Algorithm for Restoring the Electrical Resistivity of a Medium with Plane-Parallel and Coaxial-Cylindrical Interface Boundaries Based on Lateral Logging Sounding Data

Dmitry Prokhorov(✉) and Veronika Chepelenkova

Sobolev Institute of Mathematics SB RAS, 4 Koptug Avenue, Novosibirsk, Russia 630090
prokhorovdi@ipgg.sbras.ru

Abstract. This paper presents an algorithmic complex designed to restore the properties of a medium with plane-parallel and coaxial-cylindrical interface boundaries based on lateral logging sounding data. The complex includes an algorithm for direct numerical modeling of lateral logging sounding (LLS) and an algorithm for restoring the electrical resistivity of formations with drilling fluid invasion zones according to the LLS curves.

Keywords: Lateral logging sounding · Direct methods for SLAE · Nelder-Mead method

1 Introduction

Electrical resistivity is the most important parameter determining the reservoir properties of rocks. Therefore, lateral logging sounding occupies one of the leading positions in the geophysical studies of oil and gas wells. In LLS, measuring and current electrodes are located on the axis of the borehole, coinciding with the medium symmetry axis. A two-dimensional axisymmetric distribution of the resistivity of the rocks surrounding the borehole must be determined. The uniqueness of the solution in the case of the two-dimensional piecewise constant resistivity has been proven in [5].

In this paper we use alternative approach. The algorithm for solving the inverse problem is based on using the non-gradient Nelder-Mead minimization method [6,12]. This method consists of sequentially moving and deforming the simplex around the extremum point in the parameter space of the inverse problem. The Nelder-Mead method does not use the gradient of the objective function, which makes it possible to significantly simplify both the mathematical justification and the software implementation of the algorithm for solving the inverse problem.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2026
O. Gervasi et al. (Eds.): ICCSA 2025 Workshops, LNCS 15888, pp. 402–418, 2026.
https://doi.org/10.1007/978-3-031-97596-7_27

The solution to the direct problem uses the finite difference method and the direct SLAE solution method with a sparse matrix of coefficients. A finite-difference approximation of the electric field potential distribution equation reduces the problem of numerical simulation of LLS to the solution of an SLAE with a sparse matrix and a set of right-hand sides. As a rule, the direct SLAE solution method consists of three stages:

1. Symbolic factorization.
2. Numerical factorization.
3. Solution.

The first stage is the most time-consuming due to the weak parallelization of [14]. However, the developed algorithm for solving the inverse problem is constructed so that all direct problems (arising within the framework of solving the inverse problem) are approximated on the same computational grid, which leads to the fact that all direct problems have the same sparse coefficient matrix template. It means that symbolic factorization is performed exactly once at the beginning of the algorithm. Then, the matrix coefficients are changed on the fly without violating the sparse matrix template.

Based on the nature of the task (a single execution of a complex and difficult-to-parallelize operation and SLAE solution for many right-hand sides), the optimal choice is to use the NVIDIA cuDSS [1] library for graphics accelerators. Since cuDSS entails storing the matrix in GPU memory, the algorithm is designed to avoid extremely time-consuming exchanges between video and CPU memory. Therefore, the matrix is loaded into VRAM once, and then, when the environment parameters change, the coefficients are changed directly in the graphics accelerator's memory.

Using cuDSS provides a considerable performance boost in solving this problem compared to libraries focused on solving SLAE using the CPU. However, it is worth noting that due to the rapid development of artificial intelligence technologies, graphics accelerators have become an expensive and scarce commodity, which entails significant restrictions on using the developed algorithm (availability of appropriate infrastructure). Nevertheless, it is not difficult to reorient the algorithm to CPU usage. For example, the Intel MKL PARDISO [2] library can be used with Intel Xeon Phi KNL [11] multithreaded processors. Also, SLAE can be solved by iterative methods (for example, the biconjugate gradient stabilized method [15]).

2 Problem Statement

In a cylindrical coordinate system with radial symmetry, the potential distribution of the electric field is described by the equation:

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\sigma\frac{\partial U}{\partial r}\right)+\frac{\partial}{\partial z}\left(\sigma\frac{\partial U}{\partial z}\right)=-g, \tag{1}$$

where U — electric field potential, about σ — electrical conductivity, g — in case of point source of direct current with intensity I at the middle of the borehole at point z_0 is $\frac{I}{2\pi}\delta(r)\delta(z-z_0)$. For the Eq. 1, a mixed boundary value problem with boundary conditions is posed:

$$\frac{\partial U}{\partial r}\Big|_{r=0} = 0,\ U\Big|_{\sqrt{r^2+z^2}\to\infty} = 0. \tag{2}$$

3 Finite-Difference Scheme

To approximate the Eq. 1, we introduce a uniform rectangular grid with steps h_z, h_r along the directions z and r, respectively. The grid nodes are points with coordinates $z_{m_z} = h_z m_z$ and $r_{m_r} = h_r m_r$, where the indices m_z, m_r take half-integer values. Setting the grid cells:

$$C_{m_z,m_r} = [h_z(m_z-1/2), h_z(m_z+1/2)] \times [h_r(m_r-1/2), h_r(m_r+1/2)]. \tag{3}$$

The potential and conductivity are considered constant inside the grid cell, while they may have discontinuities at the cell boundary. Then the finite-difference approximation of the Eq. 1 has the following form:

$$\begin{aligned}
&\frac{\tilde\sigma_{m_z+1/2,m_r}}{h_z^2}U_{m_z+1,m_r} + \frac{\tilde\sigma_{m_z,m_r+1/2}r_{m_z,m_r+1/2}}{r_{m_z,m_r}h_r^2}U_{m_z,m_r+1} + \\
&+\frac{\tilde\sigma_{m_z-1/2,m_r}}{h_z^2}U_{m_z-1,m_r} + \frac{\tilde\sigma_{m_z,m_r-1/2}r_{m_z,m_r-1/2}}{r_{m_z,m_r}h_r^2}U_{m_z,m_r-1} - \\
&-\Bigg(\frac{\tilde\sigma_{m_z+1/2,m_r}+\tilde\sigma_{m_z-1/2,m_r}}{h_z^2} + \\
&+\frac{\tilde\sigma_{m_z,m_r+1/2}r_{m_z,m_r+1/2}+\tilde\sigma_{m_z,m_r-1/2}r_{m_z,m_r-1/2}}{r_{m_z,m_r}h_r^2}\Bigg)U_{m_z,m_r} = \\
&= -g_{m_z,m_r}.
\end{aligned} \tag{4}$$

