Study on disturbance-rejection magnitude optimum method
decay ratios

Satja Lumbar¹, Damir Vrančić¹

August 2006
1. INTRODUCTION 1

2. MAGNITUDE OPTIMUM CRITERION FOR TRACKING AND CONTROL 1

3. STUDY OF DECAY RATIOS 4

4. COMPARISON TO SOME OTHER METHODS 25

4.1 ÄSTROM AND HAGGLUND 25
4.2 PANAGOPoulos TUNING RULES 27
4.3 RESULTS 27

5. CONCLUSIONS 31

6. REFERENCES 31

APPENDIX A 32

REGCOMPAREPI_FINAL.M 32
GP_BASE2.M 35
TC2AREAS1.M 39
PIOPT.M 40
BO_PI.M 41
CONTROLLER.MDL 42
EX_FIND.M 42
REGCOMPAREPI_APFINAL.M 43
GP_BASEPI.M 48
OL_PAR1.M 49
PI_PANAGOPOULOS.M 50
PI_ASTROM.M 51
1. Introduction

PID controllers are the most widely used controllers in the process industry. It has been acknowledged that more than 95% of the control loops used in the process control is of the PID type, of which most are the PI type [1].

Today, the most often applied tuning rules for PID controllers are those based either on the measurement of process step response or on the detection of a particular point on the Nyquist curve of the process (usually one related to the ultimate magnitude and frequency of the process by using relay excitation).

Apart from standard tuning rules, such as Ziegler-Nichols, Cohen-Coon, Chien-Hrones-Reswick, or refined Ziegler-Nichols rules, more sophisticated tuning approaches have been suggested. They are usually based on more demanding process identification algorithms or tuning procedures, like non-convex optimization, gain and phase specification, IMC controller design, or identification of multiple points in frequency domain [3,4,5,6,7,8,17].

One of the frequent demands in the time domain when dealing with closed-loop regulation is, amongst other requirements, the closed-loop response decay ratio, online optimization of which is usually relatively demanding. While using the disturbance rejection magnitude optimum (DRMO) tuning method [9,10,15,16] it came to our attention, that the decay ratios of the responses have quite similar values for a diversity of process models. Furthermore the DRMO method is relatively simple to apply since it does not require any form of optimization (i.e. retuning). Our intention in this paper is to test this method on a large batch of process models, analyze the decay ratios of the closed loop responses and compare them to some other modern tuning methods.

This paper is set out as follows. Section 2 shortly describes the original MO tuning method and its modification, the DRMO method for PI controllers. A study on uniformity of the decay ratios of the DRMO method is given in Section 3. The results of this study are then compared with results of two methods that set the value of maximum sensitivity function [1,3,17] in Section 4. Lastly, the conclusions are provided in Section 5.

2. Magnitude optimum criterion for tracking and control

Fig. 1 Typical closed-loop configuration using a 1DOF controller.

Fig. 1 shows the process in a closed-loop configuration with 1DOF controller, where signals $r$, $u$, $d$, $y$ and $e$ represent a reference, controller output, input disturbance, process output and output error, respectively. One possible controller design objective is to maintain the closed-loop magnitude (amplitude) as flat and as close to unity over as wide a frequency range as possible [10].

If we assume, that there is no input disturbance ($d = 0$), the transfer function between the reference and the process output is:

$$G_{cl}(s) = \frac{Y(s)}{R(s)} = \frac{G_p(s)G_c(s)}{1 + G_p(s)G_c(s)}$$

(1)
The controller is determined in such a way that

\[ G_{CL}(0) = 1 \]  
\[ \lim_{\omega \to \infty} \left[ \frac{d^{2k}|G_{CL}(j\omega)|}{d\omega^{2k}} \right] = 0, \quad k = 1,2,...,k_{\text{max}} \]  

for as many \( k \) as possible [9,10,15].

Eq. (2) is simply fulfilled by using a controller structure containing the integral term\(^1\), because the steady-state control error is zero. The number of conditions in Eq. (3) that can be satisfied depends on controller order.

For a typical closed-loop transfer function,

\[ G_{CL}(s) = \frac{1 + f_1s + f_2s^2 + \cdots}{1 + e_1s + e_2s^2 + \cdots} \]  

the conditions (3) are fulfilled by using the following set of equations [18]:

\[ \sum_{i=0}^{2k} (-1)^i (e_i e_{2k-i} - f_i f_{2k-i}) = 0, \quad k = 1,2,...,k_{\text{max}} \]  

where \( e_0 = f_0 = 1 \) and \( k_{\text{max}} \) represents the number of controller parameters (\( k_{\text{max}} = 2 \) for PI controller structure).

Let us now calculate the parameters of the PI controller, which can be described by the following transfer function:

\[ G_c(s) = K_p + \frac{K_i}{s} = K_p \left( 1 + \frac{1}{sT_i} \right) \]  

where \( K_p, K_i, \) and \( T_i \) are proportional gain, integral gain and integral time constant respectively.

For a PI controller structure two expressions are obtained [15]

\[ K_p = \frac{A_3}{2(A_1A_2 - A_0A_3)} \]  
\[ K_i = \frac{A_2}{2(A_1A_2 - A_0A_3)} \]  

where symbols \( A_0\) to \( A_3 \) represent the so called »characteristic areas« of the process [12, 13,15]:

---

\(^1\) Under the condition that the closed-loop response is stable
Note that $A_0$ equals the steady-state gain of the process. The name “characteristic areas” is associated with the fact that they can be calculated from nonparametric process model in time domain by changing the steady state of the process and performing multiple integrations on the process input $[u(t)]$ and output $[y(t)]$ signals [7, 12]. This procedure is relatively easy to perform in practice and does not require explicit identification of the process transfer function parameters. However, by using the original MO method, disturbance rejection is degraded when dealing with lower-order processes, since slow process poles might become almost entirely cancelled by controller zeros.

This phenomenon is expected, since the MO method aims at achieving good reference tracking, so it optimizes the transfer function between the reference and the process output $(r=1, d=0)$ $G_{CL}(s)=Y(s)/R(s)$ instead of the transfer function between the input disturbance $(r=0, d=1)$ and the process output $G_{CLD}(s)=Y(s)/D(s)$. Optimizing the latter would prove itself a fruitless attempt, since this transfer function is not compatible with MO criterion as $G_{CLD}(0)=0$.

The modification of MO criteria has been proposed which optimizes a modified transfer function between the input disturbance $(r=0, d=1)$ and the process output [15]:

$$
G_{CLD}(s) = \frac{K_p}{s} + \frac{G_p(s)}{1 + G_p(s)G_C(s)}
$$

By applying the transfer function (10) into equations (4) and (5) and solving the first two equations $(k=1$ and $k=2$) for $K_p$ and $K_i$, the following expressions are obtained:

$$
K_p^2(A_0^2A_3 - 2A_0A_1A_2 + A_1^3) + 2K_p(A_0A_3 - A_1A_2) + A_3 = 0
$$

$$
K_i = \frac{(1 + K_pA_0)^2}{2A_1}
$$

In general, Eq. (12) is of the second order and can be solved analytically. Naturally, only real solutions can be used. This requirement reads as follows:

$$
A_2^2 \geq A_1A_3
$$

According to [15,11] the appropriate pair $(K_p, K_i)$ is the one with smaller absolute value for $K_p$. This conclusion leads to a mathematical expression for $K_p$:
\[ K_p = \frac{\xi_2 - \text{sgn}(\xi_1)A_1A_2^2 - A_1A_3 \sqrt{\xi_1^2}}{\xi_1} \]  

(14)

where

\[ \xi_1 = A_0^2A_1 - 2A_0A_1A_2 + A_1^3, \]
\[ \xi_2 = A_1A_2 - A_0A_3. \]  

