PINNverse: Accurate parameter estimation in differential equations from noisy data with constrained physics-informed neural networks

Marius Almanstötter[1,2*], Roman Vetter[1,2] and Dagmar Iber[1,2]

[1*]Department of Biosystems Science and Engineering, ETH Zürich, Schanzenstrasse 44, Basel, 4056, Switzerland.

[2]Swiss Institute of Bioinformatics, Schanzenstrasse 44, Basel, 4056, Switzerland.

*Corresponding author(s). E-mail(s): marius.almanstoetter@bsse.ethz.ch;
Contributing authors: vetterro@ethz.ch; dagmar.iber@bsse.ethz.ch;

Abstract

Parameter estimation for differential equations from measured data is an inverse problem prevalent across quantitative sciences. Physics-Informed Neural Networks (PINNs) have emerged as effective tools for solving such problems, especially with sparse measurements and incomplete system information. However, PINNs face convergence issues, stability problems, overfitting, and complex loss function design. Here we introduce PINNverse, a training paradigm that addresses these limitations by reformulating the learning process as a constrained differential optimization problem. This approach achieves a dynamic balance between data loss and differential equation residual loss during training while preventing overfitting. PINNverse combines the advantages of PINNs with the Modified Differential Method of Multipliers to enable convergence on any point on the Pareto front. We demonstrate robust and accurate parameter estimation from noisy data in four classical ODE and PDE models from physics and biology. Our method enables accurate parameter inference also when the forward problem is expensive to solve.

Keywords: Physics-Informed Neural Network, Differential equations, Inverse problem, Noisy data, Parameter estimation, Constraint Differential Optimization

Introduction

Accurate modeling of complex phenomena in science and engineering often necessitates solving differential equations whose parameters characterize essential physical properties. Directly measuring these parameters can be difficult or impractical, which has prompted the development of inverse methods designed to infer unknown parameters from observational data.

Traditionally, inverse problems have been tackled using two main methodological frameworks. The frequentist approach seeks parameter estimates by maximizing the likelihood [1, 2] of observed data given the model predictions [3–5]. However, likelihood optimization is complicated by loss landscapes that exhibit multiple modes and spurious local minima, resulting in solutions that strongly depend on initial guesses [6]. Consequently, multiple local minimizations from different starting points or global optimization strategies are typically employed, considerably increasing computational costs [7]. In contrast, Bayesian methods consider parameters as random variables [8], using Markov chain Monte Carlo algorithms to estimate posterior distributions, which naturally quantify parameter uncertainties [9, 10]. However, such approaches require extensive forward model evaluations, rely heavily on priors, and can face convergence challenges [11, 12].

Conventionally, forward problems that are formulated as differential equations (DEs) are solved using numerical schemes such as Runge–Kutta, Finite Differences, Finite Volumes, or Finite Elements. Recently, Physics-Informed Neural Networks (PINNs) [13, 14] have emerged as an attractive alternative. PINNs leverage deep neural networks in conjunction with automatic differentiation and gradient-based optimization to approximate the solutions of DEs. By incorporating DE constraints directly into the training objectives—including governing equations, initial and boundary conditions—PINNs learn to approximate the underlying physical laws. This strategy is unsupervised, mesh-free, and has successfully addressed a variety of forward and inverse problems as an alternative

to classical numerical methods [15–23]. Nevertheless, employing PINNs can still present challenges, such as complex loss landscapes due to soft enforcement of constraints and difficulties encountered in scenarios involving shock waves [24, 25]. Consequently, several enhancements have been proposed over recent years to enforce physical laws more effectively and to expand the applicability of PINNs. They include respecting temporal causality [26], including Fourier features in the first layer [27], curriculum-based training [25], adaptive resampling strategies during training [28, 29], adaptive weights during training [30–32] and augmenting the training process with additional equations [33].

When training PINNs, a data loss term is introduced that competes with the physics-based loss component. This interplay has recently prompted increased interest in multi-objective optimization techniques and the exploration of Pareto fronts within the context of PINNs [34–39]. These studies typically emphasize on adaptive weighting schemes to manage the relative contributions of various loss terms, or apply evolutionary algorithms, such as NSGA-II [40], explicitly targeting multi-objective optimization. However, these evolutionary methods often involve substantially greater computational expense compared to conventional PINN training. The adaptive weighting schemes introduce additional, often sensitive, hyperparameters that themselves require careful tuning, adding another layer of complexity to the problem.

In inverse problem scenarios with substantial noise in the observational data, simultaneously minimizing data and physics losses to zero is unfeasible, necessitating a deliberate trade-off between these competing objectives. Furthermore, equal balancing of losses is typically undesirable, as it can result in overfitting the data noise instead of accurately capturing the underlying physics.

To overcome these limitations, we propose a constrained optimization framework tailored to identify optimal PINN solutions that accurately fit observational data while strictly enforcing physical constraints. Previous investigations into constrained optimization approaches within PINNs have demonstrated networks structured to exactly fulfill initial and boundary conditions, while employing augmented Lagrangian methods to minimize physics and data losses [41]. Additionally, stochastic augmented Lagrangian methods have been effectively applied to train PINNs [42]. However, the augmented Lagrangian requires nested loops for training, which are typically not employed in deep learning, and increases computational complexity. To overcome this limitation, we propose employing the Modified Differential Method of Multipliers (MDMM) [43], an optimization strategy that simultaneously updates the neural network parameters, differential equation parameters, and associated Lagrange multipliers in parallel. This method is fully compatible with modern state-of-the-art optimizers, such as Adam [44] or Adan [45], and importantly, it does not incur additional computational costs beyond those of standard PINN training methods. We demonstrate on four classical examples that the proposed training paradigm outperforms standard PINNs, especially with high levels of noise in the data, and traditional optimizers such as the Nelder–Mead [46] when the initial parameter guess is inaccurate.

Results

PINNverse restates the training paradigm

PINNverse is a reformulation of the training paradigm for PINNs that addresses their fundamental limitations in solving inverse problems. The key innovation lies in the handling of multiple competing objectives that arise in PINNs. Traditional PINNs employ a composite loss function that linearly combines data-fitting terms with physics constraints using fixed weights (Fig. 1, purple box; Methods). This weighted-sum approach creates a multi-objective optimization problem that struggles with complex Pareto fronts (the set of solutions that minimize the loss), particularly those with non-convex regions where conventional gradient-based methods generally fail to find balanced solutions. A minimization with stochastic gradient descent algorithms lets even trajectories originating from the same initial point consistently converge towards different convex regions of the solution space depending on the used learning rate and starting point. The resulting reachable Pareto front represents only the convex portions of the solution landscape, which can lead to a strong emphasis of the NN solution on data enforcement (small data loss), and thus overfitting.

PINNverse restructures this approach by reformulating the problem as constrained optimization rather than weighted summation (Methods). We designate the data-fitting term as the primary objective while transforming physics-based terms (differential equations, initial conditions, and boundary conditions) into explicit constraints (Fig. 1, green box). This shift from penalty-based regularization to explicit constraint enforcement substantially changes how physics information is incorporated into the learning process.

To solve this constrained problem, we employ the Modified Differential Method of Multipliers, which enables convergence to any point on the Pareto front, including those in concave regions that traditional methods would miss. The primary advantage of the MDMM lies in its inherent capability to converge towards saddle points of the loss landscape, which constitute the optimal solutions within a Lagrangian formulation [47]. This property enables it to efficiently identify balanced solutions that simultaneously fulfill data-fitting objectives and physics-based constraints, ensuring that neither is compromised in favor of the other. Unlike conventional approaches in which certain

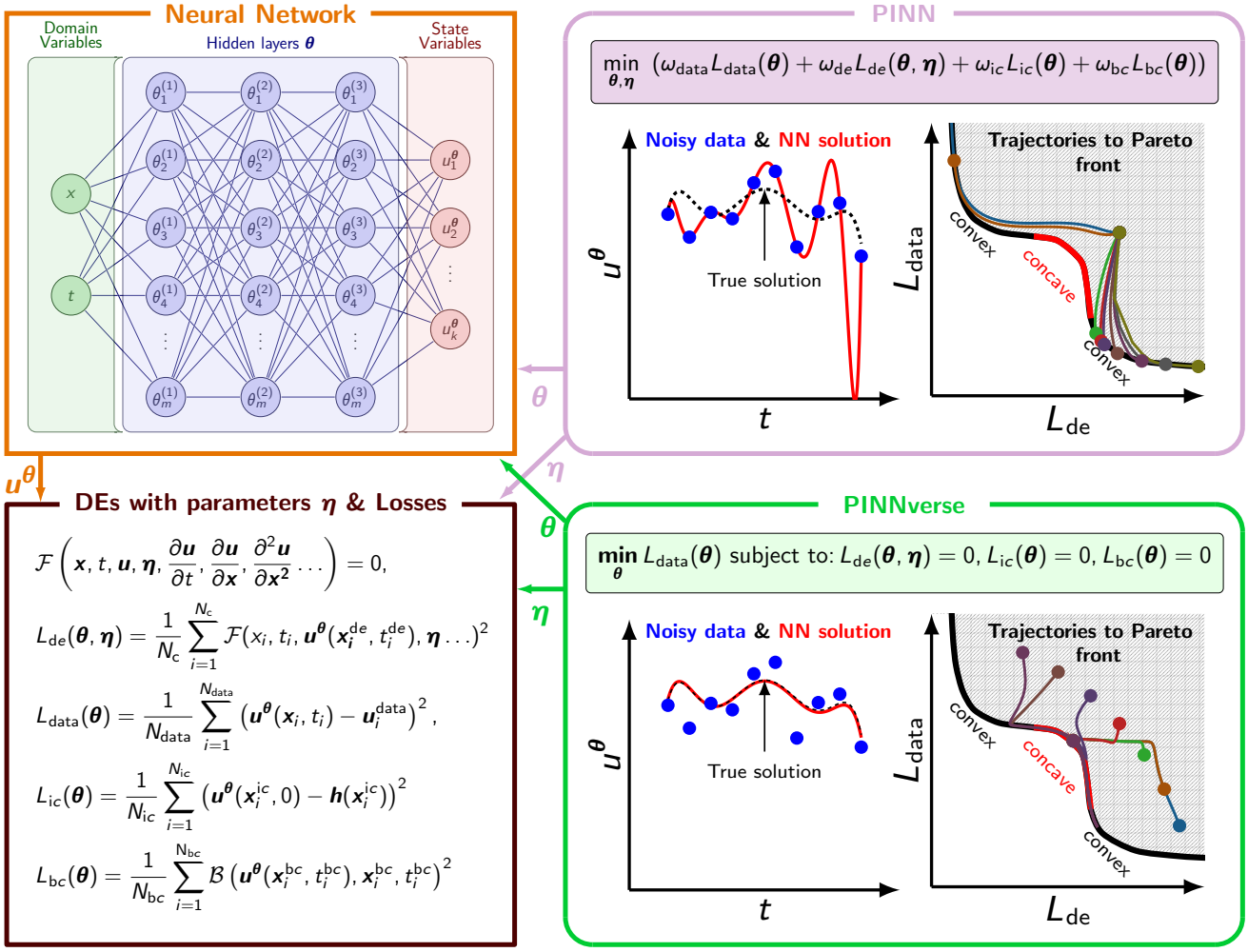

Fig. 1 Schematic representation of the difference between PINN and PINNverse. When approximating solutions to differential equations (DEs) in residual form $\mathcal{F} = 0$ with a neural network (NN), the architecture utilizes hidden layers (parameter set $\boldsymbol{\theta}$) to map input variables including spatial coordinates $\boldsymbol{x}$ and time t to the solution space of state variables, represented by a k-dimensional vector of functions $\boldsymbol{u}^{\boldsymbol{\theta}}$ (orange box). The training process of a PINN (purple box) minimizes a composite loss function incorporating terms penalizing deviations of the predicted solution $\boldsymbol{u}^{\boldsymbol{\theta}}$ from observed noisy data points $\boldsymbol{u}_i^{\text{data}}$ (blue dots) (L_{data}), as well as terms that penalize violations of the differential equations (L_{de}) and of the initial and/or boundary conditions (L_{ic}, L_{bc}). The predicted NN solution typically suffers from overfitting and misses non-convex parts (solid red line) of the Pareto front (solid black line). Examples of trajectories starting from initial points within the feasible region (dashed area) and leading to this front are schematically visualized for a 2D subspace. With PINNverse (green box), the data, IC and BC losses are formulated as external constraints under which the optimization is carried out, which avoids overfitting and allows the trajectories to converge also on convex parts of the front.

loss terms may dominate others during optimization, PINNverse ensures that all physics constraints are properly enforced while simultaneously fitting the available data.

