Вестник КРАУНЦ. Физ.-мат. науки. 2024. Т. 49. №4. С. 135-156. ISSN 2079-6641

ИНФОРМАЦИОННЫЕ И ВЫЧИСЛИТЕЛЬНЫЕ ТЕХНОЛОГИИ

https://doi.org/10.26117/2079-6641-2024-49-4-135-156

Научная статья

на Русском языке

УДК 519.6; 517.972.7; 519.642.2

Некоторые аспекты реализации программного комплекса PRPHMM 1.0 для уточнения параметров эредитарных математических моделей переноса радона в накопительной камере

Д. А. Твёрдый*[1], Е. О. Макаров[2]

[1] Институт космофизических исследований и распространения радиоволн ДВО РАН, 684034, с. Паратунка, ул.Мирная, д. 7, Россия

[2] Камчатский филиал Федерального исследовательского центра «Единая геофизическая служба РАН», 683023, г. Петропавловск-Камчатский, ул. бульвар Пийпа, д.9, Россия

Аннотация. Математические модели некоторых динамических процессов можно существенно уточнить, используя в них производные и интегралы нецелого порядка, учитывая эффекты, которые не описать с помощью обыкновенных производных. Так, например, с помощью дробных производных Герасимова-Капуто постоянного и переменного порядка можно учитывать эффект памяти в модели процесса, а порядок производной будет связан с интенсивностью процесса. В частности, авторами ранее разработана эредитарная α-модель объемной активности радона, где параметр α связан с проницаемостью среды. Однако возникает вопрос об определении оптимальных значений как α, так и других параметров модели. Для решения проблемы можно решать обратную задачу — распространенный тип задач во многих научных областях, где необходимо определить значения параметров модели на основе наблюдаемых данных, но невозможно провести прямые измерения этих параметров. Необходимость такого подхода часто возникает при работе с геологическими данными. В статье описывается программная реализация программного комплекса PRPHMM 1.0, способного восстанавливать оптимальные значения эредитарных математических моделей на основе производной Герасимова-Капуто. Адаптирован и реализован на языке MATLAB алгоритм безусловной оптимизации ньютоновского типа Левенберга-Марквардта. Реализованы подпрограммы для чтения, обработки и визуализации экспериментальных и модельных данных. Приводится тестовый пример, решающий на основе экспериментальных данных радонового мониторинга обратную задачу для эредитарной α-модели на параметры α и λ_0-коэффициент воздухообмена. Показано, что PRPHMM 1.0 позволяет для эредитарных математических моделей на восстанавливать значения параметров, близкие к оптимальным.

Ключевые слова: математическое моделирование, дробные производные, Герасимов-Капуто, эффект памяти, нелокальность, нелинейные уравнения, обратные задачи, безусловная оптимизация, алгоритм Левенберга–Марквардта, MATLAB.

Получение: 02.11.2024; Исправление: 10.11.2024; Принятие: 24.11.2024; Публикация онлайн: 28.11.2024

Для цитирования. Твёрдый Д. А., Макаров Е. О. Некоторые аспекты реализации программного комплекса PRPHMM 1.0 для уточнения параметров эредитарных математических моделей переноса радона в накопительной камере // *Вестник КРАУНЦ. Физ.-мат. науки.* 2024. Т. 49. № 4. С. 135-156. EDN: FMWIIQ . https://doi.org/10.26117/2079-6641-2024-49-4-135-156.

Финансирование. Исследование выполнено за счет гранта Российского научного фонда № 22-11-00064, https://rscf.ru/project/22-11-00064/.

Конкурирующие интересы. Конфликтов интересов в отношении авторства и публикации нет.

Авторский вклад и ответственность. Авторы участвовали в написании статьи и полностью несут ответственность за предоставление окончательной версии статьи в печать.

*__Корреспонденция:__ E-mail: tverdyi@ikir.ru

Vestnik KRAUNC. Fiz.-Mat. nauki. 2024. vol. 49. no. 4. P. 135-156. ISSN 2079-6641

INFORMATION AND COMPUTING TECHNOLOGIES

https://doi.org/10.26117/2079-6641-2024-49-4-135-156

Research Article

in Russian

MSC 85-04; 49N45; 26A33

Some Aspects of the Implementation of the PRPHMM 1.0 Software Package for Refining the Parameters of Hereditary Mathematical Models of Radon Transfer in a Storage Chamber

D. A. Tverdyi*[*1], E. O. Makarov[2]

[1] Institute of Cosmophysical Research and Radio Wave Propagation FEB RAS, 684034, Paratunka village, Mirnaya str., 7, Russia

[2] Kamchatka Branch of the Federal Research Center «Unified Geophysical Service of the Russian Academy of Sciences», 683023, Petropavlovsk-Kamchatsky, Piipa Boulevard st., 9, Russia

Abstract. Mathematical models of some dynamic processes can be significantly enhanced by using derivatives and integrals of non-integer order in them, taking into account effects that cannot be described by ordinary derivatives. For example, by using fractional Gerasimov- Caputo derivatives of constant and variable order, it is possible to take into account the memory effect in the process model, and the order of the derivative will be related to the intensity of the process. In particular, the authors have previously developed an hereditary α-model of the volumetric activity of radon, where the parameter α is related to the permeability of the medium. However, the question arises about determination of optimal values of both α and other parameters of the model. To solve the problem, it is possible to solve the inverse problem, a common type of problem in many scientific fields, where it is necessary to determine the values of model parameters from observed data, but it is impossible to make direct measurements of these parameters. The need for such an approach often arises when working with geological data. The article describes the software implementation of the PRPHMM 1.0 software package which can clarifying optimal values of hereditary mathematical models based on the Gerasimov- Caputo derivative. The Levenberg-Marquardt unconditional Newtonian optimisation algorithm is adapted and implemented in MATLAB language. Subroutines for reading, processing and visualisation of experimental and model data are implemented. A test case solving the inverse problem for the hereditary α-model for the parameters α and λ_0-air exchange coefficient on the basis of experimental radon monitoring data is presented. It is shown that PRPHMM 1.0 allows for the clarify of parameter values close to the optimum values for the hereditary mathematical models.

Key words: mathematical modeling, fractional derivatives, Gerasimov-Caputo, memory effect, nonlocality, nonlinear equations, inverse problems, unconditional optimization, Levenberg-Marquardt algorithm, MATLAB.

Received: 02.11.2024; Revised: 10.11.2024; Accepted: 24.11.2024; First online: 28.11.2024

For citation. Tverdyi D. A., Makarov E. O. Some aspects of the implementation of the PRPHMM 1.0 software package for refining the parameters of hereditary mathematical models of radon transfer in a storage chamber. *Vestnik KRAUNC. Fiz.-mat. nauki.* 2024, **49**: 4, 135-156. EDN: FMWIIQ . https://doi.org/10.26117/2079-6641-2024-49-4-135-156.

Funding. The research was funded by a grant from the Russian Science Foundation, project number 22-11-00064, which can be found at https://rscf.ru/project/22-11-00064/.

Competing interests. The authors declare that there are no conflicts of interest regarding authorship and publication.

Contribution and Responsibility. All authors contributed to this article. Authors are solely responsible for providing the final version of the article in print. The final version of the manuscript was approved by all authors.

[*]**Correspondence:** E-mail: tverdyi@ikir.ru

Введение

В последние годы возрос интерес к исследованию так называемых дифференциальных уравнений дробного порядка [1, 2], в которых неизвестная функция содержится под знаком производной дробного порядка. Это обусловлено как развитием самой теории дробного интегрирования и дифференцирования, так и приложениями таких конструкций в различных областях науки [3, 4].

Рассматривается дробное уравнение вида:

$$\partial_{0t}^{\alpha(t)} x(\varphi) = F(x(t), t), \qquad x(0) = x_0, \tag{1}$$

где, дробная производная типа Герасимова-Капуто переменного $0 < \alpha(t) < 1$ порядка имеет вид:

$$\partial_{0t}^{\alpha(t)} x(\varphi) = \frac{1}{\Gamma(1-\alpha(t))} \int_0^t \frac{dx(\varphi)}{dt} \frac{1}{(t-\varphi)^{\alpha(t)}} d\varphi, \tag{2}$$

где, $\Gamma(\cdot)$ – известная гамма-функция Эйлера.

