IFAC Conference on Advances in PID Control
PID'12
Brescia (Italy), March 28-30, 2012

Tracking Simulation Based on PI Controllers and Autotuning

Mats Friman and Pasi Airikka

Metso, Automation Business Segment, Tampere, Finland (e-mail: firstname.lastname@metso.com)

Abstract: A mechanism for automatic update of tracking simulator parameters is suggested. A tracking simulator is a simulator, which runs in real time, and which corrects its own behavior (models) by comparing real process measurements to simulator outputs. Typically, a process simulator, no matter if static or dynamic, cannot adapt its behavior to reality, but the tracking simulator has this ability due to its integrated update mechanism. A tracking simulator is commonly used for state estimation of non-linear process models (simulation models). The suggested invention utilizes standard PI controllers, which enables fast update of parameters. A major advantage with the PI controllers is that autotuning can be applied and, hence, no tuning parameters need to be plucked out of the air. With the proposed innovation, it will be easy to extend an ordinary process simulator to a tracking simulator, which can be used for many purposes, including predictive and fault tolerant control, soft sensors, prediction of future plant behavior, parameter estimation, and plant optimization.

Keywords: Dynamic Simulation, Tracking Simulation, PI control, Autotuning

1. INTRODUCTION

There are many reasons for using dynamic simulation. The threshold for using simulation tools is basically a question of enthusiasm and motivation since there is a very large selection of high-quality solvers, modelling tools, and libraries available for free. Many of these are open source (Fritzson, 2003, Åkesson et al. 2009).

Traditionally, it has been time-consuming and expensive to build and maintain simulation models, and this has efficiently hindered their industrial use in daily operation. This problem is gradually disappearing. More and more engineering platforms utilize dynamic simulation already at process design phase, and those models could be re-used during operation, without any additional cost. Process & Instrumentation diagrams that previously have been used to document process design are already replaced with dynamic simulations models. With such systems, simulation models can easily be generated from a menu, for free.

Despite the long history of using dynamic simulation, it is mostly used off-line. However, there are really huge opportunities for using dynamic process models on-line, for example in monitoring, control, and optimization tasks.

Examples of on-line usage of dynamic models include MPC controllers (Richalet et al., 1978) and Kalman Filters (Kalman, 1960). These applications, however, traditionally employ linear models. Even though several nonlinear MPC applications have been reported (Rawlings et al., 1994), the advantage of non-linear models is not justified with the added complexity in the optimization phase.

Simulation models are seldom re-used as such, e.g. from training simulators. Sometimes a plant may use linear models for MPC control and nonlinear simulations models for operator training for the same unit process. In that case, the same unit process model must be modelled twice.

Process models are built with a wide degree of complexity and accuracy. In order to clarify this aspect, standards have been developed. For example fossil fuel power plants have been divided into three levels according to their complexity and accuracy (ANSI/ISA, 1993).

A difficulty with simulation models is that the state of the process cannot, as such, be copied into the simulator. If we, for example, want to simulate an industrial process from the current state, e.g. to show what will happen in the near future, we should first initialize the state of the dynamic model to match the state of the process. This task is far from trivial. Even though some measurements are available on-line, the state of the dynamic model may depend on some actions that were implemented long ago.

One way of estimating the state of the process is to run the simulator in parallel with the process with the inputs of the process connected to the simulator, see Fig. 1a. With this connection, the outputs of the process and simulator will, however, usually diverge due to unknown or varying parameters. To overcome this difficulty, a tracking simulator is used (Fig. 1b).

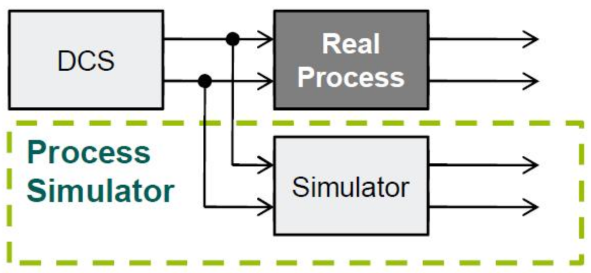

Fig. 1a. A process simulator may be connected to a real process and run in real-time. However, outputs from real process and simulator will usually diverge if no update mechanism is used.

