Research Article

Alexander V. Sokolov and Vladimir V. Voloshinov*

Model Selection by Balanced Identification: the Interplay of Optimization and Distributed Computing

<https://doi.org/10.1515/comp-2020-0116>

Received Aug 02, 2019; accepted Mar 03, 2020

Abstract: The technology of formal quantitative estimation of the conformity of mathematical models to the available dataset is presented. The main purpose of the technology is to make the model selection decision-making process easier for the researcher. The method is a combination of approaches from the areas of data analysis, optimization and distributed computing including: cross-validation and regularization methods, algebraic modeling in optimization and methods of optimization, automatic discretization of differential and integral equations, and optimization REST-services. The technology is illustrated by a demo case study. A general mathematical formulation of the method is presented. It is followed by a description of the main aspects of algorithmic and software implementation. The list of success stories of the presented approach is substantial. Nevertheless, the domain of applicability and important unresolved issues are discussed.

Keywords: nonparametric identification, inverse problems, regularization, distributed computing, Everest platform

1 Introduction

The same object, process or phenomenon can be described by different mathematical models. The model selection among possible candidates remains one of the significant problems in applied mathematics. Mathematical description depends on both the qualitative knowledge of the object (which can be formalized as mathematical state-

ments), and the availability of quantitative data, their volume, detail, reliability and accuracy. The more complex a phenomenon under consideration and the corresponding mathematical model are, the more detailed and reliable the measurement should be. Usually to find a balance between a model complexity and an available measurement an easy approach is used: various models and datasets are tested until the built model meets the research goals. There is no universally accepted approach here. In each case, various parametrization, different identification and verification procedures and a variety of software are used. In addition, the model selection criterion is not formalized and the selection of the particular model is often made subjectively.

What we need are as follows: a technology that simplifies the model selection procedure for the researcher; a unified method that gives a quantitative estimation of the conformity of the mathematical description to the available dataset for each model under study.

This paper presents a method of a *balanced identification* (also called the SVF-method, *simplicity vs fitting*). The method has been developed to fulfil the above requirements. Besides a formal description a software implementation of the method is presented. It is called the SVF-technology. Even though the method is rather mature and has been successfully used in a number of various research projects over the past few years, its general mathematical scheme has only been published recently [1]. The basis for the SVF-method are nonparametric identifications with *regularization* [2] and the well-known *cross-validation* method [3–5]. Regularization is used to get the balance between simplicity of the model (e.g. smoothness of model functions) and measure of fitting to the available data set (accuracy of reproducing experimental data by model functions). Cross-validation error is used as a quantitative estimation of the model compliance to the dataset.

In general, the SVF-method may be applied to both parametric and nonparametric identification (where the model under study contains unknown functions). Depending on the class of the models, the variational problems

*Corresponding Author: Vladimir V. Voloshinov: Institute for Information Transmission Problems (Kharkevich institute); Email: vv_voloshinov@iitp.ru

Alexander V. Sokolov: Institute for Information Transmission Problems (Kharkevich institute); Email: alexander.v.sokolov@gmail.com

arising here can be interpreted as problems of variational spline approximation [6], nonparametric regression [7], predictive modeling [5], machine learning [4] as well as others..

There is a trend in research on nonlinear regression to create algorithms of “automatic” composition of the best regression function from a predefined set of basic functions [8]. Here, the object under study is thought of as a black-box and the best regression remains a purely phenomenological description. SVF-technology is not a fully automated routine. It enables quantitative comparison of a given set of models, but it is the researcher who makes modifications to the model (e.g. include/exclude model constraint equations). Moreover, the SVF-method can be applied to structural mathematical models describing the internal structure of the phenomenon, e.g. via algebraic or integro-differential equations. One can say that the SVF-method generalizes the concept of regression with additional constraints. A known example of this is a so called monotonic regression [9].

For a given model, the SVF-method requires solving a bilevel optimization problem, where at the lower level, we have a number of independent mathematical programming problems to get the value of the objective function of the upper-level problem. Finally, the optimal values of cross-validation error (CV-error) and regularization coefficients are obtained for a given model. A root-mean-square error will be presented for reference only. Bilevel optimization is a well-known hard challenge for numerical methods [10]. Current implementation of the SVF-method is based on surrogate optimization and on the ideas of the *Pshenichny-Danilin linearisation method* [11].

Because at the lower level we have a number of independent problems, it is possible to increase performance of the technology by distributed computing. It should be noted that optimization in a distributed computing environment is already used in regression analysis, e.g. [12]. In SVF-technology the basic tool for solving independent mathematical programming problems in parallel is the Everest optimization service [13, 14] based on the Everest toolkit [15], <http://everest.distcomp.org>.

This paper is organized as follows. The next section demonstrates an example of applying the SVF-technology to the modeling of an oscillator with friction. Section 3 contains mathematical formulation of the basic concepts of the SVF-method including the formulation of the main bilevel optimization problem. Section 4 presents the main aspects of the SVF-method software implementation: usage of Pyomo package (Python Optimization Modeling Objects), <http://www.pyomo.org>; special symbolic notation that simplifies formulation of the model and regulariza-

tion rule; discretization of differential and integral equations if they are present in the model; usage of the Everest optimization service. Successful use of SVF-technology in various research works are presented in Section 5. Some important open issues related to the current implementation of SVF-technology are discussed in Section 6 followed by the acknowledgements.

2 Demonstrative example

Before proceeding with a formal description of the method it is worth considering a demonstrative “use case” on the example of a classical damped oscillator described by the following function of time t :

$$x(t) = \sin\left(\frac{\sqrt{4k-\mu}}{2}t\right) \cdot \exp\left(-\frac{\mu}{2}t\right) + \Delta x, \quad (1)$$

where: $x(t)$ is a trajectory of oscillation; $k=1.56$ (elastic coefficient); $\mu=0.4$ (friction factor); $\Delta x=1.2$ (initial displacement of oscillator).

Assume that we do not know equation (1) in advance. What we have is a dataset (measurement series) on a time interval T (hereafter $0:n$ denotes the set of numbers $\{0, 1, 2, \dots, n\}$):

$$\begin{aligned} D &= \{(z_k, t_k) : k \in K\}, K=0:k_{\max}, \\ t_k &= t_{\min} + kh_t, h_t = \frac{t_{\max} - t_{\min}}{k_{\max}} \\ t_{\min} &= -1.0, t_{\max} = 2.5, z_k = x(t_k) + \varepsilon_k, \end{aligned} \quad (2)$$

where ε_k is a random error with zero mean and variance 0.1. The data distorted in this way are presented in Figure 1. Hereafter the following notation for intervals of possible values of t and $x(t)$ will be used:

$$T = [t_{\min}, t_{\max}], X = [-0.1, 2.2] \supset \left[\min_{k \in K} z_k, \max_{k \in K} z_k \right]. \quad (3)$$

Figure 1: Raw data of demo example.

Consider the problem of determining the equation of motion (1) by the method of balanced identification via given measurements (2). For that, we will compose several mathematical models and obtain corresponding CV-errors, which will be used as a criterion when comparing models.