Уравнение (1) лежит в основе некоторых разработанных авторами математических моделей динамических процессов: солнечной активности [5], динамики смертности от COVID-19 [6] а также моделировании объемной активности радона в накопительной камере [7]. Это достигается за счёт введения эффекта памяти [8], с помощью (2), в модель того или иного процесса, где порядок производной будет связан с интенсивностью процесса. В тоже время, выбирая в модели (1) вид функции $F(A(t), t)$ описывающие различные механизмы исследуемого динамического процесса, можно моделировать различные динамические режимы.

Модельные уравнения на основе (1) решаются с помощью численных методов. В работах [9, 10] подробно исследуется математический аппарат модельных уравнений типа (1).

Отметим, что в упомянутых исследованиях параметры моделей подбирались вручную по максимуму коэффициента детерминации и коэффициента корреляции Пирсона с экспериментальными данными объемной активности радона (ОАР), учитывая имеющееся понимание природы моделируемого процесса. Такой подход является трудоемким, что приводит к идеям применения способов автоматизации подбора оптимальных параметров за счет решения соответсвующей обратной задачи.

В настоящей статье мы будем рассматривать случай, когда $\alpha(t)$ в уравнении (1) является константой (эредитарная α-модель). В статье авторов [11] были приведены результаты применения программного комплекса PRPHMM 1.0 для восставновления значений проницаемости среды и коэффициента воздухообмена на пункте радонового мониторинга Карымшина. В настоящей статье дается описание программного комплекса PRPHMM 1.0, приводится листинг модулей и пример применения.

Эредитарная α-модель ОАР

Для решения проблемы поиска оптимальных параметров можно решать обратную задачу — распространенный тип задач во многих научных областях, где необходимо определить значения параметров модели на основе наблюдаемых данных [12], но невозможно провести прямые измерения этих параметров. Необходимость такого подхода часто возникает при работе с геологическими данными, в геофизике и сейсмологии [13] и др.

Далее, для иллюстрации кода и того что он реализует, рассмотрим конкретный пример а именно эредитарную α-модель ОАР в накопительной камере. Подробнее о процессе преноса радона можно узнать из работ [14, 15], а о α-модели [7, 11].

Анализ данных ОАР и сопутствующих параметров получаемых в ходе непрерывного мониторинга, является одним из методов поиска предвестников землетрясений. Это связано с тем, что на ОАР влияют изменения напряженно-деформированного состояния среды через которую подпочвенный газ выходит на поверхность [16], поэтому радон считается известным и хорошо себя зарекомендовавшим индикатором процессов протекающий в такой среде. Мониторинг ^{222}Rn как метод поиска предвестников сейсмических событий, за последние годы прекрасно себя показал, особенно как краткосрочный предвестник (до 15 суток) [17].

Эредитарная α-модель ОАР представляется следующей задачей Коши:

$$\partial^{\alpha}_{0,t} A(\varphi) = -\lambda_0 A(t) + \lambda_0, \qquad A(0) = A_0, \tag{3}$$

где, $A(t)$ – функция решения, зависимость ОАР от времени в камере; $\partial^{\alpha}_{0,t} A(\varphi)$ – дробная производная Герасимова-Капуто [18, 19] постоянного порядка $0 < \alpha < 1$, частный случай (2); α – интенсивность переноса радона, порядок дробной производной; λ_0 – коэффициент воздухообмена (КВО) в камере.

Задача (3) решается численно в равномерной сеточной области:

$$\begin{gathered} h = T/N, \qquad \widehat{\Omega} = \{(t_i = ih) : 0 \le i < N\}, \qquad \widehat{\mathbb{A}} \in \widehat{\Omega}, \\ A(t) = A_i, \qquad 0 < A_i < 1. \end{gathered} \tag{4}$$

а для решения (3) на сетке (4) использоваться безусловно устойчивая численная схема IFDS апробированная в ряде тестовых и прикладных задач [5, 10].

Методика решения обратной задачи для α-эредитарной модели

Пусть $A_i \in \widehat{\mathbb{A}}$ – функция некоего известного класса сеточных функций, но её решение зависит от набором параметров $\overrightarrow{X} = [X_0, ..., X_{K-1}]$, где $K = 2$ – число восстанавливаемых параметров, а $X_0 = \alpha$, $X_1 = \lambda_0$. Пусть значения дискретной функции решения $A_i \in \widehat{\mathbb{A}}$ неизвестны, но известна дополнительная информация (экспериментальные данные RVA) $A_i = \theta_i = \overrightarrow{\theta}$ о решении разностной прямой задачи Коши для эредитарной α-модели RVA.

Тогда разностная обратная задача для (3) на сетке (4) – это восстановление значений $\overrightarrow{X} = [X_0, X_1]$ по известным $A_i = \theta_i$ экспериментальным данным:

$$\begin{gathered} A_i = 1 - \frac{\widehat{\partial_{0,ih}^{X_0}} A_i}{X_1}, \qquad A_i = \theta_i, \qquad 1 \le i < N, \\ \widehat{\partial_{0,ih}^{X_0}} A_i = \frac{h^{-X_0}}{\Gamma(2-X_0)} \sum_{j=0}^{i-1} \left((j+1)^{1-X_0} - j^{1-X_0} \right) \left(A_{i-j} - A_{i-j-1} \right). \end{gathered} \tag{5}$$

Для решения (5) обратимся к теории безусловной оптимизации [20]. Для этого необходимо минимизировать функционал невязки:

$$\overrightarrow{\eta} = \overrightarrow{\theta} - \omega(\overrightarrow{X}), \qquad \min\left(\Psi\left(\overrightarrow{X}\right)\right) = \frac{1}{2}\sum_{i=0}^{N-1} \eta_i^2 = \frac{1}{2}\sum_{i=0}^{N-1} (\theta_i - \omega_i)^2, \tag{6}$$

где, $\overrightarrow{\eta}$ – вектор невязки размерности $N > K$, а вектор $\omega(\overrightarrow{X}) = [\omega_0, ..., \omega_N]$ – вектор модельных данных, т. е. решение прямой задачи относительно некоторого приближения $\overrightarrow{X}$, получаемого в ходе решения обратной задачи.

Разностная обратная задача решается методом безусловной оптимизации ньютоновского типа [21], а именно итерационным методом Левенберга-Марквардта [22, 23], представимого в виде:

$$\Delta X = \left(-H^{-1}\right) \times \left(J^T \times \overrightarrow{\eta}\right), \qquad H = J^T \times J + \gamma E, \tag{7}$$

где ΔX – оптимальное приращение $\overrightarrow{X}$ для следующей итерации;E – единичная матрица размерности $K \times K$; $J = J\left(\overrightarrow{X}\right)$ – матрица Якоби размерности $N \times K$ с элементами вычисляемыми во формуле: $J_{i,k} = \frac{\partial \eta_i}{\partial X_k}, i = 0..N-1, k = 0..K-1$; производная $\frac{\partial \eta_i}{\partial X_k}$ аппроксимируется разностным оператором $J_{i,k} = \frac{\eta_i^{\delta} - \eta_i}{\delta X_k}$, где δX – заданное малое приращение $\overrightarrow{X}$; γ – параметр регуляризации метода. Если $\gamma \in \mathbb{R}_{>0}$, а также матрица Гессе H положительно определена, то тогда ΔX является направлением спуска для оптимального шага метода; Стартовое значение: $\gamma^{(0)} = \nu \cdot \max\limits_i \left(diag\left(J\left(X^{(0)}\right)^T \times J\left(X^{(0)}\right)\right)\right)$, где ν – заданная стартовая константа.

Решение обратной задачи (5) методом Левенберга-Марквардта (7), далее (IP-LB), сводится к тому, чтобы в ходе цикла, начиная с заданных постоянных $X^{(0)}$, δX, ν а также c – константы для пересчёта γ, многократно вычисляя решение прямой задачи при приближениях $\overrightarrow{X}$, получаемых в ходе решения обратной задачи, вычислить оптимальные значения $\overrightarrow{X}$. Подробнее с алгоритмом для реализующий оптимизацию вектора $\overrightarrow{X}$ можно ознакомиться в работе [24].

Критерием того, что метод сходится к оптимальному решению является $\varepsilon \le \Sigma$, где Σ – заданная точность решения IP-LB, $\varepsilon = \frac{1}{N}\sum_{i=0}^{N-1} \left[\eta_i^{\Delta}\right]^2$ – среднеквадратичная ошибка между экспериментальными и модельными данными RVA.