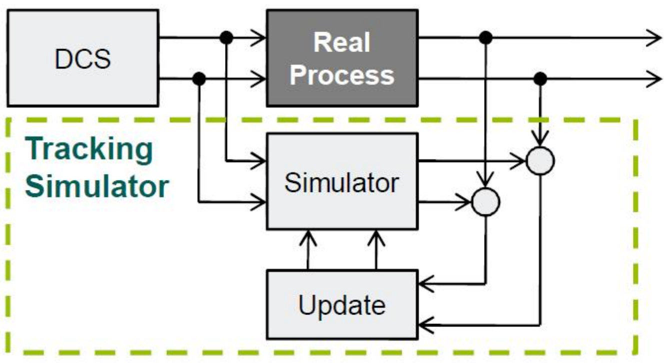

Fig. 1b. A tracking simulator is a process simulator with an update mechanism, which tries to match simulator outputs with real process outputs.

A tracking simulator is a simulator that is run in real-time, in parallel to the process. It utilizes process measurements to update its states and parameters, so that simulator outputs and process measurements converge.

A problem with tracking simulators is that the adapting mechanism may be slow due to the primitive update mechanism used. It is common to use a mechanism that updates parameters with an amount proportional to the deviation between simulator and process measurement. Although being simple, this mechanism is sometimes very slow. Another problem is the selection of the numerical values for the update, since they are usually tuned with trial and error.

In this paper we present an innovative solution, which overcomes the problems discussed above. We use PI controller for speeding up the estimation of simulation model parameters, and we use autotuning (Åström and Hägglund, 1984) techniques to make the parameter selection elegant and systematic. We believe that the solution suggested here will be useful when the use of dynamic simulation in daily operation will grow.

2. PROPOSED STRUCTURE

Before describing the proposed solution we justify out approach with a very simple motivating example. Despite being simple, it well demonstrates the drawbacks with current tracking simulators.

2.1 Motivating Example: Cooling Coffee

We use a simplified example to justify our idea and to demonstrate the problems with known technology. The example simulates cooling of coffee (Fig. 2), with an initial temperature of 60 °C and the surrounding room temperature alternating around 20 °C with the amplitude of 5 °C, and period of 600 s. The dynamics is a quite typical first-order system with the coffee temperature being the state variable.

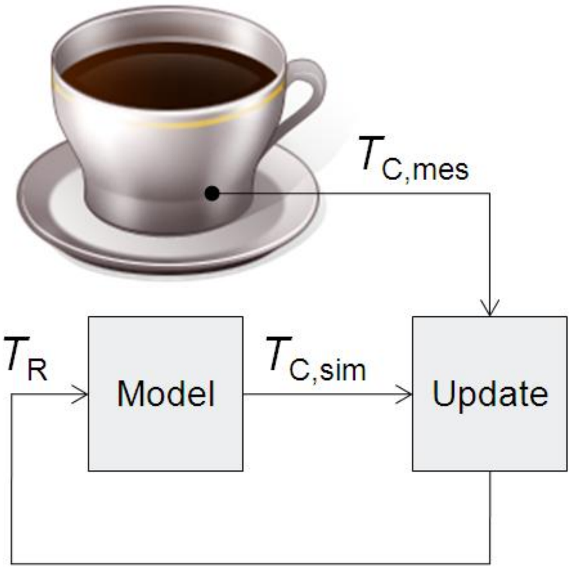

Fig. 2. Cooling Coffee example. The purpose is to estimate room temperature (T_R) by comparing real ($T_{C,mes}$)and simulated coffee temperatures ($T_{C,sim}$). The simulation block ("Model") updates coffee temperature based on estimation of room temperature, which is estimated in the update block ("Update").

The dynamics of the system is described with the equation

$$\tau \frac{dT_C}{dt} = T_R - T_C \tag{1}$$

where variable T denotes temperature, with subscripts R for room and C for coffee. τ is a time constant, which is assumed to be known ($\tau = 120$ s).

