Enforcing governing equation constraints in neural PDE solvers via training-free projections

Omer Rochman Sharabi **Gilles Louppe**

University of Liège

Abstract

Neural PDE solvers used for scientific simulation often violate governing equation constraints. While linear constraints can be projected cheaply, many constraints are nonlinear, complicating projection onto the feasible set. Dynamical PDEs are especially difficult because constraints induce long-range dependencies in time. In this work, we evaluate two training-free, post hoc projections of approximate solutions: a nonlinear optimization-based projection, and a local linearization-based projection using Jacobian-vector and vector-Jacobian products. We analyze constraints across representative PDEs and find that both projections substantially reduce violations and improve accuracy over physics-informed baselines.

1 Introduction

Neural PDE solvers for scientific simulations often break physical constraints, even when they score well on standard metrics [1, 2, 3]. In particle systems this could mean violating mass and momentum conservation [4]; in fluids, incompressibility and adherence to Navier–Stokes [5]; in medical imaging, physiologically plausible ranges for parameters [6]. Depending on the system, physical constraints can be linear or nonlinear, local or global, static or dynamical.

In many problems, physical consistency is best understood as satisfying the constraints implied by the governing PDE, the problem specification, and known physics. Per-timestep conditions, such as divergence-free in incompressible fluids, restrict admissible states but can leave temporal consistency underdetermined, since they hold at each step independently. Conservation laws can help but do not guarantee consistent time evolution. Treating the governing equation itself as a constraint couples space and time and addresses this gap, providing a limit on temporal consistency. In this case, the PDE residual often serves as an informative and, sometimes, the only feasible metric of time consistency, because it dictates how plausible states evolve. Nonetheless, even local near-consistency can result in trajectories that are inconsistent globally.

Existing constraint enforcement methods involve significant tradeoffs. Physics-informed approaches like PINNs [7], PINO [8], and their generative counterparts [9] add penalty terms to the loss function, complicating training and potentially preventing convergence [10]. Architectural enforcement guarantees constraint satisfaction [11] but reduces model expressiveness and requires custom designs for each constraint type. Helper networks [12] introduce additional training overhead and hyperparameter tuning. Constrained sampling methods for generative models [13, 14, 15, 16] require backpropagation through sampling paths with repeated constraint evaluations. Moreover, these approaches are rarely validated on complex dynamical constraints like Navier–Stokes, focusing instead on simpler linear constraints such as divergence-free conditions. Finally, PDE determinism can lead to degenerate posteriors with unique solutions, making sampling from such narrow distributions challenging.

Contributions. We focus on hard governing equation constraints from nonlinear dynamical PDEs that induce long-range temporal dependencies. We evaluate two training-free, post hoc projec-

Machine Learning and the Physical Sciences Workshop, NeurIPS 2025.

tion operators of approximate solutions: a nonlinear optimization-based projection and a local linearization-based projection implemented with Jacobian-vector and vector-Jacobian products for scalability. In experiments, both projections substantially reduce violations and improve accuracy over physics-informed training baselines.

2 Projection operators

We consider a PDE of the general form

$$\mathcal{L}[u] = f, \tag{1}$$

over $\Omega \times (0, T)$, where u is the unknown function we seek to find, Ω is the spatial domain, $(0, T)$ is the time interval, $\mathcal{L}$ is a differential operator that may be nonlinear, and f is a known source function on $\Omega \times (0, T)$. The PDE is subject to initial and boundary conditions $\mathcal{I}[u] = u_0$ and $\mathcal{B}[u] = g$, where $\mathcal{I}$ and $\mathcal{B}$ are the initial and boundary condition operators, u_0 specifies the initial state on Ω, and g specifies the boundary values on $\partial\Omega \times (0, T)$.

To enforce global consistency, the full solution must be considered at once. Let the discretized state $u \in \mathbb{R}^N$ be a vector of function evaluations at N points in space and time, $u = \{u(x_i, t_i)\}_{i=1}^N$, where the points $\{(x_i, t_i) \in \Omega \times (0, T)\}_{i=1}^N$ are chosen to be sufficiently dense to ensure accurate operator discretization. Note that we overload u to denote both the continuous function and its discretized representation. The discretized PDE problem becomes a system of nonlinear equations $h(u) = c \in \mathbb{R}^N$ where $h(u) = [\mathcal{L}_d(u), \mathcal{B}_d(u), \mathcal{I}_d(u)]$ is the concatenation of the discretized operators and $c = [f_d, g_d, u_{0,d}]$ is the concatenation of the discretized right-hand sides.

