.

In the next section PC and the problem under consideration are formally presented. In Section 3, an asynchronous PC solution to compensate delays and information losses is introduced. Results on asymptotic convergence are provided. Simulation results on a Continuous Stirred Tank Reactor (CSTR) are reported in Section 4.

2. PROBLEM STATEMENT

We consider the problem of controlling the nonlinear time-continuous process of the form

$$\dot{x} = f(x,u), \quad x(0) = x_0, \quad x \in \mathbb{R}^n, u \in \mathbb{R}^m. \tag{1}$$

It is assumed that the whole state x is available only at discrete instants t_i. The objective is to stabilize the system around the origin, i.e. $\|x\| \to 0$ for $t \to \infty$, under the state and input constraints $x \in X \subset \mathbb{R}^n$, $u \in U \subset \mathbb{R}^m$. The state constraints X, e.g. max temperature, and the input constraints U, e.g. max valve opening, are assumed to be closed sets. It is also assumed that $f(0,0) = 0$, and f is sufficiently differentiable. The controller should provide for every state measurement $x(t_i)$ a piece of input trajectory

$$u(t) = u(t; x(t_i)), \text{ for } t \in (t_i, t_{i+1}], \quad (2)$$

i.e. the calculated input trajectory is applied open loop in between consecutive recalculation times. This kind of control is commonly called sampled-data open loop feedback. Note that the recalculation times t_i do not need to be known a priori, e.g. in the event-based case when a measurement is triggered once some conditions, such as deviation from the product specifications, are met. This fits well such an asynchronous frame, and it can be used to provide better performance, thanks to the possibility of adjusting on-the-fly the recalculation frequency.

2.1 Predictive Control

In this section, we summarize the basic idea of PC (for more details see Mayne et al. (2000); Findeisen (2006)). The idea is to use a model of the process to be controlled, in order to repeatedly solve an optimization problem, based on the state prediction provided by the model itself. Then, only the first piece of trajectory is implemented and the problem is re-solved with the new measurement. The following definition will be useful for the remainder of the paper.

Definition 1. (Partition). Every series $\pi = (t_i)$, $i \in \mathbb{N}$, $t_i \in \mathbb{R}^+$, such that $t_0 = 0$, $t_i < t_{i+1}$ and $t_i \to \infty$ is called partition.

For every $t_i \in \pi$, $x(t_i)$ is measured, and

$$\min_{\bar{u}(\cdot)} \int_{t_i}^{t_i+T_p} F(\bar{x}(\tau), \bar{u}(\tau))d\tau + E(\bar{x}(t_i + T_p)), \quad (3a)$$

$$\text{s.t. } \dot{\bar{x}}(t) = f(\bar{x}(t), \bar{u}(t)), \quad \bar{x}(t_i) = x(t_i), \quad (3b)$$

$$\bar{u}(t) \in U, \quad t \in (t_i, t_{i+1}], \quad (3c)$$

$$\bar{x}(t) \in X, \quad (3d)$$

$$\bar{x}(t_i + T_p) \in \mathcal{E}, \quad (3e)$$

is solved, where $\bar{\cdot}$ denotes the controller internal variables. The solution is an optimal control signal $u^*(t; x(t_i))$, for $t \in [t_i, t_i + T_p]$, where T_p represents the finite prediction horizon. For sake of simplicity, prediction and control horizon are supposed to be equal, i.e. $T_c = T_p$. The control input is then implemented for the time-span $(t_i, t_i + \delta]$, i.e.

$$u(t) = u^*(t; x(t_i)), \text{ for } t \in (t_i, t_i + \delta], \quad (4)$$

where δ represents the interval between two consecutive recalculation times, i.e. $\delta = (t_{i+1} - t_i), \forall t_i, t_{i+1} \in \pi$. Stability can then be achieved by properly choosing the cost functional $F(x, u)$, the terminal cost $E(x)$, the terminal region $\mathcal{E} \subset X$, and the prediction horizon T_p, see Mayne et al. (2000); Fontes (2001); Findeisen (2006). As formerly mentioned, it is commonly assumed that the recalculation intervals $\delta = (t_{i+1} - t_i)$ are constant and known a priori. Here, however, these assumptions are relaxed, allowing for the recalculation instants to be time-varying and unknown a priori. The only requirement on δ is given by Assumption 2.

