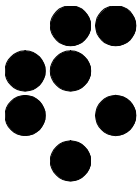
J. Stefan Institute, Ljubljana, Slovenia



Report **DP-8955**

Improving PID Controller Disturbance Rejection by Means of Magnitude Optimum

Damir Vrančić and Satja Lumbar*

*Faculty of Electrical Engineering, University of Ljubljana

May, 2004

Table of Contents

1. Introduction

Most of the PID tuning methods available so far concentrate on improving tracking performance of the closed-loop.

One of such tuning methods is the magnitude optimum (hereafter "MO") method [1,8,10,11,15,16,17], which results in a relatively fast and non-oscillatory system closed-loop response. The MO method is originally used for achieving superior reference tracking. On the other hand, by using the MO method, the process poles could be cancelled by the controller zeros. This may lead to poor attenuation of load disturbances if the cancelled poles are excited by disturbances and if they are slow compared to the dominant closed-loop poles [1]. Poorer disturbance rejection performance can be observed when controlling low-order processes. This is one of the most serious drawbacks of the MO method. In process control, disturbance rejection is usually more important than superior tracking [1,12].

Recently, a modified method for tuning parameters of PI controllers, based on the magnitude optimum method, has been developed. Namely, the original magnitude optimum (MO) has been adjusted for improving disturbance rejection by using the so-called disturbance-rejection MO (DRMO) method [18]. The results were encouraging, since disturbance rejection performance has been greatly improved, especially for lower-order processes.

Modification of the magnitude optimum method is not limited only to the PI controllers. The aim of this report is to show how the modification can be applied to PID controllers. However, since the structure of the PID controller is more complex, the calculation of the PID controller parameters cannot be performed analytically as is the case for the PI controller. Therefore, controller parameters should be calculated by numerical methods.

DRMO method for PI and PID controllers is compared to original MO method on 63 process models.

The report is set out as follows. Section 2 describes the original magnitude optimum tuning method for the PI and the PID controllers. Section 3 provides the theoretical background of disturbance rejection magnitude optimum method for the PI and the PID controller. Conclusions are given in section 5. Appendix shows the matlab files, which were used for the calculation of PI/PID controller parameters and the closed-loop simulation.

2. The original MO method

The objective of the magnitude optimum (MO) method is to maintain the closed-loop magnitude response curve as flat and as close to unity for as large bandwidth as possible for a given plant and controller structure [8,15] (see Fig. 1). Such a controller results in a fast and non-oscillatory closed-loop response for a large class of processes.

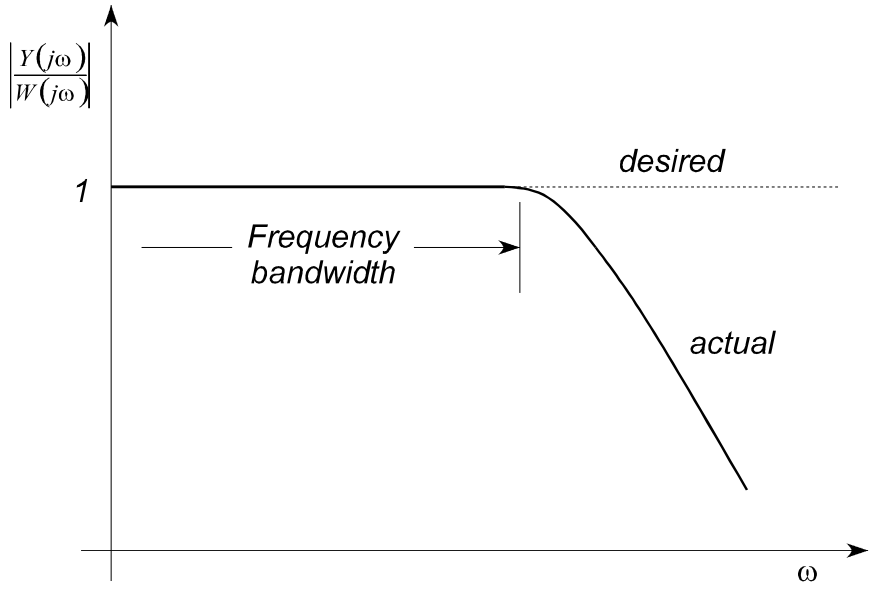


Fig. 1. Magnitude optimum (MO) criterion.

This technique can be found under other names such as modulus optimum [1] or Betragsoptimum [1,10,11], and results in a fast and non-oscillatory closed-loop time response for a large class of process models.

If G_{CL}(s) is the closed-loop transfer function from the set-point to the process output, the controller is determined in such a way that

$$\begin{aligned} & G_{CL}(0)=1 \\ & \left.\frac{d^r\left|G_{CL}(i\omega)\right|}{d\omega^r}\right|_{\omega=0}=0 \end{aligned} \tag{1}$$

for as many r as possible [1].

Let us assume that the actual process can be described by the following rational transfer function:

$$G_P(s)=K_{PR}\frac{1+b_1 s+b_2 s^2+\cdots+b_m s^m}{1+a_1 s+a_2 s^2+\cdots+a_n s^n}e^{-sT_{del}}\ , \tag{2}$$

where K_{PR} denote the process steady-state gain, and a_1 to a_n and b_1 to b_m are the corresponding parameters ($m \le n$) of the process transfer function, where n can be an arbitrary positive integer value. T_{del} denotes pure time delay.

The process transfer function (2) can be developed into the following infinite series[1]:

$$G_P = A_0 - A_1 s + A_2 s^2 - A_3 s^3 + \cdots, \tag{3}$$

where A_i are so-called characteristic areas:

$$\begin{aligned} A_0 &= K_{PR} \\ A_1 &= K_{PR}(a_1 - b_1 + T_{del}) \\ &\vdots \\ A_k &= K_{PR}\left(\begin{array}{l} (-1)^{k+1}(a_k - b_k) + \\ + \sum_{i=1}^{k} (-1)^{k+i} \frac{T_{del}{}^i b_{k-i}}{i!} \end{array} \right) + \\ &+ \sum_{i=1}^{k-1} (-1)^{k+i-1} A_i a_{k-i} \end{aligned} \tag{4}$$

The controller structure is chosen to be of the PID type (see Fig. 2) and is described by the following transfer function:

$$G_C(s) = \frac{U(s)}{E(s)} = K + \frac{K_i}{s} + \frac{sK_d}{1 + sT_f}, \tag{5}$$

where U and E denote Laplace transforms of the controller output, and the control error respectively. The controller parameters K, K_i, K_d, and T_f represent proportional gain, integral gain, derivative gain, and filter time constant, respectively.

In industrial controllers, the filter time constant T_f is usually given implicitly by defining the ratio δ between T_f, K_d and K:

$$\delta = \frac{T_f}{K_d} K \tag{6}$$

Typical values of δ are 0.05 to 0.125 [1].

---

[1] The process time delay can be developed into infinite Taylor (or Pade) series (see Vrančić et al., 1999).

Note that expression (3) is not related to any conventional process model. However, this kind of representation by the infinite series will result in simpler calculation of controller parameters.

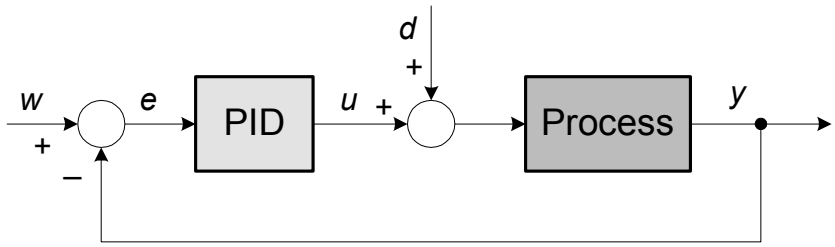


Fig. 2. The PID controller in the closed-loop with the process.

The closed-loop transfer function is the following:

$$G_{CL}(s) = \frac{G_P G_C}{1 + G_P G_C}. \tag{7}$$

When placing the process transfer function (3) and controller transfer function (5)[2] into expressions (1), the controller parameters can be calculated from process characteristic areas in the following way [17,20]:

$$\begin{bmatrix} K_i \\ K \\ K_d \end{bmatrix} = \begin{bmatrix} -A_1 & A_0 & 0 \\ -A_3 & A_2 & -A_1 \\ -A_5 & A_4 & -A_3 \end{bmatrix}^{-1} \begin{bmatrix} -0.5 \\ 0 \\ 0 \end{bmatrix} \tag{8}$$

The PI controller parameters are calculated simply as:

$$\begin{bmatrix} K_i \\ K \end{bmatrix} = \begin{bmatrix} -A_1 & A_0 \\ -A_3 & A_2 \end{bmatrix}^{-1} \begin{bmatrix} -0.5 \\ 0 \end{bmatrix} \tag{9}$$

The characteristic areas can also be calculated in time-domain from the measurement of the process steady-state change. If $u(t)$ and $y(t)$ denote process input and process output signals, then the following conditions should be obeyed to calculate the characteristic areas:

$$\begin{aligned} &\dot{y}(0) = \ddot{y}(0) = \cdots = 0 \\ &\lim_{t\to\infty} \dot{y}(t) = \lim_{t\to\infty} \ddot{y}(t) = \cdots = 0 \\ &\lim_{t\to\infty} y(t) \neq y(0) \\ &|u(t)| < \infty \end{aligned} \quad . \tag{10}$$

The calculation of areas proceeds then as follows:

[2] Note that the filter time constant (T_f) is assumed to be very small ($T_f \to 0$). If this condition is not satisfied, the exact expressions for calculating PID controller parameters are given in [20].

$$u_k = \int_0^\infty u_{k-1}(\tau)d\tau$$
$$y_k = \int_0^\infty y_{k-1}(\tau)d\tau \quad , \tag{11}$$

where

$$\begin{aligned} \Delta U &= u(\infty) - u(0) \\ \Delta Y &= y(\infty) - y(0) \\ A_0 &= \frac{\Delta Y}{\Delta U} \\ u_0(t) &= \frac{u(t) - u(0)}{\Delta U} \\ y_0(t) &= \frac{y(t) - y(0)}{\Delta Y} \end{aligned} \quad . \tag{12}$$

The areas in expression (4) can then be calculated in the following way:

$$\begin{aligned} A_1 &= A_0 u_1 - y_1 \\ &\vdots \\ A_k &= A_{k-1}u_1 - A_{k-1}u_2 + \cdots + (-1)^{k-1} A_0 u_k + (-1)^k y_k \end{aligned} \quad . \tag{13}$$

Since in practice the integration horizon should be limited, there is no need to wait until $t=\infty$. It is enough to integrate until the transient response dies out.

The given tuning procedure will be illustrated on two examples.

Case 1

The process is assumed to have the following transfer function:

$$G_P = \frac{1}{(1+s)^5} \quad . \tag{14}$$

The specific areas are calculated from (4):

$$A_0 = 1,\ A_1 = 5,\ A_2 = 15,\ A_3 = 35,\ A_4 = 70,\ A_5 = 126 \, . \tag{15}$$

The PID controller parameters are calculated from expression (8):

$$K = 1.06,\ K_i = 0.313,\ K_d = 1 \quad . \tag{16}$$

Note that in all simulation experiments, T_f is chosen as $T_f = K_d/(10K)$.

Fig. 3. shows the closed-loop response on reference change and input-disturbance. Note that the closed-loop response features relatively good tracking performance and disturbance rejection.

Case 2

The process is assumed to have the following transfer function:

$$G_P = \frac{1}{(1+s)^2(1+0.1s)} \ . \tag{17}$$

The specific areas (4) are:

$$\begin{aligned} &A_0 = 1,\ A_1 = 2.1,\ A_2 = 3.21, \\ &A_3 = 4.321, A_4 = 5.432,\ A_5 = 6.543 \end{aligned} \tag{18}$$

and the PID controller parameters (8) are as follows:

$$K = 10.19,\ K_i = 5.09,\ K_d = 5.1 \ . \tag{19}$$

Fig. 4. shows the closed-loop response on reference change and input-disturbance (d=1 at t=0s). Note that disturbance rejection compared to reference tracking becomes relatively slow. The reason is that the process is of a relatively lower order so the controller zeros practically cancel the slowest process poles [1]. Namely, the controller can be expressed in the following form:

$$G_C(s) = 5.09\,\frac{(1+s)(1+1.002s)}{s} \ . \tag{20}$$

So, both dominant poles are cancelled by controller zeros.

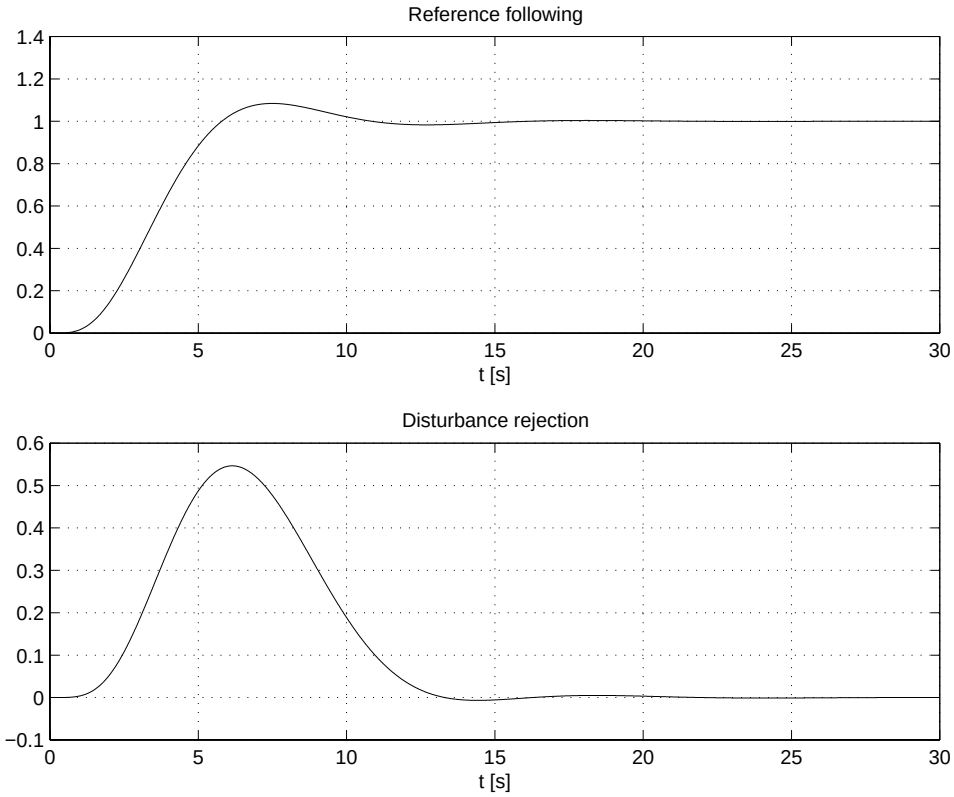


Fig. 3. The closed-loop response of the process (14) to the step change of the reference input and disturbance signal; the PID controller applied is tuned by using the MO method.

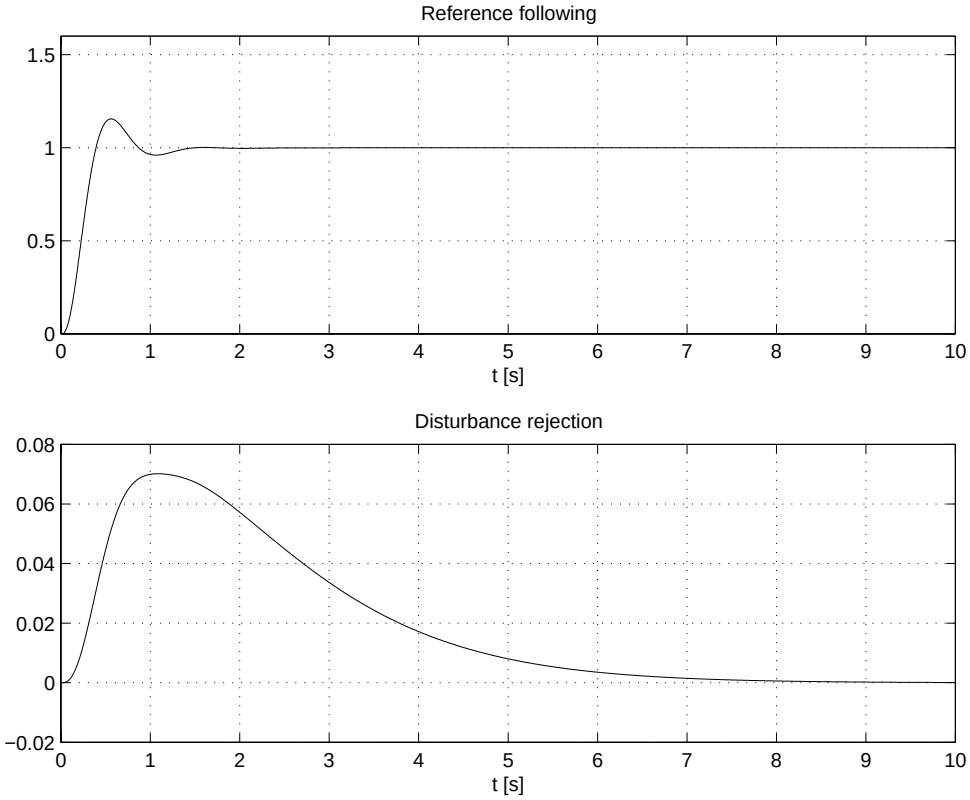


Fig. 4. The closed-loop response of the process (17) to the step change of the reference input and disturbance signal; the PID controller applied is tuned by using the MO method.

3. The disturbance-rejection MO (DRMO) method for PID controller

In the previous section it was shown that, by using the original MO method, disturbance rejection is degraded when dealing with lower-order processes, since slow process poles are almost entirely cancelled by controller zeros.

This is not unusual, since the MO method aims at achieving good reference tracking, so it optimises the transfer function $G_{CL}(s)=Y(s)/W(s)$ instead of $G_{CLD}(s)=Y(s)/D(s)$. Let us now express $G_{CL}(s)$ in terms of $G_{CLD}(s)$:

$$\begin{aligned} G_{CL}(s) &= G_{CLD}(s)G_C(s) = \\ &= G_{CLD}(s)\frac{K_i}{s}\left(1+\frac{K}{K_i}s+\frac{K_d}{K_i}s^2\right) = , \\ &= G_{CLO}(s)\left(1+\frac{K}{K_i}s+\frac{K_d}{K_i}s^2\right) \end{aligned} \tag{21}$$

where

$$G_{CLO}(s) = \frac{G_P(s)}{1+G_P(s)G_C(s)}\frac{K_i}{s} \ . \tag{22}$$

From expression (21) it can be seen that controller's zeros (the sub-expression in brackets) play an important role within $G_{CL}(s)$, which is actually "optimised" by the original MO method. On the other hand, controller's zeros may significantly degrade disturbance rejection performance. A strategy proposed herein is to optimise the transfer function $G_{CLO}(s)$ (22) instead of $G_{CL}(s)$ in expression (1).

The number of conditions which can be satisfied in expression (1) depends on controller order [20]. Namely, by using the PID controller (three independent controller parameters), the first three derivatives (r=1,2,3) can be satisfied. Note that the first expression ($G_{CL}(0)=1$) is already satisfied since the controller contains an integral term (under condition that the closed-loop response is stable).

By placing $G_{CLO}(s)$ in expression (1), the following set of equations are obtained (for r=1,2 and 3):

$$A_0^{\,2}K^2 + 2A_0K - 2A_0^{\,2}K_iK_d - 2A_1K_i + 1 = 0 \tag{23}$$

$$A_0^{\,2}K_d^{\,2} + 4A_0K_dA_2K_i + A_1^{\,2}K^2 + 2A_1K_d - 2A_2K + 2A_3K_i - 2A_0K^2A_2 - 2A_1^{\,2}K_iK_d = 0 \tag{24}$$

$$\begin{aligned} &2A_0A_4K^2 + A_2^{\,2}K^2 - 2A_1A_3K^2 - 4A_0A_4K_iK_d - 2A_3K_d + 2A_4A_5K - 2K_i - 2A_0A_2K_d^{\,2} - \\ &- 2A_2^{\,2}K_iK_d + A_1^{\,2}K_d^{\,2} + 4A_1A_3K_dK_i = 0 \end{aligned} \tag{25}$$

Since in each equation, the parameters K and K_i are of the second order, the final result can be obtained usually by means of optimisation (it cannot be solved analytically). The proper solution should fulfil the following conditions:

- K should be a real number ($K\in\Re$) and should not exceed some pre-defined range of values.

- The chosen K_i together with K and K_d should have the same sign as the process steady-state gain (K_{PR} or A_0).

Initial values for the optimisation are the values calculated by using expressions (8). Any method of optimisation (iterative search for numeric solution) that solves the system of nonlinear equations can be applied.

Possible practical tuning procedure is to calculate derivative gain from expression (8), then calculate proportional gain K from equations (23) and (24):

$$\begin{aligned}&\left(2A_0^{\,2}A_3+2A_1^{\,3}-4A_0A_1A_2\right)K^2+\left(4A_0A_3-4A_0A_1^{\,2}K_d-4A_1A_2+4A_0^{\,2}A_2K_d\right)K+\\&4A_0A_2K_d+2A_3+2A_1^{\,2}K_d+6A_0^{\,2}A_1K_d^{\,2}+2A_0^{\,4}K_d^{\,3}=0\end{aligned}\quad . \tag{26}$$

Integral gain K_i can be then calculated from equation (23):

$$K_i=\frac{\left(1+A_0K\right)^2}{2\left(A_1+A_0^{\,2}K_d\right)}\ . \tag{27}$$

Then equation (25) should be tested. Derivative gain should be modified and K (26) and K_i (27) re-calculated until equation (25) becomes zero. One of the simplest solutions for the calculation of controller parameters is to use some simple search methods like Newton's. In most cases only few iterations are required for achieving very accurate result. MATLAB files which calculate optimal PID controller parameters for disturbance rejection (according to expressions (23) to (25)) are given in appendix and in [24].

Such tuning procedure is also called *disturbance-rejection MO* (DRMO) method.

Calculation of PI controller parameters is simpler, since K_d is fixed to K_d=0. Then K and K_i are calculated from expressions (26) and (27).

To sum up, the DRMO tuning procedure goes as follows:

1. Calculate characteristic areas from the process transfer function (4) or from the process steady-state change (13),
2. calculate PID controller parameters (8) by using the original MO method (only for PID controllers),
3. calculate gains of the proportional and the integral term from expressions (26) and (27). For the PI controller, replace K_d with 0.
4. Test expression (25). Appropriately modify K_d and repeat steps 3 and 4 until (25) is satisfied.

The given tuning procedure will be illustrated on two examples.

Case 3

The process is assumed to have the same transfer function as in Case 1. The obtained PID controller parameters were calculated by using described tuning procedure (the optimisation was performed in program package Matlab [24]).

The obtained controller parameters were the following:

$$K=1.29,\ K_i=0.43,\ K_d=1.10\ . \tag{28}$$

The calculated parameters are similar to those given in expression (16). It is therefore expected that tracking and disturbance rejection performance will remain almost the same to Case 1. This assumption is confirmed by the closed-loop responses given in Fig. 5 (see solid lines).

Case 4

The process is assumed to have the same transfer function as in Case 2.

The obtained controller parameters were the following:

$$K = 24.3,\ K_i = 44.57,\ K_d = 5.10\ . \tag{29}$$

The calculated PID controller parameters are now quite different from ones given in (19). Note that the integral gain (K_i) is now much higher than before, so relatively high change of disturbance rejection performance might be expected. The closed-loop responses on reference change (w=1 at t=0) and disturbance (d=1 at t=0s) are given in Fig. 6 (solid lines). It can be seen that disturbance rejection is now quite improved in comparison to the original MO method (see Fig. 4).

However, improved disturbance rejection has its price. Namely, enlarged process overshoots after reference change can be noticed especially in Figure 6. The only way of improving deteriorated tracking performance while retaining the obtained disturbance rejection performance is using two-degrees-of-freedom (2-DOF) PID controller. One of the most simple solutions is to use the set-point weighting approach [1]. Only the integral term has been connected to the control error, while proportional and derivative terms were connected to the process output only:

$$U(s) = \frac{K_i}{s} E(s) - \left(K + \frac{sK_d}{1+sT_f} \right) Y(s). \tag{30}$$

Figures 5 and 6 show tracking performance when using 2-DOF PID controller (see dashed lines). The overshoots on reference following are now quite lower than when using a conventional PID controller.

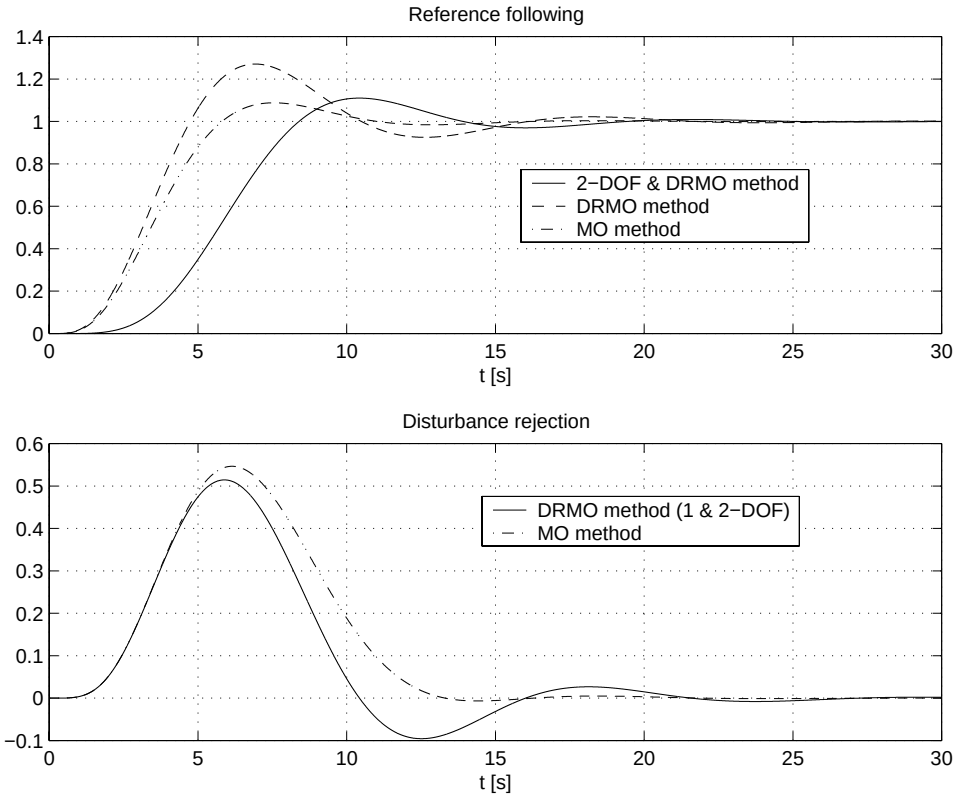


Fig. 5. The closed-loop response of the process (14) to the step reference signal (w=1 at t=0) and step disturbance signal (d=1 at t=0).

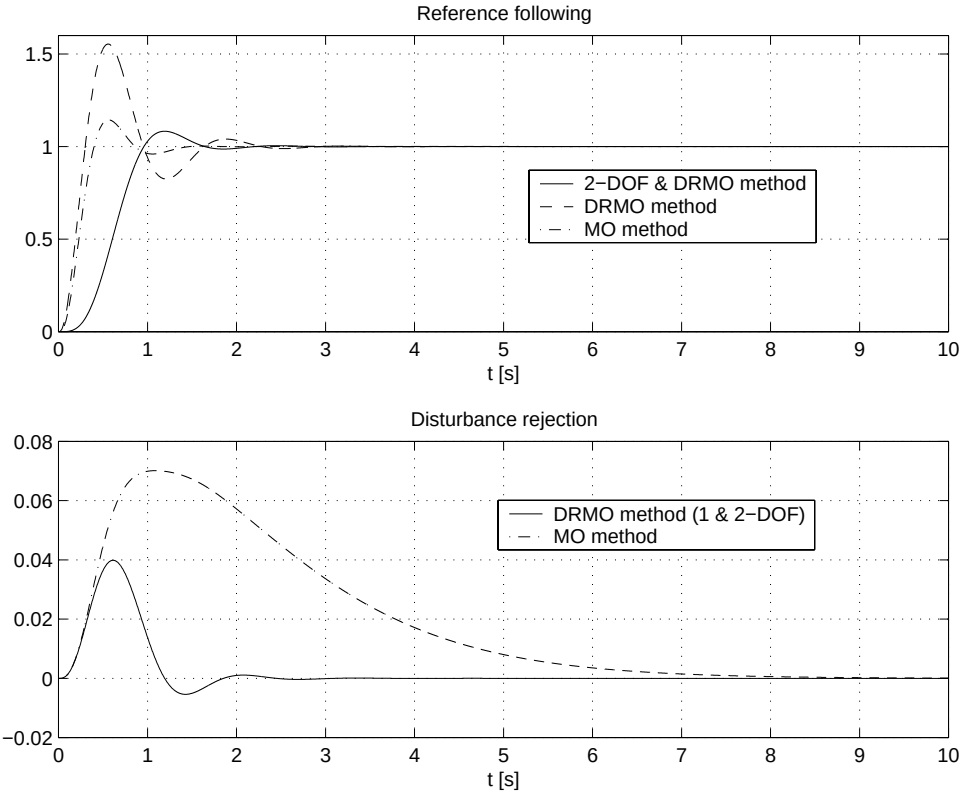


Fig. 6. The closed-loop response of the process (17) to the step reference signal (w=1 at t=0) and step disturbance signal (d=1 at t=0).

Besides the shown examples, larger number of experiments on different process models has been performed. The closed-loop responses on disturbance were compared in Appendix A for MO and DRMO methods. Appendix B gives the Matlab files that were used for the calculation of controller parameters.

It is obvious that disturbance rejection performance improves significantly when using the proposed DRMO method. The exception is the process G_{P8}, where some sub-processes give more oscillatory response when using the PID controller. The reason is that the number of expressions which can be satisfied in (1) depend on the number of controller parameters. Therefore, higher derivatives in (1) are not always satisfied and the closed-loop amplitude response (Fig. 1) may have resonant peak. This happened when using process G_{P8}.

4. Conclusions

The purpose of this report was to present a modification to the original MO method in order to achieve "optimal" disturbance rejection performance for the PID controllers. The given modification is simple and straightforward for implementation in practice. Simulation experiments on two process models have shown that the proposed approach results in improved disturbance rejection properties. The disturbance rejection performance is evidently improved for the lower-order processes. By using 2-DOF PID controller, the overshoots on reference following were kept relatively low.

On the other hand, the original MO method, as well as the proposed modification, does not guarantee stable closed-loop responses [17,22,20]. Unstable responses can be obtained when dealing with processes with oscillating poles or "strong" zeros [17], as has been shown in Appendix. Fortunately, such processes are infrequent in practice.

5. References

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

[2] Åström, K.J., Panagopoulos, H. and Hägglund, T., Design of PI Controllers based on Non-Convex Optimization. *Automatica*, 1998, **34** (5), 585-601.

[3] Besançon-Voda, A., Iterative auto-calibration of digital controllers: methodology and applications. *Control Engineering Practice*, 1998, **6**, 345-358.

[4] Deur, J., Perić, N. and Stajić, D., Design of reduced-order feedforward controller. Proceedings of the UKACC CONTROL'98 Conference, Swansea, 1998, 207-212.

[5] Ender, D.B., Process control performance: Not as good as you think. *Control Engineering*, 1993, **40** (10), 180-190.

[6] Gorez, R., "A survey of PID auto-tuning methods" , 1997, *Journal A*, **38** (1), 3-10.

[7] Hang, C.C., Åström, K.J. and Ho, W.K. "Refinements of the Ziegler-Nichols tuning formula", *IEE Proceedings – Part D*, 1991, **138** (2), 111-118.

[8] Hanus, R. “Determination of controllers parameters in the frequency domain”, *Journal A*, 1975, **XVI** (3).

[9] Ho, W.K., Hang, C.C. and Cao, L.S. “Tuning of PID Controllers Based on Gain and Phase Margin Specifications”, 12th World Congress IFAC, Sydney, Proceedings, 1993, Vol. 5, 267-270.

[10] Kessler, C., “Über die Vorausberechnung optimal abgestimmter Regelkreise Teil III. Die optimale Einstellung des Reglers nach dem Betragsoptimum”, *Regelungstechnik*, 1955, 3, 40-49.

[11] Preuss, H.P., “Prozessmodellfreier PID-regler- Entwurf nach dem Betragsoptimum”, *Automatisierungstechnik*, 1991, **39** (1), 15-22.

[12] Shinskey, F.G. (2000). PID-deadtime control of distributed processes. *Pre-prints of the IFAC Workshop on Digital Control*, Terassa, pp. 14-18.

[13] Skogestad, S. “Simple analytic rules for model reduction and PID controller tuning”, *Journal of process control*, 2003, **13**, 291-309.

[14] Strejc, V., “Auswertung der dynamischen Eigenschaften von Regelstrecken bei gemessenen Ein- und Ausgangssignalen allgemeiner Art”, *Z. Messen, Steuern, Regeln*, 1960*,* 3 (1), 7-11.

[15] Umland, J.W. and Safiuddin, M. “Magnitude and symmetric optimum criterion for the design of linear control systems: what is it and how does it compare with the others?”, *IEEE Transactions on Industry Applications*, 1990, ***26*** (3), 489-497.

[16] Voda, A. A. & I. D. Landau (1995). A Method for the Auto-calibration of PID Controllers. *Automatica*, **31** (1), pp. 41-53.

[17] Vrančić, D. (1997). Design of Anti-Windup and Bumpless Transfer Protection. *PhD Thesis*, University of Ljubljana, Ljubljana. Available on www-e2.ijs.si/Damir.Vrancic/bibliography.html

[18] Vrančić, D., and S. Strmčnik (1999). Achieving Optimal Disturbance Rejection by Using the Magnitude Optimum Method. *Pre-prints of the CSCC’99 Conference*, Athens, pp. 3401-3406.

[19] Vrančić, D., Peng Y. and Danz, C. “A comparison between different PI controller tuning methods”, Report DP-7286, J. Stefan Institute, Ljubljana, 1995.

Available on http://www-e2.ijs.si/Damir.Vrancic/bibliography.html

[20] Vrančić, D., S. Strmčnik, and Đ. Juričić (2001). A magnitude optimum multiple integration method for filtered PID controller. *Automatica*, 37, pp. 1473-1479.

[21] Vrančić, D. and Strmčnik S. “Achieving optimal disturbance rejection by means of the magnitude optimum method”, Proceedings of the CSCC'99, July 4-8, Athens, 1999, 3401-3406.

[22] Vrančić, D., Peng, Y. and Strmčnik, S. “A new PID controller tuning method based on multiple integrations”, *Control Engineering Practice*, 1999, 7, 623-633.

[23] Vrančić, D., Strmčnik, S. and Juričić, Đ. “A magnitude optimum multiple integration method for filtered PID controller”, *Automatica*, 2001, 37, 1473-1479.

[24] Vrančić, D. “Matlab functions for the calculation of controller parameters from the characteristic areas” and “Matlab Toolset for Disturbance Rejection Controller (PID)”, Available on *http://www-e2.ijs.si/damir.vrancic/tools.html.*

[25] Vrečko, D., Vrančić D., Juričić Đ. and Strmčnik S. “A new modified Smith predictor: the concept, design and tuning”, *ISA Transactions*, 2001, **40** (2), 111-121.

[26] Wang, Q. G., Hang C. C. and Bi Q. “Process frequency response estimation from relay feedback”, *Control Engineering Practice*, 1997, **5** (9), 1293-1302.

[27] Whiteley, A.L., “Theory of servo systems, with particular reference to stabilization”, *The Journal of IEE, part II*, 1946, ***93*** (34), 353-372.

Appendix A. Comparison of disturbance-rejection properties of original MO and DRMO method for 63 process models

The optimal disturbance rejection magnitude optimum (DRMO) tuning method has been tested and compared to original MO method on 63 process models. There are 9 types of process models, each with 7 different value of parameter, as given in Table 1.

Process	***Range of parameter***
$G_{P1}(s)=\frac{e^{-s}}{(1+sT)}$	$T=0.1s\ldots10s$
$G_{P2}(s)=\frac{e^{-s}}{(1+sT)^2}$	$T=0.1s\ldots10s$
$G_{P3}(s)=\frac{1}{(1+sT)(1+s)}$	$T=0.1s\ldots10s$
$G_{P4}(s)=\frac{1}{(1+sT)^2(1+s)^2}$	$T=0.1s\ldots10s$
$G_{P5}(s)=\frac{1}{(1+s)^n}$	$n=2\ldots8$
$G_{P6}(s)=\frac{1}{(1+s)(1+sT)(1+sT^2)(1+sT^3)}$	$T=0.2s\ldots0.8s$
$G_{P7}(s)=\frac{(1-sT)}{(1+s)^3}$	$T=0.1s\ldots10s$
$G_{P8}(s)=\frac{e^{-s}(1+sT)}{(1+s)^2}$	$T=0.1s\ldots1s$
$G_{P9}(s)=\frac{1}{(1+s)(1+s(1+i\alpha))(1+s(1-i\alpha))}$	$\alpha=0.1s\ldots1s$

Table 1. The tested processes with parameter range

The PI and PID controller parameters, calculated by using the MO method ((9) and (8)), are denoted as "area2PI" and "area2PID", respectively (matlab functions area2PI.m and are2PID.m are given in Appendix B). The PI and PID controller parameters calculated by using the DRMO method are denoted as "PIopt" and "PIDopt" (functions PIopt.m and PIDopt.m are given in Appendix B). Indexes in $G_{Pi,j}$ denote i = process (1-9), j = sub-process (1-7).

