Electronics and Information Technologies, 2026, 34, 165–180
http://publications.lnu.edu.ua/collections/index.php/electronics/index

UDC: 004.9:004.8:004.2]-047.27-048.34:519.853

SOFTWARE ENVIRONMENT FOR IDENTIFICATION OF NONLINEAR MATHEMATICAL MODELS USING OPTIMIZATION PROCEDURES

Bohdan Melnyk
Department of Information Systems in Management
Ivan Franko National University of Lviv,
18 Svobody Av., Lviv, 79008, Ukraine

Melnyk, B. (2026). Software Environment for Identification of Nonlinear Mathematical Models Using Optimization Procedures. *Electronics and Information Technologies*, 34, 165–180. https://doi.org/10.30970/eli.34.10

ABSTRACT

Background. For an adequate description of complex systems, it is necessary to use significantly nonlinear mathematical models. An effective approach to the identification of such models is the use of optimization procedures based on various optimization methods. For the comprehensive implementation of these procedures, it is necessary to create software that meets modern trends in the field of computer modeling.

Materials and Methods. To identify models, various optimization algorithms are used, which are developed within the framework of various optimization methods, in particular, within the framework of quasi-Newton methods or stochastic methods. The article considers the BFGS, L-BFGS, and L-BFGS-B algorithms, which belong to the first class of methods, and algorithms that implement the stochastic directed cone method, namely: the basic algorithm and its three modifications. A researcher engaged in mathematical modeling selects one of the software-implemented optimization algorithms that best corresponds to the formulated modeling problem. To do this, he must be able to evaluate the qualitative indicators of these algorithms, which are integrated in a single software environment.

Results and Discussion. The software environment for the identification of mathematical models using optimization procedures, in addition to the specific features associated with identification, must comply with modern programming paradigms and their use. In view of this, it is proposed to use the Python programming environment for its creation. The structure of the software subsystem that implements the selection of the type of mathematical model, its structural and parametric identification and evaluation, as well as integration with other subsystems of the created software environment, is considered. General requirements for ensuring its functionality are formulated. The prospects for the application of cloud IT and AI are considered.

Conclusion. The proposed software and architectural solutions allow creating a software environment that will significantly facilitate the process of computer-generated mathematical models of various complex systems. At the same time, it remains open for further modernization and expansion of functionality.

Keywords: mathematical model, identification, optimization algorithms, software, architecture.

INTRODUCTION

A large number of real objects and processes are considered complex abstract systems, which are described using nonlinear mathematical models. The process of

constructing a mathematical model is multi-stage and significantly depends on the nature of the processes being modeled, as well as on the applied modeling approaches [1, 2].

In the case when it is difficult to describe the structure of the internal connections of the system or it is not advisable from the researcher's point of view, a mathematical model of the “black box” type is used [3]. In the multidimensional case, such a model uses as variables input signals $[v_1, v_2, \dots, v_n]^T = \vec{v}$ (external disturbances that affect the behavior of the system), output signals $[y_1, y_2, \dots, y_m]^T = \vec{y}$, which reflect the external reaction of the system to external influences, state variables $[x_1, x_2, \dots, x_r]^T = \vec{x}$ (formal variables that in a certain way reflect the internal state of the system), as well as equation coefficients $[p_1, p_2, \dots, p_s]^T = \vec{p}$, which reflect the relationship between variables (model parameters).

If a dynamic system is considered, then the values of the variables change over time. In the case of fixing the values of these variables at specific points in time, a discrete model is used. The general form of a discrete model of the “black box” type with constant parameters is as follows

$$\vec{y}^{(k)} = \vec{f}\left(\vec{v}^{(k)}, \vec{x}^{(k)}, \vec{p}\right), \tag{1}$$

where $\vec{y}^{(k)}$, $\vec{v}^{(k)}$, $\vec{x}^{(k)}$ are, respectively, the vectors of output and input signals, as well as the vector of state variables, which are defined at the k-th time instant ($k = \overline{1, N}$, N is the number of time instants at which the values of the vector components are defined); $\vec{p}$ is the vector of model parameters; $\vec{f}(\bullet)$ is a certain vector function, the components of which describe the relationships between the variables.

In the process of building a mathematical model, two important stages can be distinguished among others: the stage of structural identification of the model and the stage of parametric identification of the model [4, 5]. At the stage of structural identification, the form of equations in $\vec{f}(\bullet)$ is selected. This choice significantly depends on the specific formulation of the modeling problem. In the general case, these equations are nonlinear. At the stage of parametric identification, the values of the model parameters are determined. Parametric identification procedures can be carried out either sequentially after the completion of all structural identification procedures, or in parallel with individual ones [5].

Currently, a number of methods are used for structural identification of mathematical models: methods of reduction and complication of the model structure (MSR and MSC), group method of data handling (GMDH), genetic algorithms (GA), methods based on the principles of self-organization of multi-agent systems (SO_MAS) [5]. Promising are the methods of structural identification that use an ontological approach to the construction of mathematical models [6].

In this article, we will limit ourselves to considering the stage of parametric identification of models. Therefore, we will assume that the structure of the model is previously determined on the basis of a detailed study of the behavior of the system or a certain hypothesis about the formal relationships between its input and output signals.

Note that the results published in this article are a continuation of the research presented in [7-11].

Different approaches are used for parametric identification of a mathematical model. In the case of a linear discrete dynamic model in the space of state variables, there is an effective analytical algorithm for parametric identification of the Ho-Kalman, and for a bilinear model, the analytical algorithm proposed by Isidori [4] is used. These algorithms are appropriate to apply at the initial stage of parametric identification of the model if in the structure of the generally nonlinear model it is possible to clearly distinguish the linear [7] or bilinear part. Then, the parameters of these parts of the model obtained at this stage are used as an initial approximation for the identification process of the entire nonlinear model.

For parametric identification of a mathematical model with a significantly complex structure, it is necessary to spend significant computational and especially time resources. This reduces the practical value for the researcher of the modeling process. Therefore, an approach that involves the formal division of a single complex model into simpler submodels that are identified in parallel is quite effective [4, 7]. It is obvious that parallelization of computational procedures accelerates the process of parametric identification of the entire model. And the implementation of algorithms for determining the parameters of individual submodels can hypothetically be significantly simplified. Therefore, the identification process of the general model, even with the need to coordinate the parameters of the submodels, will require less computational cost.