Критерий того, что метод попал в "ловушку" локального минимума – это отсутствие существенного изменения значений $\overrightarrow{\Delta X}$ в ходе итераций. Иначе

говоря $\Delta\left(\overrightarrow{\Delta X}\right) \to 0$, где $\Delta\left(\overrightarrow{\Delta X}\right)$ – оценка скорости «приращения приращений» определяемая:

$$\Delta\left(\overrightarrow{\Delta X}\right) = \lim_{n\to\infty}\left(\frac{1}{K}\sum_{k=0}^{K-1}\left|\Delta X_k^{(n)} - \Delta X_k^{(n-1)}\right|\right) \to 0.$$

Программная реализация алгоритма решения обратной задачи

Далее в (лист. 1–25) представлен код реализующий описанный итерационный метод Левенберга-Марквардта, для решения обратной задачи в рамках программного комплекса PRPHMM 1.0 (рис. 1) на языке MATLAB [25]. Решение основного матричного уравнения (7) представлено в коде (лист. 14–15).

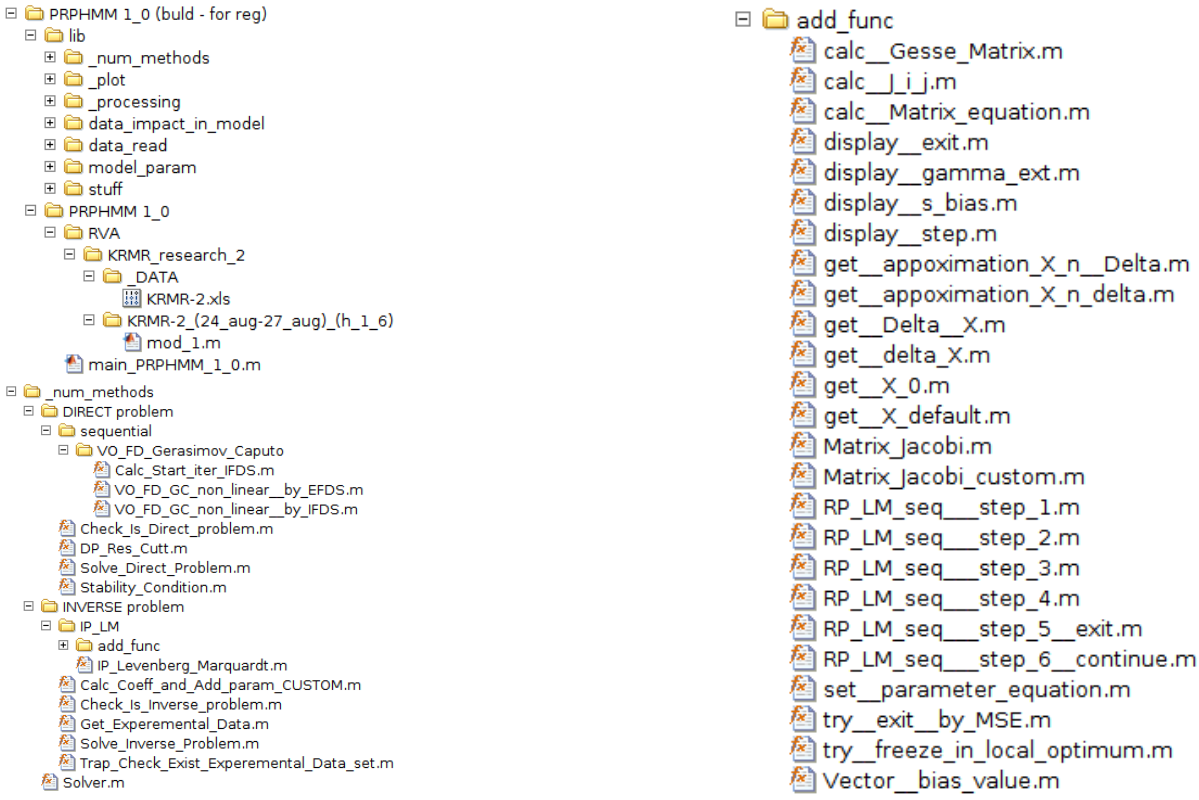

Рис. 1. Файловая структура программного комплекса PRPHMM 1.0, в частности ветвь содержащая подпрограммы для решения обратной задачи

[Figure 1. File structure of the PRPHMM 1.0 software package, in particular the branch containing subroutines for solving the inverse problem]

Листинг 1. **Calc_Coeff_and_Add_param_CUSTOM.m**

```
% пере-вычисление значений коэф. на основе описанных пользов. формул
function [ model ] = ...
    Calc_Coeff_and_Add_param_CUSTOM( model, IM )
    if( isfield( model( IM ), 'derivative' ) == 1 )
```

```
        if( isfield( model( IM ).derivative,'const_of_process') == 0 )
            disp( ['  параметр [model(' num2str(IM) ...
                   ').derivative.const_of_process] НЕ определён ' ...
                   ' -->> поэтому, будет создано  = 1 '] );
            model( IM ).derivative.const_of_process = 1;
        end
    end
    if( isfield( model( IM ), 'parameter' ) == 1 )
        names_fields     = fieldnames( model( IM ).parameter );
        num_fields_coeff = length( names_fields );
        for sub_ind = 1 : num_fields_coeff
            name_curr_field = char(names_fields(sub_ind));
            formula = ' ';
            recive_formula = sprintf(...
                'formula = model(%d).parameter.%s.define;',...
                IM, string(name_curr_field) );
            eval( recive_formula );
            model(IM).parameter.(name_curr_field).define_start_type=class(formula);
            replace_define_field = sprintf(...
                'model(%d).parameter.%s.define = ''%s'';',...
                IM, string(name_curr_field), string(formula) );
            eval( replace_define_field );
            t = model( IM ).t;
            h = model( IM ).h;
            N = model( IM ).N;
            T = model( IM ).T;
            ui = model( IM ).ui;
            if (isa(formula, 'char') == 1)
                array_values = eval(formula);
            else
                array_values(1: N + 1) = formula;
            end
            eval( sprintf( ...
                'model(%d).parameter.%s.values = array_values;',...
                IM, string( name_curr_field )) );
        end
    end
end
```

Листинг 2. **set__parameter_equation.m**

```
% пере-расчет параметров модели для решения обратной задачи
function [ model_UPDATE ] = set__parameter_equation( model, IM, X_n,X_n_m1)
    model_UPDATE = model;
    model_UPDATE( IM ).parameter.alpha.define = X_n( 1 );
    model_UPDATE( IM ).lambda = X_n( 2 );
    [ model_UPDATE ] = Calc_Coeff_and_Add_param_CUSTOM( model_UPDATE, IM );
    disp( [ '|    X_0  = ' num2str( X_n_m1(1) ) ' -> | ' num2str(X_n(1))]);
    disp( [ '|    X_1  = ' num2str( X_n_m1(2) ) ' -> | ' num2str(X_n(2))]);
end
```

Листинг 3. **Vector__bias_value.m**

```
% вычисление невязки--значения
```

```
function [ bias ] = Vector__bias_value( X, divisor )
    bias = 0.0;
    X_size = size(X,1);
    for i = 1 : X_size
        bias = bias + X( i )^2;
    end
    bias = bias / divisor;
end
```

Листинг 4. **get__appoximation_X_n_delta.m**

```
% присвоение заданого малого приращения
function [ X_n_delta ] = get__appoximation_X_n_delta( X_n, delta_X )
    X_n_delta = X_n + delta_X;
end
```

Листинг 5. **get__appoximation_X_n__Delta.m**

```
% присвоение ОПТИМАЛЬНОГО (вычисленного) приращения
function [ X_n__Delta ] = get__appoximation_X_n__Delta( X_n, Delta__X )
    X_n__Delta = X_n + Delta__X;
end
```

Листинг 6. **get__delta_X.m**

```
% СТАРОТОВОЕ - ПО умолчанию, приращение для параметров функции
function [ delta_X ] = get__delta_X( model, IM )
    delta_X( 1 ) = model( IM ).type_task.par_rest.X_0_inc_start;
    delta_X( 2 ) = model( IM ).type_task.par_rest.X_1_inc_start;
end
```

Листинг 7. **get__Delta__X.m**

```
% получение ОПТИМАЛЬНОГО (вычисленного) приращения
function [ Delta__X ] = get__Delta__X( result_M_eq )
    n_d = size( result_M_eq, 1 );
    for i = 1 : n_d
        Delta__X( i ) = result_M_eq( (n_d + 1) - i );
    end
end
```

