Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Modeling for Optimal PID Design

Olof Garpinger, Tore Hägglund

Department of Automatic Control, Lund University, Lund, Sweden
(e-mail: [olof,tore]@control.lth.se)

Abstract: Even though PID controllers have been around for a long time, few industrial controllers use derivative action and the remaining PI controllers are often designed with formula-based tuning rules rather than through computer-based optimization. This paper will delve into some of the reasons behind these choices and show potential benefits of instead using software-based PID tuning. Three commonly used tuning rules are compared to software tuning with respect to performance and robustness over a large process batch. The study shows the importance of combining a fast, accurate modeling tool with the software design method and gives guidelines for future modeling tools with regards to desired process information. With moderate process knowledge it is possible to design controllers that are much closer to optimal than the three tuning rules, with significant performance improvements as a result.

Keywords: PID control, optimization, computer software, modeling, process control.

1. INTRODUCTION

The low order of the PID controller is well-suited for use in the process industry where tuning time is of the essence. A good PID tuning method should thus both be fast and easy to carry out for the large number of control loops in a factory. This has led to the great popularity of formula-based tuning rules, which typically need some basic knowledge or model of the process. In O'Dwyer (2009), there are 1,730 PI and PID tuning rules collected. We will, however, compare some commonly used tuning rules to computer-driven optimization and argue that there should be at least one more tuning method.

Although the benefits of derivative action are well-known, it is most often turned off in industrial PID controllers. Reasons for this include increased noise sensitivity, variety of controller structure, and the difficulty of tuning 1-2 more parameters including noise filter design. To hand-tune a PID controller quickly is thus rather difficult, and there are no PID tuning rules that have gained wide acceptance in industry. In this paper, we will show the importance of combining the tuning method with a suitable modeling tool, similar to the results by Leva and Schiavo (2005). Together with our software-based design tool we will also show the potential benefits of using such a tuning method both in terms of robustness and performance.

2. THEORY

A PI controller is often parametrized in terms of proportional gain K and integral time T_i, while a PID controller also includes the derivative time T_d. In this paper we will mainly consider ideal PI and PID controllers

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

$$C_{PID}(s) = K(1 + \frac{1}{sT_i} + sT_d), \tag{2}$$

without noise filtering.

2.1 Criteria for control comparison

Closed loop requirements typically include specifications on load disturbance attenuation, robustness to process uncertainty, measurement noise and set-point tracking. Load disturbance attenuation and robustness are primary concerns in process control and will therefore be in focus here when comparing the different tuning methods. The set-point response can be handled separately, see e.g. Åström and Hägglund (2005), and the effect of noise will only be discussed briefly in the end of this paper.

Minimization of the Integrated Absolute Error (IAE)

$$IAE = \int_0^\infty |e(t)|dt, \tag{3}$$

will define optimal control performance in this paper, where $e(t)$ is the control error due to a unit step load disturbance, $d(t)$, on the process input.

Robustness to process uncertainty can be captured by the sensitivity functions

$$S(i\omega) = \frac{1}{1 + G_l(i\omega)}, \quad T(i\omega) = \frac{G_l(i\omega)}{1 + G_l(i\omega)}, \tag{4}$$

where $G_l(s) = P(s)C(s)$ is the loop transfer function with process $P(s)$ and controller $C(s)$. We will use

$$|S(i\omega)| \leq M_s, \quad |T(i\omega)| \leq M_t, \quad \forall \omega \in \mathbb{R} \tag{5}$$

to constrain IAE optimization, and

$$M_{st} = \max(|S(i\omega)|, |T(i\omega)|), \quad \forall \omega \in \mathbb{R} \tag{6}$$

to provide a robustness measure of the closed loop system. M_{st} will vary depending on process model and tuning method. Reasonable robustness is given for M_{st} ranging between 1.2-2.0.

