Application of Automatic Differentiation and Optimization with PDE Constraints to Forward and Inverse Problems

Nikolay Yavich [1] **Alexander Ryabov** [1] **Viacheslav Naumov** [1] **Sayan Ranu** [2] **N. M. Anoop Krishnan** [2]
Evgeny Burnaev [1,3] **Vladimir Vanovskiy** [1]

[1]*Skolkovo Institute of Science and Technology, Artificial Intelligence Center, Moscow, 121205, Russia.* [2]*Yardi School of Artificial Intelligence, Indian Institute of Technology Delhi, Hauz Khas, New Delhi, 110016, India.* [3]*Autonomous Non-Profit Organization Artificial Intelligence Research Institute (AIRI), Moscow, 105064, Russia.* Correspondence to: Nikolay Yavich n.yavich@skoltech.ru.

1. Introduction

Automatic differentiation (AD) is an established tool to compute gradients of functions, [1, 2, 3]. However, AD libraries typically allow an *explicit* relation between variables and functions, thus being suitable for optimization problems stemming from explicit computational schemes, see e.g. [4]. Some AD libraries support *implicit* relations between variables and functions, e.g. [5], however, their capabilities for handling sparse linear solvers are currently quite limited. The JAX [6] solver is based on QR matrix factorization, which is suitable for small and moderate-size problems, but for larger problems it is too slow and memory-consuming. Analogous PyTorch functionality [7, 8] is still under development.

This work extends earlier contributions on differentiable implicit solvers, e.g., [9, 10, 11], with a particular focus on the efficient treatment of constraints arising from large sparse systems of equations. It enables the development of a custom autograd function, which we illustrate through its application to optimization problems arising in self-supervised grid coarsening for nonlinear forward problems. We also present examples involving the solution of inverse problems for elliptic and parabolic PDEs. Our examples are motivated by applications in geoscience and involve the finite-volume method on Voronoi meshes [12].

2. Methodology

2.1 Problem Setup

Optimization in both static and evolutionary forward and inverse problems can be written in the following form,

$$L(\theta) = \sum_{k=1}^{n} \|R \cdot u^k(\theta) - u^k_{obs}\|^2 + S(\theta), \tag{1}$$

where $\theta \in \mathbb{R}^M$ is an unknown parameter requiring estimation, n is the number of temporal measurements (1 for static problems), $u^k(\theta) \in \mathbb{R}^N$ is the modelled variable, $R : \mathbb{R}^N \to \mathbb{R}^m$ is the measurement operator, $u^k_{obs} \in \mathbb{R}^m$ is measured data at the temporal point k and m meurement points, and $S(\theta)$ is a stabilizer.

Both static problems and evolutionary problems involving implicit time-stepping require solution of a large sparse system of equations,

$$A(\theta) \cdot u^k(\theta) = b^k(\theta), \quad 1 \le k \le n, \tag{2}$$

which serves as a constraint in (1).

A naive approach might be to eliminate $u^k(\theta)$ and obtain an unconstrained optimization problem,

$$L(\theta) = \sum_{k=1}^{n} \|R \cdot A(\theta)^{-1} \cdot b^k(\theta) - u^k_{obs}\|^2 + S(\theta), \tag{3}$$

However, this is not feasible because $A(\theta)^{-1}$ is a large dense matrix not suitable for manipulations including backward propagation.

2.2 Forward and Backward Propagation

We assume that the system matrix in (2) is symmetric and positive-definite, which is typically the case for reaction-diffusion problems. More general cases could be treated similarly. The matrix is large and sparse, therefore special formats are used to store only its nonzero entries.

During the forward propagation stage, we have to solve (2), which can be formally written as (omitting θ and superscript k),

$$u = A^{-1} \cdot b. \tag{4}$$

During the backward propagation stage, we have to compute the gradients of u with respect to A and b. Straightforward manipulations result in the following,

$$\nabla_b u = A^{-1} \quad \text{and} \quad \nabla_A u = -A^{-1} \cdot P \cdot u, \tag{5}$$

where P is a perturbation matrix, i.e. a matrix of zeros with just a single entry of 1.

