Active Disturbance Rejection Control (ADRC) Toolbox for MATLAB/Simulink

Wojciech Giernacki\textsuperscript{a}, Krzysztof Łakomy\textsuperscript{*}, Jacek Michalski\textsuperscript{a}, Rafal Madonski\textsuperscript{b, *}

\textsuperscript{a} Institute of Robotics and Machine Intelligence, Faculty of Control, Robotics and Electrical Engineering, Poznan University of Technology, Piotrowo 3a, 60-965 Poznan, Poland
\textsuperscript{b} Energy and Electricity Research Center, Jinan University, 519070 Zhuhai, P. R. China

Abstract

In this paper, an active disturbance rejection control (ADRC) toolbox for MATLAB/Simulink is introduced. Although ADRC has already been established as a powerful robust control framework with successful industrial implementations and strong theoretical foundations, a comprehensive tool for computer-aided design of ADRC has not been developed until now. The proposed ADRC Toolbox is a response to a growing need of both scientific community and control industry looking for a straightforward software utilization of the ADRC methodology. Its main purpose is to fill the gap between current theories and applications of ADRC and to provide an easy-to-use solution to users in various control fields wanting to employ the ADRC scheme in their applications. The ADRC Toolbox contains a single, general-purpose, drag-and-drop function block allowing to synthesize a predefined ADRC-based strategy with minimal design effort. Its efficacy is validated here in both simulations and hardware experiments, conducted using a variety of problems known from motion, process, and power control areas. The proposed ADRC Toolbox is an open-source project\textsuperscript{1}.

Keywords: active disturbance rejection control (ADRC), control system

\textsuperscript{*}Corresponding author

Email addresses: adrc.toolbox@gmail.com (Krzysztof Lakomy), krzysztof.lakomy92@gmail.com (Krzysztof Lakomy), rafal.madonski@jnu.edu.cn (Rafal Madonski)

\textsuperscript{1}The contents of this paper refer to version 1.1.3 of the ADRC Toolbox; the current version, as well as repository of previous releases, are available at: https://www.mathworks.com/matlabcentral/fileexchange/102249-active-disturbance-rejection-control-adrc-toolbox
1. Introduction

With the development of modern control techniques, new reliable tools for rapid prototyping are being developed alongside. Intensive expansion of the MATLAB/Simulink\(^2\) software environment over the past decade in both academia and industry strongly supports the current trend of using high-level programming for designing, analyzing, and applying control systems. Analyzing the market and the scientific literature, one may find a variety of such software toolboxes supporting the synthesis of control systems, with emphasis on e.g. identification of dynamic systems \([1]\), modeling and solving optimization problems \([2]\), signal differentiators \([3]\), disturbance observers \([4]\), intelligent control based on brain emotional learning \([5]\), fractional-order control \([6]\), or rapid prototyping \([7]\).

Several types of supporting software solutions have been also developed for control systems based on an active disturbance rejection control (ADRC) methodology. The ADRC is a highly practical control framework introduced to the general audience in \([8, 9]\)\(^3\) and later streamlined by \([10]\). Since its inception, it has attracted considerable attention among both scholars and industry practitioners. The effectiveness of its disturbance-centric approach \((11)\) of handling uncertainties in governed systems has been verified to date in numerous control problems seen in motion, process, and power control areas \([12, 13]\). The combination of modern control elements with pragmatism of minimum-modeling approach has made ADRC an interesting alternative to the proportional integral derivative (PID) controllers \([14]\). Recent surveys \([15, 16, 17]\) capture the significant impact ADRC has had on the control field in the last two decades. Conquering market share from the popular PID-type controllers, however, is

\(^2\)MATLAB and Simulink are registered trademarks of the MathWorks, Inc.

\(^3\)The ADRC was formally proposed in 1998 by Prof. Jingqing Han and initial papers are in Chinese - see \([8]\) and references therein.
not an easy feat and demands improvements in many areas. A lot has been achieved by now (see a summary in [18]) and it is thus not surprising that various software programs have been developed over the years to facilitate the implementation of ADRC.

In [19], for example, the ADRC-based system has been implemented in an open-source Scilab/Xcos environment. One of the benefits of using this environment is that the underlying exchange protocol allows the Xcos simulation to be carried out in batch processing in Scilab. The implemented ADRC algorithm consists of linear functions that simplify the design. In [20, 21, 22], various general-purpose function blocks were developed for industrial implementation of ADRC in programmable logic controllers (PLCs). Two versions of ADRC library for PLCs were proposed in [20], one as a standard version (with basic functionalities) and the other dedicated for experienced ADRC users. The latter provides additional features, like partial dynamic model incorporation, relative disturbance order selection, and nonlinear estimation error function. In [21, 22], the idea of general-purpose ADRC function block for PLCs has been realized from the perspective of typical process control systems. It thus addresses practically important aspects like the impact of the derivative backoff, autotuning (based only on the process step response), or reduced-order ADRC and its combination with a Smith predictor for handling system time-delay. Although the above Scilab/Xcos and PLC implementations of ADRC are not dedicated to MATLAB/Simulink nor are publicly available, they provide important information about what parameters and functionalities a practically appealing ADRC software toolbox should poses.

In [23], it was shown how to seamlessly integrate functionality of active disturbance rejection with an existing control technology using a single add-on module, developed in C language. The proposed non-invasive ADRC module was validated experimentally using a laboratory manipulator and a DSP-based control architecture, where the desired position tracking performance was achieved under unknown mass variations and sudden external disturbances. It was shown that the proposed ADRC module increased disturbance rejection
performance without any change in the existing, standard PD controller.

Regarding software support for ADRC implementation in MATLAB/Simulink, to best of our knowledge, only few approaches are currently available. Interactive tuning tools for ADRC in form of graphical user interfaces (GUIs) were developed in [24, 25]. In [24], the ADRC tuning was realized through a dedicated GUI, where the user inputs both the system model and ADRC parameters, then the tool visualizes the calculated tracking and disturbance rejection performances (based on some predefined robustness constraints). The tool can also initiate randomized Monte Carlo tests to check the robustness against system parametric uncertainty in prescribed ranges. In [25], an interactive ADRC design for flight attitude control was proposed. In particular, a CAD software based on MATLAB GUI was designed to integrate the tuning and simulation process with a stability margin tester.

A tutorial for ADRC design has been published in [26]. The work uses two examples of linear ADRC implementation for two nonlinear systems. Although the paper is more of a guide for ADRC design and application rather than a general-purpose toolbox, it nicely shows various modifications of ADRC algorithms and their generalizations to higher order systems. Furthermore, the Authors publicly shared their implementation examples on the official MATLAB File Exchange website, which is dedicated to free, open-source code sharing. A different approach from the one described above can be seen in [27, 28, 29], where MATLAB/Simulink module-based ADRC designs were presented. The works introduced the process of subsystem masking, including function modules, coding, and initialization. The papers describe drag-and-drop creation and encapsulation of different ADRC schemes by manually combing various modules (subsystems) typically seen in a conventional ADRC block diagram, like the tracking differentiator, extended state observer, and state feedback controller. Although the proposed module-based design offers flexibility in structure selection, allowing the ADRC to be customized to a given control scenario, it inevitably increases commissioning time and its difficulty, since the users need to implement and independently tune all the necessary ADRC components them-
selves.

Given the above literature overview, related to the available software solutions for ADRC implementation, a new MATLAB/Simulink toolbox is developed in this work. The proposed ADRC Toolbox expands the current state-of-the-art by distilling the best practices and most useful features into a single tool for continuous-time implementation of ADRC that provides prospect users with software that:

- contains a general-purpose ADRC function block that can be tailored to a given control scenario with minimum tuning parameters;
- is development with functionalities increasing its applicability range and practical appeal;
- can be easily incorporated into a program through drag-and-drop and offers almost plug-and-play operation;
- has a minimalistic and user-friendly graphical interface;
- is freely available to the control community as open-source software at https://www.mathworks.com/matlabcentral/fileexchange/102249-active-disturbance-rejection-control-adrc-toolbox.

The goal of the proposed ADRC Toolbox is to provide an intuitive blockset within MATLAB/Simulink environment, that will allow users to implement the ADRC structure with a relatively low effort and limited knowledge about the details of the observer-based control designs. By doing so, the developed Toolbox potentially helps ADRC to reach wider audience by adopting a suitable language and form that lowers existing barriers for application experts. The introduction of the Toolbox aims at decreasing time and effort needed to design and implement basic ADRC schemes as well as opening up new possibilities to swiftly deploy ADRC to real-world control problems.

In order to validate the efficacy of the proposed ADRC Toolbox for MATLAB/Simulink, several tests are conducted utilizing selected problems from motion, process, and power control areas that differ from each other in terms of
order, type, and structure of the controlled dynamics as well as the type of acting
external disturbance. Some of the validating tests are performed in simulation,
other are realized in laboratory hardware. Such diversified group is deliberately
selected to examine the ADRC Toolbox in various control scenarios.

The rest of the paper is organized as follows. After the notation is established
below, Section 2 revisits the key principles and components behind the ADRC
framework in the specific context of the developed software block. Section 3 de-
scribes the parameters and functionalities implemented in the introduced ADRC
Toolbox. Section 4 presents the validation of the proposed ADRC Toolbox by
showing results of various control problems solved using the Toolbox. Section 5
contains information on software development and licensing. Section 6 con-
cludes the work and gives an insight into planned future development of the
proposed ADRC Toolbox.

