IFAC Conference on Advances in PID Control
PID'12
Brescia (Italy), March 28-30, 2012 WeB2.2

Software for PID Design: Benefits and Pitfalls [⋆]

O. Garpinger [*] **T. Hägglund** [*] **L. Cederqvist** [**]

[*] *Department of Automatic Control, Lund University, Lund, Sweden, (e-mail: {olof,tore}@control.lth.se).*
[**] *Swedish Nuclear Fuel & Waste Management Company, Oskarshamn, Sweden, (e-mail: Lars.Cederqvist@skb.se).*

Abstract: The most common PID design methods in industry are based on formulas. This article will present some major advantages of instead using the power of computer based softwares for PID controller design. The Matlab based software used in this work was developed in 2007 and derives robust, IAE minimizing, PID controllers. The experiences of using this software are collected in this article and include control signal activity limitation due to measurement noise, controller design on an industrial Friction Stir Welding process and fast controller design for large batches of processes. It is shown that the properties of the software make it suitable for design of PID controllers and in PID research. There are, however, some possible design pitfalls that the user needs to be aware of. Some of these are presented as well.

Keywords: PID control, computer software, optimization, disturbance rejection, robust control, measurement noise, friction stir welding.

1. INTRODUCTION

A useful tool for control design should consider: set-point tracking, load disturbance attenuation, robustness to process uncertainty and noise. The first two requirements are typically satisfied by high gain and bandwidth, the other requirements are improved by low gain and bandwidth.

The PID controller is by far the most common controller in industry, used in vast amounts of control loops in factories. In order to handle these many loops, it is important to have a fast modeling tool as well as a fast PID design tool. The most common methods in industry are based on step response analysis, for modeling, and formula methods for design (like lambda tuning (Dahlin (1968)), AMIGO (Hägglund and Åström (2004)) and SIMC (Skogestad and Grimholt (2012))). The formula methods are usually very easy to use for PI control and only require knowledge from first order systems with time delay (FOTD)

$$P(s) = \frac{K_p}{sT+1}e^{-sL}, \tag{1}$$

with static gain K_p, time constant T and time delay L. Such systems can be characterized by their normalized time delay, $\tau = L/(L+T)$, where $\tau \approx 0$ indicates a lag-dominated process, $\tau \approx 1$ delay-dominated and intermediate τ-values balanced processes. Many formula methods can derive robust PI controllers that suit industrial processes very well. Due to the extra complexity of noise handling, however, most of them struggle to derive good PID controllers or PI controllers with limited noise sensitivity. This may be one of the reasons why the D-part of the PID controller is so seldom used in industry, even though there are processes that would obviously benefit from it. One way to get around this issue is to use computer software for PID design rather than a formula method. Such programs can be used to handle both more complex processes and additional constraints on noise sensitivity, which will limit actuator wear and tear. This article will present such a software, written in Matlab, and show some of its major benefits, as well as some pitfalls. The software is free to download from `www.control.lth.se/Research/ProcessControl/PIDControl.html`. The article is based on the experience that the authors have gained after using the software over a time period of four years. This includes controller tuning for a real industrial application, the Friction Stir Weld. The article is based on previous work, presented in Garpinger and Hägglund (2008), Garpinger (2009) and Cederqvist et al. (2012), that gives deeper knowledge on each part handled in this article. Another example of the benefits of PID software can be found in e.g. Larsson and Hägglund (2011).

[⋆] This work was funded by the Swedish Nuclear Fuel & Waste Management Company and the Swedish Research Council.

2. A SOFTWARE FOR ROBUST PID CONTROL

The Matlab software is mainly concerned with load disturbance rejection and robustness. Set-point following is generally of less interest in e.g. process industrial applications (Shinskey (1996)) and can be treated separately from load disturbance handling, as shown in Åström and Hägglund (2005). While noise sensitivity is not handled directly by the software program, it will be shown in Section 3 how it can be taken care of through repetitive controller design by the software, using different lowpass filter settings.

