Some Observations on Development and Testing of a Simple Autotuning Algorithm for PID Controllers

BOHUMIL ŠULC, STANISLAV VRÁNA,
Department of Instrumentation and Control Engineering
Czech Technical University in Prague, Faculty of Mechanical Engineering
Technická 4, Prague 6 - Dejvice
CZECH REPUBLIC
bohumil.sulc@fs.cvut.cz
stanislav.vrana@fs.cvut.cz

Abstract: - Although a wide range of control algorithms are available, PID controllers are still the most widely used type. Even control engineers who have a good knowledge of advanced control algorithms prefer PID controllers when they are asked to design the control circuit for a new device. This is because of their price and because of the simplicity and reliability of the control algorithm. The PID control algorithm enables anyone with a basic knowledge of mathematics to gain some idea of what happens inside the controller during the control process, how the manipulated value is computed and its dependence on the control error. The main disadvantage of standard PID type controllers is the absence of autotuning features. Implementation of such controllers requires time-consuming tuning, and very often the presence of expensive experts. Therefore, there is quite strong pressure to equip new PID controllers with simple autotuners. Recently, tuners based on relay experiments have become popular. However, the use of autotuners of this kind requires the control process to be stopped temporarily during the relay experiment. For these reasons, even controllers equipped with these autotuning facilities are used without any tuning, sometimes even with the factory presetting. This unsatisfactory situation has motivated us to develop a new frequency response based autotuning algorithm that does not use any model of the controlled plant and enables the controller parameters to be changed without the necessity to break the control process.

Key-Words: - PID controller, laboratory model, tuning, model-free, implementation, optimality, mathematical model, autotuning,

1 Introduction

In industry, the PID controller is still the most widely-used controller. Its function is easy to explain and, in most cases, it is the easiest controller to adjust. Although in theory controller setting is considered very important, PID controllers provide a high level of robustness, so they do not need to be tuned much. They are sometimes even implemented with the factory presetting. Deliverers of automation, so-called system integrators, often do not tune controllers, because it is expensive [8] and demanding on human resources. Nevertheless, tuning can significantly improve control performance. PID controllers offer popular setting controllers that can support tuning on-the-spot.

Two types of tuning methods are designated for tuning a PID controller: model based methods, and model-free methods.

Model-based methods use some kind of mathematical description of the behaviour of the object that is to be controlled. The model can be in the form of transfer functions, differential equations, or a state space model. This dynamic description is usually linear, because the controller design and setting methods are not prepared for a nonlinear description.

A specific group of model-based methods are those based on regions of robustness [4], [5], [14].

Model-free methods use only some of the characteristic properties of the controlled plant or control loop. The properties are usually determined via experiments, and the controller setting procedure does not require any mathematical description. Model-free controller tuning methods are popular in practical applications. It is easier to explain how they work, of course. There are many model-free tuning methods, varying from very simple to very complicated [1], [10]. Despite the number of model-free tuning methods, the most-used method is the more than 50-year-old Ziegler-Nichols method. This exists in two variants, the critical frequency variant and the step response variant [1], [12], or the relay method, a recent modification of one of them. Examples of other model-free methods are the Chien-Hrones-Reswick method, the Cohen-Coon method (both are modifications of the Ziegler-Nichols methods), the Amigo method [1], and the momentum method [11].

These methods need to break the control process temporarily when a new controller setting is being computed.

There is also a group of PID controller tuning methods based on artificial intelligence tools, such as fuzzy logic [7], [15], neural networks and genetic algorithms [9].

The popularity of Ziegler-Nichols methods is also due to the instruction given at many universities. In most cases, only this method is explained to control engineering students as a representative of model-free methods. One of the reasons why the principle of model-free methods is so rarely explained at universities may be that they appear less scientific than model-based controller tuning methods, because they are mostly not based on any mathematical theory. Another reason may be that even control theory teachers do not know how to implement model-free methods into teaching and simulations tools, because such tools are well prepared for linear modeling and time course analysis, but not so well for frequency response analysis in cases where the Fourier transform is a weak tool.

There are some model-free methods that allow the controller parameters to be changed without needing to break the control process, e.g. Foxboro Exact [2] and Honeywell Looptune, but their functions have never been published in full. Only some details of how they work have been found by reverse engineering. They are strictly connected to the controllers devised by their inventors. An unanswered question concerns the optimality of these methods. It is usually assumed that the autotuning algorithm provided with the controller can find the optimal controller setting. However, these industrial controller producers do not allow the customer to influence the optimality criterion. For example, Foxboro Exact [18] is a tuning method based on pattern recognition, and when any predefined pattern occurs the corresponding rule is chosen for changing the controller parameters. The rules are unchangeable.

This raises the question: who decides what behaviour is optimal - the user or the producer of the controller? If we choose tuning algorithms of this kind, how can we be sure that the controller setting is optimal, if the evaluating rules have to be universal for all controlled plants? Evidently, the responses of the controlled plant to disturbances should be better with the controller than without it. But is the behaviour really so significantly better than if we were to use a badly-tuned controller without autotuning features or equipped by a simpler autotuning method?

Universities prefer to teach advanced control strategies. Advanced strategies are mostly presented as superior to a PID control strategy [3], but they are not so widely (if at all) used as PID controllers in practical applications. They are based on deep mathematical and control theory, so a perfect mathematical description of these methods can be offered. This is a somewhat dangerous situation, because every potential user of such controllers also needs this knowledge of mathematics and control theory. However, not every user has graduated in this field. Unlike in the case of PID controllers or two- and three-state controllers, there is usually no explanation of what is happening inside the controller during the control process. Even students taking the relevant control engineering course are shown only the equations, and they can compare the courses of selected values of variables in a closed control loop. Except in the case of state control, there are even no tools that can show students what is happening inside the controller and so help them to really understand its function.

Industrial practice prefers controllers that do not use a mathematical model of the controller plant, because such controllers are usually cheaper and they do not need highly educated engineers to set them. The above-described shortcoming in the teaching of advanced control strategies, along with fact that a PID controller is adequate for many applications, increases the popularity of PID controllers in industrial practice.

Thus there is a conflict of interest between the universities, which prefer to teach model-based methods, and industrial practice, which prefers to use model-free tuning methods.

A PID controller equipped with our new frequency-based model-free autotuning algorithm can be considered as an advanced control strategy. However, it is simple to use and it is simple to explain how it works, which makes it acceptable for industrial practice.