Next we will show how we estimate (update) room temperature (T_R) by comparing real and simulated coffee temperatures. According to known technology (see e.g. Nakaya et.al, 2008) a parameter may be updated according to

$$p(k) = p(k-1) + Ke(k) \tag{2}$$

where $e(k)$ is the deviation (difference) between simulated and measured value (at time instant k) and K is an update constant. The parameter $p(k)$ updates its value based on its previous value $p(k\text{-}1)$. The approach suggested in this paper uses decentralized PI control, i.e.

$$p(k) = p(k-1) + K_p(e(k) - e(k-1)) + \frac{K_p}{T_i}(e(k)) \tag{3}$$

where K_p and T_i are PI controller parameters.

The results of a Cooling Coffee simulation experiment, where the two alternative update mechanisms (Eq. 2 and Eq.3) have been used, are shown in Fig.3-4. The update constants where K = -0.1667 for the conventional (Eq. 2) and $K_p = -4$; $T_i = 10$ for the method suggested in this paper (Eq. 3). The sampling time was 15 s, and the measurement was delayed with one sample in order not to favour gain selection close to infinity.

From Fig. 3 and 4 we conclude that we get a faster estimation of room temperature by using the update mechanism inspired by a PI controller (Eq. 3) compared to the single-parameter update mechanism according to Eq. 2.

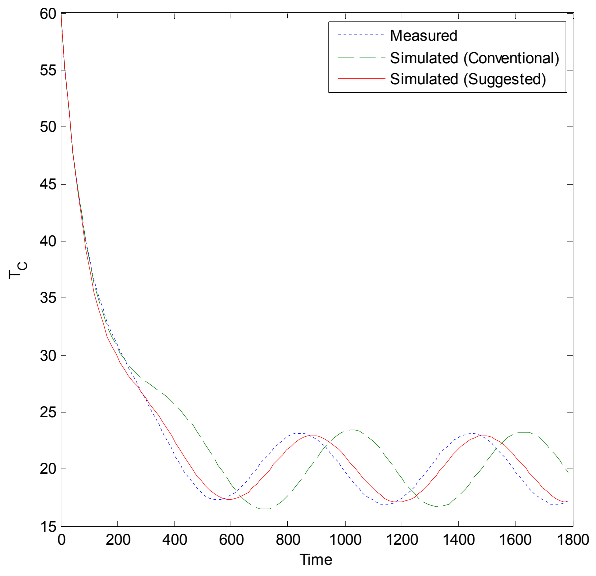

Fig. 3. Simulations of coffee temperature, where the room temperature has been updated with Eq.(2) (dashed-line) and Eq.(3) (solid line). The real measured temperature (dotted line) is included for reference.

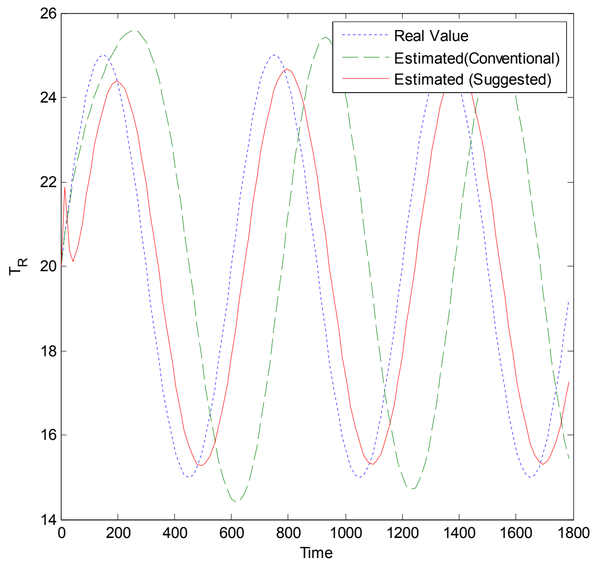

Fig. 4. Estimation of room temperature in the Cooling Coffee example, where the room temperature has been updated with Eq.(2) (dashed-line) and Eq.(3) (solid line). The unknown room temperature (dotted line) is included for reference.

2.2 *General Structure*

The motivating example introduced some benefits (most notably the speed of update) of using PI controllers for tracking simulation. A block diagram of the suggested tracking simulator is shown in Fig.5.