The program finds parameters of the controllers

$$C_{PI}(s) = K(1+\frac{1}{sT_i})\cdot\frac{1}{(T_f s+1)}, \tag{2}$$

$$C_{PID}(s) = K(1+\frac{1}{sT_i}+sT_d)\cdot\frac{1}{(s^2T_f^2/2+sT_f+1)}, \tag{3}$$

where K is the proportional gain, T_i the integral time, T_d the derivative time and T_f the lowpass filter time constant. T_f is chosen by the software user and therefore an important part of the controller design. While the above controllers are the default forms that the software uses, the software can be easily modified to handle for example; 1. discrete controllers, 2. P-, I- and PD-control, 3. lowpass filtering of the D-part alone or first order filters on the measurement signal.

The PID controller minimizes the Integrated Absolute Error (IAE)

$$\text{IAE} = \int_0^\infty |e(t)|dt, \tag{4}$$

during a unit step disturbance, d, on the process input (see Fig 1). The control error, e, is the difference between the controlled variable, y, and its reference value (not shown in the figure).

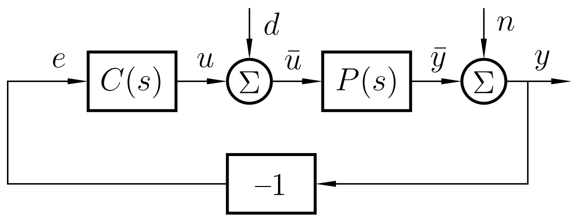


Fig. 1. A load disturbance, d, and measurement noise, n, act on the closed loop system with process $P(s)$ and PID controller $C(s)$.

The IAE optimization will guarantee stability and is constrained by H_∞ robustness conditions

$$|S_s(i\omega)| \leq M_S, \quad |T_s(i\omega)| \leq M_T, \quad \forall\omega\in\mathcal{R}^+, \tag{5}$$

$$|S_s(i\omega_S)| = M_S \text{ and/or } |T_s(i\omega_T)| = M_T, \tag{6}$$

where $S_s(s)$ and $T_s(s)$ are the sensitivity function and the complementary sensitivity function respectively. The conditions stated in (6) force at least one of the constraints to be active (in the frequency points ω_S and/or ω_T). These constraints are known to set closed loop robustness towards process variations, disturbances and non-linearities as described by e.g. Åström and Hägglund (2005). M_S and M_T are by default set to 1.4 in the software, which is known to give good robustness (e.g. phase margin 41.8° and gain margin 3.5), but they can also be changed by the user if desired. The design method is influenced by Panagopoulos et al. (2002), Hägglund and Åström (2004) and Nordfeldt (2005). Note that the process high-frequency phase-lag must be less than −180° for PID controllers and −90° for PI controllers, otherwise the closed loop can be made arbitrarily fast.

The optimization problem is non-convex due to the fixed order of the controller, which complicates the solution finding. The equality constraints do, however, simplify the optimization problem. Choosing M_S and M_T within reasonable boundaries (giving good robustness) the minimum will also seldom lie within the constraints (see Garpinger (2009)). One can, therefore, be almost certain to have found the minimum also for the optimization problem without the equality constraints (6). The optimization problem solution is found by applying the Nelder-Mead algorithm (Nelder and Mead (1965)) in the T_i-T_d plane, where K is chosen for every point such that at least one of the equality constraints is active. The IAE-values for each controller are derived using Matlab Simulink simulations and the Nelder-Mead iterations end when it is close enough to the minimum.

3. LIMITING NOISE SENSITIVITY

Neglecting noise sensitivity can lead to bad controllers, particularly for systems with lag-dominated dynamics as illustrated by the following example.

Example 1. Consider the process

$$P(s) = \frac{1}{500s+1}e^{-s}.$$

Designing a continuous time PID controller using the software, with no lowpass filter and $M_S = M_T = 1.4$, gives $K = 236.32$, $T_i = 4.91$ and $T_d = 0.41$ with $IAE = 0.0023$. Assuming that the controller is implemented on the real process with a sampling time of $h = 5$ seconds, discretizing the I-part through forward Euler, the D-part using backward Euler and the lowpass filter using Tustin discretization, it can be shown that the closed loop is not even stable. Instead using the software for discrete PID design will, on the other hand, give $K = 62.77$, $T_i = 21.05$ and $T_d = 0.88$ with $IAE = 0.37$, which gives the correct robustness. Applying white noise with unit variance on the process input will, however, give a control signal that has more than 7500 times greater variance than the noise signal itself. This controller will thus also be more or less useless, unless the measurements contain very little noise.