Experimental design

We evaluate PINNverse against PINNs and the widely adopted Nelder–Mead optimization algorithm [46], specifically using the implementation in the SciPy library [48]. Since only Nelder–Mead and PINNverse naturally support bounds, we use reasonable parameter bounds only for these two methods, and ensure parameter positivity by exponential transformation for the PINN. Our experimental framework encompasses four benchmark problems—two ordinary differential equations (ODEs) and two partial differential equations (PDEs)—to assess the generalizability, robustness and accuracy of our approach.

To isolate the effects of our methodological contribution against the PINN, we employ identical neural network architectures, optimization algorithms, learning rate schedules, and initialization procedures across both PINNverse and standard PINN implementations (Methods). In all cases we have used the Adan optimizer [45] for training. Our application of MDMM in PINNverse is based on a specific PyTorch implementation (https://github.com/crowsonkb/mdmm). This controlled experimental design ensures comparability and fairness: observed performance differences can be attributed solely to the modified gradient update strategy of PINNverse.

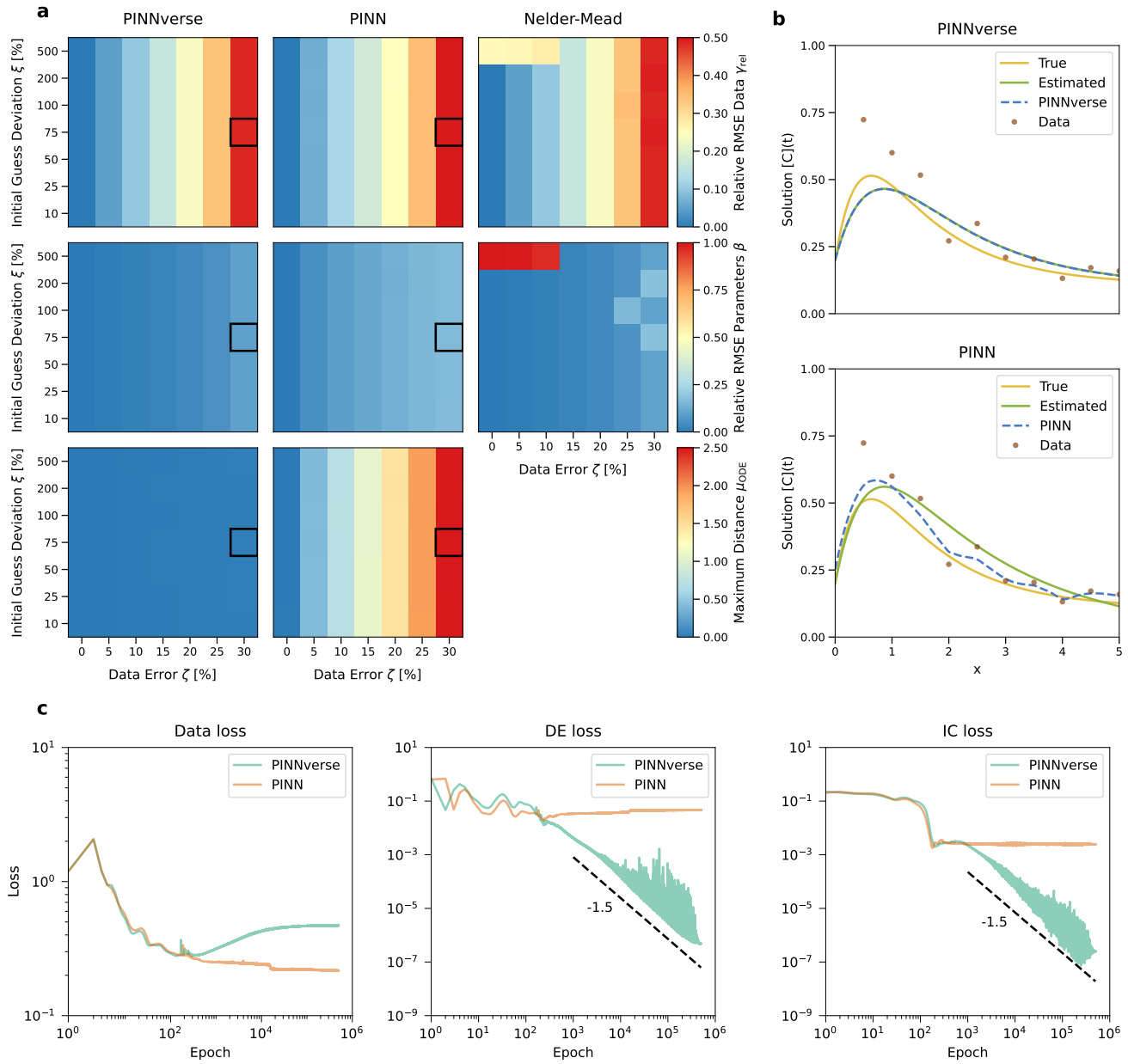

Fig. 2 Parameter estimation performance in the kinetic reaction ODE model. a, Heatmaps depicting performance metrics across varying noise levels in the data, ζ, and deviations in initial parameter guesses, ξ (Methods). The black square highlights the scenario $\zeta = 25\%$, $\xi = 75\%$ analyzed in detail in subsequent panels. **b**, Comparison of trajectories for species $[C](t)$, generated using estimated parameters (green curve), true parameters (yellow curve), neural network predictions (blue curve) and the corresponding noisy observational data (brown dots). **c**, Training loss evolution for PINNverse and conventional PINN. Data, differential equation (DE) and initial condition (IC) losses are depicted. For PINNverse, a power law was fitted to the DE and IC losses after 1000 epochs (shifted dashed lines) with indicated exponents.

Kinetic reaction model

We begin by examining a reduced version of a more detailed nonlinear reaction model previously employed in the context of parameter estimation using PINNs [49], involving four distinct species, labeled A, B, C and D:

$$A \underset{k_2}{\overset{k_1}{\rightleftharpoons}} B + C, \quad C \underset{k_4}{\overset{k_3}{\rightleftharpoons}} D,$$

where the double arrows indicate reversible reactions at indicated rates k_1 to k_4. The system of ODEs describing the temporal evolution of the concentrations reads

$$\begin{aligned}
\frac{d[A]}{dt} &= -k_1[A] + k_2[B][C], \\
\frac{d[B]}{dt} &= k_1[A] - k_2[B][C], \\
\frac{d[C]}{dt} &= k_1[A] - k_2[B][C] - k_3[C] + k_4[D], \\
\frac{d[D]}{dt} &= k_3[C] - k_4[D].
\end{aligned}$$

with initial conditions

$$[A](0) = 1.0, \quad [B](0) = 0,$$
$$[C](0) = 0.2, \quad [D](0) = 0.$$

ODE model systems of this type are widely used to characterize chemical reactions and dynamic interactions in complex biological systems. For instance, minimal models of glucose regulation of this form have been used to estimate insulin sensitivity by analyzing plasma glucose and insulin dynamics during glucose tolerance tests [50]. They are generally popular toy models for parameter inference, e.g., in the form of the oscillatory Lotka–Volterra equations [51].

We generated synthetic data by numerically solving the system at ten different time points using known (ground-truth) kinetic parameters $\boldsymbol{\eta}_{\text{true}} = [k_1, k_2, k_3, k_4] = [1.5, 0.5, 1, 0.1]$. Parameter bounds $\boldsymbol{\eta}^{\text{lower}} = [0, 0, 0, 0]$ and $\boldsymbol{\eta}^{\text{upper}} = [10, 4, 7, 0.7]$ were used in PINNverse and Nelder–Mead. The PINN and PINNverse were trained for 500,000 epochs.

Figure 2a quantifies the performance of each method with respect to data quality and uncertainty in parameter initialization. The conventional PINNs, PINNverse, and Nelder–Mead all achieve similar relative root mean squared error (RMSE, γ_{rel}, Methods) between model predictions and noisy observations (top row), growing as expected with increasing measurement noise (ζ, Methods). However, Nelder–Mead struggles when initialized far from the true parameter values ($\xi = 500\%$; Methods), getting trapped in local minima. PINNverse achieves a mean improvement factor over Nelder–Mead of approximately 370 in this scenario.

The second row of Fig. 2a evaluates the parameter estimation accuracy β (Methods). While the conventional PINN yields accurate parameters exclusively under noise-free conditions and quickly deteriorates as measurement noise increases, PINNverse and Nelder–Mead generally retain higher accuracy. PINNverse achieves a 3.8-fold improvement in β compared to standard PINNs (mean across all tested scenarios). Nonetheless, Nelder–Mead fails when initialized far from the true parameter values, and occasionally also underperforms at higher noise levels ($\zeta \geq 25\%$), where PINNverse demonstrates a 1.2-fold improvement in β.

The bottom row of Fig. 2a illustrates the maximum deviation μ_{ODE} (Methods) between the neural network predictions and the true solutions at the inferred parameters—a metric in which the advantages of PINNverse are particularly evident. While conventional PINNs violate physical constraints as measurement noise increases, PINNverse consistently remains conforming. On average across all noise-affected scenarios, PINNverse yields an 88-fold improvement in μ_{ODE} compared to the standard PINN.

To illustrate the overfitting tendencies of conventional PINNs and the physics-conforming behavior of PINNverse, a representative scenario ($\zeta = 25\%$, $\xi = 75\%$, black squares in Fig. 2a) is shown in Fig. 2b. Predictions of concentration $[C](t)$ reveal perfect alignment between PINNverse predictions (blue curve) and numerical solutions computed from the inferred parameters (green curve). Despite considerable observational noise (brown dots), PINNverse closely approximates the underlying true dynamics (yellow curve). In contrast, conventional PINN predictions deviate substantially from numerical solutions, strongly overfitting the noisy data at the expense of physical accuracy.