Given the output $\hat{u}$ of a neural PDE solver, we seek a physically consistent solution u that satisfies the constraint $h(u) = c$. If consistency is defined in the L_2 sense, then this can be formulated as finding the projection of $\hat{u}$ into the feasible manifold,

$$\min_u \|u - \hat{u}\| \quad \text{s.t.} \quad h(u) = c. \tag{2}$$

Nonlinear projection. When h is nonlinear, there is no closed form solution to Equation 2. Instead, the problem can be formulated as an unconstrained minimization problem

$$\min_u \|u - \hat{u}\| + \lambda \|h(u) - c\|, \tag{3}$$

with weight $\lambda \geq 0$, which can be solved by numerical optimization. We use the LBFGS algorithm [17] throughout this work.

Linear projection. As Equation 2 admits analytical solutions when h is linear, an alternative is to linearize the constraints about $\hat{u}$ and rearrange

$$h(u) \approx h(\hat{u}) + J_h(u - \hat{u}) \Rightarrow J_h u = c - h(\hat{u}) + J_h \hat{u}, \tag{4}$$

where J_h is the Jacobian of h computed at $\hat{u}$. We denote this linear system as $\mathcal{C}u = b$, where $\mathcal{C} = J_h$ and $b = c - h(\hat{u}) + J_h \hat{u}$. With linearized constraints, we formulate either the projection of $\hat{u}$ onto the feasible set (Equation 5) or, if we suspect $\hat{u}$ to be a poor approximation, the relaxed variant (Equation 6)

$$\min_u \|u - \hat{u}\| \quad \text{s.t.} \quad \mathcal{C}u = b \tag{5}$$

$$\min_u \|u - \hat{u}\| + \lambda \|\mathcal{C}u - b\|. \tag{6}$$

Both admit closed-form solutions [18]

$$u = \hat{u} - \mathcal{C}^\top (\mathcal{C}\mathcal{C}^\top)^{-1} (\mathcal{C}\hat{u} - b) \tag{7}$$

$$u = (I + \lambda \mathcal{C}^\top \mathcal{C})^{-1} (\hat{u} + \lambda \mathcal{C}^\top b). \tag{8}$$

The first is a projection enforcing the linearized constraints exactly when they are consistent, while the second trades constraint satisfaction and proximity to $\hat{u}$ through $\lambda \geq 0$. Therefore, we call the first projection *constrained* and the second *relaxed*. Depending on the size of the system, $\mathcal{C}$ can be too large to fully materialize, and matrix inversion is known to be unstable [19, 20, 21]. In this case, instead of direct inversion, we solve the linear systems

$$\mathcal{C}\mathcal{C}^\top X = \mathcal{C}\hat{u} - b, \tag{9}$$

$$(I + \lambda \mathcal{C}^\top \mathcal{C}) X = \hat{u} + \lambda \mathcal{C}^\top b, \tag{10}$$

and respectively compute $u = \hat{u} - \mathcal{C}^\top X$ or $u = X$, using sparse solvers such as CG [22], GMRES [23], or BiCGSTAB [24]. These solvers only require access to the matrix-vector operators, so we leverage the fact that Equations 9 and 10 can be rewritten in terms of the JVP$(v) := v \mapsto \mathcal{C}v$ and VJP$(v) := v \mapsto \mathcal{C}^\top v$ to obtain favorable scaling. Alternatively, for moderately sized systems the sparse matrix $\mathcal{C}$ can be used directly.

	Lorenz		**Kuramoto-Shivashinsky (64)**		**Navier–Stokes (64)**	
	MSE $(\times 10^{-1})\downarrow$	**Residual** $(\times 10^{-4})\downarrow$	**MSE** $(\times 10^{-2})\downarrow$	**Residual** $(\times 10^{-5})\downarrow$	**MSE** $(\times 10^{-1})\downarrow$	**Residual** $(\times 10^{-2})\downarrow$
Data-driven (baseline = MLP for Lorenz, FNO for KS/NS)						
Baseline	7.78	50.8	4.01	46.8	13	8.13
Baseline + Constrained	**7.75**	13.1	3.92	16	10.5	4.21
Baseline + Relaxed	7.76	13.3	4	39.9	12.2	3.4
Baseline + LBFGS	7.76	**1.18**	**3.91**	**4.53**	**2.63**	**0.00901**
Physics-informed (PI = PINN for Lorenz, PINO for KS/NS)						
PI	9.6	51.7	4.76	4.27	13.8	6.48
PI + Constrained	**9.57**	13.4	4.77	24.9	11.5	3.75
PI + Relaxed	9.58	13.5	4.76	3.05	13	2.94
PI + LBFGS	9.58	**1.23**	**4.75**	**1.27**	**3.21**	**0.00956**

Table 1: Results of the experiments. The evolution for KS and NS is reported on resolution 64. Lower is better. LBFGS achieves significantly lower constraint violation. In NS, where the solutions have small scale details, the MSE improvement is much more significant.