Notation. Throughout this paper, \( \mathbb{R} \) is used as the set of real num-
ers, and \( \mathbb{Z} \) as the set of integers. The \( i \)-th derivative of signal \( x(t) \) with
respect to time \( t \) is represented by \( x^{(i)} \triangleq \frac{d^i x(t)}{dt^i} \) for \( i \in \mathbb{Z} \), while \( \dot{x} \triangleq \frac{dx(t)}{dt} \)
and \( \ddot{x} \triangleq \frac{d^2 x(t)}{dt^2} \) are also used to denote the first and second time derivat-
ives. A standardized notation for certain vectors and matrices is used, i.e.,
\[
A_n \triangleq \begin{bmatrix} 0^{(n-1)\times1} & I^{(n-1)\times(n-1)} \\ 0 & 0^{1\times(n-1)} \end{bmatrix} \in \mathbb{R}^{n\times n}, \quad b_n \triangleq [0^{1\times(n-1)} 1]^T \in \mathbb{R}^n, \quad c_n \triangleq [1 0^{1\times(n-1)}]^T \in \mathbb{R}^n, \quad d_n \triangleq [0^{n-2} 1 0]^T \in \mathbb{R}^n,
\]
while \( 0 \) and \( I \) correspond, respectively, to zero and identity matrices of appropriate dimensions. Symbol \( \hat{x}(t) \) represents the estimate of variable \( x(t) \), \( \text{eig}_i(A) \) is the \( i \)-th eigenvalue of matrix \( A \), and \( 1(t) \) denotes the unit step function (Heaviside function). Signal \( x \) described as \( x \sim \mathcal{N}(\mu, \sigma^2) \) has a normal distribution with standard deviation \( \sigma \) and expected value \( \mu \). To keep the equations visually appealing, selected func-
tions within the article are written as \( f := f(t) \), hence direct time-dependence
is selectively omitted.
2. Overview of implemented ADRC algorithm

The general-purpose ADRC function block, developed within the ADRC Toolbox, encapsulates four core concepts and methods:

- **total disturbance**, which results from the aggregation of system uncertainties into a single, matched, observable term;
- **extended state**, which results from extending the original model of the controlled dynamics with an auxiliary state variable, representing solely the total disturbance;
- **extended state observer**, which is used to reconstruct the total disturbance (and other state variables, if needed) based on available signals and information about the controlled system;
- **active disturbance rejection**, which is the process of constantly updating the control signal with the estimated total disturbance, thus allowing online compensation of its effects (similar to feed-forward).

In general, the controller introduced within the ADRC Toolbox (in ver. 1.1.3) is designed to achieve a control error stabilization of a $n$-th order nonlinear single-input single-output (SISO) system, that can be described as

\[
\begin{align*}
    x^{(n)}(t) &= f^*(x, \dot{x}, \ldots, x^{(n-1)}, t) + b^*(x, \dot{x}, \ldots, x^{(n-1)}, t) u(t) + d^*(t), \\
    y^*(t) &= x(t) + w(t),
\end{align*}
\]

(1)

where $x \in \mathbb{R}$ is the controlled variable, $n \in \mathbb{Z}$ is the order of system dynamics (here also a relative order w.r.t. measured signal), $u \in \mathbb{R}$ is the control signal, $d^* \in \mathbb{R}$ represents the external disturbances, and $y^* \in \mathbb{R}$ is the system output perturbed by measurement noise $w \in \mathbb{R}$. Following [30], system (1) needs to satisfy following assumptions in order for the theoretical analysis (discussed later) to be valid.

**Assumption 1.** Variable $x(t)$, and its derivatives up to the $(n - 1)$-th order are defined on an arbitrary large bounded domain $\mathcal{D}_x \triangleq \{[x \ \dot{x} \ldots x^{(n-1)}]^\top \in \mathbb{R} : \|[x \ \dot{x} \ldots x^{(n-1)}]^\top\| < r_x\}$ for some $r_x > 0$. 

7
**Assumption 2.** Measurement noise \( w(t) \) belongs to a bounded set \( \mathcal{D}_w \triangleq \{ w \in \mathbb{R} : |w| < r_w \} \) for some \( r_w > 0 \).

**Assumption 3.** External disturbance \( d^*(t) \) and its derivative \( \dot{d}^*(t) \) belong, respectively, to bounded sets \( \mathcal{D}_{d^*} \triangleq \{ d^* \in \mathbb{R} : d^* < r_{d^*} \} \) and \( \mathcal{D}_{\dot{d}^*} \triangleq \{ \dot{d}^* \in \mathbb{R} : \dot{d}^* < r_{\dot{d}^*} \} \), for some \( r_{d^*}, r_{\dot{d}^*} > 0 \).

**Assumption 4.** Fields \( f^*(x, \dot{x}, \ldots, x^{n-1}, t) : \mathcal{D}_x \times \mathbb{R} \rightarrow \mathbb{R} \) and \( b^*(x, \dot{x}, \ldots, x^{n-1}, t) : \mathcal{D}_x \times \mathbb{R} \rightarrow \mathbb{R}/\{0\} \) are continuously differentiable Lipschitz functions.

The most common control task in terms of ADRC applications is for \( x(t) \) to follow a desired trajectory \( x_d(t) \in \mathbb{R} \), in other words, the control task is to stabilize a control error \( e(t) \triangleq x_d(t) - x(t) \) in zero. In the case of systems affected by a possibly nonlinear external disturbances and measurement noise, one can usually achieve practical stabilization, i.e. one can assure that

\[
\forall t \geq T |e(t)| < \epsilon,
\]

for some \( T > 0 \) and some arbitrarily small \( \epsilon > 0 \).

**Remark 1.** Please note, that the form of the control task formulated as (2) is general, and can address many conventionally considered control tasks, like trajectory tracking and set-point following.

**Remark 2.** Although (1) constitutes a class of nonlinear systems, the control algorithm within the ADRC Toolbox is also applicable to its simplified forms, e.g. the group of n-th order SISO linear systems.

Since the control task (2) is formulated in control error-domain, it will be more convenient to design the control system for dynamics (1) also represented in the error-domain form, i.e.,

\[
\begin{aligned}
e^{(n)}(t) &= x^{(n)}_d - f^* \left( x_d - e, \dot{x}_d - \dot{e}, \ldots, x^{(n-1)}_d - e^{(n-1)}, t \right) \\
&\quad - b^* \left( x_d - e, \dot{x}_d - \dot{e}, \ldots, x^{(n-1)}_d - e^{(n-1)}, t \right) u(t) - d^*(t), \\
y(t) &= e(t) - w(t),
\end{aligned}
\]

where \( y \triangleq x_d - y^* \in \mathbb{R} \) is the output of the control error-domain dynamics.
Assumption 5. Reference $x_d(t)$, and its derivatives up to the $(n-1)$-th order, are defined on an arbitrary large bounded domain $D_x = \{ [x_\dot{x}_d \ldots x_d^{(n-1)}]^{\top} \in \mathbb{R} : \| [x_\dot{x}_d \ldots x_d^{(n-1)}]^{\top} \| < r_{x_d} \}$ for some $r_{x_d} > 0$.

Remark 3. Assumptions 4 and 5 can be relaxed upon results shown in [31] and example from [32], where the reference signal $x_d(t)$ and fields $f^*(\cdot), b^*(\cdot)$ are allowed to have a limited number of discontinuities with bounded left- and right-hand side limits.

Remark 4. In the specific context of ADRC, the synthesis of control system expressed in error-domain has a certain beneficial characteristic, namely consecutive time-derivatives of $x_d(t)$ do not necessarily need to be known in advance. Details can be found in [33, 34].

Now, a state vector $e = [e_1 e_2 \ldots e_n]^{\top} \triangleq [e \ ˙e \ldots e^{(n-1)}]^{\top} \in \mathbb{R}^n$ can be defined for the control error-domain system (3), allowing to write down its dynamics in a following canonical observable form

$$
\begin{align*}
\dot{e}(t) &= A_e e(t) + b_n [f(e, t) + b(e, t)u(t) − d^*(t)], \\
y(t) &= c_e^{\top} e(t) − w(t),
\end{align*}
$$

(4)

where $x_d = [x_d \ ˙x_d \ldots x_d^{(n-1)}]^{\top} \in \mathbb{R}^n$, $f(e, t) := −f^*(x_d − e, ˙x_d − ˙e, \ldots, x_d^{(n-1)} − e^{(n-1)}, t)$, and $b(e, t) := −b^*(x_d − e, ˙x_d − ˙e, \ldots, x_d^{(n-1)} − e^{(n-1)}, t)$. The general concept of robust control is usually connected with the robustness against structural and parametric uncertainties of the controlled system. The most conventional formulation of the minimal amount of information required for the application of ADRC can be thus formulated as a following assumption.

Assumption 6. The structure and parameters of system (4) are highly uncertain up to a point where the dynamics order $n$ and a rough estimate $\hat{b}(\hat{e}, t) \approx b(e, t)$ are the only available information concerning the controlled system.

Remark 5. The range of admissible ratio of $b(\cdot)/\hat{b}(\cdot)$, that needs to be satisfied to keep the ADRC-based system stable, has been studied in detail in [35] and
[36], and is at least equal to
\[
\forall t \geq 0 \frac{b(e, t)}{\hat{b}(e, t)} \in \left( 0, 2 + \frac{2}{n} \right), \quad \text{for } \hat{b}(\hat{e}, t) \neq 0.
\] (5)

In the next step, following Assumption 6, system (4) can be rewritten into
\[
\begin{aligned}
\dot{e}(t) &= A_n e(t) + b_n \left[ \dot{b}(\dot{e}, t) u(t) + d(e, \dot{e}, u, t) \right], \\
y(t) &= c_n^\top e(t) - w(t),
\end{aligned}
\] (6)

where
\[
d(e, \dot{e}, u, t) = -d^*(t) + f(e, t) + \left[ b(e, t) - \hat{b}(\dot{e}, t) \right] u(t),
\] (7)
is the lumped total disturbance, aggregating the unknown components affecting the system dynamics.