Since the boundary value problem for the system of Eqs. 1 is posed in unbounded space, the finite-difference scheme is constructed in the corresponding unbounded space. It is impossible to solve such a problem in practice. Therefore, a limited computational area is allocated from the space $\Omega = [Z_{min}, Z_{max}] \times [0, R_{max}]$. At the boundaries of $z = Z_{min}$, $z = Z_{max}$ and $r = R_{max}$, the Dirichlet condition is set: $U = 0$. Let $Z_{max} = M_z h_z$ and $R_{max} = M_r h_r$. Then, the finite-difference analog of the Dirichlet condition:

$$\begin{aligned}
\frac{U_{-1/2,m_r}+U_{1/2,m_r}}{2} &= \frac{U_{M_z-1/2,m_r}+U_{M_z+1/2,m_r}}{2} = \\
&= \frac{U_{m_z,M_r-1/2}+U_{m_z,M_r+1/2}}{2} = 0.
\end{aligned} \tag{5}$$

A finite-difference analog of the Neumann condition:

$$\frac{U_{m_z,1/2}-U_{m_z,-1/2}}{h_r} = 0. \tag{6}$$

When creating SLAE, values in nodes outside the Ω domain are excluded from the expression 4. From the Dirichlet condition 5:

$$\begin{aligned} U_{-1/2,m_r} &= -U_{1/2,m_r}, \\ U_{M_z+1/2,m_r} &= -U_{M_z-1/2,m_r}, \\ U_{m_z,M_r+1/2} &= -U_{m_z,M_r-1/2}. \end{aligned} \tag{7}$$

From the Neumann condition 6:

$$U_{m_z,-1/2} = U_{m_z,1/2}. \tag{8}$$

However, this approach worsens the quality of the solution since the Dirichlet condition is not set at infinity. It is possible to distance the boundary conditions without significantly increasing computational costs by using the theory of optimal grids [4,9]. Uniform grids are used in this work for ease of implementation, and the boundary condition is set at a given distance from some computational area of interest.

The matrix of SLAE 4 is not symmetric but can be symmetrized by row and column operations. Let the initial SLAE have the form:

$$\mathcal{A}x = b, \tag{9}$$

then, multiplying from the left by the diagonal matrix, we obtain the equivalent system:

$$D_r\mathcal{A}x = D_r b. \tag{10}$$

If we apply to the matrix $\mathcal{A}$ column operation, then the transformed system will no longer be equivalent to the system 9:

$$D_r\mathcal{A}D_c\bar{x} = D_r b, \tag{11}$$

however, the solution x of the original system 9 can be obtained from the solution $\bar{x}$ of the system 11:

$$x = D_c\bar{x}. \tag{12}$$

So, the SLAE matrix 4 can be symmetrized by matrices:

$$D_r = \begin{pmatrix} \sqrt{r_{1/2,1/2}} & & & 0 \\ & \sqrt{r_{1/2,3/2}} & & \\ & & \ddots & \\ 0 & & & \sqrt{r_{M_z-1/2,M_r-1/2}} \end{pmatrix}, \tag{13}$$

$$D_c = \begin{pmatrix} \frac{1}{\sqrt{r_{1/2,1/2}}} & & & 0 \\ & \frac{1}{\sqrt{r_{1/2,3/2}}} & & \\ & & \ddots & \\ 0 & & & \frac{1}{\sqrt{r_{M_z-1/2,M_r-1/2}}} \end{pmatrix}. \tag{14}$$

That is, $D_c = D_r^{-1}$. Next, we will denote $D_r = D, D_c = D^{-1}$.

After the transformation, the finite difference scheme will take the form:

$$\begin{aligned}
&\frac{\tilde{\sigma}_{m_z+1/2,m_r}}{h_z^2}U_{m_z+1,m_r} + \frac{\tilde{\sigma}_{m_z,m_r+1/2}r_{m_z,m_r+1/2}}{\sqrt{r_{m_z,m_r}r_{m_z,m_r+1}}h_r^2}U_{m_z,m_r+1} + \\
&+\frac{\tilde{\sigma}_{m_z-1/2,m_r}}{h_z^2}U_{m_z-1,m_r} + \frac{\tilde{\sigma}_{m_z,m_r-1/2}r_{m_z,m_r-1/2}}{\sqrt{r_{m_z,m_r-1}r_{m_z,m_r}}h_r^2}U_{m_z,m_r-1} - \\
&-\left(\frac{\tilde{\sigma}_{m_z+1/2,m_r} + \tilde{\sigma}_{m_z-1/2,m_r}}{h_z^2} + \right. \\
&\left.+\frac{\tilde{\sigma}_{m_z,m_r+1/2}r_{m_z,m_r+1/2} + \tilde{\sigma}_{m_z,m_r-1/2}r_{m_z,m_r-1/2}}{r_{m_z,m_r}h_r^2}\right)U_{m_z,m_r} \\
&= -\sqrt{r_{m_z,m_r}}g_{m_z,m_r}.
\end{aligned} \tag{15}$$

The electrical conductivities at the cell faces are calculated by the harmonic mean of the conductivities of neighboring cells [10]:

$$\begin{aligned}
\tilde{\sigma}_{m_z+1/2,m_r} &= \frac{2\sigma_{m_z+1,m_r}\sigma_{m_z,m_r}}{\sigma_{m_z+1,m_r} + \sigma_{m_z,m_r}}, \\
\tilde{\sigma}_{m_z,m_r+1/2} &= \frac{2\sigma_{m_z,m_r+1}\sigma_{m_z,m_r}}{\sigma_{m_z,m_r} + \sigma_{m_z,m_r+1}}.
\end{aligned} \tag{16}$$

Since the matrix must be positively defined for the Cholesky decomposition [7], the left and right sides of the system 15 are multiplied by -1.

