

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ АЕРОНАВІГАЦІЇ, ЕЛЕКТРОНІКИ
ТА ТЕЛЕКОМУНІКАЦІЙ**

Кафедра авіаційних комп'ютерно-інтегрованих комплексів

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри

Синеглазов Віктор Михайлович

“ _____ ” _____ 2021 р.

**ДИПЛОМНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)**

**ВИПУСКНИКА ОСВІТНЬО-КВАЛІФІКАЦІЙНОГО РІВНЯ
“БАКАЛАВР”**

**ЗА СПЕЦІАЛЬНОСТЮ 151 «АВТОМАТИЗАЦІЯ ТА КОМП'ЮТЕРНО-ІНТЕГРОВАНІ
ТЕХНОЛОГІЇ»**

**ОСВІТНЬО-ПРОФЕСІЙНОЇ ПРОГРАМИ "КОМП'ЮТЕРНО-ІНТЕГРОВАНІ
ТЕХНОЛОГІЧНІ ПРОЦЕСИ І ВИРОБНИЦТВА"**

ТЕМА: Система автоматичного керування двигуном постійного струму з
ПІ-нейрорегулятором швидкості

ВИКОНАВЕЦЬ:

Білай Є.С.

КЕРІВНИК: к.т.н., ст. викл

Пантєєв Р. Л.

НОРМКОНТРОЛЕР: к.т.н., доцент

Тупіцин М.Ф.

Київ 2021

EDUCATION AND SCIENCE MINISTRY OF UKRAINE

NATIONAL AVIATION UNIVERSITY

Department of Aviation Computer-Integrated Complexes

ADMITT TO DEFENCE

Head of the department

Syneglazov V.M.

“ _____ ” _____ 2021 y.

BACHELOR WORK

(EXPLANATORY NOTE)

Specialty: 151 Automation and computer-integrated technologies

Educational professional program "Computer-integrated technological processes and production"

THEME: «Automatic DC motor control system with PI speed neuroregulator»

DONE by:

Bilai Y.S.

SUPERVISED by:

Pantyeyev R. L.

STANDARDS CONTROLLER:

Tupitsyn M.F.

Kyiv 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет аеронавігації, електроніки та телекомунікацій
Кафедра авіаційних комп'ютерно-інтегрованих комплексів

Освітній ступінь бакалавр

Спеціальність: 151 " Автоматизація та комп'ютерно-інтегровані технології"

ЗАТВЕРДЖУЮ

Завідувач кафедри

Синєглазов В.М.

“ ____ ” _____ 2021 р.

ЗАВДАННЯ

на виконання дипломної роботи студентки

Білай Єлизавети Сергіївни

- 1. Тема проекту (роботи):**“ Система автоматичного керування двигуном постійного струму з ПІ-нейрорегулятором швидкості ”
- 2. Термін виконання проекту (роботи):** з 20.01.2021 р. до 11.06.2021 р.
- 3. Вихідні данні до проекту (роботи):** Орієнтуватися на сучасні технології розробки систем керування із застосуванням нейроконтролерів.
- 4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):** 1. Аналіз та обґрунтування вибору методів керування швидкості в електромеханічній системі з двигуном постійного струму. 2. Огляд і аналіз існуючих розробок у галузі нейрокерування. 3. Дослідження проблеми нейрокерування швидкістю двигуна у двоконтурній системі підпорядкованого керування. 4. Розробка системи керування швидкістю електродвигуна із застосуванням нейроконтролера. 5. Дослідження режимів керування розробленої електромеханічної системи.

5. Перелік обов'язкового графічного матеріалу: 1. Лінійна система каскадного управління швидкістю; 2. Система регулювання швидкості приводу постійного струму із сухим тертям; 3. Схема прямого зворотного регулювання; 4. Система із сухим тертям та нейрорегулятором; 5. Система регулювання швидкості приводу постійного струму із сухим та в'язким тертям; 6. Результати роботи нейронної мережі.

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Підбір літератури	23.01 – 26.01	виконано
2	Аналіз існуючих методів керування швидкістю в електромеханічних системах з нелінійностями на прикладі сухого та в'язкого тертя	27.01 – 19.03	виконано
3	Розробка системи керування швидкістю електродвигуна на базі нейроконтролера	20.03 – 08.04	виконано
4	Створення та тренування нейромережі для роботи в електромеханічній системі з нелінійностями в якості регулятора	09.04 – 18.04	виконано
5	Дослідженні режимів роботи нелінійної електромеханічної системи з нейроконтролером	19.04 – 11.05	виконано
6	Аналіз результатів роботи моделі системи в середовищі MatLab/Simulink	12.05 – 21.05	виконано
7	Формування висновків щодо виконаної роботи	22.05 – 24.05	виконано
8	Оформлення пояснювальної записки	25.05 – 30.05	виконано
9	Створення презентації	31.05 – 11.06	виконано

7. Дата видачі завдання: “21” грудня 2021 р.

Керівник дипломної роботи

(підпис керівника)

Пантєєв Р. Л.

(П.І.Б.)

Завдання прийняв до виконання

(підпис випускника)

Білай Є.С.

(П.І.Б.)

NATIONAL AVIATION UNIVERSITY

Faculty of aeronavigation, electronics and telecommunications

Department of Aviation Computer Integrated Complexes

Educational level bachelor

Specialty: 151 "Automation and computer-integrated technologies"

APPROVED

Head of Department

Sineglazov V. M.

" ____ " _____ 2021

TASK

For the student's thesis

Bilai Y.S.

- 1. Theme of the work:** “Automatic DC motor control system with PI speed neuroregulator ”
- 2. Term of execution of the work:** from 20.01.2021 till 11.06.2021.
- 3. Initial data of the work:** To focus on modern technologies for the development of control systems using a neurocontroller.
- 4. The contents of the explanatory note (list of issues to be developed):**
 1. Analysis and discussion of the choice of speed control methods in an electromechanical system with a DC motor.
 2. Review and analysis of existing developments in the field of neurosurgery.
 3. Investigation of the problems of neuro-control of movement speed in a two-circuit system of subordinate control.
 4. Development of a control system for the speed of an electric motor using a neurocontroller.
 5. Research of control modes of the developed electromechanical system.
- 5. List of compulsory graphical materials:** 1. Linear system of speed cascade control; 2. System of speed control of a direct current drive with dry friction; 3. Scheme of direct inverse regulation; 4. System with dry friction and the neuroregulator; 5 System of speed control of a direct current drive with dry and viscous friction; 6. Results of the neural network.

6. Calendar Schedule

№	Task	Period of execution	Performance note
1	Selection of literature	23.01. – 26.01.	done
2	Analysis of existing speed control methods in electromechanical systems with nonlinearities on the example of dry and viscous friction	27.01. – 19.03.	done
3	Development of a motor speed control system based on a neurocontroller	20.03. – 08.04.	done
4	Creation and training of a neural network for work in electromechanical system with nonlinearities as a regulator	09.04. – 18.04.	done
5	Investigation of modes of operation of a nonlinear electromechanical system with a neurocontroller	19.04. – 11.05.	done
6	Analysis of the results of the system model in the MatLab / Simulink environment	12.05. – 21.05.	done
7	Forming conclusions about the work done	22.05. – 24.05.	done
8	Making an explanatory note	25.05. – 30.05.	done
9	Presentation creating	31.05. – 11.06.	done

7. Date of issuance of task: “21” december 2021.

Supervisor: _____ sr. lecturer, Ph.D Pantyeyev R. L.
(supervisors sign) (И.И.Б.)

Task is accepted for execution _____ Bilai Y.S.
(graduates sign) (И.И.Б.)

Реферат

Пояснювальна записка до бакалаврської кваліфікаційної роботи:

Система автоматичного керування двигуном постійного струму з ПП-нейрорегулятором швидкості

74 с., 31 рис., 2 табл., 2 додатка, 18 джерел

Об'єктом розробок і досліджень даної роботи є нелінійна система підпорядкованого регулювання швидкості тиристорного електроприводу постійного струму з нейрорегулятором.

Ціль роботи – синтез нелінійної системи підпорядкованого регулювання швидкості з урахуванням сухого тертя, а також системи з урахуванням сухого та в'язкого тертя; розробка для кожної з цих систем інверсної нейромоделі і використання її як нейрорегулятор швидкості, компенсуючий вищеназвані нелінійності.

Розробки проводились на основі теорії використання інверсних нейромоделей систем як нейрорегулятори в цих системах. Для дослідження розробленої системи був використаний метод математичного моделювання на персональному комп'ютері з використанням програмного пакету MatLab 5.2/Simulink, орієнтованого на моделювання систем автоматичного регулювання електроприводів.

В результаті роботи були синтезовані нелінійні системи підпорядкованого регулювання швидкості приводу, побудовані нейромоделі синтезованих систем. Для регулювання нелінійних об'єктів за допомогою отриманих нейромоделей був використаний метод “Direct inverse control”.

Проведене цифрове моделювання розроблених систем виявило ефективність використання метода “Direct inverse control” в нелінійних системах електроприводу. Отримані нелінійні системи з нейрорегулятором, компенсуючим існуючі нелінійності, відслідковують завдання з статичною помилкою 0.02% і з відставанням при цьому на один крок дискретності.

ТИРИСТОРНИЙ ЕЛЕКТРОПРИВОД ПОСТІЙНОГО СТРУМУ, СИСТЕМА ПІДПОРЯДКОВАНОГО РЕГУЛЮВАННЯ ШВИДКОСТІ, ІНВЕРСНА НЕЙРОМОДЕЛЬ, НЕЙРОРЕГУЛЯТОР, НЕЛІНІЙНІСТЬ, СУХЕ ТЕРТЯ, В'ЯЗКЕ ТЕРТЯ, ДИСКРЕТНІСТЬ, ЦИФРОВЕ МОДЕЛЮВАННЯ

Abstract

Explanatory note to the bachelor's qualification work:

Automatic DC motor control system with PI speed neuroregulator

74 pp., 31 figs., 2 tables, 2 appendices, 18 sources

The object of development and research of this work is a nonlinear system of subordinate speed control of a thyristor DC electric drive with a neuroregulator.

The purpose of the work is the synthesis of a nonlinear system of subordinate speed control taking into account dry friction, as well as a system taking into account dry and viscous friction; development for each of these systems of an inverse neuromodel and its use as a neuroregulator of speed, compensating for the above nonlinearities.

Developments were made on the basis of the theory of using inverse neuromodels of systems as neuroregulators in these systems. To study the developed system, we used the method of mathematical modeling on a personal computer using the software package MatLab 5.2 / Simulink, focused on modeling systems for automatic control of electric drives.

As a result, nonlinear systems of subordinate drive speed control were synthesized, neuromodels of synthesized systems were built. The "Direct inverse control" method was used to control nonlinear objects using the obtained neuromodels.

The conducted digital modeling of the developed systems revealed the efficiency of using the "Direct inverse control" method in nonlinear electric drive systems. The obtained nonlinear systems with a neuroregulator compensating for the existing nonlinearities track the task with a static error of 0.02% and with a lag of one step of discreteness.

THYRISTOR DC ELECTRIC DRIVE, SLAVE SPEED CONTROL SYSTEMS, INVERSE NEUROMODEL, NONLINEARITY, DRY FRICTION, VISCOUS FRICTION, DISCRETE, DIGITAL SIMULATION

CONTENT

List of abbreviations	
Introduction	
1. NEURAL NETWORKS IN AUTOMATION TECHNOLOGY	
1.1. Neural systems - branch of artificial intelligence	
1.2. General considerations of neural control for industrial processes	
2. GENERAL DESCRIPTION AND FUNCTIONING OF NEURAL NETWORKS	
3. SPECIAL PROPERTIES OF NEURAL NETWORKS	
3.1. Ability to learn	
3.2. Ability to process incorrect and incomplete information	
3.3. Adaptive behavior	
3.4. Massive parallelism.....	
3.5. Fault tolerance	
3.6. Hardware implementability.....	
4. FEEDFORWARD NETWORKS.....	
4.1. Model of a processing element	
4.2. Network layer model.....	
4.3. Model of the entire network.....	
4.4. Error back propagation.....	
5. OVERVIEW OF NEURAL REGULATIONS AND CONTROLS FOR COMPENSATING NON-LINEARITIES	



6. CLASSIFICATION OF THE TYPICAL NON-LINEARITIES IN PRACTICAL ELECTRIC DRIVE SYSTEMS

7. MODELING AND SIMULATION OF DRY FRICTION.....

8. MODELING AND SIMULATION OF DRY AND VISCOSE FRICTION

9. USE OF NEURAL NETWORKS TO COMPENSATE THE DRY FRICTION...

10. USE OF NEURAL NETWORKS TO COMPENSATE DRY AND VISCOSE FRICTION.....

Conclusions

Literature



Glossary

b_c — Coefficient of sliding friction

The moment of sliding friction: $M_c = b_c \cdot \text{sign}(\omega)$

b_N — Rolling friction coefficient

Moment of rolling friction: $M_N = b_N \cdot \omega$

I_N — Rated current of the motor

k — Torque constant

K_ω — Gain factor in the speed feedback

R_A — Connection resistance

$T_{r\omega}$ — Time constant in the counter of the TF of the PI controller

$T_{0\omega}$ — Time constant in the denominator of the TF of the PI controller

$T_{\Sigma i}$ — Equivalent time constant of the route

T_M — Mechanical starting time constant

u — Entrance to the system

y — Exit of the system

w — Weighting of the neuron

\mathbf{w} — Vector of weights

\mathbf{W} — Matrix of weights

ω_n — Rated speed

ω_{soll} — Speed setpoint

ω_{ist} — Actual speed value

Abbreviations

ANN — Artificial neural networks

N — Network

R — Route

TF — Transfer function

NN — Neural networks

AI — Artificial intelligence

INTRODUCTION

The research of neural networks has made a promising development in recent years, and neural networks have become a popular topic especially in the field of artificial intelligence research. This is particularly the case because with the help of neural networks, artificial intelligence systems can be developed that are based on the principle of human learning and are modeled on the processing of information by the nervous system.

At the same time, it is a relatively complex and abstract area of research that is very difficult to grasp and understand for a newcomer to this field. As a result, this important and promising area is not yet sufficiently widespread, especially in the industrial sector.

Several years of research gave insight into the architecture and the performance of the human brain as a control system and showed that the controller with neural networks has significant advantages over conventional controllers. The neurocontroller can effectively process a much larger amount of information. Another very important advantage is that good regulation can be achieved by learning it.

In this thesis an internship experiment to investigate the performance of neurocontrollers in non-linear electric drive systems has to be developed. For this purpose, two drive systems with cascade speed control are considered. Dry friction is noted in one of these systems and dry and viscous friction is noted in the other. Inverse neuro-model is added for the controlled system of each system. Then the obtained inverse neuromodel is built into the system according to the method "Direct inverse control" and used as the speed controller. The transfer function of the series connection of the inverse model and the controlled system should strive towards one in the method used. In this way, the non-linearities present in the system can be compensated.

1. NEURAL NETWORKS IN AUTOMATION TECHNOLOGY

1.1 Neural systems - branch of artificial intelligence

If one looks at the development of neural networks, one can see that this has been influenced by the most diverse areas and that the neural systems have developed into a largely independent discipline in recent years. The beginning of this development was made by neurobiology, which still plays a decisive role in the further development of the NN today. The first technically oriented contributions to the development of the NN come from physics, where the attempt to describe physical phenomena from the field of magnetism or thermodynamics with neural methods has led to the development of some network types.

It was only when the potential for using neural networks to solve problems in the field of artificial intelligence (AI) became clear that NN became a popular research branch in computer science within a very short time and it can now be described as the newest branch of artificial intelligence research. This is illustrated in fig. 1, which shows the relationship between the NN and other AI research areas.