3.1 Extending the optimization problem

Large controller gains are not an uncommon issue in PID controller design. AMIGO (Åström and Hägglund (2005)) and SIMC (Skogestad and Grimholt (2012)) approaches the problem by detuning of the PID parameters. In this paper, it will be handled through lowpass filter tuning. The lowpass filter time constant, T_f can be chosen in several different ways. Two of the most common approaches in industry are to either choose the filter time constant before the actual controller tuning or to relate it in some way to the controller parameters. The first approach has the advantage that the filter is actually part of the controller tuning, which is good since it will change the process and thus also the robustness if the filter is set after the controller is already tuned (like the second approach does). Choosing the lowpass filter without knowledge of how it affects the closed loop could, however, also change the process such that the closed loop becomes unnecessarily slow. Isaksson and Graebe (2002) show that the best approach would be to use four parameter design if possible, thus tuning the lowpass filter at the same time the PID controller parameters are tuned. This is also the approach presented in this paper, which was first suggested in Garpinger (2009). Some other methods using four parameter PID design are described by Kristiansson and Lennartson (2006) and Larsson and Hägglund (2011). Larsson also shows that a first order lowpass filter is optimal for PI control, while a second order filter is optimal for PID controllers. This shows that (2) and (3) are well-suited for four parameter (three for PI control) design.

Four parameter design is obtained by adding the constraint

$$\|S_k\|_2^2 = \frac{\sigma_u^2}{\sigma_n^2} \leq V_k. \tag{7}$$

S_k is the transfer function from the measurement noise n to the control signal, u. σ_u^2 is the variance of the control signal due to the measurement noise and σ_n^2 is the variance of the measurement noise itself. V_k is a user set limit on the variance gain from noise to control signal. $V_k = 1$ will thus correspond to u and n having the same variance. In Example 1, $\|S_k\|_2^2 > 7500$.

Depending on how high or low V_k is set, will also determine whether a PI or a PID controller is preferable. This can be shown by deriving optimal Youla parametrized controllers using the tool described by Wernrud (2008). These controllers are optimized using exactly the same constraints as the PID controllers (except that the robustness constraints do not need to be active), but with a much higher controller order making the optimization problem convex. Running the Youla optimization tool on the process $P(s) = 1/(s+1)^4$, with sampling time $h = 0.25$ s and four different values on $V_k = 0.04, 0.2, 1, 5$ gives the four different optimal controllers with Bode diagrams shown in Fig. 2. It is apparent that the controllers are given less freedom for lower V_k-values. The resemblance also goes first towards PID controller and then PI, when V_k is really small. For even lower values on V_k, it could even be that an I controller would be preferred. The level of variance at which a more advanced controller is preferred does of course vary from process to process.

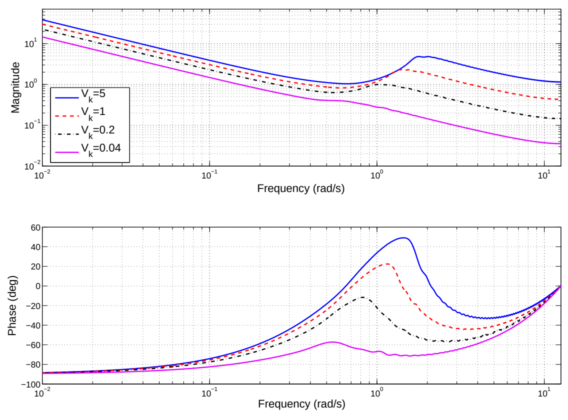


Fig. 2. Optimal Youla parametrized controllers on a fourth order lag process for four different values on V_k.

