Abstract—Practical implementations of active disturbance rejection control (ADRC) will almost always take place in discretized form. Since applications may have quite different needs regarding their discrete-time controllers, this article summarizes and extends the available set of ADRC implementations to provide a suitable variant for as many as possible use cases. In doing so, the gap between quasi-continuous and discrete-time controller tuning is being closed for applications with low sampling frequencies. The main contribution of this article is the derivation of three different discrete-time implementations of error-based ADRC. It is shown that these are almost one-to-one counterparts of existing output-based implementations, to the point where transfer functions and coefficients can be reused in unaltered form. In this way, error-based implementations become firmly rooted in the established landscape of discrete-time ADRC. Furthermore, it becomes possible to equip error-based variants with windup protection abilities known from output-based ADRC.

Index Terms—Active disturbance rejection control (ADRC), Error-based form, Controller tuning, Discretization

I. INTRODUCTION

A. Background and Related Work

Active disturbance rejection control (ADRC) [1] is a highly practical approach to solving real-world control problems. As the dominance of PID controller families in this domain is undisputed despite decades of control systems research, the widespread interest ADRC has sparked among both scientists and engineers is all the more remarkable, since it already found its way into numerous applications [2]–[5]. Some foundational milestones enabling this trajectory deep into the territory of PID control can be seen in the development of a linear version [6], an easy tuning rule [7], and its implementation in discrete-time form [8]. Later enhancements include additional features for industrial practice [9], [10], methods for improved disturbance rejection [11], or the incorporation of known plant information [12], [13].

An important barrier to be overcome for real-world adoption is not a technical one, but more a matter of understanding. With its origins in (nonlinear) state space, ADRC might appear unfamiliar to the practicing engineer. To cater such needs, a multitude of studies provides insights and understanding of ADRC from a classical, frequency-domain control engineering perspective [14]–[17].

Taking this avenue also for the actual implementation—i.e., building ADRC using transfer functions—not only eases understanding, but increases efficiency and allows to even reuse existing building blocks from classical control engineering. For discrete-time ADRC, this was put forward by [17], with an even more optimized variant in [18] that comes very close to the runtime performance of PID controllers. These are implementations we will build upon in this article.

A variant of ADRC, using the control error as estimated variable within the observer, was introduced in [19], [20], and accordingly named error-based ADRC. When viewed in transfer function form, it becomes apparent that error-based ADRC is a one-degree of freedom (1DOF) controller, as only one transfer function is needed—a feedback controller in a structure very well known from classical control engineering. Applications studied so far include quadrocopters [21], DC-DC converters [22], and thermal processes [23]. The firm connection between output- and error-based ADRC has been established in continuous-time domain [24]. To the best of our knowledge, however, the task of discretization of error-based ADRC has not yet been properly addressed, apart from an approximate solution using bilinear transform in [25]. We therefore want to extend this connection to the discrete-time domain in this article, providing ready-to-use variants that are exactly rooted in the discretization process of ADRC introduced by [8].

B. Contributions and Structure of this Article

This article aims at increasing the versatility of discrete-time ADRC in two aspects. Firstly, tuning rules for the outer state-feedback controller based on bandwidth parameterization [7] with discrete-time pole placement are introduced, which are needed to consistently achieve the desired bandwidth, even for applications with low sampling frequencies. At the same time, a bridge is being built to the commonly used continuous-time design. Existing transfer function implementations of discrete-time ADRC from [17] and [18] are being reviewed and retrofitted with these results. Secondly, discrete-time state-space and transfer function implementations of error-based ADRC [25] are developed. These exhibit strong connections with classical (output-based) variants up to equivalent transfer function coefficients, and enhance error-based ADRC with flexible control signal limitation and windup protection.

The remainder of this article is structured as follows: Sect. II briefly recapitulates continuous-time ADRC, which we will need for deriving the discrete-time implementations. Building on that, Sect. III summarizes the current practice of implementing discrete-time ADRC in state space form. As will be shown, it relies on quasi-continuous tuning of the outer feedback controller within ADRC. Sect. IV will therefore put the commonly used bandwidth parameterization
approach of the controller on the basis of discrete-time pole placement. Two existing transfer function implementations of discrete-time ADRC are revisited in Sect. V, now providing tuning equations based on the results of Sect. IV. For error-based ADRC, three different implementations are introduced in Sect. VI, all as direct counterparts of existing output-based variants. The article concludes after presenting an example in Sect. VII that compares output- and error-based ADRC in a power electronics control application.

II. SUMMARY OF CONTINUOUS-TIME ADRC

A. Structure and Control Law

All discrete-time implementations discussed in this article are based on the linear variant of ADRC introduced in [7], which we consider the most widely used form. We will briefly summarize it in this section. The control law of continuous-time ADRC in state space as shown in Fig. 1 reads:

\[ u(t) = \frac{1}{b_0} \cdot (k_1 \cdot r(t) - (k^T \cdot \hat{x}(t)) - k^{T-1} \cdot \hat{x}(t)) \]  

(1)

with \( k^T = (k_1 \ldots k_n) \) and \( \hat{x} = (\hat{x}_1 \ldots \hat{x}_{n+1})^T \).

In (1), the unity-gain factor in the state-feedback gain vector corresponds to the rejection of the total disturbance. The vector of states \( \hat{x}(t) \) is being provided by a Luenberger observer. Its plant model most commonly consists of an integrator chain disturbed by a constant input disturbance (the total disturbance \( f(t) \)): \( y^{(n)}(t) = b_0 \cdot u(t) + f(t) \). That way, only the plant order \( n \) and an estimated plant gain \( b_0 \) are required as far as modeling is concerned. The observer equations are therefore given as:

\[ \dot{x}(t) = A \cdot x(t) + b \cdot u_{lim}(t) + l \cdot \left( y(t) - c^T \cdot \hat{x}(t) \right) \]  

(2)

with \( A = \begin{pmatrix} 0^{n \times 1} & I^{n \times n} \end{pmatrix}, b = \begin{pmatrix} 0^{(n-1) \times 1} \end{pmatrix},  
\]

\( l = (l_1 \ldots l_{n+1})^T, c^T = (1 \ 0^{1 \times n}) \).

Note that in (2), the observer uses the limited control signal \( u_{lim}(t) \) instead of \( u(t) \), which is a very simple but effective method of avoiding integrator windup [10], [26]—a very important selling point for using ADRC in practice. That way, no further anti-windup measures have to be implemented, let alone to be tuned. Finally, it might also be possible that the control signal limitation takes place externally, beyond the user’s influence. For these cases, \( u_{lim}(t) \) provides a feedback path for a measured value of the actual (limited) control signal.

B. Controller Tuning

We will now briefly discuss tuning the gain vector \( k^T \) of the outer state-feedback controller from (1). This will also be the basis for developing discrete-time tuning rules in Sect. IV-A.

The \( 1/b_0 \) gain block in Fig. 1 normalizes the plant gain, which, together with feeding back the estimated total disturbance, approximately creates an undisturbed “virtual plant” with unity-gain integrator chain dynamics \( y^{(n)}(t) = u_0(t) \) to be controlled by the controller \( u_0(t) = -k^T \cdot (\hat{x}_1 \ldots \hat{x}_n)^T \).

A state space representation of this virtual plant can be given as:

\[ \dot{x}_{VP}(t) = A_{VP} \cdot x_{VP}(t) + b_{VP} \cdot u_0(t), \]

\[ y(t) = c_{VP}^T \cdot x_{VP}(t) \]  

(3)

with \( A_{VP} = \begin{pmatrix} 0^{(n-1) \times 1} & I^{(n-1) \times (n-1)} \\ 0 & 0^{1 \times (n-1)} \end{pmatrix}, b_{VP} = \begin{pmatrix} 0^{(n-1) \times 1} \\ 1 \end{pmatrix}, c_{VP}^T = \begin{pmatrix} 1 \\ 0^{1 \times (n-1)} \end{pmatrix} \).