(15)

However, one must be aware that when using reduced-order controllers, the MO method does not ensure closed-loop stability [12], which is also the case with other nonparametric tuning methods. The sufficient stability conditions are [15,20]

\[ A_0K_i \geq 0 \]
\[ A_iK_i - A_0K_p < 1 \]
\[ (-1)^j (A_jK_i - A_j^{-1}K_p) \geq 0; j = 2,\ldots \]  

(16)

3. Study of decay ratios

In process control it is always an advantage to know the shape of the closed-loop response on input disturbance in advance. In that aspect, the shape of the response should not depend on the controlled process too much. But which criterion for equality of the shape of the response is most suitable? Some methods [1,3,17] use the so called maximum sensitivity function defined as:

\[ M_s = \max_\omega \left| \frac{1}{1 + G_s(i\omega)G_c(i\omega)} \right| \]  

(17)

In the time domain a noteworthy characteristic of a closed-loop response that can be considered as a parameter of the shape of the closed-loop response is its decay ratio, which we define with the following relation:

\[ dr = \frac{B}{A} = \frac{|p_3| + |p_4|}{|p_1| + |p_2|} \]  

(18)

where \( p_1, p_2, p_3 \) and \( p_3 \) are the first, second, third and fourth of the closed-loop response as shown in Fig. 2.
The aim of this study is to evaluate a set of decay ratios over a wide batch of processes in a closed-loop with a PI controller tuned with the DRMO tuning method. Anticipation is that the method in question gives decay ratios that are relatively consistent and independent of the process model. To determine this, we used a batch of process models, covering processes of lower and higher orders, processes with delay, non-minimum phase processes and processes with zeros in left half-plane. Constants were chosen such, that the sum of all in a single process model equals 12:

\[ G_{p1} = \frac{e^{-sT_{\text{delay}}}}{1 + sT}; \]

\[ T_{\text{delay}} = \{12,11,10,6,2,1\}; \quad T = 12 - T_{\text{delay}}; \]  \hspace{1cm} (19)

\[ G_{p2} = \frac{e^{-sT_{\text{delay}}}}{(1 + sT)^2}; \]

\[ T_{\text{delay}} = \{11,10,8,6,4,1\}; \quad T = \frac{12 - T_{\text{delay}}}{2}; \]  \hspace{1cm} (20)

\[ G_{p3} = \frac{1}{(1 + sT_1)(1 + sT_2)}; \]

\[ T_1 = \{11,10,9,8,7,6\}, \quad T_2 = 12 - T_1; \]  \hspace{1cm} (21)

\[ G_{p4} = \frac{1}{(1 + sT_1)^2(1 + sT_2)^2}; \]

\[ T_1 = \{5.5,5.4,5.4,3.5,3\}, \quad T_2 = \frac{12 - 2T_1}{2}; \]  \hspace{1cm} (22)

\[ G_{p5} = \frac{1}{(1 + sT)^n}; \]
\[ n = \{3,4,5,6,7,8\} \], \[ T = \frac{12}{n} \];  

\[ G_{p6} = \frac{1}{(1+sT)(1+skT)(1+sk^2T)(1+sk^3T)}; \]

\[ n = \{0.9,0.7,0.5,0.4,0.3,0.2\} \], \[ T = \frac{11}{k+k^2+k^3} \];  

\[ G_{p7} = \frac{1-sT_z}{(1+sT_p)^3} \]

\[ T_z = \{1,2,3,4,5,6\} \], \[ T_p = \frac{12-T_z}{3} \];  

\[ G_{p8} = \frac{1+sT_z}{(1+sT_p)^3} \]

\[ T_z = \{0.5,1.2,5,4.5,5,5,7\} \], \[ T_p = \frac{12+T_z}{3} \];  

\[ G_{p9} = \frac{1}{(1+4s)(1+4s(1-i\alpha))(1+4s(1+i\alpha))} \]

\[ \alpha = \{0.2,0.3,0.4,0.5,0.7,1\} \];  

Closed-loop process responses on an input disturbance are plotted in Figs. 3 to 56. The calculated decay ratios for all the process models are depicted in Fig. 57.

**Fig. 3** Response on input disturbance \((r=0, d=1)\) of process \(G_{PI,1}\)
Fig. 4 Response on input disturbance ($r=0$, $d=1$) of process $G_{P1,2}$

Fig. 5 Response on input disturbance ($r=0$, $d=1$) of process $G_{P1,3}$

Fig. 6 Response on input disturbance ($r=0$, $d=1$) of process $G_{P1,4}$
Fig. 7 Response on input disturbance \((r=0, d=1)\) of process \(G_{P1,5}\)

Fig. 8 Response on input disturbance \((r=0, d=1)\) of process \(G_{P1,6}\)

Fig. 9 Response on input disturbance \((r=0, d=1)\) of process \(G_{P2,1}\)
Fig. 10 Response on input disturbance \((r=0, d=1)\) of process \(G_{P2,2}\)

Fig. 11 Response on input disturbance \((r=0, d=1)\) of process \(G_{P2,3}\)

Fig. 12 Response on input disturbance \((r=0, d=1)\) of process \(G_{P2,4}\)
Study on disturbance-rejection magnitude optimum method decay ratios

Satja Lumbar, Damir Vrančić

Fig. 13 Response on input disturbance \((r=0, d=1)\) of process \(G_{P2,5}\)

Fig. 14 Response on input disturbance \((r=0, d=1)\) of process \(G_{P2,6}\)

Fig. 15 Response on input disturbance \((r=0, d=1)\) of process \(G_{P3,1}\)
Fig. 16 Response on input disturbance \((r=0, d=1)\) of process \(G_{P3,2}\)

Fig. 17 Response on input disturbance \((r=0, d=1)\) of process \(G_{P3,3}\)

Fig. 18 Response on input disturbance \((r=0, d=1)\) of process \(G_{P3,4}\)
Fig. 19 Response on input disturbance \((r=0, d=1)\) of process \(G_{P3,5}\)

Fig. 20 Response on input disturbance \((r=0, d=1)\) of process \(G_{P3,6}\)

Fig. 21 Response on input disturbance \((r=0, d=1)\) of process \(G_{P4,1}\)
Fig. 22 Response on input disturbance \((r=0, \, d=1)\) of process \(G_{P4,2}\)

Fig. 23 Response on input disturbance \((r=0, \, d=1)\) of process \(G_{P4,3}\)

Fig. 24 Response on input disturbance \((r=0, \, d=1)\) of process \(G_{P4,4}\)
Fig. 25 Response on input disturbance \((r=0, d=1)\) of process \(G_{P4.5}\)

Fig. 26 Response on input disturbance \((r=0, d=1)\) of process \(G_{P4.6}\)

Fig. 27 Response on input disturbance \((r=0, d=1)\) of process \(G_{P5.1}\)
Fig. 28 Response on input disturbance \((r=0, d=1)\) of process \(G_{P5,2}\)

Fig. 29 Response on input disturbance \((r=0, d=1)\) of process \(G_{P5,3}\)

Fig. 30 Response on input disturbance \((r=0, d=1)\) of process \(G_{P5,4}\)
Fig. 31 Response on input disturbance \((r=0, d=1)\) of process \(G_{P5,5}\)

Fig. 32 Response on input disturbance \((r=0, d=1)\) of process \(G_{P5,6}\)

Fig. 33 Response on input disturbance \((r=0, d=1)\) of process \(G_{P6,1}\)
Fig. 34 Response on input disturbance \((r=0, d=1)\) of process \(G_{P6,2}\)

Fig. 35 Response on input disturbance \((r=0, d=1)\) of process \(G_{P6,3}\)