3 Experiments

We investigate three dynamical systems: the 3D Lorenz ODE [25], the 1D Kuramoto–Shivashinsky (KS) PDE [26, 27], and the 2D Navier–Stokes (NS) PDE. More details on data generation are given in App. A. In the KS and NS PDEs, we train a neural network to predict the full trajectory $x_{1:T}$ from the initial state x_0, whereas in the Lorenz system we give x_T as additional context to the network. For every equation we train two models: one that minimizes MSE only, and a physics-informed one that minimizes the MSE plus the MSE of the constraint loss $h(u) - c$. As physics-informed models, PINNs [7] are used for Lorenz, while PINOs [8] are used for KS and NS. We report test-set Mean Squared Error (MSE) and the squared L_2 norm of the constraint violation $||h(u) - c||$ (i.e. the residual) of the output of the networks without further processing, and after nonlinear and linear projections. For the NS equation, we show how the solution changes as the constraint violation reduces, and give a rough notion of the local constraint landscape. The results are shown in Table 1, with more details in App. B.

Lorenz. An MLP and a PINN-MLP are trained on the Lorenz equation. For the relaxed projection, $\lambda = 1000$ was used. All projections cut constraint violation by 70% or more and lower the MSE. The PINN-MLP has higher constraint violation despite having been trained to minimize it, as it did not train as effectively as the standard MLP under the same time and resource budget.

Kuramoto-Shivashinsky. A Fourier Neural Operator (FNO) [28] and a PINO were trained at a 64 resolution and evaluated at 64, 128, and 256 resolutions. Because KS dynamics are sufficiently resolved over these grids, the MSE is resolution-invariant, but constraint error rises with grid size. For the relaxed projection, $\lambda = 10$ was used. Projections suppress the violation at all resolutions and also reduce MSE.

Navier-Stokes. Similarly to KS, an FNO and a PINO were trained at a 64 resolution and evaluated at 64, 128, and 256. For the relaxed projection, $\lambda = ||\hat{u}||$ was used. Across resolutions, the baseline MSE stays nearly constant because large-scale structures captured by the model dominate the error. Unlike KS, NS contains smaller-scale details, which shows up as constraint violations and missing fine structure in the model output. Once those violations are reduced, the MSE drops significantly, as shown in Figure 1 and Table 1. In Figure 2 we show the local structure of the constraint by measuring its violation along the LBFGS trajectory, helping explain why LBFGS outperforms the two linearization-based projections, as seen in Table 1. They are good locally at the beginning but quickly degrade, and close to the minimum the linear approximation becomes noisy.

4 Conclusion

We demonstrated that training-free projection methods substantially outperform physics-informed training for enforcing governing equation constraints across multiple dynamical systems. LBFGS achieves near-perfect constraint satisfaction when the initial neural network output $\hat{u}$ provides a reasonable approximation, while linearization-based methods offer computational efficiency at the cost of accuracy. Our analysis reveals that linear approximations are locally effective but degrade rapidly away from the linearization point, suggesting that iterative relinearization approaches, similar to Sequential Quadratic Programming [18], could improve performance if implemented efficiently.