2 Principle of the proposed frequency response based autotuning function

In linear control theory, indicators using the Nyquist plot have been introduced as optimum control quality evaluation methods. The best known representatives are the Gain Margin and the Phase Margin. According to our experience, the frequency response method can also be used for defining some other control quality indicators on the same basis. This frequency response need not be computed only from the open loop transfer function, as in the case of linear theory (knowledge of linear models is essential), it can also be evaluated from the real responses excited by a harmonic signal which is inputted into a real (nonlinear) control circuit. Optimal values of the indicators recommended by linear theory can be used as information for the desired optimal behaviour of the control circuit, even in cases where the circuit does not show linear properties. We can also use

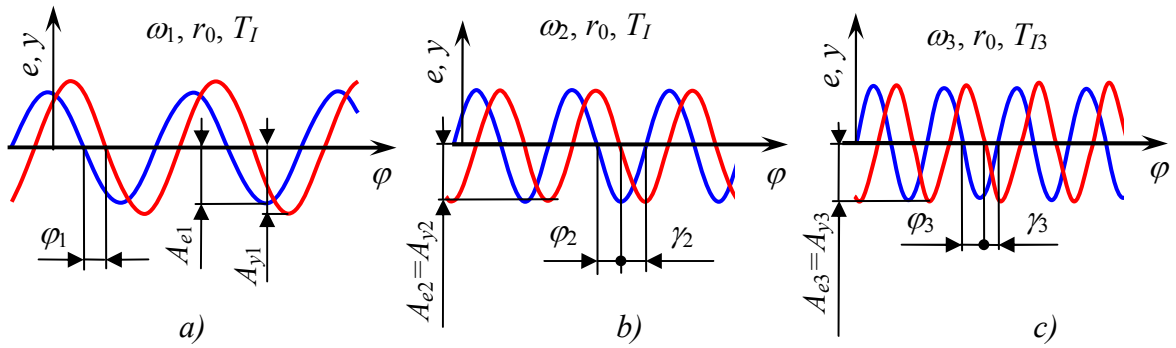

Fig.1 Three main steps in the process of controller tuning, based on a Phase Margin control quality indicator (depicted tuning of parameter T_I only) – the amplitude of the external excitation signal is denoted by symbol A having "e" in the index; while in the symbol denoting the amplitude of the response, we use "y" in the indices
a) starting situation: frequency ω_1 of the exciting signal is not of such a value when magnitude A_{e2}/A_{y2} is equal to one, and thus the phase delay φ_1 angle has nothing in common with the phase margin indicator of the control quality
b) by gradual changes of frequency ω_1, a state in the steady frequency responses has already been achieved when φ_2 represents a topical value of the phase margin $\pi - \varphi_2$, because $A_{e2}=A_{y2}$, but angle φ_2 is still not equal to the recommended level, e.g. 50°
c) the previous step is repeated with a corrected value of the time integral constant T_I in each step until the recommended optimal value of the phase margin is achieved

the actual indicator values as information on the extent to which controller retuning is necessary.

The autotuning algorithm uses the control quality indicators in the following way (the phase margin is chosen just as an example):

First, at a chosen frequency ω_1 the initial state is evaluated, i.e. the actual phase φ_1, period T_1 and the amplitudes of input and output signals A_{e1} and A_{y1} (see Fig. 1a) are determined.

Then, we manipulate the size of the exciting signal frequency ω (without changing the controller parameters) as long as a value ω_2 is reached when the amplitudes of the input and output signals are equal ($A_{e2} = A_{y2}$). Equality of the input and output signal amplitudes is the condition for determining the Phase Margin γ_2 from the phase angle (see Fig. 1b).

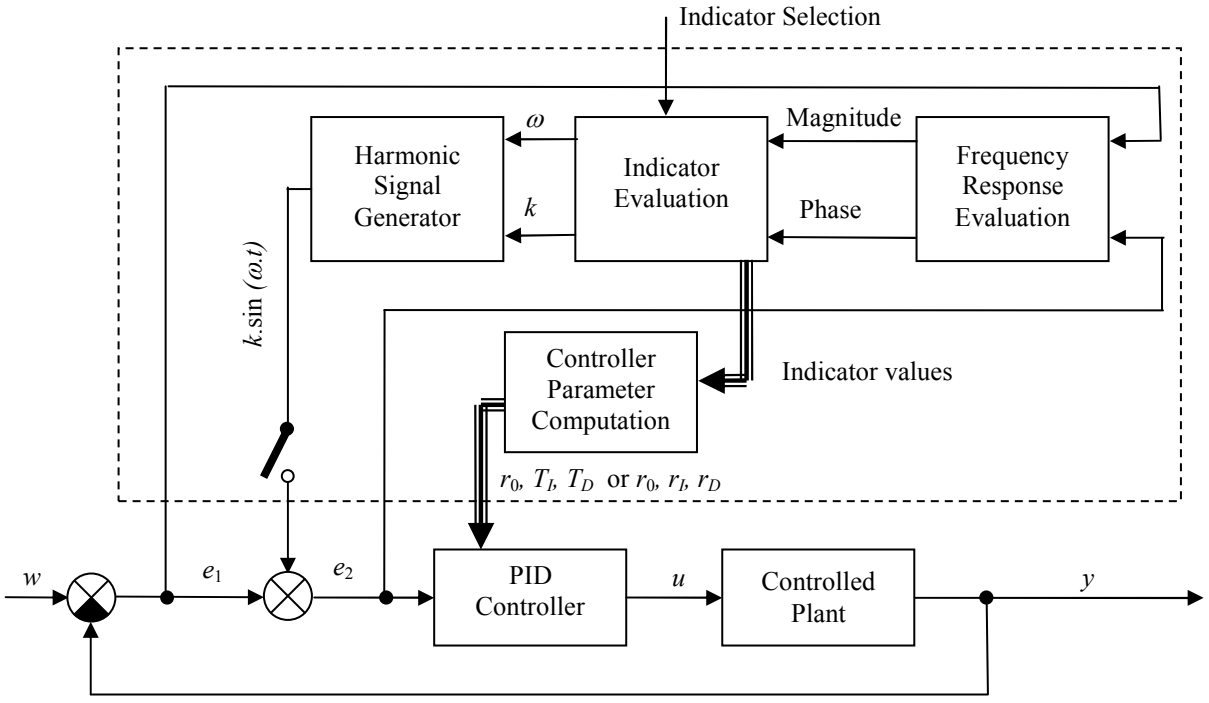

Fig.2 Block scheme of autotuning based on frequency indicators of control quality autotuning scheme

If the phase margin differs from the desired value, one of the controller parameters is changed (T_I to T_{I3}, in this example), and then the exciting frequency ω is changed again until a value ω_3 and the controller parameter are found when the Phase Margin has the desired value (see Fig. 1c).

A block scheme of frequency based autotuning is shown in Fig. 2. The block Signal Analysis analyzes the control error before and after adding a harmonic excitation signal to the control error (e_1 and e_2). It computes the Magnitude and the Phase values. Signals e_1 and e_2 are equivalent to the signals e and y in Fig. 1. The current Phase and Magnitude values are used to compute the values of the selected indicator in the block Indicator Evaluation. Block Indicator Evaluation has two outputs. The first of these is the chosen indicator value, and the second output is the properties of the added harmonic signal. The Controller Parameter Computation block contains a set of rules for computing a new controller setting aimed at achieving the desired control quality indicators values. The block Harmonic Signal Generator is a source of exciting harmonic oscillations input in the control circuit.

An example of how the algorithm works is shown in Fig. 3. For illustrative purposes, only one controller parameter is tuned The time courses of amplitude magnitude A, phase φ, the frequency of the exciting harmonic oscillation ω and the tuned controller parameter r_I are shown in the figure. At the beginning of the course in the example, the initial values are set to random values, and at the end of the course the Phase Margin = $-\pi/3$ is reached. More information about the algorithm is given in [17].

3 Approaches used in solving implementation problems

The principle of the autotuning algorithm presented here is easy, but its implementation is somewhat more complicated. It is necessary to determine correctly one or more points of the open loop frequency response. This cannot be computed from the transfer function, as it is in linear models. The difficulty in evaluation the frequency response increases with the complexity of the tool used for the testing function. It is most complicated to obtain the information in real condition, which we tested using physical models in the laboratory.

