21. AALE-Konferenz
Dresden, 12.–14. März 2025

Control Engineering for Humans with ADRC

Gernot Herbst[1]

Abstract

More than half a century of intense research in modern control theory has not stopped practitioners from continuing to use PID controllers for most real-world control problems. While not a mathematical one, this might be the most convincing proof of their robustness: if necessary, they can be implemented with minimal control engineering experience. Often, controllers are designed by individuals who are experts in their specific domain but are not dedicated control theory specialists. To address the needs of these engineers, this article wants to put an approach known as Active Disturbance Rejection Control (ADRC) in the spotlight—a method that has gradually evolved into an industry-ready alternative to PID control. Reflecting on two decades of improvements, this article portrays ADRC as a solution for everyday control problems that can be both easier to use and richer in out-of-the-box features.

Keywords

Active Disturbance Rejection Control (ADRC), State-Space Control, PID Control

1 Motivation and Goal

Often referred to as a “hidden technology” [1], automatic control is a key disciplines required to achieve the desired functionality in systems—across a wide range of industries. In many applications, the century-old family of Proportional-Integral-Derivative (PID) controllers continues to be the workhorse of control engineering and preferred choice for those tasked with designing and implementing the actual control loops [2]. In view of decades of academic efforts spent in developing alternative solutions, this prevalence is both astonishing and, perhaps, disappointing. On the other hand, it is a remarkable evidence of the accessibility and robustness of PID control: simple enough to be understood “without a PhD” [3], and tunable, if necessary, even by trial and error—they can still deliver satisfactory results even if many control loops are not perfectly tuned [4].

Why does a control paradigm obviously have to have these properties to succeed in the real world? The primary hypothesis underlying this article is that a significant percentage of control loops are set up by individuals who are domain experts in their respective industries but are not dedicated control engineering specialists familiar with all the intricacies. This article therefore aims to take a human-centric view of the typical control-related challenges faced by these individuals and to examine an alternative general-purpose control solution, known as Active Disturbance Rejection Control (ADRC), in relation to these challenges. It will be argued that ADRC provides a better overall standard feature set that make it an attractive choice practitioners should consider for day-to-day control engineering tasks.

To that end, an overview of ADRC will be provided in Sect. 2, before addressing typical issues and pitfalls in a rather informal manner in Sect. 3. This is intended to serve as a decision-making basis for determining whether ADRC addresses potential pain points experienced by those who view control engineering as just one necessary tool among others, rather than as an end in itself.

[1] University of Applied Sciences Zwickau, Faculty of Electrical Engineering, Zwickau, Germany

Open Access. © 2025 Gernot Herbst
This work is licensed under the Creative Commons Attribution 4.0 License. +

2 Introducing Active Disturbance Rejection Control (ADRC)

A general-purpose control solution known as Active Disturbance Rejection Control (ADRC) was originally developed as a nonlinear observer-based state-space controller [5]. A surge in popularity and the possibility to apply proven techniques for tuning and stability analysis came with its simplification to a linear form [6], where LUENBERGER observer and linear state feedback are being employed. How can these ingredients create a general-purpose controller, when state-based methods are typically associated with the need for accurate plant modeling?