The function g is considered a point source of direct current, that is, for a given pair of indices s_z, s_r and a given current value I_0 $g_{s_z,s_r} = I_0$, for all other indices $m_z \neq s_z \wedge m_r \neq s_r$ $g_{m_z,m_r} = 0$.

3.1 Numerical Simulation of the Lateral Logging Sounding

Let us consider a model of a medium with plane-parallel and coaxial-cylindrical section boundaries [3]. A vertical cylindrical borehole of radius r_c filled with drilling mud with resistance ρ_{wb} intersects a horizontal pack of layers of limited thickness h^i with resistances ρ_l^i, as well as unlimited (enclosing) formations with resistances $\overline{\rho_{adj}}$ and $\underline{\rho_{adj}}$. Drilling mud invasion zones (media with resistances ρ_{inv}^i) can be formed between the borehole and the layers. The distance from the axis of the borehole to the interface between the invasion zones and the layers is denoted as r_{inv}^i.

Let us describe the lateral logging sounding (LLS) process in an environment with plane-parallel and coaxial-cylindrical interface boundaries. Potential difference measurements are performed using several probes on the axis of the well with a step of Δd. The probe is defined by four points A, O, M, N (Fig. 1). A current source with a current strength of I is located at point A. The potential difference ΔU is measured between the electrodes M and N and conditionally refers to the point O, which is the middle of the segment MN.

Thus, the probe can be set by a pair of distances $L = AO$—the distance from the source to the measuring point ΔU and $l = MN$— the distance between the

receiver electrodes. Thus, each of the probes moves along the axis of the borehole in steps of Δd, measuring the potential difference ΔU at a distance of L from the current position of the source. The different positions of the source correspond to the different right sides for the SLAE 15. Thus, the numerical simulation of the LLS is reduced to solving a matrix equation of the form

$$\mathcal{A}X = B, \tag{17}$$

where the matrix $\mathcal{A}$—coincides with the SLAE matrix 15, B—is a matrix of right-hand sides, the columns of which have a form similar to that of the right-hand side of SLAE 15. The columns of the matrix X—specify the distribution of the electrical potential for the corresponding source. ΔU for a given probe position can be found by calculating the difference between the two corresponding elements of the solution column vector from X.

It is worth noting that the number of right-hand sides can be reduced since the exact position of the sources at the same current strength for different probes form the same right-hand sides. Therefore, ΔU can be obtained from a single solution vector. This optimization is implemented by grouping the measurements by the source coordinate after reading the parameters of the probes and measuring points ΔU.

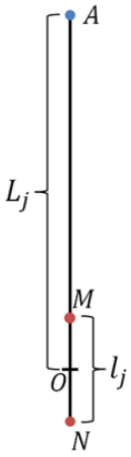

Fig. 1. The scheme of the probe

4 Inverse Problem of the Lateral Logging Sounding

Let us consider the medium with plane-parallel and coaxial-cylindrical interface boundaries, for which $r_c, \rho_{wb}, h^i, \overline{\rho_{adj}}, \underline{\rho_{adj}}$, as well as the number of layers N_l are

known. The parameters $\rho_l^i, \rho_{inv}^i, r_{inv}^i, 1 \leq i \leq N_l$ are considered unknown. They form the M data space.

In addition to the medium, N_z probes with known parameters $L_j, l_j, I_j, 1 \leq j \leq N_z$ are specified. For each probe, a set of points $x_{j,k}$ is set, where the potential difference $\Delta U_{j,k}, 1 \leq j \leq N_z, 1 \leq k \leq K_i$ is measured. The $\Delta U_{j,k}$ values are known. Denote $\Delta U_j = (\Delta U_{j,1}, ..., \Delta U_{j,K_j})$ is a vector of known potential differences for the probe j. This vector belongs to the dimension space of the j-th probe $\boldsymbol{\Delta U_j}$. Then we form the vector $\Delta U = (\Delta U_1 \cdot \Delta U_2 \cdot ... \cdot \Delta U_{N_z})$ by concatenating ΔU_k. ΔU belong to the dimension space $\boldsymbol{\Delta U}$.

Let us define the operator of the direct problem $F(m) = (F_1(m) \cdot F_2(m) \cdot ... \cdot F_{N_z}(m)) : M \to \boldsymbol{\Delta U}$, which maps the model m into $F(m) \in \boldsymbol{\Delta U}$, based on the solution of the 17 with m parameters. Then the inverse problem can be formulated as follows:

$$\begin{gathered} \text{Find } \tilde{m} \in M \mid E(\tilde{m}) = \min_{m \in M} E(m), \\ E(m) = \sum_{j=1}^{N_z} \frac{||F_j(m) - \Delta U_j||_j}{||\Delta U_j||_j}, \end{gathered} \tag{18}$$

where $||x||_j$—the 2-norm on the vector space $\boldsymbol{\Delta U_j}$.