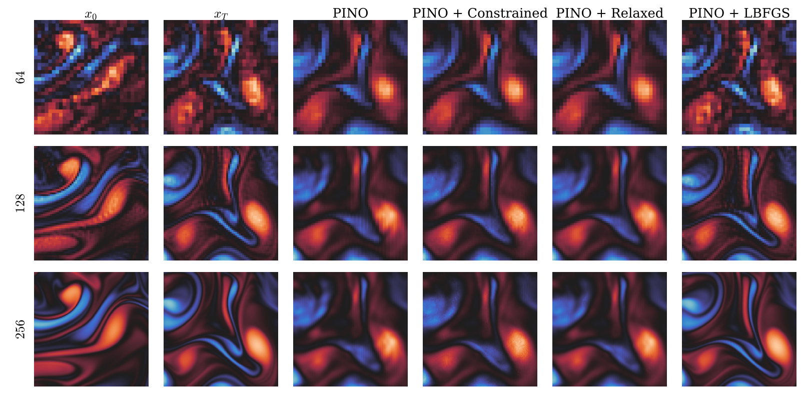

Figure 1: NS predictions comparing a PINO baseline with projection methods. Column 1 shows the initial condition x_0, Column 2 shows the ground truth x_T, and Columns 3-6 show predictions from PINO and projection methods. Only LBFGS successfully recovers the fine-scale structures; physics-informed models fail to capture details they never encountered during training.

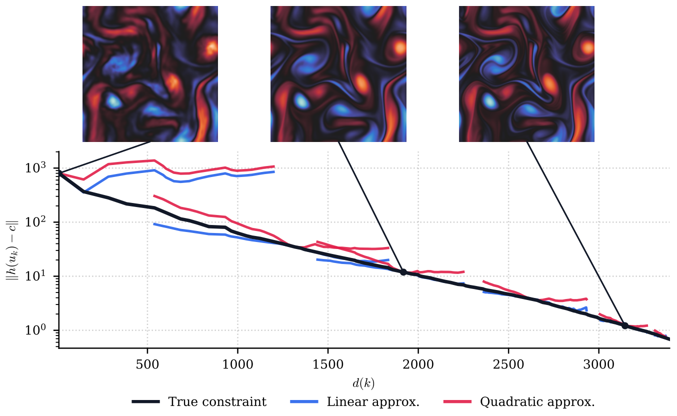

Figure 2: Constraint violation along LBFGS optimization path $u_{0:200}$ from initial guess $u_0 = \hat{u}$ to final solution $u^* = u_{200}$ (black, log scale), where the x-axis measures the size $d(k) = \sum_{i=0}^{k} \|u_{i+1} - u_i\|$ of the path $u_{0:k}$, for $k = 0$ to 200. Blue and red curves show 1st and 2nd order Taylor approximations of the violation. The curves reveal two regimes: linear approximations work well initially but become inaccurate near the solution, while quadratic approximations remain reliable throughout, explaining why LBFGS outperforms linear projection methods.

Several challenges remain for broader adoption. Current projection methods, while computationally tractable, cannot be directly integrated as differentiable layers within neural networks due to the computational cost of backpropagation through the optimization process. Reducing projection overhead is essential for scaling to larger systems and enabling end-to-end training. Additionally, extending these techniques to generative models presents opportunities for improving temporal coherence by projecting sampled trajectories onto physically consistent manifolds. Future work should focus on developing differentiable projection operators and exploring their integration with modern generative architectures.

Acknowledgments

This work was partially supported by ServicePublic de Wallonie Recherche under grant n° 2010235 – ARIAC by DIGITALWALLONIA4.AI.

References

[1] Hongkai Zheng, Wenda Chu, Bingliang Zhang, Zihui Wu, Austin Wang, Berthy T. Feng, Caifeng Zou, Yu Sun, Nikola Kovachki, Zachary E. Ross, Katherine L. Bouman, and Yisong Yue. InverseBench: Benchmarking Plug-and-Play Diffusion Priors for Inverse Problems in Physical Sciences.

[2] Liuyi Chen, Bocheng Han, Xuesong Wang, Jiazhen Zhao, Wenke Yang, and Zhengyi Yang. Machine Learning Methods in Weather and Climate Applications: A Survey. 13(21):12019.

[3] Junzhe Yin, Cristian Meo, Ankush Roy, Zeineh Bou Cher, Yanbo Wang, Ruben Imhoff, Remko Uijlenhoet, and Justin Dauwels. Precipitation Nowcasting Using Physics Informed Discriminator Generative Models.

[4] Lukas Prantl, Benjamin Ummenhofer, Vladlen Koltun, and Nils Thuerey. Guaranteed conservation of momentum for learning particle-based fluid dynamics. *ArXiv*, abs/2210.06036, 2022.

[5] Margaux Boxho, Joachim Dominique, Tariq Benarama, Michel Rasquin, Lionel Salesses, Caroline Sainvitu, Gilles Louppe, and Thomas Toulorge. Turbulent Injection assisted by Diffusion Models for Scale Resolving Simulations.