3.2 A recommended design procedure

Given a real process, the Matlab software can be used to also satisfy the constraint (7). The key is to vary the lowpass filter time constant T_f. The different PI or PID controllers derived from the software will affect both the IAE-value and V_k. For each controller, V_k can be determined using e.g. closed loop simulations, ideally with real noise data as input. One way to derive the noise data is by detrending open loop data from the process. For most processes, this can be derived e.g. during modeling tests (like step responses). The different IAE and V_k-values derived results in a trade-off curve between IAE and V_k, such as the one shown in Fig. 4, later on, which can be interpreted as how much a decrease in the noise sensitivity (or indirectly, actuator wear and tear) costs in terms of performance. If at all possible, a good choice seems to be if one can pick a controller close to where the trade-off curve bends the most (i.e. around $V_k = 3$ in Fig. 4). Outside this region, a small value on V_k may be expensive in terms of IAE and the opposite holds if good performance is wanted. The latter can be related to Example 1, where the IAE value was very low at the expense of high noise sensitivity.

It is important to remember that the software was not made for solving the extended optimization problem. For example, if $V_k = 1$ is given by $T_f = 0.1$, for some random process, it could still be that a different value on T_f, with a different set of controller parameters may give the same noise sensitivity with better overall performance. Also remember that the lowpass filter orders are different for PI and PID control. In the case that the optimal controller is a PI, the software will provide PID controllers with worse performance. Running the software on the first order process with time delay in Example 1 and $V_k = 1$ provides a good example on this. The PID controller with $K = 3.79$, $T_i = 166.5$, $T_d = 115.1$ and $T_f = 110$ gives noise sensitivity close to $V_k = 1$. Its IAE-value is, however, around 40% higher than that given by the PI controller with parameters $K = 4.62$, $T_i = 246.7$ and $T_f = 50$, which results in the same noise sensitivity. It is generally a good idea to check the performance of the PI controller when the quotient $T_f/T_d \approx 1$ for PID control. In Garpinger (2009), these kinds of issues are discussed in greater detail. It is also shown that many of the controllers given by the proposed software design procedure result in controllers that are very close to optimal Youla controllers in performance. This shows that PI and PID controllers are often the preferable choice, especially when the process is noise sensitive.

4. FRICTION STIR WELDING

In this section it will be shown how the Matlab software can be used for an industrial application, namely a Friction Stir Welding (FSW) process.

4.1 FSW of thick copper canisters for nuclear waste

FSW is a thermo mechanical solid-state process that was invented in 1991 at The Welding Institute (TWI) in Cambridge, England. A rotating non-consumable tool, consisting of a tapered probe and shoulder, is plunged into the weld metal and traversed along the joint line, see Fig. 3. Frictional heat is generated between the tool and the weld metal, causing the metal to soften, normally without reaching the melting point, and allowing the tool to traverse the joint line. The three most common input parameters are listed according to Fig. 3; 1. tool rotation rate [rpm], 2. welding speed along the joint [mm/s], and

3. axial force [kN]. A previous study by Cederqvist et al. (2008) has shown that the rotation rate is the best suited control signal and will be used in this paper.

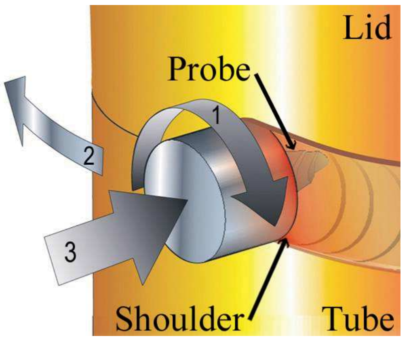


Fig. 3. Illustration of the Friction Stir Welding process to seal copper canisters.

The Swedish Nuclear Fuel and Waste Management Company (SKB) plans to join at least 12,000 lids and bases to the extruded copper tubes containing Sweden's nuclear waste, using Friction Stir Welding. The canisters produced are 5 m high, 5 cm thick with 1 m diameter. The Friction Stir Welding machine used measures the welding temperature [°C] and the torque [Nm] required to maintain the tool rotation rate. Another important variable is the power input [kW] which is proportional to the tool rotation rate multiplied with the torque. Cederqvist et al. (2008) showed that the power input is well correlated to the welding temperature.

Of all process outputs, the welding temperature is the most crucial to control. If the welding temperature gets too high, for a longer period of time, there is a risk for probe fracture. Similarly, too low temperatures may result in discontinuities in the weld. It is, therefore, very important to keep the temperature within this, so called, process window, which is roughly between 790 and 910°C for FSW on the copper canisters (when measured inside the probe).