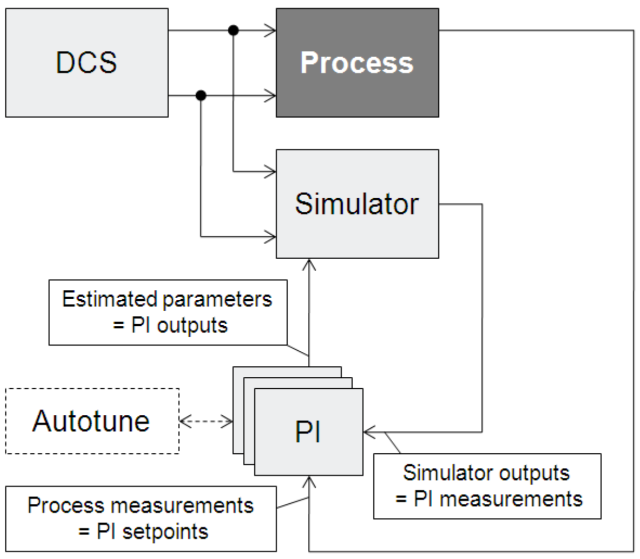
DCS
Process
Simulator
Estimated parameters = PI outputs
Autotune
PI
Simulator outputs = PI measurements
Process measurements = PI setpoints

Fig. 5. A tracking simulator update mechanism applied with PI controllers. The connections to PI controller terminals are the following: process measurement is connected to PI setpoint; simulator output is connected to PI measurement; estimated parameter is PI control signal. An autotuner is used to tune the PI controller parameters.

As seen from Fig. 5, there is a simulator running in parallel with the process. The inputs to the process are connected to the simulator. The PI controllers are connected as follows:

- outputs of the process, usually process measurements (controlled and uncontrolled) are connected to PI controllers setpoints
- outputs of the simulator (corresponding process measurements) are connected to PI controllers measurements
- the output (control value) of the PI controller is the unknown parameter, which will be used by the simulator

Above we already discussed that a (momentary) drawback of the suggested update mechanism is that we need two parameters instead of one. However, since we have described the tracking simulation problem as a control problem, which can be solved with decentralized PI control, we can use autotuning, which eliminates the need of manual parameter selection.

2.3 Summary of Implementation

Next we will shortly summarize the implementation from a practical point of view. We use the 3-input-2-output process shown in Fig. 6. as an example process. Two inputs are known (solid lines) and one is unknown (dashed line). Moreover, there are unknown process parameters, but since there are only two measured outputs, we can only estimate one parameter in addition to the unknown input.

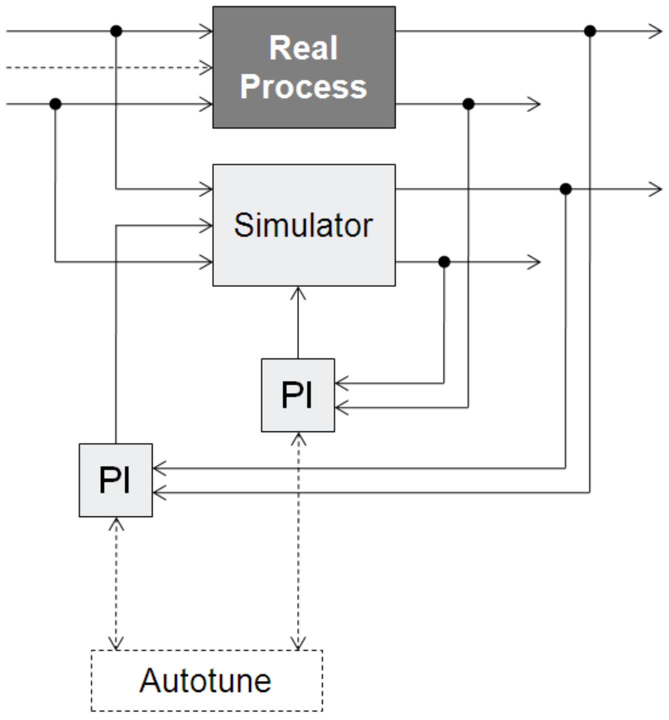

Fig. 6. An example 3-input-2-output process. One unknown input (dashed line) and one unknown parameter are identified with two PI controllers.