It should be noted that all subsequent models of the oscillator produce variational optimization problems. Solving such problems directly is challenging because off-the-shelf optimization solvers (at least those, which are stable and “mature”) cannot handle integral and differential equations. To get a numerical solution, all these problems are replaced with finite dimensional mathematical programming problems. To do this the discretization of differential and integral equations is applied (see details in Section 4).

Model 0. Spline approximation of the function $x(t)$.

Take the simplest model as a twice differentiable function:

$$x(\cdot) \in C^2(T), \quad (4)$$

and consider the following optimization problem (w.r.t. a variable $x(\cdot)$), which depends on the regularization coefficient $\alpha \geq 0$

$$F(x, K, \alpha) \doteq \frac{1}{|K|} \sum_{k \in K} (z_k - x(t_k))^2 + \alpha \int_T \left(\frac{d^2 x}{dt^2} \right)^2 dt \rightarrow \min_{x \in C^2(T)}. \quad (5)$$

For any fixed α (see [4, 6]) problem (5) has a unique solution, which is a cubic spline for the given set of points. It is presented in Figure 2(c) and looks rather reasonable.

Figure 2(a) (linear function $x(t)$) corresponds to the case $\alpha \rightarrow +\infty$, when the second regularization term of objective function (5) suppresses the first one and the optimal solution tends to be a linear regression given by the *least squares method* (**too rough an approximation**). In the case of (2b), $\alpha \rightarrow 0$, we have a problem of spline interpolation: find a curve with minimal integral curvature passing exactly through given points (**obvious overfitting**).

It is required to find the “best” value of α at which the “smooth” model function passes “close enough” to the measurements, **giving optimal balanced approximation** and smoothing out random errors (as in Figure 2(c)). The choice of an optimal coefficient α can be made by minimizing the value of cross-validation error [3–5].

The simplest variant of the cross-validation (CV) procedure (*leave-one-out*) is to use a one-point test sample (z_k, t_k) and a training sample $D \setminus \{(z_k, t_k)\}$ (see definition of D in (2)) consisting of the remaining set of measurements $K \setminus k$. Formally, for a fixed α and every k : find a solution of the corresponding optimization problems (6), cal-

Figure 2: Three variants of data approximation by a function.

culate the approximation error on the test samples and get the average CV-error (which depends on α):

$$x_{K \setminus k}^\alpha \doteq \underset{x(\cdot)}{\operatorname{Argmin}} \{ F(x(\cdot), K \setminus k, \alpha) \}, k \in K, \quad (6)$$

$$\sigma(\alpha) \doteq \sqrt{\frac{1}{|K|} \sum_{k \in K} (z_k - x_{K \setminus k}^\alpha(t_k))^2}.$$

The best α^* minimises the value of CV-error,

$$\alpha^* \doteq \underset{\alpha \geq 0}{\operatorname{Argmin}} \sigma(\alpha). \quad (7)$$

So, α^* is a solution of the bilevel optimization problem: (7) at upper-level and $|K|$ independent problems (6) at lower level. The value $\sigma^* \doteq \sigma(\alpha^*)$ is called the *modeling CV-error*. It will be used as a quantitative measure of a model’s correspondence to the available experimental data:

$$\sigma^* \doteq \sqrt{\frac{1}{|K|} \sum_{k \in K} (z_k - x_{K \setminus k}^{\alpha^*}(t_k))^2}. \quad (8)$$

Finally, the α^* gives the desired model function $x^*(\cdot)$ and its *root-mean-square error* (for reference only):

$$x^* \doteq x_K^{\alpha^*} \doteq \underset{x(\cdot)}{\operatorname{Argmin}} \{ F(x(\cdot), K, \alpha^*) \}, \quad (9)$$

$$\operatorname{rmse}^* = \sqrt{\frac{1}{|K|} \sum_{k \in K} (z_k - x^*(t_k))^2}.$$

The function obtained for Model 0 is shown in Figure 3 (the same as in Figure 2(c))

Figure 3: Model 0. SVF-approximation is cubic spline approximation.

A cubic spline is an example of nonparametric regression, which does not explain the “physics” of the process presented on Figure 1. Because the object under study seems to be a dynamical one, the following models include differential equations with unknown parameters.

Model 1. First order differential equations.

Apply the SVF-method for the model with two unknown functions $x(t)$, $f(x)$ related by the following differential equation (see the definition of T , X in (3)):

$$\frac{dx}{dt} = f(x(t)), \quad x(\cdot) \in C^2(T), \quad f(\cdot) \in C^2(X), \quad t \in T. \quad (10)$$

In this case, it is reasonable to characterize complexity of the model by smoothness of the function $f(x)$, responsible for the dynamics. The main criterion (with regularization term) will be the following (hereafter we'll use the same notation $F(\cdot)$, but with another argument list):

$$F(x, f, K, \alpha) \doteq \frac{1}{|K|} \sum_{k \in K} (z_k - x(t_k))^2 + \alpha \int_X \left(\frac{d^2 f(x)}{dx^2} \right)^2 dx. \quad (11)$$

The “balanced” solution (x^*, f^*) for the dataset (2) corresponds to the α^* that minimizes the value of CV-error:

$$\begin{aligned} (x_{K \setminus k}^\alpha, f_{K \setminus k}^\alpha) &\doteq \operatorname{Argmin}_{x(\cdot), f(\cdot)} \left\{ F(x, f, K \setminus k, \alpha) : \frac{dx}{dt} = f(x(t)) \right\}, \\ \sigma(\alpha) &\doteq \sqrt{\frac{1}{|K|} \sum_{k \in K} (z_k - x_{K \setminus k}^\alpha(t_k))^2}, \\ \alpha^* &\doteq \operatorname{Argmin}_{\alpha \geq 0} \sigma(\alpha), \end{aligned} \quad (12)$$

and x^*, f^* are determined as follows:

$$(x^*, f^*) \doteq \operatorname{Argmin}_{x(\cdot), f(\cdot)} \left\{ F(x, f, K, \alpha^*) : \frac{dx}{dt} = f(x(t)) \right\}. \quad (13)$$

The root-mean-square error is calculated as in (9). See values of σ^* and rmse^* in the second row of Table 1 and the

Figure 4: Model 1. First order ODE: optimal $x^*(t)$ and raw data.

plot of function $x^*(t)$ in Figure 4. All of them are obviously worse than those given by the Model 0. The reason is that the set of solutions of Equation (10) is too narrow!

Replacing Model 0 with Model 1 (based on a solution of ODE (10) instead of an arbitrary smooth function) reduces the possibility of describing the data (2) by the function $x(t)$. It is easy to see that only a monotonic function $x(t)$ may be a solution of (10). But such functions can not approximate points in Figure 1 well. This explains why the CV-error σ^* and rmse^* in the second row of Table 1 are bigger than those in the first row. We conclude that Model 1 is worse than Model 0.

Model 2. Second order differential equations.

We now take a model with three unknown functions $x(t)$, $v(t)$ and $f(x, v)$ related by the second order ODE:

$$\begin{aligned} \frac{d^2 x}{dt^2} &= f(x(t), v(t)), \quad \frac{dx}{dt} = v(t), \quad V \doteq [-1., 1.5], \\ x(\cdot), v(\cdot) &\in C^2(T), \quad f(\cdot) \in C^2(G) \subset \mathbb{R}^2, \quad G = X \times V. \end{aligned} \quad (14)$$

The interval V is wide enough to hold all possible values of $\dot{x}(t)$.

