arXiv:2610.27888v1 [cs.AI] 20 Oct 2026

Automatic Differentiation with Constraints in 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], and Vladimir Vanovskiy[1]

[1]Artificial Intelligence Center, Skolkovo Institute of Science and Technology, 121205 Moscow, Russia
[2]Yardi School of Artificial Intelligence, Indian Institute of Technology Delhi, Hauz Khas, New Delhi, India 110016
[3]Learnable Intelligence Group, Autonomous Non-Profit Organization Artificial Intelligence Research Institute (AIRI), 105064 Moscow, Russia

Abstract

Approximate solutions of partial differential equations (PDEs) on unstructured grids can be represented on a graph. Numerical optimization involving stiff evolutionary and static PDEs often requires solving a large system of equations, which can be treated as a constraint. In this work, we present a fully differentiable graph-based implicit solver that incorporates this constraint. We evaluate its performance across four key tasks: a) linear and nonlinear forward modeling of evolutionary and static problems, b) solving inverse problems to estimate equation coefficients, c) solving inverse problems to estimate the right-hand side, and d) learnable graph coarsening to accelerate forward modeling. The stability and differentiability of our solver enable significant reductions in the complexity of forward modeling while efficiently addressing related inverse problems. These capabilities make it a powerful tool for different physics and engineering applications, in particular for geophysics, where inverse problems are commonly solved in order to reduce the model uncertainty.

Keywords: differentiable solvers, implicit time-stepping, grid coarsening, inverse problems

*Corresponding author, n.yavich@skoltech.ru

1 Introduction

2 Introduction

In recent years, graph neural networks have emerged as a powerful tool for handling data on unstructured grids [12, 18, 33, 34]. Using message-passing mechanisms, GNNs are highly effective in capturing local dependencies and updating node features through their connections. This capability has sparked increasing interest in applying GNNs to PDE solvers, particularly in problems where the data can be represented as a graph, such as Voronoi grids and finite-volume discretizations.

The finite-volume (FV) method is a well-established numerical technique for solving PDEs, especially on unstructured grids, as it ensures local conservation properties. Integrating FV methods with GNN-based solvers provides a structured, grid-free representation of the domain, while preserving a rigorous framework for solving governing equations. In this context, the development of differentiable solvers is an especially promising research direction.

One key advancement explored in this work is the application of automatic differentiation (AD) [24] within GNN-based solvers, particularly for tasks such as graph coarsening and solving inverse problems. AD has proven to be an invaluable tool for efficient gradient-based optimization. The use of AD in inverse problems involving hyperbolic equations was demonstrated in [37].

In our work, we focus on parabolic partial differential equations, which are crucial in various applications, including engineering and the oil and gas industry. However, the high stiffness of the resulting evolutionary problems necessitates the use of implicit time-stepping schemes due to their stability properties. Additionally, we address elliptic partial differential equations, where a distinctive challenge lies in solving large and sparse linear systems.

Automatic differentiation (AD), [3, 4], is an established tool to form gradients of functions. Although some AD libraries support implicit relations between variables, e.g., [14], their capabilities for handling sparse linear solvers are currently quite limited. The JAX [15] solver is based on the QR matrix factorization, which is suitable for small and moderate size problems, but for larger problems it will be too slow and memory-consuming. Analogous Torch functionality [25, 26] is only being developed. In our work, we show how a differentiable implicit solver can be built on the basis of any automatic differentiation framework. To address this, we developed custom autograd functions tailored to our needs.

Building on this foundation, we propose an integrated approach that combines GNNs, finite-volume methods, implicit time-stepping, and automatic differentiation to create a fully differentiable pipeline for both forward and inverse modeling. Our approach ensures differentiability with respect to various components, including grid cell locations, PDE coefficients, and the right-hand side, providing a flexible and robust framework for tackling both forward modeling and inverse problems.

We should emphasize that our focus lies in design and use of asymptotically

correct (non-trainable) differentiable implicit FV solvers rather than in training surrogate models for them (see Section 3 for more details).

Specifically, we focus on the following key problems:

- Forward modeling of stiff evolutionary and static problems;
- Solving inverse problems to estimate equation coefficients;
- Solving inverse problems to estimate unknown right-hand side terms;
- Graph coarsening to accelerate forward modeling.

We demonstrate the capability of the proposed approach to effectively address all these problems, showcasing its flexibility and ease of implementation for geophysical modelling and other PDE-based applications.