Several aspects of the welds make them challenging to control. A full weld cycle can be divided into five separate sequences (depending on the position of the tool) that will each present challenges of different nature. For example, the start-up will contain a lot of fast and high-magnitude torque disturbances. Other disturbances are caused either by the tool moving in and out of preheated areas or by greater heat conduction at the joint line compared to the lid (where the welds are started). These disturbances can not be measured, but they are on the other hand also much slower than the torque disturbances. Each of these disturbances has to be counteracted to make sure the temperature is within the process window.

4.2 Cascade control

A cascaded control strategy seems ideal for the characteristics of this specific FSW application with its fast, multiplicative, torque disturbances and slower temperature counterparts. The process was divided into two subsystems. The inner process G_1 that holds the dynamics from the tool rotation rate (i.e. the control signal) to the power input, and the outer process G_2 which describes how

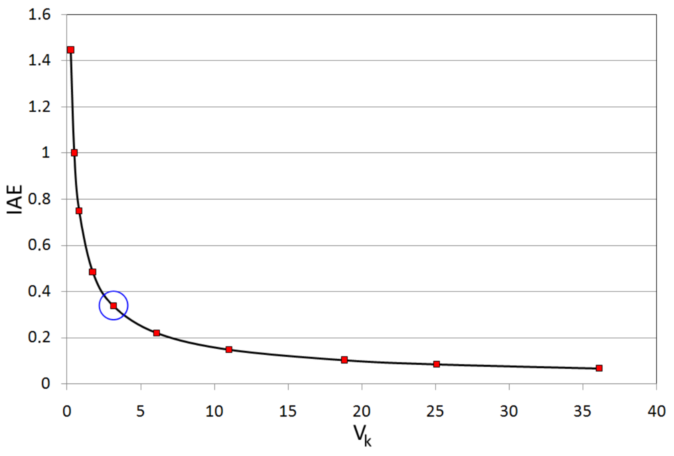


Fig. 4. Performance and noise sensitivity trade-off curve for the inner control loop. The chosen controller is marked with a blue circle.

the power input is related to the welding temperature. The inner process will be handled by a controller C_1, whose power reference will be provided by the outer loop controller C_2. The temperature reference will be set to somewhere between 840 and 850°C (close to the middle of the process window), and the sampling time is $h = 0.1$ s. A model of the inner process was determined through step response analysis and given by

$$G_1(s) = K_{p_1} \cdot \frac{\omega_1^2}{s^2 + 2\zeta_1\omega_1 s + \omega_1^2}$$
$$= 0.12 \cdot \frac{4.6^2}{s^2 + 2 \cdot 0.8 \cdot 4.6s + 4.6^2}. \qquad (8)$$

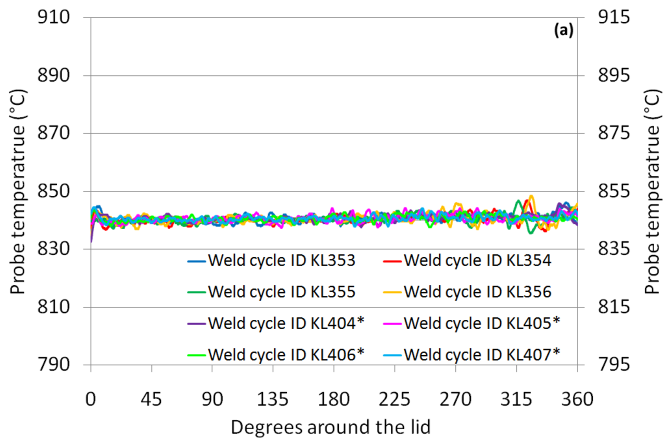
The inner controller, C_1 should be chosen to reject the torque disturbances, but the demands on the speed of the inner loop are, however, quite moderate at the same time as the torque measures contain a lot of noise. A PI controller was therefore chosen for the inner loop. Using the Matlab software on the process model G_1 showed that tuning of a lowpass filter together with the PI controller had little effect on the relative noise throughput, V_k. As a way to still be able to set V_k, one can instead vary the robustness measures M_S and M_T. Therefore, 10 different PI controllers, with 10 different M_S- and M_T-values (from 1.015 to 1.35), were tuned using the software. Fig. 4 shows the trade-off curve for the inner loop.

