UDC 681.5:62-83
DOI: 10.15587/1729-4061.2020.195719

SYNTHESIS OF AUTOMATIC SPEED CONTROL SYSTEM OF LABORATORY RESEARCH BENCH DRIVE MOTOR ON THE BASIS OF DISCRETE TIME EQUALIZER

O. Sheremet
Doctor of Technical Sciences, Associate Professor
Department of Electromechanical Systems of Automation and Electric Drive
Donbass State Engineering Academy
Akademichna str., 72, Kramatorsk, Ukraine, 84313
E-mail: sheremet-oleksii@ukr.net

O. Sadovoi
Doctor of Technical Sciences, Professor
Department of Electrotechnic and Electromechanic
Dniprovsky State Technical University
Dniprobudivska str., 2, Kamyanske, Ukraine, 51918
E-mail: sadovoyav@ukr.net

Дослідження базується на використанні методу дискретного часового еквалайзера для здійснення синтезу та практичної реалізації системи автоматичного керування швидкістю електроприводу постійного струму. Для виконання експериментальних досліджень створено лабораторно-дослідний стенд.

Синтез систем автоматичного керування методом дискретного часового еквалайзера відрізняється від традиційного підпорядкованого регулювання координат або метода узагальненого характеристичного полінома повною відмовою від використання бажаних характеристичних поліномів. Такий підхід дозволяє отримати бажані динамічні та статичні властивості системи виключно виходячи з бажаної перехідної функції, яка повинна бути наближеною до природного характеру протікання перехідних процесів (монотонного, аперіодичного або коливального).

Інтегроване середовище проектування Code Composer Studio дозволило практично реалізувати запропоновані дискретні часові еквалайзери, обернену модель об'єкта керування та блок модифікації зворотного перетворення у вигляді спеціальних підпрограм для мікроконтролера Texas Instruments TMS320F28335 – макросів на мові програмування C/C++.

Побудоване у відповідності до розробленої функціональної схеми взаємодії макросів основне тіло керуючої програми надало можливість для проведення експериментальних досліджень із застосуванням як лише основного каналу керування з одним дискретним часовим еквалайзером, так і комбінованого керування з двома дискретними часовими еквалайзерами (основним та компенсуючим). Оскільки весь програмний код, використаний під час досліджень, написано мовою програмування високого рівня C/C++ з використанням об'єктно-орієнтованих підходів, то він є апаратно незалежним від типу мікропроцесора і з легкістю може бути перенесений на іншу апаратну базу

Ключові слова: дискретний часовий еквалайзер,мікроконтролер, автоматизована система керування, двигун постійного струму

Received date 11.12.2019
Accepted date 14.02.2020
Published date 28.02.2020

1. Introduction

The synthesis of regulators is one of the main problems solved in the theory of automatic control. In a broad sense, this task is to determine the composition and structure of the automatic control system, as well as the parameters of all its components, based on some set of technical requirements [1].

In the synthesis of automated electromechanical systems, in most cases, methods are used that do not take full advantage of modern computer technology and are based on the concept of a standard characteristic polynomial [2]. The developer should select the desired characteristic polynomial from a pre-known list, based on some technical recommendations, nomograms and instructions.

The widespread occurrence and introduction of computer and microcontroller technology into industrial production and minimizing the impact of human factors are some of the most significant trends in the development of modern manufacturing processes. Against this background, the programmatic assignment of the desired dynamic properties of a technical object is relevant, in which all computational work is performed by modern computing equipment without the use of standard characteristic polynomials [3].

The process of defining the desired dynamic properties of a technical object should be carried out programmatically. So that most computational work can be performed using the program code that, without using the theory of standard characteristic polynomials, takes into account the basic dynamic features of a real control object and gives it the desired dynamic properties. The user's work with this approach is minimized – he only sets the desired transition function graphically or in the form of a set of points or values at intervals. Fig. 1 shows an example of setting the desired transition function $y(t)$.

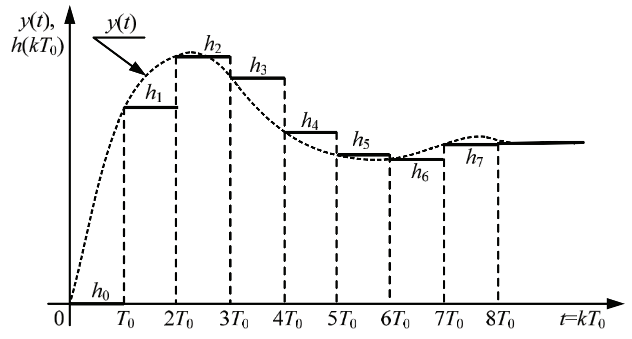


Fig. 1. Quantized transition function

The desired transition function (Fig. 1) is divided (quantized) over the whole time segment into 8 parts, the duration of each of them is T_0. At the times of quantization, the signal amplitude is fixed at the corresponding levels h_0, h_1, h_2, ... , h_7, forming a lattice function $h(kT_0)$, where k is the step number. A discrete controller that provides tuning for such quantized desired transient functions is called a discrete time equalizer [4].

2. Literature review and problem statement

The closest in technical features to control systems with discrete time equalizers are discontinuous control systems. As shown in [5], such control can be successfully applied to systems with significant nonlinearities and some parametric uncertainties. Discontinuous control systems are described by differential equations with discontinuities of functions in the right-hand sides, as a result of which the generating point characterizing the instantaneous values of the coordinates will move along parts of the phase trajectories, and with a certain ratio of parameters, this movement occurs in the sliding mode. Sliding can also be carried out within a certain surface, for example, in [6] using sliding surfaces increases the stability of nonlinear systems with uncertainties.

The basic element for the technical implementation of discontinuous control systems is a relay. Nowadays, relays are mostly programmatically implemented on relatively simple microcontrollers. For example, in [7], an Arduino Mega 2560 microcontroller is used to implement a programmable relay switch. Unlike discontinuous control systems, systems with a discrete time equalizer require complex mathematical calculations in real time and the capabilities of entry-level microcontrollers are not enough.

Controllers with a Digital Signal Processor (DSP) core have the processing power necessary for the software implementation of a discrete time equalizer. The theory and practice of industrial operation of DSP controllers are nowadays well studied and documented [8], educational institutions use DSP controllers in laboratory work on digital signal processing in real time [9].

A characteristic feature of DSP controllers is the ability to read two operands from memory and transfer them to the central processing unit (CPU) in one clock cycle. To do this, they have two independent bus systems called the "program bus" and the "data bus". This approach improves the performance of the microcontroller control system in real time and is called "Harvard architecture". It is devices with Harvard architecture that are used to solve the most complex tasks of processing information in real time. So, in [10], a project of a microprocessor device based on Harvard architecture was presented, which implements a symmetric algorithm for block encryption of data (Advanced Encryption Standard – AES). Currently, AES is one of the most common encryption algorithms and support of its acceleration at the hardware level is present in most modern microprocessors for personal computers.

Since the synthesis of regulators by the discrete time equalizer method is a development of the authors of the paper, there are no known software implementations on modern microcontrollers. The discrete time equalizer implements transition functions with a finite settling time, which is clearly demonstrated in [4].

