A PID Tuning Tool for Networked Control Systems

LASSE ERIKSSON
Control Engineering Laboratory
Helsinki University of Technology
P.O.Box 5500, FI-02015 TKK
FINLAND

Abstract: - This paper presents a software tool that can be used for PID controller tuning in networked control systems. The problem with networks, when control issues are considered, is that they cause varying delays in the measurements. A delay in the feedback loop may cause instability and it thus complicates control design, and a varying delay makes the design task even more difficult. The PID tuning procedure that is implemented in the tool is based on simulation and constrained optimization techniques. First, the control system is modeled, and then its performance is optimized with respect to a cost criterion. The networks are modeled with delay distributions in the control system simulation model. The tool provides an easy-to-use graphical user interface for tuning PID controllers manually or automatically.

Keywords: - PID controller, Optimization, Simulation, Networked systems, Varying delay, Tuning

1 Introduction

The problem of tuning the PID (Proportional – Integral – Derivative) controller has been discussed in numerous books, papers and journal articles (see e.g. [9], [12]). The most famous tuning methods are the Ziegler-Nichols (ZN) open- and closed-loop methods published already in 1942. The ZN methods give good but not optimal tuning rules.

Many tuning methods have been implemented on computers and different tuning tools are also commercially available. Some industrial products with automatic tuning capabilities are presented in detail in [12], and a more up-to-date list is available in [14]. One of the PID tuning tools is the MATLAB based BESTune PID auto-tuning software [15]. It enables automatic and optimized PID tuning on the basis of process data. The BESTune software can be implemented into several industrial controllers, but it also provides a simple graphical user interface (GUI) for the tuning. The advantages of GUIs in general are obvious. The visualization and easy usability are important properties of programs that are based on rather complex tuning methods and procedures.

Every now and then the tuning methods and tools need to be revised since new technologies come into use in industry. First, there were the analog controllers, and then became the digital controllers. Currently, distributed control systems with field buses are popular. A new hot topic is wireless technology. The use of wireless sensors and even actuators affects the tuning of the controllers, because wireless networks have problematic properties. These include packet loss and varying delays in packet delivery. These issues arise especially in sensor and actuator networks, for which the controllers need to be tuned using new tools that support appropriate methods. There are not many methods for PID controller tuning in varying delay systems, and there are even less software tools for that task.

This paper presents a new tool for PID controller tuning in networked control systems where time-varying delays are present. The tool takes advantage of the tuning concept that Koivo and Reijonen have presented in [2]. Their concept uses the well-known simulation and optimization based PID tuning method, but they have formulated it for varying time-delay systems. In this study, the tuning concept is extended, first of all, to handle discrete-time PID controllers that are the most common controllers in industry. Secondly, a graphical user interface for the tuning procedure is implemented. When comparing with the other published PID tuning tools (such as BESTune), the one presented here is designed especially for networked control systems. The network characteristics are described with different delay distributions. Another new property in the tool is that it allows introducing optimization constraints related to the delay in order to improve the stability of the control system with respect to the variance of delay.

In the paper, the networked control system is discussed in section 2 and network delays in section 3. The tuning method and the tuning tool are presented in sections 4 and 5, respectively, and conclusions are offered in section 6.

2 Networked control

Networked control has been studied in several papers and articles (see e.g. [4], [6] and [10]). Long and varying delays occur especially in the Internet, but also in other networks. Consider a wireless sensor network where the routing of packets is determined online. The longer the route, the longer the time it takes to deliver a packet to its destination. A wireless sensor network could be used for measuring process variables as in Fig.1, where the basic components of a wireless networked control system are presented. The measurements are processed in the sensor network and fused in order to get reliable information of the state of the controlled process. The fused state estimate is sent to the controller which calculates the new control signal for the actuator (Act. in Fig.1).

2.1 PID controller

The PID controller is the most common controller in control systems. For example, in the mid 1990's the PID controller was used in over 95 % of the control loops in process control [12]. The good properties of the controller can only be achieved if the controller is well tuned. The tuning of PID controllers has been considered in numerous papers and books, but nearly always in systems with constant delays. Varying delays have not been addressed very often.

