electronics

MDPI

Review

Review of DC Motor Modeling and Linear Control: Theory with Laboratory Tests

Miklós Kuczmann

Department of Power Electronics and E-Drives, Audi Hungaria Faculty of Vehicle Engineering, Széchenyi István University, H-9026 Győr, Hungary; kuczmann@sze.hu

Abstract: This review paper introduces the modeling, measurement, identification and control of direct current motors based on the state space modeling and the transfer function representation. These models are identified by real laboratory measurements, and the simulated results are compared with the measurements. Continuous-time and discrete-time PID (Proportional-Integral-Derivative) controllers, discrete-time state feedback and linear quadratic controllers are designed mathematically. The designed controllers are realized by the microcontroller Arduino UNO, and the behavior of the controllers is compared and analyzed. The noisy current signal has been measured by a discrete-time observer, steady-state Kalman filtering is also studied. The practical results of the implemented controllers support the theoretical results very well.

Keywords: direct current (DC) motor; dynamic system modeling; identification; linear control theory; PID controller; state feedback controller

1. Introduction

Direct current motors (DC motors) are a well-known type of electric motor that convert electrical energy into mechanical energy. DC motors find many applications across a wide range of industries and fields due to their ease of use, simplicity, controllability, and versatility. These motors can be found in various automotive applications, industrial conveyor belt systems, robotic arms, CNC machines, and various actuating and power systems, as well as in medical applications or in everyday devices like home appliances, consumer electronics and so forth. The literature on electrical machinery is vast, see, e.g., [1–6].

DC motors operate on the physical principle of Lorentz force, where a current-carrying conductor placed in a magnetic field experiences a force: $\vec{\mathbf{F}} = I\vec{\mathbf{dl}} \times \vec{\mathbf{B}}$, where I is the DC current flowing through the conductor part with the infinitesimal length of $\vec{\mathbf{dl}}$, and $\vec{\mathbf{B}}$ is the magnetic flux density [7,8]. This force results in the torque that causes the rotating part (rotor) to turn, resulting in mechanical motion.

The homogeneous magnetic field can be generated by windings supplied by DC current or by magnets arranged around the rotating part. The part of the motor where these windings or the magnets are placed is called the stator [1–3].

A frame made of wire is placed in a homogeneous magnetic field as shown in Figure 1 [9]. The current I is flowing through the wire of the frame. In this special case, when the vectors $\vec{\mathbf{dl}}$ and $\vec{\mathbf{B}}$ are perpendicular to each other, the Lorentz force equation has the simple form of $F = BIl$, where l is the side length of the wireframe, and $B = |\vec{\mathbf{B}}|$ is the constant magnetic flux density amplitude. The Lorentz force is constant, and perpendicular to the magnetic field and the frame side, too.

Then, a torque T is acting on the frame: $T = Fd\cos\vartheta = BIld\cos\vartheta$, where d is the distance between the two sides of the frame, and $\vartheta = \vartheta(t)$ is the angle between the frame and the horizontal ($\vartheta = 0$ in Figure 1). Since the frame can rotate, meaning that the angle can change, consequently the angle is a function of time.

check for updates

Citation: Kuczmann, M. Review of DC Motor Modeling and Linear Control: Theory with Laboratory Tests. *Electronics* **2024**, *13*, 2225. https://doi.org/10.3390/electronics13112225

Academic Editor: Pedro J. Villegas

Received: 3 May 2024
Revised: 23 May 2024
Accepted: 3 June 2024
Published: 6 June 2024

Copyright: © 2024 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

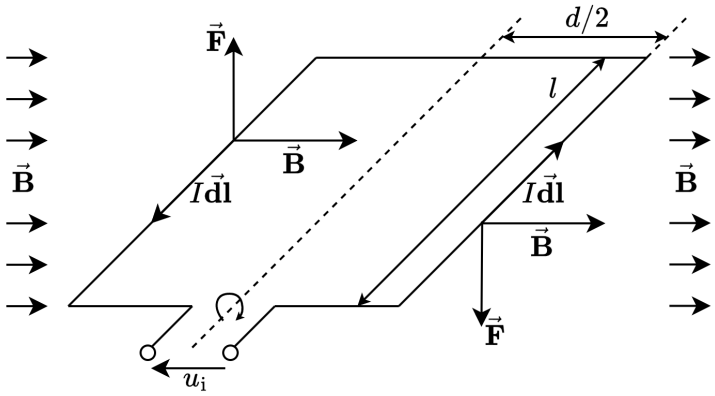


Figure 1. The Lorentz force acting on a current carrying frame.

Figure 1 shows only one frame for simplicity. The rotor in a real DC motor consists of a number of frames, which are rotated relative to each other and all frames are connected to the so-called commutator [1–3,9].

Since the motor has several frames [1–3,9], thanks to the commutators, $\cos\vartheta \approx 1$, because there is usually a frame with approximately the maximum value of the torque, and $T = KI$ can be used as an approximation. In other words, torque is proportional to the current, and K is the motor constant depending on the magnetic flux and the motor dimensions (here $K = Bld$).

The frame is rotated by the above-mentioned torque. The voltage $u_\mathrm{i} = u_\mathrm{i}(t)$ is induced according to Lenz law, i.e., to the motion: $u_\mathrm{i} = -\frac{\mathrm{d}\Phi}{\mathrm{d}t}$, where $\Phi = Bld\sin\vartheta$ is the flux of one frame, i.e., $u_\mathrm{i} = -Bld\dot{\vartheta}\cos\vartheta$. Since the motor is built up by several frames, $u_\mathrm{i} = -K\omega$ can be supposed, and $\omega = \dot{\vartheta}$ is the angular velocity of the rotor.

The motor constant K is supposed to be a constant value in this paper. It is a motor parameter identified by measurements.

As mentioned, the DC motor consists of several rotating frames [1–3,9], and only one frame acts at a moment. The switching between the frames can be realized by the commutator.

There are different types of DC motors [1–3,10], including brushed DC motors and brushless DC motors (BLDC motors). Brushed motors have brushes and a commutator to control the direction of current flow in the motor windings. Simple and economical brushed DC motors require maintenance due to brush wear and generate more electrical noise. Brushless motors use electronic commutation instead of brushes. They typically have a higher efficiency, longer lifespan, and lower maintenance requirements compared to brushed DC motors.

DC motors can be controlled in various ways, including voltage control, pulse-width modulation (PWM), or by using specialized motor controllers or drivers. These control methods allow for precise speed and torque control and precise positioning. Generally, feedback control systems use sensors to measure parameters such as motor speed, torque, or position and provide feedback to the motor controller to adjust the operation accordingly in a closed-loop system. Feedback control enables the precise control of the motor performance and allows it to maintain the desired speed, torque, or position even in the presence of disturbances or changes in operating conditions. The literature on drive is also vast, e.g., [4–6,10].

To design control systems [11–14], model based approach is the modern way [15–21], which is followed in this paper. The studied controllers are the PID controller family [9,17–29] and the state feedback controllers [10,16,18,19,21,30,31]. PID controller is a type of feedback control system widely used in industrial and automation applications to regulate the behavior of dynamic systems. PID stands for proportional, integral, and derivative, which are the three terms that make up the control action. The controller gains can be set up by the phase margin, for example. State feedback control is a modern technique used in control. The behavior of a

dynamic system is controlled by applying feedback based on the state variables of the system, i.e., state feedback control uses measurements of the system's internal state variables. Various control design techniques, such as pole placement, linear quadratic regulator, and optimal control methods can be used to determine the appropriate feedback gains. The so-called state observer is also studied in this work, which is used to estimate the internal state variables of the dynamic system based on available and deterministic input and output measurements, i.e., an observer estimates the state variables based on the system inputs and outputs. The Kalman filter as an optimal state estimator is also implemented to estimate the state of a dynamic system based on noisy measurements of the system's stochastic input and output. Here, only linear control methods and full-order observers/estimators are shown.

Modeling the dynamic behavior of a DC motor involves describing its electrical and mechanical characteristics mathematically [9,17,20,21,31–40]. The dynamic model typically considers the interaction between electrical quantities (voltage and current) and mechanical quantities (torque, speed, and position) in the form of a system of coupled differential equations in the time domain or applying the transfer function between the input and output of the system in the frequency domain or in the Laplace transform domain. Differential equation representation is also called the state space model. Linear and nonlinear models are also shown in the literature, here only linear modeling is studied, and all the nonlinearities are neglected. Major nonlinearity is the dead-zone type friction [29,35–37,39,41–43].

The dynamic model will be described step by step in Section 2. State feedback controller design is based on the state space representation of the dynamic system, while PID controllers are designed by using the transfer function. PID controllers and state feedback-based controllers are presented and designed separately in this work.

After motor modeling, the system of differential equations or the transfer function can be set up by system identification. The offline (batch method) or online (real-time method) least square parameter estimation are core methods to model identification [30,32,33,35,44–46]. The used measurement data and the steps of identification are detailed in Section 2, where the motor parameters (armature resistance and inductance, motor constant, moment of inertia, viscous friction constant) are identified directly from measurements based on the method shown in [20,30,31,34,39,47–49]. It will be proved that the mathematical model of the studied DC motor is linear so linear system modeling, identification and linear control methods can be studied in a comprehensive way. It is supposed that the motor parameters are constant and do not vary with time or temperature.

Control design can be performed by using continuous-time model and discrete-time model [11–14]. First, the continuous-time model and controller design are presented, and then the discrete-time model and design are studied in Section 2. Processing sensor data, calculating control signal, and driving the actuator to achieve desired system behavior are realized by the microcontroller Arduino UNO [50]. Finally, a simple, beginner-friendly and cheap microcontroller environment has been built to study DC motor measurements, modeling, identification and control. The implementation details are shown in Section 3.

The paper focuses on feedback control of the motor speed, torque control can not be performed by the realized test bench.

The built laboratory equipment is suitable for the comprehensive testing of small DC motors, for building and identifying the motor model, as well as for testing the designed controllers. The equipment and the summarized theoretical results can be used for educational purposes too. The aim of the paper is that the measurements with the presented test bench as well as the controller design results can be reproduced. The measurement setup can be used to test new controller algorithms, here only conventional linear techniques are presented in detail. The novelty of this review paper is the presentation, modeling, analysis, and control of a second-order linear and time-invariant system in a comprehensive way. The results based on a literature search can be applied to other second-order systems; moreover, the results can be reproduced.

Section 2 presents the realized test bench, the theoretical background of state space modeling and transfer function representation, PID controllers and state feedback design, equations of the observer as well as the Kalman filter. The results are presented in Section 3, and Section 4 is for discussion. Some important Arduino UNO codes are in the Appendices A–C.

2. Materials and Methods

2.1. DC Motor Test Bench

A simple laboratory test bench has been implemented to demonstrate the analysis of the DC motor behavior and the designed controllers. The measurement setup shown in Figure 2 can be used to perform both open-loop tests and closed-loop controls.

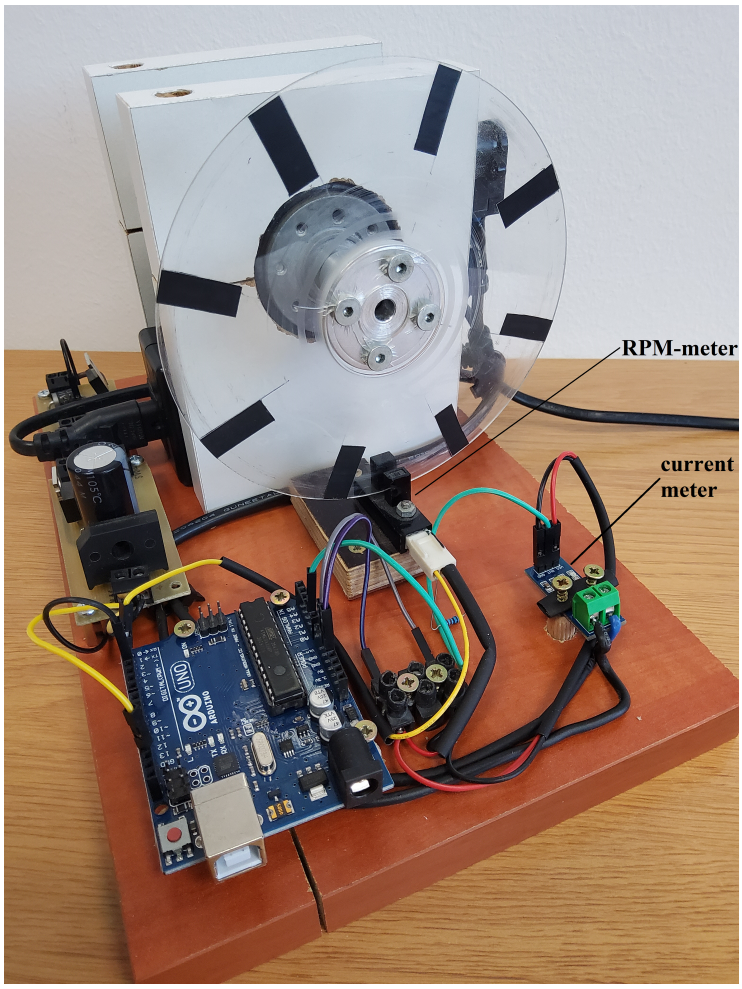


Figure 2. Equipment for testing and controlling the DC motor.

The type of permanent magnet low-power motor under test is Transtecno EC020.120 [51]. The nominal power is 20 W, and the nominal speed is 2850 RPM (round-per-minute). The motor speed is measured by a disc (with eight black glued strips evenly spaced at 45°) rotating between the transmitter and receiver sides of a Sharp GP1A22LC optocoupler [52]. The speed measurement has been implemented in the microcontroller with an interrupt. The type of Hall-element sensor used to measure the current is ACS 712 [53], which can be easily interfaced with the Arduino controller [50]. Here, the well-known Arduino UNO microcontroller has been installed. Details of the power supply and the electronics controlling the motor can be found in the thesis [54].

The open-loop measurement is presented in Figure 3 means that the Arduino microcontroller controls the motor according to a predefined time-dependent voltage signal

$u_a = u_a(t)$, and the measured current $i_a = i_a(t)$; the speed waveforms $n = n(t)$ are only recorded for further use. The measured data are fed to the computer via a serial port. This setup is suitable for testing the motor and taking measurements to be recorded for model identification. The index 'a' stands for the armature coil, which is fed by the voltage u_a and the current flowing through the coil is i_a. In this permanent magnet motor, the armature coil is wound on the rotor. Open-loop measurements are useful for system identification, because motor model parameters are not provided by the manufacturer, i.e., model parameters must be identified via measurements.

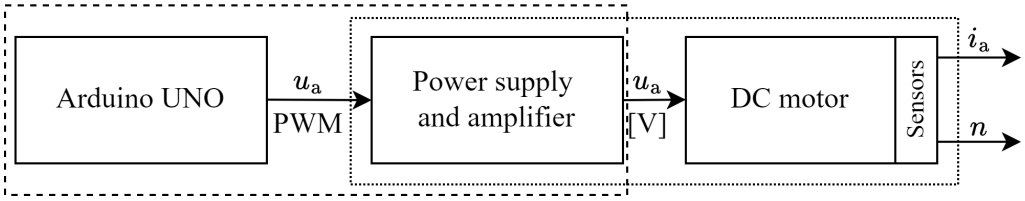


Figure 3. Block diagram of the open loop measurement system.

In a closed loop arrangement, i.e., control as known from control theory, the speed and in some cases also the current signals are fed back, as shown in Figure 4. In this work, the main control objective is speed control, so this signal is always fed back. The feedback of the current signal is indicated by a dashed line, highlighting that it is not always used (current is necessary only in the case of state feedback control, for simple PID it is not essential). In the case of control, the microcontroller does not just record the measured signals but uses them for control.