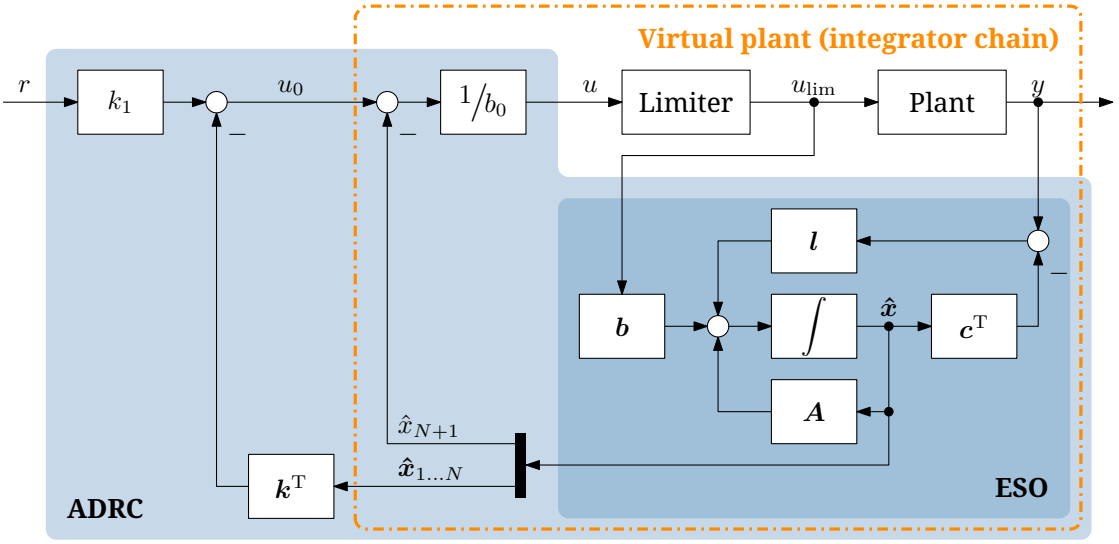

Figure 1: Control loop with continuous-time ADRC, highlighting its main ingredients and concepts.

A key ingredient is to use a generic plant model inside the observer: an N-th order integrator chain with input gain b_0 for plants of order N, i.e., $P(s) = b_0 \cdot s^{-N}$. This is enhanced by an estimate of an input disturbance of this plant model. Combined, this is usually denoted *extended state observer*, or ESO. It is important to note that the input disturbance both covers actual disturbances and the modeling error that we are obviously (and deliberately) making here. It is therefore also referred to as *generalized* or *total disturbance*, $f(t)$. Rejecting the disturbance using the estimate $\hat{f}(t)$ and normalizing the plant gain b_0, an inner control loop is formed, which, if properly tuned, behaves as a unity-gain integrator chain to an outer state-feedback loop, as shown in Fig. 1. This enables a straightforward tuning procedure solely based on desired closed-loop dynamics. The controller and observer equations are given as

$$u(t) = \frac{1}{b_0} \cdot \left(k_1 \cdot r(t) - \left(\overbrace{\begin{pmatrix} k_1 & \cdots & k_N \end{pmatrix}}^{\boldsymbol{k}^{\mathrm{T}}} \quad 1 \right) \cdot \overbrace{\begin{pmatrix} \hat{x}_1 \\ \vdots \\ \hat{x}_{N+1} \end{pmatrix}}^{\hat{\boldsymbol{x}}(t)} \right) \quad \text{with} \quad \hat{\boldsymbol{x}}(t) = \begin{pmatrix} \hat{y}(t) \\ \dot{\hat{y}}(t) \\ \vdots \\ \hat{f}(t) \end{pmatrix} \quad \text{and} \tag{1}$$

$$\dot{\hat{\boldsymbol{x}}}(t) = \underbrace{\begin{pmatrix} \boldsymbol{0}^{N\times 1} & \boldsymbol{I}^{N\times N} \\ 0 & \boldsymbol{0}^{1\times N} \end{pmatrix}}_{\boldsymbol{A}} \cdot \hat{\boldsymbol{x}}(t) + \underbrace{\begin{pmatrix} \boldsymbol{0}^{(N-1)\times 1} \\ b_0 \\ 0 \end{pmatrix}}_{\boldsymbol{b}} \cdot u(t) + \underbrace{\begin{pmatrix} l_1 \\ \vdots \\ l_{N+1} \end{pmatrix}}_{\boldsymbol{l}} \cdot \left(y(t) - \underbrace{\begin{pmatrix} 1 & \boldsymbol{0}^{1\times N} \end{pmatrix}}_{\boldsymbol{c}^{\mathrm{T}}} \cdot \hat{\boldsymbol{x}}(t) \right). \tag{2}$$

Since the observer provides estimates of the plant output and its first $N-1$ derivatives, the outer feedback controller can achieve the following closed-loop behavior with easily adjustable dynamics:

$$y(s) \approx \frac{1}{s^N} \cdot \left(k_1 r(s) - \sum_{i=1}^{N} k_i s^{i-1} y(s) \right) \quad \rightarrow \quad G_{\mathrm{CL}}(s) = \frac{y(s)}{r(s)} \approx \frac{1}{\frac{1}{k_1} \cdot s^N + \frac{k_N}{k_1} \cdot s^{N-1} + \ldots + \frac{k_2}{k_1} \cdot s + 1}. \tag{3}$$

Open Access. © 2025 Gernot Herbst
This work is licensed under the Creative Commons Attribution 4.0 License. +

3 As a Engineer Implementing Closed-Loop Control ...

3.1 ... I Am Well Versed in My Field, but Am Not a Control Engineering Expert

Control engineering is only one of disciplines required to build a product, hence many control loops are designed by engineers with a different specialization. One of the main strengths of PID control is that it can be used rather effectively even under these circumstances. This benchmark must be applied to any alternative that wants to compete in this "market". In the remainder of this section, typical issues that need to be addressed in selecting, tuning, and implementing a controller are being discussed. It will be shown if and how ADRC can make the life of engineers easier.