Generally, the continuous-time PID controller algorithm is given in time domain as

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\alpha)d\alpha + T_d\frac{de(t)}{dt} \right) \tag{1}$$

where u is the control signal and e is the error between reference y_r and process output y. The controller tuning parameters are K (gain), T_i (integration time) and T_d (derivative time). [13] The controller is often presented in the equivalent form

$$u(t) = K_p e(t) + K_i\int_0^t e(\alpha)d\alpha + K_d\frac{de(t)}{dt} \tag{2}$$

The tuning parameters are related to those in (1) by

$$K_p = K,\ \ K_i = \frac{K}{T_i},\ \ K_d = KT_d \tag{3}$$

The Laplace transform of the control signal is given in equations (4) - (7). The gain of the derivative term at high frequencies is limited using a proper approximation [13].

$$u(s) = P(s) + I(s) + D(s) \tag{4}$$

$$P(s) = Ke(s) = K_p e(s) \tag{5}$$

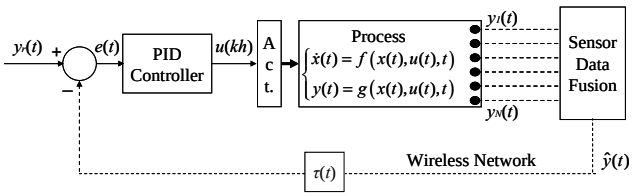

Fig.1. Wireless networked control system.

$$I(s) = \frac{K}{T_i s}e(s) = \frac{K_i}{s}e(s) \tag{6}$$

$$D(s) = KT_d s e(s) \approx K\frac{T_d s}{1 + T_d s/N}e(s) = \\ = \frac{K_d s}{1 + K_d s/\left(K_p N\right)}e(s) \tag{7}$$

Here N is a filtering time constant. At low frequencies the approximation is quite accurate, but at higher frequencies, where measurement noise occurs, the gain is limited. N is typically chosen from the range of 3 to 20. [13]

In practice, controllers are discrete-, not continuous-time. The continuous-time controller can be approximated with a discrete-time controller (sampling interval h). The proportional part of the controller is static and requires no approximation, only sampling. The backward differences method can be used in the approximation of the integral and derivative parts. The discrete-time PID controller algorithm is given in (8) - (11).

$$u(kh) = P(kh) + I(kh) + D(kh) \tag{8}$$

$$P(kh) = K_p e(kh) \tag{9}$$

$$I(kh) = I(kh - h) + K_i h e(kh) \tag{10}$$

$$D(kh) = \frac{K_d}{K_d + K_p N h}D(kh - h) + \\ + \frac{K_d K_p N}{K_d + K_p N h}\left[e(kh) - e(kh - h)\right] \tag{11}$$

2.2 Constraints in control systems

One way to improve the robustness of a controller is to tune it such that the gain and phase margins of the controlled system are sufficient. If either gain or phase margin of a system becomes negative, the stability of the system is lost. The presented controller tuning tool can take into account the desired gain and phase margins by formulating them into optimization constraints. The same principle can be used for other control system properties such as overshoot and settling time.

The gain margin is the number of decibels that the open-loop gain can be increased before it reaches 0 dB at the frequency where the open-loop phase shift is –180°. The phase margin is the additional phase lag that is allowable before reaching -180° at the frequency where the gain equals one (0 dB). [3]

3 Varying delays

The network delays depend e.g. on the system hardware and communication protocols. The delay distributions can be measured or modeled. Many delay distributions are presented in the literature, and the most relevant ones are briefly described here.

3.1 Gaussian distribution

The Gaussian distribution does not model any specific network, but it can be used as a generic delay model describing some variance in the delay around a mean value. It is possible to obtain negative delays when using Gaussian distribution, because the distribution is defined from minus to plus infinity. Since negative delays are unrealistic, the distribution should be cut at zero when modeling variable delays with it.

3.2 Gamma distribution

In a computer network such as the Internet, the delay distribution has been shown to resemble a shifted gamma distribution [8]. Even in the fastest network there is a nonzero minimum delay (therefore the shift), and the measured delay distributions have had a peak and a long tail, similar to the gamma distribution. The gamma probability distribution is given by

$$P(x) = \frac{\alpha}{\Gamma(n)}(\alpha x)^{n-1} e^{-\alpha x} \tag{12}$$

$$\Gamma(n) = \int_0^\infty t^{n-1} e^{-t}\, dt \tag{13}$$

Here n is a shape parameter (sometimes interpreted as the number of hops between first and last network nodes) and $\alpha = n / T$, where T is the mean delay.