The well-known mathematical apparatus used for the software implementation of controllers with a finite settling time (compensatory or aperiodic) [11] is not suitable for tuning to the desired transition function, defined graphically or as a set of points. Such controllers provide a finite settling time only when the model parameters exactly match the parameters of the object and the presence of deviations can lead to oscillations in a closed system. Therefore, the use of aperiodic controllers is limited only to linear objects with a significant margin of stability. In [12], a method for constructing an inverse linear-quadratic controller for systems with variable delay was proposed. However, this method is not universal; it can be applied only in dynamical systems described by linear differential equations with a quality index in the form of a quadratic functional.

Therefore, the rejection of standard characteristic polynomials in the synthesis of automated electromechanical systems requires their replacement with another mathematical apparatus, which would be the basis for creating regulators with specified quality factors in static and dynamic modes of operation. As such a basis, the desired transition functions in a numerical (discrete) form can be considered. For the technical implementation of this approach, it is necessary to create the appropriate hardware-software complex in the laboratory.

3. The aim and objectives of the study

The aim of this study is synthesizing an automatic speed control system of a drive motor based on a discrete time equalizer and its technical implementation on a laboratory research bench, which is intended for the study of direct current (DC) drives.

To achieve this aim, the following objectives were set:

– to create a laboratory research bench with a modern power semiconductor converter and a microcontroller suitable for software implementation of a discrete time equalizer;

– to analyze the general approaches to the control systems synthesis for direct current electric drives based on a discrete time equalizer with partial compensation of the control object dynamic properties;

– to determine the transfer functions and build block diagrams of the main elements of the automatic speed control system of the laboratory research bench drive motor: a discrete time equalizer, an inverse model of the control object and an inverse transformation modification unit;

– to carry out experimental research.

4. Creation of a laboratory research bench suitable for software implementation of a discrete time equalizer

A laboratory research bench created to carry out experimental research consists of the following elements:

– drive motor;
– generator;
– electric power consumers creating a load;
– ammeter;
– electric transformer;
– rectifier;
– smoothing filter;
– rheostat;
– power converter with signal microcontroller;
– feedback sensor;
– external module for performing analog-to-digital conversion;
– laptop with the necessary software.

The appearance of the laboratory research bench is shown in Fig. 2. The drive motor of the laboratory research bench is a KPA–561–U2 separately excited DC motor with a rated power of 90 W.

The KPA–561–U2 motor can operate in a direct-current network or from a rectifier, has a right rotation direction, but its short-term reversal is also possible. KPA–561–U2 motors are used to drive automatic welding machines, semi-automatic machines and other mechanisms. The shaft of the KPA–561–U2 drive motor is mechanically connected to the shaft of the DPM 290/3.8/30 motor, operating in the generator mode. The DPM 290/3.8/30 permanent-magnet motor is designed to drive various mechanisms of short-term, repeated-short-term and continuous operation in industrial automation equipment, teleautomatics, and electronic engineering.

The excitation winding of the drive motor is powered by a single-phase bridge full-wave rectifier, the input of which receives voltage from the secondary transformer winding.

The anchor winding of the drive motor is connected to the power converter board from the Texas Instruments TMDSHVMTRPFCKIT (High Voltage Motor Control and PFC Development Kit) development kit.

The TMDSHVMTRPFCKIT development kit includes the following components:

– F28035 and F28335 control boards;
– Digital Motor Control (DMC) power converter board installed in a plastic case covered by protective plexiglass;
– USB–A – USB–B cable;
– Banana Plug cable;
– external power supply;
– disk with software.

Fig. 3 shows the DMC board with an installed F28335 control board (BS1 and BS5 connectors are connected by Banana Plug cable when powering the DMC board from an external unit). The power section of the DMC consists of a DC link, a Power Factor Correction (PFC) module, and a three-phase inverter.

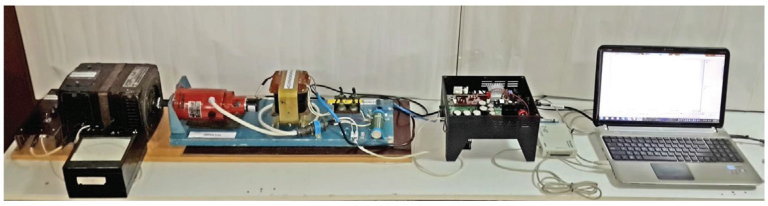


Fig. 2. Laboratory research bench

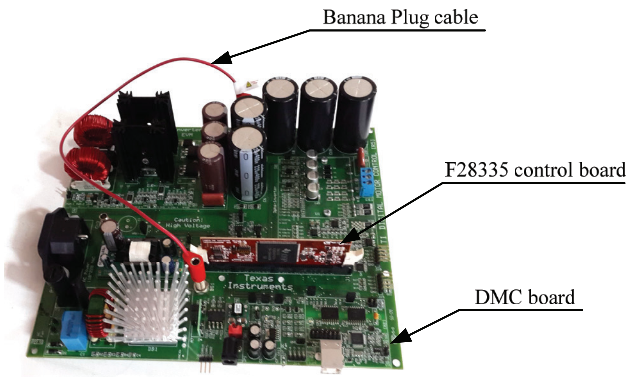


Fig. 3. DMC board with F28335 control board installed

The TMDSHVMTRPFCKIT comes with two control boards: F28035 based on the TMS320F28035 "Piccolo" microcontroller and F28335 based on the TMS320F28335 "Delfino" microcontroller (Fig. 4). TMS320F28035 and TMS320F28335 belong to the Texas Instruments C2000 family of 32-bit signal controllers. Microcontrollers of the C2000 family are specialized devices, the architecture and built-in peripherals of which are maximally adapted for efficiently solving the tasks of direct digital motor control and power energy converters control, as well as for performing complex industrial automation.

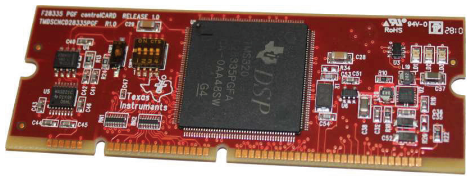


Fig. 4. F28335 control board

Based on the complexity of the algorithm of the DC motor speed control based on a discrete time equalizer, the F28335 board with the TMS320F28335 microcontroller is used in the laboratory research bench.

One of the pulse quadrature decoders (QEP) of the TMS320F28335 microcontroller is used in the laboratory research bench to connect a speed feedback sensor, which is the Hextron M40SA encoder mounted on the shaft of the KPA–561–U2 drive motor.

In order to read the information coming from the M40SA encoder during a certain time interval of the laboratory research bench functioning, an external L–Card ADC/DAC E–440 module is used, which is connected to the USB port of the laptop and has the necessary LGraph2 software.

In the laboratory research bench, the necessary information is outputted by DACs installed on the DMC board. In total, four DACs are installed on the power board, which operate on the basis of PWM DAC with filtering the resulting signal by RC filters. They allow real-time monitoring of any four user-defined variables of the control process in the program. Both the oscilloscope and the external ADC module, which is the E–440 module can be connected to the PWM DAC outputs.

The E–440 module is connected to the USB port of the laptop by USB–A – USB–B cable. Using the same cable, a digital motor control board is connected to another port of the laptop (Fig. 3). This connection allows creating the software code for the F28335 control board and loading it into the memory of the TMS320F28335 microcontroller, as well as visualizing the control process variables with the ability to write them as text files.

The components of the laboratory research bench are presented on a generalized connection scheme (Fig. 5).