The PI controller with $K = 1.07$ and $T_i = 0.36$ gives $IAE = 0.34$ and $V_k = 3.15$ (marked with a blue circle in the figure). $M_S = M_T = 1.065$ which means that the inner loop has very good robustness.

Closing the inner loop, one can run step responses to determine the outer process model

$$G_2(s) = \frac{K_{p_2}}{(sT_2+1)^2} e^{-L_2 s} = \frac{11.6}{(7s+1)^2} e^{-5s}. \qquad (9)$$

Unlike the torque signal, the temperature measurements contain very little noise and the outer loop can be tuned without taking control signal (power input reference) activity into account at all. Two PI controllers were tuned, one with $M_S = M_T = 1.4$, giving $K = 0.033$ and $T_i = 11.2$, that is active during the joint line sequence, and another with $M_S = M_T = 1.8$ ($K = 0.065$ and $T_i = 14.0$) for the start-up sequences when there are heavy



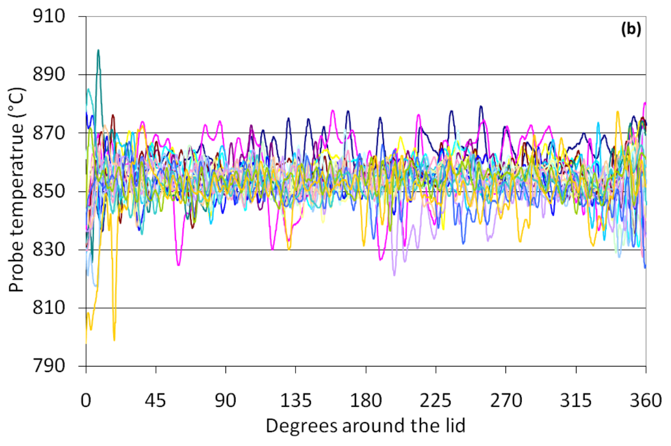


Fig. 5. Probe temperatures at the joint line using the cascade controller (a) compared to using manual control (b). A * refers to values on the right y-axis.

temperature disturbances active on the process. Note that the two controllers were designed for the inner loop in series with the outer process (9). The resulting process is a higher order system with time delay (HOTD). Fig. 5 (a) shows the temperature signals during the joint line sequences of 8 different full welds, which are well inside the process window. This also justifies use of PI controllers rather than PID, which are not needed to control the process sufficiently. For comparison, Fig. 5 (b) displays the temperature signals for 20 different weld sequences that are manually controlled. These are much closer to the limits of the process window.

4.3 Tuning for batches of processes

Due to the vast amount of canisters that will be welded, the process may be used for more than 40 years. During this time span, it is natural that the process alters in some ways and it may thus be useful to retune the controllers periodically. To not be dependent on the compatibility of Matlab versions over the years (for use of the software), it seems wise to cover up for potential process alterations already at this point in time.

Given the models (8) and (9), assume that each of the parameters are changed to either of the values $K_{p_1} = [0.071, 0.133, 0.195]$, $\omega_1 = [1.50, 7.65, 13.80]$, $\zeta_1 = [0.5, 0.75, 1.0]$, $T_2 = [2.3, 11.65, 21]$ and $L_2 = [0, 5, 10]$. These processes should cover the worst case scenarios.

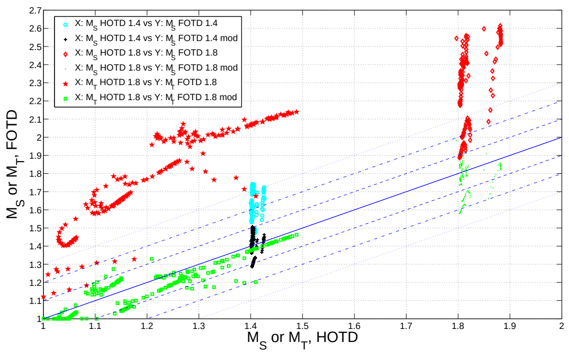


Fig. 6. Closed loop robustness for controllers determined with the HOTD model, compared to those tuned with the FOTD model.