Now, these expressions can used to compute the gradient of the loss L with respect to A and b efficiently. Notice

$$\nabla_b L = \nabla_u L \cdot \nabla_b u, \quad \text{and} \quad \nabla_A L = \nabla_u L \cdot \nabla_A u, \tag{6}$$

where $\nabla_u L$ is the gradient of the loss with respect to the solution u, propagated back from subsequent stages in the computational graph.

Substituting (5) into (6), we obtain,

$$\nabla_b L = \nabla_u L \cdot A^{-1}, \tag{7}$$

and

$$\nabla_A L = -\nabla_u L \cdot A^{-1} \cdot P \cdot u = -\nabla_b L \cdot P \cdot u. \tag{8}$$

The last expressions imply the following computational algorithm for the backward propagation stage:

1. Compute $\nabla_b L$ with (7), which is equivalent to solving of a single linear system.
2. Compute $\nabla_A L$ by multiplying the respective entries of $\nabla_b L$ and $-u$ by each other.

Therefore, computational complexity of the backward propagation stage is equivalent to the complexity solution of a single equation system. Since A is stored in a sparse matrix format, (8) is reduced to differentiating only non-zero entries.

3. Experiments

3.1 Self-supervised Grid Coarsening for Nonlinear Parabolic Equation

Consider the nonlinear parabolic equation governing compressible fluid flow in porous media,

$$\frac{\partial}{\partial t}(\phi(u)\rho(u)) - \mathrm{div}\left(\rho(u)\frac{K}{\mu}\nabla u\right) = q,\ 0 < t < T, (x, y) \in V, \tag{9}$$

Here, $K(x, y)$ is the medium permeability, $q(x, y)$ is the source. Density, ρ, and porosity, ϕ, are known functions of the pressure, $u(x, y)$. Viscosity, μ, is assumed constant.

Given modelled pressure time series at several points, resulting from the solution to (9) on a fine grid, denoted u^k_{obs}, we search for a coarser grid that would yield a solution u^k to (9) close to u^k_{obs}. This problem is formulated as minimization of (1) with θ representing the locations of the centers of the coarser grid cells.

We implemented a semi-implicit temporal discretization for numerical solution of (9) combined with the finite volume discretization on a Voronoi grid. We further linked a differentiable Voronoi mesher [13], to implement an automatically differential pipeline. The results and other details are shown in Fig A1.

3.2 Coefficient Inverse Problem

Estimation of a poorly known partial differential equation coefficients describing physical properties of some heterogeneous medium is of a paramount importance in many applications, especially in geoscience and geophysics.

We look at the problem which is known as *history matching* in petroleum engineering, [14]. We estimate the permeability, $K(x, y)$, by minimizing (1) with

$$\theta = K \quad \text{and} \quad S(K) = \beta \int_V |\nabla K|^2 dxdy, \tag{10}$$

constrained with

$$\frac{\partial u}{\partial t} - \mathrm{div}(K\nabla u) = f, \quad 0 < t < T, \quad (x, y) \in V, \tag{11}$$

and $K(x, y) > 0$. Equation (11) is discretized with the backward Euler scheme.

In our experiment, pressure was measured at two spatial locations, $m = 2$ and at 100 temporal points, $\tau = 0.1s$, $T = 10s$, Fig. A2 a) and c). The results and other details are shown in Fig. A2