3 Related Work

A significant number of recent works can be characterized as surrogate modeling. In the context of forward problems, the goal is to approximate the Green's function of the equation, as seen in [1, 20, 21]. The work of [23] involved implicit Graph Neural Networks (GNNs) for this task. Implicit deep learning, [6], also referred to as deep equilibrium models (DEQ), substitutes conventional activation functions at particular layers with a nonlinear equilibrium equation completed with a prediction equation. DEQ combined with GNNs are known as implicit GNNs, [11]. Their benefit is that data propagates instantly within the graph, rather than with finite speed as in the case of conventional deep learning. The work of [13] proposed a learnable approach for computing fluxes in the FVM, motivated by convection-dominated problems.

For inverse problems, the aim is to approximate the inverse of a potentially nonlinear forward operator using a given dataset. For instance, [35] employed GNNs for this problem. In [16], the authors enhanced GNNs with novel geometric features to improve prediction accuracy.

The main drawback of these approaches is the massive data requirements for training (ranging from gigabytes to terabytes). Also, the training is computationally expensive, often requiring hundreds to thousands of GPU hours for training. More importantly, the obtained solution is less interpretable compared to solvers employing the classical numerical schemes. However, as will be shown below, our method outperforms PINNs in solving inverse problems both in terms of quality and computational expenses.

In traditional methods for inverse problems, as described in [36], optimization is performed for each new dataset. These algorithms rely on the derivation of Jacobians, a tedious and error-prone task. The availability of automatic differentiation (AD) pipelines can help avoid this issue. For example, [37] presented an AD-based acoustic data inversion method combined with explicit Euler time discretization for the wave equation. In our work, we follow a similar

direction by demonstrating how implicit schemes can be integrated into an AD pipeline and further combined with GNNs.

Our approach is closely related to physics-informed neural networks (PINNs) [17, 27], [22], and their extensions, such as graph convolutional PINNs [9], as both methods can be considered self-supervised learning techniques.

4 Methodology

4.1 Governing equations

In this work, we mainly consider a parabolic partial-differential equation in some domain V,

$$\frac{\partial u}{\partial t} - \operatorname{div}(K \nabla u) = f, \quad 0 < t < T, \quad (x, y) \in V, \tag{1}$$

where $u(x, y, t)$ is the unknown variable, $K(x, y)$ is a positive coefficient, $f(x, y, t)$ is the source term. The above equation is completed with the initial condition $u = u^0(x, y)$ at $t = 0$ and zero Neumann boundary conditions on the domain boundary.

Assume that we are given with a set of points C_s, $1 \leq s \leq N$. We use them to generate a Voronoi grid, (1). A numerical solution to (1) can be received by combining the finite volume method (FVM) for spatial discretization [7, 19] and the implicit Euler scheme for temporal discretization,

$$D \frac{u^{k+1} - u^k}{\tau} + A \cdot u^{k+1} = D f^{k+1}, \quad 0 \leq k \leq n - 1. \tag{2}$$

where $u^k \in \mathbb{R}^N$, $0 \leq k \leq n$ is the numerical solution at the respective time step, A is the finite volume matrix, D is the diagonal matrix of Voronoi cells areas, and f^k is the source vector.

We implemented the finite volume method within the graph neural network (GNN) framework with the message passing paradigm. Our GNN solver uses weights for the edges of the graph,

$$w_{ij} = -A_{ij}. \tag{3}$$

Since every non-zero off-diagonal matrix entry corresponds to some edge of the graph, the GNN message massing mechanism could be used to implement matrix-vector multiplication, $A \cdot u^k$.

4.2 Problem Setup

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

$$L(\theta) = \sum_{k=1}^{n} \| R \cdot u^k(\theta) - u^k_{obs} \|^2 + S(\theta), \tag{4}$$

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 $S(\theta)$ is a stabilizer. The role of operator R is to simply keep in a given vector several entries (wherever sensors are located) while removing the other entries.

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

$$A(\theta) \cdot u^k(\theta) = f^k(\theta), \quad 1 \leq k \leq n, \tag{5}$$

forming a constraint to (4).

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

$$L(\theta) = \sum_{k=1}^{n} \|R \cdot A(\theta)^{-1} \cdot f^k(\theta) - u^k_{obs}\|^2 + S(\theta), \tag{6}$$

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

An alternative approach might involve solving the implicit equation through optimization techniques commonly used in deep learning, such as constructing a minimization problem for equation (5) combined with a data loss function, potentially avoiding matrix inversion. However, matrices arising from discretization of elliptic PDEs, like (5), are highly ill-conditioned. Incorporation of such a matrix into optimization problems would lead to very slow convergence of the minimization algorithms (both stochastic and deterministic). Practically, (5) is solved with either a direct or preconditioned iterative solver.