The most popular tuning method for linear ADRC was introduced in [7] as bandwidth parameterization. Its simplicity comes from a single tuning parameter, namely, the desired bandwidth \( \omega_{CL} \). In this approach, all poles of the closed loop are placed at a common location \(-\omega_{CL}\):

\[ (\lambda + \omega_{CL})^n \overset{!}{=} \det \left( A \cdot I - (A_{VP} - b_{VP}k^T) \right) \]  

(4)

Equation (5) yields the very simple controller tuning equations \( k_1 = \omega_{CL} \) (for first-order ADRC), and \( k_1 = \omega_{CL}^2, k_2 = 2\omega_{CL} \) (for second-order ADRC). We will come back to these in Sect. IV-B.

C. Observer Tuning

In a similar fashion, tuning the observer can be done by placing the poles of the observer system matrix \((A - Ic^T)\). Using bandwidth parameterization once more, an (obviously higher) bandwidth has to be chosen for the observer, which we will—following [27]—express as \( k_{ESO} \cdot \omega_{CL} \). The equation for pole placement then reads:

\[ (\lambda + k_{ESO} \omega_{CL})^{n+1} \overset{!}{=} \det \left( A \cdot I - (A - Ic^T) \right) \]  

(6)

The observer gains can be obtained from (6) after binominal expansion as follows:

\[ l_i = \frac{(n + 1)!}{(n - i + 1)!} \cdot (k_{ESO} \cdot \omega_{CL})^i \quad \forall i = 1, \ldots, n + 1. \]  

(7)
Fig. 2: Control loop with discrete-time ADRC, state space implementation

III. DISCRETE-TIME ADRC

A. Structure and Control Law

As the control law (1) of linear ADRC consists of static state feedback, its discrete-time counterpart is trivially obtained as:

\[ u(k) = \frac{1}{b_0} \cdot (k_1 \cdot r(k) - (k^T 1) \cdot \hat{x}(k)) \]  

(8)

with  
\[ k^T = (k_1 \cdots k_n). \]

When developing the discrete-time version of the extended state observer, the so-called current observer approach is being used in order to let the current estimate \( \hat{x}(k) \) be based on the most recent measurement \( y(k) \) [8], [28]. For a more detailed derivation we refer the reader to [27], and only present the resulting equations for the discrete-time observer dynamics here:

\[ \dot{\hat{x}}(k) = A_{\text{ESO}} \cdot \hat{x}(k-1) + b_{\text{ESO}} \cdot u_{\text{lim}}(k-1) + I \cdot y(k) \]  

(9)

with  
\[ I = \begin{bmatrix} l_1 & \cdots & l_{n+1} \end{bmatrix}^T, \quad A_{\text{ESO}} = A_d - I \cdot c_d^T \cdot A_d, \]

and  
\[ b_{\text{ESO}} = b_d - I \cdot c_d^T \cdot b_d. \]

In (9), \( A_d, b_d, \) and \( c_d^T \) are obtained from zero-order hold (ZOH) discretization with a sampling interval \( T \) of their respective counterparts in (2), as recommended by [8]:

\[ A_d = I + \sum_{i=1}^{\infty} \frac{A^i T^i}{i!}, \quad b_d = \left( \sum_{i=1}^{\infty} \frac{A^i T^i}{i!} \right) \cdot b, \quad c_d^T = e^T. \]

Controller (8) and observer (9) are shown as a block diagram in Fig. 2. Once again, one can recognize ADRC's signature ingredients: extended state observer (built around a simple plant model with input disturbance), disturbance rejection (unity-gain feedback of the state \( \hat{x}_{n+1} \)), and plant gain inversion (multiplication by \( 1/b_0 \) to obtain \( u(k) \)).

B. Controller Tuning

The outer state feedback controller of ADRC consists of a static feedback gain vector \( k^T \), which means that there are no dynamics to be discretized for the controller. Assuming that the sampling interval \( T \) is negligible compared to the dynamics of the closed control loop, the continuous-time design from Sect. II-B can be employed with very good accuracy for a discrete-time implementation, as well.

And indeed, to the best of the authors’ knowledge, this quasi-continuous controller design within a discrete-time linear ADRC implementation has, up to now, always been performed this way, see [29]–[34] for some examples. For applications with relatively large sampling intervals, however, the quasi-continuous design does not exactly lead to the desired closed-loop behavior. To round off ADRC controller tuning even for these edge cases, we therefore want to introduce discrete-time tuning rules for \( k^T \) in this article, which will be presented in the Sect. IV.

C. Observer Tuning

Looking back at (9), it becomes obvious that the abbreviation \( A_{\text{ESO}} \) is the system matrix of the discrete-time observer. Corresponding to the continuous-time case, we can use the pole placement approach of bandwidth parameterization for tuning the observer. Before doing so, we need to map the desired common pole from its \( s \)-plane to the \( z \)-domain location \( z_{\text{ESO}} \):

\[ \text{det} (z I - A_{\text{ESO}}) = (z - z_{\text{ESO}})^n+1 \]  

(10)

with  
\[ z_{\text{ESO}} = e^{-\omega_{\text{CL}} T}. \]

As shown before, for example in [8], [27], this results in the observer gains \( l_1 = 1 - z_{\text{ESO}}^2 \) and \( l_2 = \frac{1}{2} \cdot (1 - z_{\text{ESO}})^2 \) for the first-order case, and three gain values \( l_1 = 1 - z_{\text{ESO}}^3, l_2 = \frac{3}{2} \cdot (1 - z_{\text{ESO}})^2 \cdot (1 + z_{\text{ESO}}), l_3 = \frac{1}{3!} \cdot (1 - z_{\text{ESO}})^3 \) for second-order ADRC.

IV. DISCRETE-TIME CONTROLLER TUNING

A. Bandwidth Parameterization

In a discrete-time implementation of ADRC, all inputs and outputs are updated only once per sampling interval \( T \). The observer will, naturally, provide estimates for the outer state-feedback controller and the total disturbance only at these time instants, as well. A discrete-time design of the controller \( k^T \) therefore requires a discrete-time model of the virtual plant (3), as well. Assuming a zero-order hold behavior of the actuator, we can derive the following state-space model:

\[ \dot{\hat{x}}_{\text{VP}}(k+1) = A_{\text{VP},d} \cdot \hat{x}_{\text{VP}}(k) + b_{\text{VP},d} \cdot u(k), \]

\[ y_{\text{VP}}(k) = c_{\text{VP},d}^T \cdot \hat{x}_{\text{VP}}(k) \]  

(11)

with  
\[ A_{\text{VP},d} = I + \sum_{i=1}^{\infty} \frac{A^i T^i}{i!}, \]

\[ b_{\text{VP},d} = \left( \sum_{i=1}^{\infty} \frac{A^i T^i}{i!} \right) \cdot b_{\text{VP}}, \quad c_{\text{VP},d}^T = c_{\text{VP}}^T. \]

Similar the observer tuning procedure described in Sect. III-C, we can now apply pole placement with bandwidth parameterization for the discrete-time controller. Firstly, we map the desired location for all closed-loop poles from \( s \)- to \( z \)-domain:

\[ z_{\text{CL}} = e^{-\omega_{\text{CL}} T}, \]  

(12)

and then the design equation will read:

\[ \text{det} \left( z I - (A_{\text{VP},d} - b_{\text{VP},d} k^T) \right)^n = (z - z_{\text{CL}})^n. \]  

(13)