Fig. 36 Response on input disturbance \((r=0, d=1)\) of process \(G_{P6,4}\)
Fig. 37 Response on input disturbance \((r=0, d=1)\) of process \(G_{P6,5}\)

Fig. 38 Response on input disturbance \((r=0, d=1)\) of process \(G_{P6,6}\)

Fig. 39 Response on input disturbance \((r=0, d=1)\) of process \(G_{P7,1}\)
Fig. 40 Response on input disturbance \((r=0, d=1)\) of process \(G_{P7,2}\)

Fig. 41 Response on input disturbance \((r=0, d=1)\) of process \(G_{P7,3}\)

Fig. 42 Response on input disturbance \((r=0, d=1)\) of process \(G_{P7,4}\)
Fig. 43 Response on input disturbance \((r=0, d=1)\) of process \(G_{P7,5}\)

![Graph showing response of \(G_{P7,5}\) with PI controller to input disturbance](image)

Fig. 44 Response on input disturbance \((r=0, d=1)\) of process \(G_{P7,6}\)

![Graph showing response of \(G_{P7,6}\) with PI controller to input disturbance](image)

Fig. 45 Response on input disturbance \((r=0, d=1)\) of process \(G_{P8,1}\)

![Graph showing response of \(G_{P8,1}\) with PI controller to input disturbance](image)
Fig. 46 Response on input disturbance \((r=0, d=1)\) of process \(G_{P8,2}\)

Fig. 47 Response on input disturbance \((r=0, d=1)\) of process \(G_{P8,3}\)

Fig. 48 Response on input disturbance \((r=0, d=1)\) of process \(G_{P8,4}\)
Fig. 49 Response on input disturbance \((r=0, \; d=1)\) of process \(G_{P8.5}\)

Fig. 50 Response on input disturbance \((r=0, \; d=1)\) of process \(G_{P8.6}\)

Fig. 51 Response on input disturbance \((r=0, \; d=1)\) of process \(G_{P9.1}\)
Fig. 52 Response on input disturbance \((r=0, d=1)\) of process \(G_{P9,2}\)

Fig. 53 Response on input disturbance \((r=0, d=1)\) of process \(G_{P9,3}\)

Fig. 54 Response on input disturbance \((r=0, d=1)\) of process \(G_{P9,4}\)
Study on disturbance-rejection magnitude optimum method decay ratios

Satja Lumbar, Damir Vrančić

Fig. 55 Response on input disturbance \((r=0, d=1)\) of process \(G_{P9.5}\)

Fig. 56 Response on input disturbance \((r=0, d=1)\) of process \(G_{P9.6}\)

Fig. 57 Decay ratios for closed-loop systems with PI controller and processes \(G_{P1}\) to \(G_{P9}\)

In this figure we can observe the relative uniformity of the decay ratios for almost all processes; these decays are all within an approximately 7% range. The only exception is the non-minimal phase process \((G_{P7})\), for which we get noticeably lower decay ratios, especially for processes with
larger non-minimal phase. Processes with long delays \((G_{P1} \text{ and } G_{P2})\) also tend to have slightly smaller decay ratios, then the rest of the processes from the chosen batch, but they are still within a relatively small range. All other processes have decay ratios within a 4% range. This is confirmed by a histogram representation in Fig. 58.

![Histogram of decay ratios for selected batch of process models](Fig. 58 A histogram of the decay ratios for closed-loop systems with PI controller and processes \(G_{P1} \text{ to } G_{P9}\)

To sum up, the DRMO tuning method gives relatively uniform responses (responses with consistent decay ratios over the whole set) on input disturbance for most of the tested processes. PI controller only has trouble with zeros in right half-plane s. In light of these new results let us compare the DRMO method with some other relevant methods.

### 4. Comparison to some other methods

In this chapter we compare the decay ratios of the DRMO method with the decay ratios of two other tuning methods: Åstrom and Hagglund (Kappa-Tau or KT) tuning method [1] and the tuning according to Panagopoulos [3,17]. Let us first give a description of these methods:

#### 4.1 Åstrom and Hagglund

This method basically leans on the original Ziegler-Nichols rules. A substantial improvement is achieved if we characterize the process with three (instead of two) parameters. Maximum sensitivity

\[
M_s = \max_{\omega} \left| \frac{1}{1 + G_p(i\omega)G_c(i\omega)} \right|
\]

(28)

is used as a tuning parameter.

If the process is stable, its dynamics are characterized by three parameters: the static gain \(K_p\), the apparent lag \(T\), and apparent dead time \(L\) (Fig. 59).
The PI controller has three parameters, the gain $K_P$, the integration time $T_i$, and the setpoint weighting $b$. It is convenient to represent these parameters in dimension-free form by suitable normalization. The normalized controller gain is $K_P^*$, and the normalized integration time $T_i^*$. This normalization is the same normalization used in the Ziegler-Nichols rules. In some cases, the integration time will be normalized by $T$ instead of $L$.

The following relation between the normalized controller parameters and the normalized process parameters has been suggested [1]:

$$T_L = T + L$$

$$a = K_P^*$$

$$b = b$$

Table 1 gives the coefficients $a_0$, $a_1$, and $a_2$ of the functions of the form (30).

Table 1: Tuning formula for PI control obtained by the step-response method. The table gives parameters of functions of the form $f(\tau) = a_0e^{a_1\tau} + a_2\tau$ for the normalized controller parameters $K_P^*$, the integration time $T_i^*$, and the setpoint weighting $b$. The table was obtained by the step-response method and gives parameters of functions of the form $f(\tau) = a_0e^{a_1\tau} + a_2\tau$ for the normalized controller parameters $K_P^*$, the integration time $T_i^*$, and the setpoint weighting $b$. The table was obtained by the step-response method.

To present the results, it is convenient to reparameterize the process. Guided by the Ziegler-Nichols formula [1], we use the parameter $T_L$. The process dynamics are characterized by the parameters $a$, $L$, and $\tau$ which were also used for PI tuning rules for PID control are developed in the same way as the rules for PI control. The process is characterized by the parameters $a$, $L$, and $\tau$ which were also used for PI tuning rules for PID control.
control. The controller parameters are normalized as $aK_P$, $T_i/L$, $T_d/L$. Table 2 gives coefficients $a_0$, $a_1$, and $a_2$ of the functions of the form (30).

**Table 2** Tuning formula for PID control obtained by the step-response method. The table gives parameters of functions of the form $f(\tau) = a_0 e^{a_1 \tau + a_2 \tau^2}$ for the normalized controller parameters

<table>
<thead>
<tr>
<th>$M_c=1.4$</th>
<th>$M_c=2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_0$</td>
<td>$a_1$</td>
</tr>
<tr>
<td>$aK$</td>
<td>3.8</td>
</tr>
<tr>
<td>$T_i/L$</td>
<td>5.2</td>
</tr>
<tr>
<td>$T_d/L$</td>
<td>0.46</td>
</tr>
<tr>
<td>$T_d/T$</td>
<td>0.89</td>
</tr>
<tr>
<td>$T_d/T$</td>
<td>0.077</td>
</tr>
<tr>
<td>$b$</td>
<td>0.40</td>
</tr>
</tbody>
</table>

### 4.2 Panagopoulos tuning rules

This method [3] is based on non-convex optimization. The parameters are being adjusted until a certain value of sensitivity $M_c$ has been achieved. There are additional constraints on robustness to model uncertainties when dealing with PID controller [17].