<i>ACIC DEPARTMENT</i>				<i>NAU 21 01 31 000 EN</i>			
<i>Performed</i>	<i>Bilai Y.S.</i>			Automatic DC motor control system with PI-speed neuroregulator	<i>N</i>	<i>Page</i>	<i>Pages all</i>
<i>Supervisor</i>	<i>Pantyeyev R. L.</i>						
<i>Normcontrol</i>	<i>Tupitsyn N. F.</i>				<i>431 151</i>		
<i>Dep. haed</i>	<i>Sineglazov V. M.</i>						

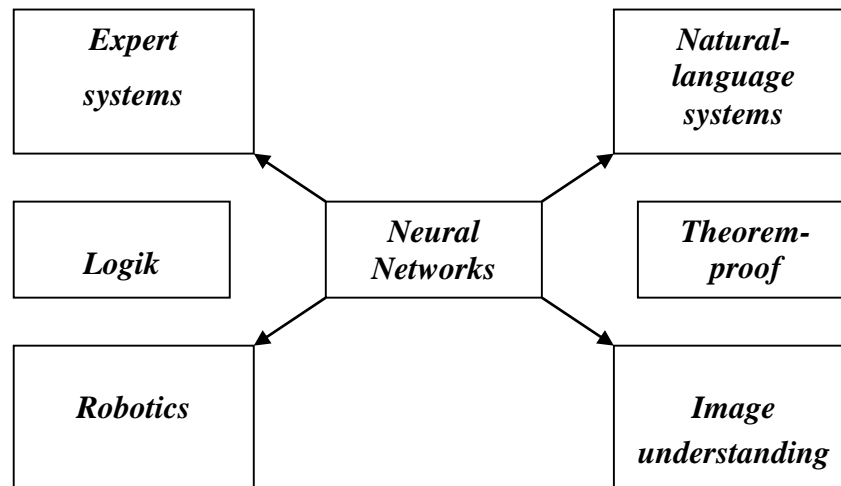


Fig. 1 - Neural networks as a research area in artificial intelligence.

Neural networks can be used successfully, for example, for problem solving in the field of image processing and natural language systems. The only area unrelated is the area of logic and theorem proof. This is the case because this area represents an approach that competes with NN. For example, one can find expert systems that are either based on methods of logic or use neural approaches, but rarely based on principles of logic and the NN at the same time (although such systems are possible). Methods of theorem proof and logic are the basic algorithms for the classic rule-based AI approaches [1].

In principle, AI research can be divided into two areas: The first research area comprises the rule- and knowledge-based approaches just mentioned, which can typically be found in expert systems in the form of a deduction component or in language-understanding systems in the form of grammars. The second research area deals mainly with methods of pattern recognition and the processing of human sensory perceptions. In these areas, neural approaches were the quickest to gain

acceptance. This also applies to the field of robotics, where a large number of control and regulation problems favor the processing of analog values with neural networks.

At a time when AI research was still mainly characterized by rule- and knowledge-based approaches, the term “machine learning” had already emerged. Here an attempt was made to develop AI systems that are based on logic-oriented approaches and that can draw rules and knowledge bases independently from examples. NN are, so to speak, a supplement to “machine learning” and have the closest relationship to this area.

Many pilot applications in artificial intelligence research in the 1980s were unsuccessful and had to be viewed as more or less failed. This was often due to the following reasons:

1. Lack of ability to learn. Rule-based systems generally have no learning ability, which in many cases meant that the rules had to be laboriously determined with the help of a detailed system analysis and then implemented in the form of a computer program. It had the following consequences: The creation of an AI application was associated with a great deal of time and money, because not only does it take a long time to find and implement all the rules, but testing and tuning the rules is also very time-consuming.
2. Lack of adaptive learning behavior. Many systems in practice are time-variant systems in which the application conditions change continuously during the application phase. Therefore, many applications require a system that is able to learn from new examples created during the application phase. A rule-based system is usually not easy to adapt. However, many AI systems

are initially designed as relatively simple systems and many special cases that actually require new special rules are only found over time through daily use of the system. The adaptive learning behavior of such a system is therefore an important property.

3. Small ability to process fuzzy information. Many problems in practice are “fuzzy problems”. The main problem with systems that are not able to process fuzzy information is the problem that it need exact input information in order to be able to derive results from it. However, if the entries do not fall outside the specified range of values, the system fails because it practically no longer knows what to do now.
4. Runtime problems. This is a very simple, common problem. However, this often leads to a rejection of the AI system, since a simplified version would not meet the requirements and a full version is too expensive.
5. Lack of integrability. This is one of the most common problems of AI applications, because almost every implemented AI system has to be integrated into existing software. It has been shown that even when using software tools that expressly have easy-to-use interfaces, considerable integration problems have arisen in many applications.

Neural networks are not a “magic bullet” for eliminating such problems. However, it has the potential to successfully run some applications that have failed for the reasons mentioned above. Neural systems can often deal with the problems mentioned above better than rule-based systems because they have the following properties [1]:



1. Ability to learn. This quality is one of the main strengths and abilities of the NN.
2. Adaptivity. It is true that not all NNs can be created adaptively in a simple way, but it is very easy to have an NN relearned with an expanded list of newly acquired examples.
3. Processing of fuzzy information. This is the typical domain of neural systems.
4. Massive parallelism and hardware implement ability of the NN have already been discussed. The great potential of the neuro-algorithms in terms of computing speed is thus available.
5. One advantage of the NN is that it is practically always about purely numerical algorithms. With the help of most of the available software shells, these can always be converted into programs that have been created, for example, in the “C ++” programming language.

1.2 General considerations of neural control for industrial processes

Research in the field of neural networks has undergone a much-noticed development over the past few years, and neural networks have become a popular topic, particularly in the field of artificial intelligence research. This is particularly the case because, with the help of neural networks, artificial intelligence systems can be created that are based on the principle of human learning and are modeled on the processing of information by the nervous system in biological organisms.

At the same time, however, neural networks are a relatively complex and abstract area of research that is very difficult to grasp and understand for a newcomer

to this field. As a result, this important and promising area is not yet sufficiently widespread, especially in the industrial sector [2].

The aim of the present work is therefore to convey the possible applications of neural networks in electrical drive technology.

In the field of process automation, mostly linear controllers are currently used, which are designed with the help of linearized, mathematical process models. For strongly non-linear processes, however, this approach leads to limited results. Another class of problems arises when the knowledge about the process can only be roughly expressed mathematically.

By using methods for black box modeling, a model of the process can be formed from measured values of process variables, which describes it with regard to its input / output behavior. Artificial neural networks (ANN) are particularly suitable here, as have a universal structure that is also able to adapt to non-linear behavior. A number of different control structures, which are based on process models, enable systematic control design for non-linear processes as well. A model is generated based on process data, which is embedded in a controller structure and used to regulate the process [3]. Since some of these control approaches are very computationally expensive, it may be necessary to have the behavior of the combination of controller structure and neural model learned by a neural controller and thus to regulate the process.

2. GENERAL DESCRIPTION AND FUNCTIONING OF NEURAL NETWORKS

Furthermore, the question should be clarified: What is a neural network? A very general answer to this question could be: A system made up of interconnected elements that can process information, called neurons. A general distinction is made between biological and artificial neural networks. In the case of a biological neural network, the neurons are nerve cells and the network is part of the nervous system of a biological organism. The processed information is biological information, which essentially consists of nerve impulses. In an artificial neural network, the neurons are implemented as mathematical or physical models with several inputs and outputs, whose mathematical behavior corresponds in principle to the biological neurons. The information processed there can generally be referred to as a sample. These can be signals, bit patterns or numerical values that are usually processed by an artificial neural network in the form of an input pattern and output in the form of an output pattern. Fig. 2 shows the schematic representation of an artificial neural network. A somewhat detailed representation is given in Fig. 3. Fig. 2 clarifies the basic functionality of an artificial neural network by representing the network as a system for pattern processing that contains an external pattern as an input, processes it internally and generates an output pattern from it.

The illustration in Fig. 3 shows the networking of the individual processing elements (neurons) with one another. The fact that all connections between the neurons contain weights is very important. In Fig. 3, some facts become clear that are generally applicable to artificial neural networks. For example, Fig. 3 shows that not all processing elements are interconnected. Also, not all neurons are exclusively connected to other neurons. Instead, the neurons can be divided into three different

<i>ACIC DEPARTMENT</i>				<i>NAU 21 01 31 000 EN</i>			
<i>Performed</i>	<i>Bilai Y.S.</i>			Automatic DC motor control system with PI-speed neuroregulator	<i>N</i>	<i>Page</i>	<i>Pages all</i>
<i>Supervisor</i>	<i>Panteyev R. L.</i>						
<i>Normcontrol</i>	<i>Tupitsyn N. F.</i>				<i>431 151</i>		
<i>Dep. haed</i>	<i>Sineglazov V. M.</i>						

classes: Some of the neurons are directly connected to the external input pattern, while another part of the neurons outputs the external output pattern [2].

A third part of the neurons is only connected to other neurons and thus has only internal and no external connections. The processing elements of a neural network are therefore usually divided into different layers, namely the input layer, the output layer and the hidden layer. The number of neurons in this last layer can be very large and it will be shown in the later course of this work that in many cases the neurons of this layer can be divided into further layers, which are generally referred to as "hidden layers".

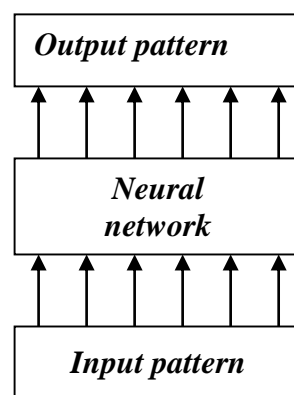


Fig. 2.1 - Schematic representation of a neural network

With the help of these findings, a more precise definition of an artificial neural network can now be made: A system for information processing with the help of simple networked elements with directed inputs and outputs and weighted connections that processes input patterns and generates the resulting output patterns.

How does information processing work in such a network? Without going into detail in this introduction, one can already give some thought to this if one remembers the formulation given above, that one can describe the behavior of neurons with a mathematical model to which a certain transfer behavior between its inputs can be described and can assign outputs, which is characterized by various parameters, in particular by the weights of the neuron inputs. A transfer function common in control engineering can serve as a model for describing a dynamic

system, the transfer behavior of which depends on the coefficients in the denominator and numerator of the transfer function. A neuron therefore generates an output signal from its input signals according to a mathematical rule. In this way, the input pattern is transformed by the neurons of the input layer and the corresponding outputs in turn become inputs of the neurons in the hidden layer, where the input pattern is further transformed. With a large number of processing layers, the input pattern is subjected to an extraordinarily complex transformation that can completely change both the shape and the dimensions of the pattern and, as a result, delivers the corresponding output pattern at the output of the neurons of the output layer.

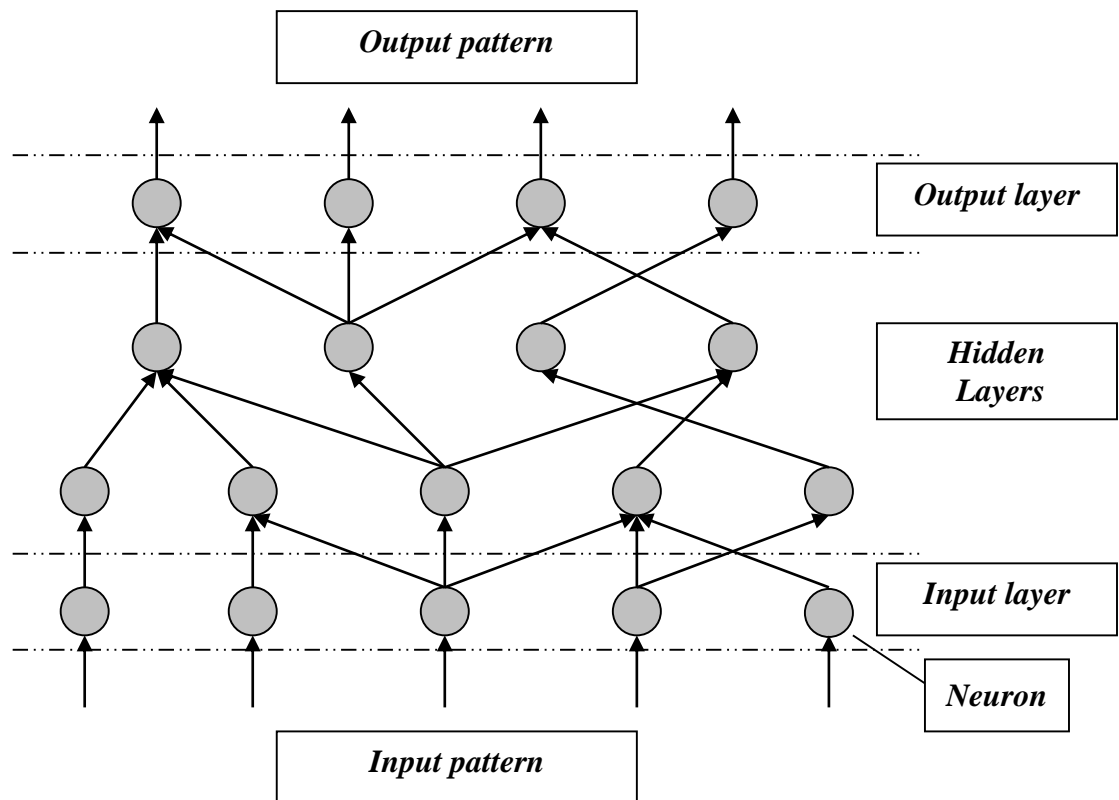


Fig. 2.2 - Detailed representation of a neural network

There is a certain mathematical relationship between the input and output patterns, which is determined by the parameters of the mathematical neuron models and the network topology of the neurons. If stick to the example of control engineering, it can still imagine a transfer function between the input and output

patterns, which is, however, very complex and non-linear and is determined by a large number of parameters [5]. In a typical NN application, the problem of which transmission behavior is present between the input and output pattern is usually not investigated when the NN parameters are specified, but the opposite problem is initially in the foreground, namely: How are the NN parameters chosen to achieve a specific transfer behavior between the input and output patterns? This problem can be solved by presenting the network with a more or less large number of sample patterns at its input or output (sample pattern pairs) and using an optimization process to try to optimize the network parameters so that it adopts the desired transmission behavior.

Here, too, the example from control engineering is again appropriate, since such a procedure corresponds in principle to a system identification and the mentioned optimization method corresponds to a parameter estimation method. In a neural network, this phase is called the learning phase or training phase, in which the network learns from examples to adapt its parameters so that it adopts the desired transmission behavior. The possibility of adapting the parameters of the neural network to the sample pattern with the help of suitable mathematical procedures gives the network its important property, namely the ability to learn from examples. Since - as mentioned above - the network parameters primarily consist of the weightings of the connections between the neurons, the weightings of a neural network are responsible for the network's ability to learn. For an NN, learning therefore means that it tries to set its weightings in such a way that it generates output patterns from the presented reference input patterns that are as similar as possible to the reference output patterns.

Once the network has learned the relationship between the reference input and output patterns in the learning phase, it can be used in the actual application phase. The NN with now firmly learned parameters is used to process a pattern processing task in which the network is presented with input patterns that it has not yet seen in the learning phase and it generates the desired output patterns from them [5]. These

can be tasks for recognizing patterns or for converting and storing patterns from a wide variety of application areas.

In summary, it can be stated that a typical NN application always consists of two phases: In the first phase - the learning or training phase - the weightings of the NN are determined using examples for input and output patterns so that the NN matches the systematic relationship between the sample pattern pairs. In the second phase - the actual application phase - the network is presented with input patterns with its learned weightings, from which it then generates output patterns and thus solves a specific pattern processing task.

3. SPECIAL PROPERTIES OF NEURAL NETWORKS

The learning ability of neural networks already described gives the neural algorithms a special property that most other algorithms do not have, but which is very advantageous and often a necessary prerequisite for solving a variety of problems - especially in the field of artificial intelligence. Because of this, neural networks have grown in popularity very quickly and continue to evolve at a rapid pace. However, neural networks have some other special properties that distinguish them from other classical methods.

3.1 Ability to learn

In addition to the facts already mentioned about the learning ability of neural networks, some interesting consequences can be derived from this fact. Nowadays most tasks are solved with the computer in a procedure in which the person himself has to analyze the task in detail and with the help of a program, the computer has to teach the procedure to solve the task in complex individual steps. A computer based on neural principles could one day be able to cope with even complex tasks by automatically learning from sample solutions for this task. With such a computer, the programming step would be replaced by the training step [3].

