Douglas-Rachford Splitting for Hybrid Differentiable Models

Abdel-Rahim Mezidi
abdel.rahim.mezidi@univ-st-etienne.fr

Jordan Patracone
jordan.patracone@univ-st-etienne.fr

Amaury Habrard
amaury.habrard@univ-st-etienne.fr

Université Jean Monnet Saint-Etienne, CNRS, Institut d'Optique Graduate School, Inria,
Laboratoire Hubert Curien UMR 5516, F-42023, SAINT- ETIENNE, France

Abstract

Hybrid differentiable models that combine neural networks with numerical solvers have shown promise in scientific machine learning applications, particularly for modeling complex physical phenomena governed by partial differential equations (PDEs). However, many real systems exhibit incomplete or partially known physics. We address this setting by introducing a proximal optimization framework based on Douglas–Rachford splitting that cleanly separates the known physics from learned corrections. Our approach reformulates learning as a structured splitting of an incomplete-physics objective and a data-driven regularizer, yielding an unrolled architecture that alternates a numerical solver on the available physics and a neural network that compensates missing effects. Theoretically, we establish connections between our Douglas-Rachford iterations and gradient flows in PDE systems. Empirically, on the Allen-Cahn benchmark problem, our method improves accuracy over standard methods.

1 Introduction

Hybrid differentiable models that combine neural networks with physics-based numerical solvers have shown promise in scientific machine learning by balancing data-driven flexibility with physical consistency [17, 14, 15]. Existing approaches include *embedded* methods that replace solver components [14, 5, 2] and *correction* methods that augment solver outputs [6, 15, 11, 9]. We focus on correction methods, particularly under incomplete physics where the PDE is only partially known and the numerical step captures only a subset of operators.

Despite their promise, current hybrid models face several limitations: (1) training instability when back-propagating through heterogeneous solver–network architectures [8], (2) limited treatment of incomplete physics where parts of the PDE are missing or uncertain, and (3) empirical architecture designs lacking principled frameworks. While differentiable solvers [15, 6] enable gradient backpropagation through PDE operators, differentiability alone does not resolve how to optimally combine known physics with learned corrections.

Beyond differentiable solvers, Plug-and-Play methods and unrolled optimization [16, 3, 12] have shown promise, especially in imaging, where learned neural networks or known denoisers/regularizers are integrated in the iterative optimization scheme. Recent hybrid designs for partially known PDEs further motivate this direction: when only a subset of operators is available, learning can compensate missing physics [17].

1st Workshop on Differentiable Systems and Scientific Machine Learning @ EurIPS 2025

We address these challenges through proximal optimization theory. Our key insight is that many PDEs are gradient flows of energy functionals, and their implicit time-stepping schemes correspond to proximal operators. By leveraging Douglas-Rachford splitting [7], we separate physical dynamics (numerical solvers) from data-driven regularization (neural networks) while ensuring principled interaction through optimization steps.

Contributions. We present: (1) a proximal optimization framework reformulating hybrid training as structured splitting of incomplete physics and learned regularization; (2) an unrolled Douglas–Rachford architecture that alternates numerical steps for available physics with neural network steps for missing effects; (3) theoretical connections between Douglas–Rachford iterations and PDE gradient flows; and (4) empirical validation, including Allen–Cahn where the numerical step models only diffusion while the network learns the reaction term.

2 Douglas-Rachford Unrolled Architecture

Many PDEs can be interpreted as gradient flows of energy functionals. For instance, the heat equation $\frac{\partial u}{\partial t} = \nabla^2 u$ is the gradient flow of the Dirichlet energy $E(u) = \frac{1}{2}\int |\nabla u|^2 dx$. Similarly, the Allen-Cahn equation $\frac{\partial u}{\partial t} = \epsilon^2 \nabla^2 u + u - u^3$ minimizes the Ginzburg-Landau free energy $E(u) = \int \left[\frac{\epsilon^2}{2}|\nabla u|^2 + \frac{1}{4}(1-u^2)^2\right] dx$. More generally, these systems follow the form $\frac{dy}{dt} = -\nabla E(y)$, where $E(y)$ represents the total energy functional governing the physical system.

When discretizing such gradient flows with the implicit Euler scheme, we obtain $\frac{y_{n+1}-y_n}{\Delta t} = -\nabla E(y_{n+1})$, which can be rewritten as a minimization problem:

$$y_{n+1} = \arg\min_y \left[\frac{1}{2\Delta t}\|y - y_n\|^2 + E(y)\right] = \text{prox}_{\Delta t E}(y_n). \tag{1}$$

This connection reveals that implicit time-stepping corresponds to the proximal operator of the energy functional [10, 1], providing a natural bridge between numerical methods and optimization theory. In the complete-physics case, the operator $\text{prox}_{\Delta t E}$ can be implemented by an implicit numerical solver (e.g., backward Euler).