The most relevant cases for a practical implementation are \( n = 1 \) and \( n = 2 \), where ADRC will be used to replace PI and
PID controllers, respectively. For the first-order case, $A_{VP,d}$ and $b_{VP,d}$ are only scalars:

$$A_{VP,d} = (1) \quad \text{and} \quad b_{VP,d} = (T).$$

This means that (13) will be reduced to the following simple equation:

$$z + (Tk_1 - 1) = z - z_{CL},$$

from which we can easily obtain the discrete-time controller tuning equation for $k_1$ as:

$$k_1 = \frac{1 - z_{CL}}{T} \quad \quad (14)$$

In the second-order case, $A_{VP,d}$ and $b_{VP,d}$ are:

$$A_{VP,d} = \begin{pmatrix} 1 & T \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad b_{VP,d} = \frac{1}{2} T^2.$$

The design equation (13) now becomes:

$$z^2 + \left( \frac{k_1 T^2}{2} + k_2 T - 1 \right) z + \left( \frac{k_1 T^2}{2} - k_2 T + 1 \right)$$

$$= z^2 - 2z_{CL} \cdot z + z_{CL}^2,$$

which we can solve for the controller gains $k_1$ and $k_2$, obtaining:

$$k_1 = \frac{(1 - z_{CL})^2}{T^2} \quad \text{and} \quad k_2 = 4 - \frac{(1 + z_{CL})^2}{2T} \quad \quad (15)$$

B. Approximation for High Sampling Frequencies

For the common case of sufficiently high sampling frequencies, we now want to provide a connection between discrete- and continuous-time controller tuning for ADRC, specifically for the practically relevant cases $n = 1$ and $n = 2$.

Let us revisit (12), remembering the power series characterization of a real-valued exponential function. This allows to express $z_{CL}$ as:

$$z_{CL} = e^{-\omega_{CL} T}$$

$$= 1 + (-\omega_{CL} T) + \frac{(-\omega_{CL} T)^2}{2!} + \frac{(-\omega_{CL} T)^3}{3!} + \ldots,$$

which we can truncate after the second term if the sampling interval $T$ is small enough to warrant $\omega_{CL} T \ll 1$. Then we obtain the following approximation for $z_{CL}$:

$$z_{CL} \approx 1 - \omega_{CL} T \quad \quad (16)$$

For the first-order case, putting (16) in (14) gives:

$$k_1 = \frac{1 - z_{CL}}{T} \approx \frac{1 - (1 - \omega_{CL} T)}{T} = \omega_{CL}, \quad (17)$$

which corresponds to the continuous-time tuning value obtained by (5). In the same way, we obtain the following approximations for second-order ADRC:

$$k_1 = \frac{(1 - z_{CL})^2}{T^2} \approx \frac{(1 - (1 - \omega_{CL} T))^2}{T^2} = \omega_{CL}^2, \quad (18)$$

$$k_2 = \frac{4 - (1 + z_{CL})^2}{2T} \approx \frac{4 - (1 + (1 - \omega_{CL} T))^2}{2T} = \omega_{CL} \cdot \left( 2 - \frac{\omega_{CL} T}{2} \right) \approx 2\omega_{CL}. \quad (19)$$

We can now summarize the first contributions of this article:

- It can be confirmed that the established quasi-continuous controller tuning can indeed be applied to discrete-time ADRC provided that the condition $\omega_{CL} T \ll 1$ holds.
- The bandwidth parameterization tuning rules for the discrete-time controller derived in Sect. IV-A are a “backward compatible” generalization of the existing practice, delivering correct results regardless of the choice of the sampling interval $T$.

C. Comparison of Continuous- and Discrete-Time Tuning

In Sect. III-B we mentioned that the established way of controller tuning within discrete-time ADRC is to simply use the continuous-time described in Sect. II-B. The above-mentioned examples [29]–[34] obviously demonstrate as well that no problems are to be expected in most practical settings. So when will the dedicated discrete-time tuning just introduced in Sect. IV-A make a difference?

Obviously the sampling interval $T$ must be small enough compared to the dynamics of the closed loop. As already evident from Sect. IV-B, we will therefore let our analysis be based on the relative factor $\omega_{CL} T$. To give a visual impression of the difference between continuous-time controller tuning as given by (4) and a discrete-time design obtained through (13), the relative factor of continuous-time (CT) vs. discrete-time (DT) controller gains are shown in Fig. 3 for first- and second-order ADRC.

For $n = 1$ (shown in Fig. 3a), the relative factor is:

$$\frac{k_1^{CT}(\omega_{CL} T)}{k_1^{DT}(\omega_{CL} T)} = \frac{\omega_{CL}}{1 - \omega_{CL} T} = \frac{\omega_{CL} T}{1 - e^{-\omega_{CL} T}} \quad (19)$$

while for $n = 2$ (shown in Fig. 3b), the following two relations are obtained:

$$\frac{k_2^{CT}(\omega_{CL} T)}{k_2^{DT}(\omega_{CL} T)} = \frac{\omega_{CL}^2}{(1 - \omega_{CL} T)^2} = \frac{(\omega_{CL} T)^2}{(1 - e^{-\omega_{CL} T})^2} \quad (20)$$

From the visualization of (19) and (20) in Fig. 3 it becomes obvious that using a continuous-time design in a discrete-time ADRC implementation may lead to serious problems above $\omega_{CL} T \gtrsim 0.1$. The controller gains obtained from continuous-time design are then overestimating the actually needed values, leading to a higher bandwidth than designed—and being closer to possible instability of the loop.

Conversely, a simple rule of thumb can be learned from Fig. 3. For relative sampling intervals below $\omega_{CL} T \lesssim 0.05$, the closed-loop behaviors of a continuous- vs. discrete-time design will hardly be discernible.

Both findings are confirmed in a simple time-domain comparison of control loops with different discrete-time ADRC implementations using quasi-continuous and actual discrete-time controller designs in Fig. 4 and Fig. 5.
Fig. 3: Relation of controller gains obtained by continuous-time design (4) vs. discrete-time design (13) depending on the relative sampling interval $\omega_{CL} \cdot T$. As a practical rule of thumb, it can be said that below $\omega_{CL} \cdot T \leq 0.05$ both continuous- and discrete-time design become indistinguishable.

Fig. 4: Comparison of closed-loop step response and control signal for a control loop with a normalized, continuous-time plant $P(s) = \frac{1}{s+1}$ and first-order ADRC implementations tuned with $b_0 = 1$, $\omega_{CL} = 10$, and $k_{ESO} = 10$. For the discrete-time implementations with a “traditional” quasi-continuous controller design described in Sect. II-B, the actual bandwidth deviates from the true continuous-time case with increasing sampling intervals—the controller gains are too high. These simulations are idealized (no measurement noise, zero delay between measurement of $y$ and update of $u$) to focus the attention on the effects of increased sampling intervals.

Fig. 5: Comparison of Fig. 4 repeated with the discrete-time controller design introduced in Sect. IV-A, instead of the quasi-continuous design previously employed for discrete-time ADRC. Continuous- and discrete-time implementations now always exhibit similar closed-loop dynamics.
As a final side note it shall be mentioned that for \( \omega_{CL} T \geq 0.1 \) there will, of course, be other problems to tackle in the control loop besides overestimated controller gains. With such large sampling intervals, any delay unaccounted for will quickly lead to oscillations or even instability.

V. Discrete-Time Transfer Function Implementations

In this section, we reiterate the derivation of two discrete-time transfer function implementations of ADRC introduced in [17] and [18], with slight variations that lead to easier understanding and simpler tuning equations. In contrast to previous work, however, we now give detailed coefficients for all transfer functions based on the discrete-time controller tuning rules developed in Sect. IV-A. This will also form the basis of introducing new discrete-time implementations of error-based ADRC in Sect. VI.