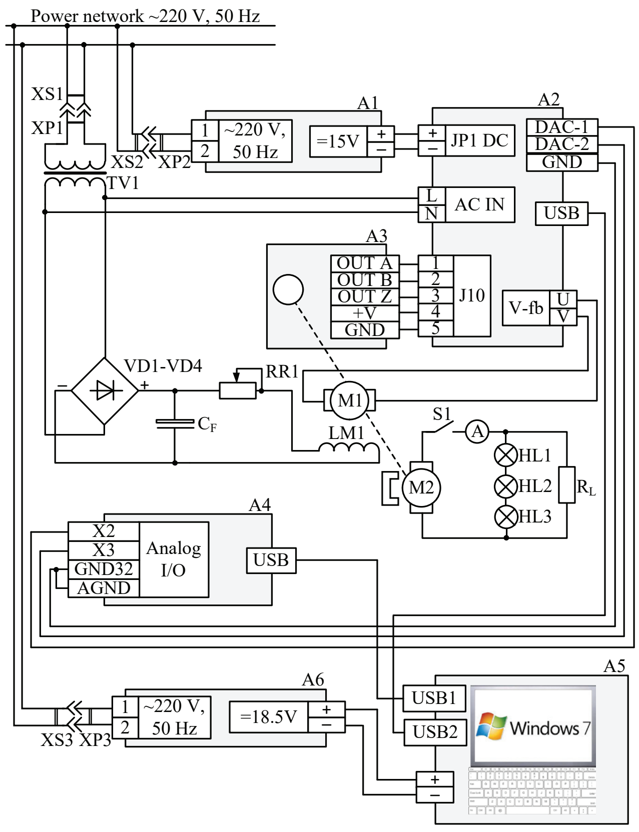


Fig. 5. Generalized connection scheme of the laboratory research bench

In Fig. 5, the following conventions are introduced:

– A1 – DMC board power supply;

– A2 – DMC board with F28335 control board installed;

– A3 –Hextron M40SA encoder;

– A4 –L–Card E–440 module;

– A5 – laptop with Windows 7 SP1 Ultimate 64–bit operating system;

– A6 – laptop power supply;

– VD1–VD4 – single-phase bridge full-wave rectifier on power diodes V10;

– TV1 – OSM1–0.4U3 transformer;

– RR1 – PEVR-25 rheostat, by moving the clamp in which an additional resistance is adjusted;

– C_F – smoothing filter;

– M1 – KPA–561–U2 drive motor with LM1 excitation winding;

– M2 –DPM 290/3.8/30 generator;

– HL1–HL3 – incandescent lamps MN 6.3–0.3;

– R_L – loading resistor PEV–25;

– S1 – toggle-switch;

– A – ammeter;

– XP1–XP3 – plugs;

– XS1–XS3 – plug sockets;

– J10 – DMC board connector for encoder connection;

– JP1 DC – DMC board connector for connecting an external power supply;

– AC IN – DMC board connector for alternating current (AC);

– DAC – DAC outputs of the DMC board, which operate on a PWM basis;

– V–fb – three-phase inverter outputs (motor anchor circuit is connected to clamps U and V).

The design environment for programming the TMS320F28335 microcontroller is Code Composer Studio (CCS) version 4.1.0.02005. CCS uses a modern concept for software development of microcontroller systems, which provides a modular approach. Each program module with this approach can be written independently of other parts of the project.

To fully expose the capabilities of the TMDSHVM-TRPFCKIT development kit in the CCS integrated design environment, Texas Instruments provides ControlSUITE software.

ControlSUITE for microcontrollers of the C2000 family is a comprehensive set of software infrastructure elements that greatly simplify software development. ControlSUITE includes all the necessary tools, from libraries of drivers and auxiliary software modules to complete typical examples of projects implemented using the TMDSHVMTRPFCKIT development kit or similar technical solutions from Texas Instruments.

5. General approaches to the synthesis of DC drive control systems based on a discrete time equalizer

Using the principle of block diagrams symmetry theoretically allows building an inertia-free closed electromechanical system and providing the transfer function of this system equal to one [13].

However, this approach does not make practical sense, since it requires energy sources of infinite power, the creation of which is impossible, based on the laws of physics. In practice, only partial compensation of the control object inertial properties makes sense. In this case, the modified principle of symmetry deserves special attention, which provides for incomplete compensation of the control object dynamic properties [14].

Then the control object will include a controllable converter with a DC motor connected to it.

The block diagram of such control object is shown in Fig. 6. The input voltage U_{inp} is supplied to the input of the control object, the speed ω is the output coordinate. Also in Fig. 6, the following designations are introduced: I_a – armature circuit current, I_s – static load current, E_{cc} – electromotive force (EMF) of the controlled converter, E_m – motor EMF.

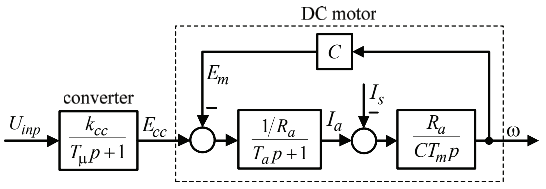


Fig. 6. Block diagram of the control object

The controlled converter in this figure is presented as an aperiodic unit with the transfer function

$$W_{cc}(p)=\frac{k_{cc}}{T_\mu p+1},$$

where k_{cc} – gain factor of the controlled converter; T_μ– time constant of the controlled converter (smallest uncompensated time constant).

The transfer function of the electromagnetic part of the motor

$$W_e(p)=\frac{1/R_a}{T_a p+1},$$

where R_a – armature circuit resistance; T_a – electromagnetic time constant of the motor.

The transfer function of the electromechanical part of the motor

$$W_m(p)=\frac{R_a}{CT_m p},$$

where C – constant coefficient of the electric motor, calculated at nominal magnetic flux; T_m – electromechanical time constant of the electric drive.

The block diagram of a direct current electric drive with control made on the basis of a discrete time equalizer with partial compensation of the control object dynamic properties is shown in Fig. 7.

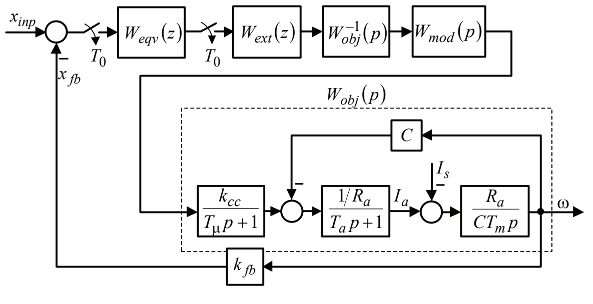


Fig. 7. Block diagram of a direct current electric drive with control made on the basis of a discrete time equalizer

In the proposed block diagram (Fig. 7), the controlled converter and the DC motor (taking into account the EMF feedback) are reduced to one transfer function $W_{obj}(p)$. The discrete time equalizer is represented by the transfer function W_{ekv}(z) and its operation with the analog part of the system is coordinated by zero-order extrapolator with the transfer function $W_{ext}(p)=(z-1)/zp$. Negative feedback is performed with the coefficient k_{fb}. The inverse model of the control object is shown as W_{obj}^{-1}. The disturbance is the static load current I_s. The transfer function of the object for the control action is as follows:

$$W_{obj}(p)=\frac{k_{cc}/\left(CT_aT_mT_\mu\right)}{p^3+\frac{T_aT_m+T_mT_\mu}{T_aT_mT_\mu}p^2+\frac{T_m+T_\mu}{T_aT_mT_\mu}p+\frac{1}{T_aT_mT_\mu}}=$$

$$=\frac{\beta_0}{p^3+\alpha_2p^2+\alpha_1p+\alpha_0}. \quad (1)$$

The obtained coefficients for the control object in accordance with the canonical form of Frobenius controllability [15]:

$$\beta_0=\frac{k_{cc}}{CT_aT_mT_\mu},\ \alpha_0=\frac{1}{T_aT_mT_\mu},$$

$$\alpha_1=\frac{T_m+T_\mu}{T_aT_mT_\mu},\ \alpha_2=\frac{\left(T_aT_m+T_mT_\mu\right)}{T_aT_mT_\mu}. \quad (2)$$

Then the block diagram of the control object using the first canonical form of controllability can be illustrated in Fig. 8.

