# Optimization of Fixed-Order Controllers Using Exact Gradients

Chriss Grimholt and Sigurd Skogestad

February 13, 2016

**Abstract**

Finding good controller settings that satisfy complex design criteria is not trivial. This is also the case for the simple three parameter PID controller. One strategy is to formulate the design problem into an optimization problem. However, when using gradient-based optimization, the algorithm often fails to converge to the optimal solution. In many cases this is a result of inaccuracies in the estimation of the gradients. In this paper we derive exact gradients for the problem of optimizing performance (IAE) with constraints on robustness ($M_S$, $M_T$). Using the exact gradients, we demonstrate increased accuracy and better convergence than with forward finite differences. The results may be easily extended to other objectives and 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 objective was to minimize the integrated absolute error (IAE) (time domain)

$$\min_K \quad \text{IAE} = \int |e(t)| \, \mathrm{d}t, \tag{1}$$

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

$$M_{ST} = \max\{M_S, M_T\} \le M^{ub}. \tag{2}$$

In this work, we solve this problem repeatedly. That is, for different values of $M^{ub}$ we generate the Pareto optimal trade-off between performance (IAE) and robustness ($M_{ST}$). In general, this is a non-convex optimization problem which we originally solved numerically using standard optimization software in MATLAB.

Initially, we solved the optimization problem using gradient-free optimization like the Nelder-Mead Simplex method (Nelder and Mead, 1965), similar to the work of Garpinger and Hägglund (2008). We used the SIMC settings (Skogestad, 2003) as initial values for the PID parameters. However, the gradient-free method was too slow for our purpose of generating trade-off curves (Grimholt and Skogestad, 2013).

We achieved significant speedup by switching to gradient-based methods (`fmincon`) in MATLAB, where the gradients of the cost function ($J$) and the constraints ($M_{\text{ST}}$) with respect to the controller parameters was found numerically using finite differences. Our experience with this approach were fairly good, but quite frequently, for example for small values of $M_{\text{ST}}$, 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 the exact (analytical) expressions for the gradients of IAE and $M_{\text{ST}} \leq M^{ub}$ which is presented in this paper. We use 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 Lapalace 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 ($M_{\text{ST}}$) and models, see Section 4.3. In addition, the approach can easily be extended to other fixed-order controller (e.g., proportional-integral-derivative-filter (PIDF) controller, Smith Predictor), and to other process models. This is discussed in Section 5. A preliminary version of this paper was presented at the PSE2015/ESCAPE25 conference (Grimholt and Skogestad, 2015).

## 2 Previous work on optimal PID tuning

The simple three parameter PID controller is the most common controller in the process industry. However, finding good parameter values by trial and error is generally difficult and also time consuming. Alternatively, model-based parameters may be found for certain cases by use of tuning rules (e.g. ZN in Ziegler and Nichols (1942), and SIMC in Skogestad (2003)).

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. In this paper, we consider a standard controller design problem where we want to optimize performance while ensuring a required robustness.

Influential work has been done to define and find optimal PID tunings. An early contribution 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 derived manly from visual inspection. Aiming at a quarter decay ratio for the time response, the ZN tuning results in a quite oscillatory and aggressive settings for most process control application.

Hall (1943) proposed finding optimal controller settings by minimizing the integrated squared error ($\text{ISE} = \int e^2 \, \mathrm{d}t$). The ISE criteria is generally selected because it has nice analytical properties. By minimizing the ISE, Hazebroek and Van der Waerden (1950) analyzed the ZN tuning rule, and proposed an improved rule. The authors noted that minimizing the ISE criterion could give rise to violent 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. (1957).

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 (1958). This paper uses an approximation of the integrated absolute error ($\text{IAE} = \int |e| \, \mathrm{d}t$) which it minimized 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 in Kristiansson and Lennartson (2006). In addition, Balchen mentions the direct link between minimizing integrated error ($\text{IE} = \int e \, \mathrm{d}t$) and maximizing the integral gain ($k_i$) for a unit step input disturbance, as given by the relationship $\text{IE} = 1/k_i$. Though the paper of Balchen is highly interesting, it seems to have been largely overlooked by the scientific community.

Schei (1994) 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. (1998) formulated this optimization problem as a set of algebraic equations which could be efficiently solved. In Panagopoulos et al. (2002) the formulation was extended to PID control, and a fast concave-convex optimization algorithm is presented in Hast et al. (2013). However, quantifying performance in term of IE can lead to oscillatory response, especially for PID controllers, because it does not penalize oscillations. Shinskey (1990) argued that the IAE is a better measure of performance, and IAE is now widely adopted in PID design (Ingimundarson and Hägglund, 2002; Åström and Hägglund, 2006; Skogestad, 2003; Huba, 2013;

Garpinger et al., 2014; Alfaro et al., 2010; Alcántara et al., 2013).