Assumption 2. Given the prediction horizon T_p, $\beta \in \mathbb{R}^+$,

$$\beta < \delta = (t_{i+1} - t_i) < T_p, \forall t_i, t_{i+1} \in \pi. \quad (5)$$

In the remainder of the paper, $\overline{\delta}$ will be used to refer to the maximum recalculation interval $\delta = (t_{i+1} - t_i)$. Additionally, the following theorem will be useful for the final results.

Theorem 3. (Asynchronous Predictive Control). Consider the closed-loop system given by (1), (3)-(4). If

i) Assumption 2 is satisfied.

ii) $\forall x_0 \in \mathcal{E} \subseteq X, \exists \bar{u}(\tau) \in U, \tau \in [0, T_p]$ where

$$x(\tau) \in \mathcal{E}, \quad (6a)$$

$$\text{for } \dot{x}(\tau) = f(x(\tau), \bar{u}(\tau)), \quad x(0) = x_0, \quad (6b)$$

$$\text{and } \frac{\partial E}{\partial x} f(x(\tau), \bar{u}(\tau)) + F(x(\tau), \bar{u}(\tau)) \leq 0. \quad (6c)$$

iii) The optimal control problem is solvable for a time t_0.

Then, $\lim_{t \to \infty} \|x(t)\| = 0$.

Proof. The proof comes directly from Findeisen (2006) and the results about PC stability, see inter alia Fontes (2001); Mayne et al. (2000). It must be ensured that δ is smaller than T_p.

Remark 4. The solution of the optimal control problem, as well as the closed loop stability, are based only on the discrete time measurements $x(t_i)$ at $t_i \in \pi$, where π does not have to be known a priori. This makes PC a very appealing solution for event-based problems.

Remark 5. Note that Theorem 3 states only asymptotic converge, and not asymptotic stability in the Lyapunov sense. The former is a weaker property, meaning that even withouth disturbances, the system can temporary drift away from the equilibrium point before converging to the equilibrium. Proving asymptotic stability would in the first step require to rigorously define stability for discrete event systems, since the partition π is not known a priori. This is way beyond the focus of this paper.

2.2 Delays and Information Losses

In a closed loop controlled system, it is quite common to face delays and/or information losses, e.g. due to components failures. Essentially, there can be three delay sources (see Figure 1):

i) *Measurement delays*, which can be due to measurement elaboration, observer reconstruction, slow sensor dynamics, but also the time required for a signal to reach the controller.

ii) *Computational delays*, which represent the time required by the controller to calculate the control input.

iii) *Actuation delays*, which can be due to slow actuator dynamics, but also signal transportation.

The following assumption on the delays is made:

Assumption 6. $\tau_s(t), \tau_c(t), \tau_a(t)$ are nondeterministic with arbitrary probability distribution, but ultimately limited, i.e.

$$\tau_s(t) \in [0, \overline{\tau_s}], \tau_c \in [0, \overline{\tau_c}], \tau_a(t) \in [0, \overline{\tau_a}]. \quad (7)$$

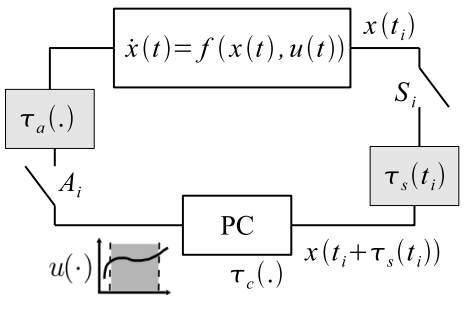

Fig. 1. Sketch of an event-based system subject to delays and information losses.

Both the sensor-to-controller and controller-to-actuator channel can suffer from information losses, which, for example, can be modeled as Bernoullian variables $A_i \sim \mathcal{B}(1-p_a)$, and $S_i \sim \mathcal{B}(1-p_s)$, such that

$$A_i = \begin{cases} 1, & \text{if a control input is received} \\ 0, & \text{otherwise} \end{cases},$$

$$S_i = \begin{cases} 1, & \text{if a measurement is received} \\ 0, & \text{otherwise} \end{cases}.$$

p_a, and p_s represent the loss probabilities for the actuation and the measurement link respectively. Sensor, controller, and actuator are event-driven, such that measurement and control information are dispatched only when necessary. It is also assumed that the following statements are fulfilled.