Let us notice the following. Firstly, automatic differentiation systems that are typically used for minimization of (4), may not process implicit relations like (5) or the respective functionality may be limited. Secondly, custom developed fast parallel solvers are commonly linked to solve (5) in practical modelling. To address both of these cases, we will derive formulae for forward and backward propagations in the next subsection.

4.3 Forward and Backward Propagation

We will assume that the system matrix in (5) is symmetric and positive-definite, what is typically the case within implicit time-stepping and static problems. Let us rewrite the system as

$$A \cdot u = b \quad \text{or} \quad \sum_j a_{ij} u_j = b_i, \tag{7}$$

for brevity. The matrix A is large and sparse thus special formats are used to store its non-zero values only, e.g. compressed sparse row (CSR) or coordinate (COO) formats.

During the forward propagation stage, we have to solve (7), what could be formally written as,

$$u = A^{-1} \cdot b. \tag{8}$$

Practical algorithms avoid storing A^{-1} and multiplication by it.

During the backward propagation stage, we have to compute the gradients of u with respect to A and b. The gradient with respect to b could be found as follows,

$$\sum_j a_{ij} \frac{\partial u_j}{\partial b_k} = \delta_{ik}, \tag{9}$$

thus,

$$\nabla_b u = A^{-1}. \tag{10}$$

The gradient with respect to A has the following form,

$$\sum_j \frac{\partial a_{ij}}{\partial a_{km}} u_j + \sum_j a_{ij} \frac{\partial u_j}{\partial a_{km}} = 0, \tag{11}$$

thus,

$$\nabla_A u = -A^{-1} \cdot P \cdot u. \tag{12}$$

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

$$\nabla_b L = \nabla_u L \cdot \nabla_b u, \quad \text{and} \quad \nabla_A L = \nabla_u L \cdot \nabla_A u, \tag{13}$$

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 (10) and (12) into (13), we receive,

$$\nabla_b L = \nabla_u L \cdot A^{-1}, \tag{14}$$

and

$$\nabla_A L = -\nabla_u L \cdot A^{-1} \cdot P \cdot u = -\nabla_b L \cdot P \cdot u. \tag{15}$$

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

1. Compute $\nabla_b L$ with (14), 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.

Thus 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, (15) is reduced to differentiating only non-zero entries.

We should emphasize there is a close conceptual relationship between forward and backward propagation in automatic differentiation (AD) and the notions of direct and adjoint states in constrained optimization. In forward mode AD, sensitivities of outputs with respect to inputs are propagated along the computational graph in the same direction as the evaluation of the primal function, mirroring the direct approach in optimization where the state equations are solved forward in time to obtain solution and its associated sensitivities. Conversely, backward (or reverse) mode AD propagates derivatives from outputs to inputs by traversing the computational graph in reverse, which is equivalent to the adjoint method in optimization: instead of directly computing sensitivities by forward perturbation, one introduces adjoint variables governed by backward equations, enabling efficient computation of gradients with respect to a large number of parameters. Thus, 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, [2, 10].

5 Experiments

5.1 Implementation

Our differentiable graph simulator is implemented based on the message passing paradigm in a graph neural networks. For implementing such a solver, we used Pytorch Geometric [8] framework. Formalas (14) and (15) were implemented in a PyTorch custom autograd function. Optimization of graph coarsening is implemented with the method for autodifferentiation of the 2D Voronoi tessellation, [30]. A schematic illustration of our pipeline is given in Fig. 1.

5.2 Learnable Graph Coarsening for the Linear Parabolic Equation

Let a polygonal domain be filled with heterogeneous porous media. We assume that a slightly compressible fluid is injected at one point and pressure is measured at some other points. With minor simplifications [5] the fluid flow is described by the parabolic equation, (1).

In these experiments, the coarsening process is guided by a graph neural network (GNN), which learns soft cluster assignments to map nodes from the fine grid to coarser clusters. Our approach builds upon [31], but instead of solving a global optimization problem for point locations, we adopt a graph coarsening method. Additionally, in this work, we replace the explicit time-stepping used in the baseline [31] with the implicit time-stepping approach, (2).