3.2 ... I Would Prefer a Standard Solution

Contrary to what the name "PID controller" might suggest, this term actually covers a whole family of different control laws and implementations: series or parallel, interacting or non-interacting forms, etc. It is therefore a necessary design decision to select one specific PID control law (or to work with what one is given), and to ensure that, for example, tuning methods are chosen that match the selected form [4].

For the linear variant of ADRC, the structure shown in Fig. 1 can be considered the standard (most widely used) form that one can build upon: a LUENBERGER observer with linear state feedback. One of the states is the *total disturbance* estimate, which is being fed back with unity gain. It acts similarly to the integrator component in PID controllers, ensuring steady-state accuracy.

3.3 ... I Want to Spend Less Time With Controller Tuning

Tuning PID controllers has been a topic in research and engineering for decades, resulting in an overwhelming variety of tuning rules to chose from [4]. Additionally, users have to take care that a selected tuning approach matches the PID control law form and the application scenario, focusing either on disturbance rejection or reference trajectory tracking.

For ADRC, the tuning procedure is being considerably simplified, as the inner loop shown in Fig. 1 normalizes gain and dynamics for the outer control loop. As already indicated by the approximate closed-loop transfer function (3), which is valid if the inner loop is fast enough, the closed-loop dynamics only depend on the controller parameters. Therefore it is possible to derive analytical equations for the controller gains depending directly on the desired dynamics, using pole placement. It is especially easy to place all closed-loop poles at a single location $-\omega_{\mathrm{CL}}$, as proposed by [7], resulting in a non-overshooting, critically-damped transient behavior. To that end, we require for the denominator of (3):

$$\frac{1}{k_1}\cdot s^N+\frac{k_N}{k_1}\cdot s^{N-1}+\ldots+\frac{k_2}{k_1}\cdot s+1 \stackrel{!}{=} \left(\frac{s}{\omega_{\mathrm{CL}}}+1\right)^N \quad \xrightarrow{k_1=\omega_{\mathrm{CL}}^N} \quad s^N+k_N\cdot s^{N-1}+\ldots+k_2\cdot s+k_1 \stackrel{!}{=} \left(s+\omega_{\mathrm{CL}}\right)^N . \tag{4}$$

Unfolding the right-hand side polynomial in (4) allows to immediately read off tuning equations for the controller gains based on the desired bandwidth of the closed loop. Its bandwidth ω_{CL} can also be selected through an easily interpretable time-domain characteristic such as the 98 % settling time, for example through a heuristically obtained relation [8]:

$$k_i = \frac{N!}{(N-i+1)!\cdot(i-1)!}\cdot\omega_{\mathrm{CL}}^{N-i+1} \quad \text{with} \quad i=1,\ldots,N \quad \text{and} \quad \omega_{\mathrm{CL}} \approx \frac{2+2\cdot N}{T_{\mathrm{settle},98\%}} . \tag{5}$$

For the observer gains, equally simple tuning equations are obtained (cf. [8] for a detailed derivation). To provide the estimated total disturbance for the inner loop as well as the state variables for the outer control loop in a timely fashion, the observer bandwidth must be higher. A straightforward tuning approach is therefore based on critically damped pole placement at a common location $-k_{\mathrm{ESO}}\cdot\omega_{\mathrm{CL}}$,

Open Access. © 2025 Gernot Herbst
This work is licensed under the Creative Commons Attribution 4.0 License. +

i.e., faster than the closed loop by a factor k_{ESO}:

$$l_i = \frac{(N+1)!}{(N-i+1)! \cdot i!} \cdot \left(k_{\mathrm{ESO}} \cdot \omega_{\mathrm{CL}}\right)^i \quad \text{with} \quad i = 1, \ldots, N+1 \quad \text{and} \quad 1 \lessapprox k_{\mathrm{ESO}} \lessapprox 10\,. \tag{6}$$

Summarizing this tuning approach, one finds that although N controller and $N+1$ observer feedback gains must be computed, this task is reduced to selecting a bandwidth or settling time for the closed loop—which is usually relatively easy using domain knowledge of the control task at hand—and an observer bandwidth factor. The latter is typically in the single-digit range and thus in a very limited search space.