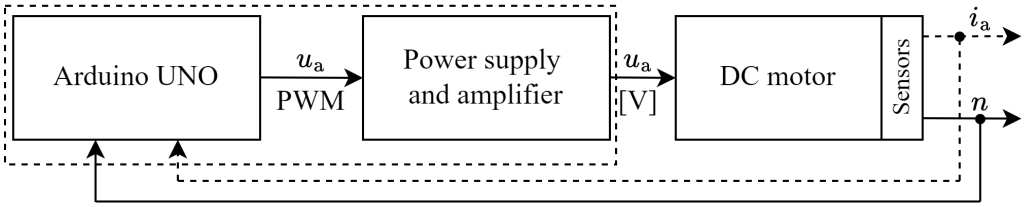


Figure 4. Block diagram of the closed loop measurement system.

Note, that the microcontroller can define the voltage u_a by a PWM value (i.e., a pulse width modulation signal with the appropriate duty cycle), and the power supply and amplifier block results the voltage in the unit of volt.

2.2. DC Motor Modeling

The DC motor equations can be written using the equivalent circuit diagram shown in Figure 5 [3–6,9,17,20,21]. The excitation of the armature is the terminal voltage $u_a = u_a(t)$, also called the armature voltage and the armature current in the rotor winding is denoted by $i_a = i_a(t)$. The electric block represents the armature circuit, which can be modeled as a simple serial-connected RL circuit, and the voltage $u_i = u_i(t)$ induced by the rotation of the rotor acts against the armature voltage so that it is connected in series with the RL circuit in the model, thus reducing the voltage on the RL-elements. The negative sign shown in the introduction is neglected by this reference direction. The mechanical block of the model describes the rotation of the rotor and the mechanical characteristics of the rotational motion.

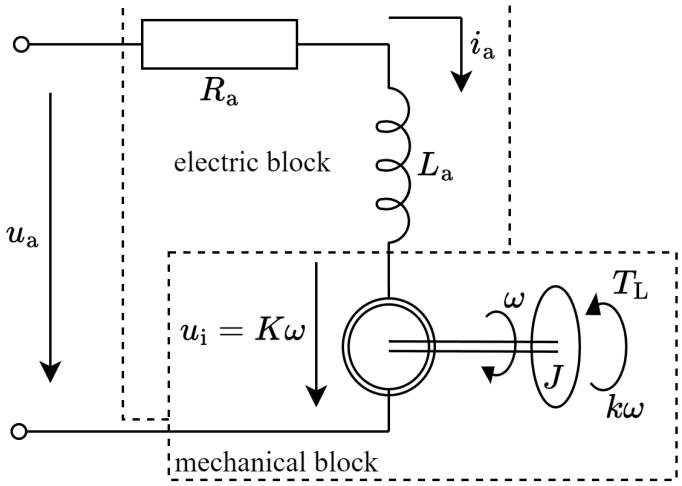


Figure 5. Equivalent circuit model of the DC motor.

The electrical equation of a DC motor based on the Kirchhoff's voltage law is as follows:

$$R_a i_a + L_a \dot{i}_a + u_i = u_a, \tag{1}$$

where R_a and L_a are the resistance and inductance of the serial-connected connected RL circuit of the armature. The induced voltage is proportional to the angular velocity of the rotor $\omega = \omega(t)$, i.e.,

$$u_i = K\omega, \tag{2}$$

where K is the constant specific to the machine. The dot above the letter i_a denotes the time derivative, $\dot{i}_a = \frac{di_a}{dt}$.

The mechanical equation of the machine is as follows:

$$J\dot{\omega} = T - T_L - k\omega, \tag{3}$$

where

$$T = Ki_a \tag{4}$$

is the torque proportional to the armature current ($T = T(t)$), $T_L = T_L(t)$ is the unknown load torque, J is the moment of inertia of the rotating mass, and k is the friction coefficient. The load torque and friction reduce the torque of the motor accordingly.

The following system of linear ordinary differential equations describes the state space model:

$$\begin{aligned} \dot{i}_a &= -\frac{R_a}{L_a} i_a - \frac{K}{L_a}\omega + \frac{1}{L_a} u_a, \\ \dot{\omega} &= \frac{K}{J} i_a - \frac{k}{J}\omega - \frac{1}{J} T_L, \end{aligned} \tag{5}$$

or with matrix notation:

$$\begin{aligned} \frac{d}{dt}\begin{bmatrix} i_a \\ \omega \end{bmatrix} &= \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{K}{L_a} \\ \frac{K}{J} & -\frac{k}{J} \end{bmatrix} \begin{bmatrix} i_a \\ \omega \end{bmatrix} + \begin{bmatrix} \frac{1}{L_a} \\ 0 \end{bmatrix} u_a + \begin{bmatrix} 0 \\ -\frac{1}{J} \end{bmatrix} T_L, \\ y &= \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} i_a \\ \omega \end{bmatrix} + 0 u_a. \end{aligned} \tag{6}$$

The unknown load torque T_L can be considered as a disturbance signal, so the input of the system is the armature voltage u_a and the output is the angular velocity ω. The state variables are the armature current and the angular velocity:

$$\mathbf{x} = \begin{bmatrix} i_\mathrm{a} & \omega \end{bmatrix}^\mathrm{T}, \tag{7}$$

moreover

$$\mathbf{A} = \begin{bmatrix} -\frac{R_\mathrm{a}}{L_\mathrm{a}} & -\frac{K}{L_\mathrm{a}} \\ \frac{K}{J} & -\frac{k}{J} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \frac{1}{L_\mathrm{a}} \\ 0 \end{bmatrix}, \quad \mathbf{c}^\mathrm{T} = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad D = 0 \tag{8}$$

are the elements of the general description of the linear state space modeling:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{b}u, \\ y &= \mathbf{c}^\mathrm{T}\mathbf{x} + Du. \end{aligned} \tag{9}$$

Note, that the angular velocity ω is often used instead of the rotation speed n. The unit of angular velocity is rad/s and the unit of rotation speed is round-per-minute, abbreviated to RPM, i.e., the number of revolutions per minute, or the number of times the motor shaft turns round in one minute. The relationship between the two is as follows:

$$n[\mathrm{RPM}] = 60\frac{\omega[\mathrm{rad/s}]}{2\pi}, \quad \text{and} \quad \omega[\mathrm{rad/s}] = 2\pi\frac{n[\mathrm{RPM}]}{60}. \tag{10}$$

In the physical model ω is used, n is used in microcontroller programs and in the presentation of the measurement results.

The state space modeling will be used in the state feedback controller design. PID controller design is based on the transfer function of the system, which can be derived as follows.

Laplace transforming the differential equation system (6), the following algebraic equations can be obtained:

$$\begin{aligned} (R_\mathrm{a} + sL_\mathrm{a})I_\mathrm{a} &= U_\mathrm{a} - K\Omega, \\ (k + sJ)\Omega &= KI_\mathrm{a} - \mathscr{L}\{T_\mathrm{L}\}. \end{aligned} \tag{11}$$

Here, $I_\mathrm{a} = I_\mathrm{a}(s) = \mathscr{L}\{i_\mathrm{a}(t)\}$, $\Omega = \Omega(s) = \mathscr{L}\{\omega(t)\}$, and $U_\mathrm{a} = U_\mathrm{a}(s) = \mathscr{L}\{u_\mathrm{a}(t)\}$ denote the Laplace transform of the armature current, angular velocity and armature voltage. The transformed variable $\mathscr{L}\{T_\mathrm{L}\}$ stands for the Laplace transform of the load torque, used in the equations but it will be omitted at the end.

The Laplace transform can be easily used to draw the block diagram shown in Figure 6. Here, it can be clearly seen that the input of the electric block is the voltage $U_\mathrm{a} - K\Omega$ and the output is the current I_a, from the first equation of (11)

$$I_\mathrm{a} = \frac{U_\mathrm{a} - K\Omega}{R_\mathrm{a} + sL_\mathrm{a}}. \tag{12}$$

This current is the input of the mechanical block, whose output is the angular velocity Ω and the voltage U_i, according to the second equation of (11):

$$\Omega = \frac{KI_\mathrm{a} - \mathscr{L}\{M_\mathrm{T}\}}{k + sJ}, \quad U_\mathrm{i} = K\Omega. \tag{13}$$

In the following, the load torque is omitted.

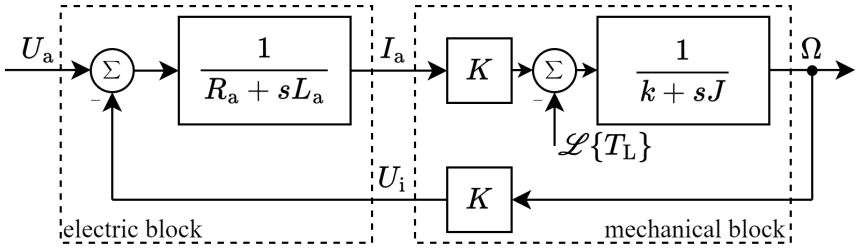


Figure 6. Block diagram model of the DC motor.

From (13)

$$I_a = \frac{k + sJ}{K}\Omega, \tag{14}$$

which is substituted into (12) to obtain the transfer function for the relationship between the angular velocity and the armature voltage:

$$W_\Omega = \frac{\Omega}{U_a} = \frac{K}{s^2 JL_a + s(kL_a + JR_a) + kR_a + K^2}, \tag{15}$$

so that a transfer function describing the relationship between the armature current and the armature voltage can be written as follows:

$$W_{I_a} = \frac{I_a}{U_a} = \frac{k + sJ}{s^2 JL_a + s(kL_a + JR_a) + kR_a + K^2}. \tag{16}$$

By neglecting the friction coefficient, the model can be simplified, i.e., if $k \approx 0$, then the transfer function of the system is as follows:

$$W_\Omega = \frac{\Omega}{U_a} = \frac{K}{s^2 JL_a + sJR_a + K^2}, \tag{17}$$

and

$$W_{I_a} = \frac{I_a}{U_a} = \frac{sJ}{s^2 JL_a + sJR_a + K^2}. \tag{18}$$

The former transfer function W_Ω, divided by K^2 in both the numerator and the denominator, can be expressed as follows:

$$W_\Omega = \frac{\Omega}{U_a} = \frac{1/K}{s^2 \frac{JR_a}{K^2}\frac{L_a}{R_a} + s\frac{JR_a}{K^2} + 1} = \frac{1/K}{s^2 \tau_m \tau_e + s\tau_m + 1}, \tag{19}$$

introduced by

$$\tau_e = \frac{L_a}{R_a}, \quad \text{and} \quad \tau_m = \frac{JR_a}{K^2}, \tag{20}$$

the so-called electric and mechanical time constants.

For identification and control design purposes, the general transfer function of the second-order system can be written independently of the physical background:

$$W_\Omega = \frac{\Omega}{U_a} = \frac{\alpha}{s^2 + \delta s + \varepsilon}, \quad \text{and} \quad W_{I_a} = \frac{I_a}{U_a} = \frac{\beta s + \gamma}{s^2 + \delta s + \varepsilon}, \tag{21}$$

where α, β, γ, δ and ε are parameters.

Identification is detailed in Section 3.2.

2.3. Linear Controller Design

Controller design can be performed after model selection and model parameter identification.

The classical PID controllers can be designed by using the transfer function which output is the motor speed. Here, the design steps based on the phase margin will be

followed. The modern state feedback controllers are based on the pole placement method, which can be designed by using the Ackermann equation or the framework of linear quadratic regulation.

These methods are presented next.

2.3.1. PID Controllers

To design the PID controllers, the block diagram shown in Figure 7 is used [11–13,18,21]. Here, $W_P = W_\Omega$ denotes the transfer function of the plant to be controlled and W_C denotes the transfer function of the controller to be designed.

The controller input $\Delta\omega$ is the difference between the reference signal ω_{ref} and the output signal ω of the plant: $\Delta\omega = \omega_{ref} - \omega$, and the output is the control signal u_a, which is the armature voltage driving the motor.

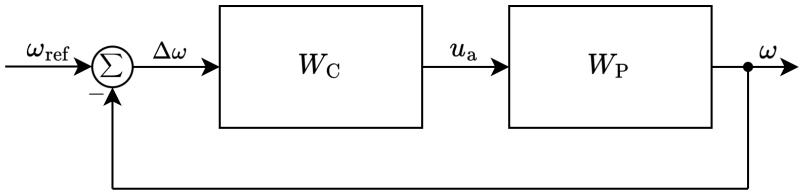


Figure 7. Block diagram of the closed loop system with PID type controllers.

The signals presented in Figure 7 can be generated using the transfer functions. Let the open loop transfer function

$$W_O = W_C W_P, \tag{22}$$

then

$$\Delta\Omega = \frac{1}{1+W_O}\Omega_{ref}, \quad U_a = \frac{W_C}{1+W_O}\Omega_{ref}, \quad \Omega = \frac{W_O}{1+W_O}\Omega_{ref} \tag{23}$$

are the Laplace transform of the error signal, the control signal, and the output signal, and Ω_{ref} is the Laplace transform of the reference signal.

The transfer function of the plant must be transformed into a multiplicative form using the time constant, i.e., from (21)

$$W_P = \frac{\Omega}{U_a} = \frac{\alpha}{s^2+\delta s+\varepsilon} = \frac{\alpha}{(s+p_1)(s+p_2)} = \frac{\alpha/p_1p_2}{(1+s\tau_1)(1+s\tau_2)}, \tag{24}$$

where τ_1 and τ_2 are the time constants of the plant. The motor usually has two stable real poles, $p_1 < 0$ and $p_2 < 0$, with which $\tau_1 = -1/p_1$ and $\tau_2 = -1/p_2$, furthermore, $\tau_1 > \tau_2$.

After this, the continuous-time PID controllers are defined as follows:

- The transfer function of the P controller is constant:

$$W_C = K_P. \tag{25}$$

- The transfer function of the PI controller contains a gain K_{PI} and a time constant T_I in the following formula:

$$W_C = K_{PI}\frac{1+sT_I}{sT_I}, \quad \text{and} \quad T_I = \tau_1. \tag{26}$$

- The transfer function of the PD controller is with a gain K_{PD} and two time constants T_D and T'_D:

$$W_C = K_{PD}\frac{1+sT_D}{1+sT'_D}, \quad \text{and} \quad T_D = \tau_2,\ T'_D = \tau_2/n, \tag{27}$$

where $n > 1$ is a controller parameter.

- The transfer function of the applied PID controller is as follows:

$$W_{\mathrm{C}} = K_{\mathrm{PID}} \frac{1 + sT_{\mathrm{I}}}{sT_{\mathrm{I}}} \frac{1 + sT_{\mathrm{D}}}{1 + sT'_{\mathrm{D}}}, \quad \text{and} \quad T_{\mathrm{I}} = \tau_1,\ T_{\mathrm{D}} = \tau_2,\ T'_{\mathrm{D}} = \tau_2/n. \tag{28}$$

This latter controller is also called a PI-PD controller.