Листинг 8. **Get_Experemental_Data.m**

```
% получение вектора эксперементальных данных
function [expr_data] = Get_Experemental_Data( model, IM, data_set )
    ind_ED = model( IM ).type_task.INDEX_exper_data;
    expr_data = data_set( ind_ED ).DATA_.VALUE;
end
```

Листинг 9. **get__X_0.m**

```
% стартовая итерация, для 1-го решения прямой задачи
function [ X_0 ] = get__X_0( model, IM )
    X_0( 1 ) = model( IM ).type_task.par_rest.X_0_start;
```

```
    X_0( 2 ) = model( IM ).type_task.par_rest.X_1_start;
end
```

Листинг 10. **get__X_default.m**

```
% получить и запомнить значения X_0 (ПО умолчанию)
function [ X_default ] = get__X_default( model, IM )
    X_default( 1 ) = model( IM ).parameter.alpha.values( 1 );
    X_default( 2 ) = model( IM ).lambda;
end
```

Листинг 11. **Matrix_Jacobi.m**

```
% вычисление матрицы Якоби
function [ J ] = Matrix_Jacobi( X_n_p1, X_n, delta_X, type_finite_diff )
    size__X_n_p1  = size( X_n_p1 , 1 );
    size__X_n     = size( X_n    , 1 );
    size__delta_X = size( delta_X, 2 );
    if( size(X_n_p1,1) ~= size(X_n,1) )
        locat = sprintf( 'Error ! dimensions wrong, %d != %d', ...
                         size__X_n_p1, size__X_n );
        error( char(locat) );
    end
    if( type_finite_diff == "upward f-d" )
        for i = 1 : size__X_n_p1
            for j = 1 : size__delta_X
                J(i,j) = ( X_n_p1(i) - X_n(i) ) / delta_X(j);
            end
        end
    end
end
```

Листинг 12. **Matrix_Jacobi_custom.m**

```
% вычисление матрицы Якоби
function [J_out] = Matrix_Jacobi_custom( J, X, delta_X,type_finite_diff,...
                                         type_calc_Jacobi )
    J_out = J;
    X_size = size(X,2);
    if( type_finite_diff == "upward f-d" )
        if( type_calc_Jacobi == "classicaly" )
            for i = 1 : X_size
                for j = 1 : X_size
                    J_out(i,j) = ( ( X(i) + delta_X(j) ) - X(i) ) / delta_X(j);
                end
            end
        else
            error( ['НЕИЗВЕСТНЫЙ : [type_calc_Jacobi]' ] );
        end
    else
        error( ['НЕИЗВЕСТНЫЙ : [type_finite_diff]' ] );
    end
end
```

Листинг 13. **calc__J_i_j.m**

```
% вычисление основной матрицы Якоби
function [ J_i_j ] = calc__J_i_j( eta_n_delta, eta_n, delta_X, n )
    locat_nd = sprintf( 'eta^( X^(%d) + delta X )', n );
    locat    = sprintf( 'eta^( X^(%d) )', n );
    [J_i_j] = Matrix_Jacobi( eta_n_delta, eta_n, delta_X, 'upward f-d' );
    display_control_sum( locat_nd, eta_n_delta );
    display_control_sum( locat,    eta_n );
    display_control_sum( 'J_i_j      ', J_i_j );
end
```

Листинг 14. **calc__Gesse_Matrix.m**

```
% вычисоение матрицы Гессе - ключевой для метода реш. обратной задачи
function [ Gesse_Matrix ] = calc__Gesse_Matrix( J_i_j, gamma_reg )
    size_J_i_j = size(J_i_j,2);
    [ gamma_E ] = Unitary_Matrix( size_J_i_j, size_J_i_j );
    gamma_PART = gamma_E * gamma_reg;
    Gesse_Matrix = (J_i_j' * J_i_j) + gamma_PART;
end
```

Листинг 15. **calc__Matrix_equation.m**

```
% вычисление решения основного матричного уравнения
function [ result_M_eq ] = calc__Matrix_equation(n, J_i_j,gamma_reg, Theta)
    [ Gesse_Matrix ] = calc__Gesse_Matrix( J_i_j, gamma_reg );
    Gesse_Matrix_Inverse = inv( Gesse_Matrix );
    J_i_j_T_mul_Theta = - 1 * J_i_j' * Theta;
    result_M_eq = Gesse_Matrix_Inverse * J_i_j_T_mul_Theta;
end
```

Листинг 16. **Solve_Inverse_Problem.m**

```
% решение обратной задачи
function [ model_OUT ] = Solve_Inverse_Problem( model, IM, expr_data )
    model_OUT = model;
    tic;
    switch( model( IM ).type_task.method )
        case( "Levenberg-Marquardt" )
            [ model_OUT ] = IP_Levenberg_Marquardt( model, IM, expr_data );
        otherwise
            error( ['НЕИЗВЕСТНЫЙ МЕТОД : [model(' num2str( IM ) ...
                    ').only_IP.method] -- решения Обратной задачи' ] );
    end
    model_OUT( IM ).time_calc = toc;
    model_OUT( IM ).RES_.var.x_raw    = model_OUT( IM ).RES_.var.x_raw';
    model_OUT(IM).RES_.diff.x_dot_raw = model_OUT(IM).RES_.diff.x_dot_raw';
    model_OUT( IM ).RES_.var.x      = model_OUT( IM ).RES_.var.x';
    model_OUT( IM ).RES_.diff.x_dot = model_OUT( IM ).RES_.diff.x_dot';
end
```

Листинг 17. **RP_LM_seq___step_1.m**

```
function [ gamma_reg ] = RP_LM_seq___step_1( k, n_d, n,count,v,X_0,delta_X)
    display__step( 1, n, count );
    J_X_0 = zeros( k,   n_d );
    [ J_X_0 ]=Matrix_Jacobi_custom(J_X_0,X_0,delta_X,"upward f-d","classicaly");
    J_X_0_T = J_X_0';
    B_0 = J_X_0_T * J_X_0;
    md_B_0( 1 : size(B_0,2) ) = 0;
    for i = 1 : size(B_0,2)
        md_B_0( i ) = B_0( i,i );
    end
    max_md_B_0 = max( md_B_0 );
    gamma_reg = v * max_md_B_0;
end
```

Листинг 18. **RP_LM_seq___step_2.m**

```
function [ s_0, eta_0 ] = RP_LM_seq___step_2( model, IM, n, count, X_0, ...
                                               X_default, expr_data, s_0_in)
    model_UPDATE = model;
    display__step( 2, n, count );
    [ model_UPDATE ] = ...
        set__parameter_equation( model, IM, X_0, X_default );
    [ model_UPDATE ] = Solve_Direct_Problem( model_UPDATE, IM );
    local_res = model_UPDATE( IM ).RES_.var.x;
    [ eta_0 ] = expr_data - local_res;
    display_control_sum( 'eta_0', eta_0 );
    s_0_previous = s_0_in;
    [ s_0 ] = Vector__bias_value( eta_0, 2 );
    display__s_bias( n, s_0, s_0_previous, 0 );
end
```

Листинг 19. **RP_LM_seq___step_3.m**

```
function [ J_i_j, eta_n_delta ] = RP_LM_seq___step_3( model, IM,n,count,...

                                       X_n, delta_X, expr_data, eta_n )
    model_UPDATE = model;
    display__step( 3, n, count );
    [ X_n_delta ] = get__appoximation_X_n_delta( X_n, delta_X );
    [ model_UPDATE ] = set__parameter_equation( model, IM, X_n_delta, X_n);
    [ model_UPDATE ] = Solve_Direct_Problem( model_UPDATE, IM );
    local_res = model_UPDATE( IM ).RES_.var.x;
    [ eta_n_delta ] = expr_data - local_res;
    [ J_i_j ] = calc__J_i_j( eta_n_delta, eta_n, delta_X, n );
end
```

