10.24425/acs.2018.124708

Archives of Control Sciences
Volume 28(LXIV), 2018
No. 3, *pages 379–400*

Flexible function block for industrial applications of active disturbance rejection controller

PAWEL NOWAK, KRZYSZTOF STEBEL, TOMASZ KLOPOT, JACEK CZECZOT,
MICHAL FRATCZAK and PIOTR LASZCZYK

In this paper, the PLC-based (Programmable Logic Controller) industrial implementation in the form of the general-purpose function block for ADRC (Active Disturbance Rejection Controller) is presented. The details of practical aspects are discussed because their reliable implementation is not trivial for higher order ADRC. Additional important novelties discussed in the paper are the impact of the derivative backoff and the method that significantly simplifies tuning of higher order ADRC by avoiding the usual trial and error procedure. The results of the practical validation of the suggested concepts complete the paper and show the potential industrial applicability of ADRC.

Key words: industrial control systems, active disturbance rejection control (ADRC), practical tuning, practical implementation, programmable logic controller (PLC)

1. Introduction

First English introduction to *ADRC* technique was made in [4] as to a novel nonlinear control algorithm with a nonlinear observer. Since then, the concept has been developed in many ways and *ADRC* technique has been established as a new paradigm in control theory [7] that links the modern controller design with simplicity and generality of conventional PID controller. However, practitioners still prefer PID-based control loops in the industrial practice [1, 9] because almost every commercial control equipment provides hardware or software PID function block supporting its fast and reliable implementation. Only a few exceptions of ready-to-use function blocks for different advanced control algorithms available in the commercial PLC (Programmable Logic Controller) market, e.g., Fuzzy Logic Control from SIEMENS®, Omron®, Schnei-

The authors are with Silesian University of Technology, Faculty of Automatic Control, Electronics and Computer Science, Institute of Automatic Control, ul. Akademicka 16, 44-100 Gliwice, Poland.

Corresponding author: J. Czeczot E-mail: jacek.czeczot@polsl.pl

This work was supported by Polish Ministry of Science and Higher Education under grants: BKM-UiUA'18 (P. Nowak and M. Fratczak) and BK-UiUA (K. Stebel, T. Klopot, J. Czeczot and P. Laszczyk). Calculations were done with the use of GeCONiI infrastructure (PO IG 02.03.01-24-099).

Received: 20.06.2018. Revised: 6.09.2018.

der Electric®, Modicon®and Internal Model Control from Allen Bradley®. At the same time, some PLC-based implementations were reported for Model Predictive Control [24–25], Predictive Functional Control [3] and for Balance-Based Adaptive Control [2, 13–15].

For the *ADRC* technique, some practical implementation aspects have been also reported. Herbst [10] discusses the time-domain discretization of the first and the second-order *ADRC* for speeding up real-time implementation and summarizes the rule of thumb for *ADRC* tuning. This concept was developed in [11], where the incremental form of *ADRC* controller is proposed and the problem of bumpless switching for reliable *ADRC* implementation is also suggested. In [22], the bumpless switching and anti-windup action are also discussed and automatic tuning tool based on robust closed-loop shaping is presented. These considerations are limited to the simplest case of first-order *ADRC* algorithm but the results are validated in the practical application to control of regenerative heater. This paper provides a significant extension of these concepts and discusses two additional novelties: the impact of the derivative backoff and the method of simplified *ADRC* tuning based only on the process step response. The unified PLC-based implementation of *ADRC* controller is also presented that encapsulates all case-independent calculations for the 1st, 2nd and 3rd order in the form of the general-purpose function block. This block has the flexibility and generality comparable with the general-purpose PID function blocks and thus, it can be considered as the alternative for practitioners who implement industrial control systems.

2. Theoretical background of ADRC

In this paper, the control of SISO (Single Input Single Output) dynamical system of the relative degree $r \geqslant 1$ is considered. The controlled output Y should be stabilized at the set point Y_{sp} by adjusting the manipulating variable (MV) u and the influence of disturbances should be rejected. In practical cases, the accurate model of the process is unknown and then, *ADRC* paradigm provides the controller design [4, 6, 8] based on the following simplified model $Y^{(r)} = h + b_0 \cdot u$, where h represents the so-called *total disturbance* that lumps all modelling uncertainties and b_0 is the adjustable scaling parameter. This model can be represented in the canonical $(r+1)$-th order extended state-space form (1) where the additional state variable x'_{r+1} represents the dynamics of the generalized *total disturbance h*.

The fundamental idea of *ADRC* methodology is to implement the Extended State Observer (*ESO*) for reconstructing the states $\underline{x}'$ from process input-output measurement data. Based on (1), the appropriate *ESO* has the unified form of the Luenberger observer (2). Its tuning requires adjusting the gains $\mathbf{L}$ from the characteristic polynomial and the rule of thumb is that the dynamics of the ob-

server should be $3 \div 10$ times faster comparing to the desired closed loop dynamics [5, 10].

$$\begin{cases} \underbrace{\frac{d}{dt}\begin{bmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_r \\ x'_{r+1} \end{bmatrix}}_{\underline{x}'} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & 0 & \dots & 0 & 0 \end{bmatrix}}_{\mathbf{A}} \cdot \underbrace{\begin{bmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_r \\ x'_{r+1} \end{bmatrix}}_{\underline{x}'} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ b_0 \\ 0 \end{bmatrix}}_{\underline{b}} u + \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}}_{\underline{e}} \frac{dh}{dt}, \\ Y = x'_1, \end{cases} \tag{1}$$

$$\begin{cases} \dfrac{d\hat{\underline{x}}'}{dt} = \mathbf{A} \cdot \hat{\underline{x}}' + \underline{b} \cdot u + \mathbf{L}\left(Y - \hat{Y}\right), \\ \hat{Y} = \hat{x}'_1. \end{cases} \tag{2}$$

Based on *ESO* (2), after substituting $h \approx \hat{x}'_{r+1}$, assuming constant set-point Y_{sp} and the linear r-th order closed-loop reference model $Y^{(r)} = k_p\left(Y_{sp} - \hat{Y}\right) - \sum_{j=1}^{r-1} k_{dj}\hat{x}'_{j+1}$, the *ADRC* controller is derived as:

$$u = \frac{1}{b_0}\left[k_p\left(Y_{sp} - \hat{Y}\right) - \sum_{j=1}^{r-1} k_{dj}\hat{x}'_{j+1} - \hat{x}'_{r+1}\right], \tag{3}$$

For the systems of the unitary relative degree, (3) has the single tuning parameter k_p. For $r > 1$, additional tuning parameters k_{dj} ($j = 1 \div r-1$) are respectively required.

3. Practical implementation issues

The *ADRC* implementation in the form of the flexible function block encapsulating all indispensable case-independent calculations requires some preliminary assumptions dealing with the e.g. acceptable *ADRC* order, time-domain discretization and tuning method.

3.1. Adjustment of ADRC order

The optimal choice of *ADRC* order n is still the open question [12, 26]. In the majority of cases, it should be adjusted equal to or lower from the relative

process degree r. Thus, to ensure flexibility, it was decided to prepare the *ADRC* general-purpose function block for the adjustable order $n \leqslant 3$. It is justified by the fact that for the majority of lag-dominated industrial processes, their effective substitute relative degree is $r \leqslant 2$ [11, 20]. For dead time-dominated processes, the dead time can be compensated by Smith predictor while the substitute effective relative degree of the residuary lag dynamics is still $r \leqslant 2$. Adjusting $n = 3$ additionally expands the functionality of the *ADRC* function block for the processes of higher-order substitute dynamics.