Process 1: $G_{P1} = \dfrac{e^{-s}}{Ts+1}$

Table 1. PI controller parameters calculated by using DRMO method (PIopt.m)

T	0.1	0.2	0.5	1	2	5	10
K	0.27529	0.2946	0.39546	0.62772	1.1643	2.8814	5.7944
K_i	0.73925	0.69833	0.6491	0.66237	0.78073	1.2555	2.0984

Table 2. PI controller parameters calculated by using MO method (area2PI.m)

T	0.1	0.2	0.5	1	2	5	10
K	0.25677	0.27442	0.36538	0.57143	1.0395	2.5165	5.0083
K_i	0.68797	0.64535	0.57692	0.53571	0.51316	0.50275	0.50076

Table 3. PID controller parameters calculated by using DRMO method (PIDopt.m)

T	0.1	0.2	0.5	1	2	5	10
K	0.52017	0.57028	0.80817	1.3081	2.3983	5.7799	10
K_i	0.97611	0.94752	0.97649	1.1468	1.5836	3.0034	4.4475
K_d	0.083731	0.10117	0.1741	0.32277	0.64648	1.6525	2.603

Table 4. PID controller parameters calculated by using MO method (area2PID.m)

T	0.1	0.2	0.5	1	2	5	10
K	0.45354	0.49247	0.66883	1.0203	1.7587	4.003	7.7514
K_i	0.86686	0.82705	0.77922	0.76014	0.75291	0.75049	0.75012
K_d	0.072096	0.086815	0.1461	0.26182	0.50615	1.2525	2.5012

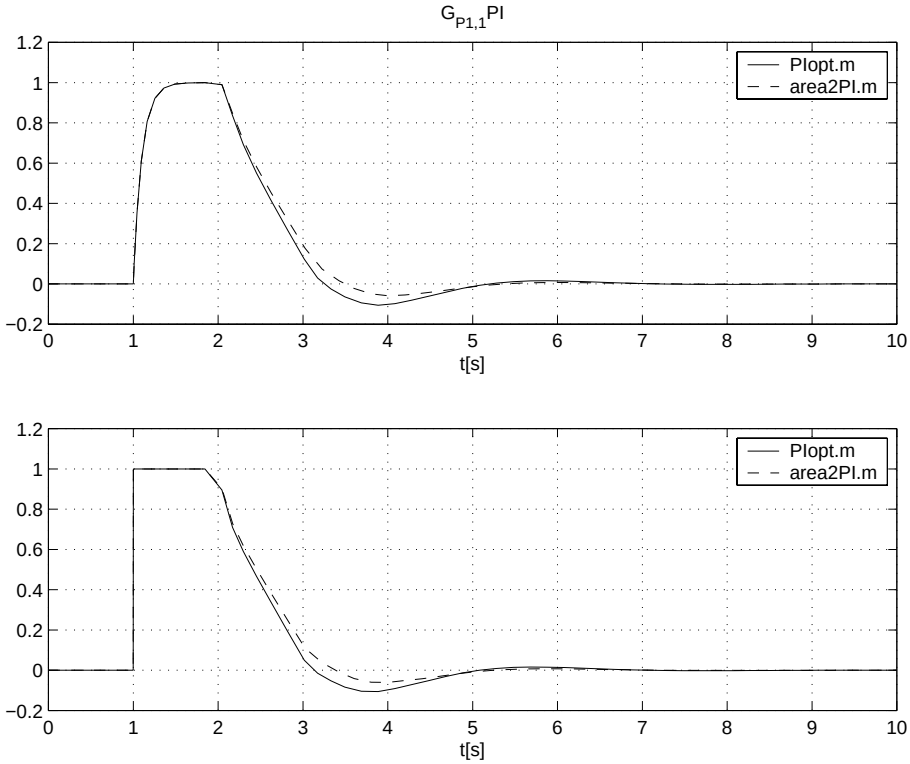


Fig. 7. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

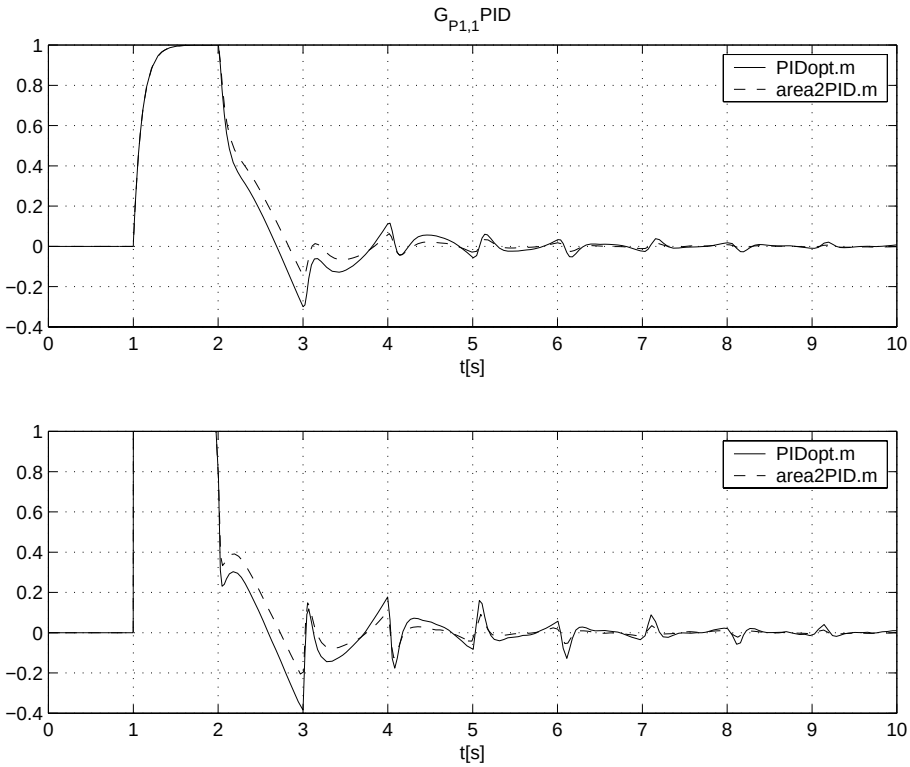


Fig. 8. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

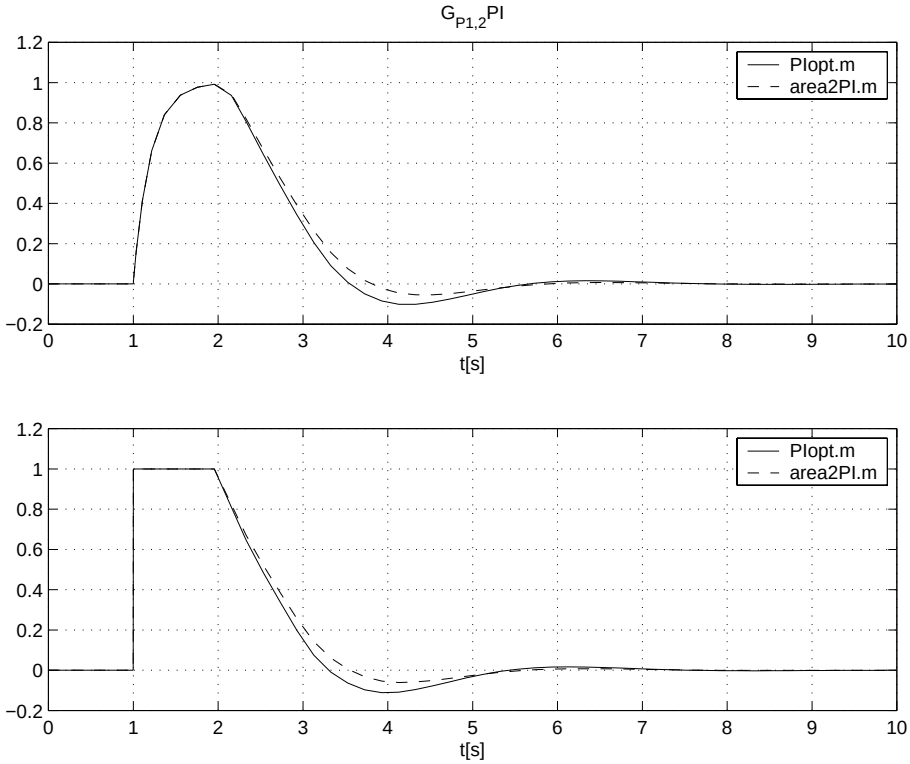


Fig. 9. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

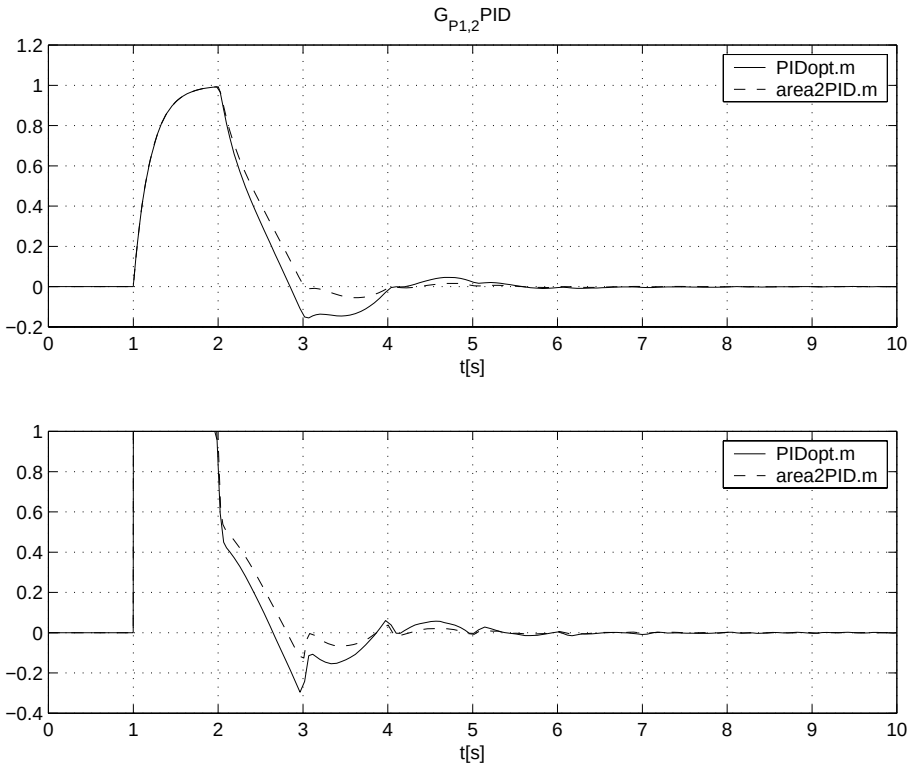


Fig. 10. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

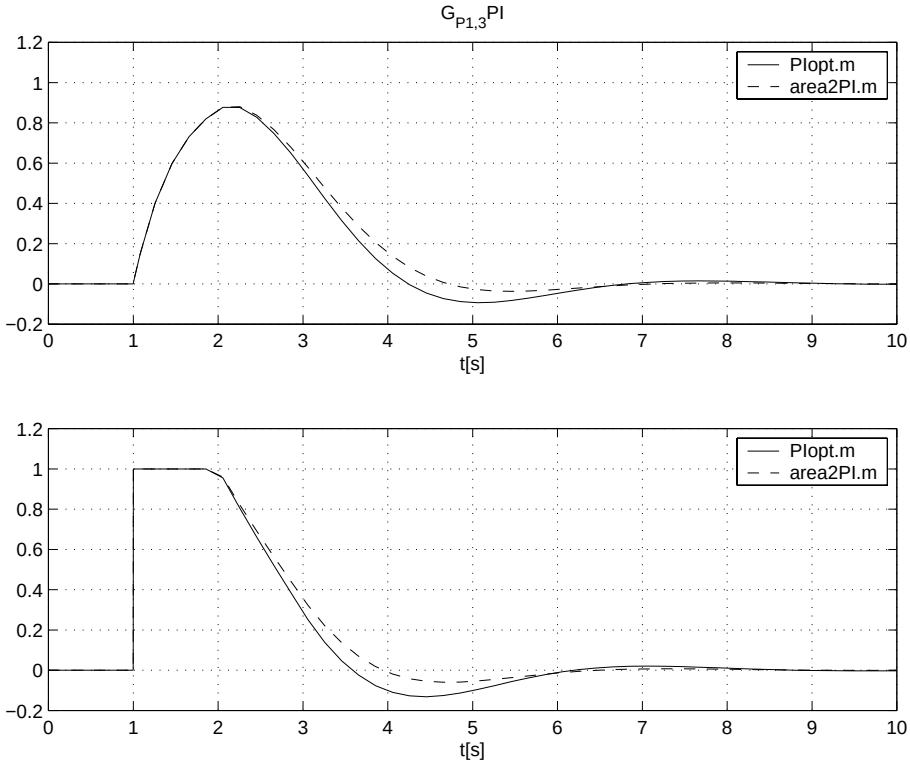


Fig. 11. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

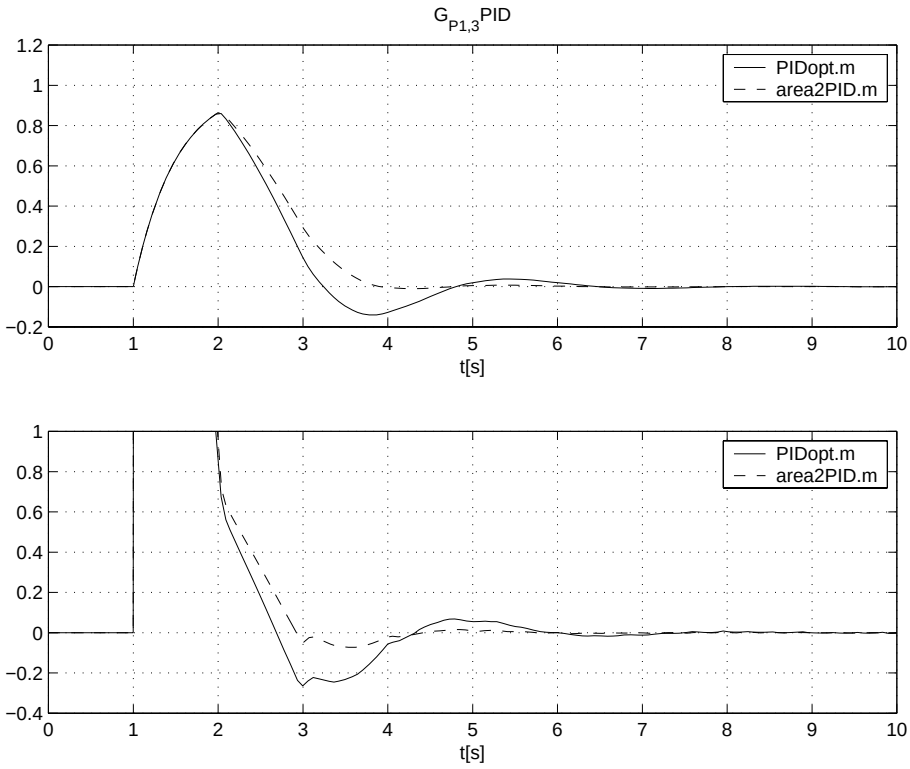


Fig. 12. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

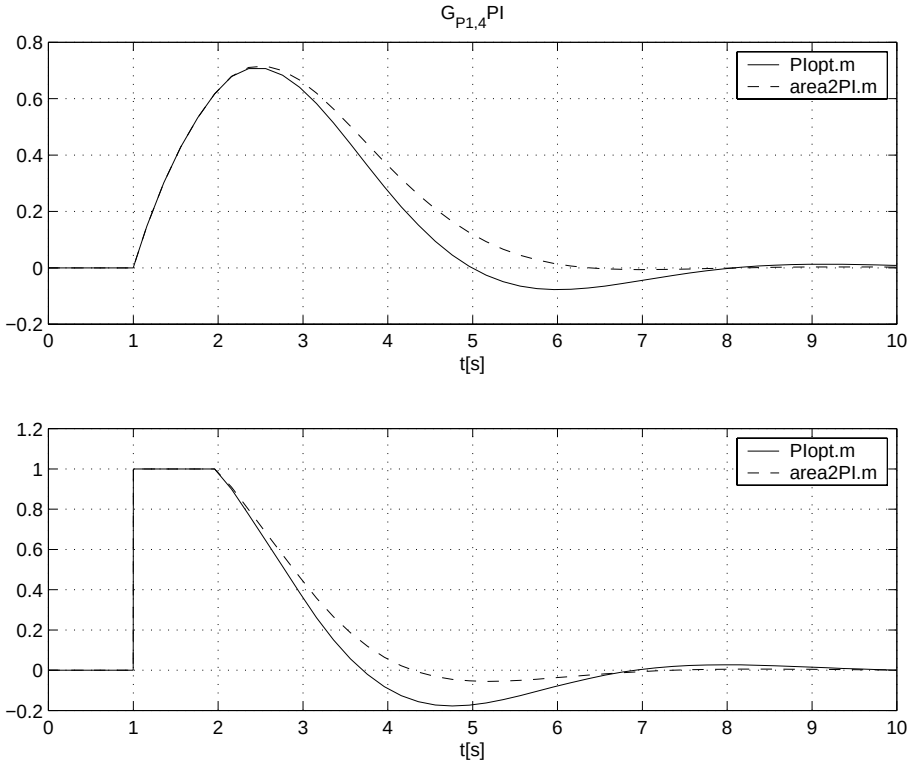


Fig. 13. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

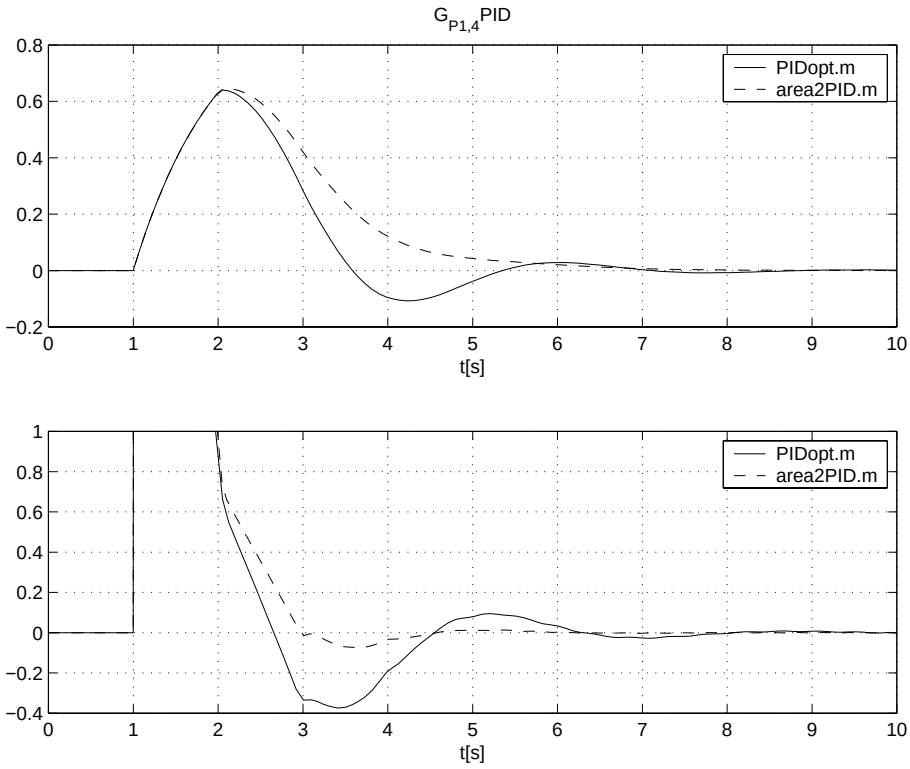


Fig. 14. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

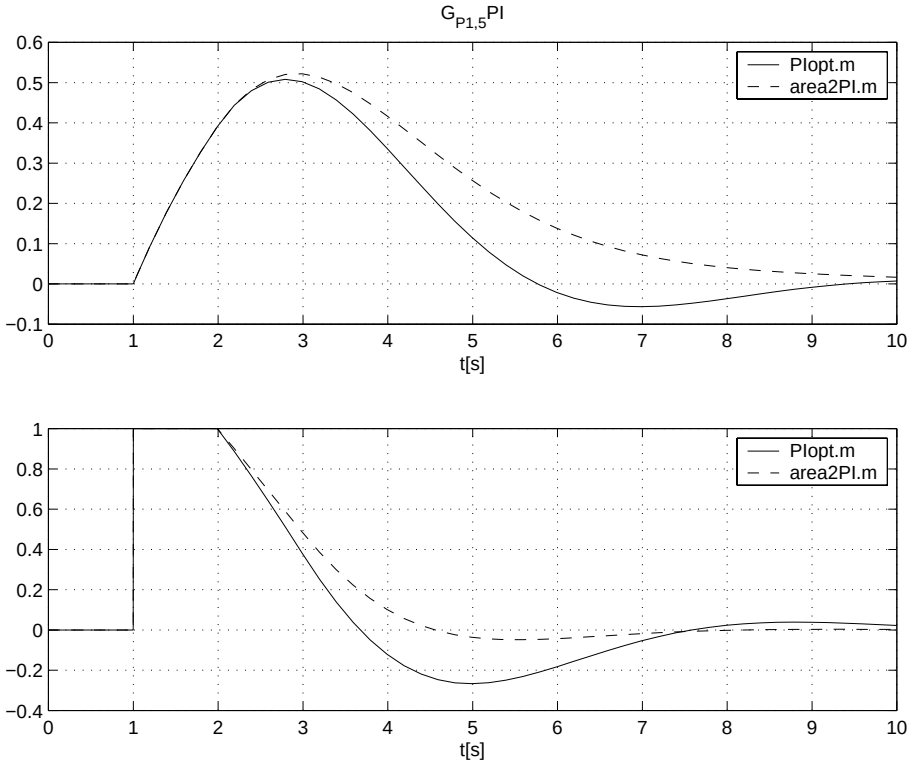


Fig. 15. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

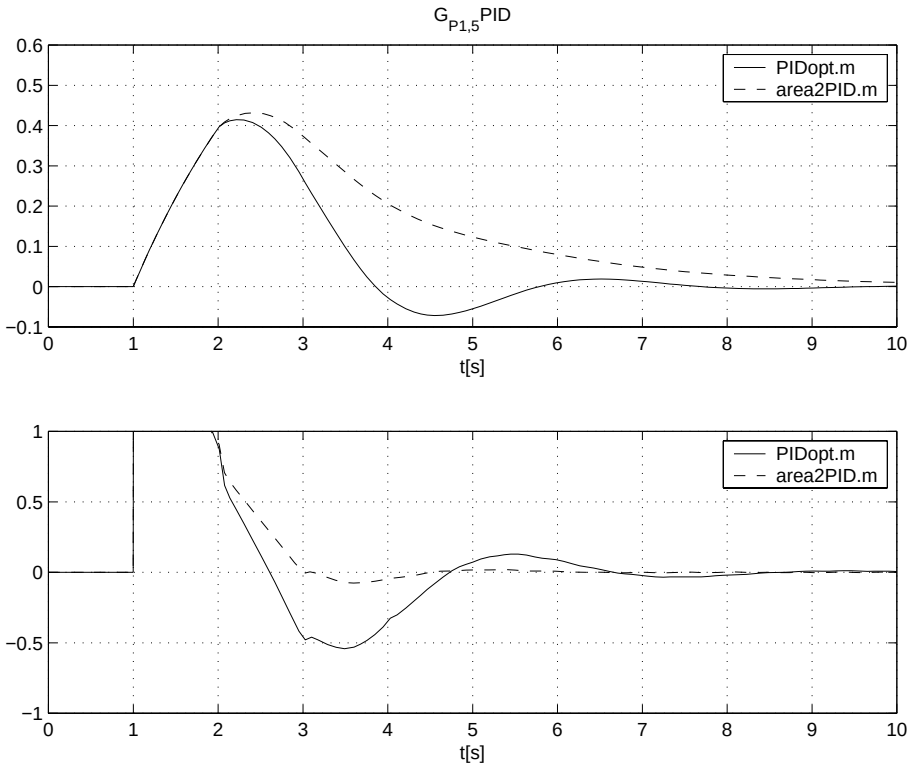


Fig. 16. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

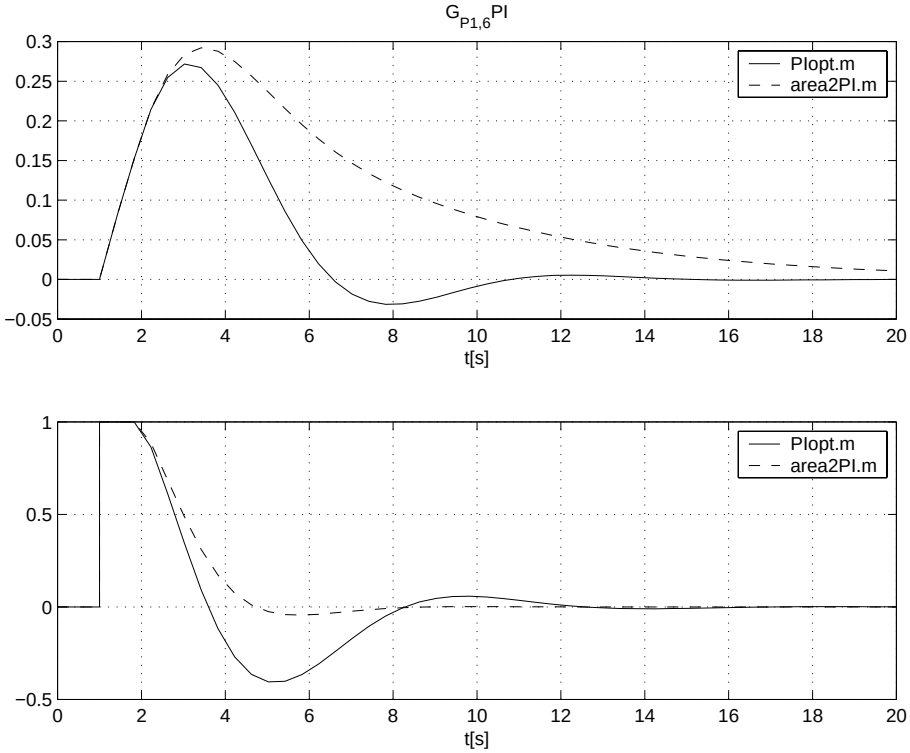


Fig. 17. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

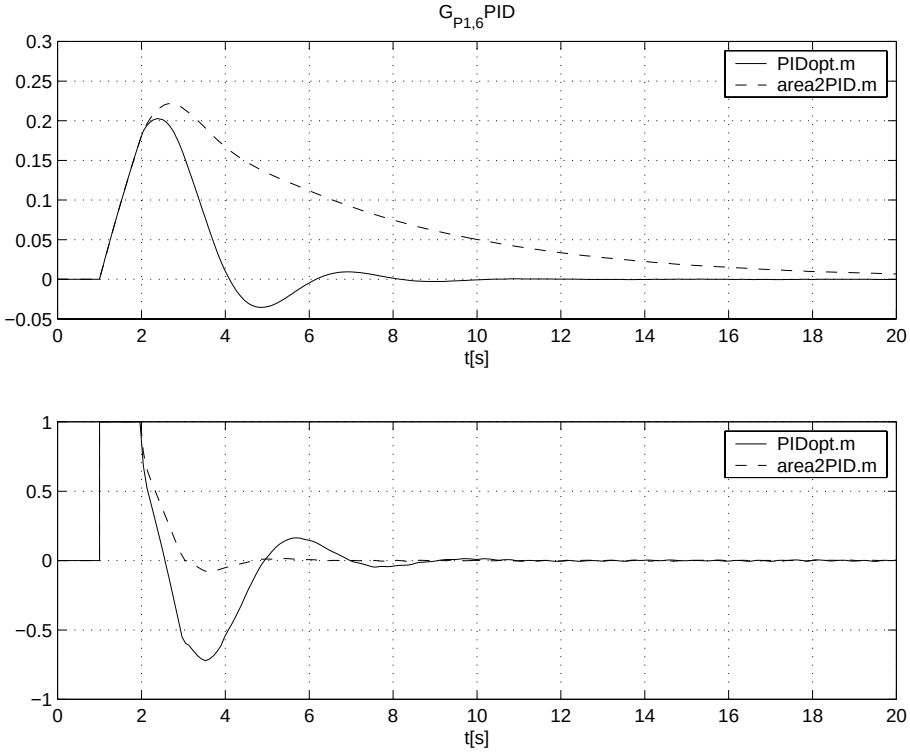


Fig. 18. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

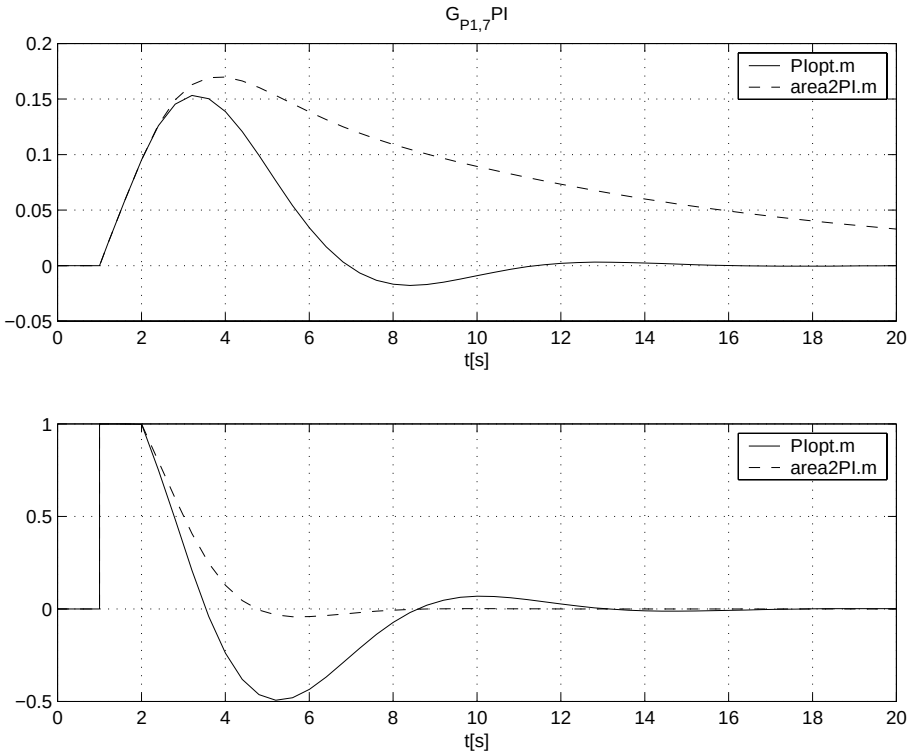


Fig. 19. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

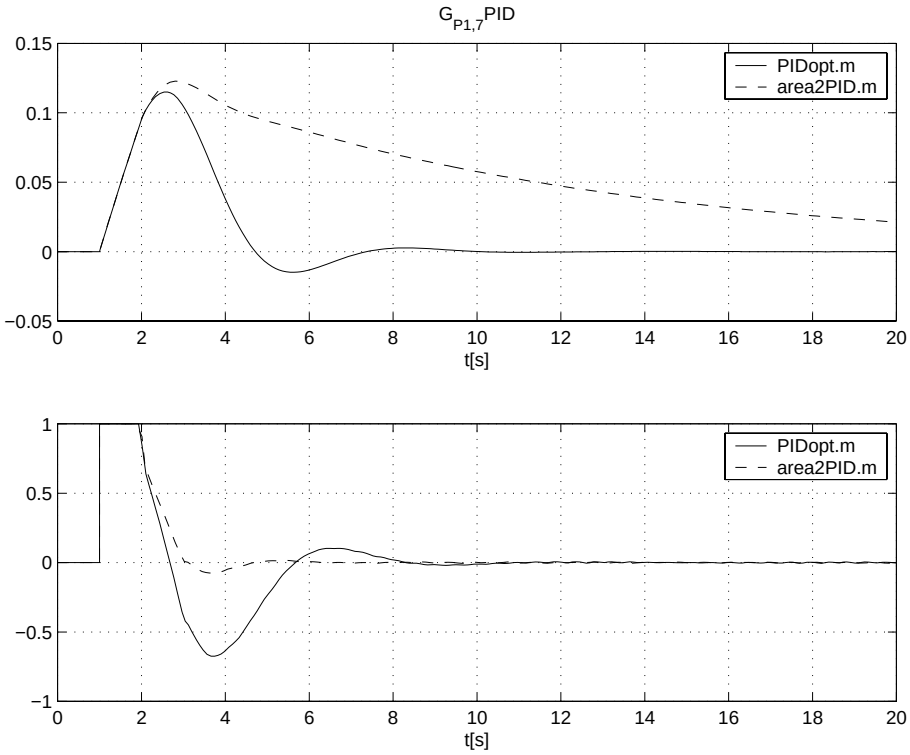


Fig. 20. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

Process 2: $G_{P2} = \dfrac{e^{-s}}{(Ts+1)^2}$

Table 5. PI controller parameters calculated by using DRMO method (PIopt.m)

T	0.1	0.2	0.5	1	2	5	10
K	0.28017	0.30615	0.39925	0.52596	0.67386	0.83417	0.90922
K_i	0.68285	0.60929	0.48947	0.38809	0.28018	0.15292	0.086789

Table 6. PI controller parameters calculated by using MO method (area2PI.m)

T	0.1	0.2	0.5	1	2	5	10
K	0.2613	0.28519	0.37097	0.49	0.63532	0.80504	0.89041
K_i	0.63441	0.56085	0.43548	0.33	0.22706	0.11864	0.06621

Table 7. PID controller parameters calculated by using DRMO method (PIDopt.m)

T	0.1	0.2	0.5	1	2	5	10
K	0.53421	0.6074	0.91052	1.5458	3.1847	10	10
K_i	0.90922	0.84558	0.79859	0.86365	1.1617	2.3984	1.2407
K_d	0.094402	0.1278	0.28533	0.75204	2.5374	14.225	27.761

Table 8. PID controller parameters calculated by using MO method (area2PID.m)

T	0.1	0.2	0.5	1	2	5	10
K	0.46478	0.52193	0.74857	1.1829	2.1192	5.0577	10
K_i	0.80398	0.72995	0.62429	0.56098	0.52383	0.50525	0.5
K_d	0.081299	0.10983	0.24387	0.64161	2.1565	12.664	49.996