A big problem for significantly nonlinear models is the lack of universal analytical algorithms for their parametric identification. The solution to this problem is the use of an optimization approach. It consists in the fact that as a result of solving the optimization problem, such model parameters are found that allow it to adequately reproduce real processes [8]. Usually, the adequacy of a “black box” model is estimated by the deviation of the real output signals $\vec{y}^{*(k)}$ from the simulated output signals $\vec{y}^{(k)}$. The expression that reflects this deviation defines the objective function in the corresponding optimization problem:

$$Q(\vec{p}) = g\left(\left\|\vec{y}^{*(k)} - \vec{y}^{(k)}\right\|\right) \to \min_{\vec{p}} Q, \tag{2}$$

where $Q(\vec{p})$ is the objective function, the arguments of which are the parameters of the model (1); g is a functional of various types, in particular [8], which reflects the deviation of the real output signals $\vec{y}^{*(k)}$ from the output signals $\vec{y}^{(k)}$ of the model (1).

To solve the optimization problem of type (2), various deterministic and stochastic optimization methods can be used. The choice of a specific method depends on many circumstances [9]. Therefore, the researcher should be able to choose from some methods the one that is most successful for the given problem of parametric identification of a mathematical model. Such a choice involves testing different optimization methods on a certain class of problems [10]. For this, it is necessary to integrate as many algorithms as possible that implement these methods and their modifications into a single software environment [11]. In addition, this environment should be an element of the software package that is directly used for both creating and using the created models. That is why the purpose of this article is to design the architecture of this software environment and develop its individual software modules, which implement and integrate various optimization algorithms used in the process of parametric identification of mathematical models of complex objects and processes.

MATERIALS AND METHODS

The defining element of the considered software complex for the identification of mathematical models is a set of modules that implement optimization algorithms. For example, we will consider only a few of them. The completed implementation of the created software complex should cover as many algorithms as possible.

In this article, we focus on three algorithms that are modifications of the deterministic quasi-Newton method and four variations of the stochastic optimization method. It is important that deterministic methods have their advantages and disadvantages compared to stochastic ones [12]. Therefore, in practical mathematical modeling, there is a need to apply both of these approaches to solving various optimization problems.

Quasi-Newton methods

A feature of quasi-Newton optimization methods that distinguishes them from Newtonian methods is that the Hessian matrix is not directly calculated, but is approximated by another matrix [13]. This, in particular, reduces the computational and time costs of the optimization process. Examples of quasi-Newton optimization algorithms are the classical Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm and its modifications, namely, limited-memory BFGS (L-BFGS) and L-BFGS with box constraints (L-BFGS-B).

BFGS algorithm

As is known, the optimization process is iterative. At each of its iterations, the direction and step of moving to the minimum point of the objective function at the next iteration are determined. The direction and step can be determined directly or indirectly through the parameters of searching for the next point on the trajectory of movement to the minimum, which are inherent to a particular optimization method.

The BFGS algorithm is based on the idea that at each i-th iteration, the inverse Hessian matrix is approximated by the approximate matrix B_i, which is obtained based on the calculation of the first partial derivatives of the objective function [14]. As a result, the direction of further searching for the minimum of the objective function is obtained, and the values of its arguments are calculated at the $i+1$-th iteration. Regarding the optimization problem of type (2), the vector of parameters of the mathematical model at the $i+1$-th iteration is calculated by the formula

$$\vec{p}_{i+1} = \vec{p}_i - \gamma \cdot B_i \cdot g_F(\vec{p}_i), \tag{3}$$

where γ is the optimal step length in the chosen search direction; $g_Q(\vec{p}_i)$ is the gradient of the objective function $Q(\vec{p})$.

L-BFGS algorithm

The basic BFGS algorithm assumes that to determine the matrix B_i at the i-th iteration, it is necessary to store the gradients calculated at all previous iterations. This increases the memory consumption of the computing system and, as a result, slows down the optimization process. In the L-BFGS algorithm, this drawback is eliminated by reducing the number of previous iterations, the results of which must be taken into account at the current iteration [15].

L-BFGS-B algorithm

The L-BFGS-B algorithm is a modification of the L-BFGS algorithm. It provides for imposing restrictions on the arguments of the objective function [16], which makes it possible to make the search for the next point on the path to the minimum more directed.

Software implementation of the quasi-Newton algorithm

The BFGS and L-BFGS-B algorithms are implemented in the scipy.optimize library of the Python environment. The programmatic call of these algorithms is carried out through a universal function [17]

```
minimize(fun, x0, args=(), method=None, jac=None, hess=None, hessp=None,
         bounds=None, constraints=(), tol=None, callback=None, options=None),
```

where the value of parameter `method` ‘BFGS’ or ‘L-BFGS-B’ indicates the corresponding algorithm.

Stochastic directed cone methods

In the directed cone method, at each i-th iteration of the main stage of the optimization process, the further search direction on the way to the minimum point of the objective function is chosen arbitrarily within a certain cone. The vertex of this cone lies at the point

$\vec{p}_i$. The segment of space that limits the cone is determined by its deployment angle Ψ_i and height h_i. Several random test points are selected based on the cone. Among them, the one in which the objective function has the smallest value is chosen. It becomes the vertex of the cone, which is built in the next iteration [4, 8, 10].

Basic algorithm of the directed cone method

The basic algorithm of the directed cone method, proposed by Rastrigin, and its initial adaptation, carried out by Bukashkin [4, 8], assume that the angle of deployment and the height of the cone remain constant at all iterations of the optimization process ($\Psi_i = const, h_i = const, i = 1,2, ...$). This makes this process inefficient [9, 10]. Therefore, algorithms have been proposed that provide for the adaptation of the mentioned optimization parameters to the conditions determined by the current behavior of the objective function.

