Journal of Process Control 71 (2018) 130–138

Contents lists available at ScienceDirect

Journal of Process Control

journal homepage: www.elsevier.com/locate/jprocont

Optimization of fixed-order controllers using exact gradients

Chriss Grimholt, Sigurd Skogestad*

Department of Chemical Engineering, Norwegian University of Science and Technology (NTNU), Trondheim, Norway

ARTICLE INFO

Article history:
Received 9 July 2015
Received in revised form 9 August 2018
Accepted 5 September 2018

Keywords:
PID control
Controller design
Controller tuning
Optimization
Exact gradients
IAE
Ms

ABSTRACT

Finding good controller settings that satisfy complex design criteria is not trivial. This is also the case for simple fixed-order controllers including the three parameter PID controller. To be rigorous, we formulate the design problem into an optimization problem. However, the algorithm may fail to converge to the optimal solution because of inaccuracies in the estimation of the gradients needed for this optimization. In this paper we derive exact gradients for the problem of optimizing performance (IAE for input and output disturbances) with constraints on robustness (M_S, M_T). Exact gradients give increased accuracy and better convergence properties than numerical gradients, including forward finite differences. The approach may be easily extended to other control objectives and other fixed-order controllers, including Smith Predictor and PIDF controllers.

1. Introduction

The background for this paper was our efforts to find optimal proportional-integral-derivative (PID) controllers for first order plus time delay (FOPTD) processes. The PID controller may be expressed as

$$K_{\text{PID}}(s;p) = k_p + k_i/s + k_d s \tag{1}$$

where k_p, k_i, and k_d are the proportional, integral, and derivative gain, respectively. The objective was to minimize for the simple feedback system in Fig. 1, the integrated absolute error (IAE) (time domain)

$$\min_K \quad \text{IAE} = \int_0^\infty |e(t)|\,\mathrm{d}t, \tag{2}$$

for given disturbances subject to achieving a given frequency domain robustness M_{ST}

$$M_{ST} = \max\{M_S, M_T\} \le M^{ub}. \tag{3}$$

where M_S and M_T are the peaks of the sensitivity and complementary sensitivity functions, respectively. In general, this is a non-convex optimization problem which we initially solved using standard optimization software in MATLAB. We used gradient-free optimization like the Nelder–Mead Simplex method [1], similar to the work of Garpinger and Hägglund [2], and used SIMC settings [3] as initial values for the PID parameters. However, the gradient-free method was slow and unreliable for our purpose of generating trade-off curves between IAE and M_{ST}, which involves repeated optimizations with changing M_{ST}-values [4].

We achieved significant speedup by switching to gradient-based methods (`fmincon` in MATLAB), where the gradients of the cost function (J) and the constraints (M_{ST}) with respect to the controller parameters were found numerically using finite differences. Our experience with this approach were fairly good, but quite frequently, for example for small values of M_{ST}, it did not converge to a solution. Surprisingly, this also occurred even though the initial guess was close to the optimum. It turned out that the main problem was not the non-convexity of the problem or the possibility for local minima, but rather inaccuracies in the estimation of the gradients when using finite-differences.

This led us to derive exact (analytical) expressions for the gradients of IAE and $M_{ST} \le M^{ub}$. This approach is based on the chain rule, that is, first we derive the gradient of IAE with respect to the control error $e(t)$, then the gradient of $e(t)$ with respect to the controller $K(s)$, and then the gradient of $K(s)$ with respect to the controllers parameters p. Some of the gradients are derived in the Laplace domain, but they are evaluated in the time domain, based on a delayed state space realization.

The derivation of these gradients is the main part of the paper (Section 4). Our experience with exact gradients has been very good, and we achieve convergence for a much wider range of conditions in terms of constraints (M_{ST}) and models, see Section 4.4. In addition, the approach can easily be extended to other fixed-

* Corresponding author.
E-mail address: sigurd.skogestad@ntnu.no (S. Skogestad).

https://doi.org/10.1016/j.jprocont.2018.09.001
0959-1524/

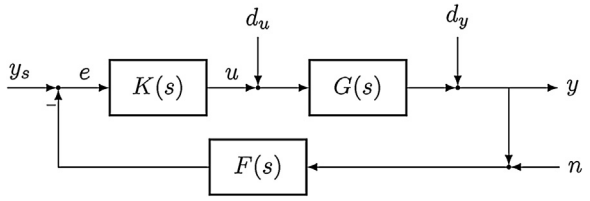

Fig. 1. Block diagram of the closed loop system. In this paper we assume $y_s = 0$, $n = 0$, and $F(s) = 1$. $K(s)$ is the feedback controller, $G(s)$ is the process, and $F(s)$ is the measurement filter (part of the controller).

order controller (e.g., proportional-integral-derivative-filter (PIDF) controller, Smith Predictor), and to other process models. This is discussed in Section 5. Note that the gradient of IAE is not calculated using direct sensitivity results [5], but calculated through Laplace transforms. This greatly simplifies the derivation and the numerical results are good.

A preliminary version of this results was presented at the PSE2015/ESCAPE25 conference [6]. The main contribution compared to the conference paper is to provide a derivation of the gradients and focus on general fixed-order controllers instead of just PID.

2. Previous related work on optimal PID tuning

The simple three-parameter PID controller is the most common controller in the process industry. However, finding good tuning parameter by trial and error is generally difficult and also time consuming. Alternatively, parameters may be found for by use of tuning rules (e.g. Ziegler and Nichols [7], and the SIMC-rules of Skogestad [3]). Nevertheless, when the complexity of the design increases, it is beneficial to switch to optimization-based design. This approach can handle complex process models, non-standard controller parametrizations, and special requirements on controller performance and robustness.

The first systematic contribution on PID tuning is the famous Ziegler-Nichols ZN tuning rule published in 1942 under the title *Optimum Setting for Automatic-Controllers*. However, the "optimum" PID settings were mainly derived from visual inspection of the closed-loop response, aiming at disturbance attenuation with a quarter decay ratio, which results in a quite oscillatory and aggressive response for most process control application.

Hall [8] proposed finding optimal controller settings by minimizing the integrated squared error (ISE = $\int e^2\,\mathrm{d}t$). The ISE criteria is generally selected because it has nice analytical properties. By minimizing ISE, Hazebroek and Van der Waerden [9] analyzed the ZN tuning rule, and proposed an improved rule. The authors noted that minimizing the ISE criterion could give rise to large fluctuations in the manipulated variable, and that the allowable parameters must be restricted for these processes. The analytical treatment of the ISE-optimization was further developed in the influential book by Newton et al. [10].