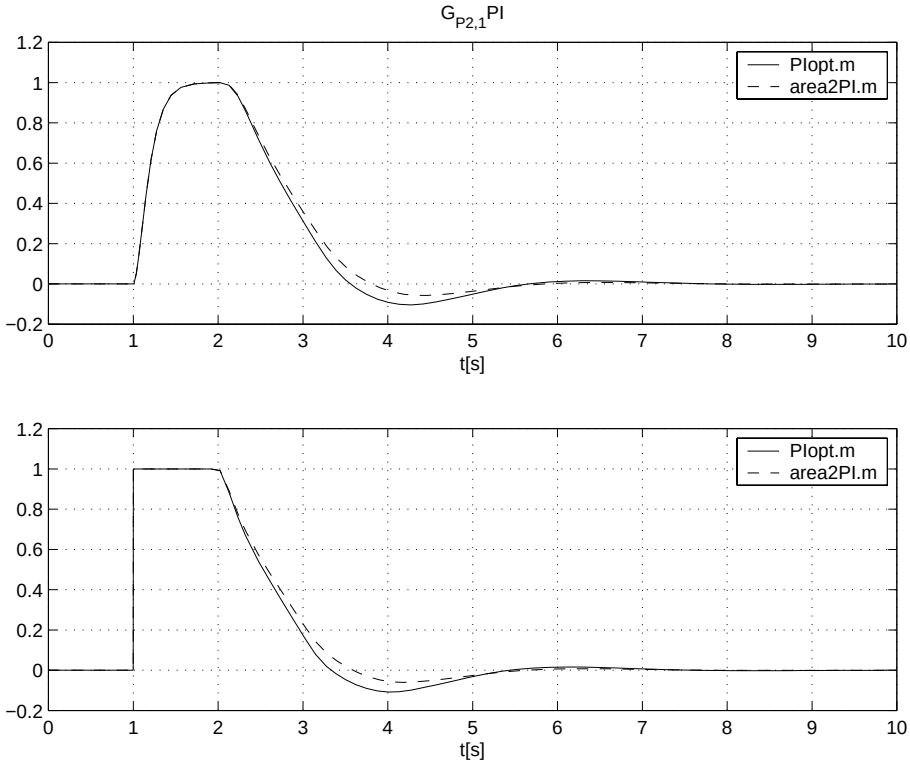


Fig. 21. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

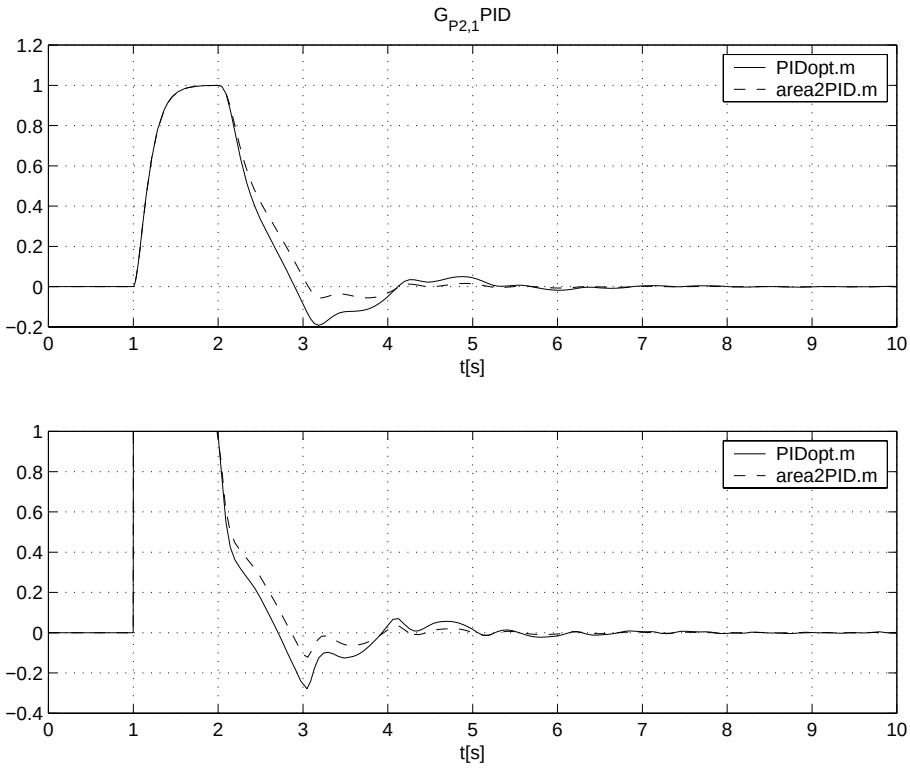


Fig. 22. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

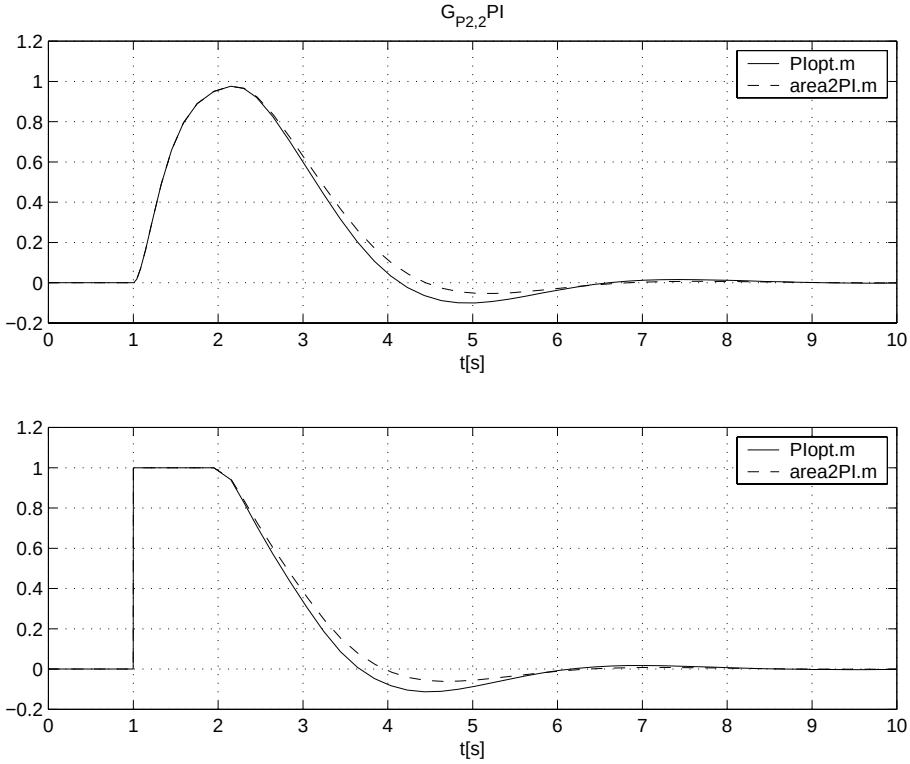


Fig. 23. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

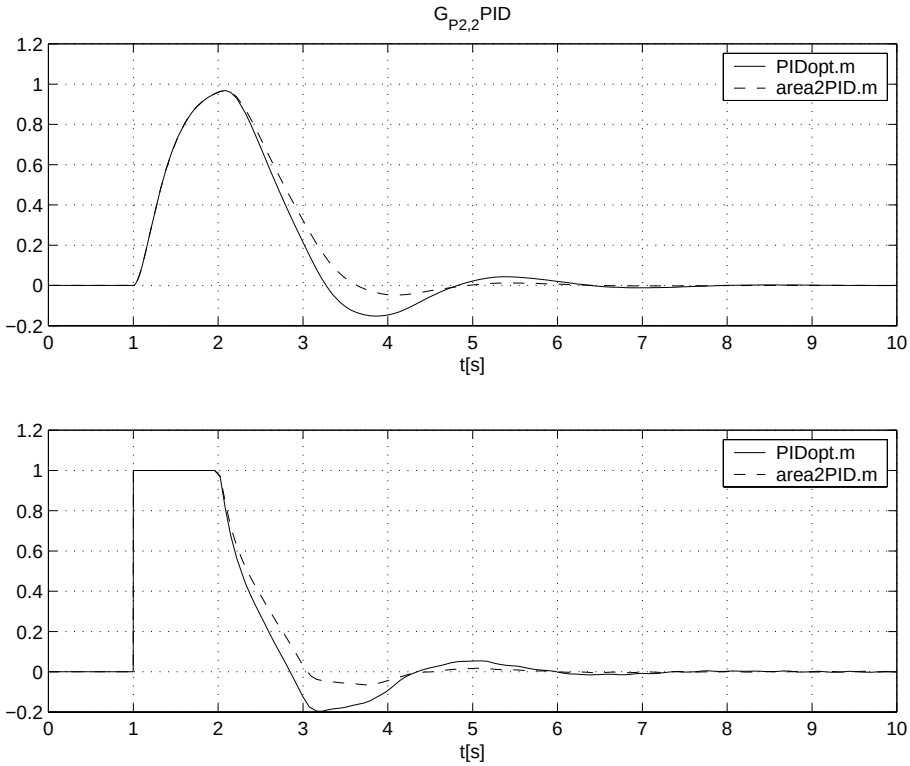


Fig. 24. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

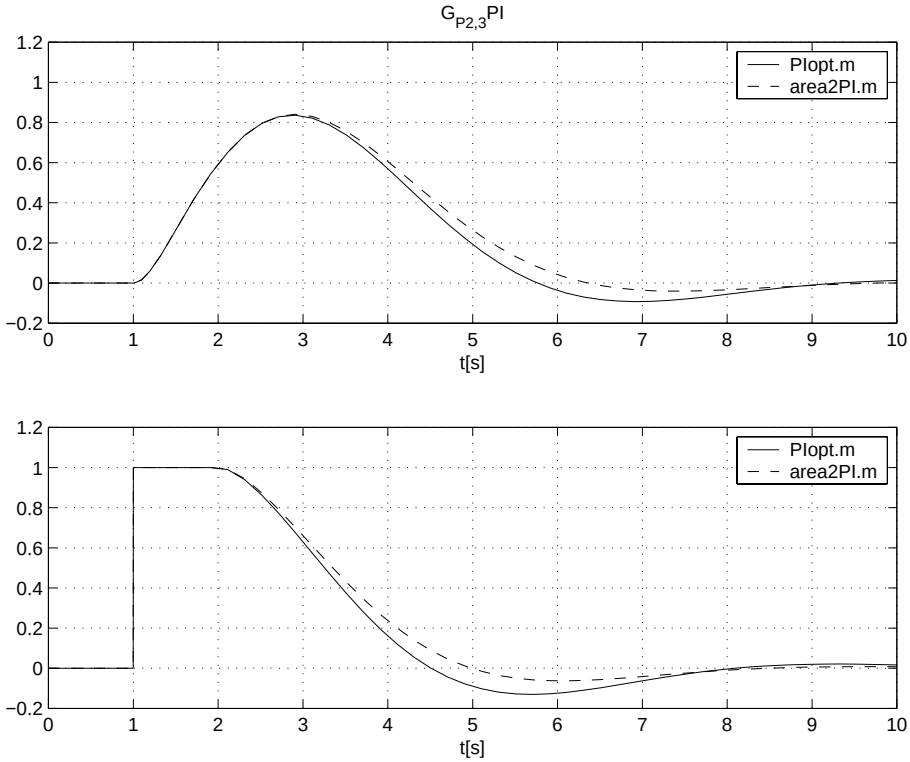


Fig. 25. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

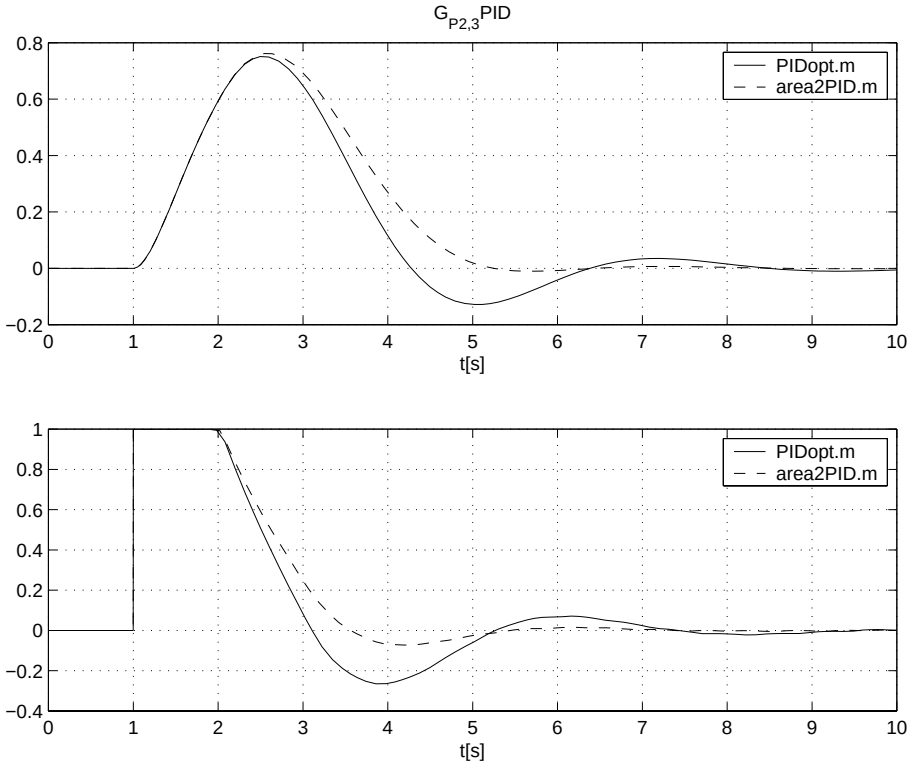


Fig. 26. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

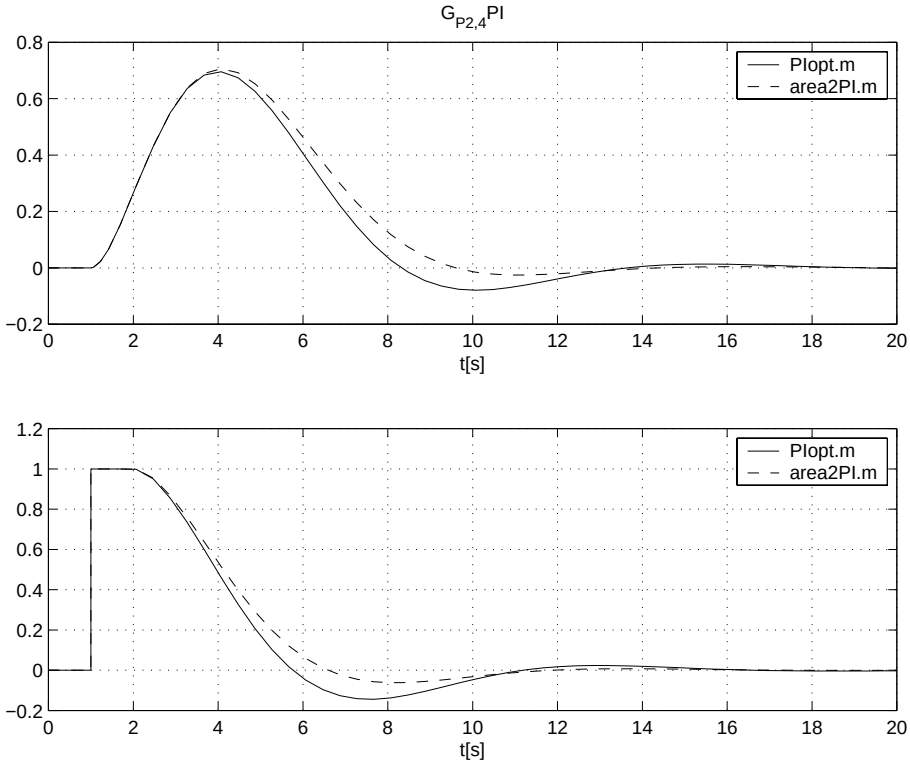


Fig. 27. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

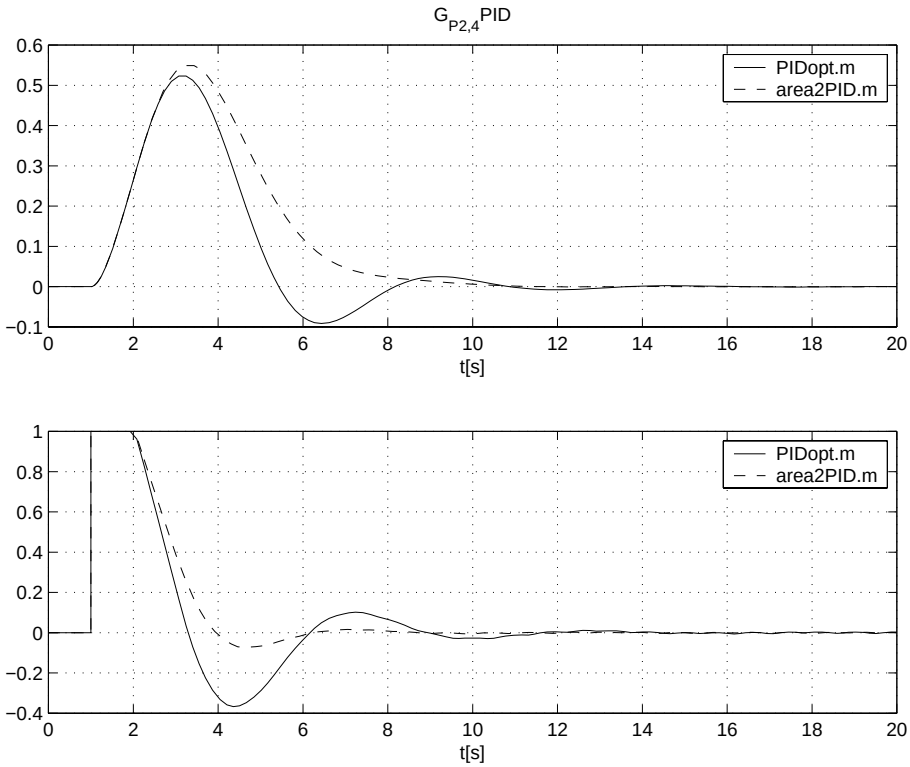


Fig. 28. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

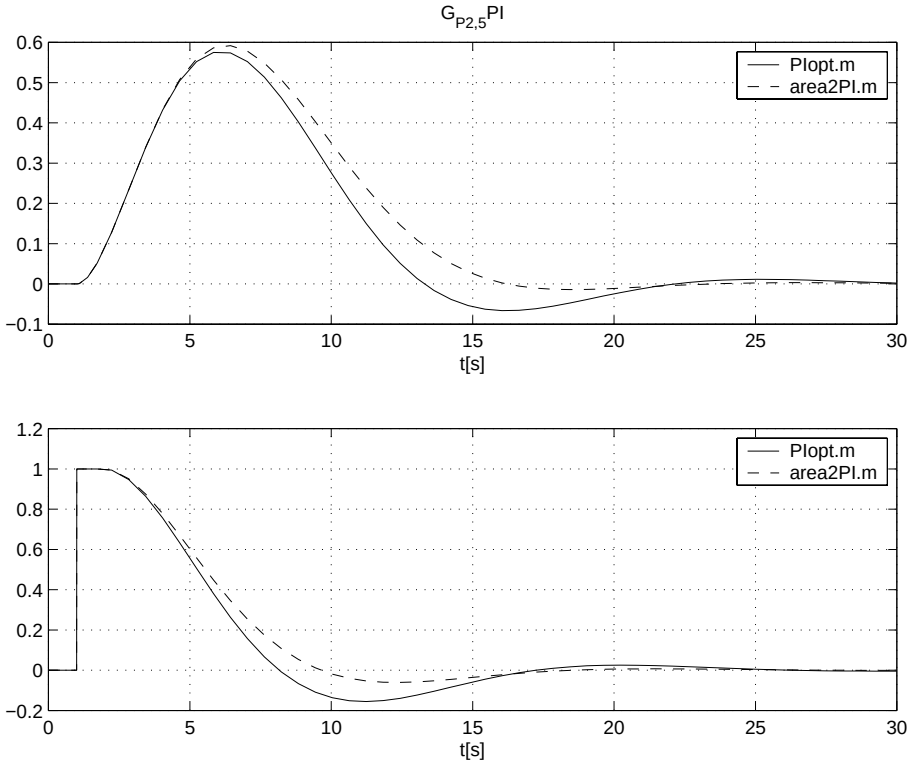


Fig. 29. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

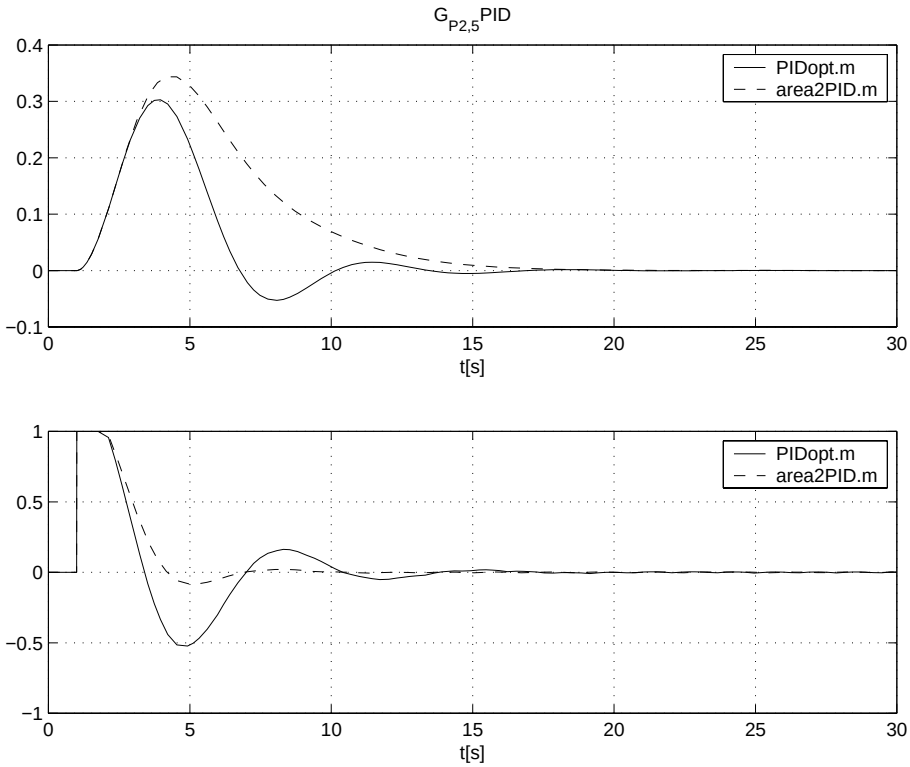


Fig. 30. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

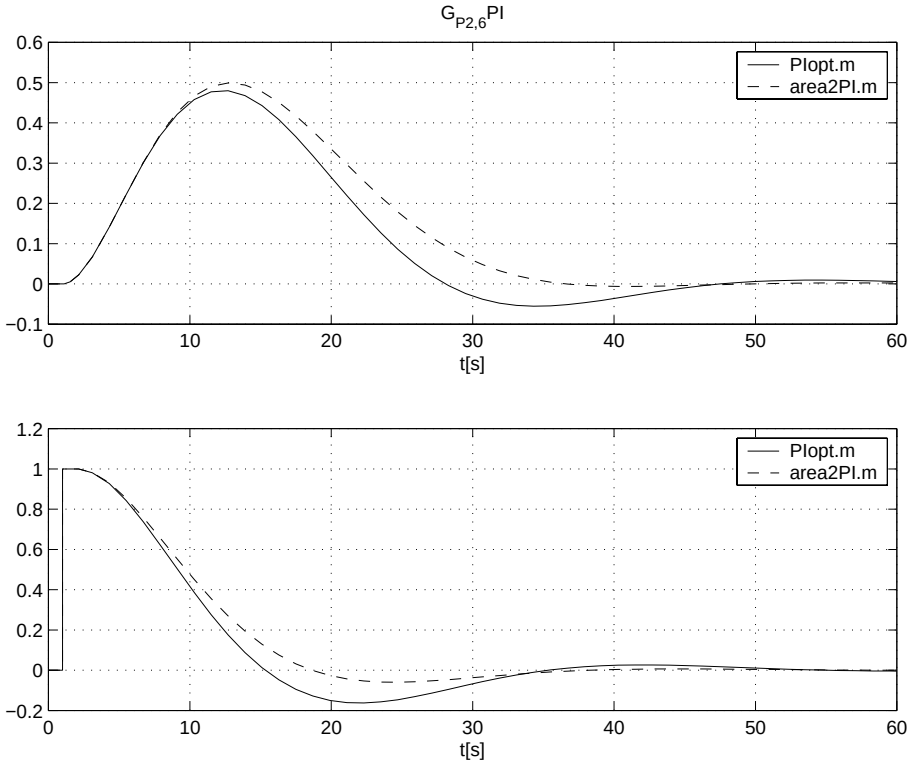


Fig. 31. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

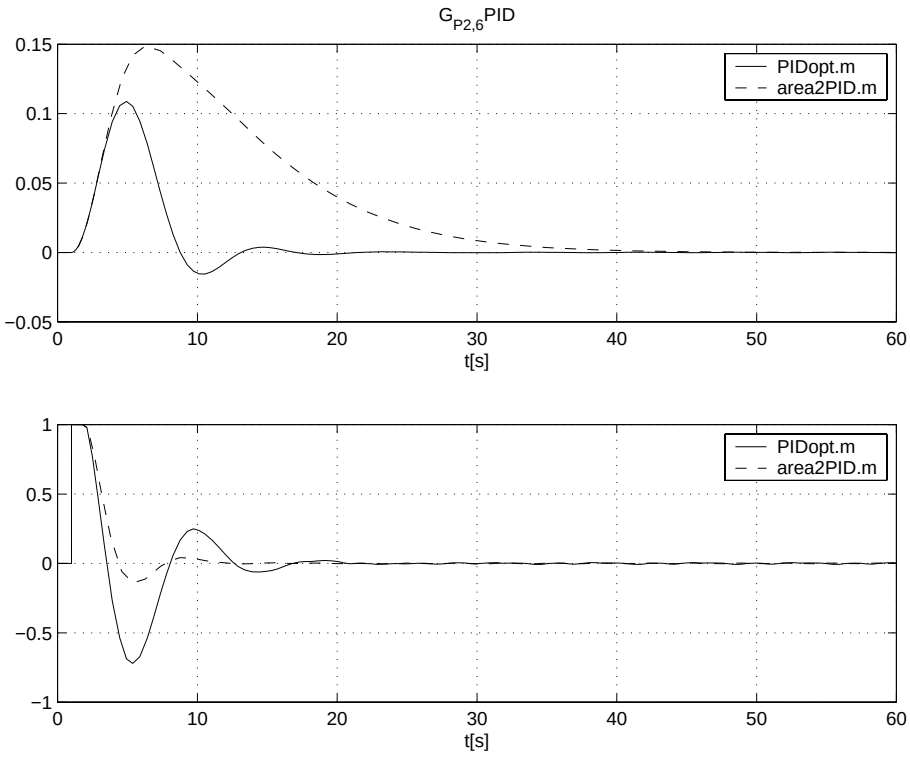


Fig. 32. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

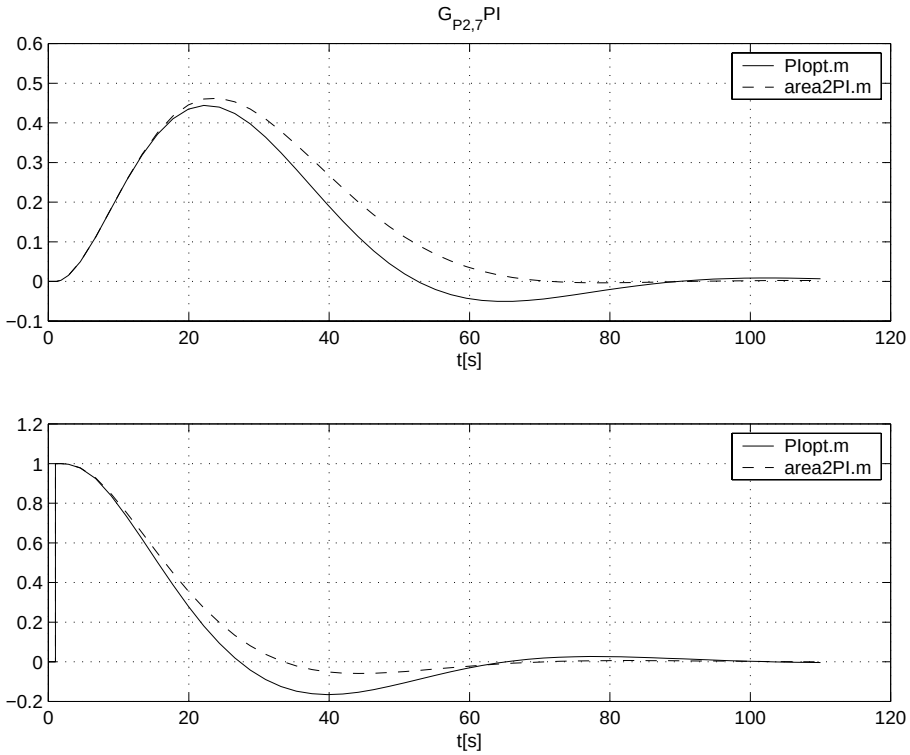


Fig. 33. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

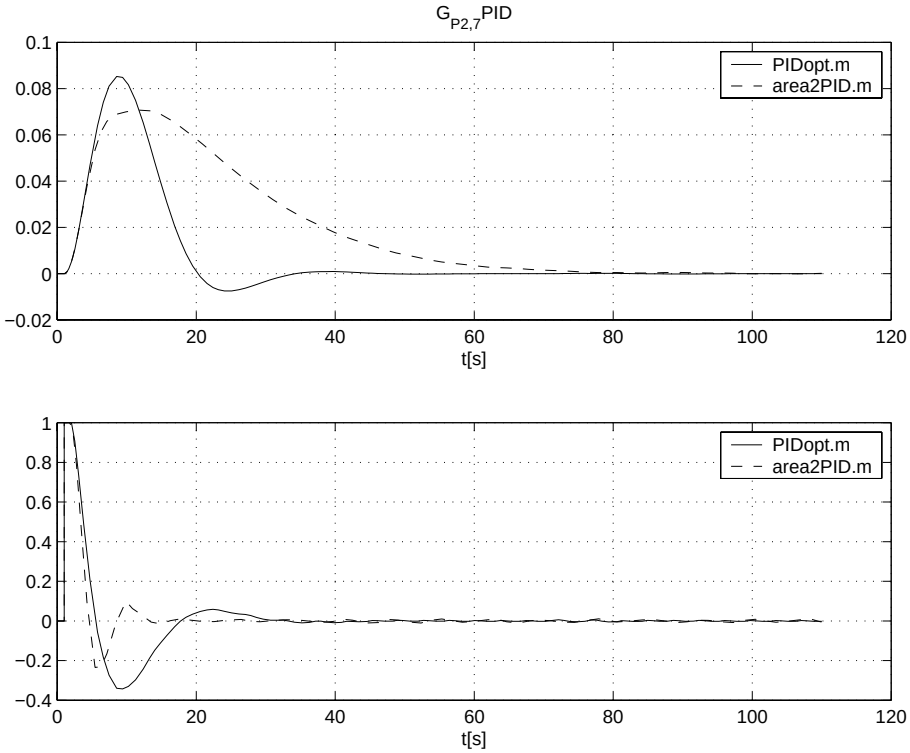


Fig. 34. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

Process 3: $G_{P3} = \dfrac{1}{(s+1)(Ts+1)}$

Table 9. PI controller parameters calculated by using DRMO method (PIopt.m)

T	0.1	0.2	0.5	1	2	5	10
K	5.05	2.6	1.25	1	1.25	2.6	5.05
K_i	16.637	5.4	1.6875	1	0.84375	1.08	1.6638

Table 10. PI controller parameters calculated by using MO method (area2PI.m)

T	0.1	0.2	0.5	1	2	5	10
K	5.05	2.6	1.25	1	1.25	2.6	5.05
K_i	5.0455	2.5833	1.1667	0.75	0.58333	0.51667	0.50455

Table 11. PID controller parameters calculated by using DRMO method (PIDopt.m)

T	0.1	0.2	0.5	1	2	5	10
K	10	10	10	10	10	10	10
K_i	40.789	28.842	18.241	12.899	9.1207	5.7684	4.0789
K_d	0.38324	0.89762	1.8166	2.6904	3.6332	4.4881	3.8324

Table 12. PID controller parameters calculated by using MO method (area2PID.m)

T	0.1	0.2	0.5	1	2	5	10
K	10	10	10	10	10	10	10
K_i	9.5455	8.75	7	5.25	3.5	1.75	0.95455
K_d	0.45	1.2333	2.9167	4.5	5.8333	6.1667	4.5

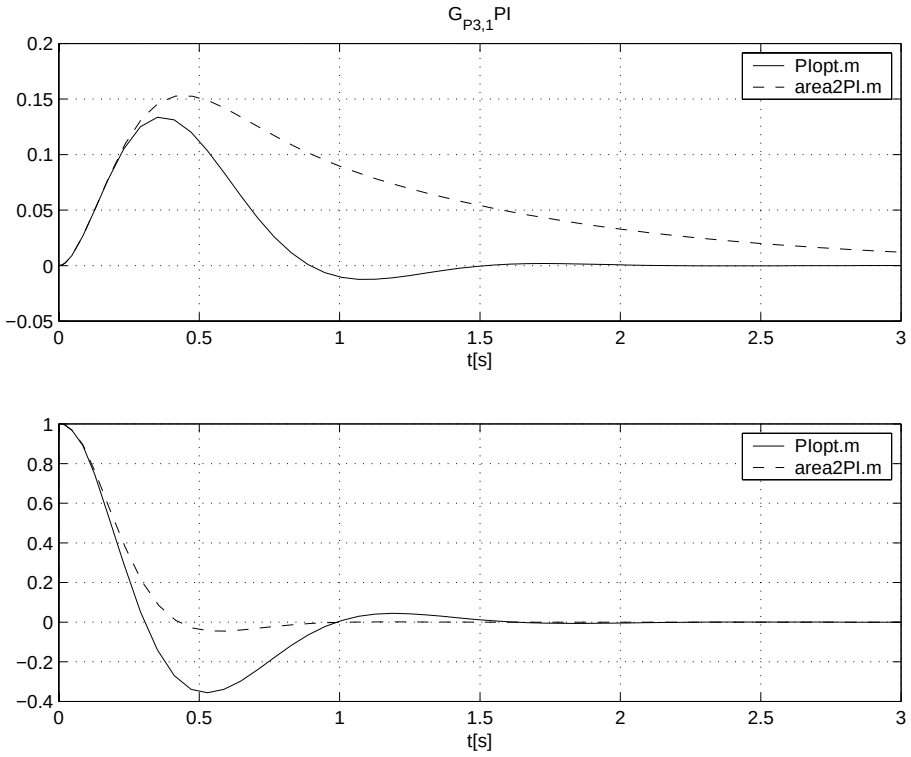


Fig. 35. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

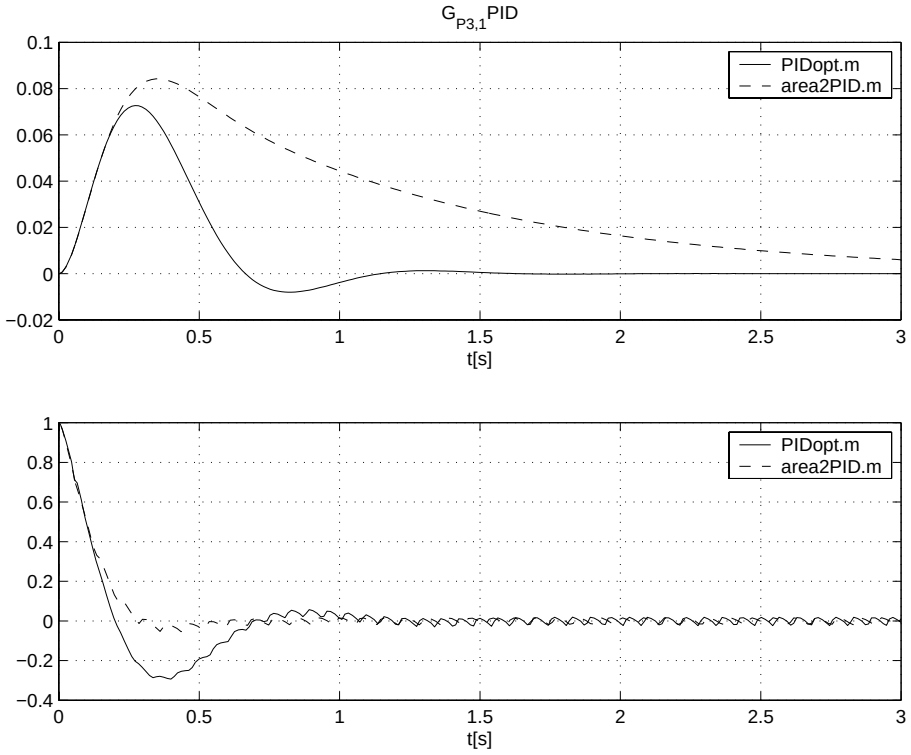


Fig. 36. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

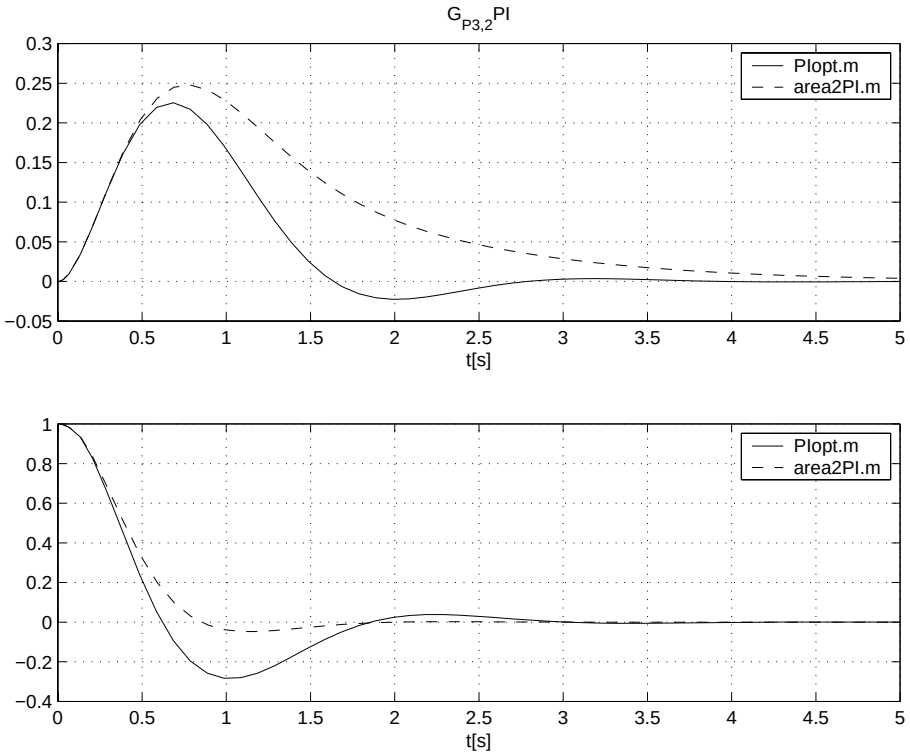


Fig. 37. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

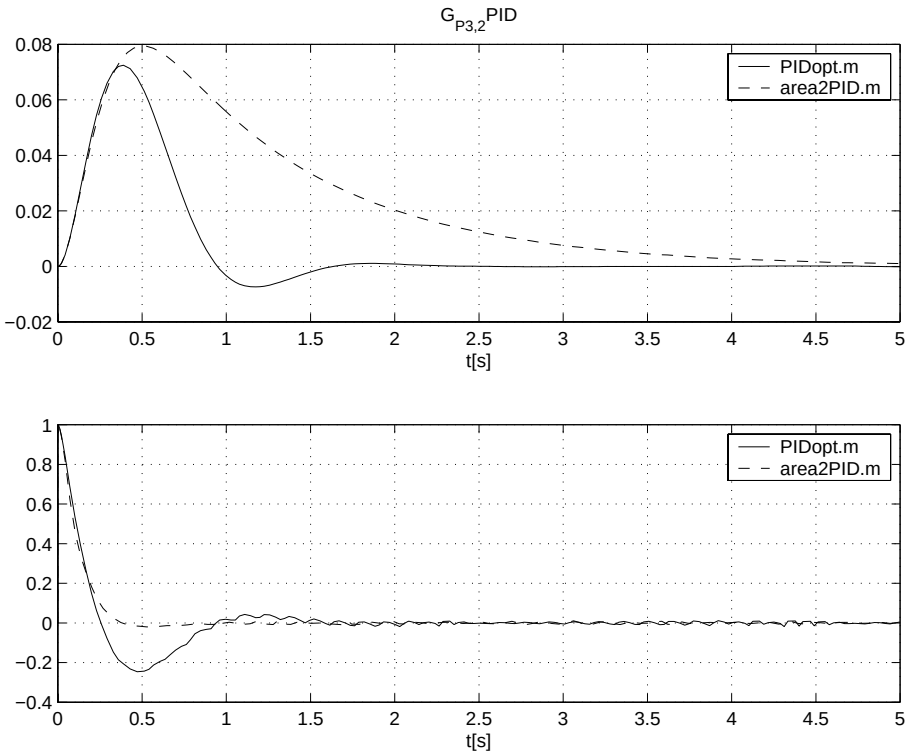


Fig. 38. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

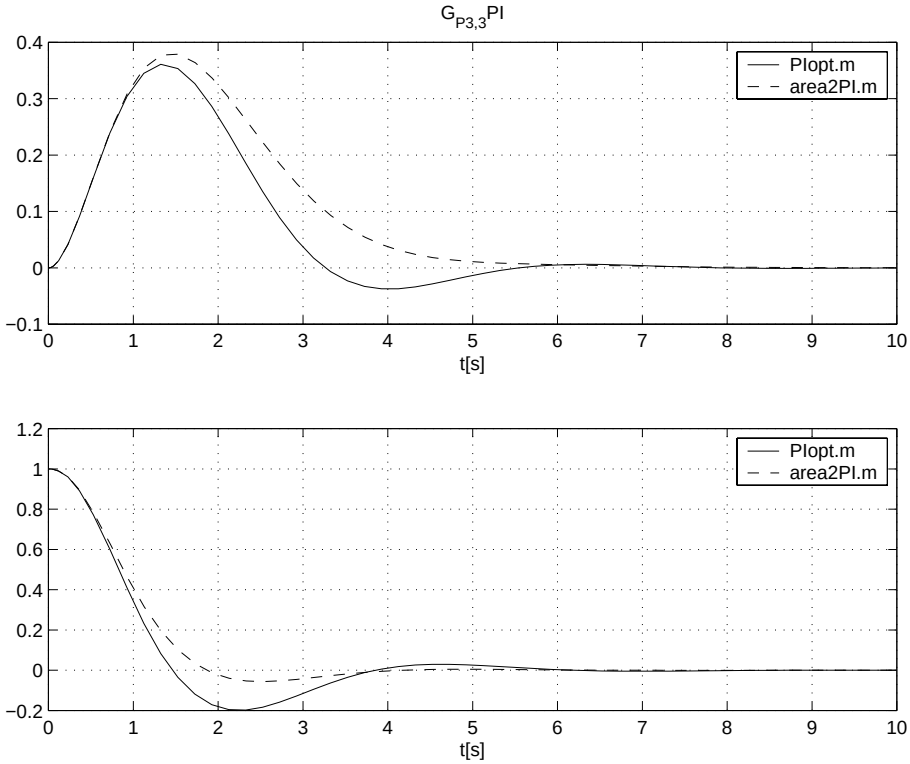


Fig. 39. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

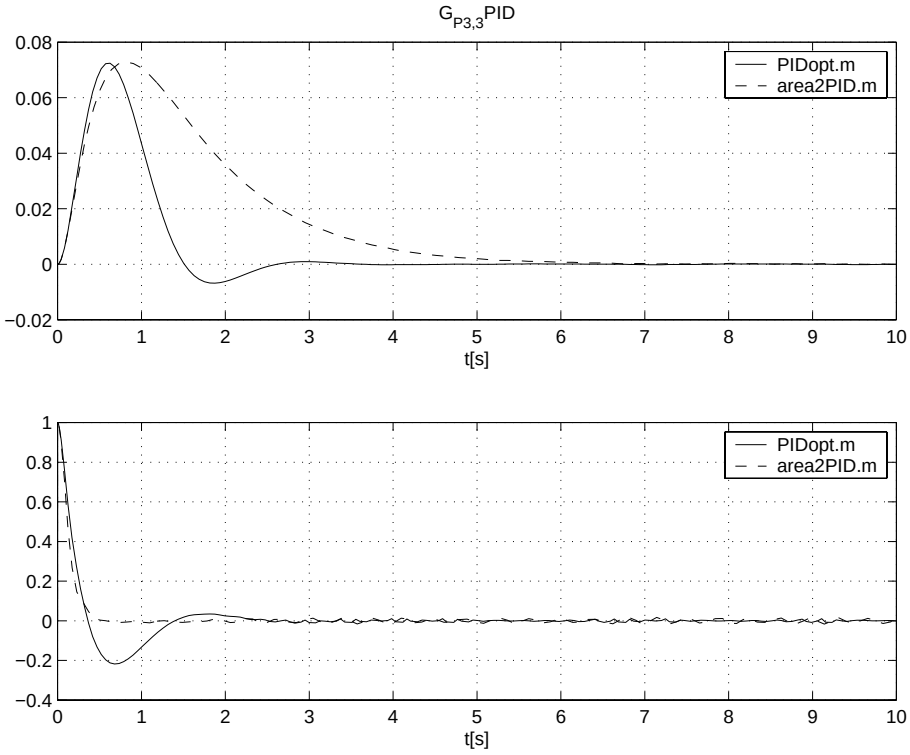


Fig. 40. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

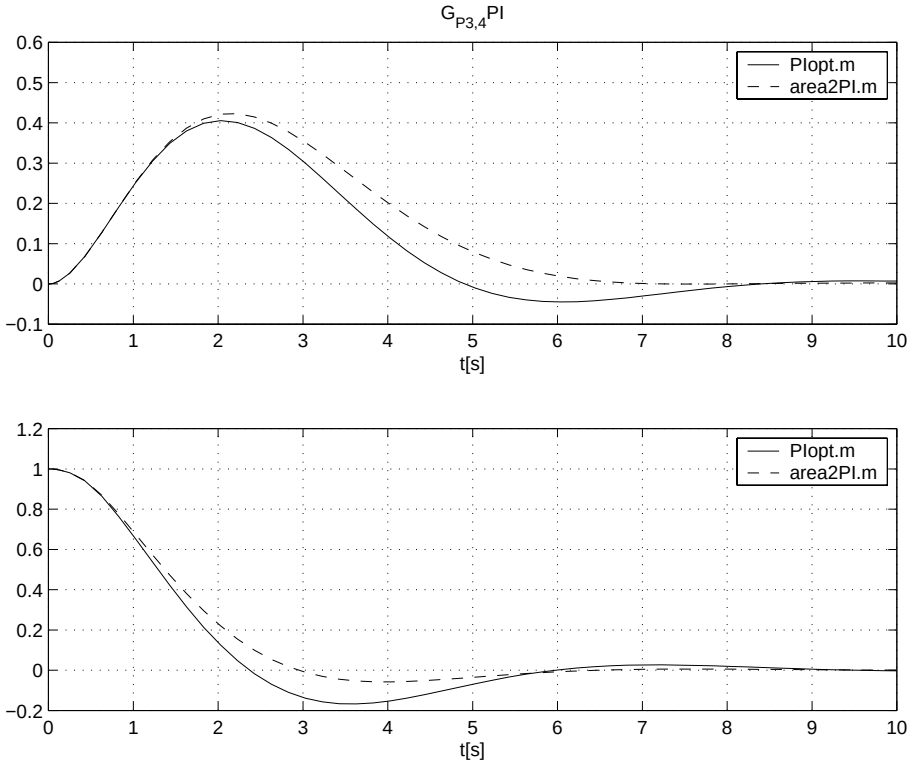


Fig. 41. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

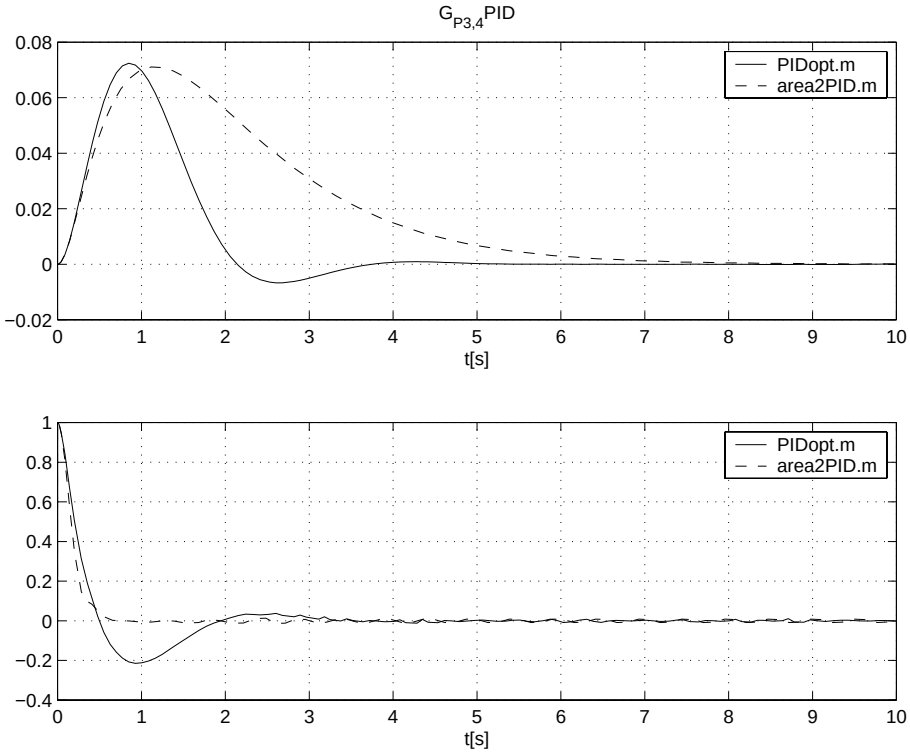


Fig. 42. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

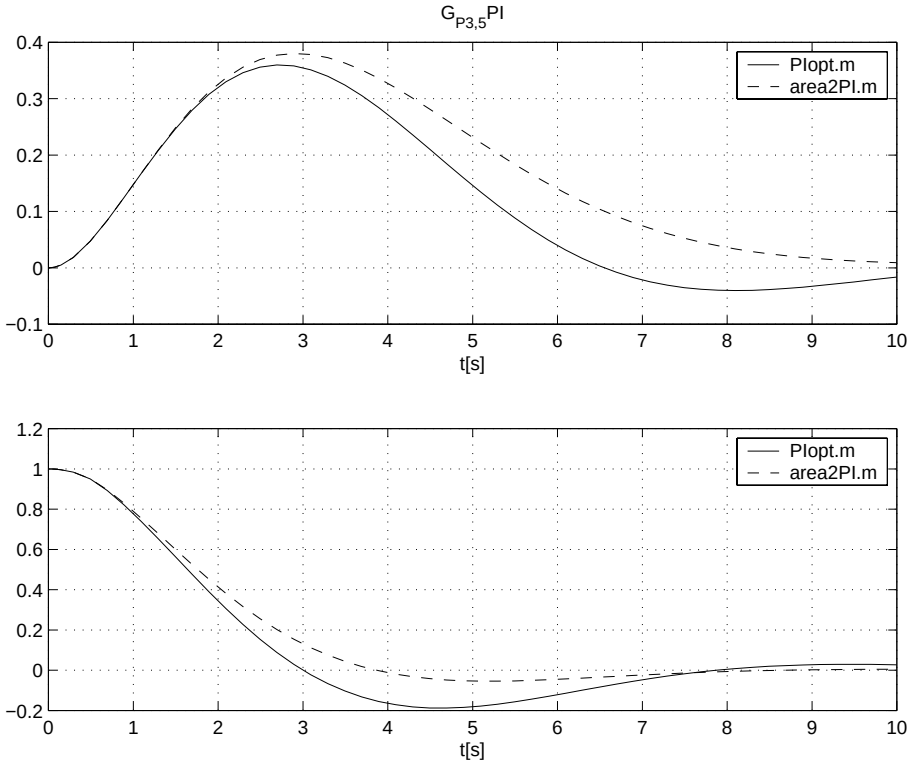


Fig. 43. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

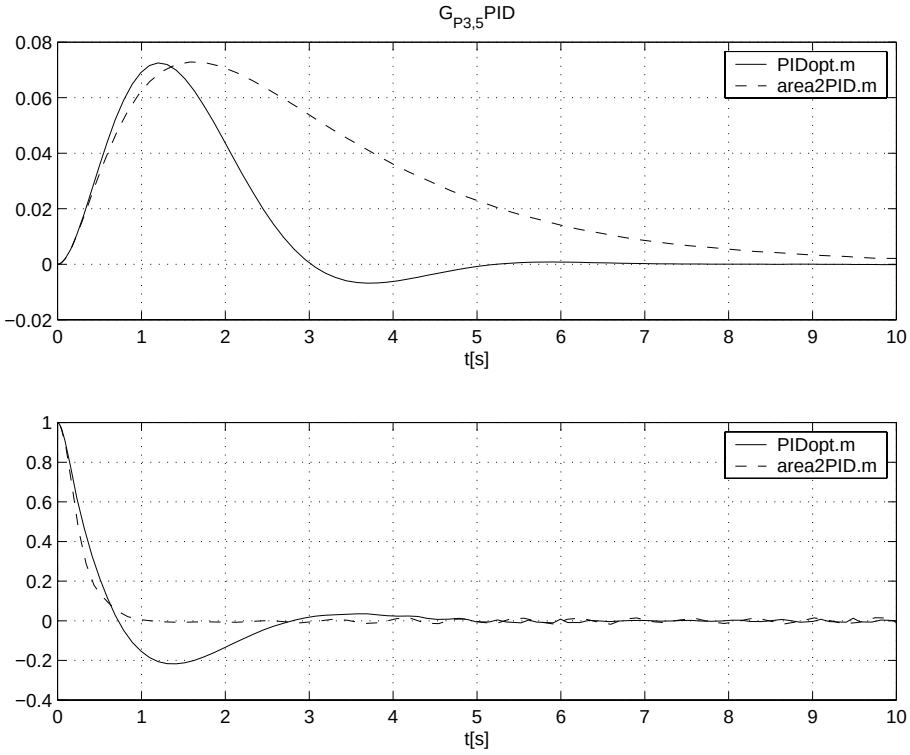


Fig. 44. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

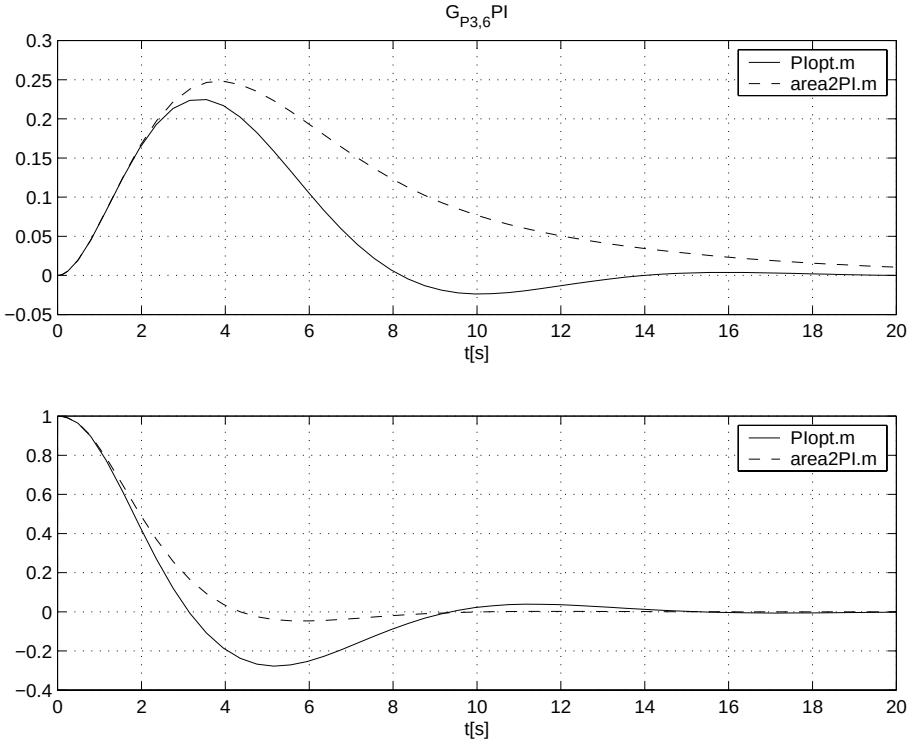


Fig. 45. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

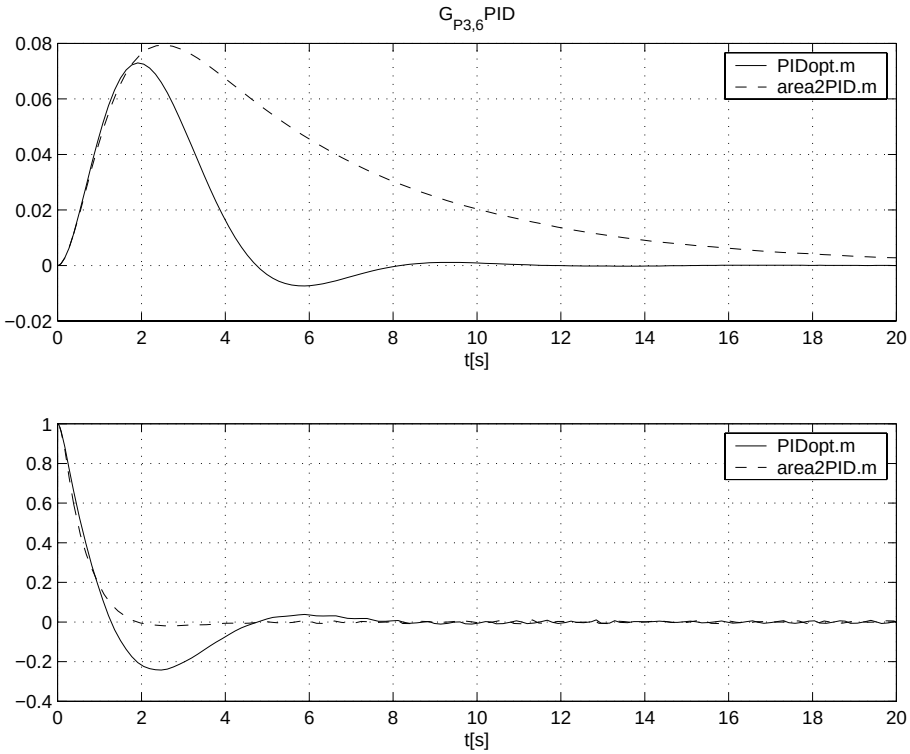


Fig. 46. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

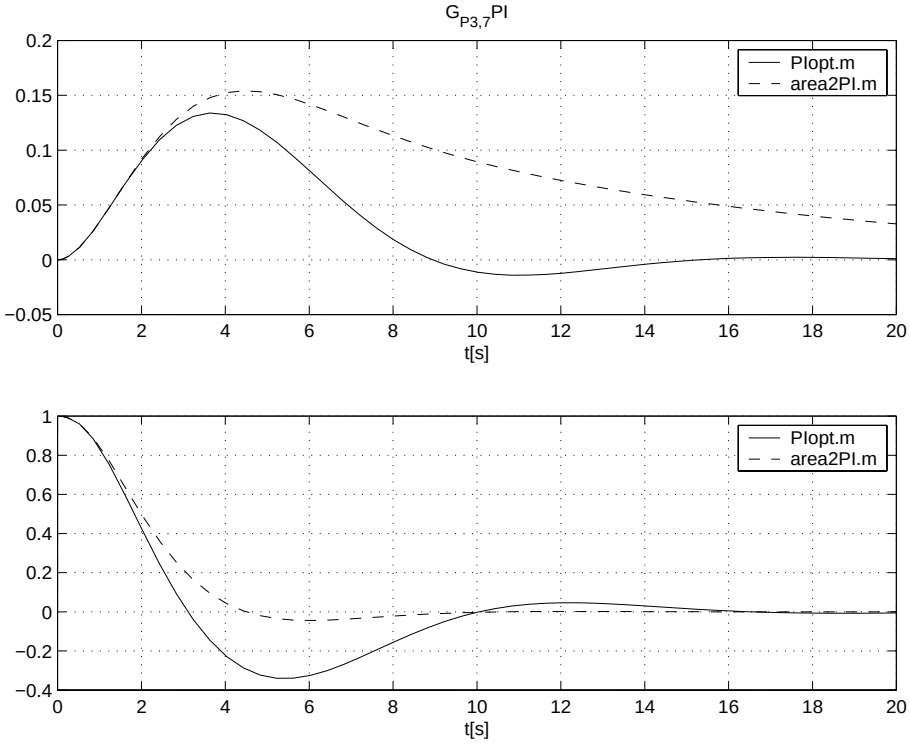


Fig. 47. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

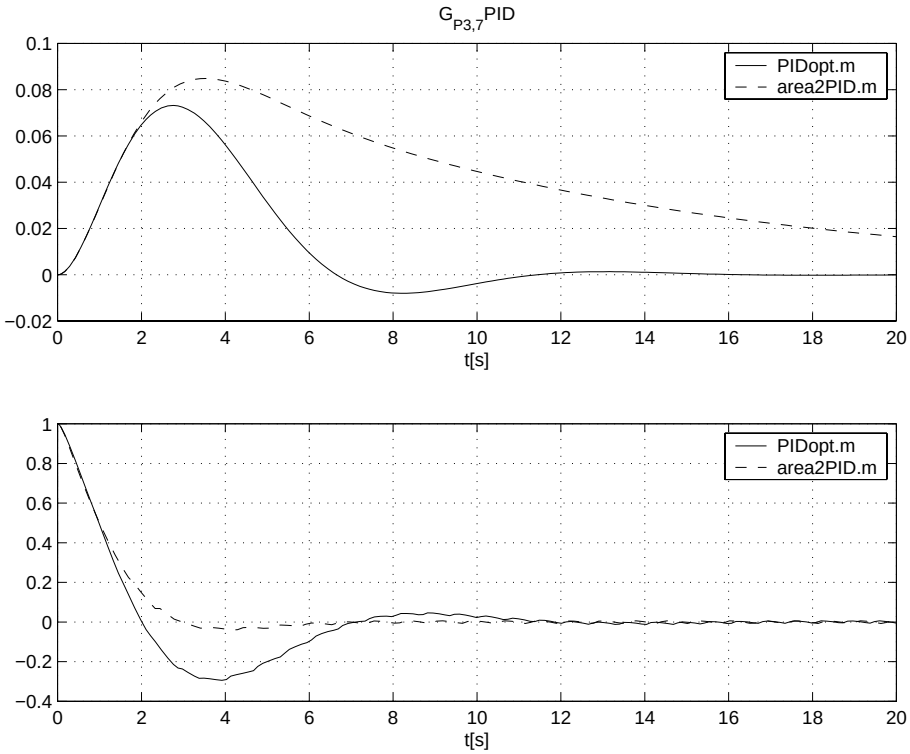


Fig. 48. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

Process 4 : $G_{P4} = \dfrac{1}{(s+1)^2(Ts+1)^2}$

Table 13. PI controller parameters calculated by using DRMO method (PIopt.m)

