Σ mathematics

Special Issue Reprint

# Applied and Computational Mathematics for Digital Environments, 2nd Edition

Edited by
Liliya Demidova

mdpi.com/journal/mathematics

MDPI

# Applied and Computational Mathematics for Digital Environments, 2nd Edition

# Applied and Computational Mathematics for Digital Environments, 2nd Edition

Editor

**Liliya Demidova**

MDPI

Basel • Beijing • Wuhan • Barcelona • Belgrade • Novi Sad • Cluj • Manchester

*Editor*
Liliya Demidova
Institute of Informational
Technologies
MIREA—Russian Technological
University
Moscow
Russia

*Editorial Office*
MDPI AG
Grosspeteranlage 5
4052 Basel, Switzerland

This is a reprint of articles from the Special Issue published online in the open access journal *Mathematics* (ISSN 2227-7390) (available at: https://mdpi.com/si/160500).

For citation purposes, cite each article independently as indicated on the article page online and as indicated below:

| Lastname, A.A.; Lastname, B.B. Article Title. *Journal Name* **Year**, *Volume Number*, Page Range. |
|---|

**ISBN 978-3-7258-2269-0 (Hbk)**
**ISBN 978-3-7258-2270-6 (PDF)**
**doi.org/10.3390/books978-3-7258-2270-6**

© 2024 by the authors. Articles in this book are Open Access and distributed under the Creative Commons Attribution (CC BY) license. The book as a whole is distributed by MDPI under the terms and conditions of the Creative Commons Attribution-NonCommercial-NoDerivs (CC BY-NC-ND) license.

# Contents

# About the Editor

**Liliya A. Demidova**

Liliya A. Demidova is a Professor at the MIREA–Russian Technological University (Moscow, Russia). She is the author of 7 books and over 200 papers published in reputable journals in Russian and English. Her research interests include fuzzy logic, machine learning, data mining, decision support, time series forecasting, and image processing.

# Preface

Digitalization and digital transformation are currently affecting many areas of human activity which include but are not limited to industry, medicine, economics, education, and ecology. This rapid expansion poses new challenges for researchers whose goal is to develop principles and technologies of applied and computational mathematics that would serve as guidelines for studying such phenomena.

This reprint is constructed as follows. It comprises 10 papers covering various topics on the usage of applied and computational mathematics in practical problems in digital environments. These papers were published in Volumes 11 (2023) and 12 (2024) of the MDPI Mathematics journal, and were accepted for publication in the Special Issue "Applied and Computational Mathematics for Digital Environments, 2nd Edition".

The topics of interest include, among others, scientific research, applied tasks, and problems in the following areas:

- Building mathematical, structural, and information models of intelligent computer systems for monitoring and managing the parameters of digital environments;
- Software and mathematical technologies in the implementation of the intelligent monitoring and computer control of digital environments' parameters;
- Th application of mathematical models, Internet of Things technologies, machine learning, and artificial intelligence for big data analysis of digital environments;
- Mathematical models and algorithms for identifying stable patterns between the parameters of digital environments and their complex and separate influence;
- Mathematical modeling, machine learning, and their implementation within the concept of "smart" digital environments.

This reprint may be of interest to researchers involved in finding solutions to actual practical problems in digital environments through the use of applied and computational mathematics.

As the Guest Editor of the Special Issue "Applied and Computational Mathematics for Digital Environments, 2nd Edition" of the MDPI Mathematics journal, I am grateful to all contributing authors. I also express my gratitude to all of the reviewers for their valuable comments, which made it possible to improve the quality of the submitted papers, and to the administrative staff of MDPI publications for their support in completing this project. Special thanks are due to Dr. Syna Mu, Journal Development Editor, for his excellent collaboration, valuable assistance, and support.

**Liliya Demidova**
*Editor*

mathematics

*Article*

# Universal Stabilisation System for Control Object Motion along the Optimal Trajectory

**Askhat Diveev [†] and Elena Sofronova [*,†]**

Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, 44, build. 2, Vavilova Str., Moscow 119333, Russia; aidiveev@mail.ru
* Correspondence: sofronova_ea@mail.ru
† These authors contributed equally to this work.

**Abstract:** An attempt to construct a universal stabilisation system that ensures the object motion along specified trajectory from certain class is presented. If such a stabilisation system is constructed, then only the problem of optimal control is solved, but for a model of the object, which includes a stabilisation system and a subsystem with a reference model for generating a specified trajectory. In this case, the desired control is the control in the reference model. Statement of complete optimal control problem includes two problems, optimal control problem and stabilisation system synthesis problem for motion along given trajectory in the state space. Numerical methods for solving these problems based on evolutionary computation and symbolic regression are described. It is shown that when solving the stabilisation system synthesis problem, it is possible to obtain a universal system that provides stabilisation of the object motion relative to any trajectory from a certain class. Therefore, it is advisable to formulate an optimal control problem for an object with a motion stabilisation system. A computational example of solving the problem for the spatial motion of a quadrocopter is given.

**Keywords:** optimal control; control synthesis; stabilisation system; evolutionary algorithm; symbolic regression

**MSC:** 49M25; 68W50

**Citation:** Diveev, A.; Sofronova, E. Universal Stabilisation System for Control Object Motion along the Optimal Trajectory. *Mathematics* **2023**, *11*, 3556. https://doi.org/10.3390/math11163556

Academic Editor: Liliya Demidova

Received: 15 July 2023
Revised: 1 August 2023
Accepted: 14 August 2023
Published: 17 August 2023

**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

## 1. Introduction

When solving the optimal control problem in the classical Pontryagin statement [1], a solution is obtained in the form of a time-dependent control function. Such a solution cannot be directly implemented in the real control object, since it leads to the construction of an open-loop control system that is not sensitive to the real current position of the control object. The control system without feedback is sensitive to small perturbations. The control object with an open-loop control system cannot reach the terminal state with a given accuracy and provide the optimum value of the given quality criterion. Therefore, in most cases, the optimal control problem is considered as an initial problem for obtaining an optimal program control and an optimal program trajectory. In order to implement the obtained solution in a real control object, it is necessary to further solve the problem of stabilisation system synthesis for the control object motion along the obtained optimal program trajectory.

Two things should be noted here. First, solving the control synthesis problem is often no less complicated than solving the original optimal control problem, because in the synthesis problem, the control function is sought as a function of the state vector. Second, solving the control synthesis problem changes the dynamics of the control object. The mathematical model of the control object in the form of differential equations contains the control function in the right side, so the optimal control found for the original mathematical model may no longer be optimal for the mathematical model of the object with a motion stabilisation system. In any case, a system that provides feedback control is necessary to implement the solution of the optimal control problem, although this does not follow from the most classical statement of the optimal control problem.

The statement of the control problem, where it is necessary to find the control function of the state vector, was also proposed by R. Bellman a long time ago [2]. To solve this problem, a dynamic programming method was developed that allows numerically find the value of the control vector for each value of the state vector. If we solve the problem in the Bellman statement using the dynamic programming method for an initial state, as in the Pontryagin statement, the resulting solution will also be sensitive to changes in the initial conditions, and the dependence of the control function on the state vector will not provide an adequate response to disturbances. Solving the Bellman optimal control problem by dynamic programming for a set of initial states faces the problem of the curse of dimensionality.

In [3], a refined statement of the optimal control problem is formulated. The classical statement is complemented by the requirement that the resulting optimal trajectory has a non-empty neighbourhood with attractor properties. It means that the resulting optimal control should ideally be a special solution of the differential equation, an attractor. To achieve this, the control is first sought as a function of time and state, and to implement the stabilisation system, the initial state is replaced by the initial state domain.

Ensuring additional requirements can be obtained in various ways, for example, by reformulating the optimal control problem into a problem of general control synthesis for a given area of initial states [3]; then, each particular solution from a given area provides the optimal value of a given quality criterion. Such a task is computationally difficult. Another approach is to solve the control synthesis problem in order to ensure stability with respect to the terminal state, but in this case, we do not guarantee that the optimal value of the given quality criterion can be obtained.

To stabilise the movement of the control object along the optimal trajectory, theoretical works [4] propose linearising the model relative to the trajectory and obtaining a linear non-stationary model of the object. For the stability of such an object, linear feedback is proposed. In general, stability for a non-stationary object is not an unsolved problem. In most practical works, points are set on the tracked trajectory and the control object is made stable relative to these points. PI and PID controllers are used for this purpose [5–8]. Movement on stable trajectory points slows down the movement of the object near the stability point, so the optimum value of the criterion is not maintained. In the work [9], a system for tracking the trajectory of a quadcopter is built based on stabilization of the quadcopter speed in the horizontal plane along a straight line. For this purpose, a proportional regulator is built based on the Lyapunov function. Movement along the trajectory of straight segments with a constant speed is not optimal. It is necessary to track the trajectory not only in space but also in time.

In [10], an approach to solving the optimal control problem by the synthesized control method is considered. According to this approach, first the synthesis problem to ensure the stability of the object relative to the equilibrium point in the state space is solved, and the optimal control problem in the original statement is solved at the second stage, where the optimal positions of the stable equilibrium points are found. Optimal control is achieved by changing the positions of stable equilibrium points after a given time interval. Synthesised control is a universal approach to solving the optimal control problem in the class of feasible systems, but in any particular case it may have several solutions. Each of the solutions may be differently sensitive to disturbances of the initial conditions. The practical advantage of the synthesized control is that the synthesis problem of ensuring the stability of the equilibrium point is solved at a preliminary stage, i.e., at the stage of creating a control system, and the solution of the optimal control problem by choosing the position of the equilibrium points can be solved on the on-board computer for a specific current situation. Solving the problem of synthesising a control system on an on-board computer is usually complicated due to the high computational cost.

The problem of synthesis of motion stabilisation along the optimal trajectory was first considered in [11], where the method of symbolic regression was applied to solve the synthesis problem. Before that, the symbolic regression method was used to solve the problem of general control synthesis, without solving the optimal control problem. As a result, we obtain a control system that includes a reference model for generating the optimal trajectory in time. Studies have shown that the stabilisation system depends on the type of optimal trajectory. In practice, it means that the use of such a control system is difficult, that is, when the situation changes and a new optimal control problem arises, it is necessary not only to solve the optimal control problem, but to re-solve the stabilisation system synthesis problem, which is unacceptable for the on-board computer.

In contrast to the previous works, in this paper, it is proposed to use a universal stabilisation system. The universality of the stabilisation system is that one system of motion stabilisation for a particular object is obtained for different types of trajectories. The obtained stabilisation system can be used for other trajectories as well. There may be trajectories for which this stabilisation system is not suitable, but this requires additional research, which is currently underway. The stabilisation system should ensure the stabilisation of control object motion along the given optimal trajectories from a certain class. For this purpose, first the synthesis problem is solved for one stabilisation system for several given trajectories. The class of trajectories is considered as a training set, and the stabilisation system synthesis is the learning of control system for a given training set. Next, this stabilisation system is applied to the motion along the trajectory that was not included in the training set. To solve the stabilisation system synthesis problem, methods of symbolic regression are applied [12,13].

The rest of the paper is organized as follows. The statements of optimal control problems and the synthesis of a motion stabilisation system along a given trajectory, as well as a new optimal control problem for an object that includes a reference model and a motion stabilisation system relative to the trajectory obtained using the reference model in the control system, are presented in Section 2. Such stabilisation system is called universal. Next, the network operator method, one of methods of symbolic regression, is described in Section 3. An example of the universal stabilisation system synthesis for the spatial movement of a quadcopter and the use of this system when the quadcopter is moving along a complex trajectory is given in Section 4. Computational experiments are followed by a conclusion in Section 5 and a discussion in Section 6, respectively.

## 2. Statement of Complete Optimal Control Problem

We consider some statements of the optimal control problem that make it complete in terms of implementing of the control problem solution in a real object. To implement a solution, it is necessary to obtain a closed-loop control system, so that a found control function depends on the state space vector.

The classical statement of the optimal control problem does not allow obtaining a closed-loop control system, so a found control function depends only on time. In the following, a classical statement of the optimal control problem is considered.

### 2.1. Optimal Control Problem

The mathematical model of a control object is given in the form of an ordinary differential equation system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \tag{1}$$

where $\mathbf{x}$ is a state vector of control object, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} = [x_1 \dots x_n]^T$, $\mathbf{u}$ is a control vector, $\mathbf{u} \in \mathrm{U} \subseteq \mathbb{R}^m$, U is a compact set that defines control constraints. For example, values of control vector components can be bounded from above and below

$$\mathbf{u}^- \leqslant \mathbf{u} \leqslant \mathbf{u}^+, \tag{2}$$

$\mathbf{u}^- = [u_1^- \dots u_m^-]^T$, $\mathbf{u} = [u_1 \dots u_m]^T$, $\mathbf{u}^+ = [u_1^+ \dots u_m^+]^T$.

For System (1), the initial state is given:

$$\mathbf{x}(0) = \mathbf{x}^0. \tag{3}$$

The terminal state is given as

$$\mathbf{x}(t_f) = \mathbf{x}^f, \tag{4}$$

where $t_f$ is a terminal time of achievement of the terminal state. The terminal time is not specified, but it is limited; $t_f \leqslant t^+$, $t^+$ is a given time limit.

The quality criterion is given in the common integral form

$$J_0 = \int_0^{t_f} f_0(\mathbf{x}, \mathbf{u})dt \to \min_{\mathbf{u}\in \mathrm{U}}. \tag{5}$$

When solving the problem by direct numerical approach, the terminal state (4) is reached with a certain accuracy, which is included in the quality criterion (5). Therefore, the quality criterion for the numerical solution of the problem has the following form:

$$J_1 = \int_0^{t_f} f_0(\mathbf{x}, \mathbf{u})dt + p_1\|\mathbf{x}^f - \mathbf{x}(t_f)\| \to \min_{\mathbf{u}\in \mathrm{U}}, \tag{6}$$

where $p_1$ is a weight coefficient;

$$t_f = \begin{cases} t, \text{ if } t < t^+ \text{and } \|\mathbf{x}^f - \mathbf{x}(t)\| \leqslant \varepsilon_1 \\ t^+, \text{ otherwise} \end{cases}, \tag{7}$$

where $\varepsilon_1$ is an accuracy of achievement of the terminal state (4).

In the classical optimal control problem, a control function is sought as a function of time:

$$\mathbf{u} = \mathbf{v}(t) \in \mathrm{U}. \tag{8}$$

To implement the solution of the optimal control problem, it is necessary to synthesise the stabilisation system of a motion along the obtained optimal trajectory. This stabilisation system should change the mathematical model of the control object in such a way that the optimal trajectory in the state space acquires a non-zero neighbourhood with attractor properties. In the statement of the optimal control problem, either these requirements should be included and then the implementation of this property is conducted at the discretion of the control system designer, or it is necessary to add a statement of the stabilisation system synthesis problem of the motion along the optimal trajectory, and then solve these two problems together sequentially.

Thus, the control synthesis problem for stabilising the motion along the optimal trajectory is described as follows.

### 2.2. Stabilisation System Synthesis

The same mathematical model of control object as in the optimal control problem (1) is used.

Instead of the one initial state (3), a domain of initial states is specified. For a numerical solution, the initial state domain is given in the form of a finite set of points

$$\mathrm{X}_0 = \{\mathbf{x}^{0,1}, \ldots, \mathbf{x}^{0,K}\}. \tag{9}$$

The terminal state is given (4).

The optimal trajectory is a time function

$$\mathbf{x}^*(t),\ t \in (0; t_f). \tag{10}$$

It is necessary to find the optimal control as a function of the deviation of the state space vector from the optimal trajectory

$$\mathbf{u} = \mathbf{h}(\mathbf{x}^* - \mathbf{x}). \tag{11}$$

If the control function (11) is placed in the right part of ODE system (1), then the following system is obtained:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{h}(\mathbf{x}^* - \mathbf{x})). \tag{12}$$

The control function (11) should minimize the sum of maximum deviations of all particular solutions of the system (12) from initial states of the given domain (9)

$$J_2 = \sum_{i=1}^{K} \left( \max_{t \in (0; t_{f,i})} \|\mathbf{x}^* - \mathbf{x}(t, \mathbf{x}^{0,i})\| + p_1 \|\mathbf{x}^f - \mathbf{x}(t_{f,i})\| \right) \to \min_{\mathbf{u} \in \mathrm{U}}, \tag{13}$$

where $t_{f,i}$ is a time of achievement the terminal state of the particular solution from the initial state $\mathbf{x}^{0,i}$ defined by Equation (7), $\mathbf{x}(t, \mathbf{x}^{0,i})$ is a particular solution of the system (12) from the initial state $\mathbf{x}^{0,i}$.

There is an ambiguity in this problem statement: How to obtain the value of the optimal trajectory (10) at a given time? When solving the optimal control problem in the first stage, the optimal control function is sought as a function of time, but not an optimal trajectory. When searching by a direct approach, the optimal control function is usually approximated by a piece-wise continuous function. The result is an analytical mathematical expression for the control function. The optimal trajectory in the general case has no mathematical expression and it is obtained numerically after simulation of the control object model (1) with the optimal control function. To obtain the optimal trajectory and use it to solve the stabilisation system synthesis problem in the second stage, the results of the simulation of the control object model (1) with the optimal control function can be kept as an array of time points and values of the state space vector at this moment. Another way is to simulate the control object model (1) with the optimal control function together with the control object model used for the synthesis of the stabilisation system. Then, instead of the control object model (1) and the optimal trajectory (10) in the statement of the stabilisation system synthesis problem, a model with two subsystems is used:

$$\begin{array}{rcl} \dot{\mathbf{x}} & = & \mathbf{f}(\mathbf{x}, \mathbf{u}), \\ \dot{\mathbf{x}}^* & = & \mathbf{f}(\mathbf{x}^*, \mathbf{u}^*). \end{array} \tag{14}$$

The first subsystem is the mathematical model of the control object with a sought-after control function for stabilising the movement relative to the optimal trajectory. The second subsystem is the reference model that generates the optimal trajectory.

It should be noted that the stabilisation system changes the dynamics of the control object and the optimal control for the control object without a stabilisation system can be non-optimal for the control object with a stabilisation system. In order to solve the synthesis problem of the stabilisation system, a machine learning control by symbolic regression is used. The stabilisation system in a common case cannot provide stabilisation for any trajectory, and therefore it should be synthesised for each new optimal trajectory.

### 2.3. Optimal Control Problem for Object with Motion Stabilisation System

In this work, a universal stabilisation system synthesis problem is considered. This universal stabilisation system provides motion of a control object along any trajectory from some class. To solve the universal stabilisation system synthesis problem, machine learning control by symbolic regression is performed for the same model of the control object and for some given trajectories at the same time. Suppose such a universal stabilisation system is obtained. If it is known that an optimal trajectory belongs to the class of trajectories stabilised by the universal stabilisation system, then the optimal control problem can be solved for the control object with a stabilisation system.

The problem statement of the optimal control problem for a control object with a universal stabilisation system of movement along the trajectory has the following description.

The mathematical model of the control object with a universal stabilisation system and with a reference model for the generation of the program trajectory is given as

$$\begin{array}{rcl}\dot{\mathbf{x}} & = & \mathbf{f}(\mathbf{x}, \mathbf{h}(\mathbf{x}^* - \mathbf{x})), \\ \dot{\mathbf{x}}^* & = & \mathbf{f}(\mathbf{x}^*, \mathbf{u}),\end{array} \tag{15}$$

where the function of stabilisation system $\mathbf{h}(\mathbf{x}^* - \mathbf{x})$ satisfies the constraints on control for any values of its arguments

$$\mathbf{h}(\mathbf{x}^* - \mathbf{x}) \in \mathrm{U}. \tag{16}$$

The initial state is given in (3). The terminal state is given in (4). The quality criterion is as follows:

$$J_3 = \int_0^{t_f} f_0(\mathbf{x}, \mathbf{h}(\mathbf{x}^* - \mathbf{x}))dt + p_1 \|\mathbf{x}^f - \mathbf{x}(t_f)\| \to \min_{\mathbf{u} \in \mathrm{U}}. \tag{17}$$

It is necessary to find a control function as a time function

$$\mathbf{u} = \mathbf{v}^*(t) \in \mathrm{U}, \tag{18}$$

with the constraint (16) that the particular solution of the system (15) from the given initial state (3) reaches the given terminal state (4) with the optimal value of the given criterion (17).

In this problem, a control function is sought as a function of time and a closed-loop system with feedback control is obtained.

## 3. Symbolic Regression for Solving the Control Synthesis Problem

Symbolic regression is a unique computational technique that allows finding the mathematical expressions of the desired functions. Note that artificial neural networks can also approximate any function, but they do not find the structure of the function. The structure of an artificial neural network is determined by the type of network. It can vary regularly over a certain range by changing the number of layers and the number of neurons in each layer.

The universal approximation of functions by an artificial neural network is provided by a large number of parameters. The determination of parameter values as a result of network training allows obtaining the required values of the desired function for the entire set of specified values of its arguments. The technique of using an artificial neural network is similar to the technique of digging a hole using only a shovel. Obviously, any hole can be dug with a shovel if a large number of workers is used; in the neural network, these are parameters. But why is it impossible to use more advanced mechanisms, such as excavators, in approximation problems? These are nonlinear transformations. They should not be used just because they are difficult to use and require qualification.

Note that nonlinear effects are characteristic of physical and natural phenomena. Many models of physical processes are nonlinear. Only for nonlinear differential equations it is possible to obtain a stable limit cycle or an attractor property for a manifold of non-zero dimension.

There are problems that are difficult to solve with an artificial neural network. One example is the control synthesis problem. In this problem, it is necessary to find a function that is part of the mathematical model of the control object. Each new function changes the dynamic properties of the object, so we cannot define the behaviour of the object with the optimal control function in advance since the form of the desired function is unknown at the time of the search. The lack of a training example complicates the application of an artificial neural network. The function should be searched only by the value of the quality criterion. Paper [14] explicitly states that the future of artificial intelligence is not connected with incomprehensible neural networks, but with quite clear symbolic regression.

Here, the symbolic regression method is used to solve the control synthesis problem. All symbolic regression methods encode mathematical expressions in the form of special codes. To encode a mathematical expression, first, the alphabet of elementary functions is defined. The search for a mathematical expression is performed by a special genetic algorithm on the space of codes of mathematical expressions. In a special genetic algorithm, the crossover operation is performed taking into account the code of the mathematical expression so that after the crossover operation, two correct codes of new mathematical expressions are obtained.

The best known symbolic regression method is genetic programming (GP) [15]. GP encodes a mathematical expression in the form of a computational tree. On the leaves of the tree are the arguments of the mathematical expression. Each node of the tree represents an elementary function. The number of branches leaving the node is equal to the number of arguments of the elementary function. The crossover operation in GP involves randomly selecting nodes in the parent trees and swapping the subtrees originating from those nodes. GP is not the most convenient method of symbolic regression because after the crossover operation the codes of mathematical expressions change length and the number of identical arguments of a mathematical expression should be equal to the number of occurrences of that argument in the desired mathematical expression. GP has also been applied to solve control problems [16,17].

There are now about twenty symbolic regression methods. In this paper, one of them, the network operator method (NOP) [12], is used to solve the control synthesis problem. NOP uses only functions with one or two arguments in the alphabet of elementary functions. It encodes a mathematical expression in the form of a directed graph. Source nodes of the graph are associated with the arguments of the mathematical expression. Other nodes of the graph are associated with the functions of two arguments. The edges of the graph are associated with functions of one argument.

We consider an example of encoding a mathematical expression by the network operator method. A mathematical expression is given:

$$y = a\exp(-bx_1)(\sin(cx_2) + \cos(dx_2)), \tag{19}$$

where $a$, $b$, $c$, $d$ are constant parameters, $x_1$, $x_2$ are variables. Parameters and variables are arguments of mathematical expression (19).

To encode the mathematical expression, we use the following functions:

(1) functions with one argument:

$$\begin{aligned} \mathrm{F}_1 \quad = \quad & \{f_{1,1}(z) = z, f_{1,2}(z) = -z, f_{1,3}(z) = \exp(z), \\ & f_{1,4}(z) = \sin(z), f_{1,5}(z) = \cos(z)\}; \end{aligned} \tag{20}$$

(2) functions with two arguments:

$$\mathrm{F}_2 = \{f_{2,1}(z_1, z_2) = z_1 + z_2, f_{2,2}(z_1, z_2) = z_1 \cdot z_2\}. \tag{21}$$

Functions with two arguments should be commutative, associative and have a unit element,

$$f_{2,i}(e_i, z) = f_{2,i}(z, e_i) = z,$$

where $e_i$ is a unit element of function $f_{2,i}(z_1, z_2)$.

Figure 1 shows the directed graph of NOP for mathematical expression (19).

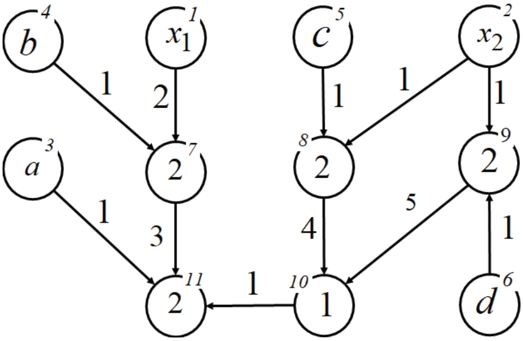


**Figure 1.** The network operator graph for the mathematical expression.

In the network operator graph, the arguments of the mathematical expression are shown in the source node. The numbers of functions with two arguments are shown in other nodes. Numbers of functions with one argument are displayed next to the edges. The nodes are indexed in their upper parts. If the node indices are sorted such that the index of node where the edge comes out is of a lesser value than that of the index of nodes where the edge comes in, then the network operator matrix is upper triangular.

In PC memory, the network operator is presented as an integer matrix, that has a structure like that of the network operator graph adjacency matrix. The network operator matrix for the graph in Figure 1 has the following form:

$$\mathbf{\Psi} = [\psi_{i,j}] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}, \ i, j = 1, \ldots, L = 11, \tag{22}$$

where $L$ is a number of nodes in the network operator graph, $\dim(\mathbf{\Psi}) = L \times L$.

In the network operator matrix, lines with zeros on the main diagonal are linked with source-nodes. Other non-zero elements on the main diagonal are the function numbers with two arguments. Non-zero elements above the main diagonal $\psi_{i,j} \neq 0$ are function numbers with one argument.

For calculation values of the mathematical expression by network operator, a vector of nodes is defined. Initially, the vector of nodes consists of mathematical expression arguments and unit elements of the corresponding functions with two arguments.

For considered mathematical expression, the initial vector of nodes has the following form:

$$\mathbf{z}^{(0)} = [x_1 \ x_2 \ a \ b \ c \ d \ 1 \ 1 \ 1 \ 0 \ 1]^T, \tag{23}$$

where 1 is a unit element for function of multiplication $f_{2,2}(z_1, z_2) = z_1 \cdot z_2$, 0 is a unit element for function of summary $f_{2,1}(z_1, z_2) = z_1 + z_2$.

The calculation of mathematical expression is performed by the following equation:

$$z_j^{(i)} \leftarrow \begin{cases} f_{2,\psi_{j,j}}(z_j^{(i-1)}, f_{1,\psi_{i,j}}(z_i^{i-1})), \text{ if } \psi_{i,j} \neq 0 \\ z_j^{(i-1)}, \text{ otherwise} \end{cases}, \ i = 1, \ldots, L-1, \ j = i+1, \ldots, L. \tag{24}$$

To find an optimal mathematical expression by the network operator method, the variation genetic algorithm is used. This algorithm implements the principle small variations of basic solution [18]. According to this principle, small variations of code are defined and only one basic solution is encoded. Other possible solutions are coded as sets of small variations. A code of small variation is an integer vector of four components,

$$\mathbf{w} = [w_1\ w_2\ w_3\ w_4]^T, \tag{25}$$

where $w_1$ is a type of small variation, $w_2$ is the line number, $w_3$ is the column number, $w_3 \geq w_2$, $w_4$ is a new value of a network operator matrix element.

In the network operator, four types of small variations are used: $w_1 = 0$ is an exchange of the function with one argument: if $\psi_{w_2,w_3} \neq 0$, then $\psi_{w_2,w_3} \leftarrow w_4$; $w_1 = 1$ is an exchange of the function with two arguments: if $\psi_{w_2,w_2} \neq 0$, then $\psi_{w_2,w_2} \leftarrow w_4$; $w_1 = 2$ is an insertion of the additional function with one argument: if $\psi_{w_2,w_3} = 0$, then $\psi_{w_2,w_3} \leftarrow w_4$; $w_1 = 3$ is an elimination of the function with one argument: if $\psi_{w_2,w_3} \neq 0$ and $\exists \psi_{w_2,j} \neq 0$, $j > w_2$, $j \neq w_3$ and $\exists \psi_{i,w_3} \neq 0$, $i \neq w_2$, then $\psi_{w_2,w_3} \leftarrow 0$.

In the search algorithm, a population of possible solutions is used. Any possible solution other than the basic solution is encoded as a set of small variation vectors

$$\mathrm{W}_i = \{\mathbf{w}^{1,1}, \ldots, \mathbf{w}^{i,d}\},\ i = 1, \ldots, H, \tag{26}$$

where $d$ is a depth of variation, a parameter of the search algorithm, $H$ is a number of possible solutions in the population.

Any possible solution $\mathbf{\Psi}_i$ is obtained by small variations of the basic solution $\mathbf{\Psi}_0$. Variation vector is an operator changing of the network operator matrix. Therefore, for any possible solution, one can write the following equation:

$$\mathbf{\Psi}_i = \mathrm{W}_i \circ \mathbf{\Psi}_0 = \mathbf{w}^{i,d} \circ \mathbf{w}^{i,d-1} \circ \cdots \circ \mathbf{w}^{i,1} \circ \mathbf{\Psi}_0. \tag{27}$$

To perform a crossover operation, two possible solutions are selected randomly:

$$\begin{aligned} \mathrm{W}_\alpha &= \{\mathbf{w}^{\alpha,1}, \ldots, \mathbf{w}^{\alpha,d}\}, \\ \mathrm{W}_\beta &= \{\mathbf{w}^{\beta,1}, \ldots, \mathbf{w}^{\beta,d}\}. \end{aligned} \tag{28}$$

The crossover point is selected randomly, $c \in \{1, \ldots, d\}$. Two new possible solutions are obtained by the exchange of tails after the crossover point of selected possible solutions

$$\begin{aligned} \mathrm{W}_{H+1} &= \{\mathbf{w}^{\alpha,1}, \ldots, \mathbf{w}^{\alpha,c-1}, \mathbf{w}^{\beta,c}, \ldots, \mathbf{w}^{\beta,d}\}, \\ \mathrm{W}_{H+2} &= \{\mathbf{w}^{\beta,1}, \ldots, \mathbf{w}^{\beta,c-1}, \mathbf{w}^{\alpha,c}, \ldots, \mathbf{w}^{\alpha,d}\}. \end{aligned} \tag{29}$$

## 4. Computation Experiment

Consider the optimal control problem of the spatial motion of a quadcopter. The mathematical model of the control object is

$$\begin{aligned} \dot{x}_1 &= x_4, \\ \dot{x}_2 &= x_5, \\ \dot{x}_3 &= x_6, \\ \dot{x}_4 &= u_4(\sin(u_3)\cos(u_2)\cos(u_1) + \sin(u_1)\sin(u_2)), \\ \dot{x}_5 &= u_4\cos(u_3)\cos(u_1) - g, \\ \dot{x}_6 &= u_4(\cos(u_2)\sin(u_1) - \cos(u_1)\sin(u_2)\sin(u_3)), \end{aligned} \tag{30}$$

where $g = 9.80665$.

For a given model of the control object, it is necessary to build a system for stabilising the motion along a given spatial trajectory, where the shape of the trajectory is not known in advance. For this purpose, we first create a universal stabilisation system. Solving the optimal control problem, we obtain several different trajectories, and then solve the stabilisation system synthesis problem for all the trajectories.

### 4.1. Synthesis of Universal Stabilisation System

Usually, the development of a system for stabilising the movement of an object along a given trajectory [19] is to study the mathematical model of the control object, determine the control channels, determine the deviation of the object from the trajectory or from the nearest point [7] located on the trajectory and inserting the regulator into the control channel for qualitative compensation of the deviation. Sometimes model predictive controls with a simplified model of the control object are used to quickly determine the deviation [20]. In this work, an attempt is made to develop a universal system for stabilising the movement along a given trajectory based on machine learning control by symbol regression. In this approach, the analytical study of the mathematical model of the control object is entrusted to the computer, which itself finds the necessary control channels and inserts the necessary regulators there.

To generate program trajectories, we formulate an optimal control problem for a given object (30).

Initial and terminal states are given as

$$\mathbf{x}(0) = \mathbf{x}^0 = [0\ 5\ 0\ 0\ 0\ 0]^T, \tag{31}$$

$$\mathbf{x}(t_f) = \mathbf{x}^f = [10\ 5\ 10\ 0\ 0\ 0]^T, \tag{32}$$

where $t_f$ is a time of achievement of the terminal state, $t_f$ is not given, but limited, $t_f \leqslant t^+ = 5.6$.

Phase constraints are included in the quality criterion,

$$\varphi_i(\mathbf{x}) = r_i - \sqrt{(x_{1,i} - x_1)^2 + (x_{3,i} - x_3)^2} \leqslant 0,\ i = 1, \dots, M, \tag{33}$$

where $M$ is a number of obstacles, $M = 2$, $r_i$ is a radius of obstacle $i$, $r_1 = 2$, $r_2 = 2$, $(x_{1,i}, x_{3,i})$ are coordinates of their centers, $x_{1,1} = 2.5$, $x_{3,1} = 2.5$, $x_{1,2} = 7.5$, $x_{3,2} = 7.5$.

The control object should move through some specified areas. Changing the position of the these areas affects the optimal trajectory shape. This condition is also included in the quality criterion,

$$\delta_i^{(k)}(\mathbf{x}(t)) = \min_{0 \leqslant t \leqslant t_f} \left\{ \sqrt{(z_{1,i}^{(k)} - x_1(t))^2 + (z_{3,i}^{(k)} - x_3(t))^2} - d_i^{(k)} \right\} \leqslant 0, \tag{34}$$

where $(z_{1,i}^{(k)}, z_{3.i}^{(k)})$ are coordinates of the area centers on the horizontal plane, $i = 1, \dots, S$, $k = 1, \dots, P$, $d_i^{(k)}$ is a size of area, $S$ is a number of areas, $S = 4$, $d_i^{(k)} = 0.6$, $P$ is a number of optimal control problems, $P = 4$, $k$ is a current optimal control problem.

The quality criterion is

$$J_4^{(k)} = t_f + p_1 \|\mathbf{x}^f - \mathbf{x}(t_f)\| + p_2 \sum_{i=1}^{M} \int_0^{t_f} \vartheta(\varphi_i(\mathbf{x}))dt + p_3 \sum_{j=1}^{S} \vartheta(\delta^{(k)}(\mathbf{x}(t))) \to \min_{\mathbf{u} \in \mathrm{U}}, \tag{35}$$

where $p_1 = 2$, $p_2 = 3$, $p_3 = 3$, $\vartheta(\alpha)$ is the Heaviside step function

$$\vartheta(\alpha) = \begin{cases} 1, \text{if } \alpha > 0 \\ 0, \text{otherwise} \end{cases}. \tag{36}$$

To obtain a variety of trajectories, we consider the conditions for the object to pass through specified areas when solving the optimal control problem. Various optimal trajectories were obtained as a result of different locations of the specified areas.

The training set contained four trajectories in defined locations in the required areas.

The coordinates of required area centers are

$$\begin{gathered}
z_{1,1}^{(1)} = 2.5, z_{3,1}^{(1)} = 0.4, z_{1,2}^{(1)} = 5.0, z_{3,2}^{(1)} = 2.0, z_{1,3}^{(1)} = 7.5, z_{3,3}^{(1)} = 4.5, z_{1,4}^{(1)} = 9.6, z_{3,4}^{(1)} = 7.5, \\
z_{1,1}^{(2)} = 2.5, z_{3,1}^{(2)} = 0.4, z_{1,2}^{(2)} = 4.5, z_{3,2}^{(2)} = 2.5, z_{1,3}^{(2)} = 5.5, z_{3,3}^{(2)} = 7.5, z_{1,4}^{(2)} = 7.5, z_{3,4}^{(2)} = 9.6, \\
z_{1,1}^{(3)} = 0.0, z_{3,1}^{(3)} = 2.0, z_{1,2}^{(3)} = 2.0, z_{3,2}^{(3)} = 5.0, z_{1,3}^{(3)} = 5.0, z_{3,3}^{(3)} = 8.0, z_{1,4}^{(3)} = 8.0, z_{3,4}^{(3)} = 10.0, \\
z_{1,1}^{(4)} = 0.4, z_{3,1}^{(4)} = 2.5, z_{1,2}^{(4)} = 2.5, z_{3,2}^{(4)} = 4.5, z_{1,3}^{(4)} = 7.5, z_{3,3}^{(4)} = 5.5, z_{1,4}^{(4)} = 9.6, z_{3,4}^{(4)} = 7.5.
\end{gathered}$$

When solving optimal control problems, we used a direct approach. For this purpose, we divided the time axis into equal intervals and looked for the values of constant parameters at the boundaries of the intervals. Taking into account the control constraints, the piecewise linear approximation of the control function has the following form:

$$u_i^{(k)} = \begin{cases} u_j^+, \text{ if } \hat{u}_j^k > u_j^+ \\ u_j^-, \text{ if } \hat{u}_j^k < u_j^- \\ \hat{u}_j^k, \text{otherwise} \end{cases} \tag{37}$$

where

$$\hat{u}_j^{(k)} = q_{j+mi}^{(k)} + (q_{j+m(i+1)}^{(k)} - q_{j+mi}^{(k)})\frac{t - i\Delta t}{\Delta t}, \tag{38}$$

$i = 1, \ldots, N$, $j = 1, \ldots, m$, $N$ is a number of time intervals, $\Delta t$ is a time interval,

$$N = \left\lfloor \frac{t^+}{\Delta t} \right\rfloor = \left\lfloor \frac{5.6}{0.4} \right\rfloor = 14. \tag{39}$$

For numerical solution a hybrid algorithm [21] was applied. A hybrid algorithm is based on three well-known algorithms: the genetic algorithm [22], particle swarm optimization [23], and the grey wolf optimizer [24].

The total $(N+1)m = 15 \times 4 = 60$, $\mathbf{q}^{(k)} = [q_1 \ldots q_{60}]^T$ parameters were found.

The solutions of the optimal control problem obtained by the hybrid evolutionary algorithm are given in the Data Availability section.

The parameters of hybrid evolutionary algorithm are the number of possible solutions in population—1024, the number of generations—512, the number of evolutionary transformations in each population—512. For GA, the number of bits for integer part—4, the number of bits for fructional part—12, the probability of mutation—0.75. For GWO, the number of leaders—4. For PSO, the parameters are $k_\alpha = 0.729$, $k_\beta = 0.85$, $k_\gamma = 0.15$, $k_\sigma = 1$, and the number of randomly selected solutions to choose the informant—4.

The projections of optimal trajectories on the horizontal plane of the four optimal control problems are shown in Figures 2–5. In the figures, red circles indicate obstacles that define phase constraints of the problem. Small dotted circles indicate the specified areas mandatory for trajectories to follow.

After obtaining the optimal program trajectories, the problem of synthesis of the object motion stabilisation system along these trajectories was solved (1)–(2), (9)–(13). It was necessary to find one stabilisation system for all four program trajectories from the training set according to criterion (13). The domain of initial conditions (9) contained $K = 26$ vectors of initial states.

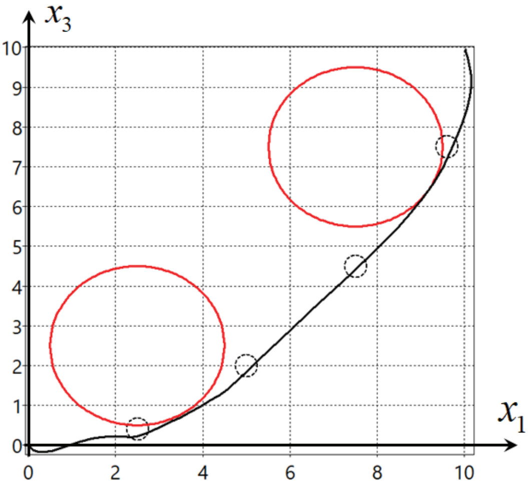


**Figure 2.** Projection of optimal trajectory 1 on the horizontal plane.

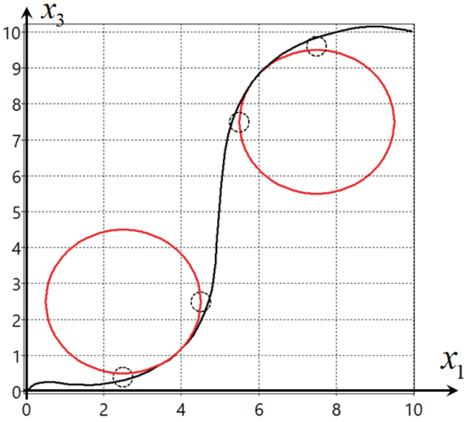


**Figure 3.** Projection of optimal trajectory 2 on the horizontal plane.

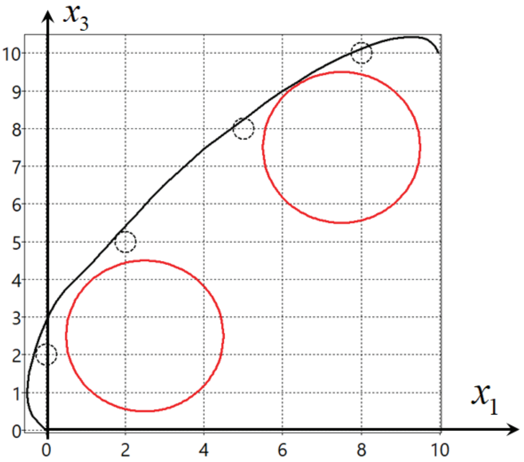


**Figure 4.** Projection of optimal trajectory 3 on the horizontal plane.

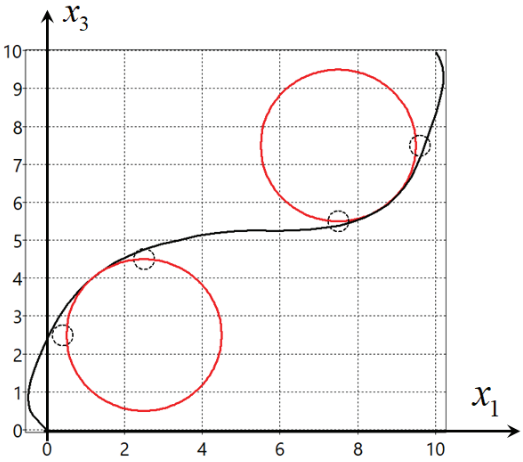


**Figure 5.** Projection of optimal trajectory 4 on the horizontal plane.

To solve the synthesis problem, the network operator method [12] implemented in the software package developed by the authors was used. The parameters of the algorithm were the number of chromosomes in the initial population—512; the number of generations—128; the number of couples in one generation—128; the dimension of network operator 36 × 36; the number of variations of one possible solution—5. The computational time for the universal stabilisation system synthesis on PC with CPU Intel Corei7 2.8 GHz for four trajectories was approximately 10 min.

The following solution was obtained:

$$u_i = \begin{cases} u_i^+, \text{if } \tilde{u}_i > u_i^+ \\ u_i^-, \text{if } \tilde{u}_i < u_i^- \\ \tilde{u}_i, \text{otherwise} \end{cases}, \tag{40}$$

where

$$\tilde{u}_1 = \mu(A) + \rho_{19}(q_2), \tag{41}$$

$$\tilde{u}_2 = (\tilde{u}_1 - \tilde{u}_1^3)\vartheta(q_6(x_6^* - x_6) + q_3(x_3^* - x_3)), \tag{42}$$

$$\tilde{u}_3 = \tilde{u}_2 + \rho_{17}(q_1 \arctan(x_1^* - x_1) + q_4(x_4^* - x_4)), \tag{43}$$

$$\begin{aligned} \tilde{u}_4 \quad = \quad & \tilde{u}_3^2 + \text{sgn}(B + \mu(A) + \rho_{19}(q_2))\sqrt{|B + \mu(A) + \rho_{19}(q_2)|} + \\ & \vartheta(C) + \sin(D) + \text{sgn}(-A\arctan(E)F) + \arctan(G) + \sqrt[3]{F} + \\ & \sin(q_4(x_4^* - x_4)) + \exp(q_2(x_2^* - x_2)) + \sqrt{q_1}, \end{aligned} \tag{44}$$

$$A = q_6(x_6^* - x_6) + q_3(x_3^* - x_3) + \vartheta(q_6(x_6^* - x_6)),$$

$$B = H + \tanh(D) + \exp(E) + \sqrt[3]{F} + \exp(q_5(x_5^* - x_5)),$$

$$C = D + \tanh(-A\arctan(E)F) + \rho_{18}(F),$$

$$D = -A\arctan(E)F + \sqrt[3]{G} + \text{sgn}(A) +$$

$$\sin(q_1 \arctan(x_1^* - x_1) + q_4(x_4^* - x_4)) + \cos(q_3(x_3^* - x_3)),$$

$$E = F^3 + q_1 \arctan(x_1^* - x_1) + q_4(x_4^* - x_4) + \vartheta(q_5(x_5^* - x_5)) +$$

$$\arctan(q_4) - (x_6^* - x_6) + (x_5^* - x_5)^2,$$

$$F = \sin(q_6(x_6^* - x_6)) + q_5(x_5^* - x_5) + (q_2+1)(x_2^* - x_2) + \cos(q_1) - (x_2^* - x_2)^3,$$

$$G = \rho_{19}(A) + E + \ln(|q_6(x_6^* - x_6) + q_3(x_3^* - x_3)|) + \exp(F) +$$
$$\vartheta(q_5(x_5^* - x_5)) + \operatorname{sgn}(x_5^* - x_5) + (x_2^* - x_2)^3,$$

$$H = \exp(C) + \cos(q_6(x_6^* - x_6)) + \operatorname{sgn}(C)\sqrt{|C|} + \operatorname{sgn}(D)\sqrt{|D|} + \sin(q_5(x_5^* - x_5)),$$

$$\vartheta(\alpha) = \begin{cases} 1, \text{ if } \alpha > 0 \\ 0, \text{ otherwise} \end{cases},$$

$$\mu(\alpha) = \begin{cases} \alpha, \text{ if } |\alpha| < 1 \\ \operatorname{sgn}(\alpha), \text{ otherwise} \end{cases},$$

$$\rho_{17}(\alpha) = \operatorname{sgn}(\alpha)\ln(|\alpha| + 1),$$

$$\rho_{18}(\alpha) = \operatorname{sgn}(\alpha)(\exp(|\alpha|) - 1),$$

$$\rho_{19}(\alpha) = \operatorname{sgn}(\alpha)\exp(-|\alpha|),$$

$q_1 = 12.10181$, $q_2 = 4.23291$, $q_3 = 15.55688$, $q_4 = 14.70337$, $q_5 = 7.75635$, $q_6 = 10.45923$.

Figures 6–9 show projections of one optimal trajectory (in blue) and eight perturbed trajectories (in black) from the domain of initial conditions (9). The figures show that the same stabilisation system (40)–(44) provides the object motion in the neighbourhood of all four optimal trajectories. It should be considered universal.

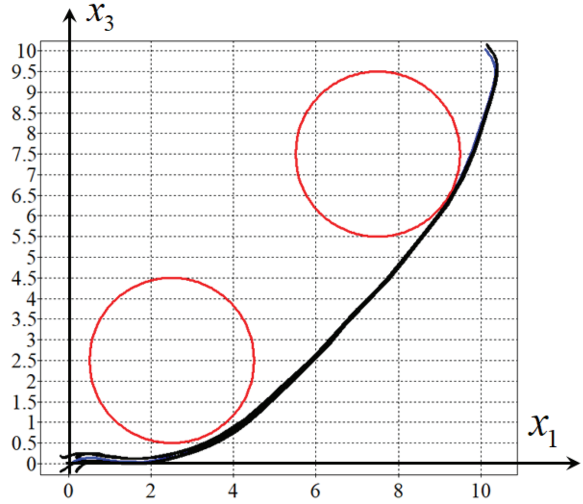


**Figure 6.** Projections of optimal trajectory 1 and perturbed trajectories from eight initial states on the horizontal plane.

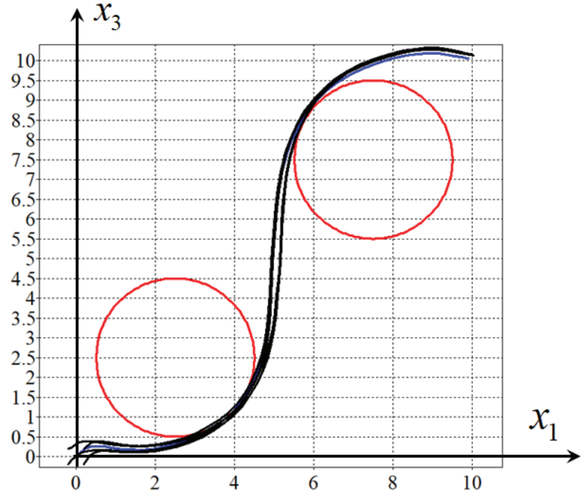


**Figure 7.** Projections of optimal trajectory 2 and perturbed trajectories from eight initial states on the horizontal plane.

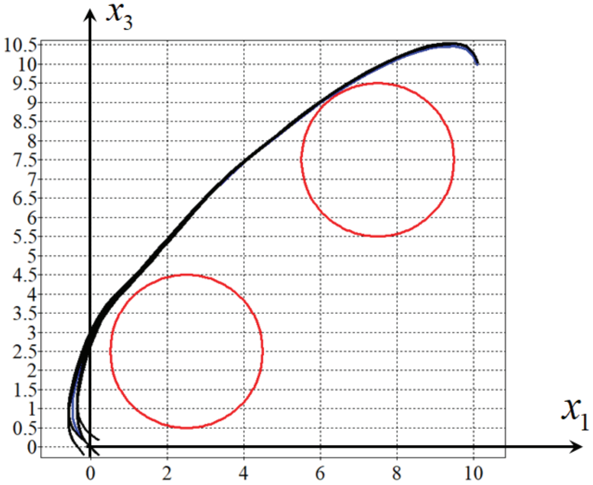


**Figure 8.** Projections of optimal trajectory 3 and perturbed trajectories from eight initial states on the horizontal plane.

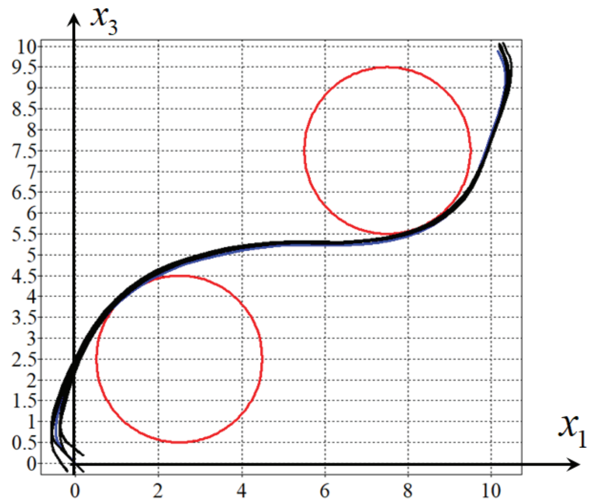


**Figure 9.** Projections of optimal trajectory 4 and perturbed trajectories from eight initial states on the horizontal plane.

### 4.2. Solution of Complete Optimal Control Problem for Object with Motion Stabilisation System

In the second computational experiment, we solved the complete optimal control problem for an object with a universal stabilisation system (40)–(44). The spatial motion of the quadrotor along a closed trajectory was considered. The mathematical model of the control object with a universal stabilisation system has the following form:

$$\begin{aligned}
\dot{x}_1 &= x_4,\\
\dot{x}_2 &= x_5,\\
\dot{x}_3 &= x_6,\\
\dot{x}_4 &= h_4(\mathbf{x}^*-\mathbf{x})(\sin(h_3(\mathbf{x}^*-\mathbf{x}))\cos(h_2(\mathbf{x}^*-\mathbf{x}))\cos(h_1(\mathbf{x}^*-\mathbf{x}))\\
&\quad+\sin(h_1(\mathbf{x}^*-\mathbf{x}))\sin(h_2(\mathbf{x}^*-\mathbf{x}))),\\
\dot{x}_5 &= h_4(\mathbf{x}^*-\mathbf{x})\cos(h_3(\mathbf{x}^*-\mathbf{x}))\cos(h_1(\mathbf{x}^*-\mathbf{x}))-g,\\
\dot{x}_6 &= h_4(\mathbf{x}^*-\mathbf{x})(\cos(h_2(\mathbf{x}^*-\mathbf{x}))\sin(h_1(\mathbf{x}^*-\mathbf{x}))\\
&\quad-\cos(h_1(\mathbf{x}^*-\mathbf{x}))\sin(h_2(\mathbf{x}^*-\mathbf{x}))\sin(h_3(\mathbf{x}^*-\mathbf{x}))),\\
\dot{x}_1^* &= x_4^*,\\
\dot{x}_2^* &= x_5^*,\\
\dot{x}_3^* &= x_6^*,\\
\dot{x}_4^* &= u_4(\sin(u_3)\cos(u_2)\cos(u_1)+\sin(u_1)\sin(u_2)),\\
\dot{x}_5^* &= u_4\cos(u_3)\cos(u_1)-g,\\
\dot{x}_6^* &= u_4(\cos(u_2)\sin(u_1)-\cos(u_1)\sin(u_2)\sin(u_3)),
\end{aligned} \tag{45}$$

where $\mathbf{x} = [x_1 \dots x_6]^T$, $\mathbf{x}^* = [x_1^* \dots x_6^*]^T$.

For the control object, the initial conditions coincide with the terminal conditions

$$\mathbf{x}^0 = \mathbf{x}^f = [0\ 5\ 0\ 0\ 0\ 0]^T. \tag{46}$$

Four phase constraints are

$$\varphi_i(\mathbf{x}) = r_i - \sqrt{(x_{1,i}-x_1)^2+(x_{3,i}-x_3)^2} \leqslant 0,\ i = 1,\dots,M, \tag{47}$$

where $M$ is the number of obstacles, $M = 4$, $r_i$ is the radius of obstacle $i$, $r_i = 2$, $i = 1,\dots,4$, $(x_{1,i}, x_{3,i})$ are coordinates of their centers, $x_{1,1} = 5$, $x_{3,1} = 0$, $x_{1,2} = 10$, $x_{3,2} = 5$, $x_{1,3} = 5$, $x_{3,3} = 10$, $x_{1,4} = 0$, $x_{3,4} = 5$.

The three specified areas are as follows:

$$\delta_i(\mathbf{x}(t)) = \min_{0 \leqslant t \leqslant t_f} \left\{ \sqrt{(z_{1,i} - x_1(t))^2 + (z_{3,i} - x_3(t))^2} - d_i \right\} \leqslant 0, \tag{48}$$

where $(z_{1,i}, z_{3.i})$ are the coordinates of the area centers on the horizontal plane, $i = 1, \ldots, S$, $S$ is the number of areas, $S = 3$, $d_i$ is the size of the area, $d_i = 0.6$.

The coordinates of the required area centers are $z_{1,1} = 10$, $z_{3,1} = 0$, $z_{1,2} = 10$, $z_{3,2} = 10$, $z_{1,3} = 0$, $z_{3,3} = 10$.

The optimal control problem was solved on the basis of the direct approach by a hybrid evolutionary algorithm [21]. To solve the problem, we set $t^+ = 15.2$ and time interval $\Delta t = 0.4$. There were 38 intervals in total. For each interval boundary, $M = 4$ controls had to be found. Thus, the total $(N+1)m = 39 \times 4 = 156$, $\mathbf{q} = [q_1 \ldots q_{156}]^T$ parameters were found.

The values of the found parameters for optimal control are given in the Data Availability section. The value of the quality criterion (35) is $J_4 = 15.1221$. Figure 10 shows the projection of the new found optimal trajectory on the horizontal plane.

Figure 11 shows a new optimal trajectory (in blue) and eight perturbed trajectories (in black) for an object with a universal stabilisation system. As it can be seen from the figure, all perturbed trajectories are in the neighbourhood of the optimal trajectory and satisfy the phase constraints. A value of the quality criterion (35) for the object with a universal stabilisation system without perturbations (blue) is $J_4 = 16.341$.

To estimate the results, a comparable experiment was performed. For models with and without a stabilisation system, the initial states were subjected to random perturbations,

$$x_i(0) = x_i^0 + 2\beta_0(\xi(t) - 1), \tag{49}$$

where $\xi(t)$ is a random numbers generator. Function $\xi(t)$ returns a random number from 0 to 1 after each call, $\beta_0$ is a level of perturbations.

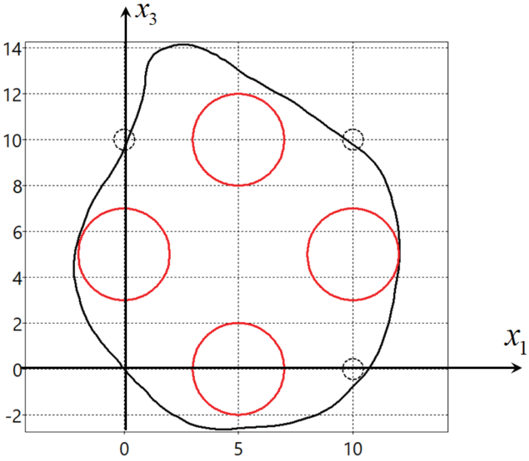


**Figure 10.** Projection of a new optimal trajectory on the horizontal plane.

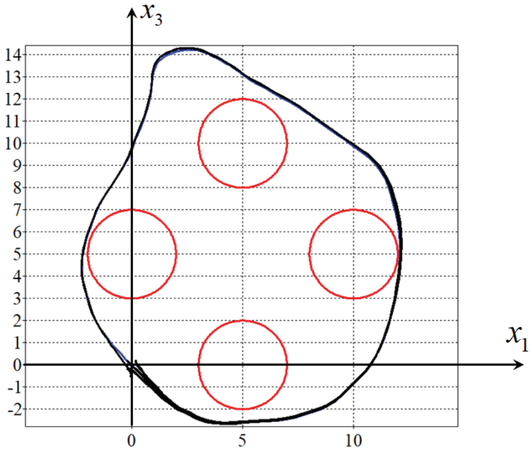


**Figure 11.** Projections of a new optimal trajectory (blue) and perturbed trajectories (black) on the horizontal plane.

The values of quality criterion (35) for perturbed initial states at $\beta_0 = 0.1$ for object with and without stabilisation system are given in Table 1.

The last two lines show average value and standard deviation (SD) on experiments. As it can be seen from Table 1, an object model without a stabilisation system is essentially more sensitive to the perturbation of initial conditions.

**Table 1.** Sensitivity of solutions to perturbations of initial states.

| No | Direct | Stabilisation |
|---|---|---|
| 1 | 20.5112 | 16.4147 |
| 2 | 18.5402 | 16.4715 |
| 3 | 19.3551 | 16.4456 |
| 4 | 18.3839 | 16.3356 |
| 5 | 20.0805 | 16.4408 |
| 6 | 18.7307 | 16.4417 |
| 7 | 19.0710 | 16.4456 |
| 8 | 21.3337 | 16.4120 |
| 9 | 20.3913 | 16.4495 |
| 10 | 18.5464 | 16.4710 |
| Avg. | 19.4940 | 16.4323 |
| SD | 1.02170 | 0.04012 |

## 5. Conclusions

The paper presents a statement of the complete optimal control problem in which, according to the classical statement, it is necessary to find the control function and the optimal program trajectory, and to implement the solution, it is necessary to solve the synthesis problem of motion stabilisation along the program trajectory. To solve the stabilisation system synthesis problem, machine learning of control by the symbolic regression method is used.

For the first time, it is proposed to synthesise a universal stabilisation system that ensures the motion of an object along different trajectories from some class. To synthesise a universal stabilisation system, a training set of different trajectories is generated. As a result of solving the problem of stabilisation system synthesis, we obtain one universal stabilisation system which provides object motion along all trajectories from the training set.

The obtained solution was tested on stabilisation of the object motion along the trajectory, which was not included in the training set. An example of solving the complete optimal control problem for quadcopter motion in space with four obstacles was given. The optimal trajectory was a closed curve, which passed through the specified areas and avoided the obstacles.

Training of the stabilisation system was performed on trajectories that differed significantly from the example. The training set included four trajectories that were obtained as a result of solving the optimal control problem of quadcopter spatial motion from a given initial point to a given terminal point in space with two phase constraints. The trajectories differed in that they avoided obstacles from different sides. The computational experiment showed that the universal stabilisation system provided qualitative motion of the object along the closed optimal trajectory, which was not included in the training set.

## 6. Discussion

Future research is aimed at expanding of the class of trajectories that are included in the training set. It is important to identify and investigate the properties of trajectories that cannot be stabilised by the obtained stabilisation system, i.e., to determine the limits of applicability of the proposed universal stabilisation system.

Furthermore, the construction of universal stabilisation systems for different control objects will exclude the most time-consuming stage of synthesis of the trajectory motion stabilisation system from the solution of the optimal control problem. If for some object it is necessary to construct several stabilisation systems for different classes of trajectories, it is necessary to synthesise such stabilisation systems and further use them to solve different optimal control problems.

**Author Contributions:** Conceptualization, A.D. and E.S.; methodology, A.D.; software, A.D. and E.S.; validation, E.S.; formal analysis, A.D.; investigation, E.S.; resources, A.D.; writing—original draft preparation, A.D. and E.S.; writing—review and editing, E.S.; visualization, A.D.; supervision, A.D. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported by the Ministry of Science and Higher Education of the Russian Federation, project No. 075-15-2020-799.

**Data Availability Statement:** The data that support the findings of this study are openly available in https://cloud.mail.ru/public/4CBU/sBEA3B2wE, access date: 1 July 2023.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Pontryagin, L.S.; Boltyanskii, V.G.; Gamkrelidze, R.V.; Mishchenko, E.F. *The Mathematical Theory of Optimal Process*; Gordon and Breach Science Publishers: New York, NY, USA; London, UK; Paris, France; Montreux, Switzerland; Tokyo, Japan, 1985; 360p.
2. Bellman, R.E.; Dreyfus, S.E. *Applied Dynamic Programming*; Princeton Legacy Library: Princeton, NJ, USA, 1962.
3. Diveev, A.I. Refinement of Optimal Control Problem for Practical Implementation of Its Solution. *Dokl. Math.* **2023**, *107*, 28–36. [CrossRef]
4. Brockett, R.W. Asymptotic Stability and Feedback Stabilization. In *Differential Geometric Control Theory*; Brockett, R.W., Millman, R.S., Sussmann, H.J., Eds.; Birkhauser: Boston, MA, USA, 1983; pp. 181–191.
5. Walsh, G.; Tilbury, D.; Sastry, S.; Murray, R.; Laumond, J.P. Stabilization of Trajectories for Systems with Nonholonomic Constraints. *IEEE Trans. Autom. Control* **1994**, *39*, 216–222. [CrossRef]
6. Samir, A.; Hammad, A.; Hafez, A.; Mansour, H. Quadcopter Trajectory Tracking Control using State-Feedback Control with Integral Action. *Int. J. Comput. Appl.* **2017**, *168*, 1–7. [CrossRef]
7. Nguyen, A.T.; Xuan-Mung, N.; Hong, S.-K. Quadcopter Adaptive Trajectory Tracking Control: A New Approach via Backstepping Technique. *Appl. Sci.* **2019**, *9*, 3873. [CrossRef]
8. Mohamed, M.J.; Abbas, M.Y. Design a Fuzzy PID Controller for Trajectory Tracking of Mobile Robot.*Eng. Technol. J.* **2018**, *36A*, 100–110. [CrossRef]
9. Ma, T.; Wong, S. Trajectory Tracking Control for Quadrotor UAV. In Proceedings of the 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO), Macau, Macao, 5–8 December 2017.

10. Diveev, A.; Sofronova, E. Synthesized Control for Optimal Control Problem of Motion Along the Program Trajectory. In Proceedings of the 2022 8th International Conference on Control, Decision and Information Technologies (CoDIT), Istanbul, Turkey, 17–20 May 2022; pp. 475–480.
11. Diveev, A.; Sofronova, E.; Konyrbaev, N.; Bexeitova, A. Stabilization of Movement Along an Optimal Trajectory and Its Solution. *Eng. Proc.* **2023**, *33*, 12.
12. Diveev, A.I.; Sofronova, E.A. The network operator method for search of the most suitable mathematical equation. In *Bio-Inspired Computational Algorithms and Their Applications*; InTech: Rijeka, Croatia, 2012; pp. 19–42.
13. Diveev, A.; Kazaryan, D.; Sofronova, E. Symbolic regression methods for control system synthesis. In Proceedings of the 2014 22nd Mediterranean Conference on Control and Automation, MED 2014, Palermo, Italy, 16–19 June 2014; pp. 587–592.
14. Quade, M.; Abel, M.; Isele, T. Machine learning control—Explainable and analyzable methods. *Phys. D Nonlinear Phenom.* **2020**, *412*, 132582. [CrossRef]
15. Koza, J.R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*; MIT Press: Cambridge, MA, USA, 1992; 819p.
16. Koza, J.; Keane, M.; Yu, J.; Bennett, F., III; Mydlowec, W. Automatic Creation of Human-Competitive Programs and Controllers by Means of Genetic Programming. *Genet. Program. Evolvable Mach.* **2000**, *1*, 121–164. [CrossRef]
17. Marchetti, F.; Minisci, E. Genetic Programming Guidance Control System for a Reentry Vehicle under Uncertainties. *Mathematics* **2021**, *9*, 1868. [CrossRef]
18. Sofronova, E.A.; Diveev, A.I. Universal Approach to Solution of Optimization Problems by Symbolic Regression. *Appl. Sci.* **2021**, *11*, 5081. [CrossRef]
19. Karnani, C.; Raza, S.; Asif, A.; Ilyas, M. Adaptive Control Algorithm for Trajectory Tracking of Underactuated Unmanned Surface Vehicle (UUSV). *J. Robot.* **2023**, *2023*, 4820479. [CrossRef]
20. Chipofya, M.; Lee, D.J.; Chong, K.T. Trajectory Tracking and Stabilization of a Quadrotor Using Model Predictive Control of Laguerre Functions. *Abstr. Appl. Anal.* **2015**, *2015*, 916864. [CrossRef]
21. Diveev, A. Hybrid evolutionary algorithm for optimal control problem. In *Intelligent Systems and Applications. IntelliSys 2022*; Arai K., Ed.; Lecture Notes in Networks and Systems; Springer: Cham, Switzerland, 2023; Volume 543, pp. 726–738.
22. Davis, L. *Handbook of Genetic Algorithms*; Van Nostrand Reinhold: New York, NY, USA, 1991.
23. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; Volume IV, pp. 1942–1948.
24. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]

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

mathematics

Article

# Adaptive Synthesized Control for Solving the Optimal Control Problem

**Askhat Diveev [†] and Elizaveta Shmalko [*,†]**

Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, 44/2, Vavilova Str., Moscow 119333, Russia; aidiveev@frccsc.ru

* Correspondence: e.shmalko@frccsc.ru

† These authors contributed equally to this work.

**Abstract:** The development of artificial intelligence systems assumes that a machine can independently generate an algorithm of actions or a control system to solve the tasks. To do this, the machine must have a formal description of the problem and possess computational methods for solving it. This article deals with the problem of optimal control, which is the main task in the development of control systems, insofar as all systems being developed must be optimal from the point of view of a certain criterion. However, there are certain difficulties in implementing the resulting optimal control modes. This paper considers an extended formulation of the optimal control problem, which implies the creation of such systems that would have the necessary properties for its practical implementation. To solve it, an adaptive synthesized optimal control approach based on the use of numerical methods of machine learning is proposed. Such control moves the control object, optimally changing the position of the stable equilibrium point in the presence of some initial position uncertainty. As a result, from all possible synthesized controls, one is chosen that is less sensitive to changes in the initial state. As an example, the optimal control problem of a quadcopter with complex phase constraints is considered. To solve this problem, according to the proposed approach, the control synthesis problem is firstly solved to obtain a stable equilibrium point in the state space using a machine learning method of symbolic regression. After that, optimal positions of the stable equilibrium point are searched using a particle swarm optimization algorithm using the source functional from the initial optimal control problem statement. It is shown that such an approach allows for generating the control system automatically by computer, basing this on the formal statement of the problem and then directly implementing it onboard as far as the stabilization system has already been introduced.

**Keywords:** stabilization; optimization; symbolic regression; synthesized control; evolutionary computations; quadcopter model; ordinary differential equations

**MSC:** 49M99

**Citation:** Diveev, A.; Shmalko, E. Adaptive Synthesized Control for Solving the Optimal Control Problem. *Mathematics* **2023**, *11*, 4035. https://doi.org/10.3390/math11194035

Academic Editor: Liliya Demidova

Received: 2 September 2023
Revised: 19 September 2023
Accepted: 20 September 2023
Published: 22 September 2023

**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

## 1. Introduction

Long ago, Leonard Euler spoke about the optimal arrangement of everything in the world: "For since the fabric of the universe is most perfect and the work of a most wise Creator, nothing at all takes place in the universe in which some rule of maximum or minimum does not appear". Striving for optimality is natural in every sphere.

In order to optimally move an autonomous robot to a certain target position, currently, as a standard, engineers first solve the problem of optimal control, obtain the optimal trajectory, and then solve the additional problem of moving the robot along the obtained optimal trajectory. In most cases, the following approach is used to move the robot along a path. Initially, the object is made stable relative to a certain point in the state space. Then, the stability points are positioned along the desired path and the object is moved along the trajectory by following these points from one point to another [1–7]. The difference between the existing methods is in solving the control synthesis problem to ensure stability relatively to some equilibrium point in the state space and in the location of these stability points.

Often, to ensure stability, the model of the control object is linearized relative to a certain point in the state space. Then, for the linear model of the object, a linear feedback control is found to arrange the eigenvalues of the closed-loop control system matrix on the left side of the complex plane. Sometimes, to improve the quality of stabilization, control channels or components of the control vector are defined that affect the movement of an object along a specific coordinate system axis of the state space. Then controllers, as a rule PI controllers, are inserted into these channels with the coefficients that are adjusted according to the specified control quality criterion [3,4]. In some cases, analytical or semi-analytical methods are used to solve the control synthesis problem and build nonlinear stable control systems [5,7]. But the stability property of the nonlinear model of the control object, obtained from the linearization of this model, is generally preserved only in the vicinity of a stable equilibrium point.

The main drawback of the approach when the control object is moved along the stable points on the trajectory is that even if this trajectory is obtained as a solution of the optimal control problem [8], then the movement itself will never be optimal. To ensure optimality, it is necessary to move along the trajectory at a certain speed, but when approaching the stable equilibrium point, the speed of the control object tends to zero.

The optimal control problem generally does not require ensuring the stability of the control object. The construction of a stabilization system that provides the stability of the object relative to some equilibrium point in the state space is carried out by the researcher to achieve predictable behavior of the control object in the vicinity of a given trajectory.

The optimal control problem in the classical formulation is solved for a control object without any stabilization system; therefore, the resulting optimal control and the optimal trajectory will not be optimal for this object with a further introduced stabilization system. It follows that the classical formulation of the optimal control problem [9] is missing something as far as its solution cannot be directly implemented in the real object, since this leads to an open-loop control. The open-loop control system is very sensitive to small disturbances, but they are always possible in real conditions, since no model accurately describes the control object. In order to achieve optimal control in a real object, it is necessary to build a feedback control system, which should provide some additional properties, for example stability relative to the trajectory or points on this trajectory. The authors of [10] proposed an extended formulation statement of the optimal control problem, which has additional requirements established for the optimal trajectory. The optimal trajectory must have a non-empty neighborhood with a property of attraction. Performing these requirements provides implementation of the solution of the optimal control problem directly in the real control object.

In [11,12], an approach to solving the extended optimal control problem on the base of the synthesized control is presented. This approach ensures obtaining a solution of the optimal control problem in the class of practically implemented control functions. According to this approach, initially, the control synthesis problem is solved. So, the control object becomes stable in the state space relatively to some equilibrium point. In the second stage, the optimal control problem is solved by determination of optimal positions of the stable equilibrium point. Switching stable points after a constant time interval ensures moving the control object from initial state to the terminal one optimally according to the given quality criterion. Optimal positions of stable equilibrium points can be far from the optimal trajectory in the state space; therefore, a control object does not slow down its motion speed. Studies of synthesized control in various optimal control problems have shown that such control is not sensitive to perturbations and can be directly implemented in a real object [13,14].

In synthesized control, the optimal control problem is solved for a control object already with a stabilization system. Another advantage of synthesized control is that the position of the stable point does not change during the time interval; that is, an optimal control function is solved using the class of piece-wise constant functions, which simplifies the search for the optimal solution.

It is possible that piece-wise constant control in the synthesized approach finds several optimal solutions with practically the same value of the quality criterion. This circumstance prompted in us the idea to find among all almost-optimal solutions one that is less sensitive to perturbations. This approach is called adaptive synthesized control.

In this work, a principle of adaptive synthesized control is proposed in Section 2, methods for solving it are discussed in Section 3 and further in the Section 4, a computational experiment to determine the solution of the optimal control problem for the spatial motion of quadcopter by adaptive synthesized control is considered.

## 2. Adaptive Synthesized Control

Consider the principle of adaptive synthesized control for solving the optimal control problem in its extended formulation [10].

Initially, the control synthesis problem is solved to provide stability of the control object relatively some point in the state space. In the problem, the mathematical model of the control object in the form of ordinary differential equation system is given.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \tag{1}$$

where $\mathbf{x}$ is a state vector, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u}$ is a control vector, $\mathbf{u} \in \mathrm{U} \subseteq \mathbb{R}^m$, U is a compact set that determines restrictions on the control vector.

The domain of admissible initial states is given

$$\mathrm{X}_0 \subseteq \mathbb{R}^n. \tag{2}$$

To solve the problem numerically, the initial domain (2) is taken in the form of the finite number of points in the state space:

$$\tilde{\mathrm{X}}_0 = \{\mathbf{x}^{0,1}, \dots, \mathbf{x}^{0,K}\}. \tag{3}$$

Sometimes, it is convenient to set one initial state and deviations from it:

$$\mathbf{x}^{0,j} = \mathbf{x}^0 - \mathbf{\Delta}_0 + 2 \odot (j)_2 \mathbf{\Delta}_0,\ j = 1, \dots, 2^n - 1, \tag{4}$$

where $\mathbf{x}^0$ is a given initial state, $\mathbf{\Delta}_0$ is a deviations vector, $\mathbf{\Delta}_0 = [\Delta_1 \dots \Delta_n]^T$, $\odot$ is Hadamard product of vectors, $(j)_2$ is a binary code of the number $j$. In this case $K = 2^n - 1$.

The stabilization point as a terminal state is given by

$$\mathbf{x}^{f_1} \in \mathbb{R}^n. \tag{5}$$

It is necessary to find a control function in the form

$$\mathbf{u} = \mathbf{h}(\mathbf{x}^{f_1}\mathbf{x}) \in \mathrm{U}, \tag{6}$$

where $\mathbf{h}(\mathbf{x})$: $\mathbb{R}^n \to \mathbb{R}^m$, such that it minimizes the quality criterion

$$J_0 = \sum_{i=0}^{K} \Big( t_{f_1,i} + p\|\mathbf{x}^{f_1} - \mathbf{x}(t_{f_1,i}, \mathbf{x}^{0,i})\| \Big) \to \min, \tag{7}$$

where $t_{f_1,i}$ is the time of achieving the terminal state (5) from the initial state $\mathbf{x}^{0,i}$, $t_{f_1,i}$ is determined by an equation

$$t_{f_1,i} = \begin{cases} t, \text{ if } t < t^+ \text{and } \|\mathbf{x}^{f_1} - \mathbf{x}(t, \mathbf{x}^{0,i})\| \leq \varepsilon_0 \\ t^+, \text{ otherwise} \end{cases}, \tag{8}$$

$\mathbf{x}(t, \mathbf{x}^{0,i})$ is a particular solution from initial state $\mathbf{x}^{0,i}$, $i = 1, \dots, K$, of the differential Equation (1) with an inserted control function (6)

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{h}(\mathbf{x}^{f_1} - \mathbf{x})), \tag{9}$$

$\varepsilon_0$ is a given accuracy for hitting to terminal state (5), $t^+$ is a given maximal time for control process, $p$ is a weight coefficient.

Further, using the principles of synthesized optimal control the following optimal control problem is considered. The model of control object in the form (9) is used

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{h}(\mathbf{x}^* - \mathbf{x})), \tag{10}$$

where the terminal state vector (5) is changed into the new unknown vector $\mathbf{x}^*$, which will be a control vector in the considered optimal control problem.

In accordance with the classical formulation of the optimal control problem, the initial state of the object (10) is given

$$\mathbf{x}^0 \in \mathbb{R}^n. \tag{11}$$

In the engineering practice, there can be some deviations in the initial position; therefore, in adaptive synthesized control, instead of one initial state (11) the set of initial states used are defined by Equation (4). The vector of initial deviations $\boldsymbol{\Delta}_0$ is defined as a level of disturbances.

The goal of control is defined by achievement of the terminal state

$$\mathbf{x}^f \in \mathbb{R}^n. \tag{12}$$

The quality criterion is given

$$J_1 = \int_0^{t_f} f_0(\mathbf{x}, \mathbf{h}(\mathbf{x}^* - \mathbf{x}))dt \to \min, \tag{13}$$

where $t_f$ is a terminal time, $t_f$ is not given but is limited, $t_f \leq t^+$, $t^+$ is a given limit time of control process.

According to the principle of synthesized control, it is necessary to choose time interval $\Delta t$ and to search for optimal constant values of the control vector $\mathbf{x}^{*,i}$ for each interval

$$\mathbf{x}^* = \mathbf{x}^{*,i}, \text{if } (i-1)\Delta t \leq t < i\Delta t,\ i = 1 \dots, M, \tag{14}$$

where $M$ is a number of intervals

$$M = \left\lfloor \frac{t^+}{\Delta t} \right\rfloor. \tag{15}$$

So the system (10) with the found optimal constant values of the control vector (14) in the right-hand side of differential equations has a particular solution which reaches the terminal state (12) from the given initial state (11) with an optimal value of the quality criterion (13).

Algorithmically, in the second stage of the adaptive synthesized control approach, the optimal values of the vector $\mathbf{x}^*$ are found as a result of the optimization task with the following quality criterion, which takes into account the given grid according to the initial conditions:

$$J_2 = \sum_{i=1}^{K} \left( \int_0^{t_{f,i}} f_0(\mathbf{x}, \mathbf{h}(\mathbf{x}^{*,i} - \mathbf{x}))dt + p\|\mathbf{x}^f - \mathbf{x}(t_{f,i}, \mathbf{x}^{0,i})\| \right) \to \min_{\mathbf{x}^*}, \tag{16}$$

where $K$ is number of initial states, $t_{f,i}$ is determined by Equation (8).

## 3. Methods of Solving

As described in the previous section, the approach based on the principle of adaptive synthesized optimal control consists of two stages.

To implement the first stage of the approach under consideration for solving the control synthesis problem (1)–(9), any known method can be used. For linear systems, for example, methods of modal control [15] can be applied, as well as such analytical methods such as backstepping [16,17] or synthesis based on the application of the Lyapunov function [18]. In practice, stability is ensured through linearization of the model (1) in the terminal state and setting PI or PID controllers in control channels [19,20]. All known analytical and technical methods have their limitations, which mostly depend on the type of the model used to describe the control object. The mathematical formulation of the stabilization problem as a control synthesis problem is needed to apply numerical methods and automatically obtain a feedback control function. Today, to solve the synthesis problem for nonlinear dynamic objects of varying complexity, modern numerical methods of machine learning can be applied [21]. Among different machine learning techniques, only symbolic regression allows searching both for the structure of the needed mathematical function and its parameters. In our case, the needed function is a control function. So, in the present paper machine learning by symbolic regression [22,23] is used.

Methods of symbolic regression search for the mathematical expression of the desired function in the encoded form. These methods differ in the form of this code. The search for solutions is performed in the space of codes by a special genetic algorithm.

Let us demonstrate the main features of symbolic regression on the example of the network operator method (NOP), which was used in this work in the computational experiment. To code a mathematical expression NOP uses an alphabet of elementary functions:

- Functions without arguments or parameters and variables of the mathematical expression

$$\mathrm{F}_0 = \{f_{0,1} = x_1, \dots, f_{0,n} = x_n, f_{0,n+1} = q_1, \dots, f_{0,n+r} = q_{n+r}\}; \tag{17}$$

- Functions with one argument

$$\mathrm{F}_1 = \{f_{1,1}(z) = z, f_{1,2}(z), \dots, f_{1,W}(z)\}; \tag{18}$$

- Functions with two arguments

$$\mathrm{F}_2 = \{f_{2,1}(z_1, z_2), \dots, f_{2,V}(z_1, z_2)\}. \tag{19}$$

Any elementary function is coded by two digits: the first one is the number of arguments, the second one is the function number in the corresponding set. These digits are written as indexes of elements in the introduced sets of the alphabet (17)–(19). The set of functions with one argument must include the identity function $f_{1,1}(z) = 1$. Functions with two arguments should be commutative, associative and have a unit element.

NOP encodes a mathematical expression in the form of an oriented graph. Source-nodes of the NOP-graph are connected with functions without arguments, while other nodes are connected with functions with two arguments. Arcs of the NOP-graph are connected with functions with one argument. If on the NOP-graph some node has one input arc, then the second argument is a unit element for the function with two arguments connected with this node.

Let us define the following alphabet of elementary functions:

$$\begin{array}{rcl} \mathrm{F}_0 & = & \{f_{0,1} = x_1, f_{0,2} = x_2, f_{0,3} = q_1, f_{0,4} = q_2\}; \\ \mathrm{F}_1 & = & \{f_{1,1}(z) = z, f_{1,2}(z) = -z, f_{1,3}(z) = \cos(z), f_{1,4}(z) = \sin(z)\}; \\ \mathrm{F}_2 & = & \{f_{2,1}(z_1, z_2) = z_1 + z_2, f_{2,2}(z_1, z_2) = z_1 z_2\}. \end{array} \tag{20}$$

With this alphabet the following mathematical expressions can be encoded in the form of NOP:

$$\begin{aligned} y_1 &= \cos(x_1)\sin(x_2) - \sin(x_1)\cos(x_2); \\ y_2 &= \cos(q_1 x_1 + q_2 \sin(x_2)); \\ y_3 &= q_1 \sin(q_2 x_2) + q_2 \cos(q_1 x_1); \\ y_4 &= q_1 \sin(q_2 \cos(x_1)) + q_2 \cos(q_1 \sin(x_2)). \end{aligned} \tag{21}$$

The NOP-graphs of these mathematical expressions are presented in Figure 1. The nodes of the graph are numbered. Inside each node there is either the number of a binary operation or an element of the set of variables and parameters $F0$, and the arcs of the graph indicate the numbers of unary operations.

In the computer memory, the NOP-graphs are presented in the form of integer matrices.

$$\boldsymbol{\Psi} = [\psi_{i,j}],\ i,j = 1,\dots,L. \tag{22}$$

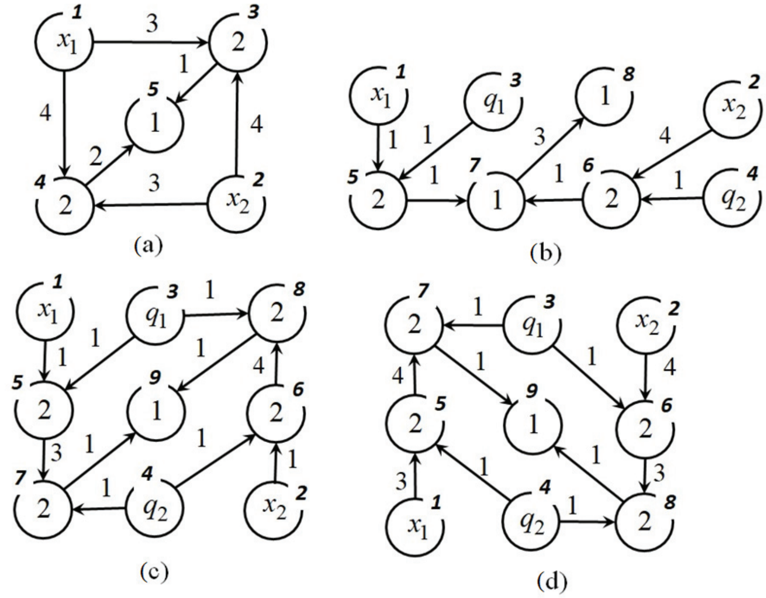


**Figure 1.** NOP-graphs for mathematical expressions (21), (**a**) $y_1$, (**b**) $y_2$, (**c**) $y_3$, (**d**) $y_4$.

As the NOP-nodes are enumerated in such a way that the node number from which an arc comes out is less than the node number to which an arc enters, then the NOP-matrix has an upper triangular form. Every line of the matrix corresponds some node of the graph. Lines with zeros in the main diagonal corresponds to source-nodes of the graph. Other elements in the main diagonal are the function numbers with two arguments. Non-zero elements above the main diagonal are the function numbers with one argument.

NOP-matrices for the mathematical expressions (21) have the following forms:

$$\boldsymbol{\Psi}_1 = \begin{bmatrix} 0 & 0 & 3 & 4 & 0 \\ 0 & 0 & 4 & 3 & 0 \\ 0 & 0 & 2 & 0 & 2 \\ 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},\ \boldsymbol{\Psi}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\boldsymbol{\Psi}_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \; \boldsymbol{\Psi}_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{23}$$

To calculate a mathematical expression by its NOP-matrix, initially, the vector of nodes is determined. The number of components of the vector of nodes equals to the number of nodes in a graph. The initial vector of nodes includes variables and parameters in positions that correspond to source nodes, as well as other components equal to the unit elements of the corresponding functions with two arguments. Further, every line of the matrix is checked. If element of the matrix does not equal zero, then corresponding element of the vector of nodes is changed. To calculate mathematical expression by the NOP-matrix, the following equation is used:

$$z_j^{(i)} \leftarrow \begin{cases} f_{2,\psi_{j,j}}(z_j^{(i-1)}, f_{1,\psi_{i,j}}(z^{(i-1)_i})), \text{ if } \psi_{i,j} \neq 0 \\ z_j^{(i-1)}, \text{ otherwise} \end{cases} , i = 1, \dots, L-1, \; j = i+1, \dots, L, \tag{24}$$

where

$$z_i^{(0)} = \begin{cases} f_{0,i}, \text{ if } \psi_{i,i} = 0 \\ e_{\psi_{i,i}}, \text{ otherwise} \end{cases} , \tag{25}$$

$e_j$ is a unit element for function with two arguments $f_{2,j}(z_1, z_2)$,

$$f_{2,j}(e_j, z) = f(z, e_j) = z. \tag{26}$$

Consider an example of calculating the second mathematical expression in (21) on its NOP-matrix $\boldsymbol{\Psi}_2$.

The initial vector of nodes is

$$\mathbf{z}^{(0)} = [x_1 \; x_2 \; q_1 \; q_2 \; 1 \; 1 \; 0 \; 0]^T.$$

Further, all strings in the matrix $\boldsymbol{\Psi}_2$ are checked and non-zero elements are found.

$$\psi_{1,5} = 1, \; z_5^{(1)} = f_{2,2}(z_5^{(0)}, f_{1,1}(z_1^{(0)})) = 1 \cdot f_{1,1}(z_1^{(0)}) = 1 \cdot x_1 = x_1,$$

$$\psi_{2,6} = 4, \; z_6^{(2)} = f_{2,2}(z_6^{(1)}, f_{1,4}(z_1^{(1)})) = 1 \cdot f_{1,4}(z^{(1)}) = 1 \cdot \sin(x_2) = \sin(x_2),$$

$$\psi_{3,5} = 1, \; z_5^{(3)} = f_{2,2}(z_5^{(2)}, f_{1,1}(z_3^{(2)})) = x_1 \cdot q_1 = q_1 x_1,$$

$$\psi_{4,6} = 1, \; z_6^{(4)} = f_{2,2}(z_6^{(3)}, f_{1,1}(z_4^{(3)})) = \sin(x_2) \cdot q_2 = q_2 \sin(x_2),$$

$$\psi_{5,7} = 1, \; z_7^{(5)} = f_{2,1}(z_7^{(4)}, f_{1,1}(z_5^{(4)})) = 0 + q_1 x_1 = q_1 x_1,$$

$$\psi_{6,7} = 1, \; z_7^{(6)} = f_{2,1}(z_7^{(5)}, f_{1,1}(z_6^{(5)})) = q_1 x_1 + q_2 \sin(x_2),$$

$$\psi_{7,8} = 3, \; z_8^{(7)} = f_{2,1}(z_8^{(6)}, f_{1,3}(z_7^{(6)})) = 0 + \cos(q_1 x_1 + q_2 \sin(x_2)) = \cos(q_1 x_1 + q_2 \sin(x_2)).$$

The last mathematical expression coincides with the needed mathematical expression for $y_2$ (21).

So, we considered the way of coding in the NOP method. Then, to search for an optimal mathematical expression in some task, the NOP method applies a principle of small variations of a basic solution. According to this principle, one possible solution is encoded in the form of the NOP-matrix $\mathbf{\Psi}_0$. This solution is the basic solution and it is set by a researcher as a good solution. Other possible solutions are presented in the form of sets of small-variation vectors. A small variation vector consists of four integer numbers

$$\mathbf{w} = [w_1\ w_2\ w_3\ w_4]^T, \tag{27}$$

where $w_1$ is a type of small variation, $w_2$ is a line number of the NOP-matrix, $w_3$ is a column number of NOP-matrix, $w_4$ is a new value of an NOP-matrix element. There are four types of small variations: $w_1 = 0$ is an exchange of the function with one argument, if $\psi_{w_2,w_3} \neq 0$, then $\psi_{w_2,w_3} \leftarrow w_4$; $w_1 = 1$ is an exchange of the function with two arguments, if $\psi_{w_2,w_2} \neq 0$, then $\psi_{w_2,w_2} \leftarrow w_4$; $w_1 = 2$ is an insertion of the additional function with one argument, if $\psi_{w_2,w_3} = 0$, then $\psi_{w_2,w_3} \leftarrow w_4$; $w_1 = 3$ is an elimination of the function with one argument, if $\psi_{w_2,w_3} \neq 0$ and $\exists\psi_{w_2,j} \neq 0$, $j > w_2$, $j \neq w_3$ and $\exists\psi_{i,w_3} \neq 0$, $i \neq w_2$, then $\psi_{w_2,w_3} \leftarrow 0$.

The initial population includes $H$ possible solutions. Each possible solution $i \in \{1, \ldots, H\}$ except the basic solution is encoded in the form of the set of small variation vectors

$$\mathrm{W}_i = (\mathbf{w}^{i,1}, \ldots, \mathbf{w}^{i,d}),\ i \in \{1, \ldots, H\}, \tag{28}$$

where $d$ is a depth of variations, which is set as a parameter of the algorithm.

The NOP-matrix of a possible solution is determined after application of all small variations to the basic solution

$$\mathbf{\Psi}_i = \mathbf{w}^{i,d} \circ \ldots \circ \mathbf{w}^{i,1} \circ \mathbf{\Psi}_0,\ i \in \{1, \ldots, H\}, \tag{29}$$

Here, the small variation vector is written as a mathematical operator changing matrix $\mathbf{\Psi}_0$.

During the search process sometimes the basic solution is replaced by the current best possible solution. This process is called a change of an epoch.

Consider an example of applying small variations to the NOP-matrix $\mathbf{\Psi}_3$. Let $d = 3$ and there are three following small variation vectors:

$$\mathbf{w}^1 = [0\ 3\ 5\ 2]^T,\ \mathbf{w}^2 = [2\ 5\ 6\ 3]^T,\ \mathbf{w}^3 = [2\ 8\ 9\ 4]^T$$

After application of these small variation vectors to the NOP-matrix $\mathbf{\Psi}_3$, the following NOP-matrix is obtained:

$$\mathbf{\Psi}_5 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 3 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

This NOP-matrix corresponds to the following mathematical expression:

$$y_5 = \sin(q_1 \sin(q_2 x_2 \cos(-q_1 x_1))) + q_2 \cos(-q_1 x_1).$$

Similar to a search engine, a genetic algorithm is used. To perform the main genetic operation of crossover, two possible solutions are selected randomly

$$\begin{aligned} \mathrm{W}_\alpha &= (\mathbf{w}^{\alpha,1}, \ldots, \mathbf{w}^{\alpha,d}), \\ \mathrm{W}_\beta &= (\mathbf{w}^{\beta,1}, \ldots, \mathbf{w}^{\beta,d}). \end{aligned} \tag{30}$$

A crossover point is selected randomly $c \in \{1, \dots, d\}$. Two new possible solutions are obtained as the result of exchanging elements of the selected possible solutions after the crossover point:

$$\begin{array}{rcl} \mathrm{W}_{H+1} & = & (\mathbf{w}^{\alpha,1}, \dots, \mathbf{w}^{\alpha,c}, \mathbf{w}^{\beta,c+1}, \dots, \mathbf{w}^{\beta,d}), \\ \mathrm{W}_{H+2} & = & (\mathbf{w}^{\beta,1}, \dots, \mathbf{w}^{\beta,c}, \mathbf{w}^{\alpha,c+1}, \dots, \mathbf{w}^{\alpha,d}). \end{array} \tag{31}$$

The second stage of the synthesized principle under consideration is to solve the problem of optimal control via determination of the optimal position of the equilibrium points. Studies have shown that for a complex optimal control problem with phase constraints, evolutionary algorithms allow the system to cope with such problems. Good results were demonstrated [24] by such algorithms as a genetic algorithm (GA) [25], a particle swarm optimization (PSO) algorithm [26–28], a grey wolf optimizer (GWO) algorithm [29] or a hybrid algorithm [24] involving one population of possible solutions and all three evolutionary transformations of GA, PSO and GWO selected randomly.