2.2 Modeling

Since the modeling time should be as short as possible, it is reasonable to believe we only have time for quick

experiments that provides limited process knowledge. We will therefore assume that our models are of low order, either a First Order Time-Delayed (FOTD) system

$$P_m(s) = \frac{K_p}{sT+1}e^{-sL}, \tag{7}$$

or a Second Order Time-Delayed (SOTD) model

$$P_m(s) = \frac{K_p}{(sT_1+1)(sT_2+1)}e^{-sL}, \tag{8}$$

with the special case $T_1 = T_2$. Processes can be characterized based on the normalized time delay $\tau = L/(L+T)$ (FOTD) or $\tau = L/(L+T_1+T_2)$ (SOTD), ranging from 0 to 1. A process is *lag-dominated* if τ is small, *delay-dominated* if τ is large, and *balanced* if τ is around 0.5.

A common way to determine K_p, L and T in (7) is based on an open loop *step response* of the process. K_p is the steady state gain. The apparent time delay L is the t-coordinate of the intersection of the steepest tangent with the time axis, and $L+T$ is the time when the step response has reached 63% of its steady state value. We call this method the *63%-rule*.

Another way to determine either an FOTD or SOTD model is through reduction of a higher order process model with the so called *half-rule*, see Skogestad (2003).

A *relay test* is made in closed loop where the control signal switches amplitude whenever the process output crosses a certain hysteresis threshold. This method is less sensitive to disturbances than the step test and keeps the process closer to its set-point during the modeling experiment. However, it typically only gives information about one frequency point in the process spectrum and it is seldom used for deriving models like (7) and (8).

2.3 Tuning methods

We have chosen to compare our own software-based tuning method with three commonly used tuning rules: Lambda tuning; SIMC; and AMIGO.

Lambda tuning Lambda tuning is today widely adopted in the process industry, see e.g. Sell (1995). Modeling is typically based on measured step responses and the *63%-rule* is used to obtain an FOTD model. The desired closed-loop time constant T_{cl} is used as a tuning parameter, for which we have used the classic choice $T_{cl} = T$ in this paper even though there are better recommendations for delay-dominated processes. Lambda tuning does not refer to any specific robustness, but here we have chosen to compare it to IAE optimal controllers with $M_s = M_t = 1.4$.

SIMC Skogestad (2003) introduced modifications of the Lambda tuning method called SIMC, that improves performance especially for lag-dominant processes. An FOTD (PI) or SOTD model (PID) is obtained by model reduction using the *half-rule*. SIMC is closely related to Lambda tuning, but uses the desired closed-loop time constant $T_{cl} = L$, which typically gives a sensitivity close to $M_{st} = 1.6$.

A modified method for PI control, here called SIMC+, was presented in Skogestad and Grimholt (2012) to improve performance for delay-dominated systems. For PI control we therefore use SIMC+ and for PID control we use the original SIMC rule.

AMIGO The AMIGO method, Hägglund and Åström (2004), was obtained by applying constrained optimization to a large test batch of process models and then use parameter fitting to find the tuning rules. The parameters of an FOTD model are determined by the *63%-rule*. The controller is tuned for a robustness of $M_s = M_t = 1.4$..

SWORD Our own SoftWare-based Optimal Robust Design (SWORD) of PI and PID controllers was first introduced in Garpinger and Hägglund (2008). Using a linear process model of any order and any robustness constraints on the sensitivity and complementary sensitivity functions, one can find the IAE-optimal controller. Here, we will choose $M_s = M_t$ for simplicity. The user can also specify a first (PI) or second order (PID) measurement noise filter before the optimization. This can then be used to set an upper limit for the control signal activity due to measurement noise, as shown in Garpinger (2009).

3. COMPARISON OF THE TUNING METHODS

3.1 Approach

It is reasonable to believe that the four tuning methods would be used together with the *63%-rule* in practice since the step response test is the most common modeling experiment in industry. SIMC and SWORD will also be compared when used on perfect process models, which means that SWORD will use an exact model of the process while SIMC will use models derived with the *half-rule* from the exact model. Given the need for modeling speed, however, it is unlikely that one would have access to an accurate model in every process case.

The four tuning methods will be compared with respect to IAE and M_{st} for the batch of processes common in process industry, that was presented in Hägglund and Åström (2004) and used to derive the AMIGO rules. The integrating processes in the batch are left out of our study since the Lambda tuning method does not handle such systems.