Algorithm with adaptation of the search step length

The height of the cone h_i basically determines the distance between two neighboring (i-th and $i+1$-th) points on the path of finding the minimum of the objective function. Therefore, h_i can be considered as the search step at the i-th iteration. Depending on the local behavior of the objective function, the current search step should be adapted according to the formula [4, 10]

$$h_{i+1} = h_i \cdot \left(\frac{t_i}{t_{ef}}\right)^{\frac{1}{z-1}}, \tag{4}$$

where $t_i = \tilde{d}/d$ (d is the total number of attempts made with the search step length h_i and $\tilde{d}$ is the number of successful attempts when the objective function decreased); t_{ef} ($0 < t_{ef} < 1$) is a parameter that is set to normalize t_i; z is the dimensionality of the objective function.

In general, the search step should be reduced near the minimum point of the objective function (to determine the minimum as accurately as possible), and at a considerable distance from the minimum point, on the contrary, it should be increased (to speed up the optimization process) [11].

Algorithm with simultaneous adaptation of the cone deployment angle and the search step length

In order to make the optimization process at individual stages, where the objective function allows, more targeted, it was proposed to reduce the cone deployment angle. Such angle adaptation is associated with the need for simultaneous adaptation of the search step length. In this case, a decrease in the cone deployment angle should be accompanied by a corresponding increase in the search step length and vice versa [4, 8].

It has been experimentally proven that the best result of the optimization search is achieved when the adaptation at the i-th iteration of the cone deployment angle and the search step length occurs according to the corresponding formulas [4, 10]

$$\Psi_{i+1} = \delta^{\mu \cdot \alpha - 1} \cdot \Psi_i, \tag{5}$$

$$h_{i+1} = h_i \cdot \left(\frac{t_i}{t_{ef}}\right)^{\frac{1}{z-1}} \cdot \delta^{-(\mu \cdot \alpha - 1)}, \tag{6}$$

where $\delta > 1$ is some constant (the larger the value of this constant, the faster the cone deployment angle adapts); α is a coefficient that characterizes the deviation of the cone axis at two consecutive iterations; μ $(1 \leq \mu \leq 2)$ is some coefficient.

Tunneling algorithm

The tunneling algorithm aims to significantly reduce the time of the optimization process. This is achieved by significantly increasing the search step length in a statistically justified direction on certain sections of the path to the minimum point of the objective function [4, 10].

Software implementation of directed cone method algorithms

For the software implementation of the algorithms of the directed cone method, original software modules were developed. Initially, they were implemented in the Delphi software environment [18]. For example, below is a fragment of a Delphi program for calculating a linear dynamic model with constant parameters in the space of state variables:

$$\begin{cases} \vec{x}^{(k+1)} = F \cdot \vec{x}^{(k)} + G \cdot \vec{v}^{(k)} \\ \vec{y}^{(k+1)} = C \cdot \vec{x}^{(k+1)} + D \cdot \vec{v}^{(k+1)}, \end{cases} \tag{7}$$

where F, G, C, D are the matrices of model parameters of the corresponding dimensions.

```
K: TVector;
F, G, C, D: TMatrix;
NewX: TVector;
VectorDeclare(Params.K, 'K', StateDimension);
MatrixDeclare(Params.F, 'F', StateDimension, StateDimension);
MatrixDeclare(Params.G, 'G', StateDimension, InputDimension);
MatrixDeclare(Params.C, 'C', OutputDimension, StateDimension);
atrixDeclare(Params.D, 'D', OutputDimension, InputDimension);
SetLength(Params.NewX, StateDimension);
var
  i, j, k, l: Integer;
  S: Double;
// Calculation of the vector y
for i := 0 to OutputDimension - 1 do
begin
  S := 0;
  for j := 0 to StateDimension - 1 do
    S := S + Params.C[i][j] * Params.x[j];
{
  for j := 0 to InputDimension - 1 do
    S := S + Params.D[i][j] * v[j];
}
  y[i] := S;
end;

// Calculating the new value of the vector x
for i := 0 to StateDimension - 1 do
begin
  S := Params.K[i];
  for j := 0 to StateDimension - 1 do
    S := S + Params.F[i][j] * Params.x[j];
  for j := 0 to InputDimension - 1 do
    S := S + Params.G[i][j] * v[j];
  Params.NewX[i] := S;
end;
```

```
for i := 0 to StateDimension - 1 do
  Params.x[i] := Params.NewX[i];
```

However, later, in order to adapt to modern programming systems, the optimization algorithms of the directed cone method were modernized and implemented in the Python environment. This made it possible to integrate these algorithms into a single software system with other algorithms that are standardly implemented in this environment, for example, with algorithms of quasi-Newton optimization methods. Note that the created Python version of the software allows you to import previously programmatically implemented models in the Delphi environment, in particular the one presented above. This allows you to use already developed model structures in the Python environment.

Integration of optimization algorithms

For practical modeling, the researcher needs to be given the opportunity to choose one of several optimization algorithms. Therefore, these algorithms must be integrated with each other at the software level. The logic of using integrated optimization algorithms in the process of parametric identification of a mathematical model is reproduced in **Fig. 1**. This scheme considers the possibility of choosing among the following optimization algorithms: two quasi-Newton algorithms – BFGS and L-BFGS-B, and three algorithms of the directed cone method (DCM) – with adaptation of the search step length (DCM-S), with simultaneous adaptation of the cone deployment angle and search step length (DCM-A-S), with tunneling (DCM-T). Note that in the final implementation of the software complex, it is necessary to provide for the integration of other optimization algorithms.