The inverse model transfer function is

$$W_{obj}^{-1}(p)=\frac{1}{W_{obj}(p)}=\frac{1}{\beta_0}\left(p^3+\alpha_2p^2+\alpha_1p+\alpha_0\right). \quad (3)$$

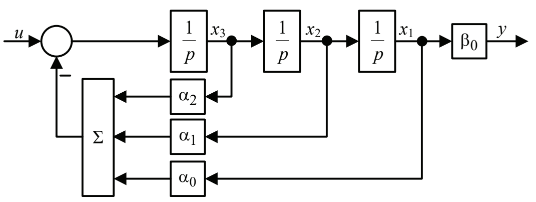


Fig. 8. Block diagram of the control object in the first canonical form of controllability

So, the synthesis method of automated electromechanical systems based on a discrete time equalizer can be used for direct current electric drives. The synthesis must be carried out taking into account the modified principle of symmetry, which consists in installing a block for inverse transformation modification (integrator) between the inverse model of the control object and the control object itself. With this approach, realistically achievable dynamic modes of operation are obtained and the first order of astatism is also ensured.

6. Definition of transfer functions and building of block diagrams of the automatic control system main elements

Let us synthesize an automatic speed control system of the laboratory research bench drive motor based on a discrete time equalizer, using technical specifications and taking into account the design features of the equipment included in its composition. In this case, the synthesis results should allow the possibility of their software implementation using the software discussed in the previous section.

Since the program implementation of the control system in the integrated CCS design environment, the overall program cycle is performed at a frequency of 10 kHz, all variables that are in the main body of the control program are updated in real time in 0.0001 s. Between the iterations (cycles) of the general program cycle, the variables hold their previous values. When it comes to executing some elements of the program code below 10 kHz, they are removed from the main body of the control program and formalized as special macro subroutines.

To interrogate the macro, a counter is set in the main body of the control program. When the counter reaches the value corresponding to the required number of iterations of the general program cycle, the macro is accessed and the variables associated with it in the main body of the control program are updated. After interrogating the macro, the counter is reset to the initial value and everything is repeated again. For example, if such a counter is set to 10, then the corresponding macro will be accessed at every tenth iteration of the general program cycle, therefore, the macro code will be executed with a frequency of 1 kHz.

Thanks to this approach, individual parts of the discrete control system, implemented on the basis of the TMS320F28335 microcontroller of the C2000 family, are able to operate at a frequency lower than the frequency of the overall program cycle. Typically, in examples of projects supplied by Texas Instruments as part of the ControlSUITE software, interrogating of some sensors and external peripherals is implemented at a frequency below the frequency of the overall program cycle.

Based on the block diagram shown in Fig. 7, the following provisions can be formulated regarding the software implementation of its individual elements:

1. The discrete time equalizer should be implemented as a separate subprogram – a macro, which is interrogated at a frequency below the frequency of the general program cycle.

2. It is advisable to process the signal coming from the M40SA encoder using a standard macro from the ControlSUITE software and interrogate it with the frequency of the general program cycle.

3. There is no need to use an extrapolator as a separate element of the control system. The function of data extrapolation is performed by the logic of the program code: all variables store their values in the microcontroller memory between iterations of the general program cycle.

4. The inverse model of the control object and the inverse transformation modification block should be represented in the form of separate macros, however, they should not be interrogated by the counter, but directly in the general program cycle. If the macro polling frequency of the discrete time equalizer is significantly lower than the macro polling frequency of the inverse model of the control object and the inverse transformation modification block, then the corresponding blocks of the discrete control system can be considered quasi-analog.

Thus, the elements of the block diagram shown in Fig. 7, when performing a software implementation, are the following subprograms:

– *EQUALIZER_MACRO* – a macro of the discrete time equalizer; experimental studies use several variants of this macro that have settings for various quantized desired transition functions;

– *MIRROR_MACRO* – a macro of the inverse control object model;

– *MODIFIER_MACRO* – a macro of the inverse transformation modification block;

– *QEP_MACRO*– a standard encoder macro that comes with ControlSUITE software.

In addition to *QEP_MACRO*, other standard macros are also used in the program code. For example, the *PWM_MACRO* inverter control macro or *PWMDAC_MACRO* macro for interacting with DAC of a DMC board. Standard macros are used in the program code either without changes or with minor modifications. The functional diagram of these macros interaction is presented in Fig. 9.

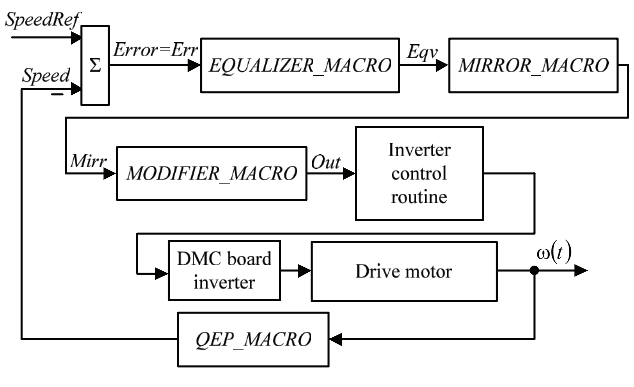


Fig. 9. Functional diagram of macros interaction

Based on Fig. 9, the reference signal is represented in relative units using the *SpeedRef* variable, which is set by the user in the main body of the control program. The *QEP_MACRO* macro in the main body of the control program is associated with the *Speed* variable that represents the drive motor speed in relative units. The difference between the *SpeedRef* and *Speed* variables is represented in the main body of the control program as the *Error* variable, which is then transferred to the macro of the discrete time equalizer as *Err*. The *EQUALIZER_MACRO* macro passes its original *Eqv* variable to the input of the *MIRROR_MACRO* macro. The *MIRROR_MACRO* macro has an output variable *Mirr*, which is transmitted to the input of the *MODIFIER_MACRO* macro, the output variable *Out* of which is supplied to the inverter control routine and is used to determine the duty cycle of the PWM signal at clamps U and V of the three-phase inverter, to which the armature circuit of the drive motor is connected.

The inverter control routine and the macro for working with the encoder are taken from the libraries supplied with ControlSUITE. *MIRROR_MACRO*, *MODIFIER_MACRO* and *EQUALIZER_MACRO* are implemented based on the results of the synthesis of the automatic speed control system of the laboratory research bench drive motor on the basis of discrete time equalizer.

Given the above, we perform the calculations necessary for the further software implementation of the *MIRROR_MACRO*, *MODIFIER_MACRO* and *EQUALIZER_MACRO* macros.

Based on the block diagram of the control object shown in Fig. 8, its inverse model is determined by the transfer function (3), where the coefficients β_0, α_2, α_1, α_0 are determined by the formulas (2).