For each process in the batch, we have derived one *63%-rule* FOTD model to use with all tuning methods, as well as *half-rule* FOTD and SOTD models to also use with the SIMC method. PI and PID controllers were derived based on these models, after which M_{st} and IAE were derived with respect to the nominal process. The IAE-values were compared with the PI and PID controllers giving minimal $IAE = IAE_{opt}$ with $M_s = M_t = 1.4$ for Lambda tuning, AMIGO, SWORD and $M_s = M_t = 1.6$ for the SIMC-methods. The measure $100 \cdot IAE/IAE_{opt}$ was used to compare performance to the optimal (100%).

3.2 Comparison

The results from the comparison of PI controllers are collected in Fig. 1. The variation in both closed loop robustness and performance is large for the Lambda method, even if we disregard delay-dominant processes. On the other hand, it seems quite easy to predict both of them if

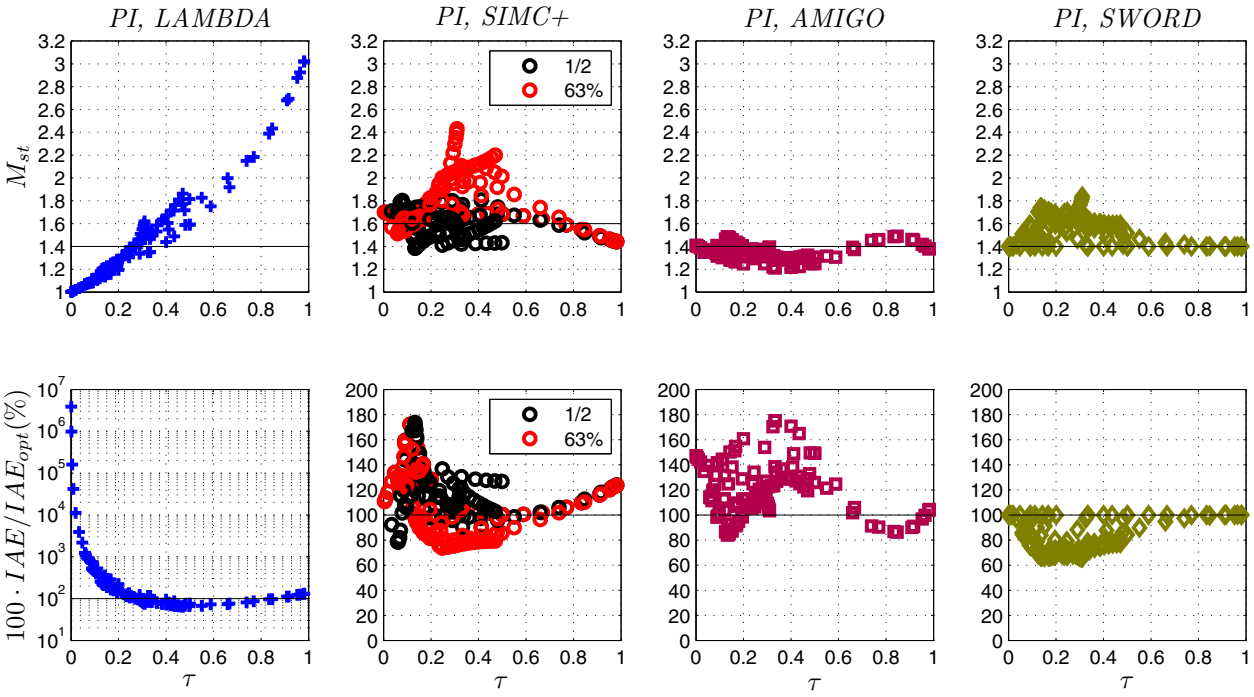

Fig. 1. Comparison of the four different tuning methods for PI control. The upper plots compare robustness with respect to the nominal process. The lower plots compare nominal closed loop performance to optimal performance given a robustness associated with the specific methods. 1/2 denotes controllers derived from *half-rule* models and 63% denotes controllers given by *63%-rule* models. Notice the log-scaled performance plot for the lambda method.