This pronounced bias of PINNs toward data loss minimization, neglecting physics and initial condition losses due to inherent non-convexities in the multi-objective landscape, is also apparent in convergence plots (Fig. 2c). In contrast, PINNverse achieves substantial simultaneous reductions in physics and initial condition losses consistent with algebraic convergence beyond 1000 epochs: $L_{\text{de,ic}} \sim \text{epoch}^{-a}$ with exponents $a = 1.5255 \pm 0.0009$ for DE and $a = 1.5108 \pm 0.0005$ (s.e.) for IC.

FitzHugh–Nagumo model

Originally introduced as a simplification of the Hodgkin–Huxley model [52], the FitzHugh–Nagumo (FHN) equations [53, 54] capture essential neuronal phenomena such as excitability and refractoriness. Their mathematical tractability makes them a powerful tool for analyzing neuronal firing patterns and stochastic behavior. Accurate parameter estimation in the FHN model is essential for quantitatively interpreting and predicting neuronal responses to different stimuli. Techniques utilizing noisy voltage-clamp data have effectively recovered intrinsic neuronal parameters [55], while maximum likelihood methods based on spike timing have characterized neuronal responses without amplitude information [56]. Recently, approximate Bayesian computation approaches with structure-preserving numerical schemes have robustly estimated parameters from stochastic neuronal data, enhancing the model's applicability [57]. Additionally, the FHN equations have gained prominence in physics-informed neural network literature [58, 59], which is why they are our next benchmark.

We use the classical two-dimensional spatially homogeneous form of the FitzHugh–Nagumo model:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = u - \frac{u^3}{3} - v,$$
$$\frac{\mathrm{d}v}{\mathrm{d}t} = \frac{u + a - bv}{r},$$

with initial conditions

$$u(0) = 0, \quad v(0) = 0$$

where $u(t)$ is the excitable (membrane potential-like) variable and $v(t)$ represents the slow recovery variable.

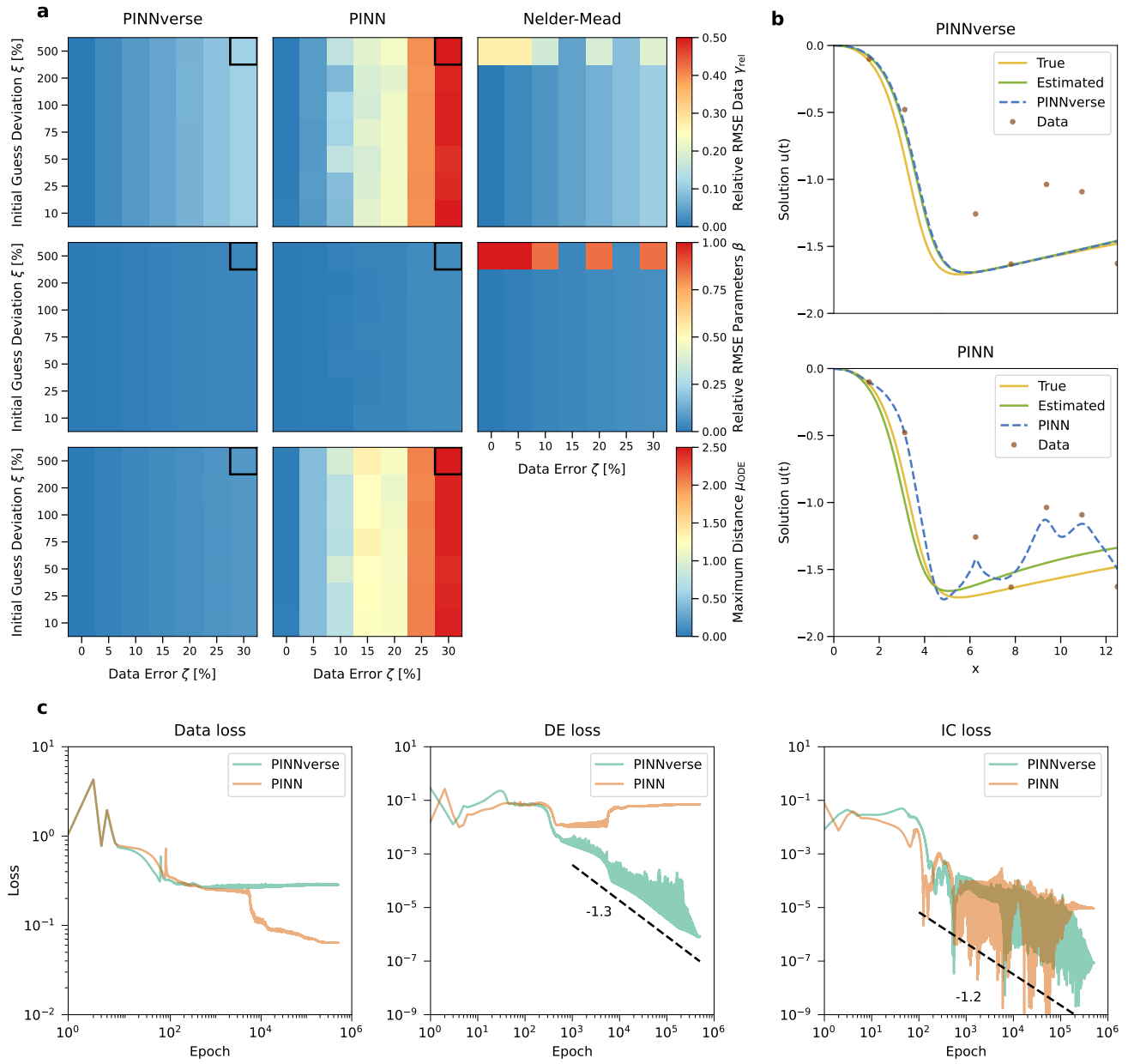

Fig. 3 Parameter estimation performance in the FitzHugh–Nagumo ODE model. a, Heatmaps depicting performance metrics across varying noise levels in the data, ζ, and deviations in initial parameter guesses, ξ (Methods). The black square highlights the scenario $\zeta = 25\%$, $\xi = 500\%$ analyzed in detail in subsequent panels. **b**, Comparison of trajectories for the excitable variable $u(t)$, generated using estimated parameters (green curve), true parameters (yellow curve), neural network predictions (blue curve), and the corresponding noisy observational data (brown dots). **c**, Training loss evolution for PINNverse and conventional PINN. Data, differential equation (DE) and initial condition (IC) losses are depicted. For PINNverse, a power law was fitted to the DE and IC losses after 1000 epochs (shifted dashed lines) with indicated exponents.

Parameters a, b, and r regulate threshold dynamics, recovery rates, and timescale separation, respectively.

To subject the optimization methods to a particularly tough challenge with sparse data, we use only seven data points as input, calculated from the solution for $\boldsymbol{\eta}_{\text{true}} = [a, b, r] = [0.7, 0.8, 12.5]$ plus noise (Methods). Parameter bounds $\boldsymbol{\eta}^{\text{lower}} = [0, 0, 0]$ and $\boldsymbol{\eta}^{\text{upper}} = [10, 10, 100]$ were set. The same number of training epochs was used as for the kinetic reaction model.

The observed trend is broadly consistent with the reaction model, although with several noteworthy differences (Fig. 3a). The standard PINN consistently exhibits substantially higher relative data error (γ_{rel}) compared to PINNverse, particularly as the measurement noise level increases. Across all tested scenarios, PINNverse achieves a 4.7-fold mean improvement in γ_{rel}. While the Nelder–Mead algorithm performs similarly well with good initial guesses, it again fails when initialized far from the true parameters ($\xi = 500\%$), where PINNverse outperforms it by a factor of approximately 221.

PINNverse also yields a substantially better parameter estimate than the PINN (Fig. 3a, middle row), with a 3.5-fold improvement in β (mean over all tested scenarios). Under the challenging condition of an initial parameter guess deviation of $\xi = 500\%$, the Nelder–Mead algorithm fails to find the right parameter also in this benchmark. Furthermore, the solutions predicted by the standard PINN deviate markedly from the numerical reference solution, whereas PINNverse predictions closely match it (Fig. 3a, bottom row).

In Fig. 3b we provide a representative example illustrating these observations through component $u(t)$ for the scenario with largest data noise and deviation in the initial guess (black squares in Fig. 3a). The PINNverse prediction closely follows the true trajectory, while that of the PINN deviates by overfitting the noisy data. Also when the estimated parameters are used in a numerical solver, the computed solution match the true solution only for PINNverse, not for the PINN.

In convergence plots (Fig. 3c), a similar picture is observed for the FHN equations as in the reaction model: The PINN systematically fails to reduce the DE and IC losses after about 1000 epochs. PINNverse, on the other hand, reduces them further, approximately following power laws $L_{\mathrm{de,ic}} \sim \mathrm{epoch}^{-a}$ with slightly lower algebraic convergence rates $a = 1.3340 \pm 0.0007$ and $a = 1.159 \pm 0.001$ (s.e.) for DE and IC, respectively.

Fisher–KPP model

The Fisher–KPP equation, pioneered by Fisher and Kolmogorov, Petrovsky and Piskunov [60, 61], is the first of two PDEs in our test suite. It has emerged as a fundamental mathematical framework to model spatiotemporal population dynamics across diverse biological contexts, from two-dimensional cell spreading during skeletal tissue regeneration [62] to motility and proliferation in circular barrier assays relevant to wound healing [63] and malignant cell proliferation influenced by TGF-β signaling [64]. Recently, PINNs have also been explored for solving this equation [65, 66]. We use its one-dimensional form with the following initial and boundary conditions:

$$\begin{aligned} \frac{\partial u}{\partial t} &= D\frac{\partial^2 u}{\partial x^2} + \rho u(1-u) \\ u(x,0) &= \frac{1}{10}e^{-x} \\ \frac{\partial u(x,t)}{\partial x} &= 0 \quad \text{at} \quad x \in \{0, 10\} \end{aligned}$$

Here, $u(x,t)$ represents the normalized population density at position x and time t. D is the diffusion coefficient characterizing the random motility of the cells. The nonlinear source term $\rho u(1-u)$ accounts for logistic growth, representing cell proliferation at rate ρ, constrained by carrying capacity limitations. The exponential initial condition represents a localized cell distribution gradually decreasing with distance. Zero-flux (Neumann) boundary conditions ensure that there is no cell transfer across the domain boundaries. A particularly challenging feature of this PDE is that it admits traveling wave solutions whose propagation velocity is determined by the underlying model parameters.

To generate the synthetic measurement data, we used ground-truth kinetic parameters $\boldsymbol{\eta}^{\mathrm{true}} = [\mathrm{D}, \rho] = [0.5, 1]$ and perturbed the exact solution at a total of 18 points in time and space as before, at observation times $t = 1$ and $t = 2$. $\boldsymbol{\eta}^{\mathrm{lower}} = [0.1, 0.5]$ and $\boldsymbol{\eta}^{\mathrm{upper}} = [0.5, 6]$ served as parameter bounds for PINNverse and Nelder–Mead. We trained the PINNverse and PINN for 300,000 epochs.