The first appearance of a "modern" optimization formulation, which includes a performance vs. robustness trade-off, similar to the one used in this paper, is found in Balchen [11]. Balchen minimizes the integrated absolute error (IAE = $\int |e|\,\mathrm{d}t$) while limiting the peak of the sensitivity function (M_S). The constraint on M_S may be adjusted to ensure a given relative dampening or stability margin. This is similar to the formulation used much later in Kristiansson and Lennartson [12]. In addition, Balchen mentions the direct link between minimizing integrated error (IE = $\int e\,\mathrm{d}t$) and maximizing the integral gain (k_i) for a unit step input disturbance, as given by the relationship IE = $1/k_i$. Though the paper of Balchen is very interesting, it seems to have been largely overlooked by the scientific community.

Schei [13] followed up this idea and derived proportional-integral PI settings by maximizing the integral gain k_i subject to a given bound on the sensitivity function M_S. Åström et al. [14] formulated this optimization problem as a set of algebraic equations which could be efficiently solved. In Panagopoulos et al. [15] the formulation was extended to PID control, and a fast concave-convex optimization algorithm is presented in Hast et al. [16]. However, quantifying performance in terms of IE can lead to oscillatory response, especially for PID controllers, because it does not penalize oscillations. Shinskey [17] argued that the IAE is a better measure of performance, and IAE is now widely adopted in PID design [3,18–23].

3. Problem formulation

3.1. Feedback system

We consider the linear feedback system in Fig. 1, with disturbances entering at both the plant input (d_u) and plant output (d_y). Because the response to setpoints can always be improved by using a two degree-of-freedom (DOF) controller, we do not consider setpoints (y_s) in the initial design of the feedback controller. That is, we assume $y_s = 0$. It is worth noticing that, from a feedback point of view, a disturbance entering at the plant output (d_y) is equivalent to a setpoint change. Measurement noise (n) enters the system at the measured output (y). This system can be represented by four transfer functions, nicknamed "the gang of four",

$$S(s) = \frac{1}{1 + G(s)K(s)}, \quad T(s) = 1 - S(s),$$
$$GS(s) = G(s)S(s), \qquad KS(s) = K(s)S(s).$$

Their effect on the control error and plant input is,

$$-e = y - y_s = S(s)\,d_y + GS(s)\,d_u - T(s)\,n, \tag{4}$$

$$-u = KS(s)\,d_y + T(s)\,d_u + KS(s)\,n. \tag{5}$$

Although we could use any fixed-order controller, we consider in this paper mostly the parallel (linear in the parameter) PID controller,

$$K_{\mathrm{PID}}(s;p) = k_p + k_i/s + k_d s, \tag{6}$$

$$p = \begin{pmatrix} k_p & k_i & k_d \end{pmatrix}^T, \tag{7}$$

where k_p, k_i, and k_d is the proportional, integral, and derivative gain, respectively. The controller can equivalently be written in the form

$$K_{\mathrm{PID}}(s;p) = k_c\left(1 + \frac{1}{\tau_i s} + \tau_d s\right), \tag{8}$$

where $k_c = k_p$, $\tau_i = k_c/k_i$, and $\tau_d = k_d/k_c$ is the proportional gain, integral time, and derivative time, respectively. Note that for $\tau_i < 4\tau_d$, the parallel PID controllers has complex zeros. We have observed that this can result in several peaks or plateaux for the magnitude of sensitivity function in the frequency domain $|S(j\omega)|$. As shown later, this becomes important when adding robustness specifications on the frequency behaviour.

3.2. Performance

We quantify controller performance in terms of the integrated absolute error (IAE),

$$\mathrm{IAE}(p) = \int_0^\infty |e(t;p)|\,\mathrm{d}t, \tag{9}$$

when the system is subject to step disturbances. We include both input (d_u) and output (d_y) disturbances and choose the weighted cost function

$$J(p) = 0.5\left(\varphi_{dy}\text{IAE}_{dy}(p) + \varphi_{du}\text{IAE}_{du}(p)\right) \tag{10}$$

where φ_{dy} and φ_{du} are normalization factors. It is necessary to normalize the resulting IAE_{du} and IAE_{dy}, to be able to compare the two terms in the cost function (10), and it is ultimately up to the user to decide which normalization method is most appropriate. Numerically the exact value of these variables are not very important, and in this paper we have, similar to previous work [24,4], selected the normalisation factors to be the inverse of the optimal IAE values for reference controllers (e.g. PI, PID) tuned for a step change on the input (IAE_{du°) and output (IAE_{dy°), respectively.

$$\left(\varphi_{du} = \frac{1}{\text{IAE}_{du^\circ}} \quad \text{and} \quad \varphi_{dy} = \frac{1}{\text{IAE}_{dy^\circ}}.\right)$$

This normalisation is similar to the one used in Shinskey [17]. To ensure robust reference controllers, they are required to have $M_S = M_T = 1.59$.[1] Note that two different reference controllers are used to obtain the IAE° values, whereas a single controller $K(s;p)$ is used to find $\text{IAE}_{dy}(p)$ and $\text{IAE}_{du}(p)$, when minimizing the cost function $J(p)$ in (10).

3.3. Robustness

In this paper, we have chosen to quantify robustness in terms of the largest sensitivity peak, $M_{ST} = \max\{M_S, M_T\}$ [2], where

$$M_S = \max_\omega |S(j\omega)| = \|S(j\omega)\|_\infty,$$
$$M_T = \max_\omega |T(j\omega)| = \|T(j\omega)\|_\infty,$$

and $\|\cdot\|_\infty$ is the H_∞ norm (maximum peak as a function of frequency).

For stable process, M_S is usually larger than M_T. In the Nyquist plot, M_S is the inverse of the closest distance to the critical point $(-1, 0)$ for the loop transfer function $L(s) = G(s)K(s)$. For robustness, a small M_S-value is desired, and generally M_S should not exceed 2. A reasonable M_S-value is about 1.6, and notice that $M_S < 1.6$ guarantees the following good gain and phase margins: GM > 2.67 and PM > 36.4° [25].

From our experience, using the sensitivity peak as a single constraint $|S(j\omega)| \le M^{ub}$ for all ω can lead to poor convergence. This is because the optimal controller can have several peaks of equal magnitude at different frequencies (see Fig. 3), and the optimizer may jump between peaks during iterations. Each peak has a different gradient with respect to the PID parameters. To avoid this problem, instead of using a single constraint, we use multiple constraints obtained by gridding the frequency response,

$$|S(j\omega)| \le M^{ub}, \quad \text{for all} \quad \omega \quad \text{in} \quad \Omega, \tag{11}$$

where $\Omega = [\omega_1, \omega_2, \ldots, \omega_n]$ is a finite set of frequency points. This gives one inequality constraint for each grid frequency. In addition to handling multiple peaks, this approximation also improves convergence for *infeasible* initial controllers because more information is supplied to the optimizer. On the downside, the approximation results in a somewhat reduced accuracy and an increased computational load. However, we found that the benefit of improved convergence makes up for this.

[1] For those that are curious about the origin of this specific value $M_S = 1.59$, it is the resulting M_S value for a Simple Internal Model Control (SIMC) tuned PI controller with $\tau_c = \theta$ on FOPTD process with $\tau \le 8\theta$.