the normalized time delay, τ, is known. If SIMC+ is used together with the *half-rule* (1/2), the robustness will vary roughly between 1.4 and 1.8. Assuming use of *63%-rule* (63%) models instead, the robustness will vary between 1.45 and 2.45, resulting in poor robustness for quite a few processes. The variation in the performance of the SIMC+ method, on the other hand, does not depend that much on the modeling method. Even though the AMIGO method does not need as advanced models as SIMC+, the robustness varies less, between $1.2 \leq M_{st} \leq 1.5$ with performance on par with SIMC+. On the other hand, AMIGO does not come with a tuning parameter like SIMC+, Lambda tuning and SWORD, which means that one can not trade robustness for better performance and vice versa. Using SWORD with a perfect process model gives controllers that are exactly as good as the optimal controllers. However, if *63%-rule* models are used, the robustness will instead vary between 1.4 and 1.85. Notice that there is a clear correlation between loss in robustness and gain in performance.

The results from the PID controller comparison are collected in Fig. 2. Lambda tuning results in poor performance for $\tau < 0.3$ and poor robustness for $\tau > 0.5$. Since SIMC needs an SOTD model to work, we have only used the *half-rule* for the comparison. The spread in both robustness and performance is on par or better than AMIGO, but the need for a good model is still very limiting for this method. For most processes, the robustness of the AMIGO method is within ± 0.2 from the design values $M_s = M_t = 1.4$. Performance is good for $\tau > 0.3$, but almost as widespread as the Lambda method for $\tau \leq 0.3$. SWORD is obviously in need of a different modeling method than the *63%-rule*.

3.3 Visions for better tuning methods

The comparison shows that there is a great deal of variation in both robustness and performance for all four tuning methods. PI control can be improved considerably and it is easy to understand why people in industry hesitate to use PID control. Lambda tuning is intuitive and easy to use, but varies too much in quality. The SIMC methods and SWORD needs too accurate models to work properly and, while AMIGO is the best out of the four tuning methods it still lacks a tuning variable. Clearly, there is room for an improved tuning method.

A properly working SWORD method, with M_{st} close to the design values and almost optimal IAE, would have great benefits. One could use $M_s = M_t$ as a tuning variable and get much better control performance than the other methods given the same maximum value of M_{st}. The biggest challenge is to find a fast, robust and simple modeling tool that provides good enough models for the tuning method to work. Step response modeling seems to limit the four tuning methods and we will therefore investigate possibilities to use relay modeling instead. The aim is to handle tuning with robustness constraints from $M_s = M_t = 1.4$ to 1.8 and provide guidelines for autotuning of PI and PID controllers.

4. MODEL QUALITY

Ideally, we would like a process model that preserves closed loop robustness as well as performance. For simplicity, we will focus on robustness in this article and hope that good performance follows. We would thus like our models to be as accurate as possible around the frequency for which M_{st} is given with optimal control.