3.4 ... I Also Want to Spend Less Time With Plant Modeling

Apart from the plant order N, ADRC requires only knowledge of a single plant model parameter, the input gain b_0. Due to its importance, it is sometimes referred to as the *critical gain parameter*. To illustrate how one can obtain b_0 easily for typical process dynamics, let us bring first- and second-order differential equations into the form of a disturbed integrator chain $y^{(N)}(t) = b_0 \cdot u(t) + f(t)$:

$$T \cdot \dot{y}(t) + y(t) = K \cdot u(t) \quad \rightarrow \quad \dot{y}(t) = \underbrace{\frac{K}{T}}_{b_0} \cdot u(t) + \underbrace{\frac{-1}{T} \cdot y(t)}_{f(t)} \tag{7}$$

$$T^2 \cdot \ddot{y}(t) + 2DT \cdot \dot{y}(t) + y(t) = K \cdot u(t) \quad \rightarrow \quad \ddot{y}(t) = \underbrace{\frac{K}{T^2}}_{b_0} \cdot u(t) + \underbrace{\frac{-2D}{T} \cdot \dot{y}(t) - \frac{1}{T^2} \cdot y(t)}_{f(t)} \tag{8}$$

To obtain b_0 when using ADRC, one can therefore build upon either analytical modeling or a large variety of experimentally driven system identification methods in time or frequency domain. Considering the fact that the other tuning parameters of ADRC are largely based on the desired properties of the closed loop, as discussed in Sect. 3.3, b_0 can be said to be the only true "unknown" factor that must be selected—if there is no other option, by trial and error. One can therefore conclude that ADRC, although based on more complicated building blocks than PID control, is easier to tune.

3.5 ... I Would Prefer Not to Have to Worry About Integrator Windup

Avoiding integrator windup is essential when implementing PID control. To the disadvantage of its users, there are different variants to choose from [9], which can be divided into rather pragmatic ad-hoc approaches and much more involved model-based solutions [10, 11].

Since ADRC is based on an observer, a very simple solution for the windup problem exists [12]: As indicated in Fig. 1, the observer only needs to be aware of the actual (limited) controller output value. This eliminates the task of selecting an anti-windup method as in the PID world, which sometimes even requires to tune another loop (for example in the PID back-calculation method). On an equation level, the only necessary change is to use u_{lim} instead of u in the observer equation (2):

$$\dot{\hat{\boldsymbol{x}}}(t) = \boldsymbol{A} \cdot \hat{\boldsymbol{x}}(t) + \boldsymbol{b} \cdot u(t) + \boldsymbol{l} \cdot \left(y(t) - \boldsymbol{c}^{\mathrm{T}} \cdot \hat{\boldsymbol{x}}(t)\right) \quad \rightarrow \quad \dot{\hat{\boldsymbol{x}}}(t) = \boldsymbol{A} \cdot \hat{\boldsymbol{x}}(t) + \boldsymbol{b} \cdot u_{\mathrm{lim}}(t) + \boldsymbol{l} \cdot \left(y(t) - \boldsymbol{c}^{\mathrm{T}} \cdot \hat{\boldsymbol{x}}(t)\right)\,. \tag{9}$$

3.6 ... I Feel Lost in State Space

A key advantage of PID control is that its components (proportional, integral, derivative) can be easily understood. It is therefore clearly a disadvantage of a competing approach like ADRC if state-space methods such as an observer are being used, which require more control-related background knowledge. However, it is possible to understand the dynamics of ADRC using frequency domain concepts, which engineers are typically more familiar with. This allows to gain insights into the working principles, compare against classical solutions, or even implement ADRC using transfer functions [13].