Assuming that the inner controllers have already been tuned properly, four types of outer controllers will be derived for all 243 possible process combinations. Two for the cases $M_S = M_T = 1.4, 1.8$ and two when either using the more advanced process models (8) and (9) (High Order Time Delayed, HOTD model) or when using the approximate first order process with delay (see (1)) that has time delay L_Y and time constant T_Y (FOTD model). The FOTD model can easily be determined using simple step response modeling, which has the advantage that the controller designer does not need to have an extensive education in automatic control, which may be necessary for deriving the HOTD model.

Comparing the different controller tunings, the FOTD models often give more aggressive PI controllers than the HOTD models, both concerning proportional and integral gain. This can lead to closed loop processes with much worse robustness than for the process it is designed for. Fig. 6 shows M_S and M_T for the HOTD designed controllers compared to the FOTD designed. Plotting the FOTD controller parameters against the τ-values of the processes, reveals that lag-dominant processes more often give controllers with bad robustness. A simple modification of the FOTD PI parameters give the controllers marked with green and black, in the figure, which have much better robustness overall. The performance of the modified controllers range between 10% better than the HOTD controllers to 30% worse, which seems acceptable. The modified PI controllers had the PI parameters $K_{mod} = K/1.35, T_{i_{mod}} = T_i/0.9$ $(\tau < 0.3)$, $K_{mod} = K/1.35, T_{i_{mod}} = T_i$ $(\tau \geq 0.3)$ (for $M_S = M_T = 1.4$) and $K_{mod} = K/1.15, T_{i_{mod}} = T_i/0.7$ $(\tau < 0.3)$, $K_{mod} = K/1.25, T_{i_{mod}} = T_i/0.95$ $(\tau \geq 0.3)$ (for $M_S = M_T = 1.8$).

Depending on the process parameters, the FOTD model parameter T_Y can vary between 4 and 48, while the same limits for L_Y are 1 and 26. Processes with lower τ-value has to be gridded closer together not to lose the robustness frames given in Fig. 6. The controllers for $M_S = M_T = 1.8$ will also need a finer grid than 1.4 to achieve the same robustness goals. Therefore, T_Y and L_Y were gridded with 10,208 different points for $M_S = M_T = 1.4$ and 32,817 points for 1.8. All 43,025 PI controller designs took roughly 2 days to carry out on a fairly regular laptop. Fig. 7 shows how the proportional gain K, for the outer controller, depends on T_Y and L_Y when $M_S = M_T = 1.8$. These

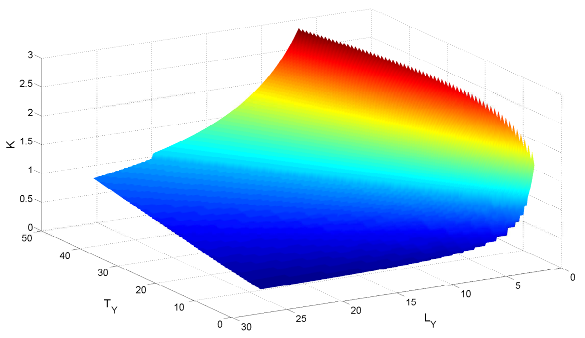


Fig. 7. Controller gains for a variety of FOTD process models, derived with the robustness criteria $M_S = M_T = 1.8$ in the Matlab software.

controller parameter values can e.g. be put into any matrix and then be used when the controllers are retuned.

A different, and perhaps better, approach would be to use formula methods on the processes instead of designing this many controllers. Trying AMIGO on the batch of processes shows that it will give nearly the same performance and robustness as the modified FOTD controllers did. It does, however, not apply for $M_S = M_T = 1.8$. Anyways, the only matter of real importance here is that the software makes it possible to derive this many controllers in relative short time. This can be used in e.g. PID controller research to get a better understanding of how the controllers affect the closed loop, which was shown when comparing the HOTD controllers to the original FOTD ones.