A. Transfer Function Implementation

The first step towards a discrete-time transfer function representation of the state-space implementation is made by \( z \)-transform of both controller (8) and observer (9):

\[
u(z) = \frac{1}{b_0} \cdot \left( k_1 \cdot r(z) - (k^T 1) \cdot \hat{x}(z) \right), \quad (21)
\]

\[
\hat{x}(z) = z^{-1} \cdot A_{ESO} \cdot \hat{x}(z) + z^{-1} \cdot b_{ESO} \cdot u_{lim}(z) + I \cdot y(z). \quad (22)
\]

Putting (21) in (22) leads to the closed-loop dynamics of the discrete-time observer. To be able to do that, we sacrifice the separate feedback path of the limited control signal \( u_{lim} \) back to the observer, as this feedback loop will now be "convoluted" in the closed-loop observer dynamics. With \( u(z) = u_{lim}(z) \) one obtains:

\[
\hat{x}(z) = \Phi_{ESO} \cdot \left( z^{-1} \cdot \frac{k_1}{b_0} \cdot b_{ESO} \cdot r(z) + I \cdot y(z) \right). \quad (23)
\]

In (23), we introduced the matrix \( \Phi_{ESO} \) as a shorthand notation:

\[
\Phi_{ESO} = \left( I - z^{-1} \cdot \left( A_{ESO} - \frac{1}{b_0} \cdot b_{ESO} \cdot (k^T 1) \right) \right)^{-1}. \quad (24)
\]

Transfer functions will relate the reference signal and the measured plant output to the updated controller output. The vector of estimated state variables \( \hat{x}(z) \) therefore needs to be eliminated. That can be achieved by putting (23) back into (21). The control law can now be written as:

\[
u(z) = \frac{k_1}{b_0} \cdot \left( 1 - z^{-1} \cdot (k^T 1) \cdot \Phi_{ESO} \cdot \frac{1}{b_0} \cdot b_{ESO} \right) \cdot r(z) \]

\[
- \left( \frac{1}{b_0} \cdot (k^T 1) \cdot \Phi_{ESO} \cdot I \right) \cdot y(z). \quad (25)
\]

Equation (25) obviously represents a two-degrees-of-freedom control structure, albeit not a typical (or desirable) one. A discrete-time controller structure often encountered in practice is shown in Fig. 6 and consists of a feedback controller \( C_{FB}(z) \) and a prefilter \( C_{PF}(z) \) as follows:

\[
u(z) = C_{FB}(z) \cdot \left( C_{PF}(z) \cdot r(z) - y(z) \right). \quad (26)
\]

This structure is also shown in Fig. 6. We can transform (25) into that form by factoring out the feedback transfer function. Now we can read off the transfer functions of both feedback controller \( C_{FB}(z) \) and prefilter \( C_{PF}(z) \) as follows:

\[
C_{FB}(z) = \frac{1}{b_0} \cdot \left( k^T 1 \right) \cdot \Phi_{ESO} \cdot I, \quad (27)
\]

\[
C_{PF}(z) = \frac{k_1}{b_0} \cdot \left( 1 - z^{-1} \cdot (k^T 1) \cdot \Phi_{ESO} \cdot \frac{1}{b_0} \cdot b_{ESO} \right) \cdot \left( k^T 1 \right) \cdot \Phi_{ESO} \cdot I. \quad (28)
\]

As its counterpart in the continuous time domain, the discrete-time feedback controller transfer function \( C_{FB}(z) \) includes an integrator—otherwise the control loop would not be able to reject constant disturbances and follow constant reference signal values with zero steady-state error, obviously. In [17] it was put forward to factor this pole at \( z = 1 \) out of \( C_{FB}(z) \):

\[
C_{FB}(z) = \Delta C_{FB}(z) \cdot \frac{1}{1 - z^{-1}}. \]

Implementing \( C_{FB}(z) \) as a series connection of an accumulator and a discrete-time filter comes with the following advantages:

- One dependent coefficient of the denominator polynomial is removed, which means there is one coefficient less to compute and to store in an implementation. This also saves one multiplication at runtime.
- The \( \Delta C_{FB}(z) \) part of the feedback controller now exhibits the incremental [35] or velocity [36] form of a digital controller. Implementing the accumulator with clamping therefore provides an easy but quite effective protection against integrator windup.

This is also the reason why (25) had to be brought into the form (26)—otherwise there would be a second integrator, namely in the transfer function from \( r \) to \( u \). For the overall behavior, these two structures are equivalent in a mathematical sense, but having two integrators (one between \( r \) and \( u \) and one between \( y \) and \( u \)) would result in two integrator states running away to infinity for any nonzero reference \( r \).

Putting everything together and applying discrete-time observer and controller tuning as described in Sect. III-C and Sect. IV-A, one obtains the following two transfer functions.

![Fig. 6: Discrete-time transfer function implementation of ADRC](image-url)
TABLE I: Discrete-time transfer function implementation of first-order ADRC: coefficients of the transfer functions $C_{FB}(z)$ and $C_{PF}(z)$ from (29) and (30), obtained by discrete-time pole placement and bandwidth parameterization with $z_{CL} = e^{-\omega_{CL}T}$ (closed loop) and $z_{ESO} = e^{-z_{ESO}T}$ (observer).

| Coefficient | Value for first-order ADRC |
|-------------|----------------------------|
| $\alpha_1$  | $-z_{CL}^2z_{ESO}$         |
| $\beta_0$   | $\frac{1}{b_0T} \left[ 2z_{CL}^2z_{ESO} - 2z_{ESO} - z_{CL} + 1 \right]$ |
| $\beta_1$   | $\frac{1}{b_0T} \left[ 2z_{CL}z_{ESO} - 2z_{CL}z_{ESO}^2 + z_{ESO} - 1 \right]$ |
| $\gamma_0$  | $\frac{1 - z_{CL}}{b_0T}$   |
| $\gamma_1$  | $\frac{-2z_{ESO} \cdot (1 - z_{CL})}{b_0T}$ |
| $\gamma_2$  | $\frac{z_{ESO}^2 \cdot (1 - z_{CL})}{b_0T}$ |

$C_{FB}(z)$ and $C_{PF}(z)$:

$$C_{FB}(z) = \frac{\sum_{i=0}^{n} \beta_i z^{-i}}{1 + \sum_{i=1}^{n} \alpha_i z^{-i}} \cdot \frac{1}{1 - z^{-1}}, \quad (29)$$

$$C_{PF}(z) = \frac{\sum_{i=0}^{n} \beta_i z^{-i}}{1 + \frac{1}{b_0} \sum_{i=1}^{n} \beta_i z^{-i}}. \quad (30)$$

For the first- and second-order case of ADRC, which are particularly relevant in practice, we provide a detailed set of equations for the $\alpha$, $\beta$, $\gamma$ coefficients based on the continuous-time tuning parameters $\omega_{CL}$ (desired closed-loop bandwidth) and $k_{ESO}$ (relative observer bandwidth factor) in Table I and Table II, respectively. These results are—in the sense of Sect. IV-C—a more accurate generalization of the coefficients previously reported in [17], which used a quasi-continuous design for the controller gains. In (30) we factored out $\frac{1}{b_0}$, which allows to reuse the $\beta$ coefficients for $C_{PF}(z)$ and give more compact tuning equations for the $\gamma$ coefficients.

Note that this discrete-time transfer function implementation does not fully replicate the behavior of its state-space ancestor. On the other hand, transfer functions can be implemented more efficiently, as their number of coefficients grows only linearly with the order of the controller $n$, as opposed to quadratic growth in the state-space case. The price for obtaining a very familiar structure with prefilter and feedback controller was to eliminate the separate feedback path of the limited controller output $u_{lim}$. This results in loss of the “out-of-the-box” windup protection coming with ADRC—a loss that can be partially compensated by implementation of a clamped integrator as part of $C_{FB}(z)$. Luckily, it is not always an either/or situation. In the second transfer function approach, that we will introduce in Sect. V-B, state-space features will be combined with transfer function efficiency.