## 4. Computational Experiment

Consider the optimal control problem for the spatial motion of a quadcopter. In the problem, the quadcopter should move for a minimum time on a closed-loop circle from the given initial state to the same terminal state, avoiding collisions with obstacles and passing through the given areas.

### 4.1. Mathematical Model of Spatial Movement of Quadcopter

In the general case, the mathematical model of a quadcopter as a hard body has the following form:

$$\begin{array}{rcl} \ddot{x} & = & F(\cos(\gamma)\sin(\theta)\cos(\psi) + \sin(\gamma)\sin(\psi))/m; \\ \ddot{y} & = & F\cos(\gamma)\cos(\theta)/m - g; \\ \ddot{z} & = & F(\cos(\gamma)\sin(\theta)\sin(\psi) + \sin(\gamma)\cos(\psi))/m; \\ \ddot{\gamma} & = & ((I_{yy} + I_{zz})\dot{\theta}\dot{\psi} + M_x)/I_{xx}; \\ \ddot{\psi} & = & ((I_{zz} + I_{xx})\dot{\gamma}\dot{\theta} + M_y)/I_{yy}; \\ \ddot{\theta} & = & ((I_{xx} + I_{yy})\dot{\gamma}\dot{\theta} + M_z)/I_{zz}. \end{array} \tag{32}$$

where $F$ is a summary thrust force of all drone screws, $m$ is a mass of drone, $g$ is acceleration of gravity, $g = 9.80665$, $M_x$, $M_y$, $M_z$ are control moments around the respective axes.

Figure 2 shows how the angles of a quadcopter turn are linked with its axes.

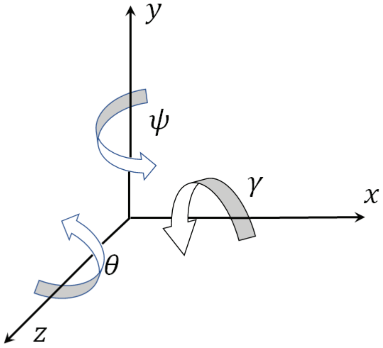


**Figure 2.** Inertial coordinate system for quadcopter.

To transform the model (32) to a vector record, the following designations are entered: $x = x_1$, $y = x_2$, $z = x_3$, $\dot{x}_1 = x_4$, $\dot{x}_2 = x_5$, $\dot{x}_3 = x_6$, $\gamma = x_7$, $\psi = x_8$, $\theta = x_9$, $\dot{\gamma} = x_{10}$, $\dot{\psi} = x_{11}$, $\dot{\theta} = x_{12}$, $M_1 = M_x$, $M_2 = M_y$, $M_3 = M_z$.

As a result the following mathematical model is received:

$$
\begin{aligned}
\dot{x}_1 &= x_4;\\
\dot{x}_2 &= x_5;\\
\dot{x}_3 &= x_6;\\
\dot{x}_4 &= F(\cos(x_7)\sin(x_9)\cos(x_8)+\sin(x_7)\sin(x_8))/m;\\
\dot{x}_5 &= F(\cos(x_7)\cos(x_9)/m-g;\\
\dot{x}_6 &= F(\cos(x_7)\sin(x_9)\sin(x_8)+\sin(x_7)\cos(x_8))/m;\\
\dot{x}_7 &= x_{10};\\
\dot{x}_8 &= x_{11};\\
\dot{x}_9 &= x_{12};\\
\dot{x}_{10} &= ((I_{yy}+I_{zz})x_{11}x_{12}+M_x)/I_{xx};\\
\dot{x}_{11} &= ((I_{zz}+I_{xx})x_{10}x_{12}+M_y)/I_{yy};\\
\dot{x}_{12} &= ((I_{xx}+I_{yy})x_{10}x_{11}+M_z)/I_{zz};
\end{aligned}
\tag{33}
$$

where $\mathbf{x}$ is a state space vector, $\mathbf{x} = [x_1 \dots x_n]^T$, $\mathbf{M}$ is a vector of control moments, $\mathbf{M} = [M_1\ M_2\ M_3]^T$.

As a rule, quadcopters are manufactured with some angle stabilization systems. This means that a drone can be stabilized at any angle for some interval. The system of angle stabilization provides a stable location of the drone relatively, given angles by control moments:

$$M_i = w_i(x_7^* - x_7, x_8^* - x_8, x_9^* - x_9, x_{10}, x_{11}, x_{12}),\ i = 1,2,3. \tag{34}$$

Assume that the angular stabilization system works out the given angles of the quadcopter quickly enough, at least in comparison with spatial movement. In this case we can assume that the control of the spatial movement of the quadcopter is carried out using the angular position of the drone and the thrust force. Let us define components of the spatial control vector: $x_7 = u_1$, $x_8 = u_2$, $x_9 = u_3$, $F/m = u_4$.

As a result we receive the following model of spatial quadcopter movement:

$$
\begin{aligned}
\dot{x}_1 &= x_4;\\
\dot{x}_2 &= x_5;\\
\dot{x}_3 &= x_6;\\
\dot{x}_4 &= u_4(\cos(u_1)\sin(u_3)\cos(u_2)+\sin(u_1)\sin(u_2));\\
\dot{x}_5 &= u_4\cos(u_1)\cos(u_3)-g;\\
\dot{x}_6 &= u_4(\cos(u_1)\sin(u_3)\sin(u_2)+\sin(u_1)\cos(u_2)).
\end{aligned}
\tag{35}
$$

In this work, this model is used to obtain optimal control for the spatial motion of the quadcopter.

### *4.2. The Optimal Control Problem for Spatial Motion of Quadcopter*

The model (35) of the control object is given. Here, $\mathbf{x}$ is a state space vector, $\mathbf{x} \in \mathbb{R}^6$, $\mathbf{u}$ is a control vector $\in \mathrm{U} \in \mathbb{R}^4$. U is a compact set that defines restrictions on values of control vector components,

$$
\begin{array}{rcccl}
u_1^- = -\pi/12 & \le & u_1 & \le & \pi/12 = u_1^+,\\
u_2^- = -\pi & \le & u_2 & \le & \pi = u_2^+,\\
u_3^- = -\pi/12 & \le & u_3 & \le & \pi/12 = u_3^+,\\
u_4^- = 0 & \le & u_4 & \le & 12 = u_4^+.
\end{array}
\tag{36}
$$

According to the principle of synthesized control, initially the control synthesized problem (1)–(9) is solved. The model (35) is used as a model of the control object. To construct the set of initial states (4), the following vector of deviations is used:

$$\Delta_0 = [2\ 2\ 2\ 0\ 0\ 0]^T. \tag{37}$$

In the problem, initial state and terminal state were equal:

$$\mathbf{x}^0 = \mathbf{x}^f = [0\ 5\ 0\ 0\ 0\ 0]^T. \tag{38}$$

For calculation of the quality criterion (7), the following parameters are used: $t^+ = 2$, $\varepsilon_0 = 0.1$, $p = 2$.

To solve the control synthesis problem, the network operator method [23] was used. NOP found the following solution:

$$u_i = \begin{cases} u_i^+, \text{ if } \hat{u}_i > u_i^+ \\ u_i^-, \text{ if } \hat{u}_i < u_i^- \\ \hat{u}_i, \text{ otherwise} \end{cases}, i = 1, \dots, m = 4, \tag{39}$$

where

$$\hat{u}_1 = \mu(C), \tag{40}$$

$$\hat{u}_2 = \hat{u}_1 - \hat{u}_1^3, \tag{41}$$

$$\hat{u}_3 = \hat{u}_2 + \rho_{19}(W + \mu(C)) + \rho_{17}(A), \tag{42}$$

$$\hat{u}_4 = \hat{u}_3 + \ln(|\hat{u}_2|) + \text{sgn}(W + \mu(C))\sqrt{|W + \mu(C)|} + \rho_{19}(W) +$$

$$\arctan(H) + \text{sgn}(F) + \arctan(E) + \exp(q_2(x_2^f - x_2)) + \sqrt{q_1}, \tag{43}$$

$$C = q_6(x_6^f - x_6) + q_3(x_3^f - x_3),\ W = V + \tanh(G) + \exp(D),$$

$$A = q_1(x_1^f - x_1) + q_4(x_4^f - x_4),\ H = G + \tanh(F) + \rho_{18}(B),$$

$$F = E + C + \arctan(D) - B,\ E = D + \text{sgn}(x_5^f - x_5) + (x_2^f - x_2)^3,$$

$$V = \exp(H) + \cos(q_6(x_6^f - x_6)) + \text{sgn}(D)\sqrt{|D|},\ G = F + \sqrt[3]{E} + \sin(A),$$

$$B = \sin(q_6(x_6^f - x_6)) + q_5(x_5^f - x_5) + q_2(x_2^f - x_2) + \cos(q_1) + \vartheta(x_2^f - x_2),$$

$$D = \rho_{17}(C) + B^3 + A + \vartheta(q_5(x_5^f - x_5)) + (x_5^f - x_5)^2,$$

$$\mu(z) = \begin{cases} z, \text{ if } |z| < 1 \\ \text{sgn}(z), \text{ otherwise} \end{cases},\ \rho_{17}(z) = \text{sgn}(z)\ln(|z| + 1),$$

$$\rho_{18}(z) = \text{sgn}(z)(\exp(|z|) - 1),\ \rho_{19}(z) = \text{sgn}(z)\exp(-|z|),$$

$q_1 = 7.26709$, $q_2 = 11.46143$, $q_3 = 12.77026$, $q_4 = 3.20630$, $q_5 = 8.38501$, $q_6 = 5.56250$.

In the second stage, the optimal control problem is considered. In the problem, the mathematical model (35) is given. The initial state coincides with the terminal state (38).

It is necessary to find a control in the form of points in the state space (14). For synthesized control it is necessary to minimize the following quality criterion:

$$J_3 = t_f + p_1\|\mathbf{x}^f - \mathbf{x} + (t_f)\| + p_2\sum_{i=0}^{N}\int_0^{t_f}\vartheta(\varphi_i(\mathbf{x}))dt +$$

$$p_3\sum_{j=1}^{S} p_3\vartheta(\min_t |\delta_j(\mathbf{x})| - \varepsilon) \to \min_{\mathbf{x}^*}, \tag{44}$$

where $p_1 = 2$, $p_2 = 3$, $p_3 = 3$,

$$\varphi_i(\mathbf{x}) = r_i - \sqrt{(x_{i,1} - x_1)^2 + (x_{i,3} - x_3)^2}, \tag{45}$$

$i = 1, \ldots, N = 4$, $r_1 = r_2 = r_3 = r_4 = 2$, $x_{1,1} = 5$, $x_{1,3} = 0$, $x_{2,1} = 10$, $x_{2,3} = 5$, $x_{3,1} = 5$, $x_{3,3} = 10$, $x_{4,1} = 0$, $x_{4,3} = 5$,

$$\delta_j(\mathbf{x}) = \sqrt{(y_{i,1} - x_1)^2 + (y_{j,3} - x_3)^2}, \tag{46}$$

$j = 1, \ldots, S = 7$, $y_{1,1} = 5$, $y_{1,3} = -2$, $y_{2,1} = 10$, $y_{2,3} = 0$, $y_{3,1} = 12$,+ $y_{3,3} = 5$, $y_{4,1} = 10$, $y_{4,3} = 10$, $y_{5,1} = 5$, $y_{5,3} = 12$, $y_{6,1} = 0$, $y_{6,3} = 10$, $y_{7,1} = 2$, $y_{7,3} = 5$, $\varepsilon = 0.6$.

In the optimal control problem, the terminal time $t_f$ is determined by the Equation (8) with $t^+ = 14.4$, $\varepsilon_0 = 0.1$. It is necessary to find coordinates of control points on each time interval, $\Delta t = 0.8$. The desired vector includes $3M$ parameters, where

$$M = \left\lfloor \frac{t^+}{\Delta t} \right\rfloor = \left\lfloor \frac{14.4}{0.8} \right\rfloor = 18, \tag{47}$$

that is, $\mathbf{q}^* = [q_1 \ldots q_{54}]^T$. The hybrid evolutionary algorithm has found the following optimal solution:

$$\begin{gathered}
\mathbf{x}^{*,1} = [4.83910\ 1.14025\ -5.22899]^T,\ \mathbf{x}^{*,2} = [11.07056\ 6.79389\ -2.48647]^T, \\
\mathbf{x}^{*,3} = [9.19808\ 1.54674\ 15.87195]^T,\ \mathbf{x}^{*,4} = [-0.12204\ 0.12276\ -1.82381]^T, \\
\mathbf{x}^{*,5} = [-4.08347\ 2.93658\ 5.89553]^T, \mathbf{x}^{*,6} = [\ 16.72896\ 2.18022\ 2.27907]^T, \\
\mathbf{x}^{*,7} = [1.18106\ 2.56582\ 14.41088]^T,\ \mathbf{x}^{*,8} = [8.67198\ 5.78737\ -2.90409]^T, \\
\mathbf{x}^{*,9} = [8.59478\ 2.73948\ 11.33252]^T,\ \mathbf{x}^{*,10} = [-1.25924\ [-1.97448\ -1.42747]^T, \\
\mathbf{x}^{*,11} = [2.45445\ 7.42257\ -0.38164]^T,\ \mathbf{x}^{*,12} = [8.68306\ -0.78496\ 15.41667]^T, \\
\mathbf{x}^{*,13} = [0.60972\ 7.02724\ 7.66403]^T,\ \mathbf{x}^{*,141} = [-0.59975\ 0.39324\ -1.31307]^T, \\
\mathbf{x}^{*,15} = [-2.39004\ 7.95279\ 3.02003]^T,\ \mathbf{x}^{*,16} = [2.52642\ 6.69332\ 9.17356]^T \\
\mathbf{x}^{*,17} = [-0.95896\ 4.42529\ -0.36318]^T,\ \mathbf{x}^{*,18} = [-0.01193\ 5.02821\ 15.40007]^T.
\end{gathered} \tag{48}$$

For the found solution (48), the value of the quality criterion is $J_3 = 14.7010$.

In Figure 3, projections of the optimal trajectory on the horizontal plane $\{x_1; x_3\}$ are presented. Here, red circles are phase constraints described by (45), small black circles are passing areas described by (46) and small black boxes are control points (48).

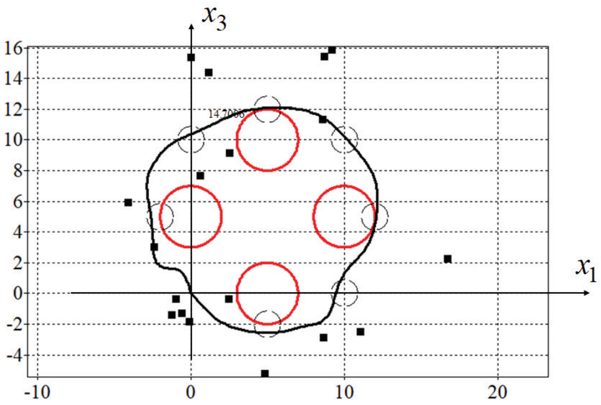


**Figure 3.** Optimal trajectory for synthesized control.

For the new adaptive synthesized control proposed in this paper, the set of initial states is determined by Equation (3) with deviation vector

$$\mathbf{\Delta}_0 = [0.2\ 0.2\ 0.2\ 0\ 0\ 0]^T. \tag{49}$$

It is necessary to find the same number of control points according to the following quality criterion:

$$J_4 = \sum_{k=1}^{K} \left( t_{f,k} + p_1 \|\mathbf{x}^f - \mathbf{x}(t_{f,i}, \mathbf{x}^{0,i}) + p_2 \sum_{i=0}^{N} \int_0^{t_f} \vartheta(\varphi_i(\mathbf{x}))dt + \right.$$

$$\left. p_3 \sum_{j=1}^{S} p_3 \vartheta(\min_t |\delta_j(\mathbf{x})| - \varepsilon) \right) \to \min_{\mathbf{x}^*}, \tag{50}$$

where $K = 7$, $t_{f,k}$ is defined by Equation (8). Other parameters of the criterion are the same as for the criterion (44).

Again, the hybrid algorithm was applied and the following optimal solution has been found:

$$\begin{gathered}
\mathbf{x}^{*,1} = [17.46361\ 1.14030\ \ -8.00000]^T,\ \mathbf{x}^{*,2} = [11.07060\ 6.79390\ \ -2.48650]^T, \\
\mathbf{x}^{*,3} = [9.19810\ 2.05890\ 15.87207]^T,\ \mathbf{x}^{*,4} = [-0.27800\ 2.51633\ \ -1.99493]^T, \\
\mathbf{x}^{*,5} = [-3.98430\ 2.27048\ 13.40976]^T,\ \mathbf{x}^{*,6} = [17.18235\ 0.26253\ 2.19246]^T, \\
\mathbf{x}^{*,7} = [-3.56784\ 3.44842\ 14.94369]^T,\ \mathbf{x}^{*,8} = [4.53881\ 2.20612\ \ -2.99328]^T, \\
\mathbf{x}^{*,9} = [9.06419\ 2.49928\ 11.30274]^T,\ \mathbf{x}^{*,10} = [-0.16333\ \ -1.88939\ \ -0.75766]^T. \\
\mathbf{x}^{*,11} = [2.17956\ 6.92983\ \ -1.06412]^T,\ \mathbf{x}^{*,12} = [10.24873\ \ -0.51465\ 5.82840]^T, \\
\mathbf{x}^{*,13} = [1.12164\ 2.84506\ 7.93804]^T,\ \mathbf{x}^{*,14} = [0.10678\ 3.23489\ \ -1.55778]^T, \\
\mathbf{x}^{*,15} = [-2.54374\ 0.99732\ 2.82005]^T,\ \mathbf{x}^{*,16} = [7.41006\ 6.49634\ 12.02799]^T, \\
\mathbf{x}^{*,17} = [-0.67510\ 4.03845\ \ -0.28527]^T,\ \mathbf{x}^{*,18} = [-0.18037\ 4.62980\ 6.89661]^T.
\end{gathered} \tag{51}$$

A value of the quality criterion (50) for one initial state $\mathbf{x}(0) = [0\ 5\ 0\ 0\ 0\ 0]^T$, is $J_4 = 15.6090$. In Figure 4, projections of the optimal trajectory on the horizontal plane $\{x_1; x_3\}$ found by the adaptive synthesized control (51) are presented.

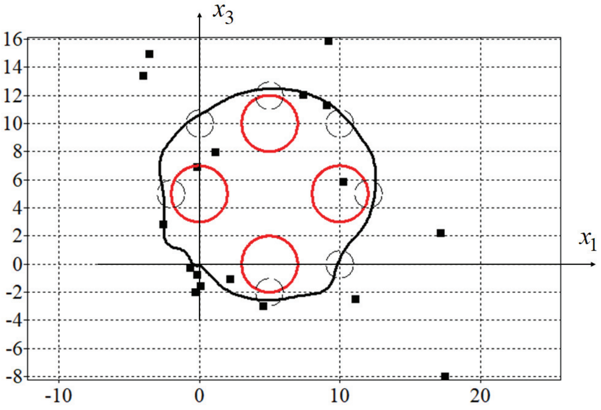


**Figure 4.** Optimal trajectory for adaptive synthesized control.

Since the initial state in the problem coincided with the terminal state, in order to force the control object to move along a closed path, mandatory conditions for passing through certain areas were added to the quality criterion. For trajectories that meet the criteria for passing through the specified areas, the value of the quality criterion will not change at $p_3 = 0$. This is seen in Figures 3 and 4 as both trajectories pass through all specified areas.

Let us check the sensitivity of the obtained solutions to random perturbations of the initial state

$$x_i(0) = x_i^0 + \beta_0(2\xi(t) - 1),\ i = 1, \dots, n = 6, \tag{52}$$

where $\xi(t)$ is a function generator of random noise, which returns a random number from interval $(0;1)$ at every call, $\beta_0$ is a constant level of noise.

In Figures 5 and 6, the optimal (in blue) and eight perturbed trajectories (in black) for $\beta_0 = 0.1$ for the solutions obtained by synthesized (Figure 5) and adaptive synthesized (Figure 6) control are presented.

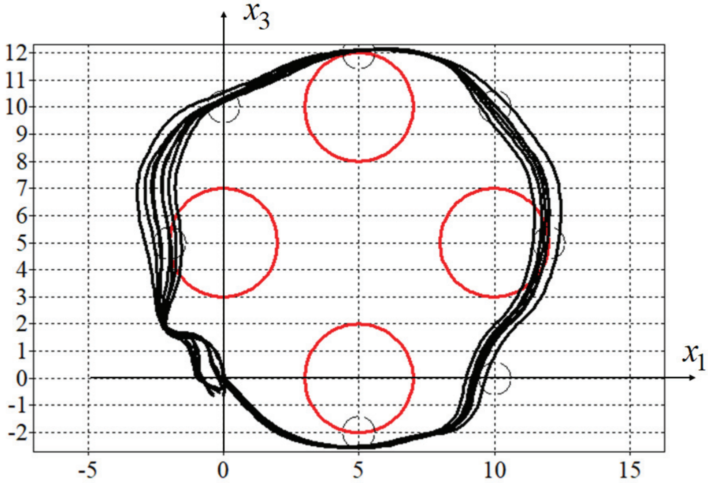


**Figure 5.** Optimal and eight disturbance trajectories of synthesized control.

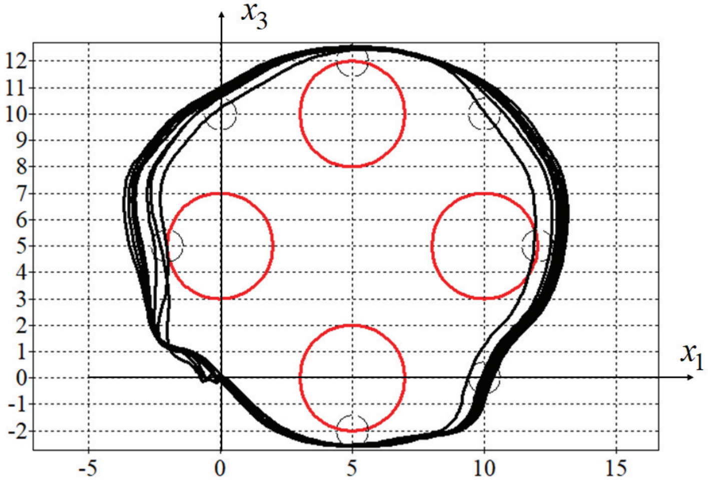


**Figure 6.** Optimal and eight disturbance trajectories of adaptive synthesized control.

For comparison, for a model (35) without stabilization systems (39), the problem of optimal control directly was solved, where control was sought in the form of a piece-wise linear function, taking into account restrictions (36).

$$u_i = \begin{cases} u_i^+, \text{ if } \tilde{u}_i > u_i^+ \\ u_i^-, \text{ if } \tilde{u}_i < u_i^- \\ \tilde{u}_i, \text{ otherwise} \end{cases}, i = 1, \dots, m = 4, \tag{53}$$

where

$$\tilde{u}_i = (q_{i+jm} - q_{i+(j-1)m})\frac{t-(j-1)\Delta t}{\Delta t} + q_{i+(j-1)m},\ i = 1,2,3,4, \quad (54)$$

$(j-1)\Delta t \leq t < j\Delta t$, $j \in \{1, \dots, K+1\}$, $q_i$ is a component of desired parameters vector, $i = 1, \dots, m(M+1)$,

$$\mathbf{q} = [q_1 \dots q_{m(M+1)}]^T. \quad (55)$$

In this work, we set the same time interval $\Delta t = 0.8$; therefore, from (15) $M = 18$, and it is necessary to find $m(M+1) = 4 \cdot 19 = 76$ parameters, $\mathbf{q} = [q_1 \dots q_{76}]^T$.

To solve the optimal control problem, the same hybrid algorithm was used. As a result, the following solution was obtained:

$$\begin{aligned}\mathbf{q} \quad = \quad & [12.57045\ -4.58471\ -2.74617\ 2.90422\ -9.26325\ -0.10990\\ & -2.63222\ 18.36841\ 0.00816\ -17.35177\ 0.00165\ 4.28718\\ & -11.81492\ 1.88489\ -8.01206\ 15.87943\ 10.84894\ 0.06505\\ & 9.65475\ 19.51903\ 2.79860\ -4.06408\ -0.88992\ 10.50507\\ & -19.19030\ 17.90240\ 12.52431\ 19.00010\ 4.76513\ -11.97648\\ & 0.00010\ 8.85464\ 2.92334\ 0.14238\ 8.60919\ 7.83194\\ & 5.74904\ -8.35383\ -3.42757\ 12.87671\ 18.58717\ 15.43057\\ & 9.06137\ 12.55621\ -1.54628\ 1.47314\ 2.40706\ 8.67602\\ & 0.00091\ -11.91236\ -19.94063\ 17.08304\ 19.92640\ -1.33145\\ & -7.77258\ 15.54094\ -19.93278\ -17.37121\ -9.31290\ 5.03257\\ & -0.90297\ -5.22021\ 0.62653\ 4.21368\ -2.04314\ -0.53192\\ & 0.09353\ 14.25213\ -0.11587\ 9.05588\ -0.03270\ 11.23667\\ & 0.03826\ -16.78047\ 0.18220\ 19.81652]^T.\end{aligned} \quad (56)$$

In Figure 7, the projection of the optimal trajectory obtained by the direct method is presented.

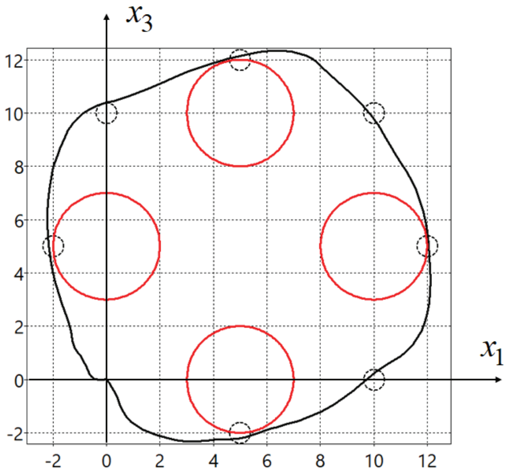


**Figure 7.** Optimal trajectory of direct control.

In Table 1 there are values of the quality criterion (44) of ten experiments for perturbed solutions obtained by the synthesized (column Synthesized), the adaptive synthesized (column Adaptive) control and the direct solution (Direct). In two last strings of the table, average values of the functionals and standard deviations for all experiments are presented.

As can be seen from Figures 5 and 6 and Table 1, the solutions obtained by adaptive synthesized control are less sensitive to perturbations of initial states than the solutions obtained by simple synthesized control or, especially, by the direct approach.

**Table 1.** Sensitivity of decisions to perturbations of initial states.

| No | Synthesized | Adaptive | Direct |
|---|---|---|---|
| 1 | 14.7651 | 15.4892 | 19.2082 |
| 2 | 20.7377 | 15.4829 | 19.8854 |
| 3 | 15.2888 | 15.6947 | 16.7706 |
| 4 | 16.9743 | 15.4935 | 16.2334 |
| 5 | 18.6159 | 16.0397 | 19.2815 |
| 6 | 19.5227 | 15.7950 | 19.3866 |
| 7 | 20.0937 | 15.4178 | 16.8263 |
| 8 | 17.5416 | 16.1424 | 23.3437 |
| 9 | 20.1225 | 17.0695 | 19.6251 |
| 10 | 19.9257 | 15.3893 | 20.8163 |
| Av | 18.3588 | 15.8014 | 19.1377 |
| SD | 2.1234 | 0.5167 | 2.1285 |

## 5. Conclusions

A new method for solving the problem of optimal control in the class of implemented functions, an adaptive synthesized control principle is presented. Unlike synthesized control, the new method takes into account the perturbations of the initial state when solving the optimal control problem. Therefore, the value of the quality criterion is calculated as the sum of the quality criterion values for the different initial states. As a result of this approach, a solution is chosen in such a way that for the origin initial state it may not give the best quality criterion value, but in the case of disturbances of the initial state, the quality criterion value changes slightly.

## 6. Discussion

Obtaining a solution based on replacing the optimal solution is less optimal, but also less sensitive to disturbances. At first glance, this seems obvious and can be applied to any method of solving the optimal control problem. However, this is not the case. A direct solution to the optimal control problem results in control in the form of a time function and an open-loop control system. Perturbation of the initial conditions for such a system gives large variations in quality criterion values, which cannot be reliably estimated from the average value due to the large variance.

The synthesized control method firstly makes the control object stable relative to some equilibrium point in the state space. This means that the perturbed and unperturbed trajectories at each point in time move towards a stable equilibrium point. The adaptive synthesized control method sets the positions of the equilibrium points so that all disturbed trajectories are located in some tube that does not violate phase constraints whenever possible.

In the future, when using the adaptive synthesized control method, it is necessary to assess the required size of the initial state region and reduce the number of initial state points, since this significantly increases the time for finding the optimal solution.

**Author Contributions:** Conceptualization, E.S.; methodology, A.D. and E.S.; software, A.D. and E.S.; validation, E.S.; formal analysis, A.D.; investigation, E.S.; writing—original draft preparation, A.D. and E.S.; writing—review and editing, E.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported by the Ministry of Science and Higher Education of the Russian Federation, project No. 075-15-2020-799.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Egerstedt, M. Motion Planning and Control of Mobile Robots. Ph.D. Thesis, Royal Institute of Technology, Stockholm, Sweden, 2000.
2. Walsh, G.; Tilbury, D.; Sastry, S.; Murray, R.; Laumond, J.P. Stabilization of trajectories for systems with nonholonomic constraints. *IEEE Trans. Autom. Control* **1994**, *39*, 216–222. [CrossRef]
3. Samir, A.; Hammad, A.; Hafez, A.; Mansour, H. Quadcopter Trajectory Tracking Control using State-Feedback Control with Integral Action. *Int. J. Comput. Appl.* **2017**, *168*, 1–7. [CrossRef]
4. Allagui, N.Y.; Abid, D.B.; Derbel, N. Autonomous navigation of mobile robot with combined fractional order PI and fuzzy logic controllers. In Proceedings of the 2019 16th International Multi-Conference on Systems, Signals & Devices (SSD), Istanbul, Turkey, 21–24 March 2019; pp. 78–83. [CrossRef]
5. Chen, B.; Cao, Y.; Feng, Y. Research on Trajectory Tracking Control of Non-holonomic Wheeled Robot Using Backstepping Adaptive PI Controller. In Proceedings of the 2022 7th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS), Tianjin, China, 1–3 July 2022; pp. 7–12. [CrossRef]
6. Karnani, C.; Raza, S.; Asif, A.; Ilyas, M. Adaptive Control Algorithm for Trajectory Tracking of Underactuated Unmanned Surface Vehicle (UUSV). *J. Robot.* **2023**, *2023, 4820479*. [CrossRef]
7. Nguyen, A.T.; Nguyen, X.-M.; Hong, S.-K. Quadcopter Adaptive Trajectory Tracking Control: A New Approach via Backstepping Technique. *Appl. Sci.* **2019**, *3873*, 3873. [CrossRef]
8. Lee, E.B.; Marcus, L. *Foundations of Optimal Control Theory*; Robert & Krieger Publishing Company: Malabar, Florida, 1967; 576p.
9. Pontryagin, L.S.; Boltyanskii, V.G.; Gamkrelidze, R.V.; Mishchenko, E.F. *The Mathematical Theory of Optimal Process. L. S. Pontryagin, Selected Works*; Gordon and Breach Science Publishers: New York, NY, USA; London, UK; Paris, France; Montreux, Switzerland; Tokyo, Japan, 1985; Volume 4, 360p.
10. Shmalko, E., Diveev, A. Extended Statement of the Optimal Control Problem and Machine Learning Approach to Its Solution. *Math. Probl. Eng.* **2022**, *2022*, 1932520. [CrossRef]
11. Diveev, A.; Shmalko, E.Y.; Serebrenny, V.V.; Zentay, P. Fundamentals of synthesized optimal control. *Mathematics* **2020**, *9*, 21. [CrossRef]
12. Shmalko, E.Y. Feasibility of Synthesized Optimal Control Approach on Model of Robotic System with Uncertainties. In *Electromechanics and Robotics*; Smart Innovation, Systems and Technologies; Ronzhin, A., Shishlakov, V., Eds.; Springer: Singapore, 2022; Volume 232.
13. Shmalko, E. Computational Approach to Optimal Control in Applied Robotics. In *Frontiers in Robotics and Electromechanics. Smart Innovation, Systems and Technologies*; Ronzhin, A., Pshikhopov, V., Eds.; Springer: Singapore, 2023; Volume 329.
14. Diveev, A.; Shmalko, E.Y. Stability of the Optimal Control Problem Solution. In Proceedings of the 8th International Conference on Control, Decision and Information Technologies, CoDIT, Istanbul, Turkey, 17–20 May 2022; pp. 33–38.
15. Simon J.D.; Mitter S.K. A theory of modal control. *Inf. Control* **1968**, *13*, 316–353. [CrossRef]
16. Jouffroy, J.; Lottin, J. Integrator backstepping using contraction theory: A brief methodological note. In Proceedings of the 15th IFAC World Congress, Barcelona, Spain, 21–26 July 2002.
17. Zhou H.; Liu, Z. Vehicle Yaw Stability-Control System Design Based on Sliding Mode and Backstepping Contol Approach. *IEEE Trans. Veh. Technol.* **2010**, *59*, 3674–3678. [CrossRef]
18. Febsya, M.R.; Ardhi, R.; Widyotriatmo, A.; Nazaruddin, Y.Y. Design Control of Forward Motion of an Autonomous Truck-Trailer using Lyapunov Stability Approach. In Proceedings of the 2019 6th International Conference on Instrumentation, Control, and Automation (ICA), Bandung, Indonesia, 31 July–2 August 2019; pp. 65–70. [CrossRef]
19. Zihao, S.; Bin, W.; Ting, Z. Trajectory Tracking Control of a Spherical Robot Based on Adaptive PID Algorithm. In Proceedings of the 2019 Chinese Control And Decision Conference (CCDC), Nanchang, China, 3–5 June 2019; pp. 5171–5175. [CrossRef]
20. Liu, J.; Song, X.; Gao, S.; Chen, C.; Liu K.; Li, T. Research on Horizontal Path Tracking Control of a Biomimetic Robotic Fish. In Proceedings of the 2022 International Conference on Mechanical and Electronics Engineering (ICMEE), Xi'an, China, 21–23 October 2022; pp. 100–105. [CrossRef]
21. Duriez, T.; Brunton, S.L.; Noack, B.R. *Machine Learning Control–Taming Nonlinear Dynamics and Turbulence*; Springer International Publishing: Cham, Switzerland, 2017.
22. Awange, J.L.; Paláncz, B.; Lewis, R.H.; Völgyesi, L. Symbolic Regression. In *Mathematical Geosciences*; Springer: Cham, Switzerland, 2018.
23. Diveev, A.I.; Shmalko, E.Y. *Machine Learning Control by Symbolic Regression*; Springer: Cham, Switzerland, 2021; 155p.
24. Diveev, A.; Shmalko, E. Machine Learning Feedback Control Approach Based on Symbolic Regression for Robotic Systems. *Mathematics* **2022**, *10*, 4100. [CrossRef]
25. Davis, L. *Handbook of Genetic Algorithms*; Van Nostrand Reinhold: New York, NY, USA, 1991.
26. Eberhardt, R.C.; Kennedy, J.A. Particle Swarm Optimization. In Proceedings of the IEEE International Conference on Neural Networks, Piscataway, NJ, USA, 27 November–1 December 1995; pp. 1942–1948.
27. Eltamaly, A. A Novel Strategy for Optimal PSO Control Parameters Determination for PV Energy Systems. *Sustainability* **2021**, *13*, 1008. [CrossRef]

28. Salehpour, E.; Vahidi, J.; Hosseinzadeh H. Solving optimal control problems by PSO-SVM. *Comput. Methods Differ. Equ.* **2018**, *6*, 312–325
29. Mirjalili, S.; Mirjalil, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]

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

mathematics

MDPI

*Article*

# Exploratory Landscape Validation for Bayesian Optimization Algorithms

**Taleh Agasiev * and Anatoly Karpenko**

Department of Computer-Aided Design Systems, Bauman Moscow State Technical University,
Moscow 105005, Russia; apkarpenko@mail.ru
* Correspondence: agtaleh@mail.ru

**Abstract:** Bayesian optimization algorithms are widely used for solving problems with a high computational complexity in terms of objective function evaluation. The efficiency of Bayesian optimization is strongly dependent on the quality of the surrogate models of an objective function, which are built and refined at each iteration. The quality of surrogate models, and hence the performance of an optimization algorithm, can be greatly improved by selecting the appropriate hyperparameter values of the approximation algorithm. The common approach to finding good hyperparameter values for each iteration of Bayesian optimization is to build surrogate models with different hyperparameter values and choose the best one based on some estimation of the approximation error, for example, a cross-validation score. Building multiple surrogate models for each iteration of Bayesian optimization is computationally demanding and significantly increases the time required to solve an optimization problem. This paper suggests a new approach, called exploratory landscape validation, to find good hyperparameter values with less computational effort. Exploratory landscape validation metrics can be used to predict the best hyperparameter values, which can improve both the quality of the solutions found by Bayesian optimization and the time needed to solve problems.

**Keywords:** Bayesian optimization; Gaussian process; surrogate modeling; hyperparameter tuning; exploratory landscape analysis; exploratory landscape validation; variability map of objective function

**MSC:** 90C26; 90C56; 90C59

**Citation:** Agasiev, T.; Karpenko, A. Exploratory Landscape Validation for Bayesian Optimization Algorithms. *Mathematics* **2024**, *12*, 426. https://doi.org/10.3390/math12030426

Academic Editor: Liliya Demidova

Received: 25 December 2023
Revised: 20 January 2024
Accepted: 23 January 2024
Published: 28 January 2024

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

## 1. Introduction

Continuous optimization problems with high computational complexity in terms of objective functions arise in many fields of engineering, material design, automatic machine learning, and others [1]. In real-world optimization problems, evaluating the value of an objective function requires time-consuming computational experiments or simulations. The time needed to solve an optimization problem is primarily defined by the total number of objective function evaluations performed by the optimization algorithm. When solving computationally expensive optimization problems, the main termination criterion for the algorithm is the maximum number of objective evaluations, which is usually called the computational budget [2].

Problems with a limited computational budget are best solved by optimization algorithms that utilize surrogate models of the objective function [2]. The most promising candidates for objective evaluation are identified by refining and exploring surrogate models at every optimization iteration. Hence, both the quality of surrogate models and the parameters of the model exploration procedure can have an impact on the optimization algorithm's efficiency. Many studies have focused on tuning the parameters of the model exploration procedure, specifically the type and the parameters of the acquisition function, as well as the parameters of the algorithm for optimizing it [3,4]. This article focuses on ways to improve the efficiency of optimization algorithms by increasing the quality of surrogate models.

Surrogate models are usually built by approximation algorithms with tunable hyperparameters. Numerous studies have been conducted to find the best hyperparameter values that maximize the accuracy of surrogate models and, hence, the optimization algorithm's efficiency [5–7]. The process of finding such hyperparameter values is called hyperparameter tuning. Here are some examples of hyperparameters being tuned for different types of surrogate models:

- Degree of polynomial regression models;
- Size of the hidden layer for models based on neural networks;
- Kernel type and parameters for models based on Gaussian process (GP) regression;
- Number of trees and tree depth for models based on random forest regression.

This article considers the tuning of kernel parameters for models based on GP regression, which includes, for example, the scale mixture parameter $\alpha$ of the rational quadratic kernel or the $\nu$ parameter of the Matern kernel [8].

In order to perform hyperparameter tuning, the hyperparameter efficiency metric is defined based on the quality estimate of a surrogate model built with a given hyperparameter vector. The model quality is usually measured on a test sample of points using some error approximation metric, such as mean absolute error, mean squared error, $R^2$ score, etc. [9,10]. In cases of a strict computational budget, maintaining a separate test sample by evaluating a costly objective function is not sufficient. To increase the reliability of hyperparameter efficiency estimates, resampling procedures are used, which involve the building of multiple models for each hyperparameter vector, for example, the well-known cross-validation procedure [10]. Using cross-validation for hyperparameter efficiency estimation significantly increases the computational cost of hyperparameter tuning. This article introduces a new approach for estimating hyperparameter efficiency called exploratory landscape validation, along with new efficiency metrics that require less computational effort, and that, in some cases, outperform cross-validation in terms of solution quality. One of the metrics is based on a so-called ranking preservation indicator [11], calculated on an extended training sample. Another metric is evaluated by comparing the variability maps of an objective function [12] constructed for a training sample and a surrogate model.

Much of the recent research on the parameter tuning of optimization algorithms, which also includes the hyperparameter tuning of surrogate models, has focused on using exploratory landscape analysis (ELA) algorithms to estimate the characteristic features of optimization problems [13–15]. The hyperparameter prediction process is based on the following idea. Given the ELA feature vectors, which incorporate the specifics of problems, and the best hyperparameter values found for those problems, a machine learning (ML) algorithm is used to build a tuning model to predict the best hyperparameter values or the best approximation algorithm [12,16,17]. According to such an ELA-ML approach, the tuning model is then used to identify hyperparameter values that are reasonably effective when solving similar problems based on the feature vectors of those problems. In [16], for example, the authors constructed tuning models that are based on different classification algorithms to predict the best surrogate modelling algorithm using the ELA features of the problem. Despite the fact that additional computational effort is required to collect the training data and build a tuning model, the increase in efficiency of the optimization algorithm outweighs the computational costs in a long-term perspective. This article also explores the benefits of using tuning models that are built using the proposed metrics for finding effective hyperparameter values.

The rest of the article is organized in the following way. Section 2 presents a statement for a global continuous optimization problem, describes the canonical form of the Bayesian optimization algorithm, and formalizes the hyperparameter tuning and prediction problems. In Section 3, a new approach to the quality estimation of surrogate models, called landscape validation, is introduced, and includes a variety of criteria for selecting and predicting the best hyperparameter values. Section 4 starts with the general setup for computational experiments, followed by a description of the experiments performed with the hyperparameter tuning and hyperparameter prediction approaches based on the proposed hyperparameter efficiency metrics; then, the experimental results are discussed.

## 2. Bayesian Optimization with Hyperparameter Tuning and Prediction

The following statement of the continuous global optimization problem $q$ is considered:

$$\underset{X \in D_X \subset \mathbb{R}^{|X|}}{min} f(X) = f(X^*) = f^*, \tag{1}$$

where $X = \left(x_1, ..., x_{|X|}\right)$ is the vector of continuous variables of size $|X|$; $f(X) \in \mathbb{R}^1$ is a scalar objective function; $D_X$ is a convex region of the permissible variables' values; $X^*$ is the vector of the variables' values for which the objective function has the minimal value $f^*$. It is assumed that the region $D_X$ is formed by the set of inequalities $x_i^- \leq x_i \leq x_i^+$, $i \in (1 : |X|)$, where $x_i^-$ and $x_i^+$ are the lower and upper bounds of the $i$-th variable, respectively. The problem (1) is referred to as the base optimization problem.

Due to the high computational complexity of the objective function, the total number of objective evaluations allowed for solving the problem (1) is limited to $N_{max}$, which is called the computational budget of the problem. When solving computationally expensive base problems, common practice is to build and explore surrogate models $\hat{f}(X)$ of the objective function $f(X)$. By using a surrogate model $\hat{f}(X)$, promising areas of the search region $D_X$ can be located approximately without utilizing the budget $N_{max}$. The algorithm for building surrogate models usually has tunable hyperparameters, the vector of the values of which is denoted as $P = \left(p_1, ..., p_{|P|}\right)$.

In this section, the Bayesian optimization algorithm is described and the problem statements for hyperparameter tuning and hyperparameter prediction for the surrogate modeling algorithm are formulated. The aim of this section is to outline and formalize common approaches for solving computationally expensive optimization problems, on the basis of which, new approaches to the hyperparameter efficiency estimation will be presented in the next section.

### 2.1. Bayesian Optimization Algorithm

In contrast to other surrogate-based optimization algorithms, Bayesian optimization algorithms use surrogate models that are based on Gaussian process (GP) approximation. In GP models, each point $X$ of the search region $D_X$ is associated with a normal distribution of the predicted objective values $\mathcal{N}\left(\mu_{\hat{f}}(X), \sigma_{\hat{f}}^2(X)\right)$, where $\mu_{\hat{f}}$ is the mean and $\sigma_{\hat{f}}$ is the standard deviation of $\hat{f}$ values. Having the probability distribution over $\hat{f}$ allows for selection of promising points for the objective function evaluation based on both predicted objective values and uncertainty estimates of those predictions. Formally, promising point selection is defined as the maximization problem of a so-called acquisition function. Hence, the tunable parameters of a Bayesian optimization algorithm include the type and parameters of the acquisition function, as well as the hyperparameters of the GP approximation algorithm, which are the type and parameters of a covariance function, also referred to as a kernel [5].

The Bayesian optimization algorithm includes the following steps (Figure 1):

1. Generate the initial sample $L^0 = \left\{\left(X^i, f^i\right), \ i \in (1 : N_{init})\right\}$ for training the first GP model, where $N_{init} < N_{max}$ is the initial sample size. Points $X^i \in D_X$ are chosen either randomly or according to one of the designs of the experiment algorithm, e.g., the Latin hypercube sampling (LHS) algorithm [18];
2. Perform iterations $r \in (1 : N_{iter})$, where $N_{iter} = N_{max} - N_{init}$ as follows:
    a. Build the surrogate model $\hat{f}^r(X)$ using the current training sample $L^r$ and the vector $P$ of the hyperparameter values;

b. Select the next point $X^{i+1}$ for the objective function evaluation by optimizing an acquisition function, e.g., by minimizing the lower confidence bound (LCB) [19] function as follows:

$$X^{i+1} = \underset{X \in D_X \subset \mathbb{R}^{|X|}}{\arg\min} \ \mu_{\hat{f}}(X) - \kappa\sigma^2_{\hat{f}}(X), \tag{2}$$

where $\kappa$ is a tunable parameter;

c. Evaluate the objective function for point $X^{i+1}$ to obtain the corresponding value $f^{i+1}$ and extend the training sample $L^{r+1} = L^r \cup \left(X^{i+1}, f^{i+1}\right)$.

3. The point that has the minimal corresponding objective value $(X^*, f^*) \in L^{N_{iter}}$ is considered as the solution of the base problem (1).

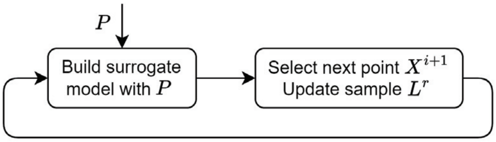

$P$
Build surrogate model with $P$
Select next point $X^{i+1}$ Update sample $L^r$

**Figure 1.** Bayesian optimization with fixed hyperparameter values.

Although the formal optimality of the best-found point $X^*$ is not guaranteed in any sense, the same notation as in the problem definition (1) is used for simplicity. The rest of the article focuses on ways to improve the optimization algorithm's efficiency by tuning the GP hyperparameters.

### 2.2. Hyperparameter Tuning for a Bayesian Optimization Algorithm

The efficiency of Bayesian optimization can be improved by selecting, at each iteration, the vector $P^*$ that is the best for the training sample $L^r$ according to the hyperparameter efficiency metric $\phi(L^r, P)$. Given the set of allowed hyperparameter values $D_P \subset \mathbb{R}^{|P|}$, the best vector $P^*$ is found by solving the following hyperparameter optimization problem at step 2a (Figure 2):

$$\underset{P \in D_P \subset \mathbb{R}^{|P|}}{opt} \quad \phi(L^r, P) = \phi(L^r, P^*), \tag{3}$$

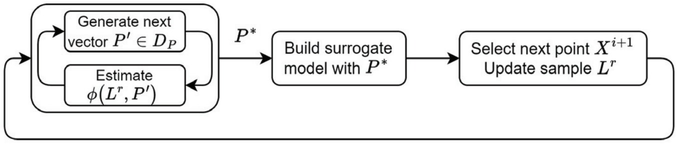

Generate next vector $P' \in D_P$
Estimate $\phi(L^r, P')$
$P^*$
Build surrogate model with $P^*$
Select next point $X^{i+1}$ Update sample $L^r$

**Figure 2.** Bayesian optimization with hyperparameter tuning.

The hyperparameter efficiency metric $\phi(L^r, P)$ is commonly defined based on approximation accuracy metrics, e.g., mean squared error, on a test sample.

In most cases, the size of sample $L$ in Bayesian optimization is already not sufficient for the search space dimension $|X|$ due to a very limited budget $N_{max}$. Since it would not be practical either to split sample $L$ into train and test parts or to spend the budget $N_{max}$ on collecting and updating a separate test sample, the cross-validation procedure is used to estimate the average accuracy of the surrogate models built with the given vector $P$. The corresponding hyperparameter efficiency metric $\phi_{CV}(L^r, P)$ is calculated as follows:

$$\phi_{CV}(L^r, P) = \frac{1}{K}\sum_{k=1}^{K} R^2(L^r_k, P) = \frac{1}{K}\sum_{k=1}^{K}\left[1 - \frac{\sum\left(f^i_k - \hat{f}^i_k\right)^2}{\sum\left(f^i_k - \overline{f}_k\right)^2}\right], \tag{4}$$

where $K$ is the total number of cross-validation folds, $R^2\left(L_k^r, P\right)$ is the coefficient of determination, $L_k^r \subset L^r$ is the test sample for the $k$-th fold, $f_k^i$ and $\hat{f}_k^i$ are the known and predicted objective values at the $k$-th fold correspondingly, and $\overline{f}_k$ is the mean of the known objective values at the $k$-th fold.

Using the metric $\phi_{CV}(L^r, P)$ when solving the problem (3) requires building as many GP models for each vector $P$ as there are cross-validation iterations. Since building a GP model has $O\left(|L|^3\right)$ complexity, where $|L|$ is the training sample size, the hyperparameter tuning process may become time consuming and unprofitable. In this article, it is proposed to develop new hyperparameter efficiency metrics that reduce the computational complexity of solving the problem (3) while maintaining the accuracy of solutions comparable to the metric $\phi_{CV}(L^r, P)$.

### *2.3. Hyperparameter Prediction for a Bayesian Optimization Algorithm*

Solving the problem (3) from scratch at each iteration of the Bayesian optimization is a straightforward but time-consuming approach to hyperparameter tuning. Modern approaches to parameter tuning or the selection of optimization algorithms involve combining ELA and ML to predict the most efficient algorithm or parameters for solving the base problem (1) [16,17]. The ELA-ML approach relies on the assumption that the best optimization algorithm or parameter values for solving similar optimization problems will also be nearly identical. The similarity of optimization problems can be estimated by a similarity measure between the corresponding ELA feature vectors.

Although many of the known ELA-ML approaches are developed for the best optimization algorithm selection, the same idea can be adapted for hyperparameter tuning the following way. The process of solving the problem (3) is divided into separate phases as is shown in Figure 3.

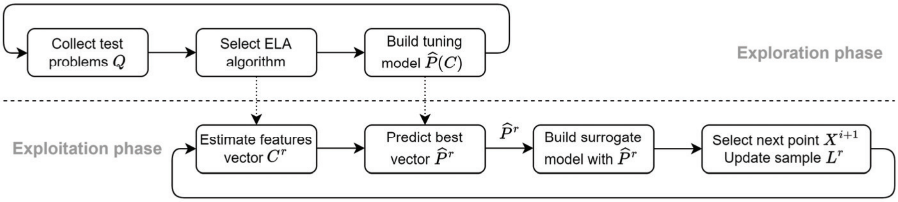

Collect test problems $Q$
Select ELA algorithm
Build tuning model $\widehat{P}(C)$
Exploration phase
Exploitation phase
Estimate features vector $C^r$
Predict best vector $\widehat{P}^r$
$\widehat{P}^r$
Build surrogate model with $\widehat{P}^r$
Select next point $X^{i+1}$ Update sample $L^r$

**Figure 3.** Bayesian optimization with hyperparameter prediction.

Exploration phase—before solving the base problem (1):

1. Define a representative set $Q = \{q_i, i \in (1 : M_Q)\}$ of test optimization problems $q_i$, where $M_Q$ is the number of test problems;
2. Generate random training samples $\{L_{i,j},\ j \in [1 : M_L]\}$ of different sizes $\left|L_{i,j}\right| \le N_{max}$, where $M_L$ is the number of samples for each problem $q_i$;
3. For each sample $L_{i,j}$, calculate the vector of ELA features $C_{i,j}$ and find the vector $P_{i,j}^*$ that is the best according to some metric $\phi\left(L_{i,j}, P\right)$ by solving the problem (3);
4. Build a tuning model $\hat{P}(C)$ using the set of known pairs $\left\{C_{i,j}, P_{i,j}^*\right\}$, the total number of which is $M_Q \times M_L$.

Exploitation phase—at step 2a of the Bayesian optimization algorithm:

1. Calculate the vector of features $C^r$ using the current training sample $L^r$;
2. Predict the best hyperparameter values $\hat{P}^r = \hat{P}(C^r)$ using the tuning model built at step 4 of the exploration stage;
3. Build the surrogate model $\hat{f}^r(X)$ using the current training sample $L^r$ and the vector $\hat{P}^r$ of the predicted hyperparameter values.

The efficiency of hyperparameter prediction is generally defined by the set $Q$, the feature vector $C$, and the metric $\phi(L, P)$. The exploration phase involves most of the computational expenses of solving the problem (3), which makes it practical to apply even costly metrics, such as the metric $\phi_{CV}(L, P)$. As illustrated in Figure 3, the test problems set $Q$ can be refined and extended permanently, even during the exploitation phase. New metrics are proposed in Section 3 to speed up the quality estimation of surrogate models for hyperparameter tuning and prediction. The efficiency of hyperparameter prediction with the metric $\phi_{CV}(L, P)$ and with the proposed new metrics will be examined in Section 4. The scope of the article does not include formation of a representative set $Q$ of the test problems, or identification of the most suitable vector $C$ of ELA features.

## 3. Exploratory Landscape Validation

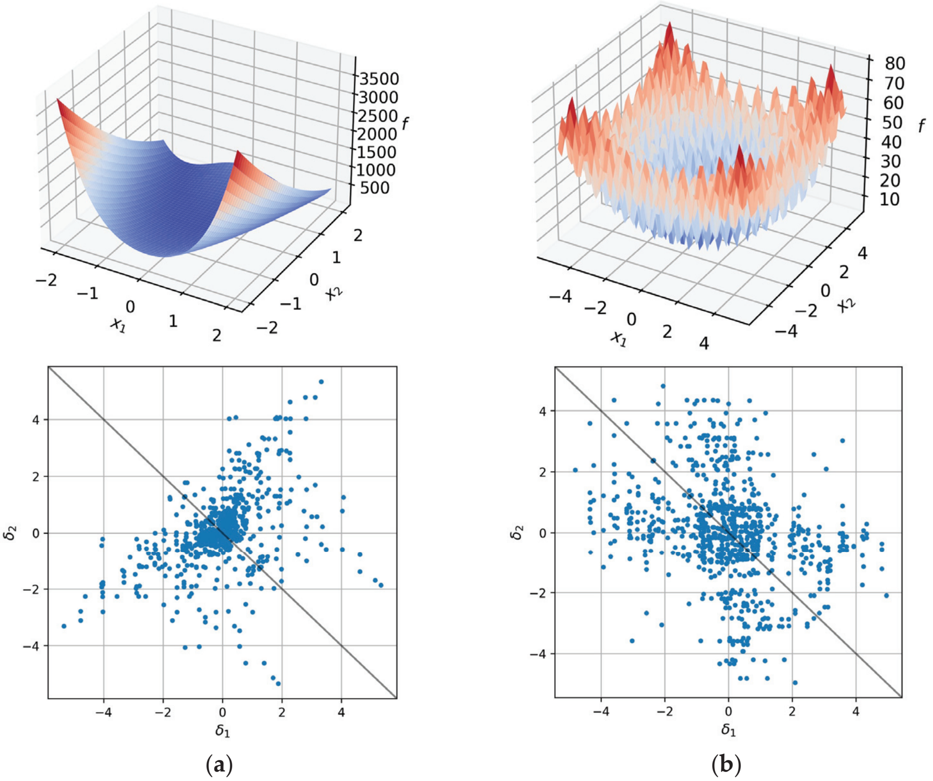
This section presents new metrics for estimating the efficiency of vector $P$ on a given sample $L$. The new metrics evaluate not only the approximation accuracy of the surrogate models, as the metric $\phi_{CV}(L, P)$ does, but also certain properties of them that could be crucial for the optimization algorithm's performance. Since the metrics are developed on the basis of an ELA algorithm, namely the variability map (VM) algorithm [12], we refer to them as exploratory landscape validation (ELV) metrics.

There are known metrics for estimating the quality of surrogate models that are not based on approximation accuracy and can, hence, be considered ELV metrics. For example, the ranking preservation (RP) metric [11], which we will refer to as $\phi_{RP}(L, P)$, estimates the level of preservation of comparative relations between pairs of objective values approximated by the given surrogate model. According to the authors, the metric $\phi_{RP}(L, P)$ is calculated using an independent test sample of points, that is not practical to collect in the context of a strict budget for objective function evaluations. At the same time, when using interpolating approximation algorithms, such as GP, the metric value calculated for the training sample will be close to the maximum possible value. To be able to use the metric $\phi_{RP}(L, P)$ for hyperparameter tuning, an algorithm for generating an extended training sample $L_{ext}$ is introduced in this paper. The proposed metrics, which are based on extended samples $L_{ext}$, are referred to as $\phi_{RP}(L_{ext}, P)$ and $\phi_{AD}(L_{ext}, P)$. The metric $\phi_{AD}(L_{ext}, P)$ is unique in the sense that it estimates the quality of surrogate models by directly comparing the VMs of those models with the VM of the training sample.

Section 3.1 starts with an improved algorithm for building a VM, followed by a discussion of VM quality assessment. Next, in Section 3.2, an algorithm for constructing an extended variability map (EVM) from a VM is suggested to enhance the reliability of landscape validation by extending the training sample. Based on that, new EVM-based landscape validation metrics $\phi_{RP}(L_{ext}, P)$ and $\phi_{AD}(L_{ext}, P)$ are proposed and explained in Section 3.3. Thus, the proposed ELV approach consists of the process of building VMs, extending VMs, and calculating values of one of the suggested metrics based on an extended training sample when solving the problem (3).

### *3.1. Variability Map of an Objective Function*

VMs were first proposed to estimate ELA features of the function $f(X)$ based on a given sample $L$ [12]. A VM is built by collecting a set of triples $T = \{t_j, j \in (1 : |T|)\}$ from the sample $L$, where $|T|$ is the total number of triples. Each triple $t_j = \left(i_1^j, i_2^j, i_3^j\right)$ is composed of points $\left(X^{i_1}, X^{i_2}, X^{i_3}\right)$ that are neighboring in $X$ space. The triples for each point of sample $L$ are collected from all of its neighboring points. Two points are considered neighbors if the distance between them is less than the maximum distance between pairs of the closest points in sample $L$, taken with some correction factor. Using the corresponding objective values $\left(f^{i_1}, f^{i_2}, f^{i_3}\right)$ of the collected triples, the pair of increment values $\left(\delta_1^j, \delta_2^j\right)$ is calculated. The values $\delta_1^j, \delta_2^j$ characterize increments of the objective function between pairs of points $(i_1, i_2)$ and $(i_2, i_3)$, respectively. The set of increment values $\left\{\left(\delta_1^j, \delta_2^j\right), j \in (1 : |T|)\right\}$ then forms the VM and can be visually represented as a cloud of points on the plane $0\delta_1\delta_2$, as shown in Figure 4.



(**a**) (**b**)

**Figure 4.** Landscape plots and corresponding VMs for (**a**) Rosenbrock and (**b**) Rastrigin test optimization functions. VM's increment values are represented by blue dots.

In cases where sample $L$ is irregular, such as when point density varies a great deal, using a max–min distance estimate for triple collecting, as suggested in [12], may not be a sustainable strategy. We propose a new algorithm for VM building based on angular ranges, which works better with irregular samples as well.

The new algorithm consists of the steps listed below:

1. Select a random point $X^{i_2}$ from sample $L$ and find the closest point $X^{i_3}$:

$$i_3 = \underset{i_3 \in [1:|L|], i_3 \neq i_2}{\arg\min} \; d_{i_2,i_3}, \tag{5}$$

   where $d_{i_2,i_3} = \left|\left|X^{i_3} - X^{i_2}\right|\right|$ is the Euclidean distance between the points;

2. Find all the points $\left\{X^k, k \in [1:|L|], k \neq i_2,\ k \neq i_3\right\}$ that satisfy the conditions:

$$\begin{gathered} d_{k,i_2} < d_{k,i_3}, \\ d_{k,i_2} < \overline{d}_{i_2}, \\ \alpha_{k,i_2,i_3} \geq \pi/2, \end{gathered} \tag{6}$$

   where $\overline{d}_{i_2}$ is the mean distance from $i_2$-th point to all the other points, $\alpha_{k,i_2,i_3}$ is the angle formed by $X^k$, $X^{i_2}$, $X^{i_3}$ points in the $X$ space. The first condition is a quick check that reduces the number of points for which the angle needs to be calculated;

3. For each angle range $[\alpha^-, \alpha^+]$ from the set of ranges $\{[90,120], [120,150], [150,180]\}$, that is formed by splitting the range $(90,180)$ into three equal ranges, do the following:

a. find a point $X^{i_1} \in \left\{X^k\right\}$ that has the minimal distance $d_{i_1,i_2}$ in that angle range:

$$\begin{aligned} X^{i_1} &= \underset{X^{i_1} \in \{X^k\}}{\arg\min}\, d_{i_1,i_2}, \\ \alpha^- &< \alpha_{i_1,i_2,i_3} \leq \alpha^+; \end{aligned} \tag{7}$$

b. extend the set of triples $T$ with a new one $t = (i_1, i_2, i_3)$;

c. increase the distances $d_{i_1,i_2}$ and $d_{i_2,i_3}$ by a factor of 2 so that the other points are considered if $i_2$ is randomly selected in the next iterations.

4. If the maximum number of triples $|T|$ is not reached, move to step 1.

The presented algorithm has several tunable parameters. The set of angle ranges is formed by splitting the range of allowed angles $(90, 180)$ into three equal ranges. It is recommended to use a lower bound of at least 90 degrees as triples with smaller angles may not be informative for landscape analysis purposes [12]. At the same time the angle formed by a triple of points in the $X$ space is limited to 180 degrees. The number of splits determines the maximum number of triples to be collected for each considered point. It is recommended to increase the number of splits for bigger dimensions $|X|$. Both the lower bound of the allowed angle range and the number of splits are tunable parameters of the algorithm. The maximum number of triples $|T|$ is another parameter of the algorithm that is usually set as the multiple of the total number of points $|L|$, e.g., by multiplying $|L|$ by the number of angle ranges. Increasing the distance between the points in step 3c helps to avoid selecting the same points for the next triples. It is recommended to multiply the distance between $X^{i_1}$ and $X^{i_2}$ at least by 1.5 and completely remove $X^{i_3}$ from the $X^{i_2}$ neighbors (e.g., by setting the corresponding distance value to infinity).

In Figure 5, the difference between the old and the new algorithm for collecting VM triples based on an irregular sample of points is illustrated. The training sample of size $|L| = 30$ is represented by black dots in the $X$ space with dimension $|X| = 2$. The points of the collected triples are connected by multicolor lines. It can be seen from the figure that the new algorithm provides better "coverage" of the $X$ space with the same number of triples. In Figure 5b, triples form connections between more distant points and fill the gaps in $X$ space caused by an irregular sample structure. The new algorithm generates triples with a lesser number of shared pairs of points, which is better for the landscape validation algorithm, further described below, and which is based on extended training samples.

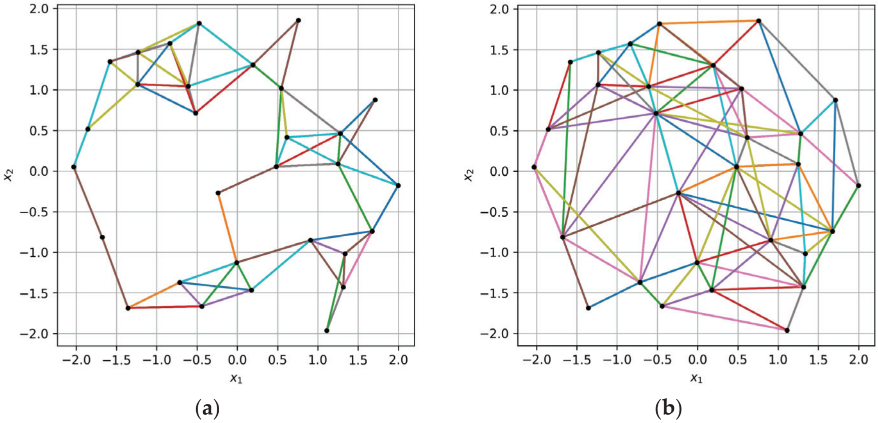


**Figure 5.** Triples of points collected (**a**) based on the max–min distance between the points and (**b**) with the proposed algorithm based on angular ranges. Black dots represent the training sample points in $X$ space ($|X| = 2$), and multicolor lines connect the points of the collected triples.

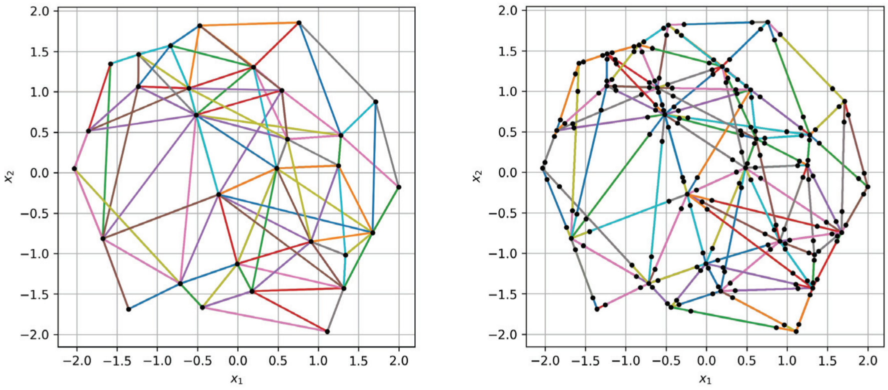
The quality of a VM-building algorithm can be formally measured by the average distance between the points of the triples and average angles formed by those points. Formalizing the "coverage" quality and analyzing the connection between the quality of the VM and the efficiency of hyperparameter tuning or prediction is beyond the scope of this article.

### *3.2. Extended Variability Map of an Objective Function*

A training sample-based landscape validation is not suitable for interpolation techniques like GP; at the same time, it is not practical to evaluate an expensive objective function for additional points in such cases. We propose the following method for extending the training sample $L$ using an extended version of a VM, built on that sample.

1. Collect the set of triples $T = \{t_j, j \in (1 : |T|)\}$ as it was described in Section 3.1;
2. For each triple $t \in T$, where $t = (i_1, i_2, i_3)$, perform the following steps:
    a. split the vector $X^{i_1}X^{i_2}$ into three parts by the points $X^{k_1}$ and $X^{k_2}$, so that the points $X^{i_1}, X^{k_1},\ X^{k_2}, X^{i_2}$ are arranged in the given order on the same line in $X$ space, where $k_1, k_2 > |L|$ for the new points $X^{k_1},\ X^{k_2}$;
    b. calculate the approximate objective values $\widetilde{f}^{k_1},\ \widetilde{f}^{k_2}$ for the new points $X^{k_1},\ X^{k_2}$, respectively, using a linear interpolation between the known points $\{(X^{i_1}, f^{i_1}), (X^{i_2}, f^{i_2})\}$;
    c. update the training sample $L_{ext} = L_{ext} \cup \left\{\left(X^{k_1}, \widetilde{f}^{k_1}\right), \left(X^{k_2}, \widetilde{f}^{k_2}\right)\right\}$;
    d. update the extended set of triples $T_{ext} = T_{ext} \cup \{(i_1, k_1, k_2), (k_1, k_2, i_2)\}$;
    e. repeat steps a-d for the vector $X^{i_2}X^{i_3}$.

The set of triples $T_{ext} = \{t_j, j \in (1 : |T_{ext}|)\}$ and the corresponding increment values $\left\{\left(\delta_1^j, \delta_2^j\right), j \in (1 : |T_{ext}|)\right\}$ form the extended variability map (EVM). The EVM is based on an extended training sample $L_{ext}$, which includes the new points linearly interpolated between the points of triples $T$. The examples of the original training sample $L$ and the extended sample $L_{ext}$, built with the proposed algorithm, are shown in Figure 6. It is clear from Figure 6b that the new points are placed along the triples of the original sample presented in Figure 6a.



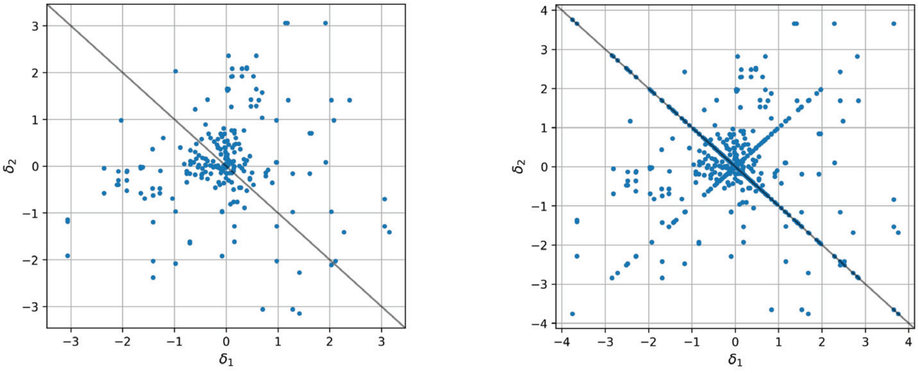
(**a**) (**b**)

**Figure 6.** The points and triples of the (**a**) original and (**b**) extended samples. Black dots represent points of the original and the extended training samples in $X$ space ($|X| = 2$), and multicolor lines connect the points of the collected triples.

Note that the set of triples $T_{ext}$ does not include the original set $T$, while the sample $L_{ext}$ also includes the points from $L$. It is recommended to locate the new points $X^{k_1}, X^{k_2}$ closer to the points $X^{i_1}, X^{i_2}$, e.g., by making logarithmic steps when splitting the vector

$X^{i_1}X^{i_2}$, since it may positively affect the accuracy of landscape validation. The closer the new points are to the original ones, the stronger the requirements for the surrogate model to preserve comparative relations between pairs of those points.

In Figure 7, the VM of the original sample and the EMV of the extended sample can be seen. Since the new points are linearly interpolated, the additional points of the EVM (Figure 7b) are located on the diagonals $\delta_1 = \delta_2$ and $\delta_1 = -\delta_2$.



(**a**) (**b**)

**Figure 7.** VM plots for the (**a**) original and (**b**) extended samples. VM's increment values are represented by blue dots.

## *3.3. Landscape Validation Metrics*

To measure the landscape consistency between a surrogate model and the original sample $L^r$ using the extended training sample $L^r_{ext}$, the value of $\phi_{RP}(L^r_{ext}, P)$ metric is calculated the following way:

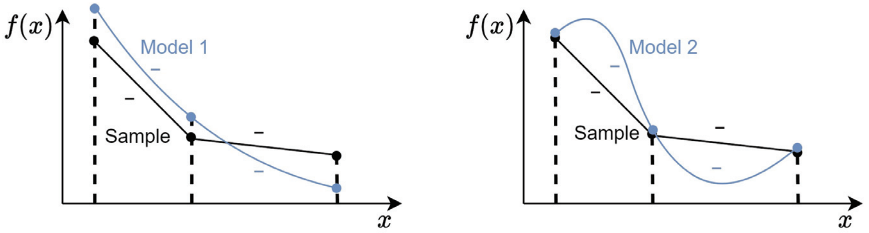
1. Build a surrogate model $\hat{f}(X)$ using the training sample $L^r$ and the vector $P$ of hyperparameter values;
2. Using the model $\hat{f}(X)$, calculate $\hat{f}^i$ values for all the points $X^i$ of the extended sample $L^r_{ext}$, where $i \in (1 : |L_{ext}|)$. The corresponding values form the sample $\hat{L}^r_{ext}$;
3. Given the samples $L^r_{ext}$ and $\hat{L}^r_{ext}$, that have the common $X^i$ values, calculate the ranking preservation metric:

$$\phi_{RP}(L^r_{ext}, P) = \frac{1}{\binom{|L^r_{ext}|}{2}} \sum_{i=1}^{|L^r_{ext}|} \sum_{j=i+1}^{|L^r_{ext}|} \begin{cases} 1, \; if \; comp(f^i, f^j) = comp\left(\hat{f}^i, \hat{f}^j\right) \\ 0, \; otherwise \end{cases}, \qquad (8)$$

where $comp(f^i, f^j)$ is the result of the comparison of $f^i$ and $f^j$ values with the possible outcomes: less, equal, more.

Note that $L^r_{ext}$ includes both the original values $f^i$ from $L$ and the linearly interpolated values $\widetilde{f^i}$. The value of the metric $\phi_{RP}(L^r_{ext}, P)$ in the range $(0, 1)$ measures the ratio of pairwise comparisons of objective values that are preserved by the model $\hat{f}(X)$.

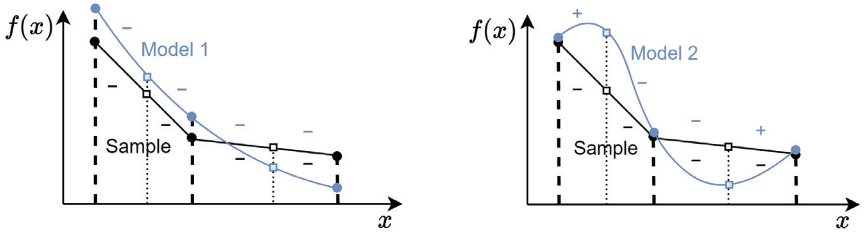
Figure 8 shows an example of models with different levels of ranking preservation. The training sample points are shown in black, while the predicted model values are shown in blue. The highest level of ranking preservation is achieved for Model 1, shown in Figure 8a, as for any pair of points $x^1$, $x^2$ in the given range the ranking preservation condition $comp(f^1, f^2) = comp\left(\hat{f}^1, \hat{f}^2\right)$ is met. Model 2 in Figure 8b is violating the rankings near the edge points of the training sample. Although Model 1 preserves the ranking better, both Model 1 and Model 2 will have the highest value of the metric $\phi_{RP}(L, P)$ when estimated based on the training sample $L$ only.



(**a**) (**b**)

**Figure 8.** Example of models with (**a**) high and (**b**) low ranking preservation levels, $|X| = 1$.

Figure 9 illustrates the idea of using the extended training sample to identify the level of ranking preservation. Although Model 2, shown in Figure 9b, has a better accuracy on the training sample, the violation of extra point ranking indicates that the model may not be suitable for optimization purposes. Although the metrics $\phi_{RP}(L_{ext}, P)$ and $\phi_{CV}(L, P)$ are correlated, they are not identical, meaning that models with similar accuracy may preserve the landscape of the training sample differently.



(**a**) (**b**)

**Figure 9.** Example of models with (**a**) high and (**b**) low ranking preservation levels for the extended training sample, $|X| = 1$

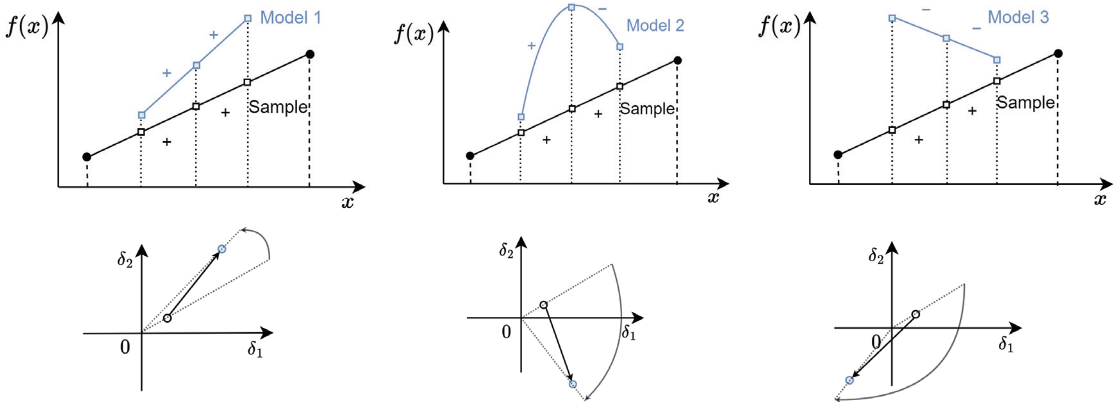
We propose another landscape validation metric $\phi_{AD}(L_{ext}, P)$, called the angular divergence (AD) of the EVM. The metric is based on the extended sample $L_{ext}$, the extended set of triples $T_{ext}$ and the corresponding increment values $\left\{\left(\delta_1^j, \delta_2^j\right), j \in (1 : |T_{ext}|)\right\}$. Instead of directly measuring the consistency of the $\hat{f}(X)$ model's landscape using the original sample $L$, it measures the consistency of the corresponding EVMs in the following way:

1. Build a surrogate model $\hat{f}(X)$ using the training sample $L^r$ and the vector $P$ of hyperparameter values;
2. Using the model $\hat{f}(X)$, calculate $\hat{f}^i$ values for all the points $X^i$ of the extended sample $L^r_{ext}$, where $i \in (1 : |L^r_{ext}|)$. The calculated values form the sample $\hat{L}^r_{ext}$;
3. For all the triples from $T^r_{ext}$, calculate the increment values $\left\{\left(\hat{\delta}_1^j, \hat{\delta}_2^j\right), j \in (1 : |T^r_{ext}|)\right\}$ using the sample $\hat{L}^r_{ext}$;
4. Assuming each pair of the increment values $\left(\delta_1^j, \delta_2^j\right)$ correspond to the vector $\Delta^j$ in the $\delta_1\delta_2$ space, and each pair of the values $\left(\hat{\delta}_1^j, \hat{\delta}_2^j\right)$—to the vector $\hat{\Delta}^j$, calculate the angular divergence metric based on the cosine similarity of the vectors $\Delta^j$ and $\hat{\Delta}^j$:

$$\phi_{AD}(L^r_{ext}, P) = \frac{1}{|T^r_{ext}|} \sum_{j=1}^{T^r_{ext}} \frac{\Delta^j \cdot \hat{\Delta}^j}{||\Delta^j|| \, ||\hat{\Delta}^j||}, \tag{9}$$

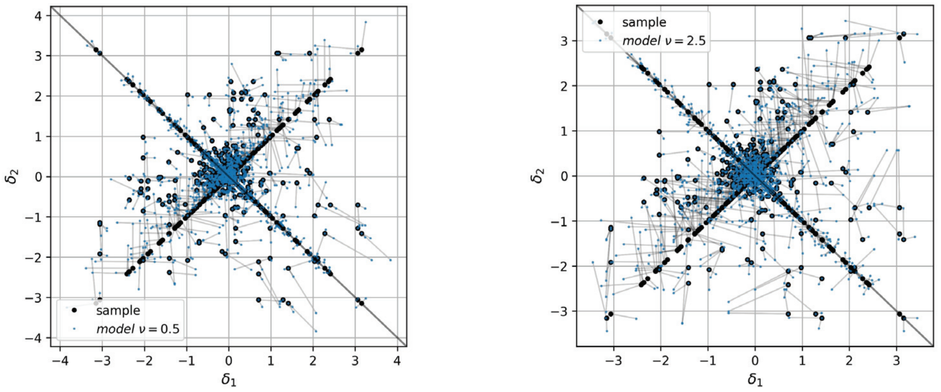
where $\Delta^j \cdot \hat{\Delta}^j$ is a dot product of the corresponding vectors.

The metric $\phi_{AD}(L_{ext}, P)$ is calculated as an average angle of rotation of EVM points around the central point of the $0\delta_1\delta_2$ plane. The rotation angle is determined by the model's ability to preserve the ratio between the $\delta_1^j$ and $\delta_2^j$ values, and not the absolute magnitude of those values. In Figure 10, the example of the AD calculation of a single triple for models with different levels of ranking preservation is shown. Model 1, presented in Figure 10a, preserves the signs of the increment values, so the angle between the EVM points that correspond to the sample and the model is relatively small. Model 2, shown in Figure 10b, alters the sign of the second increment value, hence the angular divergence for the corresponding EVM point is around 90 degrees. The signs of both increments are altered by Model 3 shown in Figure 10c, which results in an even larger angular divergence. Figure 11 illustrates the angular divergence for the whole EVMs built for GP models with different values of the Matern kernel parameter. The extended sample increments are shown in black points, while the increments calculated for the models are shown in blue. The value of the metric $\phi_{AD}(L_{ext}, P)$ of the first model shown in Figure 11a is lower than for the second model shown in Figure 11b (13 and 19 degrees on average, correspondingly).



(**a**) (**b**) (**c**)

**Figure 10.** Example of angular divergence of a single triple and the corresponding point on a VM for models with (**a**) high, (**b**) medium, and (**c**) low ranking preservation levels, $|X| = 1$.



(**a**) (**b**)

**Figure 11.** EVMs of GP models with Matern kernel parameter (**a**) $\nu = 0.5$ and (**b**) $\nu = 2.5$.

## 4. Computational Experiment

In this section, the results of computational experiments performed with the proposed landscape validation metrics are presented. The baseline is set by estimating the Bayesian optimization efficiency with fixed hyperparameter values. In the first experiment, the efficiency of Bayesian optimization with hyperparameter tuning based on different landscape validation metrics is estimated. The second experiment evaluates the efficiency of the hyperparameter prediction approach based on ELA-ML approach with different landscape validation metrics.

### *4.1. General Setup*

The Bayesian optimization efficiency is estimated on the set $Q = \{q_i, i \in (1 : M_Q)\}$ of optimization problems that are based on the BBOB set of 24 test functions, that are widely used for optimization efficiency studies [20]. The BBOB set is accessed by using Python interface of the IOHexperimenter package [21]. Each function in the BBOB set is available for arbitrary dimensions $|X|$ and the numbers of so-called instances obtained by a random shift in $X$ space. The set $Q$ of test optimization problems is composed of BBOB test functions that have dimensions $|X| = 2, 4, 8$ and the fixed instance number. Hence the total number of test problems is $M_Q = 72$.

To solve test problems $Q$, an open Python implementation of the Bayesian optimization algorithm in the ByesOpt package is used [22]. The surrogate models are built using the Matern kernel, which has a tunable parameter $\nu$ with a recommended value of $\nu^{rec} = 2.5$ specified in the package. The set of allowed $\nu$ values is fixed to $D_\nu = \{0.5,\ 1.5,\ 2.0,\ 2.5,\ 3.0, inf\}$ for hyperparameter tuning and prediction. By default, the package uses the LCB acquisition function with $\kappa = 2.576$ to select the next point for objective evaluation [22].

The set of ELA features $C = (c_1, \ldots c_{84})$ is used to categorize the test problems for hyperparameter prediction. The vector $C$ includes 41 features evaluated by the pFlacco package [23] and 43 features based on VM [12]. Only ELA features that are based on a fixed sample of points are used since additional objective evaluations are not permissible with a fixed computational budget. The ELA features that require cell mapping of the search space are also excluded. Due to the exponential growth of the total number of cells with the dimension $|X|$, the sample sizes will not be sufficient for cell mapping feature calculation.

In the following sections, the method of solving the tuning problem (3) is referred to as a strategy of the Bayesian optimization algorithm. The efficiency of the following strategies is analyzed:

- Fixed vector $P$—no hyperparameter tuning;
- Hyperparameter tuning with the considered metrics:
  - metric $\phi_{CV}(L, P)$ with 5 folds (see Equation (4) in Section 2.2);
  - metric $\phi_{RP}(L_{ext}, P)$ (see Equation (8) in Section 3.3);
  - metric $\phi_{AD}(L_{ext}, P)$ (see Equation (9) in Section 3.3).
- Predicted vector $\hat{P}$ (see Section 2.3);
- The metrics $\phi_{RP}(L_{ext}, P)$ and $\phi_{AD}(L_{ext}, P)$ will be referred to as $\phi_{RP}(L, P)$ and $\phi_{AD}(L, P)$, respectively, to simplify the experiment description.

In the experiments, the efficiency of the Bayesian optimization algorithm with a given strategy is measured by the solution's quality and by the computational cost of solving a test problem. The solution's quality is determined by the best objective value found during the run. The computational cost is assessed by estimating the average time required to solve a test problem, which includes the time spent on building surrogate models and calculating values of the selected metric. Since test problems are used, the time spent on objective evaluation is negligible. The time required to build a tuning model is not taken into account when using the hyperparameter prediction strategy. The experiments were performed on a computer with an Intel E5-2643 processor.

### *4.2. Bayesian Optimization with Hyperparameter Tuning*

To compare the efficiency of Bayesian optimization with fixed hyperparameter values and hyperparameter tuning, the experiment with the following steps is performed.

1. For each problem $q_i \in Q$ with the objective function $f_i(X)$, generate the number of random initial samples $\left\{L_{i,j}^0,\ j \in (1:10)\right\}$, each sample of size $\left|L_{i,j}^0\right| = 5|X|$;
2. Using each initial sample $L_{i,j}^0$, perform iterations $r \in (1:10|X|)$ of the Bayesian optimization algorithm, as described in Section 2.1;
3. Using each initial sample $L_{i,j}^0$, perform iterations $r \in (1:10|X|)$ of the Bayesian optimization algorithm with hyperparameter tuning, as described in Section 2.2, with each of the following metrics: $\phi_{CV}\left(L_{i,j}^r, P\right)$, $\phi_{RP}\left(L_{i,j}^r, P\right)$ and $\phi_{AD}\left(L_{i,j}^r, P\right)$. The best hyperparameter values are selected from the set $D_v$;
4. Estimate the average of the best objective values $\overline{f}_i^*$ found in 10 random runs with fixed hyperparameter values in step 2 and with hyperparameter tuning based on the metrics $\phi_{CV}\left(L_{i,j}^r, P\right)$, $\phi_{RP}\left(L_{i,j}^r, P\right)$ and $\phi_{AD}\left(L_{i,j}^r, P\right)$ in step 3:

$$\overline{f}_i^* = \frac{1}{10} \sum_j f_{i,j}^*, \tag{10}$$

   where $f_{i,j}^*$ is the best objective value found for the problem $q_i$ in the $j$-th run;
5. For each problem $q_i$ select the metric $\phi_i(L, P)$ with which the best objective value was found on average.

In the described experiment, each of the 72 test optimization problems was solved using 10 random initial samples and four different strategies with the computational budget $15|X|$−2880 runs of the Bayesian optimization algorithm in total.

### *4.3. Bayesian Optimization with Hyperparameter Prediction*

The experiment aims to measure the efficiency of using the hyperparameter prediction approach based on the ELA-ML framework by performing the following steps for each optimization problem $q_i$, $i \in (1:72)$. The experiment is split into exploration and exploitation phases as it is described in Section 2.3.

Exploration phase—evaluate ELA features, find the best hyperparameter values:

1. Remove problems from the set $Q$ that are based on the same BBOB function as the current problem $q_i$, including those with different dimensions $|X|$, so that the remaining problems compose the set $Q_i = \{q_k, k \in (1:69)\}$;
2. For each problem $q_k$ generate the random samples $\{L_{k,s}, s \in (1:300)\}$ with different sizes $|L_{k,s}| \in (5|X|,\ 15|X|)$. Within the given range, 20 sample sizes are chosen and 15 random samples of each size are generated in $D_X$;
3. For each sample $L_{k,s}$ calculate the vector of ELA features $C_{k,s}$, where $|C_{k,s}| = 84$ and find the best hyperparameter values $P_{k,s}^*$ by solving the problem (3) with the set of allowed values $D_v$. As the hyperparameter efficiency metric use the metric $\phi_k(L, P)$, which showed the best performance for the problem $q_k$ in Section 4.2;
4. Use the set of pairs $\left\{C_{k,s}, P_{k,s}^*\right\}$ as a training sample to build a tuning model $\hat{P}_i(C)$ by using the random forest classifier implemented in the scikit-learn package [24].

Exploitation phase—use the tuning model $\hat{P}_i$ for hyperparameter prediction:

1. For the current problem $q_i$, generate the number of random initial samples $\left\{L_{i,j}^0,\ j \in (1:10)\right\}$, each sample of size $\left|L_{i,j}^0\right| = 5|X|$;
2. Using each initial sample $L_{i,j}^0$ perform iterations $r \in (1:10|X|)$ of the Bayesian optimization algorithm with hyperparameter prediction by using the tuning model $\hat{P}_i$, as described for the exploitation phase in Section 2.3;

3. Estimate the average of the best objective values $\overline{f}_i^*$ found in 10 random runs with hyperparameter prediction.

Each tuning model $\hat{P}_i$ is built using 20,700 pairs of observations, all of which are collected for problems based on different BBOB functions. The experiment is structured in such a way that it is comparable to the cross-validation procedure. Each problem $q_i$ is excluded from the exploration phase to build a tuning model and is solved during the exploitation phase with the hyperparameter values predicted by that model. The hyperparameter values are predicted based on the ELA features similarity between the problem $q_i$ and the set of problems $\{q_k\}$ analyzed during the exploration phase. In our case, the accuracy of hyperparameter prediction for a particular problem $q_i$ depends, among other factors, on the presence of BBOB functions with a similar landscape in the set $\{q_k\}$.

### 4.4. Experimental Results

The results of the first experiment are summarized in Table 1. For each problem $q_i$, the best strategy is selected that provides the best value $\overline{f}_i^*$. For each considered strategy, the table shows the number of problems $q_i$ where the best result was found while using that strategy. It should be noted that for certain test problems, such as with the linear slope objective function, multiple strategies were able to find the optimal solution.

**Table 1.** The number of best-solved problems with the fixed hyperparameter value and the hyperparameter tuning approach using different metrics.

| Bayesian Optimization Strategy | Number of Problems with the Best $\overline{f}_i^*$ |
|---|---|
| Fixed vector $P$ | 12 |
| Hyperparameter tuning with: | |
| metric $\phi_{CV}$ | 24 |
| metric $\phi_{RP}$ | 24 |
| metric $\phi_{AD}$ | 21 |

With the fixed hyperparameter values, the number of best-solved problems is the lowest, as expected. Using the proposed metrics for hyperparameter tuning leads to the best results on a wider range of problems. Based on the results, it can be assumed that different problems require different approaches to hyperparameter tuning, i.e., different metrics.

Table 2 summarizes the results of the second experiment in the same manner. On top of that, the average of the best-found values $\overline{f}_i^*$ and the average time required to solve a test problem are provided. Since the scale of objective values of the test problems vary a great deal, the best-found values $\overline{f}_i^*$ were normed to the range $(0;1)$, so that 0 and 1 correspond to the best and worst values $f_{i,j}^*$ found for the problem $q_i$ in all the runs with different strategies.

**Table 2.** The experimental results with the fixed hyperparameter value, the hyperparameter tuning approach using different metrics and the hyperparameter prediction approach.

| Bayesian Optimization Strategy | Number of Problems with the Best Value $\overline{f}_i^*$ | Average of Normed Best Values $\overline{f}_i^*$ | Average Time for Solving a Problem, s |
|---|---|---|---|
| Fixed vector $P$ | 9 | 0.379 | 24 |
| Hyperparameter tuning with: | | | |
| metric $\phi_{CV}$ | 19 | 0.303 | 457 |
| metric $\phi_{RP}$ | 13 | 0.305 | 211 |
| metric $\phi_{AD}$ | 17 | 0.305 | 173 |
| Predicted vector $\hat{P}$ | 26 | 0.290 | 144 |

It is evident that the proposed hyperparameter prediction approach provides the best results for a larger number of problems. The metric $\phi_{CV}$ that requires cross-validation of the surrogate models is the most time-consuming but also the most accurate metric. However, with the proposed metrics $\phi_{RP}$ and $\phi_{AD}$ the results of comparable quality can be found with over 50% less effort. By using the hyperparameter prediction strategy, the problems can be solved with even less time (up to 70%), while the quality of the results is about 5% better than for the most accurate metric $\phi_{CV}$.

Fixedmetric $\phi_{CV}$metric $\phi_{RP}$metric $\phi_{AD}$PredictedWhen benchmarking optimization algorithms, average efficiency estimates are often not sufficient for making informed conclusions. Figure 12 presents the experimental results in the form of so-called performance profiles [25]. The performance profile of each strategy is constructed by calculating the number of test problems with a better value $\overline{f}_i^* \leq \overline{f}'$ for all possible values $\overline{f}' \in (0,1)$. As a result, the performance profile plot shows the number of test problems as a function of the quality of the solution. For example, the two most effective strategies for finding near-best values $\overline{f}_i^* \leq 0.1$ use the fixed vector $P$ and hyperparameter tuning with the metric $\phi_{CV}$. Figure 13 presents the same performance profiles for all the algorithm runs with random initial samples (10 random runs for each of the 72 problems).

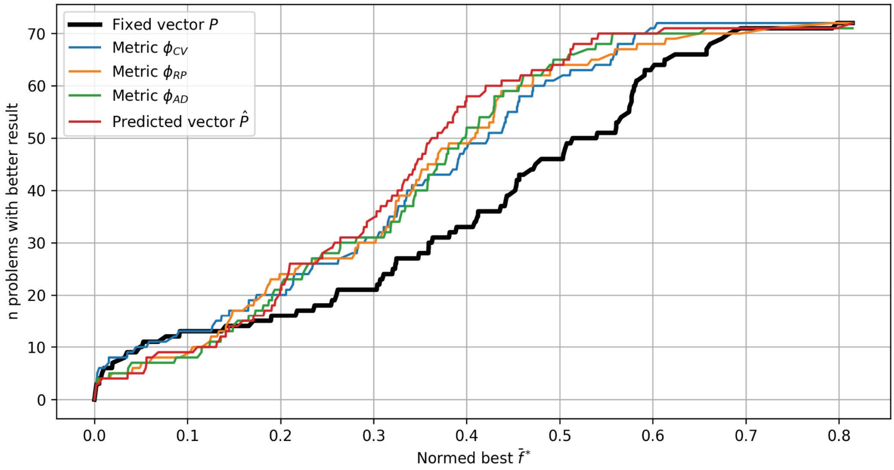


**Figure 12.** Performance profile plots for different strategies of Bayesian optimization: the number of test problems solved with better values $\overline{f}^*$.