3.2. Time domain ADRC discretization

ADRC implementation in the form of the general-purpose function block should support the calculations for *ADRC* order $n \leqslant 3$. It was decided to avoid conditional code execution and to encapsulate the calculations for $n = 3$, reconfigurable for $n < 3$. Consequently, *ESO* (2) computation is derived using Euler method and zero-order holder, with integration step equal to the control sampling time T_s [sec]:

$$\begin{aligned}
\hat{x}'_{4,i} &= \hat{x}'_{4,i-1} + \underbrace{T_s \cdot l_4 \cdot \left(Y_i - \hat{Y}_{i-1}\right)}_{\Delta\hat{x}'_{4,i}}, \\
aux_i &= \hat{x}'_{4,i} + b_0 \cdot u_{i-1}, \\
\hat{x}'_{3,i} &= \hat{x}'_{3,i-1} + \underbrace{T_s \cdot \left(l_3 \cdot \left(Y_i - \hat{Y}_{i-1}\right) + k3 \cdot aux_i\right)}_{\Delta\hat{x}'_{3,i}}, \\
\hat{x}'_{2,i} &= \hat{x}'_{2,i-1} + \underbrace{T_s \cdot \left(p2 \cdot \hat{x}'_{3,i} + l_2 \cdot \left(Y_i - \hat{Y}_{i-1}\right) + k2 \cdot aux_i\right)}_{\Delta\hat{x}'_{2,i}}, \\
\hat{Y}_i &= \hat{Y}_{i-1} + \underbrace{T_s \cdot \left(p1 \cdot \hat{x}'_{2,i} + l_1 \cdot \left(Y_i - \hat{Y}_{i-1}\right) + k1 \cdot aux_i\right)}_{\Delta\hat{Y}_i},
\end{aligned} \tag{4a}$$

and the final form of *ADRC* control law is computed as:

$$u_i = \frac{1}{b_0}\left[k_p \cdot \left(Y_{sp,i} - \hat{Y}_i\right) - k_{d1} \cdot \hat{x}'_{2,i} - k_{d2} \cdot \hat{x}'_{3,i} - \hat{x}'_{4,i}\right], \tag{4b}$$

where i is discretization instant. The code is configured by the binary switches $p1$, $p2$, $k1$, $k2$, $k3$, based on two least significant bits of the binary representation of the adjusted *ADRC* order n as: $p1 = n_1$, $p2 = n_1 \cdot n_0$, $k1 = \overline{n}_1 \cdot n_0$, $k2 = n_1 \cdot \overline{n}_0$, $k3 = p2$, and on the values of tuning parameters l_1, l_2, l_3, l_4, k_p, k_{d1}, k_{d2} that are adjusted as described in previous section and for different values of n, some of them can be zero.

Despite of adjusted value of n, state variable $\hat{x}'_{4,i}$ always represents a current estimation of the *total disturbance h*. At the same time, respectively, $\hat{x}'_{2,i}$ and $\hat{x}'_{3,i}$ always represent a current estimation of the consecutive time derivatives: $Y_i^{(1)}$ (for $n = 2$ and $n = 3$) and $Y_i^{(2)}$ (for $n = 3$). The backward calculations ensure that at each cycle, the latest information is propagated from $\hat{x}'_{4,i}$ to $\hat{Y}_i$ and finally to the control law (4b). This approach is equivalent to the concept of the *current observer* [10–11, 17, 19] but in the form suitable for higher-order *ADRC* implementation. An additional benefit from this approach is the easy implementation of bumpless switching, which is not trivial for higher order *ADRC* – this problem is discussed later in the paper.

3.3. ADRC tuning

Another open question is optimal/robust *ADRC* tuning [12, 26]. For practitioners, it is extremely important to have reliable and easy-to-apply tuning method that can be based only on limited knowledge on the process dynamics (e.g. on the approximating model of its step response). So far, the most suitable method was suggested in [5]. The controller bandwidth should be adjusted as $\omega_c = (3 \div 5)/\tau_{set}$, for the desired closed-loop settling time τ_{set}, depending on the process relative order. Then, the *ESO* bandwidth should be adjusted as $\omega_o = (3 \div 10) \cdot \omega_c$ but in the practice, its higher limit should be based on the *ADRC* sampling time T_s and on the impact of the measurement noise. Then, *ADRC* and *ESO* tunings from (4) can be easily computed from ω_c and ω_o [5, 10, 12, 19], subject to adjusted *ADRC* order n.

Finally, assuming that *ADRC* order matches the relative process order, the value of the scaling parameter b_0 can be adjusted depending on this order and on the parameters of linearised process dynamics. Thus, not only the process gain k but also the process time constants should be known for proper *ADRC* tuning: T_1 for $n = 1$, $T_1 \geqslant T_2$ for $n = 2$ and $T_1 \geqslant T_2 \geqslant T_3$ for $n = 3$. Then, respectively, the "true" value of b_0 should be adjusted as k/T_1, $k/(T_1 \cdot T_2)$ and $k/(T_1 \cdot T_2 \cdot T_3)$. However, in the industrial practice, the tuning is usually based on *FOPDT* (First Order + Dead Time) approximation of the process step response but for higher order processes, the same *FOPDT* approximation can correspond to process dynamics of different order and of different combination of time constants. Thus, the reliable method is suggested for adjusting the scaling factor b_0 for the second and third order processes, based only on *FOPDT* approximation of the step response.

The concept comes directly from the *half-rule* [21] dedicated to determining the *FOPDT* approximation of higher order processes based on their known dynamics. In this paper, the *inverse half-rule* is proposed for the second and third order processes without dead time. The procedure starts from determining the

parameters of *FOPDT* approximation of the process step response: process gain k, its substitute time constant T and dead time T_0. This approximation can be obtained by any reliable method and then, the following cases should be considered subject to the assumed process order:

- for $n = 2$, the time constants $T_1 \geqslant T_2$ can be directly calculated as $T_1 = T - T_0$, $T_2 = 2 \cdot T_0$ and then, the scaling factor can be calculated as $b_0 = k/(T_1 T_2)$.
- for $n = 3$, the time constants $T_1 \geqslant T_2 \geqslant T_3$ cannot be explicitly determined so the corresponding scaling factor for *ADRC* cannot be directly determined as $b_0 = k/(T_1 T_2 T_3)$. It results from the fact that for third-order processes, this problem cannot be solved explicitly because potentially, there are many combinations of $T_1 \geqslant T_2 \geqslant T_3$ that can result in the same *FOPDT* approximation. Thus, instead, it is suggested to use *inverse half-rule* to determine the acceptable range of the scaling factor $b_0 \in \left[b_{0,\min}, b_{0,\max}\right]$. Each value chosen within this range must correspond to time constants $T_1 \geqslant T_2 \geqslant T_3$ that represent the same *FOPDT* approximation parameters T, T_0 determined from process step response. Consequently, the respective optimization problem can be defined formally in the following way. Let us define $b_{0,\min} = k/\left(T_{1\max} \cdot T_{2\max} \cdot T_{3\max}\right)$, $b_{0,\max} = k/\left(T_{1\min} \cdot T_{2\min} \cdot T_{3\min}\right)$, where:

$$\left(T_{1\max}, T_{2\max}, T_{3\max}\right) = \max_{T_1,T_2,T_3} \left(T_1 \cdot T_2 \cdot T_3\right), \tag{5a}$$

$$\left(T_{1\min}, T_{2\min}, T_{3\min}\right) = \min_{T_1,T_2,T_3} \left(T_1 \cdot T_2 \cdot T_3\right), \tag{5b}$$

subject to the same inequality constraint:

$$T_1 \geqslant T_2 \geqslant T_3 > 0, \tag{6}$$

and two equality constraints from *half rule*:

$$T = T_1 + \frac{T_2}{2}, \qquad T_0 = \frac{T_2}{2} + T_3. \tag{7}$$

After including the equality constraints into the objective function $(T_1 \cdot T_2 \cdot T_3)$, solving the optimization problem (5a) always leads to the following formulas:

$$\begin{aligned} T_{1\max} &= \frac{(2 \cdot T - T_0) + \sqrt{(T - T_0)^2 + T \cdot T_0}}{3}, \\ T_{2\max} &= 2 \cdot (T - T_{1\max}), \\ T_{3\max} &= T_0 - \frac{T_{2\max}}{2}, \end{aligned} \tag{8}$$

and these formulas always satisfy the inequality constraint (6). Thus, the lower limitation $b_{0,\min} = k/(T_{1\max} \cdot T_{2\max} \cdot T_{3\max})$ can be always determined based only on the formulas (8). Similarly, solving the optimization problem (5b) leads to:

$$\begin{aligned} T_{1\min} &= \frac{(2 \cdot T - T_0) - \sqrt{(T - T_0)^2 + T \cdot T_0}}{3}, \\ T_{2\min} &= 2 \cdot (T - T_{1\min}), \\ T_{3\min} &= T_0 - \frac{T_{2\min}}{2}, \end{aligned} \tag{9}$$

but in this case, the minimum is located on one of the equality constraints (7) and then, from (7), the limit values of time constants can be calculated as: $T_1^- = T - 0.5 \cdot T_2^+$, $T_1^+ = T - 0.33 \cdot T_0$, $T_2^- = T_3^+ = T_0/1.5$, $T_2^+ = \min(T/1.5, 2 \cdot T_0)$, $T_3^- = T_0 - 0.5 \cdot T_2^+$. Consequently, for (5b), the upper limit is $b_{0,\max} = \max(k/(T_1^- \cdot T_2^+ \cdot T_3^-), k/(T_1^+ \cdot T_2^- \cdot T_3^+))$. Finally, after computing the acceptable range for the scaling factor $b_0 \in [b_{0,\min}, b_{0,\max}]$, the lower limitation $b_0 = b_{0,\min}$ is advised for ADRC tuning.

In both cases, the value of b_0 should be considered as the starting point for further retuning. The suggested method allows for an approximation of the initial "safe" value of b_0, which makes the tuning faster and reliable. It is a very useful tool because, in the majority of practical industrial cases, multiple trials and error tuning are not acceptable.

The proposed method is illustrated for two example transfer functions representing different configurations of the pneumatic setup described later in the paper and for their *FOPDT* approximations obtained from step response by two-point method [18]:

$$K_1(s) = \frac{0.56}{(9.5s+1)(4.2s+1)} \approx \frac{0.56}{11s+1} e^{-3.3s}, \tag{10a}$$

$$K_2(s) = \frac{0.73}{(7.3s+1)(2.2s+1)(1.4s+1)} \approx \frac{0.73}{8.3s+1} e^{-3s}. \tag{10b}$$

Then, assuming that only the process order is known, the suggested *inverse half-rule* was applied from both *FOPDT* approximations. The results are shown in Table 1.

For (10a) and for the 2nd order ADRC, the *inverse half-rule* approximation of b_0 is very close to its "true" value. For (10b) and the 3rd order ADRC, the "true" value of b_0 falls within the range $[b_{0,\min}, b_{0,\max}]$ calculated from *inverse half-rule* and its value is closer to the lower limitation $b_{0,\min}$. Readers should note that the accuracy of *inverse half-rule* approximation depends on the method of determining the *FOPDT* model.

Table 1: Results of *inverse-half rule* for the processes (10)

Process	"True" value of b_0	*Inverse half-rule* approximation of b_0
$K_1(s)$	$b_0 = 0.014$	$b_0 = 0.011$
$K_2(s)$	$b_0 = 0.0325$	$b_0 \in [0.0236, 0.1022]$

4. Additional functionalities for reliable industrial applications

4.1. Constraints of manipulating variable and equivalent of anti-windup action

In practice, the manipulated variable is always constrained as $u \in (u_{\min}, u_{\max})$, which results from technological requirements or from limitations of the output of the control device. Usually, these limitations are constant but in some cases, they can vary subject to disturbances variations and the *ADRC* function block should provide the inputs that allow for their continuous update.

For *ADRC* controller, integration of the control error e is substituted by integrating the modelling error (see $\hat{x}_4'$ in *ESO* (4a)), which compensates for modelling inaccuracies and ensures offset-free control. Consequently, when the *ADRC*-based control system operates at the stable operating point and when the manipulated variable u is bounded (which always holds for constrained controller output), the process output is bounded as well and the limiting value of the additional state estimate $\hat{x}_4'$ is also bounded. Thus, in opposition to the conventional integral action, the value of $\hat{x}_4'$ remains constant after converging, even if the static control error $e \neq 0$ exists in the system due to the constraints of the controller output. However, it is potentially dangerous when the convergence of $\hat{x}_4'$ does not stop when the controller output u has reached its saturation $u_{\min}$ or $u_{\max}$. Then, the reversal of the MV u is expected immediately when the controller output u falls within its operating range and it requires implementing the equivalence of anti-windupaction. The estimate $\hat{x}_4'$ is computed by *ESO* (4a) only if the controller output u calculated by the control law (4b) is at its acceptable range $(u_{\min}, u_{\max})$. When u falls outside this range, the update of $\hat{x}_4'$ is automatically frozen while the other *ESO* states should be successively computed without any interference.

4.2. Bumpless switching

For *ADRC*ontroller, the mismatch between the output of the process and of its simplified model is compensated by on-line updating the value of $\hat{x}_4'$ in (4a) by continuous integration of modelling error. Thus, for the bumpless switching from MAN to AUTO, the internal process model should be continuously updated not only in AUTO mode but also in MAN mode [11, 15–16]. It ensures that this

model matches the process output even when the controller operates in MAN mode. Then, this switching can be bumpless.

When *ADRC* function block operates in MAN mode, the value of u is adjusted manually as u_{man} and *ESO* (4a) is computed on-line ensuring that the current value of $\hat{x}'_4$ compensates all inaccuracies of the model (1) for $u = u_{man}$. This technique is effective when switching takes place at the steady state (for control error $e = 0$). Otherwise, if switching takes place in transient (for $e \neq 0$), the value of the estimate $\hat{x}'_4$ should be adjusted to satisfy the control law (4b) for the manually adjusted value u_{man}. In other words, in MAN mode, the inverse control law with $u = u_{man}$ should be solved for the value of $\hat{x}'_4$:

$$\hat{x}'_{4,i} = -b_0 u_{man} + k_p \left(Y_{sp,i} - \hat{Y}_{i-1}\right) - k_{d1}\hat{x}'_{2,i-1} - k_{d2}\hat{x}'_{3,i-1} + l_4 \left(Y_i - \hat{Y}_{i-1}\right), \quad (11)$$

In the first equation of (4a), at the moment of switching, the value of $\hat{x}'_{4,i}$ is computed by (11). Then, in AUTO mode, the *ESO* calculations are carried out according to (4a).