3.3 Wireless network delay distribution

The delay distribution of a wireless network resembles two or more successive gamma distributions, where the distributions are shifted unequal times corresponding to retransmissions [6]. The first gamma distribution has a higher peak than the others, since most of the packets do not need retransmissions.

4 Optimal controller tuning

The presented tuning tool uses simulation and optimization techniques. The process for which the controller is tuned has to be modeled. Also the network or at least its delay characteristics need to be modeled, because they affect the performance of the control system. Both models may be based on measurements or they can be derived analytically. The performance criterion, which is often process specific, has to be given beforehand. There are several optimization criteria that are frequently used and modified for different optimization tasks. This section discusses the cost criteria and optimization methods that are used in the tool.

4.1 Cost functions

The ITAE (Integral of Time-weighted Absolute Error) cost function depends on the absolute value of the error between reference and process output signals. The error is weighted with the time of occurrence of the error. The absolute value ensures the monotonous growth of the cost function.

$$J_{ITAE} = \int_0^\infty t\,|e(t)|\, dt = \int_0^\infty t\,|y_r(t) - y(t)|\, dt \tag{14}$$

Other similar cost criteria can also be determined. For example, IAE (Integral of Absolute Error), ISE (Integral of Square Error) or ITSE (Integral of Time-weighted Square Error) can be used. They are given, respectively, in (15) - (17). [12]

$$J_{IAE} = \int_0^\infty |e(t)|\, dt \tag{15}$$

$$J_{ISE} = \int_0^\infty (e(t))^2\, dt \tag{16}$$

$$J_{ITSE} = \int_0^\infty t\,(e(t))^2\, dt \tag{17}$$

These cost criteria are well known, and they can be used as a starting point. It is easy to modify them, and to derive more suitable versions for the control system in concern. It should be noted that the cost criteria above do not depend on the controller output, i.e. the control signal, at all. Optimization of these criteria may thus result in very sensitive controllers that are fast, but they come with the cost of high control signal usage. Often in practice, there are limitations concerning the usage of control signal.

Another cost criterion is more suited for controller tuning. Consider the optimal control law: the square of process states and the square of control

signals are weighted, summed and integrated over time. This applies if the reference signal is zero and the aim is to control all the states to zero. But if the reference is unequal to zero, the cost function needs to be revised. This is done in (18), where "IERC" (Integral of Weighted Sum of Square Error and Required Control Signal Error) criterion is presented.

$$J_{IERC} = \int_0^\infty \left[w_1 \left(e(t) \right)^2 + w_2 \left(y_r(t) - g \cdot u(t) \right)^2 \right] dt \quad (18)$$

The weights w_1 and w_2 define how much the error and the control signal usage are considered in the optimization. The bigger the w_1, the less the control signal is weighted and vice versa. The static gain g of the process scales the control and the reference signals to the same level. Thus the difference $y_r(t) - g \cdot u(t)$ equals zero, when the control signal is on the level that is required to have the process output on the reference signal level. [1]

4.2 Constrained optimization

There are many methods for unconstrained optimization (e.g. steepest descent or quasi-Newton's methods) that can be used if the decision variables (controller parameters) are not bounded and there are no other constraints affecting the optimization. Often in practice, there are limitations or requirements in the control system, and these can be expressed as constraints. There are different methods for solving unconstrained and constrained optimization problems. A general constrained optimization problem can be expressed as

$$\begin{aligned} &Min\ f(x) \\ &\text{s.t. } g(x) = 0 \\ &\quad h(x) \le 0 \\ &\quad x \in \mathbb{R}^n \end{aligned} \quad (19)$$

where f is the objective function to be minimized. If single-objective optimization is considered, f is a scalar-valued function. In a multi-objective case f is a vector. The feasible set for variables x is determined in (19) by *equality* and *inequality constraints* or $g(x)$ and $h(x)$, respectively. [5]