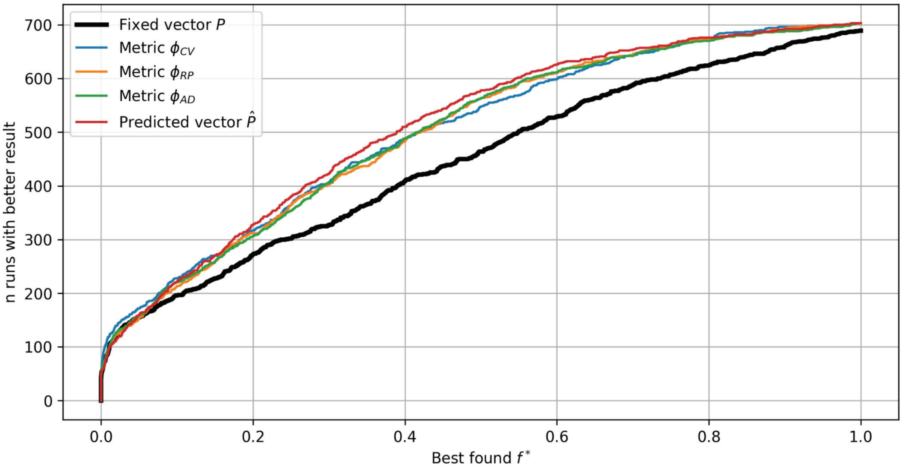


**Figure 13.** Performance profile plots for different strategies of Bayesian optimization: the number of runs with better values $f^*$.

In Figure 14 the performance profile plots are shown for the number of test problems as a function of the time required to solve a problem. It can be clearly seen that the plot has «jumping» segments due to the time difference between solving the problems of different dimensions $|X| = 2, 4, 8$. The computational complexity grows with the problem dimension slower for the metric $\phi_{AD}$ than for the metric $\phi_{RP}$. The hyperparameter prediction strategy is relatively expensive for smaller dimensions due to the computational costs involved in estimating the features vector $C$ and evaluating the tuning model $\hat{P}(C)$. Using the fixed vector $P$ and selecting hyperparameter values with the metric $\phi_{CV}$ are obviously the strategies with the best and the worst time efficiency, respectively. Figure 15 presents the same performance profiles for all the algorithm runs with random initial samples (10 random runs for each of the 72 problems).

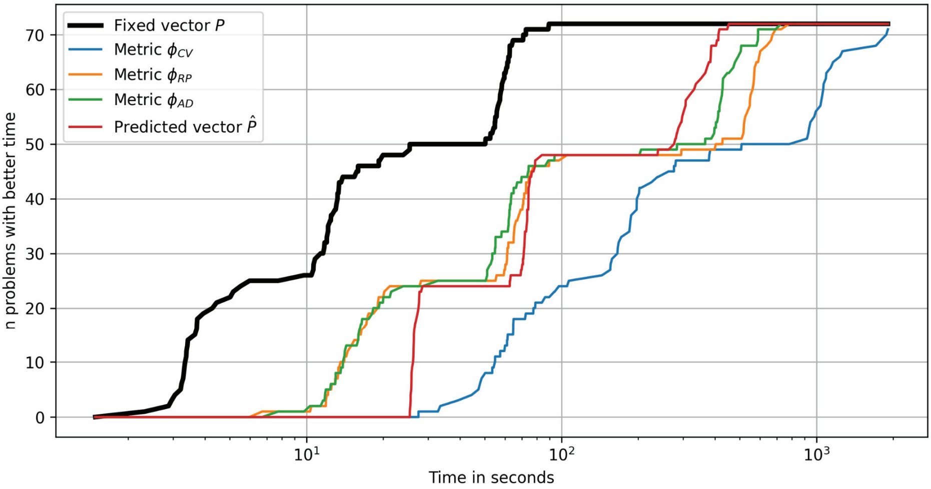


**Figure 14.** Performance profile plots for different Bayesian optimization strategies: the number of test problems solved in less time.

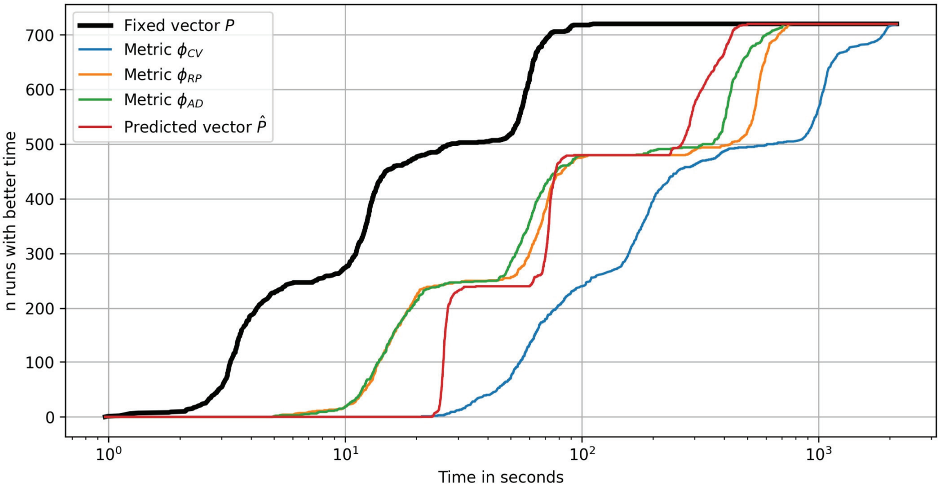


**Figure 15.** Performance profile plots for different Bayesian optimization strategies: the number of runs completed in less time.

## 5. Discussion

The presented exploratory landscape validation approach enables the finding pf effective hyperparameter values without relying on approximation accuracy estimations of surrogate models. In cases of limited computational budget and hence relatively small training samples, landscape validation metrics provide a faster way to estimate the efficiency of hyperparameter values. The presented metrics summarize the essential differences between the landscapes of surrogate models and training samples, that are generally characterized by ELA features. The study, however, has the following potential limitations. The experiments were performed with test optimization problems, but the applicability of the proposed metrics is mainly determined by the computational complexity of objective evaluation and hence the computational budget. In the event of a higher complexity of objective evaluation and relatively small budget, the computational impact even of the cross-validation metric may become negligible. On the other hand, if the budget is large enough, the increase in the optimization algorithm's efficiency may not be sufficient to justify the computational cost of hyperparameter tuning with any of the considered metrics. Analyzing applicability conditions for different tuning strategies when solving practical optimization problems could be the focus of future research. It is also promising to explore other ways of estimating the level of landscape feature preservation by surrogate models based on the known ELA methods, for example, by direct comparison of ELA feature vectors. Landscape validation can be based on various ways of measuring the differences between the variability maps of samples and surrogate models.

It was shown that landscape validation metrics can be used to both find and predict the best hyperparameter values during Bayesian optimization. Another potential extension of this work is to analyze the efficiency of hyperparameter prediction with different ELA algorithms and approximation algorithms used for building a tuning model. The efficiency of hyperparameter prediction is also affected by the level of similarity between the problems being solved at exploration and exploitation phases, which is difficult to formalize. One possible approach would be to use the ELA features of the test problem set to define the region of allowed feature values for the exploitation-phase problems. If the new problem has unrelated feature values, then the efficiency of the predicted hyperparameter values cannot be guaranteed; therefore, the default hyperparameter values should be used for that problem. In such cases, the test problem set should be refined to keep it representative of the problems solved during the exploitation phase.

## 6. Conclusions

The article considers different approaches to improving the efficiency of Bayesian optimization algorithms by selecting the best hyperparameter values of the surrogate modeling algorithm. The optimal vector of hyperparameter values is found based on a hyperparameter efficiency metric, which defines the way of measuring the quality of a surrogate model built with different vectors. The hyperparameter tuning problem is being solved at each iteration of Bayesian optimization, so using computationally demanding metrics may lead to a significant increase in the time spent solving the problem.

When solving computationally expensive optimization problems, the number of objective evaluations allowed is relatively small, as well as the size of the training sample for building a surrogate model. The commonly used efficiency metric for such cases is the cross-validation score, which requires building multiple surrogate models on different subsets of the training sample. In this article, a new approach is introduced called exploratory landscape validation (ELV), which includes the proposed hyperparameter efficiency metrics to assess the quality of surrogate models without considering the approximation error estimates. The experiments showed that hyperparameter tuning with the new metrics can provide solutions of comparable quality with less than half the time required when using the cross-validation metric. The experimental results also indicate that different metrics provide the best solutions for different optimization problems.

Another way of reducing the costs of hyperparameter tuning is to build a model that predicts the best hyperparameter values during Bayesian optimization based on ELA features estimated from the training sample. The hyperparameter prediction approach is based on collecting a test set of problems, estimating the ELA features of those problems, finding the best hyperparameter values according to an efficiency metric, and building a tuning model. In general, each optimization problem has its own best-suited efficiency metric for hyperparameter tuning. In the computational experiment, the tuning models are built to predict the hyperparameter values that are most effective according to the metrics chosen individually for each test optimization problem. With the suggested hyperparameter prediction approach and individual efficiency metrics, better-quality solutions were found in less than 70% of the time needed by a hyperparameter tuning approach based on cross-validation score. Even though additional computational expenses are required to create a tuning model, they are insignificant when compared to potential permanent improvement in the optimization algorithm's efficiency.

**Author Contributions:** Conceptualization, A.K. and T.A.; methodology, A.K.; software, T.A.; validation, T.A.; formal analysis, T.A.; investigation, T.A.; writing—original draft preparation, T.A.; writing—review and editing, T.A.; visualization, T.A.; supervision, A.K. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** The source code and the results of the computational experiments are openly available at https://github.com/agataleh/boela_bench (accessed on 6 January 2024).

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Alizadeh, R.; Allen, J.K.; Mistree, F. Managing computational complexity using surrogate models: A critical review. *Res. Eng. Des.* **2020**, *31*, 275–298. [CrossRef]
2. Palar, P.S.; Liem, R.P.; Zuhal, L.R.; Shimoyama, K. On the use of surrogate models in engineering design optimization and exploration: The key issues. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, Prague, Czech Republic, 13–17 July 2019; Association for Computing Machinery: New York, NY, USA; pp. 1592–1602.
3. Jariego Perez, L.C.; Garrido Merchan, E.C. Towards Automatic Bayesian Optimization: A first step involving acquisition functions. In Proceedings of the 19th Conference of the Spanish Association for Artificial Intelligence, Advances in Artificial Intelligence, Malaga, Spain, 22–24 September 2021; Springer: Cham, Switzerland; pp. 160–169.
4. Gan, W.; Ji, Z.; Liang, Y. Acquisition functions in Bayesian optimization. In Proceedings of the 2nd International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE), Zhuhai, China, 24–26 September 2021; IEEE: Piscataway Township, NJ, USA; pp. 129–135.
5. Palar, P.S.; Parussini, L.; Bregant, L.; Shimoyama, K.; Zuhal, L.R. On kernel functions for bi-fidelity Gaussian process regressions. *Struct. Struct. Multidiscip. Multidiscip. Optim. Optim.* **2023**, *66*, 37. [CrossRef]
6. Yu, H.; Tan, Y.; Sun, C.; Zeng, J. A comparison of quality measures for model selection in surrogate-assisted evolutionary algorithm. *Soft Comput. Comput.* **2019**, *23*, 12417–12436. [CrossRef]
7. Bischl, B.; Binder, M.; Lang, M.; Pielok, T.; Richter, J.; Coors, S.; Thomas, J.; Ullmann, T.; Becker, M.; Boulesteix, A.L.; et al. Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2023**, *13*, e1484. [CrossRef]
8. Williams, C.K.I.; Rasmussen, C.E. *Gaussian Processes for Machine Learning*; MIT Press: Cambridge, MA, USA, 2006; Volume 4, pp. 83–89.
9. Bhosekar, A.; Ierapetritou, M. Advances in surrogate based modeling, feasibility analysis, and optimization: A review. *Comput. Comput. Chem. Chem. Eng. Eng.* **2018**, *108*, 250–267. [CrossRef]
10. Garbo, A.; German, B.J. Performance assessment of a cross-validation sampling strategy with active surrogate model selection. *Struct. Multidiscip. Optim.* **2019**, *59*, 2257–2272. [CrossRef]
11. Diaz-Manriquez, A.; Toscano, G.; Coello Coello, C.A. Comparison of metamodeling techniques in evolutionary algorithms. *Soft Comput.* **2017**, *21*, 5647–5663. [CrossRef]
12. Agasiev, T.A. Characteristic feature analysis of continuous optimization problems based on Variability Map of objective function for optimization algorithm configuration. *Open Comput. Sci.* **2020**, *10*, 97–111. [CrossRef]
13. Škvorc, U.; Eftimov, T.; Korošec, P. Understanding the problem space in single-objective numerical optimization using exploratory landscape analysis. *Appl. Soft Comput.* **2020**, *90*, 106138. [CrossRef]

14. Renau, Q.; Doerr, C.; Dreo, J.; Doerr, B. Exploratory landscape analysis is strongly sensitive to the sampling strategy. In Proceedings of the 16th International Conference on Parallel Problem Solving from Nature, Leiden, The Netherlands, 5–9 September 2020; Springer: Cham, Switzerland; pp. 139–153.
15. Kerschke, P.; Preuss, M. Exploratory landscape analysis. In Proceedings of the Companion Conference on Genetic and Evolutionary Computation, Lisbon, Portugal, 15–19 July 2023; Association for Computing Machinery: New York, NY, USA; pp. 990–1007.
16. Saini, B.S.; López-Ibáñez, M.; Miettinen, K. Automatic surrogate modelling technique selection based on features of optimization problems. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, Prague, Czech Republic, 13–17 July 2019; Association for Computing Machinery: New York, NY, USA; pp. 1765–1772.
17. Kerschke, P.; Trautmann, H. Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. *Evol. Comput.* **2019**, *27*, 99–127. [CrossRef] [PubMed]
18. Viana, F.A.; Venter, G.; Balabanov, V. An algorithm for fast optimal Latin hypercube design of experiments. *Int. J. Numer. Methods Eng.* **2010**, *82*, 135–156. [CrossRef]
19. Wang, X.; Jin, Y.; Schmitt, S.; Olhofer, M. Recent advances in Bayesian optimization. *ACM Comput. Surv.* **2023**, *55*, 1–36. [CrossRef]
20. Varelas, K.; El Hara, O.A.; Brockhoff, D.; Hansen, N.; Nguyen, D.M.; Tušar, T.; Auger, A. Benchmarking large-scale continuous optimizers: The bbob-largescale testbed, a COCO software guide and beyond. *Appl. Soft Comput.* **2020**, *97*, 106737. [CrossRef]
21. de Nobel, J.; Ye, F.; Vermetten, D.; Wang, H.; Doerr, C.; Bäck, T. Iohexperimenter: Benchmarking platform for iterative optimization heuristics. *Evol. Comput.* **2023**, 1–6. [CrossRef] [PubMed]
22. Nogueira, F. Bayesian Optimization: Open Source Constrained Global Optimization Tool for Python. 2014. Available online: https://github.com/bayesian-optimization/BayesianOptimization (accessed on 6 December 2023).
23. Prager, R.P.; Trautmann, H. Pflacco: Feature-Based Landscape Analysis of Continuous and Constrained Optimization Problems in Python. *Evol. Comput.* **2023**, 1–25. [CrossRef] [PubMed]
24. Hao, J.; Ho, T.K. Machine learning made easy: A review of scikit-learn package in python programming language. *J. Educ. Behav. Stat.* **2019**, *44*, 348–361. [CrossRef]
25. Moré, J.J.; Wild, S.M. Benchmarking derivative-free optimization algorithms. *SIAM J. Optim.* **2009**, *20*, 172–191. [CrossRef]

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

mathematics

MDPI

*Article*

# Fractional-Differential Models of the Time Series Evolution of Socio-Dynamic Processes with Possible Self-Organization and Memory

**Dmitry Zhukov [1,*], Konstantin Otradnov [1] and Vladimir Kalinin [2,*]**

[1] Institute of Radio Electronics and Informatics, MIREA—Russian Technological University, 78 Vernadsky Avenue, 119454 Moscow, Russia; otradnov@mirea.ru

[2] Department of Applied Informatics and Intelligent Systems in the Humanitarian Sphere, Patrice Lumumba Peoples' Friendship University of Russia, 6 Miklukho-Maklaya Str., 117198 Moscow, Russia

[*] Correspondence: zhukov_do@mirea.ru (D.Z.); vkalininz@mail.ru (V.K.)

**Abstract:** This article describes the solution of two problems. First, based on the fractional diffusion equation, a boundary problem with arbitrary values of derivative indicators was formulated and solved, describing more general cases than existing solutions. Secondly, from the consideration of the probability schemes of transitions between states of the process, which can be observed in complex systems, a fractional-differential equation of the telegraph type with multiples is obtained (in time: $\beta, 2\beta, 3\beta, \ldots$ and state: $\alpha, 2\alpha, 3\alpha, \ldots$) using orders of fractional derivatives and its analytical solution for one particular boundary problem is considered. In solving edge problems, the Fourier method was used. This makes it possible to represent the solution in the form of a nested time series (one in time t, the second in state x), each of which is a function of the Mittag-Leffler type. The eigenvalues of the Mittag-Leffler function for describing states can be found using boundary conditions and the Fourier coefficient based on the initial condition and orthogonality conditions of the eigenfunctions. An analysis of the characteristics of time series of changes in the emotional color of users' comments on published news in online mass media and the electoral campaigns of the US presidential elections showed that for the mathematical expectation of amplitudes of deviations of series levels from the size of the amplitude calculation interval ("sliding window"), a root dependence of fractional degree was observed; for dispersion, a power law with a fractional index greater than 1.5 was observed; and the behavior of the excess showed the presence of so-called "heavy tails". The obtained results indicate that time series have unsteady non-locality, both in time and state. This provides the rationale for using differential equations with partial fractional derivatives to describe time series dynamics.

**Keywords:** differential equations with fractional derivatives; time series; self-organization; presence of memory; non-stationarity; fractality of time series; sociodynamic processes

**MSC:** 37M10

**Citation:** Zhukov, D.; Otradnov, K.; Kalinin, V. Fractional-Differential Models of the Time Series Evolution of Socio-Dynamic Processes with Possible Self-Organization and Memory. *Mathematics* **2024**, *12*, 484. https://doi.org/10.3390/math12030484

Academic Editor: Dongfang Li

Received: 15 December 2023
Revised: 18 January 2024
Accepted: 29 January 2024
Published: 2 February 2024

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

## 1. Introduction

When modeling the dynamics of various processes, three types of fractional-differential equations can be considered: equations with a fractional derivative by coordinate (in the case of one-dimensional space), equations with a fractional derivative by time, as well as mixed-type equations, including fractional operators both by coordinate and by time.

It should be noted that in different problems, depending on the nature of the processes under investigation, the concept of "coordinate" may have different meanings. For instance, in most physical studies, the coordinate, or in the case of three dimensions, coordinates, refer to spatial dimensions. When modeling the dynamics of time series, the coordinate is defined as the variable that describes the changes in the levels of the series and is measured in units corresponding to the nature of the observed process. For instance, if fractional-differential equations are used to analyze and model the dynamics of market indicators, the levels of the series represent the prices of stocks, commodities, currency ratios, etc.

Differential equation with fractional partial derivatives in the general case has the following form (this is a fractional diffusion equation):

$$\frac{\partial^\beta \rho(x,t)}{\partial t^\beta} = D\frac{\partial^\alpha \rho(x,t)}{\partial x^\alpha} \tag{1}$$

where $\alpha$ and $\beta$ are the exponents of the fractional derivative (according to Caputo); $\rho(x,t)$ is the probability density function of observing state $x$ at time $t$, depending on both time $t$ and state $x$; and $D$ is a constant coefficient. When describing diffusion, $\rho(x,t)$ represents the concentration of the diffusing substance, and $D$ stands for the diffusion coefficient.

In general, the fractional derivative of order $\nu$ for the function $\psi$(x) is defined as follows [1]:

$$\frac{d^\nu \psi(x)}{dx^\nu} = \begin{cases} \frac{1}{\Gamma(-\nu)}\cdot\int_a^x \frac{f(\xi)d\xi}{\{x-\xi\}^{\nu+1}}, \nu < 0 \\ \frac{1}{\Gamma(1-\nu)}\cdot\frac{d}{dx}\int_a^x \frac{f(\xi)d\xi}{\{x-\xi\}^{\nu}}, 0 \le \nu < 1 \\ \frac{1}{\Gamma(2-\nu)}\cdot\frac{d^2}{dx^2}\int_a^x \frac{f(\xi)d\xi}{\{x-\xi\}^{\nu-1}}, 1 \le \nu < 2 \\ \frac{1}{\Gamma(3-\nu)}\cdot\frac{d^3}{dx^3}\int_a^x \frac{f(\xi)d\xi}{\{x-\xi\}^{\nu-2}}, 2 \le \nu < 3 \\ \cdots \end{cases}$$

or

$$\frac{d^\nu \psi(x)}{dx^\nu} = \frac{1}{\Gamma(1-[\nu])}\cdot\frac{d^{(max\{\nu\})}}{dx^{(max\{\nu\})}}\int_a^x \frac{f(\xi)d\xi}{\{x-\xi\}^{[\nu]}}$$

where $[\nu]$ denotes the fractional part of the $\nu$ value in the exponent, and max$\{\nu\}$ represents rounding the fractional value of the $\nu$ to the nearest integer.

The definition of Caputo fractional derivatives differs from that of Riemann–Liouville. In the Caputo approach, the procedure initiates by differentiating the function with an integer order n, surpassing the non-integer order after rounding to the nearest integer (max$\{\nu\}$). Subsequently, the obtained result experiences integration with an order of $1 - [\nu]$.

At present, an analytical solution to the fractional diffusion Equation (1) has only been obtained for the case where $0 < \beta \le 1$ and $1 \le \alpha \le 2$ [2–9]. Solutions beyond the boundaries of these specified values of $\alpha$ and $\beta$ are not presented in the literature.

For instance, in [4], a general solution to Equation (1) is described. Relying on the theory of Lie groups, the authors in [4] introduce a one-parameter family of scaling transformations: $\overline{x} = \lambda^a x$; $\overline{t} = \lambda^b t$; $\overline{\rho}(\overline{x},\overline{t}) = \lambda^c \rho(x,t)$, where $a$, $b$ and $c$ are constants, and $\lambda$ is a real parameter confined within an open interval $I$ containing $\lambda = 1$.

By employing scaling transformations, it is possible to derive [4]

$$\frac{\partial^\beta \overline{\rho}(\overline{x},\overline{t})}{\partial t^\beta} - D\frac{\partial^\alpha \overline{\rho}(\overline{x},\overline{t})}{\partial x^\alpha} = \lambda^{c+b\beta}\frac{\partial^\beta \rho(\overline{x},\overline{t})}{\partial \overline{t}^\beta} - D\lambda^{c+a\alpha}\frac{\partial^\alpha \rho(\overline{x},\overline{t})}{\partial \overline{x}^\alpha}$$

If $a/b = \beta/\alpha$ and denoting $\gamma = c/b$, then by making the substitution $z = xt^{-\beta/\alpha}$ in Equation (1) and utilizing the Lie group method, it is possible to obtain a general scale-invariant solution ($\gamma \ge 0$) for this equation (given $0 < \beta \le 1$ and $1 \le \alpha \le 2$) [4]:

$$\rho(x,t) = t^\gamma \sum_{j=0}^{n} C_j \cdot z^{\alpha-j} \cdot W_{(-\beta,1+\gamma-\beta+\frac{\beta}{\alpha}j),(\alpha,1+\alpha-j)}(z^\alpha/D) \tag{2}$$

where $C_j$ are arbitrary real constants, $1 \leq j \leq n$; $\gamma$ is the scaling coefficient; and $W_{(-\beta, 1+\gamma-\beta+\frac{\beta}{\alpha}j),\ (\alpha,\ 1+\alpha-j)}(z^\alpha/D)$ represents a Wright function of the form

$$W_{(-\beta,1+\gamma-\beta+\frac{\beta}{\alpha}j),(\alpha,1+\alpha-j)}(z^\alpha/D) = \sum_{k=0}^{\infty} \frac{\{z^\alpha/D\}^k}{\Gamma\left(1+\gamma-\beta+\frac{\beta}{\alpha}j-\beta k\right)\Gamma(1+\alpha-j+\alpha k)} \tag{3}$$

$$\rho(x,t) = t^\gamma \sum_{j=0}^{n} C_j \cdot z^{\alpha-j} \cdot \sum_{k=0}^{\infty} \frac{\{z^\alpha/D\}^k}{\Gamma\left(1+\gamma-\beta+\frac{\beta}{\alpha}j-\beta k\right)\Gamma(1+\alpha-j+\alpha k)} \tag{4}$$

It should be noted that a series of solutions is obtained for the case of integer $\alpha$ and $0 < \beta \leq 1$ [4]. In the case of $\alpha = 2$, scale-invariant solutions of Equation (1) for the entire real line are expressed [4] using the Wright function as follows:

$$\rho(|x|,t) = t^\gamma \sum_{k=0}^{\infty} \frac{\left\{|z|/\sqrt{D}\right\}^k}{\Gamma\left(1+\gamma-\frac{\beta}{2}k\right)k!} \tag{5}$$

$$\rho(|x|,t) = t^\gamma \sum_{k=0}^{\infty} \frac{\{|z|/D\}^k}{\Gamma(1+\gamma-\beta k)k!} \tag{6}$$

$$\rho(|x|,t) = t^\gamma \sum_{k=0}^{\infty} \frac{|z|^{\alpha-1} \cdot \{|z|^\alpha/D\}^k}{\Gamma\left(1+\gamma+\frac{\beta}{\alpha}-\beta[k+1]\right)\Gamma(\alpha[k+1])} \tag{7}$$

The obtained solutions coincide, for $\gamma = 0$, with known solutions presented in [6–9] (the condition of scale invariance transformation of the probability density function $\overline{\rho}(\overline{x},\overline{t}) = \rho(x,t)$, is related to the preservation of total probability under any transformations).

For practical application of Equation (4) in the analysis and modeling of time series behavior, it is necessary to determine the constants $C_j$. In practice, this depends on the specific nature of the problem being solved to describe the dynamics of the time series. The determination of constants $C_j$ is a separate issue for discussion, similar to the convergence of the series in Equation (4).

Due to the complexity of the problems and substantial challenges in obtaining analytical solutions, numerical methods are frequently employed in practice. For instance, in [10], numerical methods were utilized to investigate solutions over an interval for an equation of the form

$$\frac{\rho(x,t)}{\partial t} = D\frac{\partial^\alpha \rho(x,t)}{\partial x^\alpha} \text{ at } 1 < \alpha < 2$$

Meanwhile, in [11], the solution over an interval for an equation of this form was obtained:

$$\frac{\partial^\beta \rho(x,t)}{\partial t^\beta} = D\frac{\partial^2 \rho(x,t)}{\partial x^2} \text{ at } 1 < \beta \leq 1$$

The function $\rho(x,t)$ can, for instance, be interpreted as the probability density of observing a particular value level within a time series at time $t$, if these levels can randomly vary over time.

In the case of a normal distribution law, the dispersion $\sigma^2(t)$ exhibits a linear relationship with time ($\sigma^2(t) \sim t$) If there is a slower growth of $\sigma^2(t)$ concerning $t$ ($\sigma^2(t) \sim \sqrt[\beta]{t}$, where $0 < \beta < 1$), such a process is classified as subdiffusion [12,13].

If there is a faster growth observed for $\sigma^2(t)$ concerning t ($\sigma^2(t) \sim t^\beta$, where $1 < \beta < 2$), such a process is classified as superdiffusion [14]. The distribution function describing subdiffusive processes is obtained from the solution of a fractional-differential equation of the form $\frac{\partial^\beta \rho(x,t)}{\partial t^\beta} = D\frac{\partial^2 \rho(x,t)}{\partial x^2}$, and superdiffusion from the equation $\frac{\partial \rho(x,t)}{\partial t} = D\frac{\partial^\alpha \rho(x,t)}{\partial x^\alpha}$, where $\alpha$ and $\beta$ are the exponents of fractional derivatives, and $D$ is a certain coefficient (diffusion coefficient). Additionally, processes of anomalous diffusion [15,16] can be observed, which is described by a mixed-type fractional-differential equation (Equation (1)).

Currently, the most developed and studied fractional-differential models are the ones created to describe various kinds of physical problems and are mainly devoted to the study of the processes of physical kinetics, abnormal diffusion [17] and relaxation in various environments [18–24], as well as wave processes [25–28].

Since this article will further consider the conclusion of the stochastic fractional-differential equation with multiples (in time: $\beta$, $2\beta$, $3\beta$ …… and status: $\alpha$, $2\alpha$, $3\alpha$ ……), orders of fractional derivatives and the analytical solution for one particular boundary problem (with an arbitrary value of $\beta$ and $\alpha = 1$) is presented; then, as a comparison and representation of the novelty of the results obtained, we give a description of the solution of similar wave-type fractional-differential equations. Work [25] deals with the solution of the fractional Zener wave equation:

$$\nabla^2\mathcal{U}(x,y,z,t) - \frac{1}{c_0^2}\frac{\partial^2\mathcal{U}(x,y,z,t)}{\partial t^2} + \tau_\sigma^\alpha \frac{\partial^\alpha}{\partial t^\alpha}\nabla^2\mathcal{U}(x,y,z,t) - \frac{\tau_\epsilon^\beta}{c_0^2}\frac{\partial^{\beta+2}\mathcal{U}(x,y,z,t)}{\partial t^{\beta+2}} = 0$$

where $\mathcal{U}(x,y,z,t)$, in the case of elastic waves, is the amount of displacement of medium particles from the equilibrium position at a point with coordinates x, y, z at time t; $C_0$ is the wave velocity; and $\tau_\epsilon$ и $\tau_\sigma$ are positive time constants. In this case, $\alpha$ and $\beta$ are different values of fractional derivatives over time and operator $\nabla^2$ is not fractional.

This equation can be derived from the fractional ratio of stress–strain of Zener in the propagation of elastic waves and accounting and in the linearized conservation of mass and momentum. The Ziner wave equation allows us to describe three different attenuation modes with characteristics of the power law [25]. Models based on this equation have very important applications in medical elastography and acoustics.

Work [27] provides a detailed description of the solution to the Zener equation for the case of $\alpha = \beta$ ($\alpha$ and $\beta$ can be derivatives of fractional positive numbers specifying time derivatives $t$). This result is very important since solving any fractional-differential equations in analytic form has significant problems.

A further development of the wave equation with a fractional derivative in time was obtained in work [28] to describe the behavior of waves in non-Newtonian liquids.

An example of an application of fractional-differential equations of the diffusion type to describe the dynamics of social processes is presented in work [29].

Fractional derivation by the state variable $x$ and time derivative $t$ allow you to describe non-local processes in which the transition to a certain state of the system (or process) $x$ at time $t$ depends not only on the local characteristics of the process or the behavior of the system in the vicinity of the point $x$ under consideration. but also on the values of $x$ obtained throughout the entire interval under study, that is, it globally depends on the distribution over all states of $x$ and on the history of the process (memory) throughout the time $t$. Using fractional equations according to both time and coordinates allows for memory effects. The effect of non-locality in time $t$ and non-locality in state $x$ on the probability density of detecting a system or process in state $x$ is qualitatively different. Non-locality in time affects the probability density at the initial moment of time, which can lead to self-organization, and non-locality in $x$ affects the asymptotic behavior of the probability density at very late moments of time [12–16].

## 2. Data Processing and Analysis of Observed Time Series

In several studies [30–33], it was demonstrated that observed time series of social processes exhibit fractality, while the systems they describe showcase memory and self-organization. For instance, analyzing the dependence of the mean and dispersion of amplitude changes in time series on the interval of calculating these amplitudes reveals complex relationships. For example, their dispersion is dependent on the size of the "sliding" window as a fractional root, significantly differing from, for instance, a normal distribution law.

An analysis of the observed data can be used to justify the possibility of using fractional-differential equations in modeling complex processes.

We analyzed the time series of the emotional attitude of users of the "RIA Novosti" portal toward the news published during the day, for a period of 1460 days from 1 January 2019 to 31 December 2022. The following emotions were chosen as the subjects of this study: "like" and "dislike".

In addition to analyzing the activity of users commenting on news on social media, an analysis of the preferences of voters of presidential electoral campaigns in the United States in 2012 and 2016 was carried out (data taken from the resource: http://www.realclearpolitics.com/epolls/, accessed on 27 April 2018).

For a preliminary analysis of the time series dynamics and determining their characteristics (for example, the possible presence of memory), the normalized range method of Hurst [34] can be employed. This method allows for the determination of their fractal dimension and classification of behavior type.

It is possible to normalize the range of sequence levels, $R$, using the standard deviation of these values, S, and represent their ratio (normalized range) as an equation: $R/S = C\tau H \; (log[R/S] = H * log[\tau] + log[C])$, where C is a constant, $\tau$ is the number of observations (levels of the sequence) comprising the considered time series, and the H exponent is known as the Hurst coefficient or exponent.

The presence of breakpoints in the $R/S \; (\tau)$ dependency may indicate the existence of characteristic time scales and/or periodicities. The value of the Hurst coefficient $H$ allows for the classification of time series according to the nature of their behavior [35].

For the time series, the Hurst coefficient for the "like" emotion turned out to be 0.22 and for "not like", it was 0.24. For the election campaign for the US presidential election in 2012, the Hurst indicator for the temporary series of the preferences of Obama voters turned out to be 0.29 and for Romney voters, it was 0.22. In 2016, the Hurst's indicator for a temporary series of the preferences of Clinton (Hillary) voters turned out to be 0.36 and for Trump voters, it was 0.30.

In all cases, the H value is significantly less than 0.5 and, therefore, the observed time series are anti-persistent (ergodic). Since the values of the Hurst coefficient are significantly different from 0.5, it follows from this that the structure of these series has fractality, and the processes described by it can have short-term memory [35].

The possible presence of memory should be taken into account in the model describing the dynamics of processes observed in complex social systems, for example, such as user activity when commenting on the news of online mass media or a change in voter preferences during electoral campaigns.

In order to justify the possibility of using the dynamics of processes in complex systems of fractional-differential equations for modeling, it is necessary to check whether the conditions of non-locality of behavior in time (variable $t$) and variable ($x$) describing the level of the series (for example, the proportion of "likes" and "dislikes" when commenting on news resources, or the proportion of voters with a preference for one candidate over another) are met.

For this, there were studies on the dependence of the mathematical expectation, variance, asymmetry (third moment of distribution) and excess (fourth moment of distribution) of the amplitudes of the deviations of the levels of time series from the calculation time intervals (dimensions of the "sliding window") of these amplitudes.

The study of the behavior of "excesses" allows you to show the presence of so-called "heavy tails" (when the graph of the distribution function lies above the graph of the Gaussian distribution function). In practice, "light tails" can also be observed. The presence of "tails" other than the Gaussian distribution indicates the non-locality of the process over the variable $x$. The study of the behavior of expectation can show the presence of non-stationary (or vice versa, stationary), and the study of the behavior of variance can show non-locality over time (variable $t$). The identified non-localities can be described using fractional derivatives.

The observed value of mathematical expectation, dispersion, asymmetry and excess can be calculated by the following equations:

$$\mu(t) = \frac{\sum_{j=1}^{N} x_j(t)}{\sum_{l=1}^{M} n_l}$$

$$\sigma^2(t) = \frac{\sum_{j=1}^{N}\{x_j(t) - \mu(t)\}^2}{\sum_{l=1}^{M} n_l}$$

$$As(t) = \frac{\sum_{j=1}^{N}\{x_j(t) - \mu(t)\}^3}{\sigma^3 \sum_{l=1}^{M} n_l}$$

$$Ex(t) = \frac{\sum_{j=1}^{N}\{x_j(t) - \mu(t)\}^4}{\sigma^4 \sum_{l=1}^{M} n_l} - 3$$

where $\sum_{l=1}^{M} n_l$ is calculated from the number of $n_l$ amplitudes.

For example, when executing the normal distribution law for a stationary time series $(x,t) = \frac{1}{2\sqrt{\pi \mathrm{D}t}} \cdot e^{-\frac{x^2}{4 \cdot D \cdot t}}$, the amount of expectation $\mu(t)$ would have to be zero:

$$\mu(t) = \int_{-\infty}^{+\infty} x \cdot \rho(x,t)dx = \frac{1}{2\sqrt{\pi \mathrm{D}t}} \int_{-\infty}^{+\infty} x \cdot e^{-\frac{x^2}{4 \cdot D \cdot t}} dx = 0$$

where $x$ is the value of the amplitude, and t is the interval of its calculation time (the value of the "sliding window").

When fulfilling the normal law, a linear dependence on the "sliding window" value would have to be observed for amplitude dispersion:

$$\sigma^2(t) = \int_{-\infty}^{+\infty} x^2 \rho(x,t)dx = \frac{1}{2\sqrt{\pi \mathrm{D}t}} \int_{-\infty}^{+\infty} x^2 e^{-\frac{x^2}{4 \cdot D \cdot t}} dx = 2Dt$$

Processing shows that

1. Mathematical expectations of changes in the amplitudes of time series levels depend on the time interval for calculating these changes (the "sliding window"). This indicates the unsteadiness of the time series under consideration and the inability to describe their parameters using the normal distribution law.
2. The values of the dispersion of the amplitudes of the change in user activity on news commentary depends on the time interval for calculating these amplitudes (the "sliding window") in a complex way: it is proportional to the fractional degree from the time interval from which they are calculated. The fractional dependence on the time interval indicates that the studied processes have a non-locality in time t (i.e., have a consequence or memory).
3. The studies of the excess distribution of amplitudes show the presence of the so-called "heavy tails", which are significantly greater than the normal distribution (where the excess is 3). With significant positive deviation values, the distribution function decreases more slowly at a distance from the average than with small values. If the deviation is more than three, the distribution density plot will be higher than the normal distribution plot and lower than three. This indicates that the processes under consideration not only have non-locality in time t, but also non-locality in state—let us designate it as x.

## 3. Setting a Study Objective

Studies devoted to the analysis of the dynamics and forecasting of the development of processes in complex systems are of high importance both from the point of view of science and from a practical perspective. A variety of studies have been conducted in this area [36–45]. Of these, we highlight the use of the theory and methods of neural networks [36–40], the use of the fuzzy logic apparatus [41], and the creation of non-parametric models based on chaos theory and methods of supporting regression vectors [42]. Also, the development of sets of rules based on genetic algorithms [43,44], as well as the use of self-organizing adaptive models [45] should be distinguished.

However, these studies have not taken into account the problems associated with studying processes in complex systems. Complex systems such as socioeconomic systems can be defined as structures involving at least one human element. On the one hand, they are characterized by stochasticity due to the influence of various random factors. In addition, there is uncertainty associated with the sometimes irrational behavior of people. However, on the other hand, the presence of the human factor also creates conditions for self-organization in such systems and can enable the storage of a memory of previous states of the processes occurring in these systems.

The joint action of these factors leads to the emergence of organized complexity, or “emergence”. Its origin cannot be reduced to a simple addition of element characteristics; it is the result of the formation of system relationships and adaptive redistribution of functions between elements.

The presence of all the listed features of complex systems leads to the need to find new approaches and methods for analyzing and modeling the dynamics of the processes observed in them.

In conclusion, we note that non-local processes in which the transition to a given state x depends not only on the local characteristics or behavior of the system in the vicinity of the point in question at a given moment in time, but also from the values taken by it over the entire studied interval of the values of the series levels at any time, which can be described on the basis of differential equations with fractional partial derivatives along t and x (and not only of the diffusion type).

Taking into account all the above, we can conclude that the development of new methods for describing the dynamics of stochastic processes based on differential equations with fractional partial derivatives may allow us to take into account the presence of self-organization and memory (not Markov processes that take into account the consequences).

Sociodynamic processes are widespread and have a wide variety of manifestations. One of the most important objects where sociodynamic processes are observed is the Internet, particularly, news and blogs, under which, users of social networks and mass media leave their comments or express their emotional attitude towards them. These are one of the most important online phenomena and can act as indicators of public opinion and mood.

## 4. Theoretical Model

### *4.1. Setting and Solving the Boundary Problem for Arbitrary Values of Derivatives of Diffusion-Type Fractional-Differential Equations*

Given that the processes observed in complex systems have features that vary over a wide range of values, for the construction of the model, it is impossible to consider only the non-locality of the state of the system $x$ (variable describing the level of the time series) or only non-locality of time $t$. In addition, it is necessary to consider a more general case with arbitrary fractional values of $\alpha$ and $\beta$, and not just the case of $0 < \beta \leq 1$ and $1 \leq \alpha \leq 2$. Or, we should consider differential equations containing multiple fractional derivatives, both in time and in state (this will be discussed later in one of the paragraphs in the section on the theoretical model).

To analyze and model the time series of processes observed in complex systems, you can consider the edge problem of the form

$$\rho(0,t) = \rho(L,t) = 0$$

with the initial condition given by the delta function:

$$\rho(x,0) = \delta(x-0)$$

The choice of such boundary conditions is due to the fact that when considering, for example, the dynamics of the amplitudes of change in the levels of the time series (in economics, this is called volatility), the state of $x$ can change within a certain permissible range of values from 0 to L, and at the time of the beginning of observations ($t = 0$), any change must be equal to 0. The $\rho$ function $(x,\ t)$ can be considered as the probability density for the amplitude x over the time interval t.

Work [46] considers the solution of the boundary problem (on the segment [0, 1]) based on Equation (1) at $\beta = 1$: $\rho(0,t) = \rho(1,t) = 0$ with the initial condition given by the delta function $\rho(x,0) = \delta(x-0)$.

$$\rho(x,t) = \sum_{n=1}^{\infty} C_n e^{\lambda_n D\cdot t}\cdot x^{\alpha-1} \sum_{k=0}^{\infty} \frac{\{\lambda_n x^{\alpha}\}^k}{\Gamma(\alpha[k+1])} \tag{8}$$

where $C_n$ are the coefficients of the Fourier series and $\lambda_n$ are the zeros of the Leffler-Mittag function: $E_{\alpha,\alpha}(\lambda_k) = \sum\limits_{k=0}^{\infty} \frac{\{\lambda_k\}^k}{\Gamma(\alpha[k+1])}$.

To determine the $C_n$ coefficients [46],

$$\varphi(x) = \rho(x,0) = \sum_{n=1}^{\infty} C_n x^{\alpha-1} \sum_{k=0}^{\infty} \frac{\{\lambda_n x^{\alpha}\}^k}{\Gamma(\alpha[k+1])}$$

Note that the function system $\theta_n = \left\{ x^{\alpha-1} \sum\limits_{k=0}^{\infty} \frac{\{\lambda_n x^{\alpha}\}^k}{\Gamma(\alpha[k+1])} \right\}_{k=1}^{\infty}$ forms a basis in $L_2$ (0, 1) [47]. Since it is not orthogonal, it is necessary to also create a system of functions

$$\phi_n(x) = \left\{ (1-x)^{\alpha-1} \sum_{k=0}^{\infty} \frac{\{\lambda_n (1-x)^{\alpha}\}^k}{\Gamma(\alpha[k+1])} \right\}_{n=1}^{\infty}$$

that are biorthogonal to the $\theta_n$ system [48].

After that, unknown $C_n$ coefficients can be determined through the dot product $\phi_n$ and $\varphi(x)$ [46]: $(\varphi(x)\cdot\phi_n(x))$ using the initial condition.

It should be noted that the determination of the $C_n$ coefficients and zeros of the Leffler-Mittag function performed in work [46] was not carried out, but the existence of a solution to the boundary problem formulated by the authors was investigated in a general form.

Consider the solution of the formulated boundary problem for the case of arbitrary fractional values of $\alpha$ and β (not only the case of $\beta = 1$ or $0 < \beta \leq 1$ and $1 \leq \alpha \leq 2$).

In this case, the equations obtained in [2–11,46] cannot be used and it is necessary to look for other solutions. In this regard, we suggest the following. First, using the Fourier method, imagine the $\rho(x,t)$ as $\rho(x,t) = X(x)\cdot T(t)$. After substituting into Equation (1) and separating the variables, we obtain

$$\frac{\partial^{\alpha} X(x)}{\partial x^{\alpha}} + \frac{\lambda}{D} X(x) = 0 \tag{9a}$$

$$\frac{\partial^{\beta} T(t)}{\partial t^{\beta}} + \lambda T(t) = 0 \tag{9b}$$

where $\lambda$ is some constant that appears due to the fact that after substituting $\rho(x,t) = X(x) \cdot T(t)$ into Equation (1), the left side will depend only on the variable $t$, and the right only on the variable $x$.

To define the $X(x)$ function, we encountered an problem finding eigenvalues. The solution to this problem is presented in the works [21,49,50]. In these studies, a solution has only been proven for eigenvalues of $\lambda_n$ that are function zeros:

$$E_{\alpha,\alpha}(\lambda_n x^{\alpha}) = \sum_{n=0}^{\infty} \frac{\left\{\frac{\lambda_n}{D} \cdot x^{\alpha}\right\}^n}{\Gamma(\alpha[n+1])}$$

There are eigenfunctions that are particular solutions of Equation (9a). Zeros can be found using the given boundary conditions $\rho(0,t) = \rho(L,t) = 0$. The eigenfunctions are equal to

$$X_n = \left\{ x^{\alpha-1} \sum_{n=0}^{\infty} \frac{\left\{\frac{\lambda_n}{D} \cdot x^{\alpha}\right\}^n}{\Gamma(\alpha[n+1])} \right\}_{n=1}^{\infty}$$

where $\lambda_n$ are the zeros of the Leffler-Mittag function.

Using the Laplace transform method for the function $T(t)$, one can write

$$p^{\beta}\overline{G(p)} - \frac{1}{p^{1-\beta}} - \lambda\overline{G(p)} = 0$$

Next,

$$\overline{G(p)} = \frac{1}{p^{1-\beta}} \cdot \frac{1}{p^{\beta} - \lambda} = \frac{1}{p} \cdot \frac{1}{1 - \frac{\lambda}{p^{\beta}}} = \frac{1}{p} \sum_{q=0}^{\infty} \left\{ -\frac{\lambda}{p^{\beta}} \right\}^q = \sum_{q=0}^{\infty} \frac{(-1)^q \lambda^q}{p^{\beta+1}}$$

Let us make the inverse Laplace transform and go from $p$ to $t$:

$$T_n(t) = \sum_{q=0}^{\infty} \frac{(-1)^q \lambda_n^q t^{\beta q}}{\Gamma(\beta q + 1)}$$

Thus, the general solution to the formulated edge problem can be written as

$$\rho(x,t) = x^{\alpha-1} \sum_{n=0}^{\infty} C_n \sum_{q=0}^{\infty} \frac{(-1)^q \lambda_n^q t^{\beta q}}{\Gamma(\beta q + 1)} \frac{\left\{\frac{\lambda_n}{D} \cdot x^{\alpha}\right\}^n}{\Gamma(\alpha[n+1])} \tag{10}$$

To find the zeros of the Mittag-Leffler function, you can use the boundary condition $\rho(L,t) = 0$ and the results obtained in [51–55], which examined the behavior of the zeros of this function for various $\alpha$ values, which allows us, taking into account the boundary condition, to obtain the following result.

In the case of $\alpha < 2$, $\alpha \in C$, zeros are $\lambda_n$ and determined using the equation

$$\lambda_n^{\pm} = e^{\pm i \frac{\pi}{2} \alpha} \left\{ \frac{2\pi n}{L} \right\}^{\alpha} \left\{ 1 + O\left\{ \frac{\log n}{n} \right\} \right\}$$

When $\alpha$ = 2,

$$\lambda_n = \left\{ \frac{\pi n}{L} \right\}^2 \left\{ 1 + O\left\{ \frac{1}{n} \right\} \right\}$$

In the case of $\alpha > 2,\ \alpha \in C$, all $z_n$ functions are large enough modulo zeros that $E_{\alpha,\alpha}(z) = \sum_{n=0}^{\infty} \frac{\{z\}^n}{\Gamma(\alpha[n+1])}$ and are described by the equation

$$z_n = \lambda_n L^{\alpha} = -\left\{ \frac{\pi}{\sin\left[\frac{\pi}{\alpha}\right]} \left\{ n + 1/2 + \frac{\alpha - 1}{\alpha} \right\} + \Omega_n \right\}^{\alpha}$$

At 6 > $\alpha$ > 2, $\Omega_n = O\left\{ n^{-\alpha \cdot \tau} \cdot e^{-\pi n \cdot ctg\left[\frac{\pi}{\alpha}\right]} \right\}$

$$\tau = \begin{cases} \frac{1}{\alpha},\ n \neq 0 \\ \frac{1+\alpha}{\alpha},\ n = 0 \end{cases}$$

At $\alpha$ = 6, $\Omega_n = O\left\{ e^{-\pi n \cdot ctg\left[\frac{\pi}{\alpha}\right]} \right\}$

At $\alpha$ > 6, $\Omega_n = O\left\{ \frac{e^{-\pi n \cdot \left[cos\left[\frac{\pi}{\alpha}\right] - cos\left[\frac{3\pi}{\alpha}\right]\right]}}{sin\left[\frac{\pi}{\alpha}\right]} \right\}$

Next, consider the definition of Fourier series coefficients $C_n$ through the dot product $(\varphi_n(x) \cdot \phi_m(x))$, where

$$\varphi_n(x) = C_n x^{\alpha-1} \cdot \sum_{n=0}^{\infty} \frac{\{\lambda_n x^{\alpha}\}^n}{\Gamma(\alpha[n+1])}$$

$$\phi_m(x) = (L - x)^{\alpha-1} \sum_{m=0}^{\infty} \frac{\{\lambda_n (L - x)^{\alpha}\}^m}{\Gamma(\alpha[m+1])}$$

Using the biorthogonality property of functions $\varphi(x)$ and $\phi_n$ in $L_2(0,\ L)$,

$$\int\limits_0^L \varphi_n(x) \cdot \phi_m(x) dx = \begin{cases} 1,\ n = m \\ 0, n \neq m \end{cases}$$

Next, we assume that with a weight equal to 1 on the segment [0, L], the following condition should be met:

$$\int\limits_0^L \varphi_n(x) \cdot \phi_m(x) dx = \int\limits_0^L \delta(x - 0) \cdot \phi_m(x) dx$$

This allows you to define $C_n$:

$$C_n = \frac{L^{\alpha-1} \sum_{m=0}^{\infty} \frac{\{\lambda_n L^{\alpha}\}^k}{\Gamma(\alpha[k+1])}}{\int_0^L x^{\alpha-1} \cdot (L - x)^{\alpha-1} \sum_{n=0}^{\infty} \frac{\{\lambda_n x^{\alpha}\}^n}{\Gamma(\alpha[n+1])} \cdot \sum_{m=0}^{\infty} \frac{\{\lambda_n (L-x)^{\alpha}\}^m}{\Gamma(\alpha[m+1])} dx}$$

### *4.2. Derivation of Telegraph-Type Fractional-Differential Equation with Multiple Orders of Fractional Derivatives from Consideration of Probability Schemes of Transitions between Process States*

To derive the basic equation of our proposed model, we can use the approach to describe the stochastic dynamics of processes, taking into account memory, as well as possible self-organization, which we developed earlier and is described in works [30,31].

This method allows us, based on the schemes of probabilistic transitions between states, to formulate a boundary value problem regarding the probability of achieving any of the states x as a function of time $t$. One can then consider solving this problem (to obtain a theoretical approximating distribution function) based on a model that takes into account the memory about previous states and their potential self-organization.

The essence of this approach is as follows. Let us designate the current state as $x_i$ (process state). Suppose the state change time interval is $\tau$ (extremely small). It is assumed that during this time period, the $\tau$ state of the system may increase by a value of $\varepsilon$ (indicating an increasing trend) or decrease by a value of $\xi$ (indicating a decreasing trend). Let us represent the entire set of states as $X$. The state observed at time $t$ can be denoted as $x_i$ $(x_i X)$, where x represents the level of values in the time series describing the observed process. We express the value of the current time as $t = h\tau$, where $h$ is the number of transition steps between states (the transition process between states becomes quasi-continuous with an infinitesimal time interval $\tau$), and $h$ takes values of 0, 1, 2, 3, N. The current state $x_i$ at step h can increase by some amount of $\varepsilon$ or decrease by an amount of $\xi$ after the transition in step $(h + 1)$, and accordingly, turn out to be equal to $(x_i + \varepsilon)$, or $(x_i - \xi)$.

Consider the concept of the probability of finding a process in a certain state. Suppose, after a certain number of steps $h$ in the described process, it can be argued that

1. $P(x - \varepsilon, h)$—probability that it is in the state of $(x - \varepsilon)$;
2. $P(x, h)$—probability that it is in the state of $x$;
3. $P(x + \xi, h)$—probability that it is in the state of $(x + \xi)$.

Following each step, the state $x_i$ (hereinafter, the index i may be omitted for brevity) can change by an amount of $\varepsilon$ or $\xi$.

The probability of $P(x,\ h + 1)$—indicating the likelihood of the process being in the state $x$ at the next $(h + 1)$ step—will be influenced by multiple transitions (see Figure 1):

$$P(x, h + 1) = P(x - \varepsilon, h) + P(x + \xi, h) - P(x, h) \tag{11}$$

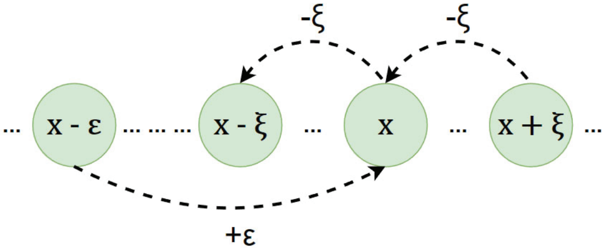


**Figure 1.** Diagram of possible transitions between process states in $h$ + 1 step.

Let us explain the Expression (11) and the diagram shown in Figure 1. The probability of transition to state $x$ in step $h$, denoted as $P(x,\ h + 1)$, is determined by the sum of the probabilities of transition to this state from states $(x - \varepsilon) :\ P(x - \varepsilon, h)$ and $(x + \xi) :\ P(x + \xi, h)$ where the system was at step $h$. From this sum, we subtract the probability of the system transitioning ($P(x, h)$) from state $x$ (where it was at step $h$) to any other state at step $h$ + 1.

In this context, we consider a Markov continuous process devoid of state memory. However, in real conditions, it is possible to store information about the previous state. To take into account memory (not Markov processes), we will determine the probabilities $P(x–\varepsilon,\ h)$, $P(x + \xi, h)$ and $P(x, h)$ through the states in the previous $h - 1$ step. In this case, the following algebraic equation can be obtained for the probability of transition:

$$P(x, h + 2) = P(x - 2\varepsilon, h) + P(x + 2\xi, h) + P(x, h) + 2\{P(x - [\varepsilon - \xi], h) - P(x - \varepsilon, h) - P(x + \xi, h)\} \tag{12}$$

In this instance, the parameter h is augmented by the quantity $m$ = 2.

Having carried out the necessary mathematical actions and considering that $t = h{\cdot}\tau$ where $t$ is the process time, $h$ is the step number, $\tau$ is the duration of one step can be obtained for any arbitrary value $m$, the following recurrent expression is the probability $P\ (x, t + m\tau)$ that the process state, for some time $t$, at memory depth $m$, will be equal to $x$:

$$P(x, t + m\tau) = \begin{cases} \sum\limits_{k,l=0}^{m} (-1)^{m-k-l} \frac{m!\cdot \mathrm{P}(x - [k\cdot\varepsilon - l\cdot\xi],\ t)}{k!\cdot l!(m-k-l)!},\ \text{при } m - k - l \geq 0 \\ 0,\ \text{при } m - k - l < 0 \end{cases} \tag{13}$$

If we differentiate Equation (13) by the variable $x$, we obtain the equation for the probability density:

$$\rho(x, t + m\tau) = \begin{cases} \sum\limits_{k,l=0}^{m} (-1)^{m-k-l} \frac{m!\cdot\rho(x-[k\cdot\varepsilon-l\cdot\xi],\, t)}{k!\cdot l!(m-k-l)!}, \text{ при } m-k-l \geq 0 \\ 0, \text{ при } m-k-l < 0 \end{cases} \tag{14}$$

In order to take into account the previously described properties of the observed processes, it is necessary to move from the obtained algebraic equation for probability density to a differential equation with derivatives of fractional orders (derivatives of Caputo). To accomplish this, we conduct the corresponding decompositions of the terms of this equation into a Taylor series over the derivatives of fractional order [16] in the vicinity of the transition point.

$$\rho(x, t + m\tau) = \rho(x, t) + \frac{\{m\tau\}^{\beta}}{\Gamma(\beta+1)} \cdot \frac{\partial^{\beta}\rho(x,t)}{\partial t^{\beta}} + \frac{\{m\tau\}^{2\beta}}{\Gamma(2\beta+1)} \cdot \frac{\partial^{2\beta}\rho(x,t)}{\partial t^{2\alpha}} + \cdots$$

$$\rho(x - [k\cdot\varepsilon - l\cdot\xi], t) = \rho(x, t) + (-1)^{\alpha}\frac{[k\cdot\varepsilon - l\cdot\xi]^{\alpha}}{\Gamma(\alpha+1)} \cdot \frac{\partial^{\alpha}\rho(x,t)}{\partial x^{\alpha}} + (-1)^{2\alpha}\frac{[k\cdot\varepsilon - l\cdot\xi]^{2\alpha}}{\Gamma(2\alpha+1)} \cdot \frac{d^{2\alpha}\rho(x,t)}{dx^{2\alpha}} + \cdots$$

Note that for fractional derivatives of $\beta$ and $\alpha$, the equation has multiples of orders $\beta$, $2\beta$, $3\beta$ and $\alpha$, $2\alpha$, $3\alpha$.

We substitute the corresponding expansions into Equation (14) and obtain, taking into account no more than the second derivatives (not to be confused with the order), the following differential equation for changing the density of the probability of detecting a process in some state $x$ depending on the value of time $t$ and memory depth $m$:

$$\rho(x,t) + \frac{\{m\tau\}^{\beta}}{\Gamma(\beta+1)} \cdot \frac{\partial^{\beta}]\rho(x,t)}{\partial t^{\beta}} + \frac{\{m\tau\}^{2\beta}}{\Gamma(2\beta+1)} \cdot \frac{\partial^{2\beta}\rho(x,t)}{\partial t^{2\beta}} = \rho(x,t) + \sum_{k,l=0}^{m} \frac{(-1)^{m-k-l}\cdot m!}{k!\cdot l!(m-k-l)!} \left\{ \begin{array}{c} (-1)^{\alpha}\frac{[k\cdot\varepsilon-l\cdot\xi]^{\alpha}}{\Gamma(\alpha+1)} \cdot \frac{\partial^{\alpha}\rho(x,t)}{\partial x^{\alpha}} + \\ +(-1)^{2\alpha}\frac{[k\cdot\varepsilon-l\cdot\xi]^{2\alpha}}{\Gamma(2\alpha+1)} \cdot \frac{d^{2\alpha}\rho(x,t)}{dx^{2\alpha}} \end{array} \right\}$$

Or

$$\frac{\{m\tau\}^{\beta}}{\Gamma(\beta+1)} \cdot \frac{\partial^{\beta}\rho(x,t)}{\partial t^{\beta}} + \frac{\{m\tau\}^{2\beta}}{\Gamma(2\beta+1)} \cdot \frac{\partial^{2\beta}\rho(x,t)}{\partial t^{2\beta}} = \sum_{k,l=0}^{m} \frac{(-1)^{m-k-l}\cdot m!}{k!\cdot l!(m-k-l)!} \left\{ \begin{array}{c} (-1)^{\alpha}\frac{[k\cdot\varepsilon-l\cdot\xi]^{\alpha}}{\Gamma(\alpha+1)} \cdot \frac{\partial^{\alpha}\rho(x,t)}{\partial x^{\alpha}} + \\ +(-1)^{2\alpha}\frac{[k\cdot\varepsilon-l\cdot\xi]^{2\alpha}}{\Gamma(2\alpha+1)} \cdot \frac{d^{2\alpha}\rho(x,t)}{dx^{2\alpha}} \end{array} \right\} \tag{15}$$

The resulting equation contains two derivatives in time and two in state, which allows you to characterize it as a telegraph-type equation. It is interesting in that it allows you to take into account previous states due to different m values and the influence of the fractional derivatives of $\alpha$ and $\beta$ of different orders.

For a member of a view equation $\frac{\partial^{\beta}\rho(x,t)}{\partial t^{\beta}}$, if the rate of change of process states changes over time, then a term of the equation of the form $\frac{\partial^{2\beta}\rho(x,t)}{\partial t^{2\beta}}$ can be, by analogy using physical kinetics, seen as acceleration. In this interpretation, the term $\frac{\partial^{2\alpha}\rho(x,t)}{\partial x^{2\alpha}}$ of the equation takes into account random transitions (diffusion wandering of the state of the system), and the term $\frac{\partial^{\alpha}\rho(x,t)}{\partial x^{\alpha}}$ will describe ordered transitions (trend or demolition), for example, or when the value of the state increases ($\varepsilon > \xi$), or decreases ($\varepsilon < \xi$).

A significant difference between Equation (15) and fractional-diffusion and fractional wave equations is that the resulting equation, being stochastic (derived from the consideration of probability schemes of transitions between states), contains multiple orders of fractional derivatives, both in time ($\beta$, $2\beta$) and state ($\alpha$ and $2\alpha$). This is a novelty that allows you to expand the class of fractional-differential equations and their application to describe the dynamics of complex systems. The Zener equation [25] contains only fractional time operators, and the fractional diffusion equations do not contain multiple derivatives.

Solving Equation (15) generally requires a separate study, and a simpler case with arbitrary values can be considered to begin with $\beta$, $\alpha = 1$ and $\varepsilon = \xi$.

Considering that

$$\sum_{k,l=0}^{m} (-1)^{m-k-l} \frac{m!}{k!\cdot l!(m-k-l)!} = 1$$

$$\sum_{k,l=0}^{m} (-1)^{m-k-l} \frac{m!}{k!\cdot l!(m-k-l)!}\cdot k = \sum_{k,l=0}^{m} (-1)^{m-k-l} \frac{m!}{k!\cdot l!(m-k-l)!}\cdot l = m$$

$$\sum_{k,l=0}^{m} (-1)^{m-k-l} \frac{m!\cdot k^2}{k!\cdot l!(m-k-l)!} = \sum_{k,l=0}^{m} (-1)^{m-k-l} \frac{m!\cdot l^2}{k!\cdot l!(m-k-l)!} = m^2$$

$$\sum_{k,l=0}^{m} (-1)^{m-k-l} \frac{m!\cdot k\cdot l}{k!\cdot l!(m-k-l)!} = m(m-1)$$

Then,

$$\frac{\{m\tau\}^{\beta}}{\Gamma(\beta+1)}\cdot\frac{\partial^{\beta}\rho(x,t)}{\partial t^{\beta}} + \frac{\{m\tau\}^{2\beta}}{\Gamma(2\beta+1)}\cdot\frac{\partial^{2\beta}\rho(x,t)}{\partial t^{2\beta}} = \frac{1}{2}\left\{m^2\varepsilon^2 - 2m(m-1)\varepsilon\xi + m^2\xi^2\right\}\frac{\partial^2\rho(x,t)}{\partial x^2} - m[\varepsilon-\xi]\frac{\partial\rho(x,t)}{\partial x} \tag{16}$$

If any state transitions have the same value, then $\varepsilon = \xi$.

$$\frac{\{m\tau\}^{\beta}}{\Gamma(\beta+1)}\cdot\frac{\partial^{\beta}\rho(x,t)}{\partial t^{\beta}} + \frac{\{m\tau\}^{2\beta}}{\Gamma(2\beta+1)}\cdot\frac{\partial^{2\beta}\rho(x,t)}{\partial t^{2\beta}} = m\varepsilon^2\frac{\partial^2\rho(x,t)}{\partial x^2} \tag{17}$$

### *4.3. Setting and Solving the Boundary Problem for a Particular Case using Telegraph-Type Fractional-Differential Equation (Arbitrary $\beta$ and $\alpha = 2$)*

Consider for Equation (17) the solution of the previously formulated boundary problem $\rho(0,t) = \rho(L,t) = 0$ with the initial condition given by the delta function: $\rho(x,0) = \delta(x-0)$.

Using the Fourier method, we present the $\rho(x,t)$ as $\rho(x,t) = X(x)\cdot T(t)$. After substituting Equation (17) and separating the variables, we obtain

$$\frac{\partial^2 X(x)}{\partial x^2} + \frac{\lambda}{m\varepsilon^2}X(x) = 0 \tag{18a}$$

$$\frac{\{m\tau\}^{\beta}}{\Gamma(\beta+1)}\cdot\frac{\partial^{\beta}\rho(x,t)}{\partial t^{\beta}} + \frac{\{m\tau\}^{2\beta}}{\Gamma(2\beta+1)}\cdot\frac{\partial^{2\beta}\rho(x,t)}{\partial t^{2\beta}} + \lambda T(t) = 0 \tag{18b}$$

where $\lambda$ is some constant that appears as a result of substituting $\rho$ $(x,t)$ = $X(x)$ · $T(t)$ into Equation (17). As a result of this substitution, the left side becomes dependent only on the variable t, and the right side only on the variable $x$. Then, you can find partial solutions of Equation (18a,b), and the general solution is represented in the form of a Fourier series.

To determine the function $X(x)$, we must solve the eigenvalue problem (Sturm–Liouville problem). This solution under given boundary conditions is well known and has the form $X_n(x) = C_n\sin(\lambda_n x)$, where $\lambda_n$ are eigenvalues ($\lambda_n = \pi n/L,\ n = 0, 1, 2, 3\ldots$).

To find a solution for $T_n(t)$, we perform the Laplace transform according to t:

$$\frac{\{m\tau\}^{\beta}}{\Gamma(\beta+1)}\left\{p^{\beta}\overline{G(p,x)} - \frac{1}{p^{1-\beta}}\right\} + \frac{\{m\tau\}^{2\beta}}{\Gamma(2\beta+1)}\left\{p^{2\beta}\overline{G(p,x)} - \frac{1}{p^{1-2\beta}}\right\} + \lambda\overline{G(p,x)} = 0$$

Next,

$$\overline{G(p,x)} = \frac{\frac{\{m\tau\}^{2\beta}}{p^{1-2\beta}\cdot\Gamma(2\beta+1)} + \frac{\{m\tau\}^{\beta}}{p^{1-\beta}\cdot\Gamma(\beta+1)}}{\frac{\{m\tau\}^{2\beta}p^{2\beta}}{\Gamma(2\beta+1)} + \frac{\{m\tau\}^{\beta}p^{\beta}}{\Gamma(\beta+1)} + \lambda} = \frac{1}{p}\cdot\frac{a_1(p)}{a_1(p)+\lambda} = \frac{1}{p}\cdot\frac{1}{1+\frac{\lambda}{a_1(p)}}$$

$$a_1(p) = \frac{\{m\tau\}^{2\beta} p^{2\beta}}{\Gamma(2\beta+1)} + \frac{\{m\tau\}^{\beta} p^{\beta}}{\Gamma(\beta+1)}$$

Let us designate $z = \frac{\lambda}{a_1(p)}$ and if $0 \leq z < 1$, we will carry out binomial decomposition:

$$\frac{1}{1-(-z)} = \sum\nolimits_{q=0}^{\infty} (-z)^q = \sum\nolimits_{q=0}^{\infty} \left\{-\frac{\lambda}{a_1(p)}\right\}^q = \sum\nolimits_{q=0}^{\infty} (-1)^q \left\{\frac{\lambda}{a_1(p)}\right\}^q =$$

$$= \sum_{q=0}^{\infty} (-1)^q \left\{ \frac{\lambda}{\frac{\{m\tau\}^{2\beta} p^{2\beta}}{\Gamma(2\beta+1)} + \frac{\{m\tau\}^{\beta} p^{\beta}}{\Gamma(\beta+1)}} \right\}^q = \sum_{q=0}^{\infty} (-1)^q \frac{\lambda^q}{\frac{\{m\tau\}^{2\beta q} p^{\beta q}}{\{\Gamma(2\beta+1)\}^q} \left\{ p^{\beta} + \frac{\Gamma(2\beta+1)}{\{m\tau\}^{\beta}\Gamma(\beta+1)} \right\}^q}$$

After substituting the result into the equation for $\overline{G(p,x)}$, we obtain

$$\overline{G(p,x)} = \sum_{q=0}^{\infty} \frac{(-1)^q \lambda^q \{\Gamma(2\beta+1)\}^q}{\{m\tau\}^{2\beta q}} \cdot \frac{p^{-\beta q-1}}{\left\{ p^{\beta} + \frac{\Gamma(2\beta+1)}{\{m\tau\}^{\beta}\Gamma(\beta+1)} \right\}^q} = \sum_{q=0}^{\infty} \frac{(-1)^q \lambda^q \{\Gamma(2\beta+1)\}^q}{\{m\tau\}^{2\beta q}} \cdot \frac{p^{-\beta q-1}}{\{p^{\beta} + \Omega\}^q}$$

$$\frac{\Gamma(2\beta+1)}{\{m\tau\}^{\beta}\Gamma(\beta+1)} = \Omega$$

We perform the inverse Laplace transform and move from the image $\frac{p^{-\beta q-1}}{\{p^{\beta}+\Omega\}^q}$ by p to its original by $t$. Consider the integral for this:

$$\begin{gathered} \int_0^{\infty} e^{-pt} \cdot t^{b-1} \cdot \sum_{j=0}^{\infty} \frac{(-\Omega t^{\mu})^j}{\Gamma(\mu j+b)} dt = \sum_{j=0}^{\infty} \frac{(-\Omega)^j}{\Gamma(\mu j+b)} \cdot \int_0^{\infty} t^{\mu j+b-1} \cdot e^{-pt} dt = \\ \sum_{j=0}^{\infty} \frac{(-\Omega)^j}{\Gamma(\mu j+b)} \cdot \int_0^{\infty} \frac{1}{p} \cdot \frac{y^{\mu j+b-1}}{p^{\mu j+b-1}} \cdot e^{-y} dy = \sum_{j=0}^{\infty} \frac{(-\Omega)^j}{p^{\mu j+b}} = \sum_{j=0}^{\infty} \frac{1}{p^b} \cdot \left\{-\frac{\Omega}{p^{\mu}}\right\}^j = \\ \frac{1}{p^b} \cdot \frac{1}{1+\frac{\Omega}{p^{\mu}}} = \frac{p^{\mu-b}}{p^{\mu}+\Omega} \end{gathered} \tag{19}$$

where $\sum\limits_{j=0}^{\infty} \frac{(-\Omega t^{\mu})^j}{\Gamma(\mu j+b)}$ is the Mittag–Leffler function, $\mu$ and $b$ are some real numbers greater than 0, and $t$ is a variable. When calculating the integral $\int_0^{\infty} t^{\mu j+b-1} \cdot e^{-pt} dt$, replacement was used: $pt = y$. It should also be borne in mind that

$$\int_0^{\infty} y^{\mu q+b-1} \cdot e^{-y} dy = \Gamma(\mu q+b)$$

We differentiate the obtained Expression (19) on the right and left by $\Omega$ $\nu$ = ($q$ − 1) times (here, the derivative of the integer is calculated, not a fractional order):

$$\int_0^{\infty} e^{-pt} \cdot t^{b-1} \cdot \left\{ \frac{d^{(q-1)}}{d\Omega^{(q-1)}} \sum_{j=q-1}^{\infty} \frac{(-\Omega t^{\mu})^j}{\Gamma(\mu j+b)} \right\} dt = (-1)^{q-1} \frac{(q-1)! \cdot p^{\mu-b}}{\{p^{\mu}+\Omega\}^q}$$

Next,

$$\int_0^{\infty} e^{-pt} \cdot t^{b-1} \cdot \sum_{j=0}^{\infty} \frac{(j+\nu)!(-t^{\mu})^{j+\nu}\Omega^j}{j! \cdot \Gamma(\mu j+\mu\nu+b)} dt = \int_0^{\infty} e^{-pt} \cdot t^{b-1} \cdot \sum_{j=0}^{\infty} \frac{(j+q-1)!(-t^{\mu})^{j+q-1}\Omega^j}{j! \cdot \Gamma\{\mu j+\mu(q-1)+b\}} dt = (-1)^{q-1} \frac{(q-1)! \cdot p^{\mu-b}}{\{p^{\mu}+\Omega\}^q}$$

Equating expressions $\frac{p^{\mu-b}}{\{p^\mu+\Omega\}^q}$ and $\frac{p^{-\beta q-1}}{\{p^\beta+\Omega\}^q}$, we find that at $\mu=\beta$ and $b=\beta(q+1)+1$, the following equality is performed:

$$\int_0^\infty e^{-pt}\cdot t^{\beta(q+1)}\cdot\sum_{j=0}^{\infty}\frac{(j+q-1)!(-t^\beta)^{j+q-1}\Omega^j}{j!\cdot\Gamma\{\beta(2q+j)+1\}}dt=(-1)^{q-1}\frac{(q-1)!\cdot p^{-\beta q-1}}{\{p^\alpha+\Omega\}^q}$$

Next, $(-1)^{q-1}\frac{(q-1)!\cdot p^{-\beta q-1}}{\{p^\alpha+\Omega\}^q}$ is equated to the expression

$$t^{\beta(q+1)}\cdot\sum_{j=0}^{\infty}\frac{(j+q-1)!(-t^\beta)^{j+q-1}\Omega^j}{j!\cdot\Gamma\{\beta(2q+j)+1\}}=(-1)^{q-1}\cdot t^{2\beta q}\sum_{j=0}^{\infty}\frac{(j+q-1)!(-\Omega t^\beta)^j}{j!\cdot\Gamma\{\beta(2q+j)+1\}}$$

Next, we find

$$(-1)^{q-1}\cdot t^{2\beta q}\sum_{j=0}^{\infty}\frac{(j+q-1)!(-\Omega t^\beta)^j}{j!\cdot\Gamma\{\beta(2q+j)+1\}}\fallingdotseq(-1)^{q-1}\frac{(q-1)!\cdot p^{-\beta q-1}}{\{p^\beta+\Omega\}^q}$$

Or, rewrite in the form

$$\frac{p^{-\beta q-1}}{\{p^\beta+\Omega\}^q}\fallingdotseq\frac{t^{2\beta q}}{(q-1)!}\sum_{j=0}^{\infty}\frac{(j+q-1)!(-\Omega t^\beta)^j}{j!\cdot\Gamma\{\beta(2q+j)+1\}}=t^{2\beta q}\left\{\frac{1}{\Gamma\{2\beta q+1\}}+\sum_{j=1}^{\infty}\frac{(-\Omega t^\beta)^j\prod_{i=1}^{j}(q+i-1)}{j!\cdot\Gamma\{\beta(2q+j)+1\}}\right\}$$

After all the necessary substitutions, we obtain

$$T_n(t)=\sum\nolimits_{q=0}^{\infty}(-1)^q\lambda_n^q\{\Gamma(2\beta+1)\}^q\cdot\left\{\frac{t}{m\tau}\right\}^{2\beta q}\left\{\frac{1}{\Gamma\{2\beta q+1\}}+\sum\nolimits_{j=1}^{\infty}\frac{(-\Omega t^\beta)^j\prod_{i=1}^{j}(q+i-1)}{j!\cdot\Gamma\{\beta(2q+j)+1\}}\right\}$$

$$\lambda_n=\pi n/L\ (n=0,1,2,3),\ \Omega=\frac{\Gamma(2\beta+1)}{\{m\tau\}^\beta\Gamma(\beta+1)}$$

Accordingly, the general solution for $\rho(x,t)$ is

$$\rho(x,t)=\sum_{n=0}^{\infty}C_nT_n(t)\cdot X_n(x)$$

Using the initial condition and orthogonality property of the function $X_n(x)=C_n\sin(\lambda_nx)$, we find that $C_n=2/L$.

## 5. Conclusions

The dynamics of the behavior of complex systems is very complex, including the presence of memory and the possibility of self-organization. It should be noted that different processes of nature can be observed at the same time, and the driving causes of which may be of a hidden nature. It should be noted that to date, the dynamics of the behavior of various physical systems have been the most well studied. In many works devoted, for example, to the study of physical kinetics, modeling is carried out without the use of fractional-differential equations. In this regard, we can mention [56], in which, a dynamic renormal group analysis of the Burgers equation was carried out, which is a nonlinear generalization of the diffusion equation taking into account the influence of random noise on the behavior of the processes studied. The obtained results were applied to the description of a growing interface with the media during polymerization and the appearance of transverse fluctuations in the directed growth of a polymer in a random medium. The occurrence of noise may not be related to the diffusion process, but may have correlations with it in space and/or time. Weak and strong noise, according to the results obtained by the authors of the article [56], lead to different scaling indicators and the appearance of correlations. For spatial correlations with sizes less than critical values, any amount of noise matters, resulting in a strong link. In the absence of temporal correlations, two modes can be observed, with sizes smaller than the critical size, either hydrodynamic behavior is determined by white noise and correlations are not important, or correlations dominate.

To some extent, the processes studied can be processes with memory and self-organization at certain scales in terms of coordinates and time.

The solution of the generalized diffusion equation containing Cardara–Parisi–Zhang nonlinearity taking into account the influence of spatially correlated noise in combination with the long-range nature of interactions was considered in works [57,58], where an approach based on renorms of groups and phase diagrams was also used.

It is possible that in terms of considering correlations in space and/or time, the use of the Burgers equation or other generalized equations of nonlinear diffusion may be one of the alternatives to fractional-differential equations.

However, models based on fractional-differential equations do not require the consideration of characteristic scales and the introduction of critical dimensions, which, for example, can be an advantage when considering processes in social and economic systems.

Studies of complex social processes, for example, electoral campaigns and user activity on social networks, show that the time series observed in practice have fractality, and the systems whose dynamics they describe have memory and show self-organization. For example, if we analyze the dynamics of changes in mathematical expectations and dispersions of amplitudes of time series levels depending on the time interval for calculating these amplitudes (using a "sliding window"), complex dependencies are observed.

For example, for the mathematical expectation, there is a root dependence of a fractional degree and for dispersion, there dependence on a power law for a fractional exponent greater than 1.5. Fractional time dependencies indicate the presence of non-locality for this variable.

The examination of the excess shows the presence of the so-called "heavy tails", with its size significantly greater than that of the normal distribution. This excess behavior indicates the presence of non-locality in the states of the time series levels.

The obtained results indicate that the process under consideration has memory and the possibility of self-organization, and its time series have unsteady, as well as non-locality, both in time and state.

Non-local processes are characterized by the fact that the transition to a certain state of the system or process depends not only on the local characteristics of the process or the behavior of the system near the point in question at the current moment in time, but also on the values obtained throughout the studied interval at previous points in time, i.e., they are globally dependent on the distribution over all states and on the history of the process (memory). Non-locality over time affects the probability density at the initial point in time, which can lead to the phenomenon of self-organization, and non-locality in the state affects the asymptotic behavior of the probability density to detect a certain state x at time t at large intervals of time.

Various types of fractional partial derivative differential equations can be used to describe such processes. Currently, a fractional diffusion equation is used to build models of the dynamics of time series with non-locality over states and time: $\frac{\partial^{\beta}\rho(x,t)}{\partial t^{\beta}} = D\frac{\partial^{\alpha}\rho(x,t)}{\partial x^{\alpha}}$. Analytical or numerical process models based on this equation are obtained only for the case of $0 < \beta \leq 1$ and $1 \leq \alpha \leq 2$. Solutions outside the limits of these values of $\alpha$ and $\beta$ are not present in the literature.

Given that the processes observed in complex systems have features that vary over a wide range of values, it is impossible to only consider non-locality of the state of the system x (variable describing the level of the time series) or of time t for the construction of the model. In addition, it is necessary to consider a more general case with arbitrary fractional values of $\alpha$ and β, and not just the case of $0 < \beta \leq 1$ and $1 \leq \alpha \leq 2$, or consider differential equations containing multiple fractional derivatives, both in time and in state.

Based on the fractional-diffusion equation, an edge problem with arbitrary values of derivatives was formulated and solved. An analytical solution was obtained that can be used in practice to analyze the dynamics of time series of processes observed in complex systems.

In addition, a fractional-differential equation of the telegraph type with multiples (in time: $\beta$, $2\beta$, $3\beta \ldots$ and state: $\alpha$, $2\alpha$, $3\alpha \ldots$) by orders of fractional derivatives and its analytical solution for one particular boundary problem was considered.

In solving edge problems, the Fourier method was used. This makes it possible to represent the solution in the form of nested time series (one in time t, the second in state x), each of which is a function of the Mittag-Leffler type. The eigenvalues of the Mittag-Leffler function for describing states can be found using boundary conditions and the Fourier coefficient based on the initial condition and orthogonality conditions of the eigenfunctions.

The resulting solutions can be tested based on the observed time series. It is necessary to calculate the parameters of the developed models ($\alpha$, $\beta$, $\varepsilon$, $D$ values) and check the accuracy of the forecasts obtained.

The novelty of the models described in this article is that the differential stochastic equation obtained when considering the probability schemes of transitions between process states (time series levels) contains multiple orders of fractional derivatives, both in time ($\beta$, $2\beta$) and state ($\alpha$ and $2\alpha$). Also, for the obtained equation, the solution of one of the edge problems is considered. This allows you to expand the class of fractional-differential equations and their use to describe the dynamics of complex systems. The Zener equation [25] contains only fractional time operators, and the fractional diffusion equations do not contain multiple derivatives.

In addition, it is new that the article presents a solution for one of the edge problems fractionally—a diffusion-type equation $\frac{\partial^\beta \rho(x,t)}{\partial t^\beta} = D\frac{\partial^\alpha \rho(x,t)}{\partial x^\alpha}$ for arbitrary $\alpha$ and $\beta$ values—while existing solutions are limited to considering fractional derivative values of $0 < \beta \leq 1$ and $1 \leq \alpha \leq 2$.

**Author Contributions:** D.Z.: conceptualization, methodology, formal analysis, writing—review and editing; K.O.: methodology, visualization; V.K.: data curation, writing—original draft. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported by the Russian Science Foundation (RSF), grant no. 23-21-00153 "Analysis and modelling dynamics of transient time series of fractal processes with memory realization (aftereffection) and self-organization based on the use of differential equations with fractional derivatives".

**Data Availability Statement:** Data are contained within the article.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Miller, K.S.; Ross, B. *An Introduction to the Fractional Calculus and Fractional Differential Equations*; John Wiley & Sons. Inc.: New York, NY, USA, 1993.
2. Mainardi, F.; Luchko, Y.; Pagnini, G. The fundamental solution of the space-time fractional diffusion equation. *Fract. Calc. Appl. Anal.* **2001**, *4*, 153–192.
3. Gorenflo, R.; Iskenderov, A.; Luchko, Y. Mapping between solutions of fractional diffusion-wave equations. *Fract. Calc. Appl. Anal.* **2000**, *3*, 75–86.
4. Luchko, Y.; Gorenflo, R. Scale-invariant solutions of a partial differential equation of fractional order. *Fract. Calc. Appl. Anal.* **1998**, *1*, 63–78.
5. Gorenflo, R.; Mainardi, F. Fractional calculus: Integral and differential equations of fractional order. In *Fractals and Fractional Calculus in Continuum Mechanics*; Carpinteri, A., Mainardi, F., Eds.; Springer: Berlin/Heidelberg, Germany, 1997.
6. Buckwar, E.; Luchko, Y. Invariance of a partial differential equation of fractional order under the Lie group of scaling transformations. *J. Math. Anal. Appl.* **1998**, *227*, 81–97. [CrossRef]
7. Engler, H. Similarity solutions for a class of hyperbolic integrodifferentialequations. *Differ. Integral Equ.* **1997**, *10*, 815–840.
8. Fujita, Y. Integrodifferential equation which interpolates the heat and the wave equations. *Osaka J. Math.* **1990**, *27*, 309–321, 797–804.
9. Gorenflo, R.; Luchko, Y.; Mainardi, F. Analytical properties and applications of the Wright function. *Fract. Calc. Appl. Anal.* **1999**, *2*, 383–414.

10. Goloviznin, V.; Kiselev, V.; Korotkin, I.; Yurkov, Y. *Some Features of Computing Algorithms for the Equations Fractional Diffusion. Preprint № IBRAE-2002-01*; Nuclear Safety Institute RAS: Moscow, Russia, 2002; p. 57.
11. Bondarenko, A.N.; Ivaschenko, D.S. Numerical methods for solving boundary problems of anomalous diffusion theory. *Sib. Electron. Math. Rep.* **2008**, *5*, 581–594.
12. Mainardi, F. *Waves and Stability in Continuous Media*; Rionero, S., Ruggeri, T., Eds.; World Scientific: Singapore, 1994.
13. Wyss, W.J. The fractional diffusion equation. *J. Math. Phys.* **1986**, *27*, 2782. [CrossRef]
14. Schneider, W.R.; Wyss, W.J. Fractional diffusion and wave equations. *J. Math. Phys.* **1989**, *30*, 134. [CrossRef]
15. Ilic, M.; Liu, F.; Turner, I.; Anh, V. Numerical approximation of a fractional-inspace diffusion equation, I. *Fract. Calc. Appl. Anal.* **2005**, *8*, 323–341.
16. Oldham, K.B.; Spanier, J. *The Fractional Calculus*; Academic Press: New York, NY, USA, 1974.
17. Metzler, R.; Göckle, W.G.; Nonnenmacher, T.F. Nonnenmacher. Fractional model equation for anomalous diffusion. *Phys. A* **1994**, *211*, 13–24. [CrossRef]
18. Zelenyj, L.M.; Milovanov, A.M. Fraktal'naya topologiya i strannaya kinetika. *Uspekhi Fiz. Nauk* **2004**, *174*, 809–852.
19. Uchajkin, V.V. Avtomodel'naya anomal'naya diffuziya i ustojchivye zakony. *Uspekhi Fiz. Nauk* **2003**, *173*, 847–874.
20. Nahushev, A.M. *Drobnoe ischislenie I Ego Primenenie*; Fizmatlit: Moscow, Russia, 2003; p. 272.
21. Samko, S.G.; Kilbas, A.A.; Marichev, O.I. *Integraly I Proizvodnye Drobnogo Poryadka I Nekotorye Ih Prilozheniya*; Nauka i Tekhnika: Minsk, Belarus, 1987; p. 688.
22. CHukbar, K.V. Stohasticheskij perenos i drobnye proizvodnye. *ZHurnal Eksp. Teor. Fiz.* **1995**, *108*, 1875–1884.
23. Kobelev, V.L.; Romanov, E.N.; Kobelev, Y.A.L. Nelinejnaya relaksaciya i diffuziya v fraktal'nom prostranstve. *DAN* **1998**, *361*, 755–758.
24. Kochubej, A.N. Diffuziya drobnogo poryadka. *Differ. Uravn.* **1990**, *26*, 660–672.
25. Näsholm, S.P.; Holm, S. On a fractional Zener elastic wave equation. *Fract. Calc. Appl. Anal.* **2013**, *16*, 26–50. [CrossRef]
26. Holm, S.; Näsholm, S.P. A causal and fractional all-frequency wave equation for lossy media. *J. Acoust. Soc. Am.* **2011**, *130*, 2195–2202. [CrossRef]
27. Näsholm, S.P.; Holm, S. Linking multiple relaxation, power-law attenuation, and fractional wave equations. *J. Acoust. Soc. Am.* **2011**, *130*, 3038–3045. [CrossRef]
28. Pandey, V.; Holm, S. Connecting the grain-shearing mechanism of wave propagation in marine sediments to fractional order wave equations. *J. Acoust. Soc. Am.* **2016**, *140*, 4225–4236. [CrossRef]
29. Demidova, L.A.; Zhukov, D.O.; Andrianova, E.G.; Sigov, A.S. Modeling Sociodynamic Processes Based on the Use of the Differential Diffusion Equation with Fractional Derivatives. *Information* **2023**, *14*, 121. [CrossRef]
30. Zhukov, D.; Khvatova, T.; Zaltsman, A. Stochastic Dynamics of Influence Expansion in Social Networks and Managing Users' Transitions from One State to Another. In Proceedings of the 11th European Conference on Information Systems Management, ECISM 2017, The University of Genoa, Genoa, Italy, 14–15 September 2017; pp. 322–329, ISBN 978-191121852-4.
31. Sigov, A.S.; Zhukov, D.O.; Khvatova, T.Y.; Andrianova, E.G. A Model of Forecasting of Information Events on the Basis of the Solution of a Boundary Value Problem for Systems with Memory and Self-Organization. *J. Commun. Technol. Electron.* **2018**, *18*, 106–117. [CrossRef]
32. Zhukov, D.; Khvatova, T.; Istratov, L. A stochastic dynamics model for shaping stock indexes using self-organization processes, memory and oscillations. In Proceedings of the European Conference on the Impact of Artificial Intelligence and Robotics, ECIAIR 2019, Oxford, UK, 31 October–1 November 2019; pp. 390–401, ISBN 978-1-912764-44-0, 978-1-912764-45-7.
33. Zhukov, D.; Khvatova, T.; Istratov, L. Analysis of non-stationary time series based on modelling stochastic dynamics considering self-organization, memory and oscillations. In Proceedings of the ITISE 2019 International Conference on Time Series and Forecasting, Granada, Spain, 25–27 September 2019; Volume 1, pp. 244–254, ISBN 978-84-17970-78-.
34. Hurst, H.E. Long-term storage capacity of reservoirs. *Trans. Am. Soc. Civ. Eng.* **1951**, *116*, 770. [CrossRef]
35. Mandelbrot, B. *The Fractal Geometry of Nature*; W. H. Freeman: San Francisco, CA, USA, 1982.
36. Nassirtoussi, A.K.; Wah, T.Y.; Ling, D.N.C. A novel FOREX prediction methodology based on fundamental data. *Afr. J. Bus. Manag.* **2011**, *5*, 8322–8330.
37. Anastasakis, L.; Mort, N. Exchange rate forecasting using a combined parametric and nonparametric self–organising modelling approach. *Expert Syst. Appl.* **2009**, *36*, 12001–12011. [CrossRef]
38. Vanstone, B.; Finnie, G. Enhancing stockmarket trading performance with ANNs. *Expert Syst. Appl.* **2010**, *37*, 6602–6610. [CrossRef]
39. Vanstone, B.; Finnie, G. An empirical methodology for developing stockmarket trading systems using artificial neural networks. *Expert Syst. Appl.* **2009**, *36*, 6668–6680. [CrossRef]
40. Demidova, L.A.; Gorchakov, A.V. Application of bioinspired global optimization algorithms to the improvement of the prediction accuracy of compact extreme learning machines. *Russ. Technol. J.* **2022**, *10*, 59–74. [CrossRef]
41. Sermpinis, G.; Laws, J.; Karathanasopoulos, A.; Dunis, C.L. Forecasting and trading the EUR/USD exchange rate with gene expression and psi sigma neural networks. *Expert Syst. Appl.* **2012**, *39*, 8865–8877. [CrossRef]
42. Huang, S.-C.; Chuang, P.-J.; Wu, C.-F.; Lai, H.-J. Chaos-based support vector regressions for exchange rate forecasting. *Expert Syst. Appl.* **2010**, *37*, 8590–8598. [CrossRef]

43. Premanode, B.; Toumazou, C. Improving prediction of exchange rates using differential EMD. *Expert Syst. Appl.* **2013**, *40*, 377–384. [CrossRef]
44. Mabu, S.; Hirasawa, K.; Obayashi, M.; Kuremoto, T. Enhanced decision making mechanism of rule-based genetic network programming for creating stock trading signals. *Expert Syst. Appl.* **2013**, *40*, 6311–6320. [CrossRef]
45. Bahrepour, M.; Akbarzadeh, T.M.-R.; Yaghoobi, M.; Naghibi, S.M.-B. An adaptive ordered fuzzy time series with application to FOREX. *Expert Syst. Appl.* **2011**, *38*, 475–485. [CrossRef]
46. Aleroev, T.S.; Hasambiev, M.V.; Isaeva, L.M. Ob odnoj kraevoj zadache dlya drobnogo differencial'nogo uravneniya advekcii-diffuzii, "Trudy MAI". *Vypusk* **2014**, *3*, 1–14.
47. Hasambiev, M.V.; Aleroev, T.S. Kraevaya zadacha dlya odnomernogo drobnogo differencial'nogo uravneniya advek-cii-diffuzii. *Vestnik MGSU* **2014**, *6*, 71–76.
48. Aleroev, T.S.; Kirane, M.; Malik, S.A. Determination of a source term for a time fraction diffusion equation with an integral type over-determining condition. *Electron. J. Differ. Equ.* **2013**, *2013*, 1–16.
49. Aleroev, T.S.; Kirane, M.; Tang, Y.-F. Boundary-value problems for differential equations of fractional order. *J. Math. Sci.* **2013**, *194*, 499–512. [CrossRef]
50. Aleroev, T.S.; Aleroeva, H.T. A problem on the zeros of the Mittag-Leffler function and the spectrum of a fractional-order differential operator. *Electron. J. Qual. Theory Differ. Equ.* **2009**, *25*, 18. [CrossRef]
51. Sedletskij, A.M. Asimptoticheskie formuly dlya nulej funkcii tipa Mittag-Lefflera. *Anal. Math.* **1994**, *20*, 117–132. [CrossRef]
52. Dzhrbashyan, M.M.; Nersesyan, A.B. O postroenii nekotoryh special'nyh biortogonal'nyh system. *Izv. Akad. Nauk Arm. SSZ Matem.* **1959**, *12*, 17–42.
53. Dzhrbashyan, M.M. Interpolyacionnye i spektral'nye razlozheniya, associirovannye s differencial'nymi ope-ratorami drobnogo poryadka. *Izv. Akad. Nauk Arm. SSZ Matem.* **1984**, *19*, 81–181.
54. Wiman, A. Uber die Nullstellen der Funktionen. *Acta Math.* **1905**, *29*, 217–234. [CrossRef]
55. Pskhu, A.V. O veshchestvennyh nulyah funkcii tipa Mittag-Lefflera. *Mat. Zametki* **2005**, *77*, 592–599.
56. Medina, E.; Hwa, T.; Kardar, M.; Zhang, Y.-C. Burgers equation with correlated noise: Renormalization-group analysis and applications to directed polymers and interface growth. *Phys. Rev. A* **1989**, *39*, 3053–3075. [CrossRef]
57. Mukherji, S.; Bhattacharjee, S.M. Nonlocality in Kinetic Roughening. *Phys. Rev. Lett.* **1997**, *79*, 2502–2505. [CrossRef]
58. Chattopadhyay, A.K. Nonlocal Kardar-Parisi-Zhang equation with spatially correlated noise. *Phys. Rev. E* **1999**, *60*, 293–296. [CrossRef]

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

mathematics

Article

# Adaptation of the Scaling Factor Based on the Success Rate in Differential Evolution

**Vladimir Stanovov * and Eugene Semenkin**

Institute of Informatics and Telecommunication, Reshetnev Siberian State University of Science and Technology, 660037 Krasnoyarsk, Russia; eugenesemenkin@yandex.ru
* Correspondence: vladimirstanovov@yandex.ru

**Abstract:** Differential evolution is a popular heuristic black-box numerical optimization algorithm which is often used due to its simplicity and efficiency. Parameter adaptation is one of the main directions of study regarding the differential evolution algorithm. The main reason for this is that differential evolution is highly sensitive to the scaling factor and crossover rate parameters. In this study, a novel adaptation technique is proposed which uses the success rate to replace the popular success history-based adaptation for scaling factor tuning. In particular, the scaling factor is sampled with a Cauchy distribution, whose location parameter is set as an nth order root of the current success rate, i.e., the ratio of improved solutions to the current population size. The proposed technique is universal and can be applied to any differential evolution variant. Here it is tested with several state-of-the-art variants of differential evolution, and on two benchmark sets, CEC 2017 and CEC 2022. The performed experiments, which include modifications of algorithms developed by other authors, show that in many cases using the success rate to determine the scaling factor can be beneficial, especially with relatively small computational resource.