The discrete-time PID controllers are defined as follows:

- The P controller is a gain:

$$W_{\mathrm{C}} = K_{\mathrm{P}}. \tag{29}$$

- The transfer function of the PI controller consists of a gain K_{PI} and a zero z_{I} defined by the formula:

$$W_{\mathrm{C}} = K_{\mathrm{PI}} \frac{z - z_{\mathrm{I}}}{z - 1}, \quad \text{where} \quad z_{\mathrm{I}} = \mathrm{e}^{-T_{\mathrm{s}}/\tau_1}. \tag{30}$$

- The transfer function of the PD controller contains a gain K_{PD} and a zero z_{D} defined by:

$$W_{\mathrm{C}} = K_{\mathrm{PD}} \frac{z - z_{\mathrm{D}}}{z}, \quad \text{where} \quad z_{\mathrm{D}} = \mathrm{e}^{-T_{\mathrm{s}}/\tau_2}. \tag{31}$$

- The transfer function of the applied PID controller is as follows:

$$W_{\mathrm{C}} = K_{\mathrm{PID}} \frac{z - z_{\mathrm{I}}}{z - 1} \frac{z - z_{\mathrm{D}}}{z}, \quad \text{where} \quad z_{\mathrm{I}} = \mathrm{e}^{-T_{\mathrm{s}}/\tau_1}, \quad z_{\mathrm{D}} = \mathrm{e}^{-T_{\mathrm{s}}/\tau_2}. \tag{32}$$

The time constants τ_1 and τ_2 are from the continuous-time model of the motor, used to define the zeros z_{I} and z_{D} of the discrete-time controllers. Note, however, that the controller gains of the continuous-time model and the discrete-time model are denoted by the same notation, of course, their values are not the same.

The sampling time period is denoted by T_{s}. The sampling frequency $f_{\mathrm{s}} = 1/T_{\mathrm{s}}$ can be defined in the microcontroller program.

The details on how to design the controller parameters with the help of the phase margin method are shown in Section 3.3.1.

2.3.2. Observer Design and Kalman Filter

In the following, we turn to the design of state-space-based controllers [11–13,18,21]. These methods make use of the state vector $\mathbf{x} = \begin{bmatrix} i_{\mathrm{a}} & \omega \end{bmatrix}^{\mathrm{T}}$, i.e., the armature current i_{a} and the angular velocity ω. The latter is also the output of the system.

The signal from the current sensor is rather noisy; however, it will be used only to verify the modeling. In many practical situations, the current signal is not available from measurements.

If a state variable cannot be measured or is deliberately not intended to be measured, the observer can be of good service in determining the deterministic signal of the state variable. In the stochastic case with noise, the state estimate given by the Kalman filter can also be used. In the following, both solutions are considered, and both are presented shortly.

To design the observer or the estimator and the state feedback controller, it is necessary to discretize the continuous-time model in the state variable description (6). The step response equivalent sampling model can be generated using the Matlab R2015a function `c2d`:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{b}u, \\ y &= \mathbf{c}^{\mathrm{T}}\mathbf{x}, \end{aligned} \quad \rightarrow \quad \begin{aligned} \mathbf{x}[k+1] &= \mathbf{\Phi}\mathbf{x}[k] + \mathbf{\Gamma}u[k], \\ y[k] &= \mathbf{c}^{\mathrm{T}}\mathbf{x}[k], \end{aligned} \tag{33}$$

where $\mathbf{\Phi}$ is the system matrix, $\mathbf{\Gamma}$ and $\mathbf{c}^{\mathrm{T}}$ are the input and output vectors of the discrete-time model, respectively. The latter is equal to the output vector of the continuous-time model. The system matrix and the input vector depend on the sampling period time T_{s}. The model generates the $(k+1)$-th value of the state vector from the k-th value of the state vector and

the input signal, and the k-th value of the system output signal from the k-th value of the state vector.

The differential equation for a continuous-time observer is as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t}\hat{\mathbf{x}} = \mathbf{F}\hat{\mathbf{x}} + \mathbf{G}y + \mathbf{H}u, \tag{34}$$

where

$$\mathbf{F} = \mathbf{A} - \mathbf{G}\mathbf{c}^{\mathrm{T}}, \quad \mathbf{H} = \mathbf{b}, \tag{35}$$

after the vector $\mathbf{G}$ is designed with the pole placement method to obtain a fast and stable observer.

A continuous-time observer is not used in the following, because solving the differential Equation (34) on a microcontroller is rather time-consuming, instead the discrete-time observer has been implemented. The continuous-time observer will only be used to validate the results of the discrete-time observer.

A continuous-time observer can be designed if the plant is observable, i.e., if the observability matrix

$$\mathbf{M}_{\mathrm{o}} = \begin{bmatrix} \mathbf{c}^{\mathrm{T}} \\ \mathbf{c}^{\mathrm{T}}\mathbf{A} \\ \mathbf{c}^{\mathrm{T}}\mathbf{A}^{2} \\ \vdots \\ \mathbf{c}^{\mathrm{T}}\mathbf{A}^{n-1} \end{bmatrix} \tag{36}$$

is invertible, and n is the number of the states. Now $n = 2$.

In the implementation, the discrete-time observer shown in Figure 8 has been realized. The input of the block representing the DC motor contains the holding unit and the output contains the sampling unit. The observer has two inputs, the deterministic $u[k]$ and $y[k]$, and the output is the estimation of the state vector $\hat{\mathbf{x}}[k+1]$, meaning that the measurement of the states is not necessary, because the state can be computed.

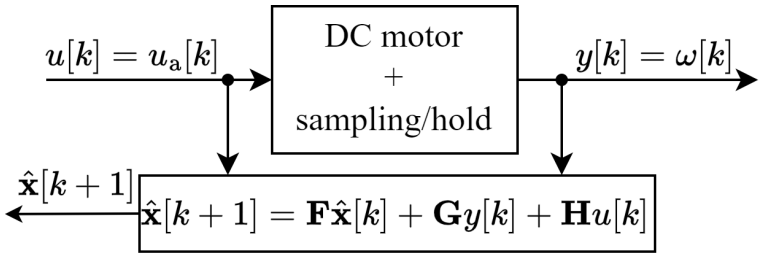


Figure 8. The discrete-time observer/estimator.

The difference equation for the discrete-time observer is as follows:

$$\hat{\mathbf{x}}[k+1] = \mathbf{F}\hat{\mathbf{x}}[k] + \mathbf{G}y[k] + \mathbf{H}u[k], \tag{37}$$

where

$$\mathbf{F} = \mathbf{\Phi} - \mathbf{G}\mathbf{c}^{\mathrm{T}}, \quad \mathbf{H} = \mathbf{\Gamma}, \tag{38}$$

after the vector $\mathbf{G}$ is designed with the pole placement method to obtain a fast and stable observer. Pole placement can be designed by the Ackermann's formula:

$$\mathbf{G} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix}^{\mathrm{T}} \left(\mathbf{M}_{\mathrm{o}}^{-1}\right)^{\mathrm{T}} \varphi_{\mathrm{o}}\left(\mathbf{\Phi}^{\mathrm{T}}\right), \tag{39}$$

where $\mathbf{M}_\mathrm{o}$ is the observability matrix, defined next, $\varphi_\mathrm{o}(\lambda)$ is the characteristic polynomial of the observer with the desired observer eigenvalues $\lambda_{\mathrm{o},i}$ $(i = 1, \cdots, n)$,

$$\varphi_\mathrm{o}(\lambda) = (\lambda - \lambda_{\mathrm{o},1})(\lambda - \lambda_{\mathrm{o},2}) \cdots (\lambda - \lambda_{\mathrm{o},n}). \tag{40}$$

A discrete-time observer can be designed if the discrete plant model is observable, i.e., if the discrete-time observability matrix

$$\mathbf{M}_\mathrm{o} = \begin{bmatrix} \mathbf{c}^\mathrm{T} \\ \mathbf{c}^\mathrm{T}\mathbf{\Phi} \\ \mathbf{c}^\mathrm{T}\mathbf{\Phi}^2 \\ \vdots \\ \mathbf{c}^\mathrm{T}\mathbf{\Phi}^{n-1} \end{bmatrix} \tag{41}$$

is invertible, and n is the number of the states. Now, $n = 2$.

The design of the Kalman filter, suppose that the discrete-time model of the plant is given by the following equations instead of (33):

$$\begin{aligned} \mathbf{x}[k+1] =& \mathbf{\Phi}\mathbf{x}[k] + \mathbf{\Gamma}u[k] + \mathbf{v}[k], \\ y[k] =& \mathbf{c}^\mathrm{T}\mathbf{x}[k] + w[k], \end{aligned} \tag{42}$$

where the vector $\mathbf{v}[k]$ and the scalar $w[k]$ represent the system noise and the output noise, respectively, i.e., the noise model is appended to the state space representation of (33). It is supposed that the noises are Gaussian white noise with zero expected value and the covariances

$$\mathbf{R}_\mathrm{v} = \mathbb{E}\{\mathrm{v}[k]\mathrm{v}[k]^\mathrm{T}\}, \quad R_\mathrm{w} = \mathbb{E}\{w[k]^2\} \tag{43}$$

are obtained from measurements. All the other covariances are supposed to be zero.

Instead of the iterative algorithm for Kalman filter, the steady-state approximation has been implemented in this work, because the DC motor state space representation is a time invariant model. In the more general former method the Kalman gain is given iteratively, in the latter, the steady-state Kalman gain is given as the solution of the so-called Filter Algebraic Riccati Equation (FARE).

The difference equation of the Kalman filter is the same as that of the discrete-time observer (37); moreover, the matrix $\mathbf{F}$ and the vector $\mathbf{H}$ are calculated in the same way, but the vector $\mathbf{G}$, the so-called steady-state Kalman gain is calculated via the discrete-time FARE:

$$\mathbf{\Phi}\mathbf{P}\mathbf{\Phi}^\mathrm{T} - \mathbf{\Phi}\mathbf{P}\mathbf{c}^\mathrm{T}\Big(\mathbf{c}\mathbf{P}\mathbf{c}^\mathrm{T} + R_\mathrm{w}\Big)^{-1}\mathbf{c}\mathbf{P}\mathbf{\Phi}^\mathrm{T} + \mathbf{R}_\mathrm{v} - \mathbf{P} = \mathbf{0}, \tag{44}$$

from which the steady-state estimation error covariance matrix $\mathbf{P}$ can be obtained, and than the steady-state Kalman gain is constant,

$$\mathbf{G} = \mathbf{\Phi}\mathbf{P}\mathbf{c}^\mathrm{T}\Big(\mathbf{c}\mathbf{P}\mathbf{c}^\mathrm{T} + R_\mathrm{w}\Big)^{-1}. \tag{45}$$

It is highlighted that the Kalman estimator has two inputs, the stochastic $u[k]$ and $y[k]$, and the output is the estimation of the state vector $\hat{\mathbf{x}}[k+1]$, meaning that the measurement of the states is not necessary, because the state can be estimated.

Section 3.3.2 details the numerical results.

2.3.3. Digital State Feedback Controllers

The block diagram of the system implementing the state feedback control is shown in Figure 9 [11–13,18,21]. The input of the motor is the held value of the control signal $u_\mathrm{a}[k]$ with the sampling period time T_s, and the output is the sampled angular frequency $\omega[k]$ or the rotor speed $n[k]$.

The current signal is not measured directly by the current sensor but is generated by the state observer or by the Kalman filter.

In order to make the speed response to a step reference signal change with zero steady-state error, an integrator is built into the feedforward branch of the closed loop system. The integrator is approximated by the left-side rectangular rule. The integrator input here is $e[k] = \Delta\omega[k]$ and the output is denoted by $\delta[k]$:

$$\frac{1}{s} \cong \frac{T_s z^{-1}}{1 - z^{-1}} = \frac{T_s}{z - 1} = \frac{\mathcal{Z}\{\delta[k]\}}{\mathcal{Z}\{e[k]\}}, \tag{46}$$

where $\mathcal{Z}\{\cdot\}$ is the z-transform of the signal, and then $\delta[k+1]$ can be written in the following recursive form:

$$\delta[k+1] = \delta[k] + T_s e[k]. \tag{47}$$

The appended equation of the state space model extended with the integrator can be written as follows:

$$\begin{bmatrix} x[k+1] \\ \delta[k+1] \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi} & \mathbf{0} \\ T_s \mathbf{c}^T & 1 \end{bmatrix} \begin{bmatrix} x[k] \\ \delta[k] \end{bmatrix} + \begin{bmatrix} \mathbf{\Gamma} \\ 0 \end{bmatrix} u[k] = \mathbf{\Phi}^\star \mathbf{x}^\star[k] + \mathbf{\Gamma}^\star u[k], \tag{48}$$

for which a state feedback controller can be designed using the Ackermann's formula:

$$\begin{bmatrix} \mathbf{k}^T & k_I \end{bmatrix} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix}^T (\mathbf{M}_c^\star)^{-1} \varphi_{cl}(\mathbf{\Phi}^\star), \tag{49}$$

where $\mathbf{M}_c^\star$ is the controllability matrix of the appended system, $\varphi_{cl}(\lambda)$ is the characteristic polynomial of the desired closed loop system with the eigenvalues $\tilde{\lambda}_i$ $(i = 1, \cdots, n+1)$,

$$\varphi_{cl}(\lambda) = (\lambda - \tilde{\lambda}_1)(\lambda - \tilde{\lambda}_2) \cdots (\lambda - \tilde{\lambda}_{n+1}). \tag{50}$$

The eigenvalue for the integrator is also designed, this is the last ($(n+1)$-th) eigenvalue. Finally, the controllability matrix of the appended system is defined as follows:

$$\mathbf{M}_c = \begin{bmatrix} \mathbf{\Gamma}^\star & \mathbf{\Phi}^\star \mathbf{\Gamma}^\star & (\mathbf{\Phi}^\star)^2 \mathbf{\Gamma}^\star & \cdots & (\mathbf{\Phi}^\star)^{n-1} \mathbf{\Gamma}^\star \end{bmatrix}. \tag{51}$$

The system is controllable, if the controllability matrix is invertible, and n is the number of the states. Now, $n = 2$.

Linear quadratic control (LQ-control) is a kind of optimal control, which minimizes the function defined by

$$J(\mathbf{x}^\star[k], u[k]) = \frac{1}{2} \sum_{k=0}^{\infty} \left[(\mathbf{x}^\star[k])^T \mathbf{Q}^\star \mathbf{x}^\star[k] + R u^2[k] \right], \tag{52}$$

and the appended system of difference equations of the closed loop system is a linear constraint of the optimization problem.

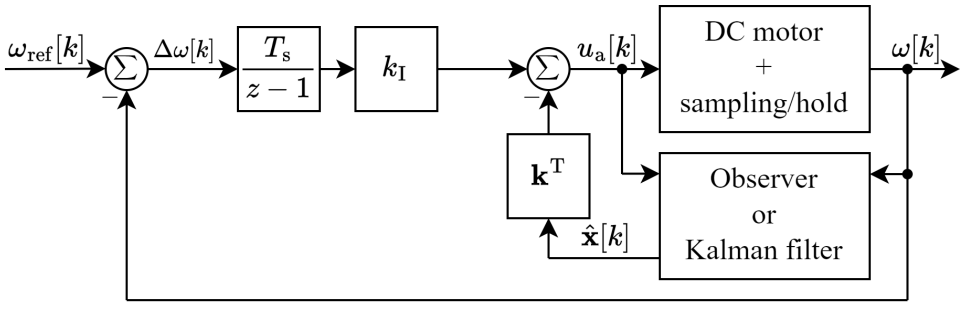


Figure 9. DC motor control by state feedback controller extended by integrator and state observer/estimator.

The first term of the functional provides a stable closed loop system, while the second term can influence the control energy.