Now, the regularization term of the SVF-criterion includes two coefficients α_x , α_v and partial derivatives of $f(x, v)$:

$$\begin{aligned} F(x, f, K, \alpha_x, \alpha_v) &\doteq \frac{1}{|K|} \sum_{k \in K} (z_k - x(t_k))^2 + \\ &+ \iint_G \left[\alpha_x^2 \left(\frac{\partial^2 f}{\partial x^2} \right)^2 + 2\alpha_x \alpha_v \left(\frac{\partial^2 f}{\partial x \partial v} \right)^2 + \alpha_v^2 \left(\frac{\partial^2 f}{\partial v^2} \right)^2 \right] dx dv. \end{aligned} \quad (15)$$

In this case, the SVF-method finds two optimal coefficients α_x^* and α_v^* minimizing the CV-error $\sigma(\alpha_x, \alpha_v)$, which is calculated similarly to Equations (12). Calculating solutions $x^*(t)$, $f^*(x, v)$, σ^* and rmse^* are similar to those, which were used for Models 0 and 1 (see Equations (13), (9)). The values σ^* , rmse^* are presented in Table 1 and isolines of the function $f^*(x, v)$ and trajectory $(x^*(t), \dot{x}^*(t))$ in Figure 5. The plot of function $x^*(t)$ is skipped because it is almost the same as in Figure 3 for Model 0.

So, we can draw the following conclusions:

1. The errors for Model 2 are almost the same as for Model 0, so the transition from an arbitrary smooth function to a solution of the ODE (14) does not reduce capabilities of describing raw data (2);
2. The level lines in Figure 5 are parallel and equidistant, so the graph of $f^*(x, v)$ is a plane, so $f^*(x, v)$ is an affine function (a linear one plus a constant).

Figure 5: Model 2. Isolines of $f^*(x, v)$ and trajectory $(x^*(t), \dot{x}^*(t))$.

Model 3. Oscillator with friction.

Affinity of $f(x, v)$ allows us to simplify the right hand side of ODE (14):

$$\frac{d^2x}{dt^2} = -k(x - \Delta x) - \mu \frac{dx}{dt}, \quad (16)$$

where k, μ and Δx are unknown parameters to be identified by fitting to the raw data (2).

The results of the SVF-method for Model 3 are presented in Table 1. You can see that the predictive error (σ^*) is smaller than in other cases. Nevertheless, the plot of function $x^*(t)$ is almost equivalent to that of Figure 3.

Table 1 illustrates the proposed methodology of comparing successive model changes. If the CV-error becomes smaller after the modification, then “we are on the right path”. So, Model 2 is better than Model 0, and Model 3 is better than Model 2. However, replacing of Model 0 with Model 1 was wrong.

Root-mean-square error (rmse^*) is another quantitative characteristic of the model. Ideally, rmse^* has to be equal to the data measurements error.

Thus, we can draw the conclusion that the Model 3 (oscillator with viscous friction) fits the available data best. Indeed, Formula (1), which was used to generate the data, is one of the solutions of the motion Equation (16) - a well-known model of the oscillator with friction.

Table 1: Cross-validation error (σ^*) and root-mean-square error (rmse^*) for different models (by oscillator's motion data).

No	Model	σ^*	rmse^*
0	A function. Spline-approximation	0.1144	0.0907
1	First order differential equation	0.2093	0.1869
2	Second order differential equation	0.1146	0.0924
3	Oscillator with friction	0.1106	0.0932

3 Balanced identification method

In the previous section, the SVF-method has been described on a model of a damped oscillator. Below is a general description of the method. It includes formalization of the basic concepts. These are as follows: the model itself; the data and model error; the main criterion (with regularization); the cross-validation scheme.

Mathematical model

A mathematical model is a set of statements (hypothesis) on the properties of the object under study. They are: equations and inequalities (e.g. $\dot{x}(t)=f(x)$, $\dot{x}(t)\geq 0$); logical expressions (e.g. $\exists t: x(t)=0$); definitions of sets and additional requirements on feasible domain of variables (e.g. $t\in T$, $G=X\times V$, $x(\cdot)\in C^2[G]$); etc.

We use a formal description of a mathematical model in the form of a system of equations:

$$M(x) = 0, \quad (17)$$

where each element of the vector M corresponds to one of the model's statements (hypothesis). For simplicity, the equal sign ($=$) is used in (17), but it can be substituted by inequalities ($\geq$), the adhesion sign ($\in$), logical conditions, etc. For example, a model (10) is written in the form of the following four statements:

$$\begin{aligned} \frac{dx}{dt} &= f(x(t)), \\ x(\cdot) &\in C^2(T), \\ f(\cdot) &\in C^2(X), \\ t &\in T. \end{aligned} \quad (18)$$

In many cases the model may be presented in the following form, as is customary in optimization literature:

$$x \in Q \doteq \{M(x) = 0, x \in S \subseteq X\}, \quad (19)$$

where x is a vector of variables, Q is a feasible set defined by a number of constraints and additional requirements:

- $M(\cdot)$ is a system of equations;
- X is a space of model variables (possibly a composition of Euclidean and functional spaces);
- S is a set defined by additional simple constraints, e.g. inequalities (like ≤ 0), logical expressions, integer value condition on variables x , etc.

The variable x may include both the functional variables (e.g. the ones describing the trajectory of a dynamic system in Section 2), and the desired model parameters (e.g. elastic and viscous friction coefficients in the same demo).

Data and model error

Usually, a set defined in (19) is too wide. As such, identifying the model means selecting the best variables $x \in Q$ corresponding to the measurements of the object under study. The set of measured data may be described as follows:

$$D = \{(z_k, P_k(x)) : k \in K\}, z_k \in \mathbb{R}^{n_z}, P_k(\cdot) : X \rightarrow \mathbb{R}^{n_z}, \quad (20)$$

where k is the index of the measurement from a finite set K , z_k is the vector of measured values, and $P_k(\cdot)$ is an operator expressing the measured values via model variables. The values of z_k may contain measurement errors $\varepsilon_k = \|z_k - P_k(x)\|$. Below, for simplicity, the source data set D is identified with the set K consisting of indices of pairs $(z_k, P_k(x))$.

Main criterion

Consider a problem of minimization of the balanced criterion (model simplicity vs. data fitting) that depends on x , K and a vector of regularization coefficients $\alpha \in \mathbb{R}^{n_\alpha}$:

$$F(x, K, \alpha) = \frac{1}{|K|} F_F(x, K) + F_S(\alpha, x) \rightarrow \min_{x \in Q}, \quad (21)$$

where $F_F(\cdot)$ is the part of the criterion responsible for the data fitting, and $F_S(\cdot)$ is the part responsible for model complexity (for smoothness as a rule).