It can also be shown that, under certain conditions, neural networks are able to systematically extract the essential relationships between these pairs of patterns, which are common to all of these pairs of patterns despite their differences, from a very large number of relatively different pairs of sample patterns. This is another very important property, as it makes it clear that the network is not only able to produce a pure "input-output mapping" between the input and output patterns, but that it can learn the essential systematics of the pattern processing task and can run

<i>ACIC DEPARTMENT</i>				<i>NAU 21 01 31 000 EN</i>			
<i>Performed</i>	<i>Bilai Y.S.</i>			Automatic DC motor control system with PI-speed neuroregulator	<i>N</i>	<i>Page</i>	<i>Pages all</i>
<i>Supervisor</i>	<i>Panteyev R. L.</i>						
<i>Normcontrol</i>	<i>Tupitsyn N. F.</i>				<i>431 151</i>		
<i>Dep. head</i>	<i>Sineglazov V. M.</i>						

from a multitude of examples to a generalization of the individual copies presented.

The NN can exploit this ability to generalize by being able to correctly process input patterns in the application phase that it never saw in the training phase. Some of these patterns can be completely different from the examples presented in the learning phase and can still be processed correctly, as they correspond to the same systematics of the learning examples and the network has recorded this general systematics during learning. This ability to generalize learning examples to examples never seen before gives the learning ability of the neural networks robustness [4]. This is necessary in order to be able to successfully cope with real applications from practice.

3.2 Ability to process incorrect and incomplete information

All network types have the ability to process incomplete and incorrect information. In many cases, it can be generating a correct output pattern from an input pattern that is noisy. In this way, incorrect input patterns can be processed correctly or transformed into error-free input patterns. It also applies to incomplete input patterns (this is the special case of incorrect patterns), which can be completed with the help of NN. Here, too, neural networks are better than most “classic methods”, since these methods usually fail in the case of incorrect or incomplete entries. Ordinary systems are designed to process correct and complete data, while neural networks can process so-called “fuzzy information”. However, a lot of information in everyday life is “fuzzy”, for example a word that is spoken quickly, which one did not understand exactly and which the human brain usually still processes successfully and correctly. The human brain is particularly capable of processing fuzzy information and it is therefore clear that NN can be used particularly successfully in the machine processing of voice and image signals.

3.3 Adaptive behavior

In addition to the ability to learn during the training phase, some networks also have the ability to continue learning in the application phase and consequently have an adaptive behavior. In this case, the weightings found in the learning phase are not kept constant in the application phase, but are continuously adapted to the current conditions, i.e. readapted, with the help of the patterns that are presented to the network during the application phase. But not all network types have this capability. However, it is very desirable in many cases, for example when NN work together with time-variant systems in which an adaptation to slowly changing conditions is necessary.

3.4 Massive parallelism

The massive parallelism of networks can already be seen from the consideration of Figure 3. This implies that the neurons can be viewed as autonomous systems. Their internal operations are independent of each other and they only communicate with each other through the weighted links. As a result, an NN can be viewed as a network of independent, parallel working individual systems. It has already been mentioned that the behavior of a neuron can be simulated with the help of a mathematical or physical model. A mathematical model is realized with the help of a computer program and the calculation of the entire network is carried out as a simulation of all neuron models coupled to one another on a conventional computer. However, due to the massive parallelism, one can also imagine realizing the simulation programs for the individual neurons on an extra processor each, which can be a very simple microprocessor. The overall network could then be implemented as a connection between all individual processors. This would have the advantage that the simulation programs of the individual neurons would then no longer have to run sequentially on a conventional computer, but that they could run in parallel on all available individual processors. An enlargement of the network would then result in an increase in the number of processors and would not have a disadvantageous effect on the computing time for the network, since each processor can contribute its computing power to the overall performance of the network at

normal speed. This would significantly increase the overall performance of the network [5]. It should also be noted that because of the very simple design of such processors, the individual elements of such a hardware network would be very cheap and large networks with very high computing power can be implemented cost-effectively. The massive parallelism of neural networks thus represents a possible approach for replacing conventional computers in the future with computers that work in parallel, which can then provide a multiple of computing power.

3.5 Fault tolerance

The fault tolerance of the neural systems is closely linked to the property of massive parallelism and hardware implement ability. In the case of a large and massively parallel system made up of several thousand elements working in parallel, there is a likelihood that in the case of a parallel computer in which a processor fails during operation, this can lead to a total failure of the entire system because Each processor in such a system assumes an important function and in particular the failure of communication with the other processors can lead to considerable problems. In such a case, the parallel computer could still be operated with a smaller number of processors, but a disruption of the program currently running would certainly be unavoidable at the time of the processor failure and all applications would have to be adapted to the lower number of processors, e.g. by recompiling and restarting.

In the case of an NN, it can be observed that in most cases a failure of individual neurons does not lead to any significant change in network behavior. This only applies in the event that the number of failed elements is relatively small compared to the total number of neurons in the network. In most cases, the transition to a significantly poorer network behavior is fluid, i.e. with an increasing number of failed elements, a continuous change in network behavior is associated and a rapid total network failure if a certain number of elements fails. not to watch.

This very advantageous, fault-tolerant behavior can be explained using some of the properties of neural networks that have already been listed: Due to the massive parallelism mentioned, the overall functionality of the network is very widely distributed over a large number of elements. Each individual element (neuron) has a very simple structure and does not have a particularly high level of performance on its own. The effectiveness of these systems is achieved through the strong connectivity of the elements with each other and the effective distribution of the task to be solved over the total number of neurons, as well as the effective control of the interaction of the neurons with the help of the weights. Therefore, the failure of a simple element can hardly affect the overall behavior, as long as enough other elements are still present. The fault tolerance is also favored by the fact that in most cases the network processes fuzzy information.

The property of fault tolerance is particularly important for the applicability of neural systems under critical operating conditions, in which an extremely high availability of the system must be guaranteed. Examples of this are applications in space, in flight safety, in the military sector, in the monitoring of complex technical systems. In such cases, an NN can also be designed to be particularly fault-tolerant by designing it deliberately oversized, i.e. using a larger number of neurons in the "hidden layer" than is absolutely necessary. In this case, the overall behavior of the is particularly well distributed and the failure of some elements is all the easier to cope with.

3.6 Hardware implementability

One also thinks of the simulation of an NN with the help of a physical model, e.g. realized by an analog electrical circuit or with the help of optical processes. It is important to note that a hardware implementation can lead to an enormous advantage in computing time. If one thinks of the advances in microelectronics and optoelectronics, one can also easily imagine that neural networks with high computing speed and a large number of neurons in the smallest dimensions can be

realized in this way. This has the following consequences for the practical applicability of NN: NN can be “tailor-made” for certain applications and they can be implemented as chips. Small dimensions and high processing speeds make real-time applications possible under a wide variety of operating conditions without the need for expensive control computers. Applications in the immediate vicinity of the machine (e.g. in engines, in cars, etc.) can be implemented. These applications can also be implemented inexpensively through mass production. Overall, it can be stated that practically all the advantages that the development of microelectronics has brought for the use of digital processors can be transferred in a similar way to neural networks.

It is also interesting to note that NN are one of the few paradigms in AI research that can be efficiently implemented in hardware. Such an attempt has also been made, for example, for rule-based paradigms from the field of logic and inference systems and has had an impact on the development of LISP machines, for example. However, due to their high prices, large dimensions and incompatibility with other systems, these have not been able to establish themselves. Most rule-based systems are and will remain pure software implementations, which in most complex applications requires the use of a computer and can thus lead to a restriction of the range of applications.

4. FEEDFORWARD NETWORKS

The most common basic network architecture is that of feedforward networks. A Net with the feedforward architecture only has connections between the neurons in one direction, namely from one layer to a “higher” layer. Higher means here that this layer is closer to the starting layer than a layer below it. A feedforward network in Figure 5 would only have connections from the bottom to the top. There are no connections between from one layer to a layer below, and there are also no connections between the neurons of a layer [6]. In the normally used feedforward network architectures, there are always only connections between one layer and the next higher layer directly above it.

The network architecture shown in Fig. 4.1 is one of the most frequently used architectures for feedforward networks, with two hidden layers and one input and one output layer. The connection between the input pattern and the input layer only serves to present the pattern and does not yet contain any weightings or totals. Typical summations of the weighted inputs only come into play with the neurons of the first hidden layer and are continued by the neurons of the second hidden layer and the output layer. Our network (fig. 5) has three active layers that contribute to the transformation of the input pattern into the output pattern. That is why one speaks of a three-layer network.

Feedforward networks can have at least one active layer, but in many cases they have three active layers. Two-layer networks or networks with more than three layers are also used.

<i>ACIC DEPARTMENT</i>				<i>NAU 21 01 31 000 EN</i>			
<i>Performed</i>	<i>Bilai Y.S.</i>			Automatic DC motor control system with PI-speed neuroregulator	<i>N</i>	<i>Page</i>	<i>Pages all</i>
<i>Supervisor</i>	<i>Pantyeyev R. L.</i>						
<i>Normcontrol</i>	<i>Tupitsyn N. F.</i>				<i>431 151</i>		
<i>Dep. head</i>	<i>Sineglazov V. M.</i>						

Feedforward networks have a number of other typical properties:

- Inputs and outputs are continuous
- Using the sigmoid function as an activation function
- Mostly different dimensions of input and output patterns
- Main application in pattern classification and assignment
- Mathematical description as a static system (the output pattern is calculated from the input pattern in a single "forward step") [6]

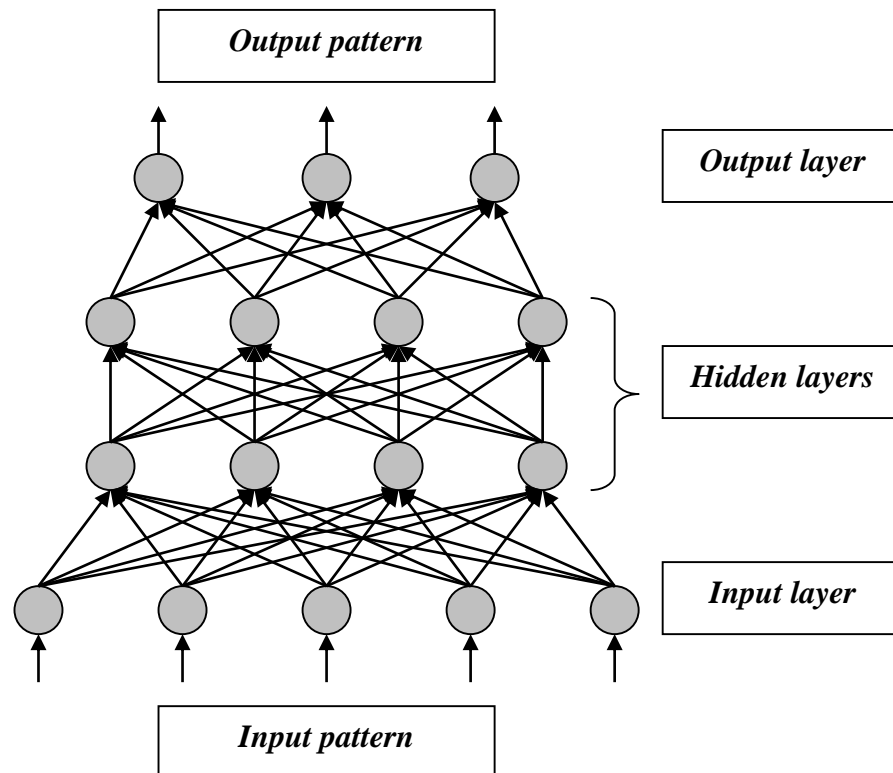


Fig. 4.1 - Architecture of a feedforward network

4.1 Model of a processing element

The detailed structure of a single processing element can be seen in Figure 6. Input variables $x_1 \dots x_n$ are initially weighted with the weighting factors $w_1 \dots w_n$. The actual neuron is represented by the two functions $G(x)$ and $F(G)$. The function $G(x)$ is referred to as the propagation function and in most cases is a pure summation function that provides the sum of the weighted input variables as the output variable. An important property of neural networks is their non-linear behavior, which is generated in that the output of the propagation function $G(x)$ is further processed by a non-linear function $F(G)$, the so-called activation function.

Our neuron also contains an element at the input of the variable θ , which takes on the function of a “bias” and is fed to the input of the NN without weighting [10]. It ensures that with a wide variation of the inputs, the output of the processing element is on average - depending on the choice of θ - is positive or negative.

In many cases, θ can be set to zero. But there are also some cases in which it makes sense to set this variable to a value other than zero.

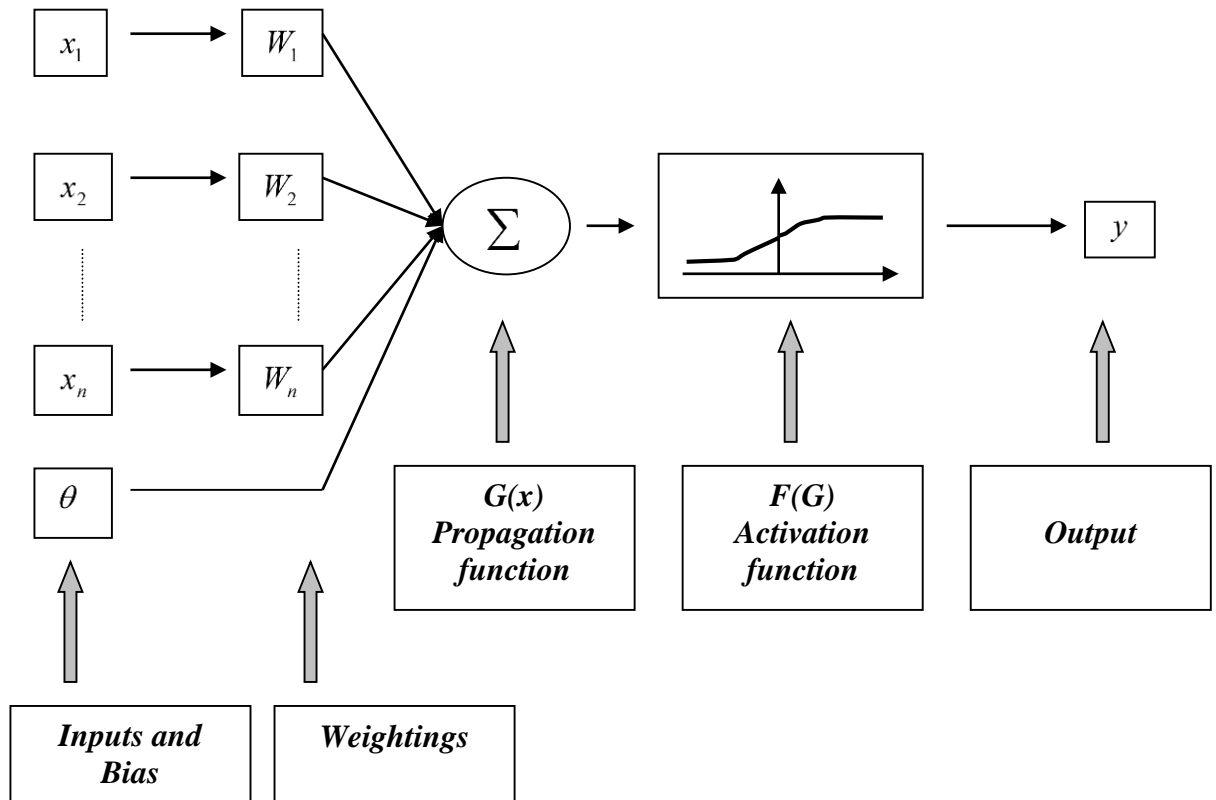


Fig. 4.2 - Structure of a single processing element

On the basis of the explanations given above, the following equation results for the output variable y of the processing element:

$$y = F\left(\sum_{i=1}^n w_i \cdot x_i + \theta\right) \quad (4.1)$$

In order to efficiently represent equation (1) in vector notation, the vectors \mathbf{x} and \mathbf{w} can be introduced:

$$\mathbf{x} = [x_1, x_2, \dots, x_n, 1]^T \quad (4.2)$$

$$\mathbf{w} = [w_1, w_2, \dots, w_n, \theta]^T \quad (4.3)$$

This means that equation (1) can also be written in the following form:

$$y = F(\mathbf{x}^T \cdot \mathbf{w}) \quad (4.4)$$

From equation (3) it follows that the factor can formally be viewed as an additional weighting factor which, based on equation (2), receives a constant input variable from $x_{n+1} = 1$. For the hard limiter, which is one of the most commonly used activation functions, one gets:

$$y = \begin{cases} 1 & \text{for } \mathbf{x}^T \cdot \mathbf{w} \geq 0 \\ 0 & \text{for } \mathbf{x}^T \cdot \mathbf{w} < 0 \end{cases} \quad (4.5)$$

and for the commonly used sigmoid function:

$$y = \frac{1}{1 + e^{-(\mathbf{x}^T \cdot \mathbf{w} + a)}} \quad (4.6)$$

This function converges to 1 for $s \rightarrow \infty$ and tends towards zero for $s \rightarrow -\infty$.