Open Access. © 2025 Gernot Herbst
This work is licensed under the Creative Commons Attribution 4.0 License. +

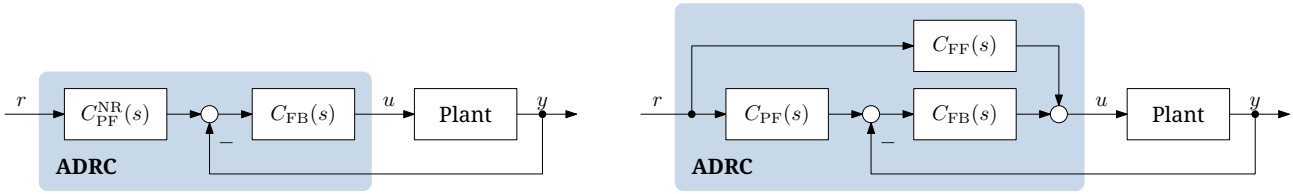

Figure 2: To better understand and analyze ADRC from a traditional (frequency domain) perspective, it can be represented using either a common two-degrees-of-freedom structure consisting of a (non-realizable) prefilter and a feedback controller, or a realizable variant with an additional feedforward transfer function.

Two possible transfer function representations of linear ADRC are shown in Fig. 2. The first one is a well-known two-degrees-of-freedom (2DOF) structure consisting of a prefilter and a feedback controller, with the following transfer functions for the N-th order case (e.g. $N = 1$ for first-order plants):

$$C_{\mathrm{FB}}(s) = \frac{K_{\mathrm{I}}}{s} \cdot \frac{1 + \sum_{i=1}^{N} \beta_i \cdot s^i}{1 + \sum_{i=1}^{N} \alpha_i \cdot s^i} \quad \text{and} \quad C_{\mathrm{PF}}^{\mathrm{NR}}(s) = \frac{1 + \sum_{i=1}^{N+1} \gamma_i \cdot s^i}{1 + \sum_{i=1}^{N} \beta_i \cdot s^i} \,. \tag{10}$$

Obviously the feedback controller $C_{\mathrm{FB}}(s)$ consists of an integrator and an N-th order lead-lag filter. This means that for the important first- and second-order cases, $C_{\mathrm{FB}}(s)$ corresponds to a PI or PID controller with a first- or second-order low-pass filter, respectively. Due to the order of its numerator polynomial, the prefilter $C_{\mathrm{PF}}^{\mathrm{NR}}(s)$ is not realizable, hence tagged “NR”. It is, however, possible to find a realizable structure using one additional component, a feedforward transfer function $C_{\mathrm{FF}}(s)$ [13]. This is shown on the right-hand side in Fig. 2, and will reduce the order of the prefilter numerator to N, making it realizable. Readers interested in the derivation and coefficient values will find all details in [8].

In summary, one can therefore say that ADRC provides a control law that is very well understandable as a classical controller, but with simpler tuning, and with the features enabled by an observer-based approach, such as easier implementation of windup protection.

3.7 … I Do Not Want to Worry About Discretization

Most practical implementations of a control law will be in discrete-time form, for example using a microcontroller or a programmable logic controller (PLC). In addition to the many decisions that PID users have already had to make, an approach for discretization has to be selected (rectangular or trapezoidal rules, zero-order hold equivalent etc.), and one has to decide whether the individual P, I, D components are being discretized, or the overall controller transfer function.

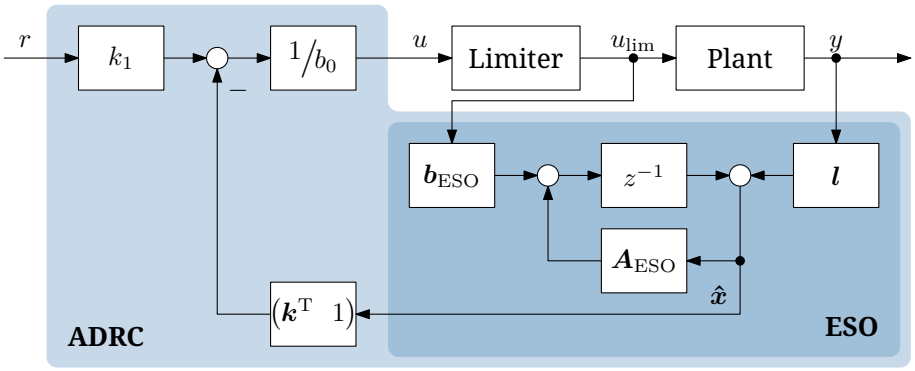

Figure 3: Discrete-time implementation of ADRC obtained through zero-order hold discretization of the observer.