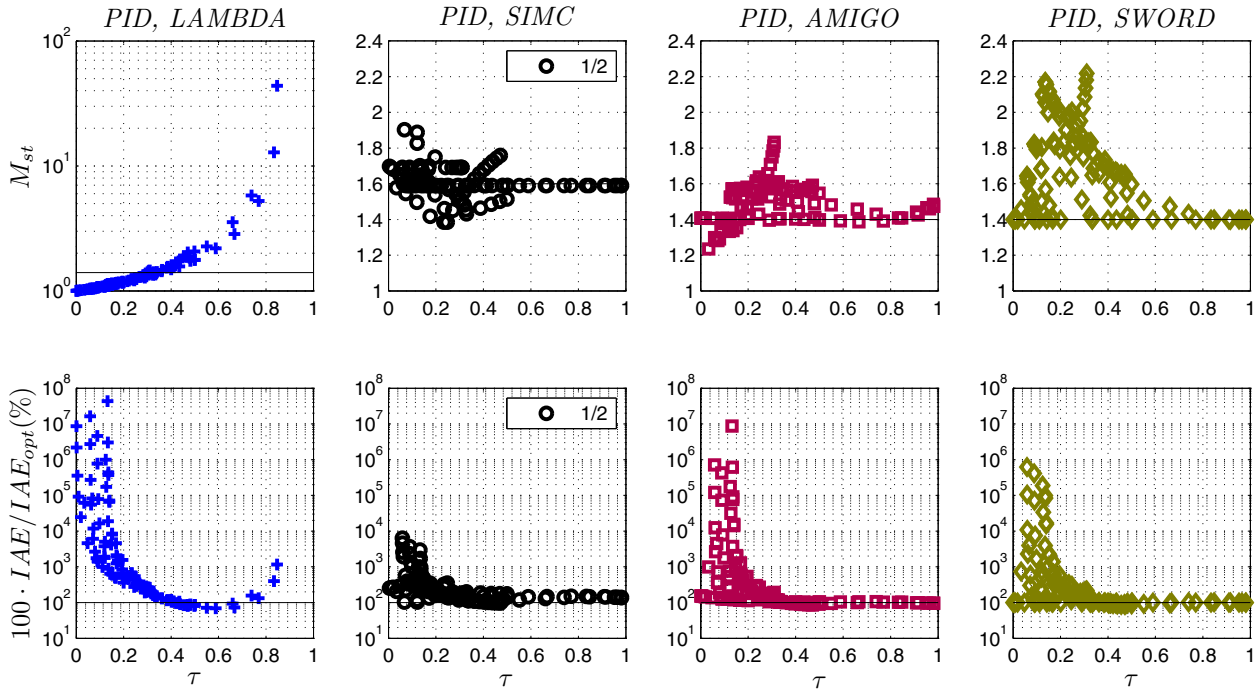

Fig. 2. Comparison of the four different tuning methods for PID control. The upper plots compare robustness with respect to the nominal process. The lower plots compare nominal closed loop performance to optimal performance given a robustness associated with the specific methods. 1/2 denotes controllers derived from *half-rule* models. Notice that several plots have log-scales.

Assume that our relay test can give us process knowledge around a single phase angle, $\phi°$, of the process, which should it be? Say that we derive FOTD models (7) and SOTD models (8), with $T_1 = T_2$, using exact process information about the static process gain, K_p, and around the phase ϕ. The static gain is only used to simplify the modeling and we would have preferred if the model was based only on information around ϕ. To see how important the static gain information is, we have also investigated models with a 10% static gain error, $P_m(0) = 1.1K_p$, and found little to no difference in the results. Therefore, the rest of the study will assume perfect knowledge about the static gain, K_p. Such FOTD and SOTD models were derived for phase angles $\phi = -105, -110, ..., -250, -255°$ on a representative subset of the process batch and SWORD was used to obtain IAE-optimal PI and PID controllers for each model with the design values $M_s = M_t = 1.4$. The closed loop robustness M_{st} was calculated for each relay-based model and the intervals of phase angles for which $1.35 \leq M_{st} \leq 1.45$, were noted. Figure 3 shows these intervals for PI and PID control. For PI control, only FOTD models were used and for PID control SOTD models, with $T_1 = T_2$, were used for all processes except for the FOTD processes. The red circles in the plots show the largest phase angle, within the range of investigated ϕ, that satisfies the given robustness interval, while the blue crosses indicate the least phase angle. All process models within this interval will thus also satisfy the robustness interval. For PI control, this means that all process models based on phase angles between at least $-105°$ and $-130°$ will give accurate closed loop robustness. For PID control, the dependence is more complex, but prior knowledge of τ can help.

Given the information from the plots, we want process knowledge somewhere around the phase angles

$$\phi(\tau) = -125°, \quad \tau \in [0, 1] \tag{9}$$

for PI control and

$$\phi(\tau) = \min(135\tau - 235, -125)°, \quad \tau \in [0, 1] \tag{10}$$

for PID control. These functions are plotted as green, dash-dotted, lines in Fig. 3. The reason why the functions are closer to the lower boundary (crosses) than the upper (circles) is because we want our tuning method to work for M_s- and M_t-values larger than 1.4. Such closed loop systems will typically have greater bandwidth and should thus use lower values of ϕ. In the next Section, we will show that these two choices of functions are reasonable.