4.3. Derivative backoff and anti-backoff action

Derivative backoff [23] occurs for controllers with derivative action. When the measurement of the controlled variable Y reaches the sensor saturation (or returns from it), the jump of the derivative action appears that is opposite to what is expected when the sensor operates normally. This phenomenon is important for *ADRC* practical implementation because *ESO* (4a) computation is based on the on-line measurement of Y. Thus, the sensor saturation does introduce derivative backoff and for the *ADRC* order $n > 1$, derivative backoff always appears. The complexity of the required anti-backoff action increases with the increment of the order n. At the same time, anti-backoff action suggested for PID controller [23] cannot be applied for *ADRC* due to the lack of the integral action.

The impact of derivative backoff is presented in Fig. 1 (solid lines) for the closed-loop system with the example process (10b) and the *ADRC* controller ($n = 3$, $b_0 = 0.023$, $\omega_o = 1.75$, $\omega_c = 0.35$). The sensor saturation was adjusted as $Y_{sat} = 1.9$ and the load disturbance was applied at $t = 10$.

Assuming that Y_{sat} is known, the suggested anti-backoff strategy is based on (4). When the saturation is reached, all estimates of the time derivatives of the controlled variable $\hat{x}'_2$ and $\hat{x}'_3$ become zero, which results in the jump of the manipulating input. This jump can be avoided by proper adjusting the value of $\hat{x}'_4$. At the time of reaching the saturation, (4a) can be rewritten and solved for:

$$\begin{aligned}
\hat{x}'_{4,i} &= -\left(k1 \cdot l_1 + k2 \cdot l_2 + k3 \cdot l_3\right) \cdot \left(Y_{sat} - \hat{Y}_{i-1}\right) - \left(k1 + k2 + k3\right) \cdot b_0 \cdot u_{i-1}, \\
\hat{x}'_{3,i} &= -p2 \cdot \left(l_2 \cdot \left(Y_{sat} - \hat{Y}_{i-1}\right) - k2 \cdot aux_i\right), \\
\hat{x}'_{2,i} &= -p1 \cdot \left(l_1 \cdot \left(Y_{sat} - \hat{Y}_{i-1}\right) - k1 \cdot aux_i\right)
\end{aligned} \quad (12a)$$

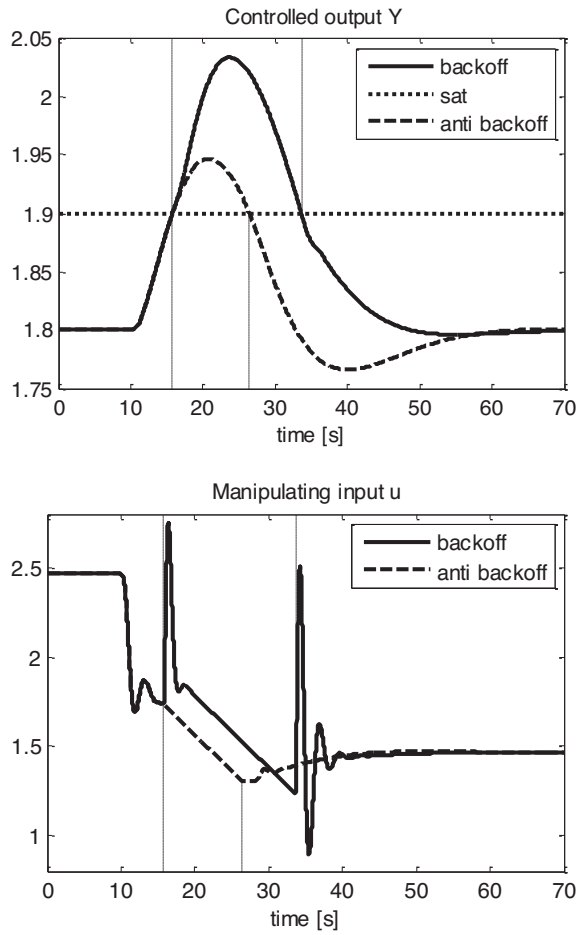

Figure 1: Impact of derivative backoff and preventing an anti-backoff action for ADRC-based closed loop system. Vertical lines indicate the moments of reaching the sensor saturation and of returning. The dotted line represents the sensor saturation Y_{sat}

and then, after inversing (4b) and introducing (12a), for $\Gamma = (p1 \cdot k_{d1} \cdot l_1 + p2 \cdot k_{d2} \cdot l_2 + k1 \cdot l_1 + k2 \cdot l_2 + k3 \cdot l_3)$:

$$\hat{Y}_i = \frac{k_p \cdot Y_{sp,i} + \Gamma \cdot Y_{sat}}{k_p + \Gamma}. \tag{12b}$$

The values (12a) must be computed once the controlled variable has reached the saturation. Then, they should be applied as the initial values for (4) and the computation should continue until the measurement of Y returns from the saturation and anti-backoff action for this returning should be applied. This action utilizes the current values of $\Delta\hat{Y}_i$, $\Delta\hat{x}'_{2,i}$ and $\Delta\hat{x}'_{3,i}$ defined in (4a) but they depend

on the slope of returning from the saturation that cannot be predicted. Thus, after returning from saturation, the current value of the manipulating variable u_i should be frozen for a certain period of time τ_{wait} to allow for *ESO* (4a) convergence. Then, (4) can be computed normally. This period of time should be determined from the adjusted observer bandwidth as $\tau_{wait} = (3 \div 4)/\omega_o$.

The results of the suggested preventing anti-backoff action is shown in Fig. 1 with dashed lines. Very significant jumps in the manipulating variable resulting from derivative backoff are smoothed at the moments of reaching and returning from saturation. However, the significant underoscillation appears in the control performance. Its impact increases for more aggressive *ADRC* tuning and can be limited if tuning is more conservative. However, in practical cases, it is very difficult to determine the price of anti-backoff action in the control performance. One can be only sure that this action smoothes the variations of the manipulating variable, which proves its reliability. Thus, after a huge number of experiments and due to large uncertainty on the expected results, it was decided not to implement anti-backoff action in the general purpose *ADRC* function block. However, the concept works and the user can consider its implementation.

5. General concept of ADRC function block

General-purpose *ADRC* implementation was designed as a library function block that should be easily accessed and utilized in a standard manner. It encapsulates all process-independent calculations (4), as well the additional functionalities discussed previously. Its simplified diagram with its input-output specification defined to allow for easy and on-line parameterization is presented in Fig. 2. *Tuning & configuration* adjusts tunings and configuration switches for (4). This block also provides errors information when any errors occur during operation. *ESO* computes (4a) and *Control law* computes the manipulated variable u_{raw} by (4b). *MV constraints* provide limitations for the manipulating variable u applied to the process.