## 3 Problem Formulation

### 3.1 Feedback system

In this paper, we consider the linear feedback system in Figure 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 2 degree-of-freedom (DOF) controller, we do not need to consider setpoints ($y_s$) in the initial design of the feedback controller. It is worth noticing that, from a feedback point of view, disturbances entering at the plant output 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*,

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

$$GS(s) = G(s)\,S(s), \qquad KS(s) = K(s)\,S(s).$$

Their effect on the control error and plant input is,

$$-e = y - y_s = S(s)\,d_y + GS(s)\,d_u - T(s)\,n, \tag{3}$$

$$-u = KS(s)\,d_y + T(s)\,d_u + KS(s)\,n. \tag{4}$$

Although we could use any fixed-order controller, we consider in this paper to use the parallel PID controller,

$$K_{\text{PID}}(s;p) = k_p + k_i/s + k_d s = k_c\left(1 + \frac{1}{\tau_i s} + \tau_d s\right), \tag{5}$$

$$\text{where} \quad p = \begin{pmatrix} k_p & k_i & k_d \end{pmatrix}^T, \tag{6}$$

and $k_p = k_c$, $k_i = k_c/\tau_i$, and $k_d = k_c\,\tau_d$ is the proportional, integral, and derivative gain, respectively, and $\tau_i$ and $\tau_d$ are the integral and derivative times. Note that for $\tau_i < 4\tau_d$, the parallel PID controllers have 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

In this paper, we quantify controller performance in terms of the integrated absolute error (IAE),

$$\mathrm{IAE}\,(p) = \int_0^{t_f} |e(t;\,p)|\ \mathrm{d}t, \tag{7}$$

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

$$J(p) = 0.5\left(\varphi_{dy}\ \mathrm{IAE}_{dy}(p) + \varphi_{du}\ \mathrm{IAE}_{du}(p)\right) \tag{8}$$

where $\varphi_{dy}$ and $\varphi_{du}$ are normalization factors. It is necessary to normalize the resulting $\mathrm{IAE}_{du}$ and $\mathrm{IAE}_{dy}$, to be able to compare the two terms in the cost function (8), and it is ultimately up to the user to decide which normalization method is most appropriate. However, in this paper (similar to previous work) we have 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 ($\mathrm{IAE}^{\circ}_{du}$) and output ($\mathrm{IAE}^{\circ}_{dy}$), respectively.

$$\varphi_{du} = \frac{1}{\mathrm{IAE}^{\circ}_{du}} \quad \text{and} \quad \varphi_{dy} = \frac{1}{\mathrm{IAE}^{\circ}_{dy}}.$$

This normalisation is similar to the one used in Shinskey (1990). To ensure robust reference controllers, they are required to have $M_{\mathrm{S}} = M_{\mathrm{T}} = 1.59$ [1]. Note that two different reference controllers are used to obtain the $\mathrm{IAE}^{\circ}$ values, whereas a single controller $K(s;p)$ is used to find $\mathrm{IAE}_{dy}(p)$ and $\mathrm{IAE}_{du}(p)$, when minimizing the cost function $J(p)$ in (8).

## 3.3 Robustness

In this paper, we have chosen to quantify robustness in terms of the largest sensitivity peak, $M_{\mathrm{ST}} = \max\{M_{\mathrm{S}}, M_{\mathrm{T}}\}$ (Garpinger and Hägglund, 2008), where

$$M_{\mathrm{S}} = \max_{\omega} |S(j\omega)| = \|S(j\omega)\|_{\infty},$$
$$M_{\mathrm{T}} = \max_{\omega} |T(j\omega)| = \|T(j\omega)\|_{\infty},$$

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

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

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 typical "good" 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$> 36.4°$ (Rivera et al., 1986).

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

$$|S(j\omega)| \le M^{ub} \qquad \text{for all } \omega \text{ in } \Omega, \tag{9}$$

where $\Omega$ is the set of selected 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 reduced accuracy and increased computational load; However, we found that the benefit of improved convergence makes up for this.