### 4.3 Results

Previously described sets of rules for PI control [1,3,15] have been compared on the following processes:

$$G_{p1} = \frac{1}{(s+1)^3}$$ \hspace{1cm} (32)

$$G_{p2} = \frac{1}{(s+1)(1+0.2s)(1+0.04s)(1+0.008s)}$$ \hspace{1cm} (33)

$$G_{p3} = \frac{e^{-15s}}{(s+1)^3}$$ \hspace{1cm} (34)

$$G_{p4} = \frac{1-2s}{(s+1)^3}$$ \hspace{1cm} (35)

$$G_{p5} = e^{-s}$$ \hspace{1cm} (36)

The PI controller parameters for all three methods are given in table 3.

Fig. 60 to Fig. 64 show the closed-loop input disturbance responses for processes $G_{p1}$ to $G_{p5}$. In Fig. 65 we can observe the decay ratios for all three tuning methods.
Table 3 Controller parameters for processes \( G_{P1} \) to \( G_{P5} \). First two rows contain DRMO parameters. In next eight rows the superscript indexation denotes the method used. Each index is composed of a letter and a number. Letter A represents tuning rules according to Åstrom and Hagglund and letter P represents tuning rules according to Panagopoulos. The number represents the maximum sensitivity \( M_s \).

<table>
<thead>
<tr>
<th>( G_{P1} )</th>
<th>( G_{P2} )</th>
<th>( G_{P3} )</th>
<th>( G_{P4} )</th>
<th>( G_{P5} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( K_P )</td>
<td>0.651</td>
<td>2.176</td>
<td>0.276</td>
<td>0.328</td>
</tr>
<tr>
<td>( K_i )</td>
<td>0.455</td>
<td>4.041</td>
<td>0.045</td>
<td>0.176</td>
</tr>
<tr>
<td>( K_P^{A14} )</td>
<td>0.535</td>
<td>1.32</td>
<td>0.077</td>
<td>0.141</td>
</tr>
<tr>
<td>( K_i^{A14} )</td>
<td>0.334</td>
<td>2.289</td>
<td>0.018</td>
<td>0.11</td>
</tr>
<tr>
<td>( K_P^{P14} )</td>
<td>0.633</td>
<td>1.93</td>
<td>0.164</td>
<td>0.179</td>
</tr>
<tr>
<td>( K_i^{P14} )</td>
<td>0.325</td>
<td>2.591</td>
<td>0.027</td>
<td>0.101</td>
</tr>
<tr>
<td>( K_P^{A2} )</td>
<td>1.145</td>
<td>3.036</td>
<td>0.280</td>
<td>0.340</td>
</tr>
<tr>
<td>( K_i^{A2} )</td>
<td>0.715</td>
<td>5.266</td>
<td>0.064</td>
<td>0.266</td>
</tr>
<tr>
<td>( K_P^{P2} )</td>
<td>1.22</td>
<td>4.13</td>
<td>0.266</td>
<td>0.294</td>
</tr>
<tr>
<td>( K_i^{P2} )</td>
<td>0.685</td>
<td>6.988</td>
<td>0.048</td>
<td>0.184</td>
</tr>
</tbody>
</table>

Fig. 60 Response on input disturbance \((r=0, d=1)\) of process (32)

Fig. 61 Response on input disturbance \((r=0, d=1)\) of process (33)
Study on disturbance-rejection magnitude optimum method decay ratios

Satja Lumbar, Damir Vrančić

Fig. 62 Response on input disturbance \((r=0, d=1)\) of process (34)

Fig. 63 Response on input disturbance \((r=0, d=1)\) of process (35)

Fig. 64 Response on input disturbance \((r=0, d=1)\) of process (36)
Study on disturbance-rejection magnitude optimum method decay ratios

Satija Lumbar, Damir Vrančić

Fig. 65 Decay ratios of processes (32) to (36) for DRMO tuning method (+), KT tuning method (□) and Panagopoulos tuning method (○). Note that for Panagopoulos tuning and KT tuning only decays of responses with $M_s=2$ are presented in this figure, since responses with $M_s=1.4$ have zero decay for many processes.

The decay ratios of the DRMO tuning method span over a range of approximately 7.5%, while the other two methods have significantly wider fields of values of the decay ratios [%]. We conclude that the responses on input disturbance, when the PI controller parameters are tuned with DRMO method, are a great deal more uniform in terms of decay ratios than responses that we get by tuning the PI controller with any of other two methods.

Let us now observe the maximum of sensitivity function (28) for all processes and all discussed tuning methods in Fig. 66.

Fig. 66 Maximum sensitivity function for processes $G_{P1}$ to $G_{P5}$ for DRMO method (+), Panagopoulos method (* for $M_s=1.4$ and □ for $M_s=2$) and the KT method (○ for $M_s=1.4$ and ◊ for $M_s=2$). From the above figure it becomes clear that Panagopoulos method guarantees absolutely uniform maximum sensitivity over the tested batch of process models, while the maximum sensitivity of the KT method varies significantly, which is logical, since there is no optimization involved in this method. Judging by the results in this chapter, we can speculate, that the DRMO method is a better criterion for uniformity of responses then optimizing the maximum sensitivity to a certain value.
5. Conclusions

Disturbance rejection magnitude optimum method gives good responses on input disturbance \( (r=0, d=1) \) for many processes that are characteristic to process and chemical industries. The aim of this paper was to determine whether these responses are uniform in terms of the decay ratios. This was established on a wide batch of processes in Chapter 3, where it was shown, that decay ratios are within a range of approximately 4% when using a PI controller. Problems arise when controlling processes with zeros. With PI controller we obtain decay ratios, that significantly deviate from the afore mentioned decay ratio field when controlling processes with zeros in right half-plane.

In this aspect, when comparing the DRMO method with other methods it becomes clear, that DRMO gives responses that are more uniform then those obtained with other tuning methods. We can draw a conclusion that by using the DRMO tuning method, the responses on the input disturbance are quite predictable unless the process is approximated with a non-minimal phase model.

6. References


APPENDIX A

List of files:
reg_comparePI_final.m
gp_base2.m
tc2areas1.m
PIopt.m
bo_pi.m
controller.mdl (same as pid_proc_CL.mdl)
ex_find.m
reg_comparePI_APfinal.m
gp_basePI.m
OL-par1.m
PI_Panagopoulos.m
PI_Astrom.m

*reg_comparePI_final.m*

% For a selected batch of processes (gp_base2.m) simulates the responses on % an input disturbance and plots them. It also calculates and plots the % decay ratios.

close all;
clear;

PARPI=[]; %for storing controller parameters
ABratios=[]; %for storing decay ratios
refval=0; %reference signal
disval=abs(refval-1); %disturbance signal
imag1=sqrt(-1); %define imaginary number
kl=4;                      %at which peak do we start when measuring decay ratio
Kmax=10000;                %maximum proportional controller gain
Kpr=1;                     %steady state gain of the process
Tstop=300;                 %simulation time

for i=1:9
    a=i;
    for j=1:6
        b=j;

        clear num den Tdelay

        pogoj=1;

        [num,den,Tdelay] = gp_base2 (a,b); %define the process
        [A0,A1,A2,A3,A4,A5,A6,A7] = tc2areas1 (num,den,Tdelay,Kpr); %calculate characteristic areas of the process
den2=den;                               %stores the denominator of the process for later use
        [Ki2,K2] = PIopt (A0,A1,A2,A3,Kmax);    %calculate DRMO controller parameters
        numC=[K2 Ki2];                          %nominator of the controller transfer function
denC=[1 0];                             %denominator of the controller transfer function

        [ampP,fazaP,w2]=bo_pi(num,den,Tdelay);  %transform the process into frequency domain
        [ampC,fazaC,w2]=bo_pi(numC,denC,0);     %transform the controller into frequency domain

        AA = -imag1.*(Ki2./w2').*ampP.*(cos(fazaP) + imag1.*sin(fazaP));
        BB = 1 + ampP.*ampC.*(cos(fazaP+fazaC) + imag1.*sin(fazaP+fazaC));
        S=1./BB;                                %the sensitivity function
        U=AA./BB;                               %modified sensitivity function

        %%%%%plot and save the sensitivity and modified sensitivity function of the cl system
        figure;
        semilogx(w2,abs(U));hold on;
        semilogx(w2,abs(S),'--');
        cd rezultatiPI
        saveas(gcf,['PI',num2str(a),'',num2str(b),'sens'],'fig')
        print('-deps',['PI',num2str(a),'',num2str(b),'sens'])
        close
        cd ..