3.3 Inverse Source Problem

Another type of inverse problem that is commonly encountered in geosciences and geophysics is to estimate the right-hand side of a partial differential equation. We will proceed with a setting that is referred to as the *self-potential method*, [15]. The goal is to estimate direct current source density based on the observations of the electric potential,

$$-\mathrm{div}(\sigma\nabla u) = f, \quad (x, y) \in V, \tag{12}$$

where $\sigma(x, y)$ is known electrical conductivity, $u(x, y)$ is the electric potential, $f(x, y)$ is the unknown source density. The source density is estimated by minimizing (1) with $n = 1$,

$$\theta = f \quad \text{and} \quad S(f) = \int_V (\gamma|\kappa\nabla f|^2 + \delta f^2) dxdy, \tag{13}$$

where the stabilizer promotes smoothness with κ being a variable coefficient needed to compensate decrease of sensitivity with depth; the last term is a stabilizer promoting sources with smaller energy; γ and δ are trade-off scalar parameters.

The results and other details are shown in Fig A3. The results also include a comparison versus those of the PINN implemented in the DeepXDE package [16].

4. Conclusions

In this work, we integrated the finite-volume method and implicit time-stepping to develop a fully differentiable modeling pipeline, applying it to both forward and inverse problems of geoscience. Our PyTorch-based pipeline is flexible and easy to implement, enabling the integration of new, fast solvers for large sparse linear systems.

For comparison, [17, 18] presented results on accelerating forward modeling using graph coarsening, which is somewhat similar to our approach. However, their method encountered significant time-step limitations due to the use of an explicit scheme, a common drawback of such approaches [19].

Our learnable coarsening example demonstrated that the obtained coarse grid maintains high fidelity to the original simulation while significantly accelerating the modeling process reducing grid size in 10 times (meaning at least 10x speedup). With implicit time-stepping we easily tackled long-term simulations in highly heterogeneous media.

We should emphasize equivalence of the presented methodology and the notions of direct and adjoint states in constrained optimization. The duality between forward and backward modes in AD reflects the duality between direct and adjoint formulations in constrained optimization, with both frameworks exploiting structural properties of the underlying equations to reduce computational cost in large-scale problems, between backpropagation and the adjoint state method, [20, 21].

Acknowledgments

The study was supported by the Russian Science Foundation grant No. 24-41-02035, https://rscf.ru/en/project/24-41-02035/

References

[1] Uwe Naumann. *The art of differentiating computer programs : an introduction to algorithmic differentiation*. SIAM, 1st edition, 2012.

[2] Michael Bartholomew-Biggs, Steven Brown, Bruce Christianson, and Laurence Dixon. Automatic differentiation of algorithms. *Journal of Computational and Applied Mathematics*, 124(1):171–190, 2000. Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations.

[3] Atılım Günes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *J. Mach. Learn. Res.*, 18(1):5595–5637, January 2017.

[4] Weiqiang Zhu, Kailai Xu, Eric Darve, and Gregory C. Beroza. A general approach to seismic inversion with automatic differentiation. *Computers and Geosciences*, 151:104751, June 2021.

[5] JAX: High performance array computing. https://jax.readthedocs.io/en/latest/index.html. Accessed: 2024-10-01.

[6] JAX: A sparse direct solver using qr factorization. https://jax.readthedocs.io/en/latest/_autosummary/jax.experimental.sparse.linalg.spsolve.html. Accessed: 2025-01-29.

[7] PyTorch: Compute the solution of a square system. https://pytorch.org/docs/stable/generated/torch.linalg.solve.html. Accessed: 2025-01-30.

[8] torch.sparse. https://pytorch.org/docs/stable/sparse.html. Accessed: 2025-01-30.

[9] François Fraysse and Richard Saurel. Automatic differentiation using operator overloading (adoo) for implicit resolution of hyperbolic single phase and two-phase flow models. *Journal of Computational Physics*, 399:108942, December 2019.

[10] Dongzhuo Li, Kailai Xu, Jerry M. Harris, and Eric Darve. Coupled time-lapse full-waveform inversion for subsurface flow problems using intrusive automatic differentiation. *Water Resources Research*, 56(8), August 2020.