This block can be executed in PLC by calling it as the subroutine in the control loop that should be executed with the sampling time T_s, in the same way, in which the general-purpose built-in library PID function block is implemented in the industrial control applications. The user-defined inputs can be aggregated into three groups:

- process-related inputs extracted from measurement data: controlled variable Y and on-line measurement of manipulated variable u_{real},
- tuning and parameterization inputs: *ADRC* order n, parameters b_0 and ω_o, ω_c, sampling time T_s and the time constant of the reference trajectory T_{ref} that potentially filters the variations of the set point Y_{sp},

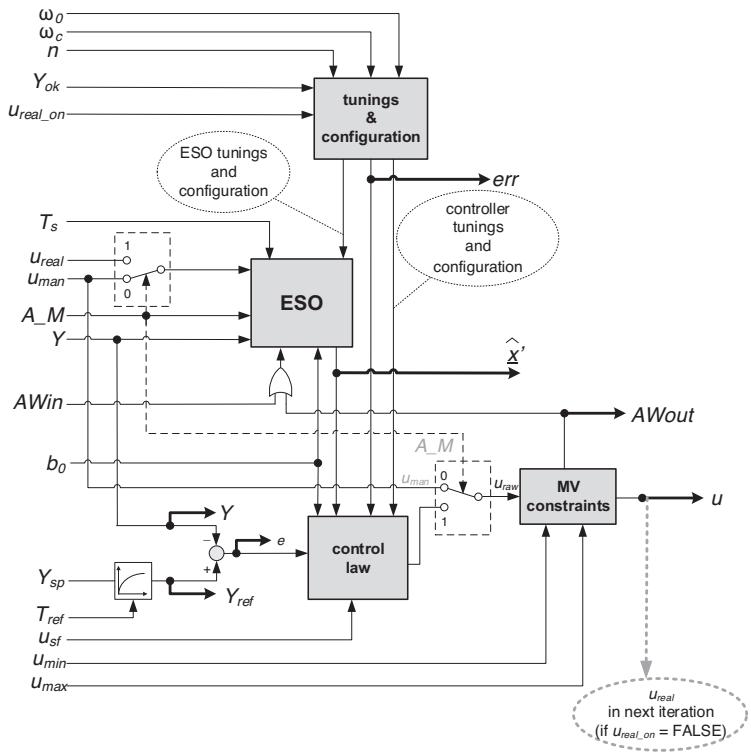

Figure 2: Schematic diagram of the *ADRC* function block. User-configurable inputs are grouped on the left and the accessible outputs are indicated by thick lines

- other parameterization inputs required for reliable control in different control structures: constraints for manipulated variable ($u_{\min}$, $u_{\max}$), binary input for switching between AUTO and MANUAL modes (*A_M*), the value of the manipulated variable u_{man} adjusted manually at MANUAL mode, binary input (*AWin*) enabling the anti-windup action when the block operates as the primary controller in the cascade structure, binary input Y_{ok} that determines validity of the measured controlled variable Y, binary input u_{real_on} that determines accessibility of on-line measurement data for manipulating variable (this signal should be connected to the input u_{real}) and u_{sf} determining safe value of manipulated variable applied to the process in emergency cases.

The natural choice for *ADRC* function block outputs are: manipulating variable u applied to the process, control error e, controlled variable Y and the binary signal *AWout* used when the *ADRC* function block operates in the secondary loop in the cascade structure (this signal allows to enable anti-windup action in the primary controller when the computed value of u has reached its saturation).

However, for monitoring the performance of the block, some additional outputs were defined: desired reference trajectory Y_{ref}, state vector $\underline{\hat{x}'}$ computed by *ESO* (4a) and *err* that contains an error code.

The input u_{real} should be used when the real value of manipulating variable applied to the process is measurable online. Then, this data should be connected to u_{real} and binary input u_{real_on} should be enabled. The current measurement data of real manipulating variable improves *ESO* (4a) computation because if the dynamics of the actuator is significant, in transients, the real value of manipulating variable applied directly to the process can be different from the one forced by the controller output. Thus, when the signal u_{real} is accessible, it is preferred to use it for more accurate *ESO* (4a) computation. Otherwise, only the value of u computed by (4b) is accessible and it should be used for *ESO* (4a) computation. Then, nothing should be connected to u_{real} and binary input u_{real_on} should be disabled.

6. Practical validation of ADRC function block

Practical validation of the general-purpose *ADRC* function block was based on the pneumatic setup shown schematically in Fig. 3. It represents the third order dynamical process that consists of three serially connected tanks of the respective volumes $V_1 = 5$ [L], $V_2 = 2$ [L] and $V_3 = 0.75$ [L]. The relative pressures at each tank are denoted as p_1, p_2, p_3 [bar]. The system is supplied with the air of the relative pressure p_s [bar] and between the tanks, the air flows through the constant pneumatic resistances R_{pa}, R_{pb}, R_{pc} [1/(m·s)]. From the last tank, the air flows out through the adjustable pneumatic resistance R_{pout} [1/(m·s)] and the relative pressure outside the tank is denoted as $p_4 = 0$[bar]. The supplying relative pressure p_s is adjusted by the proportional valve MPPES-3-1 from Festo®within the range 0–5 [bar]. All the pressures p_s, p_1, p_2, p_3 are measured on-line by the SDE1 pressure sensors and the pneumatic resistance R_{pout} at the outlet from the third tank can be changed by switching between two pneumatic valves of different resistance. All process signals are connected to analog I/O modules in ET-200 module connected to PLC SIEMENS® S7-1500. All data is visualized by SIEMENS® panel KTP 1000 and stored in SIEMENS® S7-1500 unit.

The control goal is defined to stabilize the process output $Y = p_3$ at the set-point Y_{sp} by manipulating supplying pressure $u = p_s$, both for tracking and for disturbance rejection. For the latter, two different disturbances are applied. One is the load disturbance emulated by adding Δp_s directly to the manipulating variable. The other is switching between two different pneumatic resistances R_{pout}. This switching not only disturbs the process but also significantly changes its dynamics.

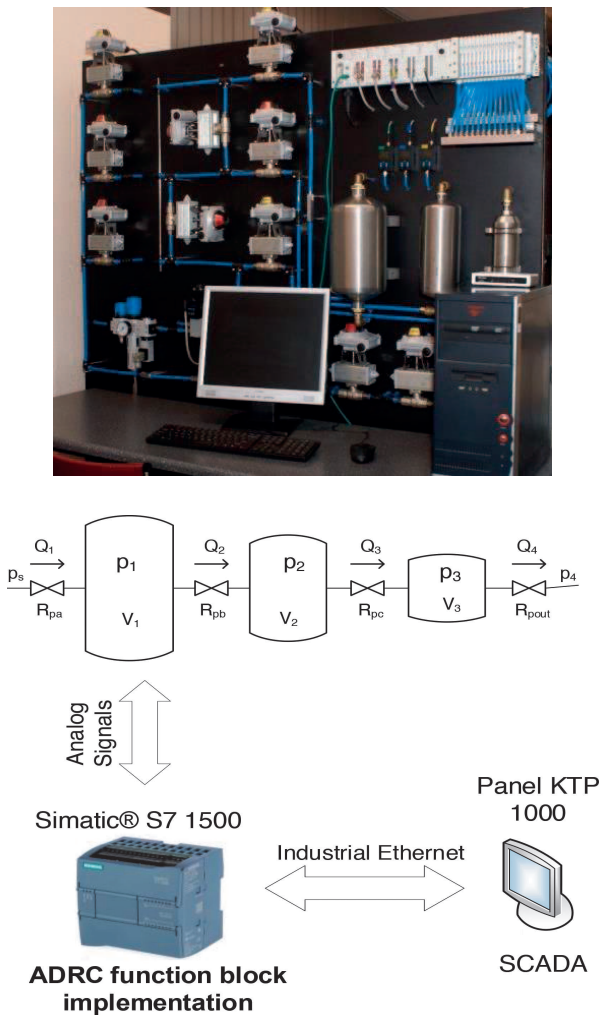

Figure 3: Pneumatic setup and its simplified scheme. The lower diagram also shows the simplified hardware diagram of the setup configuration

For the experimental tests, the general-purpose library ADRC function block shown in Fig. 4 was implemented in SCL (Structured Control Language) in TIA Portal environment. This block was prepared according to the input-output specification and it includes functionalities presented above in the paper. The block was executed in OB35 block with the sampling time $T_s = 0.1$ [s], jointly with the scaling functions, for clarity not shown in Fig. 4. The choice of the sampling time is justified by the process dynamics and in the practice when the industrial processes are considered, the sampling time can be adjusted even larger without any significant influence on the control performance.