4.3 Sequential quadratic programming – SQP

Sequential quadratic programming (SQP) is an advanced method for solving constrained nonlinear optimization problems, and it is also used in the tool. The SQP algorithm consists of three main phases. At each iteration the Hessian matrix of the Lagrangian function is first updated. The Lagrangian is given by

$$L(x, \lambda) = f(x) + \sum_i \lambda_i g_i(x) + \sum_j \lambda_j h_j(x) \quad (20)$$

where λ_i and λ_j are the Lagrange multipliers. Secondly, a quadratic programming subproblem is solved and the solution is used to calculate a new search direction. Finally, the step length and the next iterate are calculated using a proper line search method. [7]

4.4 Optimization procedure

The optimization problem can be given constraints concerning e.g. the closed loop system performance, values of the controller parameters and the worst case delay of a network. These constraints need to be satisfied in order to successfully terminate the optimization. Constraints can be formulated such that the robustness of the control system is considered. The tuning tool uses a five-step simulation based optimization procedure shown in Fig.2 for solving the controller tuning parameters.

First, the initial parameters of the controller are chosen. Default or user given values may be used. After the initialization step, the iterative part of procedure is initiated. At each iteration, the closed loop system is simulated using the available controller parameters and the cost function is evaluated. After that a termination test is performed in order to see if it is necessary to continue the iteration. If the test is not passed, the SQP optimization algorithm updates the controller parameters based on the cost function value and optimization constraints. The iteration continues by simulating the system with the new controller parameters. If the test is passed, the iteration stops and the optimal parameters are available. The solution must satisfy all the constraints.

5 A tool for PID tuning

The described tuning procedure and the presented tool are implemented with MATLAB software. To enable an easy usage of the tuning program, a graphical user interface was built with several interactive properties. These include the selection of process, reference signal and controller parameters, drawing options, optimization criteria, delay characteristics and disturbance properties etc.

5.1 The graphical user interface – GUI

The graphical user interface of the tool is shown in Fig.3. In the upper left corner of the tool it is possible to select a linear transfer function process model for which the controller is tuned. Although only linear models are used here, the tuning method applies

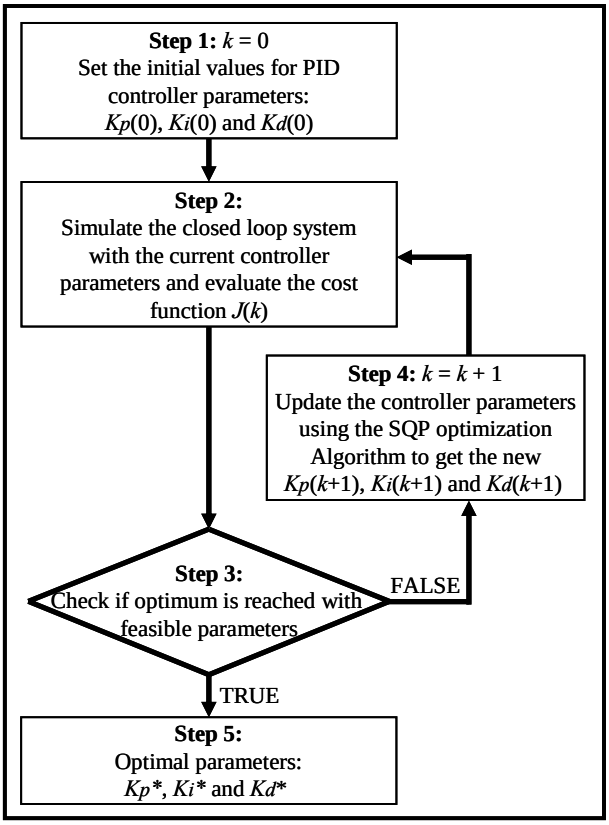

Fig.2. Optimal PID tuning (modified from [11]).

for both linear and nonlinear processes. Linearity is required in the tool for simplicity, but the tool could be extended for nonlinear processes.

In the middle of the top part of the tool a reference signal can be selected for the simulations. There are three possible reference signals implemented currently: a step, a sine wave and a series of steps. All these signals have certain parameters that can be chosen, e.g. the step time.