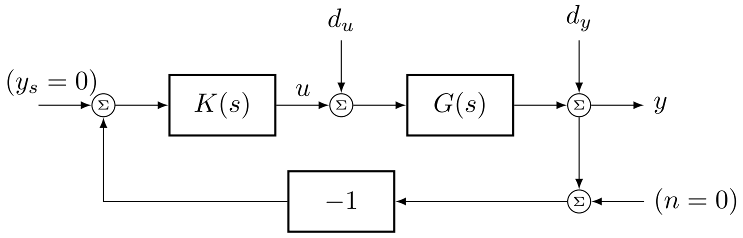

Figure 1: Block diagram of the closed loop system, with controller $K(s)$ and plant $G(s)$.

## 3.4 Summary of the optimization problem

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

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

$$\text{subject to} \quad c_{\text{S}}(p) = \left|S(j\omega;\, p)\right| - M_{\text{S}}^{ub} \leq 0 \quad \text{for all } \omega \text{ in } \Omega \tag{11}$$

$$c_{\text{T}}(p) = \left|T(j\omega;\, p)\right| - M_{\text{T}}^{ub} \leq 0 \quad \text{for all } \omega \text{ in } \Omega, \tag{12}$$

where $M_{\text{S}}^{ub}$ and $M_{\text{T}}^{ub}$ are the upper bound on $\left|S(j\omega)\right|$ and $\left|T(j\omega)\right|$, respectively. Usually we select $M^{ub} = M_{\text{S}}^{ub} = M_{\text{T}}^{ub}$, which is the same as a constraint on $M_{\text{ST}}$. Typically, we use for $\Omega$, $10^4$ logarithmically spaced grid points with a frequency range from $0.01/\theta$ to $100/\theta$, where $\theta$ is the time delay of the process. If there is a trade-off between performance and robustness, at least one robustness constraints in (11) or (12) 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 respects to a parameter vector $p$ is defined as

$$\nabla_p f(p) = \begin{pmatrix} \frac{\partial f}{\partial p_1} & \frac{\partial f}{\partial p_2} & \cdots & \frac{\partial f}{\partial p_{n_p}} \end{pmatrix}^T, \tag{13}$$

where $n_p$ is the number of parameters. In this paper, $p_i$ refers to parameter $i$, and the partial derivative $\frac{\partial f}{\partial p_i}$ is called the sensitivity of $f$. For simplicity we will use short hand notation $\nabla \equiv \nabla_p$. The sensitivities can be approximated by forward finite differences

$$\frac{\partial f}{\partial p_i} \approx \frac{f(p_i + \Delta p_i) - f(p_i)}{\Delta p_i}, \tag{14}$$

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

## 4.1 Cost function gradient

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

$$\nabla J(p) = 0.5 \left( \varphi_{dy}\ \nabla \text{IAE}_{dy}(p) + \varphi_{du}\ \nabla \text{IAE}_{du}(p) \right) \tag{15}$$

The IAE sensitivities are difficult to evaluate symbolically. However, the sensitivities can be found in a fairly straightforward manner by developing them such that the integrals can be evaluated numerically.

By taking advantage of the fixed structure of the problem, we have developed general expressions for the gradient. 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.

From the definition of the IAE in (7) and assuming that $|e(t)|$ and $\text{sign}\left\{e(t)\right\} \nabla e(t)$ is continuous, the sensitivity of IAE can be expressed as (see Appendix B for details)

$$\nabla \text{IAE}_{dy}(p) = \int_0^{t_f} \text{sign}\left\{e_{dy}(t)\right\} \nabla e_{dy}(t)\, \mathrm{d}t, \tag{16}$$

$$\nabla \text{IAE}_{du}(p) = \int_0^{t_f} \text{sign}\left\{e_{dy}(t)\right\} \nabla e_{dy}(t)\, \mathrm{d}t. \tag{17}$$

Introducing the Laplace transform, we see from (3) that

$$e_{dy}(s) = S(s)\, d_y \quad \text{for output disturbances and} \tag{18}$$

$$e_{du}(s) = GS(s)\, d_u \quad \text{for input disturbances.} \tag{19}$$

By using the chain rule, we get the error sensitivities as a function of parameter sensitivity of the controller $K(s)$ (See Appendix B.4)

$$\nabla e_{dy} = -GS(s)\ S(s)\ \nabla K(s)\, d_y \tag{20}$$

$$\text{and} \quad \nabla e_{du} = -GS(s)\ GS(s)\ \nabla K(s)\, d_u, \tag{21}$$

Specifically, for the PID controller defined in (5), the controller parameter sensitivities $\nabla K(s)$ are

$$\nabla K_{\text{PID}}(s) = \begin{pmatrix} 1 & 1/s & s \end{pmatrix}^T \tag{22}$$

For example, to evaluate the gradient of $\text{IAE}_{du}$ in (17) for PID control when considering a unit step disturbance ($d_u = 1/s$), we first obtain the time response of $\nabla e_{d_y}(t)$ by performing an impulse response simulation of a state space realization of the following system,

$$\nabla e_{d_u}(s) = -GS(s)\; GS(s) \begin{pmatrix} 1/s & 1/s^2 & 1 \end{pmatrix}^T . \tag{23}$$

Typical numerical results for $\nabla e_{d_u}$ are shown in the lower plot of Figure 4. The gradient of the IAE is then calculated by evaluating the IAE integral (17) using numerical integration techniques like the trapezoidal method.