3.4. Summary of the optimization problem

In summary, the optimization problem can be stated as follows,

$$\underset{p}{\text{minimize}} \quad J(p) = 0.5\left(\varphi_{dy} \ \text{IAE}_{dy}(p) + \varphi_{du} \ \text{IAE}_{du}(p)\right) \tag{12}$$

$$\text{subject to} \quad c_S(p) = |S(j\omega;p)| - M_S^{ub} \le 0 \quad \text{for all} \quad \omega \quad \text{in} \quad \Omega \tag{13}$$

$$c_T(p) = |T(j\omega;p)| - M_T^{ub} \le 0 \quad \text{for all} \quad \omega \quad \text{in} \quad \Omega, \tag{14}$$

where M_S^{ub} and M_T^{ub} are the upper bound on $|S(j\omega)|$ and $|T(j\omega)|$, respectively. Typically, we select $M^{ub} = M_S^{ub} = M_T^{ub}$. From experience, selecting Ω as 10^4 logarithmically spaced frequency points with a frequency range from $0.01/\theta$ to $100/\theta$, where θ is the effective time delay of the process. If there is a trade-off between performance and robustness, at least one robustness constraints in (13) or (14) will be active.

A simple pseudo code for the cost function is shown in Algorithm 1 in Appendix A. It is important that the cost function also returns the error responses, such that they can be reused for the gradient. The pseudo code for the constraint function is shown in Algorithm 2 in Appendix A. The intention behind the pseudo codes is to give an overview of the steps involved in the calculations.

4. Gradients

The gradient of a function $f(p)$ with respect to a parameter vector p is defined as

$$\nabla_p f(p) = \begin{pmatrix} \frac{\partial f}{\partial p_1} & \frac{\partial f}{\partial p_2} & \cdots & \frac{\partial f}{\partial p_{n_p}} \end{pmatrix}^T, \tag{15}$$

where n_p is the number of parameters. In this paper, p_i refers to parameter i, and the partial derivative $\partial f/\partial p_i$ is called the sensitivity of f. For simplicity we will use short hand notation $\nabla \equiv \nabla_p$.

4.1. Forward finite differences

The sensitivities can be approximated by forward finite differences (FFD)

$$\frac{\partial f}{\partial p_i} \approx \frac{f(p_i + \Delta p_i) - f(p_i)}{\Delta p_i}, \tag{16}$$

which require $(1 + n_p)$ perturbations. Because we consider both input and output disturbances, this results in at total of $2(1 + n_p)$ time response simulations.

One of the dominating factors affecting the accuracy of the finite difference approximation of the sensitivities is the accuracy of the time response simulation. This can easily be seen in the following example. The computed IAE-value ($\tilde{\text{IAE}}$) we get from integrating (9) can be written into the true IAE and the integration error δ.

$$\tilde{\text{IAE}} = \text{IAE} \pm \delta \tag{17}$$

Using forward finite differences (FFD) (16), and assuming for simplicity only one parameter p, the computed gradient becomes

$$\nabla\tilde{\text{IAE}} = \frac{\text{IAE}_1 \pm \delta_1 - \text{IAE}_2 \pm \delta_2}{\Delta p} + O(\Delta p) = \frac{\Delta \text{IAE}}{\Delta p} + \frac{\Delta \delta}{\Delta p} + O(\Delta p) \tag{18}$$

where Δp is the perturbation in parameter, and $O(\Delta p)$ is the truncation error. The worst-case gradient error becomes

$$E_{\text{FFD}} = \frac{2\delta}{\Delta p} + O(\Delta p). \tag{19}$$

As the perturbation size $\Delta p \to 0$, the truncation error $O(\Delta p) \to 0$. However, the simulation error is magnified as $2\delta/\Delta p \to \infty$. On the other hand, the impact of simulation error can be reduced

by increasing the perturbation size Δp, but then truncation error might became an issue. It this paper we have chosen to use MATLAB's default perturbation size of $\sqrt{\epsilon}$, where ϵ is the machine precession. In order to derive simple expressions which are less prone to errors, we will instead use the gradient expressions derived below.

4.2. Cost function gradient

The gradient of the cost function $J(p)$ can be expressed as

$$\nabla J(p) = 0.5 \left(\varphi_{dy} \quad \nabla \text{IAE}_{dy}(p) + \varphi_{du} \quad \nabla \text{IAE}_{du}(p) \right) \tag{20}$$

The IAE sensitivities are difficult to evaluate analytically, but they can be found in a fairly straightforward manner by expressing them such that the integrals can be evaluated numerically.

By taking advantage of the fixed structure of the problem, we develop general expressions for the gradient. When the parameter sensitivity of the controller $K(s)$ is found, evaluating the gradient is a simple process of combining and evaluating already defined transfer functions. This enables the user to quickly find the gradients for a linear system for any fixed-order controller $K(s)$.

From the definition of the IAE in (9) and assuming that $|e(t)|$ is smooth, that is that $|e(t)|$ and $\text{sign}\{e(t)\} \nabla e(t)$ are continuous, the sensitivity of the IAE can be expressed as (see Appendix B for details)

$$\nabla \text{IAE}_{dy}(p) = \int_0^{t_f} \text{sign}\{e_{dy}(t)\} \nabla e_{dy}(t) \, \mathrm{d}t, \tag{21}$$

$$\nabla \text{IAE}_{du}(p) = \int_0^{t_f} \text{sign}\{e_{dy}(t)\} \nabla e_{dy}(t) \, \mathrm{d}t. \tag{22}$$

Introducing the Laplace transform, we see from (4) that

$$e_{dy}(s) = S(s) d_y \quad \text{for output disturbances and} \tag{23}$$

$$e_{du}(s) = GS(s) d_u \quad \text{for input disturbances.} \tag{24}$$

By using the chain rule [19], we can write the error sensitivities as a function of the parameter sensitivity of the controller $K(s)$ (see Appendix B.4)

$$\nabla e_{dy}(s) = -GS(s) \quad S(s) \quad \nabla K(s) \, d_y \tag{25}$$

$$\textit{and} \quad \nabla e_{du}(s) = -GS(s) \quad GS(s) \quad \nabla K(s) \, d_u, \tag{26}$$

For the PID controller defined in (6), the controller sensitivities $\nabla K(s)$ with respect to the parameters $p = \begin{pmatrix} k_p & k_i & k_d \end{pmatrix}^T$ are

$$\nabla K_{\text{PID}}(s) = \begin{pmatrix} \frac{\partial K_{\text{PID}}}{\partial k_p} & \frac{\partial K_{\text{PID}}}{\partial k_i} & \frac{\partial K_{\text{PID}}}{\partial k_d} \end{pmatrix}^T = \begin{pmatrix} 1 & 1/s & s \end{pmatrix}^T \tag{27}$$