T	0.1	0.2	0.5	1	2	5	10
K	0.83582	0.72738	0.57557	0.52786	0.57557	0.72738	0.83582
K_i	0.76597	0.62164	0.41374	0.2918	0.20687	0.12433	0.076597

Table 14. PI controller parameters calculated by using MO method (area2PI.m)

T	0.1	0.2	0.5	1	2	5	10
K	0.80916	0.69512	0.54545	0.5	0.54545	0.69512	0.80916
K_i	0.59507	0.49797	0.34848	0.25	0.17424	0.099593	0.059507

Table 15. PID controller parameters calculated by using DRMO method (PIDopt.m)

T	0.1	0.2	0.5	1	2	5	10
K	9.6365	4.1038	2.008	1.7028	2.008	4.1038	9.6365
K_i	11.309	3.2957	1.1248	0.70571	0.5624	0.65913	1.1309
K_d	2.8019	1.552	1.0222	1.1757	2.0443	7.7599	28.019

Table 16. PID controller parameters calculated by using MO method (area2PID.m)

T	0.1	0.2	0.5	1	2	5	10
K	5.174	2.8083	1.5833	1.375	1.5833	2.8083	5.174
K_i	2.5791	1.3785	0.69444	0.46875	0.34722	0.27569	0.25791
K_d	2.599	1.444	0.95139	1.0938	1.9028	7.2201	25.99

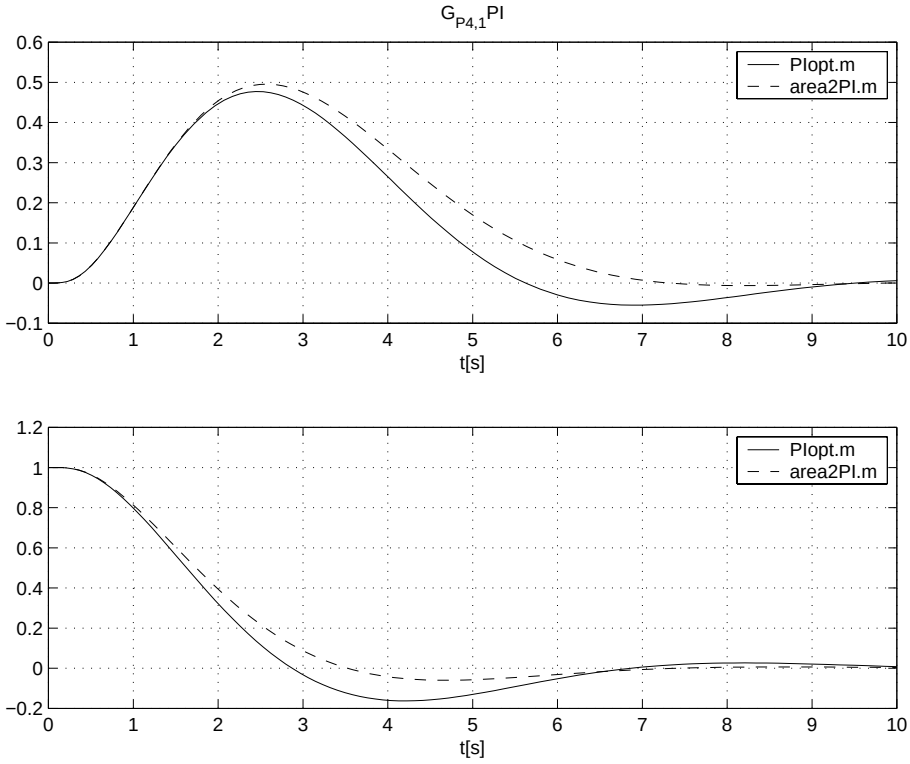


Fig. 49. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

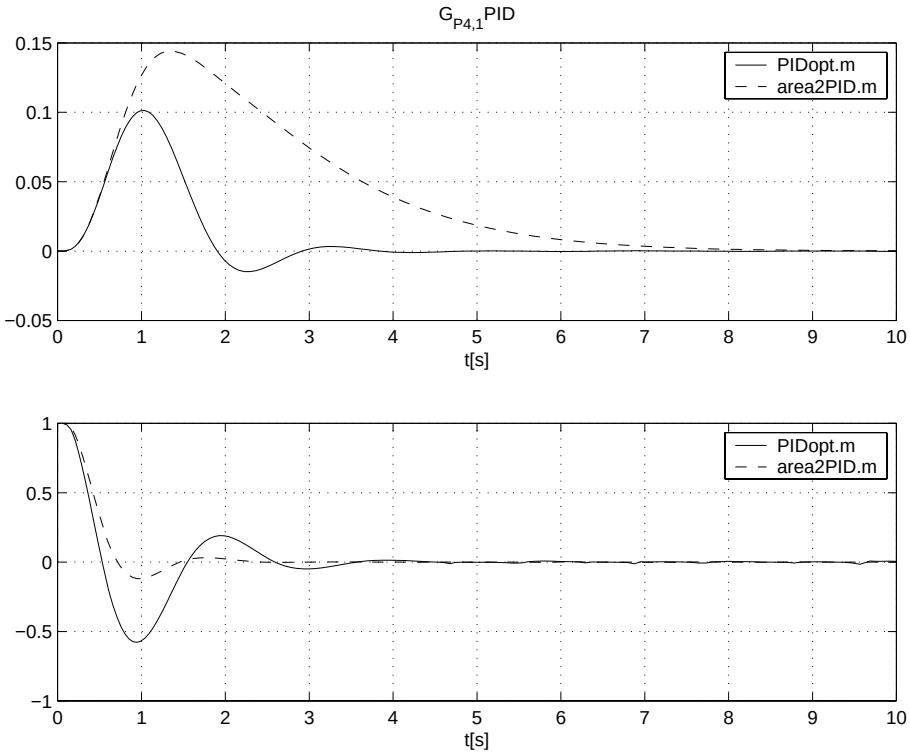


Fig. 50. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

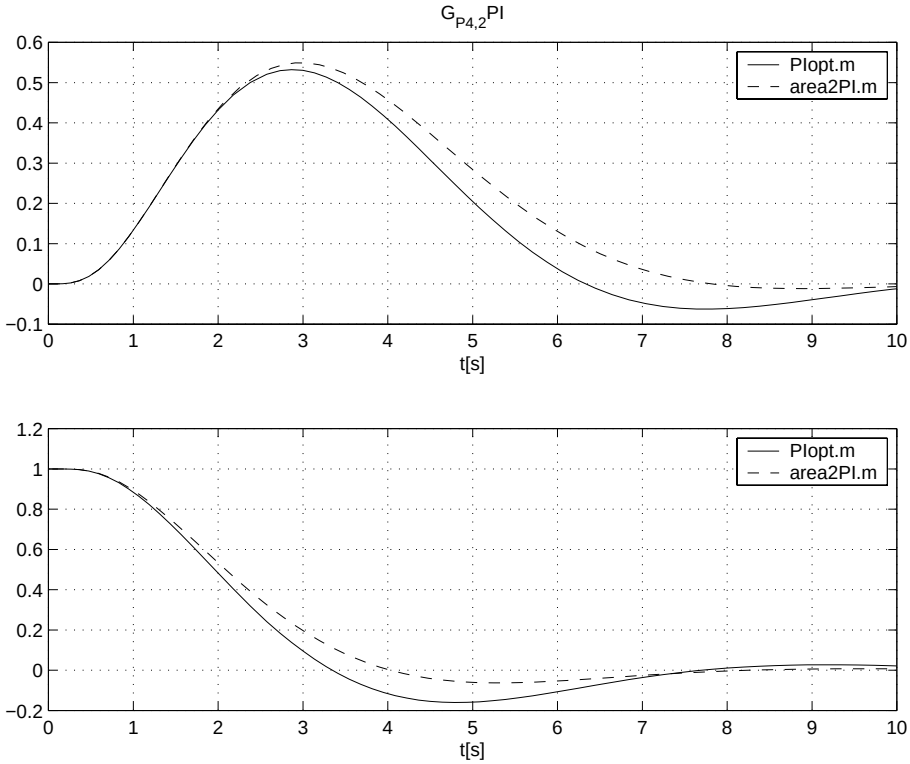


Fig. 51. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

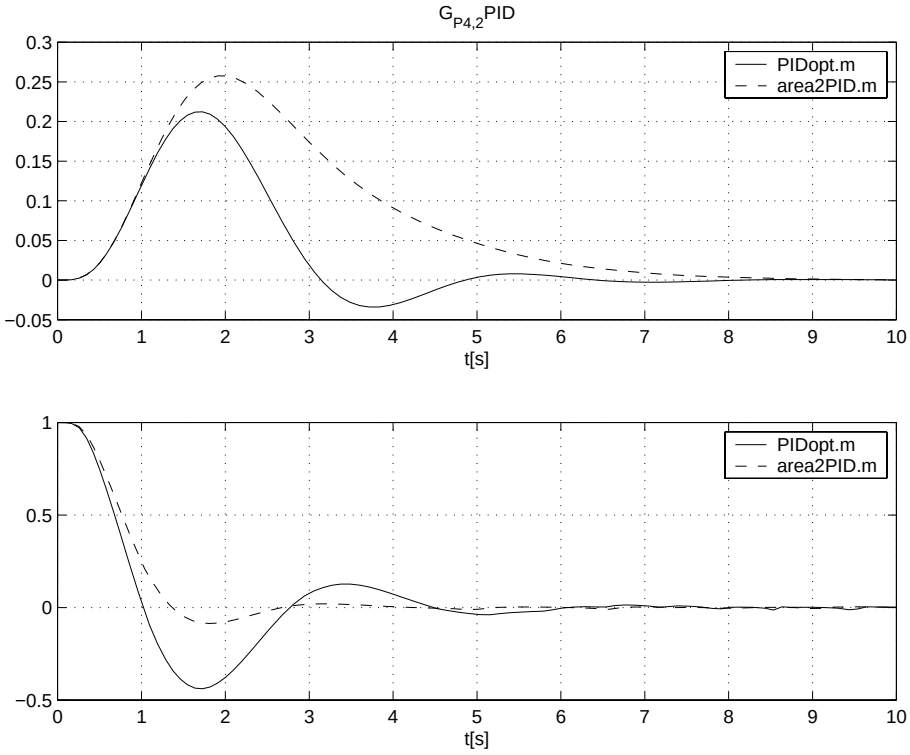


Fig. 52. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

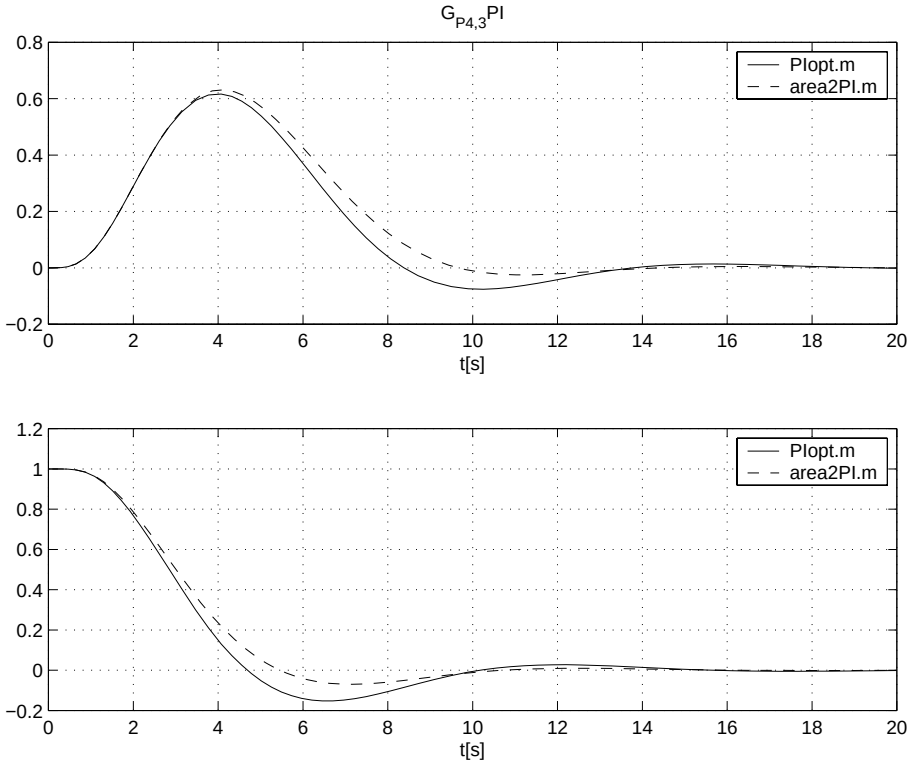


Fig. 53. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

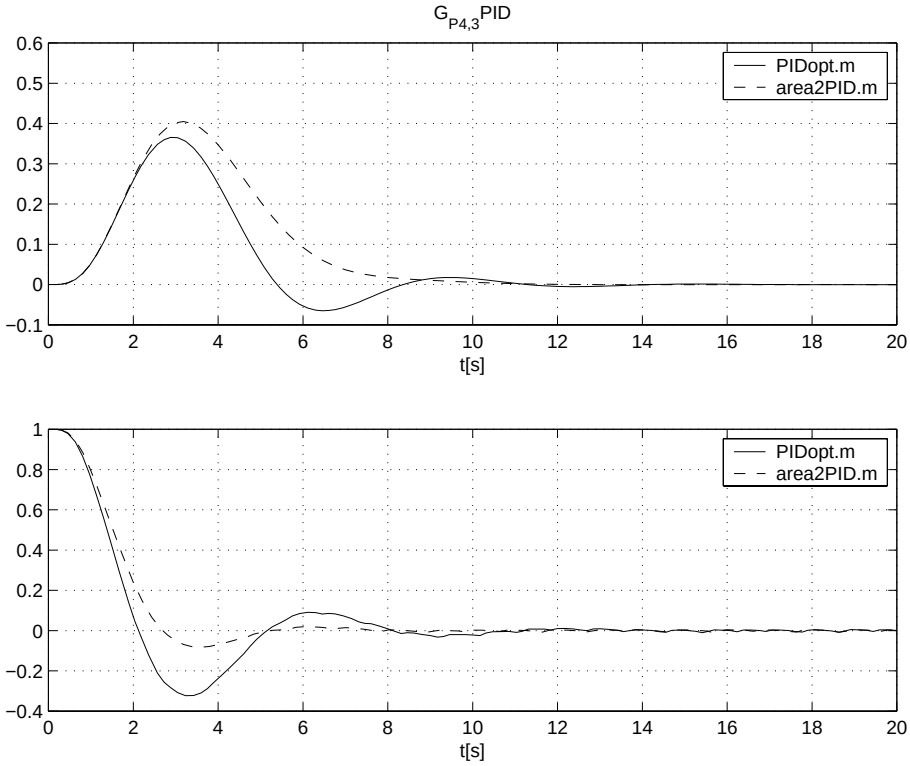


Fig. 54. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

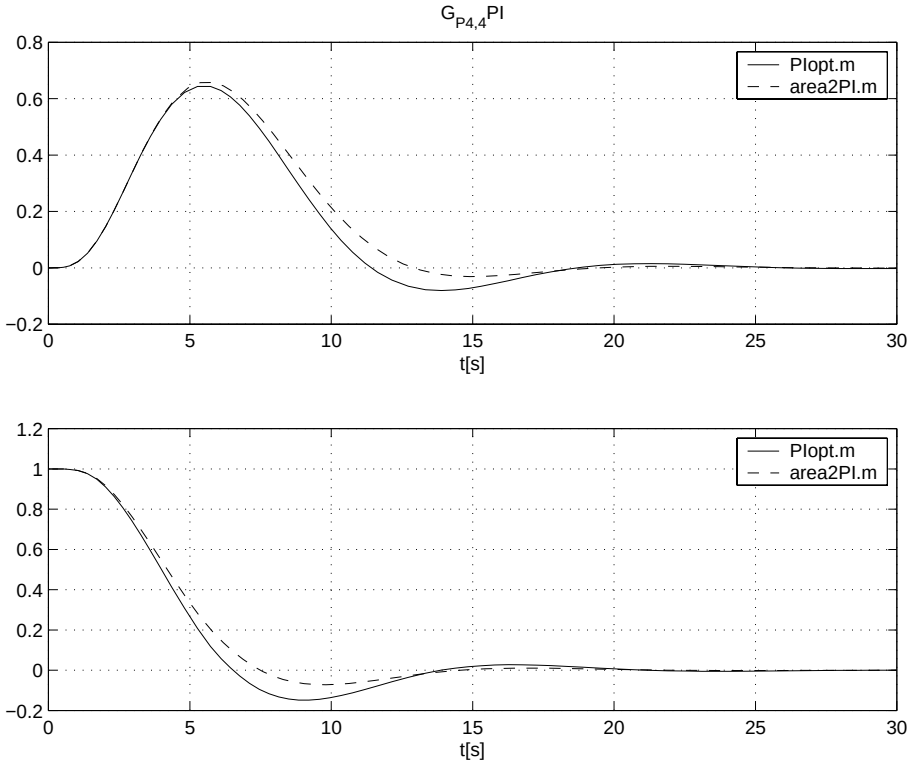


Fig. 55. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

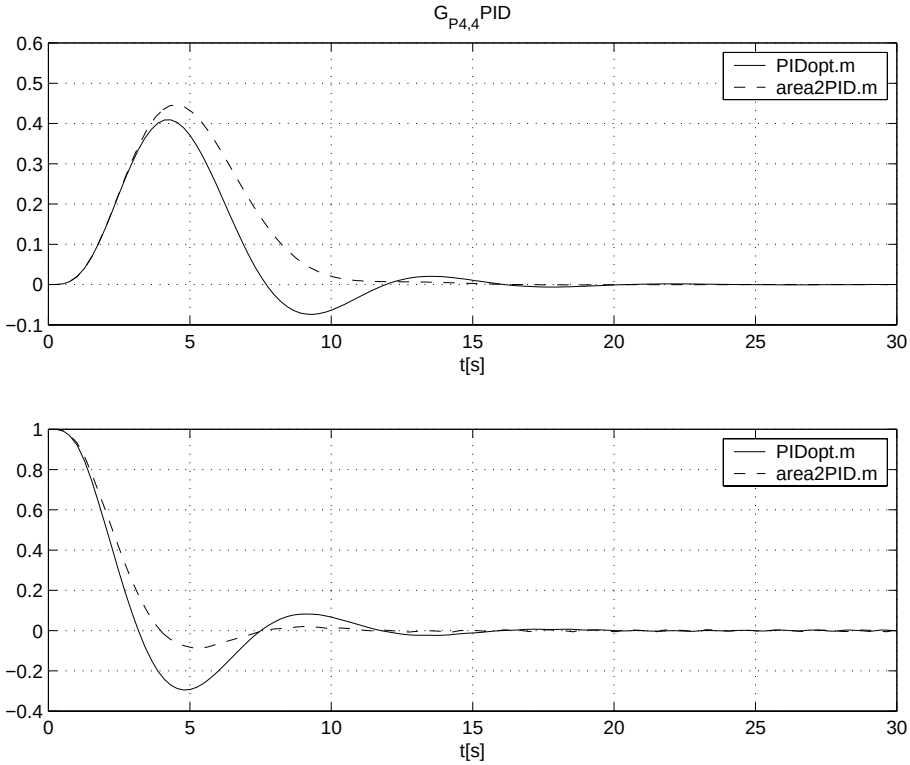


Fig. 56. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

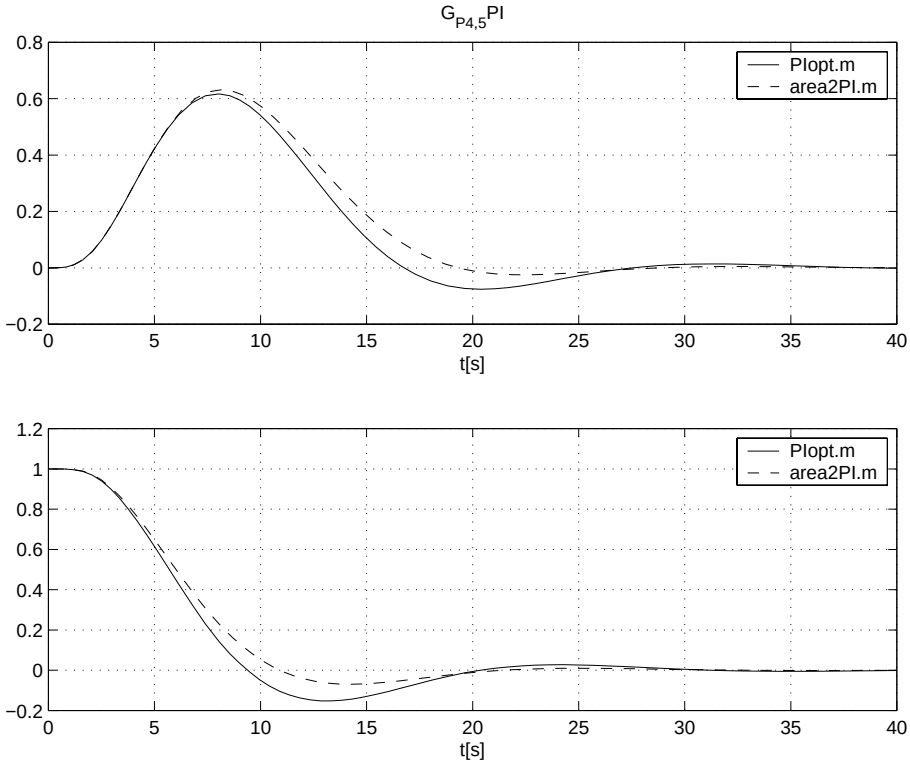


Fig. 57. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

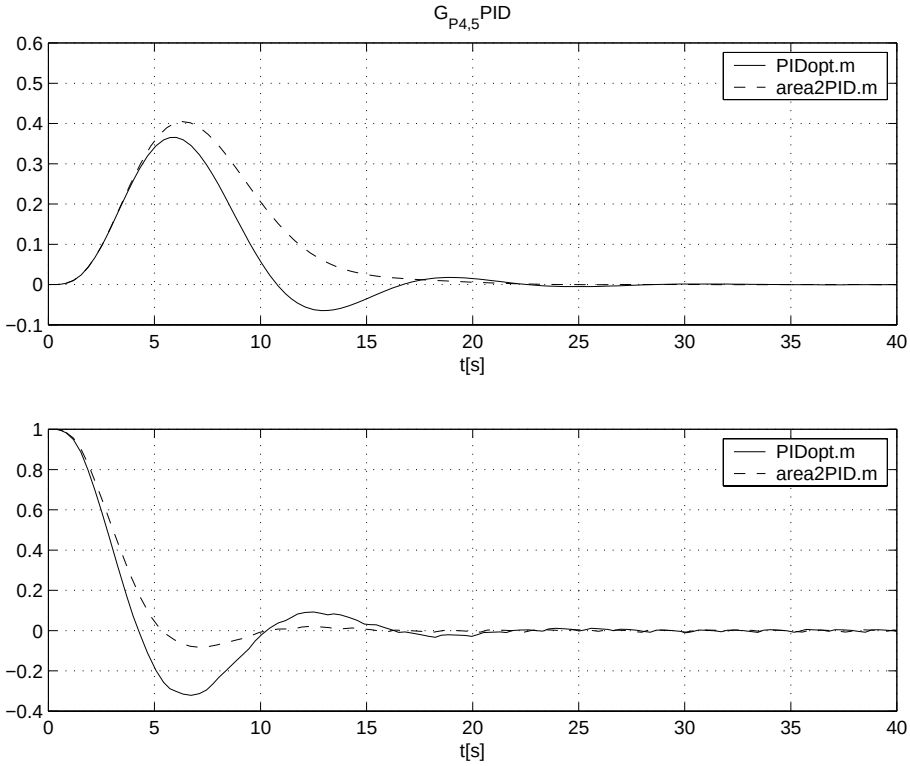


Fig. 58. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

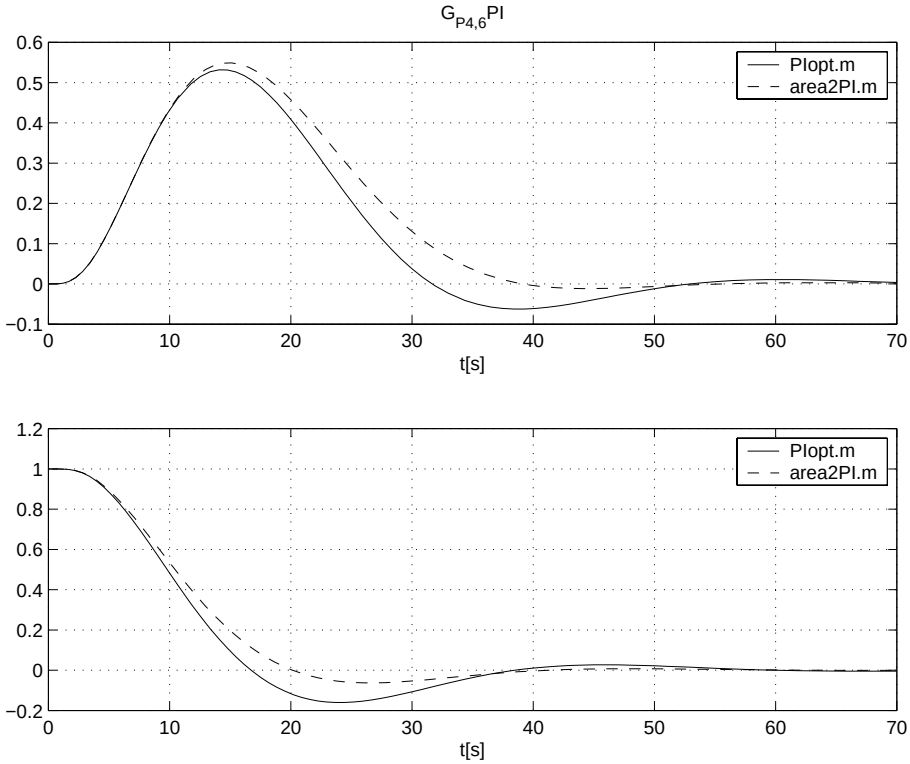


Fig. 59. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

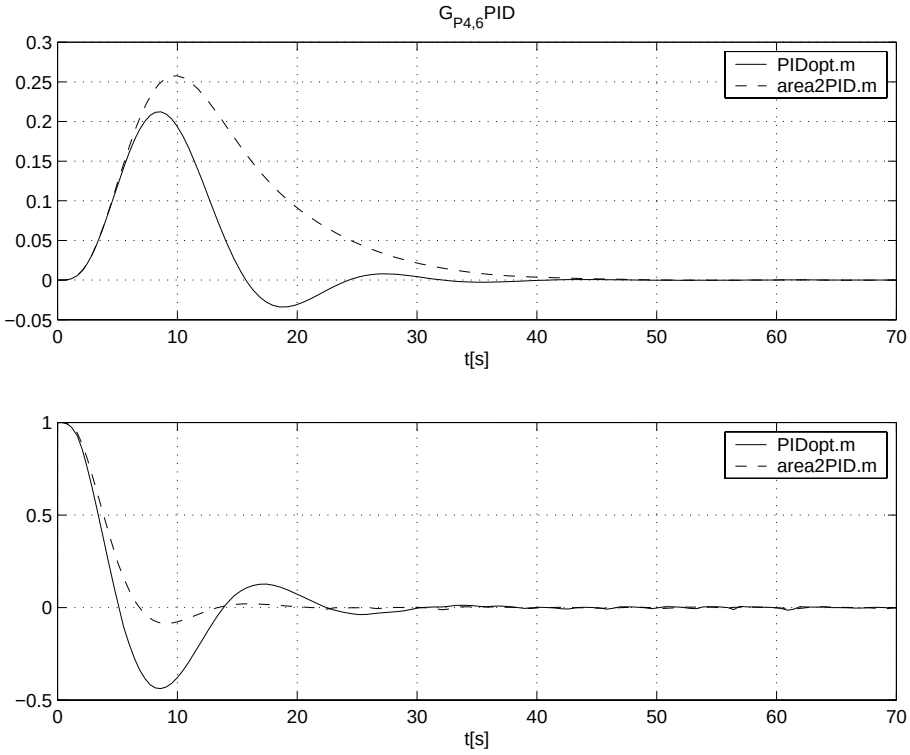


Fig. 60. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

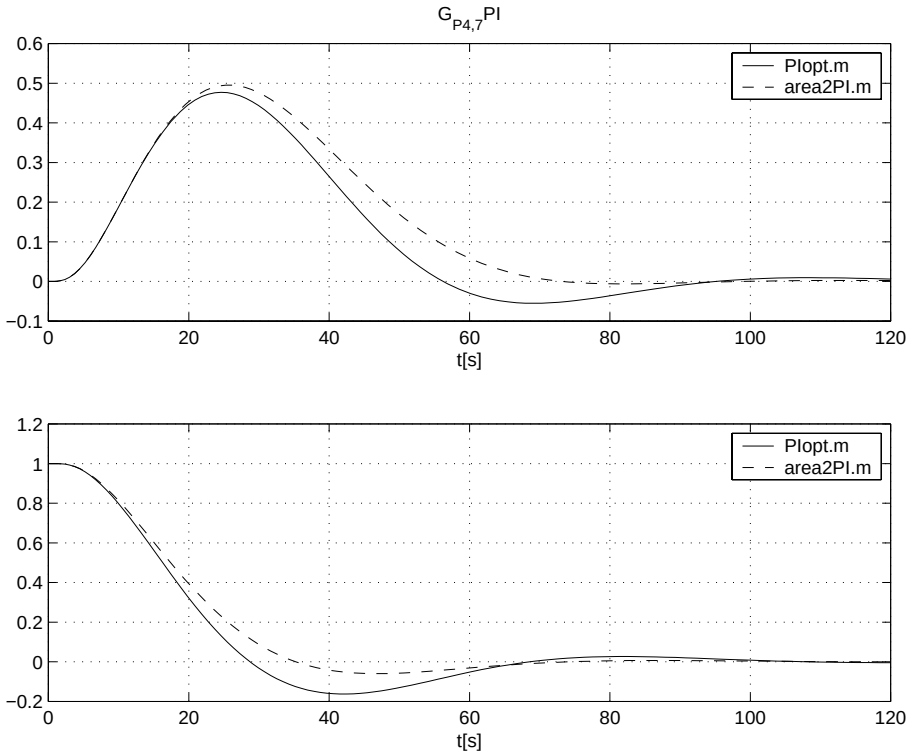


Fig. 61. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

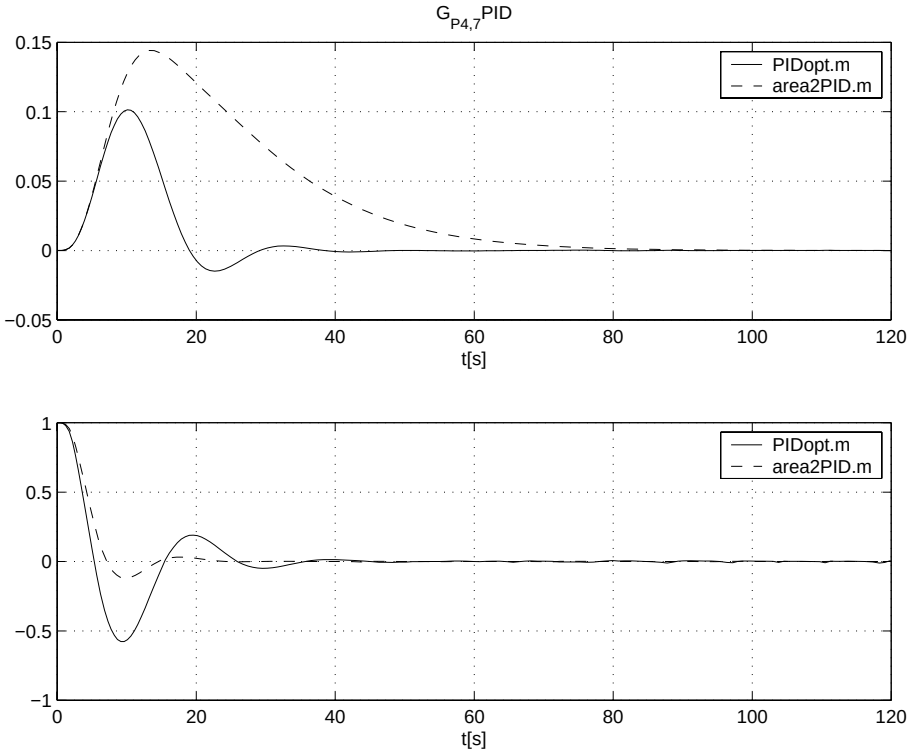


Fig. 62. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

Process 5: $G_{P5} = \dfrac{1}{(s+1)^n}$

Table 17. PI controller parameters calculated by using DRMO method (PIopt.m)

n	2	3	4	5	6	7	8
K	1	0.65153	0.52786	0.46447	0.42591	0.4	0.38139
K_i	1	0.45459	0.2918	0.21447	0.16944	0.14	0.11926

Table 18. PI controller parameters calculated by using MO method (area2PI.m)

n	2	3	4	5	6	7	8
K	1	0.625	0.5	0.4375	0.4	0.375	0.35714
K_i	0.75	0.375	0.25	0.1875	0.15	0.125	0.10714

Table 19. PID controller parameters calculated by using DRMO method (PIDopt.m)

n	2	3	4	5	6	7	8
K	10	2.9541	1.7028	1.2939	1.0917	0.97142	0.89165
K_i	12.899	1.7372	0.70571	0.43102	0.30807	0.23908	0.19512
K_d	2.6904	1.5	1.1757	1.1038	1.1014	1.128	1.1694

Table 20. PID controller parameters calculated by using MO method (area2PID.m)

n	2	3	4	5	6	7	8
K	10	2.3125	1.375	1.0625	0.90625	0.8125	0.75
K_i	5.25	0.9375	0.46875	0.3125	0.23438	0.1875	0.15625
K_d	4.5	1.5	1.0938	1	0.98438	1	1.0313