To design the ADRC controller, an extended state \( z = [e^\top \ d]^\top \in \mathbb{R}^{n+1} \) needs to be defined first, which dynamics, upon (6), can be written as
\[
\begin{aligned}
\dot{z}(t) &= A_{n+1} z(t) + b_{n+1} \dot{d}(z, \dot{z}, u, t) + d_{n+1} \dot{b}(\dot{e}, t) u(t), \\
y(t) &= c_{n+1}^\top z(t) - w(t).
\end{aligned}
\] (8)

**Remark 6.** Upon Assumptions 1-6, it can be claimed that the derivative of the total disturbance is bounded, and \( \dot{d} \in D_{\dot{d}} = \{ \dot{d} \in \mathbb{R} : |\dot{d}| < r_{\dot{d}} \} \) for some \( r_{\dot{d}} > 0 \).

Since equation (6), the estimates of vector \( \hat{z} = [\hat{e}^\top \ \dot{d}]^\top \) (or its components) have been used. These deliberate manipulations of introducing estimated signals into the system dynamics were indirectly caused by the form of system output, that allows to measure only the first component of vector \( z \) (see (8)), and thus, the rest of the extended state components have to be estimated using an extended state observer (ESO), that here takes the form
\[
\dot{\hat{z}}(t) = A_{n+1} \hat{z}(t) + d_{n+1} \dot{\hat{d}}(\hat{z}, \dot{\hat{z}}, u, t) + l_{n+1} \left[ y(t) - c_{n+1}^\top \hat{z}(t) \right],
\] (9)
where the observer gain vector \( l_{n+1} = [l_1 \ l_2 \ \ldots \ \ l_{n+1}]^\top \in \mathbb{R}^{n+1} \) is bandwidth-parametrized (see [10]) with a single parameter \( \omega_o > 0 \) in a way that
\[
l_i \triangleq \frac{(n + 1)!}{i!(n + 1 - i)!} \omega_o^i, \quad \text{for } i \in \{1, \ldots, n+1\}.
\] (10)
Now, knowing how to calculate the values of $\hat{z}$, one can introduce a robust active disturbance rejection controller in the form of

$$u(t) \triangleq \frac{1}{b(e, t)} \left[ -k_n \hat{e}(t) - \hat{d}(t) \right],$$

(11)

where $k_n = [k_1 \ k_2 \ldots \ k_n] \in \mathbb{R}^{1\times n}$. All of the controller gains are bandwidth-parametrized with a single parameter $\omega_c > 0$, such that

$$k_i \triangleq n! \omega_i^\top \omega_c, \quad \text{for } i \in \{1, \ldots, n\}.$$  

(12)

Remark 7. The presence of control signal $u(t)$ in the total disturbance $d(\cdot)$, visible in (7), does not cause instability of the system as long as condition (5) is satisfied.

After the application of (9) and (11) into (8) and (6), the subsystems describing the dynamics of the observation error $\tilde{z}(t) \triangleq z(t) - \hat{z}(t)$ and the closed-loop control error $\hat{e}(t)$ can be written down, respectively, as

$$\begin{align*}
\dot{\tilde{z}}(t) &= (A_{n+1} - l_{n+1}c_{n+1}) \tilde{z}(t) + b_{n+1} \dot{d}(t) - l_{n+1} w(t), \\
\dot{\hat{e}}(t) &= (A_n - b_n k_n) e(t) + [k_n 1] \hat{z}(t).
\end{align*}$$

(13)

The theoretical analysis of dynamical systems similar to (13) have been studied in [37], [32], and [38], and can be summarized with following lemmas.

Lemma 1. For $\omega_0 > 1$, and under Assumptions 1-6, the observation error subsystem in (13), parametrized by (10), is input-to-state stable w.r.t. inputs $\dot{d}$ and $w$, and locally satisfies

$$\|\tilde{z}(t)\| \leq \omega_0 c_1 \|\tilde{z}(0)\| \exp \left( -c_2 \omega_0 t + \frac{1}{\omega_0} c_3 \left( r_d + \omega_0^{n+1} c_4 r_w \right) \right),$$

(14)

for some constants $c_1, c_2, c_3, c_4 > 0$.

Lemma 2. For $\omega_c > 1$, and under Assumptions 1-6, the control error subsystem in (13), parametrized by (12), is input-to-state stable, and locally satisfies

$$\|e(t)\| \leq \omega_c^{n-1} c_5 \|e(0)\| \exp \left( -c_6 \omega_c t + \omega_c^{n-2} c_7 \sup_{t \geq 0} \|\tilde{z}(t)\| \right),$$

(15)

for some $c_5, c_6, c_7 > 0$. 

11
3. Functionalities and parameters of the ADRC Toolbox

The introduced ADRC Toolbox for MATLAB/Simulink contains a maximally simplistic, user-friendly, and general-purpose function block, allowing the engineers to effectively use the concept of ADRC but with low requirement about the knowledge of the method itself. The goal is to simplify the practical implementation of the architecture presented in Sect. 2. The prepared drag-and-drop ADRC function block is a single input, single output block, where the input signal is the control error \( e \) and the output signal is the control signal \( u \). In MATLAB/Simulink environment, the developed block is seen as shown in Fig. 1 and it comes with a GUI presented in Fig. 2.

In the current version of the ADRC function block (ver. 1.1.3), four mandatory parameters need to be selected:

- order of system dynamics \( n \) – see (1);
- estimate of input gain parameter \( \hat{b}(\cdot) \) – see (6);
- observer bandwidth \( \omega_o \) – see (9) and (10);
- controller bandwidth \( \omega_c \) – see (11) and (12).

Additionally, the ADRC function block allows to enable two practically important functionalities, namely control signal saturation and anti-peaking mechan-
Figure 2: The graphical user interface (GUI) for the proposed ADRC function block.
ism. A thorough description of the aforementioned parameters and functionalities, together with the heuristics and interpretations concerning tuning are given in the following subsections.

3.1. Order of system dynamics $n$

The dynamics order $n$ determines the dimensions of the observer state vector $\hat{z}$, and thus influences the number of observer equations. In practice, it is sometimes possible to achieve satisfying control precision even with the ADRC controller with inappropriately set value $n$ (especially while following slowly varying trajectories), however, it is not recommended as it can easily lead to system instability when changing other parameters. Hence, order $n$ should reflect the user’s best knowledge about the relative order of the controlled system dynamics.

From a theoretical point of view, according to Lemmas 1 and 2, the higher the order $n$, the more fragile control structure is w.r.t. measurement noise ($r_w$ multiplied by $\omega_n^{n+1}$), and the transient stage of the estimation error reaches larger values caused by observer peaking (first component of (14) multiplied by $\omega_n^{n}$). The order of system dynamics also determines the range of acceptable input gain assessment $\hat{b}(\cdot)/\hat{\beta}(\cdot)$, discussed in Remark 5. In case of unknown system order, a good starting point is setting $n$ to 1 or 2, as it represents a lot of real physical systems with acceptable level of approximation [39].

3.2. Estimate of input gain $\hat{b}(\cdot)$

The approximation range of $b(\cdot)$ is reasonably wide (see Remark 5), implicating a certain level of ADRC robustness w.r.t. parametric uncertainty of $b(\cdot)$. It is, however, recommended to be as precise as possible when setting the value $\hat{b}(\cdot)$. Not only it can lead to the system instability when set outside the aforementioned acceptable range, but also can cause the total disturbance to be harder to estimate, since the difference between the real input gain parameter $b(\cdot)$ and its assumed value $\hat{b}(\cdot)$ is incorporated within the total disturbance, represented by (7). Furthermore, when the value $\hat{b}(\cdot)$ approaches the boundaries of
it may cause some unwanted oscillations in the system behavior. The input gain parameter $\hat{b}(\cdot)$ is also a scaling factor of the control signal (11), so it also has an impact on the system performance when the admissible range of control signals is limited by a relatively narrow saturation set.

3.3. Observer bandwidth $\omega_o$

With the chosen bandwidth-parametrization, described with (10), a single parameter $\omega_o$ determines the values of all observer gains. Such approach places each pole of the observation error dynamics state matrix (see (13)) as

$$\text{eig}_i \left( A_{n+1} - l_{n+1} e_{n+1}^T \right) = -\omega_o, \quad \text{for } i \in \{1, ..., n+1\}. \quad (16)$$

Besides its simplistic form, such approach results in a relatively good performance, even comparing to the situations when the eigenvalues are selected separately (see [37]).

According to the theoretical results from Lemma 1, relatively higher values of observer bandwidth have several effects on the system performance:

- estimation error of the extended state is less affected by the derivative of total disturbance influencing dynamics (8), since the component connected with $\dot{d}$ is multiplied by $1/\omega_o$;
- performance of the estimation system w.r.t. the measurement noise is highly influenced, since the impact of noise $w$ is multiplied by $\omega_o$;
- peaking phenomenon reaches higher values due to the multiplier of the first component in (14) equal to $\omega_o$, but lasts shorter, since the speed of exponential decay connected with the peaking phenomenon is proportional to $\omega_o$.