The tuning tool supports both continuous- and discrete-time PID controllers. The controller type can be selected in the upper right corner of the tool. There are also three sliders that are used for manual tuning of the controllers. Each slider represents a certain parameter of the PID controller. Once a slider is moved and released, the closed loop system is automatically simulated. Thus it is possible to tune the controller manually by moving the sliders back and forth, and by comparing the results. This is an important property, since often reasonable initial conditions need to be found before the optimization can be executed successfully. Using the sliders the controller initial parameters are easily found. Sometimes it is useful to have a possibility to set controller parameters very accurately without simulating the system. If some parameter values are known, they can be given into text boxes next to the sliders. The sliders are then automatically moved to the corresponding positions, but the system is not simulated.

In the middle part of the tool, the user can select the signals which are shown in the result figures. The recorded signals are reference, response (output), input (control), delay and disturbances. The results can be drawn in separate figures or in the scope on the tool. The former property is useful when comparing certain sets of parameters, and the latter when the controller is tuned. There are two scopes on the tool. The upper one represents the delay used in the simulations and the lower one can be used for displaying the signals.

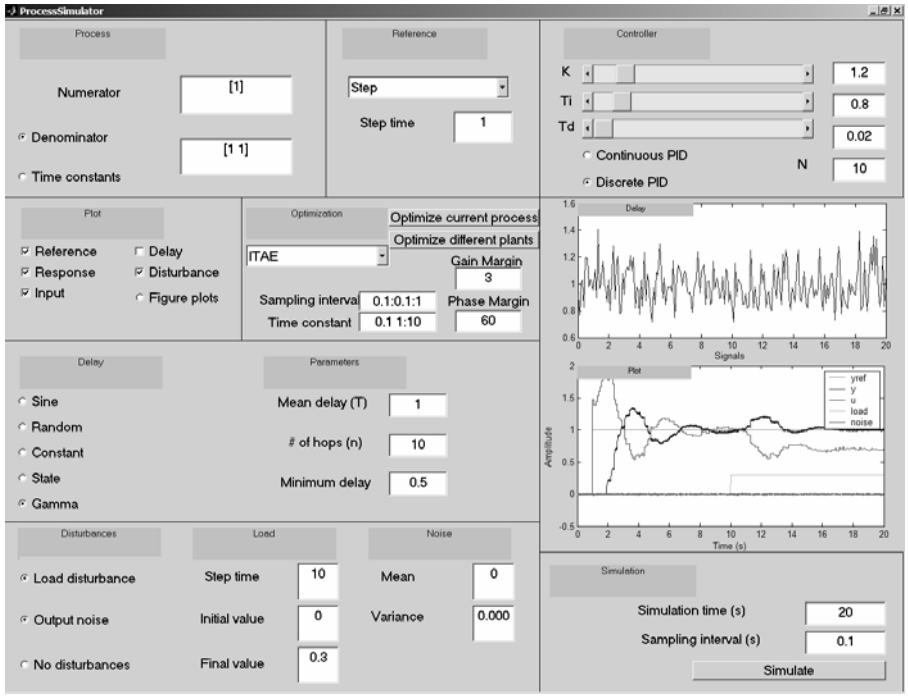

Fig.3. The PID controller tuning tool – GUI.

In the middle of the tool, there is a set of buttons and text boxes with which the optimization options are set. The five cost functions given in (14) - (18) can be used in the optimization. With the text boxes the user can set the required gain and phase margins for the system, and these are formulated into optimization constraints in the program. It is possible to optimize a controller for a certain process, but it is also possible to give several controller sampling intervals or even a set of process time constants for which controllers are optimized.

There are six delay types that can be used for describing the delay characteristics of a networked control system. All the delays are process output delays (see $\tau(t)$ in Fig.1) and the delay types are: constant, sinusoidal, state-dependent, random Gaussian distributed, random uniformly distributed and Gamma distributed. The parameters of the delays, such as mean value and variance of distributions, can be adjusted.

The effect of load disturbances and measurement noise can be examined with the buttons in the lower left corner of the tool. Noise brings realism into the simulations, since noise is always present in real systems. Load disturbances are also common, and the controller must be able to compensate these disturbances as well. If the disturbances are present in the simulations and during the optimization, more suitable controller parameters are obtained.