5. RESULTS

Equation (9) was used to determine relay FOTD models for the whole process batch in the same way modeling was carried out in Section 4. PI controllers with $M_s = M_t = 1.4, 1.6$ and 1.8 were then determined through SWORD and compared with IAE-optimal PI controllers for the same robustness values. The results are plotted in Fig. 4 and show that the choice of the phase angle ϕ is almost perfect for PI control with $M_s = M_t = 1.6$. The robustness varies between $1.58 \leq M_{st} \leq 1.62$ and the performance is within 10% higher than the optimum. For $M_s = M_t = 1.4$ the performance variation is the same, but M_{st} varies between $1.4 \leq M_{st} \leq 1.45$. The design choice of $M_s = M_t = 1.8$, will also have reasonable variations with $1.72 \leq M_{st} \leq 1.8$ and IAE less than 30% worse than optimum. Notice that for $M_s = M_t = 1.4$ we have slightly more aggressive controllers than optimum, while for $M_s = M_t = 1.8$ we are more conservative.

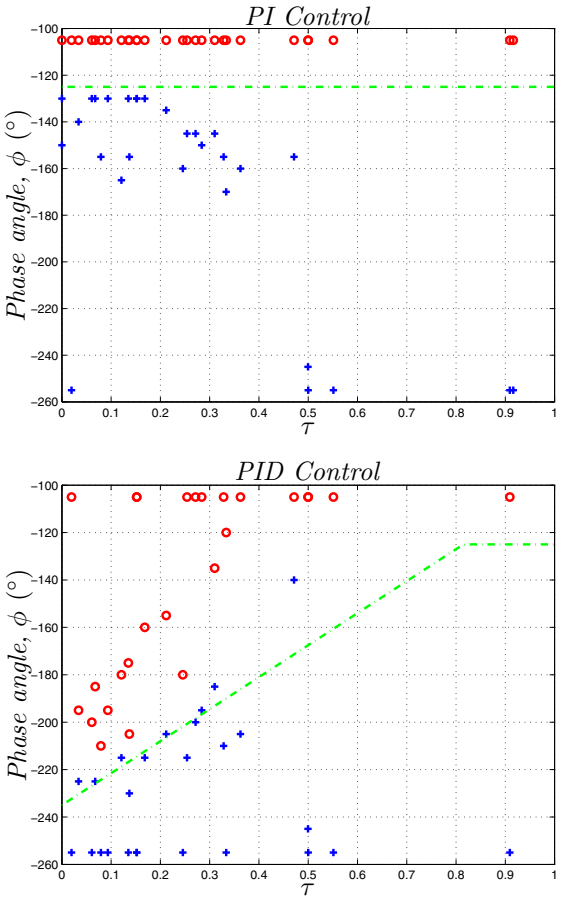

Fig. 3. The plots (PI upper, PID lower) show phase angle intervals between the blue crosses (lower boundary) and red circles (upper boundary) for which the phase angle models need to be accurate to preserve M_{st}. The green dash-dotted lines show reasonable phase angle functions $\phi(\tau)$.