In many cases $|e(t)|$ and sign $\{e(t)\}\ \nabla e(t)$ is not continuous on the whole time range. For example, $e(t)$ will have a discrete jump for step setpoint changes. For this assumption to be valid for such a case, the integration must be split up into subintervals which are continuous. Breaking 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 can be reduced to a negligible size. Therefore, the integration has not been split up into subintervals for the case study in Section 4.3.

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 cost function (15), $2n_p$ simulations are needed for evaluating (20) and (21), and 2 simulations are needed evaluate the error (18) and (19) , 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 (14), but the accuracy is much better.

## 4.2 Constraint gradients

The gradient of the robustness constraints, $c_{\text{S}}(p)$ and $c_{\text{T}}(p)$, can expressed as (See Appendix C)

$$\nabla c_{\text{S}}(j\omega;p) = \nabla|S(j\omega)| = \frac{1}{|S(j\omega)|}\Re\left\{S^*(j\omega)\nabla S(j\omega)\right\} \qquad \text{for all } \omega \text{ in } \Omega \tag{24}$$

$$\nabla c_{\text{T}}(j\omega;p) = \nabla|T(j\omega)| = \frac{1}{|T(j\omega)|}\Re\left\{T^*(j\omega)\nabla T(j\omega)\right\} \qquad \text{for all } \omega \text{ in } \Omega \tag{25}$$

| Gradient type | | Cost function | Optimal parameters | | | number of |
|---|---|---|---|---|---|---|
| Cost-function | Constraints | $J(p^\star)$ | $k_p$ | $k_i$ | $k_d$ | iterations |
| 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 |

Table 1: Comparison of optimal solutions when using different combinations of gradients.

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),

$$\nabla S(j\omega) = -GS(j\omega)\ S(j\omega)\ \nabla K(j\omega) \tag{26}$$

$$\nabla T(j\omega) = \nabla\left(1 - S(j\omega)\right) = -\nabla S(j\omega) \tag{27}$$

The gradients of the constraints is then evaluated at each frequency $\omega$ in $\Omega$. A pseudo code for the gradient calculation is show in Algorithm 4 in Appendix A.

## 4.3 Case study

The exact gradients were implemented and computed for the following FOPTD process.

$$G(s) = \frac{e^{-s}}{s+1}, \quad \text{IAE}^\circ_{dy} = 1.56, \quad \text{IAE}^\circ_{du} = 1.42, \tag{28}$$
$$M_{\text{S}}^{ub} = M_{\text{T}}^{ub} = 1.3, \quad \text{and} \quad p_0 = \begin{pmatrix} 0.2 & 0.02 & 0.3 \end{pmatrix}^T.$$

To make the system proper, a first-order filter was added to the controller,

$$K(s) = K_{\text{PID}}(s)\frac{1}{\tau_f s + 1}. \tag{29}$$

The filter time constant was selected to be 1/1000 of the delay. That is, $\tau_f = 0.001$. It will therefore not influence the optimization problem in any significant way, other than maxing it solvable. The error response (simulation length of 25 time units) and cost function sensitivity was found by fixed step integration (with number of steps $n_{\text{steps}} = 10^4$), with the initial point $p_0$. The problem was solved using MATLAB's `fmincon` with the active set algorithm. To

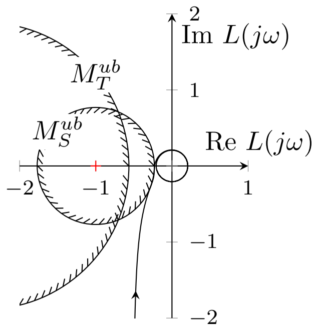

Figure 2: Nyquist plot of $L(j\omega)$ for the optimal controller for the problem given in (28).

compare, forward finite differences was used to find numerical gradients. This problem has two equal $M_s$ peaks at the optimum (Figures 2 and 3), and is a typical example of a problem exhibiting cyclic behaviour when using a single $M_{ST} \leq M^{ub}$ constraint. If it was not for the filter, the problem would have an infinite number of equal peaks. The optimal error response with sensitivities are shown in Figure 4.