Linear quadratic control (LQ-control) scheme also results in the state feedback realization, so the same design steps are applied as in the case of the state feedback control. The feedback vector with which the closed-loop system states and the control signal can minimize the functional calculated differently, as the solution to the Discrete Algebraic Riccati Equation (DARE):

$$(\mathbf{\Phi}^\star)^{\mathrm{T}}\mathbf{X}^\star\mathbf{\Phi}^\star - (\mathbf{\Phi}^\star)^{\mathrm{T}}\mathbf{X}^\star\mathbf{\Gamma}^\star\Big((\mathbf{\Gamma}^\star)^{\mathrm{T}}\mathbf{X}^\star\mathbf{\Gamma}^\star + R\Big)^{-1}(\mathbf{\Gamma}^\star)^{\mathrm{T}}\mathbf{X}^\star\mathbf{\Phi}^\star + \mathbf{Q}^\star - \mathbf{X}^\star = \mathbf{0}, \tag{53}$$

from which the matrix $\mathbf{X}^\star$ can be obtained, and then the feedback gains are as follows:

$$\begin{bmatrix}\mathbf{k}^{\mathrm{T}} & k_{\mathrm{I}}\end{bmatrix} = \Big((\mathbf{\Gamma}^\star)^{\mathrm{T}}\mathbf{X}^\star\mathbf{\Gamma}^\star + R\Big)^{-1}(\mathbf{\Gamma}^\star)^{\mathrm{T}}\mathbf{X}^\star\mathbf{\Phi}^\star. \tag{54}$$

Numerical results are shown in Section 3.3.2.

3. Results

3.1. Measuring Some Motor Characteristics

With the open loop arrangement (Figure 3), the static, i.e., the steady-state conditions can be easily investigated. Some data are shown in Table 1.

Table 1. Measured data in the stationary state.

u_{a}[PWM]	50	60	70	80	90	100	110	120
u_{a}[V]	0.50	0.93	1.25	1.76	2.21	2.65	3.08	3.47
i_{a}[A]	-	-	-	-	0.04	0.05	0.06	0.08
n[RPM]	90	245	376	520	650	792	930	1062
ω[rad/s]	9.43	25.66	39.38	54.45	68.07	82.94	97.39	111.21
u_{a}[PWM]	130	140	150	160	170	180	190	200
u_{a}[V]	3.95	4.42	4.89	5.30	5.71	6.12	6.55	6.97
i_{a}[A]	0.11	0.12	0.16	0.17	0.19	0.23	0.25	0.26
n[RPM]	1215	1356	1511	1648	1772	1923	2058	2192
ω[rad/s]	127.23	142.00	158.23	172.58	185.56	201.38	215.51	229.55
u_{a}[PWM]	210	220	230	240	250	255		
u_{a}[V]	7.46	7.91	8.31	8.80	9.82	10.4		
i_{a}[A]	0.28	0.30	0.32	0.35	0.39	0.40		
n[RPM]	2346	2489	2625	2786	3120	3308		
ω[rad/s]	245.67	260.65	274.89	291.75	326.73	346.41		

The Arduino UNO uses a PWM signal to adjust the input voltage. The PWM output is 8-bit, i.e., the PWM value can be an integer between 0 and 255, thus determining the duty cycle, i.e., the armature voltage driving the motor.

By exciting the motor with different PWM values (u_{a}[PWM], the first row of Table 1), the steady-state input voltage u_{a}[V], the steady-state armature current i_{a}[A], the rotor speed n[RPM] and the angular velocity ω[rad/s] can be measured. The steady state is typically reached within 1 second.

First, the relationship between the integer PWM input value and the stationary motor state variables is examined. In this case, the amplifier circuit is understood to be in the same block as the motor, as highlighted by the dotted line in the block diagram in Figure 3.

Second, the relationship between the input voltage measured in volts and the state variables in the stationary state is discussed. In this second case, the amplifier circuit is understood as a block with the microcontroller, as highlighted by the dashed line in Figure 3.

In the former case, the input-output relationship is nonlinear, while in the latter case, it is linear. In the controller design, the linear approach is used, and this approach is more

closely related to the state space description of the motor since its input is the physically interpretable armature voltage measured in volts.

Figure 10 shows one of the static characteristics of the motor, the relationship between the 8-bit PWM excitation signal and the rotor speed. It can be seen that this is nonlinear, and given by the following characteristic:

$$n[\mathrm{RPM}] = \begin{cases} 0 & \text{if} \quad u_\mathrm{a}[\mathrm{PWM}] < 47, \\ 14.7u_\mathrm{a}[\mathrm{PWM}] - 683.3 & \text{if} \quad u_\mathrm{a}[\mathrm{PWM}] \geq 47. \end{cases} \tag{55}$$

This means that the motor shaft will not rotate until the 8-bit PWM excitation reaches (rounded up to the integer value) 47. The small circle indicates the measurement results and the solid line indicates the approximate values according to (55).

All the interpolations performed by linear parameter estimation technique called the least square method [11,12].

In Figure 10, a possible linear approximation is also plotted with a dashed line, with the equation $n[\mathrm{RPM}] = 12.973u_\mathrm{a}[\mathrm{PWM}]$, the slope of which is calculated from the last measured data. This will not be discussed further.

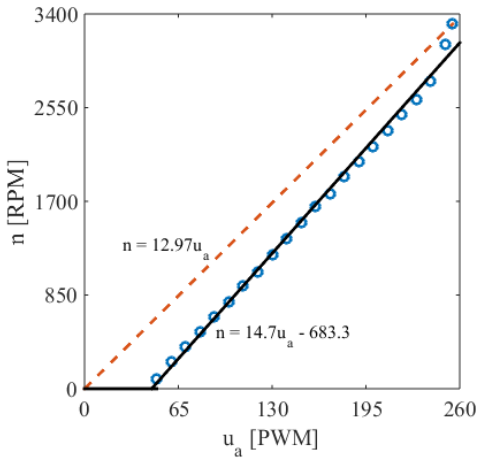


Figure 10. The characteristic $n[\mathrm{RPM}] - u_\mathrm{a}[\mathrm{PWM}]$ is nonlinear.

The same type of armature current-PWM excitation relationship is described by

$$i_\mathrm{a}[\mathrm{A}] = \begin{cases} 0 & \text{if} \quad u_\mathrm{a}[\mathrm{PWM}] < 71, \\ 0.002059u_\mathrm{a}[\mathrm{PWM}] - 0.14784 & \text{if} \quad u_\mathrm{a}[\mathrm{PWM}] \geq 71. \end{cases} \tag{56}$$

The PWM value of the excitation is related to the voltage measured on the motor terminals by a volt-meter instrument as follows:

$$u_\mathrm{a}[\mathrm{V}] = \begin{cases} 0 & \text{if} \quad u_\mathrm{a}[\mathrm{PWM}] < 43, \\ 0.045521u_\mathrm{a}[\mathrm{PWM}] - 1.9336 & \text{if} \quad u_\mathrm{a}[\mathrm{PWM}] \geq 43. \end{cases} \tag{57}$$

The relationship between the armature voltage of the motor measured in volts and the rotor speed is linear according to the measurements shown in Figure 11. The linear approximation is given by

$$n[\mathrm{RPM}] = 318.077u_\mathrm{a}[\mathrm{V}]. \tag{58}$$

The small circle in Figure 11 indicates the measurement results, solid line indicates the approximate values according to the above equations.

The armature current-armature voltage relationship can also be considered linear to a good approximation (see Figure 12):

$$i_\mathrm{a}[\mathrm{A}] = 0.03846u_\mathrm{a}[\mathrm{V}]. \tag{59}$$

The characteristic above 0.2A is considered linear, below that the larger deviation is due to the inaccuracy of the sensor, and linear behavior is assumed for the rest of the paper.

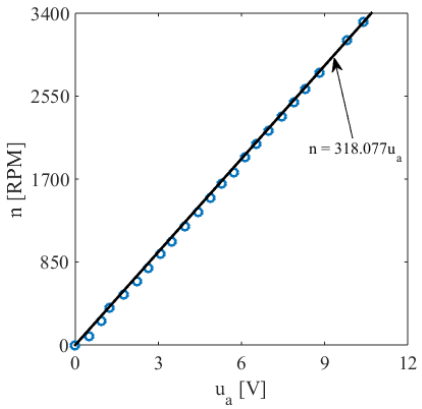


Figure 11. The characteristic $n[\mathrm{RPM}] - u_\mathrm{a}[\mathrm{V}]$ is linear.

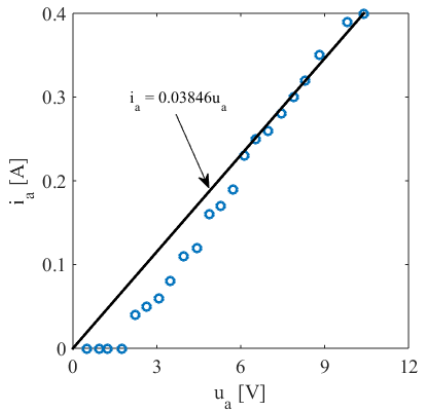


Figure 12. The characteristic $i_\mathrm{a}[\mathrm{A}] - u_\mathrm{a}[\mathrm{V}]$ is approximately linear.

3.2. Identification of Motor Parameters

In the steady-state the derivatives vanish in (1)–(4), and the following equations can be obtained:

$$u_\mathrm{a} = R_\mathrm{a} i_\mathrm{a} + K\omega, \quad \text{and} \quad K i_\mathrm{a} = k\omega, \tag{60}$$

from which K and k can be expressed:

$$K = \frac{u_\mathrm{a} - R_\mathrm{a} i_\mathrm{a}}{\omega}, \quad \text{and} \quad k = \frac{K i_\mathrm{a}}{\omega}. \tag{61}$$

It can be seen that the constants K and k depend on the value of the resistance R_a.

The block diagram of Figure 6 can be split into two separate parts as shown in Figure 13. The input of the electric part is the voltage $u_\mathrm{a} - K\omega$ and the output is the current signal i_a. The current is the input of the mechanical part, the output of which is the angular velocity ω. Thus, if the input-output time functions of interest are available from measurements, the unknown coefficients in the blocks can be determined very accurately after a good choice of armature resistance. The inductance of the armature branch L_a and the moment of inertia of the mechanical part J can be identified independently. The identification is, therefore, carried out in the following order:

$$R_\mathrm{a} \to (K, k) \to (L_\mathrm{a}, J).$$

Using the Matlab R2015a `fminbnd` function, the motor parameters can be determined using an iterative procedure based on the golden section method as follows:

(1) Initialization of R_a on the interval $[0.1, \cdots, 10]\Omega$ according to the rules of the golden section method;
(2) K and k are defined by (61) as a function of the resistance R_a, averaged over the measured stationary results (only the values between $u_a[\text{PWM}] = 150$ and $u_a[\text{PWM}] = 255$ in Table 1 are used);
(3) Determine the value of the inductance L_a using the function `fminbnd` and the differential equation of the electric block shown in Figure 13;
(4) Determine the value of the moment of inertia J using the function `fminbnd` and the differential equation of the mechanical block shown in Figure 13;
(5) Modify R_a according to the golden section method, and go back to step 2 until the algorithm converges.

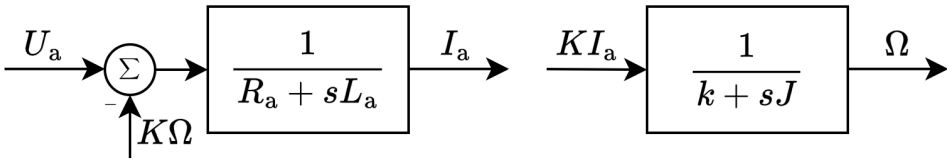


Figure 13. Electric and mechanical parts of the block diagram

In step 3 and in step 4, the following differential equations must be solved:

$$\dot{i}_a = -\frac{R_a}{L_a} i_a + \frac{1}{L_a}(u_a - K\omega), \tag{62}$$

and

$$\dot{\omega} = -\frac{k}{\Theta}\omega + \frac{1}{\Theta}(K i_a). \tag{63}$$

In the former case, the time function of the current signal, in the latter the time function of the angular velocity is obtained. The calculated time functions are compared with the measured results and an error is calculated. To conduct this, the motor is excited with a PWM signal of 250, for example, and the time functions of angular velocity and current in the transient are measured. Given the measured data, the inputs of the electric and mechanical block can be generated separately and the differential equation can be solved numerically using one of the Matlab R2015a `ode` functions.

The identification results are as follows:

$$\begin{aligned} R_a &= 6.21\ \Omega, \quad L_a = 164\ \text{mH}, \\ K &= 0.0234 \frac{\text{V}}{\frac{\text{rad}}{\text{s}}}, \quad k = 2.63 \cdot 10^{-5} \frac{\text{VA}}{\left(\frac{\text{rad}}{\text{s}}\right)^2}, \quad J = 2.31 \cdot 10^{-5}\ \text{kgm}^2. \end{aligned} \tag{64}$$

Substituting the parameters obtained from the identification in SI units into the state space description (6), the following model can be obtained:

$$\frac{\text{d}}{\text{d}t}\begin{bmatrix} i_a \\ \omega \end{bmatrix} = \begin{bmatrix} -37.8087 & -0.1425 \\ 1012.9 & -1.136 \end{bmatrix}\begin{bmatrix} i_a \\ \omega \end{bmatrix} + \begin{bmatrix} 6.0843 \\ 0 \end{bmatrix} u_a. \tag{65}$$

Substituting the above-mentioned physical parameters into the relations (15) and (16), the following transfer functions are obtained:

$$W_\Omega = \frac{\Omega}{U_a} = \frac{6176}{s^2 + s39 + 188}, \quad \text{and} \quad W_{I_a} = \frac{I_a}{U_a} = \frac{s6.1 + 6.9}{s^2 + s39 + 188}. \tag{66}$$

For the above state space description, for comparison, the following transfer functions are calculated by the Matlab R2015a function `ss2tf`:

$$W_\Omega = \frac{\Omega}{U_a} = \frac{6162.9}{s^2 + s38.9446 + 187.3085}, \quad W_{I_a} = \frac{I_a}{U_a} = \frac{s6.0843,1 + 6.9115}{s^2 + s38.9446 + 187.3085}. \tag{67}$$

Another, simpler solution to identify the continuous-time transfer function of the second-order system is as follows.

The parameters in the transfer functions (21) can be determined using the Matlab R2015a `fminsearch` function. The error to be minimized is the difference between the angular velocity and current response calculated using the searched transfer function and the measured time functions.

The simulated angular velocity and armature current are obtained as a solution of the following state space model:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} &= \begin{bmatrix} -\delta & -\varepsilon \\ 1 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u_a, \\ y &= \mathbf{c}^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \end{aligned} \tag{68}$$

where x_1 and x_2 are notional state variables and y is the notional output. The output vector $\mathbf{c}^T = [0\ \alpha]$ gives the angular velocity, and $\mathbf{c}^T = [\beta\ \gamma]$ gives the armature current.

To conduct the identification, the motor is excited with a PWM signal of 250, for example, and the time functions of angular velocity and current in the transient are measured.

The identified parameters are as follows:

$$\alpha = 5434, \quad \beta = 5.53, \quad \gamma = 3.82, \quad \delta = 32.5, \quad \varepsilon = 165, \tag{69}$$

i.e., the transfer functions are as follows:

$$W_\Omega = \frac{\Omega}{U_a} = \frac{5434}{s^2 + s32.5 + 165}, \quad \text{and} \quad W_{I_a} = \frac{I_a}{U_a} = \frac{s5.53 + 3.82}{s^2 + s32.5 + 165}. \tag{70}$$

Figure 14 shows the results of the switch-on transients for different input voltage values. The measured time functions of motor speed and armature current are compared with the model results. For speed, the results are plotted for 100, 150, 200 and 250 8-bit PWM voltage values, for current only the results for 150 and 250 PWM excitations are shown for better visibility. It is clearly visible that the current signal is quite noisy.

The time functions for the 250 PWM value have been used for the identifications.