The implementation of the tracking simulator is done as follows. We assume that a simulator that matches the real process is available.

1. Known process inputs (controls) are connected to simulator.
2. Selected unknown inputs/parameters are connected to PI controllers. All PI controllers are in manual mode with output equal to an educated guess of the particular input/parameter.
3. Ad hoc values are used for all other inputs/parameters.
4. Each PI controller measurement is connected to a simulator output and PI controller setpoint is connected to corresponding process measurement.
5. Each PI controller is autotuned and connected to automatic.
6. Some PI controllers may need retuning because of interactions between control loops.

In practice, input flows of industrial processes are usually known (e.g. valve positions), but properties of input flows, e.g. temperature, concentrations, pressure, are often unknown.

2.4 Alternatives to PI control: The Nelder-Mead and the MIT Methods

It is well known that PI controllers can only be used when the sign of process gain is known and does not change with time. In the Cooling Coffee example, where we tried to estimate the room temperature, it was clear that a higher room temperature implies a higher coffee temperature (ie. it has a positive process gain). Modest changes in the magnitude are tolerated, but constant sign of the gain is a prerequisite for successful PI control. However, this is not always guaranteed in the parameter estimation task, and hence other solutions must be on hand for the tricky cases. One possible solution is to apply the well known Nelder-Mead algorithm (Nelder and Mead, 1965).

Another alternative is to apply the MIT rule, which is well known from adaptive control theory (see e.g. Åström and Wittenmark, 1989). Here the idea is to multiply the error seen by the PI controller with the "sensitivity derivative". In that case, gains and time constants may successfully be estimated with PI controllers.

3. INDUSTRIAL EXAMPLE

We will use an economizer from a biomass fired power plant as an example. The economizer is a heat exchanger that utilizes hot flue gases in order to warm up the feed water.

The model has 13 inputs (parameters are also counted as inputs): temperature before economizer (feed water and flue gas), pressure drop over heat exchanger (feed water and flue gas), specific heat capacity (feed water and flue gas), heat transfer coefficient, density (feed water and flue gas), total heat exchanger volume (feed water and flue gas), capacity value (i.e. volumetric flow obtained by a pressure drop of 1 bar) (feed water and flue gas). There are two outputs, i.e. output temperatures (feed water and flue gas).

Real industrial data, collected from a biomass powered boiler was used in the examples.

With two outputs in the model, and corresponding process measurements, we can update two unknown parameters. We chose to update the following two parameters: 1) the heat transfer coefficient, because it is useful to know from maintenance and economic perspective and 2) the specific heat capacity of flue gas because it has a large impact on the model and large variations in the data due to variation of moisture content in the biomass fuel.

The variables of interest are shown in Fig. 7. We use variable T for temperature with indices "F" for flue gas, "W" for feed water, "in" for measurement before, and "out" for measurement after economizer. $C_{p,F}$ is the specific heat capacity of flue gas and hA is the heat transfer coefficient in the economizer.

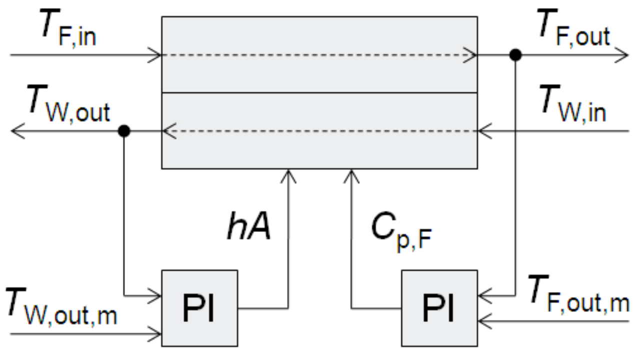

Fig. 7. Schematic diagram of the economizer example. Flue gases with input temperature $T_{F,in}$ is used to warm up water with input temperature $T_{W,in}$. Water temperature after economizer ($T_{W,out}$) is used to estimate hA, i.e. the heat transfer coefficient. The flue gas temperature after economizer ($T_{F,out}$) is used to estimate the specific heat capacity of flue gas.