**Keywords:** differential evolution; parameter adaptation; numerical optimization

**MSC:** 68W50; 68T20; 65K10

**Citation:** Stanovov, V.; Semenkin, E. Adaptation of the Scaling Factor Based on the Success Rate in Differential Evolution. *Mathematics* **2024**, *12*, 516. https://doi.org/10.3390/math12040516

Academic Editors: Jonathan Blackledge and Liliya Demidova

Received: 7 December 2023
Revised: 16 January 2024
Accepted: 29 January 2024
Published: 7 February 2024

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

## 1. Introduction

Over the last several years, the differential evolution (DE) [1] algorithm has become one of the most commonly used numerical optimization techniques within the evolutionary computation (EC) community. DE is used in unconstrained, constrained, multi-objective, dynamic, multimodal and other cases [2], when the problem being solved has numerical parameters. Hence, developing more advanced and efficient DE variants is an important research direction for many evolutionary algorithms.

The main problem of DE stems from its main advantage: only three main parameters, scaling factor $F$, crossover rate $Cr$ and population size $N$, control most of the optimization process, which makes DE highly sensitive to settings. This problem has been significantly mitigated by parameter adaptation techniques proposed in jDE [3], JADE [4], SHADE [5] and some other schemes [6]. However, there still seems to be room for further improvement. The existing adaptation techniques rely on subsequent immediate improvements, which may constitute a greedy approach. In particular, the popular success history-based adaptation tunes the scaling factor $F$ and crossover rate $Cr$ values based on the most successful values in the last generations, but this does not guarantee that these values will result in good coverage of the search space and good convergence in the long term. This problem was considered in [7], where the effects of bias in parameter adaptation were studied. Hence, some other sources of information could be used to determine parameter values, for example, the number of improved solutions at every generation.

In this study, the scaling factor $F$ adaptation technique is proposed, which is based on the success rate (SR) value, i.e., the ratio of the number of successful new solutions divided by the total number of individuals. This value is not used explicitly in any of the DE variants to the best of our knowledge, although it is part of the averaging in the JADE and SHADE (Success History Adaptive Differential Evolution) algorithms. The idea of using the success rate came from analyzing the results of the genetic programming (GP) algorithm applied to design novel parameter adaptation techniques [8]. In particular, it was observed that GP heavily relies on the success rate in many of its solutions. Here, a refined variant of this idea is presented, more specifically, a *c*th order root of the success rate is utilized. To evaluate the efficiency of the new approach for $F$ sampling, it is tested on several algorithms, including L-SHADE-RSP [9], NL-SHADE-RSP [10], NL-SHADE-LBC [11] and L-NTADE [12]. The experiments are performed on two benchmark sets, namely the Congress on Evolutionary Computation (CEC) competitions from 2017 [13] and 2022 [14].

The main features of this study can be outlined as follows:

1. The success rate adaptation of the scaling factor improves the performance of most DE variants and requires the same settings independent of the algorithm;
2. The proposed adaptation scheme shows small dependence on the computational resource or problem dimension;
3. Compared to success history adaptation, success rate adaptation performs better with relatively small computational resource.

The Section 2 contains a short overview of parameter adaptation in differential evolution, the Section 3 describes the proposed approach, the Section 4 contains the experimental setup and results; after that, in Section 5, a discussion of the results is provided, and Section 6 concludes the paper.

## 2. Related Work

### *2.1. Differential Evolution*

The main focus of our study is on differential evolution, so we only consider its variants and modifications. The reason for this is that, as latest competitions (such as CEC 2017, CEC 2021, CEC 2022) show, other techniques, such as genetic algorithms, particle swarm optimization, and other biology inspired algorithms are not competitive in a single-objective black-box setup. The only exclusion may be the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithm and its hybrids with DE.

The main idea of the differential evolution algorithm proposed in [15] is to use difference vectors between individuals to produce new solutions during mutation. The algorithm starts with the initialization of $N$ individuals, $x_i = (x_{i,1}, x_{i,2}, \dots, x_{i,D})$, $i = 1, \dots, N$, within the bounds:

$$S = \{x_i \in R^D | x_i = (x_{i,1}, x_{i,2}, \dots, x_{i,D}) : x_{i,j} \in [x_{lb,j}, x_{ub,j}]\}, \tag{1}$$

where $D$ is the number of variables. Most of the studies use the uniform random generation of individuals:

$$x_{i,j} = x_{lb,j} + rand \times (x_{ub,j} - x_{lb,j}). \tag{2}$$

After initialization and target function evaluation, the mutation step begins. Many mutation strategies have been proposed for DE, but the two most popular ones are rand/1 and current-to-pbest/1:

$$v_{i,j} = x_{r1,j} + F \times (x_{r2,j} - x_{r3,j}), \tag{3}$$

$$v_{i,j} = x_{i,j} + F \times (x_{pbest,j} - x_{i,j}) + F \times (x_{r1,j} - x_{r2,j}), \tag{4}$$

where $x_i$ is called the target vector, $v_i$ is the mutant or donor vector, $F$ is the scaling factor, *pbest* is the index of one of the $p$% best individuals, and indexes $r1$, $r2$, $r3$ and *pbest* are chosen randomly so that they are different from each other and the current individual with index $i$. The mutation is performed for all solutions in the population, $i = 1, 2, \dots, N$.

After a set of donor vectors is generated, the crossover step is performed. The most widely used method is binomial crossover, described as follows:

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } rand(0,1) < Cr \text{ or } j = jrand \\ x_{i,j}, & \text{otherwise} \end{cases}, \tag{5}$$

where $u_i$ is called the trial vector, $Cr$ is the crossover rate parameter, and $jrand \in [1, D]$, generated randomly. In other words, binomial crossover combines the information in target vector $x_i$ and mutant vector $v_i$ to produce the trial vector, and $jrand$ is required to make sure that the trial vector is different from the target one.

The mutation may generate vectors which are outside the $[x_{lb,j}, x_{ub,j}]$ range, so a bound constraint handling technique is required in DE. One of the widely used ones is the midpoint-target method [16], which works as follows:

$$u_{i,j} = \begin{cases} \frac{x_{lb,j}+x_{i,j}}{2}, & \text{if } v_{i,j} < x_{lb,j} \\ \frac{x_{ub,j}+x_{i,j}}{2}, & \text{if } v_{i,j} > x_{ub,j} \end{cases}. \tag{6}$$

If some of the components of the trial vector overshoot the boundaries, then the coordinate of the parent (target vector) is used to move towards the boundary without reaching it. We note that this step can be applied after mutation or after crossover.

The last step in DE is the selection, which here plays the role of a replacement mechanism. If the trial vector, $u_i$, is better in terms of the target function value compared to target vector $x_i$, then replacement occurs:

$$x_i = \begin{cases} u_i, & \text{if } f(u_i) \leq f(x_i) \\ x_i, & \text{if } f(u_i) > f(x_i) \end{cases}. \tag{7}$$

Although this selection mechanism is known to be simple and efficient, there are some attempts to modify it, for example, by using information about neighbuorhood [17] or replacing other individuals [12].

### 2.2. Parameter Adaptation in Differential Evolution

Most of the modern studies which are focused on DE or apply DE to some problem use one of the previously proposed parameter adaptation techniques. Here, it would be impractical to cover all of the existing parameter adaptation techniques, so interested readers are advised to refer to surveys such as [18,19] as well as some recent papers considering the problem [7].

Some of the earliest successful experiments on parameter adaptation were presented in the jDE algorithm [3], where, on each iteration $t$ before the mutation and crossover steps of the search loop, new control parameters are generated in the following way:

$$F_{i,t+1} = \begin{cases} random(F_l, F_u), & \text{if } random(0,1) < \tau_1 \\ F_{i,t}, & \text{otherwise} \end{cases}, \tag{8}$$

$$CR_{i,t+1} = \begin{cases} random(0,1), & \text{if } random(0,1) < \tau_2 \\ CR_{i,t}, & \text{otherwise} \end{cases}. \tag{9}$$

In the equations above, values $F_l = 0.1$ and $F_u = 0.9$ represent the lower and upper boundaries for $F$, and the parameters controlling the frequency of change $\tau_1$ and $\tau_2$ are set to 0.1. If, during selection, improvement occurs, then the new values for $F$ and $Cr$ are saved. This technique was successfully applied in several highly efficient versions of jDE, including the jDE100 [20] and j2020 [21] algorithms, which have shown competitive results in CEC 2019 and 2020 competitions, respectively.

Another important algorithm is JADE, proposed in [4]. Despite the fact that JADE introduced one of the most used mutations, current-to-pbest/1, it also proposed sampling of $F$ and $Cr$ in a way similar to the SaDE algorithm [22]. However, instead of using fixed means, in JADE, $F$ and $Cr$ are sampled with Cauchy and normal distributions around $\mu_F$ and $\mu_{Cr}$:

$$\begin{cases} F = randc(\mu_F, 0.1) \\ Cr = randn(\mu_{Cr}, 0.1) \end{cases}, \tag{10}$$

where $randc(\mu, s)$ and $randn(\mu, s)$ are random numbers sampled from Cauchy and normal distribution with location parameter $\mu$ and scale parameter $s = 0.1$. The update of $\mu$ values is performed as follows:

$$\begin{cases} \mu_F = (1-c) \cdot \mu_F + c \cdot mean_L(S_F) \\ \mu_{Cr} = (1-c) \cdot \mu_{Cr} + c \cdot mean_A(S_{Cr}) \end{cases}, \tag{11}$$

where $mean_A()$ is the arithmetic mean, $c$ is the update parameter, $S_F$ and $S_{Cr}$ contain the successful values of $F$ and $Cr$, i.e., values which produce individuals better than parents, and $mean_L()$ is the Lehmer mean:

$$mean_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F}. \tag{12}$$

The success of JADE's parameter adaptation has led to the development of the even more efficient Success History Adaptation (SHA), proposed in the SHADE algorithm [5]. Unlike JADE, in SHADE, there are $H$ memory cells, each containing a pair $(M_{F,h}, M_{Cr,h})$, i.e., mean values to be used for sampling $F$ and $Cr$. These values are used in a similar way as in Equation (10):

$$\begin{cases} F = randc(M_{F,h}, 0.1) \\ Cr = randn(M_{Cr,h}, 0.1) \end{cases}. \tag{13}$$

The $h$ index is randomly sampled from $[1, H]$ before each mutation and crossover. At the end of the generation, the new means are calculated using a weighted Lehmer mean [23], where weights are set based on the improvements $\Delta f = |f(u_j) - f(x_j)|$, stored in $S_{\Delta f}$:

$$mean_{wL,F} = \frac{\sum_{j=1}^{|S_F|} w_j S_{F,j}^2}{\sum_{j=1}^{|S_F|} w_j S_{F,j}}, \; mean_{wL,Cr} = \frac{\sum_{j=1}^{|S_{Cr}|} w_j S_{Cr,j}^2}{\sum_{j=1}^{|S_{Cr}|} w_j S_{Cr,j}}, \tag{14}$$

where $w_j = \frac{S_{\Delta f\, j}}{\sum_{k=1}^{|S|} S_{\Delta f\, k}}$. Next, one of the memory cells with index $k$, iterated every generation, is updated:

$$\begin{cases} M_{F,k}^{t+1} = 0.5(M_{F,k}^t + mean_{wL,F}) \\ M_{Cr,k}^{t+1} = 0.5(M_{Cr,k}^t + mean_{wL,Cr}) \end{cases}, \tag{15}$$

where $t$ is the current generation number. If $k > H$, then it is set to $k = 1$.

As for the population size, in L-SHADE [24], the Linear Population Size Reduction (LPSR) was proposed, which is a widely used technique nowadays. The LPSR works by determining new population size at every generation as follows:

$$N_{g+1} = round\left(\frac{N_{min} - N_{max}}{NFE_{max}} NFE\right) + N_{max}, \tag{16}$$

where $NFE$ is the current number of target function evaluations, $NFE_{max}$ is the total available computational resource, $N_{max}$ and $N_{min}$ are the initial and final number of individuals, $g$ is the generation number. The main idea of LPSR consists in spreading across the search space at the beginning and concentrating at the end of the search. LPSR allows achievement of significant improvements in performance if the computational resource limit is known. However, some other studies modify it and use non-linear reduction; for example, [10].

In [9], it was shown that adding tournament or rank-based selection strategies to sample the indexes of individuals for further mutation may be beneficial. The exponential rank-based selection was implemented by selecting an individual depending on its fitness in a sorted array, with the ranks assigned as follows:

$$rank_i = e^{\frac{-kp \cdot i}{N}}, \tag{17}$$

where $kp$ is the parameter controlling the pressure and $i$ is the individual number. Larger ranks are assigned to better individuals, and discrete distribution is used for selection.

It is hard to underestimate the importance of the L-SHADE (SHADE with LPSR) algorithm: the success history adaptation was further developed and improved in many studies [6]. To provide some examples, the jSO algorithm [25] proposed specific rules for parameter adaptation, limiting the $F$ and $Cr$ values depending on computational resource, and used in L-SHADE-RSP [9] with rank-based selection, proposed in [26], DB-LSHADE proposed distance-based adaptation, where weights are based on the Euclidean distance instead of fitness improvement [27,28]. In [7], the effect of modified Lehmer means was considered. Nevertheless, most of the adaptive DE variants tend to apply small modifications to the SHA general scheme without changing it dramatically. The main problem of SHA is that it follows immediate improvements in fitness and tunes $F$ and $Cr$ to them, which is a greedy approach. The biased adaptation introduced with the Lehmer mean in the JADE algorithm was, in fact, proposed to eliminate this effect by sampling higher $F$ values than they should be. A more detailed consideration of the effects of bias on parameter adaptation was performed in [7].

Some other modifications of DE include population regeneration in case of premature convergence [29], modifications for binary search space [30], Gaussian–Cauchy mutation [31] and using an ensemble of mutation and crossover operators [32]. Some works focus on proposing new mutation strategies, such as the triple competitive approach recently proposed in [33]. In [34], the Unbounded DE (UDE) was proposed, where all the individuals generated throughout the whole search space are saved and used in the search. Such an approach shows some interesting effects of using old solutions in the search process.

## 3. Proposed Approach

Developing new algorithmic schemes, including parameter adaptation techniques, could be a tedious process requiring prolonged experimentation and brand new ideas. In a recent study [8], an attempt to use genetic programming solving symbolic regression as a knowledge extraction technique was performed, and as a result, it was discovered that the success rate can be an important source of information for the adaptation of scaling factor $F$. Based on this idea, as well as some additional experiments, success rate-based adaptation is proposed.

Success rate is calculated as the ratio of the number of successful solutions $NS$ to the current population size $N$:

$$SR = \frac{NS}{N}. \tag{18}$$

In other words, $NS$ is equal to the amount of elements in $S_F$ or $S_{Cr}$. Success rate shows, in general, how well the algorithm performs: if the search is efficient, then $NS$ is quite high, and if the population is stuck in a local optimum or optima, then it is low. Success rate is used to calculate the $MF$ value, which acts as a mean for sampling $F$:

$$MF = SR^{1/c}, \tag{19}$$

where $c$ is the parameter value. Some of the solutions found by GP used the square root of the success rate. However, further experiments have shown that using the cube root or even larger $c$ values may produce better results. The $MF$ value is then used as follows:

$$F = randc(MF, 0.1). \tag{20}$$

Joining together all the above equations, the scaling factor for every mutation can be calculated as follows:

$$F = randc\left(\left(\frac{NS}{N}\right)^{1/c}, 0.1\right). \tag{21}$$

As in many other methods, if $F > 1$, it is set to 1, and if $F < 0$, it is sampled again until positive. Such a technique is very simple compared to SHADE, or even JADE and jDE, as it does not require additional values to be stored or the average to be calculated. Moreover, it can be applied to any DE algorithm without significant effort.

Figure 1 shows the dependence of $MF$ on $SR$ with different $c$ parameter values.

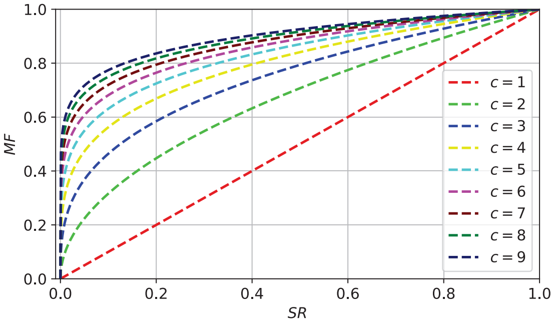


**Figure 1.** $MF$ values for different $c$ parameter values.

As shown in Figure 1, small success rate values (<0.05) lead to $MF$ values between 0 and 0.5, while larger success rates result in $MF$ being larger than 0.5.

The main advantage of the proposed technique is its simplicity: to apply it, a single Equation (21) is required, whereas SHA needs to store memory cells, successful parameter values, calculate weights, Lehmer means and update memory cells (Equations (13)–(15)). This creates additional computational effort. To illustrate this, in Figure 2, the two techniques are compared. In the next section, when the modifications of DE variants are considered, the steps of success history adaptation are replaced by a single step of success rate adaptation.

$$F = randc\left(\left(\frac{NC}{N}\right)^{1/c}, 0.1\right)$$

$$\begin{gathered}
F = randc(M_{F,k}, 0.1) \\
mean_{wL} = \frac{\sum_{j=1}^{|S|} w_j \cdot S_j^2}{\sum_{j=1}^{|S|} w_j \cdot S_j} \\
w_j = \frac{\Delta f_j}{\sum_{k=1}^{|S|} \Delta f_k} \\
\Delta f_j = |f(u) - f(x_j)| \\
M_{F,k}^{t+1} = 0.5(M_{F,k}^{t} + mean_{(wL,F)}) \\
H\ memory\ cells
\end{gathered}$$

**Figure 2.** Comparison of steps required for success rate-based adaptation and success history-based adaptation.

The proposed technique was implemented for L-SHADE-RSP, NL-SHADE-RSP, NL-SHADE-LBC, L-NTADE and other algorithms, and the results are considered in the next section.

## 4. Experimental Setup and Results

### *4.1. Benchmark Functions and Parameters*

The main purpose of the experiments in this study was to determine the efficiency of success rate-based adaptation in different scenarios. For this, two benchmark sets were chosen, namely the CEC 2017 [13] and 2022 [14] Single Objective Bound Constrained Numerical Optimization problems. These benchmarks have different numbers of functions, 30 and 12, respectively, different dimensions and computational resource. For CEC 2017, the test functions were defined for $D = 10$, 30, 50 and 100, and for CEC 2022 $D = 10$ and 20. The number of available function evaluations was set to $NFE_{max}$ = 10,000$D$ for CEC 2017, and for CEC 2022 it was set to $NFE_{max} = 2 \times 10^5$ for 10$D$ and $NFE_{max} = 1 \times 10^6$ for 20$D$. All the experiments with algorithms and their modifications were performed according to competition rules.

All the tested algorithms were implemented in C++, and ran on eight AMD Ryzen 3700 PRO processors with eight cores each under Ubuntu Linux 20.04; the experiments were paralleled using OpenMPI 4.0.3, and post-processing was performed using Python 3.8.5.

To compare different results, two main techniques were applied. The first was the Mann–Whitney rank sum statistical test with normal approximation and tie-breaking, with significance level set to $p = 0.01$. In addition to the result of the test (win/tie/loss), the standard $Z$ score values were calculated and summed over all test functions. We note that $Z = \pm 2.58$ corresponds to $p = 0.01$. The second is the Friedman ranking procedure, applied in the Friedman statistical test. Here, it was used to compare a set of algorithms or their variants, and the ranks were assigned to the results obtained on every run and function independently, and then summed together. We note that unlike CEC 2017, in CEC 2022, the ranking of results included not only the best achieved values, but also the total computational resource spent, and this ranking was used in both the Mann–Whitney test and Friedman ranking.

### *4.2. Numerical Results*

In order to consider the proposed modification from different points, in the following subsections, several aspects were considered, including the effects of SR-based adaptation on the performance of different existing algorithms when success history-based adaptation is replaced; the influence of the available computational resource and mutation strategies; comparison with the best methods on two sets of benchmark functions. Also, a visualization of the parameter adaptation process was considered for a better understanding of the effects on the differential evolution algorithm.

#### 4.2.1. Modification of Existing Algorithms

As was mentioned above, the success rate-based adaptation for $F$ replaced the success history adaptation in four tested algorithms. As for the crossover rate $Cr$ adaptation, it was unchanged. In the case of the L-SHADE-RSP algorithm, which uses a specific mutation strategy with $F$ and $F2$, when using the success rate, $F2$ was set equal to $F$, and specific rules based on the current resource from jSO algorithm were removed. The parameters for all the algorithms were set according to those used in corresponding papers.

The majority of modern DE algorithms rely on the success history adaptation method introduced in SHADE, so the purpose of the study was to compare the adaptation methods, and not specific algorithms. For this purpose, in Tables 1–10, we modified L-SHADE-RSP, NL-SHADE-RSP, NL-SHADE-LBC, L-NTADE, jSO, LSHADE-SPACMA, APGSK-IMODE, MLS-LSHADE and MadDE with success rate-based adaptation and compared the results.

Tables 1 and 2 present the comparison of the original L-SHADE-RSP with the modified L-SHADE-RSP (SR) with different $c$ values. Each cell in the tables contains the number of wins/ties/losses and the total standard score in brackets. Larger standard score means that the modified algorithm is better.

**Table 1.** L-SHADE-RSP (SR) vs. L-SHADE-RSP, CEC 2017.

| $c$ | **10$D$** | **30$D$** | **50$D$** | **100$D$** |
|---|---|---|---|---|
| 1 | 3/18/9 (−45.16) | 0/9/21 (−157.01) | 0/7/23 (−163.79) | 2/5/23 (−183.13) |
| 2 | 3/23/4 (−12.68) | 4/11/15 (−74.72) | 4/8/18 (−87.52) | 4/9/17 (−91.82) |
| 3 | 5/23/2 (4.32) | 5/16/9 (−28.41) | 5/15/10 (−25.87) | 8/12/10 (−16.28) |
| 4 | 4/26/0 (19.57) | 2/23/5 (−18.11) | 5/20/5 (2.37) | 11/11/8 (11.60) |
| 5 | 5/23/2 (14.58) | 3/20/7 (−31.34) | 1/25/4 (−11.09) | 12/10/8 (16.57) |
| 6 | 6/19/5 (8.65) | 2/18/10 (−41.83) | 3/20/7 (−33.75) | 12/11/7 (4.25) |

**Table 2.** L-SHADE-RSP (SR) vs. L-SHADE-RSP, CEC 2022.

| $c$ | **10$D$** | **20$D$** |
|---|---|---|
| 1 | 7/3/2 (29.61) | 3/3/6 (−11.30) |
| 2 | 7/3/2 (30.91) | 4/4/4 (4.29) |
| 3 | 6/3/3 (21.43) | 4/6/2 (18.58) |
| 4 | 4/2/6 (−8.13) | 2/8/2 (−3.00) |
| 5 | 4/4/4 (−8.23) | 2/6/4 (−14.21) |
| 6 | 3/5/4 (−9.17) | 1/7/4 (−16.18) |

As can be seen from Tables 1 and 2, the $c$ parameter significantly influences performance, and if for CEC 2017 $c = 1$, i.e., $MF = SR$, is a bad choice; for CEC 2022 it is a reasonable setting for 10$D$. The setting which leads to good performance is $c = 4$ for CEC 2017. In this case, for 10$D$, 50$D$ and 100$D$, there are more improvements than losses, but for 30$D$ the success history adaptation works better. In the case of the CEC 2022 benchmark, the best choice is$c = 3$.

In Tables 3 and 4, the comparison of the basic NL-SHADE-RSP and the modified NL-SHADE-RSP (SR) is presented.

**Table 3.** NL-SHADE-RSP (SR) vs. NL-SHADE-RSP, CEC 2017.

| $c$ | **10$D$** | **30$D$** | **50$D$** | **100$D$** |
|---|---|---|---|---|
| 1 | 0/13/17 (−77.34) | 1/6/23 (−134.10) | 1/9/20 (−128.17) | 6/6/18 (−91.88) |
| 2 | 0/25/5 (−18.32) | 7/21/2 (17.65) | 14/13/3 (45.81) | 20/6/4 (89.04) |
| 3 | 2/27/1 (8.46) | 14/14/2 (64.24) | 15/13/2 (100.78) | 22/7/1 (134.60) |
| 4 | 7/22/1 (28.38) | 15/14/1 (81.37) | 17/11/2 (111.83) | 22/6/2 (141.35) |
| 5 | 8/21/1 (33.83) | 16/13/1 (73.35) | 18/10/2 (111.43) | 21/7/2 (139.24) |
| 6 | 6/24/0 (29.73) | 13/14/3 (66.20) | 17/11/2 (111.59) | 22/5/3 (135.73) |

**Table 4.** NL-SHADE-RSP (SR) vs. NL-SHADE-RSP, CEC 2022.

| $c$ | **10$D$** | **20$D$** |
|---|---|---|
| 1 | 4/4/4 (2.46) | 0/6/6 (−39.78) |
| 2 | 5/6/1 (14.26) | 1/8/3 (−8.21) |
| 3 | 6/2/4 (14.37) | 5/4/3 (5.05) |
| 4 | 6/2/4 (8.26) | 5/3/4 (4.18) |
| 5 | 5/3/4 (9.20) | 5/3/4 (6.02) |
| 6 | 6/2/4 (11.19) | 5/4/3 (8.10) |

The situation with NL-SHADE-RSP is quite different as this method was not tuned for CEC 2017 or CEC 2022. Nevertheless, small values of the $c$ parameter lead to poor performance on both benchmarks, and increasing $c$ up to 3 or 4 produces much better results. After $c = 4$, performance gains become smaller.

Tables 5 and 6 show the results of NL-SHADE-LBC and its modified version.

**Table 5.** NL-SHADE-LBC (SR) vs. NL-SHADE-LBC, CEC 2017.

| $c$ | 10$D$ | 30$D$ | 50$D$ | 100$D$ |
|---|---|---|---|---|
| 1 | 0/17/13 (−63.46) | 2/8/20 (−138.49) | 5/8/17 (−103.11) | 7/4/19 (−108.71) |
| 2 | 2/24/4 (−13.13) | 7/13/10 (−15.57) | 18/3/9 (62.92) | 16/8/6 (63.66) |
| 3 | 4/24/2 (21.91) | 14/12/4 (73.17) | 23/6/1 (161.08) | 23/5/2 (161.72) |
| 4 | 8/21/1 (40.55) | 16/13/1 (97.39) | 25/4/1 (185.04) | 25/5/0 (190.54) |
| 5 | 10/20/0 (53.79) | 16/13/1 (106.44) | 25/5/0 (186.03) | 27/3/0 (199.10) |
| 6 | 10/20/0 (56.07) | 17/12/1 (106.95) | 25/4/1 (176.66) | 26/4/0 (193.90) |

**Table 6.** NL-SHADE-LBC (SR) vs. NL-SHADE-LBC, CEC 2022.

| $c$ | 10$D$ | 20$D$ |
|---|---|---|
| 1 | 6/3/3 (26.91) | 3/3/6 (-19.44) |
| 2 | 2/3/7 (−23.34) | 1/4/7 (−37.09) |
| 3 | 3/2/7 (−25.30) | 2/5/5 (−20.46) |
| 4 | 2/3/7 (−26.51) | 3/4/5 (−17.18) |
| 5 | 2/3/7 (−28.37) | 3/4/5 (−20.26) |
| 6 | 2/3/7 (−28.53) | 3/4/5 (−18.72) |

As can be seen from Tables 5 and 6, the situation with NL-SHADE-LBC is quite different. As this method was specifically developed for the CEC 2022 benchmark, its parameter adaptation was not tuned for the CEC 2017 benchmark, and applying success rate-based adaptation significantly improves results, with up to 27 improvements out of 30 functions in 100$D$. However, for CEC 2022, the success rate adaptation was not able to deliver better performance, but still $c = 4$ or higher is a reasonable choice for both benchmarks. The reason why SR-based adaptation worked worse with NL-SHADE-LBC is that the parameter adaptation of this algorithm was specifically tuned for the CEC 2022 benchmark.

Tables 7 and 8 contain the results of the L-NTADE algorithm and its modification.

**Table 7.** L-NTADE (SR) vs. L-NTADE, CEC 2017.

| $c$ | 10$D$ | 30$D$ | 50$D$ | 100$D$ |
|---|---|---|---|---|
| 1 | 2/10/18 (−105.68) | 0/8/22 (−192.68) | 0/4/26 (−218.98) | 1/2/27 (−219.12) |
| 2 | 6/15/9 (−11.94) | 3/14/13 (−65.01) | 4/8/18 (−108.84) | 5/3/22 (−124.60) |
| 3 | 9/15/6 (22.08) | 12/15/3 (43.24) | 8/17/5 (19.67) | 8/9/13 (-6.39) |
| 4 | 11/14/5 (37.30) | 14/15/1 (75.39) | 15/11/4 (69.26) | 15/11/4 (70.63) |
| 5 | 10/16/4 (44.53) | 14/15/1 (73.95) | 17/10/3 (67.77) | 15/11/4 (65.51) |
| 6 | 11/17/2 (36.40) | 9/19/2 (48.88) | 13/11/6 (25.88) | 11/12/7 (24.52) |

**Table 8.** L-NTADE (SR) vs. L-NTADE, CEC 2022.

| $c$ | 10$D$ | 20$D$ |
|---|---|---|
| 1 | 5/3/4 (11.08) | 3/2/7 (−22.99) |
| 2 | 6/4/2 (25.53) | 3/3/6 (−6.99) |
| 3 | 6/4/2 (25.45) | 3/5/4 (1.07) |
| 4 | 5/5/2 (26.17) | 2/6/4 (−1.82) |
| 5 | 3/8/1 (13.29) | 2/6/4 (−10.64) |
| 6 | 2/6/4 (−10.49) | 2/7/3 (−12.62) |

Although L-NTADE has two populations and has a different algorithmic scheme, applying success rate-based adaptation improved the results in both benchmarks. As before, small $c$ values are inefficient. However, setting $c = 4$ for CEC 2017 or probably even larger values offers significant performance benefits.

In order to test the applicability of the proposed approach to other algorithms, several of them for which the source codes were available were modified and the modifications were compared to the original results. For the CEC 2017 benchmark, in the jSO (derived from L-SHADE) [25], the mutation step, which uses two factors, $F$ and $F2$, was modified to have only a single one, and the control rules for $F$ were deactivated. For LSHADE-SPACMA [35], the scaling factor sampling in the DE part was replaced by Equation (19). The comparison is shown in Table 9.

**Table 9.** Comparison of other modified approach vs. original versions, CEC 2017.

| Algorithm | 10$D$ | 30$D$ | 50$D$ | 100$D$ |
|---|---|---|---|---|
| jSO (SR, $c = 4$) vs.<br>jSO [25] | 1/21/8<br>(−37.08) | 8/16/6<br>(21.04) | 6/14/10<br>(−25.27) | 11/9/10<br>(−2.29) |
| LSHADE-SPACMA (SR, $c = 4$) vs.<br>LSHADE-SPACMA [35] | 3/22/5<br>(−22.61) | 1/24/5<br>(−41.67) | 0/22/8<br>(−64.32) | 0/23/7<br>(−42.11) |

As can be seen from Table 9, the modification of jSO with $c = 4$ may be beneficial in some cases, for example, in the 30D scenario, but in 10D the number of losses was much larger than the number of wins. As for LSHADE-SPACMA, using success rate-based adaptation leads to decreased performance. The main reason why SR-based adaptation failed to improve the performance in these cases could be that both jSO and LSHADE-SPACMA were specifically tuned for the CEC 2017 benchmark, and the SR-based adaptation is a more general approach, which was not tuned for these particular algorithms.

In Table 10, the same modification was applied to APGSK-IMODE (only in IMODE part) [36], MLS-LSHADE [37] and MadDE [38]. Due to a simpler benchmark in terms of computation time, the experiments were performed with different $c$ values.

**Table 10.** Comparison of other modified approaches vs. original versions, CEC 2022.

| Algorithm | 10$D$ | 20$D$ |
|---|---|---|
| APGSK-IMODE (SR, $c = 2$) vs.<br>APGSK-IMODE [36] | 4/5/3<br>(3.66) | 1/5/6<br>(−24.33) |
| APGSK-IMODE (SR, $c = 3$) vs.<br>APGSK-IMODE [36] | 4/4/4<br>(12.88) | 1/8/3<br>(−14.19) |
| APGSK-IMODE (SR, $c = 4$) vs.<br>APGSK-IMODE [36] | 3/4/5<br>(−8.81) | 1/8/3<br>(−12.24) |
| APGSK-IMODE (SR, $c = 5$) vs.<br>APGSK-IMODE [36] | 3/4/5<br>(−8.81) | 1/8/3<br>(−12.24) |
| MLS-LSHADE (SR, $c = 2$) vs.<br>MLS-LSHADE [37] | 4/3/5<br>(−2.05) | 0/4/8<br>(−49.53) |
| MLS-LSHADE (SR, $c = 3$) vs.<br>MLS-LSHADE [37] | 2/3/7<br>(−29.50) | 0/6/6<br>(−41.34) |
| MLS-LSHADE (SR, $c = 4$) vs.<br>MLS-LSHADE [37] | 3/1/8<br>(−32.78) | 0/5/7<br>(−41.91) |
| MLS-LSHADE (SR, $c = 5$) vs.<br>MLS-LSHADE [37] | 1/3/8<br>(−37.41) | 0/5/7<br>(−40.96) |

**Table 10.** *Cont.*

| Algorithm | 10$D$ | 20$D$ |
|---|---|---|
| MadDE (SR, $c = 2$) vs.<br>MadDE [38] | 5/4/3<br>(12.46) | 2/6/4<br>(−9.56) |
| MadDE (SR, $c = 3$) vs.<br>MadDE [38] | 4/5/3<br>(14.78) | 3/8/1<br>(13.48) |
| MadDE (SR, $c = 4$) vs.<br>MadDE [38] | 4/5/3<br>(13.10) | 3/8/1<br>(15.39) |
| MadDE (SR, $c = 5$) vs.<br>MadDE [38] | 3/6/3<br>(4.88) | 3/6/3<br>(0.68) |

The effect of success rate-based scaling factor adaptation is different for different algorithms. For example, the performance of APGSK-IMODE was improved only in the 10$D$ case, with $c = 2$ and $c = 3$, but for MLS-LSHADE the effect was mainly negative. As for the MadDE algorithm, it was mostly improved by the modification, especially for $c = 3$ and $c = 4$.

### 4.2.2. Effect of the Available Computational Resource

The main difference between the two considered benchmarks is the number of computations available for every function. The experiments shown above demonstrated that there is a possibility that amount of resource may influence the performance of success rate-based adaptation. To test this hypothesis, a set of experiments was performed for the L-NTADE algorithm on CEC 2022, where the number of function evaluations $NFE_{max}$ was decreased to 10%, 20% and so on up to 100%. Figure 2 shows the Friedman ranking of the results in the form of heatmaps.

In Figure 3, the ranking was performed independently in every column, as there is no sense in comparing algorithms with different available resource. The best ranks are shown in red. In the 10$D$ case, if the resource is relatively small, the success rate adaptation is significantly better than SHA. However, as the $NFE_{max}$ grows, the SHA may take the lead. In the 20$D$ case, however, the SR adaptation is always better, and the best settings are $c = 5$ or $c = 6$, although $c = 4$ is very similar. From these results, it can be concluded that, first of all, success rate-based adaptation is not highly influenced by available resource, but there is a difference in performance compared to SHA.

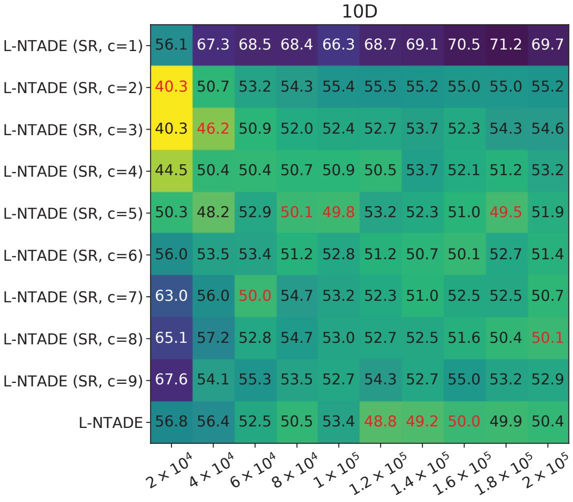


**Figure 3.** *Cont.*

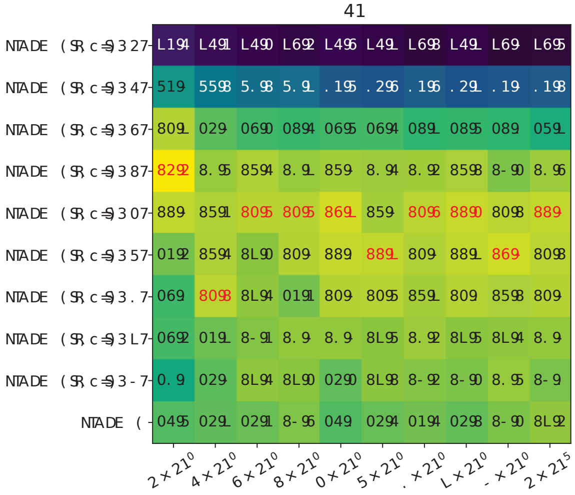


**Figure 3.** Comparison of L-NTADE with and without success rate adaptation with different available resource $NFE_{max}$, Friedman ranking.

### 4.2.3. Visualization of Parameter Adaptation Process

For a better understanding of what SR actually does during the search process, Figures 4 and 5 show the graphs of $F$ values set by the SHA and SR methods in the case of the L-SHADE-RSP algorithm.

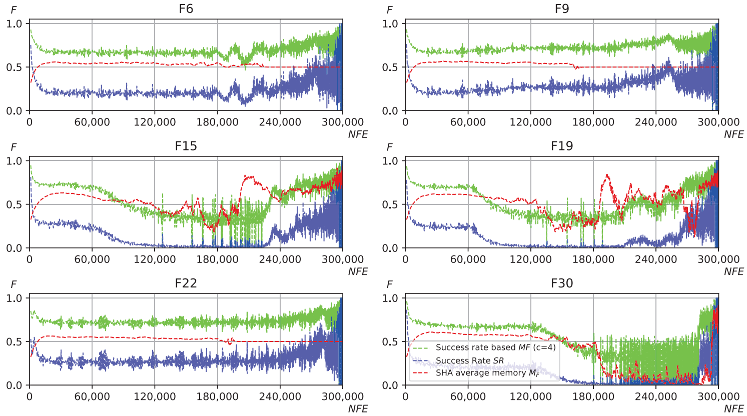


**Figure 4.** Graphs of parameter adaptation of L-SHADE-RSP with and without success rate adaptation, CEC 2017, 30$D$, selected functions.

The graphs in Figure 4 show that the behavior of the two considered adaptation methods is quite different, and sometimes even opposite. For example, at the beginning of the search, the success rate is high, and so is the $MF$ value, and they both decrease, while SHA makes larger $F$ values. If the success rate drops close to 0, this makes the $MF$ values oscillate between 0 and 0.5, while SHA slowly tunes values in the memory cells, as seen on functions F15 and F19. If the search process is going well, SR tends to maintain relatively high $F$ values, and switches to an oscillating mode if the success rate is low.

Figure 5 shows similar trends, but they are seen much better due to larger resource. For example, for functions F4, F7 and F8, the success history adaptation leads to memory cell values close to zero at some points in the middle of the search, but success rate adaptation tries every possible $F$ value from 0 to around 0.4 when the success rate drops low. Also, oscillation tends to become higher closer to the end of the search. This is simply the effect of smaller a population size.

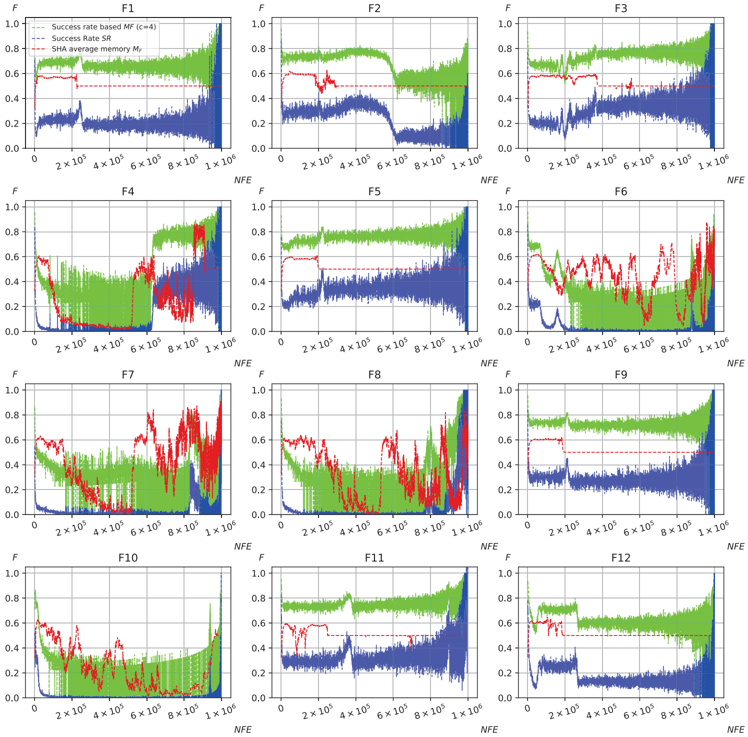


**Figure 5.** Graphs of parameter adaptation of L-SHADE-RSP with and without success rate adaptation, CEC 2022, 20$D$.

### 4.2.4. Success Rate-Based Adaptation With Different Mutation Strategies

One of the possible explanations of the high performance of success rate-based adaptation for $F$ could be that it works in combination with the current-to-pbest strategy. If the success rate is low, smaller $F$ values force more exploration, while $F$ closer to 1 leads to a move towards one of the $p\%$ best solutions. To test this effect of success rate-based adaptation, a set of experiments was performed on the standard L-SHADE algorithm with different mutation strategies, such as rand/1, rand/2, current-to-best/1, current-to-rand/1, best/1 and best/2. The $c$ parameter was also altered from 1 to 6, and the original L-SHADE was tested. Next, the Friedman ranking procedure was applied to compare the performance of SHA with SR. The ranks were assigned for each mutation strategy (column) independently. The experiments were repeated for both CEC 2017 and CEC 2022 benchmarks. The parameters of L-SHADE were the following: $N = 60 \times D^{\frac{2}{3}}$, $pbest = 0.2$, $H = 5$, initial $M_F = 0.5$, $M_{Cr} = 0.5$, archive was not used. Figures 6 and 7 contain the heatmaps of the comparison.

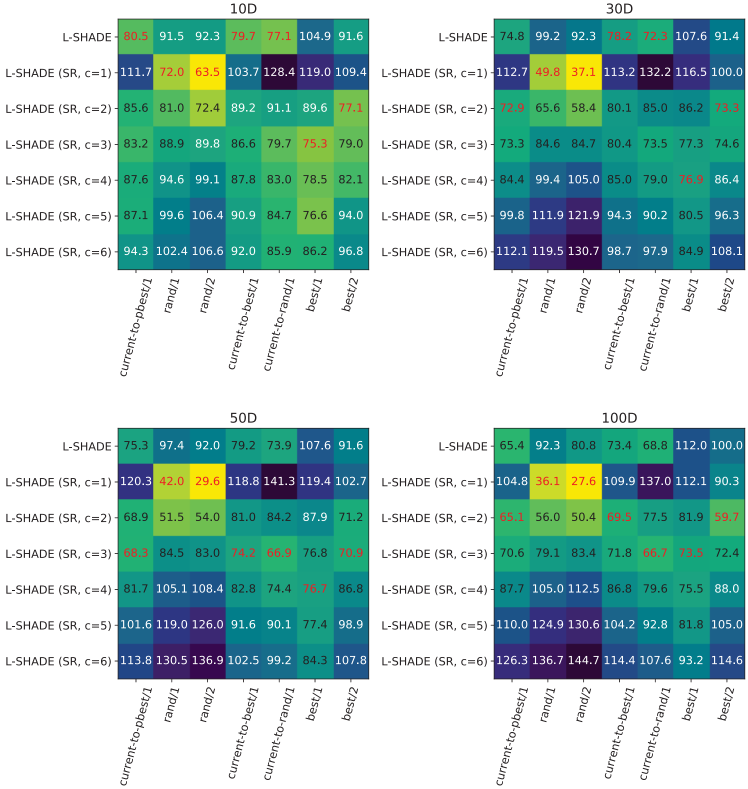


**Figure 6.** Comparison of L-SHADE with different strategies, with and without success rate adaptation, CEC 2017, Friedman ranking.

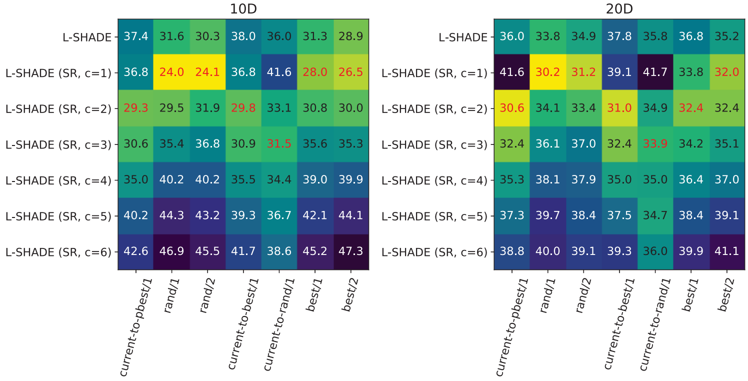


**Figure 7.** Comparison of L-SHADE with different strategies, with and without success rate adaptation, CEC 2022, Friedman ranking.

The heatmaps for the CEC 2017 benchmark show that in low-dimensional cases like 10D and 30D, the SHA may perform better than SR, but only for current-to-pbest/1, current-to-best/1 and current-to-rand/1. As for the other strategies, rand/1 and rand/2 performed much better with the modified approach, and so did best/1 and best/2, although the difference here was smaller. In high-dimensional cases, SR is always performing better, especially when the rand/2 strategy is used. Different strategies prefer different $c$ values;

for example, rand/1 and rand/2 only use $c = 1$, i.e., linear dependence. In fact, this means that the success rate is used directly. Other strategies, such as best/1 and best/2, prefer $c = 2$, $c = 3$ or $c = 4$, but this value is always smaller for best/2 compared to best/1. The same is true when comparing rand/1 and rand/2: the effect of smaller $c$ values is larger in the case of rand/2. The current-to-best and current-to-rand strategies are similar to current-to-pbest ones, and the best settings for them are close. The comparison on CEC 2022 demonstrates the same trends, but in general smaller $c$ values are preferred for all strategies except current-to-rand ones. Also, on this benchmark, the SHA is never the best choice.

To compare different strategies with current-to-pbest, the best performing ones from every dimension and every benchmark were chosen and compared to current-to-pbest with SR. The standard SHA was not considered here. The Mann–Whitney statistical tests for both benchmarks are shown in Tables 11 and 12.

**Table 11.** L-SHADE (SR) with different mutation strategies, current-to-pbest strategy vs. best variants, CEC 2017.

| Strategy | 10$D$ | 30$D$ | 50$D$ | 100$D$ |
|---|---|---|---|---|
| current-to-pbest/1 vs.<br>rand/1 | 10/11/9<br>(26.18) | 19/9/2<br>(110.36) | 24/4/2<br>(150.35) | 18/8/4<br>(133.91) |
| current-to-pbest/1 vs.<br>rand/2 | 11/16/3<br>(56.51) | 20/9/1<br>(134.71) | 25/3/2<br>(174.02) | 20/6/4<br>(146.97) |
| current-to-pbest/1 vs.<br>current-to-best/1 | 15/10/5<br>(45.26) | 5/22/3<br>(17.98) | 4/22/4<br>(3.20) | 11/10/9<br>(19.94) |
| current-to-pbest/1 vs.<br>current-to-rand/1 | 11/17/2<br>(52.97) | 19/11/0<br>(119.34) | 19/9/2<br>(135.04) | 22/7/1<br>(164.06) |
| current-to-pbest/1 vs.<br>best/1 | 9/17/4<br>(20.28) | 15/11/4<br>(80.90) | 23/4/3<br>(108.31) | 20/6/4<br>(107.48) |
| current-to-pbest/1 vs.<br>best/2 | 11/16/3<br>(27.58) | 17/11/2<br>(80.53) | 24/4/2<br>(123.93) | 20/6/4<br>(108.48) |

**Table 12.** L-SHADE (SR) with different mutation strategies, current-to-pbest strategy vs. best variants, CEC 2022.

| Strategy | 10$D$ | 20$D$ |
|---|---|---|
| current-to-pbest/1 vs. rand/1 | 4/5/3 (1.77) | 3/5/4 (−5.97) |
| current-to-pbest/1 vs. rand/2 | 4/5/3 (8.87) | 2/7/3 (−1.72) |
| current-to-pbest/1 vs. current-to-best/1 | 1/7/4 (−15.58) | 1/8/3 (−8.00) |
| current-to-pbest/1 vs. current-to-rand/1 | 7/4/1 (35.73) | 5/6/1 (22.00) |
| current-to-pbest/1 vs. best/1 | 5/3/4 (1.91) | 5/4/3 (12.18) |
| current-to-pbest/1 vs. best/2 | 3/4/5 (−10.48) | 6/2/4 (13.10) |

In the case of the CEC 2017 benchmark in Table 11, the current-to-pbest strategy dominates other strategies in terms of performance in all cases. However, the closest competitor is the current-to-best which performed similar in the 50D case, which might indicate a non-optimal choice of the *pbest* parameter for current-to-best. As for the results on CEC 2022, the current-to-pbest outperformed most of the strategies but the current-to-best had several wins in both the 10D and 30D cases. Also, the rand/1 strategy has shown some competitive performance in 20D, as well as best/2 in 10D. Considering the improvements that success rate-based adaptation was able to deliver for these strategies compared to success history adaptation (Figures 6 and 7), one may conclude that SR can be efficiently applied to other strategies, and even make them competitive.

### 4.2.5. Comparison With Alternative Approaches

Tables 13–15 contain the comparison of the L-NTADE (SR) algorithm with $c = 4$ to other modern DE algorithms on the CEC 2017 benchmark, and Tables 16–18 on the CEC 2022 benchmark. Table 13 shows the results of Mann–Whitney statistical tests, Table 14 contains the Friedman ranking results, the ranking that is used in a Friedman statistical test to compare several sets of measurements; smaller total ranks are better. Table 15 shows the recently proposed U-scores [39]. Tables 16–18 contain the same results for CEC 2022. The U-scores represent a trial-based dominance method for comparing several optimization methods using Mann–Whitney U tests. For CEC 2017, the convergence speed is not taken into consideration, while for CEC 2022 the number of function evaluations to reach the desired accuracy is considered.

**Table 13.** L-NTADE (SR) vs. alternative approaches, Mann–Whitney tests, CEC 2017.

| Algorithm | 10*D* | 30*D* | 50*D* | 100*D* |
|---|---|---|---|---|
| L-NTADE (SR, $c = 4$) vs.<br>LSHADE-SPACMA [35] | 9/18/3<br>(25.76) | 17/6/7<br>(75.97) | 14/3/13<br>(17.98) | 14/1/15<br>(1.50) |
| L-NTADE (SR, $c = 4$) vs.<br>jSO [25] | 6/21/3<br>(7.27) | 19/10/1<br>(139.25) | 22/6/2<br>(153.40) | 24/1/5<br>(147.83) |
| L-NTADE (SR, $c = 4$) vs.<br>EBOwithCMAR [40] | 3/19/8<br>(−37.99) | 16/9/5<br>(79.00) | 18/7/5<br>(110.30) | 22/2/6<br>(131.61) |
| L-NTADE (SR, $c = 4$) vs.<br>L-SHADE-RSP [9] | 3/24/3<br>(2.69) | 18/11/1<br>(127.15) | 20/8/2<br>(131.90) | 20/6/4<br>(121.52) |
| L-NTADE (SR, $c = 4$) vs.<br>NL-SHADE-RSP [10] | 12/9/9<br>(17.82) | 23/4/3<br>(173.49) | 29/1/0<br>(250.21) | 29/0/1<br>(241.25) |
| L-NTADE (SR, $c = 4$) vs.<br>NL-SHADE-LBC [11] | 6/20/4<br>(6.58) | 21/8/1<br>(176.68) | 28/2/0<br>(227.05) | 27/1/2<br>(210.08) |
| L-NTADE (SR, $c = 4$) vs.<br>L-NTADE [12] | 11/14/5<br>(37.30) | 14/15/1<br>(75.39) | 15/11/4<br>(69.26) | 15/11/4<br>(70.63) |

**Table 14.** L-NTADE (SR) vs. alternative approaches; Friedman ranking. Smaller ranks are better, CEC 2017.

| Algorithm | 10*D* | 30*D* | 50*D* | 100*D* | Total |
|---|---|---|---|---|---|
| LSHADE-SPACMA [35] | 139.42 | 127.42 | 100.55 | 85.06 | 452.45 |
| jSO [25] | 134.97 | 131.33 | 132.14 | 139.09 | 537.53 |
| EBOwithCMAR [40] | **116.15** | 123.19 | 124.94 | 131.67 | 495.94 |
| L-SHADE-RSP [9] | 133.93 | 126.37 | 121.91 | 122.36 | 504.58 |
| NL-SHADE-RSP [10] | 146.25 | 196.87 | 226.37 | 226.63 | 796.13 |
| NL-SHADE-LBC [11] | 133.80 | 177.80 | 193.38 | 191.89 | 696.88 |
| L-NTADE [12] | 143.55 | 110.41 | 100.85 | 101.93 | 456.75 |
| L-NTADE (SR, $c = 4$) | 131.92 | 86.60 | 79.85 | 81.37 | 379.75 |

As can be seen from Table 13, the L-NTADE algorithm with success rate-based adaptation is able to outperform most of the alternative algorithms in both benchmarks. EBOwithCMAR performed better in the 10D case, and LSHADE-SPACMA showed comparable in the 100D case. Table 14 supports these results, ranking L-NTADE with $c = 4$ the best algorithm overall, and even the standard L-NTADE is very competitive. In Table 15, the comparison yields similar results, and L-NTADE with $c = 4$ ranks first, followed by LSHADE-SPACMA and L-NTADE.

**Table 15.** L-NTADE (SR) vs. alternative approaches, U-scores. Larger ranks are better [39], CEC 2017.

| **Algorithm** | **10*D*** | **30*D*** | **-** |
|---|---|---|---|
| LSHADE-SPACMA [35] | 263,193 | 293,839.5 | - |
| jSO [25] | 272,469 | 282,122.5 | - |
| EBOwithCMAR [40] | 322,146.5 | 303,458.5 | - |
| L-SHADE-RSP [9] | 276,570.5 | 296,012 | - |
| NL-SHADE-RSP [10] | 244,170 | 112,109 | - |
| NL-SHADE-LBC [11] | 276,146 | 161,327 | - |
| L-NTADE [12] | 248,186 | 336,712.5 | - |
| L-NTADE (SR, $c = 4$) | 281,959 | 399,259 | - |
| Algorithm | 50*D* | 100*D* | Total |
| LSHADE-SPACMA [35] | 364,005.5 | 404,653 | 1,325,691 |
| jSO [25] | 281,506 | 262,631 | 1,098,728.5 |
| EBOwithCMAR [40] | 298,859 | 282,227 | 1,206,691 |
| L-SHADE-RSP [9] | 305,628.5 | 305,817 | 1,184,028 |
| NL-SHADE-RSP [10] | 34,664.5 | 34,687 | 425,630.5 |
| NL-SHADE-LBC [11] | 121,800.5 | 124,111 | 683,384.5 |
| L-NTADE [12] | 362,005 | 359,221 | 1,306,124.5 |
| L-NTADE (SR, $c = 4$) | 416,371 | 411,493 | 1,509,082 |

**Table 16.** L-NTADE (SR) vs. alternative approaches, CEC 2022.

| **Algorithm** | **10*D*** | **20*D*** |
|---|---|---|
| L-NTADE (SR, $c = 4$) vs.<br>APGSK-IMODE [36] | 8/2/2<br>(36.86) | 10/0/2<br>(52.60) |
| L-NTADE (SR, $c = 4$) vs.<br>MLS-LSHADE [37] | 4/2/6<br>(−9.67) | 4/1/7<br>(−19.78) |
| L-NTADE (SR, $c = 4$) vs.<br>MadDE [38] | 8/2/2<br>(39.05) | 8/2/2<br>(43.60) |
| L-NTADE (SR, $c = 4$) vs.<br>EA4eigN100 [41] | 2/6/4<br>(−15.49) | 5/3/4<br>(5.51) |
| L-NTADE (SR, $c = 4$) vs.<br>NL-SHADE-RSP-MID [42] | 3/4/5<br>(−5.98) | 6/2/4<br>(17.59) |
| L-NTADE (SR, $c = 4$) vs.<br>L-SHADE-RSP [9] | 5/3/4<br>(14.23) | 4/6/2<br>(12.24) |
| L-NTADE (SR, $c = 4$) vs.<br>NL-SHADE-RSP [10] | 6/3/3<br>(24.51) | 7/4/1<br>(34.78) |
| L-NTADE (SR, $c = 4$) vs.<br>NL-SHADE-LBC [11] | 2/3/7<br>(−30.20) | 4/4/4<br>(−3.75) |
| L-NTADE (SR, $c = 4$) vs.<br>L-NTADE [12] | 5/5/2<br>(26.17) | 2/6/4<br>(−1.82) |

The comparison on the CEC 2022 benchmark in Table 16 shows that L-NTADE with $c = 4$ is outperformed by NL-SHADE-LBC and EA4eigN100 in the 10D case, but in 20D the difference between them decreases. However, the MLS-LSHADE is able to deliver better performance in the 20D case. The Friedman ranking in Table 17 sets EA4eigN100 in first place, followed by NL-SHADE-LBC, and MLS-LSHADE and L-NTADE with $c = 4$. However, when using U-scores in Table 18, L-NTADE with $c = 4$ takes second place after NL-SHADE-LBC, and MLS-LSHADE has an almost identical total rank. We note that L-NTADE used the same parameter settings in both benchmarks, and still was able to show highly competitive results in both of them.

**Table 17.** L-NTADE (SR) vs. alternative approaches, Friedman ranking. Smaller ranks are better, CEC 2022.

| Algorithm | 10$D$ | 20$D$ | Total |
|---|---|---|---|
| APGSK-IMODE [36] | 81.57 | 87.92 | 169.48 |
| MLS-LSHADE [37] | 64.05 | 50.13 | 114.18 |
| MadDE [38] | 87.15 | 87.02 | 174.17 |
| EA4eigN100 [41] | 39.17 | 53.02 | 92.18 |
| NL-SHADE-RSP-MID [42] | 58.92 | 71.25 | 130.17 |
| L-SHADE-RSP [9] | 66.65 | 59.07 | 125.72 |
| NL-SHADE-RSP [10] | 88.23 | 83.58 | 171.82 |
| NL-SHADE-LBC [11] | 49.77 | 56.73 | 106.50 |
| L-NTADE [12] | 64.58 | 55.92 | 120.50 |
| L-NTADE (SR, $c = 4$) | 59.92 | 55.37 | 115.28 |

**Table 18.** L-NTADE (SR) vs. alternative approaches, U-scores. Larger ranks are better [39], CEC 2022.

| Algorithm | 10$D$ | 20$D$ | Total |
|---|---|---|---|
| APGSK-IMODE [36] | 38,228 | 28,381.5 | 66,609.5 |
| MLS-LSHADE [37] | 52,255 | 62,548.5 | 114,803.5 |
| MadDE [38] | 35,727 | 30,402.5 | 66,129.5 |
| EA4eigN100 [41] | 50358 | 60,202 | 110,560 |
| NL-SHADE-RSP-MID [42] | 56,740.5 | 44,291.5 | 101,032 |
| L-SHADE-RSP [9] | 44,754.5 | 54,543.5 | 99,298 |
| NL-SHADE-RSP [10] | 35,363.5 | 32,772.5 | 68,136 |
| NL-SHADE-LBC [11] | 64,330 | 56,998.5 | 121,328.5 |
| L-NTADE [12] | 51,346 | 57,894 | 109,240 |
| L-NTADE (SR, $c = 4$) | 56,897.5 | 57,965.5 | 114,863 |

In order to compare the computational complexity of the proposed approach, the experiment on the CEC 2022 benchmark was performed according to the rules of the CEC 2022 competition rules [14]. The T0, T1 and T2 values are the estimations of time required for the processor to perform mathematical evaluations, target function evaluation time and the algorithm runtime, respectively. Table 19 compares L-NTADE with the SHA and the SR adaptation methods.

**Table 19.** Computational complexity of L-NTADE with SHA and the SR adaptation methods.

| **L-NTADE** | | | | |
|---|---|---|---|---|
| $D$ | $T0$ | $T1$ | $T2$ | $(T2-T1)/T0$ |
| $D=10$ | $2.5\times10^{-2}$ | $2.1\times10^{-2}$ | $1.358\times10^{-1}$ | 4.592 |
| $D=20$ | $2.5\times10^{-2}$ | $5.0\times10^{-2}$ | $1.704\times10^{-1}$ | 4.816 |
| **L-NTADE (SR, $c=4$)** | | | | |
| $D$ | $T0$ | $T1$ | $T2$ | $(T2-T1)/T0$ |
| $D=10$ | $2.5\times10^{-2}$ | $2.1\times10^{-2}$ | $1.192\times10^{-1}$ | 3.928 |
| $D=20$ | $2.5\times10^{-2}$ | $5.0\times10^{-2}$ | $1.658\times10^{-1}$ | 4.632 |

As can be seen from Table 19, applying SR reduces the amount of time, as the calculation of the weighted Lehmer mean for $F$ in success history-based adaptation requires significant computational effort.

## 5. Discussion

The main advantage of the proposed success rate-based adaptation of the scaling factor is its simplicity. As simple as the original DE, it relies on a naturally present value in the algorithm, namely the number of successful individuals, i.e., the number of solutions that were improved according to the selection step. This number is, in fact, calculated by the majority of existing DE algorithms. Success rate-based adaptation does not require complex calculations of means, averaging over time and other techniques, and still performs very well. Moreover, the experiments have shown that the sensitivity to the $c$ parameter is very low: the same value, $c=4$, works on different benchmarks, different functions and different computational resource. This property makes for a universal approach, which seems to work well enough in most scenarios. Although there are cases when the success history adaptation may deliver better performance, we believe that this can be mitigated by further development of SR-based adaptation schemes.

As for the reasons for high performance, and why $c=4$ is a good choice in many cases, the following explanation seems reasonable. The SR in most tested algorithms worked together with the current-to-pbest mutation strategy, which has a specific structure. If $F>0.5$, then the new solutions are attracted closer to some of the best ones, although this also means that larger steps in other directions will be made. If $F<0.5$, this means that the attention towards better solutions is smaller, and the algorithm searches more around the current positions of each individual. What the success rate adaptation does is it switches between these two behaviors. If the search is going well, it makes DE generate solutions closer to better ones, and the $F$ values are relatively stable. However, if the algorithm is stagnating, a wider search is beneficial. In this case, success rate adaptation makes $MF$ oscillate in the $[0,0.5]$ range, sampling smaller $F$ values and trying to escape local optima. In this manner, applying a simple curve like $SR^{1/4}$ produces the desired behavior of DE without complex adaptation mechanisms.

The experiments with other mutation strategies revealed that SR-based adaptation works not only with the current-to-pbest strategy, but also with other ones, and in all cases there was a setting of $c$ which lead to improved performance. Moreover, in most cases, any $c$ value resulted in better performance than SHA. In the case of the rand/1 and rand/2 strategies, this improvement was very significant, and even allowed for current-to-pbest to be outperformed in some cases. This might mean that the SR-based adaptation can be used as a universal mechanism, independent of the mutation strategy, although the $c$ value should be set accordingly.

Replacing the original success history adaptation in other DE-based algorithms, such as APGSK-IMODE, MLS-LSHADE, MadDE, jSO and LSHADE-SPACMA, did not always improve performance; nevertheless, the results were mostly comparable. Some of the modified algorithms represented hybrids with other methods, such as local search or covariance matrix adaptation, and probably better results could be achieved with a more careful tuning of the hybridization.

Of course, the presented experiments cannot cover all the possibilities of success rate-based adaptation. However, it is worth mentioning some of the possible directions of further studies, which include:

1. Developing a crossover rate adaptation scheme based on the success rate,
2. Connecting the success rate to the *pbest* parameter,
3. Combining the advantages of the success history and success rate adaptation.

## 6. Conclusions

In this study, a new adaptation technique for the scaling factor parameter in differential evolution was proposed. The new method relies on the success rate, and the performed experimental analysis showed that the new method is insensitive to computational resource, works with different mutation strategies, and can be included in various algorithms.

**Author Contributions:** Conceptualization, V.S. and E.S.; methodology, V.S. and E.S.; software, V.S.; validation, V.S. and E.S.; formal analysis, E.S.; investigation, V.S.; resources, E.S. and V.S.; data curation, E.S.; writing—original draft preparation, V.S.; writing—review and editing, V.S.; visualization, V.S.; supervision, E.S.; project administration, E.S.; funding acquisition, V.S. and E.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the Ministry of Science and Higher Education of the Russian Federation within limits of state contract No. FEFE-2023-0004.

**Data Availability Statement:** Data are contained within the article.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| GA | Genetic Algorithms |
| GP | Genetic Programming |
| EC | Evolutionary Computation |
| DE | Differential Evolution |
| CEC | Congress on Evolutionary Computation |
| SHADE | Success History Adaptive Differential Evolution |
| LPSR | Linear Population Size Reduction |
| LBC | Linear Bias Change |
| RSP | Rank-based Selective Pressure |
| L-NTADE | Linear population size reduction Newest and Top Adaptive Differential Evolution |

## References

1. Price, K.; Storn, R.; Lampinen, J. *Differential Evolution: A Practical Approach to Global Optimization*; Springer: Berlin/Heidelberg, Germany, 2005.
2. Das, S.; Suganthan, P. Differential evolution: a survey of the state-of-the-art. *IEEE Trans. Evol. Comput.* **2011**, *15*, 4–31. [CrossRef]
3. Brest, J.; Greiner, S.; Boškovic, B.; Mernik, M.; Žumer, V. Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. *IEEE Trans. Evol. Comput.* **2006**, *10*, 646–657. [CrossRef]
4. Zhang, J.; Sanderson, A.C. JADE: Adaptive Differential Evolution With Optional External Archive. *IEEE Trans. Evol. Comput.* **2009**, *13*, 945–958. [CrossRef]
5. Tanabe, R.; Fukunaga, A. Success-history based parameter adaptation for differential evolution. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico, 20–23 June 2013; pp. 71–78. [CrossRef]
6. Piotrowski, A.P.; Napiorkowski, J.J. Step-by-step improvement of JADE and SHADE-based algorithms: Success or failure? *Swarm Evol. Comput.* **2018**, *43*, 88–108. [CrossRef]
7. Stanovov, V.; Akhmedova, S.; Semenkin, E. Biased Parameter Adaptation in Differential Evolution. *Inf. Sci.* **2021**, *566*, 215–238. [CrossRef]

8. Stanovov, V.; Akhmedova, S.; Semenkin, E. The automatic design of parameter adaptation techniques for differential evolution with genetic programming. *Knowl. Based Syst.* **2022**, *239*, 108070. [CrossRef]
9. Stanovov, V.; Akhmedova, S.; Semenkin, E. LSHADE Algorithm with Rank-Based Selective Pressure Strategy for Solving CEC 2017 Benchmark Problems. In Proceedings of the 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
10. Stanovov, V.; Akhmedova, S.; Semenkin, E. NL-SHADE-RSP Algorithm with Adaptive Archive and Selective Pressure for CEC 2021 Numerical Optimization. In Proceedings of the 2021 IEEE Congress on Evolutionary Computation (CEC), Kraków, Poland, 28 June–1 July 2021; pp. 809–816. [CrossRef]
11. Stanovov, V.; Akhmedova, S.; Semenkin, E. NL-SHADE-LBC algorithm with linear parameter adaptation bias change for CEC 2022 Numerical Optimization. In Proceedings of the 2022 IEEE Congress on Evolutionary Computation (CEC), Padua, Italy, 18–23 July 2022.
12. Stanovov, V.; Akhmedova, S.; Semenkin, E. Dual-Population Adaptive Differential Evolution Algorithm L-NTADE. *Mathematics* **2022**, *10*, 4666. [CrossRef]
13. Awad, N.; Ali, M.; Liang, J.; Qu, B.; Suganthan, P. *Problem Definitions and Evaluation Criteria for the CEC 2017 special Session and Competition on Single Objective Bound Constrained Real-Parameter Numerical Optimization*; Technical Report; Nanyang Technological University: Singapoure, 2016.
14. Kumar, A.; Price, K.; Mohamed, A.K.; Suganthan, P.N. *Problem Definitions and Evaluation Criteria for the CEC 2022 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization*; Technical Report; Nanyang Technological University: Singapoure, 2021.
15. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
16. Biedrzycki, R.; Arabas, J.; Jagodziński, D. Bound constraints handling in Differential Evolution: An experimental study. *Swarm Evol. Comput.* **2019**, *50*, 100453. [CrossRef]
17. Kumar, A.; Biswas, P.P.; Suganthan, P.N. Differential evolution with orthogonal array-based initialization and a novel selection strategy. *Swarm Evol. Comput.* **2022**, *68*, 101010. [CrossRef]
18. Das, S.; Mullick, S.; Suganthan, P. Recent advances in differential evolution—An updated survey. *Swarm Evol. Comput.* **2016**, *27*, 1–30. [CrossRef]
19. Al-Dabbagh, R.D.; Neri, F.; Idris, N.; Baba, M.S.B. Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy. *Swarm Evol. Comput.* **2018**, *43*, 284–311. [CrossRef]
20. Brest, J.; Maucec, M.; Bovsković, B. The 100-Digit Challenge: Algorithm jDE100. In Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC), Wellington, New Zealand, 10–13 June 2019; pp. 19–26.
21. Brest, J.; Maucec, M.; Bosković, B. Differential Evolution Algorithm for Single Objective Bound-Constrained Optimization: Algorithm j2020. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020; pp. 1–8.
22. Qin, A.K.; Suganthan, P.N. Self-adaptive differential evolution algorithm for numerical optimization. In Proceedings of the 2005 IEEE Congress on Evolutionary Computation, Edinburgh, UK, 2–5 September 2005; Volume 2; pp. 1785–1791 .
23. Bullen, P. *Handbook of Means and Their Inequalities*; Springer: Amsterdam, The Netherlands, 2003. [CrossRef]
24. Tanabe, R.; Fukunaga, A. Improving the search performance of SHADE using linear population size reduction. In Proceedings of the IEEE Congress on Evolutionary Computation, CEC, Beijing, China, 6–11 July 2014; pp. 1658–1665. [CrossRef]
25. Brest, J.; Maučec, M.; Boškovic, B. Single objective real-parameter optimization algorithm jSO. In Proceedings of the IEEE Congress on Evolutionary Computation, Donostia, Spain, 5–8 June 2017; pp. 1311–1318. [CrossRef]
26. Gong, W.; Cai, Z. Differential Evolution With Ranking-Based Mutation Operators. *IEEE Trans. Cybern.* **2013**, *43*, 2066–2081. [CrossRef] [PubMed]
27. Viktorin, A.; Senkerik, R.; Pluhacek, M.; Kadavy, T.; Zamuda, A. Distance based parameter adaptation for Success-History based Differential Evolution. *Swarm Evol. Comput.* **2019**, *50*, 100462. [CrossRef]
28. Bujok, P.; Kolenovsky, P. Differential Evolution with Distance-based Mutation-selection Applied to CEC 2021 Single Objective Numerical Optimisation. In Proceedings of the 2021 IEEE Congress on Evolutionary Computation (CEC), Kraków, Poland, 28 June–1 July 2021; pp. 849–856.
29. Yang, M.; Cai, Z.; Li, C.; Guan, J. An improved adaptive differential evolution algorithm with population adaptation. In Proceedings of the Annual Conference on Genetic and Evolutionary Computation, Amsterdam, The Netherlands, 6–10 July 2013.
30. Santucci, V.; Baioletti, M.; Bari, G.D. An improved memetic algebraic differential evolution for solving the multidimensional two-way number partitioning problem. *Expert Syst. Appl.* **2021**, *178*, 114938. [CrossRef]
31. Chen, X.; Shen, A. Self-adaptive differential evolution with Gaussian–Cauchy mutation for large-scale CHP economic dispatch problem. *Neural Comput. Appl.* **2022**, *34*, 11769–11787. [CrossRef]
32. Yi, W.; Chen, Y.; Pei, Z.; Lu, J. Adaptive differential evolution with ensembling operators for continuous optimization problems. *Swarm Evol. Comput.* **2021**, *69*, 100994. [CrossRef]
33. Yang, Q.; Qiao, Z.Y.; Xu, P.; Lin, X.; Gao, X.D.; Wang, Z.J.; Lu, Z.Y.; Jeon, S.W.; Zhang, J. Triple competitive differential evolution for global numerical optimization. *Swarm Evol. Comput.* **2024**, *84*, 101450. [CrossRef]

34. Kitamura, T.; Fukunaga, A. Differential Evolution with an Unbounded Population. In Proceedings of the 2022 IEEE Congress on Evolutionary Computation (CEC), Padua, Italy, 18–23 July 2022.
35. Mohamed, A.; Hadi, A.A.; Fattouh, A.; Jambi, K. LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, Spain, 5–8 June 2017; pp. 145–152.
36. Mohamed, A.W.; Hadi, A.A.; Agrawal, P.; Sallam, K.M.; Mohamed, A.K. Gaining-Sharing Knowledge Based Algorithm with Adaptive Parameters Hybrid with IMODE Algorithm for Solving CEC 2021 Benchmark Problems. In Proceedings of the 2021 IEEE Congress on Evolutionary Computation (CEC), Kraków, Poland, 28 June–1 July 2021; pp. 841–848.
37. Cuong, L.V.; Bao, N.N.; Binh, H.T.T. T*echnical Report: A Multi-Start Local Search Algorithm with L-SHADE for Single Objective Bound Constrained Optimization*; Technical Report; SoICT, Hanoi University of Science and Technology: Hanoi, Vietnam, 2021.
38. Biswas, S.; Saha, D.; De, S.; Cobb, A.D.; Das, S.; Jalaian, B. Improving Differential Evolution through Bayesian Hyperparameter Optimization. In Proceedings of the 2021 IEEE Congress on Evolutionary Computation (CEC), Kraków, Poland, 28 June–1 July 2021; pp. 832–840.
39. Price, K.V.; Kumar, A.; Suganthan, P. Trial-based dominance for comparing both the speed and accuracy of stochastic optimizers with standard non-parametric tests. *Swarm Evol. Comput.* **2023**, *78*, 101287. [CrossRef]
40. Kumar, A.; Misra, R.K.; Singh, D. Improving the local search capability of Effective Butterfly Optimizer using Covariance Matrix Adapted Retreat Phase. In Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, Spain, 5–8 June 2017; pp. 1835–1842.
41. Bujok, P.; Kolenovsky, P. Eigen Crossover in Cooperative Model of Evolutionary Algorithms Applied to CEC 2022 Single Objective Numerical Optimisation. In Proceedings of the 2022 IEEE Congress on Evolutionary Computation (CEC), Padua, Italy, 18–23 July 2022.
42. Biedrzycki, R.; Arabas, J.; Warchulski, E. A Version of NL-SHADE-RSP Algorithm with Midpoint for CEC 2022 Single Objective Bound Constrained Problems. In Proceedings of the 2022 IEEE Congress on Evolutionary Computation (CEC), Padua, Italy, 18–23 July 2022.

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

mathematics

MDPI

*Article*

# Decision-Making on the Diagnosis of Oncological Diseases Using Cost-Sensitive SVM Classifiers Based on Datasets with a Variety of Features of Different Natures

**Liliya A. Demidova**

Institute of Information Technologies, Federal State Budget Educational Institution of Higher Education, MIREA—Russian Technological University, 78, Vernadsky Avenue, 119454 Moscow, Russia; liliya.demidova@rambler.ru

**Abstract:** This paper discusses the problem of detecting cancer using such biomarkers as blood protein markers. The purpose of this research is to propose an approach for making decisions in the diagnosis of cancer through the creation of cost-sensitive SVM classifiers on the basis of datasets with a variety of features of different nature. Such datasets may include compositions of known features corresponding to blood protein markers and new features constructed using methods for calculating entropy and fractal dimensions, as well as using the UMAP algorithm. Based on these datasets, multiclass SVM classifiers were developed. They use cost-sensitive learning principles to overcome the class imbalance problem, which is typical for medical datasets. When implementing the UMAP algorithm, various variants of the loss function were considered. This was performed in order to select those that provide the formation of such new features that ultimately allow us to develop the best cost-sensitive SVM classifiers in terms of maximizing the mean value of the metric $MacroF_1 - score$. The experimental results proved the possibility of applying the UMAP algorithm, approximate entropy and, in addition, Higuchi and Katz fractal dimensions to construct new features using blood protein markers. It turned out that when working with the UMAP algorithm, the most promising is the application of a loss function on the basis of fuzzy cross-entropy, and the least promising is the application of a loss function on the basis of intuitionistic fuzzy cross-entropy. Augmentation of the original dataset with either features on the basis of the UMAP algorithm, features on the basis of the UMAP algorithm and approximate entropy, or features on the basis of approximate entropy provided the creation of the three best cost-sensitive SVM classifiers with mean values of the metric $MacroF_1 - score$ increased by 5.359%, 5.245% and 4.675%, respectively, compared to the mean values of this metric in the case when only the original dataset was utilized for creating the base SVM classifier (without performing any manipulations to overcome the class imbalance problem, and also without introducing new features).

**Keywords:** oncological disease; cost-sensitive SVM classifier; features; UMAP algorithm; loss function; entropy; fractal dimension

**MSC:** 68Q32; 68T05

**Citation:** Demidova, L.A. Decision-Making on the Diagnosis of Oncological Diseases Using Cost-Sensitive SVM Classifiers Based on Datasets with a Variety of Features of Different Natures. *Mathematics* **2024**, *12*, 538. https://doi.org/10.3390/math12040538

Academic Editor: Liangxiao Jiang

Received: 5 January 2024
Revised: 28 January 2024
Accepted: 5 February 2024
Published: 8 February 2024

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

## 1. Introduction

At present, the processes of digital transformation are becoming more and more apparent and sought-after in many spheres of human society, including the spheres of medicine and healthcare. First of all, digital transformation in the spheres of medicine and healthcare is a complex continuous process that involves a complete restructuring of the fundamental principles of the functioning of medical organizations at all hierarchy levels, as well as the concept of their work with patients [1]. Now, the implementation of innovative digital technologies is aimed at establishing high standards of healthcare delivery and moving towards the "4P medicine" model [1,2], which integrates preventive, personalized, participatory and predictive aspects of medical practice.

Digital transformation in sphere of healthcare consists of the transition from standard approved clinical approaches recommended for use in the examination and treatment of patients, to personal and individual approaches, as well as the prevention of certain diseases through timely early diagnosis, the constant monitoring of patients, active involvement of patients in treatment process, etc. [2]. An important key aspect of the digital transformation in the sphere of healthcare involves creating the prerequisites for reducing morbidity and mortality, as well as for increasing the active life expectancy of a person. Advanced health monitoring technologies should help not only to detect certain diseases in their early stages, but also to prevent disease progression through the application of innovative treatments for certain diseases.

The design of intellectual analysis tools for processing medical data of large and ultra-large volumes with the involvement of advanced machine learning (ML) [3–7] and deep learning (DL) [8–12] technologies provides an opportunity to receive and analyze new previously hidden knowledge both directly in the medical sphere and in related ones. These technologies are actively applied in solving various problems of medical diagnosis and, in particular, in solving problems of diagnosis of oncological diseases (ODs). The logistic regression algorithm [13], k-nearest neighbors (kNN) algorithm [14], support vector machine (SVM) algorithm [15], random forest (RF) algorithm [16] and DL algorithms [8–10] are usually applied when creating classifiers to solve OD diagnosis problems. Such classifiers are created on the basis of datasets which accommodate information about both patterns with diagnosed ODs of various types and patterns with unconfirmed ODs (i.e., normal patterns) [13–20].

Recently, applied and computational mathematics tools have become increasingly in demand, especially in the sense of digital transformation in healthcare. At the present time, the integration of applied and computational mathematics into digital health platforms is an important factor for efficient analysis and processing of medical data. These tools provide more accurate detection of pathologies at early stages of development, which is critical for the successful prevention of diseases, including cancer [13,21–23]. The use of applied and computational mathematics methods in the digital transformation of healthcare not only improves the accuracy of diagnosis but also contributes to the formation of innovative methods of disease prevention and treatment. As a result, the basis for personalized and predictive healthcare strategies is created, including strategies of cancer prevention and diagnosis.

Oncological diseases (ODs) are considered to be one of the most critically dangerous diseases due to the potentially severe consequences for patients, especially with late diagnosis [24,25]. These consequences include severe pain and serious psychological distress. Treatment for ODs can take a very long time. In addition, it is associated with huge financial costs both on the part of patients and on the part of the state. ODs are dangerous immune violations that cause abnormal cells to divide and grow in some organ of the patient. In addition, these immune violations can very quickly cover the patient's entire body and lead to a sad outcome. Clearly, timely early diagnosis of ODs is crucial so that the doctor can promptly choose an effective method for treating the patient. Unfortunately, the problem of early diagnosis of ODs is challenging and extremely difficult, because obvious characteristic symptoms do not manifest themselves until late in the course of the disease, so even innovative treatments may not be effective.

One approach to early diagnosis of ODs involves analyzing the results of various tests, for example, gene tests (GTs) [13,26] and protein tests (PTs) [13,27,28]. In the last few years, experts have favored PT-based OD diagnostic tools involving blood protein markers [13–16,26]. While GTs are static, PTs are dynamic, so PTs, when performed in a timely manner, can detect the disease onset and monitor its progression [13]. In addition, PTs are performed non-invasively. Also, they are cheap. It is assumed that PT-based diagnostic technologies provide the forecasting of risks of cancer development for 1–3 years in advance and, therefore, allow us to carry out advanced prevention and diagnosis of ODs.

Various protein markers are present in the blood. It is known that for different types of ODs, the values of blood protein markers (BPMs) differ from each other [13]. It can be reasonably assumed that taking into account the entire spectrum of BPMs should provide an increase in the accuracy of diagnostics of various diseases. The use of data mining (DM) tools with the involvement of ML and DL technologies will allow us to reveal the relationships between the values of BPMs hidden in PT data for different types of ODs [13,14,20].

The main problem that arises when solving medical diagnosis problems is the class imbalance of the dataset [14,20]. As a rule, the number of normal data patterns describing situations with the absence of any ODs greatly exceeds the number of pathologic ones describing situations of OD of one type or another. As a result, the pattern class of normal data represents the majority class, while the pattern classes of pathologic data represent the minority classes (minority classes). When solving the problem of early diagnosis of ODs, the target classes, i.e., the classes whose correct classification of patterns is most important, are the minority classes. In addition, the medical diagnosis problem itself is usually a multiclass classification problem.

The problem of creating a multiclass classifier on the basis of an imbalanced dataset is very difficult, as the classifier must be trained to accurately classify the patterns of different classes that are imbalanced. Currently, a lot of approaches to overcoming the class imbalance problem have been proposed [29,30]. The most commonly used approaches are those that implement various class balancing algorithms realizing the strategies of oversampling [31–34], undersampling [34,35] and their combinations, as well as approaches that implement cost-sensitivity learning (CSL) and take into account the cost of incorrect decisions [14,36]. In [29], the authors show that, currently, there is no universal approach to address class imbalance. They propose a taxonomy that covers methods to eliminate class imbalance such as through performing cost-sensitive classification or through data sampling. The authors show that a lot of DM problems are cost-sensitive and class-imbalanced. In [30], the authors note that it is common to use data-level approaches, algorithm-level approaches, ensemble approaches and hybrid approaches to deal with class imbalance. They present a systematic literature review and perform an analysis of the studies presented in more than 400 papers from 2002 to June 2017. This analysis emphasizes the significant impact that inherent problems in the data have on the results obtained from classification problems. In addition, the analysis covers methods for handling imbalanced data and methods used to deal with skewed data distributions. The authors reveal trends and gaps in this sphere of research and discuss directions for future research.

Obviously, in each specific case, when applying one or another method to overcoming the class imbalance problem, it is necessary to check whether undesirable effects have not appeared, for example, in the form of a loss of representative data during undersampling, the appearance of mistaken or redundant data during oversampling, a significant increase in time for classifier development, a significant decrease in accuracy for classifying patterns of the majority class, etc. [20]. In this regard, when evaluating the quality of the created classifier, it is advisable to perform a thorough analysis of different classification quality metrics using the test set, particularly metrics which allow taking the dataset imbalance into account; e.g., such metrics as $F_1 - score$ and balanced accuracy can be used for this purpose. In addition, it is advisable to use k-fold cross-validation to empirically assess the generalization ability of the created classifier.

The problem of working with data accommodating information on blood protein markers is addressed in a number of scientific papers [13,21–25,37,38]. In particular, in [13], the so-called CancerSEEK test based on the logistic classifier (LC) is considered. The authors of the pilot study [13] choose eight types of ODs described by the patterns of this dataset due to the fact that these types are most often found in residents of Western countries and, additionally, because in clinical practice, blood tests were not applied for the early identification of ODs. As a result, the dataset contains information about patterns belonging to nine classes: eight of them correspond to eight types of ODs, and another one corresponds to patterns for which no OD has been diagnosed. Also, the authors consider each individual's sex, levels of eight proteins and facts of mutations in 1933 various genomic locations. Based on the results of the experiments, the authors argue that blood protein markers reflect most of the information about the localization of ODs, because mutations in genes are most often not tissue-specific.

Later, there appeared research that proposed approaches to the development of classifiers that diagnose ODs based on datasets that accommodate data only on blood protein marker values [14,20], that is, they do not take into account data about the values of gene markers. Thus, in [14], the authors propose a cost-sensitive three-class kNN classifier created on the basis of the three-class imbalanced dataset extracted from the dataset used in [13]. The new dataset accommodates only patterns describing information for 39 blood protein markers mapped to 39 features. The authors extract additional information hidden in the 39-dimensional data patterns using sample entropy and approximate entropy, form two new features and add them to the used dataset, hoping to improve the data classification quality. In the research [20] performed earlier by the author of this paper, the aspects of kNN [14,39] and SVM [15,39] classifier development using sampling class balancing tools are considered. In this study, oversampling strategies are applied to balance classes [31–34]. In addition, the research explored various approaches to the formation of new features based on the methods for calculating entropy [40–45], Hjorth parameters [46,47] and fractal dimension [48,49], as well as on the basis of the UMAP (Uniform Manifold Approximation and Projection) algorithm [50–53], which is a nonlinear dimension reduction algorithm. New features were created to be added to the original dataset in different combinations, as well as to independently use these combinations as datasets when developing classifiers.

In general, we can note the high interest of scientists and practitioners in developing approaches to diagnosing cancer based on blood protein markers. At the same time, ideas are proposed for the development of both binary classifiers aimed at identifying any one cancer disease and multiclass classifiers seeking to identify different classes of diseases, which is much more difficult.

The aim of this research is to develop efficient classifiers of ODs (in terms of providing high values of classification quality metrics) using modern DM tools and ML techniques. We propose to develop SVM classifiers [15,20,39] using cost-sensitive algorithms that allow for the different estimation of classification errors in majority and minority classes when working with the original dataset and the extended datasets created on the basis of the original dataset using different tools for forming new features. In particular, when forming new features, as in [20], it is planned to use:

- The UMAP algorithm [50–53], which implements the nonlinear dimensionality reduction of data;
- Methods for calculating the approximate entropy (AE) [45] as well as Higuchi fractal dimension (HFD) [48] and Katz fractal dimension (KFD) [48].

While working with the UMAP algorithm, the plan is to investigate how different loss functions affect the results of embedding the original dataset into a lower-dimensional space. Also, we plan to research how the choice of loss function (LF) affects the quality of the extended datasets and the quality of the SVM classifiers developed from them.

In addition, it is planned to perform a comparative analysis of the SVM classifiers created in this study with the SVM classifiers created in [20], both in terms of the classification quality metrics and the time taken to train and test the SVM classifiers.

The rest of the paper is organized as follows. Section 2 is devoted to a review of works related to the presented research. Section 3 summarizes the design aspects of cost-sensitive SVM classifiers used to address the class imbalance problem in datasets. Furthermore, the applied quality metrics for multiclass classification are emphasized. In addition, a brief description of the principles of the UMAP algorithm and the various LFs used in it, which affect the results of embedding the original dataset from a high-dimensional space into a low-dimensional space, is also provided. Moreover, the methods for computing the approximate entropy, Higuchi fractal dimension and Katz fractal dimension are mentioned. Section 4 presents the experimental results. First, a brief background to this research is given; in particular, a description of the approach to generating datasets and the previously obtained results of the creation of multi-class classifiers using oversampling algorithms are discussed. Then, aspects of the analysis of the original three-class datasets on the basis of the UMAP algorithm using various LFs are considered. Furthermore, the results of developing cost-sensitive SVM classifiers based on various datasets are discussed. Section 5 discusses the obtained results. Section 6 provides conclusions and purposes for future research. The section Appendix A contains the names of concepts and their abbreviations in Table A1 and reference information on the datasets and the composition of their features in Table A2. The section Appendix B contains figures visualizing the graphical dependencies for the LFs used in the UMAP algorithm when the original dataset is embedded in two-dimensional space. Section Appendix C contains information on the results of statistical tests with the developed models.

## 2. Related Work

In a pilot study [13], the authors propose to evaluate the levels of proteins and mutations in extracellular deoxyribonucleic acid and apply this information in a nine-class CancerSEEK test based on LC. The dataset applied during the development of the CancerSEEK test is available in the supplementary materials to the paper [13] under the name aar3247_cohen_sm_tables-s1-s11.xlsx. It should be noted that new data are constantly being added to this dataset. A regularly updated version of this dataset is openly presented in the repository titled as Catalog of Somatic Mutations in Cancer (COSMIC) [54] under the title NIHMS982921-supplement-Tables_S1_to_S11.xlsx. In developing the CancerSEEK test, the authors use 1005 patterns of data from patients with clinically identified "Breast", "Colorectum", "Esophagus", "Liver", "Lung", "Ovary", "Pancreas" or "Stomach" ODs. In doing so, they propose to consider each individual's sex, levels of eight proteins and facts of mutations in 1933 various genomic locations. The authors believe that the facts of the gene mutations or the growth in the level of any of the eight proteins allows a pattern of data to be classified as a pattern with detectable OD. They use 10-fold cross-validation to calculate values of the classification quality metrics and show that for all types of ODs, the mean value of such metrics as sensitivity is about 70%; however, there are great differences by classes: the lowest value of this metric, equal to 33%, is found for the breast class, and the highest value of this metric, equal to 98%, is found for the ovarian class. Also, authors show that the value of this metric depends significantly on the stage of the disease: the lower the disease stage number, the lower the sensitivity metric value.