Листинг 20. **RP_LM_seq___step_4.m**

```
function  [ s_1, X_n__Delta, model_UPDATE ] = ...
                    RP_LM_seq___step_4( model, IM, n, count, J_i_j, ...
                            gamma_reg, Theta, X_n, expr_data, s_1_in )
    model_UPDATE = model;
    display__step( 4, n, count );
    [ result_M_eq ] = calc__Matrix_equation( n, J_i_j, gamma_reg, Theta );
```

```
    [ Delta__X ] = get__Delta__X( result_M_eq );
    [ X_n__Delta ] = get__appoximation_X_n__Delta( X_n, Delta__X );
    [ model_UPDATE ] = set__parameter_equation( model, IM, X_n__Delta,X_n);
    [ model_UPDATE ] = Solve_Direct_Problem( model_UPDATE, IM );
    local_res = model_UPDATE( IM ).RES_.var.x;
    [ eta_n__Delta ] = expr_data - local_res;
        display_control_sum( 'eta_n__Delta  ', eta_n__Delta );
    s_1_previous = s_1_in;
    [ s_1 ] = Vector__bias_value( eta_n__Delta, 2 );
    display__s_bias( n, s_1, s_1_previous, 1 );
end
```

Листинг 21. **RP_LM_seq___step_5__exit.m**

```
function [ is_exit, X_n__SUMM, count_summ ] = ...
                        RP_LM_seq___step_5__exit( model, IM, n, count, ...
                                                  n_d, model_OUT, expr_data, ...
                                                  X_n__SUMM, X_n,count_summ, ...
                                                  count_summ_max )
    display__step( 5, n, count );
    is_exit = 0;
    SIGMA = model( IM ).type_task.par_.SIGMA;
    local_res = model_OUT( IM ).RES_.var.x;
    if( try__exit__by_MSE( expr_data, SIGMA, n, local_res ) == 1 )
        is_exit = 1;
        return;
    end
    if( count_summ < count_summ_max )
        X_n__SUMM = X_n__SUMM + X_n;
        count_summ = count_summ + 1;
    end
    if( count_summ == count_summ_max )
        if( try__freeze_in_local_optimum( X_n__SUMM, X_n, SIGMA, n,...
                                          count_summ_max ) == 1 )
            is_exit = 1;
            return;
        end
        X_n__SUMM( 1 : n_d ) = 0;
        count_summ = 0;
    end
end
```

Листинг 22. **RP_LM_seq___step_6__continue.m**

```
function [ n, X_n, s_0, s_1, X_n__Delta, gamma_reg, model_OUT ] =...
            RP_LM_seq___step_6__continue( model, IM, c, n, count, ...
                                          s_0, s_1, gamma_reg, ...
                                          model_OUT_last, expr_data, ...
                                          X_n__Delta, delta_X )
    model_OUT = model_OUT_last;
    previous_gamma = -1;
    display__step( 6, n, count );
    if( s_0 > s_1 )
        display__exit( n, 's_0', '>', 's_1', s_0, s_1,...
```

```
                        ' n++, Пересчёт шагов  3  4 ' );
        s_0 = s_1;
        X_n = X_n__Delta;
        previous_gamma = gamma_reg;
        gamma_reg = gamma_reg / c;
        n = n + 1;
        display__gamma_ext( n, 'gamma_reg', '/', 'gamma_reg', 'c',...
                           gamma_reg, previous_gamma, c );
        if( n >= 1 )
            local_res = model_OUT_last( IM ).RES_.var.x;
            eta_n = expr_data - local_res;
        end
        [ J_i_j, eta_n_delta ] = ...
                    RP_LM_seq___step_3( model, IM, n, count, ...
                    X_n, delta_X, expr_data, eta_n );
        [ s_1, X_n__Delta, model_OUT ] = ...
                    RP_LM_seq___step_4( model, IM, n, count, J_i_j, ...
                                        gamma_reg,eta_n_delta, X_n, ...
                    expr_data, s_1 );
    else
        previous_gamma = gamma_reg;
        gamma_reg = gamma_reg * c;
        display__gamma_ext( n, 'gamma_reg', '*', 'gamma_reg', 'c',...
                gamma_reg, previous_gamma, c );
        display__exit( n, 's_0', '<=', 's_1', s_0, s_1,...
                           ' Пересчёт  4 ' );
        [ s_1, X_n__Delta, model_OUT ] = ...
                    RP_LM_seq___step_4( model, IM, n, count, J_i_j, ...
                    gamma_reg,eta_n_delta, ...
                     X_n, expr_data, s_1 );
    end
end
```

Листинг 23. **IP_Levenberg_Marquardt.m**

```
function [ model_OUT ] = IP_Levenberg_Marquardt( model, IM, expr_data )
    model_OUT = model;
    N         = model( IM ).N;
    n         = 0;
    count     = 0;
    COUNT__TIME_OUT_RP = model( IM ).type_task.par_.TIME_OUT;
    k         = model( IM ).type_task.par_rest.k;
    n_d       = k;
    c         = model( IM ).type_task.par_.c;
    v         = model( IM ).type_task.par_.v;
    gamma_reg = 0;
    s_0             = 0.0; s_1             = 0.0;
    s_0_previous    = 0.0; s_1_previous    = 0.0;
    % ======= обявление матриц и векторов необходимых для алгоритма =======
    X_default           = get__X_default( model, IM );
    X_0( 1 : k )        = 0;
    X_n( 1 : k )        = 0;
    delta_X( 1 : n_d )  = 0;
    J_i_j               = zeros( N, n_d );
```

```
    X_n__SUMM( 1 : n_d ) = 0;
    % ================== СТАРТ решения обратной задачи ==================
    [ X_0 ]       = get__X_0( model, IM );
    [ delta_X ]   = get__delta_X( model, IM );
    [ gamma_reg ] = RP_LM_seq___step_1( k, n_d, n, count, v, X_0, delta_X);
        disp( ['gamma_reg = ' num2str(gamma_reg)]);
    [ s_0, eta_0 ] = RP_LM_seq___step_2( model, IM, n, count, X_0, ...
                                         X_default, expr_data, s_0 );
    if( n == 0 )
        X_n = X_0;
        eta_n = eta_0;
    end
    [ J_i_j, eta_n_delta ] = RP_LM_seq___step_3( model, IM, n,count, X_n,...
                          delta_X, expr_data, eta_n );
    [ s_1, X_n__Delta, model_OUT ] =
        RP_LM_seq___step_4( model, IM, n, count, J_i_j, gamma_reg, ...
                    eta_n_delta, X_n,expr_data, s_1 );
    % ========================== ГЛАВНЫЙ ЦИКЛ ==============================
    count_summ = 0;
    count_summ_max = 5;
    while( count < COUNT__TIME_OUT_RP )
        [ is_exit, X_n__SUMM, count_summ ] = ...
             RP_LM_seq___step_5__exit( model, IM, n, count, n_d, model_OUT,...
                     expr_data, X_n__SUMM, X_n, count_summ, count_summ_max );
        if( is_exit == 1 )
            break;
        end
        [ n, X_n, s_0, s_1, X_n__Delta, gamma_reg, model_OUT ] = ...
            RP_LM_seq___step_6__continue( model, IM, c, n, count, s_0, s_1, ...
                        gamma_reg, model_OUT, expr_data, X_n__Delta, delta_X );
        count = count + 1;
    end
end
```

Листинг 24. **try__exit__by_MSE.m**

```
% проверка : выход по достижению заданого среднего квадратического ?
function [bool] = try__exit__by_MSE( expr_data,SIGMA, n,result__X_n__Delta)
    bool = 0;
    MSE = - 1.0;
    M   = size( expr_data, 1 );
    eta_mse = expr_data - result__X_n__Delta;
    MSE = Vector__bias_value( eta_mse, M );
    if( MSE <= SIGMA )
        bool = 1;
        display__exit( n, 'MSE', '<=', 'SIGMA', MSE, SIGMA, ' EXIT ' );
    else
        display__exit( n,'MSE','>','SIGMA',MSE,SIGMA,' ВЫЧИСЛЯЕМ ДАЛЬШЕ ');
    end
end
```