To assess the performance on the Fisher–KPP, we quantified the *absolute* deviation between model prediction and data, γ_{abs} (Methods), because solution values can approach zero. PINNverse is seen to consistently and substantially outperform the PINN whenever measurement noise is non-zero (Fig. 4a), by all metrics considered. PINNverse achieves a 12-fold mean improvement in γ_{rel} and 48-fold in β across all tested scenarios. Here, a peculiar weakness of the normal PINN becomes apparent: While the solution learned by the PINN strongly overfits the noisy data (Fig. 4b), the numerical solution obtained by solving the PDE with the parameters inferred by the PINN does not approximate the data well, because the parameter estimate is poor (Fig. 4a, middle column). PINNverse, on the other hand, robustly finds the optimal parameters and reasonably approximates the data (Fig. 4a, left column).

In convergence plots (Fig. 4c), we observe that PINNverse manages to reduce all physical loss terms (DE, IC and BC), while the classical PINN starts overemphasizing the data loss beyond about 1000 epochs at the expense of struggling with the physical losses. With PINNverse, approximate algebraic convergence in the number of epochs is observed: $L_{\mathrm{de,ic,bc}} \sim \mathrm{epoch}^{-a}$ with $a = 1.422 \pm 0.001$, $a = 1.265 \pm 0.002$ and $a = 1.413 \pm 0.002$ (s.e.) for DE, IC and BC, respectively.

Burgers' equation

As a final demanding PDE benchmark, we test PINNverse on Burgers' equation [67], which has been widely employed to study various nonlinear wave phenomena, including water infiltration into soil [68], nonlinear acoustic shock-wave propagation validated by experimental N-waveforms from spark sources [69], and shock-wave propagation in the human brain resulting from explosive blasts [70]. Burgers' equation has frequently been featured in recent physics-informed neural network literature [13, 28, 33, 71]. We use its viscous form

$$\begin{aligned} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} &= \nu\frac{\partial^2 u}{\partial x^2}, \\ u(0,x) &= -\sin(\pi x) \end{aligned}$$

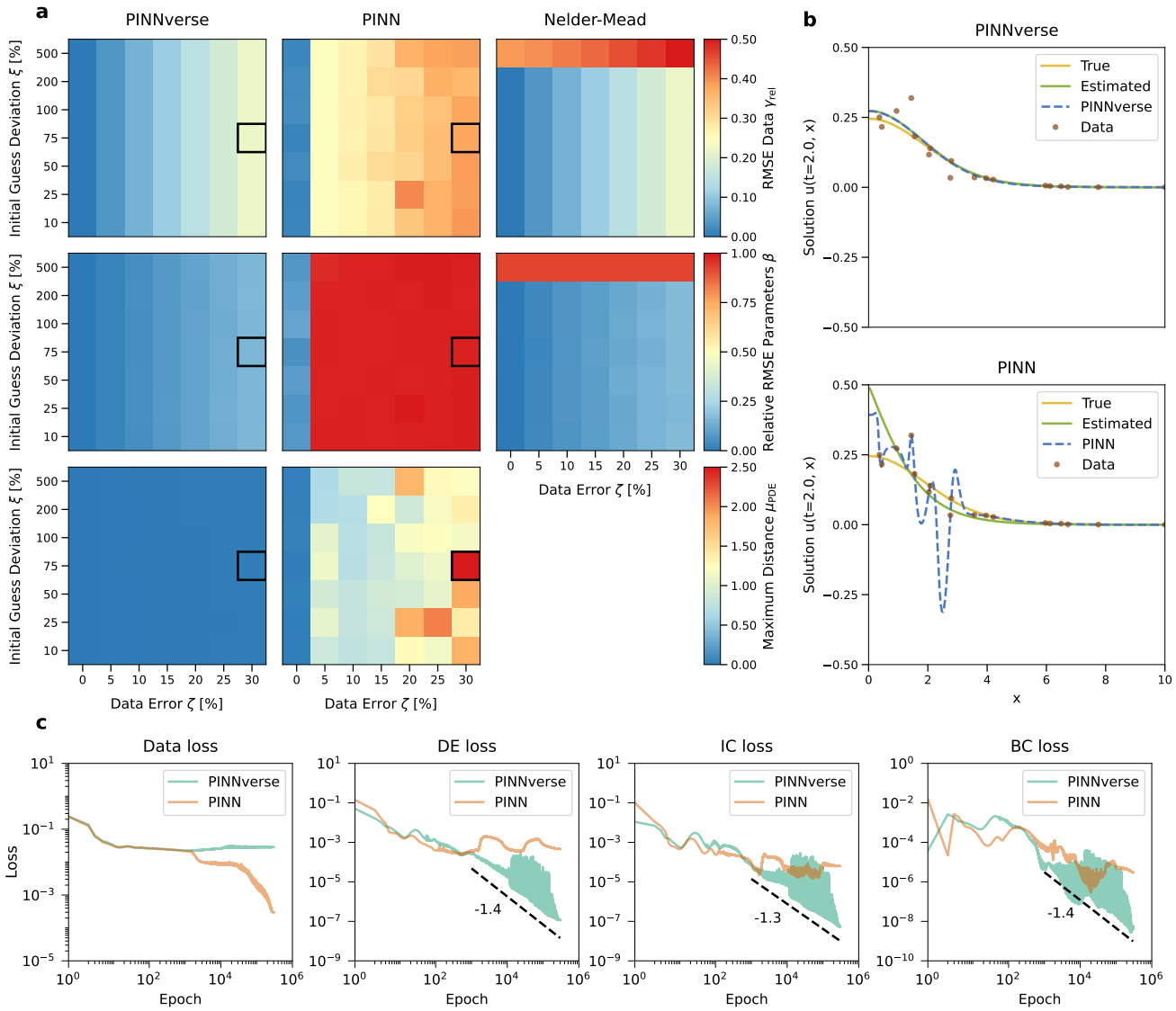

Fig. 4 Parameter estimation performance in the Fisher–KPP PDE model. a, Heatmaps depicting performance metrics across varying noise levels in the data, ζ, and deviations in initial parameter guesses, ξ (Methods). The black square highlights the scenario $\zeta = 25\%$, $\xi = 75\%$ analyzed in detail in subsequent panels. **b**, Comparison of trajectories for the cell concentration $u(x)$ at time point $t = 2$, generated using estimated parameters (green curve), true parameters (yellow curve), neural network predictions (blue curve), and the corresponding noisy observational data (brown dots). **c**, Training loss evolution for PINNverse and conventional PINN. Data, differential equation (DE), initial condition (IC) and boundary condition (BC) losses are depicted. For PINNverse, a power law was fitted to the DE, IC and BC losses after 1000 epochs (shifted dashed lines) with indicated exponents.

$$u(t, x) = 0 \quad \text{at} \quad x \in \{-1, 1\}$$

where $u(t, x)$ represents a velocity field (or an analogous quantity, such as traffic density), and ν is the viscosity (or diffusion) coefficient. When $\nu > 0$, the solutions are smooth, balancing nonlinearity and diffusion. In the limit $\nu \to 0$, shock waves can form, making Burgers' equation a prototypical model to study shock formation and related phenomena in fluid dynamics.

To generate our synthetic measurements, we selected a ground-truth parameter $\boldsymbol{\eta}^{\text{true}} = \nu = 0.01$, placing the system firmly within the shock wave regime. Lower and upper parameter limits were set to $\boldsymbol{\eta}^{\text{lower}} = 0$ and $\boldsymbol{\eta}^{\text{upper}} = 0.07$, respectively. A total of 14 perturbed data points were recorded at observation times $t = 0.2$ and $t = 0.4$. To overcome the inherent limitation of standard neural networks in representing high-frequency spatial variations, known as spectral bias [27], we applied a Fourier feature mapping to the spatial input coordinate (Methods). 150,000 epochs were used for training.

All three optimization strategies achieve similar RMSEs (Fig. 5a, top row). However, in terms of parameter estimation accuracy (Fig. 5a, middle row), PINNverse consistently surpasses the standard PINN across all examined scenarios, achieving, on average, a 33-fold improvement in β. The PINN delivers reliable parameter estimates exclusively under noise-free conditions; as soon as measurement noise is introduced, the accuracy of parameter estimates drastically declines,