In [14], the authors develop a cost-sensitive three-class kNN classifier on the basis of imbalanced dataset patterns describing data only for 39 blood protein markers. Each pattern belongs to one of the following classes: "Normal", "Ovary" and "Liver". The authors refused the idea of developing a nine-class classifier, as was performed in [14], because of the poor separability of the patterns of the nine classes. They expand the original dataset containing 39 features with 2 new features formed using sample entropy and approximate entropy, and they use the extended 41-dimensional dataset to create a cost-sensitive kNN classifier. In this research, the dataset contained 897 patterns belonging to one of three classes in the such ratio as "Normal":"Ovary":"Liver" = 799:54:44. The values of metrics such as $Precision$, $Recall$, $MacroF_1 - score$ and $AUC$ were equal to 0.807, 0.833, 0.819 and 0.920, respectively. The overall accuracy of the classifier was equal to 0.952.

In study [20], the author of this paper develops kNN [14,39] and SVM [15,39] classifiers using such class balancing tools as SMOTE (Synthetic Minority Oversampling Technique) [31], Borderline SMOTE-1 [32], Borderline SMOTE-2 [32] and ADASYN (ADaptive SYNthetic sampling approach) [33], which allow us to restore class balance based on oversampling strategies. The creation of classifiers is performed on the basis of both the original dataset and the new datasets designed on the basis of the original dataset using different tools for forming new features. Thus, five methods of entropy calculation [40–45], such as approximate entropy (AE), sample entropy (SE), singular value decomposition entropy (SVDE), spectral entropy (SPE) and permutation entropy (PE); two methods of Hjorth parameter calculation [46,47], such as Hjorth complexity and Hjorth mobility (HC and HM); and three methods for calculating fractal dimensions [48,49], such as Higuchi fractal dimension (HFD), Katz fractal dimension (KFD) and Petrosian fractal dimension (PFD) were used to form potentially new features from the 39-dimensional patterns of the original dataset. Based on the results of the analysis of values of the mean and the standard deviation (SD), calculated for each class, as well as correlation assessments, the methods for calculating approximate entropy AE, Higuchi fractal dimension HFD and Katz fractal dimension KFD were selected for further use. In addition, the UMAP algorithm (Uniform Manifold Approximation and Projection) [50–53], which implements non-linear dimensionality reduction on the data by embedding them in a lower-dimensional space, was applied to form new features from 39-dimensional patterns of the original dataset. The dimensionality reduction was performed not only to 2-dimensional space (as it is usually carried out when solving data visualization problems in two-dimensional space), but also to other dimensions (from 3 to 38) in order to select the best dimensionality reduction option in the context of solving the problem of achieving higher data classification quality.

The research in reference [20] is the first attempt to create datasets using different tools for generating new features by recovering data that are hidden in 39 features of the original dataset and then selecting the best tools for generating new features. In particular, the selection of new feature formation tools was founded on the correlation analysis of potentially new features among themselves, as well as on their ability to separate patterns that belong to different classes, using the mean and the SD values of features for each class. This approach to the creation of a group of datasets describing the subject area was first applied in the field of medical diagnostics, including the identification of ODs using blood protein markers. These datasets were further subjected to balancing using SMOTE, Borderline SMOTE-1, Borderline SMOTE-2 and ADASYN oversampling tools followed by choosing the best of them. The best oversampling tools were used in developing kNN and SVM classifiers followed by choosing the best classifiers in terms of maximizing the mean value of the metric $MacroF_1 - score$. In this research, the best kNN classifier was created using the original dataset extended by the feature on the basis of approximate entropy, and the best SVM classifier was created using the original dataset extended by the feature on the basis of approximate entropy and 28 features on the basis of the UMAP algorithm. The mean values of the metric $MacroF_1 - score$ of kNN and SVM classifiers increased by 16.138% and 4.219%, respectively, in comparison with the mean values of this metric in the case when the original dataset was applied to create kNN and SVM classifiers. Thus, the mean values of the metric $MacroF_1 - score$ of the best kNN and SVM classifiers were 0.878 (with the SD value equal to 0.050) and 0.914 (with the SD value equal to 0.050), respectively. In addition, in [20], it was shown that it is promising to work with other considered tools of new feature formation, because their use in the datasets applied in the creation of classifiers also provided an improvement in data classification quality, although not as significant as the best classifiers mentioned above. Thus, the feasibility of using the proposed approach to form a group of datasets describing the subject area has been experimentally proven.

Despite the clearly significant results of the study performed in [20], we should note the drawbacks of the approach to solving the class imbalance problem, which implements work with oversampling tools that involve the synthesis of new patterns (perhaps never hypothetically possible). First, oversampling may lead to even more class mixing (in the case when classes are already poorly separable from each other) due to the impossibility of a priori accurate knowledge about the spatial geometry of data patterns because of the refusal (due to substantial time expenditures) to conduct additional research to study the spatial geometry of data patterns. However, in a sense, this disadvantage can be offset by evaluating the quality of the developed classifiers while screening out the unreliable ones. Second, oversampling always leads to an increase in the time cost of classifier development, both because of the need to synthesize new patterns and because of the need to develop classifiers on significantly larger datasets.

Due to this, it is reasonable to consider approaches to overcoming the class imbalance problem, in which algorithms that take into account the cost sensitivity of wrong decisions are implemented. CSL, and, in particular, cost-sensitive algorithms, have been addressed and applied in a large number of research works. For example, such an approach is used in the aforementioned work [14] related to the development of a three-class kNN classifier for diagnosing ODs.

In [36], the authors note that CSL accounts for the misclassification cost and possibly costs of other types as well. The purpose of CSL is to minimize the total cost. CSL handles various classification errors differently. In particular, the classification cost of marking the positive data pattern as negative may not be equal to the classification cost of marking the negative data pattern as positive. Non-cost-sensitive learning does not account for the misclassification cost.

In [55], the authors show that the class imbalance problem (CIP) is one of the serious ML problems. Training on very imbalanced data leads to the fact that classifiers will be overloaded with data patterns from majority classes; therefore, the false negative rate will be high. They note that many methods are currently known to address the class imbalance problem, including sampling methods and CSL methods, but these methods are usually applied independently of each other. The authors propose two empirical methods on the basis of sampling methods and CSL methods. The first method suggests to create SVM classifiers conjoining sampling methods with CSL methods. The second method suggests to use CSL methods with a locally optimized cost matrix. The authors show that the first method allows reducing the misclassification costs, while the second method allows improving the classifier performance.

In [56], the authors argue that CSL has made significant efforts to address the CIP, but in practice, it is almost impossible to assess the misclassification cost exactly. Additionally, they show that the classification quality depends on the used feature subsets from the dataset and the values of the classifier parameters. The authors embed evaluation metrics such as $AUC$ (Area Under Curve) and $G - mean$ (Geometric Mean) into the objective function to improve the classification quality of the cost-sensitive SVM classifier. They offer the method that tries to find the best combination of feature subsets, values of misclassification costs and values of the classifier parameters. The authors show that the proposed method is more efficient than commonly used sampling methods.

In [57], the authors propose robust cost-sensitive classifiers developed via the modification of the target functions of ML algorithms such as decision tree, random forest, extreme gradient boosting and logistic regression and apply them to efficiently predict medical diagnoses. Unlike sampling methods, the authors' approach does not change the original data distribution. The authors implement standard versions of the algorithms mentioned above and compare them with cost-sensitive versions. The cost-sensitive classifiers take into account the imbalanced class distribution during training, leading to more robust performance compared to classifiers on the basis of sampling methods.

In [58], the authors show that although methods such as cost-sensitive methods, sampling methods and ensemble learning methods can improve classification accuracy for minority class patterns, they are restricted by the problems of selection of cost parameter values and overfitting. The authors suggest a hybrid approach that includes data block construction, dimensionality reduction and ensemble creation with DL neural network classifiers. The effectiveness of the proposed hybrid approach is validated by experimental results using eight unbalanced datasets evaluated in terms of $Recall$, $G - mean$ and $AUC$.

In [59], the authors analyze CSL aspects and postulate its importance in medicine. They note that doctors are interested in models which can seek to minimize several types of healthcare-related costs such as the attribute cost (e.g., the diagnostic test cost) and the misclassification cost (e.g., the false negative test cost). They show that the diagnostic tests and the misclassification errors have high financial and human costs. The authors propose ideas for dealing with CSL and its medical applications and provide an overview of research on CSL, including approaches and methods for the creation and evaluation of cost-sensitive classifiers.

Currently, especially in 2022–2023, there is a significant increase in the number of studies addressing aspects of the use of ML and DL technologies in the context of solving the problem of early diagnosis of cancer [3–7,11–14,20,26,37,38,60–66]. At the same time, many of them are aimed at solving the problem of diagnosing ODs on the basis of biomarkers, including blood protein markers [13,14,20–24,27,28,37,38,60]. However, most research solves the problem of binary classification with the identification of one specific disease, for example, breast [22], liver [12] or lung [37] cancer. It is obvious that the problem of multiclass classification is, on the one hand, more complex, but, on the other hand, the data used in its solution contain more complex dependencies, the restoration of which should help the rapid diagnosis of oncological diseases [13,14,20].

## 3. Materials and Methods

### 3.1. Aspects of Developing Cost-Sensitive SVM Classifiers

Various ML and DL algorithms can be used in the development of data classifiers, including multi-class classifiers, such as kNN [14,20,39,67,68], SVM [15,20,39,69,70], LR (Logistic Regression) [13,71] and RF [16,72,73], as well as algorithms on the basis of certain neural network architectures [8–12]. In addition, an increase in the quality of data classification can be achieved by applying cascade algorithms and ensembles on the basis of ML and DL algorithms. In this case, when developing certain classifiers, it is possible to use the default values of the parameters of the eponymous algorithms or to adjust the values of these parameters by applying, for example, grid search algorithms or well-established population optimization algorithms [74].

It should be noted that there are no universal approaches to classifier development which would guarantee that the classifier developed with their application will ensure high-quality classification for every task. In particular, the quality of data classification will depend both on the key features of the mathematical apparatus used in classifier development and on the specifics of the dataset used in classifier development, including its balance. In some cases, the time spent on classifier development, as well as on making classification decisions, may be of fundamental importance. Obviously, preference should be given to classifiers that provide high data classification quality with minimum time cost. For example, the kNN algorithm can be characterized as an algorithm that requires minimal time to develop a classifier as well as to make classification decisions, unlike the RF algorithm. The SVM algorithm is generally less time-consuming than the RF algorithm but more time consuming than the kNN algorithm. It is for this reason that the author of the study in [20] used the kNN and the SVM algorithms in the creation of the eponymous classifiers.

In this study, only the SVM algorithm is considered, and a cost-sensitive one at that. This choice is made in connection with the previously stated goal of the study. The rejection of the kNN algorithm used by the author of this study along with the SVM algorithm in [20] can be justified by the fact that the SVM classifiers provided a higher quality of data classification in the earlier study. Therefore, it was decided to conduct an additional study to see if the SVM classifiers could be developed with even higher quality by utilizing cost-sensitive learning principles.

Let $U = \{\langle x_1, y_1\rangle, \ldots, \langle x_s, y_s\rangle\}$ be the dataset applied in the creation of the SVM classifier, where $x_i$ $(i = \overline{1,s})$ is the pattern described by $q$ features; $x_i \in X$; $X$ is the set of patterns; $y_i$ is the class labels of the pattern $x_i$ $(i = \overline{1,s})$; $y_i \in Y = \{1, \ldots, M\}$; $Y$ is the set of pattern class labels; $s$ is the number of patterns in the dataset $U$; $M$ is number of classes in the dataset $U$ [39].

Suppose that the SVM classifier is trained on $S$ patterns. Additionally, the SVM classifier is tested on $s - S$ patterns. In this case, the k-fold cross-validation procedure can be used to assess the quality of the SVM classifier.

The base SVM algorithm assumes that $M = 2$, that is, the classification is binary, and implements binary data classification by constructing a hyperplane that separates the classes [39,62]. In this case, each data pattern $x_i \in X$ corresponds to a class label $y_i \in Y = \{-1; +1\}$ $(i = \overline{1, s})$.

When developing a binary SVM classifier, we must solve the problem of constructing a hyperplane that separates the classes. This problem can be reduced, accordingly with the Kuhn–Tucker theorem, to a quadratic programming problem that contains only dual variables $\lambda_i$ $(i = \overline{1, S})$ [39,69]:

$$\begin{cases} \frac{1}{2} \cdot \sum\limits_{i=1}^{S} \sum\limits_{j=1}^{S} \lambda_i \cdot \lambda_j \cdot y_i \cdot y_j \cdot \kappa\left(x_i, x_j\right) - \sum\limits_{i=1}^{S} \lambda_i \to \min\limits_{\lambda}, \\ \sum\limits_{i=1}^{S} \lambda_i \cdot y_i = 0, \\ 0 \le \lambda_i \le C,\ i = \overline{1, S}, \end{cases} \tag{1}$$

where $\kappa\left(x_i, x_j\right)$ is the kernel function; $C$ $(C > 0)$ is the value of the regularization parameter.

The kernel function $\kappa\left(x_i, x_j\right)$ can be linear, radial basis, polynomial or sigmoid, the last three of which carry out a transition to a higher-dimensional space than the original feature space in order to provide better separability of pattern classes from each other.

In the proposed study, as in [20], the radial basis function (RBF) kernel is applied. Such function can be defined as $\kappa\left(x_i, x_j\right) = exp\left(-\frac{\left(x_i - x_j\right) \cdot \left(\left(x_i - x_j\right)\right)}{2 \cdot \sigma^2}\right)$, where $\sigma$ $(\sigma > 0)$ is the kernel function parameter.

When creating a binary SVM classifier with an a priori defined RBF kernel function $\kappa\left(x_i, x_j\right)$, we must define the value of the parameter $\sigma$ and the value of the regularization parameter $C$ [62], which assures the minimum classification error. The task of searching for optimal values of these parameters can be solved on the basis of grid search (GS) algorithms or population optimization algorithms.

Support vectors, which are patterns of the original dataset located near the hyperplane that separates classes, are defined as a result of solving problem (1). They present all information about the class separation rules. For the support vectors, the values of the dual variables $\lambda_i$ satisfy the condition $\lambda_i \neq 0$ [75].

The classification rule, according to which the membership class of pattern $x$ is determined, has the following form [39,69]:

$$F(x) = sign\left(\sum_{i=1}^{S} \lambda_i \cdot y_i \cdot \kappa(x_i, x) + b\right) \tag{2}$$

where $b = \omega \cdot x_i - y_i$; $\omega = \sum_{i=1}^{S} \lambda_i \cdot y_i \cdot x_i$.

In order to form classification decisions in the case of multiclass classification, i.e., when the number of classes is greater than 2, either the OvR (One-vs-Rest) strategy or the OvO (One-vs-One) strategy is applied [75].

Since the purpose of this research is to create a multi-class cost-sensitive SVM classifier, we can use different values of the regularization parameter $C_j$ $(j = \overline{1, M})$ for different classes by defining them as $C_j = weight_j \cdot C$, where $weight_j$ is the weight coefficient of the $j$-th class. Thus, $weight_j$ can be defined as $\frac{S}{M \cdot S_j}$, where $S_j$ is the number of patterns in the $j$-th class; $\sum_{j=1}^{M} S_j = S$. Furthermore, we can consider arbitrary combinations of weights for different classes, assigning large weights $weight_j$ $\left(j = \overline{1, M}\right)$ to small classes. As a result, it will be possible to select such combinations of weights that guarantee high-quality data classification according to some quality metric at not the highest costs (penalties) for classification errors.

When evaluating the quality of multiclass classification, it is reasonable to use metrics such as $Accuracy$, $MacroPrecision$, $MacroRecall$ and $MacroF_1 - score$ [20,76]. Such metrics as $MacroPrecision$, $MacroRecall$ and $MacroF_1 - score$ are useful and effective when developing classifiers using imbalanced datasets [76].

We can calculate $Accuracy$, $MacroPrecision$ and $MacroRecall$ as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{3}$$

$$MacroPrecision = \frac{1}{M} \sum\nolimits_{m=1}^{M} \frac{TP_m}{TP_m + FP_m}, \tag{4}$$

$$MacroRecall = \frac{1}{M} \sum\nolimits_{m=1}^{M} \frac{TP_m}{TP_m + FN_m}, \tag{5}$$

where $TP$ is the total number of true positive patterns; $TN$ is the total number of true negative patterns; $FP$ is the total number of false positive patterns; $FN$ is the total number of false negative patterns; $M$ is the total number of classes; $TP_m$ is the number of true positive patterns with the $m$-th class label; $FP_m$ is the number of false positive patterns with the $m$-th class label; $FN_m$ is the number of false negative patterns with the $m$-th class label.

The metric $MacroF_1 - score$ is based on metrics $MacroPrecision$ and $MacroRecall$. This metric allows us to simultaneously consider information about the precision and recall of the decisions formed by the classifier. It can be calculated as follows [20,76]:

$$MacroF_1 - score = 2 \cdot \frac{MacroPrecision \cdot MacroRecall}{MacroPrecision + MacroRecall}. \tag{6}$$

A high value of the metric $MacroF_1 - score$ means that the classifier has good performance on all the classes, whereas a low value of the metric $MacroF_1 - score$ means that the classes are poorly predictable [76].

In the proposed study, just like in [20], the SVM classifier having the maximum value of the metric $MacroF_1 - score$ is recognized as the best one.

### *3.2. Aspects of New Feature Generation*

#### 3.2.1. Generation of Features on the Basis of the UMAP Algorithm with Different Loss Functions

The UMAP algorithm carries out nonlinear dimensionality reduction by embedding patterns whose feature values are defined in high-dimensional space into lower-dimensional space. When performing such an embedding, the UMAP algorithm preserves local and global data structures that are defined in the high-dimensional space better [50,53] than similar algorithms, for example, the t-SNE (t-distributed stochastic neighbor embedding) algorithm [77].

Let $X = \{x_1, x_2, \ldots, x_s\}$ be some dataset. The UMAP algorithm embeds $q$-dimensional patterns $x_i$ $(i = \overline{1, s})$ described by $q$ features into $h$-dimensional space $(h \leq q)$.

At the first stage, the UMAP algorithm constructs a fuzzy weighted undirected graph. At the second stage, the UMAP algorithm optimizes the LF [50].

At the first stage, $k$ nearest neighbors [78] are found for each pattern $x_i$ $(i = \overline{1, s})$, and then distances $d_{il}$ $(l = \overline{1, k})$ to $k$ nearest neighbors are calculated based on some distance metric (e.g., on the basis of the Euclidean metric). Next, the values $\rho_i$ that define the distances to the nearest neighbor are found for each pattern $x_i$ $(i = \overline{1, s})$.

Then, a binary search is carried out to look for the values $\sigma_i$ that satisfy the condition:

$$\sum\nolimits_{l=1}^{k} e^{\left(\frac{\rho_i - d_{il}}{\sigma_i}\right)} = \log_2 k. \tag{7}$$

An array $\mathcal{M}_i$ is formed for each pattern $x_i$ $(i = \overline{1, s})$. The array's component $\mu_{ij}$ $(\mu_{ij} \in [0,\ 1])$ is a fuzzy number that defines how similar the $i$-th and the $j$-th patterns belonging to dataset $X$ are. If the patterns $x_i$ and $x_j$ are not neighbors, the component $\mu_{ij}$ of the array $\mathcal{M}_i$ is assumed to be 0. If the patterns $x_i$ and $x_j$ are neighbors, the component $\mu_{ij}$ of the array $\mathcal{M}_i$ is calculated as follows:

$$\mu_{ij} = e^{\left(\frac{\rho_i - d_{ij}}{\sigma_i}\right)}. \tag{8}$$

As a result, a weighted adjacency matrix $Matr \in \mathbb{R}^{s\times s}$ is formed in which the $i$-th row is defined on the basis of components from the array $\mathcal{M}_i$ $(i = \overline{1,s})$. The matrix $Matr$ is asymmetric. It defines the fuzzy weighted oriented graph that encodes the pairwise similarity of patterns $x_i$ $(i = \overline{1,s})$.

At the second stage, the UMAP algorithm carries out the symmetrization of the matrix $Matr$ on the basis of the probabilistic t-conorm:

$$\mu_{ij} \leftarrow \mu_{ij} + \mu_{ji} - \mu_{ij}\cdot\mu_{ji} \;\left(i = \overline{1,s}\,;\; j = \overline{1,s}\right), \tag{9}$$

where $\mu_{ll} = 0$ $(l = \overline{1,s})$.

Representations of $q$-dimensional patterns $x_i$ $(i = \overline{1,s})$ in $h$-dimensional space as $h$-dimensional patterns $y_i$ $(i = \overline{1,s})$ are computed using spectral embedding [50] $(h \leq q)$. As a result, a dataset $Y = \{y_1, y_2, \ldots, y_s\}$ is generated.

The base UMAP algorithm implements optimization using an LF [79] that is a weighted fuzzy cross-entropy with reduced repulsion:

$$L(Matr, Y) = \sum_{i=1}^{s}\sum_{j=1}^{s}\left(\mu_{ij}\ln\frac{\mu_{ij}}{\nu_{ij}} + \frac{\sum_{k=1}^{s}\mu_{ik}}{2s}\ln\left(\frac{1-\mu_{ij}}{1-\nu_{ij}}\right)\right), \tag{10}$$

where $Matr \in \mathbb{R}^{s\times s}$ is a symmetric adjacency matrix containing fuzzy values that determine the pairwise similarity of patterns of high dimensionality (i.e., of dimensionality $q$) from the dataset $X$; $Y \in \mathbb{R}^{s\times h}$ is the representation of $s$ patterns of low dimensionality (i.e., of dimensionality $h$); $\mu_{ij} \in [0,1]$ is the number that defines the fuzzy similarity of the $i$-th and the $j$-th patterns of high dimensionality that belong to the dataset $X$; $\nu_{ij} \in [0,1]$ is the number that defines the fuzzy similarity of the $i$-th and the $j$-th patterns of low dimensionality that belong to the dataset $Y$.

Pairwise similarity of the $i$-th and the $j$-th patterns of low dimensionality that belong to the dataset $Y$ is defined as:

$$\nu_{ij} = \left(1 + a d_{ij}^{2b}\right)^{-1}, \tag{11}$$

where $d_{ij}$ is the distance between the $i$-th and the $j$-th patterns of low dimensionality that belong to the dataset $Y$, calculated based on some distance metric (e.g., based on the Euclidean metric); $a$ and $b$ are coefficients fitted using the nonlinear least squares method (11) on the curve:

$$\psi_{ij} = \begin{cases} 1, & d_{ij} \leq d_{min} \\ e^{(d_{min} - d_{ij})}, & d_{ij} > d_{min} \end{cases}, \tag{12}$$

where $d_{ij}$ is the distance between the $i$-th and $j$-th patterns of low dimensionality that belong to the dataset $Y$; $d_{min}$ is the parameter $(d_{min} \in (0,\ 1])$ that influences the density of clusters created in the low-dimensional space during the optimization of the LF.

In the proposed study, as in the earlier study [20], the Euclidean metric is used as the distance metric.

The base UMAP algorithm applies the stochastic gradient descent (SGD) algorithm to optimize the LF (10) [50]. Pattern representations $y_i$ $(i = \overline{1,s})$ of low dimensionality from the dataset $Y$ are refined at each iteration of the SGD algorithm during minimization of the LF (10).

In addition to the LF (10), other LFs can be used.

We will additionally use the LFs described in [53] and presented below.

The LF based on fuzzy cross-entropy can be written as [53]

$$L_1(Matr, Y) = \sum_{i=1}^{s} \sum_{j=1}^{s} \left( \mu_{ij} \ln \frac{\mu_{ij}}{\nu_{ij}} + (1 - \mu_{ij}) \ln \left( \frac{1 - \mu_{ij}}{1 - \nu_{ij}} \right) \right). \tag{13}$$

The LF based on symmetric fuzzy cross-entropy can be written as [53]

$$L_2(Matr, Y) = \sum_{i=1}^{s} \sum_{j=1}^{s} \left( (\mu_{ij} - \nu_{ij}) \ln \left( \frac{\mu_{ij}(1 - \nu_{ij})}{\nu_{ij}(1 - \mu_{ij})} \right) \right). \tag{14}$$

The LF based on intuitionistic fuzzy cross-entropy can be written as [53]

$$L_3(Matr, Y) = \sum_{i=1}^{s} \sum_{j=1}^{s} \left( \mu_{ij} \ln \frac{\mu_{ij}}{\frac{1}{2}\mu_{ij} + \frac{1}{2}\nu_{ij}} + (1 - \mu_{ij}) \ln \left( \frac{1 - \mu_{ij}}{1 - \frac{1}{2}(\mu_{ij} + \nu_{ij})} \right) \right). \tag{15}$$

In the following, for convenience of presentation, we will refer to the LF (10) based on weighted fuzzy cross-entropy with reduced repulsion as $L_4$.

LFs (10) and (13)–(15) determine how the embedding of $q$-dimensional patterns $x_i$ ($i = \overline{1, s}$) into $h$-dimensional space ($h \leq q$) will look like.

It should be noted that in the author's study [50], function (10) is stated as an LF, but in fact, in the author's version of the UMAP algorithm software (v. 0.5.5) library in Python [70], the LF is not explicitly specified, and the optimization process itself when performing pattern embedding from a high-dimensional space to a low-dimensional space can be described precisely by the LF (10), as proven by the authors of study [79].

The main parameters involved in the work of the UMAP algorithm are the parameters $k$ and $d_{min}$.

$k$ is the number of nearest neighbors that are found for each pattern in the high-dimensional space. This parameter is responsible for controlling balance between local and global structures in the data. Low values of $k$ (*n_neighbors* in the software library [80]) will force the UMAP algorithm to pay attention to a very local structure. Large values of $k$ will force the UMAP algorithm to pay attention to larger neighborhoods of each pattern during assessment of the topological structure of the data, but in this case, it is possible to lose small detail structure wanting to cover more data. Traditionally, $k = 15$. In our research, we will enumerate values for parameter $k$ from the range [10, 20] with a step of 5.

$d_{min}$ is the threshold distance ($d_{min} \in (0, 1]$). This parameter influences the density of clusters created in the low-dimensional space during the optimization of the LF. It is responsible for controlling how densely UMAP algorithm packs points together. It defines the minimum distance between the patterns in the low-dimensional space. Low values of $d_{min}$ (*min_dist* in the software library [80]) will lead to clumpier embeddings. Larger values of $d_{min}$ will make it possible to avert from packing patterns together and focus on the preserving of the broad topological structure. Traditionally, $d_{min} = 0.1$. In our research, we will enumerate values for parameter $d_{min}$ from the range [0.1, 0.3] with a step of 0.1.

The UMAP algorithm also works with the parameter $h$, which determines the dimension of low-dimensional space. Traditionally, $h = 2$. In our research, we will enumerate values for parameter $h$ from the range [2, 38] with a step of 2 to investigate the performance of the UMAP algorithm when embedding data in spaces with dimensions other than 2.

### 3.2.2. Generation of Features on the Basis of the Approximate Entropy, the Higuchi Fractal Dimension and the Katz Fractal Dimension

The study carried out in [20] has shown the feasibility of using the methods of calculating the approximate entropy AE [45], Higuchi fractal dimensionality HFD [48] and Katz fractal dimensionality KFD [48] for the creation of new features. These are the methods that will be used in the proposed research. They are described in detail in [20].

### 3.2.3. Computational Complexity of Developing Classifiers

The assessment of the computational complexity of developing the proposed SVM classifiers can be performed as follows. Let $s$ be the number of patterns in the dataset and $q$ be the number of features.

The computational complexity of the standard SVM classifier training has both a quadratic component and a cubic one [81]. It increases at least like $s^2$ when the value of the regularization parameter $C$ is small and like $s^3$ when the value of the regularization parameter $C$ is large [81,82]. In general, the computational complexity of the standard SVM classifier training can be estimated as $O\left(qs^3\right)$.

The computational complexity of the UMAP algorithm realization can be estimated as $O\left(qs^2\right)$ [50,83].

The computational complexity of the approximate entropy calculation method can be estimated as $O\left(q^2\right)$ [84,85]. Because we calculate the approximate entropy for each pattern in the dataset, the total computational complexity can be estimated as $O\left(q^2 s\right)$.

The computational complexity of the methods for calculating the Higuchi fractal dimension and the Katz fractal dimension can be estimated as $O\left(q^2\right)$ [48]. Because we calculate the fractal dimensions for each pattern in the dataset, the total computational complexity can be estimated as $O\left(q^2 s\right)$.

Thus, the main computational complexity comes from training the SVM classifier. However, state-of-the-art SVM realizations typically have computational complexity that scales between $O(s)$ and $O\left(s^{2.3}\right)$, if we assume that the number of features $q$ is not large compared to the number patterns $s$ [86]. We can improve the computational complexity to $O(s)$ using parallel mixture [87]. Also, we can use such modern solvers as the Pegasos SVM [88] and the quantum SVM [89] to improve the computational complexity of standard SVM classifier training. Hence, this complexity can be reduced to quadratic (and even lower), both taking into account the specifics of the dataset and through the use of modern solvers. Though, these are only empirical observations and not theoretical guarantees [82].

Working with fractal dimensions does not make a significant contribution to the computational complexity of the developed classifiers. The computational complexity of the UMAP algorithm implementation and the computational complexity of the approximate entropy calculation method (taking into account working with all patterns in the dataset) are comparable.

Thus, the computational complexity of developing one SVM classifier in this research in the worst case is determined as $O\left(s^3\right)$, but in some cases it can be estimated as $O\left(s^2\right)$. When working with modern solvers, we can obtain computational complexity of $O\left(s^2\right)$ (if the UMAP algorithm is used when creating a dataset) and even less (if only entropy and fractal dimension calculation methods are used when creating a dataset).

## 4. Experimental Studies

All experimental studies were performed in the interactive cloud environment Google Colab. We used the Python 3.10 programming language for software development, since it allows us to work with a large number of different software libraries, including libraries that implement ML algorithms.

A three-class dataset on oncological diseases accommodating data on 39 serum protein markers (39 features) for 910 patterns was used in experimental studies. The following classes are considered: "Normal" (corresponding to the case when no OD was detected), "Liver" and "Ovary". Each protein marker in this dataset corresponds to a feature in the dataset. A list of serum protein markers can be found in [20]. This dataset is extracted from the original nine-class dataset, which contains 1817 patterns of classes such as "Normal", "Breast", "Colorectum", "Esophagus", "Liver", "Lung", "Ovary", "Pancreas" and "Stomach". Such a dataset with nine classes is publicly available in the COSMIC repository [54]. However, as shown in [14,20], the nine-class dataset is characterized by both poor separability of classes from each other and significant imbalance. In this regard, an attempt to work to restore class balance through the use of sampling strategies will not give the desired result. The use of oversampling strategies can lead to an even greater deterioration of the situation in the context of class separability as a result of even greater mixing of patterns of different classes, and the use of undersampling strategies can lead to the unreasonable deletion of representative patterns belonging to the "Normal" class, which is the biggest one, that is, the majority class. In [14,20], it is shown that the three-class dataset is also poorly balanced.

In [20], the visualization results using the UMAP algorithm in two-dimensional space for the original nine-class dataset and for the three-class dataset are presented. At the same time, it is assumed that the class separation in the three-class dataset should be better than in the original one. In this regard, it was the three-class dataset that was chosen for further research in [20]. It should be noted that the class ratio in a three-class dataset is as follows: "Normal":"Liver":"Ovary" = 812:44:54.

The UMAP algorithm has a library implementation in Python [80], proposed by the authors of this algorithm [50]. The authors argue that UMAP is a stochastic algorithm, so they use elements of randomness both to speed up the approximation steps and during the solution of optimization problems using the SGD algorithm.

Figure 1 shows the two-dimensional visualization for the three-class dataset using the library implementation of the UMAP algorithm with default parameter values of *n_neighbors* = 15, *min_dist* = 0.1, *metric* = *'euclidean'* and *random_state* = 42, where *n_neighbors* is the number of neighbors taken into account when assessing local and global properties of a diverse data structure [80]; *min_dist* is the parameter that determines the minimum distance at which data patterns can be located in low-dimensional space [80]; *random_state* is the parameter responsible for the initialization of UMAP algorithm and the reproducibility of results [80]; *metric* is the parameter that defines the metric used when calculating the distance between patterns [80]. Each two-dimensional point in Figure 1 corresponds to a 39-dimensional data pattern. The points corresponding to different classes are marked with different colors.

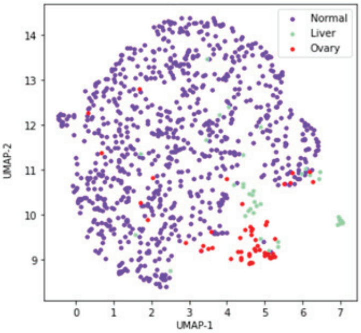


**Figure 1.** Two-dimensional visualization for three-class dataset using the library implementation of the UMAP algorithm [80] with the default parameter values (*n_neighbors* = 15, *min_dist* = 0.1, *metric* = *'euclidean'*, *random_state* = 42). The points corresponding to different classes are marked with different colors.

Since the dataset has three classes, the classification problem is multi-class, and its solution includes the development of a multi-class classifier.

### 4.1. Brief Background of This Study

Previously, in [20], approaches to the creation of three-class kNN and SVM classifiers based on datasets generated in various ways were investigated. The original dataset was tested for feature correlation. This test showed that there was no strong correlation between all the features. The correlation index values for all pairs of features, except one, turned out to be less than 0.6. Only for one pair of features with numbers 34 and 35 (sHER2/sEGFR2/sErbB2 (pg/mL) and sPECAM-1 (pg/mL)), the value of the correlation index was 0.604. Thus, the feasibility of using all features when performing further research was proven. Also, different options for extracting features from those in question were previously analyzed as well. In particular, five methods for calculating entropies such as approximate entropy (AE), sample entropy (SE), singular value decomposition entropy (SVDE), spectral entropy (SPE) and permutation entropy (PE); two methods for calculating Hjort parameters such as Hjorth complexity and Hjorth mobility (HC and HM); and three methods for calculating fractal dimensions such as Higuchi fractal dimension (HFD), Katz fractal dimension (KFD) and Petrossian fractal dimension (PFD) were considered. These methods were applied to the three-class dataset whose feature values were not subjected to preliminary scaling to [0, 1] to generate potential new features.

The values of entropies, Hjorth parameters and fractal dimensions were combined according to pattern class labels. Then, for each class, the mean value and the SD value of the potential new feature were calculated.

For each potential new feature, based on the results of the analysis of values of the mean and the SD for each class, the following conclusions were made. The biggest differences between the classes are observed for potentially new features based on the entropies AE and SE, as well as based on the fractal dimensions HFD and KFD. At the same time, the SD values for the aforementioned potentially new features are not large (and only for the feature based on the fractal dimension KFD they are slightly larger). It was these four potential new features that were selected for further consideration and examined for correlation with each other. We discovered that the potential new features based on the entropies AE and SE highly correlate with each other (the correlation score value is equal to 0.931). The feature based on the sample entropy SE was removed from consideration since it had a lesser correlation with the target feature that defines the pattern class label [20]. We also discovered that the potential new features based on the fractal dimensions HFD and KFD weakly correlate with each other (the correlation score value is equal to 0.141).

Thus, the preliminary analysis showed the feasibility of using the new feature on the basis of the approximate entropy AE as well as new features on the basis of the fractal dimensions HFD and KFD.

The additional experiments showed a slight excellence of the feature on the basis of the approximate entropy AE over the feature on the basis of the sample entropy SE in terms of maximizing the mean value of the metric $MacroF_1 - score$.

In addition, the library implementation of the UMAP algorithm [63] was used in [20] to embed a three-class 39-dimensional dataset into spaces of lower dimensions $h$ ($h = 2, \ldots, 38$) to form new features.

Thus, the new generated features belonged to one of the following three groups:

- Feature on the basis of the approximate entropy AE;
- Features on the basis of the fractal dimensions HFD and KFD;
- Features on the basis of the UMAP algorithm.

We created the new datasets either by appending various combinations of new features of the three groups mentioned above to the original dataset or by appending various combinations of new features of the first two groups mentioned above to the features of the third group formed on the basis of the UMAP algorithm. As a result, we developed classifiers of 12 types: we developed classifiers of 4 types once (since we did not use the UMAP algorithm in the creation of the corresponding datasets) and classifiers of another 8 types over and over again (since we used the UMAP algorithm in the creation of the corresponding datasets for $h = 2, \ldots, 38$, where $h$ is the dimension of the space into which the original dataset is embedded).

Figure 2 schematically shows all 12 ways for creating datasets. Table A2 in Appendix A contains information on the composition of features for all 12 datasets.

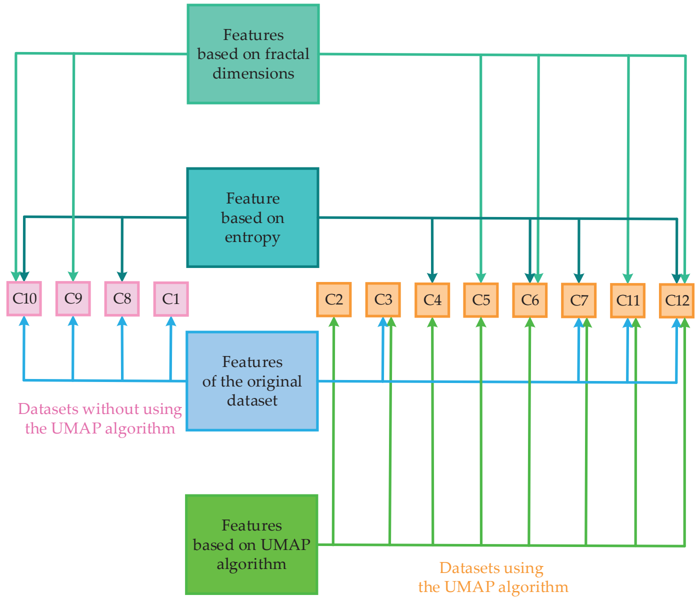

Features based on fractal dimensions
Feature based on entropy
C10
C9
C8
C1
C2
C3
C4
C5
C6
C7
C11
C12
Features of the original dataset
Datasets without using the UMAP algorithm
Features based on UMAP algorithm
Datasets using the UMAP algorithm

**Figure 2.** Scheme of creating of all 12 datasets applied in the research for the development of classifiers.

The developed classifiers subsequently have the same names as the datasets on the basis of which they are developed. Thus, the designation C7 in Figure 2 corresponds to a dataset obtained by supplementation of the original three-class dataset with the feature on the basis of the approximate entropy AE, as well as the features on the basis of the UMAP algorithm for a certain dimension $h$ ($h = 2,\ldots, 38$) of space in which the original three-class dataset is embedded. Based on dataset C7 for a certain space dimension $h$, homonymous classifier C7 is developed. All 12 considered three-class datasets are unbalanced, since the three-class dataset C1 extracted from the original nine-class dataset is unbalanced.

It should be noted that all the principles formulated above for forming new groups of datasets can be applied when developing classifiers on the basis of any ML algorithms, since they only have an impact on the stage of preparing datasets applied in the creation of classifiers.

In [20], the CIP was overcome by applying oversampling algorithms such as SMOTE [31], Borderline SMOTE-1 [32], Borderline SMOTE-2 [32] and ADASYN [33]. Then, the best oversampling algorithms in terms of maximizing the mean value of the metric $MacroF_1 - score$ were selected. In this case, all datasets were either immediately used to develop classifiers or were first subjected to oversampling based on SMOTE, Borderline SMOTE-1, Borderline SMOTE-2 and ADASYN algorithms.

We found that Borderline SMOTE-1 is the best oversampling algorithm for developing kNN classifiers. Also, we found that the base SMOTE algorithm is the best oversampling algorithm for developing SVM classifiers.

The choice of these different oversampling algorithms was justified by the fact that these particular algorithms made it possible to provide the best classification quality assessed using the metric $MacroF_1 - score$ for the kNN and SVM classifiers.

The following conclusions were made based on the experimental results:

- SVM classifiers outperform kNN classifiers, both in the absence of oversampling and in the case of its use, in terms of maximizing the mean value of the metric $MacroF_1 - score$.

- In the case of oversampling, SVM classifiers provided the best classification quality assessed using the metric $MacroF_1 - score$, but the time spent on their development increased significantly.

In this regard, it was decided to continue researching the capabilities of SVM classifiers. In order to solve the CIP, it was decided to use the cost-sensitive algorithms instead of oversampling algorithms. If a positive result is obtained from experiments using cost-sensitive algorithms, we will be able to significantly reduce the time spent on creating SVM classifiers and, possibly, improve the data classification quality.

The method for finding the best SVM classifiers is described in detail in [20]. It involves searching through a grid of parameter values that provide the maximum value of the metric $MacroF_1 - score$.

In particular, we applied a grid search to find the values of parameters such as $C$ and $gamma$, that, respectively, determine the regularization parameter and the parameter of the RBF kernel. The values of parameter $C$ and parameter $gamma$ varied in the range [0.4, 2] with a step of 0.1. We used the default values from the software implementation of the SVM algorithm in the scikit-learn library of Python as the values of the remaining parameters.

We used the metric $MacroF_1 - score$ as the main classification quality metric. This was performed to minimize the negative impact of the existing class imbalance in the original dataset C1 on the classification performance.

We applied a grid search to find the optimal values of the SVM classifier parameters [20] using stratified 10-fold cross-validation [90,91] with three-time repetition and the multi-class OvO strategy. We calculated the mean value of the metric $MacroF_1 - score$ with the corresponding SD value. Also, we calculated the mean values of the metrics $Accuracy$, $MacroPrecision$ and $MacroRecall$ with the corresponding SD values for the best classifiers. In addition, we determined the hyperparameter values for the best classifiers.

Table 1 demonstrates, for reference, the mean values of the metric $MacroF_1 - score$ and the corresponding SD values for the best classifiers of 12 types created without the application of oversampling algorithms [20]. Based on the information in Table 1, we can select best potential types of classifiers, which, in the future, first of all, should be paid attention to when developing classifiers based on CSL principles.

**Table 1.** Characteristic values of the best classifiers of different types created without the application of oversampling algorithms and cost-sensitive algorithms on the basis of the metric $MacroF_1 - score$.

| **Characteristic** | **Classifiers** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **C1** | **C2** | **C3** | **C4** | **C5** | **C6** | **C7** | **C8** | **C9** | **C10** | **C11** | **C12** |
| Mean | **0.877** | 0.837 | **0.864** | 0.799 | 0.840 | 0.846 | **0.873** | **0.885** | **0.878** | **0.880** | **0.866** | **0.876** |
| Standard deviation | 0.078 | 0.095 | 0.074 | 0.101 | 0.090 | 0.090 | 0.075 | 0.079 | 0.078 | 0.074 | 0.078 | 0.080 |

Values greater than 0.850 are highlighted in bold.

As we can see, classifiers C3, C7, C8, C9, C10, C11 and C12 have the mean values of the metric $MacroF_1 - score$ exceeding 0.850. The mean values of the metric $MacroF_1 - score$ of these classifiers are highlighted in bold in Table 1. In the future, it will be interesting to evaluate the behavior of classifiers of these types if CSL principles will be applied for their development.

We found that that even the best SVM classifiers do not have good quality metric values due to class imbalance: the values of the metric $MacroF_1 - score$ are small, while all classifiers make many errors on patterns belonging to minority classes [20].

### *4.2. Experiments to Implement the Concept of This Study*

When performing experiments in this study, it was decided to:

- Apply the principles of cost-sensitive algorithms for developing classifiers;
- Explore the possibility of generating new features using the UMAP algorithm, which uses five variants of the LF.

We plan to select the parameter values of the UMAP algorithm on the basis of the GS algorithm.

It should be noted that, as in [20], methods for calculating the approximate entropy AE, the Higuchi fractal dimension HFD and the Katz fractal dimension KFD will be used to generate new features.

We plan to use the following variants for the LF:

- Variant available in the library implementation in Python [80];
- Four variants described in [53] and available in the library implementation in Python [92].

Unfortunately, the LF in the library implementation in Python [80] is set implicitly, although the authors really use an SGD algorithm to solve an optimization problem when embedding the dataset from the high-dimensional space into the low-dimensional space; that is, in fact, the LF (13) declared by the authors in [50] is not used. The authors of study [79] tried to explicitly write down the formula for the LF, relying on the library implementation in Python [80]. They concluded that the LF resembles formula (10). In this case, with some error, they reproduced the work of the UMAP algorithm in the library implementation in Python [80]. In what follows, we will call the implicit LF used in the library implementation [80] as $L_0$ and the LF (10) as $L_4$.

The LFs in [92] are defined in accordance with (10) and (13)–(15). In this case, the optimization problem is solved using full gradient descent (FGD). The parameter value of *random_state* equal to 42 is used only when initializing the embedding of patterns into low-dimensional space.

We suggest using the following methodology when performing experiments.

1. Create datasets C1–C12 in accordance with the scheme presented in Figure 2, for a fixed combination of the parameter values (*h*, LF type, *n_neighbors, min_dist*), where *h* is the dimension of the low-dimensional space (*h* is selected from the range [2, 38] with a step of 2); the LF type is defined as one of the types in the list [$L_0$, $L_1$, $L_2$, $L_3$, $L_4$]; *n_neighbors* is the number of nearest neighbors of the data pattern in the high-dimensional space (*n_neighbors* is selected from the range [10, 20] with a step of 5); *min_dist* is the threshold distance that influences the density of clusters created in the low-dimensional space (*min_dist* is selected from the range [0.1, 0.3] with a step of 0.1). During the experiments, a walk through the grid is carried out for combinations of parameter values (*h*, LF type, *n_neighbors, min_dist*).
2. Develop cost-sensitive SVM classifiers based on datasets C1–C12 for a fixed combination of penalty values ($weight_1$, $weight_2$, $weight_3$) for the misclassification of patterns of different classes.
3. Assess the data classification quality using the cost-sensitive SVM classifiers based on datasets C1–C12 on the basis of stratified 10-fold cross-validation. Select classifiers that maximize the mean value of the metric $MacroF_1 - score$ (6). Analyze of the mean values of the metrics $Accuracy$ (3), $MacroPrecision$ (4) and $MacroRecall$ (5). Assess the time spent training and testing the cost-sensitive SVM classifiers.

At step 1 of the methodology, we prepare different datasets that vary from each other in the composition of features. Additional information on the formation of such datasets is also reflected in [20].

At step 2 of the methodology, we can use the options for setting penalties for classification errors proposed in Section 3.1. Selecting fine values involves a certain amount of time. It can be performed by walking through the grid or by analyzing the most intuitively selected combinations of penalty values based on the following considerations: the smaller the number of patterns in a class, the greater the penalty for misclassifying patterns of this class. At the same time, considering various combinations of penalties and maximizing the mean value of the metric $MacroF_1 - score$, it is advisable to choose combinations with smaller penalties with similar mean values of the metric $MacroF_1 - score$. For the three-class dataset under study, we preferred the combination of penalties in the form ($weight_1$, $weight_2$, $weight_3$) = (1, 10, 10).

At step 3 of the methodology, it makes sense to pay special attention to the mean values of the metrics $MacroF_1 - score$ and $MacroRecall$: we maximize the first in order to select the best classifier, and the second allows us to assess whether there is an increase in the data classification quality in terms of sensitivity to identifying truly existing diseases. In addition, with similar values of the data classification quality for SVM classifiers, it makes sense to give preference to those whose training and testing time will be lesser (paying attention to time costs makes sense, for example, when comparing the cost-sensitive classifiers and classifiers created using oversampling technologies). It should be noted that when choosing classifiers for further use, it is advisable to pay attention to the results of certain statistical tests.

Figure 1 shows the two-dimensional visualization of the three-class dataset using the UMAP algorithm, for which there is a library implementation in Python [80], with the values of parameters *n_neighbors, min_dist, random_state* and *metric* set as default. These are the values of the parameters that were used in [20]. It should be noted that, unfortunately, this library implementation does not provide an explicit output of the values of the LF, and the results of the embedding, even with fixed values of the parameters *n_neighbors* and *min_dist*, depend on the initialization of the UMAP algorithm using the parameter *random_state.* In addition, the final results of the UMAP algorithm's application depend on the version of the library implementation, which is in a state of constant improvement, as well as on the version of the libraries associated with it, including the Numba library [93], which is responsible for parallelizing calculations and allows for speeding up the operation of the UMAP algorithm.

In the proposed study for the UMAP algorithm with LFs $L_0$, $L_1$, $L_2$, $L_3$ and $L_4$, the GS was performed for the values of parameters *n_neighbors* and *min_dist,* providing development of the best cost-sensitive SVM classifiers in terms of maximizing the mean value of the metric $MacroF_1 - score$. In our research, we enumerate values for the parameter *n_neighbors* in the range [10, 20] with a step of 5, and enumerate values for the parameter *min_dist* in the range [0.1, 0.3] with a step of 0.1. In addition, we enumerate values for the parameter that defines the space dimension $h$ in the range [2, 38] with a step of 2.

The tuple of parameter values (*n_neighbors*, *min_dist*) which allowed us to obtain a cost-sensitive SVM classifier of a specific type with the maximum possible mean value of the metric $MacroF_1 - score$ was considered the best.

It should be emphasized that the creation of cost-sensitive SVM classifiers was carried out. At the same time, different ratios of penalties for classification errors were considered for different classes, but in the end, a ratio of the form 1:10:10 was chosen, respectively, for the classes "Normal", "Liver" and "Ovary".

With this ratio of penalties for classification errors, the best cost-sensitive classifier C1 has a mean value of the metric $MacroF_1 - score$ equal to 0.907 (with the SD value equal to equal to 0.052). In what follows, we will consider this mean value of the metric $MacroF_1 - score$ to be the base (threshold) value. It is with this value that we will compare the mean values of the metric $MacroF_1 - score$ of cost-sensitive SVM classifiers of other types in order to select the truly best one, that is, superior to classifier C1, developed based on a cost-sensitive SVM algorithm.

Table 2 shows the main characteristics of the base classifier C1 [20], created on the basis of the original three-class 39-dimensional dataset not subjected to any manipulation, balanced classifier C1 using SMOTE algorithm [20], balanced classifier C7 using the SMOTE algorithm (at $h = 28$) [20] and the base cost-sensitive classifier C1, developed on the basis of the original three-class 39-dimensional dataset, which was not subjected to any manipulation.

**Table 2.** Characteristics of such SVM classifiers as the base C1, balanced C1, balanced C7 (at $h = 28$) and the base cost-sensitive C1.

| **Characteristic** | **Classifiers** | | | |
|---|---|---|---|---|
| | **Base C1 [20]** | **Balanced C1 Using SMOTE Algorithm [20]** | **Balanced C7 (at $h$=28) Using SMOTE Algorithm [20]** | **Base Cost-Sensitive C1** |
| Number of features in the dataset | 39 | 39 | 68 | 39 |
| *gamma* | 1.2 | 1 | 0.7 | 0.8 |
| $C$ | 2.0 | 0.4 | 0.7 | 0.4 |
| $MacroF_1 - score$ (mean/SD) | 0.877/0.078 | 0.910/0.064 | 0.914/0.050 | 0.907/0.052 |
| *Accuracy* (mean/SD) | 0.973/0.015 | 0.977/0.015 | 0.978/0.012 | 0.977/0.013 |
| *MacroRecall* (mean/SD) | 0.843/0.088 | 0.907/0.081 | 0.907/0.065 | 0.907/0.066 |
| *MacroPrecision* (mean/SD) | 0.950/0.053 | 0.929/0.058 | 0.937/0.048 | 0.923/0.053 |
| Training time (mean/SD), s. | 0.123/0.008 | 0.886/0.214 | 0.489/0.021 | 0.169/0.023 |
| Quality metrics calculation time (mean/SD), s. | 0.007/0.001 | 0.013/0.004 | 0.008/0.001 | 0.011/0.003 |

It should be noted that the best kNN and SVM classifiers created on the basis of oversampling strategies and presented in [20] outperformed the cost-sensitive kNN classifier developed in [14] in terms of maximizing the mean value of the metric $MacroF_1 - score$. In addition, the best SVM classifier [20] outperformed the kNN classifier [20] in terms of the same indicator. In this regard, the main attention in the proposed study is paid specifically to the aspects of creating cost-sensitive SVM classifiers that have a higher data classification quality than previously developed classifiers.

As can be seen from Table 2, the base cost-sensitive classifier C1 has a higher mean value of the metric $MacroF_1 - score$ than the base classifier C1. However, this value in case of the cost-sensitive classifier C1 is less than that of classifier C1 balanced using the SMOTE algorithm [20] and classifier C7 balanced using the SMOTE algorithm (at $h = 28$) [20]. At the same time, the time spent on developing and testing base cost-sensitive classifier C1 is comparable to a similar time for the base classifier C1.

Figure 3a–e show the two-dimensional visualization of the three-class dataset using the UMAP algorithm with LFs $L_0$, $L_1$, $L_2$, $L_3$ and $L_4$ for tuples of parameter values (*n_neighbors, min_dist*) providing development of the best cost-sensitive SVM classifiers, called C3, for $h$ = 2 in terms of maximizing the mean value of the metric $MacroF_1 - score$. It should be said that, generally speaking, the best SVM classifiers, called C3, can be obtained in spaces of dimension $h$ other than 2. Each two-dimensional point in Figure 3a–e corresponds to a 39-dimensional data pattern. The points corresponding to different classes are marked with different colors.

Figure A1a–d, from Appendix B, show graphical dependencies for LF $L_1$, $L_2$, $L_3$ and $L_4$, obtained by constructing embeddings of the original 39-dimensional three-class dataset into the two-dimensional space, presented in Figure 3b–e. It should be noted that the ability to analyze and display the values of the LF $L_0$ is not provided by the library implementation [80]; therefore, graphical dependency is not shown.

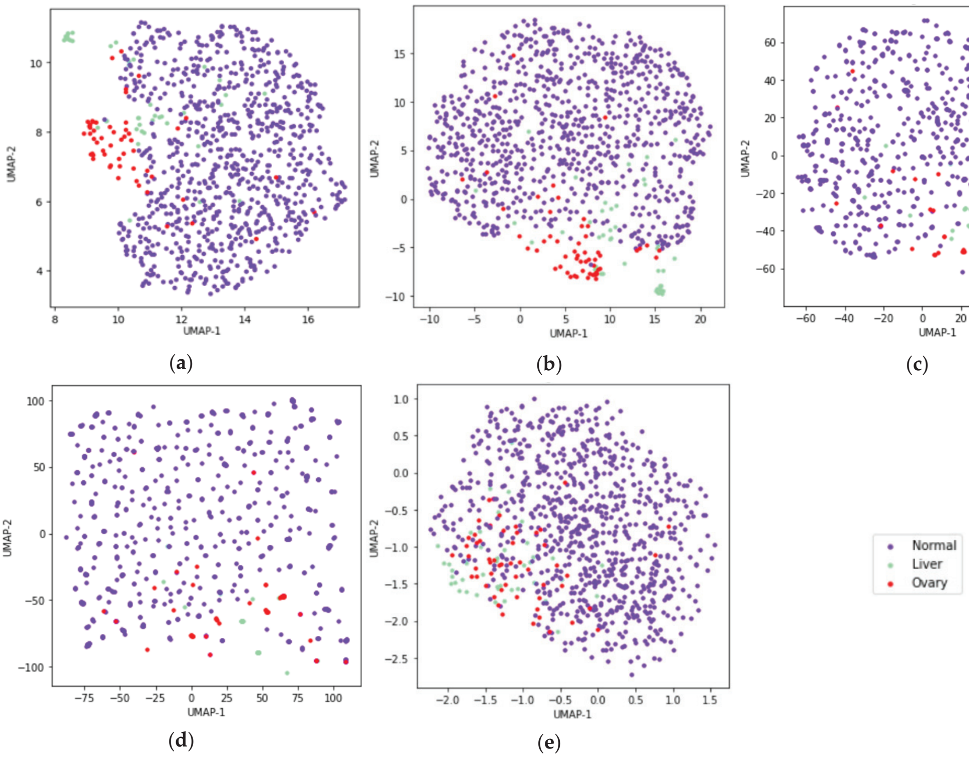


**Figure 3.** Two-dimensional visualization of the 39-dimensional three-class dataset using the UMAP algorithm with various LFs with parameter values *n_neighbors* and *min_dist*, ensuring the creation of the best cost-sensitive classifiers C3 (at $h = 2$) in the terms of maximizing the mean value of the metric $MacroF_1 - score$: (**a**) $L_0$: implicitly set LF (*n_neighbors* = 10; *min_dist* = 0.3; $MacroF_1 - score$ : mean = 0.924; SD = 0.047); (**b**) $L_1$: LF based on fuzzy cross-entropy with FGD (*n_neighbors* = 15; *min_dist* = 0.1; $MacroF_1 - score$ : mean = 0.918; SD = 0.049); (**c**) $L_2$: LF based on symmetric fuzzy cross-entropy with FGD (*n_neighbors* = 15; *min_dist* = 0.1; $MacroF_1 - score$ : mean = 0.916; SD = 0.047); (**d**) $L_3$: LF based on intuitionistic fuzzy cross-entropy with FGD (*n_neighbors* = 15; *min_dist* = 0.3; $MacroF_1 - score$ : mean = 0.914; SD = 0.055); (**e**) $L_4$: LF based on weighted fuzzy cross-entropy with FGD (*n_neighbors* = 20; *min_dist* = 0.2; $MacroF_1 - score$ : mean = 0.915; SD = 0.044). The points corresponding to different classes are marked with different colors.

The best cost-sensitive classifier C3 on the basis of the library implementation [80] has parameter values *n_neighbors* and *min_dist (*Figure 3a) different from the default values (Figure 1). At the same time, the mean values of the metric $MacroF_1 - score$ for the best cost-sensitive classifier C3 and the cost-sensitive classifier C3 with the default values of the parameters *n_neighbors* and *min_dist* are equal to 0.924 (with the SD value equal to 0.047) and 0.913 (with the SD value equal to 0.045), respectively, i.e., the best cost-sensitive classifier C3 outperformed the cost-sensitive classifier C3 with default parameter values by 1.205%. It should be noted that in both cases, the used default value of the parameters *random_state* was equal to 42. We decided to check how choosing the value of parameter *random_state* affects the final quality of the cost-sensitive classifier C3. In order to do this, we varied the values of the parameter *random_state* from 1 to 50 with a step of 1. Unfortunately, it turned out that only in 11 cases out of 50, which is 22%, we were able to receive a mean values of the metric $MacroF_1 - score$ of no less than 0.907. Moreover, only in 7 cases out of 50, which is only 14%, we were able to receive mean values of the metric $MacroF_1 - score$ of no less than 0.908: three of them turned out to be equal to 0.908, one of them turned out to be equal to 0.909, two of them turned out to be equal to 0.911 and only one of these, which is only 2%, turned out to be 0.924. In this regard, the following conclusions can be deduced: Indeed, we can reduce the dimension of space even to 2, obtaining in some cases mean values of the metric $MacroF_1 - score$ of no less than 0.907, which is not bad, because in this case, it is possible to reduce the dimension of space from 39 to 2. However, such cases occur rarely, and searching for them is associated with the additional time expenditures. The case where the mean value of the metric $MacroF_1 - score$ is 0.924 turned out to be the only one. The remaining cases in which it was possible to obtain mean values of the metric $MacroF_1 - score$ of no less than 0.907 did not occur often, and the corresponding mean values of the metric $MacroF_1 - score$ turned out to be significantly less than the found maximum mean value of 0.924. So, obviously, one should not expect that simply iterating over the values of the parameter *random_state* will quickly lead us to the desired result, namely, to mean values of the $MacroF_1 - score$ of no less than 0.924. We can say that the use of the SGD algorithm in the problem under consideration, although it makes it possible to reduce the time spent searching for a solution, in most cases leads to finding only certain local extrema. Therefore, to find better solutions, i.e., solutions close to the global extremum, repeated runs of the SGD algorithm are required. So, choosing a different value for the parameter *random_state* other than 42 does not guarantee that we will obtain a mean value of the metric $MacroF_1 - score$ for cost-sensitive classifier C3 no worse than the mean value of the metric $MacroF_1 - score$ for cost-sensitive classifier C1. We will look for classifiers that are no worse in terms of the mean value of the metric $MacroF_1 - score$ than the cost-sensitive classifier C1, assuming that, perhaps, with equal mean values of the $MacroF_1 - score$ we will be able to decrease the number of features in the datasets on the basis of which cost-sensitive classifiers will be developed. A smaller number of features in the dataset that is applied for a certain classifier development, with all other characteristics being equal for the compared classifiers, can be considered a positive property of this classifier.

As can be seen from Figure 3a–e, only four LFs, named $L_0$, $L_1$, $L_2$ and $L_4$, provide good separation of classes in the two-dimensional space. They are used in the formation of two-dimensional datasets, a visualization of which is presented in Figure 3a–c,e.

However, for the purity of the experiment and the formation of convincing conclusions, we conducted experiments with all five LFs, without abandoning the LF $L_3$, for all values of the space dimension $h$ indicated above, i.e., for values $h$ from 2 to 38 with a step of 2.

It should be noted, based on the results obtained when developing cost-sensitive classifiers C3 (at $h = 2$) based on different LFs, that the best cost-sensitive classifiers C3 have parameter values recommended for use by default in the library implementation of the UMAP algorithm in only two out of five cases [80]. It can be assumed that the best cost-sensitive SVM classifiers on the basis of the UMAP algorithm result in spaces with dimension $h$ different from 2 and may have parameter values of *n_neighbors* and *min_dist* different from those that are recommended to be used by default in the library implementation of the UMAP algorithm [80].

Next, the responses to two research questions (RQs) will be presented.

Question 1. Which types of cost-sensitive SVM classifiers are the best in terms of maximizing the mean value of the metric $MacroF_1 - score$ when using LFs $L_0$, $L_1$, $L_2$, $L_3$ and $L_4$ for different combinations of values of parameters *n_neighbors* (in the range [10, 20] with a step of 5) and *min_dist* (in the range [0.1, 0.3] with a step of 0.1), as well as different values of the space dimension $h$ (from 2 to 38 with a step of 2)?

Question 2. How often do cost-sensitive SVM classifiers of each type have mean values of the metric $MacroF_1 - score$ no lower than the base (threshold) mean value of the metric $MacroF_1 - score$ inherent to the base cost-sensitive classifier C1 and equal to 0.907, when using different LFs for different combinations of values of parameters *n_neighbors* (from 10 to 20 with a step of 5) and *min_dist* (from 0.1 to 0.3 with a step of 0.1), as well as different values of the space dimension $h$ (in the range [2, 38] with a step of 2)?

### 4.2.1. Identifying the Best Cost-Sensitive SVM Classifiers and Analysis of Their Characteristics

In order to answer Question 1, Table 3 shows the names of the best cost-sensitive SVM classifiers and their characteristics, such as the mean value and the SD value of the metric $MacroF_1 - score$, as well as the dimension of space $h$ (in the case of using the UMAP algorithm) in the format classifier name/mean/standard deviation/space dimension.

**Table 3.** Names of the best cost-sensitive SVM classifiers and their characteristics.

| **Tuple of Parameter Values (*n_neighbors, min_dist*)** | **Loss Functions** | | | | |
|---|---|---|---|---|---|
| | $L_0$ | $L_1$ | $L_2$ | $L_3$ | $L_4$ |
| (10, 0.1) | C8/0.918/0.047/- | C8/0.918/0.047/- | C8/0.918/0.047/- | C8/0.918/0.047/- | C8/0.918/0.047/- |
| (10, 0.2) | C8/0.918/0.047/- | C8/0.918/0.047/- | C8/0.918/0.047/- | C8/0.918/0.047/- | C8/0.918/0.047/- |
| (10, 0.3) | **C3/0.924/0.047/2** * | C8/0.918/0.047/- | C8/0.918/0.047/- | C8/0.918/0.047/- | C8/0.918/0.047/- |
| (15, 0.1) | C8/0.918/0.047/- | C7/0.919/0.046/22 | C7/0.917/0.048/12 | C8/0.918/0.047/- | C8/0.918/0.047/- |
| (15, 0.2) | C8/0.918/0.047/- | C7/0.919/0.042/8 | C7/0.919/0.042/6 | C8/0.918/0.047/- | C7/0.918/0.046/6 |
| (15, 0.3) | C8/0.918/0.047/- | C7/0.920/0.044/28 | *C7/0.921/0.048/26* | C8/0.918/0.047/- | C8/0.918/0.047/- |
| (20, 0.1) | C8/0.918/0.047/- | C8/0.918/0.047/- | C8/0.918/0.047/- | C3/0.918/0.045/12 | C7/0.920/0.047/34 |
| (20, 0.2) | C8/0.918/0.047/- | C7/0.920/0.052/30 | C8/0.918/0.047/- | C8/0.918/0.047/- | C11/0.919/0.054/8 |
| (20, 0.3) | C8/0.918/0.047/- | *C7/0.921/0.045/28* *** | ***C7/0.923/0.047/28*** ** | C8/0.918/0.047/- | *C3/0.921/0.045/6* |

* the classifier, which took first place in the ranking in terms of maximizing the mean value of the metric $MacroF_1 - score$, is highlighted in bold; ** the classifier, which took second place in the ranking in terms of maximizing the mean value of the metric $MacroF_1 - score$, is highlighted in bold italic font; *** the classifiers, sharing third place in the ranking in terms of maximizing the mean value of the metric $MacroF_1 - score$, are highlighted in italics.

We can see from Table 3 that the best cost-sensitive SVM classifiers for the considered tuples of the parameter values (*n_neighbors, min_dist*) turned out to be cost-sensitive classifiers C3, C7, C8 and C11. At the same time, cost-sensitive classifiers C7 and C8 most often took the lead, 11 and 30 times, respectively. Cost-sensitive classifiers C3 and C11 were leaders three times and one time, respectively.

In Table 3, the cost-sensitive SVM classifier, which is the absolute leader in terms of maximizing the mean value of the metric $MacroF_1 - score$, is highlighted in bold. This is the cost-sensitive classifier C3 with values of the mean and the SD of the metric $MacroF_1 - score$ equal to 0.924 and 0.047, respectively. The dataset used in the development of this classifier was created by supplementing the original three-class 39-dimensional dataset with new features generated on the basis of the library implementation of the UMAP algorithm [80] (i.e., using the LF $L_0$). Based on the results of previously performed experiments with the LF $L_0$, we can conclude that its use when implementing the UMAP algorithm does not give the expected effect when working with spaces of dimensions $h$ different from 2, with a fixed value of the parameter *random_state*, which affects the results of stochastic gradient descent. By default, in the library implementation of the UMAP algorithm [80], the value of the parameter *random_state* is 42. Previously performed experiments with different values of the parameter *random_state* at $h = 2$ did not improve the quality of cost-sensitive classifiers C3, so the feasibility of similar experiments with other values of the space dimension $h$ is questionable, but it is related to large time expenditures.

In Table 3, the cost-sensitive SVM classifier, which took second place in the ranking in terms of maximizing the mean value of the metric $MacroF_1 - score$, is highlighted in bold italic font. This is the cost-sensitive classifier C7 with values of the mean value and the SD of the metric $MacroF_1 - score$ equal to 0.923 and 0.047, respectively. This classifier is built on the basis of the original 39-dimensional dataset, supplemented with a feature on the basis of the approximate entropy AE as well as features on the basis of the UMAP algorithm at $h = 28$ using LF $L_2$.

In Table 3, three cost-sensitive SVM classifiers, sharing third place in the ranking in terms of maximizing the mean value of the metric $MacroF_1 - score$, are highlighted in italics. These are such cost-sensitive SVM classifiers as the following:

- Classifier C7, with values of the mean value and the SD of the metric $MacroF_1 - score$ equal to 0.921 and 0.045, respectively (this classifier is built on the basis of the original 39-dimensional dataset, supplemented with a feature on the basis of the approximate entropy AE, as well as features on the basis of the UMAP algorithm at $h = 28$ using the LF $L_1$);
- Classifier C7, with values of the mean value and the SD of the metric $MacroF_1 - score$ equal to 0.921 and 0.048, respectively (this classifier is built on the basis of the original 39-dimensional dataset, supplemented with a feature on the basis of the approximate entropy AE, as well as features on the basis of the UMAP algorithm at $h = 26$ using the LF $L_2$);
- Classifier C3, with values of the mean value and the SD of the metric $MacroF_1 - score$ equal to 0.921 and 0.045, respectively (this classifier is built on the basis of the original 39-dimensional dataset, supplemented with features on the basis of the UMAP algorithm at $h = 6$ using the LF $L_4$).

In this case, obviously, preference should be given to the cost-sensitive classifier C3, because it allows us to work with low-dimensional dataset $h$ (at $h = 6$) in the UMAP algorithm, unlike the other two cost-sensitive classifiers C7, for which the dimension of the low-dimensional space $h$ in the UMAP algorithm is equal to 26 or 28.

As can be seen from Table 3, the LF turned out to be the most successful and reliable $L_1$ in the context of its use for the formation of new features: for all analyzed tuples of parameter values (*n_neighbors*, *min_dist*), using this function allowed us to develop cost-sensitive classifiers C7, which became the best in terms of maximizing the mean value of the metric $MacroF_1 - score$ in five out of nine experiments. In this regard, we can conclude that the LF $L_1$ successfully copes with the problem of reducing dimensionality when embedding the original dataset into spaces of different dimensions $h$ (both small and large) and can be recommended for further use when working with the UMAP algorithm.

Second place in the success rating was shared by LFs $L_2$ and $L_4$. The use of the LF $L_2$ made it possible to develop the cost-sensitive classifiers C7, which became the best in terms of maximizing the mean value of the metric $MacroF_1 - score$ in four out of nine experiments.

The use of the LF $L_4$ allowed to develop two cost-sensitive classifiers C7, one cost-sensitive classifier C3 and one cost-sensitive classifier C11, which became the best in terms of maximizing the mean value of the metric $MacroF_1 - score$ in four out of nine experiments. It should be noted that although one cost-sensitive classifier C7 (at $h = 6$) has the same mean value of the metric $MacroF_1 - score$ as the cost-sensitive classifier C8, we will assume that the cost-sensitive classifier C7 (at $h = 6$) is the winner, because it has a slightly lower standard deviation value (it is equal to 0.46), while the cost-sensitive classifier C8 has a standard deviation value of 0.47. However, 6 more additional features were used during development of the cost-sensitive classifier C7.

The LFs $L_0$ and $L_3$ in the proposed study did not show significant success in solving the problem of generating new features that would improve the data classification quality: the successes of these LFs, according to the experimental results given in Table 3, can rather be called random (single, characteristic only of individual tuples of parameter values (*n_neighbors*, *min_dist*). Working with these LFs can lead to a substantial increase in time expenditure without any guarantee that the results expected from them will be obtained. It should be emphasized that even the success of the LF $L_0$ is only partial: the results depend significantly on how successfully the initialization of the UMAP algorithm was performed.

Table 4 shows the main characteristics of the five best cost-sensitive classifiers from Table 3, ranked in the first three places, as well as the main characteristics of the base cost-sensitive classifier C1 (Table 2).

**Table 4.** Characteristics cost-sensitive the winning classifiers of the rating and the base cost-sensitive classifier C1.

| **Characteristics** | **Classifiers** | | | | | |
|---|---|---|---|---|---|---|
| | **Base Cost-Sensitive C1** | **C3 (at $h$=2)** | **C7 (at $h$=28)** | **C7 (at $h$=26)** | **C7 (at $h$=28)** | **C3 (at $h$=6)** |
| Number of features in the dataset | 39 | 41 | 67 | 65 | 67 | 45 |
| Loss function | - | $L_0$ | $L_1$ | $L_2$ | $L_2$ | $L_4$ |
| *n_neighbors* | - | 10 | 20 | 15 | 20 | 20 |
| *min_dist* | - | 0.3 | 0.3 | 0.3 | 0.3 | 0.3 |
| *gamma* | 0.8 | 0.5 | 0.7 | 0.5 | 0.5 | 0.4 |
| $C$ | 0.4 | 0.8 | 0.6 | 1 | 1 | 1.3 |
| $MacroF_1 - score$ (mean/SD) | 0.907/0.052 | 0.924/0.047 | 0.921/0.045 | 0.921/0.048 | 0.923/0.047 | 0.921/0.045 |
| $Accuracy$ (mean/SD) | 0.977/0.013 | 0.980/0.012 | 0.979/0.011 | 0.980/0.012 | 0.981/0.010 | 0.979/0.012 |
| $MacroRecall$ (mean/SD) | 0.907/0.066 | 0.920/0.061 | 0.911/0.056 | 0.913/0.065 | 0.916/0.062 | 0.915/0.060 |
| $MacroPrecision$ (mean/SD) | 0.923/0.053 | 0.943/0.046 | 0.944/0.052 | 0.943/0.053 | 0.945/0.050 | 0.941/0.045 |
| Training time (mean/SD), s. | 0.169/0.023 | 0.238/0.041 | 0.133/0.013 | 0.125/0.012 | 0.130/0.008 | 0.231/0.030 |
| Quality metrics calculation time (mean/SD), s. | 0.011/0.003 | 0.017/0.007 | 0.011/0.002 | 0.012/0.001 | 0.013/0.001 | 0.017/0.007 |

As can be seen from Table 4, all winning cost-sensitive classifiers surpassed the base cost-sensitive classifier C1 in terms of maximizing of the mean value of the metric $MacroF_1 - score$ (note that it was previously decided to use the mean value of the metric $MacroF_1 - score$ of the base cost-sensitive classifier C1 as the base (threshold) values for comparison). In addition, all winning cost-sensitive classifiers outperformed the base classifier C1 (Table 2, [20]); the classifier C1, balanced using the SMOTE algorithm (Table 2, [20]); and the classifier C7, balanced using the SMOTE algorithm (at $h = 28$) (Table 2, [20]) in terms of maximizing of the mean value of the metric $MacroF_1 - score$. In this case, there is a decrease in the SD value for the metric $MacroF_1 - score$, especially compared to the same value of the base classifier C1 (Table 2, [20]).

In addition, we can notice an increase in the mean value of the metric $MacroRecall$ with a decrease in the SD value for the metric $MacroRecall$, especially compared to the same value of the base classifier C1 (Table 2, [20]).

It should be noted that the winning cost-sensitive SVM classifiers are created on the basis of datasets whose number of features is greater than the number of features in the original 39-dimensional dataset.

Table 5 shows, for reference, the mean values of the metric $MacroF_1 - score$ and the corresponding SD values for the best cost-sensitive classifiers of 12 types. Bold font in Table 5 indicates information on the cost-sensitive classifiers that outperformed the base cost-sensitive classifier C1 (information for which is marked in bold italics) in terms of maximizing the mean value of the metric $MacroF_1 - score$.

We can notice that cost-sensitive classifiers of all types, with the exception of the cost-sensitive classifier C4, improved their mean values of the metric $MacroF_1 - score$ compared to similar classifiers, during the development of which no manipulations were used to overcome the problem of class imbalance (Table 2, [20]). First of all, it should be noted that classifiers C1, C3, C7, C8, C10, C11 and C12, whose mean values of the metric $MacroF_1 - score$ exceeded the similar value for the base classifier C1 (Table 2, [20]), turned out to be more than 0.900.

**Table 5.** Characteristic values for the best cost-sensitive classifiers of different types based on the metric $MacroF_1 - score$.

| Characteristic | Classifiers | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **C1** | **C2** | **C3** | **C4** | **C5** | **C6** | **C7** | **C8** | **C9** | **C10** | **C11** | **C12** |
| Mean | ***0.907*** * | 0.874 | **0.924** ** | 0.799 | 0.851 | 0.849 | **0.923** | **0.918** | 0.898 | **0.915** | **0.919** | **0.913** |
| Standard deviation | ***0.052*** | 0.079 | **0.047** | 0.108 | 0.065 | 0.067 | **0.047** | **0.047** | 0.063 | **0.053** | **0.054** | **0.065** |

* information for the base cost-sensitive classifier C1 is highlighted in italics; ** information for classifiers that outperformed the base cost-sensitive classifier C1 in terms of maximizing the mean value of the metric $MacroF_1 - score$ is highlighted in bold.

### 4.2.2. Identification of the Best Loss Functions in the UMAP Algorithm and Analysis of Their Capabilities in the Context of the Formation of New Features

In order to answer Question 2 in Table 6 for each tuple of parameter values (*n_neighbors*, *min_dist*) it is shown how many times a cost-sensitive classifier of a certain type performed no worse than the base cost-sensitive classifier C1 in terms of maximizing the mean value of the metric $MacroF_1 - score$. Moreover, the percentage of successfulness to the total number of experiments is indicated for each tuple of parameter values (*n_neighbors, min_dist*). The total number of experiments is 19, because the dimension of space $h$ in the UMAP algorithm varies from 2 to 39 with a step of 2. Table 6 provides information only about those cost-sensitive classifiers that that were no worse than the base cost-sensitive classifier C1 more than once (in all experiments). It can be noted that, according to the information from Table 5, the cost-sensitive classifier C12 outperformed the base cost-sensitive classifier C1, but it did this only once (when using the LF $L_0$ in the UMAP algorithm (at $h = 2$, *n_neighbors* = 10 and *min_dist* = 0.3).

**Table 6.** Names of the best cost-sensitive SVM classifiers and their win rates.

| **Tuple of Parameter Values (*n_neighbors, min_dist*)** | **Loss Functions** | | | | |
|---|---|---|---|---|---|
| | $L_0$ | $L_1$ | $L_2$ | $L_3$ | $L_4$ |
| (10, 0.1) | C3: 0 (0%)<br>C7: 5 (26.316%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 0 (0%) | C3: 2 (10.526%)<br>C7: 18 (94.737%)<br>C8: 19 (100%)<br>C10: 9 (47.368%)<br>C11: 2 (10.526%) | C3: 2 (10.526%)<br>C7: 18 (94.737%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 0 (0%) | C3: 1 (5.263%)<br>C7: 2 (10.526%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 0 (0%) | C3: 3 (15.789%)<br>C7: 7 (36.842%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 1 (5.263%) |
| (10, 0.2) | C3: 1 (5.263%)<br>C7: 7 (36.842%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 0 (0%) | C3: 3 (15.789%)<br>C7: 18 (94.737%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 2 (10.526%) | C3: 2 (10.526%)<br>C7: 17 (89.474%)<br>C8: 19 (100%)<br>C10: 2 (10.526%)<br>C11: 0 (0%) | C3: 0 (0%)<br>C7: 1 (5.263%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 0 (0%) | C3: 3 (15.789%)<br>C7: 3 (15.789%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 1 (5.263%) |
| (10, 0.3) | C3: 1 (5.263%)<br>C7: 11 (57.895%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 1 (5.263%) | C3: 2 (10.526%)<br>C7: 18 (94.737%)<br>C8: 19 (100%)<br>C10: 2 (10.526%)<br>C11: 2 (10.526%) | C3: 2 (10.526%)<br>C7: 17 (89.474%)<br>C8: 19 (100%)<br>C10: 1 (5.263%)<br>C11: 0 (0%) | C3: 0 (0%)<br>C7: 2 (10.526%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 0 (0%) | C3: 3 (15.789%)<br>C7: 3 (15.789%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 2 (10.526%) |
| (15, 0.1) | C3: 1 (5.263%)<br>C7: 4 (21.053%)<br>C8: 19 (100%)<br>C10: 1 (5.263%)<br>C11: 0 (0%) | C3: 3 (15.789%)<br>C7: 18 (94.737%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 1 (5.263%) | C3: 3 (15.789%)<br>C7: 18 (94.737%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 1 (5.263%) | C3: 0 (0%)<br>C7: 4 (21.053%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 0 (0%) | C3: 4 (21.053%)<br>C7: 15 (78.947%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 3 (15.789%) |
| (15, 0.2) | C3: 2 (10.526%)<br>C7: 8 (42.105%)<br>C8: 19 (100%)<br>C10: 1 (5.263%)<br>C11: 0 (0%) | C3: 2 (10.526%)<br>C7: 18 (94.737%)<br>C8: 19 (100%)<br>C10: 3 (15.789%)<br>C11: 2 (10.526%) | C3: 0 (0%)<br>C7: 19 (100%)<br>C8: 19 (100%)<br>C10: 5 (26.316%)<br>C11: 2 (10.526%) | C3: 1 (5.263%)<br>C7: 6 (31.579%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 0 (0%) | C3: 4 (21.053%)<br>C7: 9 (47.368%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 5 (26.316%) |
| (15, 0.3) | C3: 0 (0%)<br>C7: 4 (21.053%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 0 (0%) | C3: 0 (0%)<br>C7: 18 (94.737%)<br>C8: 19 (100%)<br>C10: 4 (21.053%)<br>C11: 2 (10.526%) | *C3: 1 (5.263%)*<br>C7: 19 (100%)<br>C8: 19 (100%)<br>C10: 8 (42.105%)<br>C11: 2 (10.526%) | C3: 1 (5.263%)<br>C7: 4 (21.053%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 0 (0%) | C3: 6 (31.579%)<br>C7: 7 (36.842%)<br>C8: 19 (100%)<br>C10: 1 (5.263%)<br>C11: 4 (21.053%) |

**Table 6.** *Cont.*

| **Tuple of Parameter Values (*n_neighbors, min_dist*)** | **Loss Functions** | | | | |
|---|---|---|---|---|---|
| | $L_0$ | $L_1$ | $L_2$ | $L_3$ | $L_4$ |
| (20, 0.1) | C3: 1 (5.263%)<br>C7: 3 (15.789%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 0 (0%) | C3: 3 (15.789%)<br>C7: 19 (100%)<br>C8: 19 (100%)<br>C10: 6 (31.579%)<br>C11: 3 (15.789%) | C3: 1 (5.263%)<br>C7: 15 (78.947%)<br>C8: 19 (100%)<br>C10: 4 (21.053%)<br>C11: 1 (5.263%) | C3: 2 (10.526%)<br>C7: 2 (10.526%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 2 (10.526%) | C3: 9 (47.368%)<br>C7: 16 (84.211%)<br>C8: 19 (100%)<br>C10: 1 (5.263%)<br>C11: (36.842%) |
| (20, 0.2) | C3: 1 (5.263%)<br>C7: 6 (31.579%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 0 (0%) | C3: 4 (21.053%)<br>C7: 19 (100%)<br>C8: 19 (100%)<br>C10: 4 (21.053%)<br>C11: 3 (15.789%) | C3: 2 (10.526%)<br>C7: 18 (94.737%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 1 (5.263%) | C3: 1 (5.263%)<br>C7: 1 (5.263%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 0 (0%) | C3: 15 (78.947%)<br>C7: 17 (89.474%)<br>C8: 19 (100%)<br>C10: 2 (10.526%)<br>C11: 13 (68.421%) |
| (20, 0.3) | C3: 1 (5.263%)<br>C7: 7 (36.842%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 0 (0%) | C3: 5 (26.316%)<br>C7: 19 (100%)<br>C8: 19 (100%)<br>C10: 3 (15.789%)<br>C11: 5 (26.316%) | C3: 1 (5.263%)<br>C7: 19 (100%)<br>C8: 19 (100%)<br>C10: 11 (57.895%)<br>C11: 2 (10.526%) | C3: 1 (5.263%)<br>C7: 2 (10.526%)<br>C8: 19 (100%)<br>C10: 0 (0%)<br>C11: 0 (0%) | C3: 14 (73.684%)<br>C7: 16 (84.211%)<br>C8: 19 (100%)<br>C10: 4 (21.053%)<br>C11: 11 (57.895%) |

Analysis of the results given in Table 6 confirms the clear advantage of the LF $L_1$: its use allows us to develop cost-sensitive SVM classifiers of different types that exceed the base cost-sensitive classifier C1 in terms of maximizing of the mean value of the metric $MacroF_1 - score$.

In addition, analysis of the results given in Table 6 allows us to notice that, usually, the number of successful cost-sensitive classifiers C3 for each tuple of parameter values (*n_neighbors*, *min_dist*) is no more than three. Most often, such situations arise when the dimensionality $h$ of the low-dimensional space in the UMAP algorithm is two, four or six. However, when using the LF $L_4$ in the UMAP algorithm, the number of successful cost-sensitive classifiers C3 for each tuple of parameter values (*n_neighbors*, *min_dist*) is always at least 3, and for tuples (*n_neighbors*, *min_dist*) taking values (20, 0.2) and (20, 0.3), the number of successful cost-sensitive classifiers C3 is 15 and 14, respectively (i.e., such classifiers are successful with different dimensions $h$ of space (both small and large)).

It should be noted that all cost-sensitive SVM classifiers, indicated in Table 6, are developed based on datasets whose number of features is greater than the number of features in the original 39-dimensional dataset.

## 5. Discussion

Experimental results of creating SVM classifiers using the cost-sensitive SVM algorithm confirmed that high data classification quality can be achieved through modification of the original dataset by adding different combinations of new features on the basis of the approximate entropy AE and the fractal dimensions KFD and HFD, as well as on the basis of the UMAP algorithm. The most successful in terms of maximizing the mean value of the metric $MacroF_1 - score$ turned out to be classifiers C3, C7 and C8, developed, respectively, on the basis of the original three-class 39-dimensional datasets, supplemented, respectively, with new features on the basis of the UMAP algorithm; on the basis of the UMAP algorithm and the approximate entropy AE; as well as only on the basis of the approximate entropy AE.

All winning classifiers, C3, C7, C11, C8, C10 and C12 (Table 5), presented in descending order of the mean values of the metric $MacroF_1 - score$, outperformed the base SVM classifier (Table 2, [20]) in terms of maximizing the mean value of the metric $MacroF_1 - score$ by 5.359%, 5.245%, 4.789%, 4.675%, 4.333% and 4.105%, respectively.

All winning classifiers, C3, C7, C11, C8, C10 and C12 (Table 5), presented in descending order of the mean values of the metric $MacroF_1 - score$, outperformed the base cost-sensitive classifier C1 (Table 2) in terms of maximizing the mean value of the metric $MacroF_1 - score$ by 1.874%, 1.764%, 1.323%, 1.2135, 0.882% and 0.662%, respectively.

Also, all winning classifiers, C3, C7, C11, C8, C10 and C12 (Table 5), presented in descending order of the mean values of the metric $MacroF_1 - score$, outperformed the best classifier C1 on the basis of the SMOTE algorithm (Table 2, [20]) in terms of maximizing of the mean value of the metric $MacroF_1 - score$ by 1.538%, 1.429%, 0.989%, 0.879%, 0.549% and 0.330%, respectively.

In addition, five out of the six winning classifiers, C3, C7, C11, C8, C10 and C12 (Table 5)—namely classifiers C3, C7, C11, C8 and C10, presented in descending order of the mean values of the metric $MacroF_1 - score$—outperformed the best classifier C7 on the basis of the SMOTE algorithm (Table 2, [20]) in terms of maximizing the mean value of the metric $MacroF_1 - score$ by 1.094%, 0.985%, 0.547%, 0.438% and 0.109%, respectively. Only classifier C12 turned out to be worse than the best classifier C7 on the basis of the SMOTE algorithm (Table 2, [20]) in terms of maximizing the mean value of the metric $MacroF_1 - score$ by 0.109%.

We can see that the advantage of the best cost-sensitive classifiers over the best classifiers C1 and C7 on the basis of the SMOTE algorithm is not very large. However, it was possible to significantly reduce the time expenditures for developing and testing classifiers (Table 4) compared to similar time estimates obtained when developing classifiers using the SMOTE algorithm, which implements the strategy of oversampling new data patterns (Table 2, [20]).

So, for example, we can compare the total time spent on the training and testing of the cost-sensitive classifiers C7, recognized as the best in the proposed study and in [20]. They are created on the basis of the original 39-dimensional dataset, supplemented with features created on the basis of the approximate entropy AE and the UMAP algorithm. At the same time, in [20], the library implementation of the UMAP algorithm with the default parameter values is applied, and in the proposed study for the UMAP algorithm, the value enumeration of the parameters *n_neighbors* and *min_dist* is implemented for five LFs with the choice of the best variant. However, the CIP is solved differently in the proposed study and in [20]. The best SVM classifier in [20] is the classifier C7, in which the CIP was solved using the oversampling SMOTE algorithm. One of the best cost-sensitive SVM classifiers in the proposed study is the cost-sensitive classifier C7 (Table 4), for which the CIP was solved using the CSL concept. In this case, the LF $L_2$ was used in the UMAP algorithm, and the parameters *n_neighbors* and *min_dist* took the values 20 and 0.3, respectively. The total time spent on the training and testing of the cost-sensitive classifier C7 in the proposed study turned out to be only 1.1 times longer than the time to develop the base classifier C1 and 1.26 times less than the time to develop the base cost-sensitive classifier C1, while it took 3.48 times less than the same time for classifier C7 built using the SMOTE algorithm and being the best in [20], as well as 6.29 times less than the same time for classifier C1 built using the SMOTE algorithm [20].

The function $L_1$ that implements the calculation of fuzzy cross-entropy with FGD should be recognized as the best function in the context of working with different LFs in the UMAP algorithm in order to form new features that complement the original dataset and ensure the development of classifiers with high data classification quality. The function $L_3$ that implements the calculation of intuitionistic fuzzy cross-entropy with FGD and then the LF $L_0$ that implements the calculation of implicitly set LF should be recognized as the worst LFs. Such conclusions were made based on the efficiency of these LFs in terms of embedding of the original 39-dimensional dataset into spaces of arbitrary dimensions $h$ (from 2 to 38 with a step of 2) in the context of the formation of new datasets and the further development of a classifier with high data classification quality, superior to the quality of the base cost-sensitive classifier C1. Thus, the function $L_1$ was successful in these terms for all space dimensions $h$, while the successes of the function $L_3$ were solitary. The $L_0$ function's successes were also solitary. Although the use of the function $L_0$ made it possible to obtain the best classifier C3 in the terms of maximizing the mean value of the metric $MacroF_1 - score$, that is, the absolute winning classifier in our rating (Table 3), the use of this LF is associated with selecting the value of another parameter, namely the *random_state* parameter, affecting the final results of UMAP algorithm. This leads to additional time expenditures without a guaranteed expected result. The LFs $L_2$ and $L_4$ turned out to be less successful than the LF $L_1$ in the terms under consideration but more successful than the LFs $L_0$ and $L_3$. However, their use in the UMAP algorithm made it possible to obtain the winning classifiers in our rating (Table 3), so it is advisable to use them (in the absence of significant restrictions on the time spent on the development of high-quality classifiers).

In order to statistically test the superiority of the developed classifiers, which solve the CIP in different ways, over other classifiers, we applied the Wilcoxon signed rank test [94,95] to the obtained quality estimates of various classifiers. To obtain statistically representative results, we repeated the evaluation of each pair of classifiers using stratified 10-fold cross-validation [90,91] with three-time repetition: each time, the datasets were divided into 10 blocks, and the classifiers were evaluated on 10 different parts of each dataset. This was performed three times. Thus, each of the resulting classifier quality distributions contained a total of 30 values. The distribution obtained for the base classifier C1 based on the original dataset was compared with all other distributions obtained for other classifiers in this study. When assessing the quality of the classifiers, we considered the metrics $MacroF_1 - score$ and $Recall$, as well as estimates of the training time and testing time of the classifiers. The values of metrics $MacroF_1 - score$ and $Recall$ should be maximized. Estimates of training time and testing time for classifiers should be minimized.

According to the null hypothesis $H_0$, the two compared distributions did not have statistically significant differences [94,95]. The $p$-value was set to 0.05. The obtained results are presented in Tables A4–A7 in Appendix C. We compared the classifiers developed in this study with the base SVM classifier based on the original dataset [20], with the SVM classifier based on the original dataset and the SMOTE algorithm [20], and also with the SVM classifier based on the original dataset expanded using features based on the approximate entropy AE, the UMAP algorithm and the SMOTE algorithm [20]. In Tables A4–A7, the "=" sign means that there are no statistically significant differences between the compared distributions of the classifiers, the "+" sign means that the classifier in the row header is superior to the classifier in the column header, and the "–" sign means the opposite.

According to Table A4, classifiers C3 (at $h = 2$) with $L_0$, C7 (at $h = 28$) with $L_1$, C7 (at $h = 26$) with $L_2$, C7 (at $h = 28$) with $L_2$, C3 (at $h = 6$) with $L_4$ and balanced C7 (at $h = 28$) using the SMOTE algorithm [20] surpassed the base classifier C1 as measured by the metric $MacroF_1 - score$. The base cost-sensitive classifier C1 and the balanced classifier C1 using the SMOTE algorithm [20] have no statistically significant differences from the base C1 by this metric. When comparing the classifiers developed in the proposed study using the same metric with classifiers developed using the SMOTE algorithm [20], it was possible to reveal only the superiority of the C3 classifier (at $h = 2$) with $L_0$.

According to Table A5, all classifiers that solve the CIP in one way or another outperformed the base classifier C1 as measured by the metric $Recall$. When comparing the values of the same metric of the classifiers developed in the proposed study with classifiers developed using the SMOTE algorithm [20], it was also possible to reveal only the superiority of the C3 classifier (at $h = 2$) with $L_0$.

According to Table A6, all classifiers that solve the CIP in one way or another lose to the base classifier C1 in training time (which was expected). When comparing the training time of the classifiers developed in the proposed study with the classifiers developed using the SMOTE algorithm [20], the superiority of the cost-sensitive classifiers is observed. At the same time, the balanced classifier C7 (at $h = 28$) using the SMOTE algorithm [20] outperformed the balanced classifier C1 using the SMOTE algorithm [20] in training time (possibly due to better separability of classes).

According to Table A7, all classifiers that solve the CIP in one way or another lose to the base classifier C1 in terms of testing time (which was expected). When comparing the testing time of the classifiers developed in the proposed study with the same time of the balanced classifier C1 using the SMOTE algorithm [20], the superiority of the cost-sensitive classifiers is observed, with the exception of the classifier C3 (at $h = 2$) with $L_0$ that lost. At the same time, the balanced classifier C7 (at $h = 28$) using the SMOTE algorithm [20] outperformed all classifiers in testing time except the base classifier C1, which it lost to.

In general, the following should be noted. All classifiers that solve the CIP in one way or another and are developed on the basis of modified datasets are superior to the base classifier C1 by the metrics $MacroF_1 - score$ and $Recall$.