Open Access. © 2025 Gernot Herbst
This work is licensed under the Creative Commons Attribution 4.0 License. +

For ADRC, the observer has to be brought in discrete-time form, and an established, “canonical” choice exists based on the work performed in [14]. It consists of the zero-order hold equivalent (ZOH) and the so-called *current observer* approach—a discrete-time form of the observer that makes use of the most recent measurement $y(k)$ when computing the estimate $\hat{\boldsymbol{x}}(k)$ [15]. Figure 3 shows the resulting block diagram as derived in [8], where all equations for the matrix and vector coefficients can be found.

The discrete-time controller and observer equations for the time instant $t = k \cdot T_{\text{sample}}$ now read:

$$u(k) = \frac{1}{b_0} \cdot \left(k_1 \cdot r(k) - \begin{pmatrix} \boldsymbol{k}^{\mathrm{T}} & 1 \end{pmatrix} \cdot \hat{\boldsymbol{x}}(k) \right) \quad \text{and} \tag{11}$$

$$\hat{\boldsymbol{x}}(k) = \boldsymbol{A}_{\text{ESO}} \cdot \hat{\boldsymbol{x}}(k-1) + \boldsymbol{b}_{\text{ESO}} \cdot u(k-1) + \boldsymbol{l} \cdot y(k) \,. \tag{12}$$

The continuous-time controller and observer gains can be reused if the sampling frequency is considerably larger than the bandwidth of control loop and observer. However, using dedicated discrete-time pole placement is always preferable and will ensure that the performance specifications of a continuous-time design such as settling time are maintained in the discrete-time implementation [16]. Ready-made equations for controller and observer gains are also given in [8].

3.8 … I Fear Matrix Multiplications Are Too Heavy for My Target System

A discrete-time PID implementation has only a very limited computational footprint (runtime and memory), which make it an ideal solution for resource-constrained target systems and/or high control frequencies. The discrete-time observer equation (12) may therefore raise questions regarding the computational complexity of ADRC, as, for example, a multiplication with an $(N+1) \times (N+1)$ matrix $\boldsymbol{A}_{\text{ESO}}$ must be performed in every time step.

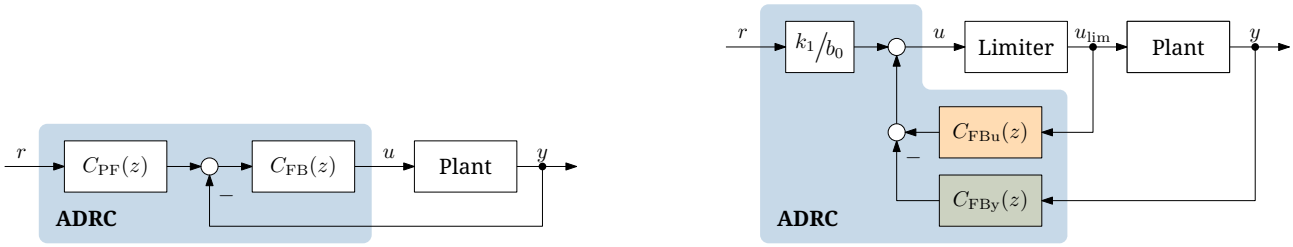

Figure 4: Transfer function forms of discrete-time ADRC. While mathematically equivalent, only the right-hand side variant with two transfer functions in the feedback loop retains the anti-windup properties of the observer-based structure—and it can be implemented more efficiently, as they share the same denominator.

Using a transfer function form of discrete-time ADRC, however, will reduce the number of coefficients and multiplications. Figure 4 shows two possible variants as derived in [16]. Similar to the continuous-time case, the left-hand side form consists of a prefilter and a feedback controller with integrator plus lead-lag filter behavior—a well-known two-degrees-of-freedom structure, in discrete-time domain even without realizability worries. The right-hand side form with two transfer functions in the feedback path is a bit unusual, yet the preferable solution. It retains the feedback of control signal limitation as in the observer-based structure, and hence the built-in windup protection. Since both transfer functions share the same denominator, they can be implemented very efficiently, as proposed by [17]:

$$u(z) = \frac{k_1}{b_0} \cdot r(z) + \frac{-\left(\sum_{i=0}^{N} \beta_i z^{-i}\right) \cdot y(z) + z^{-1} \cdot \left(\sum_{i=0}^{N} \gamma_i z^{-i}\right) \cdot u_{\text{lim}}(z)}{1 + \sum_{i=1}^{N+1} \alpha_i z^{-i}} \,. \tag{13}$$

$C_{\text{FBy}}(z) =$ $= C_{\text{FBu}}(z)$

Equations that relate the discrete-time α, β, γ coefficients to the previously introduced tuning parameters ω_{CL}, k_{ESO}, b_0 can be found in [8], such that prospective ADRC users do not have to worry about discrete-time pole placement or transformations from state space to transfer functions.