Generally, we used three types of models while implementing and testing the autotuning algorithm. The first group of controlled process models consisted of linear models. These very idealized models were usually

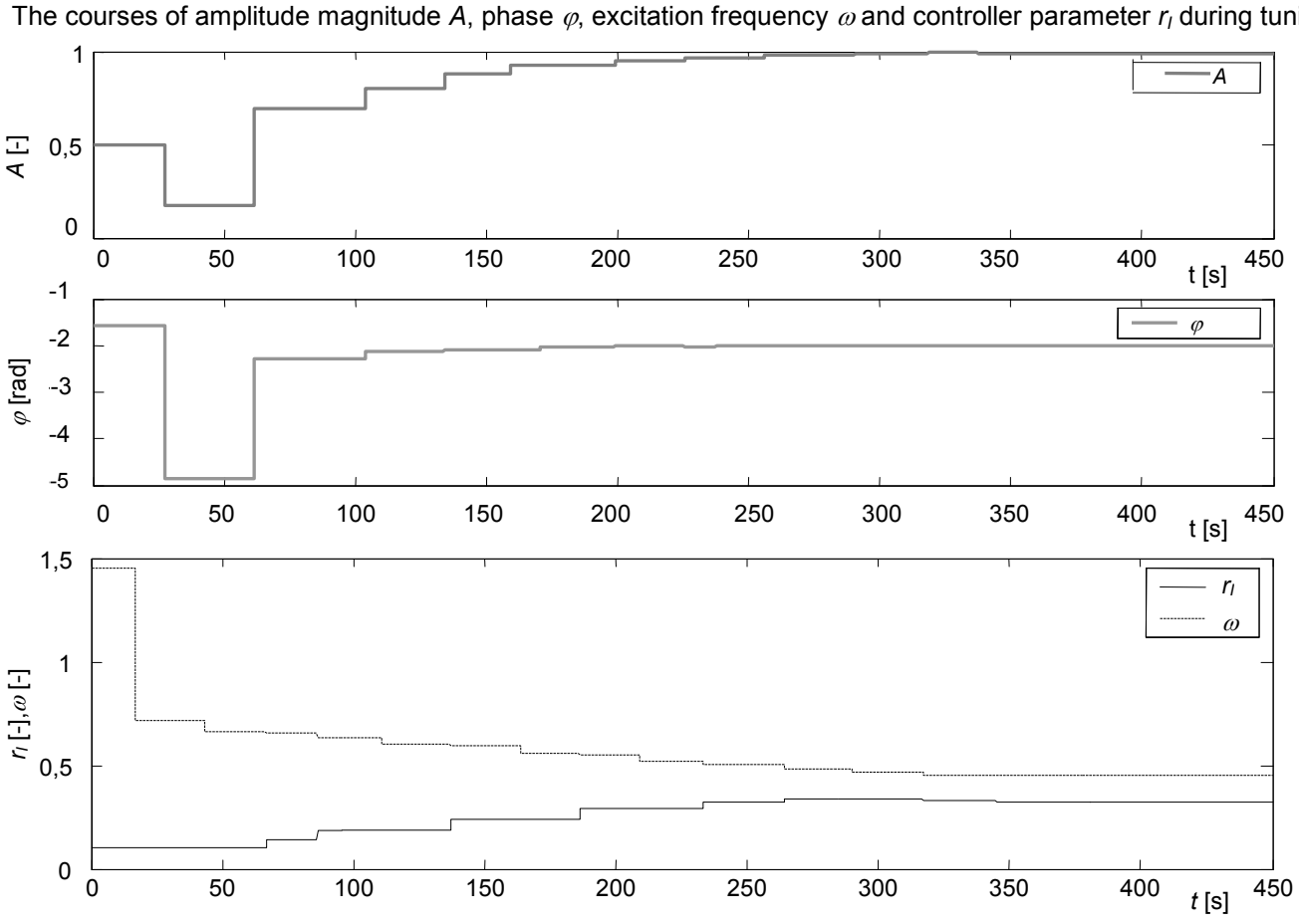

Fig.3 Example of tuning one controller parameter (parameter $r_I = r_0/T_I$ is tuned)

described by transfer functions. Their usefulness lies in their simplicity and the possibility to compute the expected results in advance and compare them with the results from simulation experiments. For example, the courses shown in Figure 3 were obtained with the use of a linear model with the transfer function

$$G_S = \frac{1}{s^2 + 1{,}2s + 1} e^{-1{,}5s} \tag{1}$$

The second type of models used here are models supplemented by some precautions providing a better simulation of physical reality than is possible using pure linear models. Such more complicated models may contain non-linearities, and their complexity may vary from models based on linear models with added non-linearity, e.g., a saturation of capacity, tank overflow, stopping the temporary increase in temperature while changing the phase from liquid state to gaseous state, etc., to pure non-linear models, which represents phenomena observable on the physical device that are significant for control. The advantage of models of this type is their easy reconfigurabilty. Some problems can occur while performing simulations with such complex models. These problems are not caused by a wrong model design, but are connected with the simulation solver. When there is a possibility, that they may occur, preventive measures must be taken.

The third and most challenging type of models used here were physical models, or replica simulation models of them.

3.1 Model of a three-tank cascade laboratory set-up

Before implementing the autotuning control algorithm on a three-tank cascade set-up, we worked with a model of it in various levels of complexity, as mentioned above. One of the advantages of such modelling is that tuning results can be obtained in less time than when a physical model is directly used. The mathematical description can be nonlinear, and is based on a volumic flow rate balance,

$$\frac{d}{dt} h_1(t) = \frac{1}{A} q_1(t) - \frac{Kv_{12}\sqrt{h_1(t) - h_2(t)}}{A} \tag{2}$$

$$\frac{d}{dt} h_2(t) = \frac{Kv_{12}\sqrt{h_1(t) - h_2(t)}}{A} + \frac{Kv_{12}\sqrt{h_3(t) - h_2(t)}}{A} - \frac{Kv_{20}\sqrt{h_2(t)}}{A} \tag{3}$$

$$\frac{d}{dt} h_3(t) = \frac{1}{A} q_3(t) - \frac{Kv_{32}\sqrt{h_3(t) - h_2(t)}}{A} \tag{4}$$

where A means the cross section (the same for all three tanks) in dm^2

Kv_{xy} means the flow rate coefficients used in the equations quantifying the (volumic) flow rate through the valve from tank x to tank y

h_x means the water level in tank x

q_x means the supply flow rate, or the mutual flow rates between the tanks (x, y = 1, 2, 3) or to the atmosphere (y = 0)

The mathematical model gives a credible representation of the three-tank cascade. The only behaviour not described by the equation involves situations when the tanks overflow, so tanks of unlimited height are considered in the simulation. However, overflow is not a typical or important state in our testing.

A Simulink representation of the mathematical model enables the initial conditions to be computed before the simulation starts, so it is not then necessary to compute the initial conditions separately outside the Simulink scheme.