In addition, the connection of two PI controllers, used to estimate the two unknown parameters hA and $C_{p,F}$, are illustrated in Fig. 7. The setpoints of the PI controllers are the temperatures, which are measured from the real plant, i.e. water and flue gas temperatures after the economizer ($T_{W,out,m}$ and $T_{F,out,m}$).

An example autotuning experiment is shown in Fig. 8-9. During time t = 10-20 min the feed water loop was identified and during t = 25-35 min the flue gas loop was identified. We used standard relay autotuning, with relay amplitudes d = 20 kW/°C for heat transfer coefficient and d = 0.4 kJ/(kg/K) for the specific heat capacity value. We employed the "integrator-plus-deadtime" model as described by Friman and Waller (1994) for process identification. The model parameters were k = 036; L = 0.29 for the feed water loop and k = 153; L = 0.25 for the flue gas loop.

For PI controller tuning we used the IMC tuning guidelines suggested by Chien and Freuhauf (1990), and selected the target speed T_{CL}, i.e. the closed-loop time constant as T_{CL} = 5L. The PI parameters where then K_C = 2.94; T_i = 3.21 for the feed water loop and K_C = 0.0080; T_i = 2.75 for the flue gas loop with these selections.

In Fig. 8-9 is furthered shown that the autotuning experiment in successful, since the simulated temperatures in Fig.8. nicely follow their setpoint values, which have been recorded from an industrial plant, when the controllers are connected to automatic model for t> 35 min. The unknown parameters settle quite well (Fig.9, from about 40 min forward).

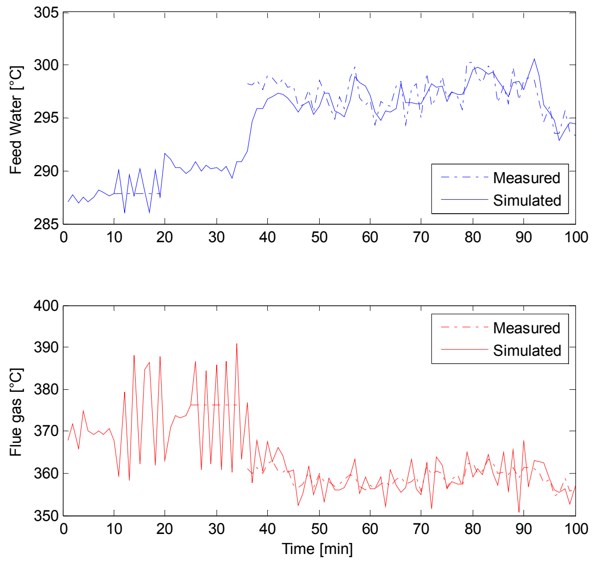

Fig. 8. Simulated values of feed water (top) and flue gas (bottom) temperatures during autotuning experiments. The feed water loop is autotuned at t = 10-20 min, and the flue gas loop is autotuned at t = 25-35 min. At t = 35 min both loops are connected in automatic mode.

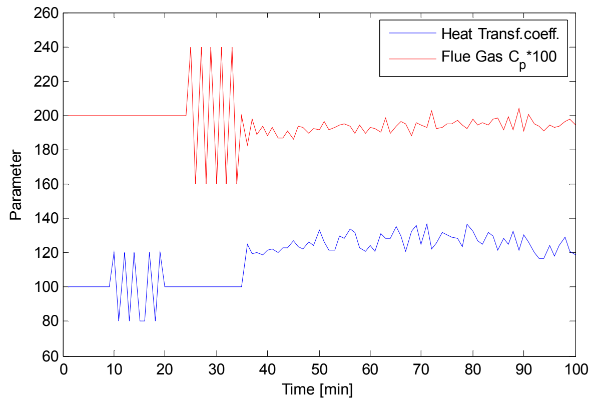

Fig. 9. Process parameters (PI control outputs) during the autotuning experiments in Fig. 8.