Open Access. © 2025 Gernot Herbst
This work is licensed under the Creative Commons Attribution 4.0 License. +

3.9 … I Need to Tailor a Custom Solution

The observer-based state-space framework allows for an organic extension of ADRC when specific requirements arise in an application that cannot be satisfactorily solved with the standard approach.

– One practically relevant case is the handling of processes that include dead times. A pragmatic extension of ADRC that improves the dynamic behavior is to feed the controller output with a delay matching the process back to the observer [18]. This allows to design for faster transients while reducing oscillations in both controlled variable and controller output.

– Another application case is the estimation and rejection of non-constant (e.g. sinusoidal) disturbances [19]. This can achieved by replacing the constant disturbance model in the observer with a model of the expected disturbance, as also demonstrated in [8].

– The transfer function forms of ADRC shown in Fig. 2 and Fig. 4, consisting of prefilter and feedback controller, give rise to the idea of replacing the prefilter with a customized filter, enabling a true 2DOF design [20]. Simply omitting the prefilter leads to a form known as *error-based ADRC* [21].

– Even though the burden of plant modeling is much reduced when using ADRC, sometimes a plant model is already or easily available. Then, of course, one does not have to throw away this knowledge. Similar to an enhanced disturbance model, the plant model can be incorporated into the extended state observer [22]. This can help to improve the dynamics of the closed loop, as the observer's job will be easier with a more detailed model.

3.10 … I Do Not Want to Start From Scratch With My Implementation

The way from an analytical description of a control law to one's own model-based or even source code implementation can be hard, especially if this path is taken for the first time. Having a reference implementation that enables quick simulations, tinkering, or simply serves as an inspiration therefore lowers entry barriers and significantly shortens the path to one's own solution. There are countless examples of PID controller implementations in C or Simulink to choose from, but due to the many combinations and choices to be made by a user (notably: PID form, discretization method, anti-windup approach), it may be hard to identify a proper one for the application at hand.

With the almost canonical choice of ZOH discretization of linear ADRC and its simple anti-windup approach, the choice is greatly simplified for users. In the field of model-based environments, notably Matlab/Simulink, a linear ADRC block is available since release R2022b in Mathworks' own add-on offering *Simulink Control Design*, cf. Fig. 5. For error-based ADRC, the so-called *ADRC Toolbox*[1] is freely available from the Mathworks File Exchange. Another free offering that covers multiple variants of continuous- and discrete-time ADRC is the *Linear ADRC Blockset*[2].

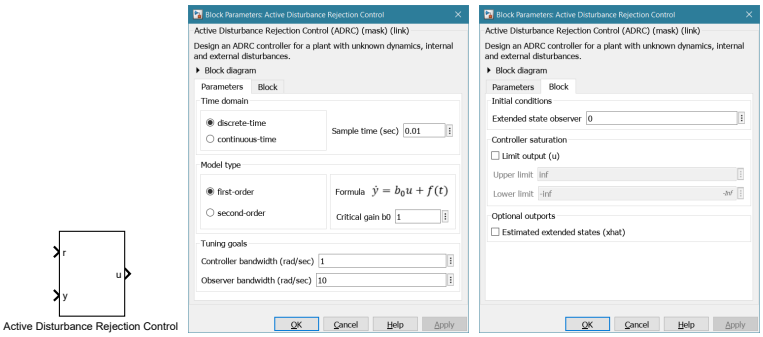

Figure 5: Symbol and parameter dialogs of the *Active Disturbance Rejection Control* block from *Simulink Control Design*.

Finally, the necessary steps to implement both state-space and transfer function forms of discrete-time ADRC as C source code are described in the book [8], which not only contains every necessary equation hinted at in this article, but is freely available as an open access publication for academics and practitioners alike.