4.2 Network layer model

Let's consider two network layers shown in Fig. 4.3. The output values in the M neurons of the upper layer are calculated from the output values of the N neurons of the lower layer.

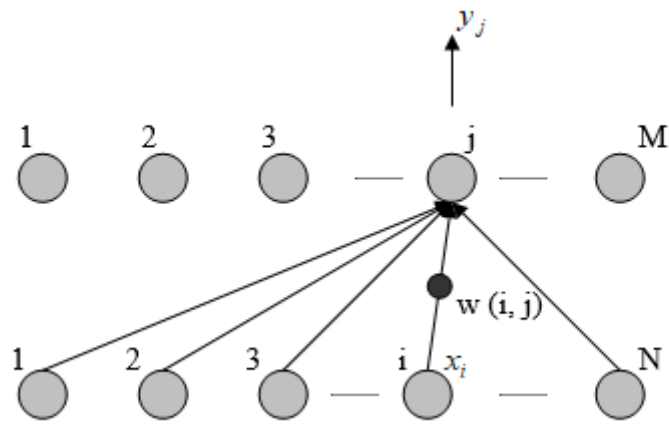


Fig. 4.3 - Calculation of the outputs of a network layer

The current index of the upper layer neurons is j , where j goes from 1 to M . The running index of the lower layer neurons is i , where i goes from 1 to N . Each of the lower layer N outputs is connected to all of the upper layer neurons. Therefore every neuron of the upper layer has N inputs which are identical to the N outputs of the lower layer. If consider only one processing element j of the output layer, it can calculate the corresponding output value with the aid of equation (4) [10]:

$$y_j = F(\mathbf{x}^T \cdot \mathbf{w}_j) \quad (4.7)$$

The vector \mathbf{w}_j thus contains the weightings of the connections from the N neurons of the lower layer to the j -th neuron of the upper layer and is thus a vector with the dimension N . These connections are shown in Figure 8. The weightings of the individual connections are not shown, except for the weighting w_{ij} , which represents the weighting between the i -th neuron of the lower layer and the j -th neuron of the upper layer. Overall, we have M weighting vectors \mathbf{w}_j ($j=1, \dots, M$) each with N components and thus a total of $N \cdot M$ weighting factors. Each vector \mathbf{w}_j contains the weights between all neurons of the lower layer and the j -th neuron of the upper layer. That's why it can write:

$$\mathbf{w}_j = [w_{1j}, w_{2j}, \dots, w_{Nj}]^T \quad (4.8)$$

With the help of equation (7) one can calculate the output value for each neuron j of the upper layer. If one considers equation (7) for all output values $j=1, \dots, M$, one can write:

$$[y_1, y_2, \dots, y_M] = F(\mathbf{x}^T \cdot [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_M]) \quad (4.9)$$

or with the introduction of the weighting matrix W :

$$\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_M] = \begin{bmatrix} w_{11} & \dots & w_{1M} \\ \dots & \dots & \dots \\ \dots & \dots & \dots \\ w_{N1} & \dots & w_{NM} \end{bmatrix} \quad (4.10)$$

The number of rows in the weighting matrix thus corresponds to the number of neurons in the lower layer and the number of columns corresponds to the number of neurons in the upper layer.

4.3 Model of the entire network

Let us describe a two-layer network, which is shown in Figure 9. The two-layer network can be viewed as two single-layer networks. The first network thus generates the output vector \mathbf{x} from the input vector \mathbf{y} . The weightings of the first layer are combined in the matrix \mathbf{W}_1 . The output vector obtained is treated by the second layer like a new input vector and made available to all neurons of the second layer with the weightings contained in the matrix \mathbf{W}_2 as shown in Figure 9. The equation (12) for the output vector \mathbf{z} can be interpreted as a double matrix transformation with non-linear distortion of the input vector.

$$\text{Layer 1:} \quad \mathbf{y}^T = F(\mathbf{x}^T \cdot \mathbf{W}_1) \quad (4.11)$$

$$\text{Layer 2:} \quad \mathbf{z}^T = F(\mathbf{y}^T \cdot \mathbf{W}_2) = F(F(\mathbf{x}^T \cdot \mathbf{W}_1) \cdot \mathbf{W}_2) = G(\mathbf{x}^T) \quad (4.12)$$

It can be seen that this transformation $G(\mathbf{x}^T)$ depends only on the assumed nonlinear activation function F of the neurons and in particular on the elements in the matrices \mathbf{W}_1 and \mathbf{W}_2 .

With these explanations, the basic principle of learning in neural networks is made clear once again: The parameters of the complex transformation G , which consist of the weightings in \mathbf{W}_1 and \mathbf{W}_2 are determined in such a way that when example vectors \mathbf{x} are presented, the associated example vectors \mathbf{z} are as good as possible can be simulated and the error that occurs is as small as possible. It follows that the learning process consists of determining the parameters in the weighting matrices.

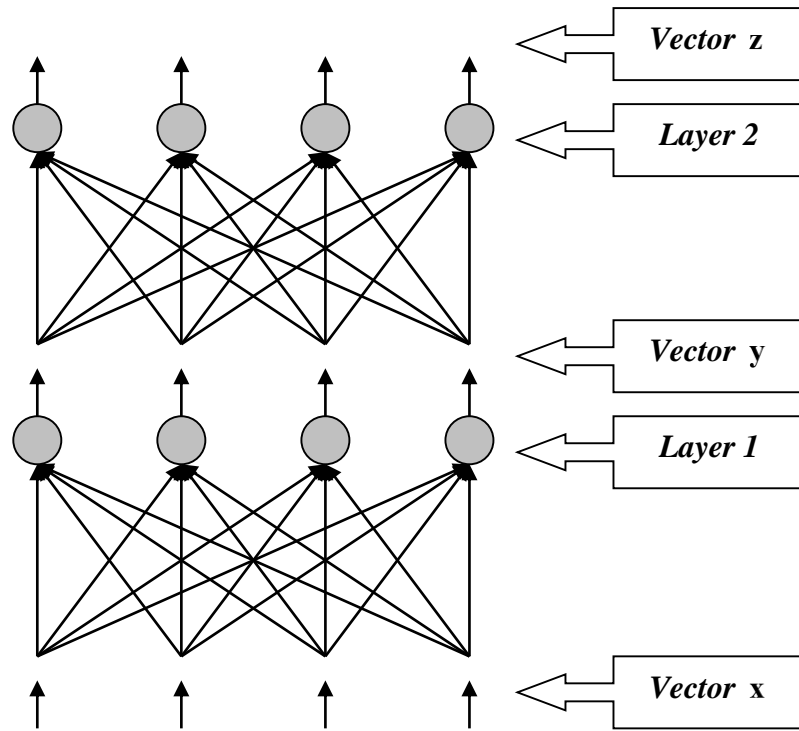


Fig. 4.4 - Two-layer neural network

4.4 Error back propagation

Backpropagation is the most common method of error recovery. But there are also some modifications of this feeling: quick propagation, elastic spreading [3]. The backpropagation network selected from layers with entry, exit and hidden rights (Fig. 10). The signal transmission is called in the forward direction. The actual and setpoint values are checked at the output. If this causes an error, it is converted backwards so that the weights of each layer are corrected. But the convergence of trust is not seen to be accomplished.

Now let's look at learning after backpropagation:

1. Initialize weights
2. Enter the learning file with neuron inputs x_i ($i=1\dots N$) and the desired output d for all learning patterns M :

$$(x_1, x_2, x_N, d)_1, (x_1, x_2, x_N, d)_2, \dots, (x_1, x_2, x_N, d)_M \quad (4.13)$$

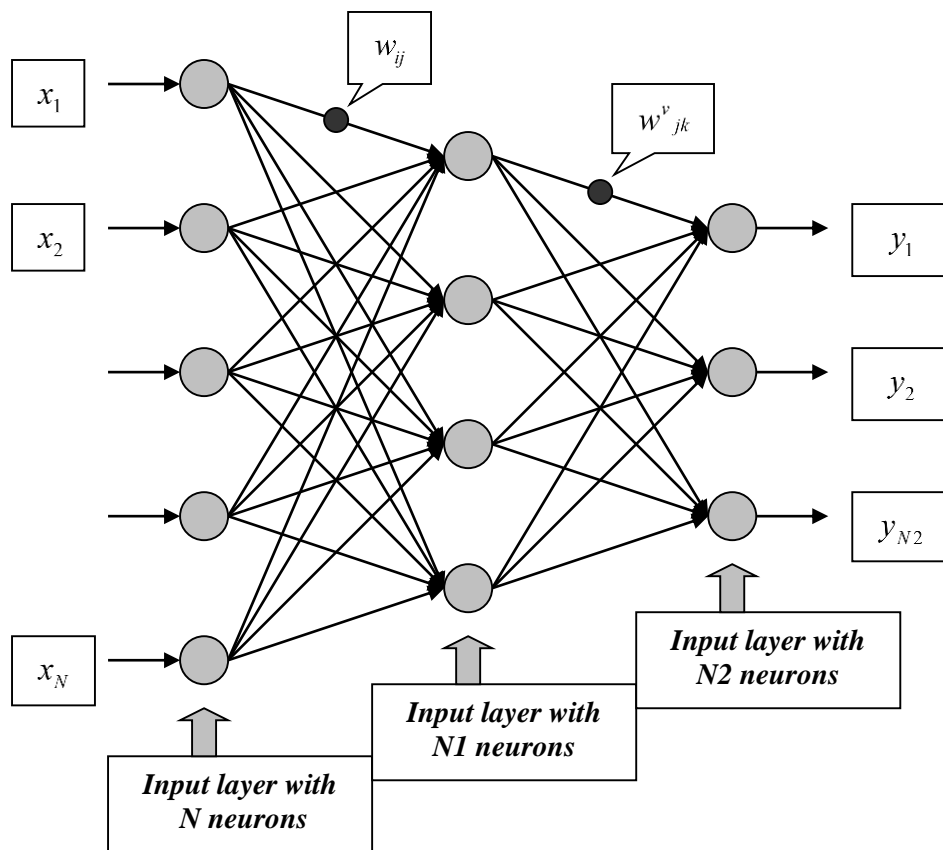


Fig. 4.5 - Two-layer neural network with N inputs and N_2 outputs

3. Calculate input transmission in forward direction
 - 3.1) hidden layer (Outputs α_j , and v_j):

$$\alpha_j = \sum_{i=1}^N w_{ij} x_i + \theta_j \quad v_j = f(\alpha_j) \quad j = (1 \dots N_1) \quad (4.14)$$

3.2) Output layer (Outputs α_k , and y_k):

$$\alpha_k = \sum_{j=1}^{N_1} w_{jk}^v v_j + \theta_k \quad y_k = f(\alpha_k) \quad k = (1 \dots N_2) \quad (4.15)$$

4. Calculate the total error for all learning patterns M n:

$$E = \sum_{p=1}^M E_p^2 = \sum_{p=1}^M (d_p - y_p)^2 \quad (4.16)$$

5. Calculate weights and thresholds value change respectively:

5.1) Output neurons:

$$w_{jk}^{v(neu)} = w_{jk}^v + \Delta w_{jk}^v \quad \theta_{k(neu)} = \theta_k + \Delta \theta_k \quad (4.17)$$

$$\Delta w_{jk} = \eta \cdot E_j \cdot x_k \quad \text{if} \quad E_j = (d_j - x_j) \cdot x_j \cdot (1 - x_j) \quad (4.18)$$

$$\Delta \theta_k = \eta \cdot E_j \cdot x_k \quad (4.19)$$

5.2) hidden neurons:

$$w_{ij}^{(neu)} = w_{ij} + \Delta w_{ij} \quad \theta_{j(neu)} = \theta_j + \Delta \theta_j \quad (4.20)$$

$$\Delta w_{ij} = \eta \cdot E^v_i \cdot x_j \quad \text{if} \quad E^v_i = \left(\sum_k E_k \cdot w_{kj} \right) \cdot x_j \cdot (1 - x_j) \quad (4.21)$$

$$\Delta \theta_k = \eta \cdot E^v_i \cdot \theta_k \quad (4.22)$$

This ends a learning step.

5. OVERVIEW OF NEURAL REGULATIONS AND CONTROLS FOR COMPENSATING NON-LINEARITIES

The use of neural networks for regulation and control is connected with the ability to recognize the state of dynamic systems.

The network inputs are state variables of the control loop. The network outputs are manipulated variables or characteristic values of the controller. The supervised learning methods are mainly used as the learning method. This creates a new feedback, namely the “learning feedback”, which works separately from the control loop feedback [8].

The concepts of control with NN are listed below:

1. State control
2. Predictive regulation
3. Adaptive control

With regard to the implementation of these concepts, the following rule structures can be designated:

1. One-network approach
2. Two-network approach

<i>ACIC DEPARTMENT</i>				<i>NAU 21 01 31 000 EN</i>			
<i>Performed</i>	<i>Bilai Y.S.</i>			Automatic DC motor control system with PI-speed neuroregulator	<i>N</i>	<i>Page</i>	<i>Pages all</i>
<i>Supervisor</i>	<i>Pantyeyev R. L.</i>						
<i>Normcontrol</i>	<i>Tupitsyn N. F.</i>				<i>431 151</i>		
<i>Dep. head</i>	<i>Sineglazov V. M.</i>						

3. Regulator network approach

Neural state control is shown in Fig. 5.1. With this concept of control with NN, the network receives the control difference $e(t)$ and its derivatives at its input, as well as the vectors of the manipulated and controlled variables $y(t)$ and $x(t)$.

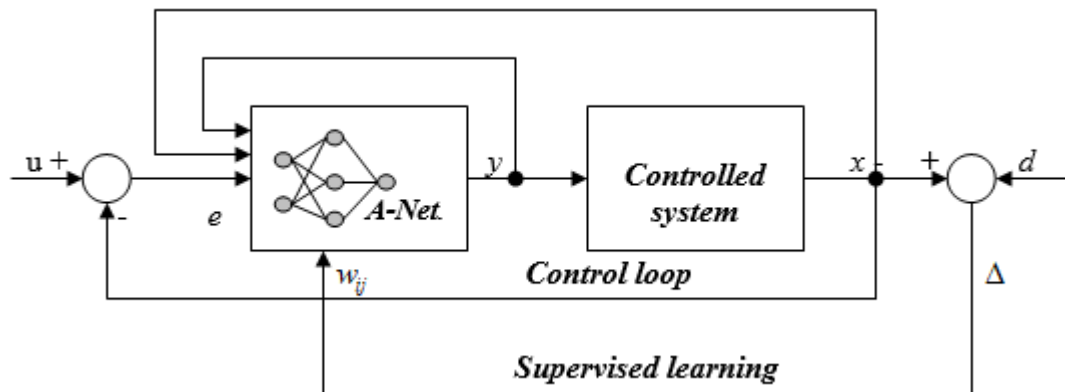


Fig. 5.1 - Neural state control. One-network approach

The output of the NN is the manipulated variable $y(t)$, which is why the network is referred to as an “action network” or “A network” in this case.

In order to optimize the control process, the trained A network should form an inverse transfer function compared to the transfer function of the controlled system [5]. Therefore, the task of the neural network is to set its own weights after the monitored learning so that the difference between the desired output d and the current output x is minimal.

The next procedure is shown in Fig. 5.2. This method belongs to the category of predictive control and in this case is implemented with a two-network method.