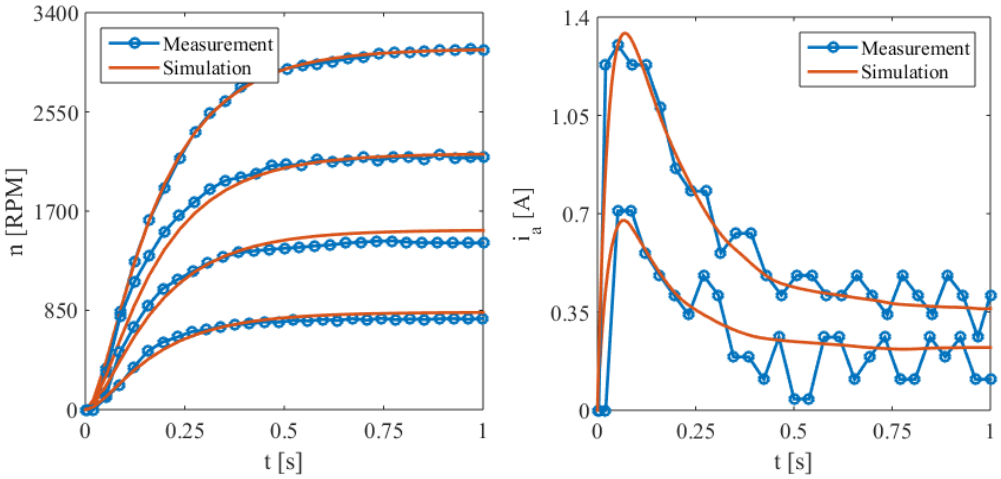


Figure 14. Switch-on transient of motor speed and armature current, comparison between measured and simulated data.

3.3. Design of Linear Controllers

3.3.1. PID Controllers

The block diagram of the closed-loop control system for speed control used in this study is shown in Figure 15. The plant to be controlled is the DC motor, whose speed $n = n(t)$ is controlled, and the reference signal is $n_{\text{ref}} = n_{\text{ref}}(t)$. The difference between the two is the error signal:

$$\Delta n = n_{\text{ref}} - n. \tag{71}$$

The controller is designed for the transfer function $W_{\text{P}} = W_{\Omega} = \Omega/U_{\text{a}}$, i.e., assuming a plant model with input of armature voltage in the unit of volt and with the output of angular velocity in rad/s. The error term must, therefore, be converted from RPM to rad/s, this is conducted by the $n \to \omega$ block. The input of the controller is, therefore, the angular velocity error $\Delta\omega$ and the output is the voltage u_{a} in the unit of volt. The output of the microcontroller produces a voltage in units of PWM, and the V $\to$ PWM block converts the voltage signal of the control algorithm into a PWM value. To conduct this, the relation (57) is rearranged:

$$u_{\text{a}}[\text{PWM}] = 21.96788 u_{\text{a}}[\text{V}] + 42.477, \tag{72}$$

and the result, of course, is rounded to the nearest integer.

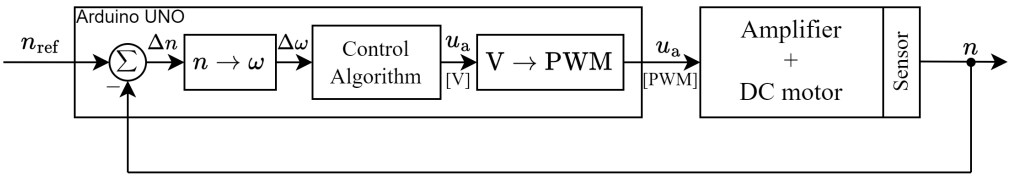


Figure 15. The realized closed loop system with PID controller.

The control algorithm also contains a lower and an upper limit below and towards which the control signal must not go. These PWM values are 50 and 250. The motor does not rotate below 50 PWM (50 is used instead of 43 in the context (57)), and the maximum value would be 255, but it is set to 250 in the microcontroller program. These limits should be taken into account when designing and checking the operation of the controller.

Design of the Continuous-Time PID Controllers

The PID controllers can be designed using the method of designing for phase margin using the open loop system Bode diagram. To conduct this, the polynomial-per-polynomial transfer function of the plant is transformed into a multiplicative form using the time constant, i.e.,

$$W_{\text{P}} = \frac{\Omega}{U_{\text{a}}} = \frac{5434}{s^2 + s32.5 + 165} = \frac{5434}{(s + 26.2)(s + 6.3)} = \frac{32.9}{(1 + s0.159)(1 + s0.038)}. \tag{73}$$

The slow time constant is $\tau_1 = 0.159$ s, and the fast one is $\tau_2 = 0.038$ s, the system poles are $p_1 = -6.3\frac{1}{\text{s}}$ and $p_2 = -26.2\frac{1}{\text{s}}$, with which $\tau_1 = -1/p_1$ and $\tau_2 = -1/p_2$.

As an example, let us design controllers for phase margin $\varphi_{\text{m}} = 60^{\circ}$. As an illustration, the design of a P-controller using Figure 16 is used. In a Matlab R2015a environment, this can be easily conducted with the function `bode`. For this:

(1) Plot the Bode diagram of the transfer function $W_{\text{O}} = W_{\text{C}} W_{\text{P}}$ of the open loop with unit gain, i.e., let $K_{\text{P}} = 1$ ($K_{\text{PI}} = 1$, $K_{\text{PD}} = 1$, and $K_{\text{PID}} = 1$, respectively), as plotted in Figure 16;

(2) find the point $-180^{\circ} + \varphi_{\text{m}}$ (which here means the point -120°) on the phase diagram, and read off the cut-off frequency ω_{c} (in the example ω_{c} = 23.8 rad/s at -117°, of course, the `bode` function can be used to obtain a more precise value, the 3° error is due to the readout given by the graphical display);

(3) read off the value of the amplitude characteristic K_{dB} in decibel units at the cut-off frequency (here $K_{\mathrm{dB}}(\omega_{\mathrm{c}}) = 15.9$ dB);
(4) determine the controller gain:

$$K_{\mathrm{controller}} = 10^{-K_{\mathrm{dB}}/20}, \tag{74}$$

i.e., $K_{\mathrm{P}} = 10^{-15.9/20} = 0.16$, which is a ratio without unit, since the `bode` function represents K_{dB} in decibels according to the relationship of $K_{\mathrm{dB}} = 20\mathrm{lg}K_{\mathrm{controller}}$.

The design can be conducted without plotting the Bode diagram directly, but then a nonlinear system of equations must be solved:

$$|W_{\mathrm{O}}(\mathrm{j}\omega_{\mathrm{c}})| = 1, \quad -180^{\circ} + \varphi_{\mathrm{m}} = \arc W_{\mathrm{O}}(\mathrm{j}\omega_{\mathrm{c}}). \tag{75}$$

This system of equations can be solved efficiently by calling the `fsolve` function. The two unknowns associated with the two equations are the controller gain $K_{\mathrm{controller}}$ and the cut-off frequency ω_{c}.

The step responses of the designed closed-loop control systems are compared in Figure 17, and some characteristic data are summarized in Table 2. Here, ω_{c} is the cut-off frequency in units of rad/s, K is the controller gain, $v_{\max}$ is the maximum value of the step response, and v_{st} is the steady-state value of the step response, $\Delta v[\%] = 100\frac{v_{\max}-v_{\mathrm{st}}}{v_{\mathrm{st}}}$ is the overshoot, $u_{\mathrm{t}} = \frac{u(0)}{u_{\mathrm{st}}}$ is the overcontrol, and finally u_{st} is the steady-state value of the control signal.

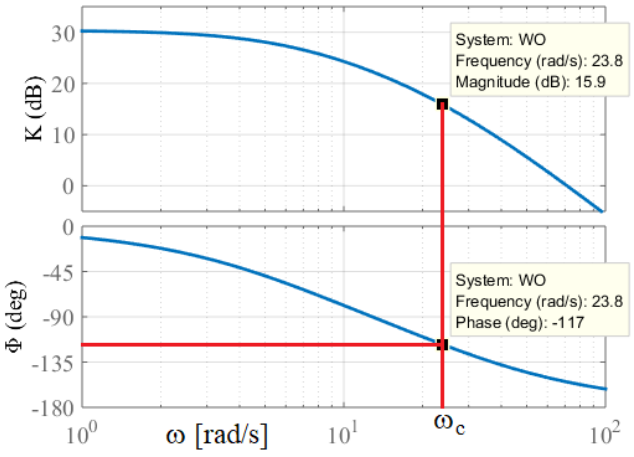


Figure 16. Design of P controller for phase margin $\varphi_{\mathrm{m}} = 60^{\circ}$.

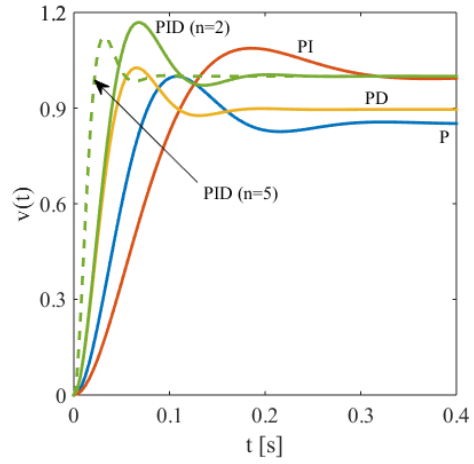


Figure 17. Comparison of closed loop step response data for the phase margin $\varphi_{\mathrm{m}} = 60^{\circ}$.

Table 2. Some data of closed-loop systems for the phase margin $\varphi_m = 60^\circ$.

Controller	ω_c[rad/s]	K	v_{max}	v_{st}	Δv[%]	u_t	u_{st}
P	25.28	0.175	0.99	0.85	17.4	6.74	0.026
PI	15.13	0.084	1.09	1.00	9.48	2.87	0.031
PD ($n = 2$)	41.78	0.261	1.03	0.89	14.6	19.2	0.027
PI-PD ($n = 2$)	41.78	0.261	1.17	1.00	14.9	17.2	0.031
PI-PD ($n = 5$)	88.57	0.517	1.12	1.00	12.4	85.1	0.031

Table 2 shows that the higher the value of ω_c, the faster the control loop is, as can be seen in Figure 17, and that increasing n further increases the controller speed. Also, the higher the value of the loop gain, the faster the controller. Further increasing K will speed up the control loop. However, the acceleration leads to oscillatory behavior, as can be clearly seen in the simulation results, and in fact, large gains can make the closed-loop system unstable. However, fast control is possible with large control signals, as can be seen from the large overcontrol values u_t. A trade-off between the two must be found, i.e., to achieve the fastest control possible, but without saturating the actuating device. From the steady-state value v_{st} it is clear that a controller including an integrator allows accurate operation for the step reference signal. Overshooting occurs in all cases, which can be reduced by increasing the phase margin, reducing the gain, or reducing n. This of course makes the control slower.

The closed loop system behavior is not very accurate when applying only a P-controller. It can be stated that the plant response can be accelerated by a PD block, the steady-state accuracy can be improved by a PI block, while PID altogether results in a fast and accurate control loop.

Discrete-Time Realization of the Continuous-Time PID Controllers

The sampled realization of the above-mentioned continuous-time PID controllers can be conducted by the step response equivalent transformation or by the unconditionally stable backward differential scheme, the right-hand-side rectangle rule, and the Tustin formula. The sampling period time is denoted by T_s, i.e., the frequency of the sampling provided by the microcontroller is $f_s = 1/T_s$.

The constant gain of the discrete-time P controller is of course the same as the gain of the continuous-time controller.

By using $s = \frac{z-1}{T_s z}$ in the transfer function of the PI controller,

$$W_C = K_{PI} \frac{1 + sT_I}{sT_I}\bigg|_{s=\frac{z-1}{T_s z}} = K_{PI} \frac{1 + \frac{z-1}{T_s z} T_I}{\frac{z-1}{T_s z} T_I} \tag{76}$$

can be written. After a short transformation of $W_C(z) = U(z)/E(z)$, the following difference equation can be obtained to obtain the discrete-time control signal:

$$u[k] = u[k-1] + K_{PI}\left(1 + \frac{T_s}{T_I}\right)e[k] - K_{PI}e[k-1]. \tag{77}$$

Using the Tustin formula $s = \frac{2}{T_s}\frac{z-1}{z+1}$ results in:

$$u[k] = u[k-1] + K_{PI}\left(1 + \frac{T_s}{2T_I}\right)e[k] - K_{PI}\left(1 - \frac{T_s}{2T_I}\right)e[k-1], \tag{78}$$

and the step response equivalent transformation gives the difference equation of

$$u[k] = u[k-1] + K_{PI}e[k] - K_{PI}\left(1 - \frac{T_s}{T_I}\right)e[k-1]. \tag{79}$$

In each of the three specifications, the k-th value of the control signal $u[k]$ can be computed iteratively by the k-th value of the error signal $e[k]$ and the $(k-1)$-th value of the error signal and the control signal. The implementing program is, therefore, the same in all three cases, but each value must be multiplied by a different weight. For a possible Arduino code, see Appendices A–C.

The PI controller output signal can be approximated by

$$u(t) = K_{\mathrm{PI}}\left(e(t) + \frac{1}{T_{\mathrm{s}}}\int_0^t e(\tau)\mathrm{d}\tau\right), \tag{80}$$

which results in the difference equation:

$$u[k] = K_{\mathrm{PI}}\sum_{i=0}^{k-1} e[i] = u[k-1] + K_{\mathrm{PI}}e[k]. \tag{81}$$

The control signal by the PD controller can be calculated by

$$u[k] = \frac{T'_{\mathrm{D}}}{T_{\mathrm{s}}+T'_{\mathrm{D}}}u[k-1] + K_{\mathrm{PD}}\frac{T_{\mathrm{s}}+T_{\mathrm{D}}}{T_{\mathrm{s}}+T'_{\mathrm{D}}}e[k] - K_{\mathrm{PD}}\frac{T_{\mathrm{D}}}{T_{\mathrm{s}}+T'_{\mathrm{D}}}e[k-1], \tag{82}$$

when applying the approximation $s = \frac{z-1}{T_{\mathrm{s}}z}$. The Tustin formula $s = \frac{2}{T_{\mathrm{s}}}\frac{z-1}{z+1}$ results in

$$u[k] = -\frac{T_{\mathrm{s}}-2T'_{\mathrm{D}}}{T_{\mathrm{s}}+2T'_{\mathrm{D}}}u[k-1] + K_{\mathrm{PD}}\frac{T_{\mathrm{s}}+2T_{\mathrm{D}}}{T_{\mathrm{s}}+2T'_{\mathrm{D}}}e[k] + K_{\mathrm{PD}}\frac{T_{\mathrm{s}}-2T_{\mathrm{D}}}{T_{\mathrm{s}}+2T'_{\mathrm{D}}}e[k-1], \tag{83}$$

and finally, the step response equivalent is as follows

$$u[k] = qu[k-1] + K_{\mathrm{PD}}\frac{T_{\mathrm{D}}}{T'_{\mathrm{D}}}e[k] + K_{\mathrm{PD}}\left(1-q-\frac{T_{\mathrm{D}}}{T'_{\mathrm{D}}}\right)e[k-1], \tag{84}$$

where $q = \mathrm{e}^{-T_{\mathrm{s}}/T'_{\mathrm{D}}}$.

Also in these specifications, the k-th value of $u[k]$ can be computed iteratively by the k-th value of the error signal $e[k]$ and the $(k-1)$-th value of the error signal and the control signal.

The PD controller output signal can be approximated by

$$u(t) = K_{\mathrm{PD}}\left(e(t) + T_{\mathrm{D}}\frac{\mathrm{d}e(t)}{\mathrm{d}t}\right) \tag{85}$$

which results in the difference equation:

$$u[k] = K_{\mathrm{PD}}e[k] + K_{\mathrm{PD}}T_{\mathrm{D}}\frac{e[k]-e[k-1]}{T_{\mathrm{s}}}. \tag{86}$$

The approximate formulae of the PI-PD controller used are much longer, these are not detailed here, see the result in the code (Appendix A).