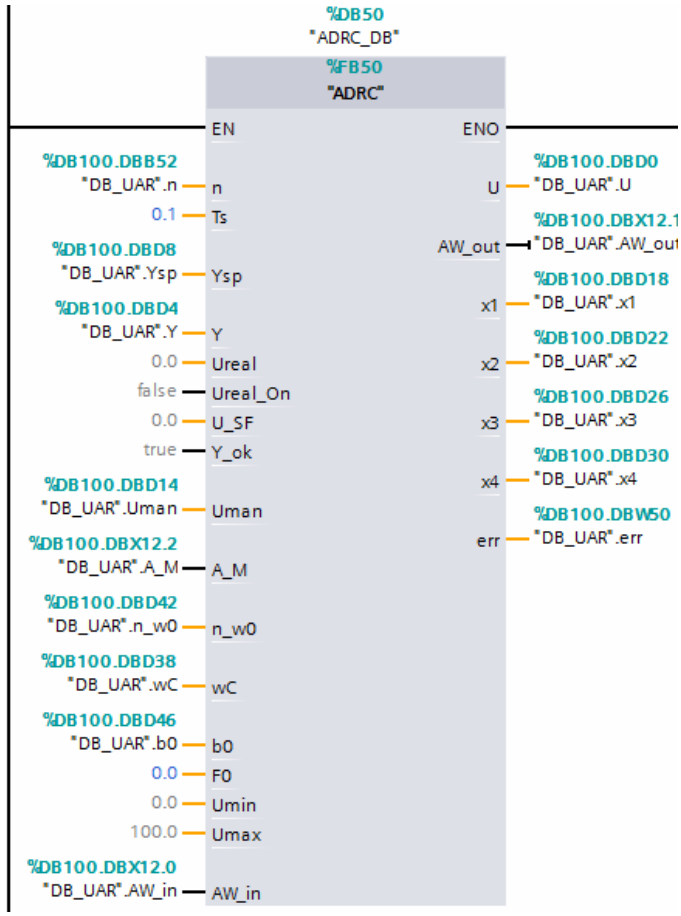

Figure 4: Implementation of the library ADRC function block for SIEMENS®S7-1500

The first stage of experiments was dedicated to validation of the suggested *inverse half-rule* for tuning ADRC controller of different order n, in the application to control the same pneumatic setup representing the third order dynamical process. The same experiment depicted in Fig. 5 was carried out for ADRC function block configured respectively for $n = 1$, 2 and 3. For each case, the scaling factor b_0 was estimated from FOPDT approximation of the process step response. For $n = 1$, the estimation was derived directly from the first order part as $b_0 = k/T$ and for $n = 2$ and 3, the *inverse half-rule* was applied. Then, the experimental retuning was needed to obtain the final adjustment of b_0 and to adjust the other ADRC tunings ω_o, ω_c. The final tunings are shown in Table 2. For $n = 1$, the initial estimation of b_0 is far from its final adjustment due to a very significant mismatch between the order of the process and the applied ADRC order. The final adjust-

ment is 3 times larger than its initial estimation. For $n = 2$, the estimation of b_0 is better and its final value is only 2 times larger. For $n = 3$, the final adjustment of b_0 is equal to the lower limitation obtained from the suggested *inverse half-rule*. In each case, from a practical viewpoint, the estimation of b_0 is accurate enough to avoid blind trial and error experimental tuning. The estimated values do not provide the best possible control performance but even if no further retuning was applied, the ADRC performance is acceptable.

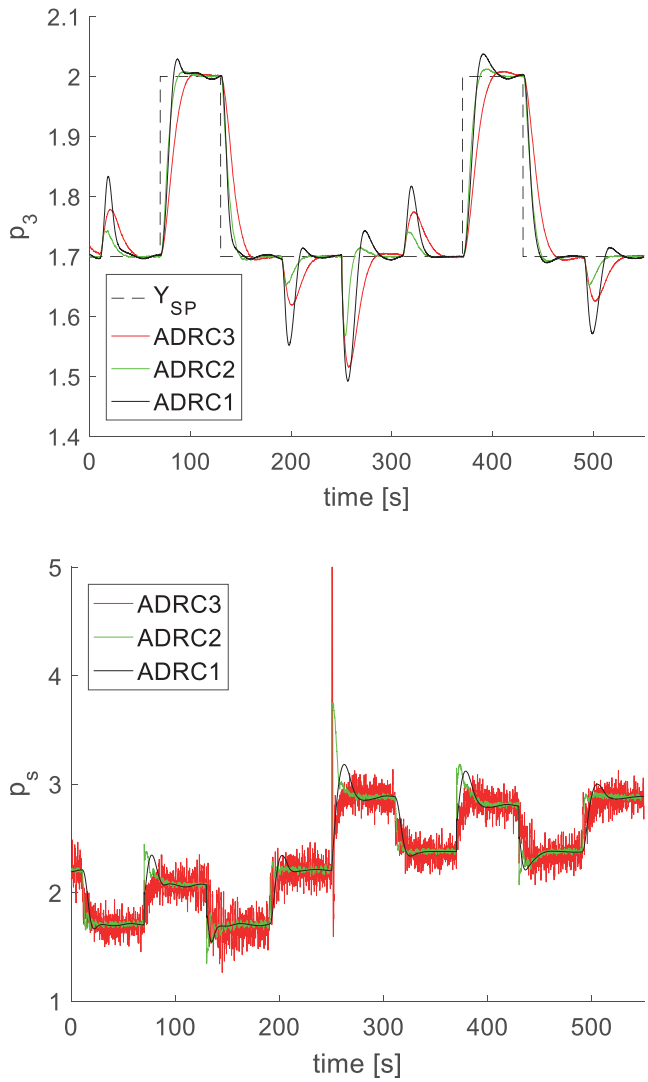

Figure 5: Experimental comparison of the control performance between ADRC of a different order ($n = 1$, 2 and 3) for the 3rd order pneumatic process. The upper diagram shows the controlled variable and the lower – manipulating variable

Table 2: Tunings of ADRC adjusted for experimental tests

ADRC order	Approximation of b_0	Final adjustment of b_0	ω_o	ω_c
$n = 1$	0.099	0.3	0.9	0.3
$n = 2$	0.025	0.05	1.75	0.35
$n = 3$	[0.025, 0.13]	0.025	1.75	0.35

The results are presented in Fig. 5. For each case of n and for the suggested tuning, ADRC controller provides acceptable control performance. However, the best tracking and disturbances rejection are obtained for $n = 2$, which is lower from the order of the real process dynamics. Retuning does not change this picture significantly. For $n = 1$, the dynamical properties of ADRC do not allow for effective compensation for the process dynamics. On the other hand, for $n = 3$, for which the best results should be expected, more aggressive tuning results in large chattering in the manipulating variable without significant benefit in the control performance. Thus, it can be expected that for higher order process dynamics, the ADRC of the second order should be suggested.