In practice, we often face incomplete physical models where the true energy $E(y)$ is unknown or intractable. Instead, we have access to a simplified energy $E'(y)$ that captures only partial physics. To compensate for this gap and better fit experimental data, we introduce a learned regularization term $R(y)$. The complete time-stepping problem becomes:

$$y_{n+1} = \arg\min_y [F(y) + \Delta t\, R(y)] \tag{2}$$

where $F(y) = \frac{1}{2}\|y - y_n\|^2 + \Delta t\, E'(y)$ combines the temporal discretization with incomplete physics. However, this composite minimization cannot be directly solved like before.

To address this challenge, we leverage Douglas-Rachford splitting, an algorithm designed for minimizing sums of functions through their individual proximal operators. Then, for any k in $\mathbb{N}^*$, the algorithm is written as:

$$\begin{cases} y^{(k+1)} = \text{prox}_{\lambda F}(z^{(k)}) = \text{prox}_{\frac{\lambda\,\Delta t}{1+\lambda}E'}\left(\dfrac{z^{(k)} + \lambda y_n}{1+\lambda}\right), & y^{(0)} = y_n, \\ z^{(k+1)} = z^{(k)} + \gamma\Big[\text{prox}_{\lambda\,\Delta t\, R}(2y^{(k+1)} - z^{(k)}) - y^{(k+1)}\Big] & y^{(k)} \xrightarrow[k\to\infty]{} y_{n+1} \end{cases} \tag{3}$$

where $\lambda > 0$ and $\gamma \in (0, 2)$. The first step is the proximal of the incomplete physics E' and the second step is the proximal of R. We unroll a fixed number of iterations.

Assumptions. If E' and R are proper, closed, convex and their proximal operators are single-valued, DR with $\gamma \in (0, 2)$ converges to a minimizer of $F + R$.

Remark: In our formulation, the NN learns a proximal operator $\text{prox}_{\lambda\Delta t\, R}$. By definition it needs to be the resolvent of a maximally monotone operator. Practically, this entails (i) nonexpansiveness (Lipschitz constant ≤ 1), (ii) monotonicity, and (iii) consistency with an underlying convex function R.

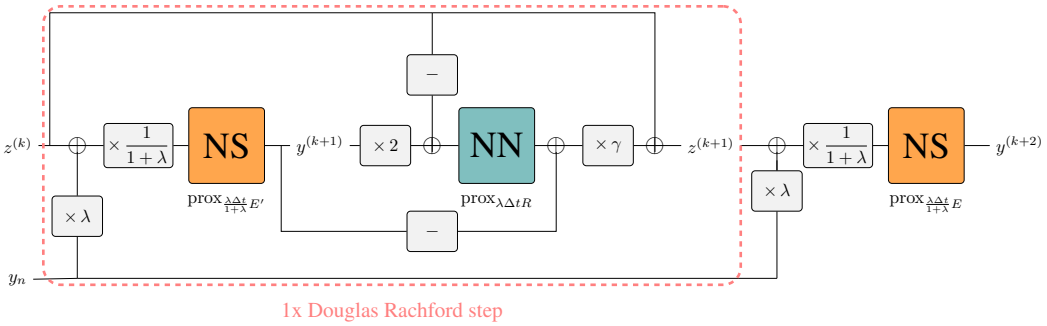

Figure 1: Complete Douglas-Rachford hybrid architecture with numerical solver (NS) and neural network (NN) components. The dashed red box highlights one Douglas-Rachford iteration step. The parameter γ controls the relaxation, with $\gamma = 1$ corresponding to standard Douglas-Rachford and $\gamma = 2$ to Peaceman-Rachford splitting.

3 Implementation

Figure 1 depicts this unrolled design. Each DR iteration alternates between:

- Numerical Solver (NS): Implements $\text{prox}_{\lambda F}$ using established PDE operators

- Neural Network (NN): Implements $\text{prox}_{\lambda\Delta t\, R}$, correcting the numerical solver's output.

The hyperparameters λ (proximal step size) and γ (relaxation parameter) can be either fixed a priori or learned during training.

We unroll a fixed number of iterations and output the primal variable y—meaning we reapply the numerical solver at the end—for rollout consistency.

Neural networks as proximal operators. Following our theoretical framework, we need the neural network to learn the second proximal operator $\text{prox}_{\lambda\Delta t R}$. While learning proper proximal operators is challenging [13, 4], we relax the constraints on the neural network, following standard Plug-and-Play framework, allowing it to learn any function. To do so, we use a standard ResNet with skip connections.