The functional $F_F(x, K)$ is a measure of the model's approximation error. Usually, we consider an additive measure, i.e. if $K = K_1 \cup K_2$ and $K_1 \cap K_2 = \emptyset$, then $F_F(x, K) = F_F(x, K_1) + F_F(x, K_2)$. For definiteness, assume that $\frac{1}{|K|} F_F(\cdot)$ is a mean-square error:

$$F_F(x, K) = \sum_{k \in K} (z_k - P_k(x))^2. \quad (22)$$

The functional $F_S(\alpha, x)$ is a measure of the model's complexity (it is one of the possible interpretations of a Tikhonov regularization functional [2]). It depends on the desired solution, and on the vector of non-negative regularization coefficients $\alpha = \{\alpha_r\}$. Each component of α is the penalty coefficients for some aspect of complexity of the solution. It is presumed that the functional $F_S(\alpha, x)$ increases monotonically over any α_r , tends to zero as $\alpha_r \rightarrow 0$, and tends to $+\infty$ as $\alpha_r \rightarrow \infty$.

The functional of complexity is being chosen based on the specificity of the object's model (17). This functional can contain different characteristics of the curvature of functions constituting the model and/or some integral characteristics, such as energy or entropy. For instance, in the demo example in Section 2, for a single-variable function $f(x)$ (see Model 1), $F_S(\alpha, f) = \alpha \int_X \left(\frac{d^2 f(x)}{dx^2} \right)^2 dx$, and for two-variable function $f(x, v)$ (see Model 2),

$$F_S(\alpha, f) = \iint_{X \times V} \left[\alpha_x^2 \left(\frac{\partial^2 f}{\partial x^2} \right)^2 + 2\alpha_x \alpha_v \left(\frac{\partial^2 f}{\partial x \partial v} \right)^2 + \alpha_v^2 \left(\frac{\partial^2 f}{\partial v^2} \right)^2 \right] dx dv.$$

For a fixed α the problem of identification for a given set of measurements K is to find the value x_K^α that minimizes the functional (21) on a feasible set (19):

$$x_K^\alpha = \underset{x}{\operatorname{Argmin}} \{F(x, K, \alpha) : x \in Q\}.$$

It is required to find a balanced solution, i.e. a value α^* that provides a reasonable compromise between the model error and its simplicity.

Cross-validation procedure

To find α^* , a cross-validation procedure is used [3]. The dataset K is subdivided into disjoint subsets K_i of statistically independent measurements:

$$K = \bigcup_{i \in I} K_i, K_i \cap K_j = \emptyset, i \neq j \quad (23)$$

(if measurement errors are random, then any non-intersecting sets will be independent).

Next, we exclude some subset K_i from the set K , and find the minimum (21) for a given value α and for the remaining set $K \setminus K_i$ (training set). Let $x_{K \setminus K_i}^\alpha$ be a solution,

$$x_{K \setminus K_i}^\alpha = \underset{x}{\operatorname{Argmin}} \{F(x, K \setminus K_i, \alpha) : x \in Q\}, \quad (24)$$

and the value $F_F(x_{K \setminus K_i}^\alpha, K_i)$ is approximation error on the test set K_i . By repeating this procedure for all subsets K_i (where $i \in I$) and by summing up the results, we obtain a cross-validation error for a given α ,

$$\sigma(\alpha) = \sum_{i \in I} F_F(x_{K \setminus K_i}^\alpha, K_i). \quad (25)$$

The desired optimal weight coefficients and the corresponding solution x^* (for a whole set of measurements K) are defined as follows:

$$\begin{aligned} \alpha^* &= \underset{\alpha \geq 0}{\operatorname{Argmin}} \sigma(\alpha), \\ x^* &= x_K^{\alpha^*} = \underset{x}{\operatorname{Argmin}} \{F(x, K, \alpha^*) : x \in Q\}. \end{aligned} \quad (26)$$

Therefore, the search for optimal coefficients α for a given dataset (20) is a *bilevel optimization problem* [10] with $|I|$ independent optimization problems (24) (these problems may be the variational ones) in the lower level to calculate the value of cross-validation error $\sigma(\alpha)$. In the upper-level, we find the α^* by minimizing the $\sigma(\alpha)$, (26).

In the case when the model error $F_F(\cdot)$ is the standard deviation (22) we have a *mean (modeling) cross-validation error* (σ^*) and a *root-mean-squared error* (rmse^*):

$$\begin{aligned} \sigma^* &= \sqrt{\frac{1}{|K|} \sum_{i \in I} \sum_{k \in K_i} [z_k - P_k(x_{K \setminus K_i}^{\alpha^*})]^2}, \\ \operatorname{rmse}^* &= \sqrt{\frac{1}{|K|} \sum_{k \in K} [z_k - P_k(x^*)]^2}. \end{aligned} \quad (27)$$

The practical applicability of the SVF-method depends on the complexity of the model of the object under study, on the number of measurements and their quality. Generally speaking, we can get a variational problem if the model variables are functions of a continuous argument. In this case, discretization is applied, e.g., when a continuous domain is replaced with a finite set of points with unknown values of the desired function at these points. Another discretization method is to replace the desired function with polynomials with unknown coefficients. As a result, all optimization problems become finite-dimensional mathematical programming ones. To solve them one can effectively use available solvers and high-performance computing environments.

4 SVF-technology as software implementation of the SVF-method

The method of balanced identification has been described above in a rather abstract manner. In this section the main details of its software implementation are presented:

- the capabilities of symbolic notation of the model's equations and of the method options;
- discretization of continuous variables, differential and integral equations;
- surrogate optimization method (Subsection 4.5) for solving bilevel optimization problem (26);
- usage of Python Pyomo package and Everest optimization service.

4.1 SVF-technology block diagram

The general scheme of “human-computer” technology that implements the balanced identification method is shown in Figure 6. Each block of the block diagram has a number in a circle. At the user level, an expert (block 1), who has an idea of the object under study, should prepare a measurement file and a task-file (block 2). The data-file then contains a table with experimental data (in text format or in MS Excel or MS Access formats).

A text task-file usually contains the name of the data-file, a mathematical description of the object (model), a list of unknown parameters as well as cross-validation specifications etc (Figure 7). These files are transferred to the client program (block 3). Here the variational problems are replaced by discretization with NLP (finite dimen-

Figure 6: SVF-technology workflow.

sional Non-Linear mathematical programming Problem), the sets for CV are generated and the Pyomo NLP Model is formalized. The constructed data structures are transmitted to the surrogate optimization subroutine (block 4), which implements an iterative numerical search for unknown model parameters and regularization coefficients to minimize the cross-validation error. This subroutine includes parallel solving of mathematical programming problems in the distributed environment of Everest optimization services. The Pyomo package converts the NLP description in the so-called NL-files (block 5), which at the server level are processed by solvers (block 7) under the control of the service for solving optimization problems (block 6). The solutions obtained by solvers (block 8) are collected (block 9) and sent back to the client level (block 4), where they are analyzed. E.g. the conditions for the completion of the iterative process are checked here.

If the conditions are not met, a surrogate optimization algorithm calculates new values of regularization coefficients and the process repeats. Otherwise, the program prepares the results (block 10), calculates the errors, writes the solution files, draws the graphs of the functions found (block 11), and presents them to the researcher (block 1). The obtained results, especially the values of the model-

ing errors, are used by experts as an argument to choose a new (or modified) model, or to stop calculations.