It is worth noting that to reduce the number of calculations, especially in the case of an excess of data, the projective operators P_j can be introduced: $\boldsymbol{\Delta U_j} \to \boldsymbol{\Delta U}$. Examples of such operators include selecting each k-th value from the vector ΔU_j, selecting values of ΔU_j corresponding to a certain range of coordinates $x_{j,k} \in [a, b]$, and not using probe data (i.e., $P_j : \boldsymbol{\Delta U_j} \to 0$ for some j). The global sensitivity analysis method [8,13] can also be used to build a representative sample of $\Delta U_{j,k}$. In terms of projections, the formulation of the inverse problem is changing:

$$\begin{gathered} \text{Find } \tilde{m} \in M \mid E(\tilde{m}) = \min_{m \in M} E(m), \\ E(m) = \sum_{j=1, P_j \not\equiv 0}^{N_z} \frac{||P_j \circ F_j(m) - P_j(\Delta U_j)||_j}{||P_j(\Delta U_j)||_j}. \end{gathered} \tag{19}$$

5 Nelder-Mead Method

The Nelder-Mead simplex algorithm is a complete search method for solving the problem of unconditional optimization of $\min f(\boldsymbol{x})$, where $f : \mathbb{R}^n \to \mathbb{R}$—objective function [6,12]. A simplex of dimension n is a convex hull of $n + 1$ vertices. Denote the simplex with vertices $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{n+1}$ as Δ. The Nelder-Mead algorithm generates a sequence of simplices approximating the minimum point of the objective function. At each iteration, the vertices of the simplex $\{\boldsymbol{x}_j\}_{j=1}^{n+1}$ are sorted according to the values of the objective function: $f(\boldsymbol{x}_1) \leq f(\boldsymbol{x}_2) \leq ... \leq f(\boldsymbol{x}_{n+1})$. The vertex $\boldsymbol{x}_1$ is considered the best, and $\boldsymbol{x}_{n+1}$ is considered the worst.

The algorithm uses four operations: reflection, expansion, contraction, and shrink. Each of the operations is set by a corresponding parameter: α (reflection), β (expansion), γ (contraction), and δ (shrink). The standard algorithm parameters are the values $\{\alpha, \beta, \gamma, \delta\} = \{1, 2, \frac{1}{2}, \frac{1}{2}\}$. The center of gravity of the n best vertices is denoted as

$$\bar{\boldsymbol{x}} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i. \tag{20}$$

In these terms, one iteration of the Nelder-Mead algorithm has the following form:

1. **Order.** Calculate the value of the function f for each of the $n+1$ vertices Δ and sort the vertices in ascending order by the value of f.
2. **Reflection.** Calculate the reflected point $\boldsymbol{x}_r = \bar{\boldsymbol{x}} + \alpha(\bar{\boldsymbol{x}} - \boldsymbol{x}_{n+1})$. Find $f_r = f(\boldsymbol{x}_r)$. If $f(\boldsymbol{x}_1) \le f_r < f(\boldsymbol{x}_n)$, replace the point $\boldsymbol{x}_{n+1}$ with the point $\boldsymbol{x}_r$.
3. **Expansion.** If $f_r < f(\boldsymbol{x}_1)$, calculate the expanded point $\boldsymbol{x}_e = \bar{\boldsymbol{x}} + \beta(\boldsymbol{x}_r - \bar{\boldsymbol{x}})$ and find the value of $f_e = f(\boldsymbol{x}_e)$. If $f_e < f_r$, replace the point $\boldsymbol{x}_{n+1}$ with the point $\boldsymbol{x}_e$, otherwise replace the point $\boldsymbol{x}_{n+1}$ with the point $\boldsymbol{x}_r$.
4. **External contraction.** If $f(\boldsymbol{x}_n) \le f_r < f(\boldsymbol{x}_{n+1})$, calculate the point of external contraction $\boldsymbol{x}_{oc} = \bar{\boldsymbol{x}} + \gamma(\boldsymbol{x}_r - \bar{\boldsymbol{x}})$ and find the value of $f_{oc} = f(\boldsymbol{x}_{oc})$. If $f_{oc} < f_r$, replace the point $\boldsymbol{x}_{n+1}$ with the point $\boldsymbol{x}_{oc}$, otherwise go to step 6.
5. **Internal contraction.** If $f_r \ge f(\boldsymbol{x}_{n+1})$, calculate the point of internal contraction $\boldsymbol{x}_{ic} = \bar{\boldsymbol{x}} - \gamma(\boldsymbol{x}_r - \bar{\boldsymbol{x}})$ and find the value of $f_{ic} = f(\boldsymbol{x}_{ic})$. If $f_{ic} < f(\boldsymbol{x}_{n+1})$, replace the point $\boldsymbol{x}_{n+1}$ with the point $\boldsymbol{x}_{ic}$, otherwise go to step 6.
6. **Shrink.** For $2 \le i \le n+1$, calculate the new points $\boldsymbol{x}_i = \boldsymbol{x}_1 + \delta(\boldsymbol{x}_i - \boldsymbol{x}_1)$.

Justification of operations:

1. Reflection: if $\boldsymbol{x}_{n+1}$ is the worst point, then the minimum is assumed to be in the opposite direction.
2. Expansion: if the reflected point is a new minimum, then it is assumed that the approximation will improve even more if the point is moved further in the same direction.
3. Contraction: if reflection does not provide a good approximation, then the minimum is assumed to lie inside the simplex.
4. Shrink: if moving away from the worst point did not give a better approximation, all points are approaching the best.

The algorithm can be stopped when one of the following conditions is met:

1. $\max_{1 \le i,j \le n+1} |f_i - f_j| < \epsilon$—the change in the values of the objective function has become too small.
2. $\max_{1 \le i,j \le n+1} ||\boldsymbol{x}_i - \boldsymbol{x}_j|| < \delta$—the change in the points of the simplex has become too small.
3. The iteration limit has been reached: $num_it > it_max$.

4. The limit on the number of function calculations f has been reached: $num_f_eval > f_eval_max$.

The input data for the Nelder-Mead algorithm is an n-dimensional simplex, that is, $n + 1$ points of an n-dimensional space. This simplex can be chosen based on an approximation based on some a priori knowledge of the objective function. Either the simplex can be chosen randomly. If the constraints on the parameters are known, the Nelder-Mead algorithm can be modified to optimize the limited area. Consider the parameter x lying in the range $[a, b]$, and introduce the equivalent normalized parameter $\tilde{x} \in [0, 1]$. The one-to-one correspondence between them is given as $x = a + (b - a)\tilde{x}$. If during the execution of iterations of the Nelder-Mead algorithm (for example, during reflection or stretching), the parameter $\tilde{x}$ falls outside the range of $[0, 1]$, then its value is assumed to be 0, with $\tilde{x} < 0$ or 1, with $\tilde{x} > 1$.