The simulation parameters such as simulation time and controller sampling time can be adjusted in the lower right corner of the tool. There is also a button that runs a single simulation after which the results are presented in the scope.

In the background, there is a rather complicated program that sets up the simulation model based on the user's selections. The model is then simulated and results are sent back to the main program that presents the results to user via the GUI.

5.2 Data structures and program blocks

The basic data type used in the tuning program is a structure that contains all the information about process, controller, delay, disturbances etc. The data structure is built when either simulation or optimization buttons are clicked on the GUI. All users' selections and simulation information are gathered from the GUI and recorded into the structure. Once the structure is built, the program inserts the simulation information from the structure into the simulation model. There are special functions in the program that are made for transferring the information easily from the GUI into the structure and onwards into the simulation model.

After the optimization procedure, the controller parameters and cost function values are added into the structure. If the sampling interval or process time constants are varied during the optimization and thus several controllers are obtained, the results are recorded into a structure array. In this way, the results are saved in one variable (the array) that is easy to handle and e.g. to save on a disk.

The execution phases of the tuning program are presented in Fig.4. It can be seen in the flow chart that there are three basic actions that the user can choose: simulation, optimization of a single process and optimization of several controllers or processes. The optimization actions execute the constrained optimization algorithm (SQP) which results in optimal controller parameters. If several processes or controllers are to be optimized at once, the program saves the latest results in the structure array and updates the controller or process information. Then the optimization is rerun with updated parameters. The results are shown to the user after the simulation or the optimization is successfully terminated.

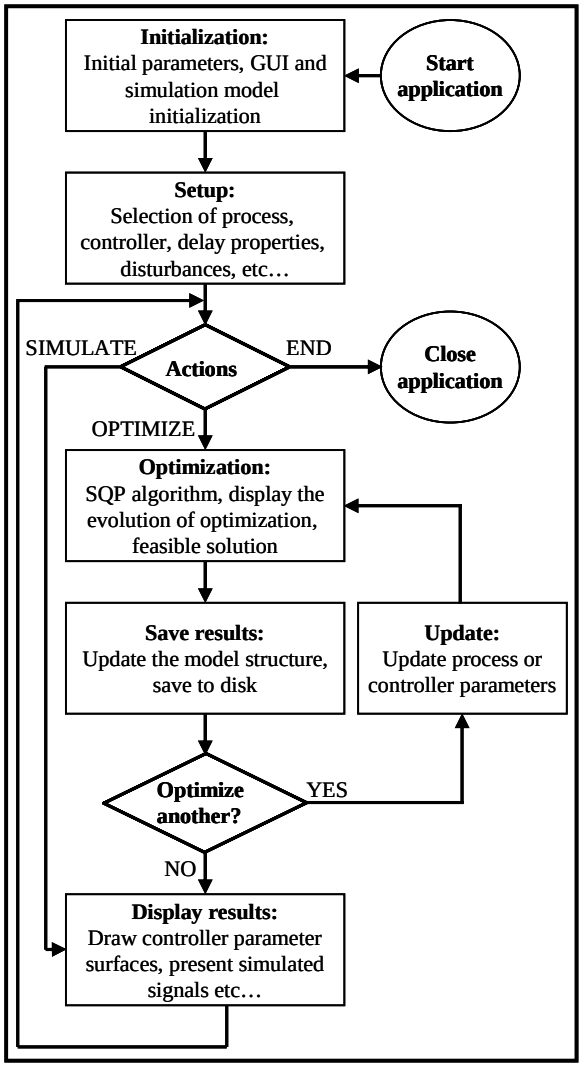

Fig.4. Program execution phases of the tuning tool.

5.3 Displaying the results

After tuning the controller, the program automatically displays the optimal tuning parameters and cost function values. If only one controller is tuned for a certain process, the results are shown in 2D-figures, where the horizontal axis represents the controller sampling time (in discrete-time PID case) and the vertical axis represents the value of the parameter. If several processes or sampling intervals are examined at once, the results may be shown as in Fig.5 in 3D-plots. There the controller parameters form surfaces as functions of sampling interval and time constant of a first order process. From this kind of figures it is easy to see how the parameters are affected by the sampling interval or process time constant.