Our GNN architecture for coarsening consists of a single graph convolution layer, followed by a point-wise multilayer perceptron (MLP) with two fully connected layers and a softmax activation at the end, which predicts the soft

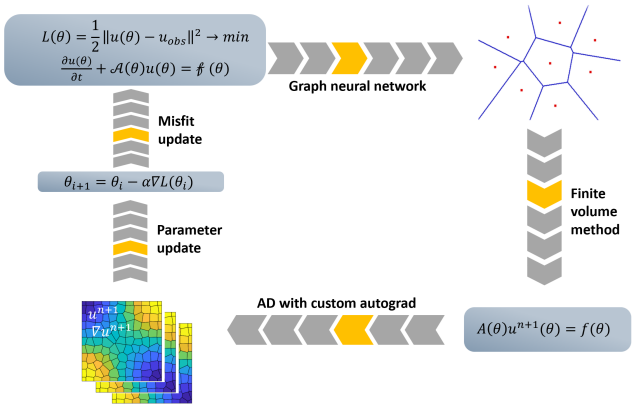


Figure 1: Differentiable modeling pipeline developed in this work combines graph neural networks, the finite-volume method, implicit time-stepping and can be applied to optimization in forward and inverse problems.

assignment matrix S_{ij} for probabilistic clustering. As in the referenced work, the features of sources and measurement points remain unchanged during the coarsening process. The feature aggregation is mathematically represented as follows:

$$\mathbf{c}_j = \frac{\sum_{i=1}^{N} S_{ij}\mathbf{x}_i}{\sum_{i=1}^{N} S_{ij}}, \tag{16}$$

where $\mathbf{c}_j$ is the centroid feature for cluster j, $\mathbf{x}_i$ is the feature vector for node i, and S_{ij} is the soft assignment of node i to cluster j. Other details on the coarsening algorithm could be found in [29].

In the first experiment, the modelling domain was an octagon and partitioned into 252 cells, Fig. 2 a). The permeability coefficient K varied from 10^{-3} to 10^4 making the problem highly heterogeneous. We modelled pressure with (2) and used the time step, τ, of 0.1 s. Pressure values were recorded at two measurement points and the respective time series are shown in Fig. 2 c). Although the measurement points are symmetric with respect to the source, they are located in the areas of different permeability, leading to different pressure growth rates.

In this experiment, minimization of (4) was performed with Adam optimizer and no stabilizer. We picked a coarse grid with 25 cells and tried to optimize it minimizing the misfit between pressure values on the coarse and fine grids over time. An approximately optimal grid was received after 200 epochs, Fig. 2 b).

In Fig. 2, we observe that the pressure time series match very well and that coarsened grid is fairly regular shaped. This illustrates robustness and stability of our framework.

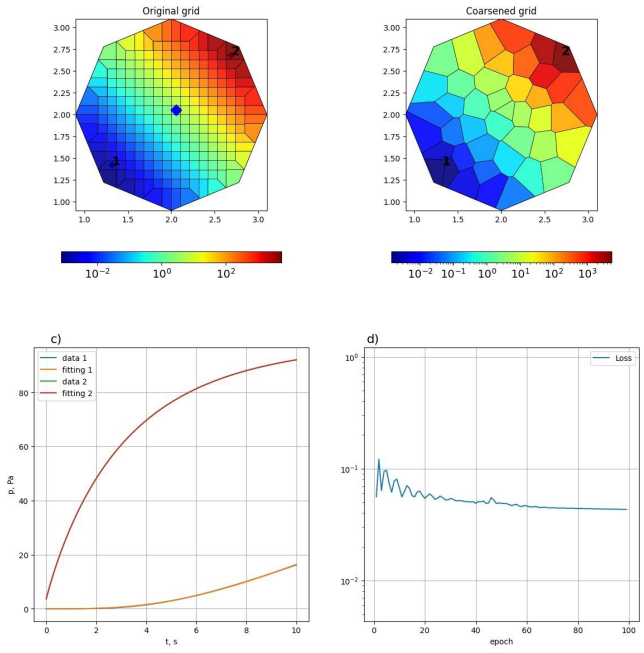


Figure 2: Computational grid coarsening for the linear parabolic equation, (1). a) Original grid, color indicates permeability, source marked with blued diamond, and two measurement points (1 and 2) marked as black diamonds; b) coarsened grid; c) data at two measurement points, modelled with the original and coarsened grids; d) loss function during optimization.