Generalizing the proposed approach to the n-dimensional case, we assume optimization is performed in the n-dimensional unit cube. If a point goes beyond its limits, it is projected onto the area's boundary.

6 Results

The true data was obtained by simulation with ANSYS. The known parameters of the one-layer medium are presented in Table 1, and the parameters that should be recovered are shown in Table 2. Similarly, the known parameters of the three-layer medium are in the Table 3, and the unknown ones are in the Table 4.

To solve the inverse problem (19), the function $E(m)$ is minimized by the Nelder-Mead algorithm. Solving a series of direct problems that arise at each iteration of the algorithm is carried out using the NVIDIA cuDSS library.

Table 1. Known parameters of the one-layer medium

r_c (m)	ρ_c $(\Omega \cdot m)$	$\overline{\rho_{adj}}$ $(\Omega \cdot m)$	$\underline{\rho_{adj}}$ $(\Omega \cdot m)$	h (m)
0.1	1.7	5.0	5.0	3.0

Table 2. Unknown parameters of the one-layer medium

r_{inv} (m)	ρ_{inv} $(\Omega \cdot m)$	ρ_l $(\Omega \cdot m)$
0.3	30	10

Table 3. Known parameters of the three-layer medium

r_c (m)	ρ_c $(\Omega \cdot m)$	$\overline{\rho_{adj}}$ $(\Omega \cdot m)$	$\underline{\rho_{adj}}$ $(\Omega \cdot m)$	h^1 (m)	h^2 (m)	h^3 (m)
0.1	1.7	5.0	5.0	3.0	5.0	3.0

Table 4. Unknown parameters of the three-layer medium

Layer number	$r_{inv}(m)$	$\rho_{inv}(\Omega \cdot m)$	$\rho_l(\Omega \cdot m)$
1	0.2	30	10
2	0.5	40	7
3	0.3	30	15

Test calculations using the single-layer medium have shown that for different pairs of ρ_{inv}, r_{inv}, obtaining a fairly close result to the observed data is possible. This effect is called the U-equivalence of lateral logging sounding curves [3]:

$$\frac{\rho_{inv} - \rho_l}{\rho_c} \ln \frac{r_{inv}}{r_c} = U_{idem}. \tag{21}$$

Thus, a curve C appears in the parameter space, each point c_i of which is a minimum in some neighborhood, except points on the curve $B_\epsilon(c_i) \backslash C$. Using a minimization method based on calculating the gradient would lead to the algorithm hitting this curve and descending along it to a global minimum. However, the Nelder-Mead algorithm does not use the gradient of the objective function; moreover, due to discretization, r_{inv} can only take a finite number of values, and therefore, the curve splits into several unrelated local minima. When the Nelder-Mead algorithm "gets stuck" in one of these minimums, it can no longer find a global one. Nevertheless, the search for a global minimum is possible. The case of a single-layer environment is described in the Subsect. 6.1, and a three-layer environment is described in the Subsect. 6.2.

6.1 One-Layer Medium

The known parameters of the one-layer medium are presented in the Table 1. The list of probes is shown in the Table 5. When solving the inverse problem, only the values of $\Delta U_{j,k}$ corresponding to points from a given interval are used. For each probe, this interval is indicated in the last column of the Table 5. The $z = 0$ level is considered the middle of the layer.

Table 5. Parameters of probes

Probe number	$L(m)$	$l(m)$	Range
1	0.45	0.1	$[-2.5, 3.0]$
2	1.05	0.1	$[-2.5, 3.5]$
3	2.25	0.5	$[-3, 5.5]$
4	4.25	0.5	$[-4.5, 9.0]$
5	8.5	1.0	$[-5, 15.0]$

A priori estimates are known for unknown medium parameters:

$$\begin{aligned} 1.5r_c \leq & r_{inv} \leq 12r_c \\ 0.5\rho_c \leq & \rho_{inv} \leq 500\rho_c \\ 0.03125\rho_{inv} \leq & \rho_l \leq 4\rho_{inv}, \end{aligned} \tag{22}$$

Substituting the known parameters values, we get:

$$\begin{aligned} 0.15 \leq & r_{inv} \leq 1.2 \\ 0.85 \leq & \rho_{inv} \leq 850 \\ 0.0265625 \leq & \rho_l \leq 3400. \end{aligned} \tag{23}$$

The size of the computational domain is $[-45.5, 45.5] \times [0, 31.2]$ with spatial sampling steps $h_z = 0.05$ and $h_r = 0.025$. This size is chosen based on the following considerations. First, the minimum coordinate of the source is determined—-13.5, then the maximum coordinate of the receiver—15.5. These values define the initial range of the calculated area along the z axis. The initial range of r is determined by the maximum radius of the invasion zone—$[0, 1.2]$. Then, the area's boundaries are moved by 30 m in each direction, resulting in the size of the computing area $[-43.5, 45.5] \times [0, 31.2]$. For symmetry concerning the point $z = 0$, the range along the z axis is expanded to $[-45.5, 45.5]$. The computational grid size at a given sampling step is 1820×1248 nodes.

For each of the values of r_c from the range $[0.2, 1.2]$ in increments of 0.1 (Table 6), the Nelder-Meade algorithm restores the remaining two parameters. In the Table 6, the value of the objective function on the found solution is written in column E. The value of Ev_num denotes the number of solved direct problems, and It_num is the number of iterations of the Nelder-Mead algorithm.