[11] Lian Liu, Bo Yang, Yi Zhang, Yixian Xu, Zhong Peng, and Dikun Yang. Calculating sensitivity or gradient for geophysical inverse problems using automatic and implicit differentiation. *Computers and Geosciences*, 193:105736, November 2024.

[12] Robert Eymard, Thierry Gallouët, and Raphaèle Herbin. *Finite volume methods*, page 713–1018. Elsevier, 2000.

[13] Sergei Shumilin, Alexander Ryabov, Serguei Barannikov, Evgeny Burnaev, and Vladimir Vanovskii. A method for auto-differentiation of the voronoi tessellation, 2024.

[14] S. Srinivasan and J.Y. Leung. *Petroleum Reservoir Modeling and Simulation: Geology, Geostatistics, and Performance Prediction*. McGraw Hill LLC, 2022.

[15] André Revil and Abderrahim Jardani. *Forward and inverse modeling*, page 110–153. Cambridge University Press, 2013.

[16] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

[17] Sergei Shumilin, Alexander Ryabov, Nikolay Yavich, Evgeny Burnaev, and Vladimir Vanovskiy. Self-supervised coarsening of unstructured grid with automatic differentiation. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 45315–45328. PMLR, 21–27 Jul 2024.

[18] Alexander Ryabov, Viacheslav Naumov, Sergei Shumilin, Nikolay Yavich, Sayan Ranu, N. M. Anoop Krishnan, Evgeny Burnaev, and Vladimir Vanovskiy. Learnable stability-aware computational grid coarsening for accelerating physics simulations. *Machine Learning: Science and Technology*, 7(1), 2026.

[19] Nikolay Yavich, Evgeny Burnaev, and Vladimir Vanovskiy. Stability of the dufort–frankel scheme on unstructured grids. *Computation*, 13(10):246, October 2025.

[20] Pierre Baldi. *Gradient descent learning algorithms: a unified perspective*, page 509–541. L. Erlbaum Associates Inc., USA, 1995.

[21] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, Second Edition*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2008.