To ensure proper gradient backpropagation, we implement the numerical solver using PhiFlow's differentiable PDE operators. In $\text{prox}_{\lambda F}$, the factor $\frac{\lambda\,\Delta t}{1+\lambda}$ sets the effective timestep of the solver, meaning λ has a direct effect on the solver behavior.

4 Experiments

Datasets. We evaluate on 2D Allen–Cahn ($\partial_t u = \epsilon^2\,\nabla^2 u + u - u^3$) on a 64×64 grid. In our setting, the full PDE is partially known: we designate a subset of terms as known in the solver and learn the remainder. For Allen–Cahn, we treat diffusion as known and learn the reaction. We generate the dataset by solving the full PDE with a high-precision solver starting from random initial conditions. We then split it into 800 training, 100 validation and 100 testing samples.

Baselines. We compare to several baseline approaches: (1) *solver-only* uses only the incomplete numerical solver without any neural network correction, representing the performance when physics knowledge is limited; (2) *NN-only* uses only the neural network without any physics-based solver, testing the pure data-driven approach; (3) *post-correction* (NS→NN) applies the numerical solver first, then corrects the output with a neural network [8]; and (4) *NS→NN→NS* performs NS→NN k times (similar to HybDR and post-correction) followed by a final numerical solver step, providing a fairer comparison with our iterative approach.

Remark: A single-step *NS→NN* post-correction pipeline can be interpreted as a forward–backward (proximal-gradient) splitting under appropriate modeling choices: treating the numerical term as the proximable part and the neural correction as a (locally) smooth update, or conversely when the NN acts as a proximable denoiser and the physics is used for a gradient step.

Training. We learn the mapping from initial state to state after 15 seconds by unrolling each model 8 times. All models are trained using the Adam optimizer with learning rate 10^{-4} and weight decay 10^{-3} on MSE loss. We train for 150 epochs with early stopping (patience 20 epochs) based on validation loss. All experiments use ResNets with 64 hidden channels and are repeated across four fixed seeds for statistical significance. Models are evaluated on the relative L2 error on the test set.

Results. Figure 2 shows the mean relative L2 error for the different approaches. Our Douglas-Rachford method (HybDR) attains the lowest error (7.5%), outperforming both the NS→NN→NS sequential architecture (9.2%) and post-correction (10.7%). The solver-only baseline exhibits a high error (94.6%) as expected due to missing physics. Notably, the neural network-only model can achieve low error but exhibits high variance with some models failing to converge. Our approach achieves competitive error while maintaining stability, suggesting that alternating optimization via Douglas-Rachford splitting more effectively leverages the interplay between physics-based solvers and neural corrections than simple sequential pipelines.

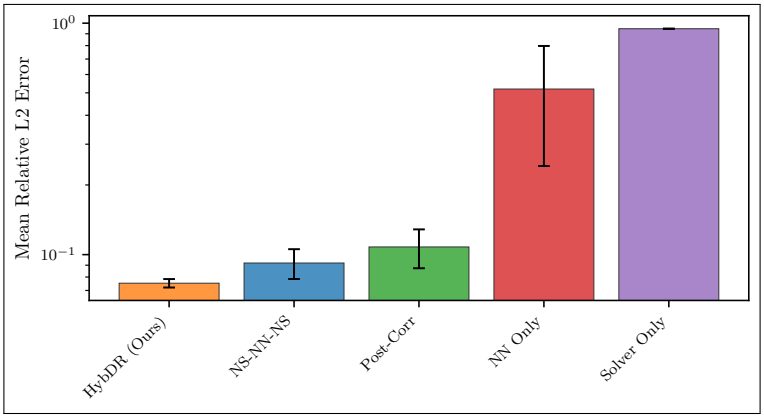

Figure 2: Mean relative L2 error on Allen-Cahn for different methods after 8 unrolled time steps. Error bars show standard error of the mean across four runs.

Making λ and γ learnable can be empirically difficult to optimize, particularly for nonconvex problems where the Lipschitz constant of the underlying operators may be high. In our experiments, we found that a fixed value of $\lambda = 0.25$ and $\gamma = 1.5$ provided stable training and good performance.

5 Conclusion

We presented a principled framework for hybrid differentiable models based on Douglas–Rachford splitting that addresses fundamental challenges in combining neural networks with numerical PDE solvers under incomplete physics. By recognizing the connection between implicit time-stepping schemes and proximal operators, we designed unrolled architectures where the Douglas–Rachford iteration naturally separates available physical dynamics (numerical solver) from learned regularization (neural network) while ensuring structured interaction through optimization steps.