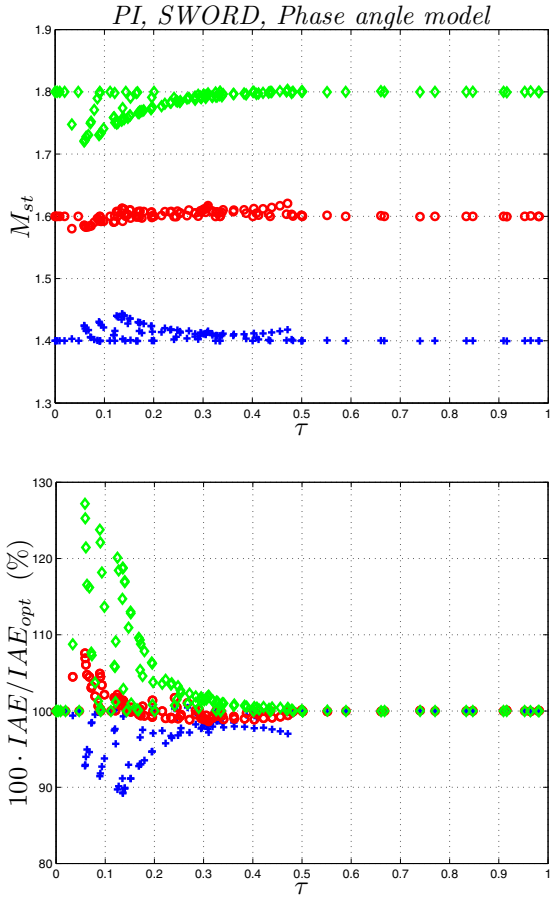

Fig. 4. Results in terms of robustness (upper plot) and performance (lower plot) when using SWORD to design PI controllers for the process batch with three design choices, $M_s = M_t = 1.4$ (blue crosses), 1.6 (red circles), 1.8 (green diamonds) on phase angle models derived using process knowledge given by (9).

PID control was handled in the same way as PI control, but with equation (10) and SOTD models ($T_1 = T_2$) for all processes except the FOTD processes. The results are shown in Fig. 5. For the design choice $M_s = M_t = 1.4$ robustness varies between $1.37 \leq M_{st} \leq 1.53$ and IAE between $90 - 165\%$ of the optimal. The corresponding values for $M_s = M_t = 1.6$ are $1.57 \leq M_{st} \leq 1.80$ and $80 - 165\%$, and for $M_s = M_t = 1.8$ they are $1.76 \leq M_{st} \leq 2.04$ and $75 - 180\%$. The robustness variation is thus almost the same in all three cases while performance variation is greater for higher values of M_s and M_t. Thus, unlike PI control, both robustness and performance deteriorates at the same time. Even so, the robustness is kept within the boundaries for decent robustness $1.2 \leq M_{st} \leq 2.0$ for all cases except one.

6. CONCLUSIONS

The comparison of the four tuning methods showed some severe shortcomings. For PI control, closed loop robustness and performance varies a lot, especially for the lambda method. The SIMC and SWORD methods need accurate models to work well and the AMIGO method lacks a tuning parameter. Furthermore, none of the methods give satisfactory PID control since the performance varies too much.

The biggest benefit of finding a method with less robustness variation is that an increase in M_s and M_t will still guarantee the same M_{st} as the other methods and at the same time improve the performance. Accuracy in performance will of course add further to this. We have focused our study on a software-based tuning method because it can easily adapt itself directly to the process when trying to find the optimal controller. Finding a good tuning rule is hard because it needs to describe every possible case, which is a difficult task especially for PID control. With the optimization software one can also use robustness as a tuning variable. Improving the robustness will thus give worse performance and vice versa, which makes it possible to trade one for the other directly and still guarantee good enough robustness. It is thus our belief that a robust software optimization tool is the future for PI and PID tuning, the question is just how it needs to be built to work properly.

Even if a really good PID software tool is available, it is imperative that it is combined with a fast modeling method that provides good enough models. In this paper, we have shown that the amount of process knowledge needed for both less robustness and performance variation is quite modest. An FOTD model (7) accurate around the phase $\phi = -125°$, with decent static gain knowledge,