Appendix A. Illustrations

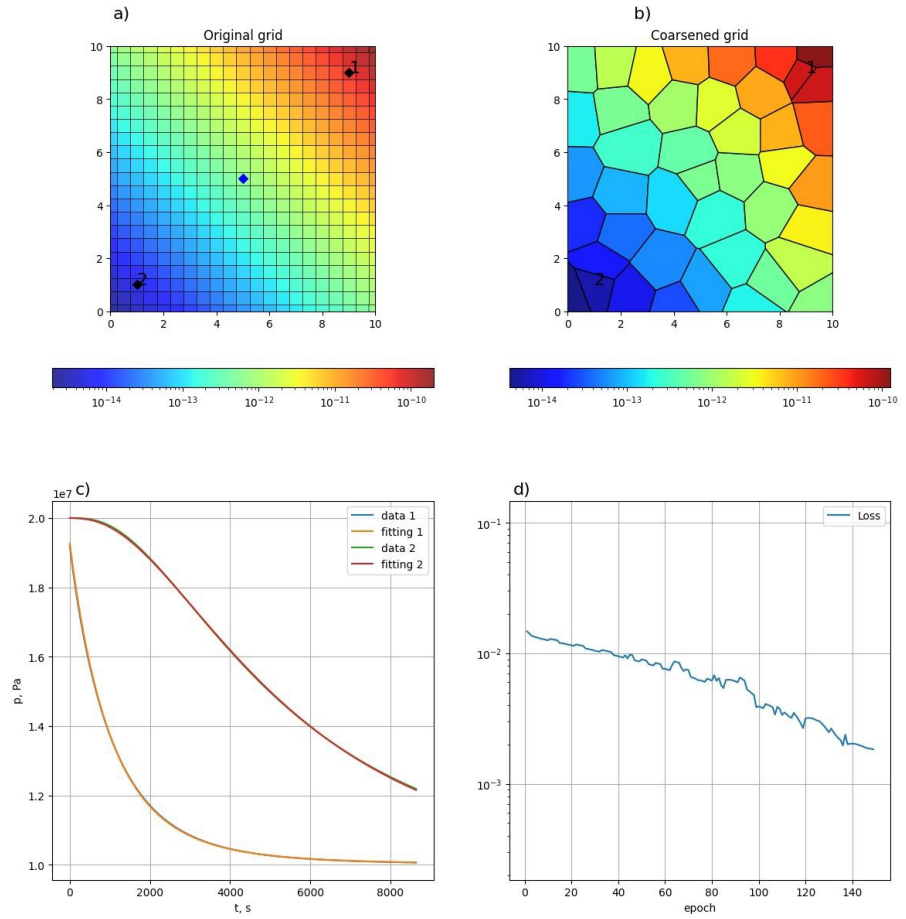

Fig. A1: Computational grid coarsening for the nonlinear parabolic equation, (9). The following dependencies were assumed, $\rho(u) = \rho_r e^{c_\rho(u-u_r)}$, $\phi(u) = \phi_r e^{c_\phi(u-u_r)}$. Initial pressure $u_0 = 2 \cdot 10^7$ Pa, well pressure in well $u_w = 10^7$ Pa, viscosity $\mu = 0.005$, $c_\rho = 10^{-8}$, $\rho_r = 850.0$, $c_\phi = 10^{-11}$, $\phi_r = 0.3$, $u_r = 2 \cdot 10^7$ Pa. Permeability K varied from 10^{-14} to 10^{-10}. a) Original grid, color indicates permeability, source marked with blued diamond, and two measurement points (1 and 2) marked as black diamonds; b) coarsened grid; c) data at two measurement points, modelled with the original and coarsened grids; d) loss function during optimization.

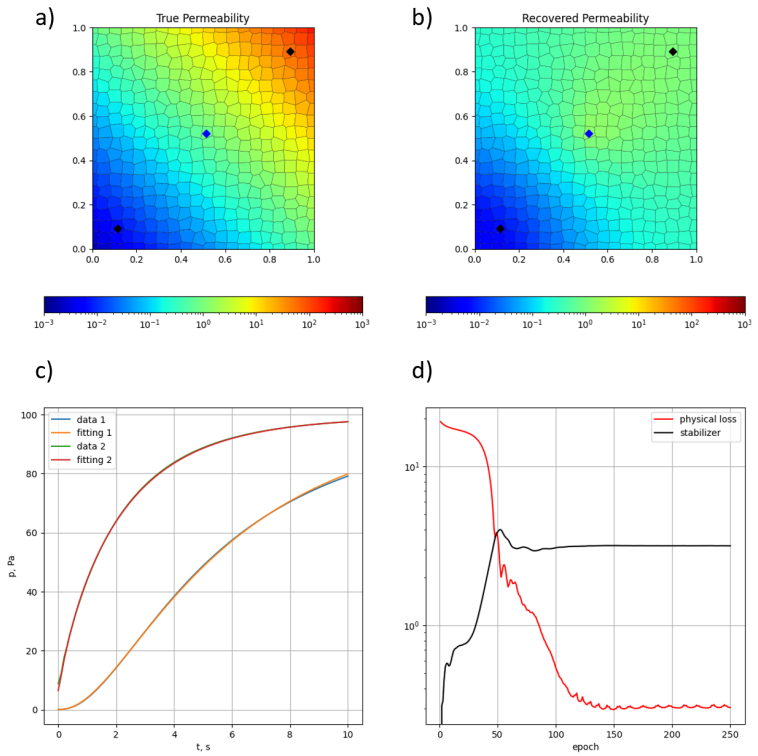

Fig. A2: Solution of the coefficient inverse problem. a) True permeability, source (blue diamond) and two measurement points (black diamonds); b) recovered permeability; c) pressure at two measurement points (data and prediction/fitting); d) loss function during 250 epochs of minimization. The Voronoi grid was formed from a graph of 400 randomly perturbed vertices. We applied the Adam optimizer to (1) stating with $K = 1$ and continued for 250 epochs. Decent data fit was archived after 150 epochs, d). The recovered permeability, b), has main features of the true one: the lower left area of lower permeability, the upper right area of higher permeability. We reduced the size of the grid from 400 to 40 cells and obtained a fairly good match of pressure time series.