Assumption 7. Either a common global time, or a set of synchronized clocks is available, such that a common unique time t is established among the components.

Assumption 8. All exchanged information is time-stamped.

3. ASYNCHRONOUS PC

Model-based approaches, such as PC, represent an intuitive and natural way to handle input/output delayed systems. In this section, we show how PC can be used in an event-based way to control asynchronous systems by preserving stability, in the sense of asymptotic convergence, and simultaneously reducing both exchanged information and computational requirements. Note that delays are very common on a daily basis, therefore the presented approach represents a good solution for a wide class of problems, e.g control under actuator/sensor slow dynamics, heavy computation, control over networks, and/or limited resources.

3.1 Compensating Delays

Measurement Delays Assume for the moment that no information is lost. When a measurement $[x(t_i)|t_i]$ is dispatched at a time $t_i \in \pi$, where $x(t_i) \in \mathcal{X}$ is the state value, while t_i is its time-stamp, if there is a measurement delay $\tau_s(t)$, then the information will be available to the controller only at $(t_i + \tau_s(t_i))$, i.e. the controller has to use some piece of information which is outdated and does not correspond to the actual state of the system under control. Therefore, it is necessary to compensate this delay in order to solve the correct control problem. Since a model of the system is available at the controller side, and no mismatch is present, under Assumptions 7-8, it is possible to determine the delay simply by comparing the time-stamp with the global time t, i.e. $\tau_s(t_i) = (t - t_i)$. By means of forward prediction through the local model, possible since it is known what input is applied to the plant (no actuation delay), one can obtain the state prediction

$$\overline{x}(t_i + \tau_s(t_i)) = x(t_i + \tau_s(t_i)). \tag{8}$$

Theorem 9. (Measurement Delay Compensation).
Given the closed loop system (1),(3)-(4), if

i) Theorem 3 is satisfied in the nominal case, i.e. without measurement delays.
ii) $T_p > \overline{\tau_s} + \overline{\delta}$.

Then, $\lim_{t\to\infty} \|x(t)\| = 0$.

Proof. The proof follows from Theorem 3, when the state prediction (8) is used to compensate the delay $\tau_s(t_i)$. More details can be found in Findeisen (2006).

Compensation of Actuation Delays Less trivial is the compensation of computational and actuation delays. In this case, in fact, if the delays are nondeterministic, the actual applied input is not known for sure to the controller. Thus, it is not possible to obtain (8) correctly since $u^*(\cdot)$ is not uniquely determined. As formerly stated, this kind of delays are common in real application, e.g. due to slow actuator dynamics or reduced computational capabilities, and if not explicitly considered can worsen considerably the closed loop performance, or bring to instability. To solve this problem, in some way the applied input must be made deterministic. This can be achieved by using future input trajectories and buffer them in the actuator till the moment they can be used, see Alldredge and M. S. Branicky (2008); Findeisen and Varutti (2009); Varutti and Findeisen (2008). In fact, since Assumption 6 must hold, one can consider the worst case for $\tau_a(t)$ and $\tau_c(t)$, namely $\overline{\tau_a}$ and $\overline{\tau_c}$, in which the state prediction

$$\overline{x}(t_i + \tau_s(t_i) + \overline{\tau_a} + \overline{\tau_c}) = x(t_i) + \int_{t_i}^{t_i+\tau_s(t_i)+\overline{\tau_a}+\overline{\tau_c}} f(x(\tau), u(\tau))d\tau \tag{9}$$

is obtained by using the measurement $x(t_i)$. (9) is then used to solve the optimal control problem and the corresponding solution is despatched to the actuator with a new time-stamp, buffered and used once its time-stamp matches with the global time t, i.e.

$$u^*(\tau; \overline{x}(t_i + \tau_s(t_i) + \overline{\tau_a} + \overline{\tau_c})), \tag{10}$$

for $\tau \in (t_i + \tau_s(t_i) + \overline{\tau_a} + \overline{\tau_c}), t_{i+1} + \tau_s(t_{i+1}) + \overline{\tau_a} + \overline{\tau_c})]$ is sent as $[u^*(\cdot)|(t_i + \tau_s(t_i) + \overline{\tau_a} + \overline{\tau_c})]$. The overall algorithm is reported in Algorithm 1, Appendix A. It can be proved that under Assumption 7, and

$$T_p > \overline{\delta} + \overline{\tau_s} + \overline{\tau_a} + \overline{\tau_c}, \tag{11}$$

Algorithm 1 stabilizes the delayed system.