B. Dual-Feedback Transfer Function Implementation

A different transfer function implementation that preserves the feedback path of the limited control signal value $u_{lim}(z)$ was introduced in [18]. The starting point of Sect. V-A is being used here, as well: the control law (21) and observer (22) in $z$-domain. The latter is being solved for $\dot{x}(z)$:

$$\dot{x}(z) = \left( I - z^{-1}A_{ESO} \right)^{-1} \cdot \left( z^{-1}B_{ESO} \cdot u_{lim}(z) + I \cdot y(z) \right).$$

Putting this equation back in (21) yields a control law depicted in Fig. 7. It features three inputs $r$, $y$, and $u_{lim}$, as in the state-space case:

$$u(z) = \frac{1}{b_0} \cdot \left( k_1 \cdot r(z) - \left( k^T \cdot 1 \right) \cdot \left( I - z^{-1}A_{ESO} \right)^{-1} \cdot \left( z^{-1}B_{ESO} \cdot u_{lim}(z) + I \cdot y(z) \right) \right)$$

$$= \frac{k_1}{b_0} \cdot r(z) - C_{FB}(z) \cdot y(z) + C_{PF}(z) \cdot u_{lim}(z). \quad (31)$$

The two feedback transfer functions $C_{FB}(z)$ and $C_{PF}(z)$ in (31) share the same denominator and can be expressed as given in (32) and (33).

$$C_{FB}(z) = \frac{1}{b_0} \cdot \left( k^T \cdot 1 \right) \cdot \left( I - z^{-1}A_{ESO} \right)^{-1} \cdot I$$

$$= \frac{\sum_{i=0}^{n} \beta_i z^{-i}}{1 + \sum_{i=1}^{n} \alpha_i z^{-i}} \quad (32)$$

$$C_{PF}(z) = -\frac{z^{-1}}{b_0} \cdot \left( k^T \cdot 1 \right) \cdot \left( I - z^{-1}A_{ESO} \right)^{-1} \cdot b_{ESO}$$

$$= \frac{\sum_{i=0}^{n} \gamma_i z^{-i}}{1 + \sum_{i=1}^{n} \alpha_i z^{-i}} \quad (33)$$

The transfer function coefficients are computed by inserting properly tuned controller gains $k^T$ and observer gains $I$ in
TABLE II: Discrete-time transfer function implementation of second-order ADRC: coefficients of the transfer functions $C_{FB}(z)$ and $C_{PF}(z)$ from (29) and (30), obtained by discrete-time pole placement and bandwidth parameterization with $z_{CL} = e^{-j\omega_{CL}T}$ (closed loop) and $z_{ESO} = e^{-k_{ESO}j\omega_{CL}T}$ (observer). Underlying system and tuning parameters are $b_0$ (plant gain), $T$ (sampling interval), $\omega_{CL}$ (closed-loop bandwidth), $k_{ESO}$ (observer bandwidth factor).

| Coefficient | Value for second-order ADRC |
|-------------|-----------------------------|
| $\alpha_1$  | $-\frac{1}{8} \cdot (1 + z_{CL})^2 \cdot (1 + z_{ESO}) + z_{CL}^2 z_{ESO} + 1$ |
| $\alpha_2$  | $z_{CL}^2 z_{ESO}$ |
| $\beta_0$   | $\frac{1}{b_0 T^2} \cdot \left[ \frac{1}{4} \cdot (1 + z_{CL})^2 \cdot (1 + z_{ESO})^3 - 2 \cdot (z_{CL}^2 z_{ESO} + 2 z_{CL} + 3 z_{ESO} - 2) \right]$ |
| $\beta_1$   | $\frac{1}{b_0 T^2} \cdot \left[ -(1 + z_{CL})^2 \cdot (1 + z_{ESO})^3 \cdot (1 + z_{CL})^2 + 6 \cdot (z_{CL}^2 z_{ESO} + 2 z_{CL} z_{ESO} + z_{ESO}^2 + z_{CL} + 2 + z_{ESO} - 1) \right]$ |
| $\beta_2$   | $\frac{1}{b_0 T^2} \cdot \left[ \frac{1}{4} \cdot (1 + z_{CL})^2 \cdot (1 + z_{ESO})^3 + 2 \cdot \left( -2 z_{CL}^2 z_{ESO} + 3 z_{CL}^2 z_{ESO} + 2 z_{CL} z_{ESO}^2 + z_{CL} z_{ESO} + 1 \right) \right]$ |
| $\gamma_0$  | $\frac{(1 - z_{CL})^2}{b_0 T^2}$ |
| $\gamma_1$  | $\frac{-3 z_{ESO} \cdot (1 - z_{CL})^2}{b_0 T^2}$ |
| $\gamma_2$  | $\frac{3 z_{ESO} \cdot (1 - z_{CL})^2}{b_0 T^2}$ |
| $\gamma_3$  | $\frac{-z_{ESO} \cdot (1 - z_{CL})^2}{b_0 T^2}$ |

(31) and solving for $\alpha$, $\beta$, and $\gamma$. Reference [18] gave tuning equations for these coefficients using bandwidth parameterization with the quasi-continuous controller design of Sect. II-B. In this paper, we provide these coefficients for first- and second-order ADRC incorporating the discrete-time controller tuning of Sect. IV-A, now providing accurate results also for larger sampling intervals. They can be found in Table III and Table IV. Note that the $\beta$ coefficients are the same as in Sect. V-A, i.e. the numerator polynomial of $C_{FB}(z)$ in (32) and $C_{FB}(z)$ in (29) are identical.

A very efficient implementation was proposed in [18], exploiting the common denominator of $C_{FB}(z)$ and $C_{PF}(z)$ and choosing a transposed ‘direct form II’ realization. This reduces the number of required storage variables during runtime to the minimum, and uses even less algebraic operations than the transfer function representation of Sect. V-A—while keeping the flexibility and behavior of the state-space implementation regarding user-defined control signal limitation and windup protection. For the first- and second-order case, the resulting controller structure is shown in Fig. 12a and Fig. 12b.

VI. ERROR-BASED ADRC

Error-based ADRC is a variant presented in generalized form in [25]. It obtained its name from feeding the control error $e = r - y$ to the observer—instead of the plant output $y$ in the “classical” case like the one shown in Sect. II. To better differentiate the two approaches, we will accordingly denote the latter as “output-based” ADRC in this section.

The relation between output- and error-based ADRC was firmly established for the continuous-time case in [24]. Building on that, we want to extend the connections between output- and error-based ADRC to the discrete time domain in this article. The only discrete-time implementation of error-based ADRC so far in [25] was based on a bilinear transformation from its continuous-time transfer function representation. As we will see in this section, we can fully build on existing discrete-time ADRC structures instead. In doing so, we will derive new, equivalent variants of discrete-time error-based ADRC that match the three implementations covered in this paper: state-space, transfer function, and dual-feedback transfer implementation: coefficients for first-order ADRC, obtained by discrete-time pole placement and bandwidth parameterization with $z_{CL} = e^{-j\omega_{CL}T}$ (closed loop) and $z_{ESO} = e^{-k_{ESO}j\omega_{CL}T}$ (observer).