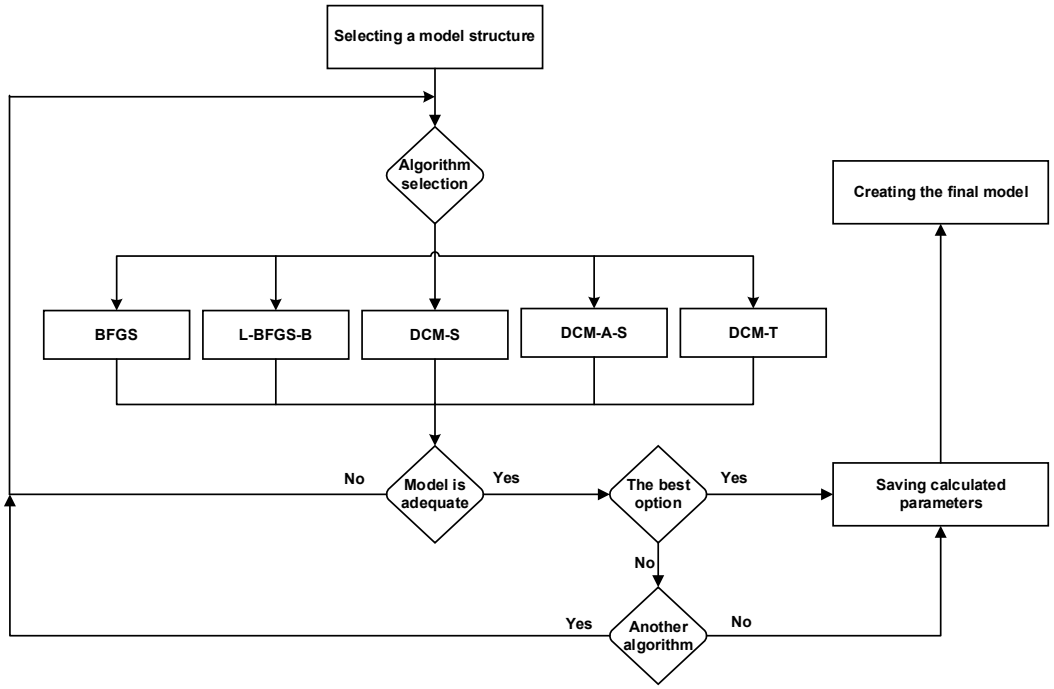

Fig. 1. The logic of using optimization algorithms integrated in a single software package in the process of parametric identification of a mathematical model

After selecting the structure of the mathematical model, the researcher selects one of the listed algorithms. It is used to perform parametric identification of the model. If, according to the criteria selected by the researcher, the model with the calculated parameters does not meet the adequacy conditions, the researcher selects another of the

permissible optimization algorithms. The return to the selection of algorithms can also occur if the researcher considers it necessary to improve the model parameters. In the case when the best option, in the researcher's opinion, is achieved, then the final model is formed.

Note that the proposed integration of optimization algorithms allows us to research the effectiveness of their application on a class of optimization problems using the procedures proposed in [9].

RESULTS AND DISCUSSION

Programming environment requirements

The previously considered approach to the integration of various optimization algorithms should be implemented programmatically as one of the modules of a more complex software system, the purpose of which is to ensure the construction of various nonlinear mathematical models. Other elements of such a system are modules that provide: the system's interconnection with automated complexes for conducting physical experiments, interfaces with information systems, interaction with artificial intelligence (AI), support for databases (DB) and data warehouses (DW), interaction with widespread computer modeling systems, support for parallel computing, carrying out the calculations necessary for the construction and use of mathematical models, and the implementation of a convenient user interface. In addition, some technological requirements are put forward for modern software systems that determine the software environment in which these systems should be created.

In view of this, the programming environment for the developed system of identification of mathematical models should have a number of defining characteristics. In particular [9]: support cross-platform [19] object-oriented [20], generalized [21] and procedural programming [22], the ability to create software in a distributed software environment [23]. For simplicity of software development, it is worth using a high-level programming language [24]. However, in order for the developed software to effectively use the functions of system software, in particular, the functions of the operating system, it is necessary that the programming language has a number of properties of a low-level language [25].

Currently popular programming languages that have the listed properties are Java and C Sharp (C#). The basic constructs of these languages are borrowed from the C++ language [26]. The ISO/IEC 14882:2024 (C++23) standard is valid for C++ [27]. Standardization stimulates leading software manufacturers, in particular such giants of the IT industry as IBM, Oracle, Intel, Microsoft, to create their own compilers from C++, or to support their development within the framework of such projects as LLVM, Embarcadero [28]. The principles on which the concept of the creation and development of this language is based provide the possibility of: implementing any algorithms, using any programming styles, creating an interface with system software, interacting with other programming languages, and creating high-performance software [29]. The latter property is important for reducing time costs in the process of solving optimization problems.

It is also important that the MATLAB software environment, popular among a large number of researchers, is written in C++ and Java [30]. The Simulink visual modeling package integrated into this environment is a convenient tool for simulating the operation of various complex objects described by mathematical models [18].

So, taking into account all the previous considerations, it is appropriate to conclude that it is worth creating the program code of the developed system in a language that is related to C++ or can easily integrate with it.

Despite the significant capabilities of C-oriented programming languages, a large number of software developers, particularly in the field of modeling, prefer Python. The most important reason for the popularity of this language is that it is a high-level language with a simple but at the same time understandable syntax. In addition, it supports cross-platform and various programming concepts, which allows it to be used in different

hardware and software environments to solve different classes of problems [31]. If it is necessary to use low-level capabilities of the C language or previously created C program modules in Python, the C Foreign Function Interface (CFFI) for Python is used [32].

A large number of libraries have been developed for the Python language, which contain various standard functions and fundamental methods for implementing computational procedures. This makes it easier for the programmer to use them when solving specific problems. NumPy [33] and SciPy [34] libraries are used for modeling.

The Python programming environment can be easily integrated with other programming environments used for modeling. For example, in the program code written in Python, you can use the functions and methods of the MATLAB programming and numerical computing platform using the MATLAB Engine API for Python, and vice versa, you can access Python code libraries and constructs from the MATLAB environment by calling a function with the prefix py. [35].

The disadvantage of the Python programming environment is that it works in interpreter mode. This reduces the speed of computational procedures, which negatively affects the effectiveness of the use of optimization methods. To speed up Python programs, the PyPy interpreter is used, the work of which is accelerated by the Just-in-Time compiler [36], or the Codon compiler, created based on the byte-coded LLVM [37] or the similar Numba compiler [38]. In addition to compiling Python code, Codon and Numba provide parallelization of computational procedures, which further speeds up their execution.

Given that the developed software system must interact at the information level with other software systems and software environments, it is necessary to ensure the import and export of data between them. One of the most successful data formats that provides information coordination of different software environments is the csv (comma-separated values) format [39]. The standard of this format [40] also provides support for cross-platform. A file with the .csv extension can be created and edited directly by the user using standard office applications, for example, in the MS Excel spreadsheet environment [39]. This data format must be accepted by the MATLAB environment [41]. Therefore, it should be used to prepare data for model identification and export data to environments that these models use in practice.