Here, it is worth mentioning that the observer bandwidth should be significantly larger than the controller bandwidth (i.e. $\omega_o \gg \omega_c$), since one of the controller goals is to compensate the total disturbance $d(\cdot)$ which, if done properly, needs to be estimated in a unquestionably quicker manner. In [10], through
simulation and experimentation, a heuristic was found that setting \( \omega_o = 5 \sim 10 \omega_c \) is a reasonable compromise between the convergence speed and the noise sensitivity in many control scenarios, hence a good starting point for further fine tuning. Moreover, as seen in Lemma 2, the precision of estimation (highly correlated with the value of \( \omega_o \)) determines the magnitude of perturbation affecting the control error upper and lower bounds.

### 3.4. Controller bandwidth \( \omega_c \)

In the process of controller tuning (12), the same bandwidth-parametrization concept is utilized as in the case of the observer, but this time by placing the eigenvalues of the state matrix in closed-loop control error subsystem (13), i.e.,

\[
\text{eig}_i (A_n - b_nk_n) = -\omega_c, \quad \text{for} \ i \in \{1, \ldots, n\}. \tag{17}
\]

The increase of parameter \( \omega_c \) causes a faster convergence of the control error from the initial conditions \( e(0) \), but also causes possibly higher overshoot, since the first component of (15) is multiplied by \( \omega_c^{-1} \). Looking at the second component of (15), higher values of \( \omega_c \) may also have, depending on the dynamics order \( n \), negative impact on the influence of estimation errors on the control performance, and thus one ought to guarantee a fast and precise extended state estimation by selecting \( \omega_o \gg \omega_c \) to reduce the impact of \( \| \hat{z}(t) \| \) on \( \| e(t) \| \).

It is also worth noting, that with the higher values of \( \omega_c \), the transients of \( d \) are faster, making them harder to estimate by the ESO. Here again, a good starting point for controller bandwidth selection is the heuristic \( \omega_o = 5 \sim 10 \omega_c \), discussed in [10].

### 3.5. Control signal saturation

Enabling the control signal saturation function in the provided ADRC function block, when one knows the physical limitations of the controlled plant may have enormous impact on the control performance. When this functionality is disabled (default setting), the observer is treating unsaturated control \( u \) as the signal affecting the system dynamics, while in reality, the dynamic system may
be only subject to its saturated value. Such situation may result in a virtual
change of $b(\cdot)$ from the observer perspective, and cause aggressive and unpre-
dictable transients. In the worst case, it can lead to control system instability
due to virtual leave of the range discussed in Remark 5. Depending on the se-
lected saturation function status in the ADRC function block, the control signal
takes the form

$$u(t) = \begin{cases} 
  u^*(t), & \text{when saturation OFF (default)}; \\
  u_{\min}, & \text{for } u^*(t) < u_{\min}; \\
  u^*(t), & \text{for } u_{\min} \leq u^*(t) \leq u_{\max}, \text{ when saturation ON}, \\
  u_{\max}, & \text{for } u^*(t) > u_{\max},
\end{cases}$$

(18)

where auxiliary variable $u^*(t) \triangleq \frac{1}{b(\hat{e}(t))} \left[-k_n \hat{e}(t) - \hat{d}(t)\right]$ is the originally defined control signal resulting from (11).

**Remark 8.** One could rightfully point out that the introduction of the saturation function (18) into the ADRC algorithm, derived in its nominal form in Sect. 2, is not consistent with Lemmas 1 and 2 (both considering saturation-less case). And although true, the presence of saturation function in disturbance observer-based control designs (especially in the context of stability) has been already been addressed in the literature (e.g. [40, 41]). Additionally, the saturation of control signal is most frequently met in the initial transient stage of the control experiments, after which the control signal does not reach the saturation and acts according to (11), which is in line with Lemmas 1 and 2.

### 3.6. Anti-peaking

The existence of peaking phenomenon in the operation of high gain observ-
ers is a well known issue that can deteriorate the control system performance
especially at the beginning of the observation process. Although many methods
can be used within the observer structure to address this issue (e.g. [42, 43]),
here a method is utilized that deals with the peaking by temporarily disabling
the control action at the beginning of the control process (meaning the control
The specific anti-peaking logic implemented in the ADRC function block affects the control signal in a following manner:

\[
u(t) = \begin{cases} 
  u^*(t), & \text{when anti-peaking OFF (default);} \\
  0 & \text{for } 0 \leq t \leq T_d; \\
  u^*(t) & \text{for } t > T_d;
\end{cases}
\]  

(19)

where \(T_d > 0\) is the time that needs to pass before applying control signal, and \(u^*(t)\) is the auxiliary variable representing the control signal after going through the saturation logic defined in (18).

**Remark 9.** Here, similarly to Remark (8) discussing the control signal saturation, the use of anti-peaking mechanism (19) in the considered ADRC algorithm makes it inconsistent with Lemmas 1 and 2 (both derived for the case without anti-peaking). The theoretical ramifications of this have been studied before, for example in [43, 42], and it was shown that, under certain conditions, relatively easy to satisfy in practice, the stability of the system can be retained.

### 3.7. Overview

A simplified diagram showing the insights of the introduced ADRC function block is shown in Fig. 3 and a scope of its current functionalities is summarized in Table 1. Depending on the function block configuration, it can be seen that \(\hat{b}\) can be chosen either constant (realized internally through the provided GUI - Fig. 2) or supplied externally as a time-dependent value from outside of the function block, which is illustrated with switching mechanisms in Fig. 3. The default configuration in the ADRC function block is constant \(\hat{b}\). One can change its source in the GUI to external and then provide (in each sample time) information about varying value of \(\hat{b}\). A variety of techniques can be used to on-line calculate \(\hat{b}(t)\), for example, a genetic algorithm [44] or extremum seeking [45]. Regardless of the chosen type of \(\hat{b}\) in the function block, its value
has to satisfy (5). In the cases of $\omega_c$ and $\omega_o$, similar approach is applied. By default, these values are to be set constant in the ADRC function block, or alternatively provided as external time-dependent values. In the latter case, their specific values can be calculated, for example, using a neural network [37], genetic algorithm [46], nonlinear function [9], or time-varying [47] function.

To summarize the above, Fig. 4 shows design steps for deploying the proposed ADRC function block.

4. Validation

In order to verify the efficacy of the proposed ADRC Toolbox with its ADRC function block, a set of tests is conducted in Examples #1 through #5. The tests, described in Table 2, utilize a group of systems from various control areas that differ from each other in terms of order, type, and structure of the
1. Identify type of controlled system
   - SISO?
   - MIMO?

2. Drag-and-drop the ADRC function block(s) from the ADRC Toolbox to the MATLAB/Simulink program

3. Connect control error(s) to the input of the ADRC block(s) and connect the output of the ADRC block(s) to the input(s) of the controlled system

4. Select relative order of the controlled system ($n$)
   - known?
   - unknown?
   - use directly
   - approximate, identify

5. Select the input gain of the controlled system ($b$)
   - known?
   - unknown?
   - use directly
   - approximate, identify

   set „Source: Internal”
   - constant?
   - variable?

   set „Source: External” and provide value of $b$ as a new input to ADRC block

6. Select the ADRC tuning parameters ($\omega_c$ and $\omega_o$)
   - known?
   - unknown?
   - use directly
   - approximate, identify

   set „Source: Internal”
   - constant?

   set „Source: External” and provide values of $\omega_c$ and $\omega_o$ as new inputs to ADRC block

7. Saturation
   - yes?
     - tick box and set $u_{\text{min}}$, $u_{\text{max}}$
   - no?
     - tick off box

8. Anti-peaking
   - yes?
     - tick box and set time $T_d$
   - no?
     - tick off box

9. Run the program. Check performance. If not satisfactory, then reevaluate system parameters (4,5), tuning gains (6), and/or functionalities (7,8)

---

* $m$ is the number of degrees of freedom of the controlled system
** such approach is not viable to all MIMO systems (e.g. underactuated systems)

Figure 4: Design steps for deploying the proposed ADRC function block.
Table 1: Overview of parameters and functionalities of the ADRC function block.

| Parameter/Functionality                                      | Symbol | Selection | Variations                  | Admissible value             |
|--------------------------------------------------------------|--------|-----------|------------------------------|------------------------------|
| Order of system dynamics (Sect. 3.1)                        | $n$    | Mandatory | Constant                     | $n \in \mathbb{Z}$           |
| Estimate of input gain (Sect. 3.2)                          | $\hat{b}$ | Mandatory | Constant / Time-dependent    | $\hat{b} \in \mathbb{R}/\{0\}$ |
| Observer bandwidth (Sect. 3.3)                               | $\omega_0$ | Mandatory | Constant / Time-dependent    | $\omega_0 \gg \omega_i$      |
| Controller bandwidth (Sect. 3.4)                            | $\omega_c$ | Mandatory | Constant / Time-dependent    | $\omega_c > 0$               |
| Control signal saturation (Sect. 3.5)                       | n/a    | Optional  | Upper limit, Lower limit     | $u_{\text{min}} < u_{\text{max}}$ |
| Anti-peaking (Sect. 3.6)                                     | n/a    | Optional  | Constant                     | $T_d > 0$                    |

Table 2: Methodology of ADRC Toolbox validation.