| Coefficient | Value for first-order ADRC |
|-------------|-----------------------------|
| $\alpha_1$  | $-2 z_{ESO}$ |
| $\alpha_2$  | $z_{ESO}^2$ |
| $\beta_0$   | $\frac{1}{b_0 T} \cdot \left[ z_{CL} z_{ESO}^2 - 2 z_{CL} z_{ESO}^2 + z_{CL}^2 + 2 \right]$ |
| $\beta_1$   | $\frac{1}{b_0 T} \cdot \left[ 2 z_{CL} z_{ESO}^2 - z_{CL} z_{ESO}^2 + z_{CL}^2 + 2 + z_{ESO} - 1 \right]$ |
| $\gamma_0$  | $z_{CL} z_{ESO}^2 - 2 z_{ESO} + 1$ |
| $\gamma_1$  | $z_{ESO}^2 - z_{CL} z_{ESO}^2$ |
| $k_1$       | $\frac{1 - z_{CL}}{b_0 T}$ |
TABLE IV: Discrete-time dual-feedback transfer function implementation: coefficients for second-order ADRC, obtained by discrete-time pole placement and bandwidth parameterization with $z_{CL} = e^{-\omega_{CL}T}$ (closed loop) and $z_{ESO} = e^{-k_{ESO} \omega_{CL}T}$ (observer)

| Coefficient | Value for second-order ADRC |
|-------------|-----------------------------|
| $a_1$       | $-3z_{ESO}$                 |
| $a_2$       | $3z_{ESO}^2$                |
| $a_3$       | $-z_{ESO}^3$                |
| $\beta_0$   | $\frac{1}{b_0T^2} \cdot \left[ \frac{1}{4} \cdot (1 + z_{CL})^2 \cdot (1 + z_{ESO})^3 - 2 \cdot \left( z_{CL}^2 z_{ESO}^3 + 2z_{CL} + 3z_{ESO} - 2 \right) \right]$ |
| $\beta_1$   | $\frac{1}{b_0T^2} \cdot \left[ -(1 + z_{CL})^2 \cdot (1 + z_{ESO})^3 + 2 \cdot (1 + z_{CL})^2 + 6 \cdot \left( z_{CL}^2 z_{ESO}^3 + 2z_{CL} z_{ESO} + z_{ESO}^2 + z_{ESO} - 1 \right) \right]$ |
| $\beta_2$   | $\frac{1}{b_0T^2} \cdot \left[ -\frac{1}{4} \cdot (1 + z_{CL})^2 \cdot (1 + z_{ESO})^3 + 2 \cdot \left( -2z_{CL}^2 z_{ESO}^3 + 2z_{CL} z_{ESO}^3 + 2z_{CL} + 3z_{ESO} + 1 \right) \right]$ |
| $\gamma_0$  | $\frac{1}{8} \cdot (1 + z_{CL})^2 \cdot (1 + z_{ESO})^3 - z_{ESO} \cdot (z_{CL}^2 z_{ESO} + 3)$ |
| $\gamma_1$  | $-\frac{1}{8} \cdot (1 + z_{CL})^2 \cdot (1 + z_{ESO})^3 + 3z_{ESO}^2 + 1$ |
| $\gamma_2$  | $z_{ESO}^3 \cdot (z_{CL}^2 - 1)$ |
| $k_i$       | $\frac{(1 - z_{CL})^2}{b_0T^2}$ |

![Fig. 8: Structure of a control loop with error-based ADRC (continuous-time state-space version) and a user-defined control signal limiter](image)

For an in-depth introduction to error-based ADRC, we refer to [25]. A continuous-time block diagram of error-based ADRC is shown in Fig. 8. In contrast to [25], we explicitly are feeding the limited control signal $u_{lim}(t)$ into observer, to provide easy windup protection as in the output-based case in Sect. II-A. The associated control law and observer dynamics are:

$$u(t) = \frac{1}{b_0} \cdot (k^T \ 1) \cdot \hat{x}(t),$$  \hspace{1cm} (34)

$$\dot{x}(t) = A \cdot \hat{x}(t) - b \cdot u_{lim}(t) + l \cdot (e(t) - c^T \cdot \hat{x}(t)).$$  \hspace{1cm} (35)

Not only the structure of these equations and the block diagram are similar to output-based ADRC: In the observer (35), $A$, $b$, and $c^T$ are exactly the same as in (2). This applies to the controller gains $k^T$ and observer gains $l$, as well, which means that all existing tuning methods can be used here without modification.

**B. Discrete-Time State-Space Implementation**

To derive the discrete-time state-space version of error-based ADRC, we can largely build on the results from output-based ADRC in Sect. III-A. The control law (34) is a static state feedback, and hence trivial to translate to the discrete time domain:

$$u(k) = \frac{1}{b_0} \cdot (k^T \ 1) \cdot \hat{x}(k).$$  \hspace{1cm} (36)

We only have to derive a discretized observer from (35). By visual inspection of Fig. 1 and Fig. 8, only two differences between the observers can be found:

- For error-based ADRC, the input $e(t)$ is being used instead of $y(t)$, and
- the limited controller output $u_{lim}(t)$ enters the observer with a negative sign.

This enables us to take a very convenient shortcut. We can simply reuse the discrete-time observer equation (9) from Sect. III-A, exchange $y(k)$ for $e(k)$, and change the sign.
for $u_{\text{lim}}(k)$. The properly discretized observer for error-based ADRC now reads:

$$\hat{x}(k) = A_{\text{ESO}} \cdot \hat{x}(k-1) - b_{\text{ESO}} \cdot u_{\text{lim}}(k-1) + l \cdot e(k),$$ (37)

with $A_{\text{ESO}}$ and $b_{\text{ESO}}$ as well as controller/observer gains being the same as in the output-based case. The block diagram shown in Fig. 9 is, consequently, also very similar to Fig. 2.

Please remember that even though this derivation could be presented in an extremely condensed form, it is the first discrete-time implementation of error-based ADRC in state space. Owing to the equivalence to output-based ADRC at the core observer dynamics, all findings of [8] regarding the best discretization strategy for linear ADRC automatically apply here as well (zero-order hold discretization in conjunction with the current observer approach).

This also means that there is no need for a “custom” discretization of error-based ADRC at transfer function level, as done in [25]—we can instead follow the same route of exact discretization representations as presented in Sect. V of this article, and will do so right next.

C. Discrete-Time Transfer Function Implementation

As the counterpart of the output-based version in Sect. V-A, a transfer function implementation of discrete-time error-based ADRC can now easily be derived by following the same steps. We once again start with the $z$-domain representation of control law (36) and observer (37), and use $u(z) = u_{\text{lim}}(z)$ as an observer input, as we will lose the dedicated feedback path of the limited controller output:

$$u(z) = \frac{1}{b_0} \cdot (k^T 1) \cdot \hat{x}(z),$$ (38)

$$\hat{x}(z) = z^{-1} \cdot A_{\text{ESO}} \cdot \hat{x}(z) - z^{-1} \cdot b_{\text{ESO}} \cdot u(z) + l \cdot e(z).$$ (39)

The closed-loop observer equation is obtained by putting (38) into (39):

$$\hat{x}(z) = \Phi_{\text{ESO}} \cdot l \cdot e(z),$$

where $\Phi_{\text{ESO}}$ is identical to (24) in the output-based case. Inserting this equation back into (38) yields the control law for the transfer function implementation of error-based ADRC, which is also shown in Fig. 10:

$$u(z) = \frac{1}{b_0} \cdot (k^T 1) \cdot \Phi_{\text{ESO}} \cdot l \cdot e(z).$$ (40)

Equivalent to the continuous-time derivation in [24], only one transfer function is required here. And, above all, exactly the same transfer function $C_{\text{FB}}(z)$ given in (29) is being used for output- and error-based ADRC, with the same $\alpha$ and $\beta$ coefficients from Table I and Table II.

Comparing control laws and block diagrams of discrete-time output- and error-based ADRC in Fig. 6 and Fig. 10, it is clear that output-based ADRC can be seen as an extension of error-based ADRC, with an additional reference signal prefilter $C_{\text{PF}}(z)$—as in the continuous-time case [24].

In output-based ADRC, the dynamics of reference tracking and disturbance rejection cannot be independently chosen, the prefilter is tuned for a critically damped closed-loop response. Being a more “bare-bones” version, error-based ADRC therefore allows to be more flexibly combined with an independently tuned prefilter for an actual two-degrees-of-freedom design.