Theorem 10. (Worst Case Compensation).
Given the nonlinear continuous time system (1) and the the predictive controller obtained from (3)-(4), and (9), by applying Algorithm 1, under (11) the closed loop system is stable, in sense of asymptotic convergence, if the nominal controller, i.e. the controller subject to no delays obtained from Theorem 3, stabilizes the system.

Proof. The proof follows directly from Theorem 3, and 9, first, by proving recursive feasibility, and then convergence –see Findeisen (2006); Findeisen and Varutti (2009); Varutti and Findeisen (2008) for more details–.

Remark 11. Assumption 7 is required to have a common time-frame among the components. This can represent a problem for fast dynamical systems, since the state-of-the-art synchronization algorithms cannot guarantee high precision.

3.2 Information Loss Compensation

It has been assumed till now that the communication is not affected by any information loss. In reality, however, information losses might occur due, for example, to unreliable communication media, or some temporary components failure. As in the delay case, the major problem is represented by losses/failures in the actuation channel. In fact, if an information loss $S_i = 0$ occurs for $t_i \in \pi$, the controller can still use the last available state

$$x(t_k), \text{ for } t_k \in \pi, \text{ s.t. } S_k = 1, \tag{12}$$

and the nominal model for (1) to calculate the prediction

$$\overline{x}(t_k + \sum_{j=k}^{i} \tau_s(t_j) + \overline{\tau_a} + \overline{\tau_c}), \text{ for } t_j \in \pi, \text{ s.t. } S_j = 0, \quad (13)$$

by using the compensation approach presented in Section 3.1, if the applied input is uniquely determined.

Remark 12. Notice that $t_k + \sum_{j=k}^{i} \tau_s(t_j)$ can be substituted by the global time t, moment in which the controller receives a new measurement.

On the contrary, if a dropout $A_i = 0$ occurs, then the control input is not uniquely known at the controller side, and thus (13) cannot be accurately calculated. In Varutti and Findeisen (2008); Findeisen and Varutti (2009) a solution for the problem was found by using *prediction consistent feedbacks*, i.e. feedbacks that under information losses are able to keep the difference between state prediction and actual state negligible and hence guarantee convergence under a limited amount of dropouts.

In this paper, a different approach is considered. In particular, the following assumptions are made:

Assumption 13. An acknowledge mechanism is available on the actuator side.

Assumption 14. The acknowledgments have high priority, and they cannot be dropped.

Assumption 15. The acknowledgments are delivered instantaneously.

This is equal to saying that for every input received by the actuator an acknowledgment with the time-stamp of the latest successfully delivered information is sent back to the controller.

We show later how Assumption 15 can be relaxed. On the contrary, Assumption 14 is a fundamental condition since it is well known from theoretical results that no handshake protocol can solve a coordination problem under acknowledgment losses –see "The two Generals Paradox", Tanenbaum (2008)–. In the case of chemical processes, however, timely submission of acknowledgments is often not a problem, since there are frequently slow measurement and actuation devices. The used communication networks are often sufficiently fast and provide the possibility of having high priority acknowledgments. Although restrictive, these conditions are necessary to allow the controller to correctly reconstruct the applied input sequence. Algorithm 2 in Appendix A illustrates the procedure to compensate simultaneously delays and information losses. Differently from Algorithm 1, the entire control input trajectory $u^*(\tau; x(t_i))$, for $\tau \in (t + \overline{\tau_a} + \overline{\tau_c}, t_i + T_p]$ is sent to the actuator with time-stamp $(t + \overline{\tau_a} + \overline{\tau_c})$. In this way, when some information is dropped either in the down- or in the up-link, the actuator can still utilize the old input trajectory to control the system. Note that in an event-based setup measurement losses are transparent to the controller. This means that no new control input is generated and dispatched to the actuator. However, since the whole control trajectory is sent, if the number of consecutive losses S_i is less than T_p, the actuator can still apply the latest received input. On the other hand, the controller can establish immediately that some information has gone lost thanks to the timer that is implicitly set by time-stamping the control inputs. In fact, if the current time exceeds the former time-stamp, from Assumption 15, it is known for sure that no new control input has arrived. An alternative would be to index the control trajectories and use an error mechanism instead, i.e. an error is sent every time some control information is lost. However, in the asynchronous case utilizing acknowledgments is more efficient, since the actuator does not have to wait till the next successfully received trajectory to realize that some information went lost. Finally, note that a non-resend policy has been chosen, i.e. if a control trajectory is lost, the controller does not transmit it again but it simply records the event in order to update its local copy of the currently applied control input. This seems to be a more logical choice since the sequence would probably arrive when its applicability time is already expired.