As seen in Table 1, the exact gradients performed better than the numerical finite differences. The biggest improvement comes from the exact cost function gradients. This shows that the optimum is relatively flat, and that the approximated cost function gradients using finite difference are not precise enough to find the true local optimum. The same test was performed for 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 (controller parameter error in the second digit). On the other hand, the exact gradient could still converge to the local optimum with as low as $n_{\text{steps}} = 500$ (control parameters error in the fifth digit).

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

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

When using central finite differences, the accuracy was increased. However, this requires $2(1+2n_p)$ step simulations.

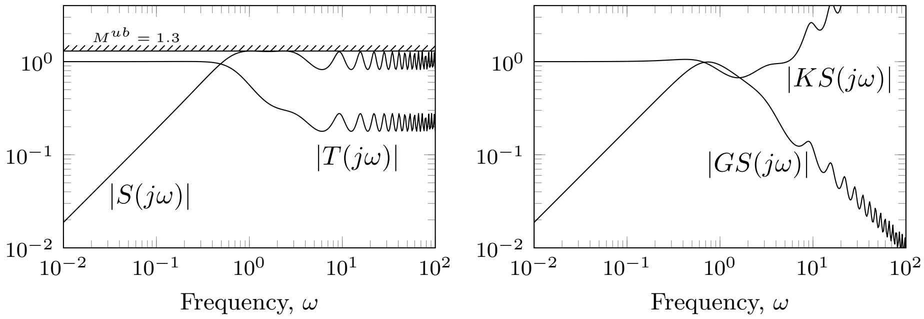

Figure 3: Frequency response of the "the gang of four" ($S(j\omega)$, $T(j\omega)$, $GS(j\omega)$, and $KS(j\omega)$) for optimal controller for the problem given in (28).

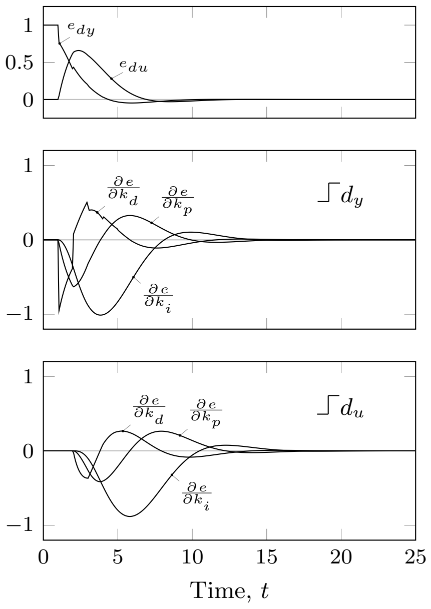

Figure 4: Optimal error response and corresponding error sensitivities for the problem given in (28).

## 5 Extensions to other fixed order controllers

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

$$K_{\text{PID}}^{\text{serial}}(s) = k_c \frac{(\tau_i s + 1)(\tau_d s + 1)}{\tau_i s}, \tag{30}$$

the controllers sensitivity is (elements in $\nabla K_{\text{PID}}^{\text{serial}}(s)$)

$$\frac{\partial K_{\text{PID}}^{\text{serial}}(s)}{\partial k_c} = \frac{(\tau_i s + 1)(\tau_d s + 1)}{\tau_i s}, \tag{31}$$

$$\frac{\partial K_{\text{PID}}^{\text{serial}}(s)}{\partial \tau_i} = -k_c \frac{(\tau_d s + 1)}{\tau_i^2 s}, \quad \text{and} \tag{32}$$

$$\frac{\partial K_{\text{PID}}^{\text{serial}}(s)}{\partial \tau_d} = k_c \frac{(\tau_i s + 1)}{\tau_i}. \tag{33}$$

Note the serial controller sensitivities must be updated during iterations, whereas they are constant for the parallel PID controller $K_{\text{PID}(s)}$, see (22) For a PIDF controller, here defined as

$$K_{\text{PIDF}}(s) = K_{\text{PID}}(s)F(s), \tag{34}$$

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

$$\nabla K_{\text{PIDF}}(s) = F(s)\nabla K_{\text{PID}}(s) + K_{\text{PID}}(s)\nabla F(s), \tag{35}$$

For example, with

$$F(s) = \frac{1}{\tau_f s + 1} \quad \text{we get} \quad \nabla F(s) = \begin{pmatrix} 0 & 0 & 0 & -\frac{s}{\tau_f s + 1} \end{pmatrix}^T. \tag{36}$$