D. Discrete-Time Dual-Feedback Transfer Function Implementation

With the discrete-time state-space implementation of error-based ADRC in Sect. VI-B at hand, a dual-feedback transfer function variant can easily be derived, as well. The $z$-transform of the observer (37),

$$\hat{x}(z) = z^{-1} \cdot A_{\text{ESO}} \cdot \hat{x}(z) - z^{-1} \cdot b_{\text{ESO}} \cdot u_{\text{lim}}(z) + l \cdot e(z),$$

is being solved for $\hat{x}(z)$, yielding:

$$\hat{x}(z) = \left( I - z^{-1} A_{\text{ESO}} \right)^{-1} \cdot \left( -z^{-1} \cdot b_{\text{ESO}} \cdot u_{\text{lim}}(z) + l \cdot e(z) \right),$$

for the transfer function implementation of error-based ADRC, dual-feedback transfer function implementation

Fig. 9: Control loop with discrete-time error-based ADRC, state space implementation

Fig. 10: Control loop with discrete-time error-based ADRC, transfer function implementation

Fig. 11: Control loop with discrete-time error-based ADRC, dual-feedback transfer function implementation
which can be inserted into the control law (38):

\[
u(z) = \frac{1}{b_0} \cdot (k^T 1) \cdot (I - z^{-1} A_{ESO})^{-1} \cdot (-z^{-1} \cdot b_{ESO} \cdot u_{\text{lim}}(z) + I \cdot e(z)).
\]

By inspection and comparison with the output-based counterpart (31), we can make the important observation that (41) can be represented using the two already introduced transfer functions \( C_{FBY}(z) \) and \( C_{FBu}(z) \):

\[
u(z) = C_{FBY}(z) \cdot e(z) + C_{FBu}(z) \cdot u_{\text{lim}}(z),
\]

where \( C_{FBY}(z) \) and \( C_{FBu}(z) \) are identical to their definition in (32) and (33). The required \( \alpha, \beta, \gamma \) coefficients can, consequently, be reused in unmodified form as given in Table III and Table IV for the first- and second-order case.

Please note that we kept the subscript “y” as part of \( C_{FBY}(z) \) in the control law (42) and its visualization in Fig. 11 to emphasize that exactly the same transfer functions from the output-based variant are being reused here, even though \( C_{FBY}(z) \) is now being fed with the control error \( e \)—which, in turn, of course also contains \( y \).

As a final step, a very efficient implementation of this control law can be found as well using the ideas from [18]. With the common denominator of \( C_{FBu}(z) \) and \( C_{FBY}(z) \), the control law (42) can be written as:

\[
u(z) = \frac{\sum_{i=0}^{n} \beta_i z^{-i}}{1 + \sum_{i=0}^{n+1} \alpha_i z^{-i}} \cdot u_{\text{lim}}(z) + \frac{\sum_{i=0}^{n} \gamma_i z^{-i}}{1 + \sum_{i=0}^{n+1} \alpha_i z^{-i}} \cdot e(z).
\]

A direct form II implementation of (43) reduces both storage variables and algebraic operations required during runtime to a minimum. The resulting implementations are shown in Fig. 12c and Fig. 12d for the first- and second-order case, allowing a direct comparison with their output-based counterparts.

VII. Example

In this section, a power electronics application (DC-DC converter) to be controlled by first-order output- and error-based ADRC is being studied. As we developed error-based ADRC implementations in this paper that exhibit exactly the same dynamic disturbance rejection behavior as their output-based counterparts, the equivalence between output- and error-based ADRC in this regard should therefore be visible. Furthermore, the anti-windup abilities shall be demonstrated that error-based ADRC gained in the implementations introduced in this paper by keeping the separate feedback path for the limited controller output in both the state-space and the dual-feedback variant. For comparability down to the level of individual sampling intervals, we will perform this comparison in a simulation environment.

An overview of the DC-DC converter and its control structure with error-based ADRC is shown in Fig. 13. The converter parameters are: \( V_{\text{Input}} = 24 \text{ V} \), \( L = 33 \mu \text{H} \), \( C = 100 \mu \text{F} \), base load \( R = 100 \Omega \), \( f_{\text{Switch}} = 50 \text{ kHz} = \frac{1}{T} \), peak-current mode slope compensation of 2 A per cycle. To limit not

1For output-based ADRC, this is a replication of the example in [18].
only the absolute values, but also the slope of the converter current during transients, the voltage controller is equipped with a magnitude and rate limitation block configured for peak current commands from the interval [0 A, 6 A] with a rate limit of ±20 A/ms. To dynamically decouple reference tracking from disturbance rejection, a first-order low-pass setpoint filter is added, with a time constant of 750 μs for a 98% settling time of 3 ms. ADRC tuning is performed with a bandwidth $\omega_{cl} = 4000$ rad/s for an 98% settling time of 1 ms, and a relative observer bandwidth $k_{eso} = 5$. As shown in [10], plant modeling for ADRC is extremely easy in this application, depending only on the output capacitor with $b_0 = \frac{1}{c} = 10000$ V/As. The sampling frequency of the discrete-time implementation corresponds to the switching frequency. To be closer to practice, measurement noise with $\sigma = 20$ mV is added to the output voltage, and a full sampling interval assumed as the delay between measurement and updated controller output.

The scenario consists of startup at (almost) no load, a jump to full load at $t = 4$ ms, a setpoint change at $t = 6$ ms, and a temporary overload condition between $t = 10$ ms and $t = 12$ ms—the latter to deliberately drive the controller into saturation. The simulation is performed twice: once with error-based ADRC (including the setpoint filter), and once with output-based ADRC (without setpoint filter). Results are presented in Fig. 14.

The simulation results confirm that the discrete-time error-based implementations introduced in this article deliver identical disturbance rejection behavior as their output-based counterparts, down to the scope of individual samples. As an additional plus, they can now benefit from the same windup protection options.

VIII. Conclusion

Practical implementations of ADRC are usually carried out in software, i.e. in a discrete time domain. This article provided both a summary and an extension of discrete-time implementations of linear ADRC. As a first contribution, controller
tuning was performed using bandwidth parameterization with discrete-time pole placement, departing from the prevalent quasi-continuous design procedure. A bridge to the existing tuning rules could be built, while also demonstrating the necessity of accurate results for low sampling frequencies. For two existing discrete-time transfer function implementations of ADRC, coefficients were rederived based on accurate discrete-time pole placement.

A second bridge was built from “classical” (output-based) to error-based ADRC in the discrete time domain. It was shown that existing implementations need only slight restructuring, and transfer functions, coefficients, tuning rules can all be reused for error-based ADRC. In one sweep, three new error-based ADRC implementations could be derived in this article, which are all direct counterparts of existing output-based variants. Table V summarizes all variants presented here along with some main characteristics, to assist with making a choice. Error- and output-based implementations now exhibit exactly the same inner-loop behavior (disturbance rejection), rendering custom discrete-time implementations of error-based ADRC—as done e.g. in [25]—unnecessary. As an important feature for real-world applications, error- and output-based ADRC can now be equipped with the same control signal limitation and windup protection capabilities.

Be it the necessity of a true two-degrees-of-freedom design, windup protection, a low footprint during runtime, or structural similarity to solutions of classical digital control—the discrete-time ADRC variants introduced and discussed in this article can now cater a larger variety of practical needs than ever. What a great time to put ADRC into practice!

**TABLE V: Overview of discrete-time implementations for output-based and error-based ADRC presented in this article**

| Implementation | Output-based | Error-based | Characteristics |
|----------------|--------------|-------------|-----------------|
| State space    | Sect. III-A  | Sect. VI-B  | Discrete-time equivalent of the original structure of linear continuous-time ADRC. Flexible control signal limitation and windup protection. Runtime: Minimum number of storage variables, maximum number of coefficients and operations. |
| Transfer function | Sect. V-A | Sect. VI-C | Strong similarity to classical digital control using feedback controller and prefilter. Control signal limitation only via clamped integrator. Runtime: Maximum number of storage variables, reduced number of coefficients and operations. |
| Dual-feedback TF | Sect. V-B | Sect. VI-D | Full replication of the state-space behavior using transfer functions, but with uncommon structure. Flexible control signal limitation and windup protection. Runtime: Minimum number of storage variables, coefficients, and operations. |

**REFERENCES**

[1] J. Han, “From PID to active disturbance rejection control,” *IEEE Transactions on Industrial Electronics*, vol. 56, no. 3, pp. 900–906, 2009.