        %%%%%plot and save the Nyquist diagram of the system
        figure;
        axis([-2 2 -2 2]); hold on;
        plot(BB-1);
        grid on;
        cd rezultatiPI


saveas(gcf,['PID',num2str(a),",num2str(b),nyquist'],'fig')
print("-deps",['PID',num2str(a),",num2str(b),nyquist'])
close
cd ..
%%%%%%simulation
if Tdelay==0; %if delay=0 we don't have to use simulink for simulation (we save
simulation time)
    deltat=0.002;
    Ttt=0:deltat:Tstop;
    numC=[K2 Ki2];
    denC=[1 0];
    Gp=tf(num,den);
    Gc=tf(numC,denC);
    G=Gp/(1+Gp*Gc);
    [y4,t4]=step(G,Ttt);
else %in case of non-zero delay we use simulink for simulation (consumes
more time)
    numC=[K2 Ki2];
    denC=[1 0];
    Gp=tf(num,den);
    Gc=tf(numC,denC);
    Gpc=Gc*Gp;
    [t,y]=sim('controller',Tstop,[],[]);
    t4=simout(:,1);
    y4=simout(:,3);
end
%%%%%%plot and save the closed-loop response
pls=3;
figure;
plot(t4(1:length(t4)/pls),y4(1:length(t4)/pls)); hold on; grid on;
title(['Process G_P_\ ,num2str(a),\_num2str(b), with PI controller']);
xlabel(['t[s]']);

cd rezultatiPI
save(['PI',num2str(a),",num2str(b)'],'t4',"y4")
saveas(gcf,['PI',num2str(a),",num2str(b)'],'fig')
print("-depsc",['PI',num2str(a),",num2str(b),color'])
close
cd ..

%%%%%% calculation of decay ratio
extrs=ex_find(t4,y4); %finds the peaks of closed loop response
AA1=abs(extrs(1,kl))+abs(extrs(1,kl+1)); %P4+P5
BB1=abs(extrs(1,kl+1))+abs(extrs(1,kl+2)); %P5+P6
razm=BB1/AA1; %decay ratio
saving the parameters
ABratios=[ABratios; razm];
PARPI=[PARPI; [K2; Ki2]];
\texttt{cd \texttt{rezultatiPI}}
save PARPI PARPI -TABS
save ABratios ABratios -TABS
\texttt{cd \ldots}

\texttt{gp\_base2.m}

\texttt{\%Batch of processes}
\begin{verbatim}
function [num,den,Tdelay] = gp_base2 (a,b);
Tdelay=0;
if (a == 1),
    num = [0 0 0 0 0 0 0 1];
    if (b == 1),
        Tdelay=12;
        elseif (b == 2),
            Tdelay=11;
            elseif (b == 3),
                Tdelay=10;
                elseif (b == 4),
                    Tdelay=6;
                    elseif (b == 5),
                        Tdelay=2;
                        elseif (b == 6),
                            Tdelay=1;
                            end;
    T=12-Tdelay;
    den=[0 0 0 0 0 0 T 1];
    elseif (a == 2),
\end{verbatim}
num = [0 0 0 0 0 0 0 1];
if (b == 1),
    Tdelay=11;
elseif (b == 2),
    Tdelay=10;
elseif (b == 3),
    Tdelay=8;
elseif (b == 4),
    Tdelay=6;
elseif (b == 5),
    Tdelay=4;
elseif (b == 6),
    Tdelay=1;
end;
T=(12-Tdelay)/2;
den=[0 0 0 0 0 T*T 2*T 1];

elseif (a == 3),
    num = [0 0 0 0 0 0 0 1];
    if (b == 1),
        T1=11;
    elseif (b == 2),
        T1=10;
    elseif (b == 3),
        T1=9;
    elseif (b == 4),
        T1=8;
    elseif (b == 5),
        T1=7;
    elseif (b == 6),
        T1=6;
    end;
    T2=12-T1;
den=[0 0 0 0 0 T1*T2 T1+T2 1];

elseif (a == 4),
    num = [0 0 0 0 0 0 0 1];
    if (b == 1),
        T1=5.5;
    elseif (b == 2),
        T1=5;
    elseif (b == 3),
        T1=4.5;
    elseif (b == 4),
        T1=4;
    elseif (b == 5),
        T1=3.5;
    elseif (b == 6),
        T1=3;
    end;
    T2=(12-2*T1)/2;
\[
\text{den} = [0 \ 0 \ 0 \ 0 \ T_1*T_1*T_2*T_2 \ (2*T_1*T_1*T_2+2*T_1*T_2*T_2) \ (T_1*T_1+4*T_1*T_2+T_2*T_2) \\
(2*T_2+2*T_1) \ 1];
\]

```matlab
elseif (a == 5),
    num = [0 0 0 0 0 0 0 0 1];
    if (b == 1),
        T=4;
        int=conv([T 1],[T 1]);
        den=conv(int,[T 1]);
    elseif (b == 2),
        T=3;
        int=conv([T 1],[T 1]);
        int=conv(int,[T 1]);
        den=conv(int,[T 1]);
    elseif (b == 3),
        T=2.4;
        int=conv([T 1],[T 1]);
        int=conv(int,[T 1]);
        int=conv(int,[T 1]);
        int=conv(int,[T 1]);
        den=conv(int,[T 1]);
    elseif (b == 4),
        T=2;
        int=conv([T 1],[T 1]);
        int=conv(int,[T 1]);
        int=conv(int,[T 1]);
        int=conv(int,[T 1]);
        int=conv(int,[T 1]);
        den=conv(int,[T 1]);
    elseif (b == 5),
        T=12/7;
        int=conv([T 1],[T 1]);
        int=conv(int,[T 1]);
        int=conv(int,[T 1]);
        int=conv(int,[T 1]);
        int=conv(int,[T 1]);
        int=conv(int,[T 1]);
        den=conv(int,[T 1]);
    elseif (b == 6),
        T=1.5;
        int=conv([T 1],[T 1]);
        int=conv(int,[T 1]);
        int=conv(int,[T 1]);
        int=conv(int,[T 1]);
        int=conv(int,[T 1]);
        int=conv(int,[T 1]);
        int=conv(int,[T 1]);
        den=conv(int,[T 1]);
    end;

elseif (a == 6),
    num = [0 0 0 0 0 0 0 0 1];
    if (b == 1),
        k=0.9;
    elseif (b == 2),
```
\[ k=0.7; \]
\[ \text{elseif}(b == 3), \]
\[ k=0.5; \]
\[ \text{elseif}(b == 4), \]
\[ k=0.4; \]
\[ \text{elseif}(b == 5), \]
\[ k=0.3; \]
\[ \text{elseif}(b == 6), \]
\[ k=0.2; \]
\[ \text{end}; \]
\[ T=11/(k+k^2+k^3); \]
\[ \text{den}=[0 \ 0 \ 0 \ (k^6)\cdot(T^3) \ (k^6\cdot T^3+k^5\cdot T^2+k^4\cdot T^2+k^3\cdot T^2) \ (k^5\cdot T^2+k^4\cdot T^2+k^3\cdot T^2+k^3\cdot T+k^2\cdot T+k\cdot T) \ (k^3\cdot T+k^2\cdot T+k\cdot T+1) \ 1]; \]