Theorem 16. (Convergence Under Information Losses). Given the closed loop system (1), (3)-(4), (13) if

i) Assumptions 13-15 are satisfied.
ii) Algorithm 2 is used.
iii) The prediction horizon T_p is such that

$$T_p > \overline{\delta} + n \cdot \overline{\tau_s} + \overline{\tau_c} + m \cdot \overline{\tau_a}, \quad (14)$$

where n,$m \in \mathbb{N}/\{0\}$ represent, respectively, the number of consecutive losses in the measurement and in the actuation.

Then, $\lim_{t\to\infty} \|x(t)\| = 0$.

Proof. The proof follows from Theorem 3 and 9. The use of acknowledgments allows the controller to reconstruct in real time the correct applied control sequence. Thus, feasibility and convergence can be proved.

As formerly stated, Assumption 15 can be relaxed by allowing the acknowledgments to be subject to nondeterministic delays $\tau_{ack}(t)$. In this case, however, the following assumption is required.

Assumption 17. $\tau_{ack}(t)$ is nondeterministic with arbitrary probability distribution, but limited, i.e $\tau_{ack}(t) \in [0, \overline{\tau_{ack}}]$.

By using a worst case compensation approach similar to Algorithm 1, and 2, one can ensure asymptotic convergence of the closed loop system by considering, instead, the state prediction

$$\overline{x}(t_k + \sum_{j=k}^{i} \tau_s(t_j) + \overline{\tau_a} + \overline{\tau_c} + \overline{\tau_{ack}}), \quad (15)$$

for $t_j \in \pi$, such that $S_j = 0$.

Corollary 18. (Convergence Under Information Losses). The closed loop system (1), (3)-(4), is stable, in the sense of asymptotic convergence, if

i) The conditions for Theorem 3, 9, 10, and Assumption 17 are satisfied.
ii) A modified variant of Algorithm 2, such that the prediction (15) is used and $[u^*(\tau; x(t_i))|ts]$, for $ts = t + \overline{\tau_{ack}} + \overline{\tau_s} + \overline{\tau_c} + \overline{\tau_a}$, $\tau \in [t + \overline{\tau_{ack}} + \overline{\tau_s} + \overline{\tau_c} + \overline{\tau_a}, t_i + T_p]$ is utilized.
iii) $T_p > n \cdot \overline{\tau_s} + \overline{\tau_c} + m \cdot \overline{\tau_a} + \overline{\tau_{ack}}$, where n,$m \in \mathbb{N}/\{0\}$ represent the number of consecutive losses in the measurement/actuation.

4. SIMULATION RESULTS FOR A CSTR

The formerly presented method has been applied to a CSTR, where an irreversible exothermic reaction, $A \rightarrow B$, takes place in a constant volume, cooled by a single coolant stream at temperature T_c –see Figure 2–. The overall

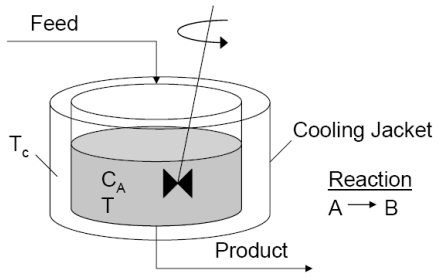

Fig. 2. Scheme of CSTR under study.

system is modeled as:

$$\dot{C}_A(t) = \frac{F}{V}(C_{Af} - C_A(t)) - k_0 C_A(t) e^{-\frac{E}{RT_r(t)}},$$
$$\dot{T}_r(t) = \frac{F}{V}(T_{rf} - T_r(t)) - \frac{k_0 \Delta H}{\rho c_p} C_A(t) e^{-\frac{E}{RT_r(t)}} + \frac{UA}{\rho c_p V}(T_c(t) - T_r(t)).$$