To perform the software implementation of the inverse model, it is more convenient to present its transfer function (3) in the following form:

$$W_{obj}^{-1}(p)=\gamma_3 p^3+\gamma_2 p^2+\gamma_1 p+\gamma_0, \tag{4}$$

where

$$\gamma_3=\frac{1}{\beta_0},\ \gamma_2=\frac{\alpha_2}{\beta_0},\ \gamma_1=\frac{\alpha_1}{\beta_0},\ \gamma_0=\frac{\alpha_0}{\beta_0}.$$

When implementing the inverse model in the form of a separate macro (*MIRROR_MACRO*), the signal entering its input is denoted as *Eqv* and the output signal as *Mirr* (Fig. 9). Since the inverse model of the control object is implemented in a discrete form, the Laplace operators *p*, included in the formula (4) and symbolizing the differentiation of the signal, should be replaced by the difference of lattice functions. The inverse model of the control object can then be described as the following difference equation:

$$\begin{aligned} Mirr(n)&=\gamma_3\cdot\nabla^3 Eqv(n)+\\ &+\gamma_2\cdot\nabla^2 Eqv(n)+\gamma_1\cdot\nabla Eqv(n), \end{aligned} \tag{5}$$

where *Mirr*(*n*) – the value of the lattice function of the *MIRROR_MACRO* macro output signal at the current step *n*; $\nabla Eqv(n)$ – the first order inverse difference for the lattice function of the *MIRROR_MACRO* macro input signal; $\nabla^2 Eqv(n)$ – the second order inverse difference for the lattice function of the *MIRROR_MACRO* macro input signal; $\nabla^3 Eqv(n)$ – the third order inverse difference for the lattice function of the *MIRROR_MACRO* macro input signal.

It should be noted that the equation (5) uses not direct differences, but inverse differences, since the calculation of direct differences requires information about the change in the input signal in the upcoming step cycles relative to the current step cycle *n*. Of course, the control program does not have such information. To calculate the inverse differences programmatically, it is necessary to have the value of the input signal in the current step cycle *n* and several of its values in previous moments (iterations of the general program cycle).

Known formulas [16] are used to calculate the inverse differences that are part of the equation (5):

$$\nabla Eqv(n) = Eqv(n) - Eqv(n-1), \tag{6}$$

where *Eqv*(*n*) – the value of the lattice function of the *MIRROR_MACRO* macro input signal at the current step*n*; *Eqv*(*n*–1) – the value of the lattice function of the *MIRROR_MACRO* macro input at the step (*n*–1).

$$\nabla^2 Eqv(n) = Eqv(n) - \\ -2\cdot Eqv(n-1) + Eqv(n-2), \tag{7}$$

where *Eqv*(*n*–2) – the value of the lattice function of the *MIRROR_MACRO* macro input signal at the step (*n*–2).

$$\nabla^3 Eqv(n) = Eqv(n) - 3\cdot Eqv(n-1) + \\ +3\cdot Eqv(n-2) - Eqv(n-3), \tag{8}$$

where *Eqv*(*n*–3) – the value of the lattice function of the *MIRROR_MACRO* macro input signal at the step (*n*–3).

The block diagram of the inverse model, constructed by equations (5)–(8), is presented in Fig. 10. Blocks z^{-1} in Fig. 10 provide a shift of the value of the corresponding lattice function by one step backwards. Accordingly, passing *Eqv*(*n*) through two blocks z^{-1}gives the value of the lattice function *Eqv*(*n*–2) and after three blocks –*Eqv*(*n*–3).

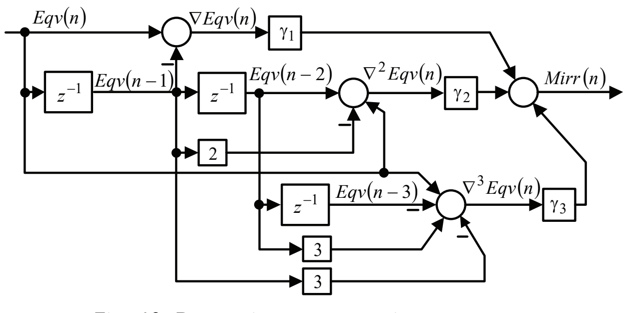


Fig. 10. Block diagram of the inverse model

The *Mirr* signal from the output of the *MIRROR_MACRO* macro acts as an input to the macro, which describes the operation of the inverse modification unit –*MODIFIER_MACRO*. The inverse transformation modifier is an integrator, so the following transformations can be performed:

$$\frac{Out(z)}{Mirr(z)} = \frac{T_0}{z-1};\ T_0\cdot Mirr(z) = z\cdot Out(z) - Out(z);$$

$$Out(z) = T_0\cdot z^{-1}\cdot Mirr(z) + z^{-1}\cdot Out(z);$$

$$Out(n) = Out(n-1) + T_0\cdot Mirr(n-1), \tag{9}$$

where *Out*(*n*) – the value of the lattice function of the output signal of the *MODIFIER_MACRO* macro at the current step*n*; *Out*(*n*–1) – the value of the lattice function of the *MODIFIER_MACRO* macro output signal at the step (*n*–1); *Mirr*(*n*–1) – the value of the lattice function of the *MODIFIER_MACRO* macro input (the *MIRROR_MACRO* macro output signal) at the step (*n*–1); T_0 – macro quantization (interrogation) period; since the interrogation of the *MIRROR_MACRO* macro is performed in the general program cycle, then T_0=0.0001 s.

From the difference equation (9) for the *MODIFIER_MACRO* macro, it follows that for each current step, the previous value of the output signal *Out*(*n*–1) will be added to the product of the previous value of the input signal *Mirr*(*n*–1) for the quantization period T_0. Due to this summation, the signal output of the *MODIFIER_MACRO* macro may be greater than one. Since all calculations in the control program are performed in relative units, and the value *Out*(*n*) is used by the inverter control subroutine to determine the PWM fill factor, the possible range of signal change at the output of the *MODIFIER_MACRO* macro is [0; 1]. Therefore, for practical use of the difference equation (9), the output signal of the *MODIFIER_MACRO* macro should be limited in the range [0; 1].

Based on the recommendations of Texas Instruments [17], in order to ensure that the output of the integrator does not deviate significantly from the set program limit in the presence of a macro output limit, the integrator should be reset according to the tracking anti-windup strategy [18]. When using this strategy, the integrator is covered by feedback on the deviation signal, which is formed as the difference between the output signal of the bounding block and its input signal. The use of the tracking anti-windup strategy for the inverse modification unit described by the difference equation (9) is shown in Fig. 11.

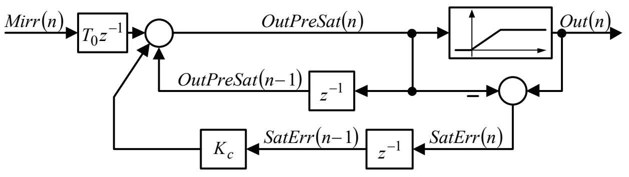


Fig. 11. Block diagram of the inverse modification unit

The nonlinear limitation unit shown in Fig. 11 is described by the following equations:

$$\left.\begin{array}{l} Out(n) = 0 \text{ when } OutPreSat(n) < 0, \\ Out(n) = OutPreSat(n) \text{ when } 0 \le OutPreSat(n) \le 1, \\ Out(n) = 1 \text{ when } OutPreSat(n) > 1, \end{array}\right\} \tag{10}$$

where *OutPreSat*(*n*) – the lattice function value of the nonlinear input signal at the current step *n*.