Листинг 25. **try__freeze_in_local_optimum.m**

```
% проверка : застревания решения возле некоторого состояния
function [ bool ] = try__freeze_in_local_optimum( X_n__SUMM, X_n__last, ...
                                                  SIGMA,n, count_summ_max )
    bool = 0;
    X_n_size = size(X_n__SUMM,2); SIGMA_pow_2 = SIGMA^2;
    X_n__SUMM_mean( 1 : X_n_size ) = 0;
    X_n__SUMM_mean = X_n__SUMM * ( 1 / count_summ_max );
    X_n__SUMM_mean_bias( 1 : X_n_size ) = 0;
    X_n__SUMM_mean_bias = X_n__SUMM_mean - X_n__last;
        disp( [ ' try__freeze_in_local_optimum : X_n__SUMM_mean_bias = '...
                num2str(X_n__SUMM_mean_bias) ] );
    X_n_MSE = Vector__bias_value( X_n__SUMM_mean_bias, X_n_size );
        disp( [ ' try__freeze_in_local_optimum : MSE ' ...
                '( X_n__SUMM_mean, X_n__last ) = ' num2str(X_n_MSE) ] );
    if( X_n_MSE <= SIGMA_pow_2 )
        bool = 1;
        display__exit( n, 'MSE( X_n )', '<=', 'SIGMA ^2', ...
                       X_n_MSE, SIGMA_pow_2, ' ВЫХОД ( Застыл !!! ) ');
    else
        display__exit( n, 'MSE( X_n )', '>', ...
                'SIGMA ^2 (проверка застревания в локальном оптимуме)', ...
                 X_n_MSE, SIGMA_pow_2, ' ПЕРЕПРОВЕРКА в следующем ОКНЕ ' );
    end
end
```

Тестовый пример

Листинг 26. Файл управляющих параметров – **mod_1**

```
RESET;
COLOR;
DEF_COUNTERS;
TVHN;
% -----------------<     Определение ... МОДЕЛЕЙ  >----------------------
model = [];
% ------------------------------------------------------------------------
%                       classic             KRMR-2_(24_aug-27_aug)_(h_1_6)
IND = IND + 1;
model(IND).name             = "Классическая модель RVA";
model(IND).N                = 10000;
model(IND).h                = 1/6;
model(IND).start_point      = 0.9;
model(IND).A_max = 1;
model(IND).lambda                   = 0.06;
model(IND).parameter.a.define       = 0;
model(IND).parameter.b.define       = - model(IND).lambda;
model(IND).parameter.c.define       = model(IND).A_max * model(IND).lambda;
model(IND).derivative.type          = 'VOGC';
model(IND).parameter.alpha.define   = 0.9999;
% -- подстройка парам. [model().] под данные
```

```
model(IND).impact_data.INDEX_data_set              = 1;
model(IND).impact_data.re_define_N                 = 1;
model(IND).impact_data.re_define_start_point       = 1;
% -- численный метод решения
model(IND).num_method.result_last_cutt             = 1;
model(IND).num_method.type                         = 'IFDS (MNM)';
model(IND).num_method.IFDS_accuracy                = 1e-3;
model(IND).num_method.IFDS_start_iter              = 'EFDS u(N)';
% -- тип решения    Direct problem
model(IND).type_task.type   =  'DIRECT problem';
% -- отрисовка
model(IND).display.is_plot_proc = 1;
model(IND).display.parameter.style   = '--';
model(IND).display.parameter.color   = COLOR_.Blue;
model(IND).display.result.style = '-';
model(IND).display.result.color = COLOR_.Blue;
% ------------------------------------------------------------------------
%                       KRMR-2_(24_aug-27_aug)_(h_1_6)
IND = IND + 1;
model(IND).name             = "Эредитарная \alpha-модель RVA";
model(IND).N                = 10000;
model(IND).h                = 1/6;
model(IND).start_point      = 0.9;
model(IND).A_max = 1;
model(IND).lambda                  = 0.055;
model(IND).parameter.a.define      = 0;
model(IND).parameter.b.define      = - model(IND).lambda;
model(IND).parameter.c.define      = model(IND).A_max * model(IND).lambda;
model(IND).derivative.type         = 'VOGC';
model(IND).parameter.alpha.define  = 0.818;
% -- подстройка парам. [model().] под данные
model(IND).impact_data.INDEX_data_set              = 1;
model(IND).impact_data.re_define_N                 = 1;
model(IND).impact_data.re_define_start_point       = 1;
% -- численный метод решения
model(IND).num_method.result_last_cutt             = 1;
model(IND).num_method.type                         = 'IFDS (MNM)';
model(IND).num_method.IFDS_accuracy                = 1e-3;
model(IND).num_method.IFDS_start_iter              = 'EFDS u(N)';
% -- тип решения    Direct problem
model(IND).type_task.type  = 'DIRECT problem';
% -- отрисовка
model(IND).display.is_plot_proc = 1;
model(IND).display.parameter.style   = '--';
model(IND).display.parameter.color   = COLOR_.Bright_green;
model(IND).display.result.style = '-';
model(IND).display.result.color = COLOR_.Bright_green;
% ------------------------------------------------------------------------
%          INVERSE           KRMR-2_(24_aug-27_aug)_(h_1_6)
IND = IND + 1;
model(IND).name             = "Обратная задача";
model(IND).N                = 10000;
model(IND).h                = 1/6;
```

```
model(IND).start_point      = 0.9;
model(IND).A_max            = 1;
model(IND).lambda           = 0.06;
model(IND).parameter.a.define  = '0 * ui';
model(IND).parameter.b.define  = '- model(IM).lambda * ui';
model(IND).parameter.c.define  = 'model(IM).A_max * model(IM).lambda * ui';
model(IND).derivative.type     = 'VOGC';
model(IND).parameter.alpha.define  = '0.9999 * ui';
% -- подстройка парам. [model().] под данные
model(IND).impact_data.INDEX_data_set             = 1;
model(IND).impact_data.re_define_N                = 1;
model(IND).impact_data.re_define_start_point      = 1;
% -- численный метод решения
model(IND).num_method.result_last_cutt            = 1;
model(IND).num_method.type                        = 'IFDS (MNM)';
model(IND).num_method.IFDS_accuracy               = 1e-3;
model(IND).num_method.IFDS_start_iter             = 'EFDS u(N)';
% -- тип решения    Inverse problem
model(IND).type_task.type   = 'INVERSE problem';
model(IND).type_task.method = 'Levenberg-Marquardt';
model(IND).type_task.INDEX_exper_data = 1;
model(IND).type_task.par_.TIME_OUT  =    50;
model(IND).type_task.par_.v         =    100;
model(IND).type_task.par_.c         =    2;
model(IND).type_task.par_.SIGMA     =    7e-3;
model(IND).type_task.par_rest.k              =   2;
model(IND).type_task.par_rest.X_0_start      =   0.05;
model(IND).type_task.par_rest.X_1_start      =   0.0025;
model(IND).type_task.par_rest.X_0_inc_start     =   0.01;
model(IND).type_task.par_rest.X_1_inc_start     =   0.0005;
% -- отрисовка
model(IND).display.is_plot_proc = 1;
model(IND).display.parameter.style   = '--';
model(IND).display.parameter.color   = COLOR_.Red;
model(IND).display.result.style         = '-';
model(IND).display.result.color         = COLOR_.Red;
% --------------<     Определение ... НАБОРОВ ДАННЫХ  >--------------------
data_set = [];
IND = 0;
% ------------------------------------------------------------------------
%                       KRMR-2_(24_aug-27_aug)_(h_1_6)
IND = IND + 1;
data_set(IND).file.name                 = 'KRMR-2.xls';
data_set(IND).file.name_add             = ...
    "Эксперементальные данные RVA (KRMR-2: 24 aug-27 aug) (h 1/6)";
data_set(IND).file.name_column_Data     = "middle (Radon1 (Bq.m3))";
data_set(IND).file.index_column_date    = [1];
% -- выборка по календарю -
data_set(IND).date_bound_op.index_type_datatime = 4;
data_set(IND).date_bound_op.filler_Second       = 0;
    data_set(IND).date_bound.start.Year       = 2020;
    data_set(IND).date_bound.start.Month      = 08;
    data_set(IND).date_bound.start.Day        = 24;
```

```
    data_set(IND).date_bound.start.Hour       = 13;
    data_set(IND).date_bound.start.Minute     = 40;
        data_set(IND).date_bound.end.Year         = 2020;
        data_set(IND).date_bound.end.Month        = 08;
        data_set(IND).date_bound.end.Day          = 27;
        data_set(IND).date_bound.end.Hour         = 03;
        data_set(IND).date_bound.end.Minute       = 30;
% -- обработка
data_set(IND).processing.normalize_on_max = true;
% -- отрисовка
data_set(IND).display.data.style           = '-';
data_set(IND).display.data.color           = COLOR_.Black;
% ---------------<     ЗАПУСК ПРОГРАММНОГО КОМПЛЕКСА  >--------------------
main_PRPHMM_1_0;
```