\[ \text{elseif}(a == 7), \]
\[ \text{if}(b == 1), \]
\[ T1=1; \]
\[ \text{elseif}(b == 2), \]
\[ T1=2; \]
\[ \text{elseif}(b == 3), \]
\[ T1=3; \]
\[ \text{elseif}(b == 4), \]
\[ T1=4; \]
\[ \text{elseif}(b == 5), \]
\[ T1=5; \]
\[ \text{elseif}(b == 6), \]
\[ T1=6; \]
\[ \text{end}; \]
\[ T2=(12-T1)/3; \]
\[ \text{den}=[0 \ 0 \ 0 \ 0 \ T2^3 \ 3\cdot T2^2 \ 3\cdot T2 \ 1]; \]
\[ \text{num}=[0 \ 0 \ 0 \ 0 \ 0 \ -T1 \ 1]; \]

\[ \text{elseif}(a == 8), \]
\[ \text{if}(b == 1), \]
\[ Tz=0.5; \]
\[ \text{elseif}(b == 2), \]
\[ Tz=1; \]
\[ \text{elseif}(b == 3), \]
\[ Tz=2.5; \]
\[ \text{elseif}(b == 4), \]
\[ Tz=4.5; \]
\[ \text{elseif}(b == 5), \]
\[ Tz=5.5; \]
\[ \text{elseif}(b == 6), \]
\[ Tz=7; \]
\[ \text{end}; \]
\[ Tp=(12+Tz)/3; \]
\[ \text{num}=[0 \ 0 \ 0 \ 0 \ 0 \ Tz \ 1]; \]
\[ \text{den}=[0 \ 0 \ 0 \ 0 \ Tp^3 \ 3\cdot Tp^2 \ 3\cdot Tp \ 1]; \]
elseif (a == 9),
num = [0 0 0 0 0 0 0 0 1];
T2=4;
if (b == 1),
alfa=0.2;
elseif (b == 2),
alfa=0.3;
elseif (b == 3),
alfa=0.4;
elseif (b == 4),
alfa=0.5;
elseif (b == 5),
alfa=0.7;
elseif (b == 6),
alfa=1;
end;
den=[0 0 0 0 0 T2*(1+alfa^2) T2^2*(3+alfa^2) 3*T2 1];
end;

tc2areas1.m

% function [A0,A1,A2,A3,A4,A5,A6,A7] = tc2areas (Tpv,Tzv,Tdelay,Kpr);
% % Function zp2areas calculates characteristic areas (A0 to A7) from the process
% % time constants:
% % % Gpr = Kpr*(1+s*Tzv(1))*(1+s*Tzv(2))*...*exp(-Tdelay*s)/((1+s*Tpv(1))*(1+s*Tpv(2))*)...

function [A0,A1,A2,A3,A4,A5,A6,A7] = tc2areas1 (num,den,Tdelay,Kpr);

num = real(num); % Just in case if complex numerator or denominator are calculated
den = real(den);

Len_num=length(num);
Len_den=length(den);

if (Len_den<9),
den=[zeros(size(1:(9-Len_den))) den];
Len_den = 9;
end

if (Len_num<Len_den),
um=[zeros(size(1:(Len_den-Len_num))) num];
end

Len_num=length(num);
Len_den=length(den);
a1 = den(Len_den-1);
\[ a_2 = \text{den}(\text{Len}_\text{den} - 2); \]
\[ a_3 = \text{den}(\text{Len}_\text{den} - 3); \]
\[ a_4 = \text{den}(\text{Len}_\text{den} - 4); \]
\[ a_5 = \text{den}(\text{Len}_\text{den} - 5); \]
\[ a_6 = \text{den}(\text{Len}_\text{den} - 6); \]
\[ a_7 = \text{den}(\text{Len}_\text{den} - 7); \]
\[ a_8 = \text{den}(\text{Len}_\text{den} - 8); \]

\[ b_1 = \text{num}(\text{Len}_\text{num} - 1); \]
\[ b_2 = \text{num}(\text{Len}_\text{num} - 2); \]
\[ b_3 = \text{num}(\text{Len}_\text{num} - 3); \]
\[ b_4 = \text{num}(\text{Len}_\text{num} - 4); \]
\[ b_5 = \text{num}(\text{Len}_\text{num} - 5); \]
\[ b_6 = \text{num}(\text{Len}_\text{num} - 6); \]
\[ b_7 = \text{num}(\text{Len}_\text{num} - 7); \]
\[ b_8 = \text{num}(\text{Len}_\text{num} - 8); \]

\[ Td_2 = \text{Tdelay}^2; \]
\[ Td_3 = Td_2^2; \]
\[ Td_4 = Td_3^2; \]
\[ Td_5 = Td_4^2; \]
\[ Td_6 = Td_5^2; \]
\[ Td_7 = Td_6^2; \]

\[ A_0 = \text{Kpr}; \]
\[ A_1 = A_0*(a_1-b_1+\text{Tdelay}); \]
\[ A_2 = A_0*(b_2-a_2-\text{Tdelay})*b_1+Td_2/2)+A_1*a_1; \]
\[ A_3 = A_0*(a_3-b_3+\text{Tdelay})*b_2-Td_2*b_1/2+Td_3/6)+A_2*a_1-A_1*a_2; \]
\[ A_4 = A_0*(b_4-a_4-\text{Tdelay})*b_3+Td_2*b_2/2-Td_3*b_1/6+Td_4/24)+A_3*a_1-A_2*a_2+A_1*a_3; \]
\[ A_5 = A_0*(a_5-b_5+\text{Tdelay})*b_4-Td_2*b_3/2+Td_3*b_2/6-Td_4*b_1/24+Td_5/120)+A_4*a_1-\]
\[ A_3*a_2+A_2*a_3-A_1*a_4; \]
\[ A_6 = A_0*(b_6-a_6-\text{Tdelay})*b_5+Td_2*b_4/2-Td_3*b_3/6+Td_4*b_2/24-Td_5*b_1/120+Td_6/720)+A_5*a_1-\]
\[ A_4*a_2+A_3*a_3-A_2*a_4+A_1*a_5; \]
\[ A_7 = A_0*(a_7-b_7+\text{Tdelay})*b_6-Td_2*b_5/2+Td_3*b_4/6-Td_4*b_3/24+Td_5*b_2/120-\]
\[ Td_6*b_1/720+Td_7/5040)+A_6*a_1-A_5*a_2+A_4*a_3-A_3*a_4+A_2*a_5-A_1*a_6; \]

\[ Plopt.m \]

% function [Ki,K] = Plopt (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;

do_pi.m

% The process transfer function is first represented by nominator, 
% denominator and time delay. This function provides equal presentation 
% with amplitude and phase

function [amp,faza,w2]=bo_pi(num,den,Tdelay);
imag1 = sqrt(-1);
Points = 2500;
w2 = logspace (-3,3,Points);

[amp,faza] = bode(num,den,w2);
faza=faza*pi/180-w2*Tdelay;