4.2 Symbolic notation of a model and identification problem (task-file)

Figure 7 shows, as an example, the task-file with the oscillator model (**Model 2**) description and parameters of the numerical method under discussion.

```
CVNumOfIter    20
CVstep         21
RunSolver      ServerParallel
Select x, t    from ../DATA/Spring5.dat
GRID:   t ∈ [ -1.,  2.5, 0.025 ]
        X ∈ [ -0.1, 2.2, 0.1   ]
        V ∈ [ -1,   1.5, 0.1   ]
VAR: x ( t )
      v ( t )
      f ( X, V ); PolyPow = 6
EQ:  d2/dt2(x(t)) == f(x(t),v(t));
      v(t) == d/dt(x(t));
OBJ: x.MSD() + f.Complexity(Penal[0],Penal[1])
Draw
EOF
```

Figure 7: Task-file for oscillator (Model 2).

The first three lines define the control parameters: “CVNumOfIter” is the maximum number (20) of iterations (of a surrogate optimization procedure), “CVstep” is the number (21) of subsets into which the data set is split for the cross-validation procedure, “RunSolver” specifies the execution mode of calculations. “ServerParallel” option means that calculations performed on the remote computing resources, and independent problems may be solved in parallel.

The next line defines a data file and the names of the columns to be read (“x” and “t”). The file contains the dataset defined in (2).

The rest of the task-file contains the “translation” of the optimization problem (14), (15) into the language of the task-file notation. In the “GRID:” section the three sets (and a discretization parameter) are specified. For example, the expression “ $t \in [-1., 2.5, 0.025]$ ” defines a grid from -1. to 2.5 with step 0.025.

The keyword “VAR:” at the beginning of a line starts a block describing the variables (parameters to be sought). The grid function “ $x(t)$ ” is defined on the grid “t”

and is linked by the names (“x” and “t”) with measurements from the file “../DATA/Spring5.dat”. A function “ $f(X,V)$ ” is defined as a sixth degree polynomial “PolyPow=6” (with unknown coefficients). In this case, the sets “x”, “v” are used for numerical integration.

In the next block (the keyword “EQ:”), two equations (14) of Model 2 are defined on the grid “t”.

Finally, the key word “OBJ:” defines the objective function consisting of two parts. The first is the data approximation error for the trajectory “ $x(t)$ ” (the first term in (15)). The second summand is the regularization term in (15) - a “measure of the complexity” of the two variables function “ $f(X,V)$ ”. The “Complexity” function calculates the regularization penalty for the “Penal” parameters. The function can be explicitly specified in the notation. For example, the first part of the integral is “ $\text{Penal}[0]**2 * \int d(X) * \int (d(X) * (d2/dX2(f(X,V)))**2)$ ”. The last two lines contain commands to draw the results (functions) and complete the calculations.

Comparing the task-file with the formal mathematical description (**Model 2**), we note the clarity and convenience of the proposed notation.

4.3 Implementation of discretization

To obtain an approximate solution, the continuous (infinite-dimensional, variational) identification problem is transformed by discretization into a finite-dimensional mathematical programming problem. The SVF-technology implements two methods of discretization: 1) replacing intervals of continuous variables with finite grid of points; 2) approximation of desired function with polynomial with unknown coefficients. Step in grid and/or polynomial degree are parameters of the discretization. So, continuous functions are replaced by a grid functions, or by polynomials. Integrals are replaced by finite sums, and differential equations - by sets of algebraic equations.

Relatively simple schemes are used. For first derivatives, the trapezoidal integration and two-point schemes “forward/central/backward” are used. For the second derivative, the three-point scheme “central” is used.

The problem is reformulated as a Pyomo model and is written into a special Python-file. A qualified user can always modify it, for example to use another discretization scheme. However, there remains the problem of evaluating the proximity of the found numerical solution (based on finite-dimensional mathematical programming problems) to the solution of the original variational problem (in functional space).

4.4 Implementation by Python & Pyomo

In the last decade Python became one of the most used programming languages in scientific computing in different areas of applied mathematics, and in optimization modeling too. Namely, Python enables to interact with AMPL-compatible solvers.

The correct programming of optimization problems (e.g. a finite-dimensional analogue of (24)) as an input data for a solver via its lower-level API may be very time consuming, especially for complex nonlinear problems. The situation becomes more cumbersome when solvers are used in a distributed computing environment. Fortunately, the state-of-the-art solvers support so called AMPL (A Modeling Language for Mathematical Programming) [16]. Our research team has used AMPL in our studies on optimization modeling for about ten years. In particular, AMPL usage enabled us to develop the distributed system of optimization services on the base of the Everest toolkit [13].

Originally, the usage of AMPL required a commercially licensed translator to generate a special NL-file containing all data of the optimization problem to be solved. This NL-file may be passed to an AMPL-compatible solver. On success, the solver returns a solution as a SOL-file, which may be read by an AMPL-translator to get the values of variables (including the dual ones). An open source alternative to an AMPL-translator appeared in 2012 as a Pyomo package [17]. Pyomo enabled the production of an AMPL-script “lifecycle” but without an AMPL-translator. A description of optimization problems, generation of NL-files and reading of SOL-files all may be done in any Python application.

```
def EQ0 (Gr,t) :
    return (
        ((x((t+0.025))+x((t-0.025))-2*x(t)) \
         /0.025**2)==f(x(t),v(t))
    )
Gr.conEQ0 = Constraint( \
    myrange(-1.0+0.025,2.5-0.025,0.025), \
    rule=EQ0 )
```

Figure 8: The first differential equation (15) as a part of Pyomo model.

It is well known that Python is not intended for intensive computing. Rather, it is a scripting language providing data exchange between different high-performance software. As to Pyomo, the bottleneck may be in the preparation of an NL-file. In situations where a delay becomes prominent, such as for big optimization problems with

dozens of thousands of variables and constraints, there are a number of tricks to reduce the elapsed time. The most difficult computations related to solving problems presented by NL-files are performed by AMPL-compatible solvers written in C, C++ or Fortran.

One of the results of the task-file processing is a special software module that reformulates the optimization problem into Python language using the Pyomo optimization modeling package. The fragment of an example of auto-generated Python code corresponding to the first differential equation (15) is shown in Figure 8.

4.5 Surrogate optimization in SVF-technology

Bilevel optimization is a well-known and difficult challenge for researchers in numerical methods [10]. A current implementation of SVF is based on an approach similar to that of *surrogate optimization* [18], i.e. upper-level objective function is approximated by its evaluation at a few points. A corresponding set of points is generated successively until some stopping condition is satisfied.

The SVF-algorithm has two essential features:

- 1) the computational cost of each iteration is rather high (it needs to solve a set of independent NLP problems);
- 2) the resulting CV-error estimate may be distorted because of an inexact solution of the above NLP-problems.

A special surrogate optimization procedure that takes into account these features has been developed. The lower-level problem is treated as a “black-box” that calculates an approximate value of $\sigma(\alpha)$ for the input vector α .