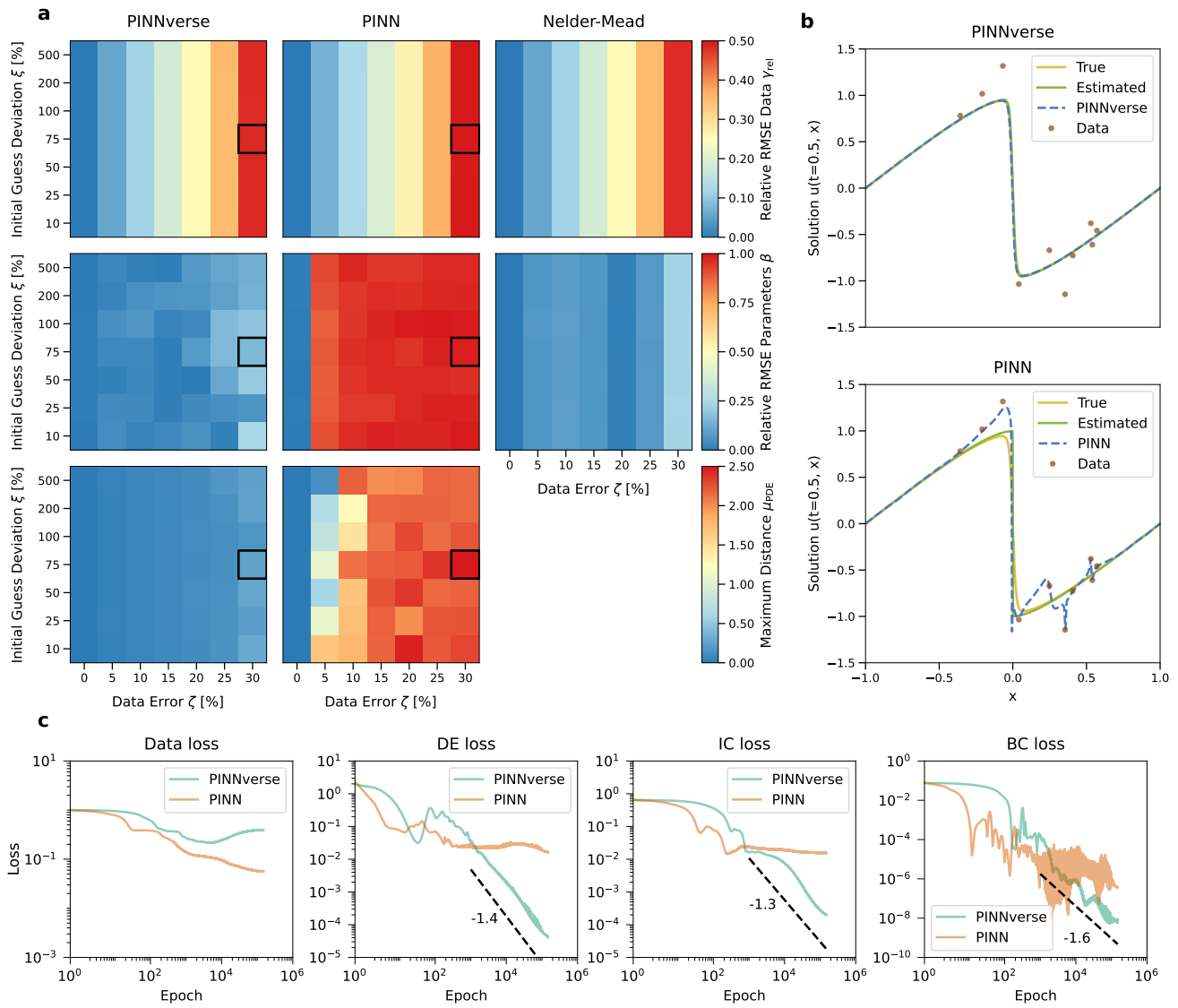

Fig. 5 Parameter estimation performance in Burgers' PDE model. a, Heatmaps depicting performance metrics across varying noise levels in the data, ζ, and deviations in initial parameter guesses, ξ (Methods). The black square highlights the scenario $\zeta = 25\%$, $\xi = 75\%$ analyzed in detail in subsequent panels. **b**, Comparison of trajectories for the dependent variable $u(x)$ at time point $t = 0.5$, generated using estimated parameters (green curve), true parameters (yellow curve), neural network predictions (blue curve), and the corresponding noisy observational data (brown dots). **c**, Training loss evolution for PINNverse and conventional PINN. Data, differential equation (DE), initial condition (IC) and boundary condition (BC) losses are depicted. For PINNverse, a power law was fitted after 1000 epochs to the DE, IC and BC losses (shifted dashed lines) with indicated exponents.

underscoring the difficulty associated with parameter estimation in shock wave regimes, where inaccurate data can hinder recovery of system dynamics.

Although Nelder–Mead does not become trapped in local minima within the tested range of initial guesses, it consistently produces less accurate parameter estimates compared to PINNverse. Quantitatively, PINNverse achieves a two-fold mean improvement in β across all evaluated scenarios.

PINNverse accurately captures the shock even under noisy conditions, whereas the PINN approach displays substantial deviations and overfitting (Fig. 5b). Consistent with all previous benchmarks, PINNverse reduces the physical losses in the Fisher–KPP equation approximately algebraically with increasing training effort: $L_{\mathrm{de,ic,bc}} \sim \mathrm{epoch}^{-a}$ with $a = 1.4348 \pm 0.0003$, $a = 1.2761 \pm 0.0006$ and $a = 1.643 \pm 0.001$ (s.e.) for DE, IC and BC, respectively (Fig. 5c).

Discussion

In the wake of the recent boom of deep learning and artificial intelligence, Physics-Informed Neural Networks have emerged as promising tools that can learn to reproduce and predict solutions to differential equations [15–19]. However, our investigation reveals fundamental limitations within the standard PINN paradigm when applied to parameter inferrence from noisy observational data subject to physical constraints. The core

issue lies in the inherent trade-off built into the conventional PINN training process: a simultaneous pursuit of data fidelity and physics compliance, often mediated by fixed weighting schemes that struggle to navigate complex Pareto fronts.

We have introduced PINNverse, a reformulated training paradigm that treats parameter estimation as a constrained optimization problem. By leveraging the Modified Differential Method of Multipliers, we achieve robust convergence to any point on the Pareto front, including those residing within concave regions typically inaccessible to standard descent-based methods. This represents a critical distinction: PINNverse does not just approximate data; it seeks solutions that simultaneously satisfy both the observational constraints and the governing physical equations—a fundamentally different approach than the weighted-sum strategy employed in conventional PINNs.

This key difference became apparent in the solutions predicted by the neural networks: standard PINNs produced substantially distorted solutions that overfit the noisy data, whereas PINNverse accurately reproduced the underlying dynamics. Similar trends were observed across all models considered. PINNverse's ability to converge on well-balanced regions of the Pareto front was being particularly pronounced in the Burgers equation, where shock wave formation is a defining characteristic. We observed superlinear convergence in the physical constraints (exponents 1.2–1.6) for PINNverse across all studied systems, whereas regular PINNs did not converge.

PINNverse was less sensitive to initial parameter guesses than Nelder–Mead. Moreover, PINNverse naturally supports parameter bounds by simply adding them to the constraints of the differential equation, initial condition and boundary condition loss. Additional constraints can be employed in the same way, further restricting the space of possible solutions. Remarkably, achieving these benefits requires minimal changes—just a few lines of code—to convert an existing standard PINN implementation to PINNverse, and does not inflict noteworthy additional computational cost.

Our findings echo the established efficacy of MDMM in other areas. Recent applications include refining CMS FastSim particle simulation accuracy [72], Bayesian Entropy Neural Networks with physics constraints [73], autonomous robotic cutting [74] and the inference/pruning of large pre-trained language models [75, 76].

Despite its demonstrated strengths, PINNverse, as a neural network-based method, inherently depends on careful selection and tuning of hyperparameters such as network architecture (number of layers, neurons per layer) and learning rate. However, in this study, explicit hyperparameter tuning was not conducted, suggesting an inherent resilience of the method to suboptimal parameter configurations.

A further strength of PINNverse, although not explicitly demonstrated here, is its computational consistency across diverse problem domains. Unlike traditional numerical methods which typically possess polynomial time complexity in the used spatio-temporal discretization, the computational performance of PINNverse depends primarily on neural network training time. With PINNverse, inverse problems with differential equations can be solved without requiring a single forward evaluation. This suggests a complexity crossover threshold beyond which our method becomes computationally advantageous compared to conventional numerical solvers, particularly for problems requiring fine discretizations, involving complex geometries, multiphysics interactions, or high-dimensional parameter spaces. This theoretical advantage deserves quantitative investigation in future research on increasingly complex models and datasets.

Our investigation utilized a "vanilla" PINN architecture as the foundational element of the PINNverse framework. However, the inherent modularity of this approach facilitates seamless integration with various known enhancements to conventional PINNs, e.g. respecting temporal causality [26], curriculum-based training [25], including Fourier features in the first layer [27], adaptive resampling strategies during training [28, 29], and augmenting the training process with additional equations [33]. We anticipate that systematic exploration of these supplementary techniques could yield further refinements in accuracy, computational efficiency, and overall performance.

Methods

Solving the inverse problem with Physics-Informed Neural Networks

We consider a general differential equation (DE) system represented in a residual form given by

$$\begin{aligned} \mathcal{F}(\boldsymbol{x}, t, \boldsymbol{u}, \boldsymbol{\eta}, \boldsymbol{u}_t, \nabla \boldsymbol{u}, ...) = 0, \quad & \boldsymbol{x} \in \Omega, \quad t \in [0, T] \\ \mathcal{B}(\boldsymbol{u}(\boldsymbol{x}, t), \boldsymbol{x}, t) = 0, \quad & \boldsymbol{x} \in \partial\Omega, \ t \in [0, T] \\ \boldsymbol{u}(\boldsymbol{x}, 0) = \boldsymbol{h}(\boldsymbol{x}), \quad & \boldsymbol{x} \in \Omega \end{aligned}$$

where $\Omega \subseteq \mathbb{R}^n$ represents the spatial domain with boundary $\partial\Omega$, and $\boldsymbol{u} : \Omega \times [0, T] \to \mathbb{R}^m$ denotes the solution field over the space-time domain. The operator $\mathcal{F}(\cdot)$ is a spatio-temporal differential operator encapsulating the governing physics of the system, which may incorporate multiple parameters $\boldsymbol{\eta} \in \mathbb{R}^p$ and various spatial and temporal derivatives of $\boldsymbol{u}$. The boundary conditions are imposed through the spatio-temporal operator $\mathcal{B}(\boldsymbol{u}, \boldsymbol{x}, t)$, which acts on the solution at the domain boundary $\partial\Omega$. The initial solution $\boldsymbol{h}(\boldsymbol{x})$ prescribes the state of the system at time $t = 0$ throughout Ω.

The forward problem consists of determining the solution $\boldsymbol{u}(\boldsymbol{x}, t; \boldsymbol{\eta})$, given known parameters $\boldsymbol{\eta}$. The inverse problem regards the parameters $\boldsymbol{\eta}$ as unknown

quantities, requiring inference from observational data at discrete spatio-temporal locations. To formalize this inverse scenario, we assume the availability of a dataset comprising N_{data} observations:

$$\{(\boldsymbol{x}_i^{\text{data}}, t_i^{\text{data}}, \boldsymbol{u}_i^{\text{data}})\}_{i=1}^{N_{\text{data}}},$$

where each datum consists of a coordinate pair $(\boldsymbol{x}_i^{\text{data}}, t_i^{\text{data}})$ and the corresponding observed solution $\boldsymbol{u}_i^{\text{data}}$. The goal is to approximate the solution using a NN model $\boldsymbol{u}^{\boldsymbol{\theta}}(\boldsymbol{x}, t)$, parameterized by neuronal weights and biases collectively denoted by $\boldsymbol{\theta}$. The optimal parameters are determined by minimizing the discrepancy between the network predictions and observed data in some chosen metric:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L_{\text{data}}(\boldsymbol{\theta}).$$

We seek the minimum in a least-squares sense here. The data loss function $L_{\text{data}}(\boldsymbol{\theta})$ can be formulated as either an absolute loss function

$$L_{\text{data}}(\boldsymbol{\theta}) = \sqrt{\frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} \left(\boldsymbol{u}^{\boldsymbol{\theta}}(\boldsymbol{x_i}, t_i) - \boldsymbol{u}_i^{\text{data}}\right)^2}$$

or, unless the data approaches zero, as a relative loss function

$$L_{\text{data}}(\boldsymbol{\theta}) = \sqrt{\frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} \left(\frac{\boldsymbol{u}^{\boldsymbol{\theta}}(\boldsymbol{x_i}, t_i) - \boldsymbol{u}_i^{\text{data}}}{\boldsymbol{u}_i^{\text{data}}}\right)^2}$$

where the vector division is taken element-wise.

To impose conformity with the physical laws described by the DE, a residual loss is introduced as

$$L_{\text{de}}(\boldsymbol{\theta}, \boldsymbol{\eta}) = \frac{1}{N_c} \sum_{i=1}^{N_c} \mathcal{F}(\boldsymbol{x_i}^{\text{de}}, t_i^{\text{de}}, \boldsymbol{u}^{\theta}(\boldsymbol{x_i}^{\text{de}}, t_i^{\text{de}}), \boldsymbol{\eta}, \ldots)^2,$$

where $\{\boldsymbol{x}_i^{\text{de}}, t_i^{\text{de}}\}_{i=1}^{N_c}$ are collocation points sampled within $\Omega \times (0, T)$.