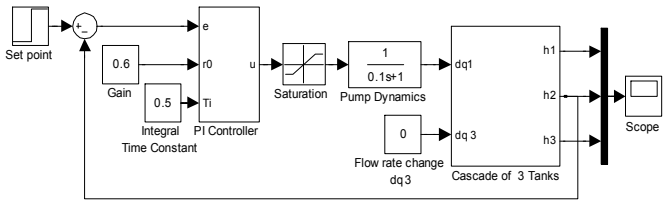

Fig.4 Simulink block scheme of the case the level in tank 2 of the cascade is controlled

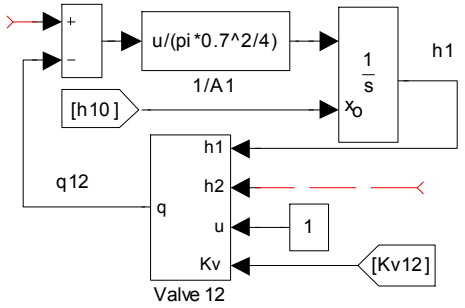

Fig.5 Water tank Simulink block scheme

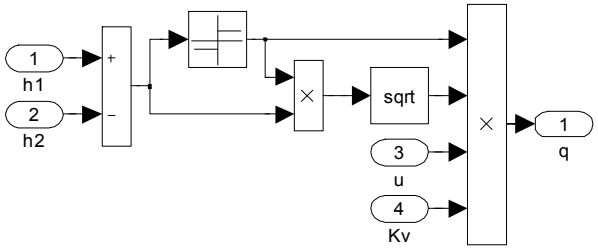

Fig.6 Simulink model for flow rate computation

Figure 4 shows the Simulink scheme of three-tank cascade laboratory model when the second tank level is controlled. Figures 5 and 6 show the main parts of the

three-tank cascade Simulink scheme, where Fig. 6 is the Simulink scheme of the equation, $|h_1 - h_2|$ is computed as $(\mathrm{sgn}(h_1 - h_2))(h_1 - h_2)$.

3.2 Model of the control valve

Another example of the non-linear models used here is a model of the control valve used for position control by means of a linear double acting pneumatic drive.

The mass flow rate through a control valve can be described by the equations

$$M(t) = G_n \varepsilon(t) \sqrt{\frac{T_0}{T_{in}}} \frac{P_{in}(t)}{P_0} \sqrt{\frac{P_{out}(t)}{P_{in}(t)} \left(1 - \frac{P_{out}(t)}{P_{in}(t)}\right)} \tag{5}$$

when $\dfrac{P_{out}(t)}{P_{in}(t)} > 0{,}5$ (subcritical case)

$$M(t) = G_n \varepsilon(t) \sqrt{\frac{T_0}{T_{in}}} \frac{P_{in}(t)}{2P_0} \tag{6}$$

when $\dfrac{P_{out}(t)}{P_{in}(t)} < 0{,}5$ (supercritical case)

where M means the mass flow rate through the valve,
G_n means the nominal flow capacity of the valve
ε means relative valve opening
T_0 means normal temperature
T_{in} means temperature of the air flowing into the valve,
P_0 means normal pressure,
P_{in} means pressure of the air flowing into the valve,
P_{out} means pressure of the air flowing out of the valve.

$$Q(t) = \frac{M(t)}{\rho} \tag{7}$$

where Q means volumic flow
M means mass flow
ρ means air density

$$\rho = \rho_0 \frac{T_0}{T} \frac{P}{P_0} \tag{8}$$

where ρ means air density
ρ_0 means normal air density
T means air temperature
T_0 means normal temperature
P means air pressure
P_0 means normal air pressure

Equations (5) and (6) can be converted into the Simulink scheme, as shown in Figure 7. Because we are considering a double acting pneumatic drive, it is necessary to use two of these schemes to model the complete control valve.

Let us assume that the pneumatic drive is connected to pressure values $P_{out1}(t)$ and $P_{in2}(t)$. Then we need to compute these values during simulation to be able to compute the volumic flows $Q_1(t)$ and $Q_2(t)$ to and from the pneumatic drive.

One way to compute values $P_{out1}(t)$ and $P_{in2}(t)$ is to derive the functions $P_{out1}(t, P_{in1}, P_{out2}, \varepsilon(t))$ and $P_{in2}(t, P_{in1}, P_{out2}, \varepsilon(t))$. This is possible; we have two conditions for joining these values

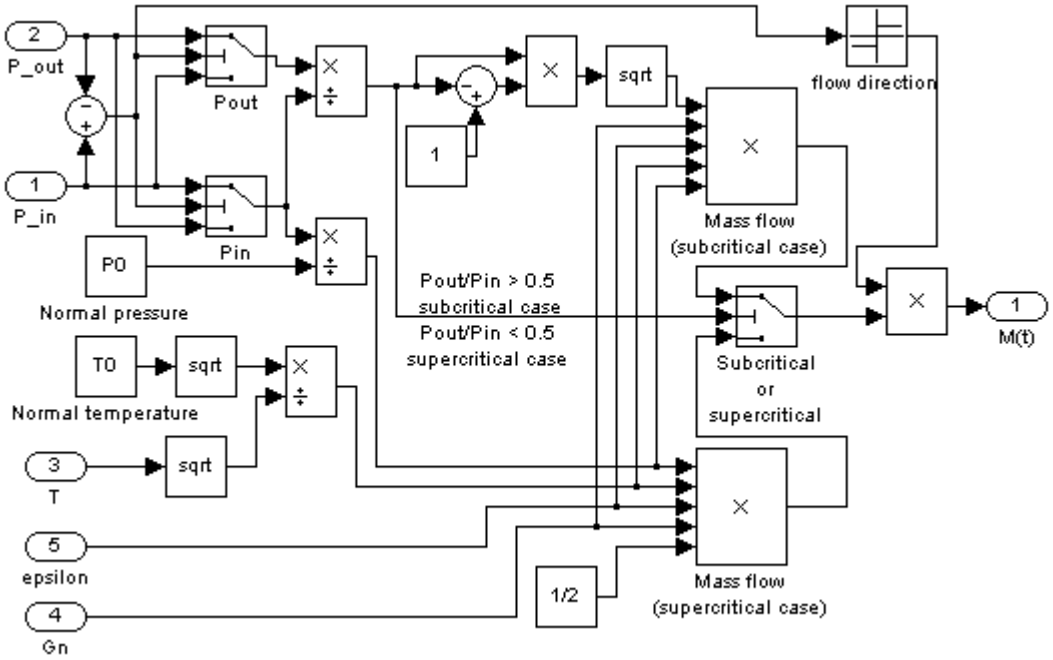

Fig.7 Simulink scheme of computation of mass flow of air flowing through control valve

$$P_{in2} = P_{out1} - \frac{m}{A} \tag{9}$$

where P_{in2} means the pressure of the air flowing out of the drive,
P_{out1} means the pressure of the air flowing into the drive,
m means the weight of the mass,
A means the area of the piston.

$$Q_1(t) = Q_2(t) \tag{10}$$

However, this solution is very complicated. The derivation of the functions $P_{out1}\left(t, P_{in1}, P_{out2}, \varepsilon(t)\right)$ and $P_{in2}\left(t, P_{in1}, P_{out2}, \varepsilon(t)\right)$ is very complicated, and we also get seven possible pairs of values P_{out1} and P_{in2}, one for the case when the flow is supercritical in both parts of control valve, two for the case when the flow is supercritical in the first part of control valve and in its second part the flow is subcritical, two for the case when the flow is subcritical in the first part of control valve and in its second part the flow is supercritical, and two for the case when the flow is subcritical in both parts of control valve.