Analysis of the Continuous-Time PID Controllers

The following reference signal contains three jump-like signal variations and one linearly varying segment, which allows us to study the behavior of closed-loop control systems:

$$n_{\mathrm{ref}} = \begin{cases} 1000, & \text{if} \quad t < 1\ \mathrm{s}, \\ 2000, & \text{if} \quad 1\ \mathrm{s} \le t < 2\ \mathrm{s}, \\ 2500, & \text{if} \quad 2\ \mathrm{s} \le t < 3\ \mathrm{s}, \\ 4750 - 750t, & \text{if} \quad 3\ \mathrm{s} \le t < 5\ \mathrm{s}. \end{cases} \tag{87}$$

The reference signal tracking of the P controller with phase margins $\varphi_m = 60°$ and $\varphi_m = 40°$ are shown in Figure 18. The measured signal (green line) is noisy, so it is easily recognizable next to the simulated noiseless curves (red curve). The reference signal is the blue line.

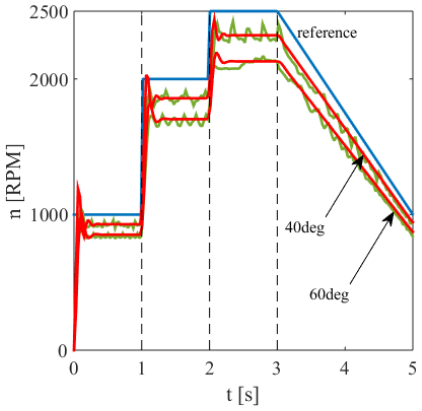


Figure 18. The reference signal tracking of the P controller with phase margins $\varphi_m = 60°$ and $\varphi_m = 40°$.

The measured and simulated results show excellent agreement. The gain of the controller is $K_P = 0.1746$ for the larger phase margin and $K_P = 0.3963$ for the smaller one, i.e., the loop gain $K = 32.9K_P$ is $K = 5.744$ and $K = 13.038$, respectively. From this, the static error can be computed according to $h = \frac{1}{1+K}$, which is 0.148 and 0.071, respectively. The measured and calculated static errors are equal to three decimals. It can be clearly seen that as the loop gain is increased, i.e., the phase margin is decreased, the overshoot in the transient is increased. The overshoot can be reduced by reducing the loop gain, but then the static error increases. It can be clearly seen that static error and overshoot reduction are conflicting demands. In the linearly decreasing section, the two curves are not parallel, and the error increases beyond all limits, since the type number of the control loop is $i = 0$, although the error appears to decrease (over the time interval considered) as the loop gain increases. It can be concluded that the problem cannot be solved satisfactorily with a purely proportional controller.

The reference tracking of the PI controller at $\varphi_m = 60°$ is shown in Figure 19, where the reference, measured and simulated signals are the blue, green and red lines, respectively. Here, the measured and simulated results show excellent agreement. In transient, there is a small overshoot which can be reduced by decreasing the gain K_{PI}, i.e., by increasing the phase margin, in this problem the overshoot at $\varphi_m = 72°$ phase margin practically disappears (then $K_{PI} = 0.0431$). In the stationary state, the error is zero, since the integrating term increases the type number to $i = 1$, and the speed follows the reference with a constant error in the linearly decreasing phase. It can be said that the PI controller is a good choice to solve this problem.

The reference tracking of the PD controller at $\varphi_m = 60°$ is shown in Figure 20, with good agreement between the measured and simulated results. Again, reference, measured and simulated signals are the blue, green and red lines, respectively. The steady-state behavior of the control loop is inherently the same as that of the P-controller. The system cannot be accelerated significantly due to the saturation of the feedback loop, the feedback signal of the designed controller is sometimes many times higher than the upper limit of the PWM excitation signal 250, which means that acceleration does not occur in the real system. This is well observed for the time step jumps $t = 0$ s and $t = 1$ s. Here, the simulated speed has an overshoot, while the measured speed is without overshoot and the run-up is below the theoretical result. The effect of the differentiating branch in the controller causes the control signal to respond to the change in the error signal over a wide range with an abrupt,

unpreferred change jerking the amplifier input. The PD controller is not recommended for this application for the above reasons.

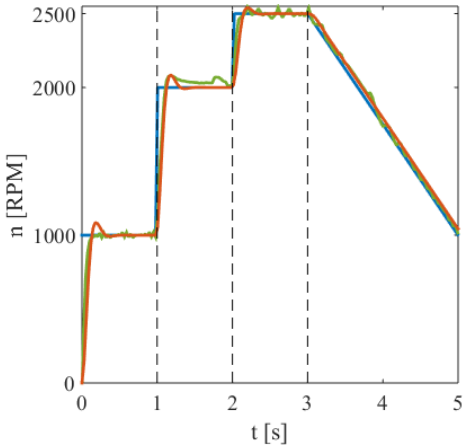


Figure 19. The reference signal tracking of the PI controller with phase margin $\varphi_m = 60°$.

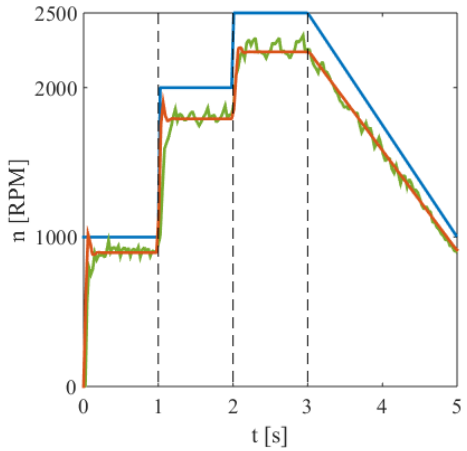


Figure 20. The reference signal tracking of the PD controller with phase margin $\varphi_m = 60°$.

The result of the PI-PD controller is not discussed separately because it is very similar to the result of the PI controller.

To set the sampling period T_s, the following rule of thumb can be used:

$$T_s = \frac{2\pi}{180\omega_c}\Delta\varphi_m, \tag{88}$$

where the maximum value of the phase decay $\Delta\varphi_m$ is set to 5°. The sampling period time is 6.9 ms, 11.5 ms, 4.1 ms, 4.1 ms and 1.9 ms for P, PI, PD, PI-PD ($n = 2$), and PI-PD ($n = 5$) controllers according to the cut-off frequency data of Table 2, respectively. By simply running the microcontroller's `loop()` function, the resulting sampling period time for this implementation is on average around 30 ms, which corresponds to a sampling frequency of roughly 30 Hz, and is larger than the values given by the rule of thumb. However, it can be stated that the controllers perform satisfactorily, very close to the theoretical results. The largest difference in the calculated and measured transient was observed for the PD controller.

Design of the Discrete-Time PID Controllers

When designing a discrete-time PID controller, the first step is to transform the continuous-time transfer function of the plant into a step response equivalent form, where

the sampling period T_s is predefined. The discrete-time transfer function of a DC motor is generally as follows:

$$W_P(z) = M\frac{z - z_1}{(z - p_1)(z - p_2)}, \tag{89}$$

where M is a constant, and the discrete-time model has one zero z_1 and two poles p_1 and p_2. These of course depend on the sampling period. For the transformation, the Matlab R2015a function `c2d` is used. As an example, the transfer function at 50 Hz and 100 Hz is as follows:

$$\begin{aligned} W_P(z) &= 0.8805\frac{z + 0.8055}{(z - 0.8817)(z - 0.5922)}, \\ W_P(z) &= 0.2442\frac{z + 0.8972}{(z - 0.9390)(z - 0.7695)}. \end{aligned} \tag{90}$$

The poles of a discrete-time system can be mapped from the poles of the continuous-time system according to the relation $p_{\text{discrete}} = e^{p_{\text{continuous}} T_s}$.

The design is conducted on a phase margin, in a similar way to the design of continuous-time controllers, by solving the following system of nonlinear equations:

$$\left|W_O\left(e^{j\omega_c T_s}\right)\right| = 1, \quad -180^\circ + \varphi_m = \arg W_O\left(e^{j\omega_c T_s}\right). \tag{91}$$

The system of equations is solved by calling the Matlab R2015a function `fsolve`. The step response of the designed closed-loop control systems are compared in Figure 21, and some characteristic data are summarized in Table 3. The nature of the step responses is of course the same as the results obtained for the continuous-time design (see Figure 17).

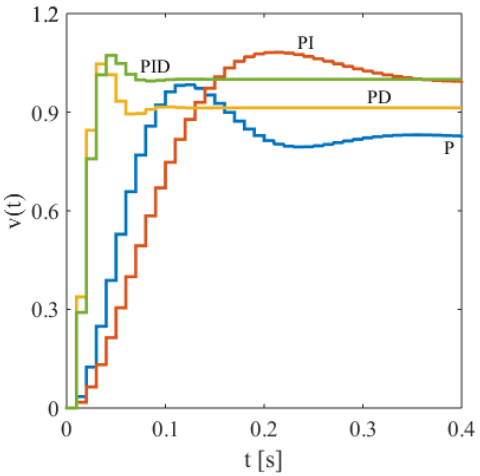


Figure 21. Comparison of closed-loop step response data for the phase margin $\varphi_m = 60^\circ$.

Table 3. Some data of closed-loop systems for the phase margin $\varphi_m = 60^\circ$, the sampling frequency is 100 Hz.

Controller	ω_c[rad/s]	K	v_{max}	v_{st}	Δv[%]	$u_t = \frac{u(0)}{u_{st}}$	u_{st}
P	21.84	0.143	0.98	0.83	18.9	5.78	0.025
PI	12.96	0.072	1.08	1.00	8.89	2.49	0.031
PD	63.66	1.385	1.05	0.91	14.6	49.9	0.027
PI-PD	53.85	1.191	1.07	1.00	7.28	39.2	0.031

By reducing the sampling period time, the results of the discrete-time model converge to the results of the continuous-time model, while the value of the control signal increases at the beginning of the transients.

The result of the controller design is a difference equation, which can be derived, for example, for the PI controller as follows. Start from the transfer function of the controller:

$$W_C(z) = \frac{U(z)}{E(z)} = K_{PI}\frac{z - z_I}{z - 1} = K_{PI}\frac{1 - z_I z^{-1}}{1 - z^{-1}}, \tag{92}$$

from which, after cross multiplication, the following equation can be written:

$$K_{PI}E(z) - K_{PI}z_I E(z)z^{-1} = U(z) - U(z)z^{-1}, \tag{93}$$

to which the following difference equation is given by the shift operator of the z-transform:

$$u[k] = u[k-1] + K_{PI}e[k] - K_{PI}z_I e[k-1]. \tag{94}$$

This equation gives the algorithm of the controller.

The PD controller can be implemented in a similar way as above:

$$u[k] = K_{PD}e[k] - K_{PD}z_D e[k-1], \tag{95}$$

and the algorithm of the PID controller is as follows:

$$u[k] = u[k-1] + K_{PID}e[k] - K_{PID}(z_I + z_D)e[k-1] + K_{PID}z_I z_D e[k-2]. \tag{96}$$

Thus, the k-th value of the control signal is the weighted sum of the $(k-1)$-th value of the control signal, the k-th, $(k-1)$-th and $(k-2)$-th values of the error signal. The controller program is very easy to implement using a general form of

$$W_C(z) = \frac{b_0 z^2 + b_1 z + b_2}{z^2 + a_1 z + a_2} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}, \tag{97}$$

which results in the difference equation:

$$u[k] = -a_1 u[k-1] - a_2[k-2] + b_0 e[k] + b_1 e[k-1] + b_2 e[k-2]. \tag{98}$$

The included coefficients are summarized in Table 4 and in Table 5 at 50 Hz and 100 Hz sampling frequencies, respectively.

Table 4. Parameters of the difference Equation (98), $\varphi_m = 60^\circ$ and $f_s = 50$ Hz.

Controller	a_1	a_2	b_0	b_1	b_2
P	0	0	0.1241	0	0
PI	1.00	0	0.0642	−0.0566	0
PD	0	0	0.4568	−0.2705	0
PI-PD	1.00	0	0.3577	−0.5272	0.1868

Table 5. Parameters of the difference Equation (98), $\varphi_m = 60^\circ$ and $f_s = 100$ Hz.

Controller	a_1	a_2	b_0	b_1	b_2
P	0	0	0.1430	0	0
PI	1.00	0	0.0720	−0.0676	0
PD	0	0	1.3851	−1.0659	0
PI-PD	1.00	0	1.1911	−2.0351	0.8607

The results for discrete-time PID controllers are not discussed separately but are very similar to those for continuous-time PID controllers. The implemented Aruino code of $f_s = 50$ Hz is presented in Appendix B for further use.

3.3.2. State Feedback Controllers

Observer Design to Measure the Current Signal

The discrete-time state-space model of the continuous-time system is obtained by the Matlab R2015a function `c2d`. The sampling period is defined to $T_s = 1/50$ s ($f_s = 50$ Hz), and

$$\mathbf{x}[k+1] = \begin{bmatrix} 0.452 & -0.001956 \\ 13.9 & 0.9552 \end{bmatrix} \mathbf{x}[k] + \begin{bmatrix} 0.08457 \\ 0.9606 \end{bmatrix} u[k], \quad y[k] = \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{x}[k]. \tag{99}$$

The eigenvalues of the system matrix, i.e., the poles of the system, are $p_1 = 0.8937$ and $p_2 = 0.5135$.

The vector $\mathbf{G}$ is determined by Ackermann's formula, with the observer's poles at 0.05. The Matlab R2015a function `acker` gives the result of

$$\mathbf{G} = \begin{bmatrix} 0.0257 \\ 0.9946 \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 0.0846 \\ 0.9606 \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} 0.4520 & -0.0277 \\ 13.8997 & -0.0393 \end{bmatrix}. \tag{100}$$

The approximation of the armature current and speed for a stepped excitation is shown in Figure 22. The measured current signal is quite noisy in this range, but the currents calculated with the continuous-time observer and the discrete-time observer are close to each other. The measured and observed speeds are in perfect agreement.

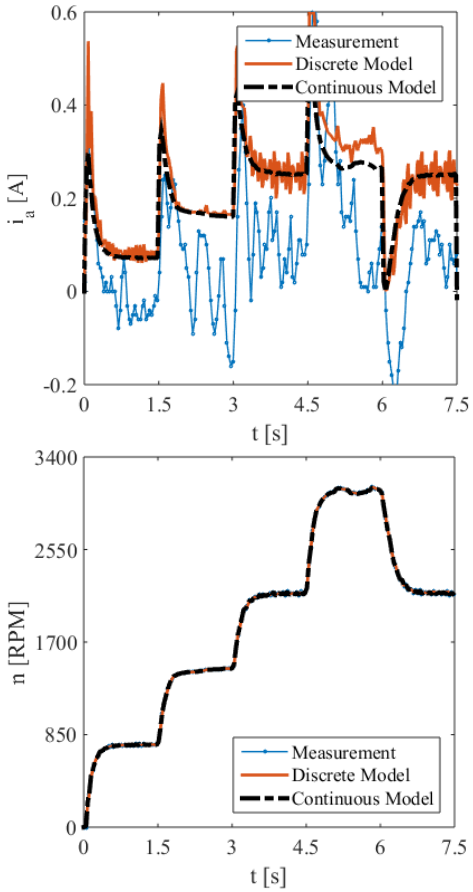


Figure 22. Comparison of measured and observed motor current and speed.

Kalman Filter to Estimate the Current Signal

To determine the covariance matrix of the Kalman filter $\mathbf{R}_v$ and the covariance R_w, the measurement result shown in Figure 23 is used when the motor is excited with a constant PWM signal of 150.