Additionally, losses for initial and boundary conditions are defined by

$$L_{\text{bc}}(\boldsymbol{\theta}) = \frac{1}{N_{\text{bc}}} \sum_{i=1}^{N_{\text{bc}}} \mathcal{B}\left(\boldsymbol{u}^{\boldsymbol{\theta}}(\boldsymbol{x}_i^{\text{bc}}, t_i^{\text{bc}}), \boldsymbol{x}_i^{\text{bc}}, t_i^{\text{bc}}\right)^2$$

$$L_{\text{ic}}(\boldsymbol{\theta}) = \frac{1}{N_{\text{ic}}} \sum_{i=1}^{N_{\text{ic}}} \left(\boldsymbol{u}^{\boldsymbol{\theta}}(\boldsymbol{x}_i^{\text{ic}}, 0) - \boldsymbol{h}(\boldsymbol{x}_i^{\text{ic}})\right)^2$$

where $\{\boldsymbol{x}_i^{\text{bc}}, t_i^{\text{bc}}\}_{i=1}^{N_{\text{bc}}}$ are boundary condition points in $\partial\Omega \times [0, T]$ and $\{\boldsymbol{x}_i^{\text{ic}}, 0\}_{i=1}^{N_{\text{ic}}}$ are initial conditional points in $\Omega \times \{t = 0\}$.

Traditionally, a composite total loss function is then formulated as [13]

$$\begin{aligned} L_{\text{pinn}}(\boldsymbol{\theta}, \boldsymbol{\eta}) = \omega_{\text{data}} L_{\text{data}}(\boldsymbol{\theta}) + \omega_{\text{de}} L_{\text{de}}(\boldsymbol{\theta}, \boldsymbol{\eta}) \\ + \omega_{\text{ic}} L_{\text{ic}}(\boldsymbol{\theta}) + \omega_{\text{bc}} L_{\text{bc}}(\boldsymbol{\theta}) \end{aligned}$$

where ω_{data}, ω_{de}, ω_{ic} and ω_{bc} are weights that balance the partial losses. Following common practice [13], all weights were set to one here.

PINNs leverage automatic differentiation to compute derivatives of the output variables $\boldsymbol{u}$ with respect to $\boldsymbol{x}$ and t, enabling evaluation of the differential operators $\mathcal{F}(\cdot)$ and boundary operators $\mathcal{B}(\cdot)$. The parameter update using gradient descent is performed as

$$\begin{aligned} \boldsymbol{\theta}^{(k+1)} &= \boldsymbol{\theta}^{(k)} - \alpha \nabla_{\boldsymbol{\theta}} L_{\text{pinn}}(\boldsymbol{\theta}^{(k)}, \boldsymbol{\eta}^{(k)}) \\ \boldsymbol{\eta}^{(k+1)} &= \boldsymbol{\eta}^{(k)} - \alpha \nabla_{\boldsymbol{\eta}} L_{\text{pinn}}(\boldsymbol{\theta}^{(k)}, \boldsymbol{\eta}^{(k)}) \end{aligned}$$

where $\alpha > 0$ is the learning rate and k the iteration index.

Pareto optimality

With noisy experimental data, the composite loss function encompasses multiple competing objectives. This represents a multi-objective optimization problem

$$\min_{\Psi} \boldsymbol{L}(\boldsymbol{\Psi}).$$

where we seek to simultaneously optimize all components of a total loss vector

$$\begin{aligned} \boldsymbol{L}(\boldsymbol{\Psi}) &= \boldsymbol{L}(\boldsymbol{\theta}, \boldsymbol{\eta}) \\ &= (L_{\text{data}}(\boldsymbol{\theta}), L_{\text{de}}(\boldsymbol{\theta}, \boldsymbol{\eta}), L_{\text{ic}}(\boldsymbol{\theta}), L_{\text{bc}}(\boldsymbol{\theta}))^{\text{T}}. \end{aligned}$$

Finding a single parameter vector $\boldsymbol{\Psi}$ that simultaneously minimizes all loss components is generally infeasible. To formalize this challenge, we adopt the concept of Pareto optimality. A parameter vector $\boldsymbol{\Psi}$ is considered (globally) Pareto optimal, if no other parameter vector $\boldsymbol{\Psi}$ exists that achieves non-increasing values across all loss functions while strictly improving at least one loss component. The collection of all candidate solutions constitutes the feasible region.

The subset of optimal objective function values represents the Pareto front [77] (Fig. 1, solid black curve), which manifests in two fundamental geometric configurations: convex and concave. A convex Pareto front is distinguished by the property that for any two points $\boldsymbol{a}$ and $\boldsymbol{b}$ on the front and any scalar $\kappa \in [0, 1]$, there exists a point $\boldsymbol{c}$ on the front such that $\kappa||\boldsymbol{a}|| + (1 - \kappa)||\boldsymbol{b}|| \geq ||\boldsymbol{c}||$. Conversely, a concave Pareto front satisfies the inequality $\kappa||\boldsymbol{a}|| + (1 - \kappa)||\boldsymbol{b}|| \leq ||\boldsymbol{c}||$.

The performance of gradient-based optimization methods is intrinsically linked to the geometry of the Pareto front. Specifically, when minimizing linearly weighted objectives, gradient descent converges exclusively to solutions located on the convex regions of

the Pareto front [78]. Consequently, regardless of the positive weighting parameters selected, points within non-convex segments of the front cannot be attained, as they do not correspond to minima of any weighted sum objective function. In contrast, for purely convex Pareto fronts, gradient-based optimization can theoretically converge to any desired point along the curve through appropriate adjustment of the weighting parameters. However, precisely controlling the final solution point along the front via weight selection is often non-trivial, as the mapping between weights and Pareto points is highly sensitive to the front's local curvature [78].

In practical applications with neural networks, Pareto fronts typically have mixed shapes with both convex and concave regions, making the tuning of PINNs notoriously difficult [39].

Inverse problem as constraint optimization

To address the limitations of standard gradient-based methods on complex Pareto fronts, we reformulate the PINN training process as a constrained optimization problem. Rather than treating all loss terms equally, we designate the data-fitting term as the primary objective while transforming the physics-based terms into constraints:

$$\begin{aligned} \underset{\boldsymbol{\theta}}{\text{minimize}} \quad & L_{\text{data}}(\boldsymbol{\theta}) \\ \text{subject to} \quad & L_i(\boldsymbol{\theta}, \boldsymbol{\eta}) = 0, \quad i \in \mathcal{I}_{\text{e}} = \{\text{de}, \text{ic}, \text{bc}\} \\ & \eta_j^{\text{lower}} \leq \eta_j \leq \eta_j^{\text{upper}}, \quad j \in \mathcal{I}_{\text{b}} = \{1, \ldots, p\} \end{aligned}$$

The parameters $\eta_j \in$ represent differential equation parameters constrained within physically plausible bounds $[\eta_j^{\text{lower}}, \eta_j^{\text{upper}}]$. These bounds ensure that the solution remains physically meaningful and prevent the neural network from exploring invalid regions.

To handle the bound constraints efficiently, we introduce an infeasibility function

$$V_j(\eta_j(\boldsymbol{\theta})) = \max(\eta_j^{\text{lower}}, \min(\eta_j(\boldsymbol{\theta}), \eta_j^{\text{upper}})) - \eta_j(\boldsymbol{\theta})$$

that measures constraint violations. This allows us to express the Lagrangian as

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\lambda}, \boldsymbol{\chi}) = L_{\text{data}}(\boldsymbol{\theta}) + \sum_{i \in \mathcal{I}_e} \lambda_i L_i(\boldsymbol{\theta}, \boldsymbol{\eta}) \\ + \sum_{j \in \mathcal{I}_b} \chi_j V_j(\eta_j), \end{aligned}$$

where λ_i and χ_j represent the Lagrange multipliers of equality and parameter bound constraints. The optimal set of neural network parameters is then obtained through a min-max formulation:

$$(\boldsymbol{\theta}, \boldsymbol{\eta})^* = \arg\min_{\boldsymbol{\theta}, \boldsymbol{\eta}} \left(\max_{\boldsymbol{\lambda} \geq 0, \boldsymbol{\chi} \geq 0} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\lambda}, \boldsymbol{\chi}) \right).$$

The target solution for the Lagrangian min-max formulation is inherently a saddle point [79]. However, such points are generally not attractors for standard gradient-based optimizers [43].

Optimization approach of PINNverse

To ultimately overcome these limitations, we employ the Modified Differential Method of Multipliers (MDMM) [43]. To the best of our knowledge, this represents the first application of the MDMM in the context of PINNs. MDMM is an optimization algorithm derived from the augmented Lagrangian formulation, also known as the Method of Multipliers. This formulation introduces quadratic penalty terms alongside the standard Lagrange multiplier terms to improve convergence properties. For the PINNverse problem, the augmented Lagrangian is defined as

$$\begin{aligned} \mathcal{L}_{\text{A}}(\boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\lambda}, \boldsymbol{\chi}, \boldsymbol{c}) = L_{\text{data}}(\boldsymbol{\theta}) \\ + \sum_{i \in \mathcal{I}_{\text{e}}} \left(\lambda_i L_i(\boldsymbol{\theta}, \boldsymbol{\eta}) + \frac{c_i}{2} L_i^2(\boldsymbol{\theta}, \boldsymbol{\eta}) \right) \\ + \sum_{j \in \mathcal{I}_{\text{b}}} \left(\chi_j V_j(\eta_j) + \frac{d_j}{2} V_j^2(\eta_j) \right), \end{aligned}$$

where $c_i > 0$ and $d_j > 0$ are the penalty coefficients for the constraints. Larger values enforce constraints more strictly. In this study, all penalty parameters were set to unity ($c_i = d_j = 1$).

A key distinction of MDMM from standard sequential augmented Lagrangian methods lies in its update dynamics. MDMM proposes simultaneous updates for both the primal variables ($\boldsymbol{\theta}$, $\boldsymbol{\eta}$) and the Lagrange multipliers ($\boldsymbol{\lambda}$, $\boldsymbol{\chi}$). For a gradient descent update this reads

$$\begin{aligned} \boldsymbol{\theta}^{(k+1)} &= \boldsymbol{\theta}^{(k)} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{A}}(\boldsymbol{\theta}^{(k)}, \boldsymbol{\eta}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\chi}^{(k)}) \\ \boldsymbol{\eta}^{(k+1)} &= \boldsymbol{\eta}^{(k)} - \alpha \nabla_{\boldsymbol{\eta}} \mathcal{L}_{\text{A}}(\boldsymbol{\theta}^{(k)}, \boldsymbol{\eta}^{(k)}, \boldsymbol{\lambda}^{(k)}, \boldsymbol{\chi}^{(k)}). \end{aligned}$$

Here $\alpha > 0$ represents the learning rate that controls the step size during each iteration of the gradient descent. Crucially, in MDMM the Lagrange multipliers are updated via gradient ascent:

$$\begin{aligned} \lambda_i^{(k+1)} &= \lambda_i^{(k)} + \alpha\, L_i(\boldsymbol{\theta}^{(k)}, \boldsymbol{\eta}^{(k)}), \quad i \in \mathcal{I}_{\text{e}} \\ \chi_j^{(k+1)} &= \chi_j^{(k)} + \alpha\, V_j(\eta_j^{(k)}), \quad j \in \mathcal{I}_{\text{b}} \end{aligned}$$

The inclusion of the quadratic penalty term, governed by c_i, d_j, is essential. As established in optimization theory [43, 80], for sufficiently large penalty parameters, the Hessian of the augmented Lagrangian with respect to the primal variables ($\nabla^2_{\boldsymbol{\theta}, \boldsymbol{\xi}} \mathcal{L}_{\text{A}}$) becomes positive definite in the subspace tangent to the constraints near a constrained minimum satisfying standard second-order sufficiency conditions. This induces local convexity and transforms the constrained minimum into an attractor for the dynamics, mitigating

the saddle-point issues associated with the standard Lagrangian that hinder simple gradient descent [43]. Note that we still need gradient ascent for the Lagrange multipliers, since the convexity only holds for the primal variables.