The best cost-sensitive classifiers developed in the proposed study outperform the base cost-sensitive C1 in terms of metrics $MacroF_1 - score$ and $Recall$; however, they do not have statistical differences among themselves in these metrics. In terms of training time and classifier testing time, all these classifiers are statistically different. Therefore, when choosing a classifier, one can focus, for example, on the minimal time required to train a classifier. Thus, classifier C7 (at $h = 26$) with $L_2$ provides minimal time of training.

In general, limitations on the applicability of the proposed approach may be due to the following reasons. First, we may experience limitations caused by the computational complexity of developing SVM classifiers using standard implementations of the SVM algorithm. However, this problem can be solved using modern SVM solvers [87–89]. Secondly, certain problems may be caused by the very nature of the used dataset. Data should be subjected to exploratory analysis and, if possible, cleared of omissions, outliers and similar defects. In the case of data of very poor quality, their preprocessing can lead to an even greater imbalance of classes, up to the loss of a significant part of the patterns belonging to minority classes. In this regard, a qualitative solution to the imbalance problem using CSL algorithms or, for example, oversampling algorithms will be questionable. In addition, the nature of the used dataset may be such that the data of different classes in it will initially be poorly separable, for example, due to the poor separability of patterns determined by blood protein markers according to their membership in different classes. In this regard, both attempts to develop SVM classifiers based on the original dataset and attempts to generate new features, for example, based on approximation entropy and the UMAP algorithm, will be unsuccessful: new features will not improve the quality of dividing data patterns into different classes. Third, it should be noted that additional experiments are needed when determining penalties for misclassifying patterns of different classes in the case of CSL or when determining the class ratio that should be achieved after restoring the balance, for example, using oversampling algorithms.

In the proposed study, we used a dataset, the properties of which were previously studied in detail in [13,14,20], and in [20] it was noted that there was no strong correlation both between the features of the original dataset and when introducing those new features approved for use.

In the proposed study, we used a dataset, the properties of which were previously studied in detail in [13,14,20], including in [20], where it was noted that there was no strong correlation both between the features of the original dataset and when introducing those approved for use new features.

It should be noted that the combination of the CSL principles and the approach proposed in [20] to the creation of datasets by forming new features using various technologies with their acquisition and application as a new dataset or adding to the original dataset may be considered appropriate. In this case, varying the values of the parameters *n_neighbors* and *min_dist*, as well as working with several LFs in the UMAP algorithm, ultimately made it possible to obtain qualitatively better cost-sensitive SVM classifiers in terms of maximizing the mean value of the metric $MacroF_1 - score$.

## 6. Conclusions

In this research, we investigated a previously suggested approach [20] for the diagnosis of cancer using blood protein markers through creation of the SVM classifiers on the basis of datasets with a variety of features of different nature. These features may correspond to blood protein markers or be constructed using methods for calculating entropy and fractal dimensions, as well as using the UMAP algorithm. These medical datasets are imbalanced.

To overcome the class imbalance problem, the concept of cost-sensitive learning was implemented, the use of which allowed the best developed SVM classifiers to outperform the base SVM classifier in data classification quality and the best SVM classifiers developed on the basis of the oversampling strategy, not only in data classification quality but also in the time spent on their development. The most successful in terms of maximizing the mean value of the metric $MacroF_1 - score$ are the following cost-sensitive SVM classifiers, listed in descending order of successfulness: C3, C7, C8, C11 and C10. The UMAP algorithm was applied to create new features in datasets used to develop classifiers C3, C7 and C11. The approximate entropy was applied to create a new feature in datasets used to develop classifiers C7, C8 and C10. The Katz and Higuchi fractal dimensions were applied to create new features in datasets used to develop classifiers C10 and C11. Each time, new features supplemented the original dataset. We showed that to create additional features on the basis of the UMAP algorithm, it is advisable to use the LFs $L_1$, $L_2$ and $L_4$, defined explicitly by formulas (13), (14) and (10). The use of an implicitly defined LF, which we called $L_0$, applied in the library implementation [80], is complicated because of the impossibility of explicitly estimating the values for the LF, although it cannot be considered unambiguously inexpedient.

The purpose of further research is to explore ways to improve data classification quality by forming new features on the basis of various dimensionality reduction algorithms, such as UMAP [53], t-SNE [77], TriMAP (Triplet Manifold Approximation) [96] and PaCMAP (Pairwise Controlled Manifold Approximation) [97], for which both the initialization of the initial embedding of patterns into low-dimensional space and the optimization of the embedding of patterns into low-dimensional space are performed in different manners. We also plan to implement a simultaneous combination of CSL and oversampling technologies with the selection of the best combinations of penalties and the best class proportions when synthesizing new data patterns.

**Funding:** This research received no external funding.

**Data Availability Statement:** The data presented in this study are openly available in [54].

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

## Appendix A

**Table A1.** Names of concepts and their abbreviations.

| Name of Concept | Abbreviation |
|---|---|
| ADASYN | ADaptive SYNthetic sampling approach |
| AE | Approximate Entropy |
| AUC | Area Under Curve |
| CIP | Class Imbalance Problem |
| COSMIC | Catalog of Somatic Mutations in Cancer |
| CSL | Cost-Sensitivity Learning |
| BPM | Blood Protein Marker |
| FGD | Full Gradient Descent |
| GT | Gene Test |
| HC | Hjorth Complexity |
| HFD | Higuchi Fractal Dimension |
| HM | Hjorth Mobility |
| KFD | Katz Fractal Dimension |
| DL | Deep Learning |
| DM | Data Mining |
| kNN | k-Nearest Neighbors |
| LF | Loss Function |

**Table A1.** *Cont.*

| Name<br>of Concept | Abbreviation |
|---|---|
| LR | Logistic Regression |
| ML | Machine Learning |
| OD | Oncological Disease |
| OvO | One-vs-One strategy |
| OvR | One-vs-Rest strategy |
| PaCMAP | Pairwise Controlled Manifold Approximation |
| PFD | Petrosian Fractal Dimension |
| PT | Protein Test |
| SGD | Stochastic Gradient Descent |
| SMOTE | Synthetic Minority Over-Sampling Technique |
| SVDE | Singular Value Decomposition Entropy |
| SVM | Support Vector Machine |
| RBF | Radial Basis Function |
| RF | Random Forest |
| RQ | Research Question |
| t-SNE | T-Distributed Stochastic Neighbor Embedding |
| TriMAP | Triplet Manifold Approximation |
| SE | Sample Entropy |
| SD | Standard Deviation |
| SPE | SPectral Entropy |
| UMAP | Uniform Manifold Approximation and Projection |

**Table A2.** Datasets and the composition of their features.

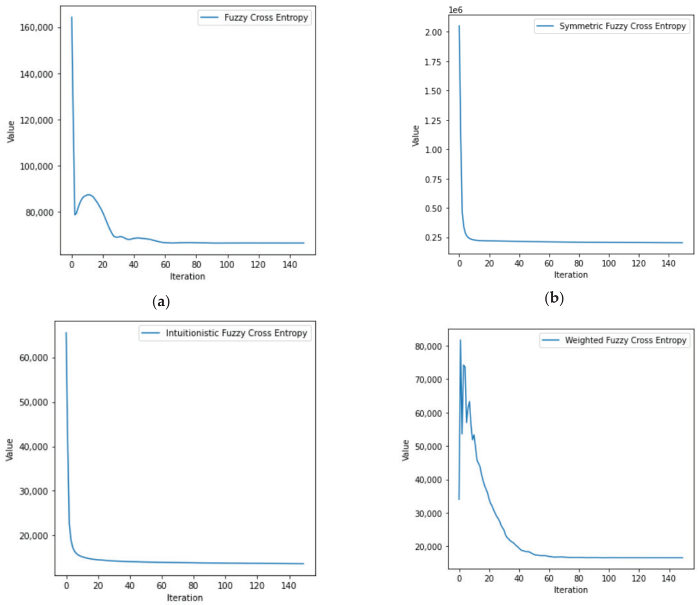
| Dataset Name | The Composition of Features |
|---|---|
| C1 | FOD * |
| C2 | FUMAP ** |
| C3 | FOD, FUMAP |
| C4 | FUMAP, FAE *** |
| C5 | FUMAP, FHKFR **** |
| C6 | FUMAP, FAE, FHKFR |
| C7 | FOD, FUMAP, FAE |
| C8 | FOD, FAE |
| C9 | FOD, FHKFR |
| C10 | FOD, FAE, FHKFR |
| C11 | FOD, FUMAP, FHKFR |
| C12 | FOD FUMAP, FAE, FHKFR |

* FOD are the Features of the Original Dataset; ** FUMAP are the Features on the basis of the UMAP algorithm; *** FAE is the Feature on the basis of Approximate Entropy; **** FHKFR are the Features on the basis of Higuchi and Katz Fractal Dimensions.

**Table A3.** Basic algorithms and description of their changeable parameters.

| Algorithm | Parameter | Parameter Value or Range with Step of Change |
|---|---|---|
| SVM | $C$ is the regularization parameter ($C$ in the scikit-learn library of Python) | [0.4, 2] with a step of 0.1 |
| | $\sigma$ ($\sigma > 0$) is the parameter of the RBF kernel (*gamma* in the scikit-learn library of Python) | [0.4, 2] with a step of 0.1 |
| UMAP | $k$ in the number of nearest neighbors that are found for each pattern in the high-dimensional space (*n_neighbors* in the software library [80]) | [10, 20] with a step of 5 |
| | $d_{min}$ is the threshold distance ($d_{min} \in (0,\ 1]$) that influences the density of clusters created in the low-dimensional space (*min_dist* in the software library [80]) | [0.1, 0.3] with a step of 0.1 |
| | $h$ is the dimension of the low-dimensional space (*n_components* in the software library [80]) | [2, 38] with a step of 2 |
| | *metric* is the distance metric in the software library [80] | Euclidean |
| | *random_state* is the parameter responsible for initialization of UMAP algorithm and reproducibility of results in the software library [80] | 42 |

## Appendix B



(**a**)

(**b**)

(**c**)

(**d**)

**Figure A1.** Graphical dependencies for LFs obtained by constructing embeddings of the original 39-dimensional three-class dataset in the two-dimensional space on the basis of the UMAP algorithm with various LFs with parameter values *n_neighbors* and *min_dist*, ensuring the creation of the best cost-sensitive classifiers C3 (at $h = 2$) in the terms of maximizing the mean value of the metric $MacroF_1 - score$) presented in Figure 3. (**a**) $L_1$: LF based on fuzzy cross-entropy with FGD (*n_neighbors* = 15; *min_dist* = 0.1; $MacroF_1 - score$ : mean = 0.918; SD = 0.049); (**b**) $L_2$: LF based on symmetric fuzzy cross-entropy with FGD (*n_neighbors* = 15; *min_dist* = 0.1; $MacroF_1 - score$ : mean = 0.916; SD = 0.047); (**c**) $L_3$: LF based on intuitionistic fuzzy cross-entropy with FGD (*n_neighbors* = 15; *min_dist* = 0.3; $MacroF_1 - score$ : mean = 0.914; SD = 0.055); (**d**) $L_4$: LF based on weighted fuzzy cross-entropy with FGD (*n_neighbors* = 20; *min_dist* = 0.2; $MacroF_1 - score$ : mean = 0.915; SD = 0.044).

## Appendix C

**Table A4.** Results of the Wilcoxon signed rank test applied to classifiers, which were estimated on the basis of the metric $MacroF_1 - score$.

| **Classifier** | **Base C1** | | **Balanced C1 Using SMOTE Algorithm [20]** | | **Balanced C7 (at *h* = 28) Using SMOTE Algorithm [20]** | |
|---|---|---|---|---|---|---|
| **Classifier** | **Sign** | ***p*-Value** | **Sign** | ***p*-Value** | **Sign** | ***p*-Value** |
| base cost-sensitive C1 | = | 0.071 | = | 0.950 | = | 0.761 |
| C3 (at $h = 2$) with $L_0$ | + | 0.002 | = | 0.200 | + | 0.034 |
| C7 (at $h = 28$) with $L_1$ | + | 0.005 | = | 0.214 | = | 0.594 |
| C7 (at $h = 26$) with $L_2$ | + | 0.004 | = | 0.252 | = | 0.462 |
| C7 (at $h = 28$) with $L_2$ | + | 0.004 | = | 0.147 | = | 0.795 |
| C3 (at $h = 6$) with $L_4$ | + | 0.008 | = | 0.178 | = | 0.795 |
| balanced C1 using SMOTE algorithm [20] | = | 0.125 | Not | Not | = | 0.795 |
| balanced C7 (at $h = 28$) using SMOTE algorithm [20] | + | 0.021 | = | 0.795 | Not | Not |

**Table A5.** Results of the Wilcoxon signed rank test applied to classifiers, which were estimated on the basis of the metric *Recall*.

| **Classifier** | **Base C1** | | **Balanced C1 Using SMOTE Algorithm [20]** | | **Balanced C7 (at $h$ = 28) Using SMOTE Algorithm [20]** | |
|---|---|---|---|---|---|---|
| **Classifier** | **Sign** | ***p*-Value** | **Sign** | ***p*-Value** | **Sign** | ***p*-Value** |
| base cost-sensitive C1 | + | 0.001 | = | 0.740 | = | 0.102 |
| C3 (at $h = 2$) with $L_0$ | + | $4.571 \times 10^{-5}$ | = | 0.153 | + | 0.039 |
| C7 (at $h = 28$) with $L_1$ | + | $2.194 \times 10^{-4}$ | = | 0.331 | = | 0.810 |
| C7 (at $h = 26$) with $L_2$ | + | $3.405 \times 10^{-4}$ | = | 0.365 | = | 0.576 |
| C7 (at $h = 28$) with $L_2$ | + | $2.682 \times 10^{-4}$ | = | 0.207 | = | 0.420 |
| C3 (at $h = 6$) with $L_4$ | + | $2.309 \times 10^{-4}$ | = | 0.283 | = | 0.909 |
| balanced C1 using SMOTE algorithm [20] | + | 0.004 | Not | Not | = | 0.724 |
| balanced C7 (at $h = 28$) using SMOTE algorithm [20] | + | $2.043 \times 10^{-4}$ | = | 0.724 | Not | Not |

**Table A6.** Results of the Wilcoxon signed rank test applied to classifiers, which were estimated on the basis time of training.

| **Classifier** | **Base C1** | | **Balanced C1 Using SMOTE Algorithm [20]** | | **Balanced C7 (at $h$ = 28) Using SMOTE Algorithm [20]** | |
|---|---|---|---|---|---|---|
| **Classifier** | **Sign** | ***p*-Value** | **Sign** | ***p*-Value** | **Sign** | ***p*-Value** |
| base cost-sensitive C1 | − | $1.863 \times 10^{-9}$ | + | $1.863 \times 10^{-9}$ | + | $1.863 \times 10^{-9}$ |
| C3 (at $h = 2$) with $L_0$ | − | $1.863 \times 10^{-9}$ | + | $1.863 \times 10^{-9}$ | + | $1.863 \times 10^{-9}$ |
| C7 (at $h = 28$) with $L_1$ | − | $1.863 \times 10^{-9}$ | + | $1.863 \times 10^{-9}$ | + | $1.863 \times 10^{-9}$ |
| C7 (at $h = 26$) with $L_2$ | − | $3.725 \times 10^{-9}$ | + | $1.863 \times 10^{-9}$ | + | $1.863 \times 10^{-9}$ |
| C7 (at $h = 28$) with $L_2$ | − | $3.725 \times 10^{-9}$ | + | $1.863 \times 10^{-9}$ | + | $1.863 \times 10^{-9}$ |
| C3 (at $h = 6$) with $L_4$ | − | $5.588 \times 10^{-9}$ | + | $1.863 \times 10^{-9}$ | + | $3.239 \times 10^{-6}$ |
| balanced C1 using SMOTE algorithm [20] | − | $1.863 \times 10^{-9}$ | Not | Not | − | $1.863 \times 10^{-9}$ |
| balanced C7 (at $h = 28$) using SMOTE algorithm [20] | − | $1.863 \times 10^{-9}$ | + | $1.863 \times 10^{-9}$ | Not | Not |

**Table A7.** Results of the Wilcoxon signed rank test applied to classifiers, which were estimated on the basis time of testing.

| **Classifier** | **Base C1** | | **Balanced C1 Using SMOTE Algorithm [20]** | | **Balanced C7 (at $h$ = 28) Using SMOTE Algorithm [20]** | |
|---|---|---|---|---|---|---|
| **Classifier** | **Sign** | ***p*-Value** | **Sign** | ***p*-Value** | **Sign** | ***p*-Value** |
| base cost-sensitive C1 | − | $2.608 \times 10^{-8}$ | + | 0.013 | − | $1.863 \times 10^{-8}$ |
| C3 (at $h = 2$) with $L_0$ | − | $1.106 \times 10^{-4}$ | − | 0.002 | − | 0.003 |
| C7 (at $h = 28$) with $L_1$ | − | $2.608 \times 10^{-8}$ | + | 0.0128 | − | $1.863 \times 10^{-8}$ |
| C7 (at $h = 26$) with $L_2$ | − | $3.725 \times 10^{-9}$ | + | $1.863 \times 10^{-8}$ | − | $2.608 \times 10^{-8}$ |
| C7 (at $h = 28$) with $L_2$ | − | $3.725 \times 10^{-9}$ | + | $1.863 \times 10^{-8}$ | − | $2.608 \times 10^{-8}$ |
| C3 (at $h = 6$) with $L_4$ | − | $2.608 \times 10^{-8}$ | + | 0.013 | − | $1.863 \times 10^{-8}$ |
| balanced C1 using SMOTE algorithm [20] | − | $1.863 \times 10^{-9}$ | Not | Not | − | $1.863 \times 10^{-9}$ |
| balanced C7 (at $h = 28$) using SMOTE algorithm [20] | − | 0.002 | + | $1.863 \times 10^{-9}$ | Not | Not |

## References

1. 2021 Global Health Care Outlook. Available online: https://www2.deloitte.com/cn/en/pages/life-sciences-and-healthcare/articles/2021-global-healthcare-outlook.html (accessed on 4 January 2024).
2. Slim, K.; Selvy, M.; Veziant, J. Conceptual innovation: 4P Medicine and 4P surgery. *J. Visc. Surg.* **2021**, *158*, S12–S17. [CrossRef] [PubMed]
3. Brar, A.; Zhu, A.; Baciu, C.; Sharma, D.; Xu, W.; Orchanian-Cheff, A.; Wang, B.; Reimand, J.; Grant, R.; Bhat, M. Development of diagnostic and prognostic molecular biomarkers in hepatocellular carcinoma using machine learning: A systematic review. *Liver Cancer Int.* **2022**, *3*, 141–161. [CrossRef]
4. Li, P.; Xu, S.; Han, Y.; He, H.; Liu, Z. Machine learning-empowered cis-diol metabolic fingerprinting enables precise diagnosis of primary liver cancer. *Chem. Sci.* **2023**, *14*, 2553–2561. [CrossRef]
5. Ma, J.; Bo, Z.; Zhao, Z.; Yang, J.; Yang, Y.; Li, H.; Yang, Y.; Wang, J.; Su, Q.; Wang, J.; et al. Machine Learning to Predict the Response to Lenvatinib Combined with Transarterial Chemoembolization for Unresectable Hepatocellular Carcinoma. *Cancers* **2023**, *15*, 625. [CrossRef] [PubMed]
6. Fu, Y.; Si, A.; Wei, X.; Lin, X.; Ma, Y.; Qiu, H.; Guo, Z.; Pan, Y.; Zhang, Y.; Kong, X.; et al. Combining a machine-learning derived 4-lncRNA signature with AFP and TNM stages in predicting early recurrence of hepatocellular carcinoma. *BMC Genom.* **2023**, *24*, 89. [CrossRef] [PubMed]
7. Iseke, S.; Zeevi, T.; Kucukkaya, A.S.; Raju, R.; Gross, M.; Haider, S.P.; Petukhova-Greenstein, A.; Kuhn, T.N.; Lin, M.; Nowak, M.; et al. Machine Learning Models for Prediction of Posttreatment Recurrence in Early-Stage Hepatocellular Carcinoma Using Pretreatment Clinical and MRI Features: A Proof-of-Concept Study. *AJR Am. J. Roentgenol.* **2023**, *220*, 245–255. [CrossRef] [PubMed]
8. Chaudhary, K.; Poirion, O.B.; Lu, L.; Garmire, L.X. Deep learning-based multi-omics integration robustly predicts survival in liver cancer. *Clin. Cancer Res.* **2018**, *24*, 1248–1259. [CrossRef] [PubMed]
9. Lv, J.; Wang, J.; Shang, X.; Liu, F.; Guo, S. Survival prediction in patients with colon adenocarcinoma via multi-omics data integration using a deep learning algorithm. *Biosci. Rep.* **2020**, *40*, BSR20201482. [CrossRef] [PubMed]
10. Lee, T.Y.; Huang, K.Y.; Chuang, C.H.; Lee, C.Y.; Chang, T.H. Incorporating deep learning and multi-omics autoencoding for analysis of lung adenocarcinoma prognostication. *Comput. Biol.* **2020**, *87*, 107277. [CrossRef]
11. Nam, D.; Chapiro, J.; Paradis, V.; Seraphin, T.P.; Kather, J.N. Artificial intelligence in liver diseases: Improving diagnostics, prognostics and response prediction. *JHEP Rep.* **2022**, *4*, 100443. [CrossRef]
12. Kawka, M.; Dawidziuk, A.; Jiao, L.R.; Gall, T.M.H. Artificial intelligence in the detection, characterisation and prediction of hepatocellular carcinoma: A narrative review. *Transl. Gastroenterol. Hepatol.* **2022**, *7*, 41. [CrossRef]
13. Cohen, J.D.; Li, L.; Wang, Y.; Thoburn, C.; Afsari, B.; Danilova, L.; Douville, C.; Javed, A.A.; Wong, F.; Mattox, A.; et al. Detection and localization of surgically resectable cancers with a multi-analyte blood test. *Science* **2018**, *359*, 926–930. [CrossRef]
14. Song, C.; Li, X. Cost-Sensitive KNN Algorithm for Cancer Prediction Based on Entropy Analysis. *Entropy* **2022**, *24*, 253. [CrossRef] [PubMed]
15. Huang, S.; Cai, N.; Pacheco, P.P.; Narrandes, S.; Wang, Y.; Xu, W. Applications of Support Vector Machine (SVM) Learning in Cancer Genomics. *Cancer Genom. Proteom.* **2018**, *15*, 41–51. [CrossRef]
16. Toth, R.; Schiffmann, H.; Hube-Magg, C.; Büscheck, F.; Höflmayer, D.; Weidemann, S.; Lebok, P.; Fraune, C.; Minner, S.; Schlomm, T.; et al. Random forest-based modelling to detect biomarkers for prostate cancer progression. *Clin. Epigenet.* **2019**, *11*, 148. [CrossRef]
17. Pan, L.Y.; Liu, G.J.; Lin, F.Q.; Zhong, S.L.; Xia, H.M.; Sun, X.; Liang, H.Y. Machine Learning Applications for Prediction of Relapse in Childhood Acute Lymphoblastic Leukemia. *Sci. Rep.* **2017**, *7*, 7402. [CrossRef]
18. Abreu, P.H.; Santos, M.S.; Abreu, M.H.; Andrade, B.; Silva, D.C. Predicting Breast Cancer Recurrence using Machine Learning Techniques: A Systematic Review. *ACM Comput. Surv.* **2017**, *49*, 52. [CrossRef]
19. Savareh, B.A.; Aghdaie, H.A.; Behmanesh, A.; Bashiri, A.; Sadeghi, A.; Zali, M.; Shams, R. A machine learning approach identified a diagnostic model for pancreatic cancer through using circulating microRNA signatures. *Pancreatology* **2020**, *20*, 1195–1204. [CrossRef]
20. Demidova, L.A. A Novel Approach to Decision-Making on Diagnosing Oncological Diseases Using Machine Learning Classifiers Based on Datasets Combining Known and/or New Generated Features of a Different Nature. *Mathematics* **2023**, *11*, 792. [CrossRef]
21. Li, G.; Hu, J.; Hu, G. Biomarker Studies in Early Detection and Prognosis of Breast Cancer. *Adv. Exp. Med. Biol.* **2017**, *1026*, 27–39. [CrossRef]
22. Loke, S.Y.; Lee, A.S.G. The future of blood-based biomarkers for the early detection of breast cancer. *Eur. J. Cancer.* **2018**, *92*, 54–68. [CrossRef] [PubMed]
23. Killock, D. CancerSEEK and destroy—A blood test for early cancer detection. *Nat. Rev. Clin. Oncol.* **2018**, *15*, 133. [CrossRef] [PubMed]
24. Kalinich, M.; Haber, D.A. Cancer detection: Seeking signals in blood. *Science* **2018**, *359*, 866–867. [CrossRef]
25. Mansur, A.; Vrionis, A.; Charles, J.P.; Hancel, K.; Panagides, J.C.; Moloudi, F.; Iqbal, S.; Daye, D. The Role of Artificial Intelligence in the Detection and Implementation of Biomarkers for Hepatocellular Carcinoma: Outlook and Opportunities. *Cancers* **2023**, *15*, 2928. [CrossRef] [PubMed]

26. Hao, Y.; Jing, X.Y.; Sun, Q. Joint learning sample similarity and correlation representation for cancer survival prediction. *BMC Bioinform.* **2022**, *23*, 553. [CrossRef]
27. Núñez, C. Blood-based protein biomarkers in breast cancer. *Clin. Chim. Acta.* **2019**, *490*, 113–127. [CrossRef]
28. Du, Z.; Liu, X.; Wei, X.; Luo, H.; Li, P.; Shi, M.; Guo, B.; Cui, Y.; Su, Z.; Zeng, J.; et al. Quantitative proteomics identifes a plasma multi protein model for detection of hepatocellular carcinoma. *Sci. Rep.* **2020**, *10*, 15552. [CrossRef]
29. Siers, M.J.; Md Zahidul, I. Class Imbalance and Cost-Sensitive Decision Trees: A Unified Survey Based on a Core Similarity. *ACM Trans. Knowl. Discov. Data.* **2020**, *15*, 4. [CrossRef]
30. Rekha, G.; Tyagi, A.K.; Reddy, V.K. A Wide Scale Classification of Class Imbalance Problem and its Solutions: A Systematic Literature Review. *J. Comput. Sci.* **2019**, *15*, 886–929. [CrossRef]
31. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
32. Han, H.; Wang, W.Y.; Mao, B.H. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In *Advances in Intelligent Computing*; Huang, D.S., Zhang, X.P., Huang, G.B., Eds.; ICIC 2005. Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3644, pp. 878–887. [CrossRef]
33. He, H.; Bay, Y.; Garcia, E.A.; Li, S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1–8 June 2008; pp. 1322–1328. [CrossRef]
34. Swana, E.F.; Doorsamy, W.; Bokoro, P. Tomek Link and SMOTE Approaches for Machine Fault Classification with an Imbalanced Dataset. *Sensors* **2022**, *22*, 3246. [CrossRef]
35. Tomek, I. Two modifications of CNN. *IEEE Trans. Syst. Man Cybern.* **1976**, *6*, 769–772. [CrossRef]
36. Ling, C.X.; Sheng, V.S. Cost-Sensitive Learning. In *Encyclopedia of Machine Learning*; Sammut, C., Webb, G.I., Eds.; Springer: Boston, MA, USA, 2011. [CrossRef]
37. Xu, R.; Wang, J.; Zhu, Q.; Zou, C.; Wei, Z.; Wang, H.; Ding, Z.; Meng, M.; Wei, H.; Xia, S.; et al. Integrated models of blood protein and metabolite enhance the diagnostic accuracy for Non-Small Cell Lung Cancer. *Biomark. Res.* **2023**, *11*, 71. [CrossRef]
38. Luan, Y.; Zhong, G.; Li, S.; Wu, W.; Liu, X.; Zhu, D.; Feng, Y.; Zhang, Y.; Duan, C.; Mao, M. A panel of seven protein tumour markers for effective and affordable multi-cancer early detection by artificial intelligence: A large-scale and multicentre case-control study. *EClinicalMedicine* **2023**, *61*, 102041. [CrossRef]
39. Demidova, L.A. Two-stage hybrid data classifiers based on SVM and kNN algorithms. *Symmetry* **2021**, *13*, 615. [CrossRef]
40. Zanin, M.; Zunino, L.; Rosso, O.A.; Papo, D. Permutation Entropy and Its Main Biomedical and Econophysics Applications: A Review. *Entropy* **2012**, *14*, 1553–1577. [CrossRef]
41. Zhang, A.; Yang, B.; Huang, L. Feature Extraction of EEG Signals Using Power Spectral Entropy. In Proceedings of the International Conference on BioMedical Engineering and Informatics, Sanya, China, 27–30 May 2008; Volume 2, pp. 435–439. [CrossRef]
42. Weng, X.; Perry, A.; Maroun, M.; Vuong, L.T. Singular Value Decomposition and Entropy Dimension of Fractals. *arXiv* **2022**, arXiv:2211.12338. [CrossRef]
43. Pincus, S.M. Approximate entropy as a measure of system complexity. *Proc. Natl. Acad. Sci. USA* **1991**, *88*, 2297–2301. [CrossRef]
44. Pincus, S.M.; Gladstone, I.M.; Ehrenkranz, R.A. A regularity statistic for medical data analysis. *J. Clin. Monit. Comput.* **1991**, *7*, 335–345. [CrossRef]
45. Delgado-Bonal, A.; Marshak, A. Approximate Entropy and Sample Entropy: A Comprehensive Tutorial. *Entropy* **2019**, *21*, 541. [CrossRef] [PubMed]
46. Hjorth, B. EEG Analysis Based on Time Domain Properties. *Electroencephalogr. Clin. Neurophysiol.* **1970**, *29*, 306–310. [CrossRef]
47. Galvão, F.; Alarcão, S.M.; Fonseca, M.J. Predicting Exact Valence and Arousal Values from EEG. *Sensors* **2021**, *21*, 3414. [CrossRef]
48. Shi, C.-T. Signal Pattern Recognition Based on Fractal Features and Machine Learning. *Appl. Sci.* **2018**, *8*, 1327. [CrossRef]
49. Bykova, M.O.; Balandin, V.A. Methodological features of the analysis of the fractal dimension of the heart rate. *Russ. Technol. J.* **2023**, *11*, 58–71. [CrossRef]
50. McInnes, L.; Healy, J.; Melville, J. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv* **2018**, arXiv:1802.03426.
51. Becht, E.; McInnes, L.; Healy, J.; Dutertre, C.A.; Kwok, I.W.H.; Ng, L.G.; Ginhoux, F.; Newell, E.W. Dimensionality reduction for visualizing single-cell data using UMAP. *Nat. Biotechnol.* **2019**, *37*, 38–44. [CrossRef] [PubMed]
52. Dorrity, M.W.; Saunders, L.M.; Queitsch, C.; Fields, S.; Trapnell, C. Dimensionality reduction by UMAP to visualize physical and genetic interactions. *Nat. Commun.* **2020**, *11*, 1537. [CrossRef]
53. Demidova, L.A.; Gorchakov, A.V. Fuzzy Information Discrimination Measures and Their Application to Low Dimensional Embedding Construction in the UMAP Algorithm. *J. Imaging* **2022**, *8*, 113. [CrossRef]
54. COSMIC | Catalogue of Somatic Mutations in Cancer. Available online: https://cancer.sanger.ac.uk/cosmic (accessed on 4 January 2023).
55. Thai-Nghe, N.; Gantner, Z.; Schmidt-Thieme, L. Cost-sensitive learning methods for imbalanced data. In Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN), Barcelona, Spain, 18–23 July 2010; pp. 1–8.
56. Cao, P.; Zhao, D.; Zaiane, O. An optimized cost-sensitive SVM for imbalanced data learning. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 280–292.

57. Mienye, I.D.; Sun, Y. Performance analysis of cost-sensitive learning methods with application to imbalanced medical data. *Inform. Med. Unlocked* **2021**, *25*, 100690. [CrossRef]
58. Tsai, H.-H.; Yang, T.-W.; Wong, W.-M.; Chou, C.-F. A Hybrid Approach for Binary Classification of Imbalanced Data. *arXiv* **2022**, arXiv:2207.02738. [CrossRef]
59. Freitas, A.; Brazdil, P.; Costa-Pereira, A. Cost-Sensitive Learning in Medicine. In *Data Mining and Medical Knowledge Management: Cases and Applications*; Berka, P., Rauch, J., Zighed, D.A., Eds.; IGI Global: Hershey, PA, USA, 2009; pp. 57–75. [CrossRef]
60. Gupta, R.; Kleinjans, J.; Caiment, F. Identifying novel transcript biomarkers for hepatocellular carcinoma (HCC) using RNA-Seq datasets and machine learning. *BMC Cancer* **2021**, *21*, 962. [CrossRef] [PubMed]
61. Lee, T.; Rawding, P.A.; Bu, J.; Hyun, S.; Rou, W.; Jeon, H.; Kim, S.; Lee, B.; Kubiatowicz, L.J.; Kim, D.; et al. Machine-Learning-Based Clinical Biomarker Using Cell-Free DNA for Hepatocellular Carcinoma (HCC). *Cancers* **2022**, *14*, 2061. [CrossRef] [PubMed]
62. Sato, M.; Tateishi, R.; Moriyama, M.; Fukumoto, T.; Yamada, T.; Nakagomi, R.; Kinoshita, M.N.; Nakatsuka, T.; Minami, T.; Uchino, K. Machine Learning–Based Personalized Prediction of Hepatocellular Carcinoma Recurrence After Radiofrequency Ablation. *Gastro Hep. Adv.* **2022**, *1*, 29–37. [CrossRef]
63. An, C.; Yang, H.; Yu, X.; Han, Z.Y.; Cheng, Z.; Liu, F.; Dou, J.; Li, B.; Li, Y.; Li, Y.; et al. A Machine Learning Model Based on Health Records for Predicting Recurrence After Microwave Ablation of Hepatocellular Carcinoma. *J. Hepatocell. Carcinoma* **2022**, *9*, 671–684. [CrossRef] [PubMed]
64. Ding, W.; Wang, Z.; Liu, F.Y.; Cheng, Z.G.; Yu, X.; Han, Z.; Zhong, H.; Yu, J.; Liang, P. A Hybrid Machine Learning Model Based on Semantic Information Can Optimize Treatment Decision for Naïve Single 3-5-cm HCC Patients. *Liver Cancer* **2022**, *11*, 256–267. [CrossRef]
65. Hsu, P.Y.; Liang, P.C.; Chang, W.T.; Lu, M.Y.; Wang, W.H.; Chuang, S.C.; Wei, Y.J.; Jang, T.Y.; Yeh, M.L.; Huang, C.I.; et al. Artificial intelligence based on serum biomarkers predicts the efficacy of lenvatinib for unresectable hepatocellular carcinoma. *Am. J. Cancer Res.* **2022**, *12*, 5576–5588.
66. Ge, S.; Xu, C.R.; Li, Y.M.; Zhang, Y.L.; Li, N.; Wang, F.T.; Ding, L.; Niu, J. Identification of the Diagnostic Biomarker VIPR1 in Hepatocellular Carcinoma Based on Machine Learning Algorithm. *J. Oncol.* **2022**, *2022*, 2469592. [CrossRef]
67. Xing, W.; Bei, Y. Medical Health Big Data Classification Based on KNN Classification Algorithm. *IEEE Access* **2020**, *8*, 28808–28819. [CrossRef]
68. Mohanty, S.; Mishra, A.; Saxena, A. Medical Data Analysis Using Machine Learning with KNN. In *International Conference on Innovative Computing and Communications*; Gupta, D., Khanna, A., Bhattacharyya, S., Hassanien, A.E., Anand, S., Jaiswal, A., Eds.; Advances in Intelligent Systems and Computing; Springer: Singapore, 2020; Volume 1166. [CrossRef]
69. Chapelle, O.; Vapnik, V.; Bousquet, O.; Mukherjee, S. Choosing multiple parameters for support vector machines. *Mach. Learn.* **2002**, *46*, 131–159. [CrossRef]
70. Yu, W.; Liu, T.; Valdez, R.; Gwinn, M.; Khoury, M.J. Application of support vector machine modeling for prediction of common diseases: The case of diabetes and pre-diabetes. *BMC Med. Inform. Decis. Mak.* **2010**, *10*, 16. [CrossRef]
71. Schober, P.; Vetter, T.R. Logistic Regression in Medical Research. *Anesth. Analg.* **2021**, *132*, 365–366. [CrossRef] [PubMed]
72. Dai, B.; Chen, R.-C.; Zhu, S.-Z.; Zhang, W.-W. Using Random Forest Algorithm for Breast Cancer Diagnosis. In Proceedings of the 2018 International Symposium on Computer, Consumer and Control (IS3C), Taichung, Taiwan, 6–8 December 2018; pp. 449–452. [CrossRef]
73. Acharjee, A.; Larkman, J.; Xu, Y.; Cardoso, V.R.; Gkoutos, G.V. A random forest based biomarker discovery and power analysis framework for diagnostics research. *BMC Med. Genom.* **2020**, *13*, 178. [CrossRef] [PubMed]
74. Cheng, S.; Liu, B.; Ting, T.O.; Qin, Q.; Shi, Y.; Huang, K. Survey on data science with population-based algorithms. *Big Data Anal.* **2016**, *1*, 3. [CrossRef]
75. Liu, J.-Y.; Jia, B.-B. Combining One-vs-One Decomposition and Instance-Based Learning for Multi-Class Classification. *IEEE Access* **2020**, *8*, 197499–197507. [CrossRef]
76. Grandini, M.; Bagli, E.; Visani, G. Metrics for Multi-class Classification: An Overview. *arXiv* **2020**, arXiv:2008.05756.
77. Van der Maaten, L.; Hinton, G.E. Visualizing Data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
78. Dong, W.; Moses, C.; Li, K. Efficient k-nearest neighbor graph construction for generic similarity measures. In Proceedings of the 20th International Conference on World Wide Web, Hyderabad, India, 28 March–1 April 2011; pp. 577–586.
79. Damrich, S.; Hamprecht, F.A. On UMAP's true loss function. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 12.
80. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. Available online: https://umap-learn.readthedocs.io/en/latest/_modules/umap/umap_.html (accessed on 4 January 2024).
81. Bottou, L.; Chapelle, O.; DeCoste, D.; Weston, J. *Support Vector Machine Solvers Large-Scale Kernel Machines*; MIT Press: Cambridge, MA, USA, 2007; pp. 1–27.
82. Tsang, I.W.; Kwok, J.T.; Cheung, P.-M. Core Vector Machines: Fast SVM Training on Very Large Data Sets. *J. Mach. Learn. Res.* **2005**, *6*, 363–392.
83. umap. Available online: https://github.com/lmcinnes/umap/issues/8 (accessed on 4 January 2024).
84. Tomčala, J. New Fast ApEn and SampEn Entropy Algorithms Implementation and Their Application to Supercomputer Power Consumption. *Entropy* **2020**, *22*, 863. [CrossRef]
85. Batu, T.; Dasgupta, S.; Kumar, R.; Rubinfeld, R. The complexity of approximating the entropy. In Proceedings of the 17th IEEE Annual Conference on Computational Complexity, Montreal, QC, Canada, 21–24 May 2002; pp. 17–26. [CrossRef]

86. Platt, J. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods—Support Vector Learning*; Schölkopf, B., Burges, C.J., Smola, A.J., Eds.; MIT Press: Cambridge, MA, USA, 1999; pp. 185–208.
87. Collobert, R.; Bengio, S.; Bengio, Y. A parallel mixture of SVMs for very large scale problems. *Neural Comput.* **2002**, *14*, 1105–1114. [CrossRef]
88. Shalev-Shwartz, S.; Singer, Y.; Srebro, N. Pegasos: Primal estimated sub-gradient solver for SVM. *Math. Program.* **2011**, *127*, 3–30. [CrossRef]
89. Gentinetta, G.; Thomsen, A.; Sutter, D.; Woerner, S. The complexity of quantum support vector machines. *Quantum* **2024**, *8*, 1225. [CrossRef]
90. Prusty, S.; Patnaik, S.; Dash, S.K. SKCV: Stratified K-fold cross-validation on ML classifiers for predicting cervical cancer. *Front. Nanotechnol.* **2022**, *4*, 972421. [CrossRef]
91. Slamet, W.; Herlambang, B.; Samudi, S. Stratified K-fold cross validation optimization on machine learning for prediction. *Sink. J. Dan Penelit. Tek. Inform.* **2022**, *7*, 2407–2414. [CrossRef]
92. umap-losses. Available online: https://github.com/worldbeater/umap-losses (accessed on 4 January 2024).
93. Numba: A High Performance Python Compiler. Available online: https://numba.pydata.org/ (accessed on 4 January 2024).
94. Demšar, J. Statistical Comparisons of Classifiers over Multiple Data Sets. *J. Mach. Learn. Res.* **2006**, *7*, 1–30.
95. Gorchakov, A.V.; Demidova, L.A.; Sovietov, P.N. Analysis of Program Representations Based on Abstract Syntax Trees and Higher-Order Markov Chains for Source Code Classification Task. *Future Internet* **2023**, *15*, 314. [CrossRef]
96. Amid, E.; Warmuth, M.K. TriMap: Large-scale Dimensionality Reduction Using Triplets. *arXiv* **2019**, arXiv:1910.00204v2. [CrossRef]
97. Wang, Y.; Huang, H.; Rudin, C.; Shaposhnik, Y. Understanding How Dimension Reduction Tools Work: An Empirical Approach to Deciphering t-SNE, UMAP, TriMap, and PaCMAP for Data Visualization. *J. Mach. Learn. Res.* **2021**, *22*, 9129–9201.

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

mathematics

MDPI

Article

# Allocation of Starting Points in Global Optimization Problems

**Oleg Khamisov [1,*], Eugene Semenkin [2,*] and Vladimir Nelyub [2]**

[1] Department of Applied Mathematics, Melentiev Energy Systems Institute, Lermontov St. 130, 664033 Irkutsk, Russia

[2] Scientific and Educational Center "Artificial Intelligence Technologies", Bauman Moscow State Technical University, 2nd Baumanskaya St., 5, 105005 Moscow, Russia; vladimir.nelub@emtc.ru

[*] Correspondence: khamisov@isem.irk.ru (O.K.); e.semenkin@emtc.ru (E.S.)

**Abstract:** We propose new multistart techniques for finding good local solutions in global optimization problems. The objective function is assumed to be differentiable, and the feasible set is a convex compact set. The techniques are based on finding maximum distant points on the feasible set. A special global optimization problem is used to determine the maximum distant points. Preliminary computational results are given.

**Keywords:** multistart; maximum distant points; multiple local minima

**MSC:** 49M37; 90C30; 65K05

## 1. Introduction

Within the concept of a "smart" digital environment, methods of mathematical modeling and machine learning are actively used to design and implement digital twins of complex technical, technological, and organizational systems. In this case, it is usually necessary to solve complex global optimization problems to automate the selection of effective structures and parameters of the corresponding models of these digital twins. The effectiveness of global optimization methods depends significantly on the choice of the initial set of solutions, which are subsequently used to find the global optimum or a good local optimum that approximates the global one. This is especially important when using global optimization methods for the continuously differentiable functions of real variables, because in this case, it is possible to obtain optimal solutions guaranteed by the strict mathematical apparatus of applied mathematics.

Let a differentiable function $f\colon \mathbb{R}^n \to \mathbb{R}$ and a convex compact set $X \subset \mathbb{R}^n$ with a nonempty interior, $\mathbf{int}(X) \neq \varnothing$, be given. The problem considered in this paper consists in finding a good local minimum using the multistart strategy. In order achieve this, it is necessary to allocate $p$ starting points $x^1, \ldots, x^p$ in $X$, such that they cover $X$ "more or less uniformly". The proposed multistart strategy is based on the CONOPT solver [1].

Various uniform sampling procedures can be used for this purpose. A survey of special methods for allocation points on spheres is presented in [2]. If $X$ is a polytope, sampling based on simplicial decomposition of $X$ is applied, as given in [3]. In [4], a class of Markov chain Monte Carlo (MCMC) algorithms for distribution points on polytopes is described. In a more general case, when $X$ is a convex body, a random walk strategy [5] based on the MCMC technique is successfully applied. A brief review of different kinds of random walk can be found in [4]. However, uniform random sampling algorithms are of exponential complexity [6]. Uniform sampling is usually used for the approximate calculation of an integral or volume of $X$. We are interested in finding a good local solution in global optimization problems. The most attractive feature of uniform sampling consists in the following: a global minimum solution can be found with a probability of one as the length of the sampling tends to infinity. However, due to the specifics of high-dimensional

**Citation:** Khamisov, O.; Semenkin, E.; Nelyub, V. Allocation of Starting Points in Global Optimization Problems. *Mathematics* **2024**, *12*, 606. https://doi.org/10.3390/math12040606

Academic Editor: Liliya Demidova

Received: 18 December 2023
Revised: 11 February 2024
Accepted: 12 February 2024
Published: 18 February 2024

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

spaces [7], random sampling is not efficient from a practical point of view. Nevertheless, uniform sampling continues to draw attention, and investigations on this topic are of serious interest [8]. Approaches based on the $p$-location problem [9] and $p$-center methodology [10] can also be used for solving the problems considered in our paper. However, we aimed to check the efficiency of a global optimization approach.

In our paper, we propose a procedure for the good allocation of points on a convex compact set $X$. The idea is to use a special global optimization problem as an auxiliary one for allocation. The special global optimization problem consists in maximizing the Euclidean norm plus a linear term over a convex compact set. Because of the particular form of the problem, it can be solved to global optimality for a sufficiently large number of variables, for example, for $n \sim 30 - 50$. In doing so, we achieve a better covering of set $X$ by a family of points. We believe that a combination of the proposed approach and advanced metaheuristics [11] will be of serious practical importance.

**The first approach.** The most attractive statement of the problem can be formalized as follows:

$$t \to \max,\ t = \|x^i - x^j\|^2,\ \ x^i, x^j \in X,\ \ 1 \le i < j \le p. \tag{1}$$

Problem (1) means that it is necessary to allocate $p$ points such that the distance between any two points is the same and is as maximal as possible. In this case, the set $\{x^1, \dots, x^p\}$ is called the set of equidistant points . However, it is well known that Problem (1) is solvable only if $p \le n+1$. When $p = n+1$, then points $\{x^1, \dots, x^{n+1}\}$ are vertices of a regular simplex. If $\|x^i - x^j\| = \delta,\ 1 \le i < j \le n+1$, all points $x^i$ belong to the sphere of radius

$$R = \delta\sqrt{\frac{n}{2(n+1)}} \tag{2}$$

centered at

$$x^c = \frac{1}{n+1}\sum_{j=1}^{n+1} x^j.$$

However, in many applications, it is necessary to allocate more than $n+1$ points.

**The second approach.** We move to another problem of the following form:

$$\min_{1 \le i < j \le p}\{\|x^i - x^j\|^2\} \to \max,\ x^i, x^j \in X. \tag{3}$$

We want to allocate $p$ points such that the minimum distance between any two of them is as maximal as possible. Problem (3) always has a solution since the objective function is continuous and the feasible set is nonempty and compact. The objective function is nonsmooth, but this can be avoided by the standard reduction of Problem (3) to the following one:

$$t \to \max,\ t \le \|x^i - x^j\|^2,\ x^i, x^j \in X,\ i, j = 1, \dots, p,\ j > i. \tag{4}$$

Two main difficulties are unavoidable when solving Problem (4). Firstly, the number of variables is equal to $\frac{p(p-1)n}{2}$. Secondly, the feasible domain is nonconvex. Hence, we have to overcome the nonconvexity of the feasible domain, but we are seriously restricted in dimension $n$.

**The third approach.** Given $p-1$ points $v^i \in X$, find point $v^p$ as a solution to the problem

$$\varphi_p(x) = \min_{1 \le j \le p-1}\{\|x - v^j\|^2\} \to \max,\ x \in X. \tag{5}$$

As a result, set $X$ is covered by $p$ balls centered at $v^1, \dots, v^p$ with radius $r_p$ equal to $\sqrt{\varphi_p(v^p)}$. We start from an arbitrary point $v^1 \in X$ and sequentially determine points $v^2, v^3, \dots$ and functions $\varphi_2, \varphi_3, \dots$ according to (5). Let $\theta(x) = 0\ \forall x \in X$ be identical a zero function on $X$.

The theoretical foundation of the approach based on solving Problem (5) is given by the following theorem.

**Theorem 1.** *The sequence of functions $\varphi_p,\ p = 2, 3, \dots$ uniformly converges to function $\theta$ over $X$.*

**Proof.** Functions $\varphi_p,\ p = 2, \dots$ are Lipschitz functions with the same Lipschitz constant. Therefore, $\varphi_p,\ p = 2, \dots$ is an equicontinuous sequence of functions. Since $X$ is a compact set, then $\varphi_p(x) \le D(X) < +\infty$, where $D(X)$ is the diameter of $X$, and functions $\varphi_p$, $p = 2, \dots$ are uniformly bounded. By construction $\varphi_p(x) \le \varphi_{p-1}(x)\ \forall x \in X$. Hence, due to the Arzelà–Ascoli theorem, $\varphi_p,\ p = 2, \dots$ is a sequence of functions uniformly convergent to a continuous function $\eta : \eta(x) \le \varphi_p(x)\ \forall x \in X,\ p = 2, \dots$. By construction $\varphi_p(v^i) = 0\ \forall i < p$; hence,

$$\eta(v^p) = 0\ \forall p. \tag{6}$$

Assume that $\lim\limits_{p\to\infty} \varphi_p(v^p) = \rho > 0$. Let $v^{p_j},\ j = 1, 2, \dots$ be a subsequence convergent to a point $v^\sharp$ such that $\eta(v^\sharp) = \rho$. From (6), due to the continuity of $\eta$, we have $\lim\limits_{j\to\infty} \eta(v^{p_j}) = \eta(v^\sharp) = 0$, a contradiction, which proves the theorem. □

Hence, we can theoretically achieve the covering of $X$ by a number of balls with sufficiently small radius. In practice, especially in high dimensions we restrict ourselves to a reasonable value of $p$.

Let us rewrite Problem (5) in a more computationally tractable form. Point $v^p$ is the maximum distant point from points $v^j, j = 1, \dots, p-1$. Since $\|x - v^j\|^2 = \|x\|^2 - 2x^\top v^j + \|v^j\|^2$ and $\min\limits_{1\le j\le p-1}\{\|x\|^2 - 2x^\top v^j + \|v^j\|^2\} = \|x\|^2 + \min\limits_{1\le j\le p-1}\{\|v^j\|^2 - 2x^\top v^j\}$, we can rewrite Problem (5) in the form

$$\|x\|^2 + t \to \max,\ t \le \|v^j\|^2 - 2x^\top v^j,\ j = 1, \dots, p-1,\ x \in X. \tag{7}$$

The feasible domain in (7) is convex, and the objective function is convex. Therefore, we have a convex maximization problem, and special advanced methods [12] can be used for solving (7).

In our paper, we develop the iterative scheme of the third approach based on solving problems of type (7). The description is the following. Take an arbitrary first point $v^1$. The other points are determined according to the solutions to problem (7) for $p = 2, 3, \dots$. Points are found sequentially: the new point is determined after finding the previous ones. This is why we call points $v^1, v^2, \dots, v^p$ obtained on the base of the iterative solution of problem (7) sequentially maximum distant pointsor simply sequentially distant points .
Notation:
$e^j,\ j = 1, \dots, n$ are unit vectors with 1 on the $j$-th position and 0 on the others;
$x_j$ is the $j$-th component of vector $x \in \mathbb{R}^n$;
$x^i$ is the $i$-th vector in a sequence of $n$-dimensional vectors $x^1, \dots, x^i, \dots$;
$x^\top y$ is the dot (inner) product of vectors $x, y \in \mathbb{R}^n$.

## 2. Allocation of Points in the Unit Ball

Assume that $X$ is the unit ball, that is,

$$X = B = \{x \in \mathbb{R}^n \colon \|x\|^2 \le 1\}.$$

In this case, Problem (5) can be solved analytically. The obtained points are called ball sequentially distant points . We start with the problem of setting the $n+1$ equidistant point in $B$ that is equivalent to inscribing a regular simplex in $B$. The distance between points can be determined from (2) with $R = 1$,

$$\delta = \sqrt{\frac{2(n+1)}{n}} = \sqrt{2}\sqrt{1 + \frac{1}{n}}. \tag{8}$$

Since the points are equidistant:

$$\|x^i - x^j\|^2 = \|x^i - x^k\|^2 \quad \Leftrightarrow \quad (x^k - x^j)^\top x^i = 0,\ 1 \le i < j < k \le n+1.$$

Due to the symmetry of $B$, we can set $x^1 = e^1 = (1, 0, \dots, 0)^\top$. Then, from (2),

$$x_1^j = x_1^k,\ 2 \le j < k \le n+1. \tag{9}$$

Since points $x^j,\ j = 2, \dots, n+1$ belong to the intersection of a plane orthogonal to $x^1$ and a boundary of $B$, we also can choose the point $x^2$ as a point with maximal zero components. Therefore, we set $x_l^2 = 0,\ l = 3, \dots, n$. The distance $\|x^1 - x^j\|^2 = (1 - x_1^2)^2 + (x_2^2)^2 = \delta^2$, and $(x_1^2)^2 + (x_2^2)^2 = 1$. From these two equations and (9), we obtain $x_1^j = -\frac{1}{n},\ j = 2, \dots,\ x_2^2 = \sqrt{\frac{(n-1)(n+1)}{n^2}}$. Now, let us repeat the same consideration for the $n-1$-dimensional ball centered at $x^2$ and obtained as an intersection of the plane $\{x \in \mathbb{R}^n : x_1 = -\frac{1}{n}\}$ and $B$. Then, we determine $x^3 = \left(-\frac{1}{n}, -\sqrt{\frac{n+1}{n} \cdot \frac{1}{n(n-1)}}, \sqrt{\frac{n+1}{n} \cdot \frac{n-2}{n}}, 0, \dots, 0\right)$. After repeating this consideration similarly for the remaining cases, we obtain the final description of the equidistant point in the unit ball:

$$x_j^k = \begin{cases} -\sqrt{\frac{n+1}{n} \cdot \frac{1}{(n-j+2)(n-j+1)}}, & 1 \le j < k, \\ \sqrt{\frac{n+1}{n} \cdot \frac{n-k+1}{n-k+2}}, & j = k, \\ 0, & k < j \le n, \end{cases} \quad ,\ k = 1, \dots, n+1. \tag{10}$$

Let us switch now to the construction of the sequentially maximum distant points. Again, due to the symmetry of $B$, the starting point $v^1 = e^1$. The next point, which is denoted by $v^{n+1}$, is determined as $v^{n+1} = \arg\max\{\|x - v^1\|^2 \colon x \in B\} = -e^1$. Point $v^2$ is a solution to the problem

$$\min\{\|x - v^1\|^2, \|x - v^{n+1}\|^2\} \to \max,\ \ x \in B. \tag{11}$$

Let us introduce the sets

$$X_{21} = \{x \in B \colon \|x - v^1\|^2 \le \|x - v^{n+1}\|\} = \{x \in B \colon x_1 \ge 0\},$$

$$X_{22} = \{x \in B \colon \|x - v^{n+1}\|^2 \le \|x - v^1\|\} = \{x \in B \colon x_1 \le 0\}.$$

Then, solving Problem (11) is reduced to solving the following two problems:

$$f_{21}(x) = \|x - v^1\|^2 \to \max,\ x \in X_{21} \tag{12}$$

and

$$f_{22}(x) = \|x - v^{n+1}\|^2 \to \max,\ x \in X_{22}. \tag{13}$$

Since $f_{21}(x) = \|x\|^2 - 2x_1 + 1 \le 2 - 2x_1\ \forall x \in B$, the upper bound for the maximum value in (12) is given by $\max\{2 - 2x_1 \colon x \in X_{21}\} = 2$ and is achieved, for example, at point $e^2$. The value $f_{21}(e^2) = 2$. Therefore, $e^2$ is a solution to problem (12). Similarly, $f_{22}(x) = \|x\|^2 + 2x_1 + 1 \le 2 + 2x_1\ \forall x \in B$, the upper bound $\max\{2 + 2x_1 \colon x \in X_{22}\} = 2$ is also achieved at $e^2$ and $f_{22}(e^2) = 2$. Hence, point $e^2$ is a solution to problem (13). The latter means that $e^2$ is a solution to Problem (11), and we can set $v^2 = e^2$.

Consider now the problem

$$\min\{\|x - v^1\|^2, \|x - v^2\|^2, \|x - v^{n+1}\|^2\} \to \max,\ x \in B. \tag{14}$$

Determine sets

$$X_{31} = \{x \in B\colon \|x - v^1\|^2 \leq \|x - v^2\|^2, \|x - v^1\|^2 \leq \|x - v^{n+1}\|^2\} =$$
$$= \{x \in B\colon -x_1 + x_2 \leq 0, x_1 \geq 0\},$$
$$X_{32} = \{x \in B\colon \|x - v^2\|^2 \leq \|x - v^1\|^2, \|x - v^2\|^2 \leq \|x - v^{n+1}\|^2\} = \{x \in B\colon x_2 \geq |x_1|\},$$
$$X_{33} = \{x \in B\colon \|x - v^{n+1}\|^2 \leq \|x - v^1\|^2, \|x - v^{n+1}\|^2 \leq \|x - v^2\|^2\} =$$
$$= \{x \in B\colon x_1 + x_2 \leq 0, x_1 \leq 0\}.$$

Problem (14) is reduced to find solutions to the three auxiliary problems

$$f_{3i}(x) = \|x - v^i\|^2 \to \max,\ x \in X_{3i},\ i = 1,2,$$

$$f_{33}(x) = \|x - v^{n+1}\|^2 \to \max,\ x \in X_{33}.$$

Again, $f_{31}(x) \leq 2 - 2x_1\ \forall x \in X_{31}$ and $f_{33}(x) \leq 2 + 2x_1\ \forall x \in X_{33}$. In both cases, the maximum value 2 is attained at the point $-e^2$. For the last auxiliary problem, we have $f_{32}(x) \leq 2 - 2x_2\ \forall x \in X_{32}$, that is, the corresponding maximum value cannot be greater than 2. Therefore, point $v^{n+2} = -e^2$ is a solution to Problem (14).

So far, four points $v^i = e^i,\ v^{n+i} = -e^i,\ i = 1,2$ are obtained. We are going to prove by induction that the same principle is true for $2n$ points: $v^i = e^i,\ v^{n+i} = -e^i,\ i = 1,\ldots,n$. The basis of induction: the hypothesis is true for $k = 2$. The induction step: let us prove that the hypothesis is true for the case $k+1$. Consider the problem

$$\min_{1\leq i\leq k} \{\|x - v^i\|^2, \|x - v^{n+i}\|^2\} \to \max,\ x \in B. \tag{15}$$

Define for $i \in K = \{1,\ldots,k\}$ the following sets

$$X_{k+1,i} = \{x \in B : \|x - v^i\|^2 \leq \|x - v^j\|^2, j \in K \setminus \{i\},\ \|x - v^i\|^2 \leq \|x - v^{n+j}\|^2, j \in K\},$$

$$X_{k+1,n+i} = \{x \in B : \|x - v^{n+i}\|^2 \leq \|x - v^j\|^2, j \in K,\ \|x - v^{n+i}\|^2 \leq \|x - v^{n+j}\|^2, K \setminus \{i\}\}.$$

Then, Problem (15) disintegrates into $2k$ problems

$$f_{k+1,i} = \|x - v^i\|^2 \to \max,\ x \in X_{k+1,i} = \{x \in B\colon x_i \geq 0, x_i \geq |x_j|,\ j \in K \setminus \{i\}\}, \tag{16}$$

$$f_{k+1,n+i} = \|x - v^{n+i}\|^2 \to \max,\ x \in X_{k+1,n+i} = \{x \in B\colon x_i \leq 0, x_i \leq -|x_j|,\ j \in K \setminus \{i\}\}. \tag{17}$$

As above, $f_{k+1,i}(x) \leq 2 - 2x_i \leq 2\quad \forall x \in X_{k+1,i},\ i \in K,\ f_{k+1,i}(e^{k+1}) = 2$ and $e^{k+1} \in X_{k+1,i}\ i \in K$. Similarly, $f_{k+1,n+i}(x) \leq 2 + 2x_i \leq 2\ \forall x \in X_{k+1,n+i},\ i \in K$. Therefore, we can take $e^{k+1}$ as a solution to Problem (15) and set $v^{k+1} = e^{k+1}$.

Let us consider now the next problem:

$$f_{k+2}(x) = \min\left\{\min_{1\leq j\leq k+1} \|x - v^j\|^2, \min_{1\leq j\leq k} \|x - v^{n+j}\|^2\right\} \to \max,\ x \in B. \tag{18}$$

Using the same arguments as earlier, it is easy now to see that $f_{k+2}(x) \leq 2\ \forall x \in B$ and $f_{k+2}(-e^{k+1}) = 2$. Hence, we can accept $-e^{k+1}$ as a solution to (18) and set $v^{n+k+1} = -e^{k+1}$.

Therefore, the first $2n$ points are determined as

$$v^i = e^i,\ v^{n+i} = -e^i,\ i = 1,\ldots,n. \tag{19}$$

The maximum distance between any two points in (19) is equal to 2, and the minimum distance between any two points is equal to $\sqrt{2}$.

Let us now determine point $v^{2n+1}$. In order to do this, we have to solve the problem

$$f_{2n+1}(x) = \min\{\min_{1\le j\le n} \|x - v^j\|^2, \min_{1\le j\le n} \|x - v^{n+j}\|^2\} \to \max,\ x \in B. \tag{20}$$

Rewrite $f$ as follows:

$$f_{2n+1}(x) = \min_{1\le j\le n} \Big\{\min\{\|x - v^j\|^2, \|x - v^{n+j}\|^2\}\Big\} = \min_{1\le j\le n} \Big\{\|x\|^2 + 1 - 2|x_j|\Big\} =$$

$$= \|x\|^2 + 1 - 2 \max_{1\le j\le n} |x_j| = \|x\|^2 + 1 - 2\|x\|_\infty \le \|x\|_1\|x\|_\infty + 1 - 2\|x\|_\infty =$$

$$= (\|x\|_1 - 2)\|x\|_\infty + 1. \tag{21}$$

The maximal value of the expression in (21) over $B$ is obviously equal to 1 and is achieved at the origin $\mathbf{0} = (0, \dots, 0)^\top$. From (19) and (20), we have $f_{2n+1}(\mathbf{0}) = 1$; hence, $v^{2n+1} = \mathbf{0}$. The maximum distance between any two points in the set $\{v^i, v^{n+i}, i = 1, \dots, n, v^{2n+1}\}$ is equal to $\sqrt{2}$, and the minimum distance is equal to 1.

The solution to the problem

$$f_{2n+2}(x) = \min\{\min_{1\le j\le n} \|x - v^j\|^2, \min_{1\le j\le n} \|x - v^{n+j}\|^2, \|x\|^2\} \to \max,\ x \in B.$$

is given by the point $v^{2n+2} = (\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}, \dots, \frac{1}{\sqrt{n}})^\top$, since $f_{2n+2}(v^{2n+2}) = 1$ and $f_{2n+2}(x) \le 1\ \forall x \in B$. Due to the symmetricity of $B$, the next $2^n - 1$ points are other vertices of cube $\tilde{C} = \{x \in \mathbb{R}^n : -\frac{1}{\sqrt{n}} \le x_j \le \frac{1}{\sqrt{n}}, j = 1, \dots, n\}$.

Finally, sequentially distant $2n + 1 + 2^n$ points for the unit ball are given by

$$v^i = e^i,\ v^{n+i} = -e^i,\ i = 1, \dots, n,\ v^{2n+1} = (0, \dots, 0)^\top, \tag{22}$$

$$v^{2n+1+i},\ i = 1, \dots, 2^n,\ \text{ are vertices of the cube }\ \tilde{C}. \tag{23}$$

The maximum distance between any two points is obviously equal to 1. Due to the symmetricity of the ball, the minimum distance can be determined as the distance between $v^{2n+2}$ and any point $v^j,\ j = 1, \dots, n$. For example, $\|v^{2n+2} - v^1\| = \sqrt{2}\sqrt{1 - \frac{1}{\sqrt{n}}}$. Points in (22) and (23) are calculated without solving the corresponding optimization problems.

The above procedures can be generalized for the allocation of points in a general ball $B(x^c, R) = \{x \in \mathbb{R}^n : \|x - x^c\| \le R\}$.

Case A. Generalization of the $n + 1$ equidistant points. We add the center $x^c$ to the set of points and obtain the following $n + 2$ ball sequentially distant points $v^1, \dots, v^{n+2}$ with (10)

$$v_j^k = \begin{cases} x_j^c - R\sqrt{\frac{n+1}{n} \cdot \frac{1}{(n-j+2)(n-j+1)}}, & 1 \le j < k, \\ x_j^c + R\ \sqrt{\frac{n+1}{n} \cdot \frac{n-k+1}{n-k+2}}, & j = k, \\ x_j^c, & k < j \le n, \end{cases} \quad ,\ k = 1, \dots, n+1, \tag{24}$$

$$v^{n+2} = x^c. \tag{25}$$

The obtained points are not equidistant. The maximum distance between any two points is equal to $R$, and the minimum distance is equal to $R\sqrt{2}\sqrt{1 + \frac{1}{n}}$ (see (8)).

Case B. Ball sequentially distant $2n+1$ points. These points are just a direct generalization of (22),

$$v^i = x^c + Re^i,\ v^{n+i} = x^c - Re^i,\ i = 1, \dots, n,\ v^{2n+1} = x^c. \tag{26}$$

The maximum distance is equal to $R$, and the minimum distance is equal to $R\sqrt{2}$.

Case C. Ball sequentially distant $2^n + 2n + 1$ points. Introduce cube $\hat{C} = \{x \in \mathbb{R}^n : x_j^c - R \le x_j \le x_j^c + R,\ j = 1, \dots, n\}$. Then, the points are determined as follows:

$$v^i = x^c + Re^i,\ v^{n+i} = x^c - Re^i,\ i = 1, \dots, n,\ v^{2n+1} = x^c, \tag{27}$$

$$v^{2n+1+i},\ i = 1, \dots, 2^n, \text{ are vertices of the cube } \hat{C}. \tag{28}$$

The maximum distance between any two points is equal to $R$, and the minimum distance is equal to $R\sqrt{2}\sqrt{1 - \frac{1}{\sqrt{n}}}$.

Let us compare the allocation of a ball sequentially $2n+1$ from (26) without the center $v^{2n+1}$ and with a uniform distribution over a unit sphere. We take the minimum distance between two points as a measure of allocation efficiency: the greater minimum distance, the better the allocation. The uniform distribution over the unit sphere is obtained using normal distribution with mean 0 and standard deviation 1 by normalization. The minimum distance between two ball sequentially distant points is $\sqrt{2} \approx 1.414$ for any $n$. If we uniformly distribute 200 points over the unit sphere in a 100-dimensional case, then the minimum distance is on average 1.098 (after 10 repetitions). Therefore, the ball sequentially distant points allocation is almost 40% better than the uniform allocation.

## 3. Mapping the Ball Sequentially Distant Points on a Compact Convex Set

Let $X$ be a convex compact set defined by a system of inequalities

$$X = \{x \in \mathbb{R}^n : g_i(x) \le 0,\ i = 1, \dots, m\},$$

$g_i,\ i = 1, \dots, m$ are convex and twice continuously differentiable functions, and $\mathbf{int}(X) \neq \varnothing$. We use the concept of an analytical center $x^a$ [13]. The point $x^a$ is the solution to the convex optimization problem

$$F(x) \to \max,\ x \in X, \tag{29}$$

$F(x) = \sum\limits_{i=1}^{m} \ln(-g_i(x))$, and $F$ is a twice continuously differentiable concave function. Since $\mathbf{int}(X) \neq \varnothing$, we have $g_i(x^a) < 0,\ i = 1, \dots, m$, so the following ellipsoid can be defined:

$$E = \{x \in \mathbb{R}^n : (x - x^a)^\top H (x - x^a) \le 1\}, \tag{30}$$

$$H = -\nabla^2 F(x^a) = \sum_{i=1}^{m} \left( \frac{1}{g_i^2(x^a)} \nabla g_i(x^a) \nabla g_i(x^a)^\top - \frac{1}{g_i(x^a)} \nabla^2 g_i(x^a) \right).$$

Then, $X \supset E$. The Hessian $H$ can be represented as $H = U^\top \Lambda U$, $U$ is an $n \times n$ orthonormal matrix with eigenvectors of $H$ as columns, and $\Lambda$ is an $n \times n$ diagonal matrix with eigenvalues $\lambda_i > 0,\ i = 1, \dots, n$ on the main diagonal. Let us introduce new variables $y = \Lambda^{\frac{1}{2}} U (x - x^a)$. Then, in variables $y$, ellipsoid $E$ in (30) is the unit ball $B = \{y \in \mathbb{R}^n : y^\top y \le 1\}$. Let $\{v^i,\ i = 1, \dots, N\}$ be ball sequentially distant points in $y$-space constructed in correspondence to the cases A ($N = n+2$), B ($N = 2n+1$) or C ($N = 2^n + 2n + 1$) from the previous section. In the $x$-space, we define points

$$w^i = x^a + U^\top \Lambda^{-\frac{1}{2}} v^i,\ i = 1, \dots, N. \tag{31}$$

Images $w^i$ of the ball equidistant points ($i = 1, \dots, n+1$) are solutions to the problem

$$t \to \max, \;\; t = (x^i - x^j)^\top H(x^i - x^j), \; x^i, x^j \in E, \quad 1 \le i < j \le n+1.$$

Images $w^i$ of the ball sequentially distant points (cases D or C, $i = 1, \dots, N$, $n = 2n+1$ or $N = 2^n + 2n + 1$) are solutions to the problem

$$\min_{1 \le j \le i-1} \{(x - w^j)^\top H(x - w^j)\} \to \max, \; x^j \in E.$$

**Example 1.** *Consider the following problem:*

$$f(x_1, x_2) = -\frac{1 + \cos(12\sqrt{(x_1 - 0.7)^2 + (x_2 - 3)^2})}{0.5((x_1 - 0.7)^2 + (x_2 - 3)^2) + 2} \to \min, \; x \in X,$$

$$X = \{x \in \mathbb{R}^2 : g_1(x) = x_1^2 - x_2 \le 0, g_2(x) = -x_1 + 3x_2 - 10 \le 0, g_3(x) = -7x_1 + x_2 \le 0\},$$

*$f$ is the shifted drop-wave function [14], and global minimum $x^* = (0.7, 3.0)$, $f(x^*) = -1$. After solving the corresponding problem* (29)*, we determine the analytical center $x^a = (0.982, 2.125)^\top$ and matrices*

$$H = -\nabla^2 F(x^a) = \begin{pmatrix} 6.806 & -1.909 \\ -1.909 & 1.210 \end{pmatrix}, \; U = \begin{pmatrix} -0.956 & -0.296 \\ 0.296 & -0.956 \end{pmatrix}, \; \Lambda = \begin{pmatrix} 7.395 & 0 \\ 0 & 0.621 \end{pmatrix}.$$

We use Case C from the previous section, so $N = 2^n + 2n + 1 = 9$ for $n = 2$. Points $v^i, \; i = 1, \dots, 9$ are determined in (27) and (28) with $R = 1$, points $w^i = x^a + U^\top \Lambda^{-\frac{1}{2}} v^i$, $i = 1, \dots 9$, points $x^{*,i}$ are stationary points determined by the CONOPT solver [1] starting from points $w^i$, and $f^{*,i} = f(x^{*,i})$ are the corresponding objective function values (see Table 1).

**Table 1.** Starting and stationary points in Example 1.

| $i$ | $v^i$ | $w^i$ | $x^{*,i}$ | $f^{*,i}$ |
|---|---|---|---|---|
| 1 | $(\;1, \;\;0)^\top$ | $(0.631, 2.233)^\top$ | $(0.700, 3.000)^\top$ | $-1.000$ |
| 2 | $(-1, \;\;0)^\top$ | $(1.333, 2.016)^\top$ | $(1.256, 1.665)^\top$ | $-0.656$ |
| 3 | $(\;0, \;\;1)^\top$ | $(0.607, 0.912)^\top$ | $(1.804, 3.935)^\top$ | $-0.656$ |
| 4 | $(\;0, -1)^\top$ | $(1.356, 3.337)^\top$ | $(0.231, 1.452)^\top$ | $-0.605$ |
| 5 | $(\;0, \;\;0)^\top$ | $(0.982, 2.125)^\top$ | $(1.227, 2.560)^\top$ | $-0.885$ |
| 6 | $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})^\top$ | $(0.469, 1.344)^\top$ | $(1.358, 2.218)^\top$ | $-0.793$ |
| 7 | $(-\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}})^\top$ | $(1.495, 2.905)^\top$ | $(0.700, 3.000)^\top$ | $-1.000$ |
| 8 | $(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})^\top$ | $(0.965, 1.190)^\top$ | $(1.357, 2.702)^\top$ | $-0.885$ |
| 9 | $(\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}})^\top$ | $(0.998, 3.058)^\top$ | $(0.700, 3.000)^\top$ | $-1.000$ |

We can see from Table 1 that the global minimum point was determined three times. In the other six cases, different stationary points were found with two points $x^{*,2}$ and $x^{*,3}$ with the same value $-0.656$, and two points $x^{*,5}$ and $x^{*,8}$ with the value $-0.885$.

Geometrical interpretation of points $w^i, \; i = 1, \dots, 9$ and the ellipsoid as a dashed curve are given in Figure 1.

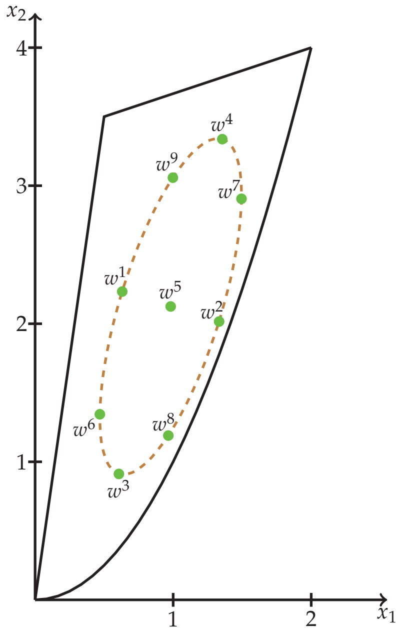


**Figure 1.** Starting points $w^i$ in feasible domain and the inscribed ellipsoid in Example 1.

The advantage of the proposed approach consists in the following: well-allocated points in "narrow and arbitrary oriented" convex compact sets can be determined since the ellipsoid (30) provides a good inner approximation of $X$.

**Example 2.** *We extend the proposed approach to solve the following problem [15]:*

$$f(x) = 5\sum_{j=1}^{4} x_j - 5\sum_{j=1}^{4} x_j^2 - \sum_{j=5}^{13} x_j \to \min\,.$$

*Set $X$ is determined by the following system:*

$$2x_1 + 2x_2 + x_{10} + x_{11} \le 0,$$

$$2x_1 + 2x_3 + x_{10} + x_{12} \le 0,$$

$$2x_2 + 2x_3 + x_{11} + x_{12} \le 0,$$

$$-2x_4 - x_5 + x_{10} \le 0,$$

$$-2x_6 - x_7 + x_{11} \le 0,$$

$$-2x_8 - x_9 + x_{12} \le 0,$$

$$-8x_1 + x_{10} \le 0,$$

$$-8x_2 + x_{11} \le 0,$$

$$0 \le x_j \le 1,\ j = 1, \ldots, 9,$$

$$0 \le x_j \le 100,\ j = 10, 11, 12,$$

$$0 \le x_{13} \le 1.$$

Points $v^i$, $i = 1, \ldots, 2n+1 = 27$ were determined according to Case B (26). Points $w^i$, $i = 1, \ldots, 27$ were computed by (31), and $x^a$ is the analytical center of $X$. Since the objective function is nonconvex and quadratic, the global minimum is achieved on the boundary of $X$. Points $u^i$ were obtained as intersections of rays $x^a + \tau(w^i - x^a), \tau \geq 0$, $i = 1, \ldots, 27$ with the boundary of $X$. Then, the multistart procedure started from points $u^i$ was applied, and the global minimum $x^* = (1,1,1,1,1,1,1,1,1,3,3,3,1)^\top$, $f(x^*) = -15$ was found.

## 4. Allocation of an Arbitrary Given Number of Points

In the previous section, the number of allocated points was equal to $n+2$ or $2n+1$ or $2^n + 2n + 1$. The allocation procedure was based on setting the points in a ball. In this section, we assume that the number of allocated points is $p$, which is different from the previous values, and, more importantly, the allocation procedure is not connected to the ball. The price for such an approach is a sequential solution to a special global optimization problem.

Problem (7) is to be iteratively solved as was announced in Section 1. This problem is a problem of the global maximization of a convex quadratic function over a bounded polyhedral set. Hence, special methods can be used for the solution.

Let the number $p$ of allocated points be given. The first point $v^1$ can be chosen arbitrarily. The remaining points are found by solving the global optimization problem

$$v^{k+1} \in \operatorname{Arg\,max}\{\|x\|^2 + t : 2x^\top v^j + t \leq \|v^j\|^2,\ j = 1, \ldots, k,\ x \in X\}, \quad k = 1, \ldots, p-1. \tag{32}$$

In solving the examples below, we used the solver SCIP [16] for finding the global maximum in Problem (32).

**Example 3.** *The number of allocated points $p = 16$, set $X = \{(x_1, x_2) : x_1 + 2x_2 \leqslant 2, x_1 \geqslant 0, x_2 \geqslant 0\}$. Since the feasible set is polytope, it was decided to start from the vertex $v^1 = (0,0)^\top$. In Figure 2, a geometrical interpretation of the allocated points is given.*

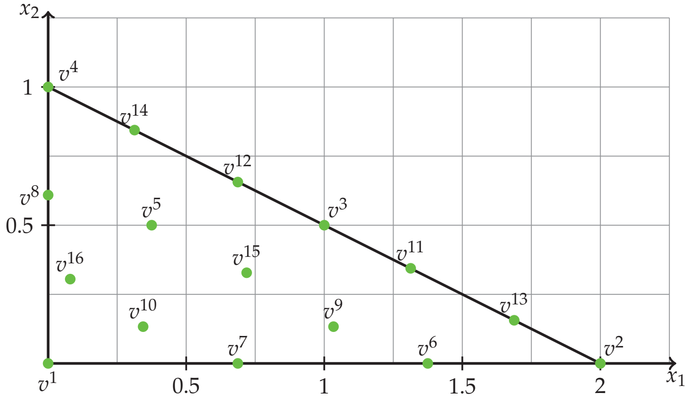


**Figure 2.** Allocation of the starting points in Example 3.

In Table 2, the coordinates of vectors $v^i$ are given, and $r^2$ is the squared maximum distance from the current point to the previous ones.

**Example 4.** *The number of allocated points $p = 16$, set $X = \{(x_x, x_2) : -x_1 + x_2 \leqslant 3,\ x_1 + 2x_2 \leqslant 15,\ 2x_1 - x_2 \leqslant 10,\ -3x_1 - 5x_2 \leqslant -15\}$. The starting vertex $v^1 = (0,3)^\top$. The determined vertices are shown in Figure 3.*

**Table 2.** Points and distances in Example 3.

| $i$ | $v_1^i$ | $v_2^i$ | $r^2$ | $i$ | $v_1^i$ | $v_2^i$ | $r^2$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | — | 9 | 1.031 | 0.133 | 0.136 |
| 2 | 2 | 0 | 4 | 10 | 0.344 | 0.133 | 0.136 |
| 3 | 1 | 0.5 | 1.25 | 11 | 1.313 | 0.344 | 0.122 |
| 4 | 0 | 1 | 1 | 12 | 0.687 | 0.656 | 0.122 |
| 5 | 0.375 | 0.5 | 0.391 | 13 | 1.688 | 0.156 | 0.122 |
| 6 | 1.375 | 0 | 0.391 | 14 | 0.313 | 0.844 | 0.122 |
| 7 | 0.867 | 0 | 0.348 | 15 | 0.719 | 0.328 | 0.109 |
| 8 | 0 | 0.609 | 0.153 | 16 | 0.080 | 0.305 | 0.099 |

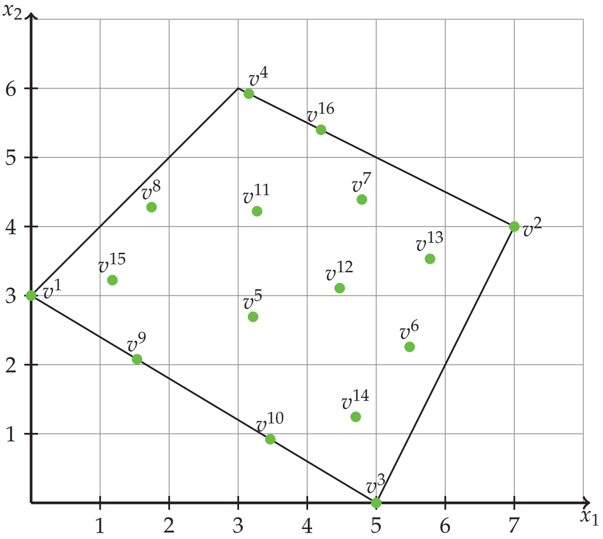


**Figure 3.** Allocation of starting points in Example 4.

Table 3 contains the coordinates of $v^i$ and again the squared maximum distances ($r^2$) from the current point to the previously found ones.

**Table 3.** Points and distances in Example 4.

| $i$ | $v_1^i$ | $v_2^i$ | $r^2$ | $i$ | $v_1^i$ | $v_2^i$ | $r^2$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 3 | — | 9 | 1.535 | 2.079 | 3.303 |
| 2 | 7 | 4 | 50 | 10 | 3.465 | 0.921 | 3.203 |
| 3 | 5 | 0 | 20 | 11 | 3.273 | 4.221 | 2.337 |
| 4 | 3.154 | 5.923 | 18.491 | 12 | 4.471 | 3.108 | 1.748 |
| 5 | 3.216 | 2.693 | 10.436 | 13 | 5.779 | 3.532 | 1.711 |
| 6 | 5.484 | 2.258 | 5.333 | 14 | 4.703 | 1.245 | 1.637 |
| 7 | 4.792 | 4.391 | 5.029 | 15 | 1.178 | 3.224 | 1.438 |
| 8 | 1.745 | 4.280 | 4.684 | 16 | 4.200 | 5.400 | 1.368 |

Example 3 shows that vertices of the given polytope are not necessarily covered by points $v^i$. The vertex $(3,6)^\top$ is not covered.

In practice, it is enough to find a new point, which is sufficiently far from the previous points. Hence, a good local solver can be used for finding the solution to Problem (32). In the testing below, we used the IPOPT solver [17] for this purpose. In the testing problems, the feasible set $X$ was a bounded polyhedral set

$$X = \{x \in \mathbb{R}^n : Ax \le b,\ \underline{x} \le x \le \overline{x}\},$$

with $m \times n$ matrix $A$. Vectors $b \in \mathbb{R}^n,\ \underline{x}, \overline{x} \in \mathbb{R}^n$ were determined randomly in a such a way that $\mathbf{int}(X) \neq \varnothing$. The first two points $v^1$ and $v^2$ are approximate solutions to the problem

$$\|x - y\|^2 \to \max,\ x \in X,\ y \in X, \tag{33}$$

where $v^1 = x^*,\ v^2 = y^*$. For solving Problem (33), the SCIP solver was used with the solution time limitation increased by 30 s. The number of points was equal to 100. The solution to the corresponding problems (32) for $k = 3, \dots, 99$ were obtained by the IPOPT solver. The last point, $v^{100}$, was obtained by the SCIP solver with the time limitation increased to 300 s. In Table 4, $n$ is the number of variables, $m$ is the number of rows in matrix $A$, $\Delta_{12} = \|v^1 - v^2\|$, $\delta$ is the obtained maximum distance from the last point $v^{100}$ to the previous ones, and T is the solving time in seconds. Testing was performed on IntelCore i7-3610QM (2.3 Ghz, 8 GB DDR3 memory).

**Table 4.** Initial and final distances for testing Problem (33).

| $n$ | $m$ | $\Delta_{12}$ | $\delta$ | T |
|---|---|---|---|---|
| 5 | 10 | 1043.004 | 243.887 | 29.125 |
| 10 | 20 | 1931.523 | 608.201 | 116.189 |
| 20 | 30 | 2972.218 | 1272.148 | 414.603 |
| 30 | 45 | 3162.046 | 1430.453 | 461.576 |
| 40 | 60 | 4074.319 | 2166.210 | 551.885 |
| 50 | 75 | 4274.107 | 2145.411 | 630.431 |

In problems with five and ten variables, globally optimal solutions were found. In other words, for example, when $n = 10$, the diameter of $X$ was equal to 1931.523, and the exact maximum distance from the 99 previous points to the point $x^{100}$ was equal to 608.201. In higher-dimensional problems, approximate solutions were determined.

## 5. Two Kinds of Multistart Strategy

We know that the feasible set $X$ can be covered by $p$ balls with centers at $v^1, \dots, v^p$ and with radius $r_p = \sqrt{\varphi(v^p)}$ (see Problem (5)). Consider the $p$ optimization problem

$$f(x) \to \min,\ \|x - v^j\|^2 \le r_p^2,\ x \in X, \tag{34}$$

where $j = 1, \dots, p$. Let $x^{\sharp,j}$, $j = 1, \dots, p$ be points obtained as a result of the application of the CONOPT solver to Problem (34) using $v^j, j = 1, \dots, p$ as the starting points. Compare Problem (34) with the following one:

$$f(x) \to \min,\ x \in X, \tag{35}$$

Let $x^{*,j}$, $j = 1, \dots, p$ be solutions of (35) obtained also by the CONOPT solver applied $p$ times also from points $v^j, j = 1, \dots, p$ as the starting points. Points $x^{\sharp,j}, j = 1, \dots, p$ have a "local nature" because of constraints $\|x - v^j\| \le r_p^2, j = 1, \dots, p$. Therefore, we can make the following assumption: the set $\Omega_p^\sharp = \{x^{\sharp,j} :\ j = 1, \dots, p\}$ contains more different local minima than the set $\Omega_p^* = \{x^{*,j}: \ j = 1, \dots, p\}$. It is not difficult to construct an example, in which all points $x^{\sharp,j}, j = 1, \dots, p$ as well as points $x^{*,j}, j = 1, \dots, p$ are points of different local minima. The first multistart strategy is connected to the construction of the sets $\Omega_p^*$. The second multistart strategy is connected to the construction of the sets $\Omega_p^\sharp$. However, in practice there can be a significant difference between these sets of points for particular cases. Let us consider the following examples.

**Example 5.** *Consider the Bird problem:*

$$f(x_1, x_2) = (x_1 - x_2)^2 + e^{(1-\sin(x_1))^2}\cos(x_2) + e^{(1-\cos(x_2))^2}\sin(x_1),$$

$$x_i \in [-2\pi, 2\pi], i = 1, 2.$$

*This problem has many local minima and two global minimum points, $x^{g,1} = (4.701, 3.152)^\top$ and $x^{g,2} = (-1.582, -3.130)^\top$ with $f(x^{g,1}) = f(x^{g,2}) = -106.765$. For $p \leq 5$, sets $\Omega_p^\sharp$ and $\Omega_p^*$ do not contain global minimum points. When $p = 6$, the set $\Omega_6^\sharp$ contains five different local minima, and one of them is a global minimum. The set $\Omega_6^*$ contains four different local minima, and one of them is a global minimum. In total, the set $\Omega_6^* \bigcup \Omega_6^\sharp$ contains six different points of minimum, and one of them is a global minimum. The set $\Omega_7^*$ contains five local minima and two of them are global minima. The set $\Omega_7^\sharp$ contains the same of local minima as $\Omega_6^\sharp$. In total, the set $\Omega_6^* \bigcup \Omega_6^\sharp$ contains seven different local minima, and two of them are global minima.*

**Example 6.** *Consider the Branin problem:*

$$f(x_1, x_2) = \left(-1.275\frac{x_1^2}{\pi^2} + 5\frac{x_1}{\pi} + x_2 - 6\right) + \left(10 - \frac{5}{4\pi}\right)\cos(x_1)\cos(x_2) +$$

$$+ \log(x_1^2 + x_2^2 + 1) + 10,$$

$$x_i \in [-5, 15], \ i = 1, 2.$$

*The global minimum is unique, $x^g = (-3.2, 12.53)^\top$, $f(x^g) = 5.559$. When $p \leq 17$, the sets $\Omega_{17}^\sharp$ and $\Omega_{17}^*$ do not contain the global minimum point. The set $\Omega_{18}^*$ contains nine different local minima, and one of them is the global minimum. The set $\Omega_{18}^\sharp$ also contains nine different local minima, and one of them is the global minimum. Sets $\Omega_{18}^*$ and $\Omega_{18}^\sharp$ do not coincide, and their union $\Omega_{18}^* \bigcup \Omega_{18}^\sharp$ contains 10 different local minima, and one of them is the global minimum.*

**Example 7.** *Consider the egg crate problem:*

$$f(x_1, x_2) = x_1^2 + x_2^2 + 25\Big(\sin^2(x_1) + \sin^2(x_2)\Big),$$

$$x_i \in [-5, 10].$$

*The global minimum is unique, $x^g = (0, 0)^\top$, $f(x^g) = 0$. The set $\Omega_5^*$ contains five different local minima, and one of them is the global minimum. When $p \leq 4$, the sets $\Omega_p^*$ do not contain the global minimum. As for the sets $\Omega_p^\sharp$, they contain the global minimum for $p \geq 26$. The set $\Omega_{26}^\sharp$ contains twenty-five different local minima, and one of them is the global minimum. In comparison, the set $\Omega_{26}^*$ contains eighteen different local minima, and one of them is the global minimum.*

**Example 8.** *Consider the Mishra problem:*

$$f(x_1,x_2) = \left[\sin^2\Big((\cos(x_1)+\cos(x_2))^2\Big) + \cos^2\Big((\sin(x_1)+\sin(x_2))^2\Big) + x_1\right]^2 +$$

$$+0.01(x_1+x_2),$$

$$x_i \in [-10,10],\ i = 1,2.$$

*The problem has the unique global minimum $x^g = (-1.987, -10)^\top$, $f(x^g) = -0.1198$. The sets $\Omega_p^*$ contain the global minimum for $p \geq 6$, while the set $\Omega_6^*$ contains five different local minima, and one of them is global. The sets $\Omega_p^\sharp$ do not contain the global minima for $p \leq 600$. The corresponding radius of each covering ball for $p = 600$ is equal to 0.625. Hence, the Mishra problem has very many "narrow" points of local minima.*

**Example 9.** *Consider the Price problem:*

$$f(x_1,x_2) = 1 + \sin^2(x_1) + \sin^2(x_2) - 0.1e^{-x_1^2-x_2^2},$$

$$x_i \in [-5,10],\ i = 1,2.$$

*The global minimum is unique, $x^g = (0,0)^\top$, $f(x^g) = 0.9$. The sets $\Omega_p^*$ contains the global minimum for $p \geq 26$. The set $\Omega_p^\sharp$ contains the global minimum for $p \geq 13$.*

**Example 10.** *Consider the Shubert problem:*

$$f(x_1,x_2) = \left(\sum_{i=1}^{5} i\cos((i+1)x_1+i)\right)\left(\sum_{i=1}^{5} i\cos((i+1)x_2+i)\right),$$

$$x_i \in [-10,10],\ i = 1,2.$$

*There are many global minima, one of them being $x^g = (-7.084, 4.858)^\top$, $f(x^g) = -186.7309$. The sets $\Omega_p^*$ start to contain a global minimum from $p = 5$. The set $\Omega_5^*$ contains only two different local minima, and one of them is global. The sets $\Omega_p^\sharp$ contain a global minimum when $p \geq 29$, and all twenty-nine local minima of the set $\Omega_{29}^\sharp$ are different.*

**Example 11.** *Consider the Trefethen problem:*

$$f(x_1,x_2) = 0.25x_1^2 + 0.25x_2^2 + e^{\sin(50x_1)} - \sin(10x_1+10x_2) + \sin(60e^{x_2}) +$$

$$+\sin(70\sin(x_1)) + \sin(\sin(80x_2)),$$

$$x_i \in [-10,10],\ i = 1,2.$$

The global minimum is unique, $x^g = (-0.0244, 0.2106)^\top$, $f(x^g) = -3.3069$. This problem has very many local minima. For example, the set $\Omega_{30}^*$ consists of thirty different local minima, with no global minimum among them. The set $\Omega_{30}^\sharp$ contains twenty-eight new different local minima in addition to the set $\Omega_{30}^*$, again with no global minimum among them. Therefore, the union $\Omega_{30}^* \bigcup \Omega_{30}^*$ contains the fifty-eight different local minima and no global minimum. Only for $p \geq 570$, the sets $\Omega_p^*$ contain the global minimum. The set $\Omega_{570}^\sharp$ contains the five hundred seventy different local minima and no global minimum. Each radius of the five hundred seventy balls, which cover the feasible set, is equal to 0.625.

In all considered examples, the following properties should be mentioned. As a rule, the sets $\Omega_p^*$ need fewer points to detect a global minimum. Example 8 with the Mishra function provides a very remarkable confirmation of this assumption: only six points were used in the set $\Omega_6^*$ to cover the global minimum, whereas even six hundred points were not enough to detect the global minimum in the case of the set $\Omega_{600}^\sharp$. The price for such a behaviour is that many points in the sets $\Omega_p^*$ are found several times, in contrast to the sets $\Omega_p^\sharp$. We also have to keep in mind that in example 9 with the Price function, the situation is opposite: thirteen points to detect the global minimum for the set $\Omega_{13}^\sharp$ and twenty-six points to detect the global minimum for the set $\Omega_{26}^*$. The number of different local minimum points in the sets $\Omega_p^\sharp$ is usually larger than in the sets $\Omega_p^*$. Nevertheless, local minimum points in the sets $\Omega_p^*$ being smaller in number, usually (not always) have lower objective function values.