The second solution is to run an iterative process that computes the values P_{out1} and P_{in2} from equations (5), (6), (7), and (8). The iterative process is based on the increase of values P_{out1} and P_{in2}, while it is tested if $Q_1(t)$ is equal to $Q_2(t)$. When they are equal, the searched values P_{out1} and P_{in2} are found.

The iterative process can be constructed as a Matlab function in the Simulink scheme. To avoid additional work with the complete iterative process in the Matlab function, we can use a second Simulink scheme of the control valve that is executed from the Matlab function. This second (slave) model is the same as the first (main) model, with the same parameters, with the exception that values P_{out1} and P_{in2} are connected to the Ramp block, which provides a continually increasing value. In this second model, the values $Q_1(t)$ and $Q_2(t)$ are compared, and when they reach the same value (when they are equal), the simulation can be stopped, because the searched values of P_{out1} and P_{in2} have been found and can be propagated to the main model. It is necessary to execute the slave model in each iteration step of execution of the main model.

A slave model executed from main model can be used not only to compute values, for computation of which it is necessary to use an iterative process. The simple slave model can also be used to show the internal values of any Matlab function

3.3 Numeric problems simulating

Models of valves, and more generally of resistances, very often form a part of simulation models that are

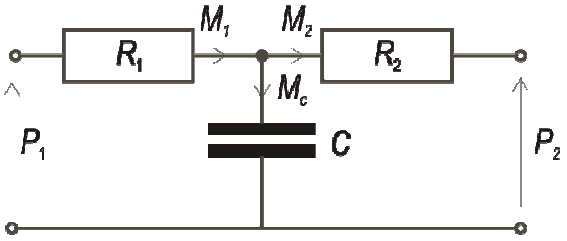

Fig.8 RC circuit scheme

created by interconnecting blocks modelling process phenomena in a simplified way. Each such block represents a certain real element, where we imagine that a certain specific physical phenomenon, lumped and separated from the others, takes place. For example, creating pressure-flowrate models we get chains of resistances (valves) and capacitors (volumes), i.e. we can create a pneumatic circuit model as an analogy of RC electric circuit models. Let us consider such a very simple RC circuit, the scheme of which is depicted in Figure 8.

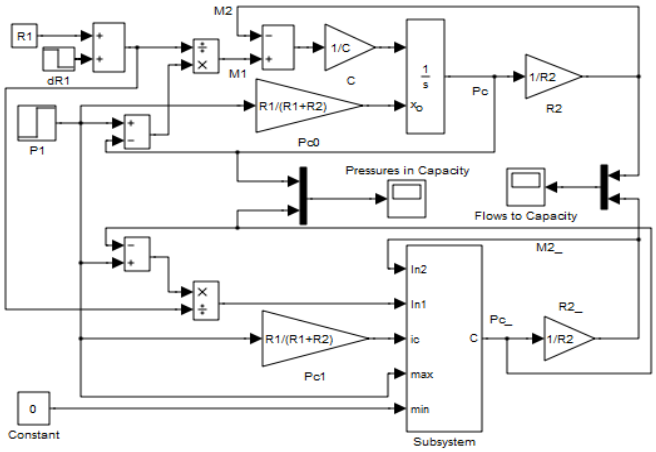

Fig.9 Standard and limited accumulation models of an RC circuit

A problem that we would like to mention in connection with simulating physical models is the numeric instability that sometimes occurs when resistances and capacities have specific values which cannot be mastered by step by step simulation with the setting of simulation conditions used here (step and solver) and in a standard simulation scheme (see the upper part of the Simulink program in Fig, 9). In such cases, something happens in each step of this numerically performed simulation that does not correspond to physical reality. Let us consider two situations in the experiments that are later demonstrated by means of results from simulation. In the first experiment, let the input pressure (voltage, temperature, etc. in analogous models of the same RC type) change from 12 to 15 kPa. As a step response to this increase after the first step, the size of which was set too large (Δt

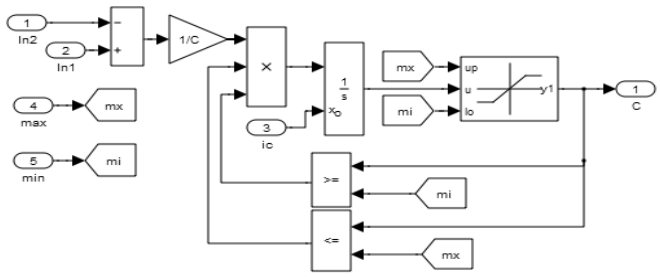

Fig.10 Subsystem performing limited accumulation

= 0,15 s for the integration method used here (Euler)), it is computed that the mass increase in the capacity is $\Delta m_1 = M_1(t)\Delta t$ and the corresponding pressure increase in the capacity is

$$\Delta P_C = \frac{RT}{V}\Delta m_1 \qquad (11)$$

There is nothing to prevent pressure P_C at the time instant $t + \Delta t$ being greater than the input pressure. This cannot happen in continuous time, because each continuous increase in the pressure P_C must lead to an immediate decrease in the flow to the capacitor. Thus there is no tendency to reverse the direction that we can observe in the discrete simulation if the pressure in the capacitor becomes greater than the input pressure. Because this change in the flow orientation is impossible for physical or technical reasons, it is necessary to add to the simulation model some precautions against it. One of reasonable solutions is to stop the integral action as soon as the state variable measuring accumulation in the capacity exceeds defined limits. In our example, the input pressure P_1 is the upper limit, and atmosphere (zero overpressure) is the lower limit. The stopping mechanism performed by the Subsystem in Fig, 10 can be clearly seen under the mask in Fig. 9. It is very similar to an antiwind-up solution.

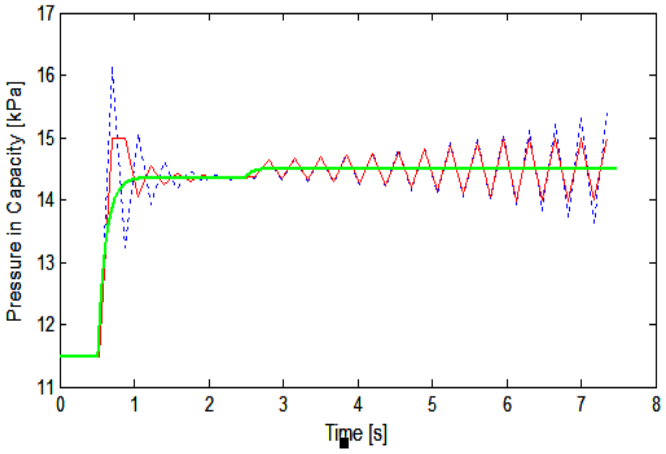

Fig.11 Step responses of an RC circuit under various simulations

The results from simulations carried out under various conditions are shown and compared in Fig.11.

The smooth course in Fig.11 is the "right" reaction to step changes in the input pressure P_1 and the value of resistor R_1, when the step size and/or the integration method have been set in such a way that the obtained course is very close to the continuous reality. The full line with some saw teeth parts depicts the response that was obtained from the model using the block Subsystem for limiting accumulation in the capacitor. The dotted line marks the response from a standard model; both responses have been obtained under the same setting of the simulation parameters.