Consequently, MDMM offers robust convergence towards a constrained minimum for sufficiently large c_i and d_j, suitable learning rate α, and initialization within the basin of attraction. Notably, this minimum can be any point on the Pareto front, even in the non-convex region.

In the theoretical derivation presented above, gradient updates were illustrated using stochastic gradient descent for simplicity. However, any gradient-based optimization algorithm is compatible with the MDMM framework. Motivated by this flexibility, we adopt the recently proposed Adan optimizer [45], an adaptive optimization technique grounded in Nesterov Momentum Estimation, specifically tailored for rapid and stable convergence in non-convex optimization landscapes.

Training

For all presented results, both the standard PINN and PINNverse were trained using neural networks comprising two hidden layers, each consisting of 20 neurons, with hyperbolic tangent activation functions. We employed a learning rate scheduler characterized by an initial linear decay from $\alpha = 10^{-2}$ down to 10^{-4} until reaching the last 30,000 epochs, after which the learning rate was kept constant at $\alpha = 10^{-4}$. For discretization, $N_{\text{de}} = 16,384$ collocation points were uniformly distributed across the interior of the temporal or spatio-temporal domains using a Sobol sequence [81]. An exception was the FitzHugh–Nagumo model, for which we used $N_{\text{de}} = 10,000$. In the two PDEs, additional collocation points, specifically $N_{\text{ic}} = N_{\text{bc}} = 1,024$, were allocated to enforce the initial and boundary conditions, respectively.

For Burgers' equation, Fourier features were used in the training. This technique transforms the coordinate into a higher-dimensional feature space using sinusoidal basis functions. Ten such basis functions corresponding to distinct frequencies were employed in our network. This augmented spatial representation, concatenated with the temporal coordinate, served as the network input, thereby enhancing its capability to resolve the sharp gradients of shock wave dynamics.

Evaluation of accuracy

We evaluated method performance under realistic conditions by introducing heteroscedastic Gaussian noise to the data,

$$\hat{y} \sim \mathcal{N}(y, \zeta y)$$

with noise levels ζ up to 30%. Additionally, we used substantially perturbed initial guesses for parameter initialization,

$$\boldsymbol{\eta}^{\text{start}} = (1 + \xi)\boldsymbol{\eta}^{\text{true}}$$

with relative deviations ξ up to 500%.

To quantitatively evaluate solution accuracy, we define the maximum distance metric μ. For ODE problems, this metric is formulated as

$$\mu_{\text{ODE}} = \max_{\substack{t\in[0,T] \\ i\in\{1,\dots,m\}}} \left| u_i^{\text{NN}}(t;\boldsymbol{\theta},\boldsymbol{\eta}) - u_i^{\text{true}}(t;\boldsymbol{\eta}^{\text{true}}) \right|$$

where $u_i^{\text{NN}}(t;\boldsymbol{\theta})$ the neural network prediction with parameters $\boldsymbol{\theta}$ for the i-th solution component at time t, and $u_i^{\text{true}}(t;\boldsymbol{\eta}^{\text{true}})$ denotes the corresponding true solution with parameters $\boldsymbol{\eta}^{\text{true}}$ obtained via high-precision numerical methods. For PDE problems, we extend this metric to incorporate spatial dimensions:

$$\mu_{\text{PDE}} = \max_{\substack{t\in\mathcal{T} \\ \boldsymbol{x}\in\Omega \\ i\in\{1,\dots,m\}}} \left| u_i^{\text{NN}}(t,\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\eta}) - u_i^{\text{true}}(t,\boldsymbol{x};\boldsymbol{\eta}^{\text{true}}) \right|$$

where Ω is the spatial domain and $\mathcal{T}$ represents the discrete set of measured time points. Note that for the PDE case we only consider the discrete time points where we have measurements, not the whole time domain. A well-trained model that adheres to the underlying physics should yield μ values approaching zero.

To assess the parameter estimation performance of the three techniques, we computed the relative root mean squared error between the true parameters and the estimated parameters:

$$\beta = \sqrt{\frac{1}{p}\sum_{j=1}^{p}\left(\frac{\eta_j^{\text{true}} - \eta_j^{\text{est}}}{\eta_j^{\text{true}}}\right)^2}$$

where p denotes the total number of parameters in the differential equation.

Additionally, we evaluated the model performance by comparing the noisy observed data with the predictions obtained by solving the differential equations using the estimated parameters in absolute and relative terms:

$$\gamma_{\text{abs}} = \sqrt{\frac{1}{N_{\text{data}}}\sum_{j=1}^{N_{\text{data}}}\left(\hat{\boldsymbol{y}}_j - \boldsymbol{u}^{\text{pred}}(t_j,\boldsymbol{x}_j;\boldsymbol{\eta}^{\text{est}})\right)^2}$$

$$\gamma_{\text{rel}} = \sqrt{\frac{1}{N_{\text{data}}}\sum_{j=1}^{N_{\text{data}}}\left(\frac{\hat{\boldsymbol{y}}_j - \boldsymbol{u}^{\text{pred}}(t_j,\boldsymbol{x}_j;\boldsymbol{\eta}^{\text{est}})}{\hat{\boldsymbol{y}}_j}\right)^2}$$

where N_{data} represents the total number of data points, $\hat{\boldsymbol{y}}_j$ denotes the j-th noisy measurement vector, and $\boldsymbol{u}^{\text{pred}}(t_j,\boldsymbol{x}_j;\boldsymbol{\eta}^{\text{est}})$ is the predicted vector at the corresponding space-time point using the estimated parameters $\boldsymbol{\eta}^{\text{est}}$.

Data availability. No new data was generated for this study.

Code availability. The complete PyTorch code, which which all our results can be reproduced, is freely available as a git repository at https://git.bsse.ethz.ch/iber/Publications/2025_almanstoetter_pinnverse.

Acknowledgements. None.

Competing interests. The authors declare no competing interests.

References

[1] R.A. Fisher, On the Mathematical Foundations of Theoretical Statistics. Philos. Trans. R. Soc. Lond. A **222**, 309–368 (1922). https://doi.org/10.1098/rsta.1922.0009

[2] J. Aldrich, R. A. Fisher and the Making of Maximum Likelihood 1912–1922. Stat. Sci. **12**, 162–176 (1997). https://doi.org/10.1214/ss/1030037906

[3] I.J. Myung, Tutorial on maximum likelihood estimation. J. Math. Psychol. **47**, 90–100 (2003). https://doi.org/10.1016/S0022-2496(02)00028-7

[4] A. Raue, C. Kreutz, T. Maiwald, J. Bachmann, M. Schilling, U. Klingmüller, J. Timmer, Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. Bioinformatics **25**, 1923–1929 (2009). https://doi.org/10.1093/bioinformatics/btp358

[5] F. Fröhlich, B. Kaltenbacher, F.J. Theis, J. Hasenauer, Scalable Parameter Estimation for Genome-Scale Biochemical Reaction Networks. PLoS Comput. Biol. **13**, e1005331 (2017). https://doi.org/10.1371/journal.pcbi.1005331

[6] W. Tu, R.W. Mayne, Studies of multi-start clustering for global optimization. Int. J. Numer. Methods Eng. **53**, 2239–2252 (2002). https://doi.org/10.1002/nme.400

[7] A.F. Villaverde, F. Fröhlich, D. Weindl, J. Hasenauer, J.R. Banga, Benchmarking optimization methods for parameter estimation in large kinetic models. Bioinformatics **35**, 830–838 (2019). https://doi.org/10.1093/bioinformatics/bty736

[8] H. Jeffreys, *Theory of Probability* (Clarendon Press, 1939)

[9] A.M. Stuart, Inverse problems: A Bayesian perspective. Acta Numer. **19**, 451–559 (2010). https://doi.org/10.1017/S0962492910000061

[10] D. Luengo, L. Martino, M. Bugallo, V. Elvira, S. Särkkä, A survey of Monte Carlo methods for parameter estimation. EURASIP J. Adv. Signal Process. **2020**, 25 (2020). https://doi.org/10.1186/s13634-020-00675-6

[11] Z. Fang, C. Da Silva, R. Kuske, F.J. Herrmann, Uncertainty quantification for inverse problems with weak partial-differential-equation constraints. Geophysics **83**, R629–R647 (2018). https://doi.org/10.1190/geo2017-0824.1

[12] M.K. Cowles, B.P. Carlin, Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review. J. Am. Stat. Assoc. **91**, 883–904 (1996). https://doi.org/10.1080/01621459.1996.10476956

[13] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. **378**, 686–707 (2019). https://doi.org/10.1016/j.jcp.2018.10.045

[14] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning. Nat. Rev. Phys. **3**, 422–440 (2021). https://doi.org/10.1038/s42254-021-00314-5

[15] E. Jamili, V. Dua, Parameter estimation of partial differential equations using artificial neural network. Comput. Chem. Eng. **147**, 107221 (2021). https://doi.org/10.1016/j.compchemeng.2020.107221

[16] Z. Mao, A.D. Jagtap, G.E. Karniadakis, Physics-informed neural networks for high-speed flows. Comput. Methods Appl. Mech. Eng. **360**, 112789 (2020). https://doi.org/10.1016/j.cma.2019.112789

[17] S. Cai, Z. Mao, Z. Wang, M. Yin, G.E. Karniadakis, Physics-informed neural networks (PINNs) for fluid mechanics: a review. Acta Mech. Solida Sin. **37**, 1727–1738 (2022). https://doi.org/10.1007/s10409-021-01148-1

[18] S. Cai, Z. Wang, S. Wang, P. Perdikaris, G.E. Karniadakis, Physics-Informed Neural Networks for Heat Transfer Problems. Int. J. Heat Mass Transf. **143**, 060801 (2021). https://doi.org/10.1115/1.4050542

[19] S. Cuomo, V.S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, F. Piccialli, Scientific Machine Learning Through Physics-Informed Neural Networks: Where we are and What's Next. SIAM J. Sci. Comput. **92**, 88 (2022). https://doi.org/10.1007/s10915-022-01939-z

[20] Q. He, D. Barajas-Solano, G. Tartakovsky, A.M. Tartakovsky, Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport. Adv. Water Resour. **141**, 103610 (2020). https://doi.org/10.1016/j.advwatres.2020.103610