5. CONCLUSIONS AND FUTURE VISIONS

The goal of the paper was to show some major benefits of using software for PID design. With fairly simple algorithms, it is possible to find the solution to the non-convex problem that gives IAE minimizing PI or PID controllers with preset robustness measures. The software is also easy to modify, robust and free to download. It can be used to find controllers that limit the noise sensitivity, which is especially important if the process is lag-dominant and if PID controllers are used. The preferred controller structure will depend on the complexity of the process and the acceptable control signal activity, V_k. It was shown that the PID design software can be used in several aspects on a real industrial application, the Friction Stir Welding process. Controller tuning for several possible process variations showed that it could be dangerous to design for first order time delayed processes if the real process is more complex. There was, however, a straightforward modification to the controllers that gave well performing closed loops with acceptable robustness. Another big benefit of the software is that it can derive controller designs on large batches of processes in reasonable time.

To develop the possibilities of PID design softwares even more, there are several areas that could use some attention. For example, it would be very useful to combine the design tool with a modeling tool to create an auto-tuner. Models of higher order than one are obviously preferable, but the question is how accurate the models need to be to allow for direct software tuning. A new software tool should also be extended to find the optimal controllers when noise sensitivity constraints are active. An alternative would be to let it automatically choose an I-, PI- or PID controller depending on which is preferable. The control activity constraint may also need to be reworked for people in industry to get a more intuitive understanding.

REFERENCES

Åström, K.J. and Hägglund, T. (2005). *Advanced PID Control*. ISA - The Instrumentation, Systems, and Automation Society, Research Triangle Park, NC 27709.

Cederqvist, L., Garpinger, O., Hägglund, T., and Robertsson, A. (2012). Cascade control of the friction stir welding process to seal canisters for spent nuclear fuel. *Control Engineering Practice*, 20(1), 35–48.

Cederqvist, L., Sorensen, C.D., Reynolds, A.P., and Öberg, T. (2008). Improved process stability during friction stir welding of 5 cm thick copper canisters through shoulder geometry and parameter studies. *Science and Technology in Welding and Joining*, 46(2), 178–184.

Dahlin, E.B. (1968). Designing and tuning digital controllers. *Instruments and Control Systems*, 42, 77–83.

Garpinger, O. (2009). Design of Robust PID Controllers with Constrained Control Signal Activity. Licentiate Thesis LUTFD2/TFRT--3245--SE, Department of Automatic Control, Lund University, Sweden.

Garpinger, O. and Hägglund, T. (2008). A Software Tool for Robust PID Design. In *IFAC World Conference*. Seoul, South Korea.

Hägglund, T. and Åström, K.J. (2004). Revisiting the Ziegler-Nichols step response method for PID control. *Journal of Process Control*, 14(6), 635–650.

Isaksson, A.J. and Graebe, S.F. (2002). Derivative filter is an integral part of PID design. *Control Theory and Applications, IEE Proceedings*, 149, 41–45.

Kristiansson, B. and Lennartson, B. (2006). Evaluation and simple tuning of PID controllers with high-frequency robustness. *Journal of Process Control*, 16, 91–102.

Larsson, P.O. and Hägglund, T. (2011). Control signal constraints and filter order selection for PI and PID controllers. In *2011 American Control Conference*. San Francisco, California, USA.

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

Nordfeldt, P. (2005). PID Control of TITO Systems. Licentiate Thesis ISRN LUTFD2/TFRT--3238--SE, Department of Automatic Control, Lund University, Sweden.

Panagopoulos, H., Åström, K.J., and Hägglund, T. (2002). Design of PID controllers based on constrained optimisation. *IEE Proceedings - Control Theory & Applications*, 149(1), 32–40.

Shinskey, F.G. (1996). *Process-Control Systems. Application, Design, and Tuning*. McGraw-Hill, New York, 4th edition.

Skogestad, S. and Grimholt, C. (2012). The SIMC method for smooth PID controller tuning. In R. Vilanova and A. Visioli (eds.), *PID Control in the Third Millennium Lessons Learned and New Approaches*. Springer.

Wernrud, A. (2008). *QTool 0.1—Reference Manual*. Department of Automatic Control, Lund University, Lund, Sweden.