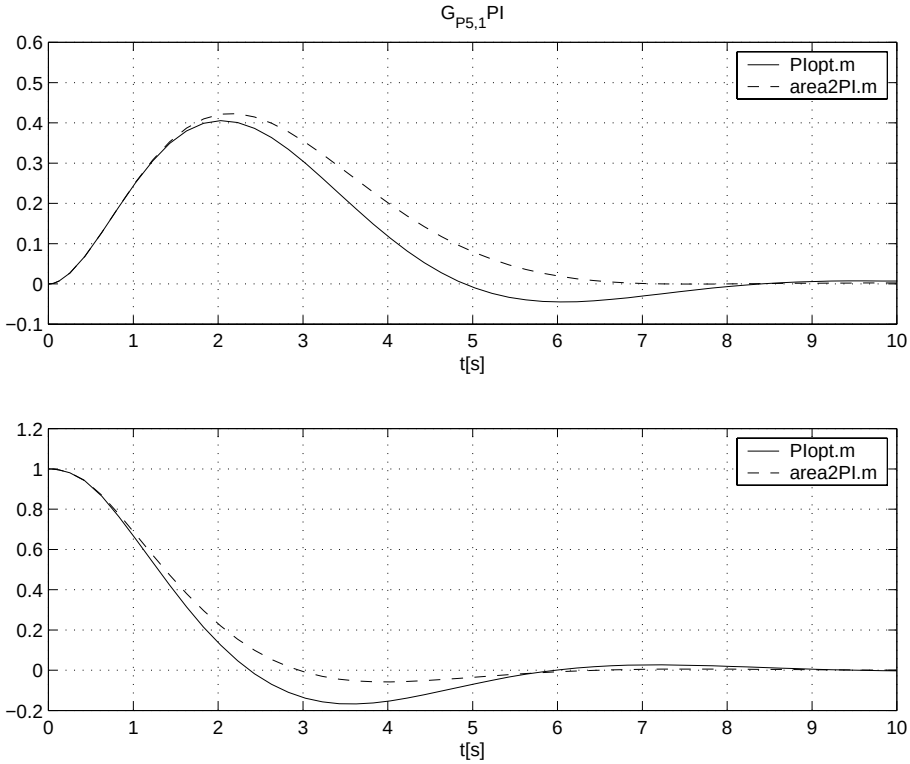


Fig. 63. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

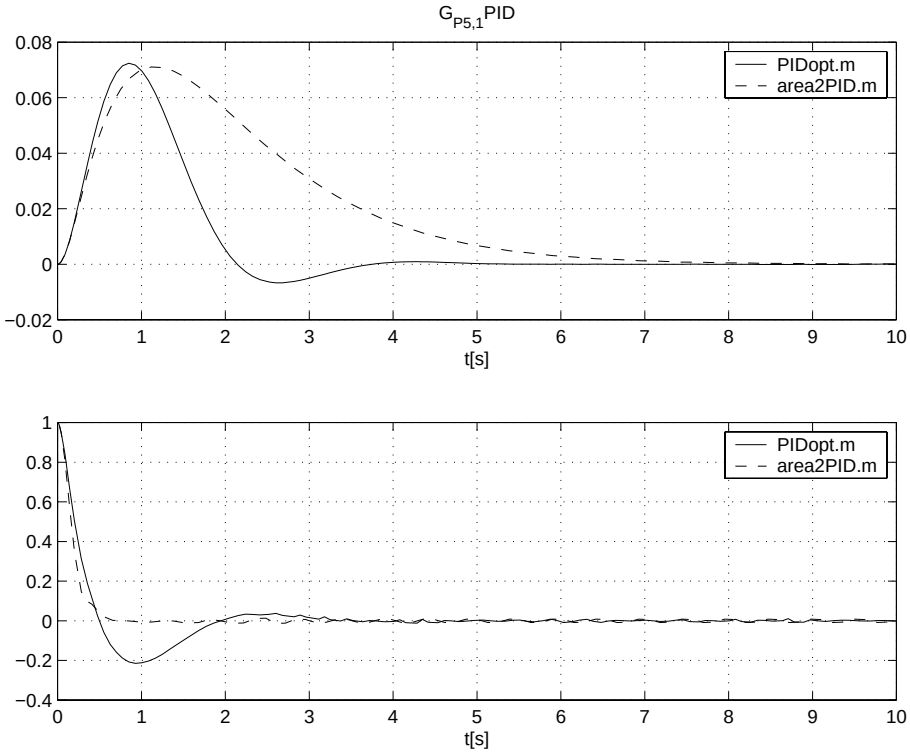


Fig. 64. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

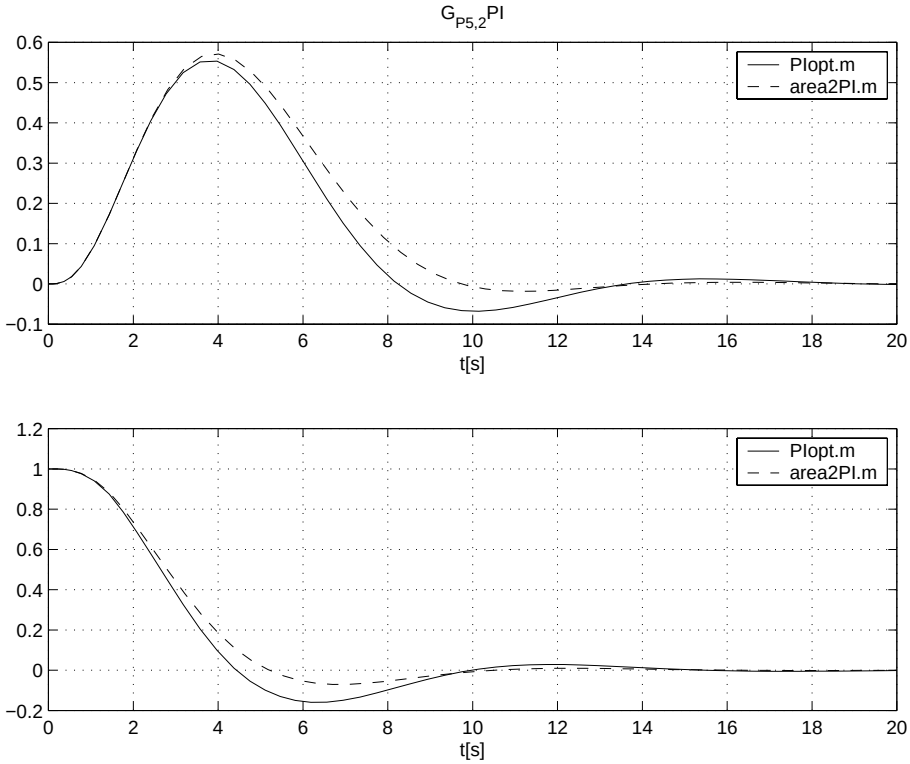


Fig. 65. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

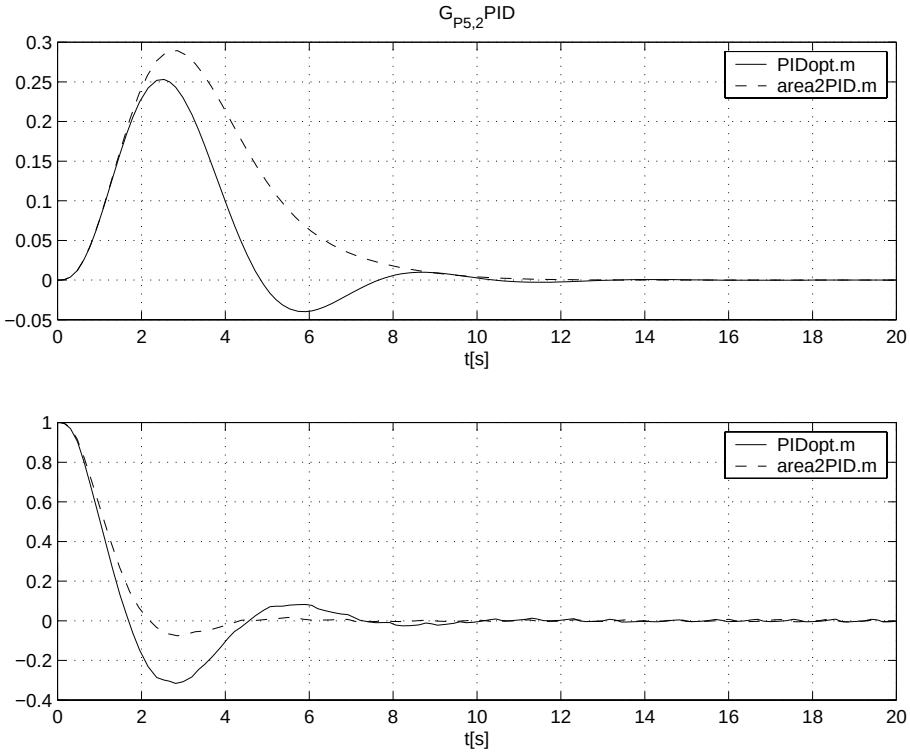


Fig. 66. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

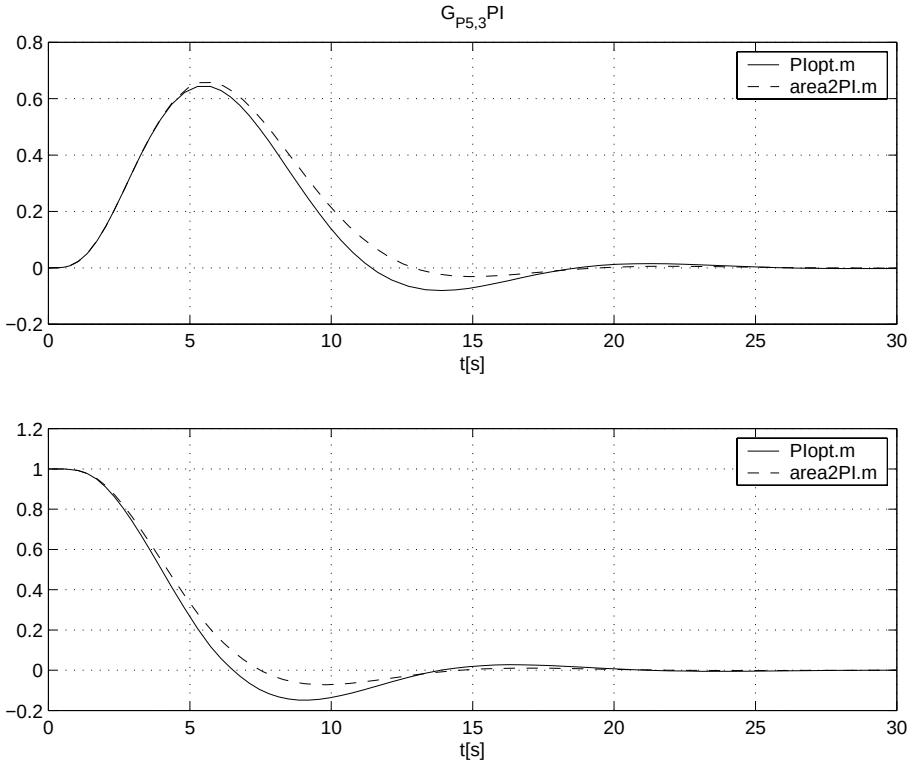


Fig. 67. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

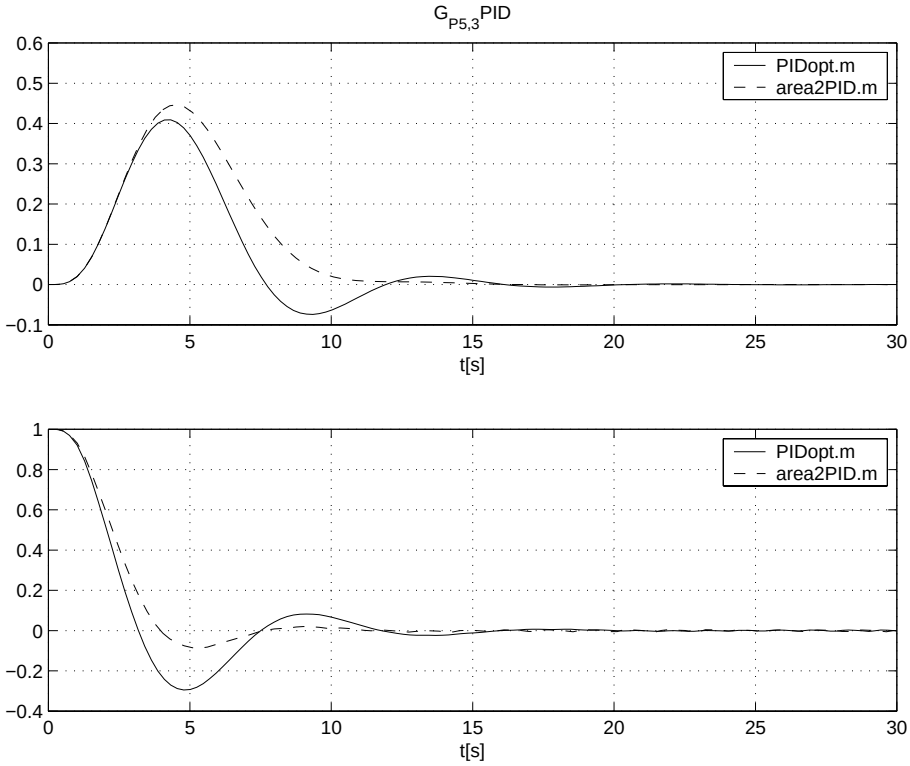


Fig. 68. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

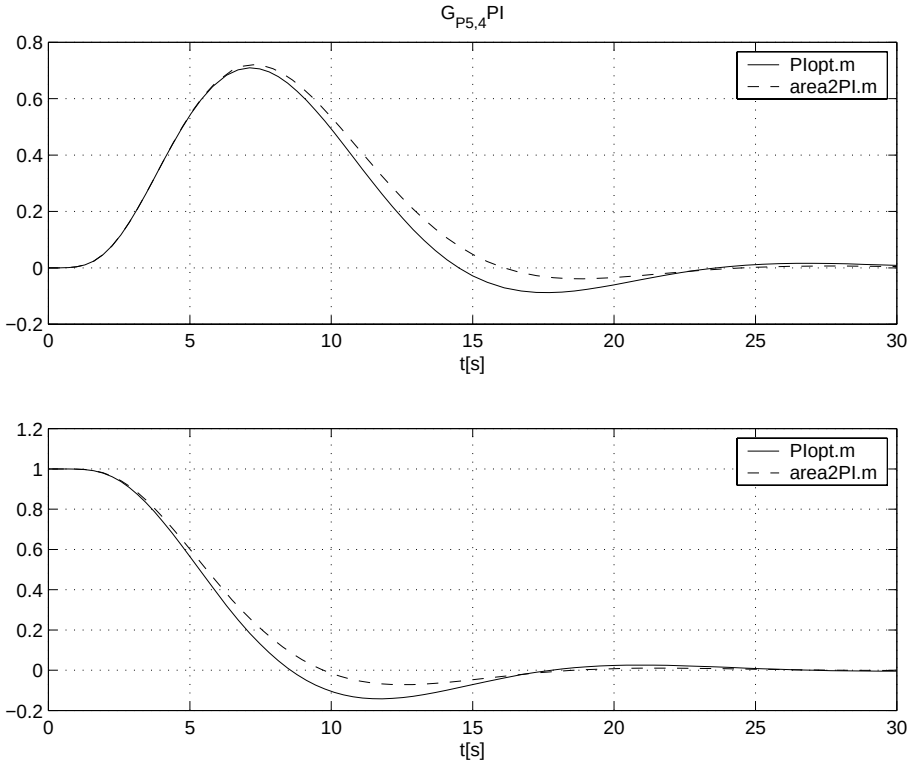


Fig. 69. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

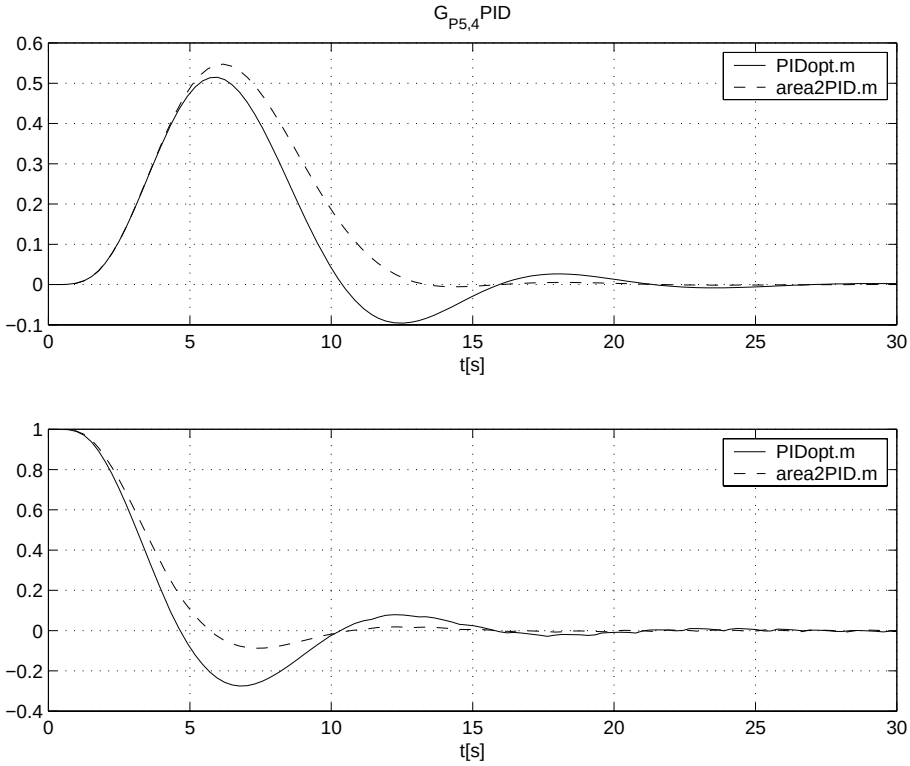


Fig. 70. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

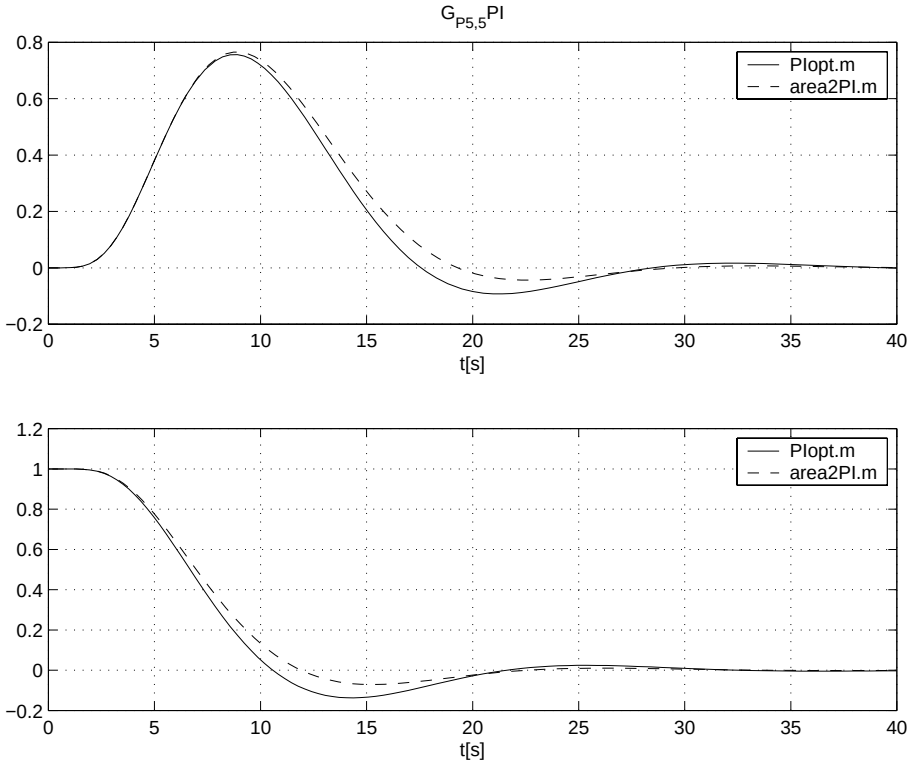


Fig. 71. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

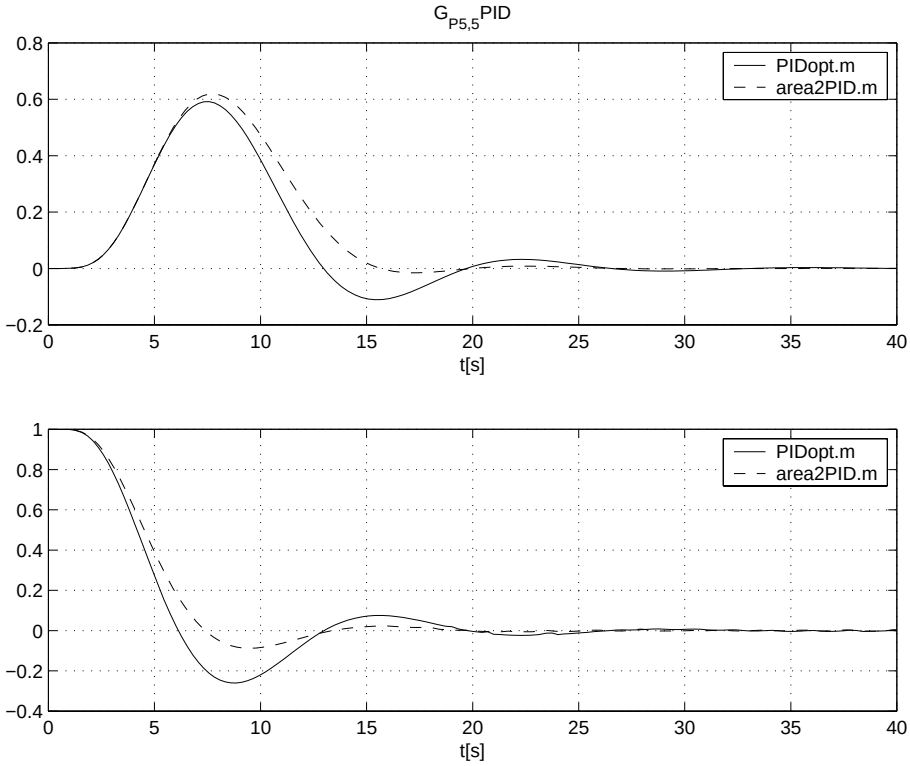


Fig. 72. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

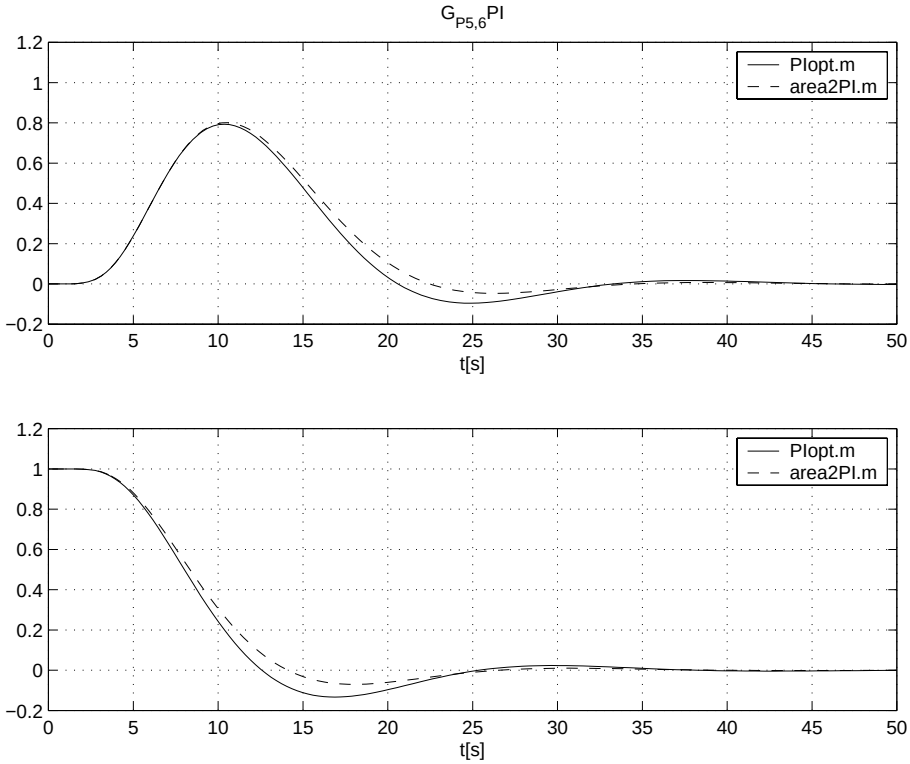


Fig. 73. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

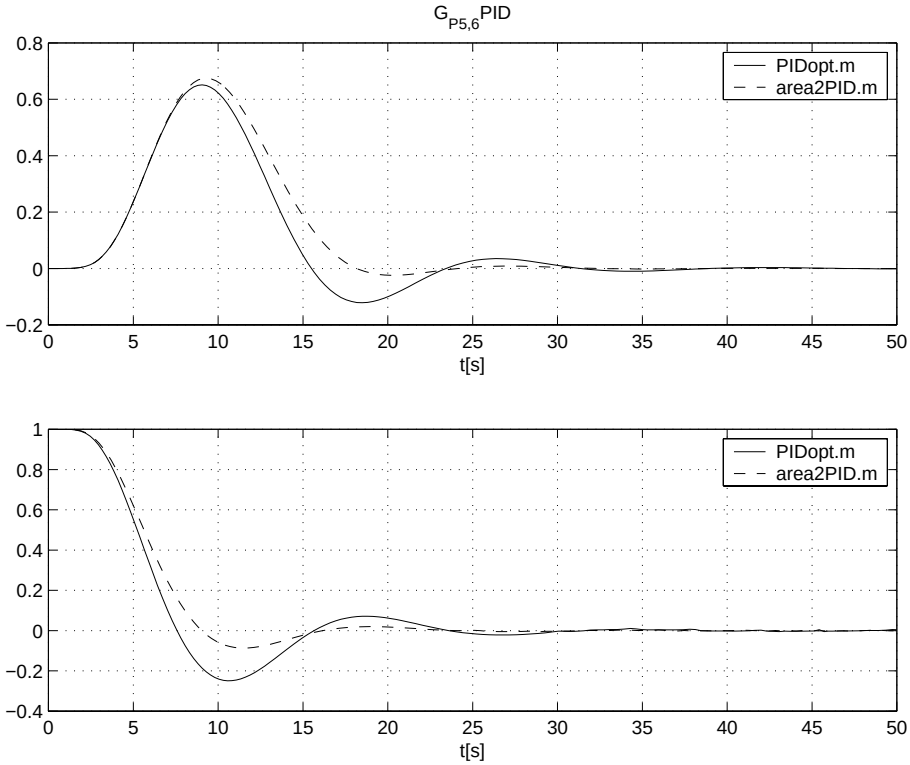


Fig. 74. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

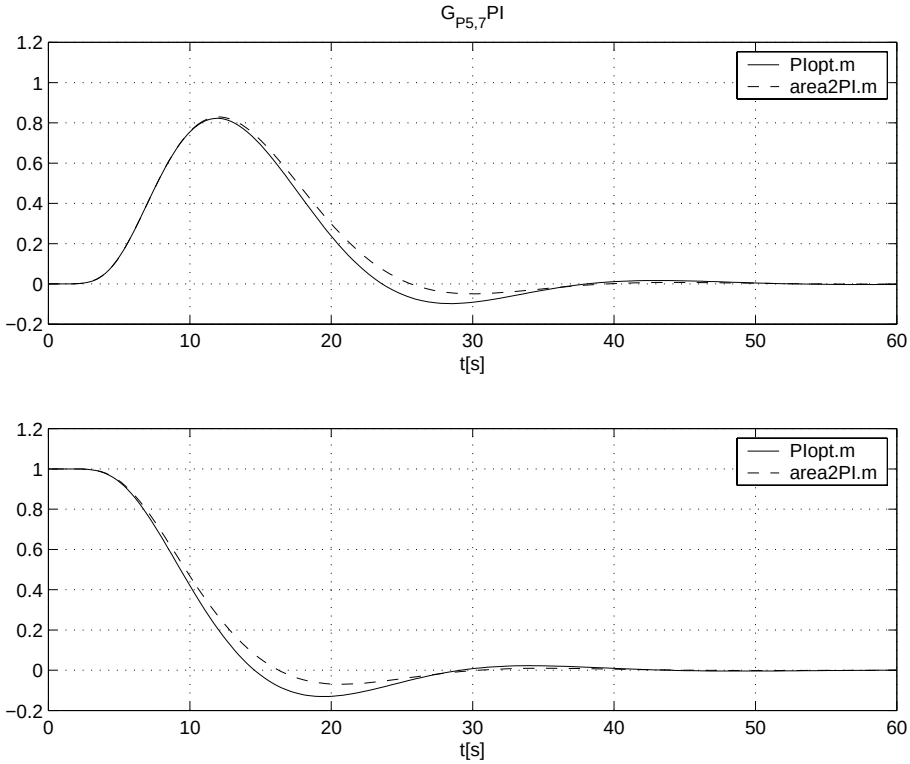


Fig. 75. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

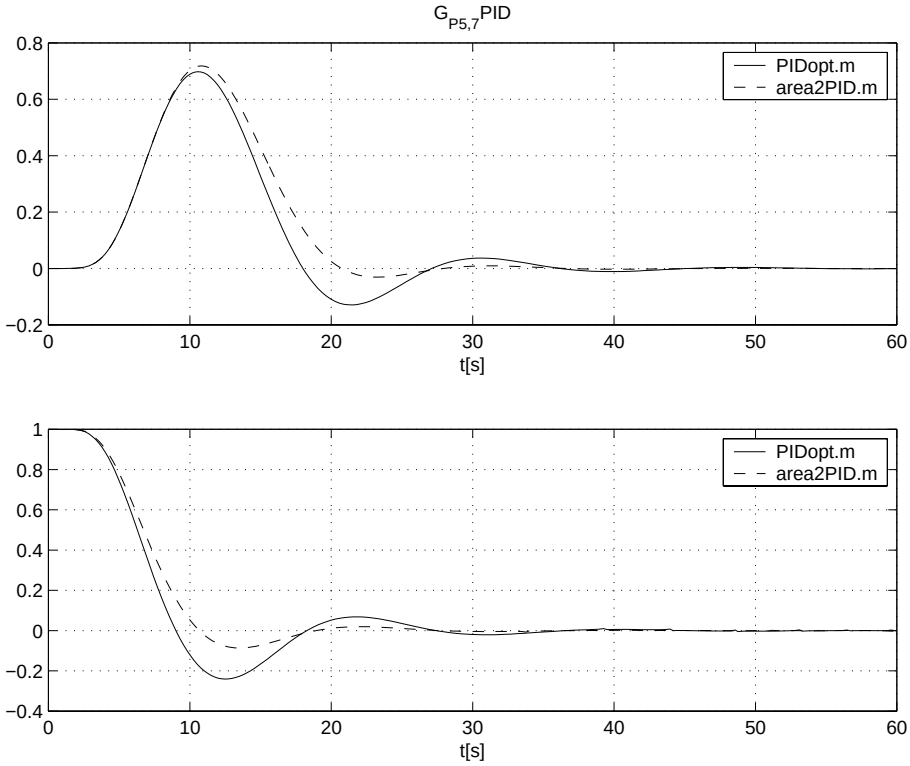


Fig. 76. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

Process 6: $G_{P6} = \dfrac{1}{(s+1)(Ts+1)(T^2s+1)(T^3s+1)}$

Table 21. PI controller parameters calculated by using DRMO method (PIopt.m)

T	0.2	0.3	0.4	0.5	0.6	0.7	0.8
K	2.1758	1.3765	0.99834	0.79029	0.66819	0.59563	0.55424
K_i	4.0407	1.9928	1.2295	0.85471	0.63944	0.50257	0.40916

Table 22. PI controller parameters calculated by using MO method (area2PI.m)

T	0.2	0.3	0.4	0.5	0.6	0.7	0.8
K	2.1	1.3167	0.95	0.75	0.63333	0.56429	0.525
K_i	2.0833	1.2821	0.89286	0.66667	0.52083	0.42017	0.34722

Table 23. PID controller parameters calculated by using DRMO method (PIDopt.m)

T	0.2	0.3	0.4	0.5	0.6	0.7	0.8
K	10	10	5.346	3.4592	2.5617	2.0956	1.8509
K_i	26.682	20.871	7.2677	3.4892	2.0629	1.4002	1.0461
K_d	1.0195	1.4817	1.1466	0.97436	0.89869	0.88878	0.93272

Table 24. PID controller parameters calculated by using MO method (area2PID.m)

T	0.2	0.3	0.4	0.5	0.6	0.7	0.8
K	10	6.0118	3.603	2.5139	1.9517	1.6443	1.4778
K_i	8.4135	4.5955	2.5265	1.6074	1.1267	0.84655	0.66998
K_d	1.582	1.4166	1.0779	0.91134	0.83813	0.82767	0.868