[2] Q. Zheng and Z. Gao, “On practical applications of active disturbance rejection control,” in *Proceedings of the 29th Chinese Control Conference*, 2010, pp. 6095–6100.

[3] ——, “Active disturbance rejection control: Some recent experimental and industrial case studies,” *Control Theory and Technology*, vol. 16, no. 4, pp. 301–313, 2018.

[4] S. E. Talole, “Active disturbance rejection control: Applications in aerospace,” *Control Theory and Technology*, vol. 16, pp. 314–323, 2018.

[5] R. Fareh, S. Khadraoui, M. Y. Abdallah, M. Baziyad, and M. Bettaieb, “Active disturbance rejection control for robotic systems: A review,” *Mechatronics*, vol. 80, p. 102671, 2021.

[6] Z. Gao, “Active disturbance rejection control: A paradigm shift in feedback control system design,” in *Proceedings of the 2006 American Control Conference*, 2006, pp. 2399–2405.

[7] ——, “Scaling and bandwidth-parameterization based controller tuning,” in *Proceedings of the 2003 American Control Conference*, 2003, pp. 4989–4996.

[8] R. Miklosovic, A. Radke, and Z. Gao, “Discrete implementation and generalization of the extended state observer,” in *Proceedings of the 2006 American Control Conference*, 2006, pp. 2209–2214.

[9] R. Madonski, Z. Gao, and K. Lakomy, “Towards a turnkey solution of industrial control under the active disturbance rejection paradigm,” in *2015 54th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, 2015, pp. 616–621.

[10] G. Herbst, “Practical active disturbance rejection control: Bumpless transfer, rate limitation, and incremental algorithm,” *IEEE Transactions on Industrial Electronics*, vol. 63, no. 3, pp. 1754–1762, 2016.

[11] R. Madonski and P. Herman, “Survey on methods of increasing the efficiency of extended state disturbance observers,” *ISA Transactions*, vol. 56, pp. 18–27, 2015.

[12] C. Fu and W. Tan, “Tuning of linear ADRC with known plant information,” *ISA Transactions*, vol. 65, pp. 384–393, 2016.

[13] R. Zhou and W. Tan, “Analysis and tuning of general linear active disturbance rejection controllers,” *IEEE Transactions on Industrial Electronics*, vol. 66, no. 7, pp. 5497–5507, 2019.

[14] G. Tian and Z. Gao, “Frequency response analysis of active disturbance rejection based control system,” in *2007 IEEE International Conference on Control Applications*, 2007, pp. 1595–1599.

[15] C. Huang and Z. Gao, “On transfer function representation and frequency response of linear active disturbance rejection control,” in *Proceedings of the 32th Chinese Control Conference*, 2013, pp. 72–77.

[16] Q. Zheng and Z. Gao, “Active disturbance rejection control: Between the formulation in time and the understanding in frequency,” *Control Theory and Technology*, vol. 14, no. 3, pp. 250–259, 2016.

[17] G. Herbst, “Transfer function analysis and implementation of active disturbance rejection control,” *Control Theory and Technology*, vol. 19, pp. 19–34, 2021.

[18] ——, “A minimum-footprint implementation of discrete-time ADRC,” in *2021 European Control Conference (ECC)*, 2021, pp. 107–112.

[19] M. M. Michalek, “Robust trajectory following without availability of the reference time-derivatives in the control scheme with active disturbance rejection,” in *Proceedings of the 2016 American Control Conference*, 2016, pp. 1536–1541.

[20] A. Mandali, L. Dong, and A. Morinec, “Robust controller design for automatic voltage regulation,” in *Proceedings of the 2020 American Control Conference*, 2020, pp. 2617–2622.

[21] T. E. Lechekhab, S. M. Manojlović, M. R. Stanković, R. Madonski, and S. M. Simić, “Robust error-based active disturbance rejection control of a quadrotor,” *Aircraft Engineering and Aerospace Technology*, vol. 89, no. 1, pp. 89–104, 2021.

[22] R. Madonski, K. Lakomy, and J. Yang, “Simplifying ADRC design with error-based framework: Case study of a DC-DC buck power converter,” *Control Theory and Technology*, vol. 19, pp. 94–112, 2021.

[23] T. Huang, G. Hu, Y. Yan, D. Zeng, and Z. Meng, “Combined feedforward and error-based active disturbance rejection control for diesel particulate filter thermal regeneration,” *ISA Transactions*, 2022.

[24] R. Madonski, G. Herbst, and M. R. Stanković, “ADRC in output and error form: Connection, equivalence, performance,” *Control Theory and Technology*, vol. 21, pp. 56–71, 2022.

[25] R. Madonski, S. Shao, H. Zhang, Z. Gao, J. Yang, and S. Li, “General error-based active disturbance rejection control for swift industrial implementations,” *Control Engineering Practice*, vol. 84, pp. 218–229, 2019.
[26] K. J. Åström and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*, 2nd ed. Princeton, NJ, USA: Princeton University Press, 2021.

[27] G. Herbst, “A simulative study on active disturbance rejection control (ADRC) as a control tool for practitioners,” *Electronics*, vol. 2, no. 3, pp. 246–279, 2013.

[28] G. F. Franklin, M. L. Workman, and D. Powell, *Digital Control of Dynamic Systems*, 3rd ed. Boston, MA, USA: Addison-Wesley Longman Publishing, 1997.

[29] R. Miklosovic and A. Radke, “High performance tracking control for the practitioner,” in *2007 American Control Conference*, 2007, pp. 3009–3014.

[30] Y. Zhang, Y. Zhang, J. Wang, and R. Ma, “An active disturbance rejection control of induction motor using DSP+FPGA,” in *2013 25th Chinese Control and Decision Conference (CCDC)*, 2013, pp. 4047–4052.

[31] M. R. Stanković, S. M. Manojlović, S. M. Simić, S. T. Mitrović, and M. B. Naumović, “FPGA system-level based design of multi-axis ADRC controller,” *Mechatronics*, vol. 40, pp. 146–155, 2016.

[32] B. Ahir and A. Nobakhti, “Hardware implementation of an ADRC controller on a gimbal mechanism,” *IEEE Transactions on Control Systems Technology*, vol. 26, no. 6, pp. 2268–2275, 2018.

[33] R. Desai, B. M. Patre, and S. N. Pawar, “Active disturbance rejection control with adaptive rate limitation for process control application,” in *2018 Indian Control Conference (ICC)*, 2018, pp. 131–136.

[34] M. Ramírez-Neria, A. Laviano-Juérez, N. Lozada-Castillo, G. Ochoa-Ortega, and R. Madonski, “Discrete-time active disturbance rejection control: A delta operator approach,” in *Advanced, Contemporary Control*, A. Bartoszewicz, J. Kabziński, and J. Kacprzyk, Eds. Cham, Switzerland: Springer International Publishing, 2020, pp. 1383–1395.

[35] Y. Peng, D. Vrančič, and R. Hanus, “Anti-windup, bumpless, and conditioned transfer techniques for PID controllers,” *IEEE Control Systems Magazine*, vol. 16, no. 4, pp. 48–57, 1996.

[36] K. J. Åström and T. Hägglund, *Advanced PID Control*. Durham, NC, USA: ISA—The Instrumentation, Systems, and Automation Society, 2006.