5.3 Learnable Graph Coarsening for the Nonlinear Parabolic Equation

We performed an experiment with learnable graph coarsening for the nonlinear parabolic equation governing compressible fluid flow in porous media,

$$\frac{\partial}{\partial t}\left(\phi(u)\rho(u)\right) - \operatorname{div}\left(\rho(u)\frac{K}{\mu}\nabla u\right) = q, \tag{17}$$

In the nonlinear equation (17), density, ρ, and porosity, ϕ, depend on the pressure,

$$\rho(u) = \rho_r e^{c_\rho(u-u_r)} \tag{18}$$

$$\phi(u) = \phi_r e^{c_\phi(u-u_r)} \tag{19}$$

where ρ_r, ϕ_r, c_ρ, c_ϕ and u_r are some constants. Viscosity μ is also assumed constant.

To solve (17), we implemented a semi-implicit temporal discretization. Let

us look at the term $\rho(u)\phi(u)$ and define $c(u)$,

$$\frac{\partial}{\partial t}\left(\phi(u)\rho(u)\right) = c(u)\frac{\partial u}{\partial t}. \tag{20}$$

Now let us return to (17),

$$c(u)\frac{\partial u}{\partial t} - \operatorname{div}\left(\rho(u)\frac{K}{\mu}\nabla u\right) = q. \tag{21}$$

The semi-implicit temporal discretization takes following form,

$$c(u^n)V\frac{u^{n+1}-u^n}{\tau} + A(u^n)\cdot u^{n+1} = Vq^{n+1}. \tag{22}$$

Notice the above system is linear although the system matrix is updated at every time step.

The experiment was run with next values of parameters: 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}, Fig. 3 a).

The results are shown in Fig 3. We reduced the size of the grid from 400 to 40 cells and obtained a fairly good match of pressure time series.

5.4 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 consider a setup similar to the previous experiment. However, we will now assume that the true permeability, K, is unknown. This problem is known as *history matching* in petroleum engineering, [32]. We will estimate the permeability by minimizing the following functional,

$$L(K) = \int_0^T \sum_{i=1}^m w_i(t)\left(u^i_{mod}(t) - u^i_{obs}(t)\right)^2 dt + \beta\int_V |\nabla K|^2 dxdy, \tag{23}$$

constrained with (1) and $K(x,y) > 0$. Here, the first term represents the data misfit, while the second term is a stabilizer promoting permeability smoothness.

In our experiment, pressure was measured at two spatial locations, $m = 2$ and at 100 temporal points, $\tau = 0.1s$, $T = 10s$, Fig. 4 a) and c). The Voronoi grid was formed from a graph of 400 randomly perturbed vertices.

We applied the Adam optimizer to (23) stating with $K = 1$ and continued for 250 epochs. Decent data fit was archived after 150 epochs, Fig. 4 d). The recovered permeability, Fig. 4 b), has main features of the true one: the lower left area of lower permeability, the upper right area of higher permeability. We want to mention that the inverse problems always imply non-uniqueness of the solution and this difference between true field and recovered one was expected.

The grid that was used in this experiment was quite distorted, Fig. 4a). We intentionally picked such a grid to demonstrate the flexibility of the presented approach.

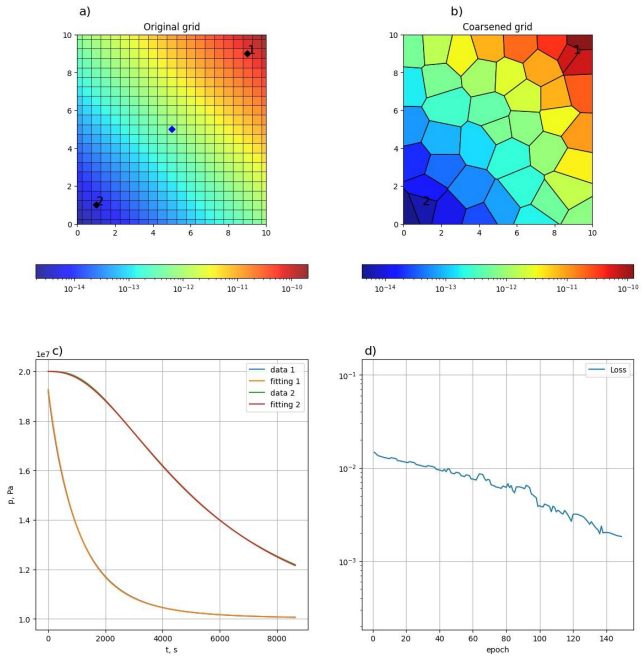


Figure 3: Computational grid coarsening for the nonlinear parabolic equation, (17). a) Original grid, color indicates permeability, source marked with blued diamond, and two measurement points (1 and 2) marked as black diamonds; b) coarsened grid; c) data at two measurement points, modelled with the original and coarsened grids; d) loss function during optimization.