Let us compare the sets $\Omega_p^*$ and $\Omega_p^\sharp$ for all tested problems and for the same number of points $p = 20$, that is, we compare the sets $\Omega_{20}^*$ and $\Omega_{20}^\sharp$. The results of the comparison are given in Table 5. Column $N_L^*(N_G^*)$ shows the number $N_L^*$ of the different local minima in the corresponding sets $\Omega_{20}^*$, with $N_G^*$ being the number of global minima among them. Similarly, column $N_L^\sharp(N_G^\sharp)$ shows the number $N_L^\sharp$ of different local minima in the sets $\Omega_{20}^\sharp$ with the number $N_G^\sharp$ of global minima among them. Column New $N_G^\sharp$ shows the number of global minima in the sets $\Omega_{20}^\sharp \setminus \Omega_{20}^*$ (new global minima). Column New $N_L^\sharp$ shows the number of local minima in the sets $\Omega_{20}^\sharp \setminus \Omega_{20}^*$ (new local minima). Column $N_L^T(N_G^T)$ shows total number $N_L^T$ of different local minima and total number $N_G^T$ of different global minima obtained by determining both sets $\Omega_{20}^*$ and $\Omega_{20}^\sharp$. For example, for the Shubert problem, we have 13(3) in the column $N_L^*(N_G^\sharp)$, which means that the corresponding set $\Omega_{20}^*$ contains thirteen different local minimum points and three of them are global minimum points. In the column $N_L^\sharp(N_G^\sharp)$, we have 18(3) that means that the corresponding set $\Omega_{20}^\sharp$ contains eighteen different local minima and three of them are global minimum points. Column New $N_G^\sharp$ shows that one new global minimum point is contained in the set $\Omega_{20}^\sharp$ in comparison to the set $\Omega_{20}^*$, and column New $N_L^\sharp$ shows that the set $\Omega_{20}^\sharp$ contains fourteen new local minimum points in comparison to the set $\Omega_{20}^*$. Finally, in column $N_L^T(N_G^T)$, we have 27(4), which means that twenty-seven different local minimum points were determined, and four of them are global minimum points.

**Table 5.** Comparison of two multistart strategies.

| Problem | $N_L^*(N_G^*)$ | $N_L^\sharp(N_G^\sharp)$ | New $N_G^\sharp$ | New $N_L^\sharp$ | $N_L^T(N_G^T)$ |
|---|---|---|---|---|---|
| Bird | 7(2) | 7(2) | 0 | 0 | 7(2) |
| Branin | 10(1) | 11(1) | 0 | 2 | 12(1) |
| Egg Crate | 15(1) | 13(0) | 0 | 8 | 23(1) |
| Mishra | 12(1) | 11(0) | 0 | 6 | 18(1) |
| Price | 6(0) | 17(1) | 1 | 12 | 18(1) |
| Shubert | 13(3) | 18(3) | 1 | 14 | 27(4) |
| Trefethen | 19(0) | 15(0) | 0 | 12 | 31(0) |

Assuming the differentiability of the objective function and finiteness of the set of local minima, it is not possible to assess the number of local minima. Therefore, we propose the following approach. Assess the number $p$ of local minima from some additional practical considerations. Then, construct the set $\Omega_p^*$ containing a good local minimum point or even a global minimum point. After that, construct the set $\Omega_p^\sharp$ to enlarge the number of local minima to catch situations similar to the Price function. Due to the very high efficiency of the CONOPT solver, finding the sets $\Omega_p^*$ and $\Omega_p^\sharp$ is not too computationally demanding. We can obtain a practical assessment of the number of minima of the objective function by using such a mixture of these two kinds of the multistart strategy. If the number of total determined local minima is not very large (for example, many of them are found many times), then we can conclude that we performed a good exploration of the objective function. Otherwise, we can reach the conclusion that the objective function is of a very complicated structure.

## 6. Testing Sequentially Distant Points in Optimization Problems

We present the results of testing the comparative efficiency of using sequentially distant and randomly generated points in solving optimization problems. Three strategies, A, B, and C, based on the cases from Section 1, are tested. Optimization problems are problems of minimizing highly nonlinear functions over a box or parallelepiped. Firstly, the maximum radius ball centered at the center of the parallelepiped is constructed. Secondly, for strategy A, $n+2$ ball sequentially points corresponding to (24)–(25) are determined. For strategy B, $2n+1$ points based on (26) are determined. For strategy C, we use the points (28) plus the center of the parallelepiped, in total $2^n+1$ points.

We used the multistart strategy with the generated points as the starting points. Strategies A, B, and C are compared with random strategies $\text{Rnd}_A$, $\text{Rnd}_B$, and $\text{Rnd}_C$ of the corresponding sizes. In strategy $\text{Rnd}_A$, $n+2$ uniformly distributed points are generated; in strategy $\text{Rnd}_B$, the number of uniformly distributed points is $2n+1$; and in strategy $\text{Rnd}_C$, the number of uniformly distributed points is $2^n+1$. In all strategies, a parallel local search process based on the CONOPT solver was started.

In Tables 6–9, the column "Duplicated Solutions" shows the number of points, which were found several times; the column "Different Solutions" shows the number of different found points; the column "Different Minimum Values" shows the number of different local minimum values among different solutions (i.e., there could be different local minimum points with the same objective value); the column "Record Value" shows the value of the objective function at the best point; in the column "Global Minimum," the sign "+" means that the global minimum was found, otherwise the sign "$-$" is used; and the column "Time" shows the total solution time in seconds. Testing was performed on an Intel Core i7-3610QM computer (2.3 GHz, 8 GB DDR3 memory). All computations were done in GAMS Demo version.

Strategies C and $\text{Rnd}_C$ were used for dimensions $n=5$ and $n=10$, since they are of exponential complexity.

Griewank function. Consider the optimization problem

$$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) \to \min,$$

$$x \in \Pi = \{x \in \mathbb{R}^n : -600 \le x_i \le 900,\ i = 1,\dots,n\}.$$

Global minimum $x^* = (0,\dots,0)^\top,\ f(x^*) = 0$. Testing results are given in Table 6. Properties of the Griewank function are studied in [18].

**Table 6.** Testing results for the Griewank function.

| Strategy | Number of Starting Points | Duplicated Solutions | Different Solutions | Different Minimum Values | Record Value | Global Minimum (+/$-$) | Time (s) |
|---|---|---|---|---|---|---|---|
| | | | | $n=5$ | | | |
| A | 7 | 0 | 7 | 7 | 0.473 | $-$ | 0.531 |
| $\text{Rnd}_A$ | 7 | 0 | 7 | 7 | 0.418 | $-$ | 0.500 |
| B | 11 | 0 | 11 | 4 | 0.118 | $-$ | 0.764 |
| $\text{Rnd}_B$ | 11 | 0 | 11 | 11 | 0.024 | $-$ | 0.843 |
| C | 33 | 0 | 33 | 33 | 0.000 | + | 2.482 |
| $\text{Rnd}_C$ | 33 | 0 | 33 | 27 | 0.000 | + | 2.559 |

**Table 6.** *Cont.*

| Strategy | Number of Starting Points | Duplicated Solutions | Different Solutions | Different Minimum Values | Record Value | Global Minimum (+/−) | Time (s) |
|---|---|---|---|---|---|---|---|
| | | | | $n = 10$ | | | |
| A | 12 | 1 | 11 | 11 | 0.000 | + | 0.858 |
| Rnd$_A$ | 12 | 4 | 8 | 8 | 0.000 | + | 0.889 |
| B | 21 | 0 | 21 | 21 | 0.000 | + | 1.388 |
| Rnd$_B$ | 21 | 5 | 16 | 13 | 0.000 | + | 1.373 |
| C | 1025 | 512 | 513 | 208 | 0.000 | + | 98.109 |
| Rnd$_C$ | 1025 | 499 | 526 | 202 | 0.000 | + | 92.259 |
| | | | | $n = 50$ | | | |
| A | 52 | 38 | 14 | 14 | 0.000 | + | 4.680 |
| Rnd$_A$ | 52 | 51 | 1 | 1 | 0.000 | + | 3.573 |
| B | 101 | 11 | 90 | 84 | 0.000 | + | 8.548 |
| Rnd$_B$ | 101 | 100 | 1 | 1 | 0.000 | + | 8.596 |
| | | | | $n = 100$ | | | |
| A | 102 | 91 | 11 | 11 | 0.000 | + | 8.938 |
| Rnd$_A$ | 102 | 101 | 1 | 1 | 0.000 | + | 9.142 |
| B | 201 | 42 | 159 | 149 | 0.000 | + | 22.089 |
| Rnd$_B$ | 201 | 200 | 1 | 1 | 0.000 | + | 19.968 |
| | | | | $n = 300$ | | | |
| A | 302 | 184 | 118 | 76 | 0.000 | + | 38.923 |
| Rnd$_A$ | 302 | 301 | 1 | 1 | 0.000 | + | 37.768 |
| B | 601 | 269 | 332 | 286 | 0.000 | + | 83.617 |
| Rnd$_B$ | 601 | 600 | 1 | 1 | 0.000 | + | 76.893 |
| | | | | $n = 500$ | | | |
| A | 502 | 398 | 104 | 71 | 0.000 | + | 76.877 |
| Rnd$_A$ | 502 | 501 | 1 | 1 | 0.000 | + | 76.581 |
| B | 1001 | 595 | 406 | 332 | 0.000 | + | 169.198 |
| Rnd$_B$ | 1001 | 1000 | 1 | 1 | 0.000 | + | 138.054 |

Rastrigin function. Consider the optimization problem

$$f(x) = 10n + \sum_{i=1}^{n} \Big( x_i^2 - 10\cos(2\pi x_i) \Big) \to \min,$$

$$x \in \Pi = \{x \in \mathbb{R}^n : -5.12 \le x_i \le 7.68,\ i = 1, \dots, n\}.$$

Global minimum $x^* = (0, \dots, 0)^\top,\ f(x^*) = 0$. Testing results are given in Table 7.

Let us make some comments on the results in Table 7. A uniform distribution of the starting points happened to be very inefficient: the best solution is very far from the optimum. Take, for example, the case $n = 300$. Strategy A found 302 different local minima with 11 different objective function values. Checking the list of local minimum points shows that there are 78 different local minimum points, with the best value being 0.995. Therefore, strategy A shows that there are quite a number of different local minima with objective value close to the optimal one. Formally, the same can be said about strategies

Rnd$_A$ and Rnd$_B$. These random strategies also found a large number of different local minima; however, the objective function values are very far from the optimal value.

Schwefel function. Consider the optimization problem

$$f(x) = 418.9829n - \sum_{i=1}^{n} x_i \sin(\sqrt{|x_i|}) \to \min,$$

$$x \in \Pi = \{x \in \mathbb{R}^n : -500 \leq x_i \leq 500,\ i = 1, \dots, n\}.$$

Global minimum $x^* = (420.9687, \dots, 420.9687)^\top$, $f(x^*) = 0$. Testing results are given in Table 8.

**Table 7.** Testing results for the Rastrigin function.

| Strategy | Number of Starting Points | Duplicated Solutions | Different Solutions | Different Minimum Values | Record Value | Global Minimum (+/−) | Time (s) |
|---|---|---|---|---|---|---|---|
| | | | | $n = 5$ | | | |
| A | 7 | 0 | 7 | 7 | 0.995 | − | 0.780 |
| Rnd$_A$ | 7 | 0 | 7 | 7 | 18.904 | − | 0.515 |
| B | 11 | 1 | 10 | 4 | 0.995 | − | 0.781 |
| Rnd$_B$ | 11 | 0 | 11 | 11 | 3.979 | − | 0.765 |
| C | 33 | 1 | 32 | 23 | 0.000 | + | 2.527 |
| Rnd$_C$ | 33 | 0 | 33 | 27 | 17.909 | − | 2.480 |
| | | | | $n = 10$ | | | |
| A | 12 | 0 | 12 | 9 | 0.995 | − | 0.858 |
| Rnd$_A$ | 12 | 0 | 12 | 12 | 39.798 | − | 0.890 |
| B | 21 | 2 | 19 | 5 | 0.000 | + | 1.576 |
| Rnd$_B$ | 21 | 0 | 21 | 21 | 39.798 | − | 1.638 |
| C | 1025 | 53 | 972 | 162 | 0.000 | + | 101.790 |
| Rnd$_C$ | 1025 | 0 | 1025 | 680 | 21.889 | − | 96.971 |
| | | | | $n = 50$ | | | |
| A | 52 | 0 | 52 | 9 | 0.000 | + | 3.354 |
| Rnd$_A$ | 52 | 0 | 52 | 52 | 198.992 | − | 3.604 |
| B | 101 | 16 | 85 | 7 | 0.000 | + | 8.549 |
| Rnd$_B$ | 101 | 0 | 101 | 101 | 198.992 | − | 8.347 |
| | | | | $n = 100$ | | | |
| A | 102 | 0 | 102 | 11 | 0.995 | − | 12.620 |
| Rnd$_A$ | 102 | 0 | 102 | 102 | 397.983 | − | 10.188 |
| B | 201 | 43 | 158 | 6 | 0.000 | + | 21.013 |
| Rnd$_B$ | 201 | 0 | 201 | 200 | 397.983 | − | 20.124 |
| | | | | $n = 300$ | | | |
| A | 302 | 0 | 302 | 11 | 0.995 | − | 37.378 |
| Rnd$_A$ | 302 | 0 | 302 | 302 | 1193.949 | − | 39.503 |
| B | 601 | 87 | 514 | 7 | 0.000 | + | 79.701 |
| Rnd$_B$ | 601 | 0 | 601 | 601 | 1193.949 | − | 74.911 |
| | | | | $n = 500$ | | | |
| A | 502 | 3 | 499 | 11 | 0.000 | + | 76.995 |
| Rnd$_A$ | 502 | 0 | 502 | 501 | 1989.915 | − | 36.331 |
| B | 1001 | 153 | 842 | 7 | 0.000 | + | 171.975 |
| Rnd$_B$ | 1001 | 0 | 1001 | 1000 | 1989.915 | − | 168.044 |

**Table 8.** Testing results for the Schwefel function.

| Strategy | Number of Starting Points | Duplicated Solutions | Different Solutions | Different Minimum Values | Record Value | Global Minimum (+/−) | Time (s) |
|---|---|---|---|---|---|---|---|
| $n = 5$ | | | | | | | |
| A | 7 | 0 | 7 | 7 | 929.319 | − | 0.515 |
| $\text{Rnd}_A$ | 7 | 0 | 7 | 7 | 475.270 | − | 0.531 |
| B | 11 | 1 | 10 | 7 | 238.915 | − | 0.764 |
| $\text{Rnd}_B$ | 11 | 0 | 11 | 11 | 455.533 | − | 0.749 |
| C | 33 | 1 | 32 | 22 | 118.438 | − | 2.199 |
| $\text{Rnd}_C$ | 33 | 0 | 33 | 31 | 455.533 | − | 2.480 |
| $n = 10$ | | | | | | | |
| A | 12 | 0 | 12 | 12 | 1562.522 | − | 0.905 |
| $\text{Rnd}_A$ | 12 | 0 | 12 | 12 | 1383.337 | − | 0.904 |
| B | 21 | 4 | 17 | 7 | 0.000 | + | 1.388 |
| $\text{Rnd}_B$ | 21 | 0 | 21 | 21 | 1223.898 | − | 1.591 |
| C | 1025 | 165 | 860 | 105 | 0.000 | + | 99.997 |
| $\text{Rnd}_C$ | 1025 | 5 | 1020 | 872 | 651.829 | − | 100.121 |
| $n = 50$ | | | | | | | |
| A | 52 | 0 | 52 | 52 | 2349.118 | − | 3.900 |
| $\text{Rnd}_A$ | 52 | 0 | 52 | 52 | 8164.279 | − | 4.040 |
| B | 101 | 25 | 76 | 23 | 0.000 | + | 8.502 |
| $\text{Rnd}_B$ | 101 | 0 | 101 | 101 | 7859.295 | + | 7.878 |
| $n = 100$ | | | | | | | |
| A | 102 | 0 | 102 | 102 | 296.108 | − | 9.016 |
| $\text{Rnd}_A$ | 102 | 0 | 102 | 102 | 16,993.912 | − | 8.611 |
| B | 201 | 55 | 146 | 31 | 0.000 | + | 21.231 |
| $\text{Rnd}_B$ | 201 | 0 | 201 | 201 | 16,948.095 | − | 18.441 |
| $n = 300$ | | | | | | | |
| A | 302 | 0 | 302 | 294 | 5909.961 | − | 33.119 |
| $\text{Rnd}_A$ | 302 | 0 | 302 | 302 | 53,468.437 | − | 32.479 |
| B | 601 | 215 | 386 | 39 | 0.000 | + | 68.984 |
| $\text{Rnd}_B$ | 601 | 0 | 601 | 601 | 50,650.766 | − | 69.732 |
| $n = 500$ | | | | | | | |
| A | 502 | 3 | 502 | 483 | 214.513 | − | 70.747 |
| $\text{Rnd}_A$ | 502 | 0 | 502 | 502 | 89,847.498 | − | 74.974 |
| B | 1001 | 328 | 673 | 43 | 0.000 | + | 168.498 |
| $\text{Rnd}_B$ | 1001 | 0 | 1001 | 1000 | 89,104.515 | − | 167.998 |

Again, pure random strategies show the worst results.

Levy function. Consider the optimization problem

$$f(x) = 10\sin^2(\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2(1 + 10\sin(\pi x_{i+1}))^2 + (x_n - 1)^2 \to \min,$$

$$x \in \Pi = \{x \in \mathbb{R}^n : -10 \le x_i \le 10,\ i = 1, \dots, n\}.$$

Global minimum $x^* = (1, \dots, 1)^\top$, $f(x^*) = 0$. Testing results are given in Table 9.

**Table 9.** Testing results for the Levy function.

| Strategy | Number of Starting Points | Duplicated Solutions | Different Solutions | Different Minimum Value | Record Value | Global Minimum (+/−) | Time (s) |
|---|---|---|---|---|---|---|---|
| | | | $n = 5$ | | | | |
| A | 7 | 0 | 7 | 5 | 0.001 | − | 0.546 |
| Rnd$_A$ | 7 | 0 | 7 | 7 | 0.337 | − | 0.515 |
| B | 11 | 1 | 10 | 6 | 1.064 | − | 0.858 |
| Rnd$_B$ | 11 | 0 | 11 | 10 | 1.064 | − | 0.764 |
| C | 33 | 0 | 33 | 28 | 0.0005 | − | 2.465 |
| Rnd$_C$ | 33 | 0 | 33 | 31 | 0.0004 | − | 2.446 |
| | | | $n = 10$ | | | | |
| A | 12 | 1 | 11 | 4 | 1.064 | − | 0.858 |
| Rnd$_A$ | 12 | 0 | 12 | 12 | 0.937 | − | 0.983 |
| B | 21 | 1 | 20 | 5 | 1.064 | − | 1.575 |
| Rnd$_B$ | 21 | 0 | 21 | 19 | 0.0005 | − | 1.388 |
| C | 1025 | 0 | 1025 | 252 | 0.0004 | − | 94.849 |
| Rnd$_C$ | 1025 | 1 | 1024 | 568 | 0.0004 | − | 94.817 |
| | | | $n = 50$ | | | | |
| A | 52 | 1 | 51 | 7 | 0.0005 | − | 4.197 |
| Rnd$_A$ | 52 | 0 | 52 | 51 | 0.0005 | − | 3.728 |
| B | 101 | 1 | 100 | 4 | 1.064 | − | 8.502 |
| Rnd$_B$ | 101 | 0 | 101 | 98 | 0.001 | − | 8.377 |
| | | | $n = 100$ | | | | |
| A | 102 | 1 | 101 | 4 | 0.0005 | − | 8.486 |
| Rnd$_A$ | 102 | 0 | 102 | 101 | 0.0005 | − | 8.361 |
| B | 201 | 1 | 200 | 5 | 1.064 | − | 18.939 |
| Rnd$_B$ | 201 | 0 | 201 | 199 | 0.002 | − | 19.355 |
| | | | $n = 300$ | | | | |
| A | 302 | 1 | 301 | 6 | 0.937 | − | 33.665 |
| Rnd$_A$ | 302 | 0 | 302 | 302 | 1.064 | − | 40.778 |
| B | 601 | 1 | 600 | 12 | 1.064 | − | 76.332 |
| Rnd$_B$ | 601 | 0 | 601 | 601 | 1.064 | − | 75.629 |
| | | | $n = 500$ | | | | |
| A | 502 | 1 | 501 | 7 | 1.064 | − | 74.210 |
| Rnd$_A$ | 502 | 0 | 502 | 502 | 1.064 | − | 94.257 |
| B | 1001 | 1 | 1000 | 20 | 0.0005 | − | 168.590 |
| Rnd$_B$ | 1001 | 0 | 1001 | 1000 | 1.064 | − | 171.942 |

The Levy function was the most difficult testing case for all strategies. Not one of them could determine the global minimum. Nevertheless, strategies A and B are relatively efficient in high-dimensional cases.

The total testing showed that the most effective was strategy B, in terms of both finding the best solution and computational efforts. This effect can be explained in the following way: strategy B explores the total area of the feasible set more efficiently than the others.

## 7. Conclusions and Future Work

Sequentially most distant points techniques were suggested for determining good starting points in multistart strategies for problems of global optimization. Preliminary testing showed that the new strategies find good local minima very fast. The sequentially distant points can be obtained either by using an inscribed ellipsoid centered at the analytical center of the feasible set or by approximately solving auxiliary global optimization problems of special types.

Our future work will be devoted to an extension of the suggested techniques to solving global optimization problems with nonconvex feasible sets and to solving special highly nonlinear problems from practical applications.

**Author Contributions:** Conceptualization, O.K. and E.S.; software, O.K.; validation, O.K.; investigation, O.K.; methodology, E.S.; formal analysis, E.S. and V.N.; resources, V.N. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** Data are contained within the article.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Available online: https://www.conopt.com/ (accessed on 17 December 2023).
2. Brauchart, J.S.; Grabner, P.J. Distributing many points on spheres: Minimal energy and disigns. *J. Complex.* **2015**, *31*, 293–326. [CrossRef]
3. Trikalinos, T.A.; van Valkenhoef, G. *Efficient Sampling from Uniform Density n-polytopes*; Technical Report; Brown University: Providence, RI, USA, 2014; 5p.
4. Chen, Y.; Dwivedi, R.; Wainwright, M.J.; Yu, B. Fast MCMC Sampling Algorithms on Polytopes. *J. Mach. Learn. Res.* **2018**, *19*, 1–86.
5. Diaconis, P. The Markov Chain Mont Carlo Revolution. *Bull. AMS* **2009**, *46*, 179–205. [CrossRef]
6. Zhigljavsky, A.A. *Theory of Global Random Search*; Kluwer Academic Publishers: Dordrecht, The Netherlands, 1991; 341p.
7. Polyak, B.; Shcherbakov, P. Why Does Monte Carlo Fail to Work Properly in High-Dimensional Optimization Problems? *J. Optim. Theory Appl.* **2017**, *173*, 612–627. [CrossRef]
8. Ugray, Z.; Lasdon, L.; Plummer, J.C.; Glover, F.; Kelly, J.; Martí, R. A Multistart Scatter Search Heuristic for Smooth NLP and MINLP Problems. In *Metaheuristic Optimization via Memory and Evolution*; Sharda, R., Voß, S., Rego, C., Alidaee, B., Eds.; Operations Research/Computer Science Interfaces Series; Springer: Boston, MA, USA, 2005; Volume 30.
9. Janáček, J.; Kvet, M.; Czimmermann, P. Kit of Uniformly Deployed Sets for p-Location Problems. *Mathematics* **2023**, *11*, 2418. [CrossRef]
10. Dupin, N.; Nielsen, F.; Talbi, E.-G. Unified Polynomial Dynamic Programming Algorithms for P-Center Variants in a 2D Pareto Front. *Mathematics* **2021**, *9*, 453. [CrossRef]
11. Sarhani, M.; Voß, S.; Jovanovic, R. Initialization of metaheuristics: Comprehensive review, critical analysis, and research directions. *Int. Trans. Oper. Res.* **2022**, *30*, 3361–3397. [CrossRef]
12. Horst, R.; Tuy, H. *Global Optimization: Deterministic Approaches*; Springer: Berlin/Heidelberg, Germany, 1996; 730p.
13. Jarre, F. Interior Point Methods for Class of Convex Programming. In *Interior Points Methods in Mathematical Programming*; Kluwer Academic Publishers: Dordrecht, The Netherlands, 1996; pp. 255–296.
14. Available online: http://www.sfu.ca/~ssurjano/drop.html (accessed on 17 December 2023).
15. Floudas, C.A.; Pardalos, P.M.; Adjiman, C.S.; Esposito, W.R.; Gümxuxs, Z.H.; Harding, S.T.; Klepeis, J.L.; Meyer, C.A.; Schweiger, C.A. *Handbook of Test Problems in Local and Global Optimization*; Springer: Dordrecht, The Netherlands, 1999; 441p.
16. Available online: https://www.scipopt.org/ (accessed on 17 December 2023).
17. Available online: https://coin-or.github.io/Ipopt/ (accessed on 17 December 2023).
18. Locatelli, M. A Note on the Griewank Test Function. *J. Glob. Optim.* **2003**, *25*, 169–174. [CrossRef]

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

mathematics

Article

# A Stabilisation System Synthesis for Motion along a Preset Trajectory and Its Solution by Symbolic Regression

**Askhat Diveev [1], Elena Sofronova [1] and Nurbek Konyrbaev [2,*]**

[1] Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, Vavilova Str., 44, Build. 2, 119333 Moscow, Russia; aidiveev@mail.ru (A.D.); sofronova_ea@mail.ru (E.S.)
[2] Institute of Engineering and Technology, Korkyt Ata Kyzylorda University, Aiteke bi Str. 29A, Kyzylorda 120014, Kazakhstan
* Correspondence: n.konyrbaev@mail.ru

**Abstract:** The problem of a stabilisation system synthesis for the motion of a control object along a given spatial trajectory is considered. The complexity of the problem is that the preset trajectory is defined in the state subspace and not in time. This paper describes a stabilisation system synthesis for motion along a trajectory specified in time and along a trajectory specified in the form of a manifold in a state space. In order to construct a stabilisation system, it is necessary to determine a distance between an object and the given trajectory at each moment in time. For trajectories that are not given in time, the determination of this distance can be ambiguous. An object may be exactly on a trajectory but at a different time. This paper proposes some approaches to solve the problem. One of the approaches is to transform a given trajectory in a state subspace into a trajectory given in time. A description of a universal method to perform this transformation is presented. In order to solve the synthesis problem automatically, without having to analyse the mathematical model of the control object, it is suggested that machine learning control by symbolic regression is used. In computational experiments, examples of stabilisation system syntheses for quadcopter motion along a given spatial trajectory are presented.

**Keywords:** machine learning control; optimal control; control synthesis; stabilisation system; symbolic regression; quadcopter

**MSC:** 49M25; 68W50

**Citation:** Diveev, A.; Sofronova, E.; Konyrbaev, N. A Stabilisation System Synthesis for Motion along a Preset Trajectory and Its Solution by Symbolic Regression. *Mathematics* **2024**, *12*, 706. https://doi.org/10.3390/math12050706

Academic Editors: Liliya Demidova and Ioannis G. Tsoulos

Received: 3 December 2023
Revised: 7 February 2024
Accepted: 21 February 2024
Published: 28 February 2024

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

## 1. Introduction

The problem of stabilising the motion of an object along a given trajectory is very common for almost all autonomous robots moving in geometric space. If a mathematical model of a control object is an ODE system with a free control vector in the right part, then the stabilisation system is a control function that changes the right part so that the ODE system, as a parametric mapping of the state space into itself, has an attractor property in the neighbourhood of the given trajectory. An argument of the control function of a stabilisation system is a deviation of the control object from the given trajectory. An ideal stabilisation system has such a control function that the ODE system of the mathematical model of the control object has a stable particular solution or a singular particular solution in the form of an attractor in the neighbourhood of the given trajectory.

Basically, two approaches are used to solve the tracking problem. The first is analytical, when the inverse problem is solved (a trajectory is specified in time and a control is sought that ensures movement along the trajectory). This approach is widely used for simple low-dimensional models and, as a rule, control is included linearly in the object model. In many papers related to trajectory tracking [1–5], the control system is built based on the analysis of the control object model. The developer of the control system studies the mathematical model of the control object, defines control channels that affect the movement of the object and determines the components of the control vector that affect a particular direction of movement of the object. Further, regulators are inserted into the control channels, which qualitatively work out movement errors along the trajectory.

The second approach is applied when the trajectory is given in space. A stabilisation system is built relative to a point in state space and then these points are placed on the trajectory [6–8]. Switching points occur sequentially and the object moves from one point to another along a trajectory. This process is known as point tracking. This approach ensures accurate movement along the trajectory providing the intervals for switching points and their locations are optimally selected. Note that the movement of a control object from one stable point to another is not uniform. The speed of the object slows down as it approaches a stable point of equilibrium. At the equilibrium point, the object stops. Therefore, point tracking is not optimal if the quality criterion depends on the time of the control process. Each of these approaches, when executed carefully enough, can produce a satisfactorily accurate trajectory.

To ensure the stability of the object with respect to the equilibrium point in point tracking, it is necessary to solve the control synthesis problem. It is necessary to search for a control function as a function of a state space vector. At present this problem does not have an exact universal numerical solution. The most general approach is to linearise the model with respect to the equilibrium point and to find the control as a linear feedback function, so that the matrix of the control system has eigenvalues in the left half of the complex plane. Such an approach does not provide any quality in the control process. Solutions of differential equations in the neighbourhood of a stable equilibrium point can be asymptotic or periodic.

The control synthesis problem is so complex that it can only be solved when the control system is created. Trajectories can be given in the process of robotic operation. The motion stabilisation system should be universal, so that this system can be used to stabilise a wide class of trajectories, i.e., one stabilisation system should be applied to movement along any trajectory from a set. In [9], a universal stabilisation system for motion along an optimal program trajectory is obtained as a result of solving the optimal control problem. The stabilisation system is constructed on the basis of machine learning control by symbolic regression. The optimal trajectory is a function of time, so the deviation of the control object from the given trajectory is simply calculated as a difference between the state vector of the control object and the current coordinate of the trajectory point in the geometric subspace at any time. This paper continues the study of the construction of a universal stabilisation system for the motion of a control object along a given trajectory. In contrast to [9], here we consider the trajectories that are not functions of time but are given in the state space in the form of a one-dimensional manifold.

Another aspect of the study deals with the automation of control system development. For a real control system, the problem of control synthesis is solved manually by studying the mathematical model of the control object, determining the control channels and inserting the regulators there. This paper presents the approach of constructing an object motion stabilisation system along the trajectory based on machine learning control by symbolic regression. Machine learning (ML) is a branch of artificial intelligence that focuses on the learning through experience and decision making of computer programs similar to humans. ML has already been effectively applied to various fields [10], for example, to develop accurate, interpretable and generalisable nonlinear models for complex natural and engineered systems [11]. A survey on the application of ML to solve large control problems is presented in [12]. In [13], the term machine learning control (MLC) was introduced as a framework to discover effective control laws for complex, nonlinear dynamics, mainly by genetic programming [14].

The machine, or more precisely the program running on the machine itself, builds a system for stabilising the movement of the object along the trajectory. Developers do not need to study the mathematical model of the control object and define control channels. A computer does it all for them. The developer does not even need to know the trajectory itself, so the stabilisation system to be created is first trained to follow any trajectory consisting of straight segments. The developer then splits the trajectory into segments and passes them to the stabilisation system in this form. The object under the control of the stabilisation system moves along a given trajectory. Thus, the novelty of the research is to propose a universal approach to stabilisation system synthesis for motion along a trajectory not given in time.

In this paper MLC is used to solve the stabilisation system synthesis problem for quadcopter motion along a trajectory given in the state subspace by a symbolic regression method, a network operator method. To estimate the performance of the proposed approach, a linear trajectory containing sharp corners, which pose difficulties for the movement of the quadcopter, was chosen.

## 2. Statement of the Stabilisation System Synthesis of an Optimal Motion

### *2.1. Stabilisation System along Trajectory Given in Time*

The mathematical model of a control object in the form of an ODE system is

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \tag{1}$$

where $\mathbf{x}$ is a state vector of the control object, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} = [x_1 \ldots x_n]^T$; the state space vector consists of two subvectors

$$\mathbf{x}^T = [\mathbf{y}^T \vdots \mathbf{z}^T],\ \mathbf{y} = [x_1 \ldots x_k]^T,\ \mathbf{z} = [x_{k+1} \ldots x_n]^T, \tag{2}$$

$\mathbf{y} \in \mathbb{R}^k$, $\mathbb{R}^k$ is a geometrical subspace, $k \in \{2, 3\}$, $\mathbf{z}$ is a subvector of state vector that contains components that are not included into geometrical subspace, $\mathbf{u}$ is a control vector, $\mathbf{u} \in \mathrm{U} \subseteq \mathbb{R}^m$ and U is a compact set that defines control constraints. For example, control vector component values may be constrained

$$\mathbf{u}^- \leqslant \mathbf{u} \leqslant \mathbf{u}^+, \tag{3}$$

$\mathbf{u}^- = [u_1^- \ldots u_m^-]^T$, $\mathbf{u} = [u_1 \ldots u_m]^T$, $\mathbf{u}^+ = [u_1^+ \ldots u_m^+]^T$.

For system (1), an initial state domain is

$$\mathrm{X}_0 \subseteq \mathbb{R}^n. \tag{4}$$

The terminal state is given in geometrical subspace

$$\mathbf{y}(t_f) = \mathbf{y}^f = [x_1^f \ldots x_k^f]^T, \tag{5}$$

where $t_f$ is a limited terminal time to reach the terminal state and $t_f \leqslant t^+$, $t^+$ is a given value.

The quality criterion is given in the following common integral form

$$J_0 = \int_0^{t_f} f_0(\mathbf{x}, \mathbf{u}) dt \to \min_{\mathbf{u} \in \mathrm{U}}. \tag{6}$$

The spatial trajectory is presented as a one-dimensional manifold

$$\theta_i(\mathbf{y}) = 0,\ i = 1, \ldots, k-1. \tag{7}$$

The terminal state (5) is located on the given trajectory,

$$\theta_i(\mathbf{y}^f) = 0,\ i = 1, \ldots, k-1. \tag{8}$$

The control function is searched as

$$\mathbf{u} = \mathbf{h}(\mathbf{y}^* - \mathbf{y}(t)) \in \mathrm{U}, \tag{9}$$

where $\mathbf{y}^*$ is the point on the manifold (7) closest to the current value of the state vector

$$\min \|\mathbf{y}^* - \mathbf{y}(t)\|, \ \theta_i(\mathbf{y}^*) = 0, \ i = 1, \dots, k-1. \tag{10}$$

For any $P$ initial states from the set (4), the following quality criterion reaches the minimum value

$$J_1 = \sum_{i=1}^{P} \left( \int_0^{t_{f,i}} (f_0(\mathbf{x}(t, \mathbf{x}^{0,i}), \mathbf{h}(\mathbf{y}^* - \mathbf{y}(t, \mathbf{x}^{0,i}))) + p_1 \|\mathbf{y}^* - \mathbf{y}(t, \mathbf{x}^{0,i})\|) dt + \right.$$

$$\left. p_2 \|\mathbf{y}^f - \mathbf{y}(t_{f,i}, \mathbf{x}^{0,i})\| \right) \to \min, \tag{11}$$

where $p_1$, $p_2$ are weight coefficients, $\mathbf{y}(t, \mathbf{x}^{0,i})$ is a particular solution of ODE system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{h}(\mathbf{y}^* - \mathbf{y})) \tag{12}$$

from initial state $\mathbf{x}^{0,i} \in \mathrm{X}_0$, $i = 1, \dots, P$, and $t_{f,i}$ is a time to reach the terminal state (5), which is calculated as

$$t_{f,i} = \begin{cases} t, \text{ if } t < t^+ \text{ and } \|\mathbf{y}^f - \mathbf{y}(t, \mathbf{x}^{0,i})\| \leqslant \varepsilon_1 \\ t^+, \text{ otherwise} \end{cases}, \ i = 1, \dots, P, \tag{13}$$

$\varepsilon_1$ is the given small positive value.

The solution of the problem is a control function (9). To solve the problem, it is necessary to find the point $\mathbf{y}^*$ on the manifold (7) closest to the current state of the control object at each moment of time. It is a rather time-consuming computational process because it requires solving an optimisation problem.

### 2.2. Stabilisation System along Trajectory Given in Space

Let us consider another approach to solve this problem. Suppose that any one-dimensional manifold can be divided into straight segments. Let $\mathbf{y}^{*,j}$, $j = 1, \dots, N$ be the join points of the straight segments in the geometric space, $\mathbb{R}^k$. Then, the length of straight segment between points $\mathbf{y}^i$ and $\mathbf{y}^{i+1}$ is

$$L_j = \|\mathbf{y}^{*,j+1} - \mathbf{y}^{*,j}\|. \tag{14}$$

The length of the manifold is

$$L = \sum_{j=1}^{N-1} L_i = \sum_{j=1}^{N-1} \|\mathbf{y}^{*,j+1} - \mathbf{y}^{*,j}\|. \tag{15}$$

If $t^+$ is the maximum time for motion on the manifold, then a module of motion speed on the manifold is

$$v = \frac{L}{t^+}, \tag{16}$$

and a time of movement on a straight segment $j$, $j = 1, \dots, N-1$, is

$$t_j = \frac{L_j}{L} t^+. \tag{17}$$

Now, let us build a reference model

$$\dot{\mathbf{y}}^* = \mathbf{v}, \tag{18}$$

where $\mathbf{v} = [v_1 \dots v_k]^T$,

$$v_i = \frac{y_i^{*,j+1} - y_i^{*,j}}{t_j}, \tag{19}$$

where $y_i^{*,j}$ is a coordinate $i$ of the point $j$, $i = 1, \dots, k$, $j = 1, \dots, N-1$.

The mathematical model of the control object includes the reference model for trajectory generation

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}), \\ \dot{\mathbf{y}}^* &= \mathbf{v}. \end{aligned} \tag{20}$$

The task requires the search for a control function, presented as (9), to minimise the value of the quality criterion (11). To address this control synthesis problem, the approach of machine learning control through symbolic regression is employed.

## 3. Network Operator Method

Symbolic regression allows finding a mathematical expression in the form of special code. Coding depends on the method chosen. To code a mathematical expression, symbolic regression uses an alphabet of elementary functions. Genetic programming (GP) [14] is the most popular symbolic regression method. GP codes mathematical expressions in the form of computation trees. Arguments of mathematical expressions are the leaves of the trees; functions are located in the nodes. The number of branches leaving the node is equal to the number of arguments of the function associated with this node. When performing the crossover operation, two codes exchange the branches exiting from the nodes selected as crossover points. After crossover, the length of the GP code may change which requires additional computational effort for analysis. Now, there are about twenty symbolic regression methods.

In this study the network operator (NOP) method developed by the authors is used [15]. The network operator method uses codes of mathematical expressions presented as directed graphs. Functions with one or two arguments form the alphabet of elementary functions. Functions with two arguments are commutative and associative, and have unit elements, so these functions with two arguments can be potentially used as functions with any number of arguments. If a function with two arguments has one argument then the second argument is a unit element of this function. In the network operator method, the source nodes of the directed graph contain arguments of the mathematical expression. The remaining nodes in the graph contain functions that require two arguments. The edges between the nodes in the graph are associated with functions that only require one argument.

### 3.1. Coding

To illustrate this concept, let us explore an example of encoding a mathematical expression using the network operator method. Consider mathematical expressions

$$\begin{aligned} y_1 &= x_1 \exp(-q_1 x_2 + q_2) \sin(q_2 x_1 + q_1), \\ y_2 &= x_2 \exp(-q_2 x_1 + q_1) \cos(q_1 x_2 + q_2), \end{aligned} \tag{21}$$

where $x_1$ and $x_2$ are variables, and $q_1$ and $q_2$ are constant parameters.

In order to transform the mathematical expression into a code, let us employ the alphabet of functions:

(1) Functions with one argument

$$\begin{aligned} \mathrm{F}_1 = \{ & f_{1,1}(z) = z, f_{1,2}(z) = -z, f_{1,3}(z) = \exp(z), \\ & f_{1,4}(z) = \sin(z), f_{1,5}(z) = \cos(z)\}; \end{aligned} \tag{22}$$

(2) Functions with two arguments

$$\mathrm{F}_2 = \{f_{2,1}(z_1, z_2) = z_1 + z_2, f_{2,2}(z_1, z_2) = z_1 \cdot z_2\}. \tag{23}$$

The identity function $f_{1,1}(z)$, which outputs the same value as its input argument, should be among the functions with one argument.

Functions with two arguments should be commutative and associative, and have a unit element,

$$f_{2,i}(e_i, z) = f_{2,i}(z, e_i) = z,$$

where $e_i$ is a unit element of the function $f_{2,i}(z_1, z_2)$. In this case, 0 is a unit element for the summation function $f_{2,1}(z_1, z_2)$ and 1 is a unit element for the multiplication function $f_{2,2}(z_1, z_2)$.

In Figure 1, the directed graph of the NOP for a given mathematical Expression (21) is depicted.

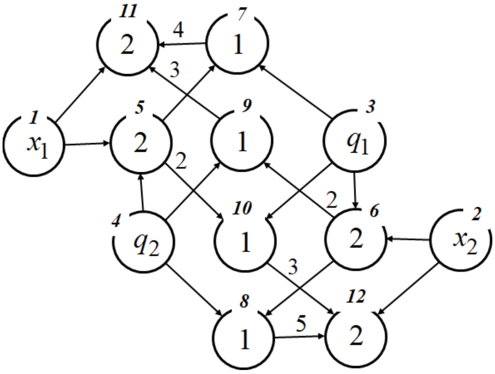


**Figure 1.** The graph of the NOP for (21).

The nodes are enumerated in their upper parts. The arguments are in the source nodes №1–№4. Nodes other than the source node contain numbers of functions with two arguments. Numbers of functions with a single argument are displayed over the edges. Nodes №11 and №12 are the sink nodes. When the node indices are arranged such that the index of the node where the edge originates is lower than the indices of the nodes where the edge terminates, the network operator matrix becomes upper triangular.

In the memory of a personal computer, the network operator is represented as an integer matrix that follows a structure similar to the adjacency matrix of the network operator graph. The network operator matrix corresponding to the graph shown in Figure 1 takes the following form

$$\mathbf{\Psi} = [\psi_{i,j}] = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}, \; i, j = 1, \ldots, D = 12, \qquad (24)$$

where $D$ is the number of nodes in the graph and $\dim(\mathbf{\Psi}) = D \times D$.

In the matrix, rows that have zeros on the main diagonal are connected to source nodes. The other elements on the main diagonal that are not zero represent the numbers of functions with two arguments. The elements above the main diagonal, denoted as $\psi_{i,j} \neq 0$, are numbers of functions with one argument.

### 3.2. Decoding

To calculate the mathematical expression using the network operator, a vector of nodes is defined. Initially, this vector consists of the arguments of the mathematical expression and the unit elements of the corresponding functions with two arguments.

The initial vector of nodes for (24) is

$$\mathbf{z}^{(0)} = [x_1\ x_2\ q_1\ q_2\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 1]^T. \tag{25}$$

Further, all rows in the network operator matrix are followed sequentially and when a non-zero element occurs components of the vector of nodes are changed

$$z_j^{(i)} \leftarrow \begin{cases} f_{2,\psi_{j,j}}(z_j^{(i-1)}, f_{1,\psi_{i,j}}(z_i^{(i-1)})), \text{if } \psi_{i,j} \neq 0 \\ z_j^{(i-1)}, \text{otherwise} \end{cases}, \ i = 1, \dots, D-1,\ j = i+1, \dots, D, \tag{26}$$

where $\psi_{i,j}$ is an element of the network operator matrix in row $i$ and column $j$.

Each component of the vector of nodes contains the results of intermediate calculations.

After viewing the $(i-1)$-th row of the component, the $z_i$ does not change.

For example, in row 1, the first non-zero element is $\psi_{1,5}$ =1. This means that the component $z_5$ of the vector of nodes changes. We define a function with two arguments by the matrix of the network operator, $\psi_{5,5} = 2$. Therefore, it is a function of $f_{2,2}(z_1, z_2)$ multiplication. The first argument of this function is the value of the current $z_5$ component of the node vector. The second component is determined by the value of a non-zero element, $\psi_{1,5} = 1$. This is a single argument function with the number 1, $f_{1,1}(z)$. The argument to this function is the value of the $z_1$ component of the node vector. As a result, we obtain

$$z_5^{(1)} \leftarrow f_{2,2}(z_5^{(0)}, f_{1,1}(z_1^{(0)})) = f_{2,2}(1, f_{1,1}(x_1)) = 1 \cdot x_1 = x_1.$$

The vector of nodes after viewing all rows of the network operator matrix has the following form

$$\mathbf{z}^{(11)} = \begin{bmatrix} x_1 \\ x_2 \\ q_1 \\ q_2 \\ q_2x_1 \\ q_1x_2 \\ q_2x_1 + q_1 \\ q_1x_2 + q_2 \\ -q_1x_2 + q_2 \\ -q_2x_1 + q_1 \\ x_1 \exp(-q_1x_2 + q_2) \sin(q_2x_1 + q_1) \\ x_2 \exp(-q_2x_1 + q_1) \cos(q_1x_2 + q_2) \end{bmatrix}. \tag{27}$$

The last two components are equal to the mathematical expressions (21).

### 3.3. Variational Genetic Algorithm

In order to find the mathematical expression that is optimal in some criterion using the network operator method, the variational genetic algorithm (VarGA) is employed. VarGA follows the principle of making slight changes, so-called small variations, to the initial solution [16]. The symbolic regression method is utilised to code one potential solution, known as the basic solution. Other solutions are presented as small variations of the basic solution. These variations are represented by an integer vector of four components

$$\mathbf{w} = [w_1\ w_2\ w_3\ w_4]^T, \tag{28}$$

where $w_1$ is a type of variation, $w_2$ is the row number, $w_3$ is the column number, $w_2 \leqslant w_3$ and $w_4$ is the new value of an element in the matrix.

Four types of small variations are defined for the network operator. The first type, denoted by $w_1 = 0$, involves substituting the function with a single argument: if $\psi_{w_2,w_3} \neq 0$, then $\psi_{w_2,w_3} \leftarrow w_4$.

The second type, $w_1 = 1$, is an exchange of the function with two arguments: if $\psi_{w_2,w_2} \neq 0$, then $\psi_{w_2,w_2} \leftarrow w_4$.

The third type, $w_1 = 2$, is an insertion of the additional function with one argument: if $\psi_{w_2,w_3} = 0$, then $\psi_{w_2,w_3} \leftarrow w_4$.

The fourth type, $w_1 = 3$, is an elimination of the function with one argument: if $\psi_{w_2,w_3} \neq 0$ and $\exists \psi_{w_2,j} \neq 0$, $j > w_2$, $j \neq w_3$ and $\exists \psi_{i,w_3} \neq 0$, $i \neq w_2$, then $\psi_{w_2,w_3} \leftarrow 0$.

Consider an example of the vector of small variations for (24)

$$\mathbf{w} = [2\ 5\ 8\ 4]^T. \tag{29}$$

The first component shows that it is a small variation of type 3 changing a zero non-diagonal element. In the network operator matrix (24) element in the fifth row $w_2 = 5$, in the eighth column, $w_3 = 8$ is equal to zero, $\psi_{5,8} = 0$. After the small variation (29), this element is changed to $\psi_{5,8} = w_4 = 4$. The edge from node 5 to node 8 appears in the graph (see Figure 2). The new edge is dash lined.

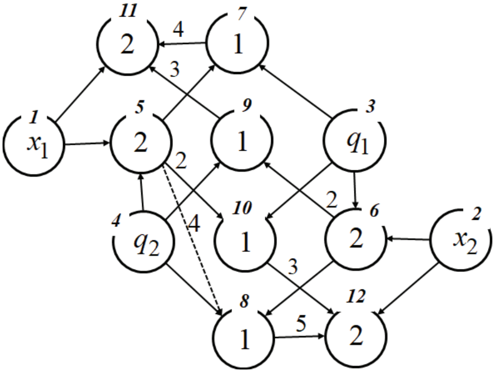


**Figure 2.** The graph of the NOP after the small variation with a new edge (dash line).

As a result a new mathematical expression is obtained

$$y_2 = x_2 \exp(-q_2 x_1 + q_1) \cos(q_1 x_2 + q_2 + \sin(x_1 q_2)). \tag{30}$$

All other possible solutions that originated from the basic solution are coded by ordered sets of vectors of small variations

$$\mathrm{W}_i = (\mathbf{w}^{i,1}, \ldots, \mathbf{w}^{i,d}), \tag{31}$$

where $i = 1, \ldots, H$, $H$ is a number of possible solutions in the population and $d$ is a depth of variation.

The crossover in VarGA is performed over ordered sets of small variations. Two possible solutions are selected randomly or as a result of tournament

$$\mathrm{W}_i = (\mathbf{w}^{i,1}, \ldots, \mathbf{w}^{i,d}),\ \mathrm{W}_j = (\mathbf{w}^{j,1}, \ldots, \mathbf{w}^{j,d}),\ i, j \in \{1, \ldots, H\}. \tag{32}$$

The crossover point is selected randomly, $c \in \{1, \ldots, d\}$. New possible solutions are obtained after the exchange of small variation vectors after the crossover point in the selected possible solutions

$$\begin{array}{rcl} W_{H+1} & = & (\mathbf{w}^{i,1}, \ldots, \mathbf{w}^{i,c}, \mathbf{w}^{j,c+1}, \ldots, \mathbf{w}^{j,d}), \\ W_{H+2} & = & (\mathbf{w}^{j,1}, \ldots, \mathbf{w}^{j,c}, \mathbf{w}^{i,c+1}, \ldots, \mathbf{w}^{i,d}). \end{array} \tag{33}$$

## 4. Computation Experiment

Consider the optimal control problem of the spatial motion of a quadcopter. The control object is presented by mathematical model

$$\begin{array}{rcl} \dot{x}_1 & = & x_4, \\ \dot{x}_2 & = & x_5, \\ \dot{x}_3 & = & x_6, \\ \dot{x}_4 & = & u_4(\sin(u_3)\cos(u_2)\cos(u_1) + \sin(u_1)\sin(u_2)), \\ \dot{x}_5 & = & u_4\cos(u_3)\cos(u_1) - g, \\ \dot{x}_6 & = & u_4(\cos(u_2)\sin(u_1) - \cos(u_1)\sin(u_2)\sin(u_3)), \end{array} \tag{34}$$

where $g = 9.80665$.

The initial state is

$$\mathbf{x}(0) = \mathbf{x}^0 = [0\ 5\ 0\ 0\ 0\ 0]^T. \tag{35}$$

The terminal state is

$$\mathbf{x}(t_f) = \mathbf{x}^f = [10\ 5\ 10\ 0\ 0\ 0]^T, \tag{36}$$

where

$$t_f = \begin{cases} t, \text{ if } t < t^+ \text{and } \|\mathbf{x}^f - \mathbf{x}\| \leq \varepsilon_1 = 0.01 \\ t^+ = 14, \text{ otherwise} \end{cases} . \tag{37}$$

The quality criterion is

$$J_2 = \int_0^{t_f} 1dt = t_f \to \min. \tag{38}$$

For a given model of the control object, it is necessary to develop a stabilisation system for movement along a given spatial trajectory. The given trajectory consists of straight segments in 3D space. To define a spatial trajectory of straight connected segments, it is enough to define the locations of connection points. In the considered example, the coordinates of the connection points are as follows

$$\begin{array}{rcl} Y^* & = & \{\mathbf{y}^{*,1} = [0\ 5\ 0]^T,\ \mathbf{y}^{*,2} = [2\ 5\ 2]^T,\ \mathbf{y}^{*,3} = [8\ 5\ 2]^T, \\ & & \mathbf{y}^{*,4} = [2\ 5\ 8]^T,\ \mathbf{y}^{*,5} = [8\ 5\ 8]^T,\ y^{*,6} = [10\ 5\ 10]^T\} \end{array} \tag{39}$$

According to the proposed approach, initially, a reference model (18) is created.

The length of the trajectory is

$$L = \sum_{i=1}^{5} L_i = 2.82843 + 6 + 8.48528 + 6 + 2.82843 = 26.14213. \tag{40}$$

Equation (17) is used to calculate the values of the reference time intervals. The following time intervals were obtained: $t_1 = 1.515$, $t_2 = 3.213$, $t_3 = 4.544$, $t_4 = 3.213$ and $t_5 = 1.515$.

The control object with the reference model is

$$\begin{aligned}
\dot{y}_1^* &= v_1, \\
\dot{y}_2^* &= v_2, \\
\dot{y}_3^* &= v_3, \\
\dot{y}_4^* &= 0, \\
\dot{y}_5^* &= 0, \\
\dot{y}_6^* &= 0, \\
\dot{x}_1 &= x_4, \\
\dot{x}_2 &= x_5, \\
\dot{x}_3 &= x_6, \\
\dot{x}_4 &= h_4(\mathbf{y}^* - \mathbf{x})(\sin(h_3(\mathbf{y}^* - \mathbf{x}))\cos(h_2(\mathbf{y}^* - \mathbf{x}))\times \\
&\quad \cos(h_1(\mathbf{y}^* - \mathbf{x})) + \sin(h_1(\mathbf{y}^* - \mathbf{x}))\sin(h_2(\mathbf{y}^* - \mathbf{x}))), \\
\dot{x}_5 &= h_4(\mathbf{y}^* - \mathbf{x})\cos(h_3(\mathbf{y}^* - \mathbf{x}))\cos(h_1(\mathbf{y}^* - \mathbf{x})) - g, \\
\dot{x}_6 &= h_4(\mathbf{y}^* - \mathbf{x})(\cos(h_2(\mathbf{y}^* - \mathbf{x}))\sin(h_1(\mathbf{y}^* - \mathbf{x})) - \\
&\quad \cos(h_1(\mathbf{y}^* - \mathbf{x}))\sin(h_2(\mathbf{y}^* - \mathbf{x}))\sin(h_3(\mathbf{y}^* - \mathbf{x}))),
\end{aligned} \tag{41}$$

where $\mathbf{h}(\mathbf{y}^* - \mathbf{x})$ is a required control function,

$$v_i = \frac{y_i^{*,j+1} - y_i^{*,j}}{t_j}, \text{ if } t_{j-1} \leqslant t < t_j, j = 1, \ldots, 5,\ t_0 = 0. \tag{42}$$

The current objective is to address the control synthesis problem and determine a control function based on the deviation between the state vector of the control object and that of the reference model

$$\mathbf{h}(\mathbf{y}^* - \mathbf{x}) = [h_1(\mathbf{y}^* - \mathbf{x}) \ldots h_4(\mathbf{y}^* - \mathbf{x})]^T. \tag{43}$$

Machine learning control by the network operator method is used. When solving the synthesis problem, the initial state (35) was replaced by a set of initial states

$$\mathrm{X}_0 = \{\mathbf{x}^{0,1}, \ldots, \mathbf{x}^{0,P}\}, \tag{44}$$

where

$$\mathbf{x}^{0,i+3j+9k+1} = [x_1^0 + (i-1)\Delta\ x_2^0 + (j-1)\Delta\ x_3^0 + (k-1)\Delta\ x_4^0\ x_5^0\ x_6^0]^T, \tag{45}$$

$i \in \{0,1,2\}, j \in \{0,1,2\}, k \in \{0,1,2\}, \Delta = 0.5$, for $i = j = k = 2$, $P = 27$.

When solving the synthesis problem, the error of movement along the given trajectory and the accuracy of reaching the terminal state for all initial states are additionally included in the criterion

$$J_3 = \sum_{i=1}^{P} \left( t_{f,i} + p_1 \int_0^{t_{f,i}} \|\mathbf{y}^* - \mathbf{y}(t, \mathbf{x}^{0,i})\| dt + p_2 \|\mathbf{x}^f - \mathbf{x}(t, \mathbf{x}^{0,i})\| \right) \to \min, \tag{46}$$

where $p_1 = 1$, $p_2 = 1$ and $t_{f,i}$ is a time to reach the terminal state from the initial state $\mathbf{x}^{0,i}$ according to (13).

Machine learning control by network operator method was performed with the following parameters: size of NOP matrix 36 × 36, graph of NOP had 12 source nodes, including 6 nodes for variables and 6 nodes for searched parameters, and 4 sink nodes for output components of control vector. Parameters of variational genetic algorithm were: number of possible solutions in initial population—512, number of crossover operations in one generation—64, number of generations—64, depth of variation—5, number of generations between change of basic solution—20, number of bits for coding a parameter—16.

Computations were performed on CPU Intel core i7, 2.8 GHz. Computational time was approx. 30 min.

The following solution was found by the network operator method

$$u_i = \begin{cases} u_i^+, \text{ if } \hat{u}_i > u_i^+ \\ u_i^-, \text{ if } \hat{u}_i < u_i^- \\ \hat{u}_i, \text{ otherwise} \end{cases}, i = 1, \ldots, m = 4, \tag{47}$$

where

$$\hat{u}_1 = \mu(C), \tag{48}$$

$$\hat{u}_2 = \hat{u}_1 - \hat{u}_1^3, \tag{49}$$

$$\hat{u}_3 = \hat{u}_2 + \rho_{19}(W + \mu(C)) + \rho_{17}(A), \tag{50}$$

$$\hat{u}_4 = \hat{u}_3 + \ln(|\hat{u}_2|) + \operatorname{sgn}(W + \mu(C))\sqrt{|W + \mu(C)|} + \rho_{19}(W) +$$

$$\arctan(H) + \operatorname{sgn}(F) + \arctan(E) + \exp(q_2(y_2^* - x_2)) + \sqrt{q_1}, \tag{51}$$

$$C = q_6(y_6^* - x_6) + q_3(y_3^* - x_3),\ W = V + \tanh(G) + \exp(D),$$

$$A = q_1(y_1^* - x_1) + q_4(y_4^* - x_4),\ H = G + \tanh(F) + \rho_{18}(B),$$

$$F = E + C + \arctan(D) - B,\ E = D + \operatorname{sgn}(y_5^* - x_5) + (y_2^* - x_2)^3,$$

$$V = \exp(H) + \cos(q_6(y_6^* - x_6)) + \operatorname{sgn}(D)\sqrt{|D|},\ G = F + \sqrt[3]{E} + \sin(A),$$

$$B = \sin(q_6(y_6^* - x_6)) + q_5(y_5^* - x_5) + q_2(y_2^* - x_2) + \cos(q_1) + \vartheta(y_2^* - x_2),$$

$$D = \rho_{17}(C) + B^3 + A + \vartheta(q_5(y_5^* - x_5)) + (y_5^* - x_5)^2,$$

$$\mu(\alpha) = \begin{cases} \alpha, \text{ if } |\alpha| < 1 \\ \operatorname{sgn}(\alpha), \text{ otherwise} \end{cases},$$

$$\vartheta(\alpha) = \begin{cases} 1, \text{ if } \alpha > 0 \\ 0, \text{ otherwise} \end{cases},$$

$$\rho_{17}(\alpha) = \operatorname{sgn}(\alpha)\ln(|\alpha| + 1),$$

$$\rho_{18}(\alpha) = \operatorname{sgn}(\alpha)(\exp(|\alpha|) - 1),$$

$$\rho_{19}(\alpha) = \operatorname{sgn}(\alpha)\exp(-|\alpha|),$$

$q_1 = 7.26709$, $q_2 = 11.46143$, $q_3 = 12.77026$, $q_4 = 3.20630$, $q_5 = 8.38501$ and $q_6 = 5.56250$.

The projections of optimal trajectory (black line) and the given trajectory (blue line) on the horizontal and vertical planes are shown in Figures 3 and 4.

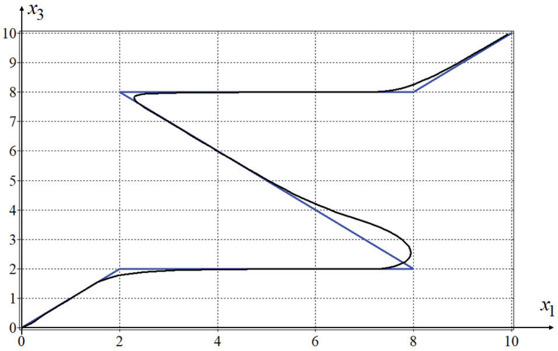


**Figure 3.** Projection of optimal trajectory (black line) and the given trajectory (blue line) on the horizontal plane $\{x_1; x_3\}$.

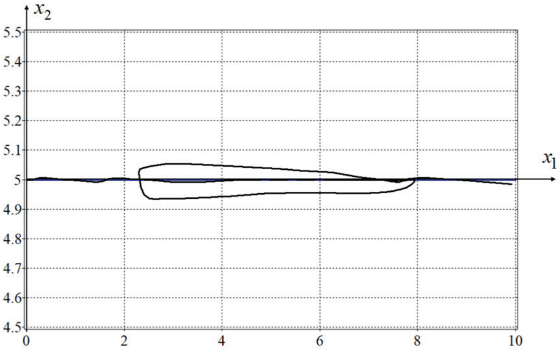


**Figure 4.** Projection of optimal trajectory (black line) and the given trajectory (blue line) on the vertical plane $\{x_1; x_2\}$.

To check the feasibility property of the obtained solution of the optimal control problem, a simulation of the control system from eight different initial states was performed: $\mathbf{x}^{0,1} = [-0.5\ 4.5\ -0.5\ 0\ 0\ 0]^T$, $\mathbf{x}^{0,2} = [-0.5\ 4.5\ 0.5\ 0\ 0\ 0]^T$, $\mathbf{x}^{0,3} = [-0.5\ 5.5\ -0.5\ 0\ 0\ 0]^T$, $\mathbf{x}^{0,4} = [-0.5\ 5.5\ 0.5\ 0\ 0\ 0]^T$, $\mathbf{x}^{0,5} = [0.5\ 4.5\ -0.5\ 0\ 0\ 0]^T$, $\mathbf{x}^{0,6} = [0.5\ 4.5\ 0.5\ 0\ 0\ 0]^T$, $\mathbf{x}^{0,7} = [0.5\ 5.5\ -0.5\ 0\ 0\ 0]^T$, $\mathbf{x}^{0,8} = [0.5\ 5.5\ 0.5\ 0\ 0\ 0]^T$.

The results of the simulation are shown in Figures 5 and 6.

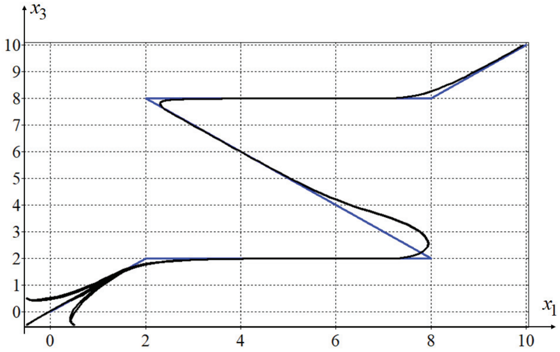


**Figure 5.** Projections of trajectories of control object from eight initial states (black lines) and the given trajectory (blue line) on the horizontal plane $\{x_1; x_3\}$.

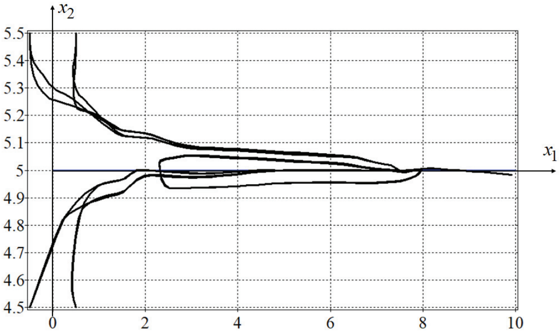


**Figure 6.** Projections of trajectories of control object from eight initial states (black lines) and the given trajectory (blue line) on the vertical plane $\{x_1; x_2\}$.

The results of the simulation show that the synthesised stabilisation system of the control object motion along the given spatial trajectory is not sensitive to disturbances; therefore it is realisable in a real object.

## 5. Discussion

The main difference of the method considered in this paper is that the trajectory stabilisation system is built entirely on the basis of machine learning by symbolic regression. Typically, a person analyses the mathematical model of the control object, identifies the control channels and determines which channels influence a particular movement of the object. Controllers are then installed in these channels and only then the parameters of the controllers are adjusted. This process is manual. A feature of machine learning control by symbolic regression is that the program is only provided with a model of the control object with a control function in the right part. The machine performs all other stages independently, without human intervention. It finds the control function in a coded form. The authors could not find a similar machine approach for the problem in other publications. Comparing the machine-learning-based approach discussed in the paper with control systems manually developed by specialists is not entirely correct, although it can be noted that the machine built a control system of equal quality. The disadvantage of the resulting control system is the complexity of the resulting mathematical expression, because the machine does not sense the complexity of calculations. We still insist that a machine search for solutions is more promising for building complex control systems than the manual approach.

In comparison to previous papers by the authors [9], in the present work, the complexity of the optimal control problem is that the given trajectory is defined in space and not in time. Instead of determining points on a given trajectory to calculate the deviation of an object from that trajectory as in prior approaches, a reference model is constructed that determines the reference motion along the trajectory. To solve the control synthesis problem, a machine learning control by symbolic regression is applied. The main goal of machine learning is to create a program to automatically write a control function. In our opinion, this approach will allow artificial intelligence to be created. Computational experiments have shown that the resulting control system has a feasibility property.

However, the proposed numerical method as all numerical methods has certain drawbacks. To obtain a solution it uses numerical techniques that include discretisation of the problem domain and approximation of the solution through an iterative computational process. Errors are accumulated and affect the final solution which is not exact, but it is still the only way to solve complex problems, such as the one presented in this paper.

Furthermore, the limitation of the proposed approach, namely, the usage of a reference model for trajectory generation, is that the velocity can be easily obtained for straight segments but might become a tricky task for differentiable trajectories.

## 6. Future Work

Future research is aimed at applying the presented approach to trajectories of different types. The applicability of this approach to differentiable trajectories, where the trajectory does not consist of straight segments, should be investigated. It is also necessary to investigate how the accuracy of the approximation of a smooth trajectory by straight segments affects the accuracy of the movement of the object along the trajectory, when the control synthesis problem has been solved for a reference model of movement on straight segments.

**Author Contributions:** Conceptualisation, A.D. and E.S.; methodology, A.D.; software, A.D. and E.S.; validation, E.S.; formal analysis, A.D.; investigation, E.S. and N.K.; resources, A.D.; writing—original draft preparation, A.D. and E.S.; writing—review and editing, E.S.; visualisation, N.K.; supervision, A.D. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the Science Committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan (Grant No. AP14869851).

**Data Availability Statement:** The data that support the findings of this study are openly available in https://cloud.mail.ru/public/rApd/Tv43xQ2QY, accessed on 1 December 2023.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Samir, A.; Hammad, A.; Hafez, A.; Mansour, H. Quadcopter Trajectory Tracking Control using State-Feedback Control with Integral Action. *Int. J. Comput. Appl.* **2017**, *168*, 1–7. [CrossRef]
2. Nguyen, A.T.; Xuan-Mung, N.; Hong, S.-K. Quadcopter Adaptive Trajectory Tracking Control: A New Approach via Backstepping Technique. *Appl. Sci.* **2019**, *9*, 3873. [CrossRef]
3. Zhang, T.; Xue, J.; Jiao, X.; Wang, Z. Adaptive Finite Time Trajectory Tracking Control of Autonomous Vehicles. In Proceedings of the 2020 4th CAA International Conference on Vehicular Control and Intelligence (CVCI), Hangzhou, China, 18–20 December 2020; pp. 684–688.
4. Cui, C.; Zhu, D.; Sun, B. Trajectory Re-planning and Tracking Control of Unmanned Underwater Vehicles on Dynamic Model. In Proceedings of the 2018 Chinese Control and Decision Conference (CCDC), Shenyang, China, 9–11 June 2018; pp. 1971–1976.
5. Wang, R.; Xue, H.; Li, Z.; Li, H.; Wang, N.; Zhao, H. Fixed-Time Trajectory Tracking Control of an Unmanned Surface Vehicle. In Proceedings of the 2020 International Conference on System Science and Engineering (ICSSE), Kagawa, Japan, 31 August–3 September 2020; pp. 1–4.
6. Walsh, G.; Tilbury, D.; Sastry, S.; Murray, R.; Laumond, J.P. Stabilization of Trajectories for Systems with Nonholonomic Constraints. *IEEE Trans. Autom. Control* **1994**, *39*, 216–222. [CrossRef]
7. Mohamed, M.J.; Abbas, M.Y. Design a Fuzzy PID Controller for Trajectory Tracking of Mobile Robot. *Eng. Technol. J.* **2018**, *36A*, 100–110. [CrossRef]
8. Ma, T.; Wong, S. Trajectory Tracking Control for Quadrotor. UAV. In Proceedings of the 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO), Macao, China, 5–8 December 2017.
9. Diveev, A.; Sofronova, E. Universal Stabilisation System for Control Object Motion along the Optimal Trajectory. *Mathematics* **2023**, *11*, 3556. [CrossRef]
10. Jordan, M.I.; Mitchell, T.M. Machine learning: Trends, perspectives, and prospects. *Science* **2015**, *349*, 255–260. [CrossRef] [PubMed]
11. Brunton, S. Machine Learning for Scientific Discovery. *Bull. Am. Phys. Soc.* **2023**, *68*. Available online: https://meetings.aps.org/Meeting/MAR23/Session/S13.1 (accessed on 1 December 2023).
12. Bensoussan, A.; Li, Y.; Nguyen, D.P.C.; Tran, M.-B.; Yam, S.C.P.; Zhou, X. Machine Learning and Control Theory. *arXiv* **2020**, arXiv:2006.05604.
13. Duriez, T.; Brunton, S.L.; Noack, B.R. *Machine Learning Control—Taming Nonlinear Dynamics and Turbulence*; Springer International Publishing: Cham, Switzerland, 2017.
14. Koza, J.R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*; MIT Press: Cambridge, MA, USA, 1992; 819p.
15. Diveev, A.I.; Sofronova, E.A. The network operator method for search of the most suitable mathematical equation. In *Bio-Inspired Computational Algorithms and Their Applications*; InTech: Rijeka, Croatia, 2012; pp. 19–42.
16. Sofronova, E.A.; Diveev, A.I. Universal Approach to Solution of Optimization Problems by Symbolic Regression. *Appl. Sci.* **2021**, *11*, 5081. [CrossRef]

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

mathematics

MDPI

Article

# An Efficient Method for Solving Problems of Acoustic Scattering on Three-Dimensional Transparent Structures

**Alexander B. Samokhin * and Ivan A. Yurchenkov ***

Institute of Information Technologies, Federal State Budget Educational Institution of Higher Education, MIREA—Russian Technological University, 78 Vernadsky Avenue, 119454 Moscow, Russia
* Correspondence: samokhin@mirea.ru (A.B.S.); yurchenkov@mirea.ru (I.A.Y.)

**Abstract:** The article contains a study of methods for solving integral equations in the context of acoustic problems. The methodology considered is applied to describe acoustic wave propagation and scattering. Efficient discretization methods are used together with iterative methods to solve the operator equations, including an apparatus for fast multiplication of the resulting post-discretization Toeplitz matrices by a vector using the fast Fourier transform. The theoretical analysis of the proposed numerical algorithm demonstrates its efficiency in terms of the required number of arithmetic operations and the memory footprint of the computing system. The presented numerical simulation demonstrates the possibility of solving the problem of acoustic wave propagation in transparent media using the proposed methods. A visualization of the obtained solutions for a practical problem with a high level of discretization of the solution volume domain is also presented.

**Keywords:** integral equations; collocation method; iterative method; modified gradient descent method; fast Fourier transform

**MSC:** 15B05; 28-08

**Citation:** Samokhin, A.B.; Yurchenkov, I.A. An Efficient Method for Solving Problems of Acoustic Scattering on Three-Dimensional Transparent Structures. *Mathematics* **2024**, *12*, 789. https://doi.org/10.3390/math12060789

Academic Editors: Jaan Janno and Nikolaos L. Tsitsas

Received: 12 December 2023
Revised: 4 March 2024
Accepted: 5 March 2024
Published: 7 March 2024

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

## 1. Introduction

The initial formulation for solving problems of mathematical physics using integral equations refers to the works of Fredholm, who first proposed the study and solution of classical problems of mathematical physics using the transformation of the original differential equations to the formulation of the problem for solving related integral equations, later called Fredholm equations. The solution of problems of mathematical physics, both in the form of differential and integral equations. at that time was impossible using numerical methods and methods of discretization of the initial problem via a grid of high dimensionality in the number of elementary partitions, as such problems were associated with impossible manual calculations. With the development of computer technology appeared the formulation and the first methods and algorithms for distributed computations over differential equations, allowing the modeling of real physical processes, and integral equations were used less frequently, due to problems with the complexity of calculations, in the solution of systems of linear algebraic equations (SLEs) over fully filled operator matrices.

Mikhlin, in his works, investigated different variations of problem statements for solving integral equations in the theory of elasticity [1], and also investigated the possibilities of obtaining analytical and numerical solutions of singular integral equations [2,3] related to the problems of mathematical physics. Colton and Kress considered surface and volume problems of acoustics and electrodynamics based on integral equations, as well as their numerical formulations [4]. In these papers, however, due to the reasons described above, there were no numerical results accompanying test problems. Miller, in a series of papers [5], also investigated modern formulations for the solution of electrodynamics problems on the basis of computational methods for solving bulk singular integral equations, using features of kernels of integral operators based on Green's functions [4], satisfying solutions for the volumetric Helmholtz equation.

Mathematical formulation and development of effective computational methods for solving problems of mathematical physics are the most important areas of science and technology development today. Modern problems are associated with large requirements for accuracy and speed of computation, in view of which the requirements of the dimensionality of discretization of problems increase. The classical formulation of the solution of differential and integral equations, as well as general computational methods for solution, are becoming insufficient to meet the needs of experimenters, researchers and manufacturers. This aspect pushes us to the necessity of developing effective methods for solving specific problem formulations based on a priori factual knowledge, assumptions about the structure of the solution, and constraints on the conditions at the boundary of the domain [6].

Overcoming these limitations, recent works on computational methods, applicable to the solution of differential equations that reduce to solving systems of linear equations (SLEs) with high-dimensional sparse operator matrices, have proposed a series of approximate methods capable of addressing the problem of increasing the discretization of the solution domain, based on homotopies. Among these are the homotopy method described in detail in [7], the multigrid method [8], the multigrid–homotopy method [9], and the wavelet multiscale method [10]. These methods are traditionally used in tasks related to solving problems of mathematical physics, including, for example, [11–13], and conducting numerical experiments with models based on partial differential equations.

Homotopy methods [7] for approximate solutions of large-dimensional SLEs, resulting from the discretization of the solution domain of the corresponding differential equation, allow the obtaining of solutions with specified accuracy on large grids. The main idea of the method is to use the concept of homotopy for a gradual transition from a simplified version of the problem to its original, more complex form, while following a path that maintains connectivity between solutions. The homotopy grid method is especially useful in cases where direct solving of the original problem is difficult due to its complexity or the presence of multiple local minima/maxima.

The algebraic multigrid method [8] represents a class of algorithms for solving systems of linear equations, particularly effective for large sparse matrices arising from the discretization of differential equations. This method does not require explicit knowledge of the geometry of the problem and is built directly on the basis of the equation coefficients, making it applicable to a wider class of problems. The key idea of multigrid methods is to use iterations at different levels of discretization (grids) to accelerate convergence to the solution. Simplified versions of the original problem are solved on coarse grids, effectively reducing error components, which decrease slowly on finer grids. This is achieved by analyzing the structure of the system matrix and grouping variables in such a way that the problem dimension decreases at each subsequent level. Various coarsening strategies, for example, based on the strength of connections between matrix elements, are used for this purpose.

Combining these two approaches, the multigrid–homotopy method [9] leverages the advantages of both in order to solve complex nonlinear problems. The homotopy method allows effective finding of an initial approximation for a nonlinear problem or a series of such approximations, while the multigrid approach ensures rapid convergence to an accurate solution at each step of homotopy. This method is particularly useful for problems where direct application of the multigrid method to a nonlinear system proves inefficient due to the complexity of the problem or lack of a good initial approximation. Combining the homotopy approach, to find an initial approximation, with the multigrid method for its refinement can significantly improve the efficiency of the iterative process.

All the above methods can adequately solve high-dimensional problems related to setting tasks in terms of differential equations; however, they may be difficult to apply to tasks based on integral equations. This is due to the complexity of calculations with fully populated operator matrices that arise when applying various discretization methods to integral formulations of mathematical physics problems.

The formulation of problems in terms of integral equations in mathematical physics is also important, as it allows the finding of the desired solution with fewer restrictions both on the form of the solution function itself and on the number of initial and boundary conditions. Solving integral equations on a grid of large dimensionality is associated with the use of fast iterative methods capable of solving full systems of linear equations in fewer iterations with a given accuracy. For this, it is necessary to introduce special discretization grids for a more accurate approximation [14] of the original solution domain, both in volume and at the boundaries between different media. This work presents one such technique for introducing a special discretization grid for volumetric integral equations, allowing for the solution of such equations using efficient matrix–vector multiplications in the process of any iterative method of solving systems of linear equations based on fast Fourier transform (FFT).

The development of integral equations theory to date is described by a large number of works in the field of proving existence and uniqueness theorems for solutions of the problems of mathematical physics, as well as the development and study of effective numerical methods for their solution. In electrodynamics, Smirnov Yu.G. proved a large number of existence and uniqueness theorems for problems of wave propagation, diffraction and polarization using the Fredholm theory for integral operators and the theory of pseudo-differential operators [15–17]. In hydrodynamics and aerodynamics, integral equations can be used in modeling stationary flows, shown in the works of Setukha [18].

In this paper we consider the problems of wave propagation in three-dimensional bounded transparent structures with inhomogeneous refraction based on the Fredholm integral equation of the second kind. Modeling problems of acoustic wave propagation in an inhomogeneous medium has a wide range of applications in various fields. For example, in oceanography, such models are used to investigate underwater sound channels and to study the influence of various factors on sound propagation in water [19]. In the process of developing printed circuit boards, defects, such as delamination, can occur during the lamination process, which can be detected by the acoustic emission method discussed in [20]. In electrodynamics, models of propagation and scattering of electromagnetic waves, built on integral equations, help to make calculations for complex processes of radiation effects and load on technical devices, necessary to their design [21,22]. Such models are also used in medical diagnostics and industrial acoustics.

Numerical methods for solving integral equations, along with the growth of computing power and memory capacity, allow the solution of more and more complex structural problems with a high degree of discretization, due to optimization and the possibility of efficient parallelization of the computations of problems with the full operator matrix. Mathematical modeling of real physical problems on the basis of the apparatus for effective methods for their solution, shown in this paper, will be useful in real-time calculations for digital twins in many physical processes, natural phenomena, technical devices and their manufacturing processes. The main part of the research, problem statements, and computational methods used in the article grew out of the theoretical basis of the research on problem statements, methods, and the algorithms for their numerical solution given in the corresponding book [23].

## 2. Materials and Methods

In this study, we consider the integral equation within a bounded region of $Q$ of Euclidean space $E_3$ [24]:

$$(1 + \alpha\, \eta(x))\, u(x) + \int\limits_Q \frac{K(x-y)}{R^m} \eta(y)\, u(y)\, dy = u^0(x), x \in Q, m \leq 3. \tag{1}$$

In Equation (1), $x = (x_1, x_2, x_3)$, $y = (y_1, y_2, y_3)$ are the points belonging to the bounded region $Q$; $R = |x - y|$ is the distance between the points of the region; and $\alpha,\ \eta,\ K,\ u^0$ are known functions, with $K(x - y)$ a differentiable coordinate function; $u(x)$ is an unknown function.

Further, suppose that Equation (1) has a unique solution in the corresponding function space. Numerical methods turn out to be uniquely possible for solving Equation (1). In this case, Equation (1), by applying the Galerkin method or the collocation method, is approximated to an SLE with a fully filled matrix.

The equation in the form of Equation (1) describes a number of important applied problems, including propagation and scattering of acoustic waves on transparent inhomogeneous obstacles [4]. In this case, $m < 3$, $\alpha = 0$ and the other functions included in Equation (1) are scalar. Then, the equation is a classical Fredholm integral equation of the 2nd kind, which can be written in the following form:

$$\begin{gathered} u(x) + k^2 \int_Q G(R)\hat{n}(x)u(y)dy = u^0(x); \\ \hat{n}(x) = \eta(x) - 1;\ G(R) = -\tfrac{\exp(ikR)}{4\pi R};\ u^0(x) = -\int_Q G(R)f(y)dy;\ x \in Q,\ ^{c} \end{gathered} \tag{2}$$

where $\eta(x)$ is the function characterizing the acoustic refraction of the definition area, and $\eta(x) = 1,\ x \notin Q$; $k$ is the wave number characterizing the properties of the simulated external radiation; $G(R) = -K(x - y)/R$ is the kernel of the integral equation, which is the solution of the corresponding differential formulation of the Helmholtz equation in three-dimensional space; $f(x)$ is the modeled external radiation function; and $u(x)$ is the unknown scalar potential field characterizing the stress for the $x \in Q$.

Previously, in [25], for the integral formulation of Equation (2), the theorem of the existence and uniqueness of the solution of the corresponding differential equation was proved under the condition of radiation at infinity, as well as under some conditions for refraction $\eta(x)$.

Equation (2) describes both the problems of acoustic wave propagation in bulk transparent media with a homogeneous scalar refraction index $\eta(x)$ and the problems of wave scattering at the boundary of media with different $\eta(x)$, where the function itself is piecewise continuous [4]. The method for solving Equation (2) does not depend on the form of $\eta(x)$.

In this paper, to demonstrate the proposed numerical methods, we propose to consider a class of acoustic problems represented by Equation (2). It is also worth noting that other classes of problems of mathematical physics can be described using integral equations [4,22,26,27].

Collocation method on a uniform grid. In practice, in order to solve integral equations, it is necessary to resort to discretization of the solution domain under consideration. To approximate the integral Equation (2), we will use the collocation method [23,28]. For three-dimensional problems, some difficulties arise in the discretization of integral equations defined in regions of complex shape.

Let us represent the area $Q$ as a union of $N_Q$ cells $\Omega(i)$, $i = 1, \ldots,\ N_Q$. Nodal points in these cells will be chosen in their centers, which are defined by the formulas [29]

$$x_l^c = \frac{\int_\Omega x_l dx}{mes\Omega},\ \ l = 1, \ldots, 3, \tag{3}$$

where $dx = dx_1 dx_2 dx_3$ represents the integration of the volumetric $l$-th partition of the solution domain $Q$, $x^c = \left(x_1^c, x_2^c, x_3^c\right)$ is the cell center $\Omega$, and $mes\Omega$ its volume. If in region $\Omega$ a differentiable function of its arguments is defined $f(x)$, then approximate equality is true:

$$\int_\Omega f(x)dx \approx f(x^c)mes\Omega. \tag{4}$$

Expression (4) will be an exact equality if $f(x)$ is a linear function of the arguments.

We will approximate the integral Equation (2) by SLE of dimension $\sim N_Q$ with respect to the values of the unknown function at the nodal points of the domain $Q$, located in the centers of the $x^{ci}$ cells $\Omega(i),\ i = 1, \ldots,\ N_Q$. We will choose the cell sizes so that the desired function varies weakly within the cell. Then, redefining the corresponding SLE can be represented in the following form [23,30]:

$$\begin{gathered}
u(i) + \sum_{j=1}^{N_Q} A(i,j)\eta(j)u(j) = u^0(i),\ i = 1, \ldots, N_Q, \\
A(i,j) = -\mathrm{k}^2 \int_{\Omega(j)} \frac{K(x^{ci}-y)}{|x^{ci}-y|} dy;\ u^0(i) = -\int_Q \frac{K(x^{ci}-y)}{|x^{ci}-y|} f(x^{ci}) dy; \\
i, = 1, \ldots, N_Q,\ u(i) = u(x^{ci}),\ u^0(i) = u^0(x^{ci}),\ \eta(i) = \eta(x^{ci}).
\end{gathered} \tag{5}$$

To calculate the integrals in Equation (5), we can use the approximate Formula (4) or more accurate numerical integration algorithms. Note that, since the nodal points are located in the center of the cells, the accuracy of approximation of the integral operators $\sim h^2$ where $h$ is the maximum cell diameter (we define the cell diameter as the maximum distance between the boundary points). For relatively small values of $N_Q \leq 10000$, we can solve the system of Equation (5) by direct or iterative methods. Below, we will outline efficient algorithms which, using iterative methods, allow us to solve system Equation (5) with much higher dimensionality.

In the kernel of the integral Equations (1) and (2), there is a term depending on the difference in Cartesian coordinates of points $x$ and $y$. However, this circumstance was not used in any way in the construction of SLE Equation (5). Below, using a uniform grid and discrete Fourier transform algorithms, we construct an efficient numerical method for solving Equation (2).

Consider a complex function $f(n)$ of discrete argument $n = 0,\ \pm 1,\ \pm 2, \ldots.$, and let us assume that $f(n)$ is a periodic function with period $N$, i.e., $f(n \pm N) = f(n)$ for any $n$. The discrete Fourier transform of the function $f(n)$ is defined by the well-known formula:

$$F[f] = f^F(k) = \sum_{n=0}^{N-1} \exp\left(i\frac{2\pi}{N}kn\right) f(n);\ k = \overline{0, N-1}, \tag{6}$$

where, obviously, the Fourier transformant $f^F(k)$m is also a periodic function with period $N$.

If we know the Fourier transformant $f^F(k)$, then we can recover the original function $f(n)$ using the inverse discrete Fourier transform:

$$F[f] = f^F(k) = \sum_{n=0}^{N-1} \exp\left(i\frac{2\pi}{N}kn\right) f(n);\ k = \overline{0, N-1}. \tag{7}$$

In general, the number of arithmetic operations $T_F(N)$, which is required to compute the discrete Fourier transform without the cost of computing functions of the form $\exp(i2\pi kn/N)$, is estimated by the formula:

$$T_F(N) \sim N^2. \tag{8}$$

When using fast discrete Fourier transform algorithms, the number of arithmetic operations required is estimated by the formula [23]:

$$T_{FF}(n) \sim N \cdot LOG(N), \tag{9}$$

where $LOG(N)$—is the integer logarithm, i.e., the sum of all prime divisors of $N$. If $N$ is a power of two, then $T_{FF}(n) \sim N \cdot log_2(N)$.

Let $A(l)$ bea periodic function of a discrete argument with period $N$. Consider the sums of the following form:

$$v(n) = \sum_{m=0}^{N-1} A(n-m)u(m), n = \overline{0, N-1}. \tag{10}$$

The sums in Equation (10) arise from multiplication of circulant matrices by a vector. Let us apply the discrete Fourier transform with period $N$ to both parts of (10). It is not difficult to show that:

$$v^F(k) = A^F(k)u^F(k), k = \overline{0, N-1}. \tag{11}$$

Using Equation (11) and fast discrete Fourier transform (FFT) algorithms, one can efficiently multiply circulant matrices by a vector. However, circulant matrices rarely appear in real-world problems. However, in many problems, in particular those discussed below, one needs to compute sums of the form Equation (10), in which the function $A(l), -(N-1) \leq l \leq (N-1)$ is arbitrary in the specified range. Such sums arise when multiplying Toeplitz matrices by the vector [31,32]. This function $A(l)$ is defined at the $(2N-1)$ integer point. Let us further define the function $A(l)$ zero at the point $l = N$ and extend it to all integer values with period *2N*. Further, for the function of the discrete argument $u(m),\ \ m = 0, \ldots, (N-1)$, let us define this as zero at the points $m = N, \ldots,\ 2N-1$. Now, consider the sums of the following form:

$$v(n) = \sum_{m=0}^{2N-1} A(n-m)u(m), n = \overline{0, 2N-1}. \tag{12}$$

It follows from the above that, at $n = 0, \ldots, N-1$ function $v(n)$ from Equation (12) coincides with the values $v(n)$ from Equation (10). Further, for quick calculation of the sums in Equation (12), we will use the formula:

$$v^F(k) = A^F(k)u^F(k), k = \overline{0, 2N-1}. \tag{13}$$

In the inverse Fourier transform, only the components of $\nu(n),\ n = \overline{0, N-1}$. Thus, it follows from Equation (9) that the number of arithmetic operations to calculate Equation (10) is estimated by the formula:

$$T_A \sim 2NLOG(2N). \tag{14}$$

Moreover, it is necessary to store in the computer memory an array with the number of elements equal to:

$$M_A \sim 2N. \tag{15}$$

Let us proceed to the discretization of the integral Equation (2). In a rectangular Cartesian coordinate system, define a parallelepiped $\Pi$, within which the region $Q$ *is* located. The edges of the parallelepiped are parallel to the coordinate axes, and the lengths of the edges are equal to $N_l h_l,\ l = 1, 2, 3$, where $h_l$ are the grid steps on the Cartesian coordinates. Then, the parallelepiped $\Pi$ can be represented as a union of cells (elementary parallelepipeds) $\Pi(p),\ p = (p_1, p_2, p_3),\ \ p_l = 0,\ \ldots, N_l - 1$. Let us define the area $\overset{\sim}{Q}$ as a union $N_Q$ of cells whose centers lie inside the area $Q$. The nodal points, in which the values of functions are defined, will be defined in the centers of cells and denoted as $x(p)$ and the values of functions in these points as $f(p)$.

The integral Equation (2) will be approximated, similarly to Equation (5), by an SLE of the following form [24]:

$$\begin{gathered} u(p) + \sum_{y(q) \in Q} A(p-q)\eta(q)u(q) = u^0(p),\ \ x(p) \in Q, \\ A(p-q) = -\mathrm{k}^2 \int_{\Pi(q)} \frac{K(x(p)-y)}{|x(p)-y|} dy,\ \ p \neq q,\ \ A(0) = 0. \end{gathered} \tag{16}$$

Since the nodal points are at the center of the cells, the accuracy of the approximation of the integral operator is $\sim h^2,\ h = \sqrt{h_1^2, h_2^2, h_3^2}$.

It follows from Equation (16) that the main computational cost of multiplying the SLE matrix by a vector (performing one iteration) is associated with the computation of sums of the form:

$$W(p) = \sum_{y(q)\in Q} A(p-q)V(q), x(p) \in Q. \tag{17}$$

To calculate $W(p)$ at the nodal points $x(p) \in Q$ requires performing $\sim N_Q^2$ arithmetic operations, where $N_Q$—is the number of nodal points in the domain $Q$. To reduce the number of arithmetic operations, we will apply the technique of fast multiplication of Toeplitz matrices by vector, as described above.

Let us define the function $V(q)$ as zero at the points $x(q)$ of the parallelepiped P, not belonging to the area $Q$. Consider the following sums:

$$W(p_1, p_2, p_3) = \sum_{q_1=0}^{N_1-1} \sum_{q_2=0}^{N_2-1} \sum_{q_3=0}^{N_3-1} A(p_1 - q_1, p_2 - q_2, p_3 - q_3)V(q_1, q_2, q_3). \tag{18}$$

It is obvious that at $x(p) \in Q$ values $W(p)$ from Equations (17) and (18) coincide. In Equation (18) the matrix function of the discrete argument $A(p)$ is defined for the values $-(N_1 - 1) \leq p_1 \leq (N_1 - 1); -(N_2 - 1) \leq p_2 \leq (N_2 - 1); -(N_3 - 1) \leq p_3 \leq (N_3 - 1)$.

Let us denote by $\Pi_2$ a parallelepiped with sides $2N_1h_1,\ 2N_2h_2$ и $2N_3h_3$. Let us continue the matrix function of the discrete argument $A(p_1, p_2, p_3)$ to all integer values $p_1, p_2, p_3$ assuming it to be periodic for each variable, with periods, respectively, $2N_1, 2N_2, 2N_3$. At the same time, let us define the function $A(p_1, p_2, p_3)$ as zero at the points where it is not defined. Next, let us define the function of the discrete argument $V(p_1, p_2, p_3)$ as zero at all nodal points $\Pi_2$, not belonging to $\Pi$, and extend this to all integer values $p_1, p_2, p_3$, assuming it is periodic for each variable, with periods, respectively, $2N_1, 2N_2, 2N_3$.

Consider the expression:

$$W(p_1, p_2, p_3) = \sum_{q_1=0}^{2N_1-1} \sum_{q_2=0}^{2N_2-1} \sum_{q_3=0}^{2N_3-1} A(p_1 - q_1, p_2 - q_2, p_3 - q_3)V(q_1, q_2, q_3). \tag{19}$$

Considering the above, it is clear that at $x(p) \in Q$ function $W(p_1, p_2, p_3)$ from Equation (19) coincides with the values from Equation (17). Below, by $\Pi$ и $\Pi_2$ we denote integer parallelepipeds with the number of discrete arguments on each axis $N_1, N_2, N_3$ и $2N_1, 2N_2, 2N_3$, respectively. Now, performing a discrete Fourier transform on each variable from both parts of Equation (19), we obtain the following equality:

$$W^F(k_1, k_2, k_3) = A^F(k_1, k_2, k_3)V^F(k_1, k_2, k_3),\ \ k \in \Pi_2. \tag{20}$$

Thus, to perform one iteration when solving the SLE Equation (16), it is necessary to perform the direct Fourier transform of the function $V(p_1, p_2, p_3)$ for each variable and the inverse transform of the function $W^F(k_1, k_2, k_3)$ (the transformation of the function $A(p_1, p_2, p_3)$ is performed once before starting the iteration procedure). The number of arithmetic operations and the amount of memory required to perform one iteration are estimated by the formulas:

$$T_A \sim 10N\ LOG(N),\ \ M_A \sim 10N,\ \ N = N_1N_2N_3. \tag{21}$$

When choosing grid steps and values $N_1, N_2, N_3$, it is necessary to be guided by the following criteria: first, the desired function does not change much within the cells; second, the region $\overset{\sim}{Q}$, consisting of cells whose centers are inside $Q$, describe $Q$ well enough.