The *OutPreSat*(*n*) signal is determined from the following difference equation:

$$OutPreSat(n) = OutPreSat(n-1) + \\ +T_0\cdot Mirr(n-1) + K_c\cdot SatErr(n-1), \tag{11}$$

where *OutPreSat*(*n*–1) – the lattice function value of the nonlinear unit input at the step (*n*–1); *SatErr*(*n*–1) – the lattice function value of the deviation signal at the step (*n*–1); K_c – the correction factor by which the speed of the integrator reset can be adjusted (in the program implementation of the macro, based on the recommendations of Texas Instruments, K_c=0.02 was set). The deviation signal at the current step *n* is defined as the difference between *Out*(*n*) and *OutPreSat*(*n*):

$$SatErr(n)=Out(n)-OutPreSat(n). \tag{12}$$

For example, if at the previous step (n–1) the signal at the input of the nonlinear limiting link was *OutPreSat*(n–1)=1.2, then the output of this link had the signal *Out*(n–1)=1. In this case, microcontroller memory on the step (n–1) had a value of *SatErr*(n–1)=–0.2 that is used in the current step n to reset the integrator.

Thus, at the output of the *MODIFIER_MACRO* macro, a signal is generated for the power switch subroutine of the inverter of the DMC board, with which the armature circuit of the drive motor is supplied with pulse voltage. The pulse frequency remains constant at 10 kHz, and the duration varies and is determined by the duty cycle. Based on the technical features of the laboratory research bench, the rectification and filtering of the voltage supplied to the power switches of the inverter is carried out by the DMC board.

The steady-state value of the rotational speed of the drive motor shaft is determined by the duty cycle of the impulse supply voltage of the armature circuit. The current speed value in the control system is obtained as the output signal Speed of the encoder *QEP_MACRO* macro. The difference between the *SpeedRef* value and the *Speed* signal forms the discrete time equalizer macro input signal *Err*:

$$Err=SpeedRef-Speed.$$

Since the control object is static, the transfer function of the discrete time equalizer in general is determined by the following formula [4]:

$$W_{eqv}(z)=\frac{Eqv(z)}{Err(z)}=\\ =\frac{a_{k-1}z^k+\sum_{i=1}^{k-1}(a_{i-1}-a_i)z^i-a_0}{T_{eq}z^k-T_{eq}k_{fb}\sum_{i=0}^{k-1}a_iz^i}, \tag{13}$$

where k – the order of the characteristic equation; a_{k-1}, a_{k-2}, ..., a_1, a_0 – the coefficients that characterize the increase in the levels of the quantized desired transition function at each quantization cycle; T_{eq} – the quantization (interrogation) period of the *EQUALIZER_MACRO* macro, which is determined based on the quantized desired transition function.

To perform a software implementation, this transfer function must be represented as a difference equation by performing the following transformations:

$$W_{eqv}(z)=\\ =\frac{1}{T_{eq}}\cdot\frac{A_kz^k+A_{k-1}z^{k-1}+A_{k-2}z^{k-2}+...+A_2z^2+A_1z+A_0}{z^k+B_{k-1}z^{k-1}+B_{k-2}z^{k-2}+...+B_2z^2+B_1z+B_0}, \tag{14}$$

The coefficients of the numerator and denominator of the transfer function (14) are determined by the following formulas:

$$\left.\begin{array}{l} A_k=a_{k-1},\\ A_i=a_{i-1}-a_i \text{ when } i=1,2,3,...,(k-1),\\ A_0=-a_0,\\ B_i=-a_i\cdot k_{fb} \text{ when } i=0,1,2,...,(k-1). \end{array}\right\} \tag{15}$$

For a software implementation of the control system in relative units, the feedback coefficient k_{fb}=1 in equations (15).

Based on the transfer function (14), the following equation can be obtained:

$$Eqv(z)=\frac{1}{T_{eq}}\cdot\begin{pmatrix} A_kErr(z)+A_{k-1}z^{-1}Err(z)+\\ +A_{k-2}z^{-2}Err(z)+...+A_2z^{-k+2}Err(z)+\\ +A_1z^{-k+1}Err(z)+A_0z^{-k}Err(z)\end{pmatrix}-\\ -B_{k-1}z^{-1}Eqv(z)-B_{k-2}z^{-2}Eqv(z)-...-B_2z^{-k+2}Eqv(z)-\\ -B_1z^{-k+1}Eqv(z)-B_0z^{-k}Eqv(z).$$

If the current step is n, then the difference equation of the discrete time equalizer in general is as follows:

$$Eqv(n)=\\ =\frac{1}{T_{eq}}\cdot\begin{pmatrix} A_kErr(n)+A_{k-1}Err(n-1)+\\ +A_{k-2}Err(n-2)+...+A_2Err(n-k+2)+\\ +A_1Err(n-k+1)+A_0Err(n-k)\end{pmatrix}-\\ -B_{k-1}Eqv(n-1)-\\ -B_{k-2}Eqv(n-2)-...-B_2Eqv(n-k+2)-\\ -B_1Eqv(n-k+1)-B_0Eqv(n-k)=\\ =\frac{1}{T_{eq}}\cdot\left(\sum_{i=0}^{k}A_{k-i}Err(n-i)\right)-\sum_{i=1}^{k}B_{k-i}Eqv(n-i). \tag{16}$$

The block diagram of the discrete time equalizer constructed by the difference equation (16) is shown in Fig. 12.

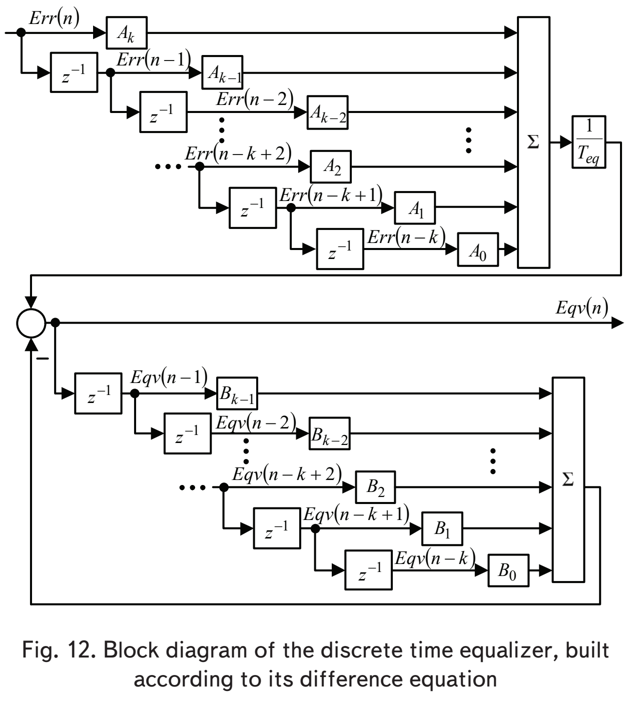


Fig. 12. Block diagram of the discrete time equalizer, built according to its difference equation

The block diagram of the discrete time equalizer shown in Fig. 12 has become the basis for its software implementation. It clearly demonstrates all the transformations that occur with the error signal *Err*(n) at any point of time. The coefficients necessary for determining the control action *Eqv*(n) are calculated according to formulas (15).

7. Experimental studies of the drive speed automatic control system

A discrete time equalizer is used in experimental studies. It has settings for the quantized desired transition function with the number of levels k=16 (Fig. 13). The quantized desired transition function in Fig. 13 is shown in relative units (r.u.); the value of n along the abscissa corresponds to the step number. The quantization period of T_{eq} during experimental studies is set in the main body of the control program, regardless of the settings in the *EQUALIZER_MACRO* macro.