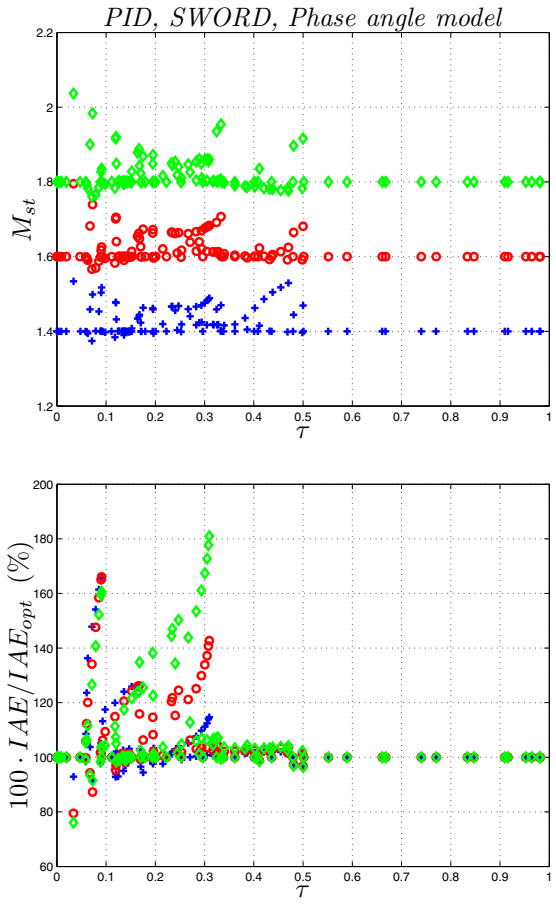

Fig. 5. Results in terms of robustness (upper plot) and performance (lower plot) when using SWORD to design PID controllers for the process batch with three design choices, $M_s = M_t = 1.4$ (blue crosses), 1.6 (red circles), 1.8 (green diamonds) on phase angle models derived using process knowledge given by (10).

is enough to provide PI control very close to optimum when used together with SWORD tuning on the whole process batch. PID controller tuning is more complex since SOTD models (8) are needed and because the necessary process knowledge depends on the normalized time delay, τ. Adding a noise filter after the process modeling will also alter τ, thus posing even greater demands on model accuracy. Finding a tuning method that works for both PI and PID control will also present a challenge since the suggested phase angles are different for the two choices.

We have suggested use of relay-based modeling even though there is little research done on relay methods for transfer function modeling. Work by Friman and Waller (1997) as well as Soltesz and Hägglund (2011), however, suggest that it should be possible to concentrate the relay tests around the suggested phase angles by use of alternative strategies. One important advantage of the relay test to other more advanced modeling methods is that it is already implemented in many commercial control systems and thus readily used.

The main purpose of this article has been to show the potential for future tuning methods rather than to present a method ready to use. SWORD is our choice of design tool, but the ideas can be used together with any other software-based tuning method. It may even provide guidelines for making better tuning rules for those who wish to continue on that track. No matter the method, however, we think that the key to develop a really good tuning method is to combine both modeling and design in the research and find balance in model accuracy and tuning speed.

7. ACKNOWLEDGEMENTS

This work was partly funded by the Swedish Foundation for Strategic Research through the PICLU center. The authors are members of the LCCC Linnaeus Center and the ELLIIT Excellence Center at Lund University.

REFERENCES

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

Friman, M. and Waller, K.V. (1997). A two-channel relay for autotuning. *Industrial and Engineering Chemistry Research*, 36(7), 2662–2671.

Garpinger, O. (2009). Design of robust PID controllers with constrained control signal activity. Licentiate Thesis ISRN 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 *Proc. 17th IFAC World Congress, 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.

Leva, A. and Schiavo, F. (2005). On the role of the process model in model-based autotuning. In *16th IFAC World Congress*. Praha, Czech Republic.

O'Dwyer, A. (2009). *Handbook of PI and PID Controller Tuning Rules*. Imperial College Press, 57 Shelton Street, Covent Garden, London WC2H 9HE, 3rd edition.

Sell, N.J. (ed.) (1995). *Process Control Fundamentals for the Pulp & Paper Industry*. Tappi Press, Technology Park, Atlanta, GA.

Skogestad, S. (2003). Simple analytic rules for model reduction and PID controller tuning. *Journal of Process Control*, 13(4), 291–309.

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 Millenium*, chapter 5, 147–175. Springer-Verlag, London.

Soltesz, K. and Hägglund, T. (2011). Extending the relay feedback experiment. In *18th IFAC World Congress*, 13173–13178. Milano, Italy.