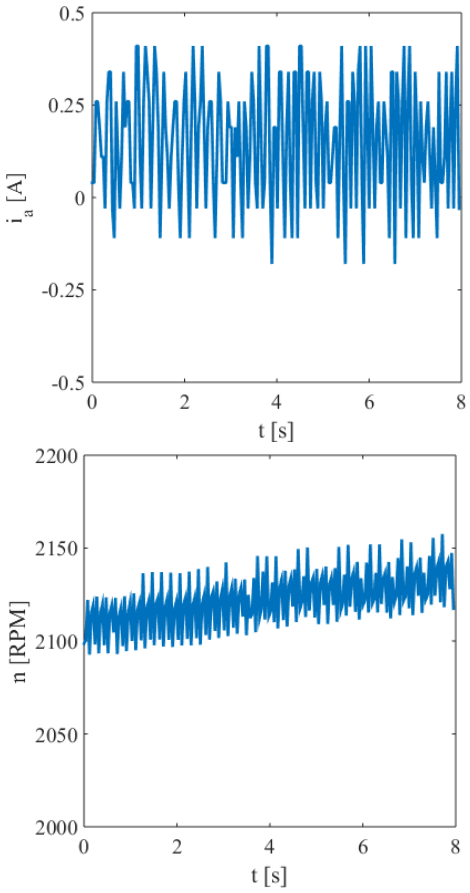


Figure 23. Measured data to set up the Kalman filter parameters.

The mean values calculated from the $n = 201$ sample are $\mu_{i_a} = 0.155$, $\mu_\omega = 222.2334$, and the covariances are as follows:

$$\mathbf{R}_\mathrm{v} = \begin{bmatrix} 0.0266 & -0.0209 \\ -0.0209 & 2.5720 \end{bmatrix}, \quad R_\mathrm{w} = 2.572. \tag{101}$$

The gain of the Kalman filter **G** can be determined by the Matlab R2015a function `dare`, which numerically solves the Riccati Equation (44). Then, **F** and **H** can be calculated:

$$\mathbf{G} = \begin{bmatrix} 0.0043 \\ 0.9583 \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 0.0846 \\ 0.9606 \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} 0.4520 & -0.0062 \\ 13.8997 & -0.0031 \end{bmatrix}. \tag{102}$$

The estimated speed signal and the estimated current signal obtained by the Kalman filter have been used in the control loop.

Design of State Feedback with Pole Placement and Ackermann's Formula

The eigenvalues of the extended system matrix, i.e., the poles of the extended system, are $p_1 = 0.8937$, $p_2 = 0.5135$, and $p_3 = 1$. The last one is due to the integrating effect, the other two poles are from the discrete-time model of the motor.

Two cases are described in more detail.

In the first case, the poles p_1 and p_2 are not moved, the pole $p_3 = 1$ is changed to $p_3 = 1/2$, i.e., the pole connected to the integrator is accelerated. This gives the following gains using Ackermann's formula (see Figure 24, where reference, measured and simulated signals are the blue, green and red lines):

$$\mathbf{p}^\mathrm{T} = \begin{bmatrix} p_1 & p_2 & p_3/2 \end{bmatrix} \quad \Rightarrow \quad \mathbf{k}^\mathrm{T} = \begin{bmatrix} 4.0835 & 0.1610 & 0.7598 \end{bmatrix}. \tag{103}$$

As a result of the experimentation with pole placement, further modification of p_3 (e.g., to 1/4 or 1/5) has practically no effect on the behavior of the closed loop. Changing the pole p_2 also has no effect on the controlled characteristic or the control signal. Moving the pole p_1 is more significant. At the following values, as in the second case, the control signal does not exceed the upper limit, but a faster settling is obtained compared to the previous one (see Figure 25, where reference, measured and simulated signals are the blue, green and red lines):

$$\mathbf{p}^{\mathrm{T}} = \begin{bmatrix} p_1/1{,}5 & p_2 & p_3/2 \end{bmatrix} \quad \Rightarrow \quad \mathbf{k}^{\mathrm{T}} = \begin{bmatrix} 5.9413 & 0.3076 & 2.8886 \end{bmatrix}. \tag{104}$$

Further acceleration, however, will result in overshooting or saturation of the actuating block.

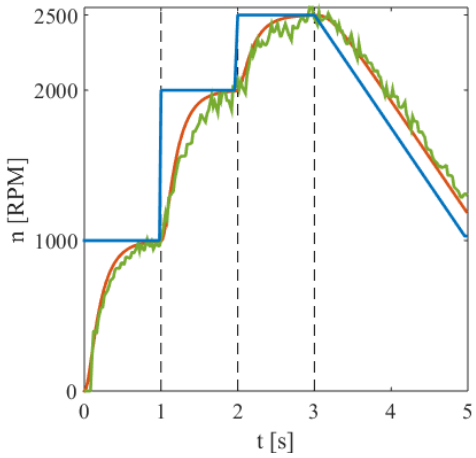


Figure 24. Closed loop system behavior designed by pole placement, case 1.

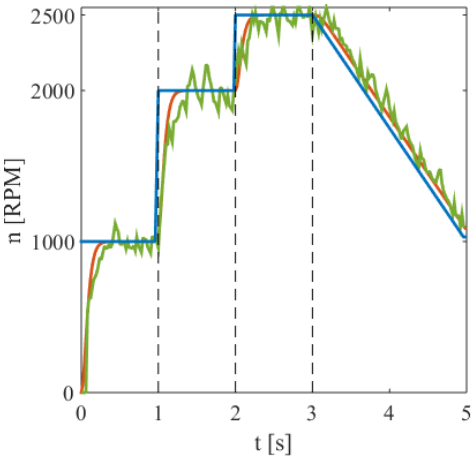


Figure 25. Closed loop system behavior designed by pole placement, case 2.

LQ-Based State Feedback Design

After some trials, the following choice leads to a very similar result to the second case obtained by Ackermann's formula:

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 100 \end{bmatrix}, \quad r = 100 \quad \Rightarrow \quad \mathbf{k}^{\mathrm{T}} = \begin{bmatrix} 2.6691 & 0.1270 & 0.8351 \end{bmatrix}. \tag{105}$$

Controller gains are obtained by the solution of (52) and (53).

A possible discrete-time state feedback controller implementation can be studied in Appendix C for further use.

4. Discussion

The DC motor analysis with real laboratory tests showed the well-known theoretical result that the DC motor can be modeled as a linear plant in the control systems. From the steady-state and switch-on measurements, the linear state space model as well as the transfer function model have been identified, and it is supposed that the model is second order. Measurement results and simulated curves show a very good agreement.

Continuous-time PID controllers have been designed via the well-known phase margin method, and the discretized form of the controllers has been implemented in the Arduino UNO microcontroller. The discrete-time model-based PID controllers are also designed and realized. The two approaches result in the same control signal as well as the same output signal. The continuous-time model is a little bit easier to understand, and the results of the discrete-time model proved that the continuous-time controller can be used efficiently; however, the discrete-time model is closer to the theory of discrete-time controller design. Moreover, discrete-time modeling can be implemented in microcontrollers. More complicated problems should be worked out in the frame of the discrete-time theory because the sampling time can be fixed at the very first step of the design.

The noisy current signal is efficiently measured by the state observer as well as by the Kalman filter. It is highlighted that the Kalman filter did not result in a much better noise reduction, meaning that the Kalman filter parameters should be adjusted as a future task as well as the noise model parameters. Practically, in this application the deterministic state observer and the Kalman filter gave the same measurement results.

State feedback controller design has been realized and the pole placement has been studied comprehensively. LQ optimal controller was also studied. The design of the parameters, i.e., the matrix $\mathbf{Q}$ and the parameter R should be improved, to reach a predefined closed loop system output and control signal [55].

There are some other linear controllers that can be realized, e.g., the two degrees of freedom controller, dead-beat controller, optimal controller with conditions on the control signal, model predictive controller and so forth. Linear control suffers from obvious limitations in terms of lack of disturbance handling and modeling uncertainties. The effect of uncertain model parameters and disturbances can be decreased by the integral term of the control loops mentioned in the paper; however, uncertainty and disturbance can be handled efficiently by robust control methods [56,57]. This paper has not worked on these questions, these can be the focus of fruitful future tasks. The paper mentioned only a full-order observer to estimate the state vector elements; however, reduced-order state estimation can be applied. This can be a future task as well.

The test bench is easy to realize, the motor, the sensors and the microcontroller are very easy to build and assemble, and it can be a useful tool for engineering students in undergraduate systems and control lectures and laboratories. This is why the Arduino codes are inserted for further studies.

There are, of course, many other microcontroller-based platforms for solving the problem presented in this review. The Arduino UNO is a cheap and simple solution that is also very useful in undergraduate control engineering education. For more complex problems that require more computing power or special tasks, other types of hardware should be used.

Funding: This research received no external funding.

Data Availability Statement: Data is contained within the article.

Acknowledgments: Special thanks to Gábor Szüts and Dániel Marcsa for building the first version of the test bench.

Conflicts of Interest: The author declares no conflicts of interest.

Appendix A

Listing A1. Arduino based code for continuous time PID-type controllers

```
#define PWM_OUT  3
#define SPEED_IN~2

int ctrl=0; // 0=P, 1=PI, 2=PD, 3=PID

volatile unsigned long t0;
volatile bool start=false;
volatile float Measured_Speed=0;

unsigned long ti0;
float Reference_Speed=1000;
int   PWM=0;
float interror=0;
float u=0, error_old=0, u_old=0, error_old2=0, u_old2=0;
// phase margin is 60 degree:
float KP=0.1746;
float KPI=0.0842, TI=0.1588;
float KPD=0.2605, TD=0.0382, TDv=TD/2;
float KPID=0.2605;
unsigned long T=1.0e6;

void setup() {
Serial.begin(9600);

pinMode(PWM_OUT, OUTPUT);
pinMode(SPEED_IN, INPUT);

TCCR2B = TCCR2B & 0b11111000 | 0x01;

attachInterrupt(digitalPinToInterrupt(SPEED_IN),
MEASURE_ISR, RISING);

analogWrite(PWM_OUT, 0);
ti0 = micros();
}

void loop() {
unsigned long t1 = micros();

if (t1<5*T){
float dt = (t1-ti0)/1e6;
ti0 = t1;

if (t1<T)                          {Reference_Speed=1000;}
else if ((t1>=T) && (t1<2*T))      {Reference_Speed=2000;}
else if ((t1>=2*T) && (t1<3*T))    {Reference_Speed=2500;}
else if ((t1>=3*T) && (t1<5*T))    {Reference_Speed=
4750-750*(float)t1/1e6;}

noInterrupts();
float error = (Reference_Speed - Measured_Speed)/60.0*2*3.1415;
```

```
interrupts();

interror =  interror + error*dt;
if (ctrl==0){
u = KP*error;
}
if (ctrl==1){
u = KPI*error+KPI/TI*interror;
u = u_old+KPI*(1+dt/TI)*error-KPI*error_old;
u = u_old+KPI*(1+dt/2.0/TI)*error
-KPI*(1-dt/2.0/TI)*error_old;
u = u_old+KPI*error-KPI*(1-dt/TI)*error_old;
}
if (ctrl==2){
u = KPD*error+KPD*TD*(error-error_old)/dt;
u = TDv/(dt+TDv)*u_old+KPD*(dt+TD)/(dt+TDv)*error
-KPD*TD/(dt+TDv)*error_old;
u = -(dt-2*TDv)/(dt+2*TDv)*u_old
+KPD*(dt+2*TD)/(dt+2*TDv)*error
+KPD*(dt-2*TD)/(dt+2*TDv)*error_old;
float q = exp(-dt/TDv);
u = q*u_old+KPD*TD/TDv*error+KPD*(1-q-TD/TDv)*error_old;
}
if (ctrl==3){
u = KPID*error+KPID/TI*interror
+KPID*TD*(error-error_old)/dt;
u = (dt+2*TDv)/(dt+TDv)*u_old-TDv/(dt+TDv)*u_old2
+KPID*((dt+TD)*(dt+TI)/TI/(dt+TDv)*error
-(dt*(TD+TI)+2*TI*TD)/TI/(dt+TDv)*error_old
+TD/(dt+TDv)*error_old2);
u = 4*TDv/(dt+2*TDv)*u_old+(dt-2*TDv)/(dt+2*TDv)*u_old2
+KPID*((dt+2*TD)*(dt+2*TI)/2/TI/(dt+2*TDv)*error
+(dt*dt-4*TI*TD)/TI/(dt+2*TDv)*error_old
+(dt-2*TD)*(dt-2*TI)/2/TI/(dt+2*TDv)*error_old2);
}

error_old2 = error_old;
u_old2     = u_old;
error_old  = error;
u_old      = u;

PWM = (int)(21.96788*u+42.477);
if (PWM>250) {PWM=250;}
if (PWM<50)  {PWM=50;}
analogWrite(PWM_OUT, PWM);

float voltage = (5.0/1023.0)*analogRead(A1);
float Measured_Current = (voltage+0.01-2.5)/0.066;

Serial.print(t1);
Serial.print(',');
Serial.print(PWM);
Serial.print(',');
Serial.print(Reference_Speed);
```

```
106 Serial.print(',');
107 Serial.print(Measured_Speed);
108 Serial.print('\n');
109 }
110 else{analogWrite(PWM_OUT, 0);}
111 }
112
113 void MEASURE_ISR()
114 {
115 if (!start){
116 start = true;
117 t0    = micros();
118 }
119 else{
120 unsigned long t1 = micros();
121 unsigned long dt = t1-t0;
122 t0 = t1;
123 Measured_Speed = 60.0e6 / dt / 8.0;
124 }
125 }
```

Appendix B

Listing A2. Arduino based code for discrete time PID-type controllers

```
1  #define PWM_OUT  3
2  #define SPEED_IN~2
3
4  volatile unsigned long t0;
5  volatile bool start=false;
6  volatile float Measured_Speed=0;
7  volatile bool control=false;
8
9  float Reference_Speed=1000;
10 float interror=0;
11 float u=0, error_old=0, u_old=0, error_old2=0;
12 int PWM=0;
13 unsigned long T=1.0e6;
14
15 void setup() {
16 Serial.begin(9600);
17
18 pinMode(PWM_OUT, OUTPUT);
19 pinMode(SPEED_IN, INPUT);
20
21 analogWrite(PWM_OUT, 0);
22
23 TCCR2B = (TCCR2B & 0b11111000) | 0x01;
24 attachInterrupt(digitalPinToInterrupt(SPEED_IN),
25 MEASURE_ISR, RISING);
26
27 TCCR1A = 0;
28 TCCR1B = 0;
29 TCNT1  = 0;
30 OCR1A  = 1249; //=(16*10^6)/(256*50)-1 (<65536) 50Hz
```

```
TCCR1B |= (1 << WGM12);
TCCR1B |= (1 << CS12);
TIMSK1 |= (1 << OCIE1A);
}

void loop() {
unsigned long t1 = micros();

if (t1<5*T){
if (control){
if (t1<T)                          {Reference_Speed=1000;}
else if ((t1>=T) && (t1<2*T))      {Reference_Speed=2000;}
else if ((t1>=2*T) && (t1<3*T))    {Reference_Speed=2500;}
else if ((t1>=3*T) && (t1<5*T))    {Reference_Speed=4750
-750*(float)t1/1e6;}

noInterrupts();
float error = (Reference_Speed---Measured_Speed)/60.0*2*3.1415;
interrupts();

u           = 0.1241*error;                          //P
u           = u_old+0.0642*error-0.0566*error_old;   //PI
u           = 0.4568*error-0.2705*error_old;         //PD
u           = u_old+0.3577*error-0.5272*error_old
+0.1868*error_old2;                //PID

error_old2  = error_old;
error_old   = error;
u_old       = u;

PWM = (int)(21.96788*u+42.477);
if (PWM>250) {PWM=250;}
if (PWM<50)  {PWM=50;}
analogWrite(PWM_OUT, PWM);

Serial.print(t1);
Serial.print(',');
Serial.print(Reference_Speed);
Serial.print(',');
Serial.print(Measured_Speed);
Serial.print('\n');

control = false;
}
}
else{analogWrite(PWM_OUT, 0);}
}

ISR(TIMER1_COMPA_vect){
control = true;
}

void MEASURE_ISR()
{
```

```
85 if (!start){
86 start = true;
87 t0    = micros();
88 }
89 else{
90 unsigned long t1 = micros();
91 unsigned long dt = t1-t0;
92 t0 = t1;
93 Measured_Speed = 60.0e6 / dt / 8.0;
94 }
95 }
```