The automatic speed control system of the drive motor of the laboratory test bench allows both control using one discrete time equalizer and combined control based on two discrete time equalizers. The T_{eq} quantization period used in the formula for the compensating discrete time equalizer will be several times (2–5) less than the quantization period of the corresponding basic discrete time equalizer. Compensating discrete time equalizers, as well as the main equalizers, are represented as the program code by the difference equation (16).

Each of the macros used in the main program code is a subroutine that is initialized in a special header file. Thanks to the application of the object-oriented approach, each of the macros is the so-called structure of the C++ language (a set of variables united by one name, which provides a generally accepted way of information storage together). Structures allow creating templates for named instances. Each of the macros used in the control program is represented by an instance of the corresponding structure.

Experimental studies are carried out according to the methodology illustrated by the diagram shown in Fig. 14 (Code Composer Studio or LGraph2 software is used at each step).

After loading the program code, which by default is implemented in the microcontroller's RAM, buttons appear on the panel of the CCS main menu for controlling the running of the program loaded into the microcontroller. Using these control buttons, the program code can be run both in real time and step by step. Also it can be temporarily paused or completely stopped.

The diagram in Fig. 14 illustrates the sequence of obtaining the results of one experimental study for a certain set of initial conditions specified in steps 1 and 2. Changing the set of initial conditions during each subsequent experiment does not affect the general scheme of experimental studies.

According to the results of each single experiment, a corresponding text file was recorded, which, in addition to the experimental data themselves, also contained additional auxiliary information, such as the date and time of the experiment, the quantity and numbers of channels, the frequency value per channel, and so on.

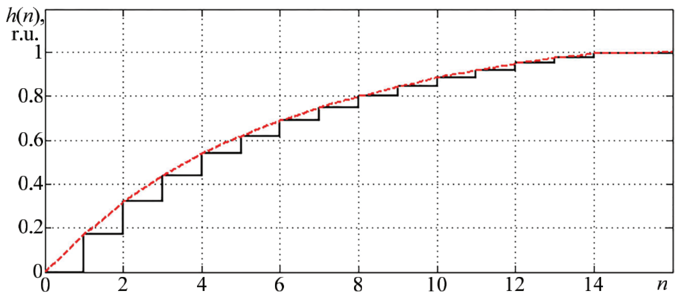
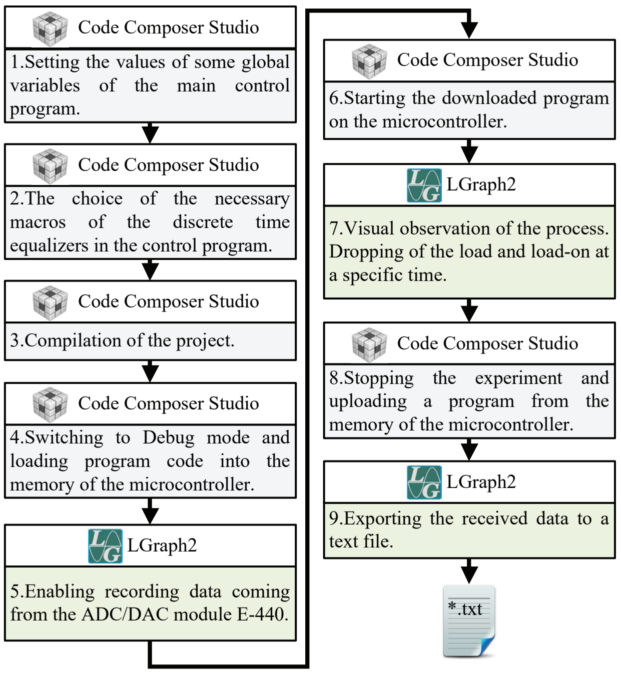


Fig. 13. Quantized desired transition function at k=16



1.Setting the values of some global variables of the main control program.

2.The choice of the necessary macros of the discrete time equalizers in the control program.

3.Compilation of the project.

4.Switching to Debug mode and loading program code into the memory of the microcontroller.

5.Enabling recording data coming from the ADC/DAC module E-440.

6.Starting the downloaded program on the microcontroller.

7.Visual observation of the process. Dropping of the load and load-on at a specific time.

8.Stopping the experiment and uploading a program from the memory of the microcontroller.

9.Exporting the received data to a text file.

Fig. 14. General scheme of experimental studies

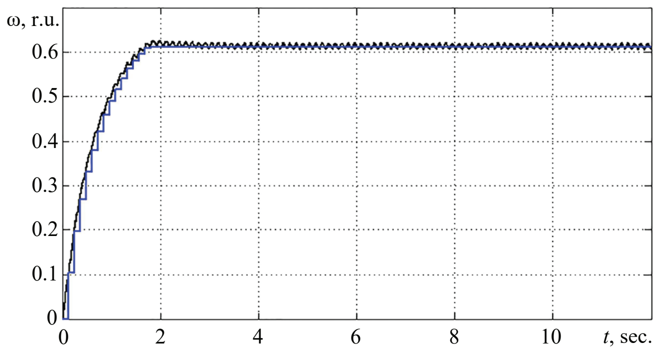


Fig. 15. Results of the experiment

The results of one of the experiments are presented in Fig. 15 (motor started at rated load up to a speed corresponding to 0.625 of the rated value). The blue color in this figure shows the quantized desired speed transition function (Fig. 13) and the black shows the obtained value of the DC motor speed.

The proposed experimental research methodology made it possible to carry out three sets of experiments differing in macros of the main and compensating discrete time equalizers. Each set consists of nine experiments that differ from each other in the values of global variables, which are responsible for the quantization period of the main and compensating discrete time equalizers, in the values of the reference speed signal, in the resampling coefficients and in utilization or not of the combined control.

In all experiments, the quantized desired transient functions were fulfilled with a slight dynamic error that occurred during the starting under load, dropping of the load and after repeated load-on.

Due to the insignificant transmission coefficient and soft static characteristics of the control object, the system required influencing the proportional component in order to increase the dynamic accuracy and reduce the sensitivity to parametric and coordinate disturbances.

The use of combined control with two discrete time equalizers made it possible to reduce the dynamic errors both during dropping of the load and load-on from 9–20 % to 1.5–3.5 %.

8. Discussion of the research results of automated electromechanical systems synthesized on the basis of a discrete time equalizer

As a result of the experimental researches, it is found that the method of synthesis of automated electromechanical systems based on a discrete time equalizer can be successfully applied to electric drives with DC motors. The practical value is the use of the modified principle of symmetry with the integrator (modification unit) reset in accordance with the tracking anti-windup strategy (Fig. 11). This fact is explained by the inability to fully compensate for the dynamic properties of the control object.

The synthesis of regulators by the discrete time equalizer method is the development of the authors of the paper; therefore, there are no known software implementations by modern microcontrollers. The Code Composer Studio integrated design environment made it possible to practically implement the proposed discrete time equalizers (16), the inverse model of the control object (5) and the inverse modification unit (10), (11) in the form of special macro routines for the TMS320F28335 microcontroller (Fig. 9).

A feature of the obtained results is the use of a high-level programming language C/C++ with object-oriented approaches, so that the code is hardware independent of the microprocessor type. Therefore, such software solutions have significant practical value and can find application in industry.