So, we are looking for an approximate minimum of the function $\sigma(\alpha)$ (25) - an optimal value function of the upper-level problem (block 4 in Figure 6). The value of $\sigma(\alpha)$ depends on solutions of $|I|$ independent optimization problems (24) for a given α (blocks 5-9 in Figure 6). The method is based on successive approximations of $\sigma(\alpha)$ by second order polynomial of α on a sequence $\{\alpha^v\}$ ($v=1, 2, \dots$). The coefficients of this polynomial are recalculated to get better approximations of the values $\sigma(\alpha^v)$ for already “passed” points α^v . Namely, the function $\sigma(\alpha)$ ($\alpha \in \mathbb{R}^{n_\alpha}$) is approximated by the following multivariate polynomial:

$$\Pi(\pi, \alpha) = \sum_{1 \leq i \leq j \leq n_\alpha} \pi_{ij} \alpha_i \alpha_j + \sum_{1 \leq i \leq n_\alpha} \pi_i \alpha_i + \pi_0, \text{ where} \quad (28)$$

$$\pi = (\pi_{ij} (1 \leq i \leq j \leq n_\alpha), \pi_i (i=1:n_\alpha), \pi_0) \in \mathbb{R}^{\frac{N(N+3)}{2}+1},$$

where vector π changes at runtime of the algorithm.

The algorithm builds a sequence of points $\{\alpha^v\}$ ($v=1, 2, \dots$) as following. Assume that at step N we have

a set of pairs $\{(\sigma^v, \alpha^v)\}_{v=1}^N$, $(\sigma^v = \sigma(\alpha^v))$.¹ In what follows these pairs will be treated as a set of points in $\mathbb{R}^{n_{\alpha}+1}$. Without loss of generality (maybe after renumbering) assume that $\sigma^N = \min_{v \in 1:N} \sigma(\alpha^v)$.

Consider a problem of approximating these points by a polynomial $\Pi(\pi, \alpha)$ (28). The smaller the distance between α^v and the “best” α^N the more weight the error $|\sigma^v - \Pi(\pi, \alpha^v)|$ will have in the following penalty function with regularization term (sum of all squared π_{ij} from (28) weighted with coefficient μ):

$$\sum_{v=1}^N \frac{(\sigma^v - \Pi(\pi, \alpha^v))^2}{e^{\|\alpha^v - \alpha^N\|}} + \mu \left(\sum_{i \leq j} \pi_{ij}^2 \right) \rightarrow \min_{\pi}. \quad (29)$$

This approximation does not pass exactly through the points $\{(\sigma^v, \alpha^v)\}_{v=1}^N$ but it is close to those points, for which σ^v is close to the minimum of $\{\sigma^v\}_{v=1}^N$.

It is easy to see that the objective function (29) is strongly convex in variables π and reaches a unique minimum at $\pi^*(\mu)$. The algorithm searches for a μ^* that minimizes the error of the $\sigma(\alpha)$ approximation (by the polynomial $\Pi(\cdot)$) at the “best” vector α^N . Another bilevel optimization problem arises:

$$\left| \Pi(\pi^*(\mu), \alpha^N) - \sigma^N \right| \rightarrow \min_{\mu \geq 0}, \quad (30)$$

where $\pi^*(\mu)$ is a solution of the lower level problem (29).

Note that in the upper-level problem we are looking for an optimal scalar value μ^* and the lower level problem is effectively solvable (as the strongly convex one). Thus, the optimal solution μ^* may be found by some of the known *line search* algorithms, e.g. “bisection method”.

After polynomial approximation $\Pi(\pi^*(\mu^*), \alpha)$ is determined, the next vector α^{N+1} is found as a solution of the following auxiliary problem, similar to the one used in the *Pshenichny-Danilin linearisation method* [11]:

$$\left(\nabla_{\alpha} \Pi(\pi^*(\mu), \alpha^N) \right)^T (\alpha - \alpha^N) + \frac{1}{2} \|\alpha - \alpha^N\|^2 \rightarrow \min_{\alpha \geq 0}, \quad (31)$$

and where $\nabla_{\alpha} \Pi(\cdot)$ is a gradient of $\Pi(\pi, \alpha)$ with respect to vector α , and where $x^T y$ means the scalar product of the vectors x and y . Let α^{N+1} be a solution of this problem (it exists and is unique).

At the next step of the algorithm we calculate the value $\sigma^{N+1} = \sigma(\alpha^{N+1})$ as a result of solving the set of problems (24). If the difference $|\sigma^{N+1} - \sigma^N|$ is less than some predefined small threshold, then the algorithm stops and returns α^N

(if $\sigma^{N+1} \geq \sigma^N$) or α^{N+1} (if $\sigma^{N+1} < \sigma^N$). If the above difference in $\sigma(\alpha)$ is higher than a predefined threshold, then the previous step of the algorithm is repeated on the extended set of pairs $\{(\sigma^v, \alpha^v)\}_{v=1}^{N+1}$.

4.6 Solvers and Everest optimization service

As a basic tool for solving independent mathematical programming problems in parallel (blocks 6-8 in Figure 6) we use the Everest optimization service [13, 14]. There are a several of services available at the URL <https://optmod.distcomp.org>. All of them are based on the Everest toolkit [15]. Each service, an Everest-application in “Everest-terminology”, is a front-end of a pool of solvers deployed and running on a number of various computing resources (desktops, standalone servers, clusters). This computing environment is extensible because the potential users can add their own computing resources to the optimization environment via the open source Everest agent <http://everest.distcomp.org/docs>.

Previous implementation of the SVF-technology is based on a *solve-ampl-stub*² Everest-application. This service solves mathematical programming problems presented by their NL-files [16] and returns SOL-files with the solutions found. The service provides unified access to the following solvers “covering” the main types of mathematical programming problems (LP/MILP/NLP/MINLP):

- Ipopt (Coin-OR Interior Point Optimizer, NLP), <https://github.com/coin-or/Ipopt>, [19];
- CBC (Coin-OR Branch-and-Cut, LP, MILP), <https://github.com/coin-or/Cbc>;
- SCIP (Solving Constraint Integer Programs, LP, MILP, MINLP), <http://scip.zib.de>, [20];
- Bonmin (COIN-OR Basic Open-source Nonlinear (convex) Mixed Integer programming, MINLP), <https://github.com/coin-or/Bonmin>.

Note that CBC and Bonmin seems to be a bit outdated now. All of the results presented above, in Section 2, and below, in Section 5, are obtained with solver Ipopt. Currently we are beginning to use a SCIP global-optimization solver. We do it carefully, because the search for global optimum usually takes much more time than the search of local optimum that Ipopt does.

Besides, we are working on an implementation of an analogue of the so-called AMPLX system [13] (an extension of AMPL-scripting to exchange data with Everest opti-

¹ At the initial stage, the new values of the vector α are obtained by small perturbations of all its components

² <https://optmod.distcomp.org/apps/vladimirv/solve-ampl-stub>

mization services) on the base of the so-called PyomoEverest, <https://github.com/distcomp/pyomo-everest>. It provides developers with simple Python API to exchange data between Pyomo optimization models and Everest optimization services. As a result, the current implementation of the SvF-technology is based on another Everest application SSOP³ (Solve Set of Optimization Problems), which solves a batch of independent problems in parallel by using the available computing resources. Everest provides the capabilities to distribute all these problems (may be hundreds in practice) among available resources. Currently SSOP supports two solvers: IPOPT if it is enough to find a local optimum in optimization problems; SCIP if a global optimum is required.