4. CONCLUSIONS

In this paper we have suggested and tested the idea of using PI controllers for tracking simulation, instead of the well known simple update mechanism in Eq. 2. We have successfully tested the idea with a decentralized 2x2 control system. However, results not shown here have demonstrated that the idea can be applied also on larger systems. In fact, since the idea is similar to ordinary decentralized control, we see no limitations in the number of parameters that can be updated according to the ideas presented here.

The parameter identification is, however, not as easy as decentralized control. In many cases there are parameters that do not have a clear direction. For example, in the Cooling Coffee example it was clear that the room temperature has a positive impact on coffee temperature. However, if our task had been to estimate the time constant, it would not be possible to apply PI control (Eq. 3) nor known technology (Eq. 2). However, we can use the MIT rule, which is well known from adaptive control, or a slower but more general solution, such as the Nelder-Mead algorithm.

Moreover, there are no structural requirements on the simulation models. The presented idea may be used on ODE/PDE/DAE models regardless of properties, such as model size, complexity, accuracy, and solver.

There are some open questions related to decentralized control, including variable pairing, decoupling, stability, closed-loop performance, and control structures (e.g. ratio and cascade controllers). However, since those challenges are basically identical to those of decentralized process control, we have not discussed these issues in detail. We basically notice two main differences between our approach and traditional decentralized process control. 1) Since processes are never designed with parameter estimation in mind, it is expected that the process, as seen by the controllers used for tracking simulation, may be very nonlinear. This problem is best handled by using modest target speed during controller tuning for the loops that enclose nonlinear behaviour. 2) It is important to understand the characteristics of each parameter during tuning and select the control speed accordingly. For example, if we are estimating a parameter that changes slowly (e.g. a heat transfer coefficient), we can favourable select a very sluggish target speed for the controller, and when we are updating some rapidly changing phenomena (usually some unknown property of input flows) we should use more aggressive controller tuning.

REFERENCES

Åkesson J., Gäfvert, M., Tummescheit, T. (2009). JModelica - an Open Source Platform for Optimization of Modelica Models. *Proceedings of MATHMOD 2009 - 6th Vienna International Conference on Mathematical Modelling*, Vienna, Austria.

ANSI/ISA-77.20-1993 (1993), Fossil Fuel Power Plant Simulators - Functional Requirements, Approved 1993, 43 p., *North Carolina, Instrument Society of America.*

Åström, K. J. and Hägglund, T. (1984). Automatic Tuning of Simple Regulators with Specifications on Phase and Amplitude Margins. *Automatica*, 20, 645-651.

Åström, K. J. and Wittenmark, B. (1989). Adaptive Control. *Addison Wesley*.

Chien, I. L. and Freuhauf, P. S. (1990). Consider IMC Tuning to Improve Performance. *Chem. Eng. Progr.* 1990 Oct., 33-41

Friman, M. and Waller K.V. (1994). Autotuning of Multiloop Control Systems. *Ind.Eng.Chem.Res,* Vol. 33, No. 7.

Fritzson, P. (2003). Principles of Object-Oriented Modeling and Simulation with Modelica 2.1, *Wiley-IEEE Press,* ISBN 0-471-471631.

Nakaya, M., Seki, T., Kawaguchi, K., Onoe, Y., Ootani, T. (2008). Model Parameter Estimation by Tracking Simulator for the Innovation of Plant Operation. *Proceedings of the 17th IFAC World Congress*, p. 2168-2173, Seoul, Korea.

Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *Computer Journal* 7: 308–313.

Kalman, R. E. (1960) A New Approach to Linear Filtering and Prediction Problems. *Transaction of the ASME—Journal of Basic Engineering*, pp. 35-45.

Rawlings, J. B., E. S. Meadows, and K. R. Muske, (1994). Nonlinear Model Predictive Control: A Tutorial and Survey ,*Proceedings of ADCHEM '94*, 203-214, Kyoto, Japan

Richalet, J. A., A Rault, J. D. Testud, and J. Papon, (1978). Model Predictive Heuristic Control: Application to Industrial Processes, *Automatica*, 13, 413