The meaning and the values of all the parameters are explained in Henson and Seborg (1997). Under the nominal condition $T_c^{nom} = 103.4$ K, the system has the three equilibrium points depicted in Figure 3. The objective is to stabilize the unstable saddle point $(0.52, 398.97)$ by manipulating the control input $u(t) = T_c(t)$ under the input constraints $T_c(t) \in [275, 370]$ K. If we represent

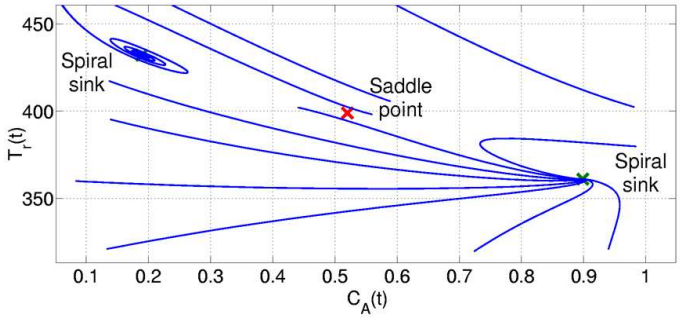

Fig. 3. Phase plot of the system for the nominal case $T_{c,nom} = 302$ K.

the state as the vector $x(t) = [C_A(t) \quad T_r(t)]^T$, the cost functional to be minimized is given by

$$J(u, x) = \int_{t_i}^{t_i + T_p} (x^T Q x + u^T Q u) d\tau,$$

where $Q = I$, $R = 1$, and $T_p = 1.5$ min. For sake of simplicity, the control trajectory is held constant between consecutive recalculation times. Terminal penalty and terminal region constraints have been chosen such that closed loop stability in the nominal case is achieved.

In Figure 4, the results for the asynchronous nominal case (no delays and no failures) are presented. Compared to a classical sampling approach with constant recalculation interval $\delta = 0.15$ min, the asynchronous controller presented in Theorem 3 obtains extremely similar results by saving up to 30% of computational effort and exchanged information. The partition π is implicitly determined by the absolute error on the product concentration, i.e. the sensor regularly checks the concentration $C_A(t)$, but it sends a measurement to the controller only when

$$\|C_A(t) - 0.52\| > \epsilon, \text{ with } \epsilon = 2 \cdot 10^{-3},$$

for longer than 0.15 minutes. In a second simulation

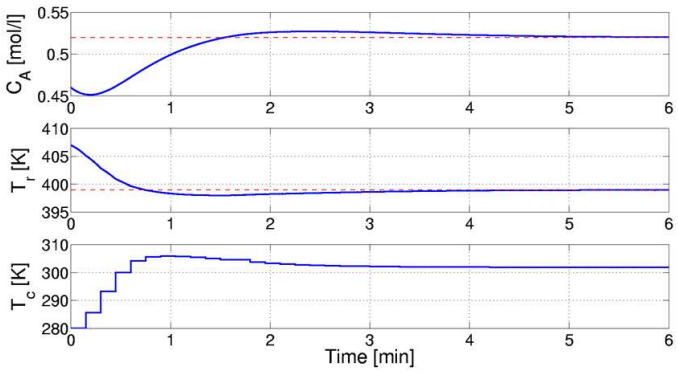

Fig. 4. Event-based controller under nominal conditions.

both measurement and actuation delays are considered. It is assumed that τ_s lies between 0 and 15 seconds, while τ_a between 0 and 5 seconds. Both are modeled as uniform variables. For sake of simplicity, no computational delay is considered. However, notice that this delay could be implicitly considered as part of the input delay τ_a. Furthermore, it is assumed that the probability loss at the actuator side p_a is equal to 5%, while no information from the sensor is lost. It is also supposed that Assumption 15 is verified. The results for the compensated and the uncompensated case are presented in Figure 5. As one can see, the proposed method is able to stabilize effectively the unstable saddle point with performance comparable to the nominal case, reported in Figure 4. Note that the controllers presented in Findeisen and Allgöwer (2004); Chen et al. (2000) are not able to handle delays on the actuation side at all.

5. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper, it was shown how PC can be used to control event-based/asynchronous systems, such as chemical processes, in which the recalculation times do not need to be known a priori. Moreover, it was depicted how delays and information losses/failures on the actuators/sensors, common in control systems, need to be taken into account to avoid instability. Whereas delays can be compensated easily with forward prediction, one can exploit bidirectional communication with the actuators in order to establish an acknowledgment mechanism to counteract information losses/failures. Two algorithms able to guarantee asymptotic convergence for nonlinear continuous time systems were presented. Through the simulation of a CSTR, it was shown firstly that asynchronous PC can actually reduce both computational requirements and exchanged information, but also compensate effectively delays and information losses while keeping closed loop stability and good performance. Future work should concentrate on how to include directly in the optimization problem the

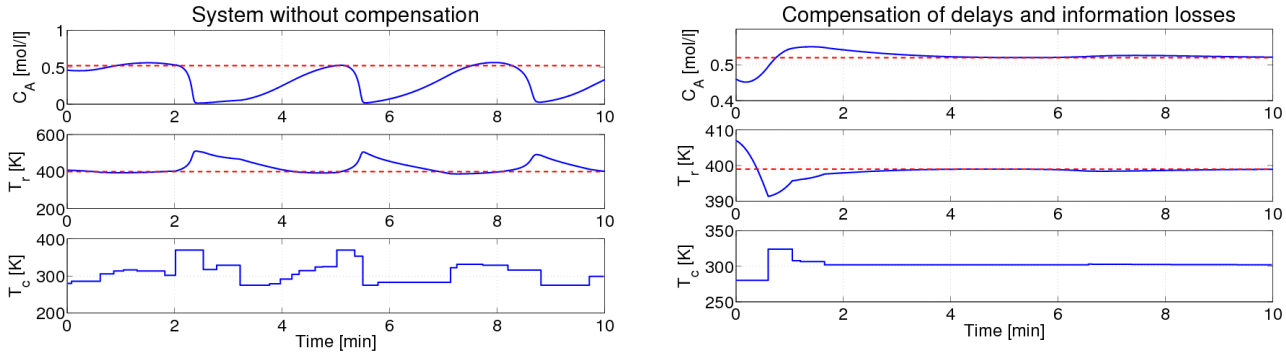

Fig. 5. Results for the closed loop system subject to delays and information losses with and without compensation.

exchanged information. Moreover, the method should be extended to include also robustness.

REFERENCES

Alldredge, G. and M. S. Branicky, V.L. (2008). Play-back buffers in networked control systems. evaluation and design. *Amer. Cont. Conf.*, 3106–3113.

Brockett, R. and Liberzon, D. (1998). Quantized feedback systems perturbed by white noise. In *Proc. 37th IEEE Conference on Decision and Control*, volume 2, 1327–1328.

Chen, W., Ballance, D., and O'Reilly, J. (2000). Model predictive control of nonlinear systems: Computational delay and stability. In *IEE Proceedings, Part D 147(4)*, 387–394.

Findeisen, R. (2006). *Nonlinear Model Predictive Control: a Sampled-Data Feedback Perspective*, volume 8. VDI Verlag.

Findeisen, R. and Allgöwer, F. (2004). Computational delay in nonlinear model predictive control. In *Proc. Int. Symp. Adv. Contr. of Chem. Proc.*, 427–432. Hong Kong, PRC.

Findeisen, R. and Varutti, P. (2009). Stabilizing nonlinear predictive control over nondeterministic communication networks. In L. Magni, D. Raimondo, and F. Allgwer (eds.), *Nonlinear Model Predictive Control: Towards New Challenging Applications*, Lecture Notes in Control and Information Sciences. Springer. In printing.

Fontes, F. (2001). A general framework to design stabilizing nonlinear model predictive controllers. *Sys. Contr. Lett.*, 42(2), 127–143.

Heemels, W.P., Sandee, J.H., and Van den Bosch, P. (2008). Analysis of event-driven controllers for linear systems. *Inter. J. of Cont.*, 81, 571–590.

Henson, M. and Seborg, D. (1997). *Feedback Linearizing Control*. Prentice Hall.

Mayne, D.Q., Rawlings, J.B., Rao, C.V., and Scokaert, P.O.M. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(6), 789–814.

Tanenbaum, A.S. (2008). *Computer Networks*. Prentice Hall PTR, 4th edition.

Varutti, P. and Findeisen, R. (2008). Compensating network delays and information loss by predictive control methods. (submitted to ECC09).

Appendix A

Algorithm 1 (Worst Case Compensation)

$\forall t_i \in \pi$; t = current time;

Sensor:

(1) Measure $x(t_i)$.
(2) Send $[x(t_i)|ts]$, with $ts = t_i$, to the controller.
(3) Go to 1.