Structure of the subsystem for constructing and evaluating a mathematical model

Let us consider the subsystem for constructing and evaluating a mathematical model. It is part of a more general system that performs a wide range of functions – from the processing of experimentally obtained data for modeling to the practical application of the constructed models. **Fig. 2** shows the general structure of the subsystem under consideration.

The subsystem includes the following functional modules:

- Interface module, which provides, on the one hand, user interaction with the subsystem, and on the other hand, the transmission of user control commands to other subsystem modules;
- Repository module is a software-implemented repository of model types defined in the developed software environment or borrowed from outside, which serve as the basis for the final identification of a specific mathematical model. The model type here is understood as the form of the mathematical description of the model (continuous, discrete, differential, integral, interval) and the initial structure of the model (linear or nonlinear with various types of nonlinearity);
- The selection module provides, at the user's request, the selection from the repository of the model type that is most advantageous for a specific modeling task;
- The structural identification module implements a user-defined structural identification algorithm based on the selected model type. In cases where parallel application of structural and parametric identification procedures is envisaged, the structural identification module directly interacts with the parametric identification module;

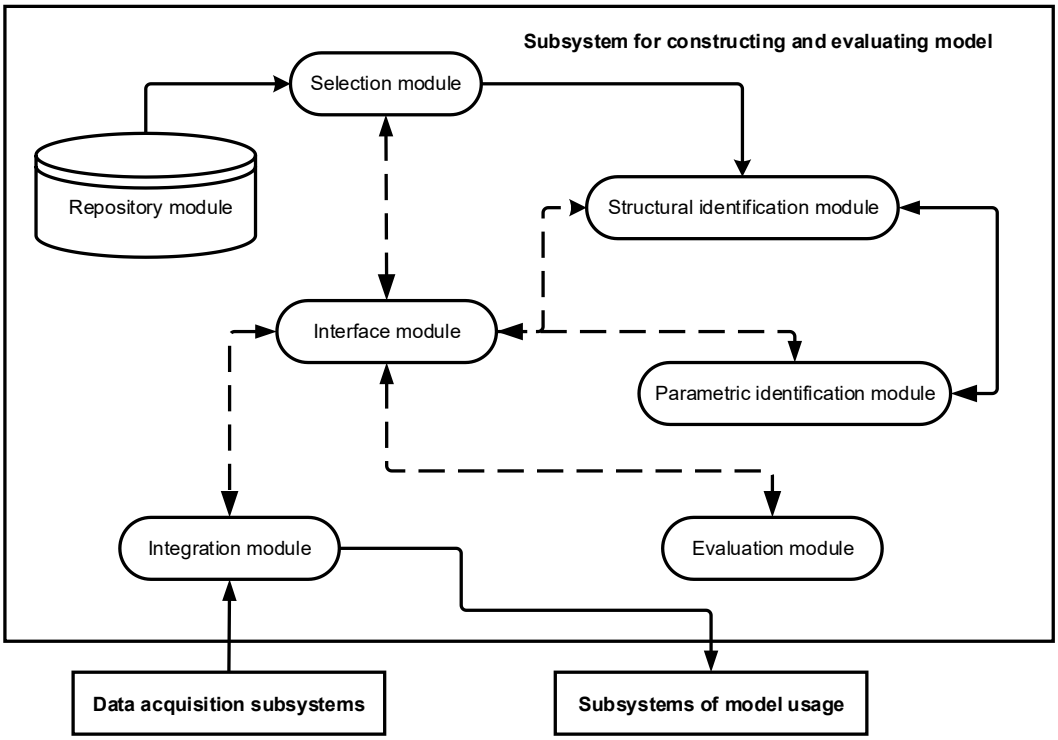

Fig. 2. General structure of the subsystem for constructing and evaluating a mathematical model

- The parametric identification module implements a user-defined parametric identification algorithm. An integral part of this module is the module for implementing optimization algorithms, which was discussed above;
- The evaluation module is used to determine the adequacy and accuracy of the constructed model according to user-defined criteria;
- The integration module provides interaction with other subsystems for:
  - interpretation of experimental data used to build the model (for example, data collected during physical experiments or during computer modeling of the studied processes, in particular in the MATLAB environment);
  - software representation of the constructed model for subsystems that use it in the process of studying the corresponding complex objects.

Technology of using the software environment

The user uses the software environment in interactive mode. The UML use cases diagram is shown in **Fig. 3**.

The software environment allows the user to activate the following main options:

- import data obtained as a result of physical or computational experiments, which are used to identify models;
- export constructed models to other software environments for their practical application;
- filling the repository of model types;
- select the type of models according to the criteria that correspond to the modeling task;
- select the method of structural identification of the model, in particular: MSR [42], MSC [43], GMDH [44], GA [45], SO_MAS [46];