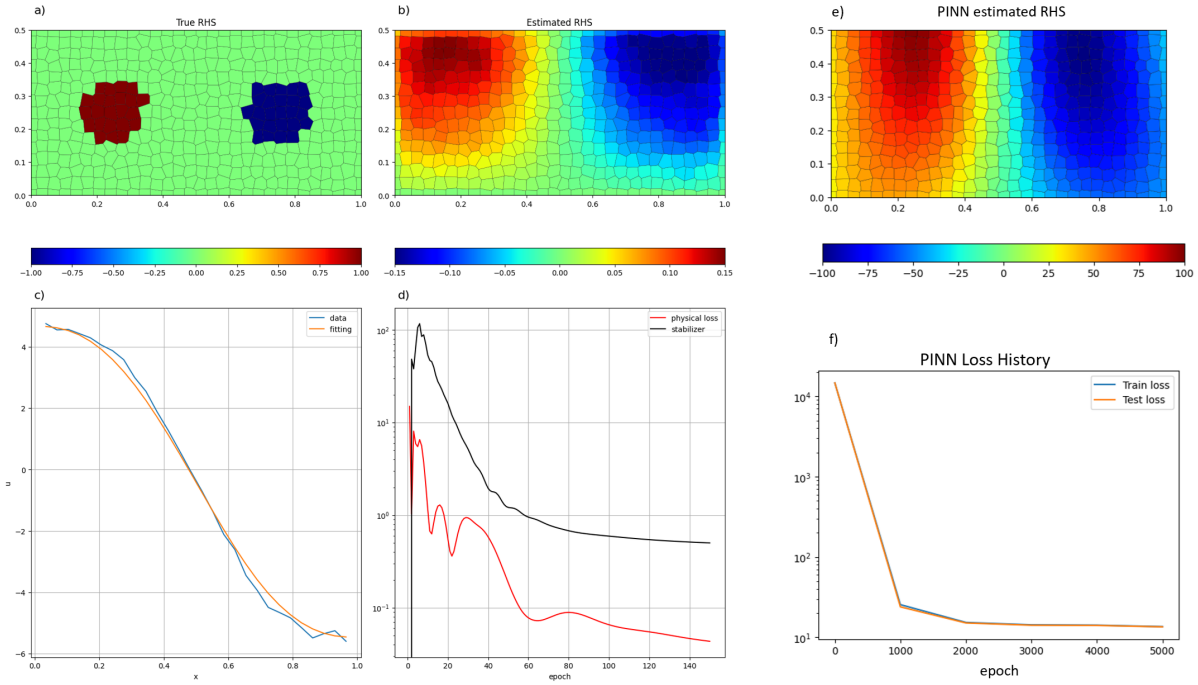

Fig. A3: Solution of the source inverse problem. a) True right-hand side f and measurement points (black diamonds); magenta diamonds indicate the side where Dirichlet boundary condition was applied; Neumann boundary condition was applied on the other three sides; b) recovered right-hand side; c) observed data at measurement points with with noise added, u_{obs}, (data) and prediction, u_{mod}, (fitting); d) loss function during 100 epochs of minimization: physical loss (first term in (13)) and stabilizers (later terms in (13)). e) recovered right-hand side f using the PINN; f) PINN loss function during 5000 epochs of minimization. For PINN solution, both the unknown solution u and the right-hand side f were parametrized with a feedforward neural network. For both our approach and PINN we see a very similar pattern in b) and e). The main difference between our result and that of PINN is in magnitudes. We also note that the convergence of the optimizer in our method was significantly faster, as shown in d) and f).