6 Conclusions

This paper presented a PID controller tuning tool which is designed especially for networked control systems. There are not many methods for designing controllers for networked systems, since the varying delays make the tuning and the control design problematic. The presented tuning procedure is based on modeling of processes and network delays. Simulation and optimization techniques are used for finding the optimal controller parameters. In the tool, there are six different delay types that can be used for characterizing the network delays. The parameters of the delays, processes and disturbances can be adjusted. The tool can utilize constrained optimization in order to improve e.g. the robustness of the control system. Particularly, in networked control systems certain constraints related to the variance of delay improve robustness. The optimization can be performed e.g. such that not even some maximum delay will endanger the stability. A graphical user interface is implemented to make the use of the tuning procedure as easy as possible.

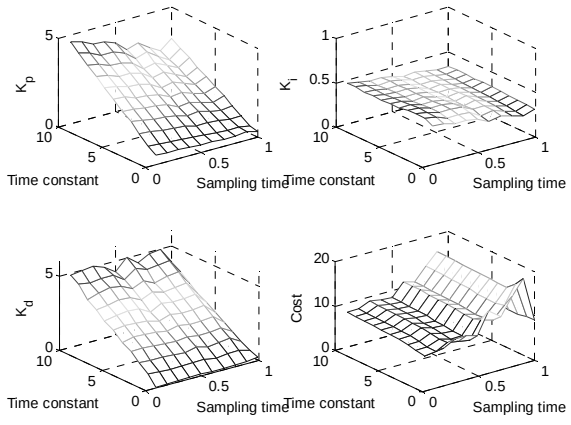

Fig.5. An example of the tuning results.

The tool is implemented with MATLAB software. Using the tools included in MATLAB, it is possible to compile the tuning tool into a stand-alone program. The compilation would enable easy distribution of the tool. The interactivity and an easy user interface enable using the tool in education, training and control design. The well-designed implementation of the tuning program facilitates the maintenance and the future development of the tool.

References:

[1] V. Hölttä, L. Palmroth, L. Eriksson, *Rapid control prototyping tutorial with application examples*, Sim-Serv, www.sim-serv.com, 2004.

[2] H. N. Koivo, A. Reijonen, Tuning of PID Controllers for Varying Time-Delay Systems, *IEEE Int. Conf. on Mechatronics (ICM'04)*, 2004.

[3] P. H. Lewis, C. Yang, *Basic Control Systems Engineering*, Prentice Hall, 1997.

[4] B. Lincoln, *Dynamic Programming and Time-Varying Delay Systems*, Ph.D. thesis, Dep. of Automatic Control, Lund Inst. of Tech., 2003.

[5] G. P. Liu, J. B. Yang, J. F. Whidborne, *Multiobjective Optimisation and Control*, Research Studies Press Ltd., 2003.

[6] V. Lucan, P. Simacek, J. Seppälä and H. Koivisto, *Bluetooth and Wireless LAN Applicability for Real-Time Control*, Automaatio2003, Helsinki, Finland, 2003.

[7] J. M. Maciejowski, *Predictive Control with Constraints*, Pearson Education Ltd., 2002.

[8] A. Mukherjee, On the Dynamics and Significance of Low Frequency Components of the Internet Load, *In Proc. of the First ACM SIGCOMM Workshop on Internet Measurement*, pp. 281 – 293, San Francisco, USA, 2001.

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

[10] N. J. Ploplys, P. A. Kawka and A. G. Alleyne, Closed-Loop Control over Wireless Networks, *IEEE Control Systems Magazine*, Vol. 24, pp. 58 – 71, 2004.

[11] A. Reijonen, *Tuning of PID Controller for Varying Time-Delay Systems*, Master's thesis, Helsinki University of Technology, 2003.

[12] K.J. Åström, T. Hägglund, *PID Controllers: Theory, Design, and Tuning*, 2nd ed., Instrument Society of America, 1995.

[13] K. J. Åström, B. Wittenmark, *Computer-controlled Systems – Theory and Design*, 3rd ed., Prentice Hall, 1997.

[14] http://www.isc-ltd.com/resource_centre/tech_pid.html, Jan, 18th 2005.

[15] http://bestune.isclever.com/, Jan, 18th 2005