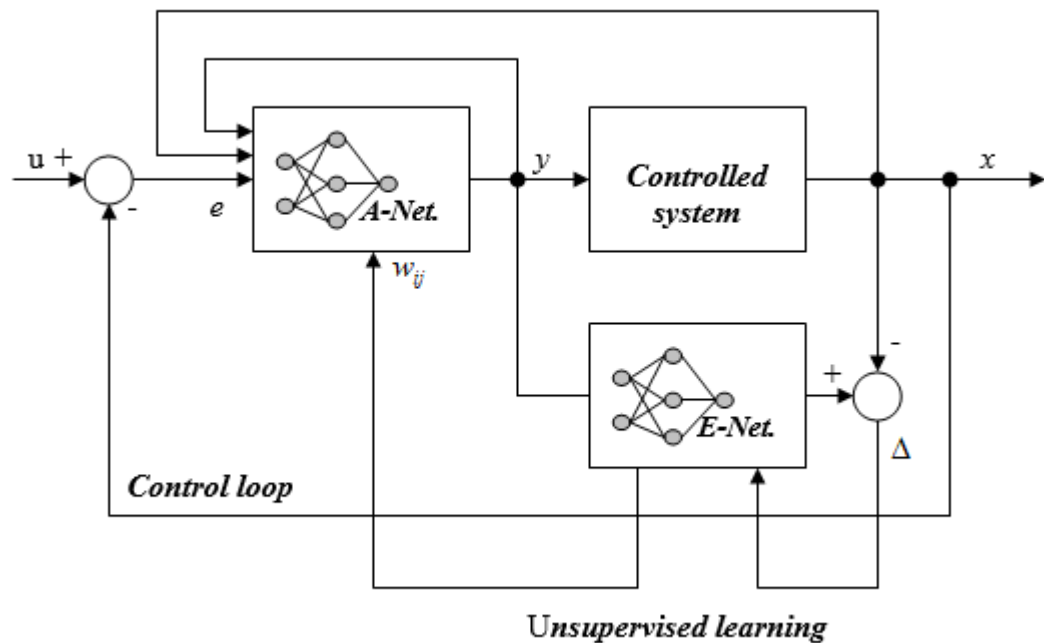


Fig. 5.2 - Neural predictive control. Two-network approach.

First, the “emulator network” (“E network”) is trained with the inputs and outputs of the controlled system. The A network is then trained, as in the previous case of the neural state control, but this time by the E network.

This procedure is referred to in the literature as “Model Predictive Control” [6] and is a powerful procedure despite a long learning period.

The combination of both methods is also possible in that the A network is not trained by the E network, but by monitored learning directly from the controlled system to an inverse transfer function. Such procedures are referred to in the literature as "Internal Model Control" [6].

The adaptive neural control is shown in Figure 5.3 using the controller-network method. The weights W_{ij} of the NN are here equivalent to the controller setting parameters.

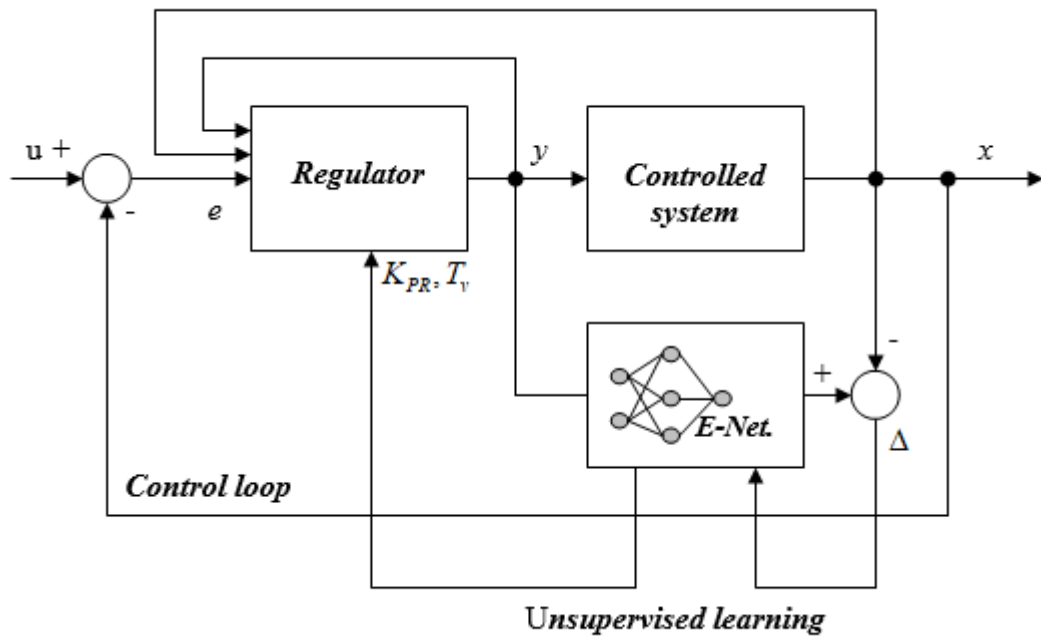


Fig. 5.3 - Neural adaptive control. Regulator network approach

This process is modified when the controller is replaced by an A network.

The NN for the control can again be implemented with one or two network processes. Fig. 5.4 shows a model-based control system with two networks. As with the new regulation, a distinction is made between the following two operating modes, learning and controlling.

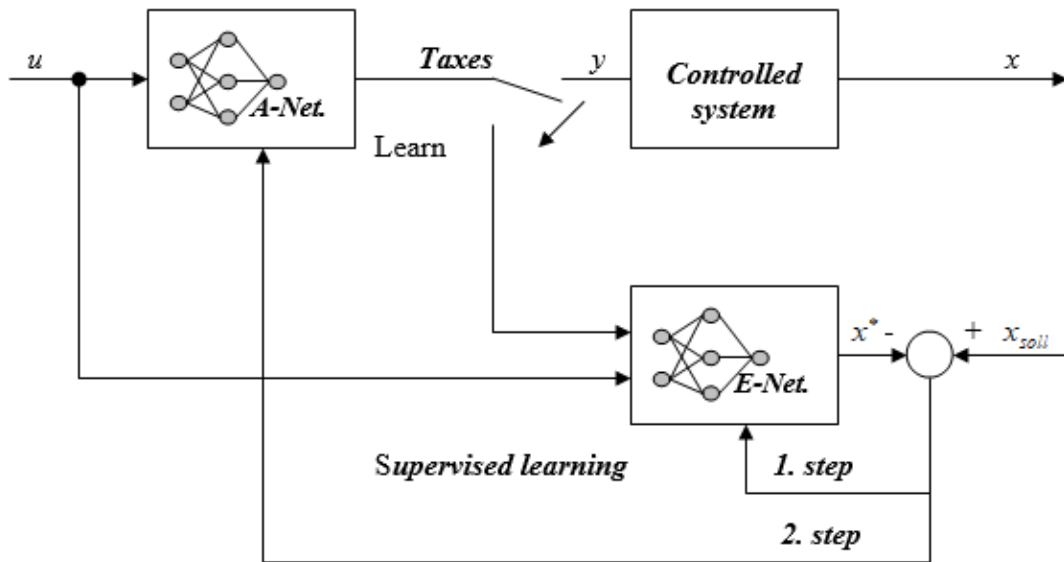


Fig. 5.4 - Neural control. Two-network approach

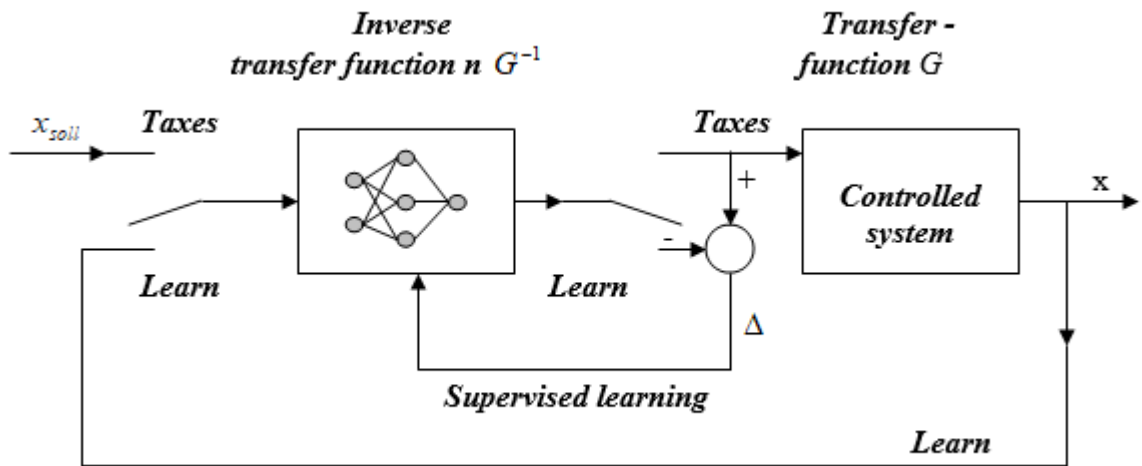


Fig. 5.5 - Neural control. One-network approach

6. CLASSIFICATION OF THE TYPICAL NON-LINEARITIES IN PRACTICAL ELECTRIC DRIVE SYSTEMS

This chapter looks at typical non-linearities in practical electric drive systems and their classification.

Definition of a non-linear system: If the system has non-linear transfer elements, then this system is non-linear.

If there are input functions or real numerical values for a transfer element, so that either the superposition principle or the amplification principle is not fulfilled, then the transfer element is called non-linear [13].

In practice, non-linear transmission elements occur in almost all technical applications. *In nature, nonlinear systems are the rule and linear systems are the exception [13].* However, linearization of most of the non-linear characteristics is permissible, so that often only one non-linear element has to be taken into account.

Non-linear control loops can only be examined in the time domain or the state (phase) level, while in linear systems an analysis and synthesis is possible in the frequency domain. Powerful design tools are known in linear theory, but no uniform non-linear systems theory exists. However, there are certain methods mainly for analyzing the stability of nonlinear systems:

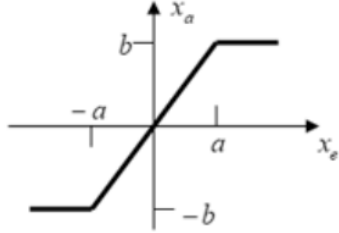
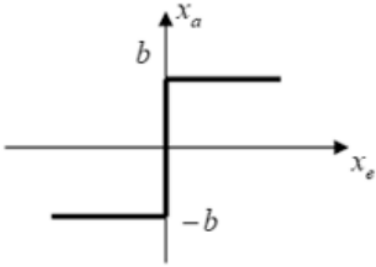
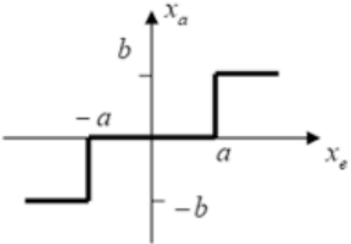
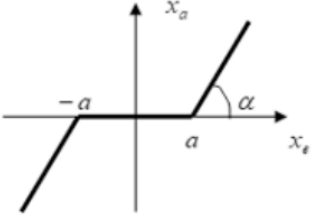
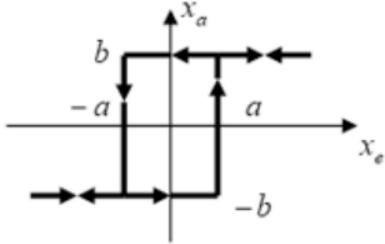
ACIC DEPARTMENT				NAU 21 01 31 000 EN			
<i>Performed</i>	Bilai Y.S.			Automatic DC motor control system with PI-speed neuroregulator	<i>N</i>	<i>Page</i>	<i>Pages all</i>
<i>Supervisor</i>	Panteyev R. L.						
<i>Normcontrol</i>	Tupitsyn N. F.				431 151		
<i>Dep. head</i>	Sineglazov V. M.						

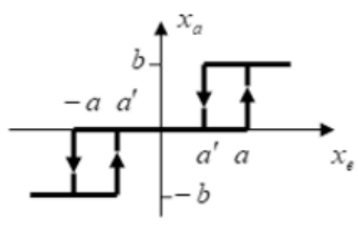
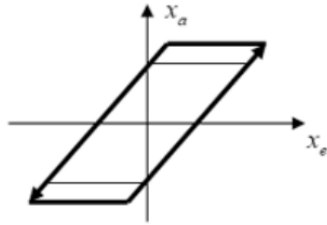
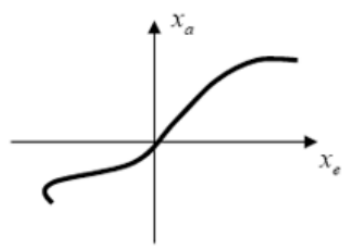
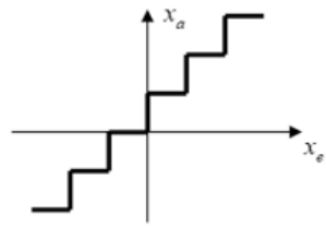
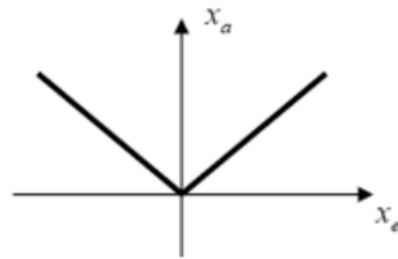
- a) Method of harmonic linearization,
- b) Phase level method,
- c) Lyapunov's second method,
- d) Stability criterion according to Popov.

There are various options for classifying non-linear transmission elements. The classification is often based on mathematical criteria, only taking into account the form of the differential equation in question. The second possibility is to use the most important non-linear properties that occur in particular in technical systems for a classification. For this purpose, one considers continuous and discontinuous non-linear system characteristics, which are compiled in Table 1. A distinction is made between clear characteristics (cases 1 to 4) and ambiguous characteristics (cases 5 to 7). The characteristics are often symmetrical to the origin of the coordinate system. A subdivision into unwanted and wanted nonlinearities is often recommended.

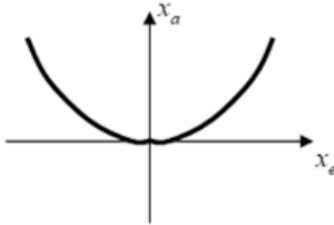


Table 6.1 - List of the most important non-linear terms

1	<p><i>Limitation</i></p> 	$x_a = \begin{cases} -b & \text{for } x_e \leq -a \\ \frac{b}{a}x_e & \text{for } -a \leq x_e \leq a \\ b & \text{for } x_e \geq a \end{cases}$
2	<p><i>Two point behavior</i></p> 	$x_a = b \operatorname{sgn} x_e = \begin{cases} -b & \text{for } x_e < 0 \\ b & \text{for } x_e > 0 \end{cases}$
3	<p><i>Three-point behavior</i></p> 	$x_a = \begin{cases} -b & \text{for } x_e < -a \\ 0 & \text{for } -a < x_e < a \\ b & \text{for } x_e > a \end{cases}$
4	<p><i>Dead zone</i></p> 	$x_a = \begin{cases} (x_e + a) \tan \alpha & \text{for } x_e < -a \\ 0 & \text{for } -a \leq x_e \leq a \\ (x_e - a) \tan \alpha & \text{for } x_e > a \end{cases}$
5	<p><i>Hysteresis behavior</i></p> 	$x_a = \begin{cases} -b & \text{for } x_e < -a \\ b \operatorname{sgn}(x_e - a \operatorname{sgn} \dot{x}_e) & \text{for } -a < x_e < a \\ b & \text{for } x_e > a \end{cases}$

6	<p><i>Three-point behavior with hysteresis</i></p> 	<p><i>Complex and difficult to visualize mathematical formulation</i></p>
7	<p><i>Gearless</i></p> 	<p><i>Complex and difficult to visualize mathematical formulation</i></p>
8	<p><i>Any non-linear characteristic</i></p> 	<p>$x_a = f(x_e)$</p>
9	<p><i>Quantization</i></p> 	<p>x_a can only gradually assume discrete values</p>
10	<p><i>Module formation</i> x_a</p> 	<p>$x_a = x_e$</p>



11	<p><i>Squaring</i></p> 	$x_a = x_e^2$
12	<p><i>Multiplication</i> <i>(Division)</i></p>	$x_a = x_{e1} \times x_{e2}$ $x_a = \frac{x_{e1}}{x_{e2}}$

No physical system is exactly linear in the mathematical sense. The non-linearity can be weak and therefore negligible, it can also be strong and have a negative (sometimes positive) effect on the dynamic behavior of a system. On the other hand, non-linear elements are sometimes deliberately used in controller design, not only because it is easy and cheap to implement (e.g. switching controllers), but also to achieve special system properties that cannot be achieved with linear elements.