At the second stage, the functionalities of the ADRC function block were tested and Fig. 6 shows the results for the most complex case (ADRC function block with $n = 3$, tuned as in Table 2). The experiment scenario includes denoted switching from MAN to AUTO, changing the setpoint Y_{sp} and operating the process at the limitations of the MV u ($u_{\min} = 0$ and $u_{\max} = 5$). The results show the bumpless switching from MAN to AUTO mode and the immediate reversal of the MV u when the controller output u falls within its operating range. The latter proves the functionality of the implemented equivalent anti-windup action.

Finally, to convince the practitioners, the comparison with the conventional PID controller was carried out. For comparison, only ADRC for $n = 2$ and tuned as in Table 2 was considered because it ensures the best control performance (see Fig. 5). PID controller was implemented based on the PID Compact function block from standard SIEMENS®TIA PORTAL library. Its tuning was carried out by the built-in auto-tuning functionalities including *pretuning* (PID_{PT} with gain $k_P = 5.69$, integral action time $T_I = 16.65$ [s] and derivative action time $T_D = 2.91$ [s]) and *finetuning* (PID_{FT} with gain $k_P = 6.31$, integral action time $T_I = 5.40$ [s] and derivative action time $T_D = 1.36$ [s]). Readers should note that in the industrial practice, the autotuning procedure is the most popular method for PID tuning so its choice for comparative studies is not accidental. The results are presented in Fig. 7 and they show that ADRC rm significantly outperforms both PID_{PT} and PID_{FT}. It ensures the best compromise between good tracking and very good disturbances rejection.

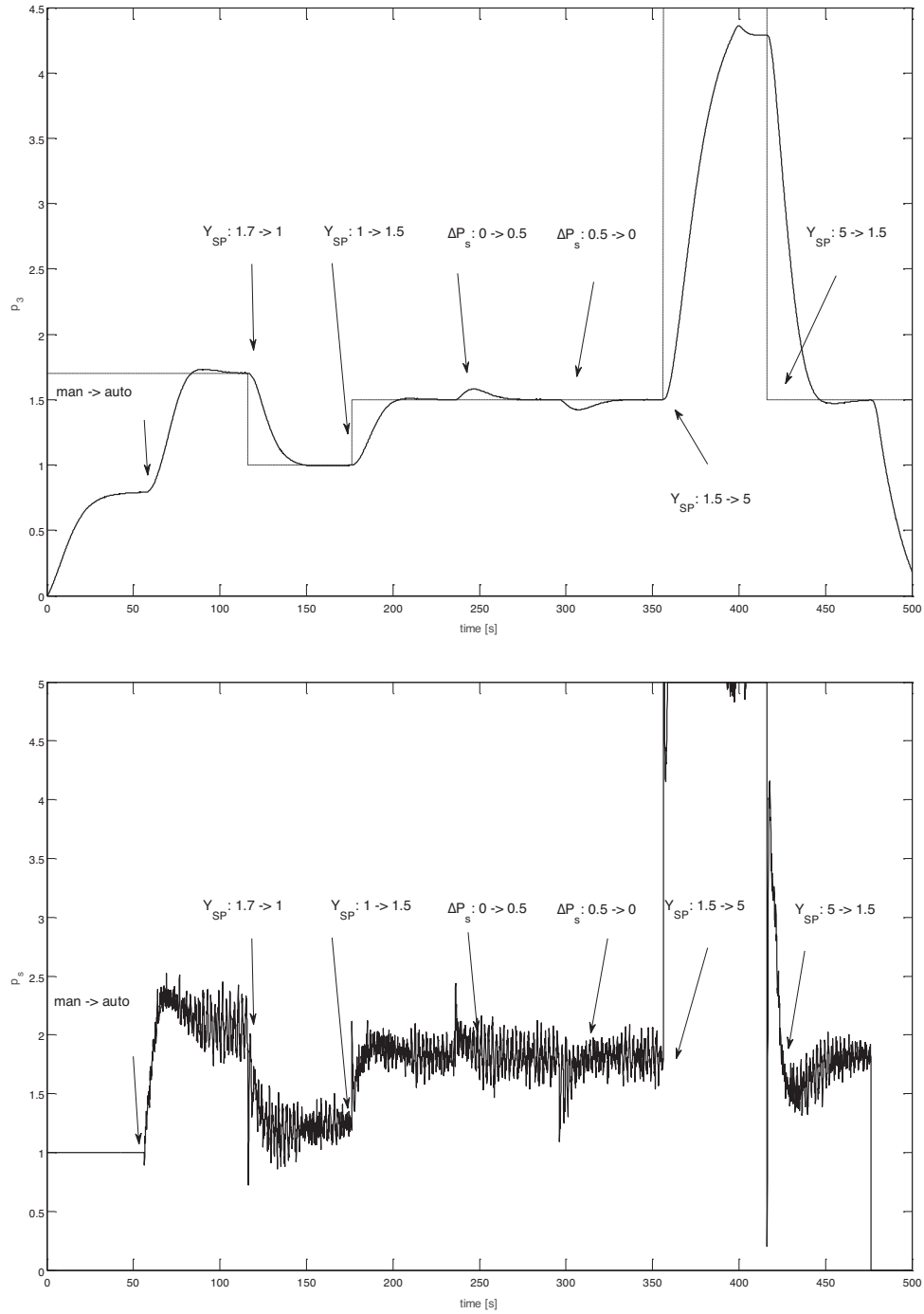

Figure 6: Experimental validation of the additional functionalities of the ADRC function block

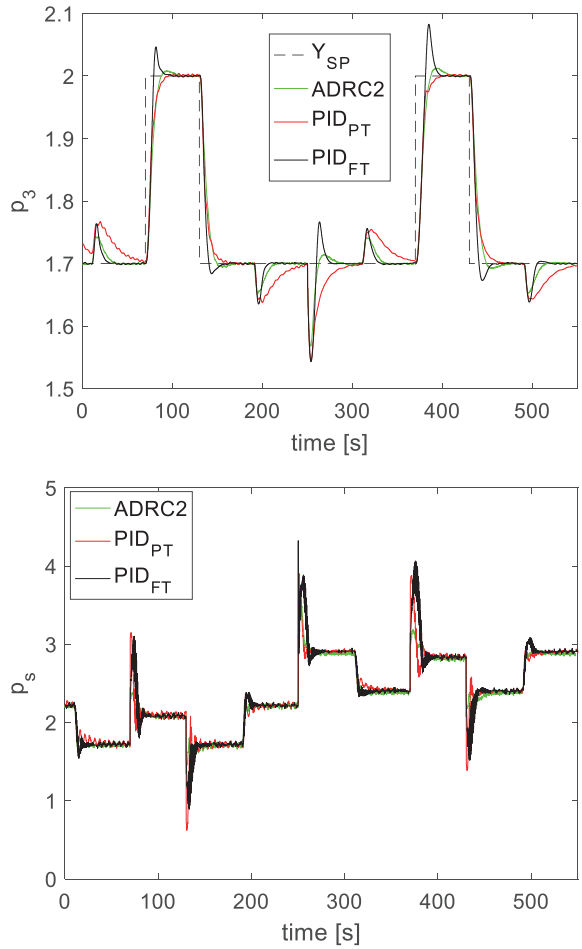

Figure 7: Experimental comparison of the control performance between ADRC ($n = 2$) and PID auto-tuned by pretuning (PID_{PT}) and by finetuning (PID_{FT}), for the 3rd order pneumatic process. The upper diagram shows the controlled variable and the lower – manipulating variable