5.5 Inverse Source Problem

Another type of an 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 *self-potential method*, [28]. The goal is to estimate direct current source density based on the observations of the electric potential. They are related by the equation,

$$-\mathrm{div}(\sigma \nabla u) = f. \tag{24}$$

The above equation was considered in a rectangular domain, completed Dirichlet on the bottom side and Neumann boundary on the other three sides. Here σ is known electrical conductivity, u is the electric potential, f is source density. Although, the equation is static, its numerical solution will involve solution of a large sparse linear system, making it similar to implicit time stepping.

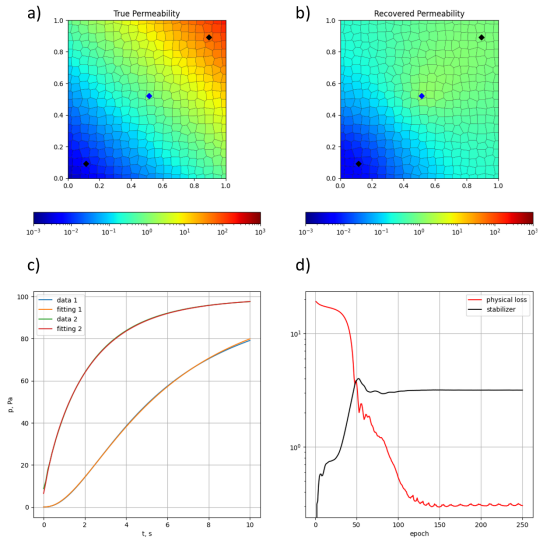


Figure 4: Solution of the coefficient inverse problem. a) True permeability, source and two measurement points (1 and 2); b) recovered permeability; c) pressure at two measurement points (data and prediction/fitting); d) loss function during 250 epochs of minimization.

The source density is estimated by minimizing the following functional,

$$L(f) = \sum_{i=1}^{n} w_i \left(u^i_{mod} - u^i_{obs}\right)^2 + \gamma \int_V |\kappa \nabla f|^2 dxdy + \delta \int_V f^2 dxdy, \tag{25}$$

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

In this experiment, the Voronoi grid was formed from a graph of 900 randomly perturbed vertices. The electric potential was measured at $n = 28$ points on the upper side of the domain, Fig. 5 a). The true right-hand side was formed by a source in the left part of the domain and a sink in the right part. Observed data, u_{obs}, was received by solving (24) and adding 10% noise. Notice this problem was highly under-determined: we estimated the values of 900 unknowns based on 28.

We applied the Adam optimizer to (25) starting with $f = 0$ and continuing for 100 epochs. We observed good data fit and convergence of the optimizer, Fig. 5 c) and d). The recovered and true right-hand sides, Fig. 5 b) and a), possess common features: positive values in the left part and negative in the

right. The recovered right-hand side is quite smeared. Also the locations of the extrema are biased. These artifacts are related to the measurement setup and the impact of stabilizers. The use of more advanced stabilizers can further improve inversion results, however we also don't expect the full coincidence of the solution to the true sources.

We evaluated our results in this experiment versus those of the PINN implemented in the DeepXDE package [22]. Both the unknown solution u and the right-hand side f were parametrized with a feedforward neural network. The network had two inputs (x and y), three dense layers of 150 neurons, and two outputs (u and f). The loss function is defined as the weighted sum of L^2 norms of residuals for the equation and boundary conditions and the difference of observed and modelled data. We had 625 random training residual points in the domain and 900 random points to test the PDE residual. The network was trained in 5000 epochs with a learning rate of 1e-3.

Fig. 5 e) shows the estimated right-hand side f using PINN. We interpolated the values of right-hand side to the same grid as was used above for easier interpretation. We see a very similar pattern in Fig. 5 b) and Fig. 5 e). The main difference between our result and that of PINN is in magnitudes. We attribute this to the following. The measured data received an integral impact of the source and sink. But since they have same magnitudes and opposite sign, the actual magnitudes give a minor impact on the data at measurement points. Our method captured the magnitudes of the source and sink more correctly than the PINN due to stabilizers (last two terms in (25)) which were absent in the PINN loss.