| Example no. (Sect.) | Plant            | Criterion |
|---------------------|------------------|-----------|
| #1 (Sect. 4.1)      | Generic plant    | Order ($n$) | Type | Structure | Ext. disturb. (d*) | Control area | Validation |
| #2 (Sect. 4.2)      | Coupled tanks    | 4         | Linear | SISO     | Step             | n/a         | Simulation |
| #3 (Sect. 4.3)      | Power converter  | 2         | Nonlinear | MIMO     | Step             | Process     | Simulation |
| #4 (Sect. 4.4)      | DC motor         | 2         | Linear | SISO     | None             | Motion      | Experiment |
| #5 (Sect. 4.5)      | Heaters          | 1         | Nonlinear | MIMO     | None             | Process     | Experiment |

controlled dynamics as well as type of acting external disturbance. Some of the control examples are performed in simulation, other are realized in hardware. As opposed to just simulation studies, physical benchmarks additionally consider real process characteristics such as discrete sampling intervals, communication overhead with the process, requirement to meet a cycle time, and mathematical modelling mismatch. Such diversified group is purposefully selected to test the ADRC Toolbox in various control scenarios. Moreover, the experimental part of the validation is conducted using only hardware components that are relatively cheap and easily accessible. Such choice of testing the proposed ADRC Toolbox exclusively using commercial off-the-shelf hardware systems is deliberate as it allows the Toolbox users to straightforwardly reproduce the results shown later in this paper. The tuning methodology in all the upcoming examples is based on guidelines from Sect. 3.
4.1. Example #1 - generic plant control

The first considered system is expressed in form of (1) as

\[
\begin{align*}
    x^{(4)}(t) &= f^\ast(x, \dot{x}, \ddot{x}, x^{(3)}, t) + b^\ast u(t) + d^\ast(t), \\
y^\ast(t) &= x(t) + w(t),
\end{align*}
\]

(20)

where \(f^\ast(x, \dot{x}, \ddot{x}, x^{(3)}, t) = -4x^{(3)} - 6\ddot{x} - 4\dot{x} - x\), \(b^\ast = 1\), and \(d^\ast(t) = 70 \cdot 1(t - 10)\).

The control objective here is to generate control signal \(u\) in a way output signal \(x\) tracks a desired trajectory \(x_d\), despite the presence of external disturbance and parametric uncertainty of the model. Additionally, two scenarios are considered in this study, namely without sensor noise \((w(t) \equiv 0)\) and with sensor noise \(w(t) \sim \mathcal{N}(0, 10^{-12})\). The ADRC function block (from the proposed ADRC Toolbox) is implemented with design parameters \(n = 4\), \(\hat{b} = 0.8\), \(\omega_c = 5\), \(\omega_o = 50\), and with turned off anti-peaking functionality (default setting). Two additional scenarios are considered, namely with the saturation turned off (default setting) and with the saturation turned on with lower and upper limits \(u_{\text{min}} = -100\) and \(u_{\text{max}} = 100\). Having (20) expressed in form of (1) as well as ADRC parameters and functionalities configured, the remaining equations of the ADRC algorithm (3)-(13) can be now straightforwardly derived for the above system, which is omitted here to avoid redundancy.

The block diagram of the implemented ADRC-based system in MATLAB/Simulink is seen in Fig. 5, in which \(G(s) = 1/(s^4 + 4s^3 + 6s^2 + 4s + 1)\). The results of using the proposed ADRC Toolbox to the above control problem are seen in Fig. 6. From the obtained results, one can conclude that the applied ADRC function block managed to realize the given control objective in a satisfactory manner for all tested scenarios. The implemented robust controller drove the control error to the vicinity of zero as well as recovered the performance after the appearance of the external disturbance at \(t = 10s\). One can also see the influences of the saturation mechanism and the measurement noise. In the case of the former, the control error convergences faster when the saturation is off but for the price of initial oscillatory behavior of the output and control signal peaking, reaching the magnitude of \(u \approx 10^6\) in the initial transient, which
probably would be not feasible for most real-world controllers. In the case of the latter, due to the particular structure of the observer and the controller (both utilizing the output signal), the noise is clearly manifested in the control signal. At the same time, for the considered system, the noise in the control signal is mostly filtered by the plant dynamics and does not have a significant impact on the controlled variable.

4.2. Example #2 - coupled tanks level control

This example considers a hydraulic system, which consists of two tanks connected by a flow channel and two independent inlet flows, each for one tank. The levels of fluids in the tanks are considered as the system outputs, hence the resultant multi-input multi-output (MIMO) system model can expressed as

\[
\begin{aligned}
\frac{dh_1(t)}{dt} &= -\frac{a}{c} \sqrt{2gh_1(t)} - \frac{a}{c} \text{sign}(\epsilon_h(t)) \sqrt{2g|\epsilon_h(t)|} + \frac{1}{c} u_1(t) + d_1^*(t), \\
y_1^*(t) &= h_1(t) + w_1(t), \\
\frac{dh_2(t)}{dt} &= -\frac{a}{c} \sqrt{2gh_2(t)} + \frac{a}{c} \text{sign}(\epsilon_h(t)) \sqrt{2g|\epsilon_h(t)|} + \frac{1}{c} u_2(t) + d_2^*(t), \\
y_2^*(t) &= h_2(t) + w_2(t),
\end{aligned}
\] (21)

where \(u_1 \text{[m}^3/\text{s]}\) and \(u_2 \text{[m}^3/\text{s]}\) are the inlet flows (control signals) for the first and second tank, respectively, \(y_1^* \text{[m]}\) and \(y_2^* \text{[m]}\) are the measured system outputs that consist of, respectively, first and the second tank fluid levels \(h_1 \text{[m]}\) and \(h_2 \text{[m]}\) and corresponding sensor noises \(w_1 \text{[m]}\) and \(w_2 \text{[m]}\), auxiliary variable \(\epsilon_h := h_1 - h_2\) represents the fluid levels difference, \(g \text{[m/s}^2\)] is the gravitational acceleration,
Figure 6: Simulation results of Example #1.

$a [m^2]$ is the cross-section area of the pipes, and $c [m^2]$ is the cross-section area of the tanks. It is assumed that the system model (21) has same parameters for both tanks and pipelines. The parameters of the system model used in the simulation are $c = 1.2 \cdot 10^{-2} m^2$, $a = 7.5 \cdot 10^{-5} m^2$, and $g = 9.81 m/s^2$. Referring to the generalized form of system description from (1), one can assign $x(t) := h_1(t)$, $f^*(x,t) = -\frac{a}{c} \sqrt{2gx(t)} - \frac{a}{c} \text{sign}(\epsilon_h(t))\sqrt{2g|\epsilon_h(t)|}$, and $b^* = \frac{1}{c}$ for the first tank dynamics and $x(t) := h_2(t)$, $f^*(x,t) = -\frac{a}{c} \sqrt{2gx(t)} + \frac{a}{c} \text{sign}(\epsilon_h(t))\sqrt{2g|\epsilon_h(t)|}$, and $b^* = \frac{1}{c}$ for the second tank dynamics.

The control objective here is to manipulate two inlet flows $u_{1/2}$ to make the output levels $h_{1/2}$ track respective reference set-points $x_{d1/d2}[m]$. The control process should be carried out despite external disturbances $d_1^*(t) = -1.4 \cdot 10^{-4} \cdot \mathbb{1}(t-90)$ and $d_2^*(t) = -0.7 \cdot 10^{-4} \cdot \mathbb{1}(t-130)$, parametric uncertainty of the model, and Gaussian sensor noises $w_{1/2} \sim N(0, 0.8 \cdot 10^{-10})$. It is worth noting that, in this example, the tanks are treated as two independent systems, therefore two independent ADRC function blocks are used, each for governing one input-to-output channel of the MIMO system. In such configuration, the influence of
the cross-couplings is treated as part of the system total disturbance.

In this example, the two ADRC blocks are implemented with design parameters $n = 2$, $\hat{b} = 0.8$, $\omega_c = 0.3$, $\omega_o = 3$ for the first tank and $n = 2$, $\hat{b} = 1$, $\omega_c = 0.12$, $\omega_o = 1.2$ for the second tank. The control signal saturation is turned on with limits $u_{\text{min}} = 0$ and $u_{\text{max}} = 4 \cdot 10^{-4}$ (where the latter value corresponds to 1440 l/h) and the anti-peaking function is turned off (default setting). The block diagram of the implemented ADRC-based system in MATLAB/Simulink is seen in Fig. 7 and the results of using the ADRC Toolbox to the above control problem are seen in Fig. 8.

From this figure, one can notice that the ADRC scheme successfully realize the control task in both control channels of the considered multidimensional system. After the transient stages, both tanks levels are kept at desired predefined values and the implemented ADRC controllers manage to quickly and accurately compensate the influence of external disturbances before it has a
visible effect on the process variables. The impact of $d_1^*$ and $d_2^*$ is, however, clearly seen in the control signals $u_{1/2}$ at $t = 90s$ and $t = 130s$, when they were abruptly forced by the control structure to generate more energy to handle the disturbances in real-time. Interestingly, the ADRC block parameters for governing the first tank dynamics have higher values of tuning parameters $\omega_c$, $\omega_o$ and lower value of estimated input gain $\hat{b}$, when compared to those for the second tank. This results in a faster convergence of the control error for the first tank but also in larger noise amplification in the control signal, latter of which could be potentially problematic in real-world implementation. These observations directly correspond to the theoretical findings described for $\hat{b}$, $\omega_o$, and $\omega_c$ in Sects. 3.2, 3.3, and 3.4, respectively.