Table 6. The results of restoring the parameters of the one-layer model

r_{inv}	ρ_{inv}	ρ_l	E	Ev_num	It_num
0.2	42.06	10.37	0.099	99	52
0.3	30.01	10.30	0.079	141	74
0.4	25.52	10.24	0.077	139	74
0.5	23.54	9.97	0.087	170	90
0.6	22.27	9.70	0.118	144	76
0.7	21.31	9.42	0.151	123	67
0.8	20.63	9.13	0.186	98	52
0.9	20.13	8.87	0.221	119	63
1.0	19.75	8.65	0.255	245	133
1.1	19.46	8.48	0.280	93	49
1.2	19.22	8.31	0.317	102	53

Thus, the minimum is reached for the values $r_c = 0.4(m)$, $\rho_{inv} = 25.52(\Omega \cdot m)$ $\rho_l = 10.24(\Omega \cdot m)$. Found parameter values for $r_c = 0.3$ almost coincides with the

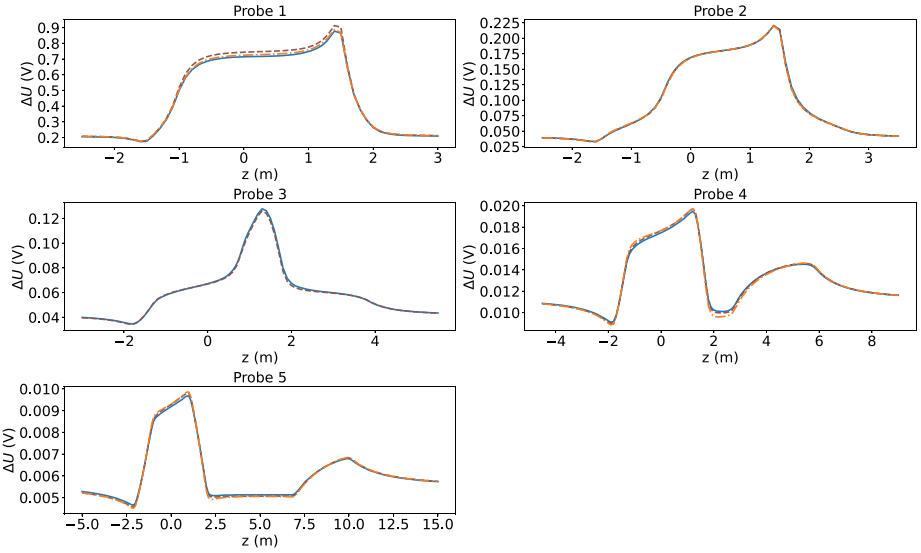

Fig. 2. LLS curves. Blue—true data. Orange—simulation result for optimal parameters. Brown - simulation result for true parameters (Color figure online)

true values. However, the value of the objective function for this set of parameters is higher than for $r_c = 0.4$. This effect is explained by the error in solving the direct problem (Fig. 2).

We also note that the found values of ρ_l are close enough to the actual value of ρ_l for any value of r_c. This observation will be used to restore the parameters of the three-layer environment.

On a device with an NVIDIA RTX 3090 Ti 24 GB graphics accelerator, the calculation took 70 min, of which 10 s was the symbolic factorization of the matrix A from the matrix Eq. 17, and the rest of the time was iterations of the Nelder-Meade algorithm. Thus, the average time to solve one direct problem without considering symbolic factorization is 2.86 s.

6.2 Three-Layer Medium

The known parameters of a single-layer medium are presented in the Table 3. The list of probes is shown in the Table 7. When solving the inverse problem, only the values of $\Delta U_{j,k}$ corresponding to every second point from the specified interval are used. For each probe, this interval is indicated in the last column of the Table 5. The $z = 0$ level is considered the middle of the second layer.

The design area configuration is set for the same reasons as in the 6.1 subsection. The size of the calculated area—$[-45.5, 45.5] \times [0, 31.2]$, spatial sampling steps—$h_z = 0.05$ and $h_r = 0.025$, computational grid size—1820×1248 nodes.

The task is solved in three stages. The first stage is to find the values of the layers resistivities ρ_l^i using an observation from the Sect. 6.1. It is assumed

Table 7. Parameters of probes

Probe number	$L(m)$	$l(m)$	Range
1	0.45	0.1	$[-10.6, 9.2]$
2	1.05	0.1	$[-10.0, 9.8]$
3	2.25	0.5	$[-8.8, 11.0]$
4	4.25	0.5	$[-6.8, 13.0]$
5	8.5	1.0	$[-7.0, 15.0]$

that even for far from true values of ρ^i_{inv} and r^i_{inv}, found by the Nelder-Mead values of ρ^i_l will be close to the true values. Therefore, random values of the parameters $r^i_{inv}, 1 \le i \le 3$ are selected from the range $[0.15, 1.2]$. The remaining six parameters are found for the selected values by the Nelder-Mead algorithm. This operation is repeated 128 times.

These calculations were launched on a node with 8 NVIDIA A100 80 GB GPUs. Using MPI technology, 128 runs of the Nelder-Meade algorithm were divided between 8 GPUs (16 runs each). The computation time is 10 h.

As a result of each run, nine parameters and the value of the objective function E were obtained. Parameter sets with a $E > 1$ are filtered out and do not participate in further parameter determination. ρ^i_l are found as a weighted average of the values of ρ^i_l among the remaining parameter sets in the sample. The weight of the set is defined as $\frac{1}{E}$. Thus, the resistivity values of the layers $\rho^1_l = 10.21, \rho^2_l = 7.10, \rho^3_l = 15.18$ are established. These values are used in the second step to restore the remaining parameters.

Restoration of the remaining parameters is similar to the process described in Sect. 6.1. At the second stage, for the given values $\rho^1_l = 10.21, \rho^2_l = 7.10, \rho^3_l = 15.18$, the parameters $r^i_{inv}, 1 \le i \le 3$ vary in the range $[0.2, 1.2]$ in 0.2 increments. For each triple r^i_{inv}, the Nelder-Mead algorithm restores the parameters $\rho^i_{inv}, 1 \le i \le 3$. $6 \times 6 \times 6 = 216$ runs of the algorithm take 4.5 h when running in parallel on 8 NVIDIA A100 80 GB GPUs. Note that the data from the fifth probe is not used in this case (i.e., $P_5 \equiv 0$); this is because, the measurements of the fifth probe primarily depend on the resistivity of the layers that have already been determined. At the same time, the positions of the source for the fifth probe do not coincide with the positions of the sources for all other probes, therefore, the exclusion of this probe from the calculation reduces the number of columns in the matrices X, B (Eq. 17).