5 Successful use cases

The technology has been used successfully in different application areas listed below.

Plant physiology [21]. The dataset (about 1600 points in time, step - 30 min) consist of transpiration and climatic (photoactive radiation, humidity and temperature) measurements. Among a number of (about 10) models of different complexity, a dynamic model of the water in the plant was chosen.

Ecology [22]. The method of balanced identification was used to describe the response of Net Ecosystem Exchange of CO₂ (NEE) to a change of environmental factors, and to fill the gaps in continuous CO₂ flux measurements in a sphagnum peat bog in the Tver region. The dataset is about 2000 points (step - 30 min). The selected model (the dependence of NEE on climatic parameters) was a superposition of one- and two-variable functions.

Meteorology [23]. The model of radon dynamics in the atmosphere was built. The dataset (about one month, about 3000 points, step 15 min) contains measurements of characteristics of the γ -radiation field by a gamma spectrometer. The model, consisting of two ODEs describes the dynamics of the volume activity of ²²²Rn and its daughter products ²¹⁴Pb and ²¹⁴Bi.

Pollution, geography, soil erosion [24]. The datasets (from 100 to 1000 points by X and Y coordinates, step - about 100 meters), consist of altitude (SRTM satellite data) and ¹³⁷Cs decay activity (airborne gamma data) measurements. A number of models were considered (functional

dependencies), including digital terrain, soil erosion and ¹³⁷Cs contamination models.

Air pollution [25]. The problem is to create a mathematical model of air pollution on the base of a 2D diffusion-transport model. The following problems were studied (see poster at <https://www.researchgate.net/publication/334523588>):

F) The estimation of distribution and sources of the atmospheric aerosol pollution (PM10) in the North of France. The dataset (about 30000 points) consists of twelve stations of the Atmo Hauts-de-France pollution measurement network.

R) The evaluation of pollutant emissions for contamination sources with known positions. Measurements of the pollution of the soil and a lake's water by metals (Cu, Ni, etc.) were obtained in the Kola Peninsula in Russia at about 100 stations.

Medicine [26]. Different models of glucose and insulin dynamics in human blood are analyzed. The datasets (about 20 measurements of glucose concentration and insulin) contain intra venous glucose tolerance test results.

Thermonuclear plasma [27, 28]. We compared several models (integro-differential, partial derivatives) describing the initial stage of the fast nonlocal transport events, which exhibit immediate response, in the diffusion time scale, of the spatial profile of electron temperature to its local perturbation, while the net heat flux is directed opposite to ordinary diffusion (i.e. along the temperature gradient). The results of experiments on a few thermonuclear installations were used. The datasets contain about 100 points.

Evolutionary theory [29]. The optimization problem, which reflects the mechanisms of regulation of the speed of evolution that provide an adequate population response to the direction and rate of environmental change, was considered. The numerical experiment results show plausible dependences of age-specific fertility on the rate of environmental changes and also describe and explain a number of evolutionary effects.

6 Discussion

The presented SVF-method and its software implementations are an interesting combination of knowledge from the areas of data analysis, optimization and distributed computing software including: cross-validation and regularization methods, algebraic modeling in optimization and methods of optimization, automatic discretization

³ <https://optmod.distcomp.org/apps/vladimirv/solve-set-opt-probs>

of differential and integral equation, optimization REST-services, etc.

Current implementation is based on state-of-the-art open source NLP-solvers (AMPL-compatible): Ipopt [19] and SCIP [20]. Optionally, optimization services [13] of extensible computing environment <https://optmod.distcomp.org> based on the Everest toolkit [15] may be used as well.

The implementation supports symbolic notation to describe the model and corresponding identification problem. This feature automates the formulation of optimization problems with less user effort. A special module generates fragments of Python codes based on a special discretization subsystem and Pyomo framework to interact with open source AMPL-compatible solvers.

A list of success stories of the presented approach is already rather long. Nevertheless, the authors realize its domain of applicability and important unresolved issues.

First of all, the method does not provide a fully automated “model generation and/or selection”. SVF-technology enables quantitative comparison within a given set of models, but it is the researcher who makes modifications to the model (e.g. change the model constraint equations).

The second issue concerns the search for a global optimum in problems (24), which is required for cross-validation. In the general case, these problems may have a lot of local extrema. Moreover, they may even contain integer variables. If so, the NLP-solver Ipopt cannot guarantee a global optimum. It is possible to use the SCIP solver implementing a branch-and-bound algorithm for mixed-integer nonlinear mathematical programming problems, but at the cost of increased solving time (and more memory consumption). Although parallel extensions of SCIP are available [30, 31] simple workarounds are used now. They are: Ipopt multi starts options; some easily verifiable necessary conditions of the global optimum. In current practice some “informal” analysis of the SVF-algorithm convergence with respect to decreasing of modeling cross-validation error is applied. In the near future we plan to perform a computational experiments where a global optimizer SCIP and its parallel extensions FiberSCIP or ParaSCIP will be used alongside Ipopt, e.g. one call of SCIP per ten calls of Ipopt.

Discretization is the third important challenge for the SVF-method. The presence of ordinary and partially differential equations, along with integral ones, is typical for models in physics, biology, medicine etc. Current implementation is based on a rather simple discretization module that implements two basic schemes: finite-difference and approximation by polynomials with unknown coefficients.

We plan to analyse capabilities of a so-called DAE subsystem of Pyomo [32] for the SVF-technology. Recall that the SVF-technology already uses the Pyomo package for programming of optimization problems and for data exchange with AMPL-compatible solvers.

One of the issues related to discretization concerns the case when the model includes composition of unknown functions $f(y)$, $x(t)$, e.g. the Model 1 (10) includes the differential equation $\dot{x} = f(x(t))$ with unknown functions $f(y)$, $x(t)$. Current implementation of SVF uses the following approach: the inner function $x(t)$ is approximated by a grid function and the outer $f(y)$ is approximated by a polynomial; that is,

$$\begin{aligned} x(t) &\sim \{x_i, t_i\} : i = 0:N_x, \text{ where } x(t_i) = x_i, \\ f(y) &\sim \sum_{k=0}^K p_k \cdot y^k. \end{aligned} \quad (32)$$

Here, the unknown variables are $\{x_i : i=0:N_x\}$ and $\{p_k, k=1:K\}$. After that, the discretization of the differential equation (10) looks like a number of equations:

$$\frac{x(t_i) - x(t_{i-1})}{\Delta_i t} = \sum_{k=0}^K p_k \cdot x(t_i)^k. \quad (33)$$

We plan to use another approach based on a smoothed piecewise approximation (e.g. see [33, 34]) of the outer function approximated by a grid as well.

Another issue concerns the domain of applicability of discretization itself, e.g. for the models based on optimal control problems. Here the applicability of discretization depends on whether the corresponding problem has any special properties, such as “chattering phenomenon” for an optimal solution. The well-known Fuller problem [35] is one of the simplest demonstrations of this effect. At first glance, it is a very simple optimal control problem on a finite time interval, but for some segment of initial conditions any optimal solution cannot be well approximated by any finite discretization of time interval.