As already mentioned, when analyzing and synthesizing nonlinear systems one will often start directly from the representation in the time domain, i.e. one must try to solve the differential equations. Simulation methods are an important aid here. Digital and hybrid computing systems are particularly suitable for simulating non-linear systems; The analog computer can also be used for minor problems.

7. MODELING AND SIMULATION OF DRY FRICTION

Depending on the frequency response of linear control loop elements, a description function is defined for non-linear control loop elements. This function takes linearization into account and is particularly suitable for considering the stability of control loops using the two-position curve method.

One has to say that the linearization of a non-linear characteristic is successful if the change in the input signal is only slight. Then it is sufficient to replace the characteristic with the tangent at the respective operating point. However, this method fails in the case of characteristics with discontinuities.

If it considers the control loop in the frequency domain, a linearization can be carried out under the following conditions:

- 1) The control is in the steady state
- 2) Restriction to only one non-linear element in the control
- 3) The calculation refers to the ideal non-linear characteristic

With a sinusoidal input signal, the input and output variables are continuous oscillations. The output signal x_a of the non-linear element is then periodic, but not harmonic. It contains harmonics of different frequencies ($2\omega, 3\omega, \dots$), which can be specified with the Fourier analysis [12].

Each control loop has damping PT1 elements so that the harmonics can be neglected. Therefore one can restrict oneself to the consideration of the fundamental oscillation x_{a1} and has carried out a practically applicable linearization.

ACIC DEPARTMENT				NAU 21 01 31 000 EN			
<i>Performed</i>	Bilai Y.S.			Automatic DC motor control system with PI-speed neuroregulator	<i>N</i>	<i>Page</i>	<i>Pages all</i>
<i>Supervisor</i>	Pantyeyev R. L.						
<i>Normcontrol</i>	Tupitsyn N. F.				431 151		
<i>Dep. head</i>	Sineglazov V. M.						

The description function mentioned above, also known as harmonic balance, is only dependent on the amplitude of the input variable. Reduced to the fundamental oscillation of the output variable is defined:

$$N(\hat{x}_e) = \frac{x_{a1}(\omega t)}{x_e(\omega t)} \quad (7.1)$$

In complex notation is input and output size:

$$x_{a1}(\omega t) = a_1 \cdot e^{j(\omega t + \pi/2)} + b_1 \cdot e^{j\omega t} \quad (7.2)$$

$$x_e(\omega t) = \hat{x}_e \cdot e^{j\omega t} \quad (7.3)$$

This results in a form of the description function that is used for further calculation:

$$N(\hat{x}_e) = \frac{b_1 + j \cdot a_1}{\hat{x}_e} \quad (7.4)$$

The dry friction corresponds approximately to switching between two specified signal states (Fig. 7.1).

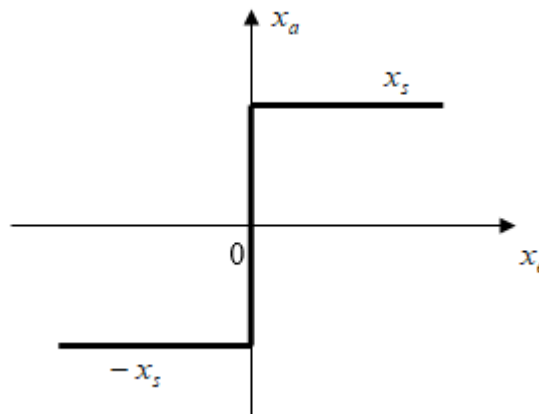


Fig. 7.1 - Characteristic curve of dry friction (two-point behavior)

Such behavior is called two-point behavior. Corresponding behavior can also be found with bimetal switches, solenoid valves, Schmitt triggers in analog and digital technology, with relay circuits and also as two-point controllers.

$$x_a = \begin{cases} x_s & \text{for } x_e \succ 0 \\ -x_s & \text{for } x_e \prec 0 \end{cases} \quad (7.5)$$

Here, the Fourier coefficient becomes $a_1 = 0$, because the static characteristic is an odd function. For b_1 then follows:

$$b_1 = \frac{2}{\pi} \cdot \int_0^{\pi} x_a(\varphi) \cdot \sin \varphi d\varphi \quad (7.6)$$

Inserting equation (20) into the integral gives:

$$b_1 = \frac{4 \cdot x_s}{\pi} \quad (7.7)$$

Thus the descriptive function of dry friction (two-point behavior) is:

$$N(\hat{x}_e) = \frac{4 \cdot x_s}{\pi \cdot \hat{x}_e} \quad (7.8)$$

The corresponding locus (Fig. 7.2) of the description function runs on the positive real axis of $0 \dots \infty$ for $x_s / \hat{x}_e = 0 \dots \infty$.

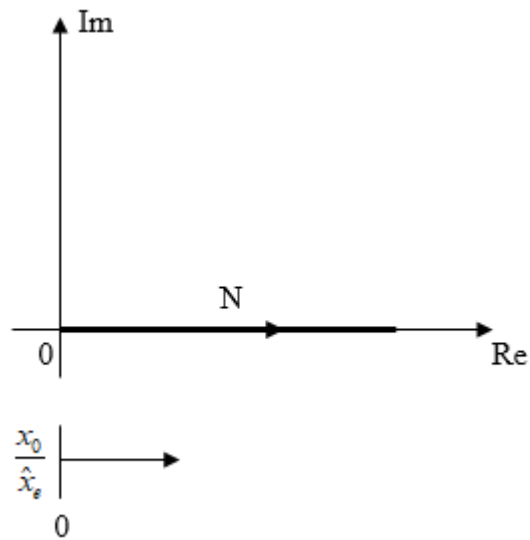


Fig. 7.2 - Locus curve for the descriptive function of dry friction (two-point behavior)

A signal flow diagram shown in Fig. 7.3 is also considered. This is the signal flow diagram for the cascade control of the speed of a DC drive. Here the current control loop is set according to the optimum amount and simulated by a PT-1 element. The speed control loop is set according to the symmetrical optimum. With the help of this structure, the influence of dry friction on the drive speed can be investigated. All parameters for this system and for the further considered signal flow plans are given in table 7.1.

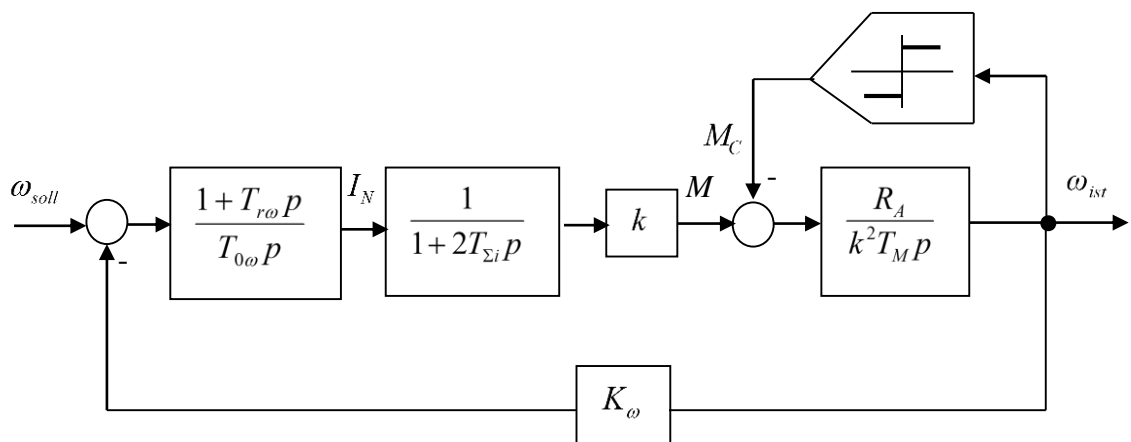


Fig. 7.3 - System of speed control of a direct current drive with dry friction

Table 7.1 - Parameters of the electric drive system

Name of the parameter	Parameter	Size
Time constant in the counter of the TF of the PI controller	$T_{r\omega}$	0.008 s
Time constant in the denominator of the TF of the PI controller	$T_{0\omega}$	0.0037 s
Equivalent time constant of the route	$T_{\Sigma i}$	0.001 s
Torque constant	k	0.28 V · s
Connection resistance	R_A	0.65 Ω
Mechanical starting time constant	T_M	0.02 s
Rated current of the motor	I_N	14 A
Rated speed	ω_n	251.2 s ⁻¹
Gain factor in the speed feedback	K_ω	1
Coefficient of sliding friction The moment of sliding friction: $M_C = b_C \cdot \text{sign}(\omega)$	b_C	0.5 N · m
Rolling friction coefficient Moment of rolling friction: $M_N = b_N \cdot \omega$	b_N	0.002 $\frac{N \cdot m}{s^{-1}}$

After simulating the system with and without dry friction, the following transition processes are obtained for the speed:

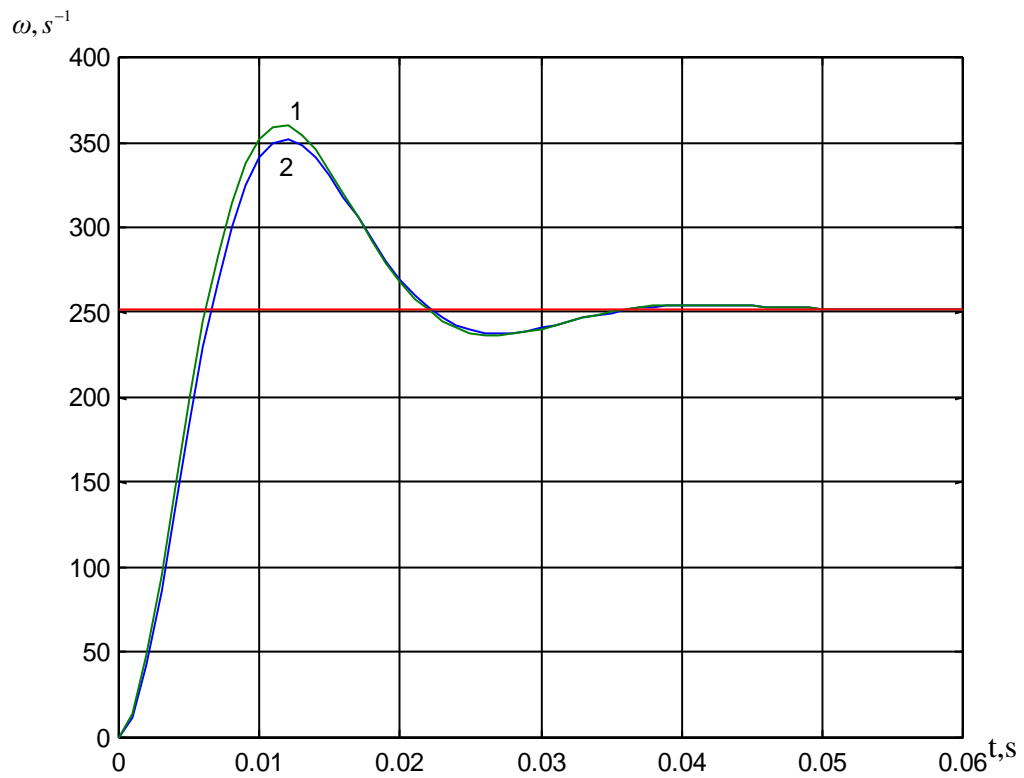


Fig. 7.4 - Step responses of the control loop. Curve 1 system without dry friction. Curve 2- system with dry friction (friction is greatly increased)

From Fig. 7.4 it can be seen that the overshoot in the system with dry friction is smaller than the overshoot in the system without dry friction.

8. MODELING AND SIMULATION OF DRY AND VISCOSE FRICTION

The static characteristic of dry and viscous friction is shown in Fig. 8.1. Internal viscous friction, which is proportional to the deformation speed of the shafts, cables, couplings, etc., has a major influence on the dynamic processes in the mechanical system.

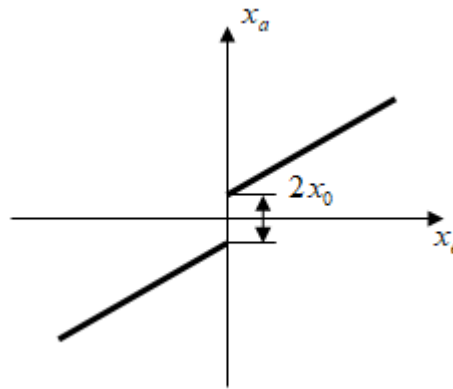


Fig. 8.1 - Characteristic curve of dry and viscous friction

The output variable can be seen in Fig. 16:

$$x_a = \begin{cases} x_0 + \hat{x}_e \cdot \sin \varphi & \text{for } x_e > 0 \\ x_0 + \hat{x}_e \cdot \sin \varphi & \text{for } x_e < 0 \end{cases} \quad (8.1)$$

Here the Fourier coefficient becomes $a_1 = 0$. For b_1 results:

$$b_1 = \frac{1}{\pi} \cdot \int_0^{\pi} x_a(\varphi) \cdot \sin \varphi d\varphi + \frac{1}{\pi} \cdot \int_{\pi}^{2\pi} x_a(\varphi) \cdot \sin \varphi d\varphi \quad (8.2)$$

ACIC DEPARTMENT				NAU 21 01 31 000 EN			
Performed	Bilai Y.S.			Automatic DC motor control system with PI-speed neuroregulator	N	Page	Pages all
Supervisor	Pantyeyev R. L.						
Normcontrol	Tupitsyn N. F.				431 151		
Dep. head	Sineglazov V. M.						

If you put the equation in the integrals, you get after calculation:

$$b_1 = \frac{4 \cdot x_0}{\pi \cdot \hat{x}_e} + \hat{x}_e \quad (8.3)$$

Thus, the descriptive function is dry and viscous friction:

$$N(\hat{x}_e) = \frac{4 \cdot x_0}{\pi \cdot \hat{x}_e} + 1 \quad (8.4)$$

The corresponding locus of the description function is shown in Fig. 8.2:

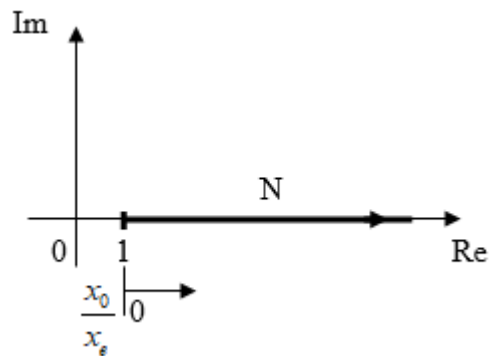


Fig. 8.2 - Locus curve for the descriptive function of dry and viscous friction

The locus of the description function runs on the positive real axis from 1 to ∞ or plotted over the quotient $n = x_0/\hat{x}_e$ from 0 to ∞ .

The signal flow diagram shown in Figure 8.3 is used to investigate the influence of dry and viscous friction on the transition processes of the speed. All parameters for this system are given in table 7.1.