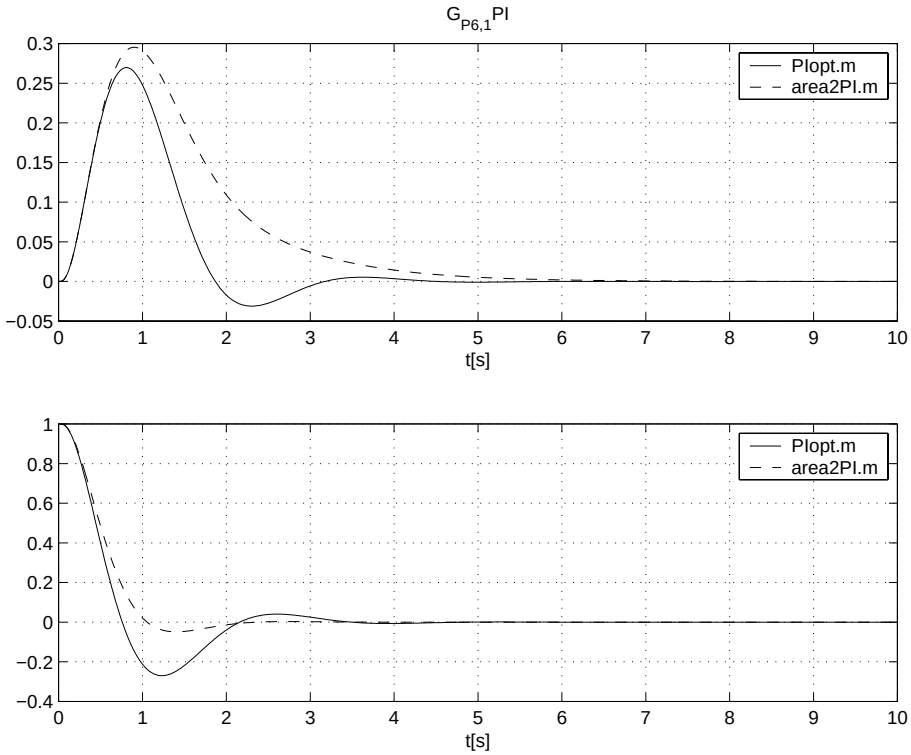


Fig. 77. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

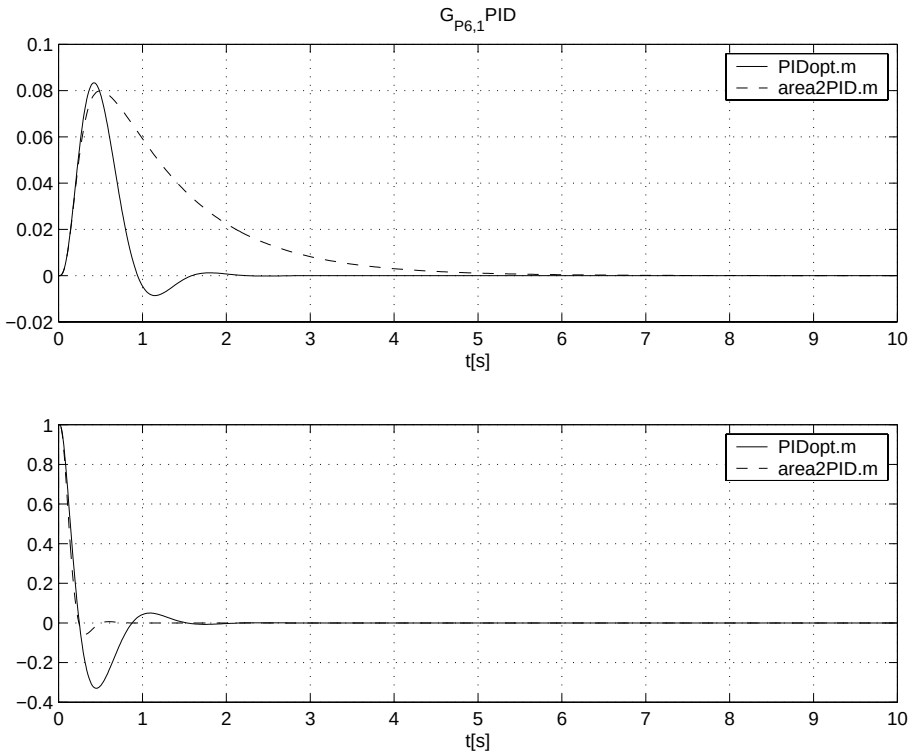


Fig. 78. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

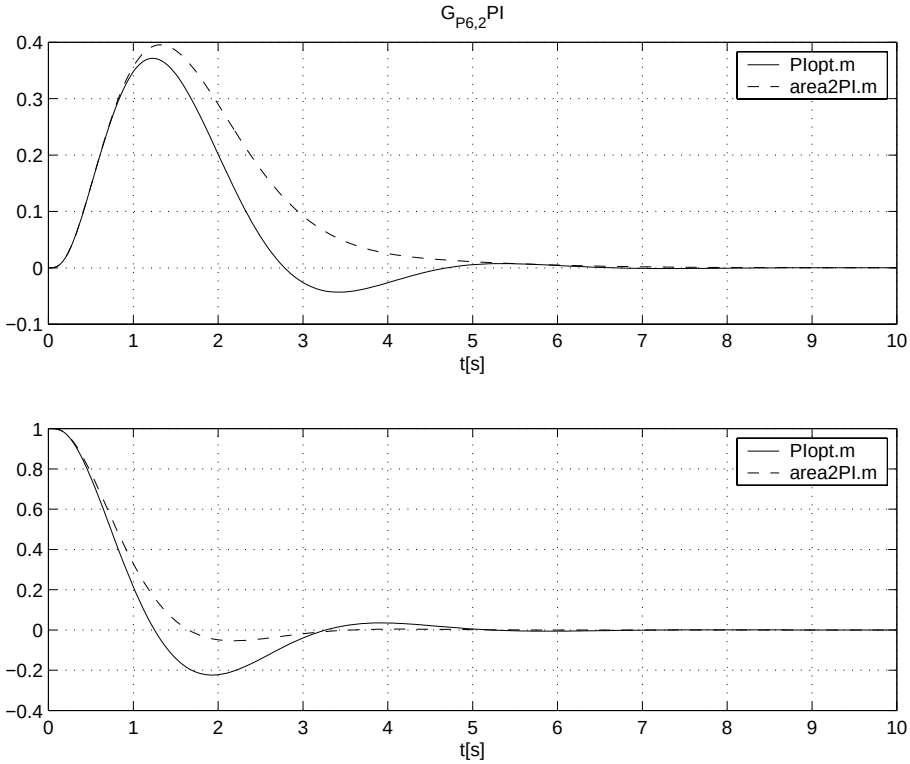


Fig. 79. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

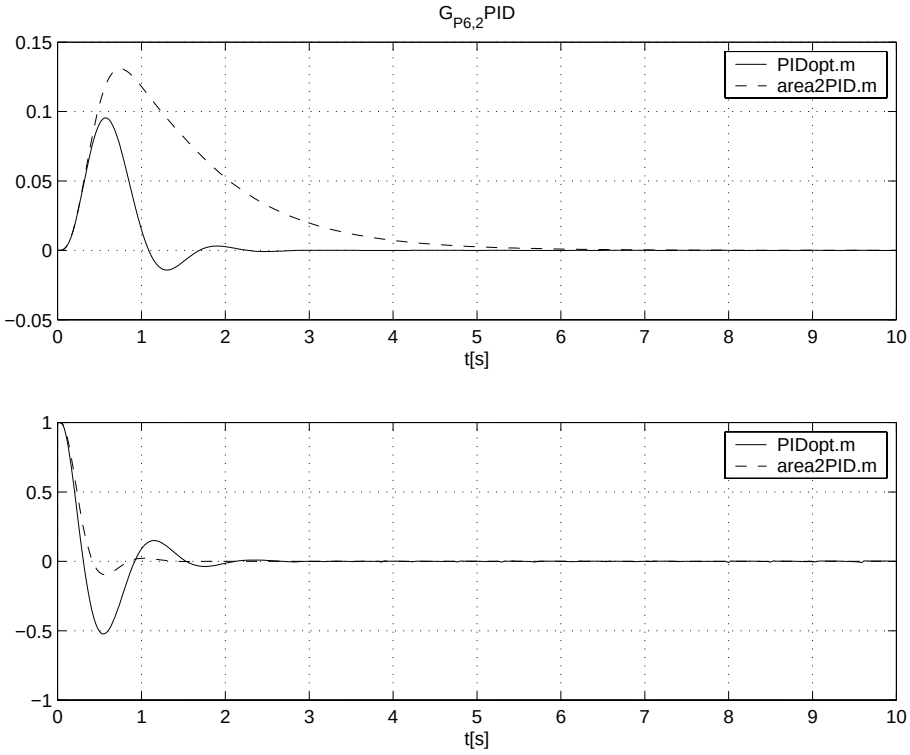


Fig. 80. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

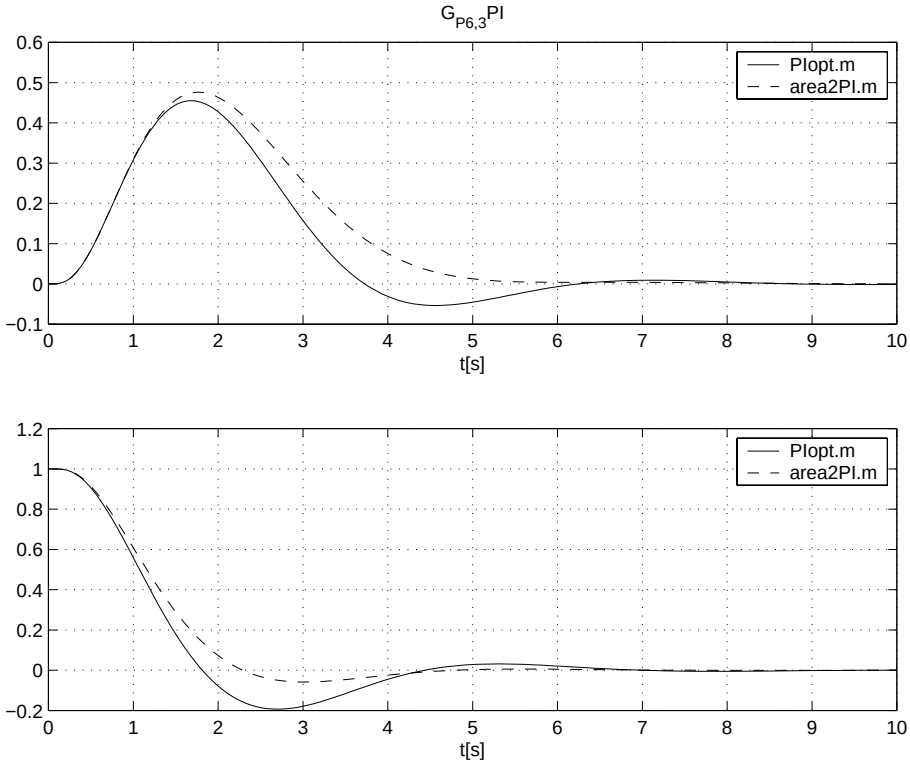


Fig. 81. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

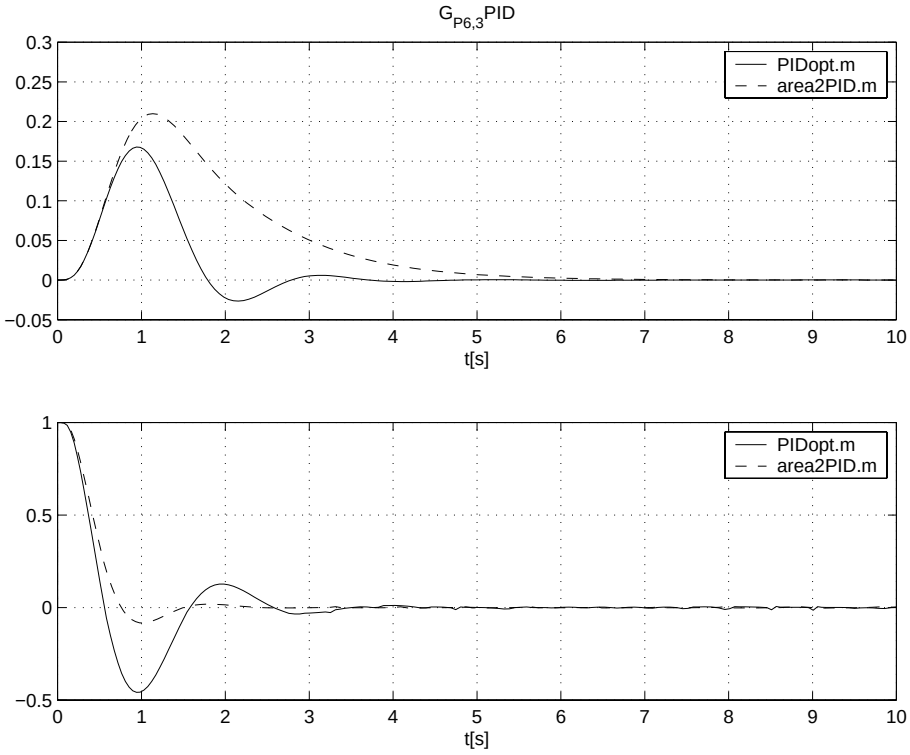


Fig. 82. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

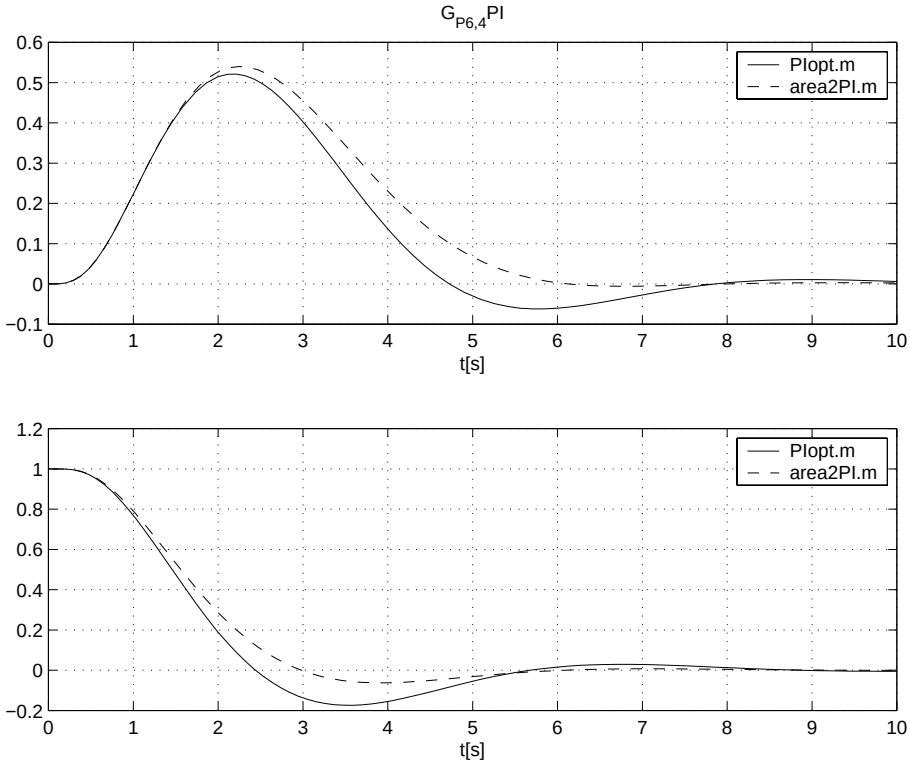


Fig. 83. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

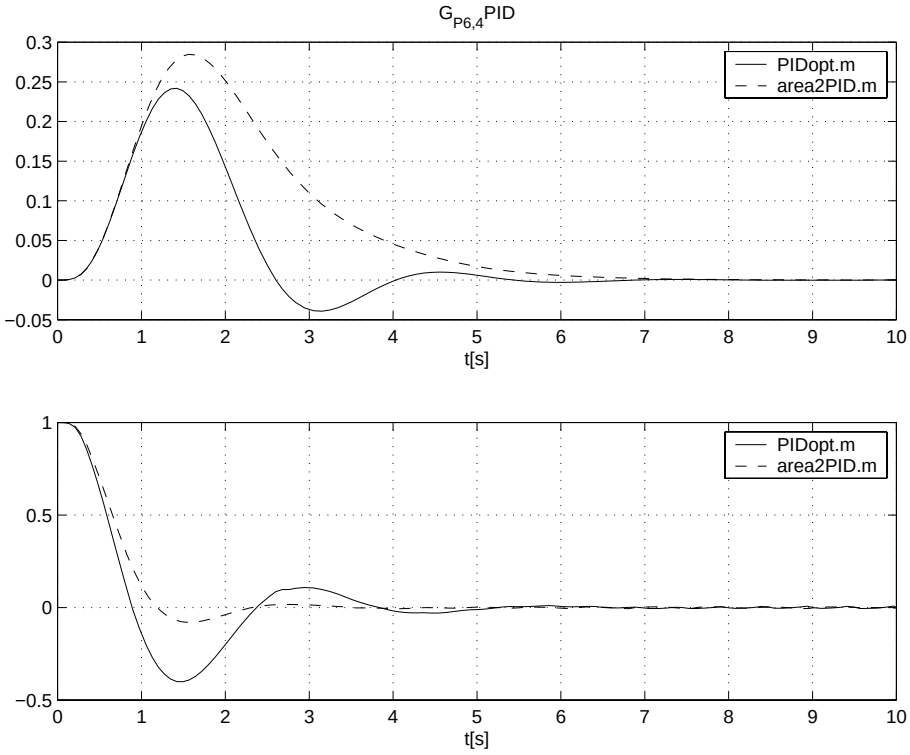


Fig. 84. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

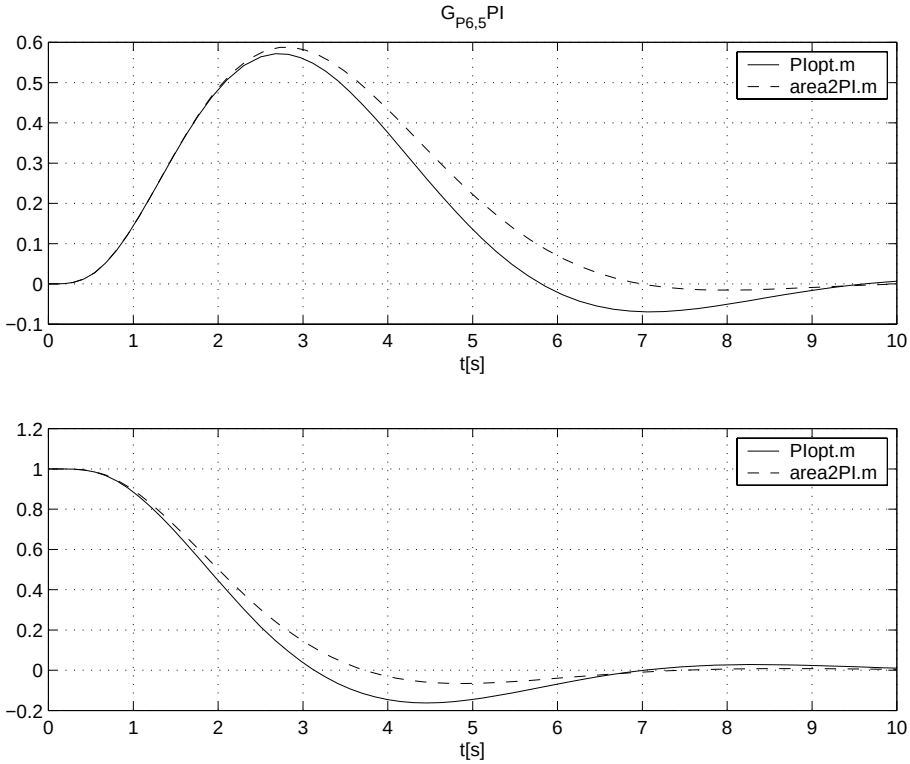


Fig. 85. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

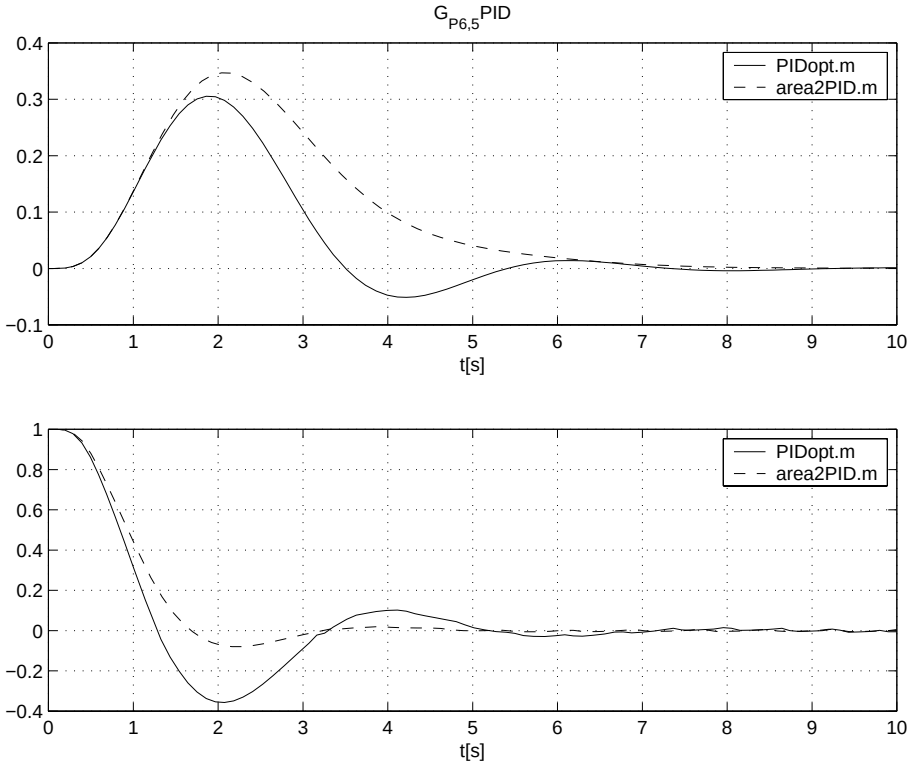


Fig. 86. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

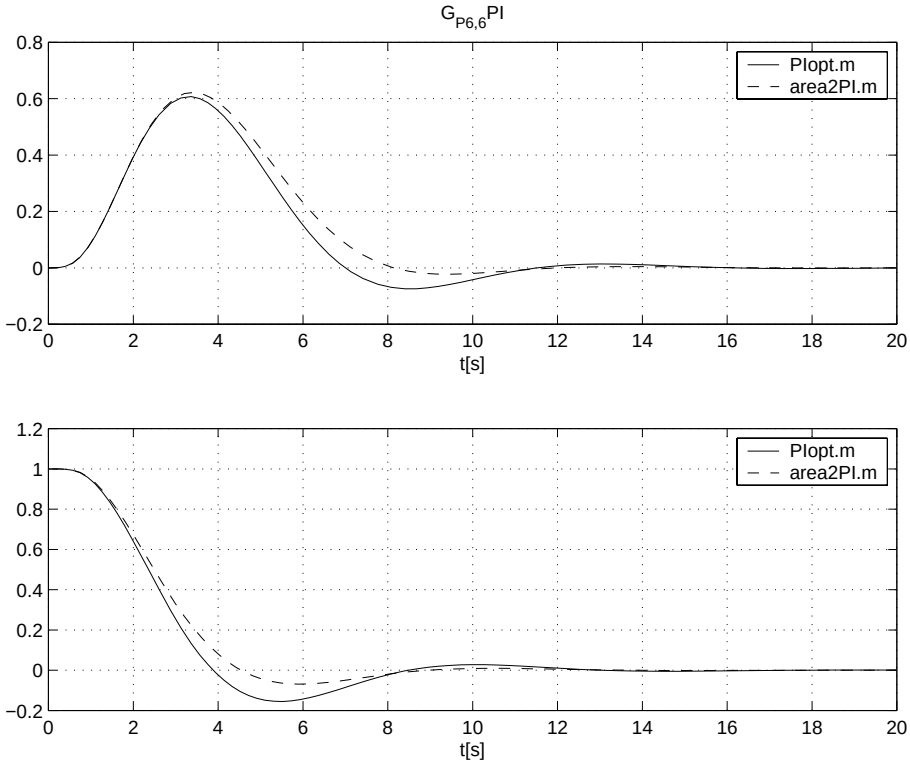


Fig. 87. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

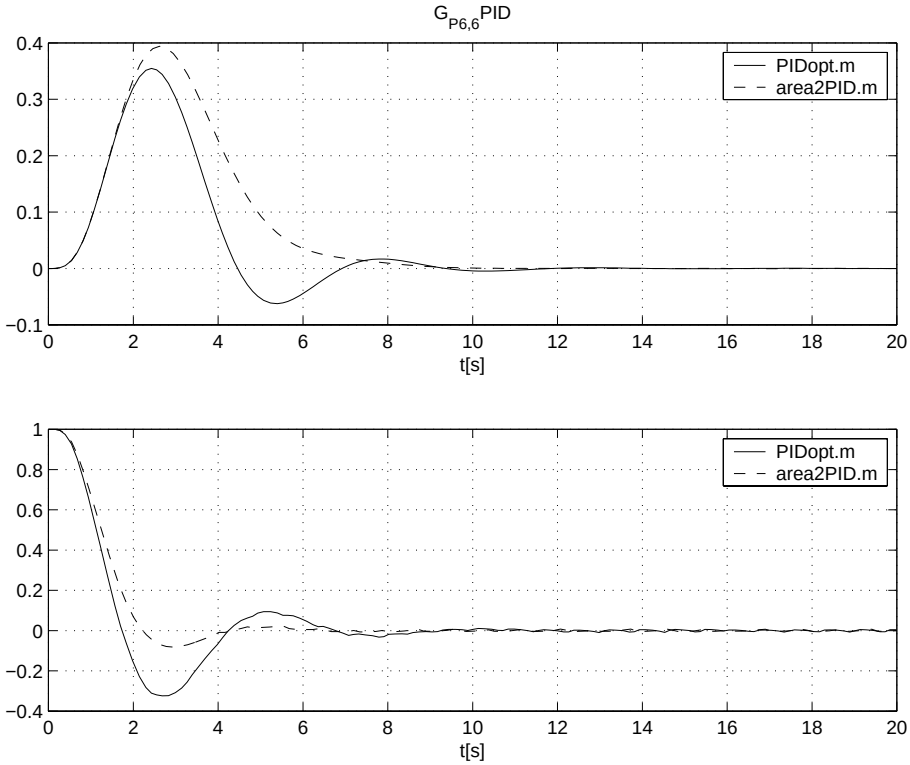


Fig. 88. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

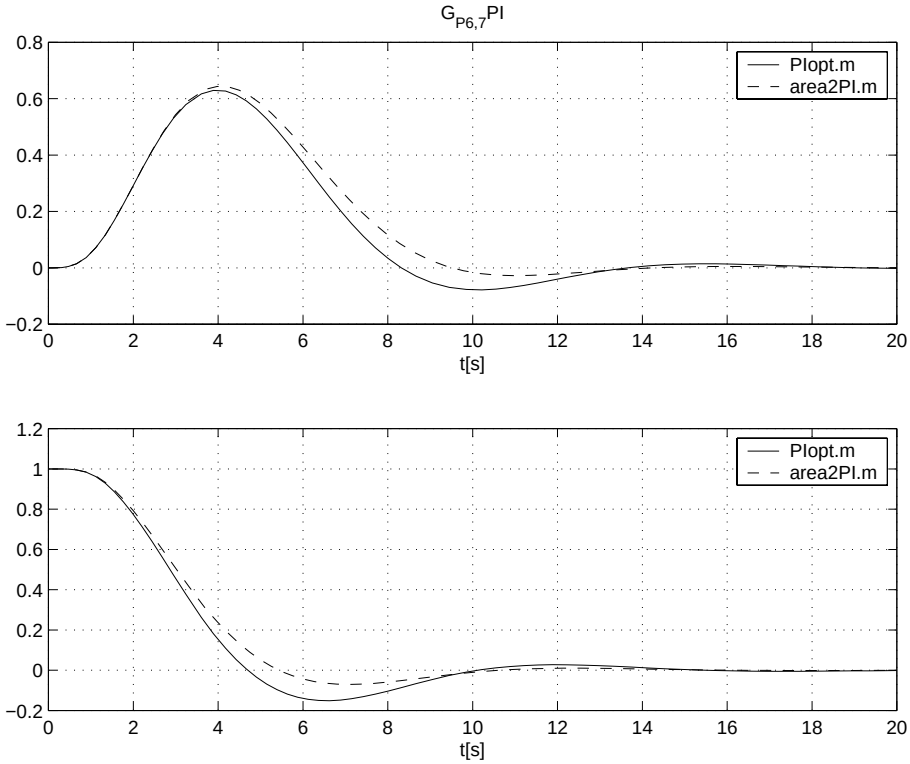


Fig. 89. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

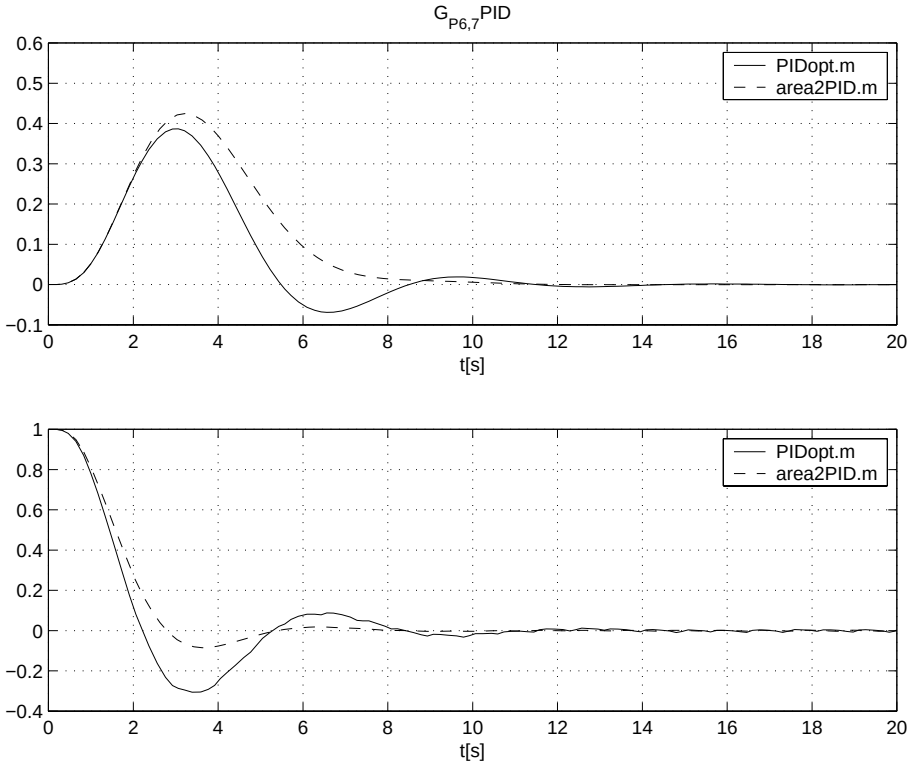


Fig. 90. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

Process 7: $G_{P7} = \dfrac{1 - sT}{(s+1)^3}$

Table 25. PI controller parameters calculated by using DRMO method (PIopt.m)

T	0.1	0.2	0.5	1	2	5	10
K	0.62237	0.59539	0.52577	0.43845	0.3276	0.18519	0.10713
K_i	0.42453	0.3977	0.33257	0.25864	0.17625	0.087791	0.047144

Table 26. PI controller parameters calculated by using MO method (area2PI.m)

T	0.1	0.2	0.5	1	2	5	10
K	0.59351	0.56452	0.49057	0.4	0.28947	0.15625	0.08794
K_i	0.35274	0.33266	0.28302	0.225	0.15789	0.082031	0.045226

Table 27. PID controller parameters calculated by using DRMO method (PIDopt.m)

T	0.1	0.2	0.5	1	2	5	10
K	2.5831	2.2898	1.695	1.1692	0.71199	0.32269	0.16786
K_i	1.4394	1.218	0.80917	0.49688	0.26718	0.10617	0.05194
K_d	1.3597	1.243	0.98778	0.73478	0.48489	0.23935	0.12965

Table 28. PID controller parameters calculated by using MO method (area2PID.m)

T	0.1	0.2	0.5	1	2	5	10
K	2.0008	1.7593	1.2824	0.875	0.53009	0.24074	0.1257
K_i	0.80672	0.70602	0.50926	0.34375	0.20602	0.092593	0.048131
K_d	1.3078	1.1574	0.85648	0.59375	0.36574	0.16898	0.088937