**controller.mdl**

```
%finds the peaks of closed loop response
function extras=ex_find(t4,y4);

zeros=[];
extras=[];
sig1=2;
counter=1;

for i=25:length(y4)
    sig2=sign(y4(i));
    if sig2==1
        if sig1==1
            zeros=[zeros, [i; t4(i); 2; counter]]; counter=counter+1;
        elseif sig1==-1
            zeros=[zeros, [i; t4(i); 1; counter]]; counter=counter+1;
        elseif sig1==0
            end
        elseif sig2==0
            elseif sig2==-1
                if sig1==1
                    zeros=[zeros, [i; t4(i); 1; counter]]; counter=counter+1;
                elseif sig1==-1
                    elseif sig1==0
                        end
                    end
                sig1=sign(y4(i));
```
end

index1=1;
counter1=1;

for ii=zeros(4,:)
    index2=zeros(1,ii)-1;
    vekt=y4(index1:index2);
    index1=zeros(1,ii);
    if zeros(3,ii)==1
        extr=max(vekt);
        extrs=[extrs, [extr;counter1]];
        counter1=counter1+1;
    elseif zeros(3,ii)==2
        extr=min(vekt);
        extrs=[extrs, [extr;counter1]];
        counter1=counter1+1;
    end

end

reg_comparePI_APfinal.m

% For a selected batch of processes (gp_base2.m) simulates the responses on
% an input disturbance (DRMO, Astrom, Panagopoulos controller parameters) and plots
% them.
% It also calculates and plots decay ratios for all three tuning methods.

close all;
clear;

PARPI=[];               %for storing controller parameters
ABratios=[];            %for storing DRMO decay ratios
ABratiosP2=[];          %for storing Panagopoulos decay ratios
ABratiosA2=[];          %for storing Astrom decay ratios

refval=0;               %reference signal
disval=abs(refval-1);   %disturbance signal
imag1=sqrt(-1);         %define imaginary number
kl=4;                   %at which peak do we start when measuring decay ratio
Kmax=10000;             %maximum proportional controller gain
Kpr=1;                  %steady state gain of the process
maxU=1.01;              %simulation time
Tstop=300;

for i=1:9
    a=i;
    clear num den Telay
pogoj=1;

[num,den,Tdelay] = gp_basePI (a);    %define the process
[A0,A1,A2,A3,A4,A5,A6,A7] = tc2areas1 (num,den,Tdelay,Kpr);  %calculate characteristic areas of the process
den2=den;     %stores the denominator of the process for later use
[Ki2,K2] = Dopt (A0,A1,A2,A3,Kmax);  %calculate DRMO controller parameters

Ms=1.4;       %setting the value of maximum sensitivity

[KK,LL,TT,aa] = OL_par1 (num, den, Tdelay);   %calculates the required open loop parameters

[KiP,P] = PI_Panagopoulos(a,Ms);  %calculate the controller parameters for Panagopoulos method

[KA, KiA, bb] = PI_Astrom (KK,TT,LL,aa,Ms); %calculate the controller parameters for Astrom method

numC=[K2 Ki2];  %nominator of the controller transfer dunction
denC=[1 0];     %denominator of the controller transfer dunction

[ampP,fazaP,w2]=bo_pi(num,den,Tdelay);  %transform the process into frequency domain
[ampC,fazaC,w2]=bo_pi(numC,denC,0);   %transform the controller into frequency domain

AA = -imag.*(Ki2./w2').*ampP.*(cos(fazaP) + imag1.*sin(fazaP));
BB = 1 + ampP.*ampC.*(cos(fazaP+fazaC) + imag1.*sin(fazaP+fazaC));
U=AA./BB;  %modified sensitivity function
S=1./BB;  %the sensitivity function

%%%%%%simulation DRMO
if Tdelay==0;  %if delay=0 we don't have to use simulink for simulation (we save simulation time)
deltat=0.002;
Ttt=0:deltat:Tstop;
numC=[K2 Ki2];
denC=[1 0];
Gp=tf(num,den);
Gc=tf(numC,denC);
G=Gp/(1+Gp*Gc);
[y4,t4]=step(G,Ttt);
else  %in case of non-zero delay we use simulink for simulation (consumes more time)
numC=[K2 Ki2];
denC=[1 0];
Gp=tf(num,den);
Gc=tf(numC,denC);
Gpc=Gc*Gp;
[t,y]=sim('pid_proc_CL',Tstop,[],[]);
t4=simout(:,1);
y4=simout(:,3);
%% calculation of DRMO decay ratio
extrs=ex_find(t4,y4);

AA1=abs(extrs(1,kl))+abs(extrs(1,kl+1));
BB1=abs(extrs(1,kl+1))+abs(extrs(1,kl+2));
razm=BB1/AA1

ABratios=[ABratios; razm];

%%% simulation Astrom Ms=1.4
if Tdelay==0;
    deltat=0.002;
    Ttt=0:deltat:Tstop;
    numC_A=[KA KiA];
    denC_A=[1 0];
    Gp_A=tf(num,den);
    Gc_A=tf(numC_A,denC_A);
    G_A=Gp_A/(1+Gp_A*Gc_A);
    [yA14,tA14]=step(G_A,Ttt);
else
    numC_A=[KA KiA];
    denC_A=[1 0];
    Gp=tf(num,den);
    Gc=tf(numC_A,denC_A);
    Gpc=Gp*Gc;
    [t,y]=sim('pid_proc_CL',Tstop,[],[]);
    tA14=simout(:,1);
    yA14=simout(:,3);
end

%%% simulation Panagopoulos Ms=1.4
if Tdelay==0;
    deltat=0.002;
    Ttt=0:deltat:Tstop;
    numC_P=[KP KiP];
    denC_P=[1 0];
    Gp_P=tf(num,den);
    Gc_P=tf(numC_P,denC_P);
    G_P=Gp_P/(1+Gp_P*Gc_P);
    [yP14,tP14]=step(G_P,Ttt);
else
    numC_P=[KP KiP];
    denC_P=[1 0];
    Gp=tf(num,den);
    Gc=tf(numC_P,denC_P);
    Gpc=Gp*Gc;
    [t,y]=sim('pid_proc_CL',Tstop,[],[]);
Study on disturbance-rejection magnitude optimum method decay ratios

Satja Lumbar, Damir Vrančić

46

tP14=simout(:,1);
yP14=simout(:,3);
end

Ms=2; %setting the value of maximum sensitivity
[KA, KiA, bb] = PI_Astrom (KK, TT, LL, aa, Ms); %calculate the controller parameters for Astrom method
[KiP, KP] = PI_Panagopoulos (a, Ms); %calculate the controller parameters for Panagopoulos method

%%%%%%simulation Astrom Ms=2
if Tdelay==0;
deltat=0.002;
Ttt=0:deltat:Tstop;
numC_A=[KA KiA];
denC_A=[1 0];
Gp_A=tf(num,den);
Gc_A=tf(numC_A,denC_A);
G_A=Gp_A/(1+Gp_A*Gc_A);
yA2,tA2=step(G_A,Ttt);
else
    numC_A=[KA KiA];
    denC_A=[1 0];
    Gp=tf(num,den);
    Gc=tf(numC_A,denC_A);
    Gpc=Gp*Gc;
    [t,y]=sim('pid_proc_CL', Tstop, [], []);
    tA2=simout(:,1);
yA2=simout(:,3);
end