[6] Mohsen Ahmadi, Debojit Biswas, Maohua Lin, Frank D. Vrionis, Javad Hashemi, and Yufei Tang. Physics-informed machine learning for advancing computational medical imaging: Integrating data-driven approaches with fundamental physical principles. 58(10):297.

[7] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, 2017.

[8] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations, 2023.

[9] Zihan Zhou, Xiaoxue Wang, and Tianshu Yu. Generating Physical Dynamics under Priors.

[10] Afila Ajithkumar Sophiya, Akarsh K Nair, Sepehr Maleki, and Senthil K. Krishnababu. A comprehensive analysis of pinns: Variants, applications, and challenges, 2025.

[11] M. Mattheakis, P. Protopapas, D. Sondak, M. Di Giovanni, and E. Kaxiras. Physical Symmetries Embedded in Neural Networks. *arXiv e-prints*, page arXiv:1904.08991, April 2019.

[12] Gonzalo E. Constante-Flores, Hao Chen, and Can Li. Enforcing hard linear constraints in deep learning models with decision rules, 2025.

[13] Jiahe Huang, Guandao Yang, Zichen Wang, and Jeong Joon Park. DiffusionPDE: Generative PDE-Solving Under Partial Observation.

[14] Christian Jacobsen, Yilin Zhuang, and Karthik Duraisamy. CoCoGen: Physically-Consistent and Conditioned Score-based Generative Models for Forward and Inverse Problems.

[15] Giacomo Baldan, Qiang Liu, Alberto Guardone, and Nils Thuerey. Flow Matching Meets PDEs: A Unified Framework for Physics-Constrained Generation.

[16] Yulong Lu and Wuzhe Xu. Generative downscaling of PDE solvers with physics-guided diffusion models.

[17] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45(1–3):503–528, August 1989.

[18] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer series in operations research and financial engineering. Springer, New York, NY, 2. ed. edition, 2006.

[19] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, second edition, 2002.

[20] Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra, Twenty-fifth Anniversary Edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2022.

[21] Gene H. Golub and Charles F. Van Loan. *Matrix Computations - 4th Edition*. Johns Hopkins University Press, Philadelphia, PA, 2013.

[22] Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49:409–435, 1952.

[23] Youcef Saad and Martin H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[24] H. A. van der Vorst. Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13(2):631–644, 1992.

[25] Edward N. Lorenz. Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, 20(2):130–141, 1963.

[26] Yoshiki Kuramoto. Diffusion-induced chaos in reaction systems. *Progress of Theoretical Physics Supplement*, 64:346–367, 02 1978.

[27] G.I. Sivashinsky. Nonlinear analysis of hydrodynamic instability in laminar flames—i. derivation of basic equations. *Acta Astronautica*, 4(11):1177–1206, 1977.

[28] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2021.

A Appendix

A.1 LORENZ data

The Lorenz dataset is generated by integrating the 3-D Lorenz–63 [25] system

$$\dot{x} = \sigma(y - x), \quad \dot{y} = x(\rho - z) - y, \quad \dot{z} = xy - \beta z. \tag{11}$$

With $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$. Trajectories are generated on-the-fly with $x_0 \sim \mathcal{N}(0, 0.1)$. Euler integration was used with a fixed step $\mathrm{d}t = 1/512$ for $T = 512$ steps, giving a time horizon $t_{\max} = 1$. Independent random seeds create disjoint training, validation, and test splits.

A.2 KURAMOTO-SHIVASHINSKY data

The 1-D Kuramoto–Shivashinsky equation [26, 27]

$$\partial_t u + \partial_{xx} u + \partial_{xxxx} u + u\, \partial_x u = 0 \tag{12}$$

is integrated on the periodic domain $x \in [0, 64]$ with a pseudospectral solver. Spatial derivatives use the FFT on grids with $N \in \{64, 128, 256\}$ points and no de-aliasing. Time stepping was done using the first-order backward-difference formula (BDF1) with fixed step $\mathrm{d}t = 10^{-1}$ for 512 steps, giving $t_{\max} = 51.2$. We generated 65,536 train, 128 validation, and 128 test simulations. During training, we used a subset of each solution, starting at step 256 and ending at step 512, corresponding to $t \in (25.6, 51.2)$. Each implicit step is solved by Newton iterations. Initial data are random sums of ten cosine modes,

$$u_0(x) = \sum_{k=1}^{10} a_k \cos\bigl(2\pi \omega_k x/64 + \phi_k\bigr), \tag{13}$$

where amplitudes $a_k \sim \mathcal{U}(0, 1)$, frequencies $\omega_k \sim \mathcal{U}(1, 5)$ and phases $\phi_k \sim \mathcal{U}(0, 2\pi)$ are sampled independently. The dataset comprises 65,536 training, 128 validation, and 128 test simulations (total 65,792), each generated with an independent initial condition and solver seed.