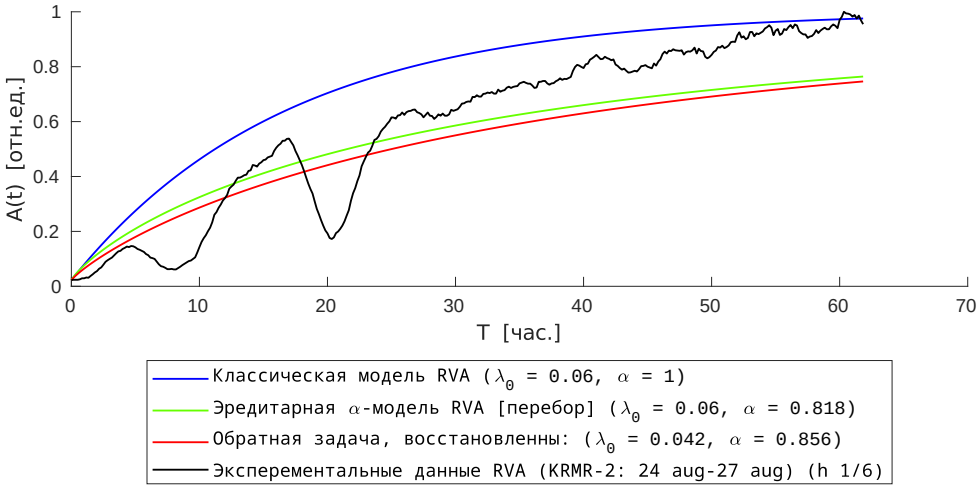

Рис. 2. Результат работы программного комплекса PRPHMM 1.0. Визуализация экспериментальных и модельных данных [11].

[Figure 2 The result of the PRPHMM 1.0 software package. Visualisation of experimental and model data [11].]

На рис. 2 приведены графики ОАР, полученные по результатам модельных расчетов (классическая модель и α-модель) в программном комплексе PRPHMM 1.0 и результатам наблюдений с пункта Камчатского филиала федерального исследовательского центра «Единая геофизическая служба РАН» Карымшина (KRMR-2) 24-27 августа 2024 г. (предоставлены к.ф.-м.н., Макаровым Е.О.). Мы видим, что решение обратной задачи позволяет давать в автоматическом режиме приемлемые оценки порядка дробной производной α и коэффициента воздухообмена λ_0 в модельном уравнении. Коэффициент воздухообмена достаточно тяжело измерить с помощью геофизической аппаратуры. В тоже время зная его можно расчитать плотность потока радона в накопительной камере, что является важным при исследований аномалий в ОАР, в том числе предшествующих сейсмическим событиям.

Заключение

Программный комплекс PRPHMM 1.0 позволяет проводить автоматизацию расчетов для уточнения значений параметров эредитарных математических моделей переноса радона в накопительной камере. В статье с помощью программного комплекса PRPHMM 1.0 и экспериментальных данных ОАР на пункте Карымшина для эредитарной α-модели переноса радона в накопительной камере были уточнены параметры порядка дробной производной α (проницаемость среды) и коэффициента воздухообмена λ_0. Расчеты показали, что значения оценимаемых параметров является адекватными. Программный комплекс PRPHMM 1.0 успешно прошел тестирование и в настоящее время ведется работа по восставновлению значний параметров функций $\alpha(t)$ для эредитарной $\alpha(t)$-модели.

Благодарности. Авторы благодарят д.ф.-м.н. Паровика Р.И. за ценные замечания при обсуждении результатов, полученных в статье.

Список литературы

1. Нахушев А. М. *Дробное исчисление и его применение.* Москва: Физматлит, 2003. 272 с. ISBN 5-9221-0440-3.
2. Uchaikin V. V. *Fractional Derivatives for Physicists and Engineers*, Background and Theory, vol. I. Berlin/Heidelberg: Springer, 2013. 373 pp. ISBN 978-3-642-33911-0 DOI: 10.1007/978-3-642-33911-0.
3. Kilbas A. A., Srivastava H. M., Trujillo J. J. *Theory and Applications of Fractional Differential Equations, 1st ed.*. Amsterdam: Elsevier Science Limited, 2006. 523 pp. ISBN 978-0444518323.
4. Самко С. Г., Килбас А. А., Маричев О. И. *Интегралы и производные дробного порядка и некоторые их приложения.* Минск: Наука и техника, 1987. 688 с.
5. Tverdyi D. A., Parovik R. I. Application of the Fractional Riccati Equation for Mathematical Modeling of Dynamic Processes with Saturation and Memory Effect // *Fractal and Fractional*, 2022. vol. 6, no. 3, pp. 163 DOI: 10.3390/fractalfract6030163.
6. Tverdyi D. A., Parovik R. I. Fractional Riccati equation to model the dynamics of COVID-19 coronavirus infection // *Journal of Physics: Conference Series*, 2021. vol. 2094, pp. 032042 DOI: 10.1088/1742-6596/2094/3/032042.
7. Tverdyi D. A., Makarov E. O., Parovik R. I. Hereditary Mathematical Model of the Dynamics of Radon Accumulation in the Accumulation Chamber // *Mathematics*, 2023. vol. 11, no. 4, pp. 850 DOI: 10.3390/math11040850.
8. Volterra V. Sur les équations intégro-différentielles et leurs applications // *Acta Mathematica*, 1912. vol. 35, no. 1, pp. 295–356 DOI: 10.1007/BF02418820.
9. Parovik R. I. Tverdyi D. A. Some Aspects of Numerical Analysis for a Model Nonlinear Fractional Variable Order Equation // *Mathematical and Computational Applications*, 2021. vol. 26, no. 3, pp. 55 DOI: 10.3390/mca26030055.
10. Tverdyi D. A., Parovik R. I. Investigation of Finite-Difference Schemes for the Numerical Solution of a Fractional Nonlinear Equation // *Fractal and Fractional*, 2022. vol. 6, no. 1, pp. 23 DOI: 10.3390/fractalfract6010023.
11. Твёрдый Д. А., Макаров Е. О., Паровик Р. И. Идентификация параметров математической α-модели переноса радона в накопительной камере по данным пункта Карымшина на Камчатке // *Вестник КРАУНЦ. Физ.-мат. науки*, 2024. Т. 48, №3, С. 95-119 DOI: 10.26117/2079- 6641-2024-48-3-95-119.
12. Tarantola A. *Inverse problem theory: methods for data fitting and model parameter estimation.* Amsterdam and New York: Elsevier Science Pub. Co., 1987. 613 pp. ISBN 0444427651.
13. Lailly P. The seismic inverse problem as a sequence of before stack migrations // *Conference on Inverse Scattering, Theory and application*, 1983, pp. 206–220.

14. Фирстов П. П., Макаров Е. О. *Динамика подпочвенного радона на Камчатке и сильные землетрясения.* Петропавловск-Камчатский: Камчатский государственный университет им. Витуса Беринга, 2018. 148 с. ISBN 978-5-7968-0691-3.
15. Фирстов П. П., Рудаков В. П. Результаты регистрации подпочвенного радона в 1997–2000 гг. на Петропавловск-Камчатском геодинамическом полигоне // *Вулканология и сейсмология*, 2003. № 1, С. 26–41.
16. Utkin V. I., Yurkov A. K. Radon as a tracer of tectonic movements // *Russian Geology and Geophysics*, 2010. vol. 51, no. 2, pp. 220–227 DOI: 10.1016/j.rgg.2009.12.022.
17. Бирюлин С. В., Козлова И. А., Юрков А. К. Исследование информативности объемной активности почвенного радона при подготовке и реализации тектонических землетрясений на примере Южно-Курильского региона // *Вестник КРАУНЦ. Науки о Земле*, 2019. Т. 4, № 44, С. 73–83 DOI: 10.31431/1816-5524-2019-4-44-73-83.
18. Герасимов А. Н. Обобщение законов линейного деформирования и их применение к задачам внутреннего трения // *АН СССР. Прикладная математика и механика*, 1948. Т. 44, № 6, С. 62-78.
19. Caputo M. Linear models of dissipation whose Q is almost frequency independent – II // *Geophysical Journal International*, 1967. vol. 13, no. 5, pp. 529–539 DOI: 10.1111/j.1365-246X.1967.tb02303.x.
20. Dennis J. E., Robert Jr., Schnabel B. *Numerical methods for unconstrained optimization and nonlinear equations.* Philadelphia: SIAM, 1996. 394 pp. ISBN 9781611971200.
21. Gill P. E., Murray W., Wright M. H. *Practical Optimization.* Philadelphia: SIAM, 2019. 421 pp.
22. Levenberg K. A method for the solution of certain non-linear problems in least squares // *Quarterly of applied mathematics*, 1944. vol. 2, no. 2, pp. 164–168 DOI: 10.1090/qam/10666.
23. Marquardt D. W. An algorithm for least-squares estimation of nonlinear parameters // *Journal of the society for Industrial and Applied Mathematics*, 1963. vol. 11, no. 2, pp. 431–441 DOI: 10.1137/0111030.
24. Твёрдый Д. А., Паровик Р. И. О задаче оптимизации для определения вида функциональной зависимости переменного порядка дробной производной типа Герасимова-Капуто // *Вестник КРАУНЦ. Физико-математические науки*, 2024. Т. 47, № 2, С. 35–57 DOI: 10.26117/2079-6641-2024-47-2-35-57.
25. Ford W. *Numerical linear algebra with applications: Using MATLAB, 1st edition.* Massachusetts: Academic Press, 2014. 628 pp. ISBN 978-0123944351 DOI: 10.1016/C2011-0-07533-6.