The Smith predictor uses an internal model of the process with time delay, $G(s)$, to predict future errors. Let $G_\circ(s)$ be the process model without delay. Using an internal PI controller $K_{\text{PI}}(s) = k_p + k_i/s$, the Smith predictor becomes,

$$K_{\text{SP}}(s) = \frac{K_{\text{PI}}(s)}{1 + K_{\text{PI}}(s)(G_\circ(s) - G(s))}, \tag{37}$$

and the gradient is

$$\nabla K_{\text{SP}}(s) = \frac{\partial K_{\text{SP}}(s)}{\partial K_{\text{PI}}(s)} \nabla K_{\text{PI}} = \frac{\nabla K_{\text{PI}}}{\left[1 + K_{\text{PI}}(s)(G(s)_\circ - G(s))\right]^2}. \tag{38}$$

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

## 6 Discussion

### 6.1 Input usage

We have not included a bound on input usage, e.g. $\|KS\|_\infty$. For simple cases, input usage can be reduced by simply making the process more robust by lowering $M_{\text{ST}}$ (Grimholt and Skogestad, 2012). That is, reducing the controller gain will make the controller more robust *and* reduce input usage. However, for unstable processes this claim might not hold. 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.

### 6.2 Noise filtering

For an ideal PID controller, the gain and thus noise amplification goes to infinity at high frequencies. To avoid excessive input movements, a measurement filter must be added. However, the measurement filter should be selected such that the performance is not significantly changed. 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. (2014). 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.

### 6.3 Circle constraint

Circle constraints (Åström and Hägglund, 2006) 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

$$\left|C - L(j\omega)\right|^2 \geq R^2 \qquad \text{for all } \omega \text{ in } \Omega. \tag{39}$$

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

$$C = -1 \quad \text{and} \quad R = 1/M^{ub}. \tag{40}$$

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

$$C = -\frac{(M^{ub})^2}{(M^{ub})^2 - 1} \quad \text{and} \quad R = \frac{M^{ub}}{(M^{ub})^2 - 1}. \tag{41}$$

Written in standard form, the constraint becomes,

$$c_{\mathrm{L}}(p) = R^2 - \left|C - L(j\omega)\right|^2 \le 0 \qquad \text{for all } \omega \text{ in } \Omega, \tag{42}$$

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

$$\nabla c_{\mathrm{L}}(j\omega) = 2\Re\left\{\left(C - L(j\omega)\right)^* \, \nabla L(j\omega)\right\} \qquad \text{for all } \omega \text{ in } \Omega. \tag{43}$$

The main computational cost is the evaluation of the transfer functions. It may seen that the circle constraint (43) 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 (24) and (25) . 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 $c_{\mathrm{S}}$ and $c_{\mathrm{T}}$. Because the mathematical operations are relatively cheap, the two gradients types are almost equivalent.

## 6.4 Direct sensitivity calculations

Our method is closely related to the direct sensitivity method, as defined in Biegler (2010), used for optimal control problems, which we applied in Jahanshahi et al. (2014). We also tested the direct sensitivity method to our problem which have time delay. We then set up the *delayed differential equations* symbolically, found the parameter sensitivities of the states, and integrated using a integrator for delayed differential equations (e.g. `dde23`). This was quite cumbersome. By taking advantage the fixed structure of the problem and the control system toolbox in MATLAB, it is much easier to differentiate the Laplace equations symbolically and obtain the *delayed differential equations* by letting MATLAB handle the conversions. This is the approach used in this work.

## 7 Conclusion

In this paper we have successfully applied the exact gradients for a typical performance (IAE) with constrained robustness $M_{\mathrm{ST}}$ 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 in MATLAB and extended to other controller formulations. 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.

## Bibliography

S. Alcántara, R. Vilanova, and C. Pedret. PID control in terms of robustness/performance and servo/regulator trade-offs: A unifying approach to balanced autotuning. *Journal of Process Control*, 23(4):527–542, 2013.

V. Alfaro, R. Vilanova, V. Méndez, and J. Lafuente. Performance/robustness tradeoff analysis of PI/PID servo and regulatory control systems. In *Industrial Technology (ICIT), 2010 IEEE International Conference on*, pages 111–116. IEEE, 2010.

Karl Johan Åström and Tore Hägglund. *Advanced PID control*. ISA-The Instrumentation, Systems, and Automation Society, 2006.