A.3 NAVIER-STOKES data

The 2-D incompressible Navier–Stokes equations in vorticity form

$$\begin{aligned} \frac{\partial \omega}{\partial t} &= -\mathbf{u} \cdot \nabla \omega + \frac{1}{\mathrm{Re}} \Delta \omega + f, \\ \mathbf{u} &= \left(\frac{\partial \psi}{\partial y}, -\frac{\partial \psi}{\partial x} \right), \\ \Delta \psi &= \omega, \end{aligned} \tag{14}$$

with the forcing $f(\omega; x, y, t) = -4\cos(4y) - 0.1\omega(x, y, t)$ are integrated on the periodic domain $[0, 2\pi]^2$ with $\mathrm{Re} = 1000$ using a pseudospectral solver. Spatial derivatives use the FFT on grids with $N \in 64, 128, 256$ and $2/3$ de-aliasing. Time stepping uses the explicit Heun method with fixed step $\mathrm{d}t = 2 \times 10^{-3}$. We record 256 snapshots at uniform intervals, giving a time horizon $T = 8.192$. Initial data are zero-mean Gaussian random fields sampled at the highest resolution and projected to each grid. We generated 4096 train, 128 validation, and 128 test simulations.

B Additional results

	MSE ($\times 10^{-1}$)	**Residual** ($\times 10^{-4}$)
MLP	7.78	50.8
MLP + Constrained	**7.75**	13.1
MLP + LBFGS	7.76	**1.18**
MLP + Relaxed	7.76	13.3
MLP-PINN	9.6	51.7
MLP-PINN + Constrained	**9.57**	13.4
MLP-PINN + LBFGS	9.58	**1.23**
MLP-PINN + Relaxed	9.58	13.5

Table 2: Lorenz results. (Left) SE between GTs and generated trajectories, averaged over all trajectories and timesteps. (Right) Mean squared L_2 norm of the constraint violation of generated trajectories.

	MSE ($\times 10^{-2}$)			**Residual** ($\times 10^{-5}$)		
	64	128	256	64	128	256
FNO	4.01	3.97	3.98	46.8	267	273
FNO + Constrained	3.92	3.72	3.73	16	71.4	74.6
FNO + LBFGS	**3.91**	**3.65**	**3.66**	**4.53**	**3.73**	**3.77**
FNO + Relaxed	4	3.87	3.88	39.9	182	185
PINO	4.76	5.56	5.55	4.27	874	896
PINO + Constrained	4.77	5.42	5.01	24.9	368	239
PINO + LBFGS	**4.75**	**4.67**	**4.66**	**1.27**	**1.11**	**1.18**
PINO + Relaxed	4.76	5.18	5.17	3.05	502	509

Table 3: Results for Kuramoto-Shivashinsky. (Left) SE between GTs and generated trajectories, averaged over all trajectories and timesteps. (Right) Mean squared L_2 norm of the constraint violation of generated trajectories. Constraint error increases with the resolution unless projection is performed.

	MSE $(\times 10^{-1})$			**Residual** $(\times 10^{-2})$		
	64	128	256	64	128	256
FNO	13	9.09	8.85	8.13	6.58	6.53
FNO + Constrained	10.5	7.4	7.36	4.21	3.1	2.97
FNO + LBFGS	**2.63**	**2.52**	**2.58**	**0.00901**	**0.00793**	**0.00813**
FNO + Relaxed	12.2	8.41	8.18	3.4	2.9	2.88
PINO	13.8	9.91	9.61	6.48	6.17	6.13
PINO + Constrained	11.5	8.27	8.13	3.75	3.07	2.97
PINO + LBFGS	**3.21**	**3.01**	**3.02**	**0.00956**	**0.00829**	**0.00832**
PINO + Relaxed	13	9.24	8.95	2.94	2.76	2.74

Table 4: Results for Navier-Stokes. (Left) SE between GTs and generated trajectories, averaged over all trajectories and timesteps. (Right) Mean squared L_2 norm of the constraint violation of generated trajectories.