Empirical validation on the Allen–Cahn problem, where the numerical step models diffusion while the network learns the reaction, demonstrates improved accuracy over standard hybrids.

Limitations and outlook. Despite the benefits, training can still exhibit instabilities due to unrolled depth and solver–network interactions; tuning λ and γ introduces additional hyperparameters; and the approach assumes access to efficient proximal operators or differentiable implementations for the physics and learned components. While our current experiments focus on a small-scale setting with ResNet architectures, future work will explore more principled neural architectures that better align with our theoretical framework. Specifically, we plan to investigate Input Convex Neural Networks (ICNNs) and Learned Proximal Networks, which are designed to learn proper proximal operators and would be more consistent with our Douglas-Rachford theoretical foundation. Additionally, we aim to extend our approach to other PDE systems beyond Allen-Cahn and study long rollout stability to assess the method's performance over extended time horizons.

6 Acknowledgments

Research conducted within the context of the Inria–IIT Associate Team.

This work has been funded by a public grant from the French National Research Agency (ANR) under the "France 2030" investment plan, which has the reference EUR MANUTECH SLEIGHT - ANR-17-EURE-0026.

This research was funded in whole or in part by the French National Research Agency (ANR) under project number ANR-25-CE23-5511-01.

References

[1] Heinz H Bauschke, Regina S Burachik, Patrick L Combettes, Veit Elser, D Russell Luke, and Henry Wolkowicz. Fixed-point algorithms for inverse problems in science and engineering. 49, 2011.

[2] Léo Bois, Emmanuel Franck, Laurent Navoret, and Vincent Vigon. An optimal control deep learning method to design artificial viscosities for discontinuous galerkin schemes. *Journal of Scientific Computing*, 101(3):70, 2024.

[3] Stanley H Chan, Xiran Wang, and Omar A Elgendy. Plug-and-play admm for image restoration: Fixed-point convergence and applications. *IEEE Transactions on Computational Imaging*, 3(1):84–98, 2016.

[4] Zhenghan Fang, Sam Buchanan, and Jeremias Sulam. What's in a prior? learned proximal networks for inverse problems. In *The Twelfth International Conference on Learning Representations*, 2024.

[5] Emmanuel Franck, Victor Michel-Dansac, and Laurent Navoret. Approximately well-balanced discontinuous galerkin methods using bases enriched with physics-informed neural networks. *Journal of Computational Physics*, 512:113144, 2024.

[6] Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.

[7] P. L. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6):964–979, 1979.

[8] Bjoern List, Li-Wei Chen, Kartik Bali, and Nils Thuerey. Differentiability in unrolled training of neural physics simulators on transient dynamics. *Computer Methods in Applied Mechanics and Engineering*, 433:117441, 2025.

[9] Jonathan F MacArt, Justin Sirignano, and Jonathan B Freund. Embedded training of neural-network subgrid-scale turbulence models. *Physical Review Fluids*, 6(5):050502, 2021.

[10] Neal Parikh and Stephen Boyd. Proximal Algorithms. *Found. Trends Optim.*, 1(3):127–239, January 2014.

[11] Jaideep Pathak, Mustafa Mustafa, Karthik Kashinath, Emmanuel Motheau, Thorsten Kurth, and Marcus Day. Using machine learning to augment coarse-grid computational fluid dynamics simulations. *arXiv preprint arXiv:2010.00072*, 2020.

[12] Ernest Ryu, Jialin Liu, Sicheng Wang, Xiaohan Chen, Zhangyang Wang, and Wotao Yin. Plug-and-play methods provably converge with properly trained denoisers. pages 5546–5557, 2019.

[13] Yu Sun, Brendt Wohlberg, and Ulugbek S. Kamilov. An online plug-and-play algorithm for regularized image reconstruction. *IEEE Transactions on Computational Imaging*, 5(3):666–681, 2019.

[14] Jonathan Tompson, Kristofer Schlachter, Pablo Sprechmann, and Ken Perlin. Accelerating eulerian fluid simulation with convolutional networks. In *International conference on machine learning*, pages 3424–3433. PMLR, 2017.

[15] Kiwon Um, Robert Brand, Yun (Raymond) Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6111–6122. Curran Associates, Inc., 2020.

[16] Singanallur V. Venkatakrishnan, Charles A. Bouman, and Brendt Wohlberg. Plug-and-Play priors for model based reconstruction. pages 945–948.

[17] Yuan Yin, Vincent Le Guen, Jérémie Dona, Emmanuel De Bézenac, Ibrahim Ayed, Nicolas Thome, and Patrick Gallinari. Augmenting physical models with deep networks for complex dynamics forecasting. volume 2021, page 124012. IOP Publishing, 2021.