Karl Johan Åström, Hélène Panagopoulos, and Tore Hägglund. Design of PI controllers based on non-convex optimization. *Automatica*, 34(5):585–601, 1998.

Jens G. Balchen. A performance index for feedback control systems based on the fourier transform of the control deviation. 247, 1958.

Lorenz T. Biegler. *Nonlinear programming: concepts, algorithms, and applications to chemical processes*, volume 10. SIAM, 2010.

Olof Garpinger and Tore Hägglund. A software tool for robust PID design. In *Proc. 17th IFAC World Congress, Seoul, Korea*, 2008.

Olof Garpinger, Tore Hägglund, and Karl Johan Åström. Performance and robustness trade-offs in PID control. *Journal of Process Control*, 24(5):568–577, 2014.

Chriss Grimholt and Sigurd 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.

Chriss Grimholt and Sigurd Skogestad. Optimal PID-control on first order plus time delay systems & verification of the SIMC rules. In *10th IFAC International Symposium on Dynamics and Control of Process Systems*, 2013.

Chriss Grimholt and Sigurd Skogestad. Improved optimization-based design of PID controllers using exact gradients. volume 37, pages 1751–1757, 2015.

Albert C. Hall. *The analysis and synthesis of linear servomechanisms*. Technology Press Massachusetts Institute of Technology, 1943.

Martin Hast, Karl Johan Aström, Bo Bernhardsson, and Stephen Boyd. PID design by convex-concave optimization. In *Proceedings European Control Conference*, pages 4460–4465, 2013.

P. Hazebroek and B. L. Van der Waerden. The optimum tuning of regulators. *Trans. ASME*, 72:317–322, 1950.

Mikulas Huba. Performance measures, performance limits and optimal PI control for the IPDT plant. *Journal of Process Control*, 23(4):500–515, 2013.

Ari Ingimundarson and Tore Hägglund. Performance comparison between PID and dead-time compensating controllers. *Journal of Process Control*, 12(8): 887–895, 2002.

E. Jahanshahi, V. de Oliveira, C. Grimholt, and Skogestad S. A comparison between internal model control, optimal PIDF and robust controllers for unstable flow in risers. In *19th World Congress*, pages 5752–5759. The International Federation of Automatic Control, 2014.

Birgitta Kristiansson and Bengt Lennartson. Evaluation and simple tuning of PID controllers with high-frequency robustness. *Journal of Process Control*, 16 (2):91–102, 2006.

J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

G. C. Newton, L. A. Gould, and J. F. Kaiser. *Analytical design of linear feedback controls*. John Wiley & Sons, New York, N. Y, 1957.

Hélène Panagopoulos, Karl Johan Åström, and Tore Hägglund. Design of PID controllers based on constrained optimisation. *IEE Proceedings-Control Theory and Applications*, 149(1):32–40, 2002.

Danlel E. Rivera, Manfred Morari, and Sigurd Skogestad. Internal model control. 4. PID controller design. *Ind. Eng. Chem. Process Des. Dev.*, 25:252–256, 1986.

Tor Steinar Schei. Automatic tuning of PID controllers based on transfer function estimation. *Automatica*, 30(12):1983–1989, 1994.

F. G. Shinskey. How good are our controllers in absolute performance and robustness? *Measurement and Control*, 23(4):114–121, 1990.

Sigurd Skogestad. Simple analytic rules for model reduction and PID controller tuning. *Journal of process control*, 13(4):291–309, 2003.

Kristian Soltesz, Chriss Grimholt, Olof Garpinger, and Sigurd Skogestad. Simultaneous design of PID controller and measurement filter by optimization. 2014.

Murray P. Spiegel. *Schaum's outline series: Theory and Problems of Laplace Transforms*. McGraw-Hill Book Company, 1965.

John G. Ziegler and Nathaniel B. Nichols. Optimum settings for automatic controllers. *trans. ASME*, 64(11), 1942.

## A Pseudo code for the calculation of gradients

**Algorithm 1:** CostFunction($p$)

```
$t \leftarrow$ uniform distributed time points
$e_{dy}(t) \leftarrow$ get step response of $S(s;p)$ with time steps $t$
$e_{du}(t) \leftarrow$ get step response of $GS(s;p)$ with time steps $t$
calculate IAE for $e_{dy}$ and $e_{du}$ using numerical integration
$J \leftarrow$ calculate cost function using (8)
**return** $(J,\ e_{dy}(t),\ e_{du}(t))$
```