Acknowledgement: The results of Section 4 were obtained within the Russian Science Foundation grant (project No. 16-11-10352). The results of Sections 1-3, 5 were obtained within the Russian Foundation for Basic Research grant (project No. 20-07-00701).

References

- [1] A.V. Sokolov and V.V. Voloshinov. Choice of mathematical model: balance between complexity and proximity to measurements.

- International Journal of Open Information Technologies*, 6(9), 2018.
- [2] A.N. Tikhonov. On mathematical methods for automating the processing of observations. In *Problems of Computational Mathematics*, pages 3–17, 1980.
 - [3] R. Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.
 - [4] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009.
 - [5] M. Kuhn and K. Johnson. *Applied predictive modeling*, volume 26. Springer, 2013.
 - [6] A.I. Rozhenko. *Theory and Algorithms of Variational Spline-Approximations*. Novosibirsk State Technical University, 2005. (in Russian).
 - [7] W. Härdle. *Applied nonparametric regression*. Number 19. Cambridge university press, 1990.
 - [8] V. Strijov and G.-W. Weber. Nonlinear regression model generation using hyperparameter optimization. *Computers & Mathematics with Applications*, 60(4):981–988, 2010.
 - [9] O. Sysoev and O. Burdakov. A smoothed monotonic regression via L2 regularization. *Knowledge and Information Systems*, 59(1):197–218, 2019.
 - [10] S. Dempe. *Foundations of bilevel programming*. Springer Science & Business Media, 2002.
 - [11] B.N. Pshenichnyi and A.A. Sosnovsky. The linearization method: Principal concepts and perspective directions. *Journal of Global Optimization*, 3(4):483–500, 1993.
 - [12] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.
 - [13] S. Smirnov, V. Voloshinov, and O. Sukhoroslov. Distributed optimization on the base of AMPL modeling language and Everest platform. *Procedia Computer Science*, 101:313–322, 2016.
 - [14] S. Smirnov and V. Voloshinov. On domain decomposition strategies to parallelize branch-and-bound method for global optimization in Everest distributed environment. *Procedia Computer Science*, 136:128–135, 2018.
 - [15] O. Sukhoroslov, S. Volkov, and A. Afanasiev. A web-based platform for publication and distributed execution of computing applications. In *Parallel and Distributed Computing (ISPD), 2015 14th International Symposium on*, pages 175–184, June 2015.
 - [16] R. Fourer, D.M. Gay, and B.W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming. Second edition*. Duxbury Press/Brooks/Cole Publishing Company, 2003. <https://ampl.com/resources/the-ampl-book>.
 - [17] W.E. Hart, C.D. Laird, J.P. Watson, D.L. Woodruff, G.A. Hackebeil, B.L. Nicholson, and J.D. Siirola. *Pyomo—optimization modeling in Python. 2nd edition*, volume 67. Springer, 2017.
 - [18] A. Forrester, A. Sobester, and A. Keane. *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.
 - [19] A. Wächter and L.T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.
 - [20] A. Gleixner, M. Bastubbe, L. Eifler, T. Gally, G. Gamrath, R. L. Gottwald, G. Hendel, C. Hojny, T. Koch, M. E. Lübbecke, S. J. Maher, M. Miltenberger, et al. The SCIP Optimization Suite 6.0. Technical Report 18-26, ZIB, Takustr. 7, 14195 Berlin, 2018.
 - [21] A.V. Sokolov, V.K. Bolondinsky, and V.V. Voloshinov. Technologies for constructing mathematical models from experimental data: applying the method of balanced identification using the example of choosing a pine transpiration model. In *National Supercomputer Forum (NSCF-2018)*, 2018.
 - [22] A.V. Sokolov, V.V. Mamkin, V.K. Avilov, D.L. Tarasov, Y.A. Kurbatova, and A. V. Olchev. Application of a balanced identification method for gap-filling in CO₂ flux data in a sphagnum peat bog. *Computer Research and Modeling*, 11(1):153–171, 2019.
 - [23] Yu.E. Lavruhin, A.V. Sokolov, and D.S. Grozdov. Monitoring of volume activity in the atmospheric surface layer based on the testimony of the spectrometer seg-017: error analysis. In *Radioactivity after nuclear explosions and accidents: consequences and ways to overcome*, pages 359–368, 2016.
 - [24] V.G. Linnik, A.V. Sokolov, and I.V. Mironenko. 137cs patterns and their transformation in landscapes of the opolye of the bryansk region. *Modern trends in the development of biogeochemistry*, pages 423–434, 2016.
 - [25] A.V. Sokolov, A.A. Sokolov, and Hervé Delbarre. Method of balanced identification in the inverse problem of transport and diffusion of atmospheric pollution. In *EGU2019-15175*, volume 26, 2019.
 - [26] A.V. Sokolov and L.A. Sokolova. Building mathematical models: quantifying the significance of accepted hypotheses and used data. In *XXI International Conference on Computational Mechanics and Modern Applied Software Systems (CMMASS'2019)*, pages 114–115, 2019.
 - [27] A.P. Afanasiev, V.V. Voloshinov, and A.V. Sokolov. Inverse problem in the modeling on the basis of regularization and distributed computing in the Everest environment. In *CEUR Workshop Proceedings*, pages 100–108, 2017.
 - [28] A.B. Kukushkin, A.A. Kulichenko, P.A. Sdvizhenskii, A.V. Sokolov, and V.V. Voloshinov. A model of recovering parameters of fast non-local heat transport in magnetic fusion plasma. *Problems of Atomic Science and Technology, Ser. Thermonuclear Fusion*, 40(1):45–55, 2017.
 - [29] A.V. Sokolov. Mechanisms of regulation of the speed of evolution: The population level. *Biophysics*, 61(3):513–520, 2016.
 - [30] Y. Shinano, T. Achterberg, T. Berthold, S. Heinz, and T. Koch. ParaSCIP: a parallel extension of SCIP. In *Competence in High Performance Computing 2010*, pages 135–148. Springer, 2011.
 - [31] Y. Shinano, S. Heinz, S. Vigerske, and M. Winkler. FiberSCIP – a shared memory parallelization of SCIP. *INFORMS Journal on Computing*, 30(1):11–30, 2017.
 - [32] B. Nicholson, J.D. Siirola, J.-P. Watson, V.M. Zavala, and L.T. Biegler. pyomo.dae: a modeling and automatic discretization framework for optimization with differential and algebraic equations. *Mathematical Programming Computation*, 10(2):187–223, 2018.
 - [33] C. Chen and O.L. Mangasarian. A class of smoothing functions for nonlinear and mixed complementarity problems. *Computational Optimization and Applications*, 5(2):97–138, 1996.
 - [34] Z. Zhou and Y. Peng. The locally Chen–Harker–Kanzow–Smale smoothing functions for mixed complementarity problems. *Journal of Global Optimization*, 74(1):169–193, 2019.
 - [35] A.T. Fuller. Relay control systems optimized for various performance criteria. volume 1, pages 520–529. Elsevier, 1960.