The use of blocks that model saturation of the capacitor is important in complex models containing more than one mutually interconnected RC submodel. They help to keep numeric stability, and the results are much closer to reality. No other model precautions are necessary to ensure inalterability of the flow orientation.

4 Laboratory model

As a laboratory model we utilize a three-tank cascade set-up. The set-up consists of three interconnected tanks (see Fig. 12). Water is supplied into Tank One and Tank Three. Each tank is equipped with a pressure sensor that is used for measuring the water level. In addition to their mutual interconnection, each of the tanks in the cascade has its own outlet valve, which enables various operational modes with different dynamics to be simulated. All valves in the laboratory model can be adjusted only manually.

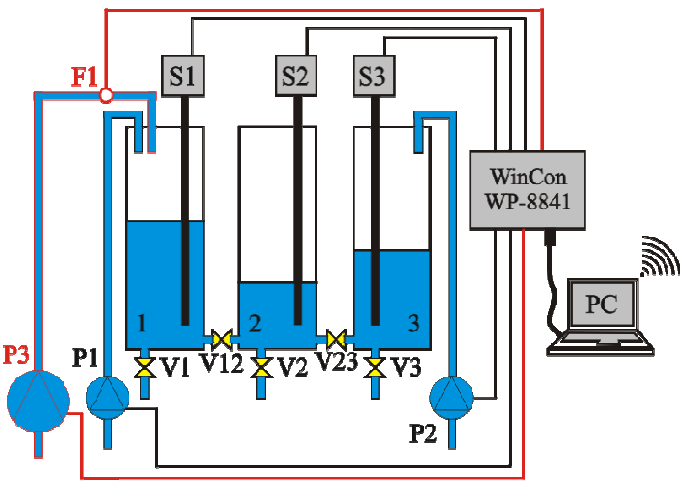

Fig.12 Three-tank cascade laboratory model scheme

Pumps P1 and P3 feed Tank One, while pump P2 feeds Tank Three. Tank Two is unfed. The hydrostatic pressure is measured by pressure sensors S1, S2, and S3, and the value is converted to the water level value. It is

also possible to measure the flow rate to Tank One through pump P3 by flow meter F1. The laboratory model is of the third order, so that an unstable control process can be achieved if parameters from the area of unstable value combinations are used for setting the controller. Originally the laboratory model contained only pressure sensors S1, S2, and S3 and pumps P1 and P2, and the circuit containing pump P3 and flow meter F1 was added later. The driving electronics of this circuit is based on a recommendation given in [6].

4.1 WinCon programmable automation controller

The fundamental hardware part is the control device. The PAC WinCon (PAC means Programmable Automation Controller) was chosen as the control device. This PAC is based on the ARM processor, and an embedded operating system is used. WinCon uses Windows CE as the operating system. We chose this type of control device because Rex control software can be used as control software, and also because of the RexLib library, which adds some block into the Simulink block library and allows direct communication between WinCon and Simulink.

The communication between Simulink and WinCon provides an opportunity to split the whole controller with the autotuning algorithm into two parts. The own control algorithm of PID type and the harmonic signal generator are placed in WinCon, and the autotuning mechanism, which is an extension of the PID algorithm, is placed in Simulink. The control algorithm placed in WinCon works independently of the connection to the autotuning algorithm in Simulink.

The selected type WinCon W-8741-G contains seven expansion module slots, two LAN connectors, two USB connectors, one RS-232 connector, one RS-485 connector, one VGA connector and one Compact Flash card slot. We use the expansion modules I-8017H (14-bit 100K sampling rate 8-channel analog input module), I-8024 (4-channel 14-bit analog output module), I-8053 (16-channel isolated digital input module - differential input), I-8064 (8-channel power relay module), I-87018 (8-channel thermocouple input module), and I-8080 module (4/8 channel counter/frequency module). The I-8017H module is used to get data from pressure sensors S1, S2, and S3 and also from flow meter F1 (the flow meter has a frequency output, and a frequency to voltage converter is used). Modules I-8024 and I-8064 drive pumps P1, P2, and P3. So both types are possible: continuous control, and on/off control of the pumps. The other modules are not used to control the laboratory model of the three-tank cascade. We use this PAC not only to control this laboratory model but also for experiments on a biomass-fired boiler [13], where the other modules, especially module I-87018, are useful.

This PAC also exists in other types, which differ in the built-in devices that they offer. For example, W-8331-G contains three expansion module slots, one LAN connector, one USB connector, one RS-232 connector, one RS-485 connector, two PS/2 connectors (keyboard and mouse), one VGA connector and one Compact Flash card slot, while W-8701-R2 contains seven expansions module slots, one LAN connector, three RS-232 connectors, one RS-485 connector and one Compact Flash card slot.

4.2 Rex control software

The tool for WinCon algorithm development and for monitoring it is a couple of programs, RexDraw and RexView. Rex Draw is designed for developing, organizing and compling algorithms, and RexView is designed for compiled configuration upload and for station activity monitoring. It can provide information about the activity of its own station and about the activity of individual algorithms. The algorithm parameters can be changed.

The supporting part of the software is the function block set library RexLib, which enables use of the Simulink extension of the Matlab program for algorithm development. The advantage of algorithm development in Simulink is that it enables the algorithm to be simulated. Simulation is not possible with the use of RexDraw or RexView. On the other hand, compilation is not possible in Simulink, so RexDraw has to be used.

The RexLib library contains all blocks which are necessary for developing the control algorithm and which are also contained in the RexDraw program. RexLib library is designated for use with Simulink. The algorithm configuration blocks are also included, so the whole algorithm can be simulated even in situations when the algorithm is divided into more than one individual block scheme. All particular algorithms can also be simulated separately.

When an algorithm designated for use outside Simulink is developed in Simulink, it is necessary to use only blocks included in the RexLib library. Other blocks can also be used, but only for simulation purposes. They cannot be a part of the algorithm that is developed. The extension of the Rex control system can be used, which allows to leave these blocks to be left in the scheme if they are named in such a way that their name contains the word simulation. Then it is not necessary to delete these blocks before final compilation of the algorithm.