Lastly, we note that the convergence of the optimizer in our method was significantly faster, as shown in Fig. 5 d) and f). We attribute this difference to the highly nonlinear nature of the loss functions in PINNs, whereas our approach involves a quadratic loss function, (25), leading to more efficient optimization.

6 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, [31] presented results on accelerating forward modeling using graph coarsening, which is somewhat similar to our approach in Section 5.2. However, their method encountered significant time-step limitations due to the use of an explicit scheme.

Both learnable coarsening examples 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.

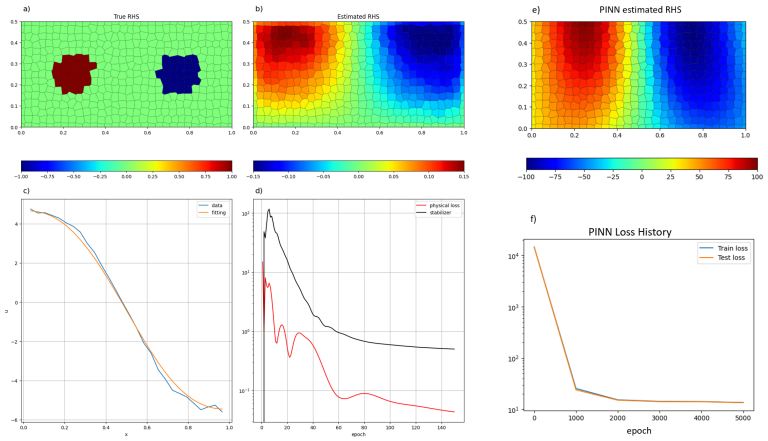


Figure 5: Solution of the source inverse problem. a) True right-hand side f and measurement points (black diamonds); magenta diamonds indicate the side where Dirichlet boundary condition was applied; Neumann boundary condition was applied on the other three sides; b) recovered right-hand side; c) observed 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 (25)) and stabilizers (later terms in (25)). e) recovered right-hand side f using the PINN; f) PINN loss function during 5000 epochs of minimization

Our method outperforms the commonly used PINN architecture in solving inverse problems both in terms of quality and computational expenses.

Our pipeline is highly efficient. In future work, we plan to address large-scale 3D problems and more complex equations, such as multi-phase fluid flow. Large-scale 3D modeling and inversion typically rely on custom preconditioned iterative solvers, as they offer both memory efficiency and fast convergence. Our pipeline's ability to easily integrate new solvers makes it particularly appealing for industrial applications.

In this work, we discussed coarsening within the context of forward modeling and data inversion. An interesting future research direction would be to combine coarsening with inversion, allowing data inversion to be performed on an optimal coarse grid in a shorter time.

7 Acknowledgement

The authors are thankful to their colleague Sergei Shumilin for many fruitful discussions. The study was supported by the Russian Science Foundation grant No. 24-41-02035, `https://rscf.ru/en/project/24-41-02035/`

References

[1] Ferran Alet, Adarsh K. Jeewajee, Maria Bauza, Alberto Rodriguez, Tomas Lozano-Perez, and Leslie Pack Kaelbling. Graph element networks: adaptive, structured computation and memory, 2019. URL https://arxiv.org/abs/1904.09019.

[2] Pierre Baldi. *Gradient descent learning algorithms: a unified perspective*, page 509–541. L. Erlbaum Associates Inc., USA, 1995. ISBN 0805812598.

[3] 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. ISSN 0377-0427. doi: https://doi.org/10.1016/S0377-0427(00)00422-2. URL https://www.sciencedirect.com/science/article/pii/S0377042700004222. Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations.

[4] 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. ISSN 1532-4435.

[5] Zhangxin Chen. *Reservoir Simulation: Mathematical Techniques in Oil Recovery*. Society for Industrial and Applied Mathematics, January 2007. ISBN 9780898717075. doi: 10.1137/1.9780898717075. URL http://dx.doi.org/10.1137/1.9780898717075.

[6] Laurent El Ghaoui, Fangda Gu, Bertrand Travacca, Armin Askari, and Alicia Tsai. Implicit deep learning. *SIAM Journal on Mathematics of Data Science*, 3(3):930–958, January 2021. ISSN 2577-0187. doi: 10.1137/20m1358517. URL http://dx.doi.org/10.1137/20M1358517.

[7] Robert Eymard, Thierry Gallouët, and Raphaèle Herbin. *Finite volume methods*, page 713–1018. Elsevier, 2000. doi: 10.1016/s1570-8659(00)07005-8. URL http://dx.doi.org/10.1016/s1570-8659(00)07005-8.