Информация об авторах

Твёрдый Дмитрий Александрович✉ – кандидат физико-математических наук, научный сотрудник лаборатории элетромагнитных излучений, Институт космофизических исследований и распространения радиоволн ДВО РАН, с. Паратунка, Россия, ORCID 0000-0001-6983-5258.

Макаров Евгений Олегович✉ – кандидат физико-математических наук, старший научный сотрудник лаборатории акустического и радонового мониторинга, Камчатский филиал федерального исследовательского центра "Единая геофизическая служба РАН г. Петропавловск-Камчатский, Россия, ORCID 0000-0002-0462-3657.

References

[1] Nakhushev A. M. Fractional calculus and its application. Moscow: Fizmatlit, 2003, 272 pp., isbn: 5-9221-0440-3 (In Russian)

[2] Uchaikin V. V. Fractional Derivatives for Physicists and Engineers. Vol. I. Background and Theory. Berlin/Heidelberg, Springer, 2013, 373 pp. DOI: 10.1007/978-3-642-33911-0.

[3] Kilbas A. A., Srivastava H. M., Trujillo J. J. Theory and Applications of Fractional Differential Equations, 1st ed. Amsterdam, Elsevier, 2006, 523 pp., isbn: 978-0444518323.

[4] Samko S. G., Kilbas A. A., Marichev O. I. Integraly i proizvodnye drobnogo poryadka i nekotorye ih prilozheniya [Fractional integrals and derivatives and some of their applications]. Science and tech: Minsk, 1987, 688 pp.(In Russian)

[5] Tverdyi D. A., Parovik R. I. Application of the Fractional Riccati Equation for Mathematical Modeling of Dynamic Processes with Saturation and Memory Effect, Fractal and Fractional, 2022, vol. 6, no. 3, pp. 163. DOI: 10.3390/fractalfract6030163.

[6] Tverdyi D. A., Parovik R. I. Fractional Riccati equation to model the dynamics of COVID-19 coronavirus infection, Journal of Physics: Conference Series, 2021, vol. 2094, pp. 032042. DOI: 10.1088/1742-6596/2094/3/032042.

[7] Tverdyi D. A., Makarov E. O., Parovik R. I. Hereditary Mathematical Model of the Dynamics of Radon Accumulation in the Accumulation Chamber, Mathematics, 2023, vol. 11, no. 4, pp. 850. DOI: 10.3390/math11040850.

[8] Volterra V. Sur les équations intégro-différentielles et leurs applications, Acta Mathematica, 1912, vol. 35, no. 1, pp. 295–356. DOI: 10.1007/BF02418820.

[9] Parovik R. I. Tverdyi D. A. Some Aspects of Numerical Analysis for a Model Nonlinear Fractional Variable Order Equation, Mathematical and Computational Applications, 2021, vol. 26, no. 3, pp. 55. DOI: 10.3390/mca26030055.

[10] Tverdyi D. A., Parovik R. I. Investigation of Finite-Difference Schemes for the Numerical Solution of a Fractional Nonlinear Equation, Fractal and Fractional, 2022, vol. 6, no. 1, pp. 23. DOI: 10.3390/fractalfract6010023.

[11] Tverdyi D. A., Makarov E. O., Parovik R. I. Identification of parameters of the mathematical α-model of radon transport in the accumulation chamber based on data from the Karymshina site in Kamchatka, Bulletin KRASEC. Physical and Mathematical Sciences, 2024, vol. 48, no. 3, pp. 95–119. DOI: 10.26117/2079-6641-2024-48-3-95-119.(In Russian)

[12] Tarantola A. Inverse problem theory: methods for data fitting and model parameter estimation, Amsterdam and New York: Elsevier Science Pub. Co., 1987, 613 pp., isbn: 0444427651.

[13] Lailly P. The seismic inverse problem as a sequence of before stack migrations, Conference on Inverse Scattering, Theory and application, 1983, pp. 206–220.

[14] Firstov P. P., Makarov E. O. Dynamics of subsurface radon in Kamchatka and strong earthquakes. Petropavlovsk-Kamchatsky, Vitus Bering Kamchatka State University, 2018, 148 pp., isbn: 978-5-7968-0691-3 (In Russian)

[15] Firstov P. P., Rudakov V. P. Results from observations of subsurface radon in 1997-2000 at the Petropavlovsk-Kamchatskii geodynamic site. Journal of Volcanology and Seismology, 2003, no. 1, pp. 26–41 (In Russian)

[16] Utkin V. I., Yurkov A. K. Radon as a tracer of tectonic movements, Russian Geology and Geophysics, 2010, vol. 51, no. 2, pp. 220–227. DOI: 10.1016/j.rgg.2009.12.022

[17] Biryulin S. V., Kozlova I. A., Yurkov A. K. Investigation of informative value of volume radon activity in soil during both the stress build up and tectonic earthquakes in the South Kuril region, Bulletin of KRASEC. Earth Sciences, 2019, vol. 4, no. 44, pp. 73–83. DOI: 10.31431/1816-5524-2019-4-44-73-83.

[18] Gerasimov A. N. Generalization of linear deformation laws and their application to internal friction problems, Applied Mathematics and Mechanics, 1948, vol. 12, pp. 529–539.
[19] Caputo M. Linear models of dissipation whose Q is almost frequency independent – II, Geophysical Journal International, 1967, vol. 13, no. 5, pp. 529–539. DOI: 10.1111/j.1365-246X.1967.tb02303.x.
[20] Dennis J. E., Robert Jr., Schnabel B. Numerical methods for unconstrained optimization and nonlinear equations. Philadelphia, SIAM, 1996, 394 pp., isbn: 9781611971200
[21] Gill P. E., Murray W., Wright M. H. Practical Optimization. Philadelphia, SIAM, 2019, 421 pp.
[22] Levenberg K. A method for the solution of certain non-linear problems in least squares, Quarterly of applied mathematics, 1944, vol. 2, no. 2, pp. 164–168. DOI: 10.1090/qam/10666.
[23] Marquardt D. W. An algorithm for least-squares estimation of nonlinear parameters, Journal of the society for Industrial and Applied Mathematics, 1963, vol. 11, no. 2, pp. 431–441. DOI: 10.1137/0111030.
[24] Tverdyi D. A., Parovik R. I. The optimization problem for determining the functional dependence of the variable order of the fractional derivative of the Gerasimov-Caputo type, Bulletin KRASEC. Physical and Mathematical Sciences, 2024, vol. 47, no. 2, pp. 35–57. DOI: 10.26117/2079-6641-2024-47-2-35-57.(In Russian)
[25] Ford W. Numerical linear algebra with applications: Using MATLAB, 1st edition. Massachusetts, Academic Press, 2014, 628 pp., isbn: 978-0123944351. DOI: 10.1016/C2011-0-07533-6

Information about the authors

Tverdyi Dmitrii Alexsandrovich – PhD (Phys. & Math.), Researcher, Electromagnetic Radiation Laboratory, Institute of Cosmophysical Research and Radio Wave Propagation, FEB RAS, Paratunka village, Russia, ORCID 0000-0001-6983-5258.

Makarov Evgeny Olegovich – PhD (Phys. & Math.), Senior Researcher, Acoustic and Radon Monitoring Laboratory, Kamchatka Branch of the Federal Research Centre "Unified Geophysical Service of the Russian Academy of Sciences Petropavlovsk-Kamchatsky, Russia, ORCID 0000-0002-0462-3657.