**Algorithm 2:** ConstraintFunction($p$)

```
$\omega \leftarrow$ logarithmically spaced frequency points in $\Omega$
$c_S \leftarrow \lvert S(j\omega) \rvert - M_S^{ub}$
$c_T \leftarrow \lvert T(j\omega) \rvert - M_T^{ub}$
$c \leftarrow$ stack $c_S$ and $c_T$ into one vector
**return** $(c)$
```

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

```
$t \leftarrow$ uniform distributed time points
for $i \leftarrow 1$ to number of parameters
    do
        $\nabla e_{dy}(t) \leftarrow$ get time response of $\nabla e_{dy}(s)$ from (20) with time steps $t$
        $\nabla e_{du}(t) \leftarrow$ get time response of $\nabla e_{du}(s)$ from (21) with time steps $t$
        $\nabla \mathrm{IAE}_{dy} \leftarrow$ numerical integration of (16) using $e_{dy}$ and $\nabla e_{dy}(t)$
        $\nabla \mathrm{IAE}_{du} \leftarrow$ numerical integration of (17) using $e_{du}$ and $\nabla e_{du}(t)$
        $\nabla J \leftarrow$ calculate from (15)
$\nabla J \leftarrow$ stack the cost function sensitivities into one vector
return $(\nabla J)$
```

**Algorithm 4:** GRADCONSTRAINTFUNCTION($p$)

```
$\omega \leftarrow$ logarithmically spaced frequency points
for $i \leftarrow 1$ to number of parameters
    do
        $\nabla c_{\mathrm{S}} \leftarrow$ evaluate (24) for frequencies $\omega$
        $\nabla c_{\mathrm{T}} \leftarrow$ evaluate (25) for frequencies $\omega$
$\nabla c \leftarrow$ combine $\nabla c_{\mathrm{S}}$ and $\nabla c_{\mathrm{T}} p_i$ into a matrix
return $(\nabla c)$
```

## 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*

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

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

$$|g(t)| = \mathrm{sign}\Big\{g(t)\Big\}\, g(t),$$

where the sign function has the following values

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

Using the product rule we get,

$$\frac{\partial}{\partial p}\left(\text{sign}\left\{g(t)\right\} g(t)\right) = \text{sign}\left\{g(t)\right\} \frac{\partial g(t)}{\partial p} + g(t)\,\frac{\partial \text{sign}\left\{g(t)\right\}}{\partial p}. \tag{44}$$

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 $t_\alpha$ to $t_\beta$ with respects to the parameter $p$ can be written as*

$$\frac{\mathrm{d}}{\mathrm{d}p}\left(\int_{t_\alpha}^{t_\beta} |g(t)|\,\mathrm{d}t\right) = \int_{t_\alpha}^{t_\beta} sign\left\{g(t)\right\}\left(\tfrac{\partial g(t)}{\partial p}\right)\mathrm{d}t \tag{45}$$

*Proof.* If $g(t)$ and its partial derivative $\frac{\partial g(t)}{\partial p}$ are continuous wrt. $t$ 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 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 $G(s)$,*

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

*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 $\frac{\partial g(t)}{\partial p}$ 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,

$$\frac{\partial S}{\partial p_i} = \frac{\partial S}{\partial L} \frac{\partial L}{\partial K} \frac{\partial K}{\partial p_i}. \tag{46}$$

We have

$$\frac{\partial S}{\partial L} = \frac{\partial}{\partial L}(1+L)^{-1} = -(1+L)^{-2} = -S^2. \tag{47}$$

and $\frac{\partial L}{\partial K} = G$ Thus,

$$\frac{\partial S}{\partial p_i} = -S^2 G \frac{\partial K}{\partial p_i}. \tag{48}$$

Similarly,

$$\frac{\partial GS}{\partial p_i} = G \frac{\partial S}{\partial p_i} = -(GS)^2 \frac{\partial K}{\partial p_i}. \tag{49}$$

## C Derivation of the exact sensitivities for the constraints

### C.1 Sensitivity of $G(j\omega)G^*(j\omega; p)$ 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 $G(j\omega)$ and $G^*(j\omega)$). Then the differential of the product of the* *two with respect to the parameter p is*

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

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

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

$$G(j\omega) = x + jy, \tag{50}$$
$$G^*(j\omega) = x - jy. \tag{51}$$

The product rule gives

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

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 $\left|G(j\omega);\, p\right|$ 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 respects 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 term 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

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

$\square$

The observant reader might have wondered why the sensitivity of the transfer function is not written using the chain-rule with

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

This is because the derivative $\frac{\partial G^*(j\omega)}{\partial G(j\omega)}$ is non-analytic and do not exist anywhere (Spiegel, 1965).