[8] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric, 2019. URL https://arxiv.org/abs/1903.02428.

[9] Han Gao, Matthew J. Zahr, and Jian-Xun Wang. Physics-informed graph neural galerkin networks: A unified framework for solving pde-governed forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 390:114502, February 2022. ISSN 0045-7825. doi: 10.1016/j.cma.2021.114502. URL http://dx.doi.org/10.1016/j.cma.2021.114502.

[10] 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. ISBN 9780898717761.

[11] Fangda Gu, Heng Chang, Wenwu Zhu, Somayeh Sojoudi, and Laurent El Ghaoui. Implicit graph neural networks, 2021. URL https://arxiv.org/abs/2009.06211.

[12] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[13] Masanobu Horie and Naoto Mitsume. Graph neural pde solvers with conservation and similarity-equivariance, 2024. URL https://arxiv.org/abs/2405.16183.

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

[15] JAX. 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.

[16] Loh Sher En Jessica, Naheed Anjum Arafat, Wei Xian Lim, Wai Lee Chan, and Adams Wai Kin Kong. Finite volume features, global geometry representations, and residual training for deep learning-based cfd simulation, 2023. URL https://arxiv.org/abs/2311.14464.

[17] E. Kharazmi, Z. Zhang, and G. E. Karniadakis. Variational physics-informed neural networks for solving partial differential equations, 2019. URL https://arxiv.org/abs/1912.00873.

[18] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017.

[19] Yu.A. Kuznetsov, O.V. Boiarkine, I.V. Kapyrin, and N.B. Yavich. Numerical analysis of a two-level preconditioner for the diffusion equation with an anisotropic diffusion tensor. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 22(4):377–391, 2007. doi: doi:10.1515/rnam.2007.018. URL https://doi.org/10.1515/rnam.2007.018.

[20] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2021. URL https://arxiv.org/abs/2010.08895.

[21] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3): 218–229, March 2021. ISSN 2522-5839. doi: 10.1038/s42256-021-00302-5. URL http://dx.doi.org/10.1038/s42256-021-00302-5.

[22] 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. doi: 10.1137/19M1274067. URL https://doi.org/10.1137/19M1274067.

[23] Matthieu Nastorg, Michele-Alessandro Bucci, Thibault Faney, Jean-Marc Gratien, Guillaume Charpiat, and Marc Schoenauer. An implicit gnn solver for poisson-like problems. *Computers & Mathematics with Applications*, 176:270–288, December 2024. ISSN 0898-1221. doi: 10.1016/j.camwa.2024.10.036. URL http://dx.doi.org/10.1016/j.camwa.2024.10.036.

[24] Uwe Naumann. *The art of differentiating computer programs : an introduction to algorithmic differentiation*. SIAM, 1st edition, 2012. ISBN ISBN 978-1-611972-06-1.

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

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

[27] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[28] André Revil and Abderrahim Jardani. *Forward and inverse modeling*, page 110–153. Cambridge University Press, 2013. doi: 10.1017/cbo9781139094252.006. URL http://dx.doi.org/10.1017/CBO9781139094252.006.

[29] 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*, submitted.

[30] Sergei Shumilin, Alexander Ryabov, Serguei Barannikov, Evgeny Burnaev, and Vladimir Vanovskii. A method for auto-differentiation of the voronoi tessellation, 2024. URL https://arxiv.org/abs/2312.16192.

[31] 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. URL https://proceedings.mlr.press/v235/shumilin24a.html.

[32] S. Srinivasan and J.Y. Leung. *Petroleum Reservoir Modeling and Simulation: Geology, Geostatistics, and Performance Prediction*. McGraw Hill LLC, 2022. ISBN 9781259834301. URL https://books.google.ru/books?id=crqeEAAAQBAJ.

[33] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

[34] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

[35] Qingqing Zhao, David B. Lindell, and Gordon Wetzstein. Learning to solve pde-constrained inverse problems with graph networks. 2022.

[36] M.S. Zhdanov. *Geophysical Inverse Theory and Regularization Problems*. Methods in Geochemistry and Geophysics. Elsevier Science, 2002. ISBN 9780080532509. URL https://books.google.ru/books?id=tHtDETv7VCoC.

[37] 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. ISSN 0098-3004. doi: 10.1016/j.cageo.2021.104751. URL http://dx.doi.org/10.1016/j.cageo.2021.104751.