A limitation characteristic of the studies is the focus on direct current electric drives. The power part of the laboratory research stand (Fig. 5) is not fully adapted for working with AC motors. In addition, to work with an AC motor, the macro of its inverse mathematical model must be additionally created.

The macros of discrete time equalizers (the block diagram in Fig. 12) and the macro of the inverse modification unit, implemented in accordance with the tracking anti-windup reset strategy (the block diagram in Fig. 11) are universal and can be used to create control systems for a variety of technical objects based on a discrete time equalizer.

The disadvantage of the proposed technical solution is the need for microcontrollers with significant processing power, for example, such as Texas Instruments C2000 (Fig. 4). The complication of the control system, the increase in the number of loops or control channels in the future will only lead to an increase in the requirements for the microcontroller, on the basis of which discrete time equalizers, inverted models, and inverse modification units are implemented. This problem can be partially solved by distributing the computing load between the individual microprocessor devices.

The synthesis of automatic control systems for electromechanical objects based on a discrete time equalizer differs from modal control, traditional subordinate coordinate control, or the method of generalized characteristic polynomial by completely rejecting the use of the desired characteristic polynomials. This approach allows obtaining the desired dynamic and static properties of the system solely on the basis of the desired transition function, which should be close to the real nature of the transition processes (monotonic, aperiodic or oscillatory). If the transition function deviates somewhat from its natural character, then this may cause overshoot when reaching a steady-state value.

Further research may be associated with the development of a subsystem for identifying the admissibility of quantized desired transition functions (Fig. 1) and the introduction of control systems with discrete time equalizers in the industry. One of the difficulties that will arise during the implementation of such systems will be the necessity of creation of sufficiently accurate control objects mathematical models.

In general, a discrete time equalizer has the prospect of hardware implementation as a standalone device that provides the ability to perform control of a technical object and comfortable dialogue with the user and implements the full range of features relevant to the concept of "Internet of Things".

9. Conclusions

1. A laboratory research bench created to carry out the experimental research made it technically possible to implement an automatic speed control system of the DC motor, synthesized on the basis of a discrete time equalizer.

2. The general approaches to the synthesis of control systems for DC electric drives based on a discrete time equalizer with partial compensation of the dynamic properties of the control object was analyzed. It was found that the synthesis must be performed taking into account the modified principle of symmetry, which consists in using an integrator for the inverse transformation modification. With this approach, realistically achievable dynamic operation modes were obtained, and the first order of astatism was also ensured.

3. The transfer functions were determined and the block diagrams of the main elements of the automatic speed control system of the laboratory research stand were built. For the technical implementation of the proposed solutions, calculation formulas were obtained that simplify the software implementation of the discrete time equalizer, the control object inverse model and the inverse transformation modification unit. The use of these formulas made it possible to obtain the coefficients of difference equations for high-order discrete time equalizers.

4. The main body of the control program, constructed in accordance with the developed functional scheme of the macros interaction, made it possible to perform the experimental studies using both the main control channel with one discrete time equalizer and the combined control with two discrete time equalizers (main and compensating). In all experiments, the quantized desired transient functions were fulfilled with a slight dynamic error. The use of combined control with two discrete time equalizers made it possible to reduce the dynamic errors both during dropping of the load and load-on from 9–20 % to 1.5–3.5 %.

References

1. Marushchak, Ya. Yu. (2005). Syntez elektromekhanichnykh system z poslidovnym ta paralelnym koryhuvanniam. Lviv: Lvivska politekhnika, 208.

2. Cerone, V., Piga, D., Regruto, D. (2014). Characteristic polynomial assignment for plants with semialgebraic uncertainty: A robust diophantine equation approach. International Journal of Robust and Nonlinear Control, 25 (16), 2911–2921. doi: https://doi.org/10.1002/rnc.3238

3. Marushchak, Y., Kopchak, B. (2015). Synthesis of Automatic Control Systems by Using Binomial and Butterworth Standard Fractional Order Forms. Computational Problems of Electrical Engineering, 5 (2), 89–94.
4. Sheremet, O., Sadovoy, O. (2016). Development of a mathematical apparatus for determining operator images of the desired quantized transition functions of finite duration. Eastern-European Journal of Enterprise Technologies, 2 (2 (80)), 51–58. doi: https://doi.org/10.15587/1729-4061.2016.65477
5. Liberzon, D., Trenn, S. (2013). The Bang-Bang Funnel Controller for Uncertain Nonlinear Systems With Arbitrary Relative Degree. IEEE Transactions on Automatic Control, 58 (12), 3126–3141. doi: https://doi.org/10.1109/tac.2013.2277631
6. Zhang, G., He, P., Li, H., Tang, Y., Li, Z., Xiong, X.-Z. et. al. (2020). Sliding Mode Control: An Incremental Perspective. IEEE Access, 8, 20108–20117. doi: https://doi.org/10.1109/access.2020.2966772
7. Zabala, P. (2017). Development of Programmable Relay Switch Using Microcontroller. American Journal of Remote Sensing, 5 (5), 43. doi: https://doi.org/10.11648/j.ajrs.20170505.11
8. Rao, K. D., Swamy, M. N. S. (2018). Digital Signal Processing. Theory and Practice. Springer. doi: https://doi.org/10.1007/978-981-10-8081-4
9. Zahradnik, P., Simak, B. (2012). Education in real-time digital signal processing using digital signal processors. 2012 35th International Conference on Telecommunications and Signal Processing (TSP). doi: https://doi.org/10.1109/tsp.2012.6256372
10. Kong, J. H., Ang, L.-M., Seng, K. P. (2010). Minimal Instruction Set AES Processor using Harvard Architecture. 2010 3rd International Conference on Computer Science and Information Technology. doi: https://doi.org/10.1109/iccsit.2010.5564522
11. Isermann, R. (2012). Digital Control Systems: Volume 1: Fundamentals, Deterministic Control (Revised Edition). Berlin: Springer-Verlag Berlin and Heidelberg GmbH & Co. KG.
12. Fukui, K., Kubo, T., Oya, H. (2013). Inverse linear quadratic regulator of neutral systems with time-varying delay. 2013 IEEE International Conference on Mechatronics and Automation. doi: https://doi.org/10.1109/icma.2013.6618131
13. Krut'ko, P. D. (1988). Obratnye zadachi dinamiki upravlyaemyh sistem. Nelineynye modeli. Moscow: Nauka, 326.
14. Sadovoy, A. V., Suhinin, B. V., Sohina, Yu. V.; Sadovoy, A. V. (Ed.) (1996). Sistemy optimal'nogo upravleniya pretsizionnymi elektroprivodami. Kyiv: ISIMO, 298.
15. Storjohann, A. (2001). Deterministic computation of the Frobenius form. Proceedings 42nd IEEE Symposium on Foundations of Computer Science. doi: https://doi.org/10.1109/sfcs.2001.959911
16. Besekerskiy, V. A., Popov, E. P. (2003). Teoriya sistem avtomaticheskogo upravleniya. Sankt-Peterburg: Professiya, 752.
17. C2000™ Digital Controller Library User's Guide. Texas Instruments, 2015. Available at: https://e2echina.ti.com/cfs-file/__key/telligent-evolution-components-attachments/00-56-00-00-00-14-79-60/PID_5F00_C2000_5F00_-Digital-Controller-Library-Users-Guide_2800_sprui31_2900_.pdf
18. Tharayil, M., Alleyne, A. (2002). A generalized PID error governing scheme for SMART/SBLI control. Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301). doi: https://doi.org/10.1109/acc.2002.1024828