To obtain $\nabla \text{IAE}_{dy}(p)$ and $\nabla \text{IAE}_{du}(p)$ in (21) and (22), we must obtain the inverse transforms of $\nabla e_{dy}(s)$ and $\nabla e_{du}(s)$ in (25) and (26). However, for processes $G(s)$ with time delay (which is the case for our problem), the sensitivity $S(s)$ will have internal delays (delays in the denominator), and there are no analytical solution to the inverse Laplace transforms of $\nabla e_{du(s)}$ and $\nabla e_{dy(s)}$, and they must be evaluated numerically. For example, to evaluate the gradient of IAE_{du} in (22) for PID control when considering a unit step disturbance ($d_u = 1/s$), we first obtain the time response of $\nabla e_{dy(t)}$ by performing an impulse response simulation of a state space realization of the following system,

$$\nabla e_{du(s)} = -GS(s) \quad GS(s) \begin{pmatrix} 1/s & 1/s^2 & 1 \end{pmatrix}^T. \tag{28}$$

Typical numerical results for ∇e_{du} are shown in the lower plot of Fig. 4. The gradient of the IAE is then calculated by evaluating the IAE integral (22) using numerical integration techniques like the trapezoidal method.

In many cases, $|e(t)|$ and $\text{sign}\{e(t)\} \nabla e(t)$ are not continuous on the whole time range. For example, $e(t)$ will have a discrete jump for step setpoint changes. For our assumptions to be valid in such cases, the integration must be split up into subintervals which are continuous. Violating this assumption will result in an inaccuracy in the calculation of the gradient at the time step of the discontinuity. However, by using very small integration steps in the time response simulations, this inaccuracy is negligible. Therefore, the integration has not been split up into subintervals for the case study in Section 4.4.

A simple pseudo code for the gradient calculation is shown in Algorithm 3 in Appendix A. Notice that the error responses from the cost function are reused (shown as inputs).

Because the gradient is evaluated by time domain simulations, the method is limited to processes that gives proper gradient transfer functions. If the gradient transfer functions are not proper, a small filter can be added to the process, controller or the gradient transfer function to make it proper. However, this will introduce small inaccuracies.

To obtain the gradient of the cost function (20), $2n_p$ simulations are needed for evaluating (25) and (26), and 2 simulations are needed to evaluate the error (23) and (24), resulting in a total of $2(1+n_p)$ simulations. This is the same number of simulations needed for the one-sided forward finite differences approximation in (16), but the accuracy is much better.

4.3. Constraint gradients

The gradient of the robustness constraints, $c_S(p)$ and $c_T(p)$, can expressed as (see Appendix C)

$$\nabla c_S(j\omega; p) = \nabla |S(j\omega)| = \frac{1}{|S(j\omega)|} \Re\{S^*(j\omega) \nabla S(j\omega)\} \quad \text{for all} \quad \omega \quad \text{in} \quad \Omega \tag{29}$$

$$\nabla c_T(j\omega; p) = \nabla |T(j\omega)| = \frac{1}{|T(j\omega)|} \Re\{T^*(j\omega) \nabla T(j\omega)\} \quad \text{for all} \quad \omega \quad \text{in} \quad \Omega \tag{30}$$

The asterisk (*) is used to indicate the complex conjugate. By using the chain rule, we can rewrite the constraint gradient as an explicit function of $\nabla K(j\omega)$ (as we did with the cost function gradient),

$$\nabla S(j\omega) = -GS(j\omega) \quad S(j\omega) \quad \nabla K(j\omega) \tag{31}$$

$$\nabla T(j\omega) = \nabla \left(1 - S(j\omega)\right) = -\nabla S(j\omega) \tag{32}$$

The gradients of the constraints is then evaluated at each frequency point ω in Ω. The constraint gradients are algebraic functions of p, and can also be found using automatic differentiation. However, due to the fixed structure of the gradients they are more easily obtained by multiplication of the known transfer function and $\nabla K(j\omega)$. A pseudo code for the gradient calculation is show in Algorithm 4 in Appendix A.

4.4. Case study

The exact gradients were implemented and computed for PID control of the design problem given in (12)–(14), with a FOPTD process

$$G(s) = \frac{e^{-s}}{s+1}. \tag{33}$$

To make the system proper, we also added a fist-order filter to the controller

$$K(s) = K_{\text{PID}}(s) \frac{1}{\tau_f s + 1}, \quad \text{with} \quad \tau_f = 0.001. \tag{34}$$

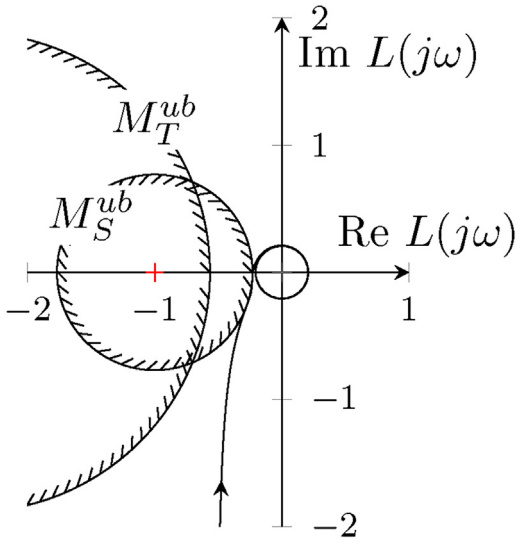

Fig. 2. Nyquist plot of $L(j\omega)$ for the optimal controller for the problem given in (33)–(36).

To ensure robust controllers, the sensitivity peaks are required to be small with

$$M_S^{ub} = M_T^{ub} = 1.3. \tag{35}$$

This problem has the following normalization factors

$$\text{IAE}_{dy}^{\circ} = 1.56, \qquad \text{IAE}_{du}^{\circ} = 1.42, \tag{36}$$

and the optimization algorithm was started with controller parameters

$$p_0 = (0.2 \quad 0.02 \quad 0.3)^T. \tag{37}$$

This case study was selected because it is a problem that looks simple, but can be surprisingly hard to solve. One reason for this is that the optimal solution has two equal M_S peaks (Figs. 2 and 3), and can therefore exhibit cyclic behavior between the iterations when using a single $M_{ST} \leq M^{ub}$ constraint. Actually, if it was not for the filter, the optimal problem solution would have an infinite number of equal peaks.

The error response (obtained with a simulation length of 25 time units) and cost function sensitivity was found by fixed step integration (with number of steps $n_{\text{steps}} = 10^4$). The problem was solved using MATLAB's `fmincon` with the active set algorithm. The optimal error response with sensitivities are shown in Fig. 4. For comparison, the problem was also solved with approximated gradients using forward finite differences with MATLAB's default perturbation size of $\sqrt{\epsilon}$ (which is supposed to give a good balance between truncation error and round-off error).