%%%%%%simulation Panagopoulos Ms=2
if Tdelay==0;
deltat=0.002;
Ttt=0:deltat:Tstop;
numC_P=[KP KiP];
denC_P=[1 0];
Gp_P=tf(num,den);
Gc_P=tf(numC_P,denC_P);
G_P=Gp_P/(1+Gp_P*Gc_P);
yP2,tP2=step(G_P,Ttt);
else
    numC_P=[KP KiP];
    denC_P=[1 0];
    Gp=tf(num,den);
    Gc=tf(numC_P,denC_P);
    Gpc=Gp*Gc;
    [t,y]=sim('pid_proc_CL', Tstop, [], []);
    tP2=simout(:,1);
yP2=simout(:,3);
end
%% calculation of Astrom Ms=2 decay ratio
if a==5
    extrsA2 = ex_find(tA2, yA2);
    AAA2 = abs(extrsA2(1, kl) + abs(extrsA2(1, kl + 1)));
    BBA2 = abs(extrsA2(1, kl + 1) + abs(extrsA2(1, kl + 2)));
    razmA2 = BBA2 / AAA2
    ABratiosA2 = [ABratiosA2; razmA2];
elseif a==5
    ABratiosA2 = [ABratiosA2; 0];
end
%
%% calculation of Panagopoulos Ms=2 decay ratio
extrsP2 = ex_find(tP2, yP2);
AAP2 = abs(extrsP2(1, kl) + abs(extrsP2(1, kl + 1)));
BBP2 = abs(extrsP2(1, kl + 1) + abs(extrsP2(1, kl + 2)));
razmP2 = BBP2 / AAP2
ABratiosP2 = [ABratiosP2; razmP2];
%
%adjustment of time scales
if a==1;
    pls=6;
elseif a==2;
    pls=30;
elseif a==3;
    pls=3/2;
elseif a==4
    pls=3;
elseif a==5
    pls=10;
end
%
%plotting and saving the responses
figure;
plot(t4(1:length(t4)/pls), y4(1:length(t4)/pls)); hold on;
plot(tA14(1:length(tA14)/pls), yA14(1:length(tA14)/pls), 'g-');
plot(tA2(1:length(tA2)/pls), yA2(1:length(tA2)/pls), 'r-');
plot(tP14(1:length(tP14)/pls), yP14(1:length(tP14)/pls), 'c-');
plot(tP2(1:length(tP2)/pls), yP2(1:length(tP2)/pls), 'k:');
legend('DRMO mu=1', 'Astrom Ms=1.4', 'Astrom Ms=2', 'Panagopoulos Ms=1.4', 'Panagopoulos Ms=2', 0);
title(['Process G_P_', num2str(a), ' with PI controller']);
grid on
xlabel('t[s]');

cd rezultatiPI_AP
save(['PI',num2str(a)],'t4','y4','tA14','yA14','tA2','yA2')
saveas(gcf,['PI',num2str(a)],'fig')
print('-deps',['PI',num2str(a)])
print('-depsc',['PI',num2str(a),'color'])
close
cd ..

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%%%%%saving the parameters and decays
PARPI=[PARPI,[K2;Ki2;KA;KiA;KP;KiP]];
cd rezultatiPI_AP
save PARPI PARPI -TABS
save ABratios ABratios -TABS
save ABratiosP2 ABratiosP2 -TABS
save ABratiosP2 ABratiosA2 -TABS
cd ..

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%%%%%plotting the decay ratios
plot(ABratios,+); hold on; grid on;plot(ABratiosA2,'r+'); plot(ABratiosP2,'g+');
legend('DRMO mU=1','Astrom Ms=2','Panagopoulos Ms=2');
plot(ABratios,:);plot(ABratiosA2,'r:'); plot((ABratiosP2,'g:'));
cd rezultatiPI_AP
print('-depsc','decay_ratios')
cd .. 

gp_basePI.m

% batch of processes

function [num,den,Tdelay] = gp_basePI (a);
Tdelay=0;
if (a == 1),
   num = [0 0 0 0 0 0 0 1];
   den = [0 0 0 0 1 3 3 1];
elseif (a == 2),
num = [0 0 0 0 0 0 0 1];
den = [0 0 0 0 0.000064 0.009984 0.25792 1.248 1];

elseif (a == 3),
    num = [0 0 0 0 0 0 0 1];
    den = [0 0 0 0 1 3 3 1];
    Tdelay = 15;

elseif (a == 4),
    num = [0 0 0 0 0 -2 1];
    den = [0 0 0 0 1 3 3 1];

elseif (a == 5),
    num = [0 0 0 0 0 0 0 1];
    den = [0 0 0 0 0 0 0 1];
    Tdelay=1;
end;

OL-par1.m

% calculates the open-loop characteristics of the process
function [KK,LL,TT,aa] = OL_par1 (num, den, Tdelay)

G_OLDp=tf(num,den,'outputdelay',Tdelay);
dt=0.01;
Tttt=0:dt:20;
[yol,tol]=step(G_OLDp,Tttt);
ymax=max(yol);
KK=ymax;

for i=2:length(yol)
    odv(i)=(yol(i)-yol(i-1))/dt;
end
index=1;

for i=2:length(odv)
    if odv(i)>odv(i-1)
        index=i;
    end
end
ktan=odv(index);
ttan=tol(index);
ytan=yol(index);

ntan=ytan-ktan*ttan;

LL=(-ntan/ktan);
TT=((0.63-ntan)/ktan)-LL;

aa=KK*(LL/TT);

**PI_Panagopoulos.m**

%calculates the controller parameters according to Panagopoulos

function [Ki, K] = PI_Panagopoulos (a,Ms);

if a==1
    if Ms==1.4
        K=0.633;
        Ti=1.95;
    elseif Ms==1.6
        K=0.862;
        Ti=1.87;
    elseif Ms==1.8
        K=1.06;
        Ti=1.82;
    elseif Ms==2
        K=1.22;
        Ti=1.78;
    end
elseif a==2
    if Ms==1.4
        K=1.93;
        Ti=0.745;
    elseif Ms==1.6
        K=2.74;
        Ti=0.672;
    elseif Ms==1.8
        K=3.47;
        Ti=0.625;
    elseif Ms==2
        K=4.13;
        Ti=0.591;
    end
elseif a==3
    if Ms==1.4
        K=0.164;
        Ti=6.16;
    elseif Ms==1.6

50
K = 0.208; 
Ti = 5.87;
elseif Ms == 1.8
    K = 0.241;
    Ti = 5.66;
elseif Ms == 2
    K = 0.266;
    Ti = 5.51;
end
elseif a == 4
    if Ms == 1.4
        K = 0.179;
        Ti = 1.78
    elseif Ms == 1.6
        K = 0.228;
        Ti = 1.69;
    elseif Ms == 1.8
        K = 0.265;
        Ti = 1.64;
    elseif Ms == 2
        K = 0.294;
        Ti = 1.60;
    end
elseif a == 5
    if Ms == 1.4
        K = 0.158;
        Ti = 0.158/0.472;
    elseif Ms == 2
        K = 0.255;
        Ti = 0.255/0.854;
    end
end

Ki = K/Ti;

*PI_Astrom.m*

% calculates the controller parameters according to Astrom

function [K, Ki, bb] = PI_Astrom (KK, TT, LL, aa, Ms)

tau = LL/(TT+LL);

K_14 = (0.29 * exp(-2.7 * tau + 3.7 * tau * tau)) / aa;
Ti_14 = (8.9*exp(-6.6*tau+3*tau*tau))*LL;
bb_14 = 0.81*exp(0.73*tau+1.9*tau*tau);
\[ K_{2} = \frac{0.78 \cdot \exp(-4.1 \cdot \tau + 5.7 \cdot \tau^2)}{a}; \]
\[ T_{i_2} = \frac{8.9 \cdot \exp(-6.6 \cdot \tau + 3 \cdot \tau^2)}{LL}; \]
\[ b_{b_2} = 0.44 \cdot \exp(0.78 \cdot \tau - 0.45 \cdot \tau^2); \]

\[
\text{if } \text{Ms}==1.4 \\
\quad K=K_{14}; \\
\quad Ki=K_{14}/T_{i_14}; \\
\quad bb=bb_{14};
\]
\[
\text{elseif } \text{Ms}==2 \\
\quad K=K_{2}; \\
\quad Ki=K_{2}/T_{i_2}; \\
\quad bb=bb_{2};
\]
\[
\text{else} \\
\quad \text{disp(’Parameter Ms must be either 1.4 or 2’)}
\]\[
\text{end}
\]