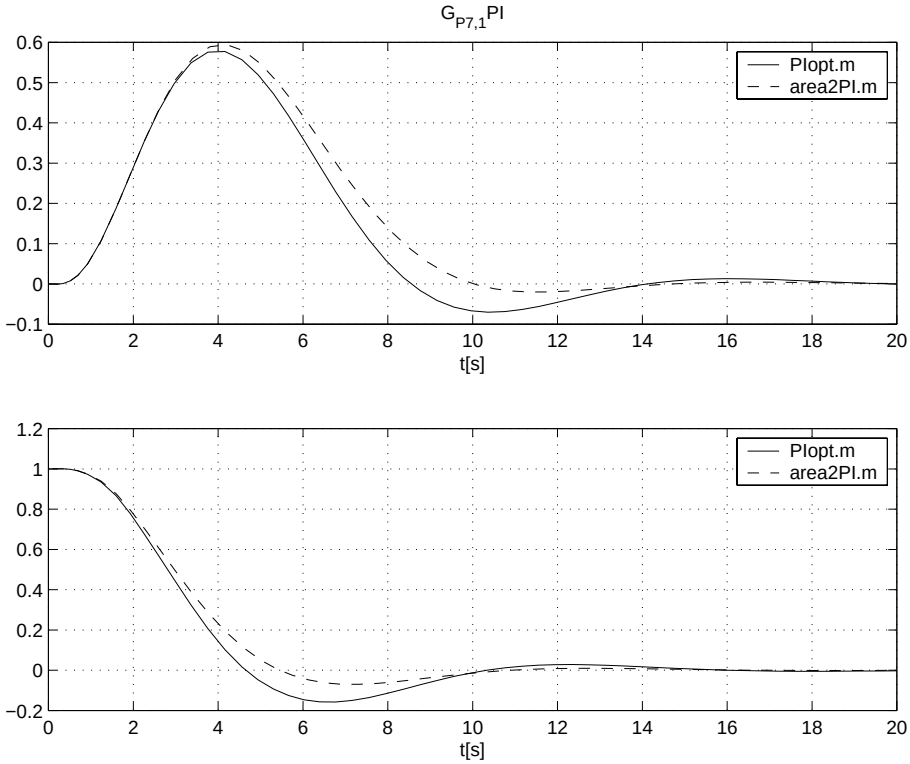


Fig. 91. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

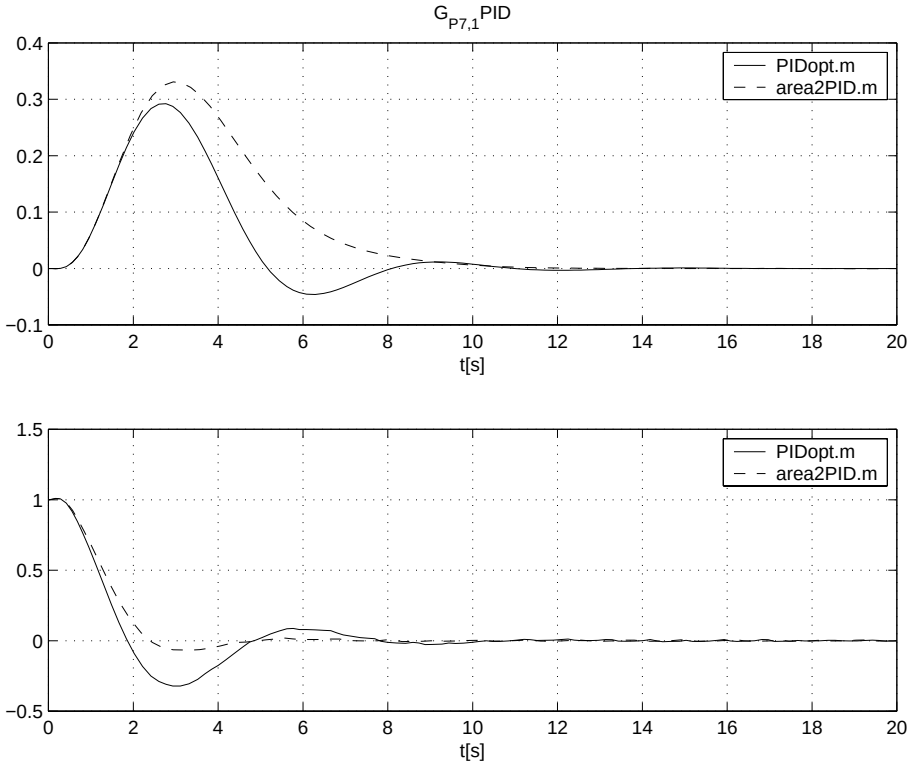


Fig. 92. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

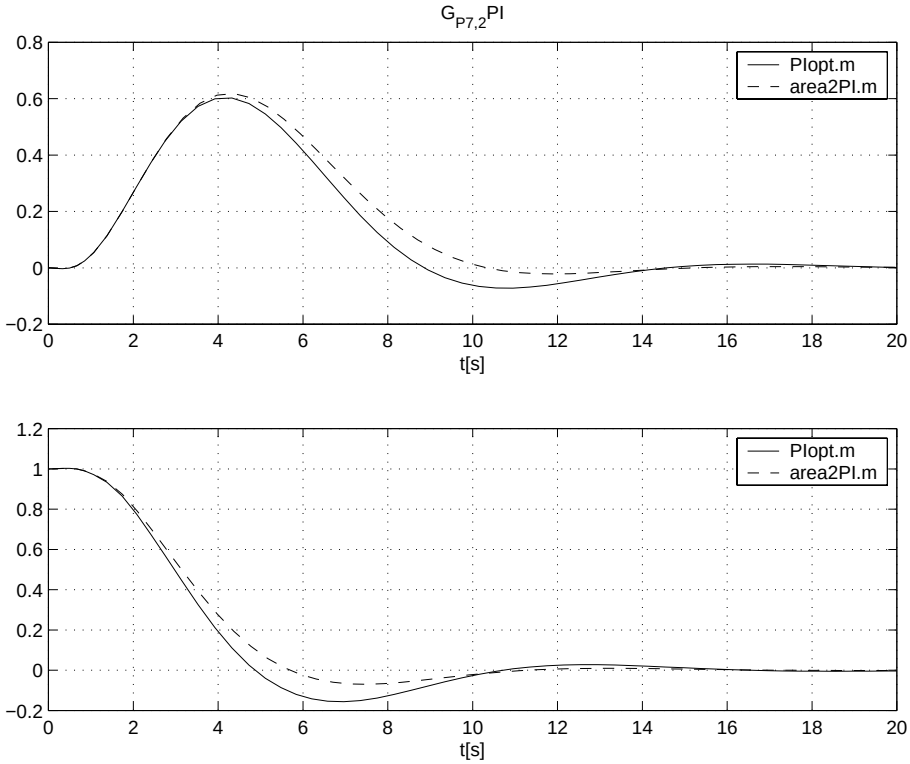


Fig. 93. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

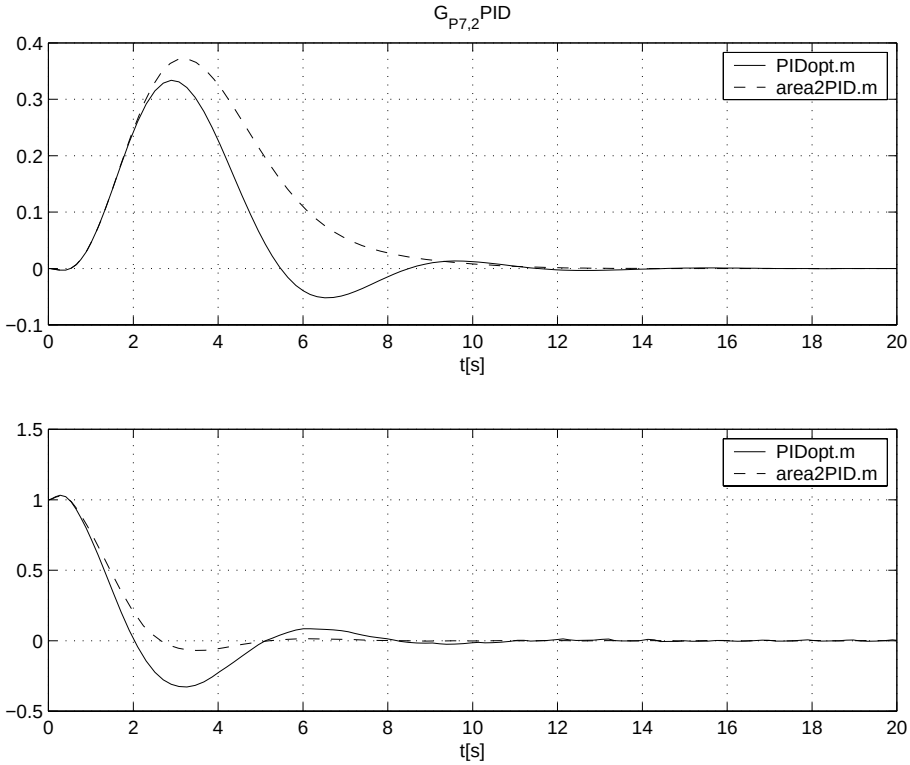


Fig. 94. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

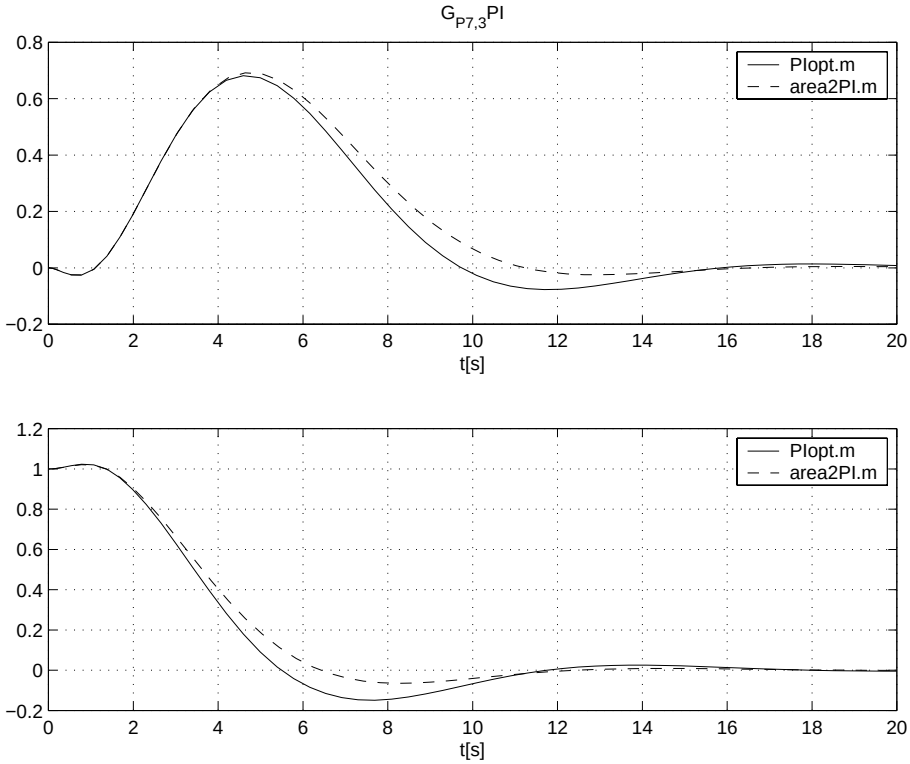


Fig. 95. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

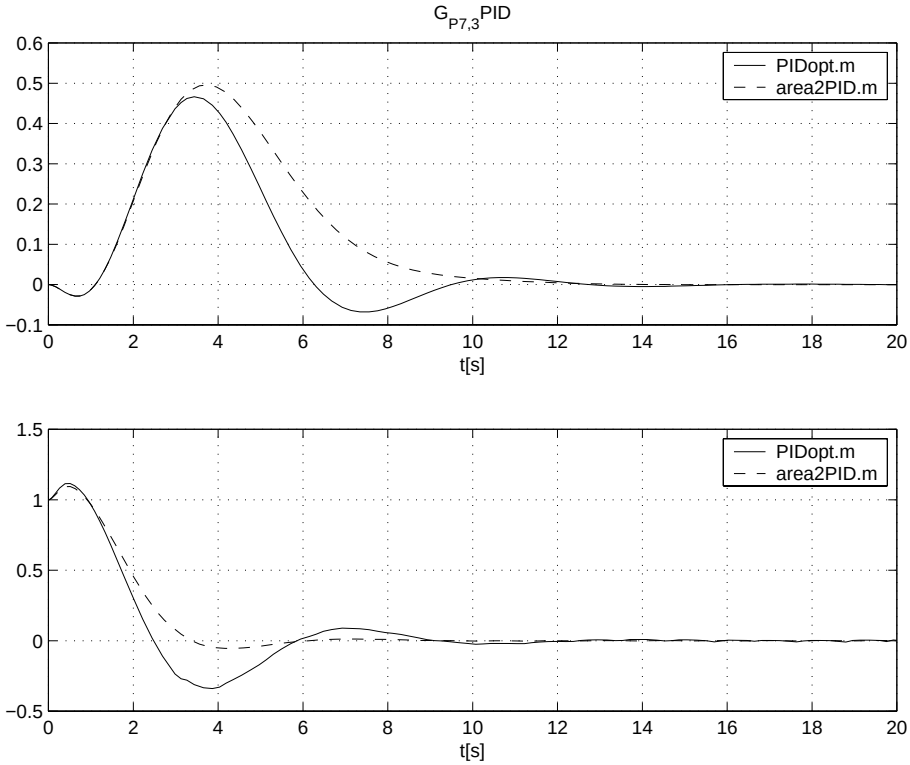


Fig. 96. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

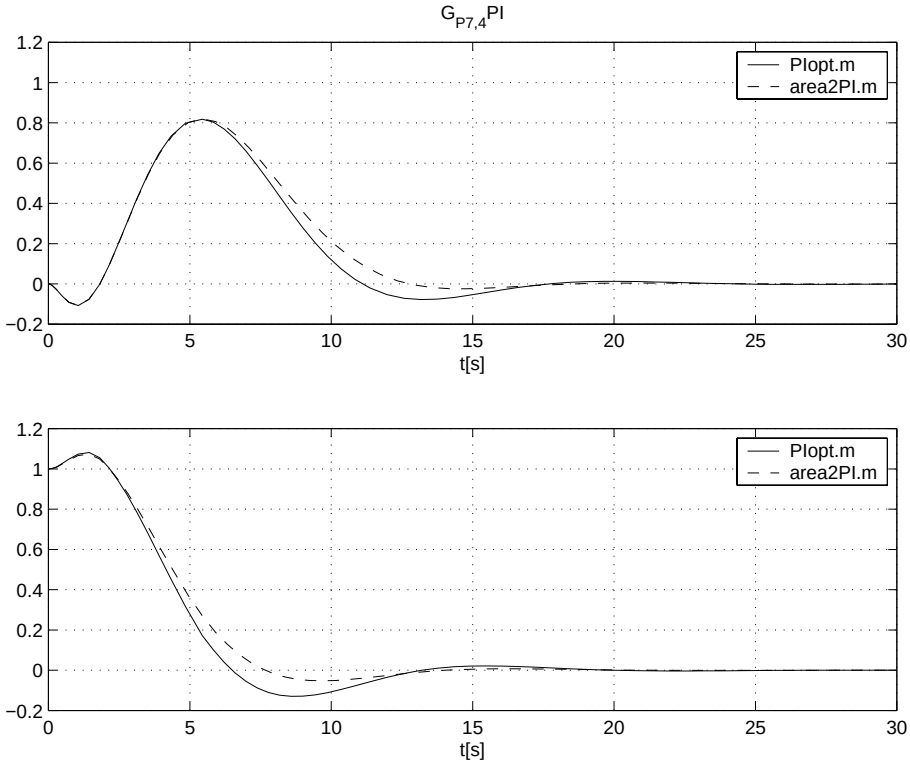


Fig. 97. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

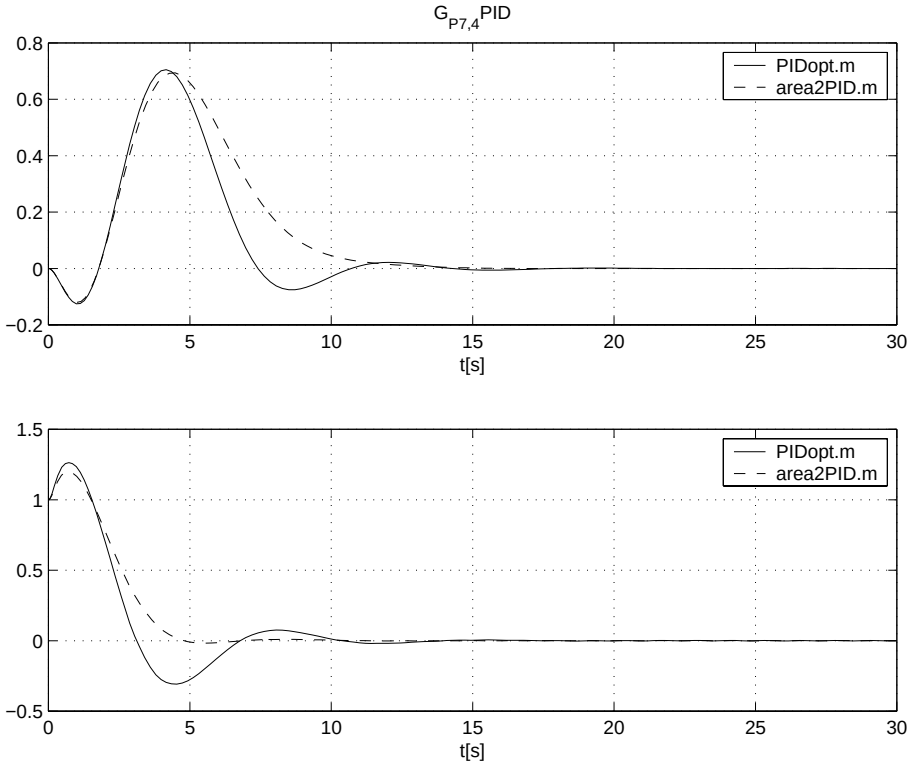


Fig. 98. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

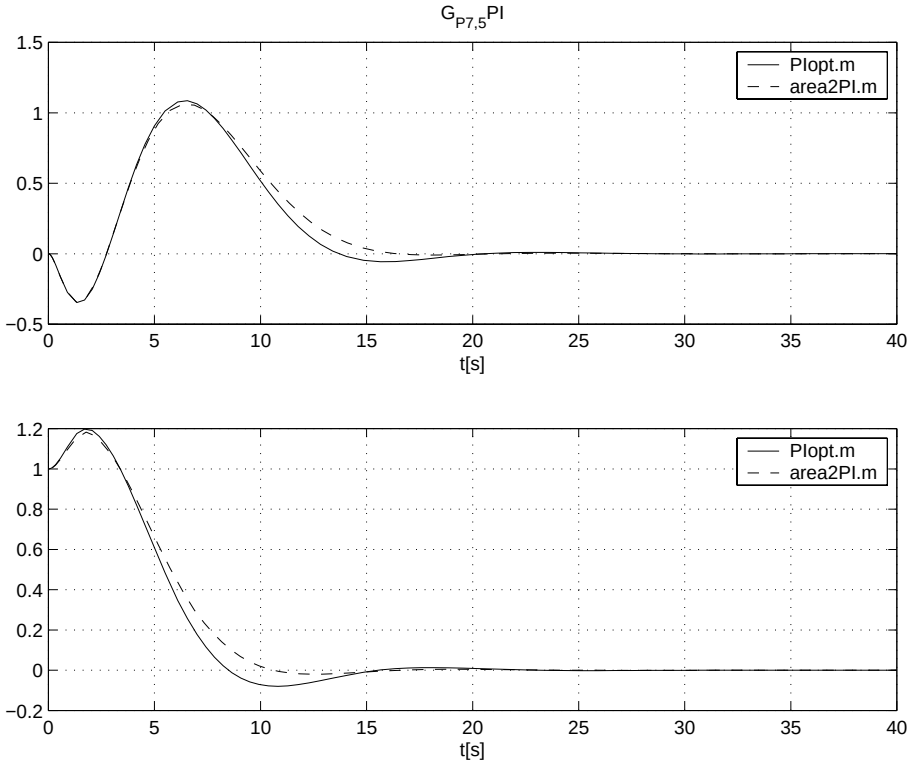


Fig. 99. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

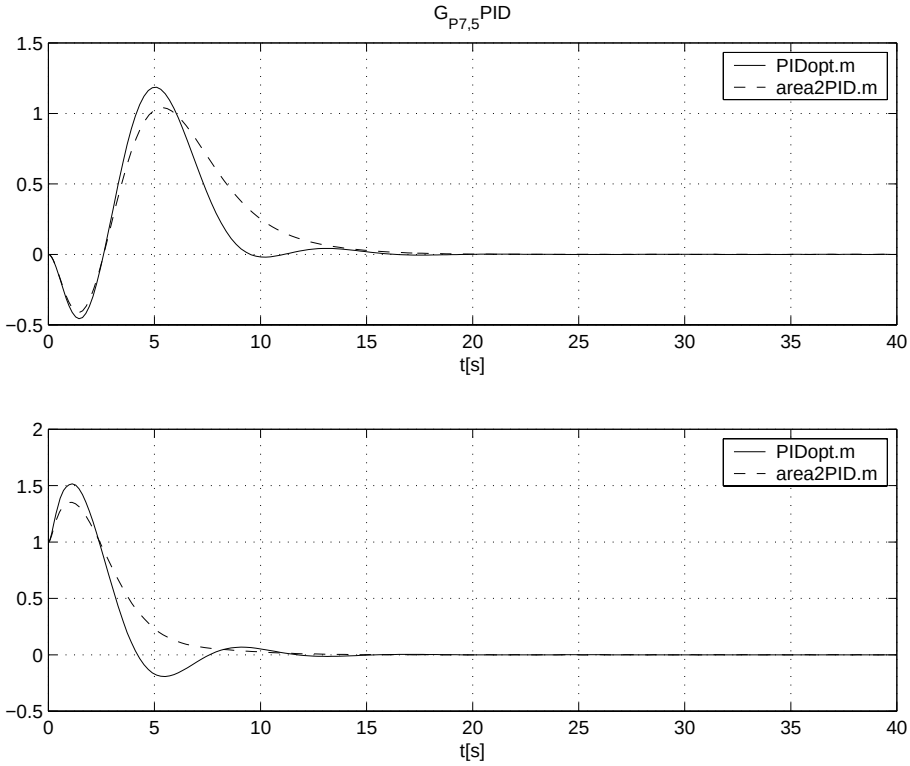


Fig. 100. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

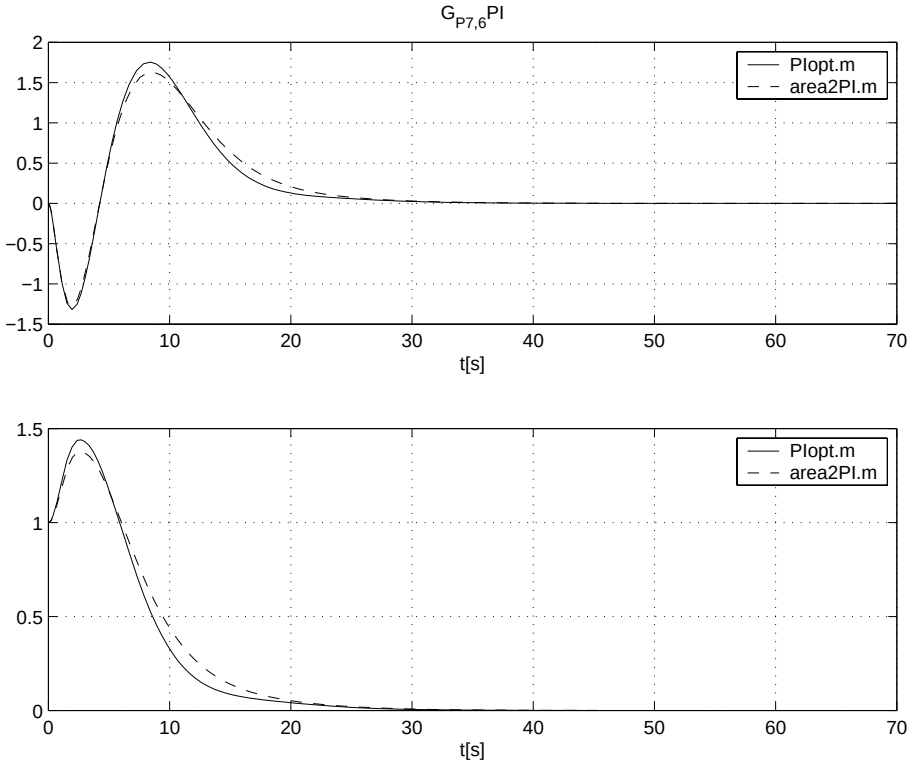


Fig. 101. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

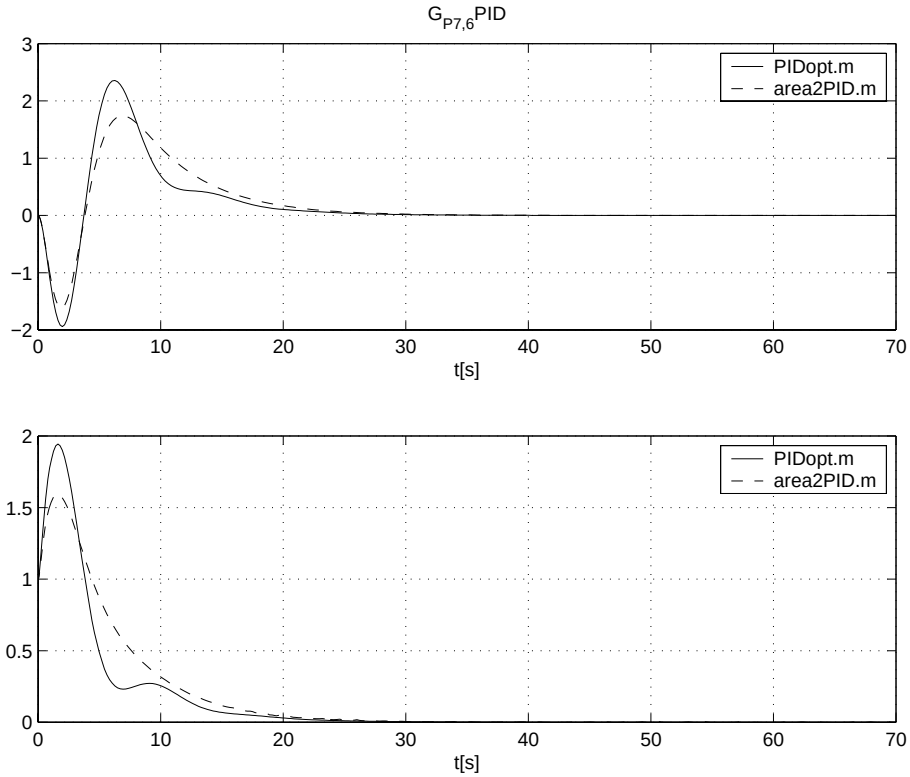


Fig. 102. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

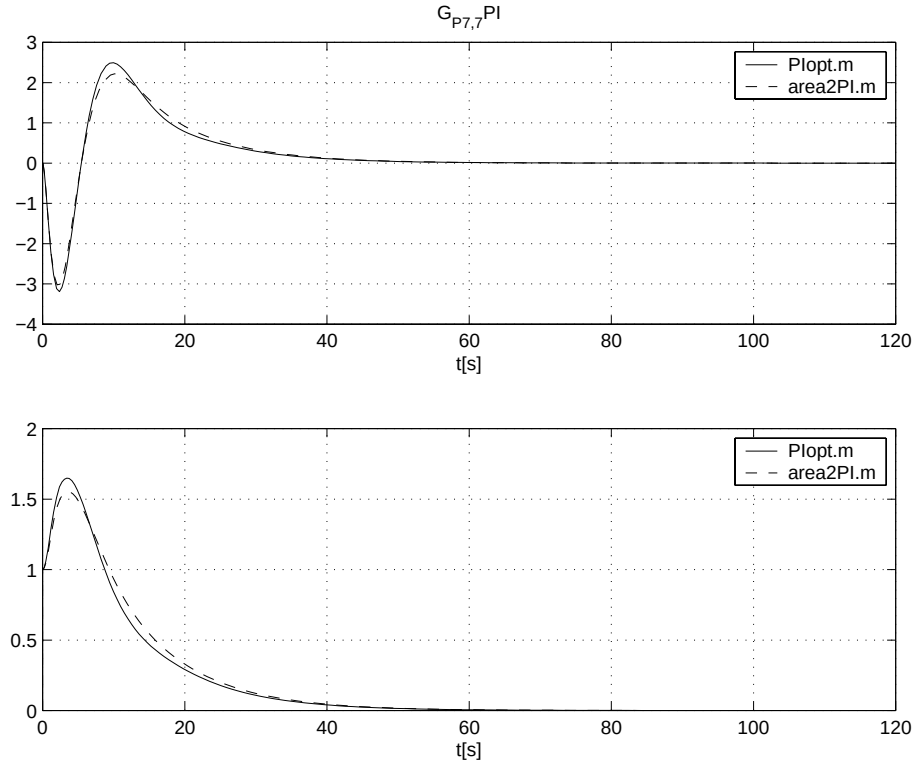


Fig. 103. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

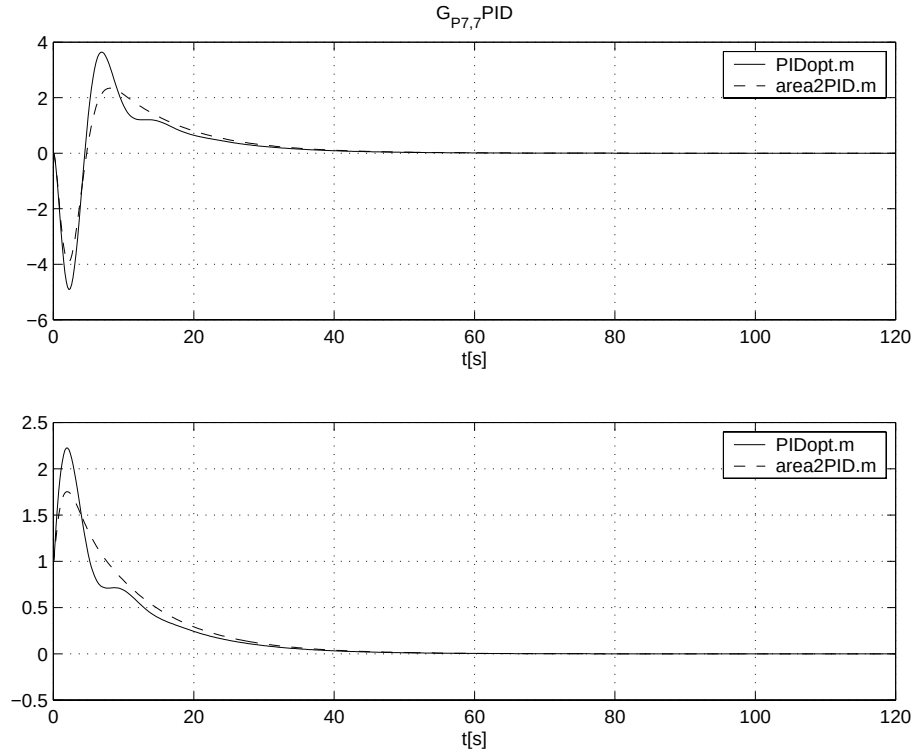


Fig. 104. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

Process 8: $G_{P8} = \dfrac{(1+sT)\cdot e^{-s}}{(s+1)^2}$

Table 29. PI controller parameters calculated by using DRMO method (PIopt.m)

T	0.1	0.2	0.3	0.4	0.5	0.7	1.0
K	0.54731	0.56955	0.59236	0.61518	0.63702	0.67016	0.62772
K_i	0.41279	0.43991	0.46956	0.50169	0.53597	0.6064	0.66237

Table 30. PI controller parameters calculated by using MO method (area2PI.m)

T	0.1	0.2	0.3	0.4	0.5	0.7	1.0
K	0.51027	0.53106	0.55195	0.57225	0.59091	0.61608	0.57143
K_i	0.34837	0.36824	0.38961	0.4124	0.43636	0.48525	0.53571

Table 31. PID controller parameters calculated by using DRMO method (PIDopt.m)

T	0.1	0.2	0.3	0.4	0.5	0.7	1.0
K	1.6973	1.8747	2.0812	2.3153	2.5559	2.5669	1.3081
K_i	0.98326	1.1317	1.3168	1.5447	1.8074	2.001	1.1468
K_d	0.7996	0.85102	0.90503	0.95775	0.99804	0.87906	0.32277

Table 32. PID controller parameters calculated by using MO method (area2PID.m)

T	0.1	0.2	0.3	0.4	0.5	0.7	1.0
K	1.2769	1.3763	1.4754	1.563	1.62	1.5378	1.0203
K_i	0.61273	0.6701	0.73161	0.79344	0.84799	0.88598	0.76014
K_d	0.68035	0.71728	0.74777	0.76403	0.75465	0.61039	0.26182

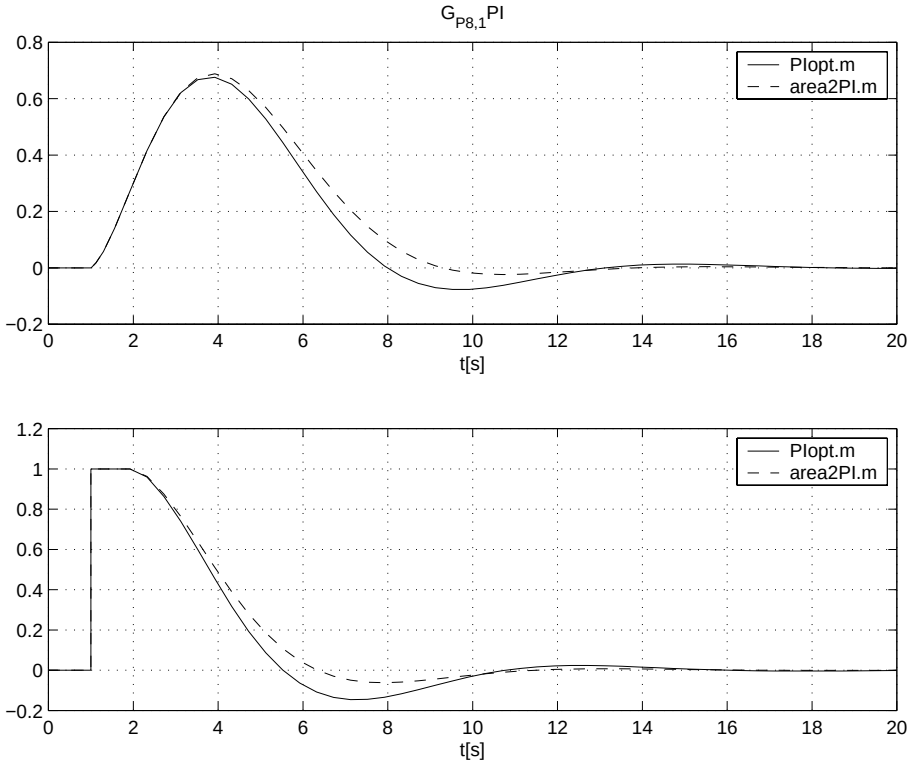


Fig. 105. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

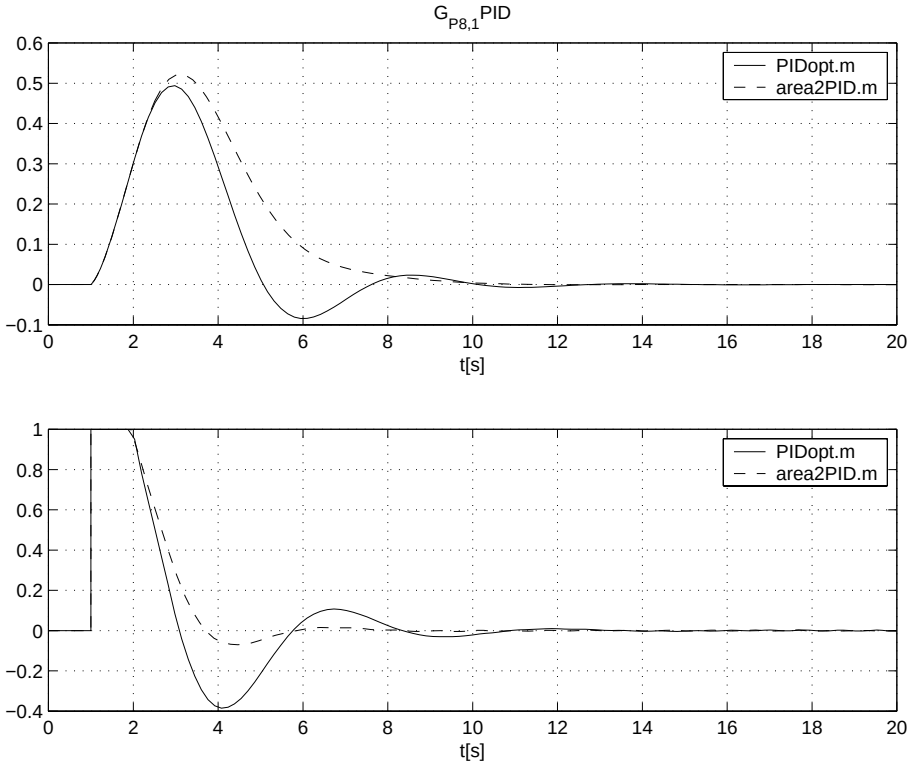


Fig. 106. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

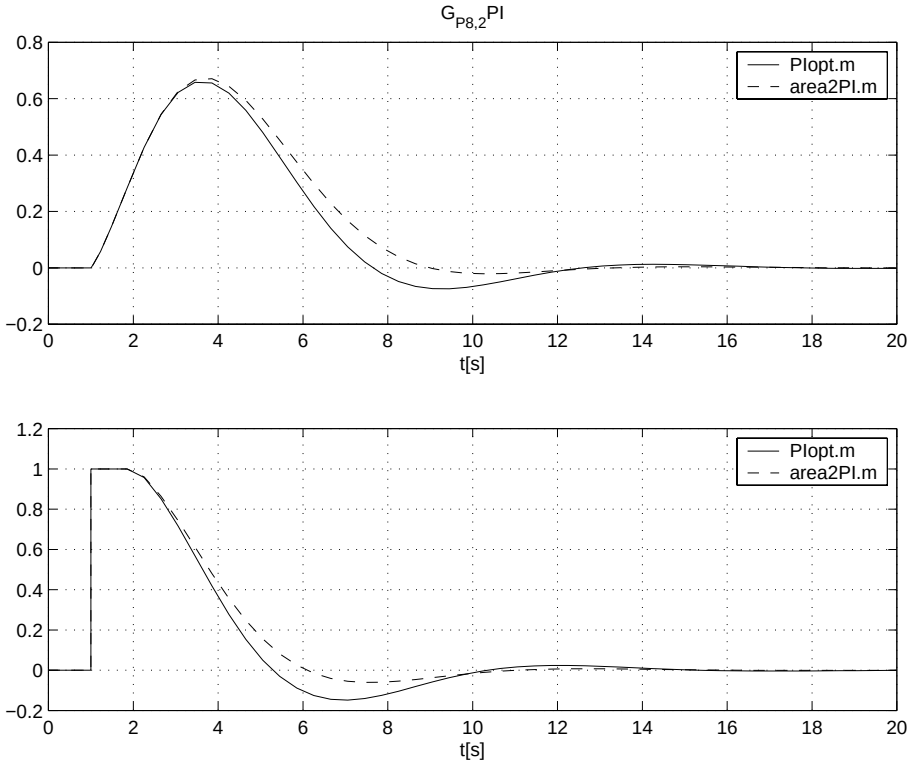


Fig. 107. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

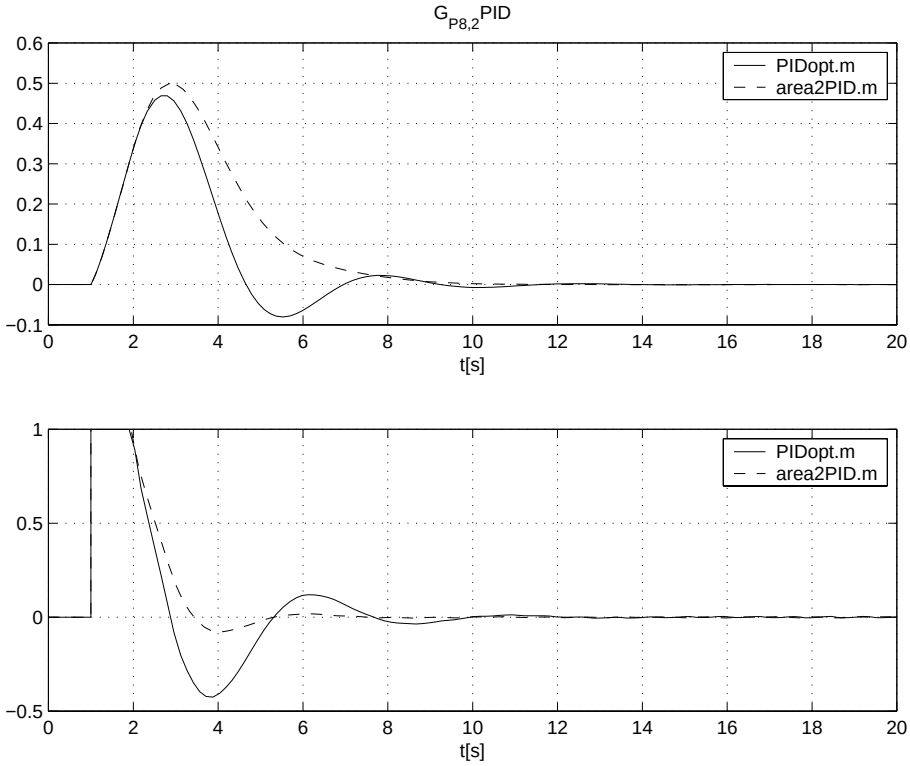


Fig. 108. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

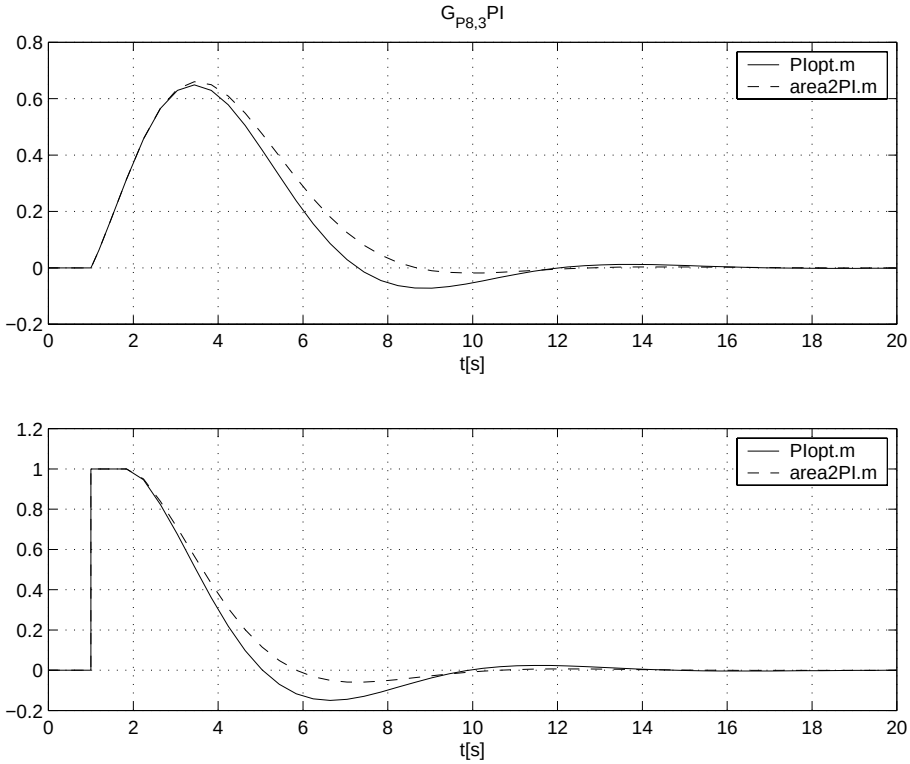


Fig. 109. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

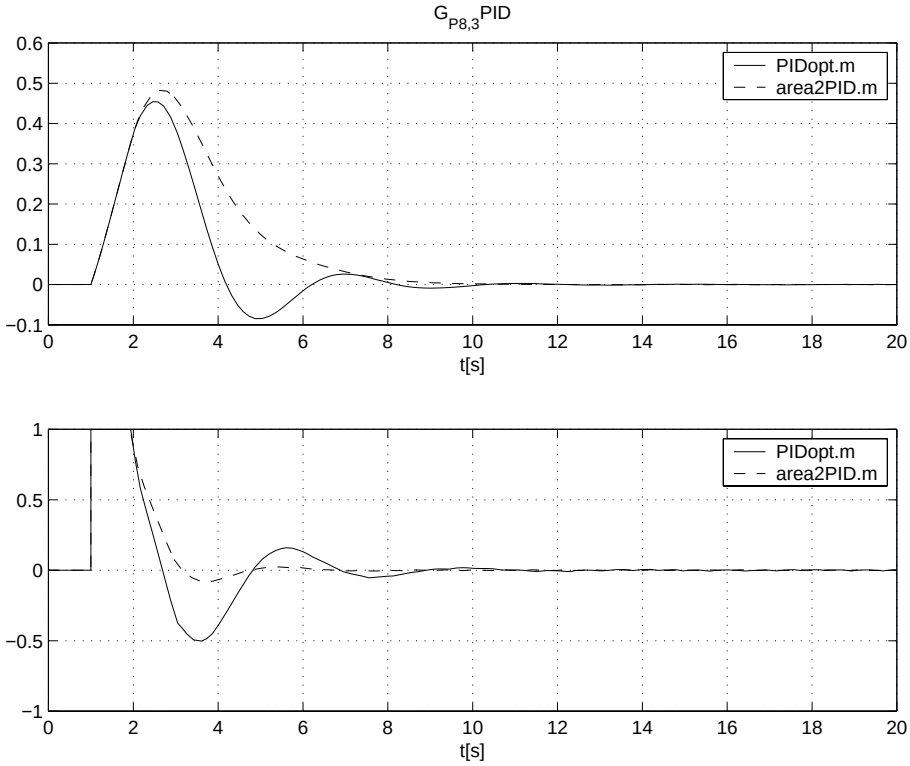


Fig. 110. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

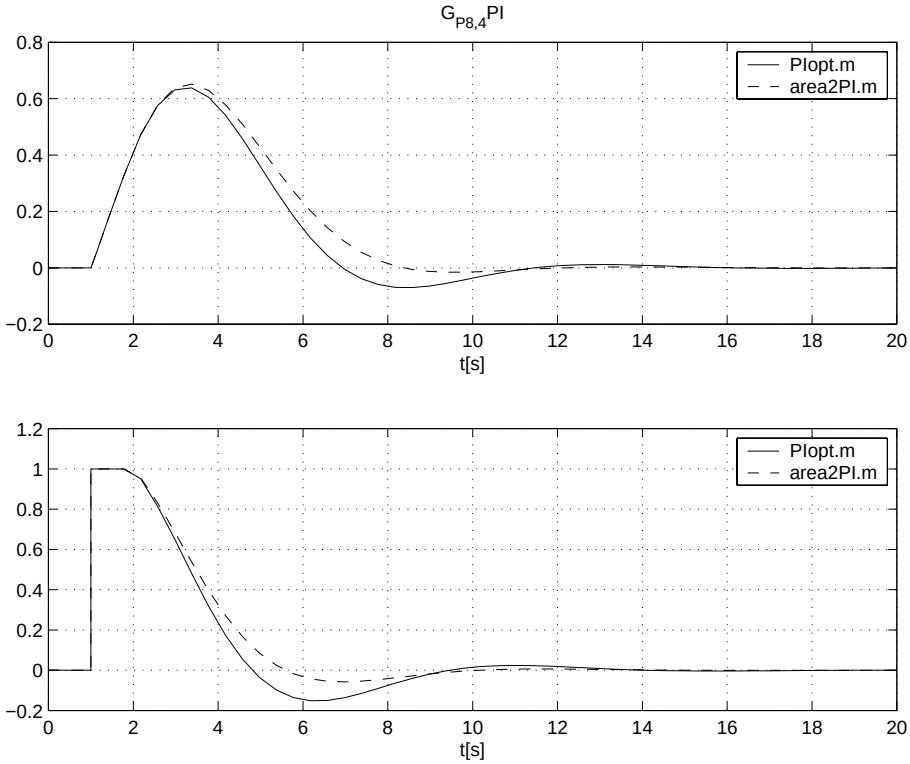


Fig. 111. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

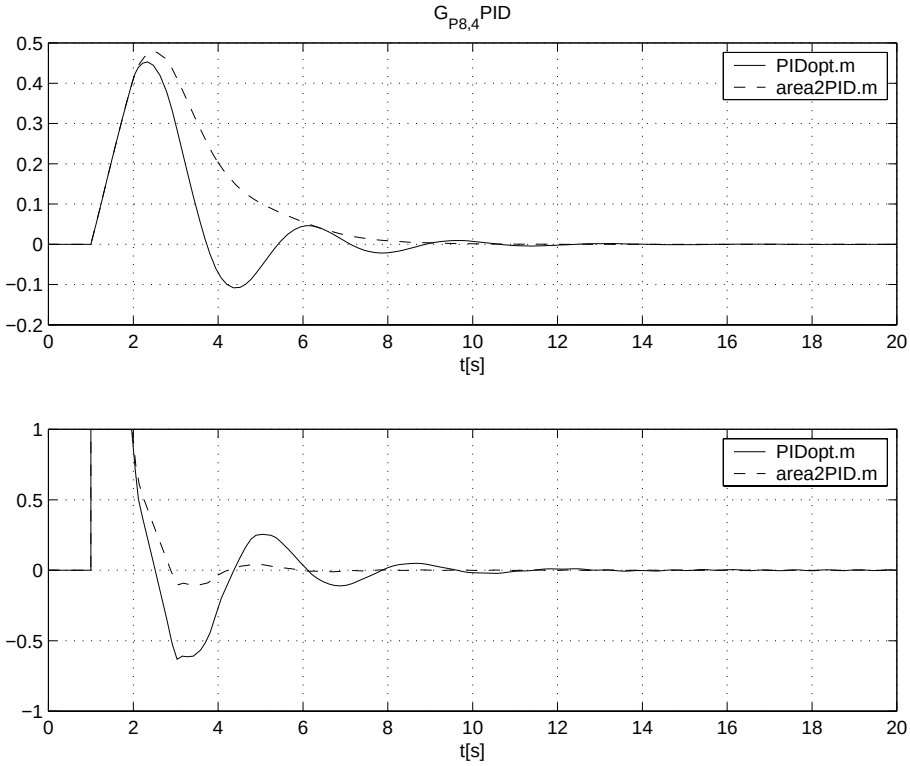


Fig. 112. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

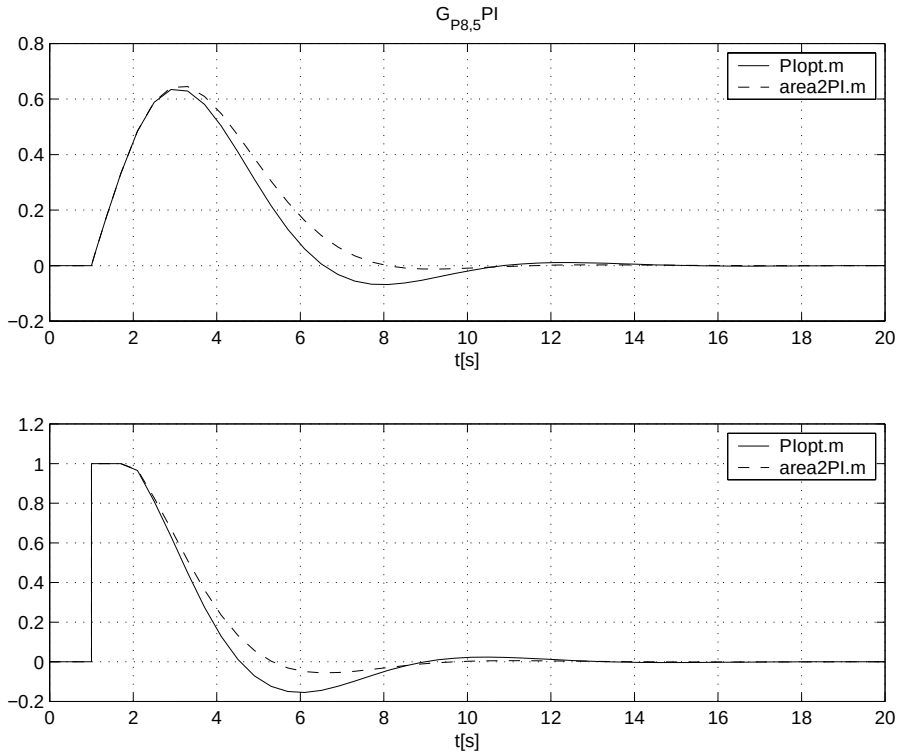


Fig. 113. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

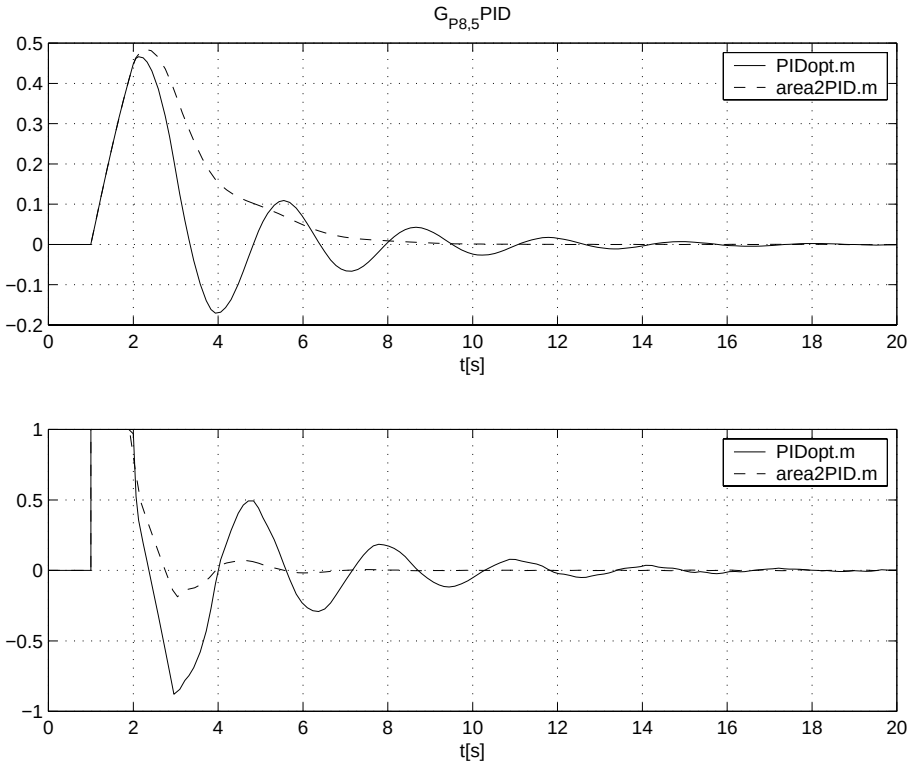


Fig. 114. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

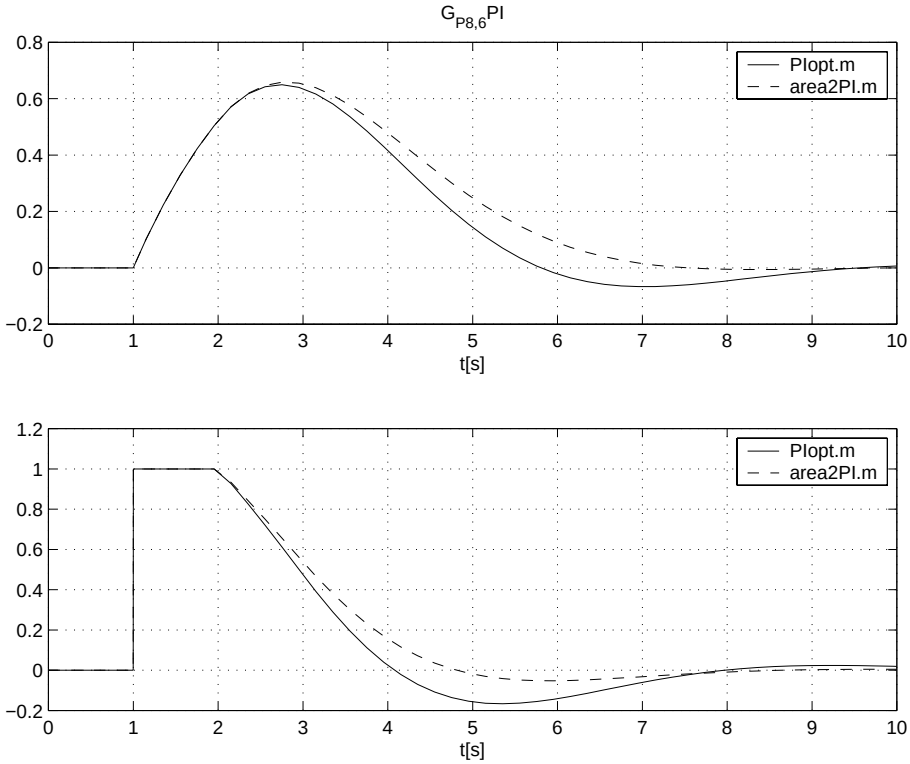


Fig. 115. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

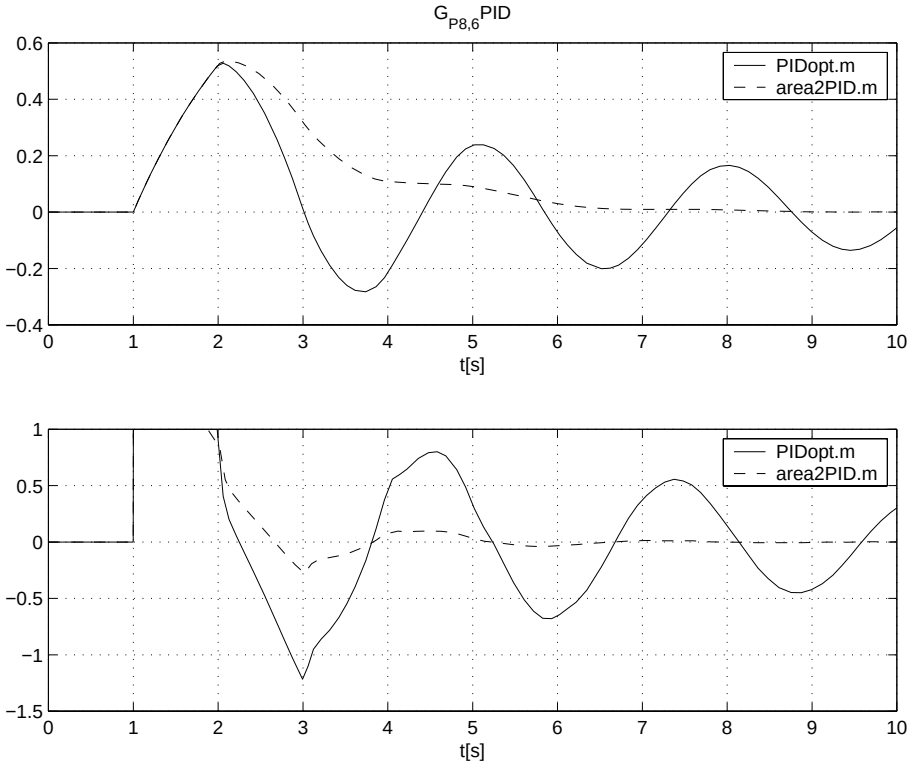


Fig. 116. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

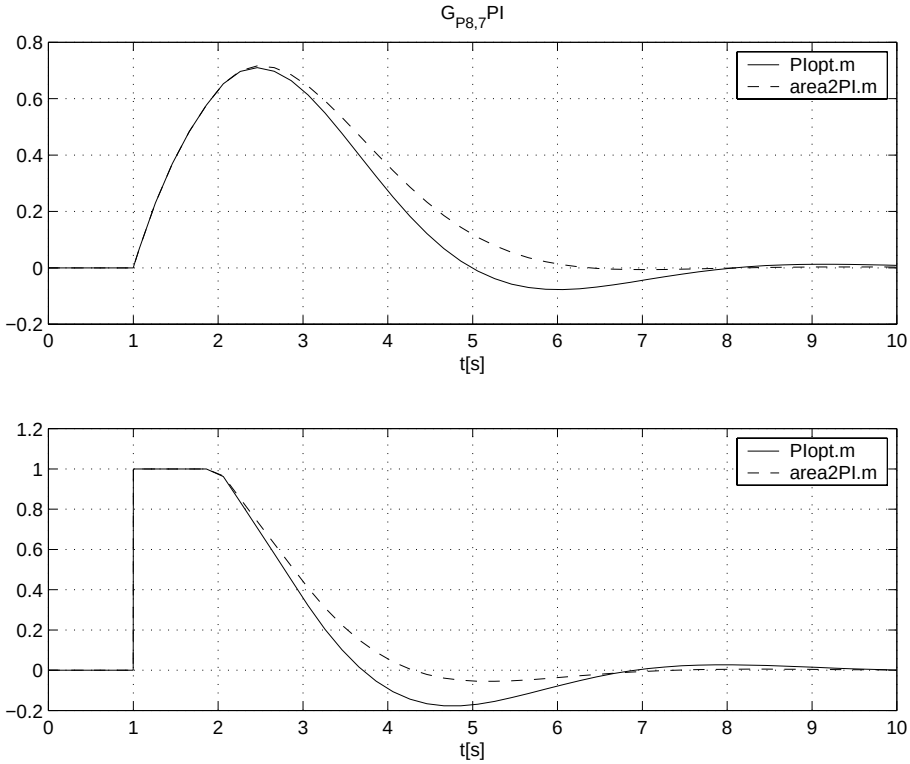


Fig. 117. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

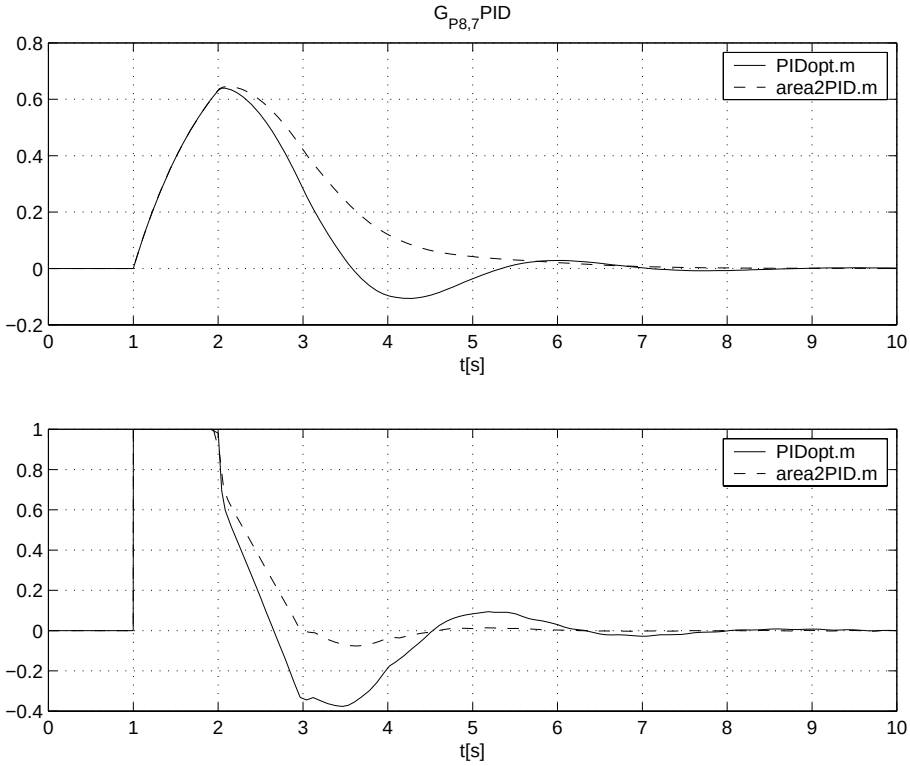


Fig. 118. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

Process 9: $G_{P9} = \dfrac{1}{(1+s)(1+s(1+i\alpha))(1+s(1-i\alpha))}$

Table 33. PI controller parameters calculated by using DRMO method (PIopt.m)

α	0.1	0.2	0.3	0.4	0.5	0.7	1.0
K	0.64665	0.63215	0.6084	0.57601	0.53582	0.43616	0.25658
K_i	0.45191	0.44399	0.43116	0.41397	0.39312	0.34376	0.26317

Table 34. PI controller parameters calculated by using MO method (area2PI.m)

α	0.1	0.2	0.3	0.4	0.5	0.7	1.0
K	0.62032	0.60644	0.58374	0.55288	0.51471	0.42038	0.25
K_i	0.37344	0.36881	0.36125	0.35096	0.33824	0.30679	0.25

Table 35. PID controller parameters calculated by using DRMO method (PIDopt.m)

α	0.1	0.2	0.3	0.4	0.5	0.7	1.0
K	2.915	2.8039	2.6367	2.4342	2.2174	1.804	1.3431
K_i	1.7087	1.6284	1.5098	1.37	1.2251	0.96184	0.68629
K_d	1.4852	1.4431	1.3799	1.3041	1.225	1.0873	1

Table 36. PID controller parameters calculated by using MO method (area2PID.m)

α	0.1	0.2	0.3	0.4	0.5	0.7	1.0
K	2.287	2.2138	2.1019	1.9636	1.8118	1.5101	1.15
K_i	0.929	0.90461	0.86731	0.8212	0.77059	0.67003	0.55
K_d	1.4852	1.4431	1.3799	1.3041	1.225	1.0873	1