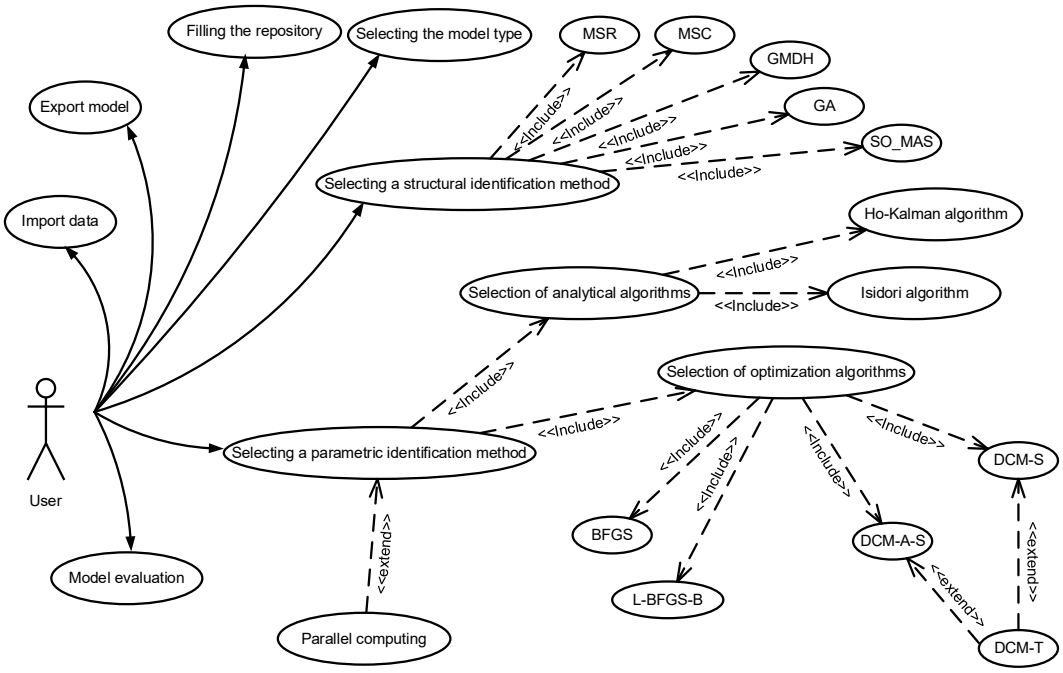

Fig 3. UML Use Case Diagram of Software Operation

- select the method of parametric identification of the model. In the case of using analytical methods, it is possible to select the Ho-Kalman algorithm [47] or the Isidori algorithm [48]. In the case of using optimization methods, the selection of BFGS, L-BFGS-B, DCM-S, and DCM-A-S algorithms is provided. At certain stages of the optimization process, the DCM-S and DCM-A-S algorithms can be supplemented with the DCM-T algorithm. If necessary, during parametric identification of the model, parallel computing algorithms can be used;
- evaluate the constructed model according to certain criteria.

Discussion

Modern software must comply with the latest approaches to its creation and use, as well as IT trends for its future development. Therefore, it is important that the developed software environment is not only functional and user-friendly but also adapted to modern requirements. In particular, it must support cloud technologies (CT) and artificial intelligence (AI) services.

Cloud infrastructure can be implemented on cloud platforms such as Amazon Web Services, Microsoft Azure [49] or Google Cloud Platform [50]. This infrastructure provides services such as scalable cloud computing, cloud data storage, supports MapReduce parallel computing technology [51] and Kubernetes cloud containerization [52], which provides automation of software deployment, scaling and management in a cloud environment, provides RESTful API services [53], which in particular allow you to load data for modeling directly from distributed sensors, and provides integration of the cloud environment with Python libraries, in particular with Matplotlib and Seaborn libraries [54], which allow you to visualize the results of modeling.

Integration of AI tools into the developed software environment increases its intellectualization by adding the function of providing recommendations on modeling procedures, processes of formalized description of models, and their practical use. The integration process is carried out using APIs provided by developers of AI chatbots. For

example, for ChatGPT, OpenAI Corporation provides cloud APIs for various programming environments, in particular for Python [55].

CONCLUSION

This study attempts to propose architectural and software solutions for creating a software environment for identifying mathematical models of complex objects, which is based on the application of an optimization approach. These solutions allow creating software that provides:

- a choice of a wide range of optimization algorithms that implement various optimization methods;
- implementation of all computational stages of mathematical model development, which include structural and parametric identification, parallelization of computational procedures, and evaluation of the resulting model;
- preparation of input data for modeling and creation of software implementation of the model for the environments of its use;
- integration with other software environments and modeling subsystems.

The choice of the Python programming system is justified, as it allows implementing all the designed functionality of the software environment and corresponds to the modern programming paradigm.

Promising directions for future research are formulated, which include the integration of the developed software into the cloud environment, as well as integration with publicly available AI mechanisms.

ACKNOWLEDGMENTS AND FUNDING SOURCES

The author received no financial support for the research, authorship, and/or publication of this article.

COMPLIANCE WITH ETHICAL STANDARDS

The author declares that he has no competing interests.

AUTHOR CONTRIBUTIONS

The author has read and agreed to the published version of the manuscript

REFERENCES

[1] Voronin, A., Lebedeva, I., & Lebedev, S. (2022). A nonlinear mathematical model of dynamics of production and economic objects. *Development Management*, 21(2), 8-15. doi.org/10.57111/devt.20(2).2022.8-15

[2] Batlovskyy, O. O., Foros, G. V., & Siforov, O. I. (2023). *Fundamentals of mathematical modeling.* Odesa: OSUIA (in Ukrainian). https://files.znu.edu.ua/files/Bibliobooks/Inshi78/0058302.pdf (in. Ukr.).

[3] Hassija, V., Chamola, V., Mahapatra, A., Singal, A., Goel, D., Huang, K., … Hussain, A. (2023). Interpreting Black-Box Models: A Review on Explainable Artificial Intelligence. *Cognitive Computation.* 16, 45–74. doi.org/10.1007/s12-559-023-10179-8

[4] Stakhiv, P. G., Kozak, Yu. Ya., & Hoholyuk, O. P. (2014). *Discrete modeling in electrical engineering and related fields*. Lviv: Publishing House of Lviv Polytechnic (in Ukrainian).

[5] Dyvak, M.P., Pukas, A.V., Porplytsia, N.P., & Melnyk, A.M. (2021). *Applied problems of structural and parametric identification of interval models of complex objects*. Ternopil: University Thought (in Ukrainian). http://dspace.wunu.edu.ua/handle/316497/43614