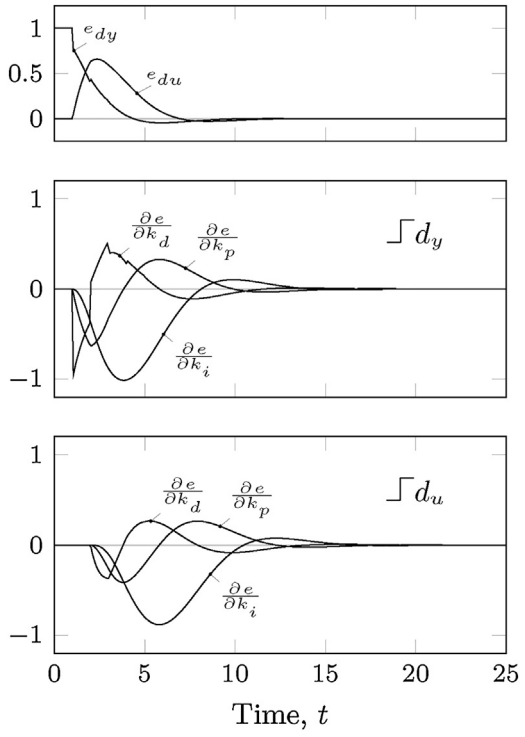

Fig. 4. Optimal error response and corresponding error sensitivities for the problem given in (33)–(36).

Table 1
Comparison of optimal solutions when using different combinations of gradients for the problem given in (33)–(36).

Gradient type		Cost function	Optimal parameters			Number of iterations
Cost-function	Constraints	$J(p^*)$	k_p	k_i	k_d	
exact	exact	**2.0598**	0.5227	0.5327	0.2172	13
fin.dif.	exact	2.1400	0.5204	0.4852	0.1812	16
exact	fin.dif.	**2.0598**	0.5227	0.5327	0.2172	13
fin.dif.	fin.dif.	2.9274	0.3018	0.3644	0.2312	11

As seen in Table 1, the exact gradients performed better than the approximated finite differences. The biggest improvement comes from using the exact cost function gradients. The large difference in controller parameters between the two approaches indicates that the optimum is relatively flat, as the small numerical inaccuracies in the simulations and small truncation errors in the finite difference

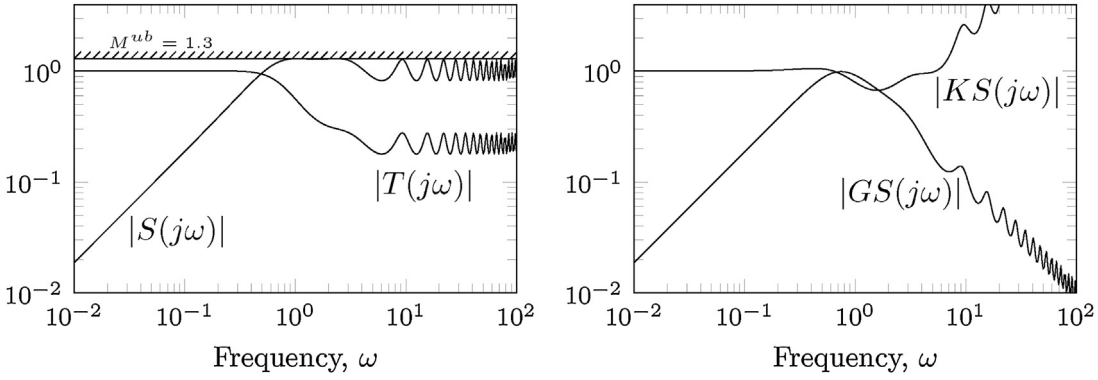

Fig. 3. Frequency response of the "the gang of four" ($S(j\omega)$, $T(j\omega)$, $GS(j\omega)$, and $KS(j\omega)$) for the optimal controller for the problem given in (33)–(36).

approximation gives gradients that falsely satisfy the tolerances of the algorithm.

As mentioned previously, one dominating factor affecting the accuracy of the finite difference approximation, is the accuracy of the time reprocess simulation. To illustrate this, the same design problem was solved with different numbers of time steps during the integration. Even with $n_{\text{steps}} = 10^5$, the forward finite differences failed to converge to the optimum. On the other hand, the exact gradient could still converge to the local optimum with $n_{\text{steps}} = 500$.

The exact gradient converged for most initial guesses that gave a stable closed-loop system. On the other hand, forward finite differences failed to find the optimum even when starting very close to the minimum, for example

$$p_0 = \begin{pmatrix} 1.001p_1^* & p_2^* & p_3^* \end{pmatrix}^T.$$

With central finite differences, the accuracy may be increased, but this requires $2(1+2n_p)$ step simulations.

5. Extensions to other fixed order controllers

As stated previously, the method is easily extended to other linear fixed-order controllers, as it only requires changing the expression for the parameter sensitivity $\nabla K(s)$ in (27). Here we will give the $\nabla K(s)$ for three other controllers: the serial PID controller, the PIDF controller, and the Smith predictor.

5.1. Serial PID controller

For the serial PID controller,

$$K_{\text{PID}}^{\text{serial}}(s) = k_c \frac{(\tau_i s + 1)(\tau_d s + 1)}{\tau_i s}, \tag{38}$$

the controller sensitivities are (elements in $\nabla K_{\text{PID}}^{\text{serial}}(s)$)

$$\frac{\partial K_{\text{PID}}^{\text{serial}}(s)}{\partial k_c} = \frac{(\tau_i s + 1)(\tau_d s + 1)}{\tau_i s}, \tag{39}$$

$$\frac{\partial K_{\text{PID}}^{\text{serial}}(s)}{\partial \tau_i} = -k_c \frac{(\tau_d s + 1)}{\tau_i^2 s}, \tag{40}$$

$$\frac{\partial K_{\text{PID}}^{\text{serial}}(s)}{\partial \tau_d} = k_c \frac{(\tau_i s + 1)}{\tau_i}. \tag{41}$$

Note the serial controller sensitivities must be updated during iterations, whereas they are constant for the parallel PID controller $K_{\text{PID}}(s)$, see (27).

5.2. PIDF controller

For a PIDF controller, here defined as

$$K_{\text{PIDF}}(s) = K_{\text{PID}}(s)F(s), \tag{42}$$

where the parameters in the filter $F(s)$ are extra degrees of freedom, the derivative can be expressed in terms of product rule,

$$\nabla K_{PIDF}(s) = F(s)\nabla K_{\text{PID}}(s) + K_{\text{PID}}(s)\nabla F(s), \tag{43}$$

For example, with a first-order filter

$$F(s) = \frac{1}{\tau_f s + 1} \quad \text{we get} \quad \nabla F(s) = \begin{pmatrix} 0 & 0 & 0 & -\dfrac{s}{(\tau_f s + 1)^2} \end{pmatrix}^T. \tag{44}$$

Note that this PIDF controller with four tuning parameters is the most general second-order controller $K(s)$, given that the controller must have integral action.