5 Experiments comparing results, and the demands of controller tuning by various methods

Three examples are shown, in Fig. 14, Fig. 15 and Fig. 16, of a controller retuning process using three different tuning methods: the relay method [1], the momentum method [11], and the frequency-based tuning method [17]. The examples demonstrate the problems of model-free controller setting on-the-spot. They also show what kind of information is missing if the explanation focuses on the theoretical background of these methods, which is often the case in publications and in university education. All experiments are performed in the following simulation experiment sequence: first, the desired water level in Tank Two is increased by $\Delta h_2 = 0{,}1$ dm, then it is decreased to the initial water level. In the second period, the controller is tuned, and, finally, the same experiment as in step one is

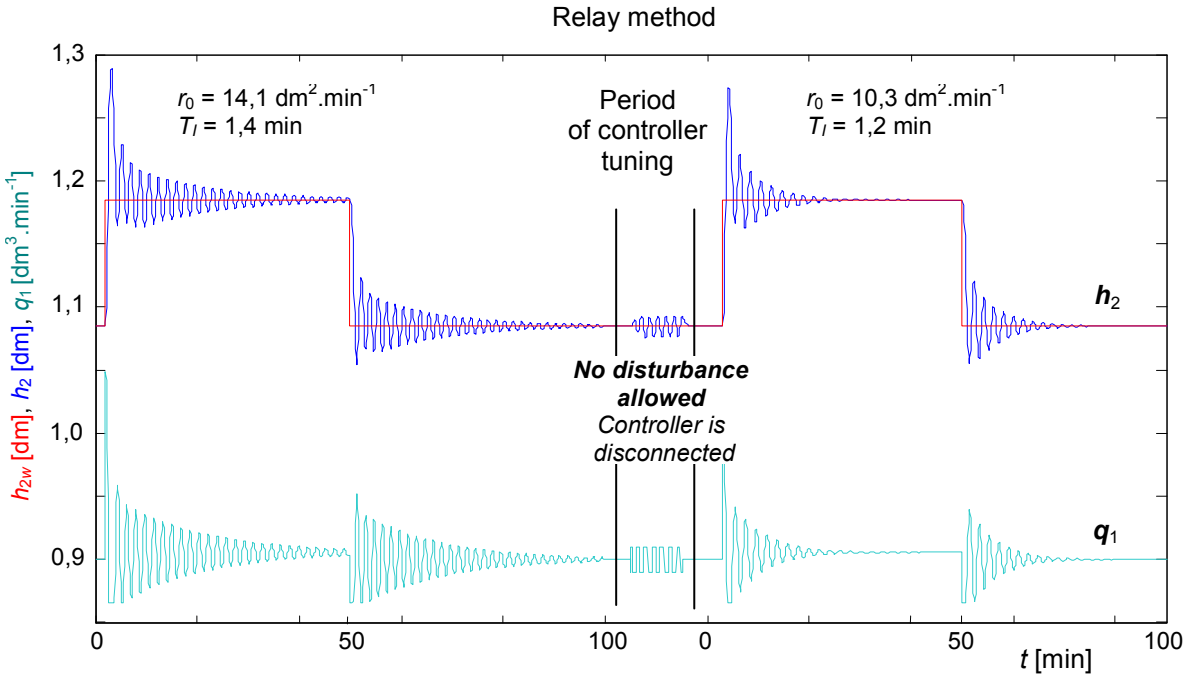

Fig.13 Relay method tuning experiment (manipulated variable q_1 depicted after dividing values by 10)

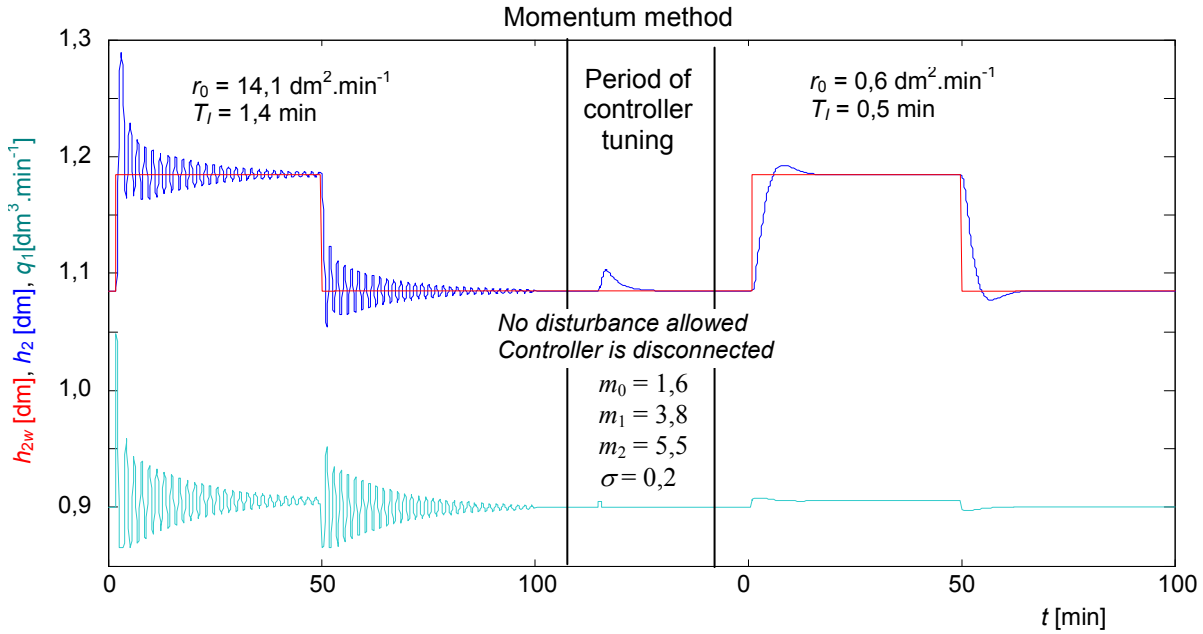

Fig.14 Momentum method tuning experiment (manipulated variable q_1 depicted after dividing values by 10)

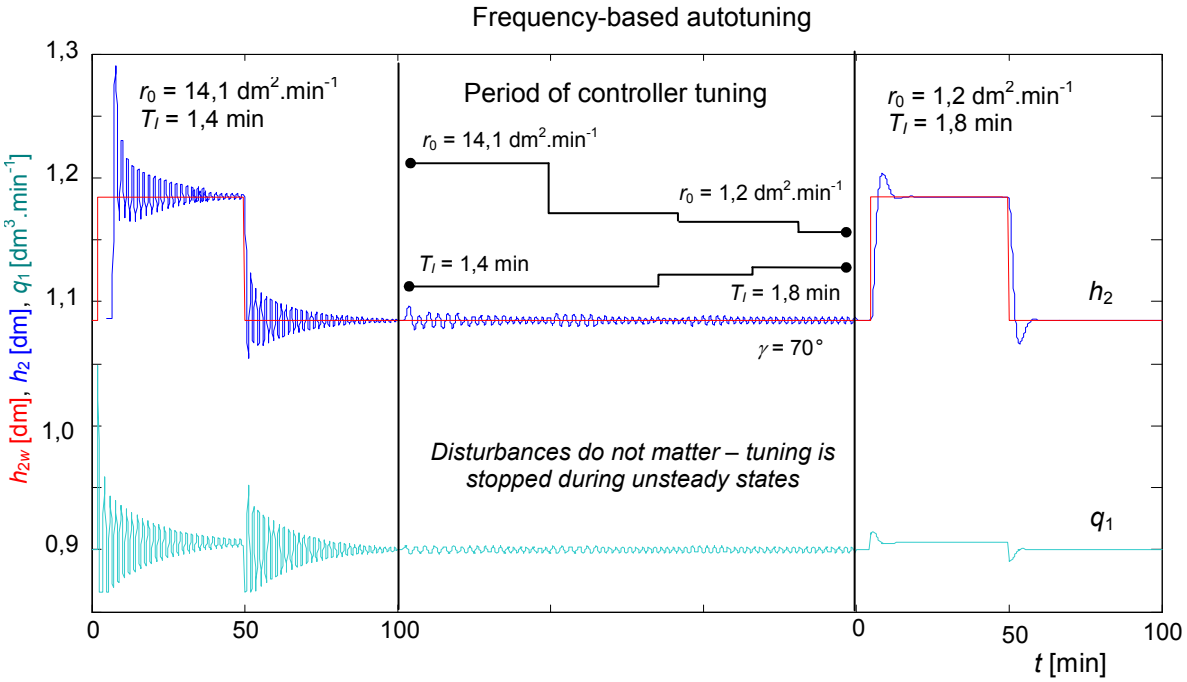

Fig.15 Frequency-based autotuning experiment (manipulated variable q_1 depicted after dividing values by 10)

repeated, but with a new controller setting. The first period demonstrates the control performance when the controller setting is unsuitable. The control process is stable, but it oscillates considerably. The second period demonstrates the retuning process, which differs for the three tuning methods used here. The third period demonstrates the control performance when the controller is retuned. The control performance is better than in the first period.