7. Conclusions

This paper shows the practical implementation of the ADRC controller in the form of the general-purpose function block. The designed block was tested experimentally and the results show its applicability in the industrial control systems. Jointly with the suggested *inverse half-rule* that allows for approximate estimation of the ADRC tuning parameter b_0, the practitioners obtain the powerful tool for fast, relatively easy and reliable ADRC implementation. This implementation is possible even in the complex control systems because the mean

execution time of the ADRC function block is about 9.1 [ms], which is comparable with the execution time of the standard SIEMENS®library PID Compact function block that is about 11 [ms]. Additional comparison of the control performance between the ADRC function block tuned according to the suggestions presented in this paper and the PID Compact function block tuned by the built-in autotuning procedure shows the superiority of ADRC. Thus, better control performance jointly with easy implementation in the form of the general-purpose function block and with the reliable tuning method makes ADRC a very interesting alternative for the application in the regulatory-level industrial control systems.

Very important problem that also should be addressed is the influence high frequency disturbances on the control performance of ADRC. When higher order of ADRC is applied, one should expect that the measurement noise can be magnified due to the presence of higher order derivative action. This phenomenon can be clearly seen in Fig. 5 (lower diagram) where chattering of the manipulating signal increases with the order of ADRC. Consequently, this effect must be considered when such a chattering is not allowed in the control system, e.g. due to limited service life of the actuator.

References

[1] K.J. ASTRÖM KJ and T. HÄGGLUND: The future of PID control, *Control Engineering Practice*, **9**(14), (2001), 1163–1175.

[2] J. CZECZOT: On possibilities of the practical implementation of balance-based adaptive control methodology, *Control and Cybernetics*, **36**(4), (2007) 967–984.

[3] H. FALLAHSOHI, C. CHANGENET, S. PLACE, C. LIGERET and X. LIN-SHI: Predictive functional control of an expansion valve for minimizing the superheat of an evaporator, *Int. Journal of Refrigeration*, **33**, (2010), 409–418.

[4] Z. GAO, Y. HUANG and J. HAN: An alternative paradigm for control system design, *Proc. of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*, **5** (2001), 4578–4585.

[5] Z. GAO: Scaling and bandwidth-parametrization based controller tuning, *Proc. of American Control Conference*, **3** (2003), 4989–4996.

[6] Z. GAO: Active Disturbance Rejection Control: A paradigm shift in feedback control system design, *Proc. of American Control Conference*, ACC, (2006), 2399–2405.

[7] Z. GAO: On the centrality of disturbance rejection in automatic control, *ISA Transactions*, **53** (2013), 850-857.

[8] J. HAN: From PID to Active Disturbance Rejection Control, *IEEE Transaction on Industrial Electronics*, **56** (2009), 900–906.

[9] HARJUNKOSKI, R. NYSTRÖM and A. HORCH: Integration of scheduling and control – Theory or practice?, *Computers and Chemical Engineering*, **33** (2009), 1909–1918.

[10] G. HERBST: A simulative study on Active Disturbance Rejection Control (ADRC) as a control tool for practitioners, *Electronics*, **2** (2013), 246–279.

[11] G. HERBST: Practical Active Disturbance Rejection Control: Bumpless Transfer, Rate Limitation, and Incremental Algorithm, *IEEE Trans. on Industrial Electronics*, **63**(3), (2016), 1754–1762.

[12] Y. HUANG and W. XUE: Active disturbance rejection control: Methodology and theoretical analysis, *ISA Transactions*, **53**(4), (2014), 963–976.

[13] T. KLOPOT, J. CZECZOT and W. KLOPOT: Flexible function block for PLC-based implementation of the balance-based adaptive controller, *Proc. of 2012 AMERICAN CONTROL CONFERENCE (ACC) Book Series: Proceedings of the American Control Conference* (2012), 6467–6472.

[14] T. KLOPOT, K. STEBEL, J. CZECZOT and P. LASZCZYK: Function block practical implementation of Balance-Based Adaptive Control for pH process, *Proc. of 39TH ANNUAL CONFERENCE OF THE IEEE INDUSTRIAL ELECTRONICS SOCIETY (IECON 2013), IEEE Industrial Electronics Society* (2013), 3549–3554.

[15] T. KLOPOT, P. LASZCZYK, K. STEBEL and J. CZECZOT: Flexible function block implementation of the balance-based adaptive controller as the potential alternative for PID-based industrial applications, *Trans. of the Institute of Measurement and Control*, **36**(10), (2014), 1098–1113.

[16] P. LASZCZYK: Programmable LabView and LabWindows/CVI PFC controller, *Proc. of the 8th IEEE International Conference on Methods and Models in Automation and Robotics, MMAR 2002* (2002), 1135–1138.

[17] R. MADOŃSKI and P. HERMAN: Survey on methods of increasing the efficiency of extended state disturbance observers, *ISA Transactions*, **56** (2015), 18–27.

[18] T.E. MARLIN: *Process Control. Designing Processes and Control System for Dynamic Performance*, McGraw-Hill, New York (1995).

[19] R. MIKLOSOVIC, A. RADKE and Z. GAO: Discrete implementation and generalization of the extended state observer, *Proc. of the American Control Conference, ACC 2006* (2006), 2209–2214.

[20] R.C. PANDA, C.C. YU and H.P. HUANG: PID tuning rules for SOPDT systems: Review and some new results, *ISA Transactions*, **43** (2004), 283–295.

[21] S. SKOGESTAD: Simple analytic rules for model reduction and PID controller tuning. *Journal of Process Control*, **11**, (2003), 291–309.

[22] L. SUN, D. LI, K. HU, K.Y. LEE and F. PAN: On Tuning and Practical Implementation of Active Disturbance Rejection Controller: A Case Study from a Regenerative Heater in a 1000 MW Power, *Plant. Ind. Eng. Chem. Res.*, **55**(23), (2016), 6686–6695.

[23] A. THEORIN and T. HÄGGLUND: Derivative backoff: The other saturation problem for PID controllers. *Journal of Process Control*, **33**, (2015), 155–160.

[24] G. VALENCIA-PALOMO and J.A. ROSSITER: Programmable logic controller implementation of an auto-tuned predictive control based on minimal plant information, *ISA Transactions*, **50** (2011), 92–100.

[25] G. VALENCIA-PALOMO and J.A. ROSSITER: Efficient suboptimal parametric solutions to predictive control for PLC applications, *Control Engineering Practice*, **19** (2011), 732–743.

[26] C. ZHAO and D. LI: Control design for the SISO system with the unknown order and the unknown relative degree, *ISA Transactions*, **53**(4) (2014), 858–872.