5.3. Smith predictor

The Smith predictor (SP) uses an internal model of the process with time delay, $G(s)$. Let $G^\circ(s)$ be the process model without delay and assume that a PID controller is used for the delay process, $K_{\text{PID}}(s) = k_p + k_i/s + k_d s$. The Smith predictor becomes,

$$K_{\text{SP}}(s) = \frac{K_{\text{PID}}(s)}{1 + K_{\text{PID}}(s)(G^\circ(s) - G(s))}, \tag{45}$$

and the gradients are

$$\nabla K_{\text{SP}}(s) = \partial K_{SP}(s)\partial K_{\text{PID}}(s)\nabla K_{\text{PID}} = \frac{\nabla K_{\text{PID}}}{[1 + K_{\text{PID}}(s)(G^\circ(s) - G(s))]^2} \tag{46}$$

where $\nabla K_{\text{PID}} = \begin{pmatrix} 1 & 1/s & s \end{pmatrix}^T$ is the gradient of the internal PID controller.

6. Discussion

6.1. Input usage and noise filtering

We have not included a bound on input usage, e.g. by considering the maximum peak of $|KS(j\omega)|$ which is $||KS||_\infty$. For simple cases, input usage can be reduced by simply making the process more robust by lowering M_{ST} [24]. That is, reducing the controller gain will make the controller more robust *and* reduce input usage. However, for unstable processes this will not hold, and increasing the controller gain can actually make the controller *more* robust and increase input usage. For such processes, an additional constraint should be added to limit input usage to a desired levels.

For the parallel PID controller in (8), the gain and thus noise amplification goes to infinity at high frequencies. To avoid excessive input movements, a measurement filter may be added as we did in (34). We assume in this paper that noise amplification is addressed separately after the initial controller design. Nevertheless, our design formulation could easily be extended to handle noise filtering more explicitly by using an appropriate constraint, e.g. $||KS||_\infty$. This is treated in a separate paper by Soltesz et al. [26]. If the measurement filter actually enhances *noise-less* performance, the note that we are no longer talking about a PID controller, but a PIDF controller with four adjustable parameters.

6.2. Circle constraint

Circle constraints [19] provides an alternative to constraining $S(j\omega)$ and $T(j\omega)$. The idea is to ensure that the loop function $L(j\omega) = G(s)K(s)$ is outside given robustness circles in the Nyquist plot. For a given circle with centre C and radius R, the robustness criteria can be expressed as

$$|C - L(j\omega)|^2 \geq R^2 \quad \text{for all} \quad \omega \quad \text{in} \, \Omega. \tag{47}$$

For $S(s)$, the centre and radius will be

$$C = -1 \quad \text{and} \quad R = 1/M^{ub}. \tag{48}$$

For $T(s)$, the centre and radius will be

$$C = -\frac{(M^{ub})^2}{(M^{ub})^2 - 1} \quad \text{and} \quad R = \frac{M^{ub}}{(M^{ub})^2 - 1}. \tag{49}$$

Written in standard form, the constraint becomes,

$$c_L(p) = R^2 - |C - L(j\omega)|^2 \leq 0 \quad \text{for all} \quad \omega \quad \text{in} \quad \Omega, \tag{50}$$

with centre C and radius R for the corresponding M^{ub} circle, respectively. The corresponding gradient is

$$\nabla c_L(j\omega) = 2\Re\left\{(C - L(j\omega))^* \nabla L(j\omega)\right\} \quad \text{for all} \quad \omega \quad \text{in} \quad \Omega. \tag{51}$$

The main computational cost is the evaluation of the transfer functions. It may be seen that the circle constraint (51) has an advantage, because you only need to evaluate $L(j\omega)$, whereas, both $S(j\omega)$ and $T(j\omega)$ must be evaluated for the constraints (29) and (30). However, by using the relation $S+T=1$ we can rewrite e.g. $T(j\omega)$ in terms of $S(j\omega)$ by $T=1-S$. Thus, we only need to evaluate $S(j\omega)$ to calculate both c_S and c_T. Because the mathematical operations are relatively cheap, the two gradients types are almost equivalent.

6.3. Direct sensitivity calculations

Our method is closely related to the more general direct sensitivity method, as defined in Biegler [5], used for optimal control problems, which we applied in Jahanshahi et al. [27]. However, processes with time delay we cannot use conventional ODE sensitivity results as the resulting feedback system has internal delay. To apply the direct sensitivity method to our problem, we set up the *delayed differential equations* (DDE) symbolically, found the parameter sensitivities of the states, and integrated using a integrator for delayed differential equations (e.g. `dde23`). Although the direct sensitivity approach works well on this problem, a lot of work went into setting up the DDE equations and sensitivities, and integrating them in Matlab. By taking advantage the fixed structure of the problem and the control system toolbox in MATLAB, it is less work to use the approach presented in this paper.

7. Conclusion

In this paper we have successfully applied the exact gradients for a typical performance (IAE) with constrained robustness M_{ST} optimization problem. Compared to gradients approximated by forward finite difference, the exact gradients improved the convergence to the true optimal. By taking advantage of the fixed structure of the problem, the exact gradients were presented in such a way that they can easily be implemented and extended to other fixed-order controllers. When the parameter sensitivity of the controller $K(s)$ is found, evaluating the gradient is just a simple process of combining and evaluating already defined transfer functions. This enables the user to quickly find the gradients for a linear system for any fixed order controller. The MATLAB code for the optimization problem is available at the home page of Sigurd Skogestad.

Appendix A. Pseudo code for the calculation of gradients

Algorithm 1. COSTFUNCTION(p)

$t \longleftarrow$ uniform distributed time points
$e_{dy}(t) \longleftarrow$ get step response of $S(s;p)$ with time steps t
$e_{du}(t) \longleftarrow$ get step response of $GS(s;p)$ with time steps t
calculate IAE for e_{dy} and e_{du} using numerical integration
$J \longleftarrow$ calculate cost function using (10)
return $(J, e_{dy}(t), e_{du}(t))$

Algorithm 2. CONSTRAINTFUNCTION(p)

$\omega \longleftarrow$ logarithmically spaced frequency points in Ω
$c_S \longleftarrow |S(j\omega)| - M_S^{ub}$
$c_T \longleftarrow |T(j\omega)| - M_T^{ub}$
$c \longleftarrow$ stack c_S and c_T into one vector
return (c)

Algorithm 3. GRADCOSTFUNCTION(p, $e_{dy}(t)$, $e_{du}(t)$)

$t \longleftarrow$ uniform distributed time points
for $i \longleftarrow 1$ **to** number of parameters
do
$\nabla e_{dy}(t) \longleftarrow$ get time response of $\nabla e_{dy}(s)$ from (25) with time steps t
$\nabla e_{du}(t) \longleftarrow$ get time response of $\nabla e_{du}(s)$ from (26) with time steps t
$\nabla \text{IAE}_{dy} \longleftarrow$ numerical integration of (21) using e_{dy} and $\nabla e_{dy}(t)$
$\nabla \text{IAE}_{du} \longleftarrow$ numerical integration of (22) using e_{du} and $\nabla e_{du}(t)$
$\nabla J \longleftarrow$ calculate from (20)
$\nabla J \longleftarrow$ stack the cost function sensitivities into one vector
return (∇J)