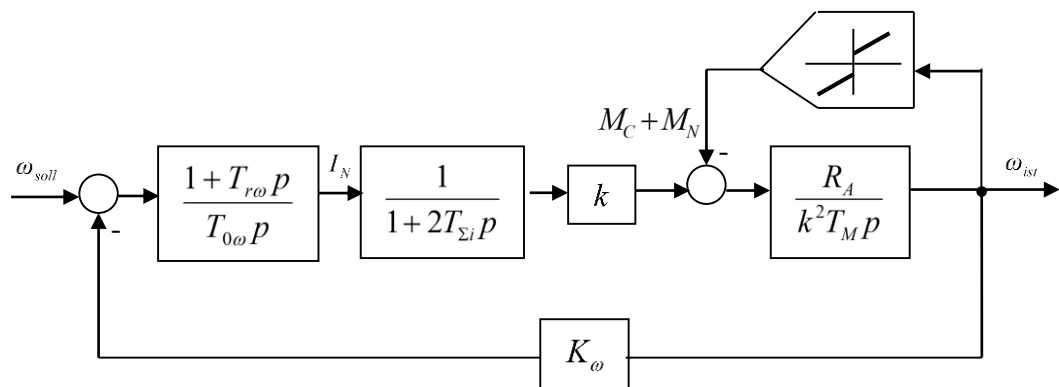


Fig. 8.3 - System of speed control of a direct current drive with dry and viscous friction

After simulating the system with and without dry and viscous friction, the following transition processes are obtained for the speed:

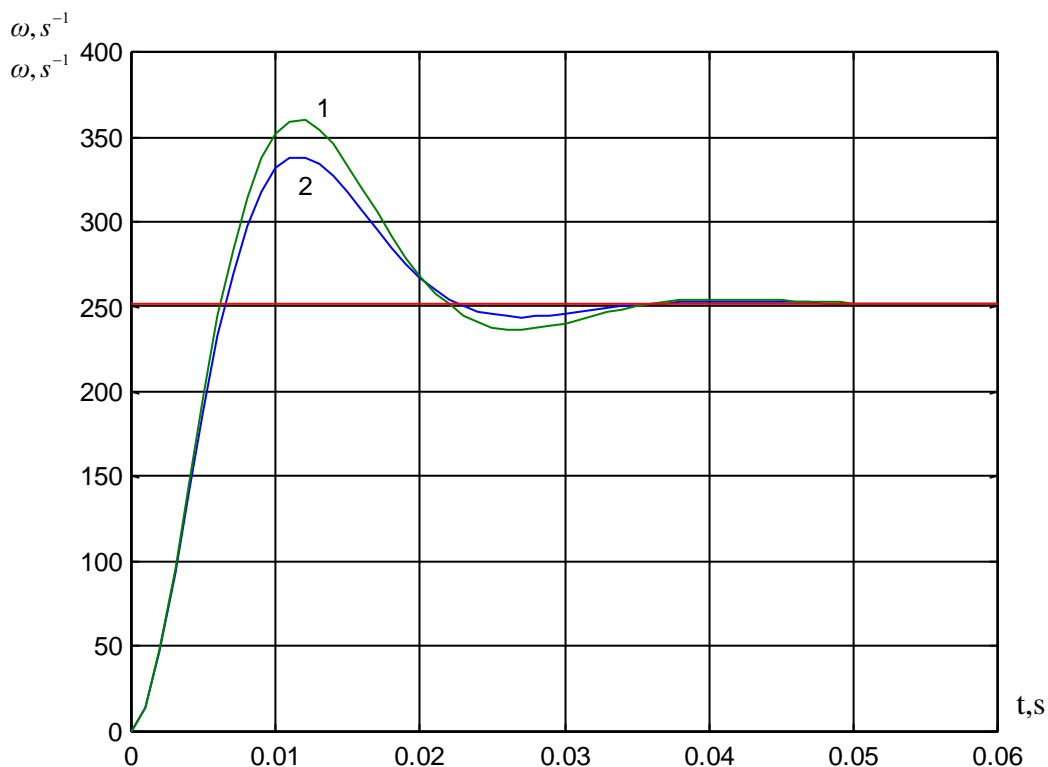


Fig. 8.4 - Step responses of the system. Curve 1 system without friction. Curve 2- system with dry and viscous friction (proportion of viscous friction is increased)

9. USE OF NEURAL NETWORKS TO COMPENSATE THE DRY FRICTION

In order to compensate for the influence of dry friction in the system, which is shown in Fig. 9.1, “Direct inverse control” is used in this work. But first an inverse model of the system should be built.

To get the inverse model, the following scheme is used:

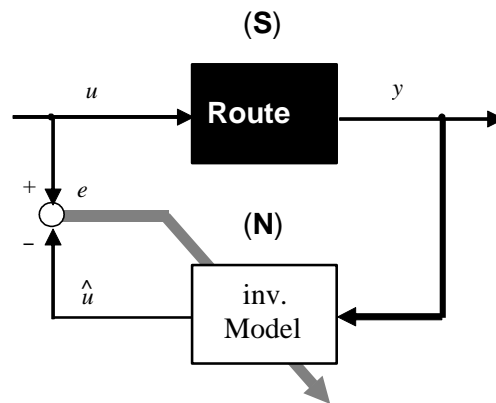


Figure 9.1 - Principle of the formation of an inverse model

The following relationship is used in the formation of the inverse NARX model:

$$\hat{u}(k) = N(y(k+1), \dots, y(k-n_y+1), u(k-1), \dots, u(k-n_u+1)) \quad (9.1)$$

The signal flow diagram, which illustrates the method of direct inverse control in, for example, $n_u = 2$ and $n_y = 2$ is shown in Fig. 9.2:

ACIC DEPARTMENT				NAU 21 01 31 000 EN			
Performed	Bilai Y.S.			<i>Automatic DC motor control system with PI-speed neuroregulator</i>	N	Page	Pages all
Supervisor	Panteyev R. L.						
Normcontrol	Tupitsyn N. F.				431 151		
Dep. head	Sineglazov V. M.						

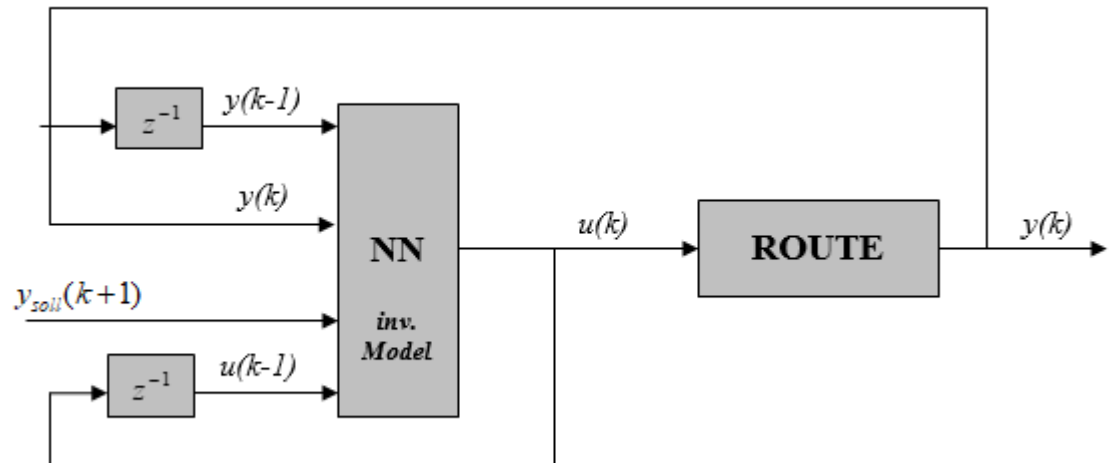


Fig. 9.2 - Direct inverse control

When using this method, the following equation is used in the control phase:

$$\hat{u}(k) = N(y_{soll}(k+1), y(k), \dots, y(k-n_y+1), \hat{u}(k-1), \dots, \hat{u}(k-n_u+1)) \quad (9.2)$$

In the application, the setpoint $y(k+1)$ is entered instead of the value $y_{soll}(k+1)$. The transfer and function of the series connection of the inverse model and the line theoretically tends to be one. In this way, non-linearities present in the system can be compensated [15].

In our case, the controlled system is part of the system shown in Fig. 7.3, from I_N to ω_{ist} . It's also $n_i = n_{\omega_{ist}} = 3$. The program for training the network is given in Appendix B and all commands used in this program are clarified in Appendix A (experiment instructions for laboratory work).

The system with the neurocontroller is shown in Figure 9.3.

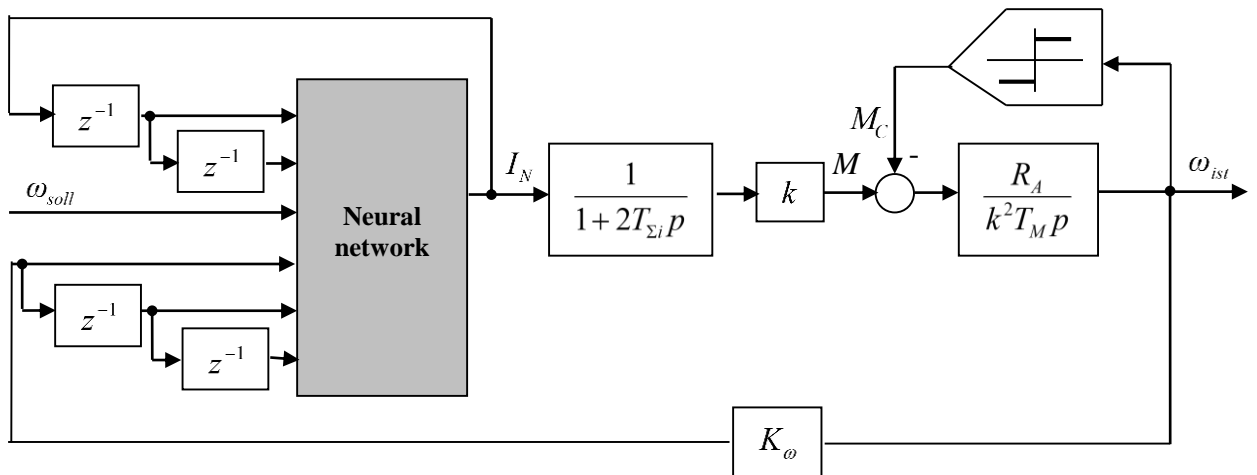


Fig. 9.3 - System with dry friction and the neurocontroller

After the simulation of this system, the transition processes for the speed are obtained, which are given in Figure 9.4. This picture shows the work of the system with the neurocontroller from a setpoint generator. It can be seen that the setpoint and the actual value of the speed almost overlap and the non-linearity is compensated. From Fig. 9.5 it becomes clear that the actual speed value is delayed by one discretization step (discretization step = 0.001 s). The error of the actual speed value compared to the speed setpoint is approximately 0.02%. It corresponds to the direct inverse control.

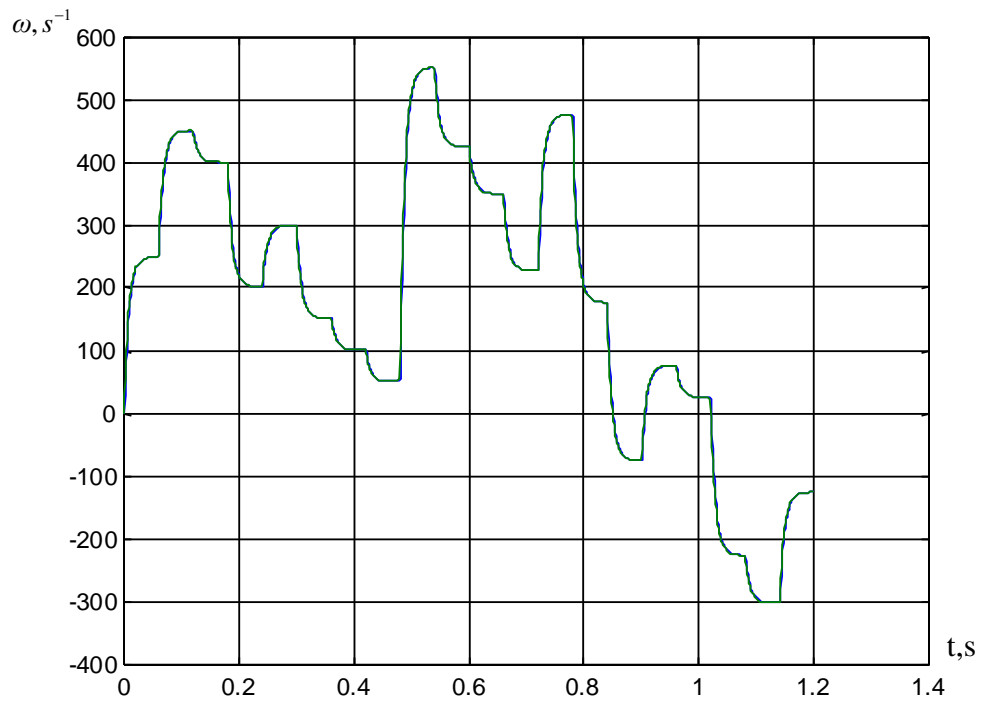


Fig. 9.4 - Work of the system with the neurocontroller from a setpoint generator

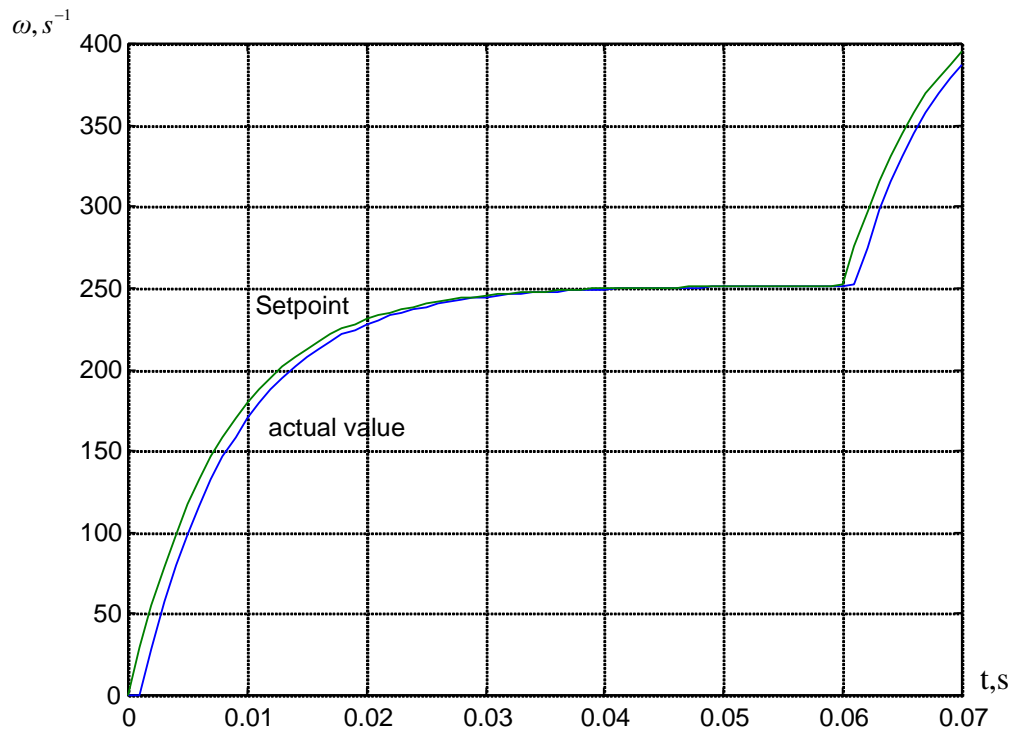


Fig. 9.5 - Enlarged part of the transition processes when the system works with the neurocontroller from a setpoint generator (discretization step = 0.001 s)

The associated motor current and actual speed value curves are shown in Fig.

9.6.

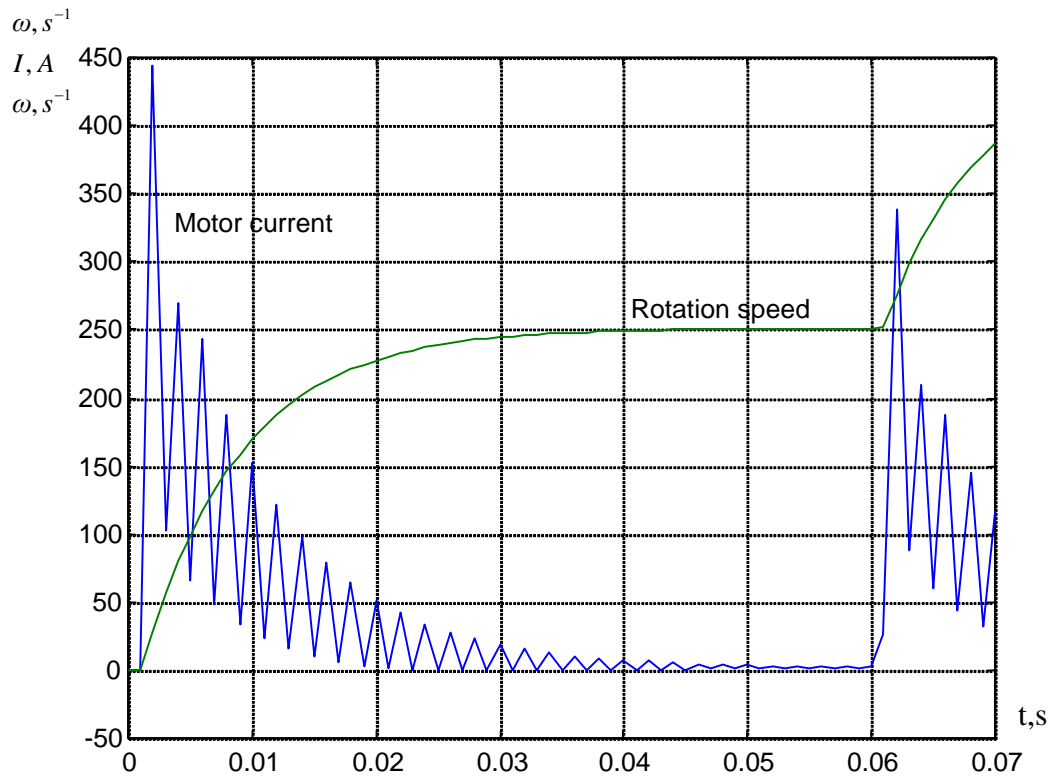


Fig. 9.6 - Motor current and actual speed value curve

10. USE OF NEURAL NETWORKS TO COMPENSATE DRY AND VISCOSE FRICTION

In order to compensate for the influence of dry and viscous friction in the system, which is shown in Fig. 10.1, direct inverse control is also used in this case. First, an inverse model of the controlled system is also built. A part of the system from I_N to ω_{ist} is taken as the controlled system. It's also $n_i = n_{\omega_{ist}} = 3$. The program for training the network is given in Appendix B and all commands used in this program are clarified in Appendix A (experiment instructions for laboratory work).