4.3. Example #3 - power converter voltage control

The system here is a DC-DC buck converter, which is a common component in power electronics. Following [32, 48], its average second order dynamic model can be formulated as

\[
\begin{align*}
\frac{d^2 v_o(t)}{dt^2} &= -\frac{1}{CR} \frac{dv_o(t)}{dt} - \frac{1}{CL} v_o(t) + \frac{V_{in}}{CL} u(t) + d^* (R_L(t), t), \\
y^*(t) &= v_o(t) + w(t),
\end{align*}
\]

(22)

where $u \in [0, 1]$ is the duty ratio (control signal), $y^*[V]$ is the measured system output that consists of the average capacitor voltage $v_o[V]$ and the sensor noise $w[V]$, $R[\Omega]$ is the load resistance of the circuit, $L[H]$ is the filter inductance, $C[F]$ is the filter capacitance, $V_{in}[V]$ is the input voltage source, and the external disturbance $d^*$ is dependent on time-varying load resistance $R_L[\Omega]$. The parameters of the power converter used in the simulation are taken from a real system utilized in [32, 48] and are $V_{in} = 20V$, $L = 0.01H$, $C = 0.001F$, and $R = 500\Omega$. Referring to the generalized form of system description from (1), one can assign $x(t) := v_o(t)$, $f^*(x, \dot{x}, t) = -\frac{1}{CR} \dot{x}(t) - \frac{1}{CL} x(t)$, and $b^* = \frac{V_{in}}{CL}$.

The control objective here is to generate control signal $u$ that will make $v_o$ track a reference capacitor output voltage trajectory $x_d[V]$ despite the acting unknown external disturbance related to load resistance $R_L = 85\sin(40\pi t) + 100,$
the presence of sensor noise \( w \sim \mathcal{N}(0, 5 \cdot 10^{-5}) \), and parametric uncertainty of the internal model dynamics.

In this example, the ADRC function block is implemented with design parameters \( n = 2, \hat{b} = 2 \cdot 10^6, \omega_c = 500, \omega_o = 3000 \), turned off anti-peaking mechanism (default setting), and turned on saturation of the control signal with \( u_{\text{min}} = 0 \) and \( u_{\text{max}} = 1 \). A block diagram of the implemented ADRC-based system in MATLAB/Simulink is seen in Fig. 9 and the results of using the proposed ADRC Toolbox to the above control problem are seen in Fig. 10. The obtained results show that the applied ADRC function block can realize the given control objective by providing satisfactory results for output voltage tracking. Although the influence of harmonic load resistance \( R_L \) has a visible effect on the shape of the control signal and the resultant output voltage, manifested through instances of abrupt spikes of \( \pm 0.2V \), the average value of the control error remained within a practically acceptable range. This example demonstrates that, although ADRC is a powerful robust control scheme, it has limitations. The challenging form of the external disturbance, on one hand, invites the ADRC user to increase the observer and controller bandwidths to achieve better disturbance rejection and reference tracking, but on the other precludes from obtaining high control performance due to the plant restrictions (e.g. finite actuation capabilities, sampling time) and ADRC function block limitations (e.g. particular structure of the implemented observer - see \([48, 34]\)). Hence, a compromise between the two ought to be found in engineering practice.
4.4. Example #4 - motor velocity control

In this example, the controlled system is a Pololu 3240 gearbox\(^4\), which consists of a 12V DC motor, 34:1 gearbox, and 48 CPR (counts per revolution) quadrature encoder integrated on the motor shaft. The system is governed through an Arduino MKR WiFi 1010 board with a dedicated Arduino MKR Motor Carrier with sampling frequency of 100Hz. The complete experimental setup can be seen in Fig. 11. Such low cost, DIY-style configuration allows for a rapid prototyping of various motion control applications.

A simplified mathematical model describing the gearmotor dynamics can be formulated as a combination of the motor electrical and mechanical parts as

\[
\begin{align*}
\frac{d^2\omega(t)}{dt^2} &= -\frac{R_a J + L_a b_s}{L_a J} \frac{d\omega(t)}{dt} - \frac{R_a b_s + k_s^2}{L_a J} \omega(t) + \frac{k_s}{L_a} u(t) + d^*(t), \\
y^*(t) &= \omega(t) + w(t),
\end{align*}
\]

(23)

where \(u[V]\) is the source voltage (control signal), \(d^*\) is the external disturbance, \(y^*[\text{rad/s}]\) is the measured system output that consists of the motor shaft angular velocity \(\omega[\text{rad/s}]\) and the sensor noise \(w[\text{rad/s}]\), \(R_a[\Omega]\) is the armature resistance,

\(^4\)https://www.pololu.com/product/3240 (last visit: 20.09.2021)
$L_a[H]$ is the armature inductance, $J[kg\cdot m^2]$ is the rotor inertia, $b_f[Nm/rad]$ is the friction coefficient, $k_\phi[Vs/rad]$ is the constant design parameter. Referring to the generalized form of system description from (1), one can assign $x(t) := \omega(t)$, $f^*(x, \dot{x}, t) = -\frac{R_a J + L_a b_f}{L_a J} \frac{d\omega(t)}{dt} - \frac{R_a b_f + k_\phi^2}{L_a J} \omega(t)$, and $b^* = \frac{k_\phi}{L_a J}$.

The control objective here is to apply control input $u$ that will make $\omega$ track a reference motor shaft angular velocity $x_d[imp/sec]$ despite the presence of sensor noise and parametric uncertainty of the internal model dynamics.

In this example, the ADRC function block is implemented with design parameters $n = 2$, $\dot{b} = 600$, $\omega_c = 40$, $\omega_o = 90$, turned off anti-peaking mechanism (default setting), and turned on saturation of the control signal with limits $u_{min} = -100$ and $u_{max} = 100$. The block diagram of the implemented ADRC-based system in MATLAB/Simulink is seen in Fig. 12 and the results of using the proposed ADRC Toolbox to the above control problem are seen in Fig. 13. The obtained experimental results show the effectiveness of the ADRC function block in hardware application. The motor output signal tracks the desired trajectory, being a combination of an unit step and a harmonic function. Expectedly, both the control signal and the control error exhibit rapid change when the system responds to the step change. Such abrupt reaction, although kept at bay by the activated saturation mechanism, could be further minimized e.g. by adding a smoothing function, as it was done in Example #3. Here, however, the step function was left deliberately to check how the control system performs in
Figure 12: Block diagram of the DC motor control system (Example #4) with the ADRC function block from the proposed ADRC Toolbox.

Figure 13: Experimental results of Example #4.

case of an instantaneous change of target signal (which happens often in engineering practice). One can notice that the harmonic component, present in both the control signal and the control error (inherited from the reference signal), has not be completely compensated by the ADRC. Although the obtained control performance may be considered satisfactory, the implemented control structure makes the control error oscillate around ±1 rad/sec. Here again, similar to Example #3, one could point to several specialized methods that would further increase harmonic disturbance rejection. This, however, is beyond the proposed ADRC Toolbox, at least in its current form.

Remark 10. In order to run and repeat the results from this hardware example, two MATLAB add-ons need to be installed: "MATLAB Support Package for Arduino Hardware" and "Simulink Support Package for Arduino Hardware."
4.5. Example #5 - temperature control

The system considered here is the "Temperature Control Lab" (TCLab)\(^5\), which is a commercially available, portable, low-cost, Arduino-based temperature control kit [49]. The TCLab consists of two heaters and two temperature sensors. The complete utilized experimental setup can be seen in Fig. 14. In this specific example, a configuration with two operating heaters is used, which constitutes a MIMO control structure. The two heater units are placed in proximity to each other to transfer heat by convection and thermal radiation. The system is governed with sampling frequency of 10Hz.

Based on [50], such system can be described with a following nonlinear mathematical model, developed based on energy balance equations representing the dynamics between the input power to each transistor and the temperature sensed by each thermistor

\[
\begin{align*}
\frac{dT_1(t)}{dt} & = f_1^*(T_1, T_2, t) + b_1^* u_1 + d_1^*(t), \\
y_1^*(t) & = T_1(t) + w_1(t), \\
\frac{dT_2(t)}{dt} & = f_2^*(T_1, T_2, t) + b_2^* u_2 + d_2^*(t), \\
y_2^*(t) & = T_2(t) + w_2(t),
\end{align*}
\]

(24)

where \(u_1[W]\) and \(u_2[W]\) are the output energy (control signals) from the first and second heater, respectively, \(y_1[\degree C]\) and \(y_2[\degree C]\) are the measured system outputs that consist of, respectively, first and second heater temperatures \(T_1[\degree C]\) and \(T_2[\degree C]\) and corresponding sensor noises \(w_1[\degree C]\) and \(w_2[\degree C]\), \(d_1^*\) and \(d_2^*\) are the external disturbances affecting particular channels, and functions \(f_i^*(\cdot) = \frac{1}{mC_p} \left[ UA(T_\infty(t) - T_i(t)) + \epsilon \sigma A(T_\infty^4(t) - T_i^4(t)) + Q_{12}(t) \right]\) and \(b_i^* = \frac{\alpha_i}{mC_p}\) for \(i \in \{1, 2\}\). Additionally, \(C_p[J/kg-K]\) is the heat capacity, \(T_\infty[\degree C]\) is the ambient temperature, \(U[W/m^2-K]\) is the heat transfer coefficient, \(A[m^2]\) is the surface area not between heaters, \(A_s[m^2]\) is the surface area between heaters, \(\epsilon\) is the emissivity, \(m[kg]\) is the mass, \(\sigma[W/m^2-K^4]\) is the Boltzmann constant, and \(\alpha_i[W/(\%\degree C)]\)

\(^5\)http://apmonitor.com/heat.htm (last visit: 20.09.2021)
heater) is the $i$-th heater factor. The heat transfer exchange $Q_{12}$ between the heaters is a combination of convective heat transfer $Q_{C12} = UA_s(T_2 - T_1)$ and radiative heat transfer $Q_{R12} = \varepsilon\sigma A(T_4^2 - T_4^1)$, defined as $Q_{12} = Q_{C12} + Q_{R12}$. Detailed derivation of the structure and parameters of (24) using physics-based and data-driven techniques can be found in [50].