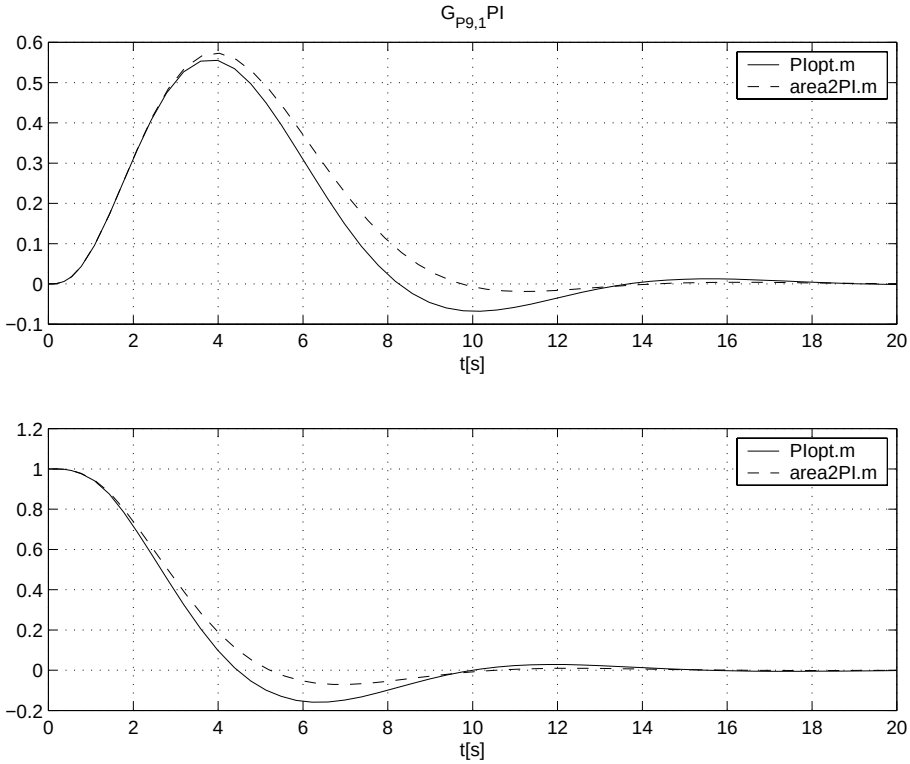


Fig. 119. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

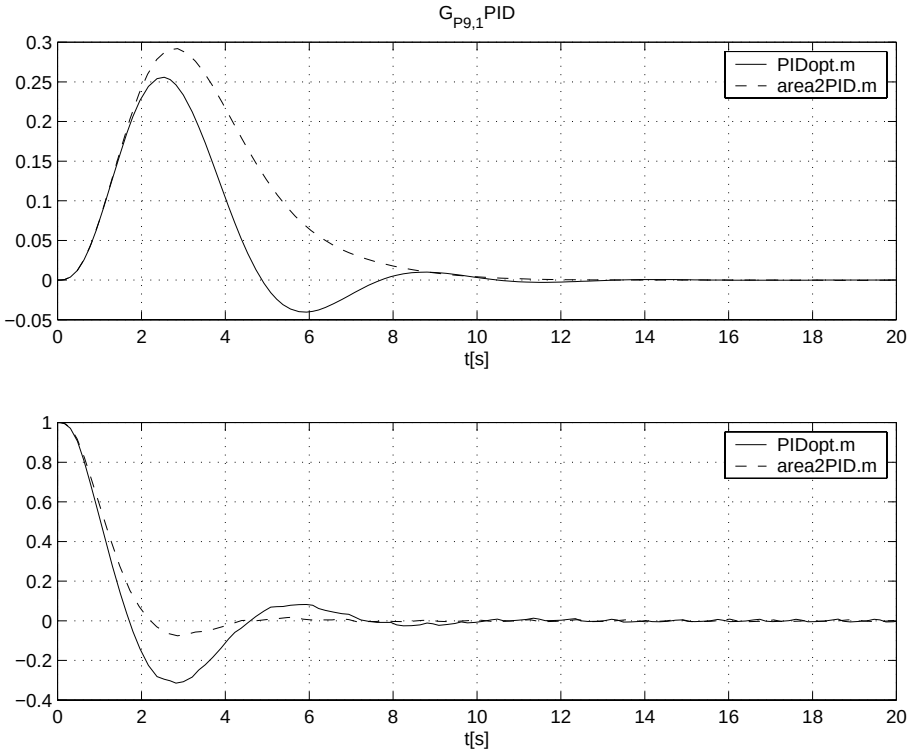


Fig. 120. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

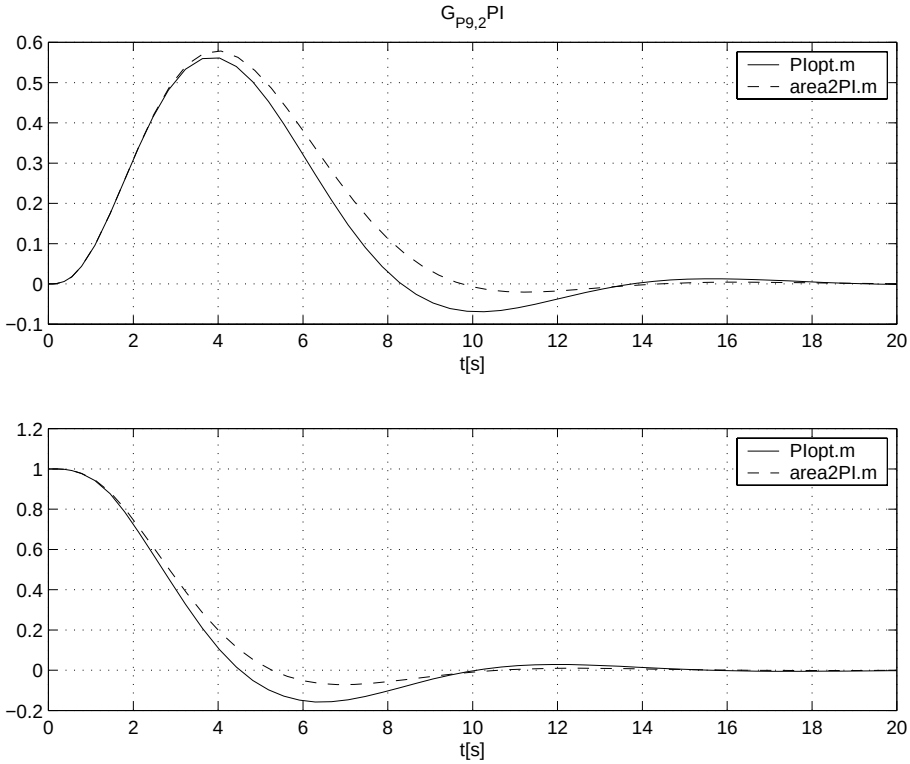


Fig. 121. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

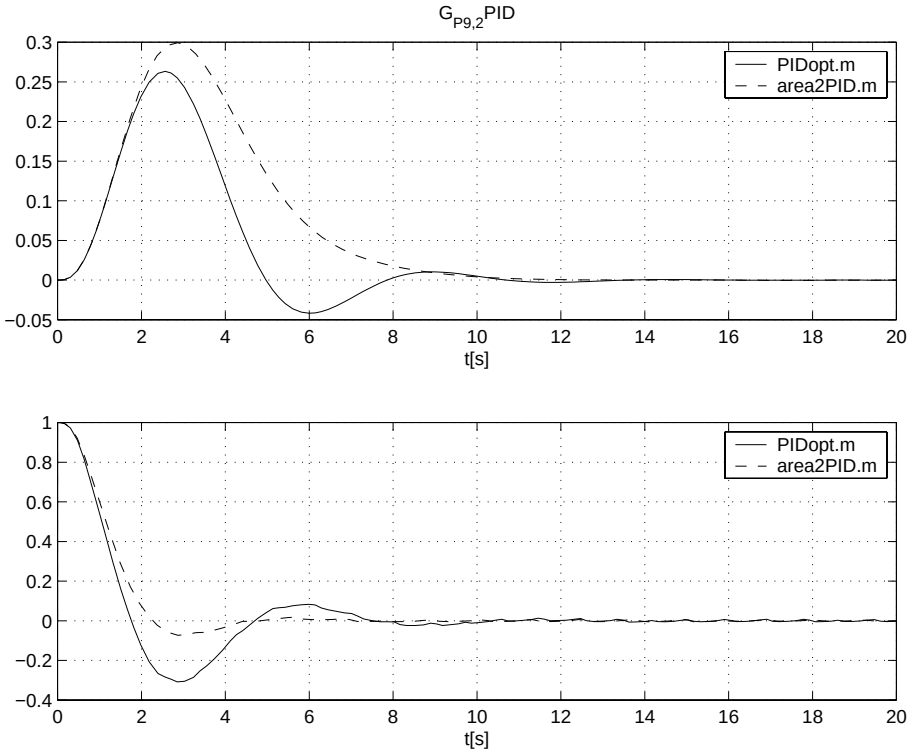


Fig. 122. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

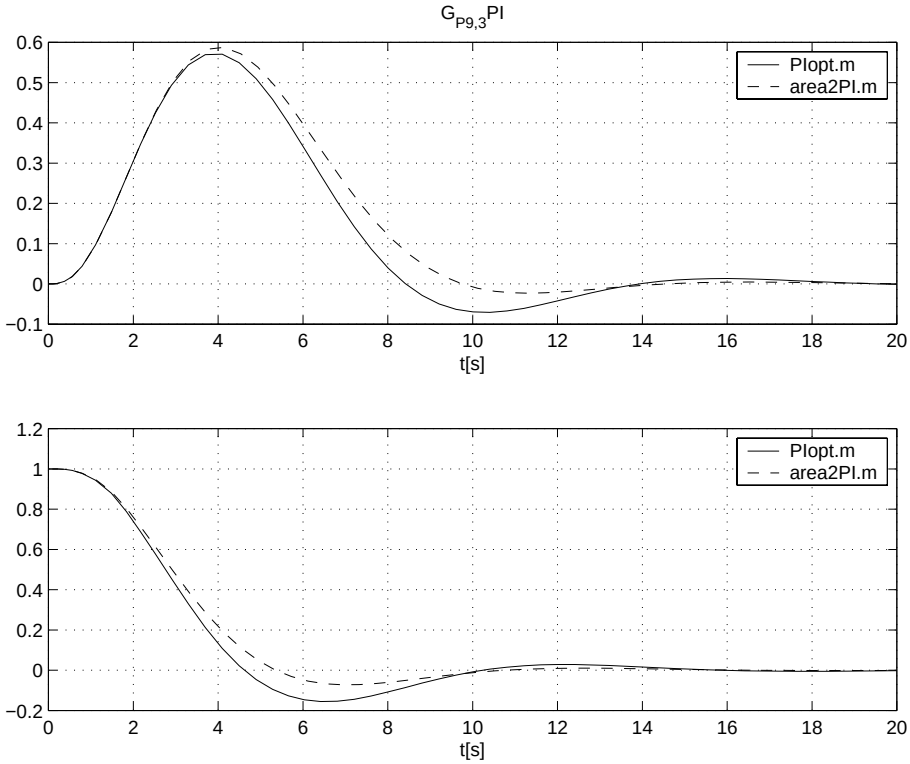


Fig. 123. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

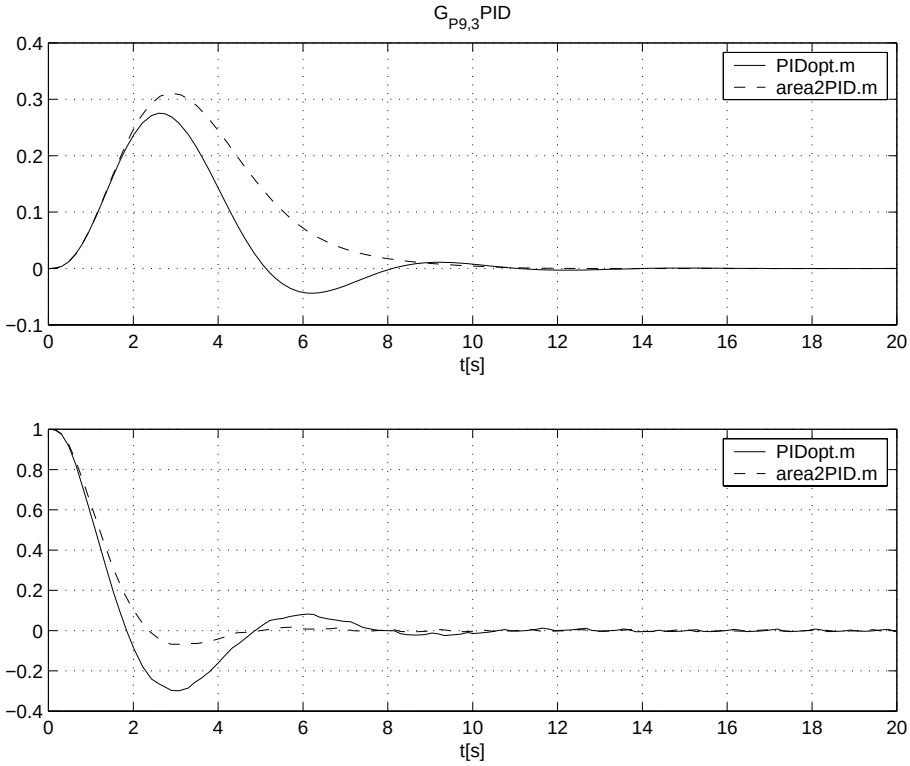


Fig. 124. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

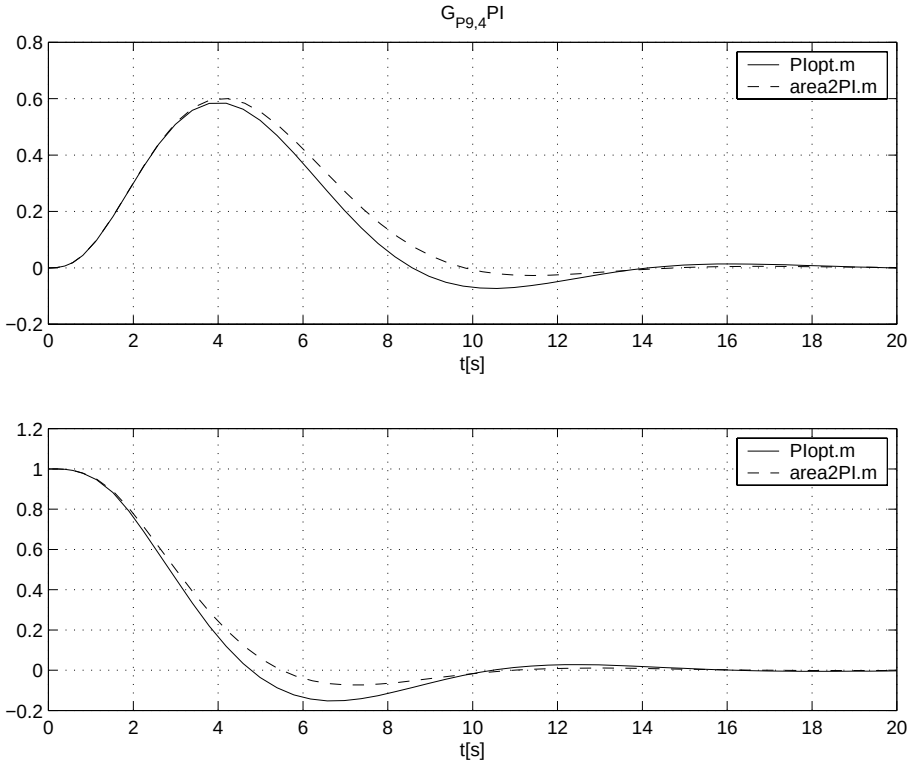


Fig. 125. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

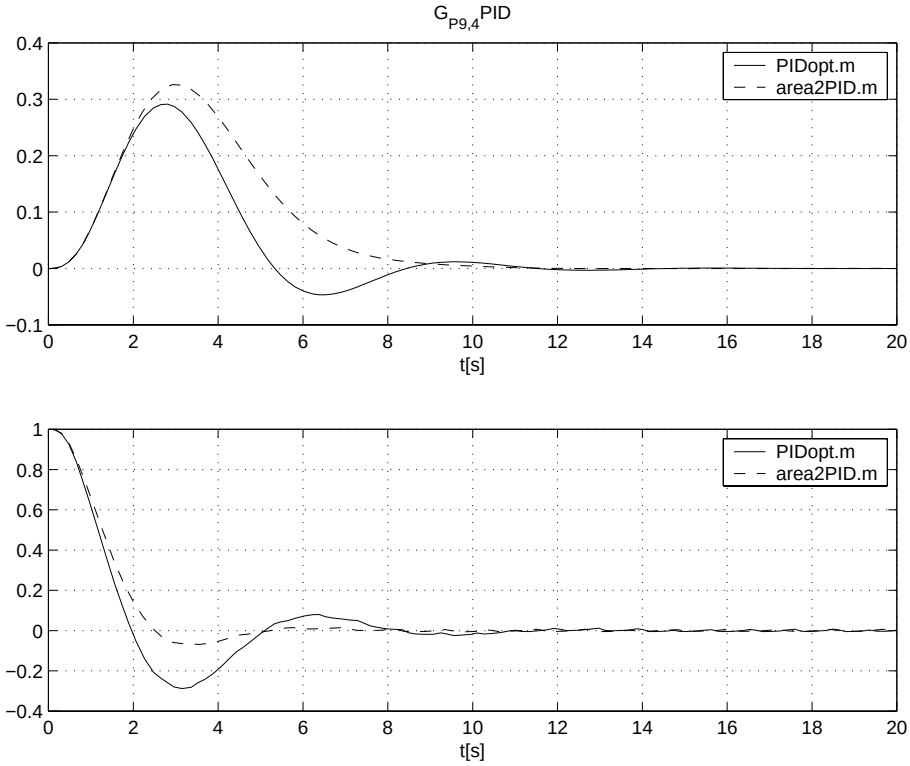


Fig. 126. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

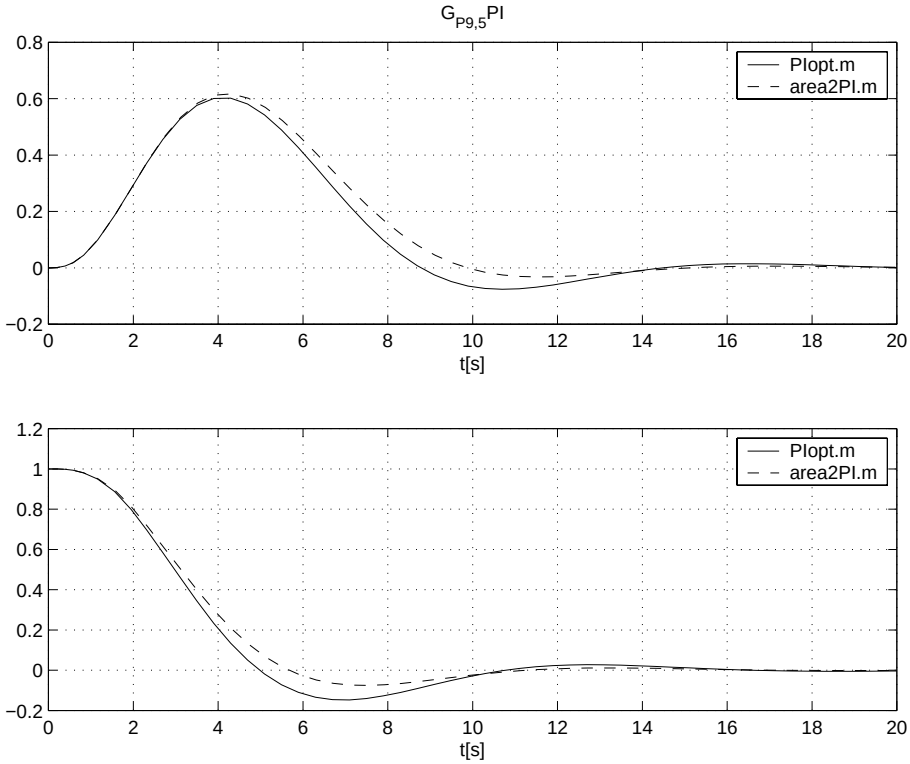


Fig. 127. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

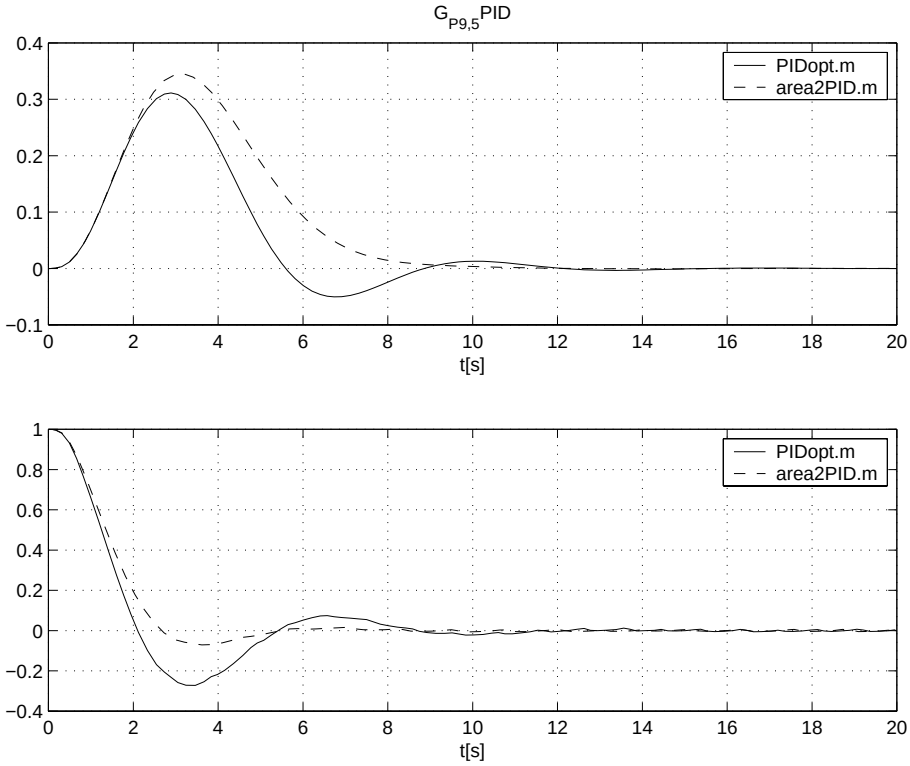


Fig. 128. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

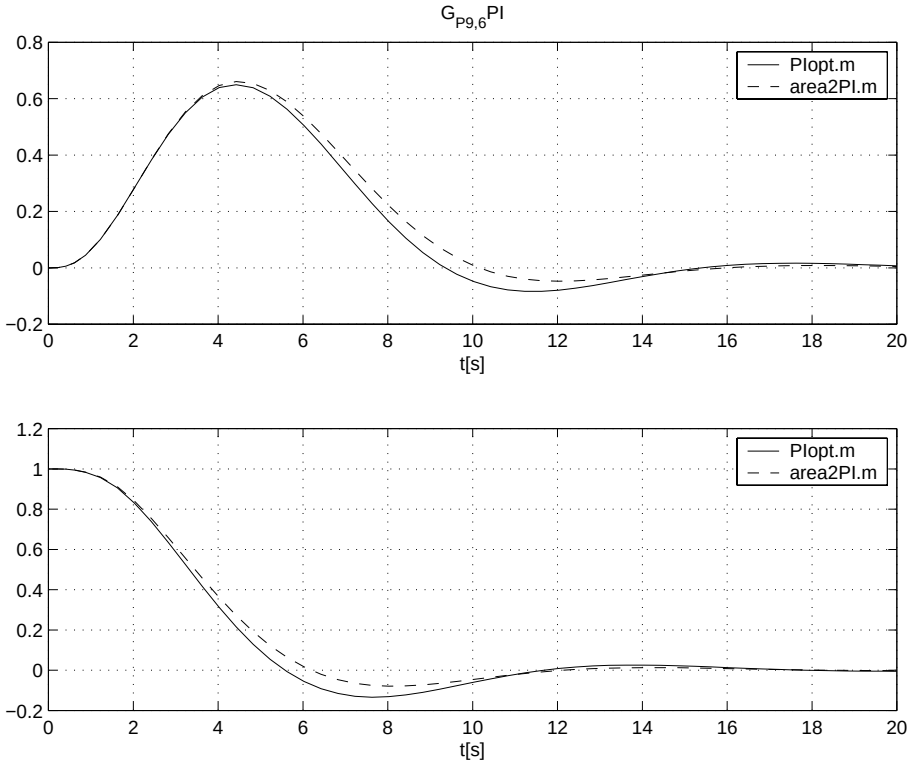


Fig. 129. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

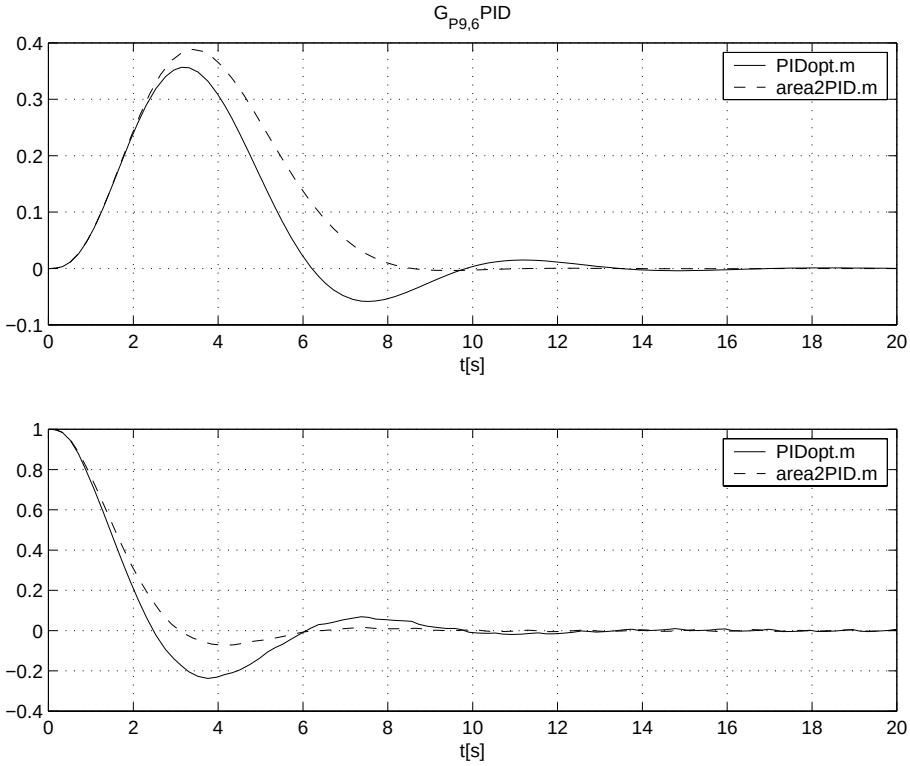


Fig. 130. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

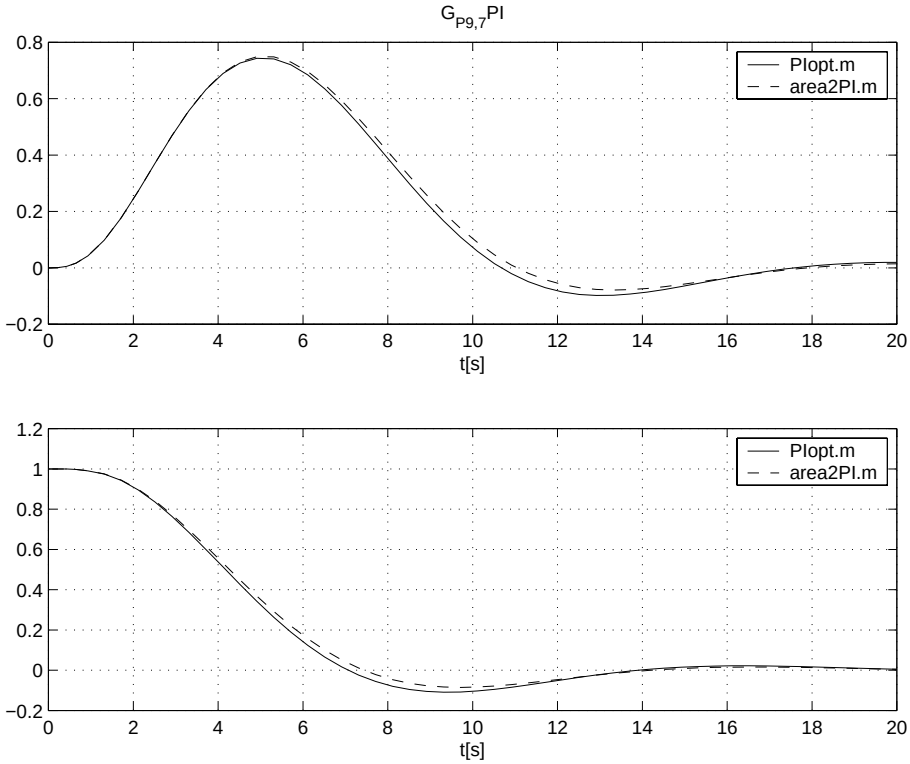


Fig. 131. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PI controllers.

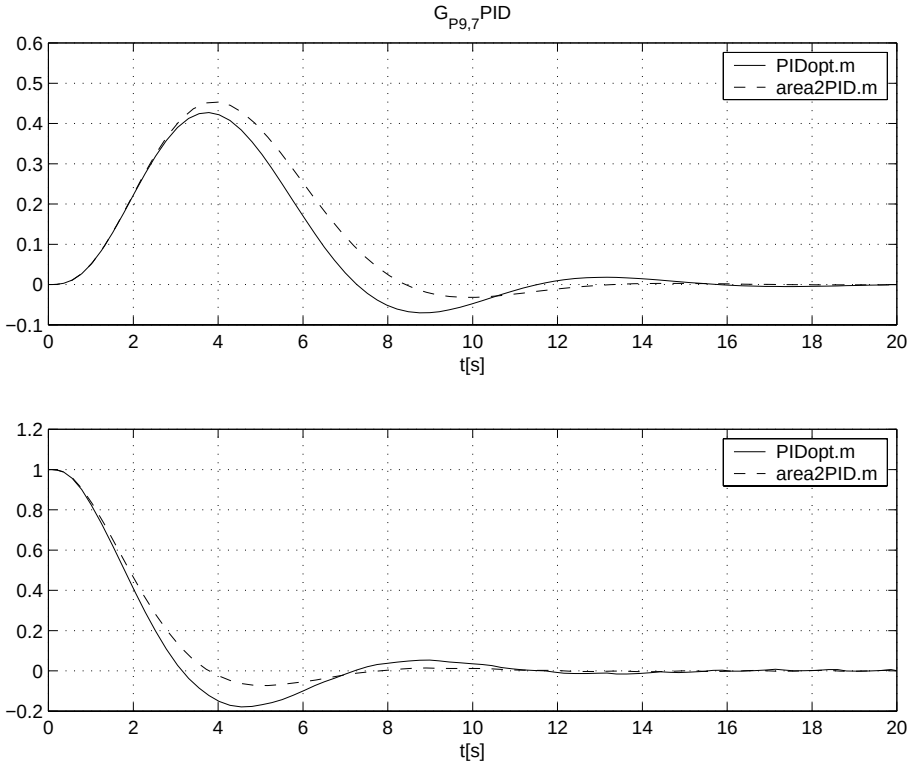


Fig. 132. Process output (upper figure) and input (lower figure) on disturbance when using DRMO (solid lines) and MO (broken lines) tuning method for PID controllers.

Appendix B. List of Matlab and Simulink files used:

- Piopt.m
- Pinopt.m
- PIDopt.m
- PIDnopt.m
- DistPID_Kd2KKi.m
- Area2PID.m
- Area2PI.m
- PIDcontroller.mdl

```
**************************************************************************
% function [Ki,K] = PIopt (A0,A1,A2,A3,Kmax);
%
% Function araa2PI calculates parameters of the PI controller:
%
% u = (Ki/s + K) * e
%
% from the measured areas of the process A0 to A3 (A0 is the process steady-state gain)
% for "optimal" disturbance rejection.
% The parameter Kmax represents the highest allowed open-loop gain K*A0


function [Ki,K] = PIopt (A0,A1,A2,A3,Kmax);

Ceta1 = A0*A0*A3 - 2*A0*A1*A2 + A1*A1*A1;
Ceta2 = A1*A2 - A0*A3;

if (Ceta1 == 0)
  Num = A3;                                          % Numerator
  Den = 2*(A1*A2-A0*A3);                  % Denominator

  if (Num == 0)
    K = 0;
  elseif (Den == 0)
    if (A0 ~= 0)
      K = Kmax/A0;
    else
      K = Kmax;
    end;
  else
    K = Num/Den;
  end;

  Tmp = K*A0;                                                % Nominal gain
```

```
  if (Tmp > Kmax) | (Tmp < 0)
    K = Kmax/A0;
  end;

else

  K = (Ceta2 - sign(Ceta2)*A1*sqrt(A2*A2-A1*A3))/Ceta1;

  Tmp = K*A0;                                          % Nominal gain

  if (Tmp > Kmax) | (Tmp < 0)
    K = Kmax/A0;
  end;

end;

Ki = 0.5*(K*A0 + 1)*(K*A0 + 1)/A1;


**************************************************************************
% function [Ki,K] = PInopt (A0,A1,A2,A3,Kmax);
%
% Function araa2PI calculates parameters of the PI controller:
%
% u = (Ki/s + K) * e
%
% from the measured areas of the process A0 to A3 (A0 is the process steady-state gain)
% for "not optimal" disturbance rejection.
% The parameter Kmax represents the highest allowed open-loop gain K*A0


function [Ki,K] = PInopt (A0,A1,A2,A3,Kmax);

[Ki,K] = area2PI (A0,A1,A2,A3,Kmax);
Ki = 0.5*(K*A0 + 1)*(K*A0 + 1)/A1;


**************************************************************************
%PIDopt.m

function [K,Ki,Kd,CF] = PIDopt (A0,A1,A2,A3,A4,A5,Kmax);

[Ki,K,Kd] = area2PID (A0,A1,A2,A3,A4,A5,Kmax);

Kdv = [0.999*Kd;Kd];
[K,Ki,CF] = DistPID_Kd2KKi (Kdv,A0,A1,A2,A3,A4,A5);
i = 1;

while (i <= 10 & abs(CF(2)-CF(1)) >= 1e-8)
  if (Kd*A0 < 1e-6)
     Kd = 1e-6;
```

```
  end
  Kd = Kd*(1-0.001*CF(2)/(CF(2)-CF(1)));
  Kdv = [0.999*Kd;Kd];
  [K,Ki,CF] = DistPID_Kd2KKi (Kdv,A0,A1,A2,A3,A4,A5);
  i = i+1;
end;

i = 1;
Kmax1 = Kmax/A0;
if (abs(CF(2)-CF(1)) < 1e-8 | K(2)*A0 > Kmax | Ki(2)*A0 < 0)
  while (i <= 10 & K(2) ~= K(1))
    Kd = Kd*(1+0.001*(Kmax1-K(2))/(K(2)-K(1)));
    Kdv = [0.999*Kd;Kd];
    [K,Ki,CF] = DistPID_Kd2KKi (Kdv,A0,A1,A2,A3,A4,A5);
    i = i+1;
  end;

end;

[K,Ki,CF] = DistPID_Kd2KKi (Kd,A0,A1,A2,A3,A4,A5);


**********************************************************************
% function [Ki,K,Kd] = PIDnopt (A0,A1,A2,A3,A4,A5,Kmax);
%
% Function araa2PID_dist calculates parameters of the PID controller:
%
% u = (Ki/s + K + Kd*s) * e
%
% from the measured areas of the process A0 to A5 (A0 is the process steady-state gain)
% for "optimal" disturbance rejection.
% The parameter Kmax represents the highest allowed open-loop gain K*A0

function [Ki,K,Kd] = PIDnopt (A0,A1,A2,A3,A4,A5,Kmax);

[Ki,K,Kd] = area2PID (A0,A1,A2,A3,A4,A5,Kmax);

al1=2*A0^2*A3+2*A1^3-4*A1*A0*A2;
be1=4*A0*A3-4*A0*A1^2*Kd-4*A2*A1+4*A0^2*Kd*A2;
ga1=4*A0*Kd*A2+2*A3+2*A1^2*Kd+6*A1*A0^2*Kd.^2+2*A0^4*Kd.^3;

if (al1==0)                                          % The first order
  K = -ga1./be1;
else                                                 % The second order
  K = 0.5*(-be1 - sqrt(be1.*be1 - 4*al1*ga1))/al1;
end;

Tmp = K*A0;                                          % Nominal gain

if (Tmp > Kmax)
  K = Kmax/A0;
```

```
    Kdv = [0.999*Kd;Kd];
    [Kv,Kiv,CF] = DistPID_Kd2KKi (Kdv,A0,A1,A2,A3,A4,A5);
  i = 1;
  Kmax1 = Kmax/A0;
  while (i <= 10 & Kv(2) ~= Kv(1))
    Kd = Kd*(1+0.001*(Kmax1-Kv(2))/(Kv(2)-Kv(1)));
    Kdv = [0.999*Kd;Kd];
    [Kv,Kiv,CF] = DistPID_Kd2KKi (Kdv,A0,A1,A2,A3,A4,A5);
    i = i+1;
  end;
  K = Kv(2);

elseif (Tmp < 0)
  K = 0;
end;

Ki = 0.5*(1 + 2*A0*K + A0^2*K^2)/(A1 + A0^2*Kd);

if (Ki*A0 < 0)
  Ki = 0;
end;


**************************************************************************
%DistPID_Kd2KKi.m

function [K,Ki,CF] = DistPID_Kd2KKi (Kd,A0,A1,A2,A3,A4,A5);

al1=2*A0^2*A3+2*A1^3-4*A1*A0*A2;
be1=4*A0*A3-4*A0*A1^2*Kd-4*A2*A1+4*A0^2*Kd*A2;
ga1=4*A0*Kd*A2+2*A3+2*A1^2*Kd+6*A1*A0^2*Kd.^2+2*A0^4*Kd.^3;

if (al1==0)                                          % The first order
  K = -ga1./be1;
else                                                 % The second order
  K = 0.5*(-be1 - sqrt(be1.*be1 - 4*al1*ga1))/al1;
end;

Ki = 0.5*(1+2*A0*K+A0^2*K.^2)./(A1+A0^2*Kd);
CF      =      -4*A0*Ki*A4.*Kd-2*A3*Kd+2*A4*K-2*A5*Ki+2*A0*K.^2*A4-2*A0*Kd.^2*A2-
2*A1*K.^2*A3-2*A2^2*Ki.*Kd+A1^2*Kd.^2+A2^2*K.^2+4*A1*Kd*A3.*Ki;


**************************************************************************
% function [Ki,K] = area2PI (A0,A1,A2,A3,Kmax);
%
% Function araa2PI calculates parameters of the PI controller:
%
% u = (Ki/s + K) * e
%
```

```
% from the measured areas of the process A0 to A3 (A0 is the process steady-state gain)
% The parameter Kmax represents the highest allowed open-loop gain K*A0


function [Ki,K] = area2PI (A0,A1,A2,A3,Kmax);

Num = A3;                                          % Numerator
Den = 2*(A1*A2-A0*A3);                             % Denominator

if (Num == 0)
  K = 0;
elseif (Den == 0)
  if (A0 ~= 0)
    K = Kmax/A0;
  else
    K = Kmax;
  end;
else
  K = Num/Den;
end;

Tmp = K*A0;                                        % Nominal gain

if (Tmp > Kmax) | (Tmp < 0)
  K = Kmax/A0;
end;

Ki = (K*A0 + 0.5)/A1;


**************************************************************************
% function [Ki,K,Kd] = area2PID (A0,A1,A2,A3,A4,A5,Kmax);
%
% Function araa2PID calculates parameters of the PID controller:
%
% u = (Ki/s + K + Kd*s) * e
%
% from the measured areas of the process A0 to A5 (A0 is the process steady-state gain)
% The parameter Kmax represents the highest allowed open-loop gain K*A0


function [Ki,K,Kd] = area2PID (A0,A1,A2,A3,A4,A5,Kmax);

Num = A3*A3 - A1*A5;                                    % Numerator
Den = 2*(A1*A2*A3+A0*A1*A5-A1*A1*A4-A0*A3*A3);          % Denominator

if (Num == 0)
  K = 0;
elseif (Den == 0)                                       % Division with zero
  if (A0 ~= 0)
    K = Kmax/A0;
```

```
  else
    K = Kmax;
  end;
else                                                % Everything is OK
  K = Num/Den;
end;

Tmp = K*A0;                                         % Nominal gain

if (Tmp > Kmax) | (Tmp < 0)          % If the calculated gain is too high or negative
  K = Kmax/A0;
end;

if (A1 ~= 0)
  Kd = (2*K*(A1*A2-A0*A3)-A3)/(2*A1*A1);
else
  Kd = 0;
end;

Tmp = Kd*A0;                                        % Nominal gain of D-term

if (Tmp < 0)                                % If derivative time constant is negative
  Kd = 0;
end;

if (Kd == 0)                                % If negative, calculate PI controller parameters
  [Ki,K] = area2PI (A0,A1,A2,A3,Kmax);
else
  Ki = (2*K*A0 + 1)/(2*A1);
end;
```

A MATLAB Simulink model pid_controller.mdl, shown on figure below has been used to test the process responses.

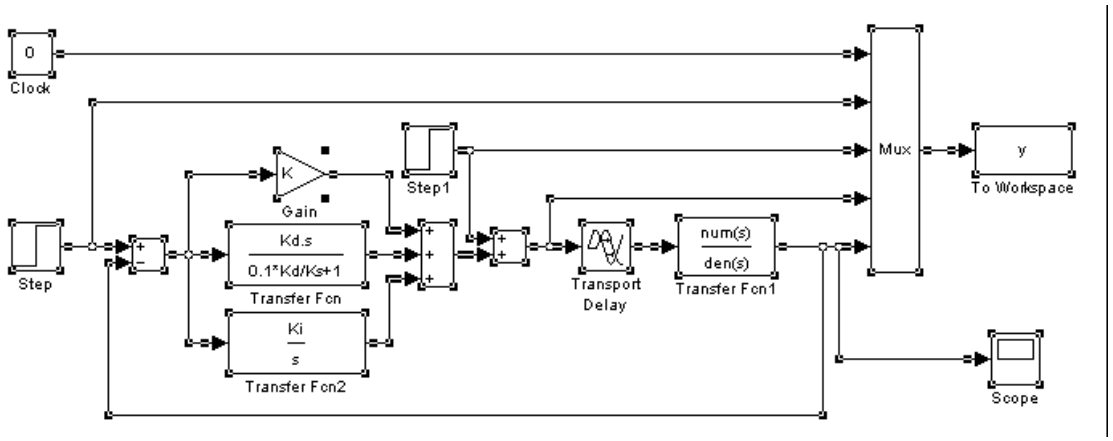


Fig.: pid_controller.mdl