Performance indicators of numerical algorithms. Having mentioned earlier that the solution of integral equations by numerical methods reduces to the solution of SLE with fully filled matrices, let us explain the main criteria for the efficiency of the algorithms, and demonstrate the visual advantage of iterative methods over direct methods in these classes of problem.

The main efficiency parameters of numerical algorithms are the number of operations $T$ required *to* solve the initial problem, the amount of memory $M$ required to implement the algorithm, the storage of $N^2$ elements of the SLE matrix with $N$ unknowns, etc. It is obvious that a large number of computational resources is required to solve the problem under consideration. In the case of the iterative method, these properties of the algorithm can be estimated using the relations:

$$T \sim LT_A, \;\; M \sim M_A, \tag{22}$$

where $T_A$ is the number of arithmetic operations required to multiply the SLE matrix by a vector, which is estimated in terms of complexity as $T_A \sim N^2$; $L$ is the number of iterations required to obtain a solution with a given accuracy; and $M_A$ is the number of unique elements of the matrix.

Let us denote by $T_G, \; M_G$ the characteristics of the direct methods for solving SLE. In comparison to this, when using the direct Gauss method to solve SLE, it is necessary to perform $T_G \sim N^3$ arithmetic operations and store in the computer memory at the order of $M_G \sim N^2$ numbers, which demonstrates the preference for using iterative methods in solving these problems.

Note that only iterative methods can be used to solve SLE Equation (16) using the considered algorithm. This is due to the fact that iterative algorithms are based on multiplications of the SLE matrix by a vector. The number of arithmetic operations and the amount of memory required to solve SLE Equation (16) are estimated by Formulas (21), (22). At the same time, the number of iterations required to obtain the solution is usually much smaller than the dimensionality of the SLE. Thus, it is possible to numerically solve the integral Equation (1) and, in the special case, Equation (2), which is reduced to the SLE of dimensionality $N_Q \geq 10^6$.

Iterative method for solving the problem. In the numerical formulation of Equation (5), and in particular Equation (16) for the case with approximation of the volumetric domain of the solution $Q$ by a system of parallelepipeds $\Pi(q), \; q \in \; Q$, we will solve the initial problem Equations (1) and (2) by the iterative method of gradient descent [33]. Let us describe the iterations and peculiarities of the method for solving a particular problem.

The task of finding a solution to an SLE is to solve the operator equation:

$$(H)u = b, \tag{23}$$

where $(H)$ is a known, in the general case a complex, matrix operator; $b$ is a known, in the general case complex, vector of the right-hand side; and $u$ is the vector whose values we want to find. When solving the problem Equations (1) and (2) by numerical methods, the values of the elements of the matrix operator $H$ are found as a result of discretization of Equations (5) and (16) of the initial integral equation and calculation of integrals over the obtained partitions $\Pi(q)$.

In addition, the action of the integral operator Equation (1) or Equation (2) with respect to the unknown vector $u$ is not the usual multiplication of a high dimensional matrix by a vector. Strictly speaking, it follows from Equation (2) that:

$$\begin{gathered} (H)u = u + A(\hat{\eta}u), \\ (H)u = u(p) + \sum_{y(q) \in Q} A(p-q)\hat{\eta}(q)u(q), p \in Q, \end{gathered} \tag{24}$$

where $A \in \mathbb{C}^{N_Q \times N_Q}$ is the matrix of coefficients of the integral equation kernel obtained as a result of calculation of Equations (4) and (5); $\hat{\eta} \in \mathbb{C}^{N_Q}$ is the vector of refraction values at each point of the $p$ of the given area partition $p \in Q$; $u \in \mathbb{C}^{N_Q}$, also in the general case, is the complex vector of unknown values of the scalar field strength at each discrete point of the given region. Each matrix-to-vector multiplication of any iterative algorithm that will solve the given problem of Equation (2) must be further defined according to Equation (24), resulting in iterated methods.

In the iterations of the modified iterative gradient descent method, the action of the conjugate operator is also presented $(H^*)$ on the vector $u$. Considering that, by definition for the matrix operator $H^* = \overline{H}^T$ where $\overline{H}$ is a matrix with complex-conjugate elements to the elements of matrix $H$, then the action of the operator Equation (24) can be rewritten in the form:

$$\begin{gathered}(H^*)u = u - \overline{\eta}\left(\overline{A}^T u\right),\\ (H^*)u = u(p) + \overline{\eta}(p) \sum_{y(q)\in Q} \overline{A}(q-p)u(q), p \in Q\end{gathered} \tag{25}$$

where $\overline{\eta}$ is also a refraction vector with complex-conjugate elements to vector $\hat{\eta}$. The iterative method aims to find an approximation $u_m \approx u$ of the unknown desired function. Iterations of the modified iterative method of gradient descent are defined as follows:

$$r_0 = (H)u_0 - b; u_1 = u_0 - \frac{\|(H^*)r_0\|^2}{\|(H)(H^*)r_0\|^2}(H^*)r_0; \tag{26}$$

$$\begin{gathered}r_m = (H)u_m - b;\ \Delta r_m = r_m - r_{m-1};\\ t_m\|\Delta r_m\|^2 + h_m Re(\Delta r_m, (H)(H^*)r_m) = Re(r_m, \Delta r_m),\\ t_m Re(\Delta r_m, (H)(H^*)r_m) + h_m\|\,(H)(H^*)r_m\|^2 = Re(r_m, (H)(H^*)r_m),\\ u_{m+1} = u_m - t_m(u_m - u_{m-1}) - h_m(H^*)r_m,\ \ m = 1, 2, \ldots.\end{gathered} \tag{27}$$

The only restriction on the iterative method for Equations (24) and (27) is the existence of a bounded inverse operator to $(H)$. Proofs of convergence of iterations for Equation (26) and convergence analysis of iterations and increasing dimension of the matrix are presented in [33].

As a criterion for stopping iterations for Equations (24) and (27), we choose the metric $\delta_m$ as the relative error of the approximated vector at step $m$ iterations:

$$\delta_m = \frac{\|u_m - u_{m-1}\|}{\|b\|} < \varepsilon, \tag{28}$$

where $u_m$ и $u_{m-1}$ are the obtained approximations of the unknown scalar field $u$ at the partition points; and $\varepsilon$ is the given accuracy of iterations, which is most often set as equal to $10^{-d+1}$, where $d$—is the number of significant digits of the computer representation of floating-point numbers.

As a result, we have a method that effectively copes with the problem of solving operator equations. The study and comparison of the iterative method for the real problem of solving SLE with a fully filled matrix was carried out in our previous work [25].

## 3. Results

Let us present the formulation of the conducted numerical experiment devoted to the solution of the acoustics problem Equation (2).

The rectangular solution domain $Q$ is characterized by linear dimensions $l = 1$ on each of the Cartesian axes $x = (x_1, x_2, x_3)$ and the point of the center of the cubic region $c = (0,\ 0,\ 0)$ at the origin of coordinates. The propagation of a plane wave in a medium is characterized by the value of the wave number $k = 15$ as well as the vector of the wave propagation direction $\overrightarrow{v} = \left(1/\sqrt{3},\ 1/\sqrt{3},\ 1/\sqrt{3}\right)$. The external function $f(x)$ from Equation (2) is modeled as complex harmonic oscillations:

$$f(x) = \exp\left(-ik\left(x, \overrightarrow{v}\right)\right),\ \ x \in Q, \tag{29}$$

with a given value of the wave number $k$ and vector of propagation direction $\overrightarrow{v}$.

Let us set for the time of the experiment the following function $\eta(x)$ of refraction of the transparent bulk medium:

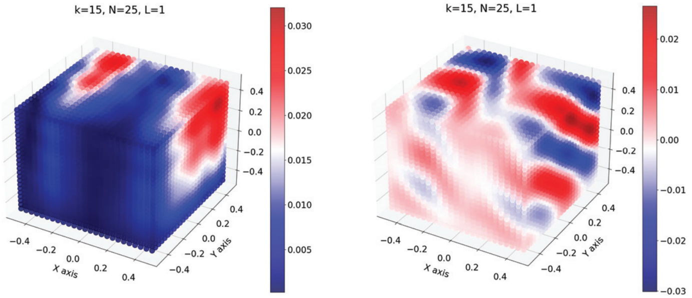
$$\eta(x) = \begin{cases} 2.5 + 1.5i = \eta_1(x), \;\; where \;\; x_1 \in (-l/4, l/4), \\ 1.0 + 0.0i = \eta_2(x), \;\; where \;\; x_1 \in \left[-\frac{l}{2}, -\frac{l}{4}\right] \cup \left[\frac{l}{4}, \frac{l}{2}\right]. \end{cases} \tag{30}$$

In fact, this formulation of the refraction function does not correspond in any way to a concrete real object, such as a transparent screen with appropriate physical properties. This value of refraction was chosen on the basis of considerations of conducting a numerical experiment with two media having different physical properties, in order to obtain the effect of refraction of a wave and attenuation of its energy when passing through a point $l/4$. A flat screen is defined by a stepwise refraction transition along the Cartesian coordinate $x_1$. As a result of Equation (30), we have a piecewise constant given function $\eta(x)$ of refraction of the region $Q$, which defines a transition from the medium with constant real refraction $\eta_2(x)$ to a constant complex refraction $\eta_1(x)$.

Discretization of the area $Q$ is according to Equation (16), into parallelepipeds of equal volume with equal sides. Modeling the problem in the cubic space, we will vary $N$, the number of such cubes, which are located along each of the axes of the Cartesian coordinate system, and the number of which along each of the coordinates is the same. Then the total number of partitions is strictly equal to $N_Q = N^3$. Within the results of this paper, we will compare the influence of the degree of discretization of the problem on the final quality of the solution of the problem for Equation (2).

In a series of numerical experiments, we computed (16) based on the modified iterative gradient descent method for Equations (24)–(27), in the private formulation of Equations (29)–(30), with specifically specified parameters. In each of the selected discretizations, for convenience, we set the stopping criterion of the iterative method $\varepsilon = 10^{-4}$ for the metric Equation (28).

The obtained solution characterizes the unknown value of the scalar field of intensity $u$ at each point of the center of partitioning of the initial region $Q$ into parallelepipeds of equal volume. Let us visualize the obtained results with sufficiently small $N = 25$ in Figure 1. The total number of obtainable elementary partitions in such a case is $N_Q = 15626$. The figure shows the obtained solution $u$ as a result of convergence of iterations for Equations (24)–(27). The first sub-drawing in Figure 1a shows the values of the $|u|$ of the potential field modulus in the center of each cube, which vary in the range of $|u| \in (0.0025, 0.0325)$. The gradation of these values is also highlighted in the graph. The sub-drawing Figure 1b shows the real part of the complex scalar field $u$ whose values vary in the range $Re\ u \in [-0.025, 0.025]$.

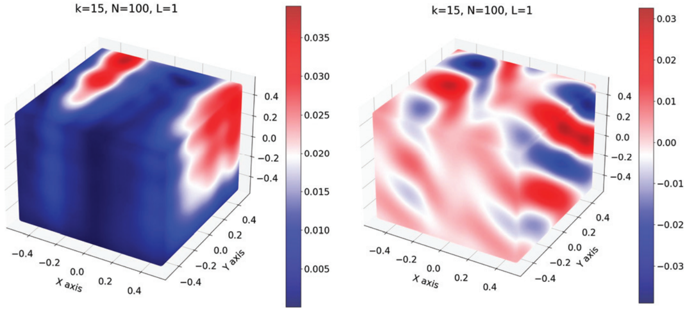


(**a**) (**b**)

**Figure 1.** Visualization of the values of the scalar field $u_m$ as a result of obtaining a solution by the iterative method with $N_Q = 15626$ at the intersection of the coordinates of a three-dimensional uniform grid; (**a**) the absolute values of the resulting field; (**b**) the real part of the complex scalar field values.

According to the character of the obtained picture, we obtain the expected type of the solution. In Figure 1b, one can notice the refraction of the plane wave Equation (29) incident from the point of $x = (0.5, 0.5, 0.5)$ against the boundary located at the point along the axis $x_1 = l/4 = 0.125$. In the limit $-l/4 < x_1 < l/4$, how the refracting wave changed the direction of its propagation can be seen, and as it passed within this region it experienced attenuation, due to which both the modulus of its value and the amplitude of its real values decreased. After passing through the point $x_1 = -l/4 = -0.125$, the propagation resumes due to the run-up of the plane wave front, as in Equation (29).

We also visualize the obtained results with a sufficiently large $\mathrm{N} = 100$ in Figure 2. The limits of the scalar field values remained the same, and also the obtained structure for the solution remained unchanged. The total number of obtainable elementary partitions in this case amounted to $\mathrm{N_Q} = 10^6$.



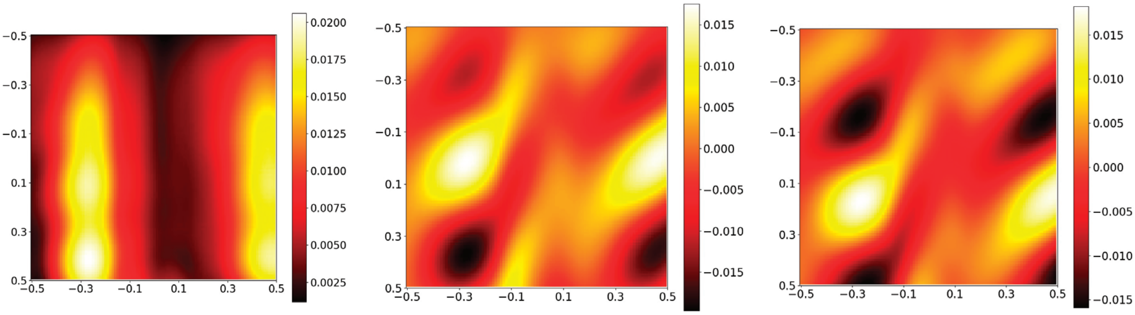
(**a**) (**b**)

**Figure 2.** Visualization of the values of the scalar field $u_m$ as a result of obtaining a solution by the iterative method with $N_Q = 100000$ at the intersection of the coordinates of a three-dimensional uniform grid; (**a**) the absolute values of the resulting field; (**b**) the real part of the complex scalar field values.

As a result, having increased the linear partitioning of the domain by approximately a factor of 4, we obtain a 64-times increase in the discretization of the domain $N_Q$, in view of which the obtained solutions show more features at the refraction boundary. Let us note the smoothness of the obtained solution, as a result of which it is possible to take into account small features both on the refraction boundary and on possible inclusions, whose sizes do not exceed the linear sizes of the obtained partitions $\Pi(q)$.

On the center slices on the axis $x_3 \approx 0$ areas $Q$ in Figure 3, we also see the picture of the refraction and attenuation of the wave, both in the modulus value, Figure 3a, and in the values of amplitudes of the real Figure 3b and the complex part of the wave Figure 3c. On the graph of the modulus approximation of the scalar field values $|u_m|$, Figure 3a, we can clearly see the transition in the range of $-l/4 < x_1 < l/4$, which is a consequence of the transition of the refraction value in this region. In Figure 3b,c, we see refraction in the pattern of propagation of the plane wave, as well as the attenuation of the amplitude of the scalar field values within $-l/4 < x_1 < l/4$.

Performing calculations with different degrees of discretization of the solution domain, we measure the efficiency of the presented iterative method, Equations (24)–(27), using values of different metrics. For solutions with different discretization of the domain $Q$, we measured the total number of matrix multiplications by the vector $m$ during the algorithm operation, the relative error of approximations of iterations $\delta_m$ counted by (28), the norm of incoherence of the obtained approximate solution $||(H)u_m - b||$, and the maximum value of the modulus of the scalar field $\max\limits_{p \in Q} |u_m(p)|$. In addition to the above metrics, we measured the program execution time in seconds ($t,\ sec$) for a given discretization ($N$) of the volume area of the solution on a personal computing device with an intel core i5-10400f processor and DDR4 RAM, with a frequency of 3200 MHz and a volume of 32 GB. Time was measured via the implementation of a C++ program in a sequential single-threaded execution of instructions.



(**a**) (**b**) (**c**)

**Figure 3.** Visualization of the values of the scalar field $u_m$ as a result of obtaining a solution by the iterative method with $N_Q = 100000$ at slices of visualization Figure 2 at central position of $x_3$ axis; (**a**) absolute values of resulting field; (**b**) real part of complex scalar field values; (**c**) image part of complex scalar field values.

Table 1 demonstrates the values of the above metrics for different numbers of partitions $N_{Q.}$ source region $Q$.

**Table 1.** Table of performance and convergence quality of the method with different numbers of partitions.

| N | $N_Q$ | m | $\delta_m$ | $\|\|(H)u_m - b\|\|$ | $\max\limits_{p\in Q}\|u_m(p)\|$ | t,s |
|---|---|---|---|---|---|---|
| 10 | 1000 | 48 | $9.652 \times 10^{-5}$ | $4.221 \times 10^{-5}$ | $5.618 \times 10^{-6}$ | 0.833 |
| 15 | 3375 | 48 | $3.587 \times 10^{-5}$ | $3.854 \times 10^{-5}$ | $3.198 \times 10^{-6}$ | 2.895 |
| 20 | 8000 | 42 | $5.366 \times 10^{-5}$ | $2.984 \times 10^{-4}$ | $1.491 \times 10^{-5}$ | 6.131 |
| 25 | 15,625 | 45 | $4.784 \times 10^{-5}$ | $2.601 \times 10^{-4}$ | $9.216 \times 10^{-6}$ | 13.065 |
| 30 | 27,000 | 57 | $7.434 \times 10^{-5}$ | $4.631 \times 10^{-4}$ | $1.100 \times 10^{-5}$ | 29.069 |
| 35 | 42,875 | 45 | $8.941 \times 10^{-5}$ | $2.603 \times 10^{-4}$ | $6.935 \times 10^{-6}$ | 36.775 |
| 40 | 64,000 | 42 | $6.731 \times 10^{-5}$ | $3.751 \times 10^{-4}$ | $6.805 \times 10^{-6}$ | 51.703 |
| 45 | 91,125 | 42 | $5.311 \times 10^{-5}$ | $3.083 \times 10^{-4}$ | $5.210 \times 10^{-6}$ | 74.258 |
| 50 | 125,000 | 54 | $2.992 \times 10^{-5}$ | $5.041 \times 10^{-4}$ | $6.793 \times 10^{-6}$ | 132.335 |
| 100 | 1,000,000 | 42 | $9.680 \times 10^{-5}$ | $1.273 \times 10^{-4}$ | $5.321 \times 10^{-6}$ | 1039.317 |

According to the results, we note that there is no dependence of the investigated values of the metrics on the number of partitions of the solution domain. It is shown that the number of matrix multiplications by vector $m$ necessary to obtain an approximation $u_m$ with a predetermined accuracy $\varepsilon$ does not increase depending on the number of partitions of the domain $Q$. However, we observe an oscillation of this value, which may be due to a poor choice of cube sizes, which resulted in an inaccurate description of the refraction region.

Metrics $\delta_m$, $||(H)u_m - b||$ do not change depending on the number of partitions; we can only note the low value of the norm of non-convexity of the integral operator with respect to the obtained solution at small discretizations $N_Q = 1000$ и $N_Q = 3375$. This circumstance can also be related to the number of final calculations or to a weak description of the solution domain of the problem. The index $\max\limits_{p\in Q}|u_m(p)|$ also varies within the error limits.

We display the graphs of the metric Equation (28) of the relative error $\delta_m$ of approximations during the iterative method for Equations (24)–(27) depending on the number of applications of the matrix-to-vector multiplication operation $m$. Figure 4 shows the value of $\log_{10}\delta_m$ depending on $m$, due to the exponential nature of the reduction of this metric. By prologarithmizing $\delta_m$, we see the convergence pattern of the method in Equations (24)–(27)

for the problem in Equations (29)–(30), in the values of powers of 10 of the base of the number system.

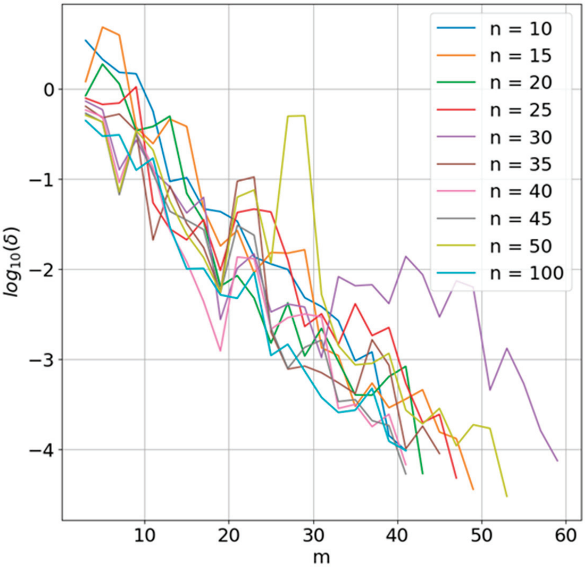


**Figure 4.** Graph of the relative error of iterations Equation (28) for the iterative method min Equations (24)–(27) as a function of the number of matrix-to-vector multiplications $m$.

In the linearized dependence shown, we see by the convergence pattern in Figure 4 an oscillating graph of metric Equation (28), demonstratively converging (in natural values) to the required value of approximation accuracy $\varepsilon$. It can be seen from the graph that, with increasing discretization (n on the graph is equal to $N$ or $N_1$, $N_2$ or $N_3$) of the initial domain, the number of iterations (matrix-to-vector multiplications) does not increase, which cardinally contributes to the speed in solving the initial formulation of the integral equation at large discretizations.

## 4. Discussion

The solution of problems using the most general approaches to approximation of complex regions is widely used in various works on mathematical modeling of wave effects on physical bulk multilayer objects of different shapes with different boundary conditions. To date, there are many ways to solve wave propagation and scattering problems in volumetric media, in particular for the problem of acoustics, in which, often, complex discretizations based on tetrahedral meshes are applied. The approach based on tetrahedra as units of elementary partitioning of the three-dimensional solution domain is due to the need for the most accurate description of complex domain boundaries to the detriment of the possibility of fast computations of problems on uniform mesh approximations.

The computational approach demonstrated in this paper will be useful in the case of a large number of different refractive inclusions that can be described by rectangular parallelepipeds. This method will also be useful in modeling subtle wave processes due to the possibility of performing efficient computations over fully filled matrices of large size, $N_Q \geq 10^6$. The condition of applicability of this computational method is the requirement of the presence of the kernel of the integral operator depending on the coordinate difference; therefore, the main computations are associated with the computation of the coefficients of the operator in the form of the Teuplice matrix. The numerical experiment presented demonstrates an effective approach to solving such a problem formulation, as well as qualitative pictures for solving the problems of acoustic wave propagation in a medium with attenuation and wave scattering at the refraction boundary of transparent bulk media.

It is noteworthy that there are no requirements for the structure of the refraction function, except for the physical limitations of the modeled materials, which makes it possible to perform mathematical modeling of the problems presented above for areas of high complexity. The main limitation will be the size of the area on which the function will be set.

The modified iterative method of gradient descent proposed for solving the problem has shown itself to be an effective method for solving the operator equation with a fully filled operator matrix. According to the results of numerical experiments, it is shown that it converges steadily when the dimensionality of the problem changes and also converges steadily for a reasonable number of matrix-to-vector multiplications (about 100 operations).

Moreover, when the dimensionality of the problem increases, the number of such operations does not change, which is associated only with the physical properties of the problem formulation. The obtained numerical scheme allows us to seriously accelerate computations in general formulations of the solution of SLE or operator equations and, with the use of special discretization, will also allow us to solve a large number of unknowns in applied modeling problems.

It is also worth noting that, due to the currently prevailing most accurate approach to describing a three-dimensional domain based on complex elementary partitions, comparative results with other studies cannot be given.

## 5. Conclusions

Today, the problem of economical fast computations for mathematical models of physical processes to provide high quality and fast computations over digital doubles of real in-situ experiments is acute. The development of efficient numerical methods for solving a narrow class of problems in mathematical physics makes it possible to solve such problems with increasing requirements on the degree of discretization due to the reduced complexity of calculations and the required memory for the implementation of the algorithm. This can be achieved by using the peculiarities of the kernels of the initial formulation in the form of solving Fredholm integral equations describing mathematical models of some real processes.

This paper shows how, taking into account the features of the integral equation kernel function and introducing a special spatial grid, it is possible to obtain a multiple acceleration of calculations and a reduction in the required memory for the realization of calculations.

The paper proposed a formulation for solving problems in mathematical physics using integral equations and iterative solution methods. The theoretical efficiency of application of fast Fourier transform in the optimization of multiplication of the Toeplitz matrix by a vector is shown. Numerical results are demonstrated on the model problem of acoustic wave propagation in a transparent bulk medium with an inhomogeneous refraction index.

In future works, it is assumed that the use of this numerical method for discretizing the volume domain of the solution, the technique of multiplying a Toeplitz matrix by a vector for optimization of iterative methods according to a given discretization form, as well as they setting up of the solution of the integral equation problem, will be applied in related tasks in the propagation and scattering of waves in electromagnetism, which will also find its application in applied areas such as radio-physics, remote sensing, radio-spectroscopy, and crystal growth.

Continuing this work will also be considered towards optimizing the numerical method of solving linear systems with Toeplitz-type matrices, both in terms of specifically optimizing multiplication with the use of multilevel Toeplitz matrices and in terms of optimizing the iterative method proposed. This direction of work will be associated with attempts to simplify the use of the proposed methodology for solving volumetric integral equations in various related tasks, as well as to speed up computations and reduce estimates of the required memory for iterations.

**Author Contributions:** Conceptualization, A.B.S.; Methodology, A.B.S.; Software, I.A.Y.; Validation, I.A.Y.; Resources, I.A.Y.; Writing—original draft, A.B.S.; Writing—review & editing, I.A.Y.; Visualization, I.A.Y. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by Russian Science Foundation, grant number 20-11-20087.

**Data Availability Statement:** No new data were created or analyzed in this study. Data sharing is not applicable to this article.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Mikhlin, S.G.; Morozov, N.F.; Paukshto, M.V. *Integral Equations in the Theory of Elasticity*; SPbSU Publishing House: Saint-Petersburg, Ruassia, 1994. (In Russian)
2. Mikhlin, S.G.; Prössdorf, S. *Singular Integral Operators*; Academie-Verlag: Berlin, Germany, 1986.
3. Mikhlin, S.G. *Multivariate Singular Integrals and Integral Equations*; Fizmatgiz: Moscow, Russia, 1962. (In Russian)
4. Colton, D.; Kress, R. *Inverse Acoustic and Electromagnetic Scattering Theory*; Springer: Berlin/Heidelberg, Germany, 1992.
5. Miller, E.K. A Selective Survey of Computational Electromagnetics. *IEEE Trans. Antennas Propag.* **1988**, *36*, 1281–1305. [CrossRef]
6. Dobrovol'skii, N.N.; Skobel'tsyn, S.A.; Tolokonnikov, L.A.; Larin, N.V. About application of number-theoretic grids in problems of acoustics. *Chebyshevskii Sb.* **2021**, *22*, 368–382. (In Russian) [CrossRef]
7. Turq, S.M. Homotopy Type Methods for Nonlinear Differential Equations. Master's Thesis, Palestine Polytechnic University, Hebron, Palestine, 2020. [CrossRef]
8. Xu, J.; Ludmil, Z. Algebraic Multigrid Methods. *Acta Numer.* **2017**, *26*, 591–721. [CrossRef]
9. Cogswell, D.A.; Szulczewski, M.L. Simulation of Incompressible Two-Phase Flow in Porous Media with Large Timesteps. *J. Comput. Phys.* **2017**, *345*, 856–865. [CrossRef]
10. Fu, S.; Li, G.; Craster, R.; Guenneau, S. Wavelet-Based Edge Multiscale Finite Element Method for Helmholtz Problems in Perforated Domains. *Multiscale Model. Simul.* **2021**, *19*, 1684–1709. [CrossRef]
11. Wei, Y.J.; Rabinovich, A. The inverse problem of permeability identification for multiphase flow in porous media. *Phys. Fluids* **2023**, *35*, 073327. [CrossRef]
12. Liu, T.; Xue, R.; Liu, C.; Qi, Y. A Regularization Homotopy Strategy for the Constrained Parameter Inversion of Partial Differential Equations. *Entropy* **2021**, *23*, 1480. [CrossRef] [PubMed]
13. Nilssen, T.K.; Karlsen, K.H.; Mannseth, T.; Tai, X.-C. Identification of diffusion parameters in a nonlinear convection–diffusion equation using the augmented Lagrangian method. *Comput. Geosci.* **2009**, *13*, 317–329. [CrossRef]
14. Vainikko, G. *Multidimensional Weakly Singular Integral Equations*; Springer: Berlin/Heidelberg, Germany, 2006.
15. Smirnov, Y.G.; Kondyrev, O.V. About Fredholm and solvability of the system of integral equations in the conjugation problem for the Helmholtz equation. *Differ. Equ.* **2023**, *59*, 1089–1097. [CrossRef]
16. Smirnov, Y.G. On the Fredholm character of the system of integral equations in the problem of electromagnetic wave propagation in a rod covered with graphene. Izvestiya vysshee obrazovaniya vysshee obrazovaniya [Izvestia of Higher Educational Institutions]. *Volga Reg. Phys.-Math. Sci.* **2023**, *3*, 74–86. (In Russian) [CrossRef]
17. Smirnov, Y.G. On the propagation of nonlinear coupled surface TE and resulting TM electromagnetic waves in a circular metal-dielectric waveguide. In *V Scientific Forum Telecommunications: Theory and Technology TTT-2021: Proceedings of the XIX International Scientific and Technical Conference, Samara, Russia, 23–26 November 2021*; Volga Region State University of Telecommunications and Informatics: Samara, Russia, 2021; pp. 221–222. (In Russian)
18. Setukha, A.V.; Tretyakova, R.M. Numerical solution of a stationary filtration problem of viscous fluid in a piecewise homogeneous porous medium by applying the boundary integral equation method. *Comput. Math. Math. Phys.* **2020**, *60*, 2076–2093. [CrossRef]
19. Chashechkin, Y.D. Acoustics and hydrodynamics of the drop impact: Two modes of sound packets emission. *Bull. Bauman Mosc. State Tech. University. Ser. Nat. Sci.* **2023**, *106*, 23–44. (In Russian) [CrossRef]
20. Uvaysov, S.U.; Luu, N.T.; Nguyen, C.D.; Vo, T.H.; Dolmatov, A.V. Detection of defects in printed circuit boards by the acoustic emission method. *Russ. Technol. J.* **2024**, *12*, 15–29. [CrossRef]
21. Nefedov, V.I.; Trefilov, D.N.; Dementiev, A.N.; Vetrova, V.V.; Kolesnikov, S.M.; Shpak, A.V. Integral equations for modeling cylindrical mirror antennas. *Russ. Technol. J.* **2017**, *5*, 124–129. (In Russian) [CrossRef]
22. Dmitriev, V.I.; Zaharov, E.V. *Integral Equations for Boundary Problems of Electromagnetics*; VSU: Moscow, Russia, 1987. (In Russian)
23. Samokhin, A.B. *Volume Singular Integral Equations of Electromagnetics*; Technosfera: Moscow, Russia, 2021. (In Russian)
24. Samokhin, A.B. Methods and effective algorithms for solving multidimensional integral equations. *Russ. Technol. J.* **2022**, *10*, 70–77. [CrossRef]
25. Samokhin, A.B.; Samokhina, A.S.; Yurchenkov, I.A. Fredholm Integral Equation for Problems of Acoustic Scattering by Three-Dimensional Transparent Structures. *Diff. Equat.* **2023**, *59*, 1256–1261. [CrossRef]
26. Vasilev, E.N. *Excitation of Bodies of Rotation*; Radio i svyaz: Moscow, Russia, 1987. (In Russian)
27. Ilinsky, A.S.; Kravcov, V.V.; Sveshnikov, A.G. *Mathematical Models of Electromagnetics*; Vysshaya shkola: Moscow, Russia, 1991. (In Russian)

28. Demidovich, B.P. Numerical methods of analysis. In *Approximation, Differential and Integral Equations*; Lan: St. Petersburg, Russia, 2021. (In Russian)
29. Nurutdinova, I.N. Integrals. In *Differential Equations*; DSU: Rostov on Don, Russia, 2021. (In Russian)
30. Samarsky, A.A.; Gulin, A.V. *Numerical Methods*; Nauka: Moscow, Russia, 2002. (In Russian)
31. Voevodin, V.V.; Tyrtyshnikov, E.E. *Computational Processes with Toeplitz Matrices*; Nauka: Moscow, Russia, 1987. (In Russian)
32. Tyrtyshnikov, E.E. *Methods of Numerical Analysis*; MSU: Moscow, Russia, 2006. (In Russian)
33. Samokhin, A.B.; Sklyar, A.Y.; Shestopalov, Y.V.; Samokhina, A.S. Iterative Gradient Descent Methods for Solving Linear Equations. *Comput. Math. Math. Phys.* **2019**, *59*, 1267–1274. [CrossRef]

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

mathematics

MDPI

*Article*

# A Method for Transforming Non-Convex Optimization Problem to Distributed Form

**Oleg O. Khamisov [1], Oleg V. Khamisov [1,*], Todor D. Ganchev [2,*] and Eugene S. Semenkin [3,*]**

[1] Depertment of Applied Mathematics, Melentiev Energy Systems Institute, 664033 Irkutsk, Russia; cygx151@gmail.com

[2] Department of Computer Science and Engineering, Technical University of Varna, 9010 Varna, Bulgaria

[3] Scientific and Educational Center "Artificial Intelligence Technologies", Baumann Moscow State Technical University, 105005 Moscow, Russia

* Correspondence: khamisov@isem.irk.ru (O.V.K.); tganchev@tu-varna.bg (T.D.G.); e.semenkin@emtc.ru (E.S.S.)

**Abstract:** We propose a novel distributed method for non-convex optimization problems with coupling equality and inequality constraints. This method transforms the optimization problem into a specific form to allow distributed implementation of modified gradient descent and Newton's methods so that they operate as if they were distributed. We demonstrate that for the proposed distributed method: (i) communications are significantly less time-consuming than oracle calls, (ii) its convergence rate is equivalent to the convergence of Newton's method concerning oracle calls, and (iii) for the cases when oracle calls are more expensive than communication between agents, the transition from a centralized to a distributed paradigm does not significantly affect computational time. The proposed method is applicable when the objective function is twice differentiable and constraints are differentiable, which holds for a wide range of machine learning methods and optimization setups.

**Keywords:** distributed optimization; non-convex optimization; gradient descent; Newton's method

**MSC:** 68W15; 68Q85

**Citation:** Khamisov, O.O.; Khamisov, O.V.; Ganchev, T.D.; Semenkin, E.S. A Method for Transforming Non-Convex Optimization Problem to Distributed Form. *Mathematics* **2024**, *12*, 2796. https://doi.org/10.3390/math12172796

Academic Editor: Jüri Majak

Received: 29 June 2024
Revised: 23 August 2024
Accepted: 29 August 2024
Published: 9 September 2024

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

## 1. Introduction

In modern society, digital technologies play an essential role in organizing our work and daily routine. Ubiquitous computing and digital technologies enable us to solve a wide range of complex problems in such important fields as ecology [1,2] and medicine [3,4].

The complexity of creating decision support systems in a digital environment requires the use of advanced technologies for designing and optimizing intelligent information processing systems. For example, within a holistic approach to integrating computational intelligence systems and human expert knowledge [5], it is possible to automatically design machine learning models with self-tuning adaptive stochastic optimization algorithms [6,7]. In this case, the processed data can remain on the problem owner's servers, which ensures trust and allows us to remain within a federated approach to learning, and the resulting models will be interpretable and explainable [8]. However, very large-scale optimization problems arising under such conditions require special computations decomposition methods [9]. At the same time, in many cases, optimization problems of this kind have properties that allow the use of rigorous mathematical methods in their solution, which makes it possible to effectively use hybrid approaches [10]. In this regard, along with the improvement in adaptive methods of computational intelligence, the evolution of traditional optimization methods is of great importance. The aim of such advancements could be focused on adaption to problems of extremely high dimensionality and federated learning through the decentralization of work and to use such properties of theirs as guaranteed convergence to the optimum at high speed, which is fundamentally important in the tasks under consideration.

Specifically, in the following, we propose a novel decentralized optimization method for non-convex optimization problems with a separable objective function and coupling equality and inequality constraints. Under standard assumptions for distributed optimization [11,12], the problem has to be solved by a set of agents communicating over a connected graph. These agents are expected to communicate synchronously and can transmit real-valued numbers to adjacent agents. Communications are performed synchronously, and communication delays and packet losses are ignored. In addition to standard conditions, agents cannot share their decision variables and objective functions as they are considered to be private information.

Decentralized optimization already proved to be an essential instrument in a wide variety of applications. Namely, application in optimal transport [13] including coordination of mobile autonomous agents [14,15] and railway traffic [16,17], power systems control [18,19] with demand response [20] as well as data analysis in sensor networks [21,21]. Finally, decentralized optimization gains increasing popularity in federated learning [22] and support vector machines [23].

### *1.1. Related Work*

While decentralized optimization has many applications of practical relevance, most of the corresponding results are dedicated to convex or strictly convex optimization. A comprehensive survey covering these areas can be found in [24]. The majority of these methods can be separated into primal [13,25], dual [26] or ADMM-based approaches [27,28]. Additionally, there exists a set of works utilizing the primal–dual approach [29,30].

The literature dedicated to non-convex or non-linear constraints is significantly more scarce. One of the proposed approaches is the application of SQP with the inner ADMM method [31]. However, in this work, coupling constraints are linear. Lagrangian methods for polynomial objective and equality constraints are presented in [32], and non-linear coupling constraints are considered. However, each constraint is associated with one of the agents and coupling is present only with the variables of adjacent agents.

Finlay, there exist several works that consider convex optimization problems with separable objective functions such that decision variables and corresponding summands of objective functions are considered private for each agent and cannot be exchanged with other participants [19,33]

### *1.2. Contribution*

Here, we propose a novel distributed optimization algorithm for non-convex optimization problems with equality and inequality constraints. It is assumed that the objective function is twice differentiable and the constraints are differentiable. In addition, communications are significantly less time consuming than oracle calls. Under these assumptions, we show that

1. The proposed method can be applied to any optimization problem with non-convex separable objective function and coupling constraints.
2. Its convergence rate is equivalent to the convergence of Newton's method with respect to oracle calls.
3. Decision variables, cost and constraint functions are not exchanged between agents.

The theoretical results are supported by numerical experiments.

### *1.3. Paper Organization*

The remainder of this article is organized as follows. Section 2 introduces the problem statement. Section 3 is dedicated to the reformulation of optimization problems with equality constraints. Section 4 outlines the distributed gradient descent algorithm. In Section 5, a problem with equality and inequality constraint and its equivalent formulation is presented. Section 6 outlines the distributed Newton method. Finally, Section 7 presents the Arrow–Hurwicz method and a numerical example.

### *1.4. Notations*

Let $\mathbf{1}_K$ denote the vector of ones of the size $K$; $I^q$ is the identity $q \times q$ matrix. Operator vec is the vectorization operator for a matrix $A$; ker $A$ is the matrix kernel. For two matrices $A$ and $B$, the Kronecker product is denoted by $A \otimes B$, and $\operatorname{diag}(A, B)$ means the extended matrix of the corresponding size with $A$ and $B$ as the blocks on the main diagonal. For a twice differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, $\nabla f$ and $\nabla^2 f$ are the gradient and Hessian matrix, respectively. For a vector function $g : \mathbb{R}^n \to \mathbb{R}^m$, its Jacobian matrix is denoted by $Jg$. For a vertex (agent) $i$ in a communication graph, a set of adjacent vertices is defined by $\operatorname{Adj}(i)$.

## **2. Problem Statement**

Let us consider a non-convex optimization problem with a coupling objective function and constraints that must be solved by a multi-agent connected network with $N$ vertices (agents). The optimization problem has the following form:

$$\min_x \left\{ F(x) = \sum_{i=1}^{N} f^i(x^i) \right\}, \tag{1a}$$

subject to

$$G(x) = \sum_{i=1}^{N} g^i(x^i) = 0, \tag{1b}$$

$$H(x) = \sum_{i=1}^{N} h^i(x^i) \leq 0. \tag{1c}$$

Here, the vector of objective variables $x \in \mathbb{R}^n$ is separated into subvectors of local variables $x^i \in \mathbb{R}^{n_i}, i \in \{1, \ldots, N\}$, $n_1 + n_2 + \cdots + n_N = n$. All functions $f^i : \mathbb{R}^{n_i} \to \mathbb{R}$, $g^i : \mathbb{R}^{n_i} \to \mathbb{R}^{\widetilde{m}}$ and $h^i : \mathbb{R}^{n_i} \to \mathbb{R}^{\widehat{m}}$, $i \in \{1, \ldots, N\}$ are smooth.

Here, we postulate that the Problem (1) has to be solved in the distributed way, which brings the following perspective. It is assumed that each subvector $x^i$ belongs to an agent $i$ and cannot be shared with the other agents, and the agent network is defined by a connected graph with the Laplacian matrix $L \in \mathbb{R}^{N \times N}$. Every node in the graph represents some agent. Communication in the network is allowed only between neighboring vertices (agents). An agent $i$ is characterized by its objective function $f^i$, equality constraint function $g^i$ and inequality constrain function $h^i$. The objective function $F$ is separable, so the main difficulty in deriving a distributed version of problem (1) is given by constraints (1b) and (1c). They are coupling (non-local) even though the constraint functions $G$ and $H$ are also separable.

Subsequently, we set the goal to develop an algorithm and present a reduction, which, for an arbitrary problem (1), creates an auxiliary problem, such that

1. The auxiliary problem has the same solution as (1);
2. Methods of gradient descent, gradient projection and quasi-Newton methods will operate as distributed optimization methods when applied to the auxiliary problem without any modification (1):

## **3. Optimization Problem with Equality Constraints**

Let us first introduce a simplified version of the problem (1);

$$\min_{x \in \mathbb{R}^n} \left\{ F(x) = f^1(x^1) + f^2(x^2) + \cdots + f^N(x^N) \right\}, \tag{2a}$$

subject to

$$g_1^1(x^1) + g_1^2(x^2) + \cdots + g_1^N(x^N) = 0, \tag{2b}$$

$$g_2^1(x^1) + g_2^2(x^2) + \cdots + g_2^N(x^N) = 0, \tag{2c}$$

$$\cdots$$

$$g_{\widetilde{m}}^1(x^1) + g_{\widetilde{m}}^2(x^2) + \cdots + g_{\widetilde{m}}^N(x^N) = 0, \tag{2d}$$

i.e., the problem defined as in (1) without the inequality constraints. We introduce into consideration an auxiliary vector $y \in \mathbb{R}^{N\widetilde{m}}$ consisting of subvectors $y^i \in \mathbb{R}^{\widetilde{m}}$, $y^i = (y_1^i, y_2^i, \dots, y_{\widetilde{m}}^i)^\top$, $i = 1, \dots, N$. Each subvector $y^i$ is connected to the constraint function $g^i$ of the vertex $i$. Consider the $j$-th scalar equality constraint from (2):

$$\sum_{i=1}^{N} g_j^i(x^i) = 0. \tag{3}$$

It can be reformulated to the following form:

$$\begin{aligned} g_j^1(x^1) + \sum_{i=1}^{N} L_{1i} y_j^i &= 0, \\ g_j^2(x^2) + \sum_{i=1}^{N} L_{2i} y_j^i &= 0, \\ &\vdots \\ g_j^N(x^N) + \sum_{i=1}^{N} L_{Ni} y_j^i &= 0, \end{aligned} \tag{4}$$

where $L_{si}$, $s, i = 1, \dots, N$ are elements of the Laplacian matrix $L$. Such a transition is performed for all $\widetilde{m}$ equality constraints. Thus, in the further consideration, we use the following notation:

$$\widetilde{g} = \mathrm{vec}\Big(g^1, g^2, \dots, g^N\Big), \tag{5}$$

$$\widetilde{L} = \Big(L \otimes I^{\widetilde{m}}\Big), \tag{6}$$

where $L \otimes I^{\widetilde{m}}$ is the Kronecker product of the Laplacian matrix $L$ and the identity matrix $I^{\widetilde{m}}$. The interpretation of representation (5) is given in Figure 1.

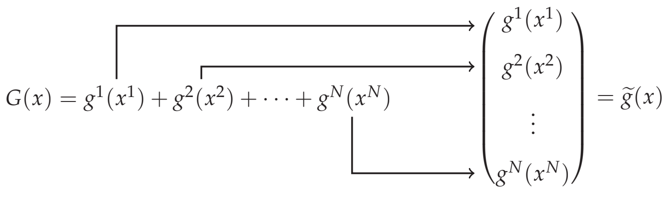


**Figure 1.** The agent network with information available at each node.

Repeating the transition from (3) to (4) for all $j = 1, \dots, \widetilde{m}$ yields the following reformulation of problem (2):

$$\min_{(x,y) \in \mathbb{R}^n \times \mathbb{R}^{N\widetilde{m}}} F(x), \tag{7a}$$

$$\widetilde{g}(x) + \widetilde{L}y = 0. \tag{7b}$$

**Lemma 1.** *The following statements are correct:*

1. *Problem* (2) *is feasible if and only if problem* (7) *is feasible.*
2. *A pair* $(x^*, y^*)$ *is the solution to problem* (7) *if and only if* $x^*$ *is the solution to* (2)*.*

**Proof.** *1.* Consider system (4) as a system of linear equations with respect to variables $(y_j^1, y_j^2, \dots, y_j^N)$ for fixed $x$ and right-hand side vector $(-g_j^1(x^1), -g_j^2(x^2), \dots, -g_j^N(x^N))$. According to the Fredholm Alternative [34] and due to the symmetricity of $L$, this system is consistent if and only if

$$\operatorname{vec}\Big(g_j^1(x^1), \dots, g_j^N(x^N)\Big) \perp \ker L, \tag{8}$$

where $\ker L = \{v \in \mathbb{R}^N : Lv = 0\}$ is the kernel of the Laplacian matrix $L$. Since the agent graph is connected, $\ker L = \{v \in \mathbb{R}^N : v = \rho \mathbf{1}_N,\ \rho \in \mathbb{R}, \rho \neq 0\}$. Then, (8) is equivalent to

$$\operatorname{vec}\Big(g_j^1(x^1), \dots, g_j^N(x^N)\Big)^\top \mathbf{1}_N = \sum_{i=1}^N g_j^i(x^i) = 0. \tag{9}$$

Repeating this consideration for all $j = 1, \dots, m$, we find that problems (2) and (7) are feasible simultaneously.
*2.* The correctness of the second statement follows from the fact that both problems have the same objective function. □

Let us consider the main property of system (4). Since $L$ is the Laplacian matrix, this system can be rewritten in the following form:

$$\begin{aligned} g_j^1(x^1) + \sum_{i \in \operatorname{Adj}(1)} (y_j^1 - y_j^i) &= 0, \\ g_j^2(x^2) + \sum_{i \in \operatorname{Adj}(2)} (y_j^2 - y_j^i) &= 0, \\ &\vdots \\ g_j^N(x^N) + \sum_{i \in \operatorname{Adj}(N)} (y_j^N - y_j^i) &= 0. \end{aligned} \tag{10}$$

In order to evaluate the $\ell$-th constraint in (10), it is necessary to know the local variables $x^\ell$, $y_j^\ell$, local function $g_j^\ell$ and variables $y_j^i$ from the neighboring vertices $i \in \operatorname{Adj}(\ell)$ only. This is the main advantage of system (10) in comparison to constraint (3), for which it is necessary to know information from all vertices of the agent network.

If we fix vector $y$, for example, $y = \tilde{y}$, then, due to the separability of function $F$ (see (2a)) and property (10), optimization with respect to the remaining vector $x$ in problem (7) can be performed separately, i.e., each agent $\ell$ independently solves the corresponding problem:

$$\min_{x^\ell} f^\ell(x^\ell), \tag{11}$$

$$g_j^\ell(x^\ell) = -\sum_{i \in \operatorname{Adj}(\ell)} (\tilde{y}_j^\ell - \tilde{y}_j^i),\ j = 1, \dots, \widetilde{m}. \tag{12}$$

Assume that problems (11) and (12) are solvable for all $\ell = 1, \dots, N$, and $\widetilde{x}^\ell$ is the corresponding solutions. The vector $\widetilde{x} = \operatorname{vec}(\widetilde{x}^1, \dots, \widetilde{x}^N)$ provides the solution of problem (7) for fixed $y = \widetilde{y}$.

## 4. Gradient Descent in Variables *y*

Variables $y$ are called communication variables. If we set $\widetilde{y}^\ell = y^{0,\ell} = 0,\ \ell = 1, \dots, N$, and problems (11) and (12) are solvable with $x^{0,\ell}$ as the corresponding solutions, then the pair $(x^0, y^0)$ is a feasible starting point for problem (7), and $F^0 = F(x^0)$ is a starting objective function record value.

Let us write down the Lagrange function for problem (7) as

$$V(x,y,\lambda) = \sum_{i=1}^{N} f^i(x^i) + \lambda^\top \Big(\widetilde{g}(x) + \widetilde{L}y\Big), \tag{13}$$

where $\lambda \in \mathbb{R}^{N\widetilde{m}}$ is the vector of Lagrange multipliers consisting of subvectors $\lambda^i = (\lambda^i_1, \lambda^i_2, \ldots, \lambda^i_{\widetilde{m}})^\top,\ i = 1, \ldots, N$. Each subvector $\lambda^i$ corresponds to constraint vector-function $g^i$ of the agent $i$. The corresponding necessary optimality conditions

$$\frac{\partial V}{\partial x} = \mathrm{vec}\Big(\nabla f^1(x^1), \ldots, \nabla f^N(x^N)\Big) + J\widetilde{g}(x)^\top \lambda = 0, \tag{14a}$$

$$\frac{\partial V}{\partial y} = \widetilde{L}\lambda = 0, \tag{14b}$$

$$\frac{\partial V}{\partial \lambda} = \widetilde{g}(x) + \widetilde{L}y = 0, \tag{14c}$$

where $J\widetilde{g}(x) = \mathrm{diag}\big(Jg^1(x^1), Jg^2(x^2), \ldots, Jg^N(x^n)\big)$ is the Jacobian of $\widetilde{g}(x)$. If we again fix $y = y^0$ and solve the corresponding problem (7) for $y = y^0$, obtaining the corresponding primal solution $x^0$ and dual solution $\lambda^0$, then conditions (14a) and (14c) will be satisfied with $x = x^0$ and $\lambda = \lambda^0$. Condition (14b) can be violated $\frac{\partial V}{\partial y} = \widetilde{L}\lambda^0 \neq 0$, since we do not perform optimization in $y$. Hence, we correct $y^0$ by the gradient descent step in $y$

$$y^1 = y^0 - \rho_0 \widetilde{L}\lambda^0.$$

In general, we obtain the following recalculation formula for $y^k$:

$$y^{k+1} = y^k - \rho_k \widetilde{L}\lambda^k. \tag{15}$$

From (14b), we have the following element-wise representation at step $k$ due to the structure of the Laplacian matrix $L$ and the structure of vector $\lambda$ :

$$\frac{\partial V(x^k, y^k, \lambda^k)}{\partial y^i_j} = \sum_{s \in \mathrm{Adj}(i)} \Big(\lambda^{k,i}_j - \lambda^{k,s}_j\Big),\ j = 1, \ldots, \widetilde{m}. \tag{16}$$

Therefore, the calculation of $\frac{\partial V}{\partial y^i_j}$ satisfies the distributed form, since $\lambda^i_j$ is a local dual variable and all $\lambda^k_j$ are dual variables from adjacent agents.

The main computational scheme for solving problem (7) is presented in Algorithm 1. We assume here that problem (7) is solvable for $y = 0$.

Since we do not make any convexity assumptions, Algorithm 1 is suggested for finding a strict local saddle point in the Lagrange function in the sense of [35]. In [36], it is pointed out that values $\rho$ at Step 4 must be chosen small enough in order to achieve convergence to a strict local saddle point. One of the recommended choices for $\rho_k$ is the following: $\rho_0 = 1,\ \rho_k = \frac{1}{\sqrt{k}}$ for $k > 1$.

**Example 1.** *Consider the problem with* $N = 4,\ n_i = 1,\ i = 1, \ldots, 4,\ \widetilde{m} = 1$ *and*

$$f^1(x^1) = (x^1)^4 + 3(x^1)^3 - (x^1)^2 - 3x^1,\ f^2(x^2) = -\sin(x^2) + 0.1(x^2)^2,$$

$$f^3(x^3) = (x^3)^2 - 4,\ f^4(x^4) = x^4,$$

$$g^1(x^1) = (x^1)^3 - 8, g^2(x^2) = (x^2)^2 - 7x^2 + 10,\ g^3(x^3) = x^3 - 1,\ g^4(x^4) = (x^4)^2 - 9.$$

*The network is described by the Laplacian matrix*

$$L = \begin{pmatrix} 2 & -1 & 0 & -1 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{pmatrix}.$$

*The starting solution of the corresponding problem* (7) *with* $y = y^0 = 0$ *is the following* $x^0 = (-2, 2, 1, -3)^\top$, $\lambda^0 = (0.417, 0.272, 2.000, 0.167)^\top$, $F(x^0) = -12.509$. *The tolerance* $\varepsilon = 0.1$. *The* $\varepsilon$*-solution was obtained after 294 iterations of Algorithm 1:* $x^{294} = (-2.131, -1.954, 0.098, -2.877)^\top$, $F(x^{294}) = -13.972$, $\rho_k = \frac{1}{\sqrt{k}}$. *The optimal solution* $x^* = (-2.182, 1.831, -0.093, -2.678)^\top$, $F(x^*) = -14.013$. *Algorithm 1 generated a sequence of feasible points with decreasing objective function values.*

**Algorithm 1** Gradient descent method (for agent $\ell$)

```
Input: $f^\ell$,$g^\ell$, $\varepsilon > 0$.
Output: $x^{\ell,*}$
Algorithm steps:
Step 1. Set $y^{0,\ell} = 0$ and $k = 0$;
Step 2. Obtain $y^{k,i}$, $i \in \mathrm{Adj}(\ell)$ from neighboring agents;
Step 3. Solve problem (11) and (12) for $\widetilde{y}^\ell = y^{k,\ell}$, $\widetilde{y}^i = y^{k,i}$, $i \in \mathrm{Adj}(\ell)$. Let $x^{k,\ell}$, $\lambda^{k,\ell}$ be the corresponding primal and dual solutions;
Step 4. Obtain $\lambda^{k,i}, i \in \mathrm{Adj}(\ell)$ from neighboring agents;
Step 5. If
    $\left| \frac{\partial V(x^k, y^k, \lambda^k)}{\partial y_j^\ell} \right| = \left| \sum_{s \in \mathrm{Adj}(\ell)} \left( \lambda_j^{k,\ell} - \lambda_j^{k,s} \right) \right| < \varepsilon \quad \forall j = 1, \ldots, \widetilde{m},$
    then go to Step 8;
Step 6. Calculate $y^{k+1,\ell}$:
    $y^{k+1,\ell} = y^{k,\ell} - \rho_k \frac{\partial V(x^k, y^k, \lambda^k)}{\partial y_j^\ell};$
Step 7. Set $k = k + 1$ and go to Step 2;
Step 8. Stop: $x^{k,\ell}$ is an $\varepsilon$-stationary point of problem (2).
```

## 5. Problem with Equality and Inequality Constraints

Similarly to the equality constraints, let us introduce vector-function $\widehat{h} : \mathbb{R}^n \to N\widehat{m}$:

$$\widehat{h} = \begin{pmatrix} h^1(x^1) \\ \vdots \\ h^N(x^N) \end{pmatrix} \tag{17}$$

and expansion of the Laplacian matrix

$$\widehat{L} = L \otimes I^{\widehat{m}}. \tag{18}$$

Then, the new optimization problem has the form

$$\min_{x \in \mathbb{R}^n, y \in \mathbb{R}^{N\widetilde{m}}, z \in \mathbb{R}^{N\widehat{m}}} \sum_{i=1}^{N} f^i(x^i), \tag{19a}$$

$$\widetilde{g}(x) + \widetilde{L}y = 0, \tag{19b}$$

$$\widehat{h}(x) + \widehat{L}z \leq 0. \tag{19c}$$

Firstly, let us prove the following lemma.

**Lemma 2.** *The following statements are correct:*

1. *Problem* (19) *is feasible if problem* (1) *is feasible.*
2. *Triplet* $(x^*, y^*, z^*)$ *is the solution to problem* (19) *if and only if* $x^*$ *is the solution to* (1).

**Proof.** In Lemma 1, it was shown that linear constraints in both problems have the same solution in $x$. Let us now consider inequality constraints in both problems. As before, we consider the constraint $j$ from (1c)

$$\sum_{i=1}^{N} h_j^i(x^i) \leq 0 \tag{20}$$

and the set of corresponding constraints $c(j)$ from (19c)

$$\operatorname{vec}\Big(h_j^1(x^1), \dots, h_j^N(x^N)\Big) + Lz \leq 0. \tag{21}$$

The sum of the rows of $L$ is zero. Thus, the sum of the inequalities (21) yields (20). Let us now show that if for some $x$ equation (21) is correct, then there always exists $x$ such that (20) holds. Vector $\operatorname{vec}\Big(h_j^1(x^1), \dots, h_j^N(x^N)\Big)$ can always be decomposed using some orthogonal basis $\mathbf{1}_N, q^2, \dots, q^N$:

$$\operatorname{vec}\Big(h_j^1(x^1), \dots, h_j^N(x^N)\Big) = \alpha\mathbf{1}_N + \sum_{i=2}^{N} \beta_i q^i = \alpha\mathbf{1}_N + q. \tag{22}$$

Vector $q$ is orthogonal to $\mathbf{1}_N$ and, consequently, is orthogonal to $\ker L$. Thus, there always exists $z$ such that $Lz = -q$. Substitution of such $z$ into left-hand side of (21) gives

$$\operatorname{vec}\Big(h_j^1(x^1), \dots, h_j^N(x^N)\Big) + Lz = \alpha\mathbf{1}_N + q + Lz = \alpha\mathbf{1}. \tag{23}$$

Additionally,

$$\sum_{i=1}^{N} h_j^i(x^i) = \operatorname{vec}\Big(h_j^1(x^1), \dots, h_j^N(x^N)\Big)^\top \mathbf{1} = (\alpha\mathbf{1}_N + q)\mathbf{1} = \alpha. \tag{24}$$

Thus, from (20) $\alpha \leq 0$ and (21), since this statement holds for all $j \in \{1, \dots, \hat{m}\}$, the lemma is proven. □

## 6. Newton's Method

Here we adapt a Newton-type approach to distributed optimization [37]. In order to carry this out, we have to derive algorithms for the initial problem (1) and the distributed problem (19) in parallel. Due to the similar structure of these problems, all variables and functions of the initial problem will be denoted with an upper index $c$, which stands for centralized.

Let us introduce Lagrange functions for problem (19):

$$V(x, y, \lambda, \mu) = f(x) + \lambda^\top(\widetilde{g}(x) + \widetilde{L}y) + \mu^\top(\widehat{h}(x) + \widehat{L}z). \tag{25}$$

The corresponding Karush–Kuhn–Tuckker conditions have the form

$$\frac{\partial V}{\partial x} = \nabla f(x) + (J\widetilde{g}(x))^\top \lambda + (J\widehat{h}(x))^\top \mu = 0, \tag{26a}$$

$$\frac{\partial V}{\partial y} = \widetilde{L}\lambda = 0 \tag{26b}$$

$$\frac{\partial V}{\partial \mu} = \widehat{L}z = 0, \tag{26c}$$

$$\widetilde{g}(x) + \widetilde{L}y = 0, \tag{26d}$$

$$(\widehat{h}(x) + \widehat{L}z)_i \mu_i = 0,\ \mu \geq 0. \tag{26e}$$

In order to replace the complimentary slackness conditions with equations suitable for the Newton method, the complementarity function $\psi : \mathbb{R}^2 \to \mathbb{R}$ is introduced. It has the following property: $\psi(x, y) = 0$ if and only if $x \geq 0, y \geq 0$ and $xy = 0$. It can be chosen in multiple ways. Here, we use the following form:

$$\psi(x, y) = \begin{cases} y, & \text{if } x > 0 \text{ or } y \geq 0, \\ -x, & \text{otherwise.} \end{cases} \tag{27}$$

Then, the KKT conditions (26) can be replaced with

$$\phi(x, y, z, \lambda, \mu) = 0, \tag{28}$$

where

$$\phi(x, y, z, \lambda, \mu) = \begin{pmatrix} \frac{\partial V}{\partial x} \\ \widetilde{L}\lambda \\ \widehat{L}\mu \\ \widetilde{g}(x) + \widetilde{L}y \\ \psi\Big(\mu, \widehat{h}(x) + \widehat{L}z\Big) \end{pmatrix}. \tag{29}$$

Next, we introduce diagonal matrices $\mathcal{A}(x, \mu)$ with elements

$$\mathcal{A}_{ii}(x, \mu) = \begin{cases} 1, & \psi_i(\mu, \widehat{h}(x) + \widehat{L}z) = \widehat{h}(x) + \widehat{L}z, \\ 0, & \text{otherwise;} \end{cases}$$

and $\mathcal{B}(x, \mu)$ with elements

$$\mathcal{B}_{ii}(x, \mu) = \begin{cases} 1, & \psi_i(\mu, \widehat{h}(x)) = -\mu, \\ 0, & \text{otherwise.} \end{cases}$$

Then,

$$\Phi(x, y, z, \lambda, \mu) = \begin{pmatrix} \frac{\partial^2 V}{\partial x^2} & 0 & 0 & (J\widetilde{g}(x))^\top & (J\widehat{h}(x))^\top \\ 0 & 0 & 0 & \widetilde{L} & 0 \\ 0 & 0 & 0 & 0 & \widehat{L} \\ J\widetilde{g}(x) & \widetilde{L} & 0 & 0 & 0 \\ \mathcal{A}J\widehat{h}(x) & 0 & \mathcal{B}\widehat{L} & 0 & \mathcal{B} \end{pmatrix} \tag{30}$$

and the values $x^{k+1}, y^{k+1}, z^{k+1}, \lambda^{k+1}, \mu^{k+1}$, corresponding to the $k$-th Newton iteration step, are calculated as the solution to the following system:

$$\Phi^k\Big(\text{vec}(x, y, z, \lambda, \mu) - \text{vec}(x^{k+1}, y^{k+1}, z^{k+1}, \lambda^{k+1}, \mu^{k+1})\Big) = \phi^k, \tag{31}$$

where

$$\Phi^k = \Phi(x^k, y^k, z^k, \lambda^k, \mu^k),\ \phi^k = \phi(x^k, y^k, z^k, \lambda^k, \mu^k). \tag{32}$$

Let us introduce the same Newton step equations for initial problem (1). For the Lagrange function, we have

$$V^c(x^c, \lambda^c, \mu^c) = f(x^c) + \lambda^{c\top} g(x^c) + \mu^{c\top} h(x^c) \tag{33}$$

and the corresponding parameters have the form

$$\phi^c(x^c, \lambda^c, \mu^c) = \begin{pmatrix} \frac{\partial V^c}{\partial x^c} \\ G(x^c) \\ \psi(\mu^c, H(x^c)) \end{pmatrix}, \tag{34a}$$

$$\Phi^c(x^c, \lambda^c, \mu^c) = \begin{pmatrix} \frac{\partial^2 V^c}{\partial x^{c2}} & (JG(x))^\top & (JH(x))^\top \\ JG(x^c) & 0 & 0 \\ \mathcal{A}(x^c, \mu^c) JH(x^c) & 0 & \mathcal{B}(x^c, \mu^c) \end{pmatrix}. \tag{34b}$$

Finally, with

$$\Phi^{c,k} = \Phi(x^{c,k}, \lambda^{c,k}, \mu^{c,k}),\ \phi^{c,k} = \phi(x^{c,k}, \lambda^{c,k}, \mu^{c,k}) \tag{35}$$

for a given Newton step, we have

$$\Phi^{c,k}\Big(\mathrm{vec}(x, y, z, \lambda, \mu) - \mathrm{vec}(x^{c,k+1}, y^{c,k+1}, z^{c,k+1}, \lambda^{c,k+1}, \mu^{c,k+1})\Big) = \phi^{c,k}. \tag{36}$$

Let us now prove the following result.

**Theorem 1.** *If the following conditions hold*

1. $x^0 = x^{c,0}$;
2. $\lambda^0_{c(i)} = \mathbf{1}\lambda^{c,0}_i / N$ *for* $i \in \{1, \dots, \widetilde{m}\}$;
3. $\mu^0_{c(i)} = \mathbf{1}\mu^{c,0}_i / N$ *for* $i \in \{1, \dots, \widehat{m}\}$;
4. *For all* $i \in \{1, \dots, \widehat{m}\}$ $z^0_{c(i)}$ *is solution of the following optimization problem*

$$\min_{t \in \mathbb{R}, z_{c(i)} \in \mathbb{R}^N} \frac{1}{2} t^\top t, \tag{37a}$$

$$\widehat{g}_{c(i)}(x^{0,i}) + L z_{c(i)} + t = 0, \tag{37b}$$

*then the convergence of* (31) *coincides with the convergence of* (36).

**Proof.** If, for the iteration $k$, conditions 1–4 hold, then, from (31), we arrive at a system of linear equations. Using $\Delta$ to denote the difference between variables at the $k$-th iteration, Equation (31) can be rewritten as

$$\frac{\partial^2 V}{\partial x^2} \Delta x + (JG(x^k))^\top \Delta\lambda + (JH(x^k))^\top \Delta\mu = \frac{\partial V}{\partial x}, \tag{38a}$$

$$\widetilde{L}\Delta\lambda = \widetilde{L}\lambda^k, \tag{38b}$$

$$\widehat{L}\Delta\mu = \widehat{L}\mu^k, \tag{38c}$$

$$J\widetilde{g}(x^k)\Delta x + \widetilde{L}\Delta y = \widetilde{g}(x^k) + \widetilde{L}y^k, \tag{38d}$$

$$\mathcal{A}(x^k, \mu^k) J\widehat{h}(x^k)\Delta x + \mathcal{A}(x^k, \mu^k)\widehat{L}\Delta z + \mathcal{B}(x^k, \mu^k)\mu^k = \psi(\mu^k, \widehat{h}(x^k) + \widehat{L}z^k). \tag{38e}$$

This set of equations describes a stationary point in the optimization problem

$$\min_{\Delta x\in\mathbb{R}^n,\Delta y\in\mathbb{R}^{\widehat{m}N},\Delta z\in\mathbb{R}^{\widehat{m}N}} \frac{1}{2}\Delta x^\top \frac{\partial^2 V}{\partial x^2}\Delta x + \Delta x^\top \frac{\partial V}{\partial x}, \tag{39a}$$

$$J\widetilde{g}(x^k)\Delta x + \widetilde{L}\Delta y = \widetilde{g}(x^k) + \widetilde{L}y^k, \tag{39b}$$

$$\left(J\widehat{h}(x^k)\Delta x + \widehat{L}\Delta z\right)_{\mathcal{J}} = \left(\widehat{h}(x^k) + \widehat{L}z^k\right)_{\mathcal{J}}, \tag{39c}$$

where $\mathcal{J} = \{i \in \{1,\dots,\widehat{m}N\} \mid \mathcal{A}_{ii}(x^k,\mu^k) = 1\}$.

Likewise, for the centralized Equation (36), we can obtain a similar optimization problem:

$$\min_{\Delta x^c\in\mathbb{R}^n} \frac{1}{2}\Delta x^{c\top} \frac{\partial^2 V^c}{\partial x^{c2}}\Delta x^c + \Delta x^{c\top} \frac{\partial V^c}{\partial x^c}, \tag{40a}$$

$$J\widetilde{g}(x^{c,k})\Delta x^c = \tilde{g}(x^{c,k}), \tag{40b}$$

$$\left(J\hat{h}(x^{c,k})\Delta x^c\right)_I = \left(\hat{h}(x^{c,k})\right)_I, \tag{40c}$$

where $I = \{i \in \{1,\dots,\widehat{m}\} \mid \mathcal{A}a_{ii}(x^{c,k},\mu^{c,k}) = 1\}$. Let us now show that (39) is an expansion of problem (40). Consider objective functions in both problems, which have first and second derivatives of the corresponding Lagrange functions. For the first derivative, we have

$$\frac{\partial V}{\partial x} = \nabla f(x^k) + (J\widetilde{g}(x^k))^\top\lambda + (J\widehat{h}(x^k))^\top\mu. \tag{41}$$

Note that due to condition 2,

$$\left((J\widetilde{g}(x^k))^\top\lambda^k\right)_i = \sum_{k=1}^{\widetilde{m}}\sum_{j\in c(i)}\frac{\partial g^k}{\partial x^i}\lambda^j_k = \sum_{k=1}^{\widetilde{m}}\sum_{j\in c(i)}\frac{\partial g^k}{\partial x^i}\frac{\lambda^c_k}{\widetilde{m}} = \lambda^c_k\sum_{k=1}^{\widetilde{m}}\sum_{j\in c(i)}\frac{\partial g^k}{\partial x^i} = \left((Jg(x^k))^\top\lambda^{k,c}\right). \tag{42}$$

The same equality holds for $((J\widehat{h}(x^k))^\top\mu^k$. Thus,

$$\frac{\partial V}{\partial x} = \frac{\partial V^c}{\partial x^c} \tag{43}$$

and, consequently,

$$\frac{\partial^2 V}{\partial x^2} = \frac{\partial^2 V^c}{\partial x^{c2}}. \tag{44}$$

As a result, the objective functions in problems (40) and (39) are equivalent. Let us now consider the relation between sets $J$ and $I$. Firstly, we focus on the optimization problem (37). It can be shown that in its optimum, $z_{c(i)}$ and $t$ are chosen so that $t_i = t_j$, since it is the only case, where $t \in \ker L$. Thus, for each $i$, the corresponding inequality constraints from (40) and (39),

$$\sum_{j=1}^{N} g^j_i(x^{c,j}) \le 0 \tag{45}$$

and

$$\widetilde{g}_{c(i)}(x) + Lz_{c(i)} \le 0 \tag{46}$$

are all active or inactive simultaneously. Thus,

$$\mathcal{J} = \bigcup_{i\in I} c(i). \tag{47}$$

As a result, problem (39) is an expansion of problem (40), and according to Lemma 1, has the same solution in $x$. Moreover, it means that for $k+1$, item 1 of the lemma is satisfied. Let us now demonstrate that the values $x^{k+1}, y^{k+1}, z^{k+1}, \lambda^{k+1}, \mu^{k+1}$ satisfy items 2–4 of the Lemma. From (38b), for each $i \in \{1, \dots, \widetilde{m}\}$, all components of $\Delta\lambda^{k+1}_{c(i)}$ are equal to each other, which gives item 2. The same approach applies for all $\mu^{k+1}_{c(i)}$, $i \in I$, and for all other $i$ $\mu^{k+1}_{c} i = 0$, which means that item 3 holds. Finally, for $z^{k+1}_{c(i)}$, $i \in I$, the corresponding constraint is active, and problem (37) is solved with $t = 0$. For all other $i$, we have $z^{k+1}_{c(i)}$, and therefore, item 4 holds. □

**Corollary 1.** *The convergence speed of Newton's method applied to problem* (19) *is equal to the convergence speed of Newton's method applied to problem* (1)*.*

**Corollary 2.** *Assume that*

- *functions $f$, $g$ and $h$ are twice differentiable in some neighborhood of the solution $x^*$ of problem* (1)*, and their second derivatives are Lipshitz-continuous in the neighborhood of the $x^*$.*
- *the constraints' gradients are linear independent in the optimum (linear independence constraint qualification);*
- *solution $x^*$ has unique corresponding dual variables $\lambda^{c,*}$ and $\mu^{c,*}$ in problem* (1)*;*
- *for $x^*$, $\lambda^{c,*}$ and $\mu^{c,*}$ we have*

$$u^\top \frac{\partial^2 L^c}{\partial x^2}(x^*, \lambda^{c,*}, \mu^{c,*}) > 0 \ \forall u \in K_+(x^*) \setminus \{0\}, \tag{48}$$

*where*

$$K_+(x^*) = \left\{u \in \ker(Jh(x^*)) \mid (Jg(x^*))_i u = 0 \ \forall i : g_i(x^*) = 0 \text{ and } \mu_i^{c,*} > 0\right\}. \tag{49}$$

*Then, in problem* (19)*, for any starting point $(x, \lambda, \mu)$ sufficiently close to $(x^*, \lambda^*, \mu^*)$, where $\lambda^0_{c(i)} = \mathbf{1}\lambda_i^{c,0}/N$ for $i \in \{1, \dots, \widetilde{m}\}$ and $\mu^0_{c(i)} = \mathbf{1}\mu_i^{c,0}/N$ for $i \in \{1, \dots, \widehat{m}\}$, Algorithm 2 converges to the solution with quadratic rate.*

This corollary result is based on the estimation of Newton's method convergence rate for problem (1) given in [37].

Finally, Algorithm 2 requires the exchange of information only in steps 6 and 8. However, during this step, the gradient descent method is used with its distributed implementation shown in the previous section. Thus, operations in Algorithm 2 are performed in the distributed form.

**Example 2.** *Consider the initial problem with the following components: $N = 4$, $\widetilde{m} = 1$, $\widehat{m} = 0$, $n_1 = n_2 = n_3 = n_4 = 1$, $f^1(x^1) = (x_1^1 - 5)^2$, $f^2(x^2) = (x_1^2 - 4)^2$, $f^3(x^3) = (x_1^3 - 3)^2$, $f^4(x^4) = (x_1^4 - 2)^2$, $g^1(x^1) = (x_1^1)^2 - 3$, $g^2(x^2) = (x_1^2)^2 - 3$, $g^3(x^3) = (x_1^3)^2 - 3$, $g^4(x^4) = (x_1^4)^2 - 3$. The network is described by the Laplacian matrix (as in example 1):*

$$L = \begin{pmatrix} 2 & -1 & 0 & -1 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{pmatrix}.$$

*Starting points $x_1^{1,0} = 2$, $x_1^{2,0} = 2$, $x_1^{3,0} = 2$, $x^{4,0} = 2$, $y_1^{1,0} = 0$, $y_1^{2,0} = 0$, $y_1^{3,0} = 0$, $y_1^{4,0} = 0$, $\lambda^{1,0} = 0.4$, $\lambda_2^{1,0} = 0.4$, $\lambda^{3,0} = 0.4$, $\lambda_1^{4,0} = 0.4$. The tolerance $\varepsilon = 0.01$. Then, in four iterations, Algorithm 2 finds an $\varepsilon$-optimal solution with components $x_1^{1,4} = 2.357$, $x_1^{2,4} = 1.886$, $x_1^{3,4} = 1.414$, $x_1^{4,4} = 0.943$, $y_1^{1,4} = -1.083$, $y_1^{2,4} = -0.472$, $y_1^{3,4} = 0.694$, $y_{=}^{4,4}0.861$, $\lambda_1^{1,4} = \lambda_1^{2,4} = \lambda_1^{3,4} = \lambda_1^{4,4} = 1.121$. The objective function value $F(x^4) = 15.088$.*

```
Algorithm 2 Newton's method
Input: $f$,$g$,$L$, $\varepsilon > 0$.
Output: $x^*$
Algorithm steps:
Step 1. Set $y^0 = 0$;
Step 2. For each $i \in \{1, \dots, N\}$ set $z_{c(i)}$ as a solution of (37) using gradient descent;
Step 3. Set $\lambda^0 = 0$ and $\mu^0 = 0$;
Step 4. Set $k = 0$;
Step 5. Solve optimization problem (39) using the gradient descent method;
Step 6. Assign
    $\begin{pmatrix} x^{k+1} \\ y^{k+1} \\ z^{k+1} \\ \lambda^{k+1} \\ \mu^{k+1} \end{pmatrix} = \begin{pmatrix} \Delta x^k \\ \Delta y^k \\ \Delta z^k \\ \Delta\lambda^k \\ \Delta\mu^k \end{pmatrix} + \begin{pmatrix} x^k \\ y^k \\ z^k \\ \lambda^k \\ \mu^k \end{pmatrix}$    (50)
Step 7. For all inactive constraints $i \in \{1, \dots, \widehat{m}\}$, solve problem (37) using gradient descent and assign its solution to $z^{k+1}_{c(i)}$;
Step 8. If $\|x^k - x^{k+1}\| > \varepsilon$, then set $k = k + 1$ and go back to step 5;
Step 9. Stop: $x^{k+1}$ is an $\varepsilon$-stationary point.
```

## 7. Application of the Arrow–Hurwicz Algorithm to Problems with Inequality Constraints

We consider here problem (19) without equality constraints

$$\min_{x \in \mathbb{R}^n, z \in \mathbb{R}^{N\widehat{m}}} \sum_{i=1}^{N} f^i(x^i), \tag{51a}$$

$$\widehat{h}(x) + \widehat{L}z \leq 0. \tag{51b}$$

The Lagrange function is given by

$$V(x, z, \mu) = f(x) + \mu^\top \left(\widehat{h}(x) + \widehat{L}x\right). \tag{52}$$

Assume that starting vectors $(x^0, z^0, \mu^0)$ are given. The Arrow–Hurwicz algorithm can be described by the following relations ([38]):

$$x^{k+1} = x^k - \rho \frac{\partial V(x^k, z^k, \mu^k)}{\partial x}, \tag{53}$$

$$z^{k+1} = x^k - \rho \frac{\partial V(x^k, z^k, \mu^k)}{\partial z}, \tag{54}$$

$$\mu^{k+1} = \max\{0, \mu^k + \rho h(x^k)\}. \tag{55}$$

As was shown above, the computational scheme (53)–(55) with $V$ defined in (52) has the distributed form. The algorithm stops when $\|x^k - x^{k+1}\| \leq \varepsilon$, where $\varepsilon > 0$ is the tolerance. The following strategy of choosing the step size $\rho$ was used. We set the initial value $\rho = 1$. If, during the iterations, the deviation $h(x^k)$ is becoming large enough, for example, greater than the practically chosen value $\overline{h}$, then $\rho$ is reset: $\rho = \frac{\rho}{2}$, and the algorithms restart from the initial starting set $(x^0, z^0, \mu^0)$. The explanation of such a restarting is the following. If the deviation $\hat{h}(x^k)$ is big enough, then the point $x^k$ is too far from the feasible domain in contrast to starting point $x^0$, which is assumed to be chosen close enough to the feasible domain. The first and main reason of using this algorithm is due to the fact that it provides a minimization procedure in the non-convex case. The second reason consists of the following. In some neighborhood of the point of minimum, the Lagrange function is usually locally convex, and if the algorithm managed to get into this neighborhood, then it determines this point of minimum.

**Example 3.** *Problem* (51) *has the following components:* $N = 4$, $\hat{m} = 2$, $n_1 = 2$, $n_2 = 1$, $n_3 = 3$, $n_4 = 2$, $f_1(x_1^1, x_2^1) = (x_1^1)^2(x_2^1)^2$, $f^2(x_1^2) = 2\sin((x_1^2 - 3)x_1^2)$, $f^3(x_1^3, x_2^3, x_3^3) = (x_1^3)^2 + (x_2^3)^2 + (x_3^3)^2$, $f^4(x_1^4, x_2^4) = (x_1^4 - x_2^4 - 1)^2$, $h_1^1(x^1, x_2^1) = x_1^1 + x_2^1 - 3$, $h_2^1(x^1, x_2^1) = (x_1^1)^2 + (x_2^1)^2 - 5$, $h_1^2(x_1^2) = (x_1^2)^2 - 4$, $h_1^3(x_1^3, x_2^3, x_3^3) = x_1^3 + x_2^3 + x_3^3 - 3$, $h_2^3(x_1^3, x_2^3, x_3^3) = x_3^3 - 2$, $h_1^4(x_1^4, x_2^4) = x^41 - x_2^4$, $h_2^4(x_1^4, x_2^4) = (x_1^4)^2 + (x_2^4)^2 - 1$.
*The interpretation of the agent network is shown in Figure 2. The Laplacian matrix is*

$$L = \begin{pmatrix} 2 & -1 & 0 & -1 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{pmatrix}.$$

*All components of the starting vectors* $x^0, z^0, \mu^0$ *were equal to 2. Tolerance* $\varepsilon = 0.001$. *After 69 iterations, the algorithm determined point* $x^{69} = (-0.162, -0.162, 0.676, 0, 0, 0, -0.119, 1.119)$, *which happened to be optimal,* $y^{69} = (3.108, 3.296, 2.525, 2.466, 1.940, 1.446, 0.427, 0.791)$, $F(x^{69}) = -1.999$.

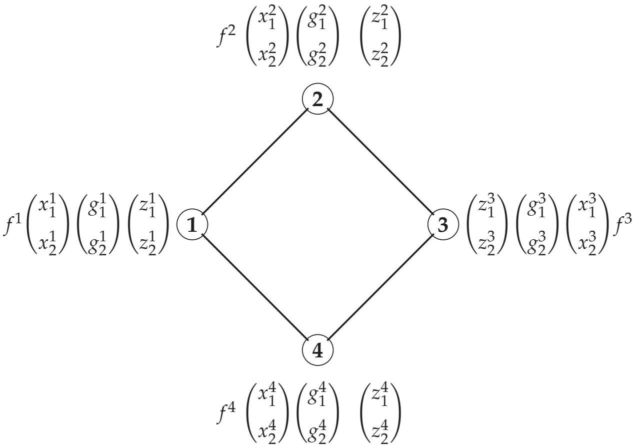

$f^2$ $\begin{pmatrix} x_1^2 \\ x_2^2 \end{pmatrix}\begin{pmatrix} g_1^2 \\ g_2^2 \end{pmatrix}$ $\begin{pmatrix} z_1^2 \\ z_2^2 \end{pmatrix}$
2
$f^1$ $\begin{pmatrix} x_1^1 \\ x_2^1 \end{pmatrix}\begin{pmatrix} g_1^1 \\ g_2^1 \end{pmatrix}\begin{pmatrix} z_1^1 \\ z_2^1 \end{pmatrix}$
1
3
$\begin{pmatrix} z_1^3 \\ z_2^3 \end{pmatrix}\begin{pmatrix} g_1^3 \\ g_2^3 \end{pmatrix}\begin{pmatrix} x_1^3 \\ x_2^3 \end{pmatrix}$ $f^3$
4
$f^4$ $\begin{pmatrix} x_1^4 \\ x_2^4 \end{pmatrix}\begin{pmatrix} g_1^4 \\ g_2^4 \end{pmatrix}$ $\begin{pmatrix} z_1^4 \\ z_2^4 \end{pmatrix}$

**Figure 2.** The agent network with information available at each node.

## 8. Conclusions

A novel approach for the decentralized solution of non-convex optimization problems was proposed. It is based on the reformulation of optimization problems to a specific form that allows the distributed implementation of modified gradient descent and Newton's methods. The main strength of the modified Newton's method is in having the same number of oracle callers as a standard Newton's method applied to the initial problem formulation. Thus, in the cases when oracle calls are more expensive than communication between agents, the transition from centralized to distributed paradigm does not significantly affect computational time. Moreover, if the convergence speed for Newton's method in application to centralized problems is quadratic, the same speed will remain for the modified decentralized algorithm.

Such properties of the proposed approach are extremely useful in solving optimization problems that arise when automating the design of decision support systems and digital twins based on a holistic approach that uses mathematical and machine learning models in conjunction with human expert knowledge [5], especially in the context of ubiquitous computing and digitalization [1].

**Author Contributions:** Conceptualization, O.O.K., O.V.K. and E.S.S.; methodology, O.O.K., O.V.K. and E.S.S.; validation, O.O.K., O.V.K., T.D.G. and E.S.S.; formal analysis O.V.K., T.D.G. and E.S.S.; investigation, E.S.S.; writing—original draft preparation, O.O.K. and O.V.K.; writing—review and editing, O.V.K., T.D.G. and E.S.S.; supervision, E.S.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** Data are contained within the article.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Ganchev, T.D. Chapter 8—Ubiquitous computing and biodiversity monitoring. In *Advances in Ubiquitous Computing*; Neustein, A., Ed.; Advances in Ubiquitous Sensing Applications for Healthcare; Academic Press: Cambridge, MA, USA, 2020; pp. 239–259. [CrossRef]
2. Ganchev, T.; Markova, V.; Valcheva-Georgieva, I.; Dobrev, I. Assessment of pollution with heavy metals and petroleum products in the sediments of Varna Lake. *Rev. Bulg. Geol. Soc.* **2022**, *83*, 3–9. [CrossRef]
3. Nandal, A.; Zhou, L.; Dhaka, A.; Ganchev, T.; Nait-Abdesselam, F. *Machine Learning in Medical Imaging and Computer Vision*; IET: London, UK, 2024.
4. Markova, V.; Ganchev, T.; Filkova, S.; Markov, M. MMD-MSD: A Multimodal Multisensory Dataset in Support of Research and Technology Development for Musculoskeletal Disorders. *Algorithms* **2024**, *17*, 187 . [CrossRef]
5. Semenkin, E. Computational Intelligence Algorithms based Comprehensive Human Expert and Data driven Model Mining for the Control, Optimization and Design of Complicated Systems. *Int. J. Inf. Technol. Secur.* **2019**, *11*, 63–66.
6. Semenkin, E.; Semenkina, M. Artificial neural networks design with self-configuring genetic programming algorithm. In Proceedings of the Bioinspired Optimization Methods and Their Applications, Maribor, Slovenia, 17–18 November 2012; pp. 291–300.
7. Akhmedova, S.; Semenkina, M.; Stanovov, V.; Semenkin, E. Semi-supervised Data Mining Tool Design with Self-tuning Optimization Techniques. In *Proceedings of the Informatics in Control, Automation and Robotics: 14th International Conference, ICINCO 2017 Madrid, Spain, 26–28 July 2017*; Revised Selected Papers; Springer: Berlin/Heidelberg, Germany , 2020; pp. 87–105.
8. Sherstnev, P.; Semenkin, E. Application of evolutionary algorithms for the design of interpretable mschine learning models in classification problems. *Control Syst. Inf. Technol.* **2022**, *22*, 17–20. [CrossRef]
9. Vakhnin, A.; Sopov, E.; Semenkin, E. On Improving Adaptive Problem Decomposition Using Differential Evolution for Large-Scale Optimization Problems. *Mathematics* **2022**, *10*, 4297. [CrossRef]
10. Krutikov, V.; Gutova, S.; Tovbis, E.; Kazakovtsev, L.; Semenkin, E. Relaxation Subgradient Algorithms with Machine Learning Procedures. *Mathematics* **2022**, *10*, 3959. [CrossRef]
11. Nedic, A.; Ozdaglar, A. Distributed Subgradient Methods for Multi-Agent Optimization. *IEEE Trans. Autom. Control* **2009**, *54*, 48–61. [CrossRef]
12. Metelev, D.; Beznosikov, A.; Rogozin, A.; Gasnikov, A.; Proskurnikov, A. Decentralized optimization over slowly time-varying graphs: Algorithms and lower bounds. *Comput. Manag. Sci.* **2024**, *21, 8*. [CrossRef]
13. Nedić, A.; Olshevsky, A.; Shi, W. Achieving Geometric Convergence for Distributed Optimization Over Time-Varying Graphs. *Siam J. Optim.* **2017**, *27*, 2597–2633. [CrossRef]
14. Tang, Y.; Deng, Z.; Hong, Y. Optimal Output Consensus of High-Order Multiagent Systems With Embedded Technique. *IEEE Trans. Cybern.* **2019**, *49*, 1768–1779. [CrossRef] [PubMed]
15. Jadbabaie, A.; Lin, J.; Morse, A. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Trans. Autom. Control* **2003**, *48*, 988–1001. [CrossRef]
16. Luan, X.; Schutter, B.; Meng, L.; Corman, F. Decomposition and distributed optimization of real-time traffic management for large-scale railway networks. *Transp. Res. Part Methodol.* **2020**, *141*, 72–97. [CrossRef]
17. Luan, X.; Schutter, B.; van den Boom, T.; Corman, F.; Lodewijks, G. Distributed optimization for real-time railway traffic management. *IFAC-PapersOnLine* **2018**, *51*, 106–111. [CrossRef]
18. Khamisov, O.O. Direct disturbance based decentralized frequency control for power systems. In Proceedings of the 2017 IEEE 56th Annual Conference on Decision and Control (CDC), Melbourne, Australia, 12–15 December 2017; pp. 3271–3276. [CrossRef]
19. Khamisov, O.O.; Chernova, T.; Bialek, J.W. Comparison of two schemes for closed-loop decentralized frequency control and overload alleviation. In Proceedings of the 2019 IEEE Milan PowerTech, Milan, Italy, 23–27 June 2019; pp. 1–6. [CrossRef]
20. Motta, V.N.; Anjos, M.; Gendreau, M. Optimal allocation of demand response considering transmission system congestion. *Comput. Manag. Sci.* **2020**, *20*, 2023.
21. Rabbat, M.; Nowak, R. Distributed optimization in sensor networks. In Proceedings of the Third International Symposium on Information Processing in Sensor Networks, IPSN 2004, Berkeley, CA, USA, 26–27 April 2004 ; pp. 20–27. [CrossRef]

22. Sadiev, A.; Borodich, E.; Beznosikov, A.; Dvinskikh, D.; Chezhegov, S.; Tappenden, R.; Takac, M.; Gasnikov, A. Decentralized personalized federated learning: Lower bounds and optimal algorithm for all personalization modes. *Euro J. Comput. Optim.* **2022**, *10*, 100041. [CrossRef]
23. Forero, P.A.; Cano, A.; Giannakis, G.B. Consensus-Based Distributed Support Vector Machines. *J. Mach. Learn. Res.* **2010**, *11*, 1663–1707.
24. Gorbunov, E.; Rogozin, A.; Beznosikov, A.; Dvinskikh, D.; Gasnikov, A., Recent Theoretical Advances in Decentralized Distributed Convex Optimization. In *High-Dimensional Optimization and Probability: With a View Towards Data Science*; Nikeghbali, A., Pardalos, P.M., Raigorodskii, A.M., Rassias, M.T., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 253–325. [CrossRef]
25. Kovalev, D.; Salim, A.; Richtarik, P. Optimal and Practical Algorithms for Smooth and Strongly Convex Decentralized Optimization. In Proceedings of the Advances in Neural Information Processing Systems, Virtual, 6–12 December 2020; Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H., Eds.; Curran Associates, Inc .: New York, NY, USA, 2020; Volume 33, pp. 18342–18352.
26. Kovalev, D.; Shulgin, E.; Richtarik, P.; Rogozin, A.V.; Gasnikov, A. ADOM: Accelerated Decentralized Optimization Method for Time-Varying Networks. In Proceedings of the 38th International Conference on Machine Learning, Virtual Event, 18–24 July 2021; Meila, M., Zhang, T., Eds.; PMLR: Westminster, UK, 2021; Volume 139, Proceedings of Machine Learning Research; pp. 5784–5793.
27. Wang, Z.; Ong, C.J.; Hong, G.S. Distributed Model Predictive Control of linear discrete-time systems with coupled constraints. In Proceedings of the 2016 IEEE 55th Conference on Decision and Control (CDC), Las Vegas, NV, USA, 12–14 December 2016; pp. 5226–5231. [CrossRef]
28. Erseghe, T. Distributed Optimal Power Flow Using ADMM. *IEEE Trans. Power Syst.* **2014**, *29*, 2370–2380. [CrossRef]
29. Yarmoshik, D.; Rogozin, A.; Khamisov, O.O.; Dvurechensky, P.; Gasnikov, A. Decentralized Convex Optimization Under Affine Constraints for Power Systems Control. In Proceedings of the 21st International Conference Mathematical Optimization Theory and Operations Research, Petrozavodsk, Russia, 2–6 July 2022; Pardalos, P., Khachay, M., Mazalov, V., Eds.; Springer: Cham, Switzerland, 2022; pp. 62–75.
30. Khamisov, O.O. Distributed continuous-time optimization for convex problems with coupling linear inequality constraints. *Math. Optim. Theory Oper. Res.* **2024**, *21*, 1619–6988. [CrossRef]
31. Stomberg, G.; Engelmann, A.; Faulwasser, T. Decentralized non-convex optimization via bi-level SQP and ADMM. In Proceedings of the 2022 IEEE 61st Conference on Decision and Control (CDC), Cancun, Mexico, 6–9 December 2022; pp. 273–278. [CrossRef]
32. Hours, J.H.; Jones, C.N. A Parametric Nonconvex Decomposition Algorithm for Real-Time and Distributed NMPC. *IEEE Trans. Autom. Control* **2016**, *61*, 287–302. [CrossRef]
33. Zhao, C.; Topcu, U.; Li, N.; Low, S. Design and Stability of Load-Side Primary Frequency Control in Power Systems. *IEEE Trans. Autom. Control* **2014**, *59*, 1177–1189. [CrossRef]
34. Shores, T. *Applied Linear Algebra and Matrix Analysis*; Springer : New York, NY, USA, 2007.
35. Evtushenko, Y. *Numerical Optimization Technique*; Springer: New York, NY, USA, 1985.
36. Hadley, G. *Nonlinear and Dynamic Programming*; Addison-Wessley Publishing Company Inc.: Boston, MA, USA, 1964.
37. Izmailov, A.F.; Solodov, M.V. *Newton-Type Methods for Optimization and Variational Problems*; Springer: Cham, Switzerland, 2014.
38. Minoux, M. *Programmation Mathémateque: Théorie et Algorithmes*; Dunod: Paris, France, 1983.

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

MDPI AG
Grosspeteranlage 5
4052 Basel
Switzerland
Tel.: +41 61 683 77 34

*Mathematics* Editorial Office
E-mail: mathematics@mdpi.com
www.mdpi.com/journal/mathematics

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

MDPI

Academic Open
Access Publishing

mdpi.com

ISBN 978-3-7258-2270-6