Appendix C

Listing A3. Arduino based code for discrete time state feedback controllers

```
1  #define PWM_OUT  3
2  #define SPEED_IN~2
3
4  volatile unsigned long t0;
5  volatile bool start=false;
6  volatile float Measured_Speed=0;
7  volatile bool control=false;
8
9  float Reference_Speed=1000;
10 float u=0, d=0;
11 int PWM=0;
12 unsigned long T=1.0e6;
13
14 // State Observer
15 float F[4]  = {0.4520, -0.0277, 13.8997, -0.0393};
16 float G[2]  = {0.0257, 0.9946};
17 float H[2]  = {0.0846, 0.9606};
18
19 // State Estimator
20 float F[4]  = {0.4520, -0.0062, 13.8997, -0.0031};
21 float G[2]  = {0.0043, 0.9583};
22 float H[2]  = {0.0846, 0.9606};
23
24 // Gains
25 float k[3]  = {2.6691,    0.1270,    0.8351};
26 float xk[2] = {0.0, 0.0};
27 float Ts;
28
29 void setup() {
30 Serial.begin(9600);
31
32 pinMode(PWM_OUT, OUTPUT);
33 pinMode(SPEED_IN, INPUT);
34
35 analogWrite(PWM_OUT, 0);
36
37 TCCR2B = (TCCR2B & 0b11111000) | 0x01;
38 attachInterrupt(digitalPinToInterrupt(SPEED_IN),
39 MEASURE_ISR, RISING);
```

```
40
41 TCCR1A = 0;
42 TCCR1B = 0;
43 TCNT1  = 0;
44 OCR1A  = 1249; //=(16*10^6)/(256*50)-1 (<65536) 50Hz
45 TCCR1B |= (1 << WGM12);
46 TCCR1B |= (1 << CS12);
47 TIMSK1 |= (1 << OCIE1A);
48
49 Ts = 0.02;
50 }
51
52 void loop() {
53 unsigned long t1 = micros();
54
55 if (t1<5*T){
56 if (control){
57 if (t1<T)                          {Reference_Speed=1000;}
58 else if ((t1>=T) && (t1<2*T))      {Reference_Speed=2000;}
59 else if ((t1>=2*T) && (t1<3*T))    {Reference_Speed=2500;}
60 else if ((t1>=3*T) && (t1<5*T))    {Reference_Speed=4750
61 -750*(float)t1/1e6;}
62
63 noInterrupts();
64 float error = (Reference_Speed---Measured_Speed)/60.0*2*3.1415;
65 float y     = Measured_Speed/60.0*2*3.1415;
66 interrupts();
67
68 d = d + error*Ts;
69 float xk0 = F[0]*xk[0] + F[1]*xk[1] + G[0]*y + H[0]*u;
70 float xk1 = F[2]*xk[0] + F[3]*xk[1] + G[1]*y + H[1]*u;
71 xk[0] = xk0;
72 xk[1] = xk1;
73 u = -(k[0]*xk[0]+k[1]*xk[1])+k[2]*d;
74
75 PWM = (int)(21.96788*u+42.477);
76 if (PWM>250) {PWM=250;}
77 if (PWM<50)  {PWM=50;}
78 analogWrite(PWM_OUT, PWM);
79
80 Serial.print(t1);
81 Serial.print(",");
82 Serial.print(Reference_Speed);
83 Serial.print(",");
84 Serial.print(Measured_Speed);
85 Serial.println();
86
87 control = false;
88 }
89 }
90 else{analogWrite(PWM_OUT, 0);}
91 }
92
93 ISR(TIMER1_COMPA_vect){
```

```
control = true;
}

void MEASURE_ISR()
{
if (!start){
start = true;
t0    = micros();
}
else{
unsigned long t1 = micros();
unsigned long dt = t1-t0;
t0 = t1;
Measured_Speed = 60.0e6 / dt / 8.0;
}
}
```

References

1. Pyrhönen, J.; Jokinen, T.; Hrabovcová, V. *Design of Rotating Electrical Machines*; John Wiley and Sons: Hoboken, NJ, USA, 2008.
2. Kopylov, I.P. *Mathematical Models of Electric Machines*; Mir Publishers, Moscow, Russia, 1984.
3. Fitzgerald, A.E.; Kingsley, C.; Umans, S.D. *Electric Machinery*; McGraw-Hill: Boston, MA, USA, 2003.
4. Veltman, A. *Fundamentals of Electrical Drives*; Springer International Publishing AG: Cham, Switzerland, 2016.
5. Pyrhönen, J.; Hrabovcová, V.; Semken, R.S. *Electrical Machine Drives Control—An Introduction*; John Wiley and Sons: Hoboken, NJ, USA, 2016.
6. Xia, C. *Permanent Magnet Brushless DC Motor Drives and Controls*; John Wiley and Sons: Hoboken, NJ, USA, 2012.
7. Stratton, J.A. *Electromagnetic Theory*; Wiley-IEEE Press: Hoboken, NJ, USA, 2007.
8. Jackson, J.D. *Classical Electrodynamics*; Wiley: Hoboken, NJ, USA, 2021.
9. Maheriya, S.C.; Parikh, P.A. A review: Modelling of brushed DC motor and various type of control methods. *J. Res.* **2016**, *1*, 18–23.
10. Afifa, R.; Ali, S.; Pervaiz, M.; Iqbal, J. Adaptive Backstepping Integral Sliding Mode Control of a MIMO Separately Excited DC Motor. *Robotics* **2023**, *12*, 105. [CrossRef]
11. Keviczky, L.; Bars, R.; Hetthéssy, J.; Bányász, C.s. *Control Engineering*; Springer: Berlin/Heidelberg, Germany, 2019.
12. Keviczky, L.; Bars, R.; Hetthéssy, J.; Bányász, C.s. *Control Engineering: MATLAB Exercises*; Springer: Berlin/Heidelberg, Germany, 2018.
13. Dukkipati, R.V. *Analysis and Design of Control Systems Using Matlab*; New Age International Limited, Ansari Road: Daryaganj, India, 2006.
14. Lantos, B.; Márton, L. *Nonlinear Control of Vehicles and Robots*; Springer: Berlin/Heidelberg, Germany, 2010.
15. Brosilow, C.; Joseph, B. *Techniques of Model-Based Control*; Prentice Hall Professional: Upper Sanddle River, NJ, USA, 2002.
16. Levine, S.W. *The Control Handbook*; CRC Press: Boca Raton, FL, USA, 1996.
17. Dubey, S.; Srivastava, S.K. A PID controlled real time analysis of DC motor. *Int. J. Innov. Res. Comput. Commun. Eng.* **2013**, *1*, 1965–1973.
18. Ogata, K. *Modern Control Engineering*; Prentice Hall: Saddle River, NJ, USA, 1990.
19. Slotine, E.; Li, W. *Applied Nonlinear Control*; Prentice Hall: Saddle River, NJ, USA, 1991.
20. Bitriá, R.; Palacín, J. Optimal PID control of a brushed DC motor with an embedded low-cost magnetic quadrature encoder for improved step overshoot responses in a mobile robot application. *Sensors* **2022**, *22*, 7817.
21. Control Tutorials for MATLAB and Simulink. Available online: https://ctms.engin.umich.edu/CTMS (accessed on 2 May 2024).
22. Iqbal, U.; Samad, A.; Nissa Z.; Iqbal, J. Embedded control system for AUTAREP—A novel autonomous articulated robotic educational platform. *Teh. Vjesn.* **2014**, *21*, 1255–1261.
23. Sagar, J.D. DC Motor Control using PID Controller. *Int. Res. J. Eng. Technol.* **2020**, *7*, 1765–1769.
24. Wati, T.; Subiyanto S. Simulation model of speed control DC motor using fractional order PID controller. *J. Phys. Conf. Ser.* **2020**, *1444*, 012022. [CrossRef]
25. Rastogi, A.; Tiwari, P. Optimal tuning of fractional PID controller for DC motor speed control using particle swarm optimization. *Int. J. Soft Comput. Eng.* **2013**, *3*, 150–157.
26. Dozein, M.G.; Gholami, A.; Kalantar, M. Speed control of DC motor using different optimization techniques based PID controller. *J. Basic Appl. Sci. Res.* **2011**, *2*, 6488–6494.
27. Jaiswal, M.; Phadnis, M. Speed control of DC motor using genetic algorithm based PID controller. *Int. J. Adv. Res. Comput. Sci. Softw. Eng.* **2013**, *3*, 247–253.
28. Lichtman, A.; Fuchs, P. Theory of PI controller and introduction to implementation for DC motor controls. In Proceedings of the 2017 Communication and Information Technologies Conference, Vysoke Tatry, Slovakia, 4–6 October 2017; pp. 1–5.

29. Michel, O.; Daou, R.A.Z.; Moreau, X. The effects of nonlinearities on the robustness of various controllers: Application on a DC motor. In Proceedings of the 2018 IEEE International Multidisciplinary Conference on Engineering Technology, Beirut, Lebanon, 14–16 November 2018; pp. 1–6.
30. Basilio, J.C.; Moreira, V. State-space parameter identification in a second control laboratory. *IEEE Trans. Educ.* **2004**, *47*, 204–210. [CrossRef]
31. Niro, L.; Chaves, W.d.S.; Kaneko, E.H.; Mollon, M.F.; Montezuma, M.A.F. Construction of and instrumentation kit for identification and control of DC motors. *Int. J. Adv. Eng. Res. Sci.* **2017**, *4*, 121–126. [CrossRef]
32. José do Carmo, M.; Teixeira, P.; Evaristo Rodrigues Costa, J.; Cláudio Gambos Lopes, L.; José Gomes, F. Mathematical modeling by identification: A case study in the laboratory of control applied to the identification of a servo mechanism. *J. Math. Model. Appl.* **2011**, *1*, 33–43.
33. Othman, K.A.; Kamal, M.M.; Mamat, N.; Hamzah, N. System identification of discrete model for DC motor positioning. In Proceedings of the 8th WSEAS International Conference on Circuits, Systems, Electronics, Control & Signal Processing, Cambridge, UK, 21–23 February 2009; World Scientific and Engineering Academy and Society (WSEAS): Stevens Point, WI, USA, 2009; pp. 212–216.
34. Bature, A.A.; Muhammad, M.; Abdullahi, A.M. Parameter Identification of a class of DC motor. *Int. J. Res. Eng. Sci.* **2013**, *1*, 69–72.
35. Kara, T.; Eker, I. Nonlinear closed-loop direct identification of a DC motor with load for low speed two-directional operation. *Electr. Eng.* **2004**, *86*, 87–96. [CrossRef]
36. Kara, T.; Eker, I. Nonlinear modeling and identification of a DC motor for bidirectional operation with real time experiments. *Energy Convers. Manag.* **2003**, *45*, 1087–1106. [CrossRef]
37. Lyshevski, S.E. Nonlinear control of mechatronic systems with permanent-magnet DC motors. *Energy Convers. Manag.* **1999**, *9*, 539–552. [CrossRef]
38. Sabir, M.M.; Khan, J.A. Optimal design of a PID controller for the speed control of DC motor by using metaheuristics techniques. *Adv. Artificaial Neural Syst.* **2014**, *2014*, 126317. [CrossRef]
39. Perez-Gomez, C.A.; Liceaga-Castro, J.U.; Siller-Alcala, I.I. Hard dead zone and friction modeling and identification of a permanent magnet DC motor non-linear model. *WSEAS Trans. Syst. Control* **2020**, *15*, 527–536. [CrossRef]
40. Beloiu, R. Dynamic determination of DC motor parameters—Simulation and testing. In Proceedings of the 6th International Conference on Electronics, Computers and Artificial Intelligence, Bucharest, Romania, 23–25 October 2014; pp. 13–18.
41. Khalil, H.K. *Nonlinear systems*; Prentice Hall: Upper Saddle River, NJ, USA, 2002.
42. Sun, L.; Liu, W.; Sano, A. Identification of a dynamical system with input nonlinearity. *IEE Proc Control Theory Appl.* **1999**, *146*, 41–51. [CrossRef]
43. Nordin, M.; Gutman, P. Controlling mechanical systems with backlash–a survey. *Automatica* **2002**, *38*, 1633–1649. [CrossRef]
44. Ljung, L. *System Identification: Theory for the User*; CRC Press: Upper Sanddle River, NJ, 1999.
45. Söderström, T.; Stoica, P. *System Identification*; Prentice Hall: Cambridge, UK, 1989.
46. Hassan, M.M.; Aly, A.A.; Rashwan, A.F. Different identification methods with application to a DC motor. *J. Eng. Sci.* **2007**, *35*, 1481–1493. [CrossRef]
47. Saab, S.S.; Kaed-Bey, A. Parameter identification of a DC motor: An experimental approach. In Proceedings of the 8th IEEE International Conference on Electronics, Circuits and Systems, Valletta, Malta, 2–5 September 2001; pp. 981–984.
48. Dub, M.; Jalovecky, R. DC motor experimental parameter identification using the Nelder-Mead simplex method. In Proceedings of the 14th Power Electronics and Motion Control Conference, Ohrid, Macedonia, 6–8 September 2010; pp. S4-9–S4-11.
49. Wu, W. DC motor identification using spoeed step responses. In Proceedings of the 2010 American Control Conference, Baltimore, MD, USA, 30 June–2 July 2010; pp. 1937–1941.
50. Arduino UNO. Available online: https://docs.arduino.cc/hardware/uno-rev3/ (accessed on 2 May 2024).
51. DC Electric Motors. Available online: https://www.transtecno.com/wordpress/wp-content/uploads/2015/06/H_EC-DC-electric-motors_210513_0521.pdf (accessed on 2 May 2024).
52. GP1A22LC Datasheet—Sharp Corporation. Available online: https://www.alldatasheet.com/datasheet-pdf/pdf/42759/SHARP/GP1A22LC.html (accessed on 2 May 2024).
53. ACS712 Datasheet—Allegro MicroSystems. Available online: https://www.alldatasheet.com/datasheet-pdf/pdf/168326/ALLEGRO/ACS712.html (accessed on 2 May 2024).
54. Szüts, G. DC Motor Speed Control (in Hungarian). Bachelor Thesis, Széchenyi István University, Győr, Hungary, 2014.
55. Kumar, V.E.; Jerome, J. Algebraic Riccati equation based Q and R matrices selection algorithm for optimal LQR applied to tracking control of 3rd order magnetic levitation system. *Arch. Electr. Eng.* **2016**, *65*, 151–168. [CrossRef]
56. Zhou, K. *Essential of Robust Control*; Prentice Hall: Upper Saddle River, NJ, USA, 1999.
57. Iqbal, J. Modern Control Laws for an Articulated Robotic Arm: Modeling and Simulation. *Eng. Technol. Appl. Sci. Res.* **2019**, *9*, 4057–4061. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.