Algorithm 4. GRADCONSTRAINTFUNCTION(p)

$\omega \longleftarrow$ logarithmically spaced frequency points
for $i \longleftarrow 1$ **to** number of parameters
do
$\nabla c_S \longleftarrow$ evaluate (29) for frequencies ω
$\nabla c_T \longleftarrow$ evaluate (30) for frequencies ω
$\nabla c \longleftarrow$ combine ∇c_S and $\nabla c_T p_i$ into a matrix
return (∇c)

Appendix B. Derivation of the exact sensitivities of the cost function

B.1 Sensitivity of the absolute value

Lemma 1. *Let $g(t;p)$, abbreviated as $g(t)$, be a function that depends on time t and parameters p. The partial derivative of the absolute value of $g(t)$ with respects to the parameter p is then*

$$\frac{\partial}{\partial p}|g(t)| = \text{sign}\{g(t)\}\frac{\partial}{\partial p}\left(g(t)\right).$$

Proof. The absolute value can be written as the multiplication of the function $g(t)$ and its sign,

$$|g(t)| = \text{sign}\left\{g(t)\right\}\, g(t),$$

where the sign function has the following values

$$\text{sign}\{g(t)\} = \begin{cases} -1 & \text{if } g(t) < 0, \\ 1 & \text{if } g(t) > 0. \end{cases}$$

Using the product rule we get,

$$\frac{\partial}{\partial p}\left(\text{sign}\left\{g(t)\right\}\, g(t)\right) = \text{sign}\left\{g(t)\right\}\, \frac{\partial g(t)}{\partial p} + g(t)\frac{\partial \text{sign}\left\{g(t)\right\}}{\partial p}. \quad (52)$$

The sign function is piecewise constant and differentiable with the derivative equal 0 for all values except $g(t)=0$, where the derivative is not defined. Hence the following conclusion is true,

$$g(t)\frac{\partial \text{sign}\left\{g(t)\right\}}{\partial p} = 0,$$

and the differential of $|g(t)|$ is as stated above. □

B.2 Sensitivity of the integrated absolute value

Theorem 1. *Let $g(t;p)$, abbreviated as $g(t)$, be a function that depends on the time t and the parameter p. The differential of the integrated absolute value of $g(t)$ on the interval from t_α to t_β with respect to the parameter p can be written as*

$$\frac{\mathrm{d}}{\mathrm{d}p}\left(\int_{t_\alpha}^{t_\beta}|g(t)|\,\mathrm{d}t\right) = \int_{t_\alpha}^{t_\beta}\text{sign}\left\{g(t)\right\}\left(\frac{\partial g(t)}{\partial p}\right)\mathrm{d}t \quad (53)$$

Proof. If $g(t)$ and its partial derivative $\partial g(t)/\partial p$ are continuous w.r.t. t and p on the intervals $\left[t_\alpha, t_\beta\right]$ and $\left[p_\alpha, p_\beta\right]$, and the integration limits are constant, then using Leibniz's rule, we can write the integral

$$\frac{\mathrm{d}}{\mathrm{d}p}\left(\int_{t_\alpha}^{t_\beta}|g(t)|\,\mathrm{d}t\right) = \int_{t_\alpha}^{t_\beta}\frac{\partial|g(t)|}{\partial p}\,\mathrm{d}t.$$

Then using Lemma 1, we obtain the stated expression. □

B.3 Obtaining the sensitivities from Laplace

Theorem 2. *Let $g(t;p)$ be a linear function that depends on time t and the parameter p, and $G(s;p)$ its corresponding Laplace transform (abbreviated $g(t)$ and $G(s)$). Then the partial derivative of $g(t)$ can be expressed in terms of the inverse Laplace transform of $G(s)$,*

$$\frac{\partial g(t)}{\partial p} = \mathcal{L}^{-1}\left\{\frac{\partial G(s)}{\partial p}\right\}.$$

Proof. Differentiating the definition of the Laplace transform with respect to the parameter p we get,

$$\frac{\partial G(s)}{\partial p} = \frac{\partial}{\partial p}\int_0^\infty e^{-st} g(t)\,\mathrm{d}t.$$

Assuming that $g(t)$ and $\partial g(t)/\partial p$ is continuous on the integration interval, Leibniz's rule gives

$$\frac{\partial G(s)}{\partial p} = \int_0^\infty e^{-st}\frac{\partial g(t)}{\partial p}\,\mathrm{d}t,$$

which is equivalent to

$$\frac{\partial G(s)}{\partial p} = \mathcal{L}\left\{\frac{\partial g(t)}{\partial p}\right\}.$$

Taking the inverse Laplace on both sides gives the stated expression. □

B.4 Sensitivity of S(s) *and* GS(s)

We have $S=(1+L)^{-1}$ where $L=GK$. Using the chain-rule on the definition of S, and dropping the argument s for simplicity,

$$\frac{\partial S}{\partial p_i} = \frac{\partial S}{\partial L}\frac{\partial L}{\partial K}\frac{\partial K}{\partial p_i}. \tag{54}$$

We have

$$\frac{\partial S}{\partial L} = \frac{\partial}{\partial L}(1+L)^{-1} = -(1+L)^{-2} = -S^2. \tag{55}$$

and $\partial L/\partial K = G$. Thus,

$$\frac{\partial S}{\partial p_i} = -S^2 G\frac{\partial K}{\partial p_i}. \tag{56}$$

Similarly,

$$\frac{\partial GS}{\partial p_i} = G\frac{\partial S}{\partial p_i} = -(GS)^2\frac{\partial K}{\partial p_i}. \tag{57}$$

Appendix C. Derivation of the exact sensitivities for the constraints

C.1 Sensitivity of G(jω)G*(jω;p) *for a specific frequency point*

Lemma 2. *Let $G(j\omega;p)$ be a general transfer function, and $G^*(j\omega;p)$ its complex conjugate (abbreviated $G(jv)$ and $G^*(j\omega)$). Then the differential of the product of the two with respect to the parameter p is*

$$\frac{\partial}{\partial p}\left(G(j\omega)\,G^*(j\omega)\right) = 2\Re\left(G^*(j\omega)\frac{\partial G(j\omega)}{\partial p}\right),$$

where $\Re\{\cdot\}$ is the real part of the argument.