The smallest error values are achieved for the values r^i_{inv} lying at the vertices of the cube $[0.2, 0.4] \times [0.4, 0.6] \times [0.2, 0.4]$. The values of the objective function and the corresponding reconstructed parameters ρ^i_{inv} are presented in the Table 8.

The minimum is reached for the parameter values $r^1_{inv} = 0.2, r^2_{inv} = 0.6, r^3_{inv} = 0.4$. Since the objective function is the sum of the relative errors for each of the four probes, the value $E = 0.0343$ means that, on average, the relative error for each probe does not exceed 1% (Fig. 3).

Table 8. The smallest values of the objective function E and the corresponding medium parameters

$r^1_{inv}(m)$	$r^2_{inv}(m)$	$r^2_{inv}(m)$	E	$\rho^1_{inv}(\Omega \cdot m)$	$\rho^2_{inv}(\Omega \cdot m)$	$\rho^3_{inv}(\Omega \cdot m)$
0.2	0.4	0.2	0.0995	29.59	45.57	39.40
0.2	0.4	0.4	0.0979	29.58	45.57	26.73
0.2	0.6	0.2	0.0531	29.00	36.49	38.14
0.2	0.6	0.4	0.0343	28.95	36.44	26.30
0.4	0.4	0.2	0.1082	19.34	45.61	39.41
0.4	0.4	0.4	0.1053	19.36	45.60	26.75
0.4	0.6	0.2	0.0558	19.28	36.44	38.26
0.4	0.6	0.4	0.0375	19.33	36.39	26.30

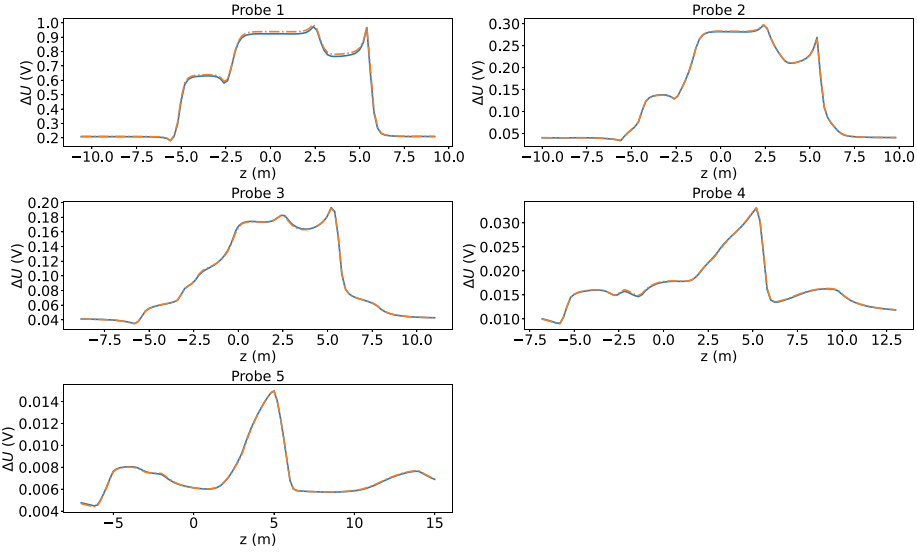

Fig. 3. LLS curves. Blue— (Color figure online)true data. Orange—simulation result for optimal parameters.

In the third stage, the solution is refined on a smaller grid with spatial sampling steps $h_z = 0.025$ and $h_r = 0.025$. Similarly to the previous stage, the parameters $r^1_{inv}, r^2_{inv}, r^3_{inv}$ vary in increments of 0.05 in the ranges $[0.2, 0.4], [0.5, 0.6], [0.2, 0.4]$ respectively. That is, the parameters are refined inside the found cube with the lowest values of the objective function at the vertices (the interval $[0.4, 0.6]$ for the parameter r^2_{inv} was reduced to $[0.5, 0.6]$ based on the fact that the value of the objective function on the right edge of the interval is significantly lower than on the left). To reduce the calculation time, instead of a priori estimates of the range ρ^i_{inv}, new ranges are used based

on the maximum and minimum values of ρ^i_{inv} from the Table 8. Thus, the search interval ρ^1_{inv} is considered to be the interval $[19, 30]$, for ρ^2_{inv}—$[36, 46]$, for ρ^3_{inv}—$[26, 40]$. Reducing the search interval significantly speeds up the Nelder-Mead algorithm running time (up to 10 min), which allows us to perform the entire calculation using 8 NVIDIA A100 80 GB GPUs in 1.5 h. The minimum error $E = 0.0340$ is reached for the set of parameters in Table 9.

Table 9. The recovered parameters of the three-layer medium

Layer number	$r_{inv}(m)$	$\rho_{inv}(\Omega \cdot m)$	$\rho_l(\Omega \cdot m)$
1	0.2	29.44	10.21
2	0.55	38.24	7.10
3	0.3	29.8	15.18

7 Conclusion

The algorithm has been developed to solve the inverse problem of lateral logging sounding using the Nelder-Mead method (deformable simplex method). Since calculating the objective function during the solution of the inverse problem requires solving a series of direct problems, the core of the complex is the solver of the direct problem of electric field potential distribution in general and the LLS problem in particular. The direct problem solver is constructed using the finite difference method. The resulting SLAE has a symmetric matrix, making it possible to use a Cholesky decomposition. Moreover, due to the constancy of the sparse matrix template, the time-consuming symbolic factorization operation is performed only once. The key advantage in solution speed is the direct SLAE solver from the NVIDIA cuDSS library. Even though cuDSS is inferior to the solvers implemented for the CPU (Intel MKL PARDISO) in terms of the execution time of symbolic factorization, since this operation is difficult to parallelize, cuDSS has a considerable advantage at the stage of calculating the solution based on the decomposed matrix.