The system with the neurocontroller is shown in Fig. 10.1.

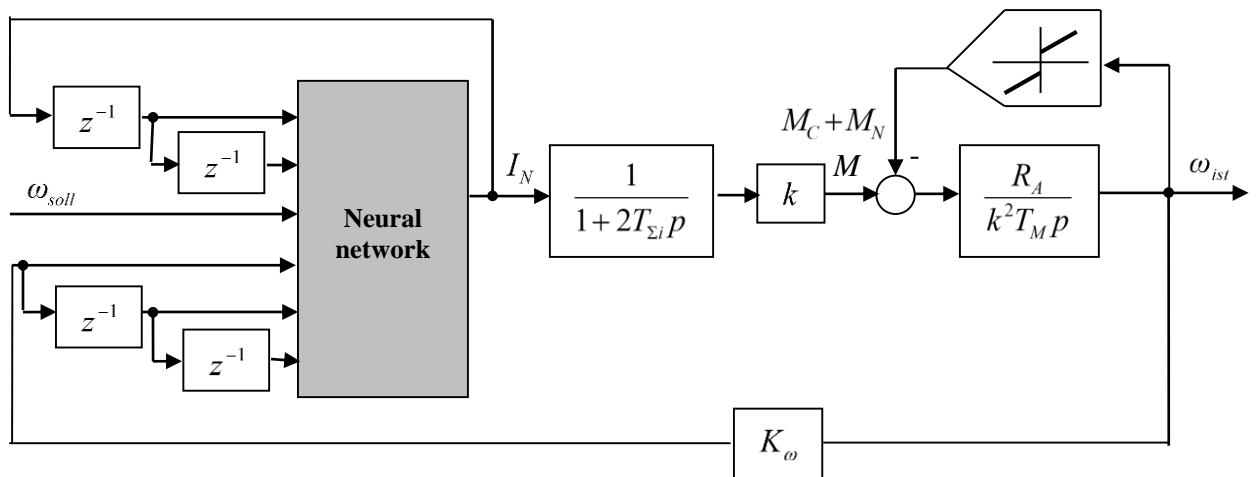


Fig. 10.1 - System with dry and viscous friction and the neurocontroller

ACIC DEPARTMENT				NAU 21 01 31 000 EN			
Performed	Bilai Y.S.			Automatic DC motor control system with PI-speed neuroregulator	N	Page	Pages all
Supervisor	Panteyev R. L.						
Normcontrol	Tupitsyn N. F.				431 151		
Dep. head	Sineglazov V. M.						

After the simulation of this system, the transition processes for the speed are obtained, which are given in Fig. 10.2. This picture shows the work of the system with the neurocontroller from a setpoint generator. It can be seen that in this case the setpoint and the actual value of the speed almost overlap and the non-linearity is also compensated. From Figure 10.3 it becomes clear that the actual speed value is delayed by one discretization step (discretization step = 0.001 s). The error of the actual speed value compared to the speed setpoint is approximately 0.02%. It corresponds to the direct inverse control.

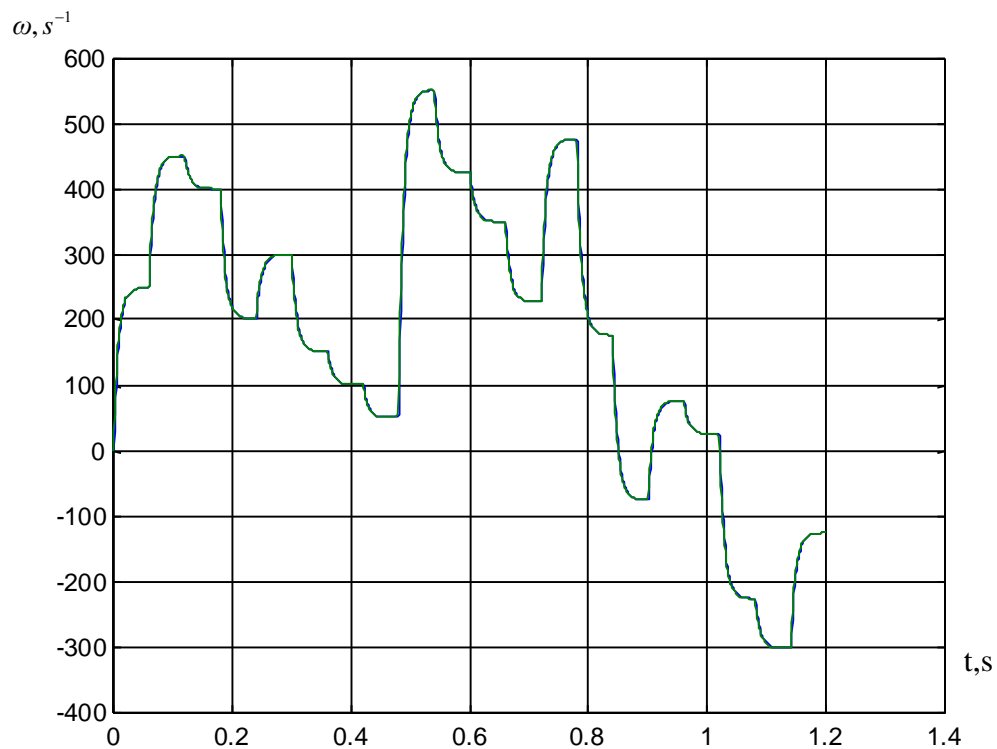


Fig. 10.2 - Work of the system with the neurocontroller from a setpoint generator

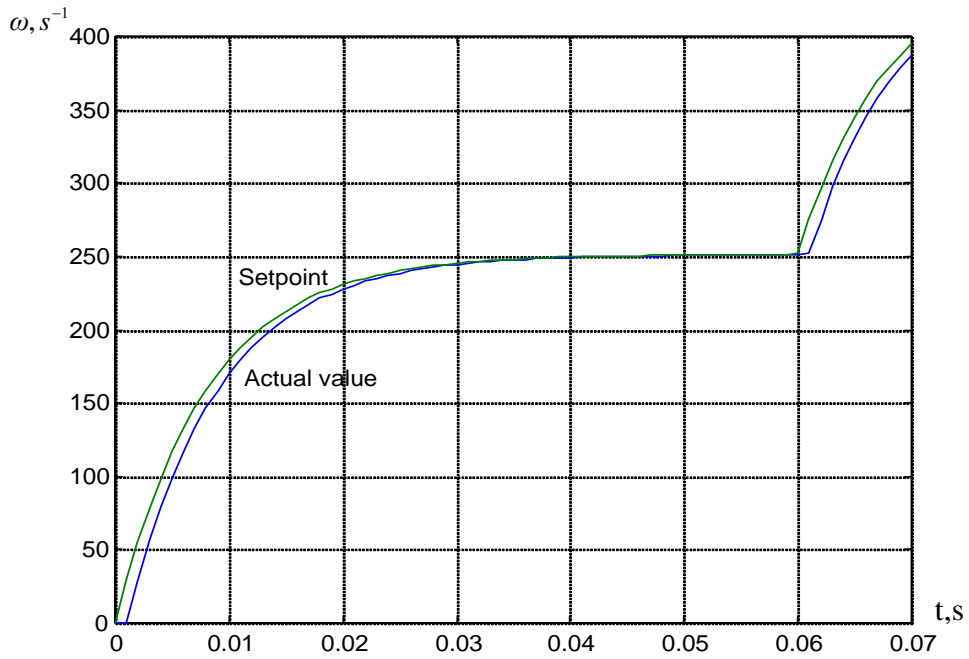


Fig. 10.3 - Enlarged part of the transition processes when the system works with the neurocontroller from a setpoint generator (discretization step = 0.001 s)

The associated motor current and actual speed value curves are shown in Figure 10.4.

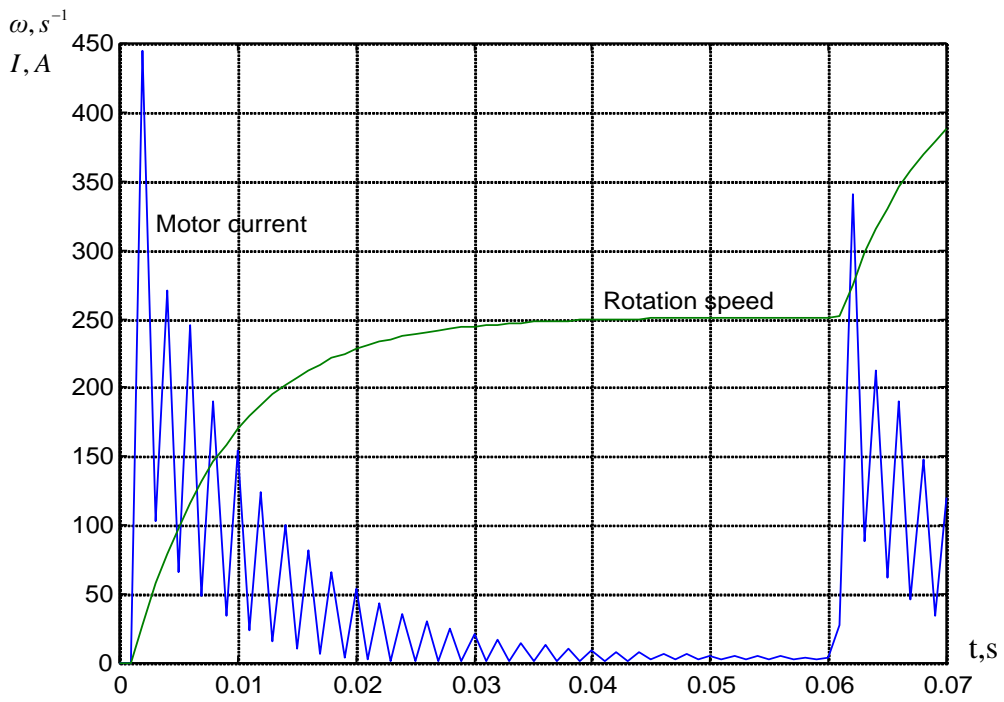


Fig. 10.4 - Motor current and actual speed value curve

Conclusions

In this thesis, two drive systems with the cascade control of the speed are considered. Dry friction is noted in one of these systems and dry and viscous friction is noted in the other. An inverse neuro-model was built on for the controlled system of each system. Then the obtained inverse neuro-model is built into the system according to the method “Direct inverse control” and used as the speed controller. The transfer and function of the series connection of the inverse model and the segment tends to be one. This compensates for non-linearities that exist in the system.

The method of mathematical modeling with a computer and the programming package MatLab 5.2 / Simulink, which is oriented towards the modeling of automated electric drive systems, was used to investigate the drive systems.

The modeling of the speed control systems carried out with the neurocontroller has shown the effectiveness of direct inverse control in non-linear electric drive systems.

The obtained non-linear electric drive systems with the neurocontroller, which compensates for the non-linearities in the systems, follow the setpoint with the static error 0.02% and are delayed by one discretization step. These results correspond to the method of direct inverse control.

On the basis of this work, an internship experiment has been developed to examine the performance of neurocontrollers in non-linear electric drive systems, which can be used in the learning process at the technical faculty.

References

1. Zakharian, S .; Ladewig-Riebler, P .: Controller setting with artificial neural networks, symposium “Modern methods of regulation and control design”, Otto von Guericke University Magdeburg, 2007.
2. Wasserman P. Neural Computing – Theory and Practice. Van Nostrand Reinhold, 1998
3. Scherer, A .: Neural Networks. Basics and Applications, Vieweg Verlag, 2017.
4. Palis, F .; Schmied, Th .; Buch, A .: Fuzzy and Neurocontrol of Mechanically Coupled Drive Systems. 2nd Magdeburg Mechanical Engineering Days, September 14th and 15th, 2005. Article No. 6 (4 pages).
5. Заенцев И. Нейронный сети: основные модели. Учебное пособие. Воронежский государственный университет, 1999.
6. Bishop C.M. Neural Networks for Pattern Recognition. Oxford University Press Inc., 2003.
7. Zakharian, S .: Neural networks for engineers: work and exercise book for control engineering applications, Vieweg Verlag, 2008.
8. Калашников В.И .; Палис Ф .; Денисенко И.В .: Теория нейросетей: учебное пособие, Донецк, Магдебург, ДонГТУ, 1997.
9. Калашников В.И .; Палис Ф .: Введение в интеллектуальные системы программирования: учебное пособие, Киев, ИСМО, 1997.
10. Pinkus A. Approximation theory of the MLP model in neural networks. Acta Numerica, 1999.
11. Jung, D .: Controller setting with neural networks, diploma thesis, FB MND, FH Wiesbaden, 2017.
12. Bavarian, B .: Introduction to Neural Networks for Intelligent Control, IEEE Control Systems Magazine, 4 (2008).
13. Haykin S. Neural Networks: A Comprehensive Foundation. NY: Macmillan, 1994.

14. Старостин С.С .: Методические указания по оформлению текстовой документации, Донецк, ДонГТУ, 1997.
15. Zurada J.M. Introduction To Artificial Neural Systems. Boston: PWS Publishing Company, 1992
16. Vogel, J .: Electric Drive Technology, 6th, completely revised. Ed., Heidelberg: Hüthig, 2018.
17. Галушкин А.И. Нейрокомпьютеры. Кн. 3. М.: ИПРЖ, 2000.
18. DARPA Neural Network Study, AFCEA International Press, 1998.

APPENDIX A.

Network training program (dry friction system)

```
i1= [0; i(1:length(i)-1)];
i2= [0; i1(1:length(i1)-1)];
w1= [w(2:length(w));w(length(w))];
w2= [0; w(1:length(w)-1)];
w3= [0; w2(1:length(w2)-1)];
P=[i1';i2';w1';w';w2';w3'];
size(P)
T=i';
size(T)
net=newff([-2.1327e+003 2.1763e+003;
-2.1327e+003 2.1763e+003;
-953.4054 661.0935;
-953.4054 661.0935;
-953.4054 661.0935;
-953.4054 661.0935],[1],{'purelin'});
net.trainParam.goal=0.01;
net.trainParam.epochs=1000;
[net,tr]=train(net,P,T);
P=[i1';i2';w1';w';w2';w3'];
y2=sim(net,P);
plot(t,y2,t,i);
grid on;
zoom on;
```

APPENDIX B.

Network training program (dry and viscous friction system)

```
i1= [0; i(1:length(i)-1)];
i2= [0; i1(1:length(i1)-1)];
w1= [w(2:length(w));w(length(w))];
w2= [0; w(1:length(w)-1)];
w3= [0; w2(1:length(w2)-1)];
P=[i1';i2';w1';w';w2';w3'];
size(P)
T=i';
size(T)
net=newff([-2.1296e+003 2.1692e+003;
-2.1296e+003 2.1692e+003;
-301.2456 552.1529;
-301.2456 552.1529;
-301.2456 552.1529;
-301.2456 552.1529],[1],{'purelin'});
net.trainParam.goal=0.0001;
net.trainParam.epochs=1000;
[net,tr]=train(net,P,T);
P=[i1';i2';w1';w';w2';w3'];
y2=sim(net,P);
plot(t,y2,t,i);
grid on;
zoom on;
```