Controller:

`buffer` $= [x(t_i)|ts]_{old}$;
`control_input` $= \{[u^*(\cdot)|ts_0]\}$;
(1) If $[x(t_i)|ts]_{new}$ arrives
  a) If $ts_{new} \leq ts_{old}$, then discard.
  b) Else `buffer` $= [x(t_i)|ts]_{new}$.
(2) $\tau_s = (t - t_i)$ for $t_i = ts$.
(3) Calculate (9), from $u^*(\cdot; x(t_i)) \in$ `control_input`.
(4) Solve the o.c.p. for (9) $\longrightarrow$
  $u^*(\tau; x(t_i))$, for $\tau \in (t_i + \tau_s(t_i) + \overline{\tau_a} + \overline{\tau_c}, t_{i+1} + \tau_s(t_{i+1}) + \overline{\tau_a} + \overline{\tau_c}]$.
(5) Send $[u^*(\tau; x(t_i))|ts]$, with $ts = (t + \overline{\tau_a} + \overline{\tau_c})$.
(6) Insert $[u^*(\tau; x(t_i))|ts]$ in `control_input`.
(7) Go to 1.

Actuator:

`buffer` $= \{[u^*(\cdot)|ts_0], \ldots, [u^*(\cdot)|ts_n]\}$, for $ts_0 < t < ts_1 \ldots < ts_n$;
`applied_input` $= [u^*(\cdot)|ts_0]$;
(1) If $[u^*(\cdot)|ts]_{new}$ arrives
  a) Insert $[u^*(\cdot)|ts]_{new}$ in `buffer`.
  b) Sort `buffer` by increasing ts.
(2) `temp` = first element of `buffer`.
(3) If $ts_{\texttt{temp}} = t$
  a) `applied_input` = `temp`.
  b) Remove first element from `buffer`.
(4) Go to 1.

Algorithm 2 (Information Loss Compensation)

$\forall t_i \in \pi$; t = current time;

Sensor:

(1) Measure $x(t_i)$.
(2) Send $[x(t_i)|ts]$, with $ts = t_i$.
(3) Go to 1.

Controller:

`buffer` $= [x(t_i)|ts]_{old}$;
`control_input` $= \{[u^*(\cdot)|ts_0]\}$;
`delivered` = `true`;
(1) If $[\cdot|ts]$ arrives:
(2) Case($[x(t_i)|ts]$):
  If $ts_{new} \leq ts_{old}$, then discard.
  Else `buffer` $= [x(t_i)|ts]_{new}$.
  $\tau_s = (t - t_i)$ for $t_i = ts$.
  Calculate (13), from $u^*(\cdot; x(t_i)) \in$ `control_input`.
  Solve the o.c.p. for (13) $\longrightarrow$
  $u^*(\tau; x(t_i))$, for $\tau \in (t + \overline{\tau_a} + \overline{\tau_c}, t_i + T_p]$.
  `delivered` = `false`.
  Send $[u^*(\tau; x(t_i))|ts]$, with $ts = (t + \overline{\tau_a} + \overline{\tau_c})$.
  Wait until `delivered`=`true` OR t $\geq$ ts.
  If `delivered` = `true`, then insert $[u^*(\tau; x(t_i))|ts]$ in `control_input`.
  Else if t $\geq$ ts, then `delivered` = `false`, use old input in `control_input`.
  Go to 1.
(3) Case($[ack|ts]$):
  Set `delivered` = `true`.
  Go to 1.

Actuator:

`buffer` $= \{[u^*(\cdot)|ts_0], \ldots, [u^*(\cdot)|ts_n]\}$,
for $ts_0 < t < ts_1 \ldots < ts_n$;
`applied_input` $= [u^*(\cdot)|ts_0]$;
(1) If $[u^*(\cdot)|ts]_{new}$ arrives
  a) Send $[ack|ts]$ to the controller.
  b) Insert $[u^*(\cdot)|ts]_{new}$ in `buffer`.
  c) Sort `buffer` by increasing ts.
(2) `temp` = first element of `buffer`.
(3) If $ts_{\texttt{temp}} = t$
  a) `applied_input` = `temp`.
  b) Remove first element from `buffer`.
(4) Go to 1.