[6] Dyvak, M.P., Melnyk, A.M., Manzhula, V.I., Spivak, I.Ya., & Porplytsia, N.P. (2024). *Knowledge-oriented systems for identification of interval mathematical models of complex dynamic and static objects*. Ternopil: University Thought (in Ukrainian).
[7] Stakhiv, P., Melnyk, B., Hoholyuk, O., & Trokhaniak, S. (2024). Application of parallel computing technology for modelling complex dynamic objects. *Computational problems of electrical engineering,* 14 (1), 30-35. https://doi.org/10.23939/jcpee2024.01.030
[8] Stakhiv, P., Gogoluk, O., Gamola, O., Melnyk, B., Maday, V., & Melnyk, N. (2023) Application of the Rastrygin's Method in Modeling of Complex Electrical Systems. *2023 24th International Conference on Computational Problems of Electrical Engineering (CPEE)*. Grybów: IEEE. https://doi.org/10.1109/CPEE59623.2023.10285315
[9] Melnyk, B., Kozak, Yu., Melnyk, N., Trokhaniak, S., Dyyak, I., & Yatsyshyn, M. (2025). Research into the Effectiveness of Using the Stochastic Optimization Method in Constructing Discrete Dynamic Models. *2025 15th International Conference on Advanced Computer Information Technologies ACIT'25*. Šibenik: IEEE, 28-34. https://doi.org/10.1109/ACIT65614.2025.11185720
[10] Melnyk, B., Dyvak, M., Melnyk, A., Tkacz, E., Banasik, A., Chwał, J., & Dzik, R. (2025). Application of the Directed Cone Method for the Identification of Mathematical Models of Electromechanical Systems. *Energies*, 18, 5949. doi.org/10.3390/en18225949
[11] Melnyk, B., Stakhiv, P., Melnyk, N., Franko, Yu., Seniv, B., & Martsenyk, Ye. (2024). Software for Implementing the Directed Cone Optimization Method. *2024 14th International Conference on Advanced Computer Information Technologies ACIT'24*. Ceske Budejovice: IEEE, 6-12. https://doi.org/10.1109/ACIT62333.2024.10712522
[12] Fulber-Garcia, V., & Aibin, M. (2024). Deterministic and Stochastic Optimization Methods. *Baeldung*. https://www.baeldung.com/cs/deterministic-stochastic-optimization
[13] Steven, G. J. (2019). Quasi-Newton optimization: Origin of the BFGS update. Notes for 18.335 at MIT. https://github.com/mitmath/18335/blob/spring21/notes/BFGS.pdf
[14] Dai, Y.-H. (2006). Convergence Properties of the BFGS Algoritm. *SIAM Journal on Optimization*. 13 (3). 693-701. https://doi.org/10.1137/S1052623401383455
[15] Singh, M., Shastri, A., & Shaw, K. (2023). *Machine Learning and Optimization for Engineering Design*. Springer Nature. https://doi.org/10.1007/978-981-99-7456-6
[16] Zhu, C., Byrd, R. H., Lu, P., & Nocedal, J. (1997). Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-Scale Bound-Constrained Optimization. *ACM Transactions on Mathematical Software*, 23 (4), 550–560. https://doi.org/10.1145/279232.279236
[17] scipy.optimize. *SciPy*. https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html#scipy.optimize.minimize
[18] Hoholyuk, O., Kozak, Yu., Rosołowski, E., & Stakhiv, P. (2019). Increasing of the effectiveness of algorithms implementation for development of discrete macromodels and their adaptation to electric circuits simulation programs. *Technical electrodynamics*, 2 (in Ukrainian). https://doi.org/10.15407/techned2019.02.003
[19] Popular programming languages for developing cross-platform applications. *Kotlin* https://kotlinlang.org/docs/multiplatform/programming-languages-cross-platform.html
[20] BasuMallick, Ch. (2022). What is OOP (Object Oriented Programming)? Meaning, Concepts, and Benefits. *Spiceworks*. https://www.spiceworks.com/tech/devops/articles/object-oriented-programming/
[21] Siek, J.G., & Lumsdaine, A. (2011). A language for generic programming in the large, *Science of Computer Programming*. 76 (5), 423-465. https://doi.org/10.1016/j.scico.2008.09.009.

[22] Coursera Staff (2025). Procedural Programming Language: What It Is and When It's Used. *Coursera*. https://www.coursera.org/articles/procedural-programming-language
[23] L'Erario, A., Fabri, J.A., Domingues, A., Duarte, A., Palácios, R., & Pessôa, M. (2012). A Distributed Software Development Environment Dynamics Model. *2012 IEEE Seventh International Conference on Global Software Engineering*, Porto Alegre, 184-184, doi: 10.1109/ICGSE.2012.13.
[24] What is High Level Language? *Geeksfor-Geeks*. https://www.geeksforgeeks.org/software-engineering/what-is-high-level-language/
[25] Şentürk, C. (2024). This Is All You Need to Know About Low-Level Languages. *Tuple*. https://www.tuple.nl/en/blog/all-you-need-to-know-about-low-level-languages
[26] Mensh T. (2026). An In-depth Look at C++ vs. Java. *Toptal*. https://www.toptal.com/developers/c-plus-plus/c-plus-plus-vs-java
[27] ISO. (2024). *ISO/IEC 14882:2024. Programming languages – C++*. https://www.iso.org/standard/83626.html
[28] Rozenberg, Z. (2025). Top C++ compilers in 2025. *Incredibuild*. https://www.incredibuild.com/blog/top-c-compilers
[29] Stroustrup, B. (2020). Thriving in a Crowded and Changing World: C++ 2006-2020. *Proceeding of the ACM Programming Languages*. 4 (HOPL), 7, 1-168 https://doi.org/10.1145/3386320
[30] Schulze, J. (2025). What Is MATLAB? Overview and FAQ. *Coursera*. https://www.coursera.org/articles/what-is-matlab
[31] Haniel, J. (2024)). What is Python used for? 11 most common use cases for Python. *JavaRush*. https://javarush.com/ua/groups/posts/dlja-chogo-vikoristovutjhsja-python
[32] CFFI. *CFFI documentation*. https://cffi.readthedocs.io/en/stable/index.html
[33] NumPy. (2025). *NumPy documentation.* https://numpy.org/doc/stable/
[34] SciPy. (2026). https://scipy.org/
[35] Using MATLAB with Python. *MATLAB*. https://www.mathworks.com/products/matlab/getting-started/using-matlab-python.html#matlab-with-python
[36] PyPy-Features. *PyPy*. https://pypy.org/features.html
[37] Welcome to Codon. *Condon*. https://docs.exaloop.io/
[38] Numba. https://numba.pydata.org/
[39] Shafranovich, Y. (2005). Common Format and MIME Type for Comma-Separated Values (CSV) Files. *SolidMatrix Technologies, Inc*. https://datatracker.ietf.org/doc/html/rfc4180
[40] Cheusheva, S. (2023). How to convert (open or import) CSV file to Excel. *Ablebits.com*. https://www.ablebits.com/office-addins-blog/convert-csv-excel/
[41] Supported File Formats for Import and Export. *MATLAB Help Center*. https://www.mathworks.com/help/matlab/import_export/supported-file-formats-for-import-and-export.html
[42] Obertyuch, R.R., & Slabkyy, A.V. (2025). *Mathematical modeling of mechanical systems*. Vinnytsia: VNTU (in Ukrainian). https://pdf.lib.vntu.edu.ua/books/2025/Obertjukh_2025_119.pdf
[43] Silvestrov, A**.,** Ostroverkhov, M**.,** Spinul, L**.,** Serdyuk, A**.,** & Falchenko, M**.** (2022). Structural and Parametric Identification of Mathematical Models of Control Objects Based on the Principle of Rational Complication. *2022 IEEE 3rd KhPI Week on Advanced Technology (KhPIWeek)*. Kharkiv: IEEE. 1-4. doi.org/10.1109/KhPIWeek57572.2022.9916340
[44] Amiri, M., & Soleimani, S. (2021). ML-based group method of data handling: an improvement on the conventional GMDH. *Complex Intelligent Systems.* 7, 2949–2960. https://link.springer.com/article/10.1007/s40747-021-00480-0#Bib1