As is shown in the figures, the courses when the tuned controller is used differ significantly. The relay method produces the most oscillating response, while the response of other methods is not oscillating.

The second major difference between the tuning periods in the figures is their length. Therefore, for the controller obtained here, frequency-based autotuning can provide better results. However, when we take into account the length of the tuning period, the whole tuning process is not so much better. Nevertheless, there is another significant difference – the need for the controller to be disconnected.

When we use the relay method to tune the controller, it is necessary to disconnect it temporarily, then to perform the relay experiment and reconnect the controller into the loop. The same is necessary when we use the momentum method to tune the controller. When we use the frequency-based tuning method, however the control function is fully operable during the tuning period. Thus, the need to disconnect the controller or restrict its function is a further consideration when comparing tuning methods.

Tuning methods should be compared in several aspects, but the only considerations mentioned here are the quality of the controller setting and the information that is required about the controlled plant/process. All such considerations should be taken into account in comparing the tuning methods focused on the needs of practical application.

6 Conclusions

A frequency-based PID controller tuning method that uses control quality indicators has been presented not only in terms of its operating principles, but mainly in terms of the experience obtained during implementation and testing with various kinds of controlled plant models (linear, non-linear, and physical). We used these types of model to some problems and their solution if the simulation should run in a correct, reality reflecting way.

The PID controller tuning methods were tested on a three-tank cascade set-up and on a simulation model of this cascade with emphasis on modeling all details corresponding to real application. This circumstance is very important, because it is only when we reflect all real conditions under which autotuning starts and runs that a serious and useful comparison of various approaches and of autotuning efficiency can be made. This is what is often missing in the literature. From our tests it follows, for example, that the relay method requires the shortest tuning time but the setting that is obtained is the least suitable. The controller is disconnected during tuning. The momentum method leads to a controller setting that needs minimal control

actions, but the controller is also disconnected during tuning. Frequency-based autotuning needs more time for controller retuning than the other methods. Although the results of all tuning methods are parameters of optimal controller setting, the obtained settings differ and lead to responses of different quality.

The significant advantage of a controller equipped with the frequency-based autotuning method is that the controller is fully serviceable during tuning. In contrast to the momentum method and the relay method, when the operator cannot influence the controller setting obtained by these methods, the frequency–based method also allows the operator to change the criteria and their optimal values at any time if the response does not satisfy the expected course.

This work has been supported by Czech Science Foundation Grant No. GAČR 101/07/1667.

References:

[1] Åström, K. J., Hägglund, T., *Advanced PID Control*. USA: ISA, Research Triangle Park, NC, 2006.

[2] Åström, K. J., Hägglund, T., Hang, C. C., Ho, W. K., Automatic tuning and adaptation for PID controllers - a survey, *Control Engineering Practice*, Volume 1, Issue 4, August 1993, pp. 699-714

[3] Åström, K. J., Hägglund T., The future of PID control, *Control Engineering Practice*, Volume 9, Issue 11. 2001. pp. 1163 – 1175

[4] Emami, T., Watkins, J. M., A Unified Approach for Sensitivity Design of PID Controllers in the Frequency Domain, *WSEAS Transactions on Systems and Control*, Vol. 4, Issue 5, pp. 221-231, May 2009.

[5] Emami, T., Watkins, J. M., Robust Performance Characterization of PID Controllers in the Frequency Domain, *WSEAS Transactions on Systems and Control*, Vol. 4, Issue 5, pp. 211-220, May 2009.

[6] Hlava, J., Šulc, B., Advanced Modelling and Control using a Laboratory Plant with Hybrid Processes, *Proceedings of the 17th World Congress The International Federation of Automatic Control, Seoul, Korea, July 6-11, 2008*. Seoul: IFAC, 2008, p. 14636-14641

[7] Huang, Sh., Lo, Y., Metal Chamber Temperature Control by Using Fuzzy PID Gain Auto-tuning Strategy, *WSEAS Transactions on Systems and Control*, Vol. 1, Issue 4, pp. 1-10, Jan. 2009.

[8] Chang, H., Lin, Sh., Tang J., Su, Y., A Commercial Economy Efficiency PID Control System and the Implementation, *WSEAS Transactions on Business and Economics*, Vol. 2, Issue 4, pp. 186-191, Oct. 2005.

[9] Chang, L., Chen, H., Tuning of Fractional PID Controllers Using Adaptive Genetic Algorithm for Active Magnetic Bearing System, *WSEAS Transactions on Systems*, Vol. 1, Issue 8, pp. 158-167, Jan. 2009.

[10] O'Dwyer, A., *Handbook of PI and PID Controller Tuning Rules*. London: Imperial College Press, 2003

[11] Schlegel, M., *A new approach to the proposal for robust industrial controllers* (in Czech: Nový přístup k robustnímu návrhu průmyslových regulátorů). Inaugural dissertation. Pilsen: University of West Bohemia, 2000

[12] Schlegel, M., Exact Revision of the Ziegler-Nichols Frequency Response Method, *Proceedings of the IASTED International Conference on Control and Application*, *Cancun, Mexico, 2002*, Calgary: Acta Press, p. 121-126.

[13] Šulc, B., Vrána, S., Plaček, V., Hrdlička, J., Lepold, M., Control for Ecological Improvement of Small Biomass Boilers, *Proceedings of the IFAC Symposium on Power Plants and Power Systems Control, Tampere, Finland, July 5-8, 2009* Tampere: IFAC, 2009.

[14] Večerek, O., Schlegel, M., Design of H∞ PI Controller by Robustness Region Method, *Process Control 2004*, Pardubice: University of Pardubice, pp. R165-1-R165-9

[15] Voloşencu, C., Pseudo-Equivalence of Fuzzy PID Controllers, *WSEAS Transactions on Systems and Control*, Vol. 4, Issue 4, pp. 163-176, Apr. 2009.

[16] Vrána, S., Čihák, J., Šulc B., Three Tank Cascade Improvement for Model-free Autotuning Testing, *XXXIV. Seminar ASR '2009 "Instruments and Control"*. Ostrava: VŠB - Technical University of Ostrava, 2009, pp. 349-356

[17] Vrána, S., Šulc, B., Control Quality Indicators in PID Controller Autotuning, *The 5th International Conference on Computing, Communications and Control Technologies: CCCT 2007 Proceedings*. Volume II. Orlando: IIIS – International Institute of Informatics and Systemics, 2007, pp. 280-285

[18] Wade, H. L., Basic and Advanced Regulatory Control: System Design and Application. 2nd ed. USA: ISA, 2004.