Referring to the generalized form of system description from (1), one can assign $x(t) := T_1(t)$, $f^*(x, t) = \frac{1}{mC_p} \left[ UA(T_\infty^1(t) - T_1(t)) + \varepsilon\sigma A(T_4^1(t) - T_4^1(t)) + Q_{12}(t) \right]$, and $b^* = \frac{\alpha_1}{mC_p}$ for the first heater dynamics and $x(t) := T_2(t)$, $f^*(x, t) = \frac{1}{mC_p} \left[ UA(T_\infty^2(t) - T_2(t)) + \varepsilon\sigma A(T_4^2(t) - T_4^2(t)) - Q_{12}(t) \right]$, and $b^* = \frac{\alpha_2}{mC_p}$ for the second heater dynamics.

In case of this example, the control objective is to adjust both heaters power outputs $u_1/2$ and transfer the thermal energy from the heaters to the temperature sensors to make $T_1/2$ keep desired temperature set-points $x_{d1/d2}$, despite the parametric uncertainty of the system model and sensor noise.

The block diagram of the implemented ADRC-based system in MATLAB/Simulink is seen in Fig. 15. One can notice that in the performed example heaters as treated as two independent systems. The equations (1) are therefore derived independently for each degree-of-freedom of the system (24). Consequently, two ADRC function blocks are used, each for governing one input-to-output channel of the system. In such configuration, the unmodeled influence of the cross-couplings is treated as part of the system total disturbance.

In this example, the two ADRC function blocks are implemented with design parameters $n = 1$, $\hat{b} = 3$, $\omega_c = 6$, $\omega_o = 15$ for the first heater and $n = 1$, $\hat{b} = 5$, $\omega_c = 6$, $\omega_o = 15$ for the second heater. Both function blocks have the anti-
peaking mechanisms turned off (default setting), and turned on saturation of the control signals with user-defined limits $u_{\text{min}} = 0$ and $u_{\text{max}} = 100$. The results of using the proposed ADRC Toolbox to the above control problem are seen in Fig. 16. The presented experimental results validate the control structure with multi ADRC function blocks as a viable solution for the considered cross-coupled MIMO plant. For the chosen tuning parameters, the target temperature set-points are tracked with acceptable margins of error. Without the necessity of precise system modeling, the implemented robust ADRC approach managed to deal with the plant modeling uncertainties, including the cross-couplings between the system degrees-of-freedom. By comparing the selected ADRC parameters, one can notice that smaller estimated input gain ($\hat{b}$) was selected for the first heater. In result, steeper slope of convergence to the desired values is seen for the first heater output, which also results in a signal overshoot of around 3°C. This is an expected result, which stands from the fact that $\hat{b}$ is also a scaling factor for the entire control signal (see (11) and discussion thereof in Sect. 3.2). Regarding the control effort, one can notice that the user-defined reference temperatures turned out to be challenging for the governed plant and pushed the control signals to extremes. This justifies why the saturation mechanism has been incorporated in the proposed ADRC function block and shows the importance of activating saturation if the admissible range of the control signal is known. This is especially important in hardware experiments, where the control signals usually operate within certain physically-limited bounds. It is important for the considered observer-based control algorithm, which works correctly only if it is fed with accurate information about the input-output behavior of the system (for details see Sect. 3.5).

Remark 11. In order to run and repeat the results from this hardware examples, extra MATLAB add-ons need to be installed (see Remark 10) as well as dedicated TClab libraries, available under the MIT licence at the producer website (mentioned earlier).
5. Software development and licensing

The presented ADRC algorithm (Sect. 2), toolbox parameters and functionalities (Sect. 3), and the validating tests (Sect. 4) shown in this paper are for the version of the ADRC Toolbox, which was current at the time of paper publication (ver. 1.1.3). The latest version of the ADRC Toolbox can be downloaded from https://www.mathworks.com/matlabcentral/fileexchange/102249-active-disturbance-rejection-control-adrc-toolbox and installed in MATLAB/Simulink following the instructions therein. Apart from the necessary installation file, the website also contains contact information to the ADRC Toolbox lead developers as well as a Q&A section. Alternatively, the ADRC Toolbox can be also installed as an add-on directly from MATLAB using the "Add-On Explorer" function.

Regarding software development of the ADRC Toolbox, all its related files,
including MATLAB code, data, apps, examples, and documentation, are packaged into a single installation file (.mltbx), which greatly simplifies sharing and installation of the ADRC Toolbox. This form of software development is convenient for the end users, who can install the ADRC Toolbox without being concerned with the MATLAB path or other installation details as the provided .mltbx file manages these details for the end users.

6. Conclusions

The work demonstrated the effectiveness and ease of use of the developed active disturbance rejection control (ADRC) toolbox for numerical simulations and hardware experiments performed in MATLAB/Simulink software environment. The ADRC Toolbox has been proposed to address the recent interest behind the ADRC methodology and the growing use of MATLAB/Simulink in both academia and industry. Through a variety of conducted case studies, it has been shown that the developed ADRC function block can be used for various applications and may be easily integrated within existing Simulink models. The implemented functionalities can help the users to swiftly and straightforwardly tailor-made their ADRC applications, thus potentially decrease the usually required time and effort. The expected impact is that the improvements on the implementation side of ADRC, introduced through the development of the ADRC Toolbox, will make ADRC an even more compelling choice for solving real-world control problems.

The developed ADRC Toolbox is available to end users as an open-source software. It is planned to be further developed with new functionalities based on the feedback from the control community. Importantly, the ADRC Toolbox is an open-source project, freely available to interested users. The current version of the ADRC Toolbox can be downloaded from https://www.mathworks.com/matlabcentral/fileexchange/102249-active-disturbance-rejection-control-adrc-toolbox.
Acknowledgement

This research was financially supported as a statutory work of the Poznan University of Technology, Poland (no. 0214/SBAD/0229). Additionally, Dr. Krzysztof Lakomy and Dr. Rafal Madonski are supported by the Fundamental Research Funds for the Central Universities project no. 21620335.

Conflict of interest

None declared.

References

[1] B. Ninness, A. Wills, A. Mills, UNIT: a freely available system identification toolbox, Control Engineering Practice 21 (5) (2013) 631–644. doi:10.1016/j.conengprac.2012.10.007.

[2] J. Löfberg, YALMIP: a toolbox for modeling and optimization in MATLAB, in: Proc. Computer Aided Control Systems Design International Symposium, 2004, pp. 284–289. doi:10.1109/CACSD.2004.1393890.

[3] B. Andritsch, M. Horn, S. Koch, H. Niederwieser, M. Wetzlinger, M. Reichhartinger, The robust exact differentiator toolbox revisited: filtering and discretization features, in: Proc. IEEE International Conference on Mechatronics, 2021, pp. 1–6. doi:10.1109/ICM46511.2021.9385675.

[4] H. Chang, H. Kim, G. Park, H. Shim, DO-DAT: a MATLAB toolbox for design & analysis of disturbance observer, IFAC-PapersOnLine 51 (25) (2018) 340–345. doi:10.1016/j.ifacol.2018.11.130.

[5] J. P. Coelho, T. M. Pinho, J. Boaventura-Cunha, J. B. de Oliveira, A new brain emotional learning Simulink toolbox for control systems design, in: Proc. IFAC World Congress, 2017, pp. 16579–16584. doi:10.1016/j.ifacol.2017.08.1912.
[6] A. Tepljakov, E. Petlenkov, J. Belikov, FOMCON: a MATLAB toolbox for fractional-order system identification and control, International Journal of Microelectronics and Computer Science 2 (2) (2011) 51–62.

[7] T. E. Lechekhab, S. Manojlovic, M. Stankovic, R. Madonski, S. Simic, Robust error-based active disturbance rejection control of a quadrotor, Aircraft Engineering and Aerospace Technology 93 (1) (2021) 89–104. doi:10.1108/AEAT-12-2019-0266.

[8] Z. Gao, Y. Huang, J. Han, An alternative paradigm for control system design, in: Proc. IEEE Conference on Decision and Control, Vol. 5, 2001, pp. 4578–4585. doi:10.1109/CDC.2001.980926.

[9] J. Han, From PID to Active Disturbance Rejection Control, IEEE Transactions on Industrial Electronics 56 (3) (2009) 900–906. doi:10.1109/TIE.2008.2011621.

[10] Z. Gao, Scaling and bandwidth-parameterization based controller tuning, in: Proc. American Control Conference, 2003, pp. 4989–4996. doi:10.1109/ACC.2003.1242516.

[11] Z. Gao, On the centrality of disturbance rejection in automatic control, ISA Transactions 53 (4) (2014) 850–857. doi:10.1016/j.isatra.2013.09.012.

[12] X. Zhang, X. Zhang, W. Xue, B. Xin, An overview on recent progress of extended state observers for uncertain systems: methods, theory and applications, Advanced Control for Applications (2021). doi:10.1002/adc2.89.