The parameters of the medium with plane-parallel and coaxial-cylindrical interfaces were restored for one layer with drilling mud invasion and three layers (also with drilling mud invasion) using the developed algorithm. The calculation for a single-layer environment took 70 min on a personal computer with a modern NVIDIA RTX 3090 Ti 24 GB GPU. Restoring the parameters of a three-layer environment is a much more time-consuming task. The total calculation time using 8 NVIDIA A100 80 GB GPUs is 14 h. This time can be significantly reduced. At the second stage, the variation of the invasion radii over the entire range can be replaced by searching the octants of the cube $[0.15, 1.2] \times [0.15, 1.2] \times [0.15, 1.2]$, which would lead to the final area $[0.2, 0.4] \times [0.4, 0.6] \times [0.2, 0.4]$ not for 216, but for 45 solutions to the inverse

problem, moreover, restoring the parameters ρ^i_{inv} for boundary values of r^i_{inv} from the range $[0.15, 1.2] \times [0.15, 1.2] \times [0.15, 1.2]$ would allow us to get a significantly smaller search range for ρ^i_{inv} (since this was done at the third stage of the solution). Solutions to the inverse problem with random parameters of the invasion zones were used to search for layers resistivities. Since the layers resistivities are being restored reasonably well, regardless of the parameters of the penetration zones, the number of launches can be significantly reduced. The proposed optimizations will reduce the total calculation time for a three-layer environment to 3–4 h (when using 8 NVIDIA A100 80 GB GPUs).

Another option to reduce the calculation time is to use gradient minimization methods. Even though the Nelder-Mead algorithm has been successfully applied to solving the inverse problem, the minimum configuration in the parameter space suggests that gradient methods can significantly increase search speed. However, gradient minimization methods significantly complicate the algorithm.

The accuracy of parameters recovery is relatively high (especially for layers electrical resistivity). Nevertheless, the recovered values of drilling mud invasion radii may be overestimated (the error value is up to one borehole radius), which in turn also introduces a small error in the electrical resistivity of the invasion zone. This problem arises due to the use of a coarse grid and the insufficient removal of boundary conditions. Both problems can be solved using optimal grid theory. The construction of an optimal grid will significantly improve the processing of boundary conditions and reduce the overall size of the grid, which in turn will allow using a smaller step of spatial discretization.

Acknowledgments. The research was supported by the Russian Science Foundation grant no. 19-77-20004-П.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. NVIDIA cuDSS. https://developer.nvidia.com/cudss. Accessed 03 Mar 2025
2. Belonosov, M.A., Kostov, C., Reshetova, G.V., Soloviev, S.A., Tcheverda, V.A.: Parallel numerical simulation of seismic waves propagation with intel math kernel library. In: Manninen, P., Öster, P. (eds.) Applied Parallel and Scientific Computing, pp. 153–167. Springer Berlin Heidelberg
3. Dahnov, V.: Electric and Magnetic Methods of Borehole Research. Nedra, Moscow (1981). in Russian
4. Druskin, V., Knizhnermann, L.: Gaussian spectral rules for the three-point second differences: I a two-point positive definite problem in a semi-infinite domain. SIAM J. Numer. Anal. **36**, 442–464 (1999)
5. Druskin, V.: On the uniqueness of the solution of the inverse problem of electrical exploration and electrical wiring for piecewise constant conductivities. News of the USSR Academy of Sciences. Phys. Earth **1** (1982), in Russian
6. Gao, F., Han, L.: Implementing the Nelder-mead simplex algorithm with adaptive parameters. Comput. Optim. Appl. **51**(1), 259–277 (2012)

7. Gilbert, J., Schreiber, R.: Highly parallel sparse Cholesky factorization. SIAM J. Sci. Stat. Comput. **13** (1990)
8. Gondyul, E., Gadylshin, K., Lisitsa, V., Vishnevsky, D.: Reducing numerical dispersion in pseudo-3D space and constructing training dataset. In: Gervasi, O., Murgante, B., Garau, C., Taniar, D., C. Rocha, A.M.A., Faginas Lago, M.N. (eds.) Computational Science and Its Applications – ICCSA 2024 Workshops, pp. 352–366. Springer Nature Switzerland
9. Ingerman, D., Druskin, V., Knizhnerman, L.: Optimal finite difference grids and rational approximations of the square root i. elliptic problems. Commun. Pure Appl. Math. **53**, 1039–1066 (2000)
10. Khachkova, T., Lisitsa, V., Reshetova, G., Tcheverda, V.: GPU-based algorithm for evaluating the electrical resistivity of digital rocks. Comput. Math. Appl. **82**, 200–211 (2021)
11. Kulikov, I., et al.: Numerical modeling of hydrodynamic turbulence with self-gravity on intel Xeon Phi KNL. In: Sokolinsky, L., Zymbler, M. (eds.) Parallel Computational Technologies, pp. 309–322. Springer International Publishing
12. Nelder, J.A., Mead, R.: A simplex method for function minimization. Comput. J. **7**(4), 308–313 (1965)
13. Sobol, I.M.: Global sensitivity indices for nonlinear mathematical models and their monte Carlo estimates. Math. Comput. Simul. **55**(1), 271–280 (2001)
14. Solovyev, S.: Application of the low-rank approximation technique in the gauss elimination method for sparse linear systems. Num. Methods Programm. **3**, 441–460 (2014)
15. Solovyev, S., Lisitsa, V.: Field-split iterative solver vs direct one for quasi-static Biot equation. In: Voevodin, V., Sobolev, S., Yakobovskiy, M., Shagaliev, R. (eds.) Supercomputing, pp. 86–99. Springer Nature Switzerland, Cham (2023)