Proof. Write the transfer function out as their the complex numbers

$$G(j\omega) = x + jy, \tag{58}$$

$$G^*(j\omega) = x - jy. \tag{59}$$

The product rule gives

$$(x-jy)\frac{\partial(x+jy)}{\partial p} + (x+jy)\frac{\partial(x-jy)}{\partial p},$$

and becomes

$$2\left(x\frac{\partial x}{\partial p} + y\frac{\partial y}{\partial p}\right) = 2\Re\left(G^*(j\omega)\frac{\partial G(j\omega)}{\partial p}\right).$$

□

C.2 Sensitivity of |G(jω);p| *for a specific frequency point*

Theorem 3. *Let $G(j\omega;p)$, abbreviated as $G(j\omega)$ be a general transfer function evaluated at the frequency $j\omega$. The partial derivative of the magnitude of $G(j\omega)$ with respect to the parameter p is then*

$$\frac{\partial|G(j\omega)|}{\partial p} = \frac{1}{|G(j\omega)|}\Re\left\{G^*(j\omega)\frac{\partial G(j\omega)}{\partial p}\right\}.$$

Proof. The derivative can be expressed in terms of the squared magnitude,

$$\frac{\partial|G(j\omega)|}{\partial p} = \frac{\partial}{\partial p}\sqrt{|G(j\omega)|^2} = \frac{1}{2|G(j\omega)|}\frac{\partial|G(j\omega)|^2}{\partial p}.$$

The squared magnitude can be written as the transfer function multiplied by its complex conjugate $G^*(j\omega)$

$$|G(j\omega)|^2 = G(j\omega)\,G^*(j\omega).$$

Using the product rule presented in Lemma 2, we get

$$\frac{\partial|G(j\omega)|^2}{\partial p} = 2\Re\left\{G^*(j\omega)\frac{\partial G(j\omega)}{\partial p}\right\}.$$

□

Some readers might have wondered why the sensitivity of the transfer function is not written using the chain-rule with

$$\frac{\partial|G(j\omega)|^2}{\partial G(j\omega)} = \frac{\partial(G(j\omega)\,G(j\omega)^*)}{\partial G(j\omega)} = G^*(j\omega) + G(j\omega)\frac{\partial G^*(j\omega)}{\partial G(j\omega)}.$$

This is because the derivative $\partial G^*(j\omega)/\partial G(j\omega)$ is non-analytic and do not exist anywhere [28].

References

[1] J.A. Nelder, R. Mead, A simplex method for function minimization, Comput. J. 7 (4) (1965) 308–313.
[2] O. Garpinger, T. Hägglund, A software tool for robust PID design, in: Proc. 17th IFAC World Congress, Seoul, Korea, 2008.
[3] S. Skogestad, Simple analytic rules for model reduction and PID controller tuning, J. Process Control 13 (4) (2003) 291–309.
[4] C. Grimholt, S. Skogestad, Optimal PID-control on first order plus time delay systems & verification of the SIMC rules, 10th IFAC International Symposium on Dynamics and Control of Process Systems (2013).
[5] L.T. Biegler, Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes, vol. 10, SIAM, 2010.
[6] C. Grimholt, S. Skogestad, Improved optimization-based design of PID controllers using exact gradients, 12th International Symposium on Process Systems Engineering and 25th European Symposium on Computer Aided Process Engineering, volume 37 (2015) 1751–1757.
[7] J.G. Ziegler, N.B. Nichols, Optimum settings for automatic controllers, Trans. ASME 64 (11) (1942).
[8] A.C. Hall, The Analysis and Synthesis of Linear Servomechanisms, Technology Press Massachusetts Institute of Technology, 1943.
[9] P. Hazebroek, B.L. Van der Waerden, The optimum tuning of regulators, Trans. ASME 72 (1950) 317–322.

[10] G.C. Newton, L.A. Gould, J.F. Kaiser, Analytical Design of Linear Feedback Controls, John Wiley & Sons, New York, NY, 1957.
[11] J.G. Balchen, A performance index for feedback control systems based on the Fourier transform of the control deviation, Acta Polytech. Scand. Math. Comput. Mach. Ser. 247 (1) (1958) 3–19.
[12] B. Kristiansson, B. Lennartson, Evaluation and simple tuning of PID controllers with high-frequency robustness, J. Process Control 16 (2) (2006) 91–102.
[13] T.S. Schei, Automatic tuning of PID controllers based on transfer function estimation, Automatica 30 (12) (1994) 1983–1989.
[14] K.J. Åström, H. Panagopoulos, T. Hägglund, Design of PI controllers based on non-convex optimization, Automatica 34 (5) (1998) 585–601.
[15] H. Panagopoulos, K.J. Åström, T. Hägglund, Design of PID controllers based on constrained optimisation, IEE P Contr. Theor. Appl. 149 (1) (2002) 32–40.
[16] M. Hast, K.J. Aström, B. Bernhardsson, S. Boyd, PID design by convex-concave optimization, Proceedings of European Control Conference (2013) 4460–4465.
[17] F.G. Shinskey, How good are our controllers in absolute performance and robustness? Meas. Control 23 (4) (1990) 114–121.
[18] A. Ingimundarson, T. Hägglund, Performance comparison between PID and dead-time compensating controllers, J. Process Control 12 (8) (2002) 887–895.
[19] K.J. Åström, T. Hägglund, Advanced PID Control, ISA – The Instrumentation, Systems, and Automation Society, 2006.
[20] M. Huba, Performance measures, performance limits and optimal PI control for the IPDT plant, J. Process Control 23 (4) (2013) 500–515.
[21] O. Garpinger, T. Hägglund, K.J. Åström, Performance and robustness trade-offs in PID control, J. Process Control 24 (5) (2014) 568–577.
[22] V. Alfaro, R. Vilanova, V. Méndez, J. Lafuente, Performance/robustness tradeoff analysis of PI/PID servo and regulatory control systems, in: 2010 IEEE International Conference on Industrial Technology (ICIT), IEEE, 2010, pp. 111–116.
[23] S. Alcántara, R. Vilanova, C. Pedret, PID control in terms of robustness/performance and servo/regulator trade-offs: a unifying approach to balanced autotuning, J. Process Control 23 (4) (2013) 527–542.
[24] C. Grimholt, S. Skogestad, Optimal PI-control and verification of the SIMC tuning rule, in: IFAC Conference on Advances in PID control (PID'12), The International Federation of Automatic Control, March, 2012.
[25] D.E. Rivera, M. Morari, S. Skogestad, Internal model control. 4. PID controller design, Ind. Eng. Chem. Process Des. Dev. 25 (1986) 252–256.
[26] K. Soltesz, C. Grimholt, S. Skogestad, Simultaneous design of PID controller and measurement filter by optimization, IET Control Theory 11 (2017) 341–348.
[27] E. Jahanshahi, V. de Oliveira, C. Grimholt, S. Skogestad, A comparison between internal model control, optimal PIDF and robust controllers for unstable flow in risers, The International Federation of Automatic Control (2014) 5752–5759.
[28] M.P. Spiegel, Schaum's Outline Series: Theory and Problems of Laplace Transforms, McGraw-Hill Book Company, 1965.