1 ADRC Toolbox: https://www.mathworks.com/matlabcentral/fileexchange/102249
2 Linear ADRC Blockset: https://www.mathworks.com/matlabcentral/fileexchange/135552

Open Access. © 2025 Gernot Herbst
This work is licensed under the Creative Commons Attribution 4.0 License. +

4 Conclusion

Over the last two decades, steady progress could be witnessed that has made ADRC a viable alternative for control problems in industrial practice, ranging from simplified tuning, efficient digital implementation variants up to ready-made software offerings.

In summary, this article argued that ADRC provides a better default feature set while leaving less details to worry about, making it an attractive option especially for engineers from other (non-control) domains in need of a control solution. Tuning is reduced to selecting a desired bandwidth (or settling time) and an observer factor. Since users often have knowledge or a good intuition about the achievable dynamics in their application, the search space for these parameters is much smaller than for PID controller gains. Plant modeling is simplified to its order and a single gain value, which might therefore be the only parameter that requires some effort to determine. And finally, less choices have to be made regarding the (discrete-time) implementation as well as an anti-windup mechanism for ADRC. Nevertheless, ADRC is open to extensions owing to the observer-based approach at its core.

Should one try to replace PID control in every application? No! But everyone should feel encouraged to seriously consider adding ADRC to their toolbox, and perhaps even as a first choice.

5 References

[1] Åström, K. J. Automatic control — the hidden technology. Advances in Control, 1999.
[2] Hägglund, T., Guzmán, J. L. Give us PID controllers and we can control the world. IFAC-PapersOnLine, 2024, 58(7).
[3] Wescott, T. PID without a PhD. Technical report, Wescott Design Services, Inc., Oregon City, OR, 2016.
[4] O'Dwyer, A. Handbook of PI and PID Controller Tuning Rules. Imperial College Press, 2009.
[5] Han, J. From PID to active disturbance rejection control. IEEE Transactions on Industrial Electronics, 2009, 56(3).
[6] Gao, Z. Active disturbance rejection control: A paradigm shift in feedback control system design. American Control Conference, 2006.
[7] Gao, Z. Scaling and bandwidth-parameterization based controller tuning. American Control Conference, 2003.
[8] Herbst, G., Madonski, R. Active Disturbance Rejection Control: From Principles to Practice. Control Engineering. Birkhäuser, Cham, 2025.
[9] Visioli, A. Practical PID Control. Advances in Industrial Control. Springer, London, 2006.
[10] Hippe, P. Windup in Control: Its Effects and Their Prevention. Advances in Industrial Control. Springer, London, 2006.
[11] Tarbouriech, S., Garcia., G., da Silva Jr., J. M. G., Queinnec, I. Stability and Stabilization of Linear Systems with Saturating Actuators. Springer, London, 2011.
[12] Åström, K. J., Murray, R. M. Feedback Systems: An Introduction for Scientists and Engineers. Princeton University Press, 2021.
[13] Herbst, G. Transfer function analysis and implementation of active disturbance rejection control. Control Theory and Technology, 2021, 19.
[14] Miklosovic, R., Radke, A., Gao, Z. Discrete implementation and generalization of the extended state observer. American Control Conference, 2006.
[15] Franklin, G. F., Workman, M. L., Powell, D. Digital Control of Dynamic Systems. Addison-Wesley Longman Publishing, Boston, 1997.
[16] Herbst, G., Madonski, R. Tuning and implementation variants of discrete-time ADRC. Control Theory and Technology, 2023, 21.
[17] Herbst, G. A minimum-footprint implementation of discrete-time ADRC. European Control Conference, 2021.
[18] Zhao, S., Gao, Z. Modified active disturbance rejection control for time-delay systems. ISA Transactions, 2014, 53(4).
[19] Madonski, R., Stanković, M. R., Ferdjali, A., Shao, S., Gao, Z. General ADRC design for systems with periodic disturbances of unknown and varying frequencies. Journal of Dynamic Systems, Measurement, and Control, 143(1), 2021.
[20] Madonski, R., Herbst, G., Stanković, M. R. ADRC in output and error form: Connection, equivalence, performance. Control Theory and Technology, 2023, 21.
[21] Madonski, R., Shao, S., Zhang, H., Gao, Z., Yang, J., Li, S. General error-based active disturbance rejection control for swift industrial implementations. Control Engineering Practice, 2019, 84.
[22] Fu, C., Tan, W. Tuning of linear ADRC with known plant information. ISA Transactions, 2016, 65.

Open Access. © 2025 Gernot Herbst
This work is licensed under the Creative Commons Attribution 4.0 License. +