[13] Q. Zheng, Z. Gao, Active disturbance rejection control: some recent experimental and industrial case studies, Control Theory and Technology 16 (4) (2018) 301–313. doi:10.1007/s11768-018-8142-x.
[14] G. Herbst, Transfer function analysis and implementation of active disturbance rejection control, Control Theory and Technology 19 (1) (2021) 19–34. doi:10.1007/s11768-021-00031-5.

[15] E. Sariyildiz, R. Oboe, K. Ohnishi, Disturbance observer-based robust control and its applications: 35th anniversary overview, IEEE Transactions on Industrial Electronics 67 (3) (2020) 2042–2053. doi:10.1109/TIE.2019.2903752.

[16] W.-H. Chen, J. Yang, L. Guo, S. Li, Disturbance-observer-based control and related methods - an overview, IEEE Transactions on Industrial Electronics 63 (2) (2016) 1083–1095. doi:10.1109/TIE.2015.2478397.

[17] R. Madonski, P. Herman, Survey on methods of increasing the efficiency of extended state disturbance observers, ISA Transactions 56 (2015) 18–27. doi:10.1016/j.isatra.2014.11.008.

[18] G. Herbst, A minimum-footprint implementation of discrete-time ADRC (2021). arXiv:2104.01943.

[19] L. Wang, L. Qing, C. Tong, Y. Yin, J. Li, S. Song, Active disturbance rejection control simulation toolbox in open-source software Scilab/Xcos, in: Proc. IEEE International Workshop on Open-source Software for Scientific Computation, 2011, pp. 71–76. doi:10.1109/OSSC.2011.6184697.

[20] R. Madonski, M. Nowicki, P. Herman, Practical solution to positivity problem in water management systems - an ADRC approach, in: Proc. American Control Conference, 2016, pp. 1542–1547. doi:10.1109/ACC.2016.7525135.

[21] P. Nowak, K. Stebel, T. Klopot, J. Czeczot, M. Fratczak, P. Laszczyk, Flexible function block for industrial applications of active disturbance rejection controller, Archives of Control Sciences 28 (3) (2018) 379–400. doi:10.24425/acs.2018.124708.
[22] P. Grelewicz, P. Nowak, J. Czeczot, J. Musial, Increment count method and its PLC-based implementation for autotuning of reduced-order ADRC with Smith Predictor, IEEE Transactions on Industrial Electronics 68 (12) (2021) 12554–12564. doi:10.1109/TIE.2020.3045696.

[23] W. Xue, R. Madonski, K. Lakomy, Z. Gao, Y. Huang, Add-on module of active disturbance rejection for set-point tracking of motion control systems, IEEE Transactions on Industry Applications 53 (4) (2017) 4028–4040. doi:10.1109/TIA.2017.2677360.

[24] L. Sun, J. Dong, D. Li, Y. Kwang, A practical multivariable control approach based on inverted decoupling and decentralized active disturbance rejection control, Industrial & Engineering Chemistry Research 55 (7) (2016) 2008–2019. doi:10.1021/acs.iecr.5b03738.

[25] S. Ma, M. Sun, Z. Chen, Interactive ADRC design for flight attitude control, in: Proc. Data Driven Control and Learning Systems, 2017, pp. 611–616. doi:10.1109/DDCLS.2017.8068142.

[26] B. Martínez, J. Sanchis, S. García-Nieto, M. M., Control por rechazo activo de perturbaciones: guía de diseño y aplicación (in Spanish), Revista Iberoamericana de Automática e Informática Industrial 18 (2021) 201–217. doi:10.4995/riai.2020.14058.

[27] H.-J. Hu, Z. Zhu, Modeling and simulation of linear active disturbance rejection controller in Simulink, in: Proc. International Conference on Electronic, Control, Automation and Mechanical Engineering, 2017, pp. 547–552. doi:10.12783/dtetr/ecame2017/18451.

[28] P. Jiang, J. Hao, X. Zong, P. Wang, Modeling and simulation of active-disturbance-rejection controller with Simulink, in: Proc. International Conference on Machine Learning and Cybernetics, 2010, pp. 927–931. doi:10.1109/ICMLC.2010.5580604.
[29] M. Kia, P. Mansouri, A. Javdani, Modeling and simulation of a single gain tuning ADRC controller in Matlab/Simulink, in: Proc. International Symposium on Industrial Electronics, 2020, pp. 154–159. doi:10.1109/ISE45063.2020.9152398.

[30] K. Lakomy, R. Madonski, Cascade extended state observer for active disturbance rejection control applications under measurement noise, ISA Transactions 109 (2021) 1–10. doi:10.1016/j.isatra.2020.09.007.

[31] Y. Huang, W. Xue, Active disturbance rejection control: methodology and theoretical analysis, ISA Transactions 53 (4) (2014) 963–976. doi:10.1016/j.isatra.2014.03.003.

[32] K. Lakomy, R. Madonski, B. Dai, J. Yang, P. Kicki, M. Ansari, S. Li, Active disturbance rejection control design with suppression of sensor noise effects in application to DC–DC buck power converter, IEEE Transactions on Industrial Electronics 69 (1) (2022) 816–824. doi:10.1109/TIE.2021.3055187.

[33] R. Madonski, S. Shao, H. Zhang, Z. Gao, J. Yang, S. Li, General error-based active disturbance rejection control for swift industrial implementations, Control Engineering Practice 84 (2019) 218–229. doi:10.1016/j.conengprac.2018.11.021.

[34] R. Madonski, M. Stanković, S. Shao, Z. Gao, J. Yang, S. Li, Active disturbance rejection control of torsional plant with unknown frequency harmonic disturbance, Control Engineering Practice 100 (2020) 104413. doi:10.1016/j.conengprac.2020.104413.

[35] W. Xue, Y. Huang, Performance analysis of 2-DOF tracking control for a class of nonlinear uncertain systems with discontinuous disturbances, International Journal of Robust and Nonlinear Control 28 (4) (2018) 1456–1473. doi:10.1002/rnc.3972.
[36] S. Chen, Y. Huang, Z. Zhao, The necessary and sufficient condition for the uncertain control gain in active disturbance rejection control (2020). arXiv:2006.11731.

[37] P. Kicki, K. Lakomy, K. M. B. Lee, Tuning of extended state observer with neural network-based control performance assessment (2021). arXiv:2103.15516.

[38] K. Lakomy, R. Patelski, D. Pazderski, ESO architectures in the trajectory tracking ADR controller for a mechanical system: a comparison, in: A. Bartoszewicz, J. Kabziński, J. Kacprzyk (Eds.), Advanced, Contemporary Control, Springer, 2020, pp. 1323–1335. doi:10.1007/978-3-030-50936-1_110.

[39] H. Sira-Ramirez, A. Luviano-Juarez, M. Ramirez-Neria, E. Zurita-Bustamante, Active Disturbance Rejection Control of Dynamic Systems: a Flatness Based Approach, Butterworth-Heinemann, 2017.

[40] L. B. Freidovich, H. K. Khalil, Performance recovery of feedback-linearization-based designs, IEEE Transactions on Automatic Control 53 (10) (2008) 2324–2334. doi:10.1109/TAC.2008.2006821.

[41] M. Ran, Q. Wang, C. Dong, Stabilization of a class of nonlinear systems with actuator saturation via active disturbance rejection control, Automatica 63 (2016) 302–310. doi:10.1016/j.automatica.2015.10.010.

[42] D. Astolfi, L. Marconi, L. Praly, A. R. Teel, Low-power peaking-free high-gain observers, Automatica 98 (2018) 169–179. doi:10.1016/j.automatica.2018.09.009.

[43] M. Ran, J. Li, L. Xie, A new extended state observer for uncertain nonlinear systems, Automatica 131 (2021) 109772. doi:10.1016/j.automatica.2021.109772.

[44] R. Madonski, A. Piosik, P. Herman, High-gain disturbance observer tuning seen as a multicriteria optimization problem, in: Proc. Mediter-
[45] R. Madonski, Z. Gao, K. Lakomy, Towards a turnkey solution of industrial control under the active disturbance rejection paradigm, in: Proc. Annual Conference of the Society of Instrument and Control Engineers of Japan, 2015, pp. 616–621. doi:10.1109/SICE.2015.7285478.

[46] B. DeBoon, S. Nokleby, C. Rossa, Multi-objective gain optimizer for a multi-input active disturbance rejection controller: application to series elastic actuators, Control Engineering Practice 109 (2021) 104733. doi:10.1016/j.conengprac.2021.104733.

[47] W. Wei, W. Xue, D. Li, On disturbance rejection in magnetic levitation, Control Engineering Practice 82 (2019) 24–35. doi:10.1016/j.conengprac.2018.09.018.

[48] R. Madonski, K. Lakomy, M. Stankovic, S. Shao, J. Yang, S. Li, Robust converter-fed motor control based on active rejection of multiple disturbances, Control Engineering Practice 107 (2021) 104696. doi:10.1016/j.conengprac.2020.104696.

[49] P. de Moura Oliveira, J. D. Hedengren, J. Rossiter, Introducing digital controllers to undergraduate students using the TCLab Arduino kit, IFAC-PapersOnLine 53 (2) (2020) 17524–17529, IFAC World Congress. doi:10.1016/j.ifacol.2020.12.2662.

[50] J. Park, R. A. Martin, J. D. Kelly, J. D. Hedengren, Benchmark temperature microcontroller for process dynamics and control, Computers & Chemical Engineering 135 (2020) 106736. doi:10.1016/j.compchemeng.2020.106736.