[45] McCall, J. (2005). Genetic algorithms for modelling and optimization. *Journal of Computational and Applied Mathematics*, 184 (1), 205-222. https://doi.org/10.1016/j.cam.2004.07.034
[46] Di Marzo Serugendo, G., Gleizes, M.-P., & Karageorgos, A. (2005). Self-organization in multi-agent systems. *The Knowledge Engineering Review*, 20(2), 165-189. doi:10.1017/S0269888905000494
[47] Petreczky, M. (2023). *Realization theory of cyber-physical systems and its application to system identification and model reduction. Optimization and Control [math.OC]*. Université de Lille. https://hal.science/tel-04144797/document
[48] Panos M. Pardalos,P.M., & Yatsenko, V. (2008). *Optimization and Control of Bilinear Systems. Theory, Algorithms, and Applications*. NY: Springer. https://link.springer.com/book/10.1007/978-0-387-73669-3
[49] Perri, D., Simonetti, M., & Gervasi. O. (2022). Deploying Efficiently Modern Applications on Cloud. *Electronics*, 11 (3), 450. https://doi.org/10.3390/electronics11030450
[50] What is Google Cloud Platform (GCP)? *Tandent*. https://ua.talend.com/resources/what-is-google-cloud-platform/
[51] Wang, Y., Goldstone, R., Yu, W., & Wang, T. (2014). Characterization and Optimization of Memory-Resident MapReduce on HPC Systems. *2014 IEEE 28th International Parallel and Distributed Processing Symposium*. Phoenix: IEEE, 799-808, https://doi.org/10.1109/IPDPS.2014.87
[52] Kubernetes And Cloud Native Architecture: A Comprehensive Guide. *Kubegrade*. https://kubegrade.com/kubernetes-cloud-native-architecture/
[53] What is RESTful API? *AWS*. https://aws.amazon.com/what-is/restful-api/
[54] Muddarla, B., & Vatti, P. R. (2024). Machine Learning in Cloud Environments: Leveraging SQL and Python for Big Data Analytics. *Nanotechnology Perceptions*. 20 (7). 12-21. https://doi.org/10.62441/nano-ntp.v20i7.3794
[55] Dyouri, A. (2026). How to Integrate ChatGPT's API With Python Projects. *Real Python*. (19.01.2026). https://realpython.com/chatgpt-api-python/

ПРОГРАМНЕ СЕРЕДОВИЩЕ ДЛЯ ІДЕНТИФІКАЦІЇ НЕЛІНІЙНИХ МАТЕМАТИЧНИХ МОДЕЛЕЙ З ВИКОРИСТАННЯМ ОПТИМІЗАЦІЙНИХ ПРОЦЕДУР

Богдан Мельник
Кафедра інформаційних систем у менеджменті,
Львівський національний університет імені Івана Франка,
Пр. Свободи, 18, Львів, 79008, Україна

АНОТАЦІЯ

Вступ. Для адекватного опису складних систем необхідно використовувати суттєво нелінійні математичні моделі. Ефективним підходом до ідентифікації таких моделей є застосування оптимізаційних процедур, які базуються на різних методах оптимізації. Для комплексної реалізації цих процедур необхідно створити програмне забезпечення, яке відповідає сучасним тенденціям в галузі комп'ютерного моделювання.

Матеріали та методи. Для ідентифікації моделей використовують різні оптимізаційні алгоритми, які розроблені в межах різних методів оптимізації, зокрема у межа квазіньютонівських методів чи стохастичних методів. У статті розглядаються алгоритми BFGS, L-BFGS, L-BFGS-B, що належать до першого класу методів, а також алгоритми, які реалізують метод стохастичного спрямованого конуса, а саме: базовий алгоритм та три його модифікації. Дослідник, який займається математичним моделюванням, вибирає один із програмно реалізованих оптимізаційних алгоритмів, які найкраще відповідають сформульованій задачі моделювання. Для цього він повинен мати змогу оцінити якісні показники цих алгоритмів, які інтегровані у єдиному програмному середовищі.

Результати. Програмне середовище для ідентифікації математичних моделей з використанням оптимізаційних процедур, окрім специфічних особливостей, пов'язаних з ідентифікацією, повинно відповідати сучаснім парадигмам програмування і його використання. Зважаючи на це, запропоновано використати для його створення середовище програмування Python. Розглянуто структуру програмної підсистеми, яка реалізує вибір типу математичної моделі, її структурну і параметричну ідентифікацію та оцінку, а також інтеграцію з іншими підсистемами створюваного програмного середовища. Сформульовані загальні вимоги для забезпечення його функціональності. Розглянуто перспективи застосування хмарних ІТ та ШІ.

Висновки. Запропоновані програмні і архітектурні рішення дають змогу створити програмне середовище, яке суттєво полегшить процес комп'ютерного створення математичних моделей різних складних систем. При цьому воно залишається відкритим для подальшої модернізації і розширення функціональних можливостей.

Ключові слова: математична модель, ідентифікація, оптимізаційні алгоритми, програмне забезпечення, архітектура.

Received / Одержано	Revised / Доопрацьовано	Accepted / Прийнято	Published / Опубліковано
04 May, 2026	12 June, 2026	15 June, 2026	29 June, 2026