[21] J. Garay, J. Dunstan, S. Uribe, F. Sahli Costabal, Physics-informed neural networks for parameter estimation in blood flow models. Comput. Biol. Med. **178**, 108706 (2024). https://doi.org/10.1016/j.compbiomed.2024.108706

[22] A.M. Tartakovsky, C. Ortiz Marrero, P. Perdikaris, G.D. Tartakovsky, D. Barajas-Solano. Learning Parameters and Constitutive Relationships with

Physics Informed Deep Neural Networks (2018). Preprint at https://arxiv.org/abs/1808.03398

[23] Z. Wei, J.C. Wong, N. Sung, A. Gupta, C.C. Ooi, P.H. Chiu, M.H. Dao, Y.S. Ong, *How to Select Physics-Informed Neural Networks in the Absence of Ground Truth: A Pareto Front-Based Strategy*, in *1st Workshop on the Synergy of Scientific and Machine Learning Modeling* (2023)

[24] O. Fuks, H.A. Tchelepi, Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. J. Mach. Learn. Model. Comput. **1**, 19–37 (2020). https://doi.org/10.1615/jmachlearnmodelcomput.2020033905

[25] A.S. Krishnapriyan, A. Gholami, S. Zhe, R.M. Kirby, M.W. Mahoney. Characterizing possible failure modes in physics-informed neural networks (2021). Preprint at https://arxiv.org/abs/2109.01050

[26] S. Wang, S. Sankaran, P. Perdikaris, Respecting causality for training physics-informed neural networks. Comput. Methods Appl. Mech. Eng. **421**, 116813 (2024). https://doi.org/10.1016/j.cma.2024.116813

[27] M. Tancik, P.P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J.T. Barron, R. Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains (2020). Preprint at https://arxiv.org/abs/2006.10739

[28] C. Wu, M. Zhu, Q. Tan, Y. Kartha, L. Lu, A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. Comput. Methods Appl. Mech. Eng. **403 Part A**, 115671 (2022). https://doi.org/10.1016/j.cma.2022.115671

[29] A. Daw, J. Bu, S. Wang, P. Perdikaris, A. Karpatne. Mitigating Propagation Failures in Physics-informed Neural Networks using Retain-Resample-Release (R3) Sampling (2023). Preprint at https://arxiv.org/abs/2207.02338

[30] Z. Xiang, W. Peng, X. Liu, W. Yao, Self-adaptive loss balanced Physics-informed neural networks. Neurocomputing **496**, 11–34 (2022). https://doi.org/10.1016/j.neucom.2022.05.015

[31] S. Wang, X. Yu, P. Perdikaris, When and why PINNs fail to train: A neural tangent kernel perspective. J. Comput. Phys. **449**, 110768 (2022). https://doi.org/10.1016/j.jcp.2021.110768

[32] L.D. McClenny, U.M. Braga-Neto, Self-adaptive physics-informed neural networks. J. Comput. Phys. **474**, 111722 (2023). https://doi.org/10.1016/j.jcp.2022.111722

[33] J. Yu, L. Lu, X. Meng, G.E. Karniadakis. Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems (2021). Preprint at https://arxiv.org/abs/2111.02801

[34] F.M. Rohrhofer, S. Posch, G.B. C. Data vs. Physics: The Apparent Pareto Front of Physics-Informed Neural Networks (2021). Preprint at https://arxiv.org/abs/2105.00862

[35] F. Heldmann, S. Berkhahn, M. Ehrhardt, K. Klamroth, PINN training using biobjective optimization: The trade-off between data loss and residual loss. J. Comput. Phys. **488**, 112211 (2023). https://doi.org/10.1016/j.jcp.2023.112211

[36] R. Bischof, M.A. Kraus, Multi-Objective Loss Balancing for Physics-Informed Deep Learning. Comput. Methods Appl. Mech. Eng. **439**, 117914 (2025). https://doi.org/10.1016/j.cma.2025.117914

[37] T. Lazovskaya, D. Tarkhov, M. Chistyakova, E. Razumov, A. Sergeeva, T. Shemyakina, Evolutionary PINN Learning Algorithms Inspired by Approximation to Pareto Front for Solving Ill-Posed Problems. Computation **11**, 166 (2023). https://doi.org/10.3390/computation11080166

[38] B. Lu, C. Moya, G. Lin, NSGA-PINN: A Multi-Objective Optimization Method for Physics-Informed Neural Network Training. Algorithms **16**, 194 (2023). https://doi.org/10.3390/a16040194

[39] J.C. Wong, A. Gupta, C.C. Ooi, P.H. Chiu, J. Liu, Y.S. Ong. Evolutionary Optimization of Physics-Informed Neural Networks: Survey and Prospects (2025). Preprint at https://arxiv.org/abs/2501.06572

[40] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**, 182–197 (2002). https://doi.org/10.1109/4235.996017

[41] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, S.G. Johnson, Physics-Informed Neural Networks with Hard Constraints for Inverse Design. SIAM J. Sci. Comput. **43**, B1105–B1132 (2021). https://doi.org/10.1137/21M1397908

[42] A. Dener, M.A. Miller, R.M. Churchill, T.S. Munson, C.S. Chang. Training neural networks under physical constraints using a stochastic augmented Lagrangian approach (2020). Preprint at https://arxiv.org/abs/2009.07330

[43] J. Platt, A. Barr, *Constrained Differential Optimization*, in *Neural Information Processing Systems*, vol. 1 (American Institute of Physics, 1987), pp. 612–621

[44] D.P. Kingma, J. Ba. Adam: A Method for Stochastic Optimization (2017). Preprint at https://arxiv.org/abs/1412.6980

[45] X. Xie, P. Zhou, Z. Li, H. Lin, S. Yan, Adan: Adaptive Nesterov Momentum Algorithm for Faster Optimizing Deep Models. IEEE Trans. Pattern Anal. Mach. Intell. **46**, 9508–9520 (2024). https://doi.org/10.1109/TPAMI.2024.3423382

[46] J.A. Nelder, R. Mead, A simplex method for function minimization. Comput. J. **7**, 308–313 (1965). https://doi.org/10.1093/comjnl/8.1.27

[47] X.L. Sun, D. Li, K.I.M. McKinnon, On Saddle Points of Augmented Lagrangians for Constrained Nonconvex Optimization. SIAM J. Optim. **15**, 1128–1146 (2005). https://doi.org/10.1137/030602770

[48] R. Virtanen, P. Gommers, T.E. Oliphant, T. Haberland, M. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S.J. van der Walt, M. Brett, J. Wilson, K.J. Millman, N. Mayorov, A.R.J. Nelson, E. Jones, R. Kern, E. Larson, C.J. Carey, İ. Polat, Y. Feng, E.W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E.A. Quintero, C.R. Harris, A.M. Archibald, A.H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 Contributors, SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nat. Methods **17**, 261–272 (2020). https://doi.org/10.1038/s41592-019-0686-2

[49] V. Bibeau, D.M. Boffito, B. Blais, Physics-informed Neural Network to predict kinetics of biodiesel production in microwave reactors. Chem. Eng. Process. **196**, 109652 (2024). https://doi.org/10.1016/j.cep.2023.109652

[50] R.N. Bergman, Y.Z. Ider, C.R. Bowden, C. Cobelli, Quantitative estimation of insulin sensitivity. Am. J. Physiol. **236**, E667 (1979). https://doi.org/10.1152/ajpendo.1979.236.6.E667

[51] L. Babadzanjanz, J. Boyle, D. Sarkissian, J. Zhu, Parameter Identification For Oscillating Chemical Reactions Modelled By Systems Of Ordinary Differential Equations. J. Comput. Methods Sci. Eng. **3**, 223–232 (2003). https://doi.org/10.3233/JCM-2003-3203

[52] A.L. Hodgkin, A.F. Huxley, A quantitative description of membrane current and its application to conduction and excitation in nerve. J. Physiol. **117**, 500–544 (1952). https://doi.org/10.1113/jphysiol.1952.sp004764

[53] F. R., Impulses and Physiological States in Theoretical Models of Nerve Membrane. Biophys. J. **1**, 445–466 (1961). https://doi.org/10.1016/S0006-3495(61)86902-6

[54] J. Nagumo, S. Arimoto, S. Yoshizawa, An Active Pulse Transmission Line Simulating Nerve Axon. Proc. IEEE **50**, 2061–2070 (1962). https://doi.org/10.1109/JRPROC.1962.288235

[55] Y. Che, L.H. Geng, C. Han, S. Cui, J. Wang, Parameter estimation of the FitzHugh-Nagumo model using noisy measurements for membrane potential. Chaos **22**, 023139 (2012). https://doi.org/10.1063/1.4729458

[56] R.O. Doruk, L. Abosharb, Estimating the Parameters of Fitzhugh-Nagumo Neurons from Neural Spiking Data. Brain Sci. **9**, 364 (2019). https://doi.org/10.3390/brainsci9120364

[57] A. Samson, M. Tamborrino, I. Tubikanec, Inference for the stochastic FitzHugh-Nagumo model from real action potential data via approximate Bayesian computation. Comput. Stat. Data Anal. **204**, 108095 (2025). https://doi.org/10.1016/j.csda.2024.108095

[58] J. Rudi, J. Bessac, A. Lenzi, *Parameter Estimation with Dense and Convolutional Neural Networks Applied to the FitzHugh-Nagumo ODE*, in *2nd Annual Conference on Mathematical and Scientific Machine Learning*, *Proceedings of Machine Learning Research*, vol. 145 (2021), pp. 1–28

[59] A. Bizzi, L. Nissenbaum, J.M. Pereira. Neural Conjugate Flows: Physics-informed architectures with flow structure (2024). Preprint at https://arxiv.org/abs/2411.08326

[60] R.A. Fisher, The Wave of Advance of Advantageous Genes. Ann. Eugen. **7**, 355–369 (1937). https://doi.org/10.1111/j.1469-1809.1937.tb02153.x

[61] A.N. Kolmogorov, I.G. Petrovskii, N.S. Piskunov, Investigation of the Equation of Diffusion Combined with Increasing of the Substance and Its Application to a Biology Problem. Bull. Moscow State Univ. Ser. A: Math. Mech. **1**, 1–25 (1937)

[62] B.G. Sengers, C.P. Please, R.O.C. Oreffo, Experimental characterization and computational modelling of two-dimensional cell spreading for skeletal regeneration. J. R. Soc. Interface **4**, 1107–1117 (2007). https://doi.org/10.1098/rsif.2007.0233

[63] M.J. Simpson, K.K. Treloar, B.J. Binder, P. Haridas, K.J. Manton, D.I. Leavesley, D.L.S. McElwain, R.E. Baker, Quantifying the roles of cell motility and cell proliferation in a circular barrier assay. J. R. Soc. Interface **10**, 20130007 (2013). https://doi.org/10.1098/rsif.2013.0007

[64] S.E. Wang, P. Hinow, N. Bryce, A.M. Weaver, L. Estrada, C.L. Arteaga, G.F. Webb, A Mathematical Model Quantifies Proliferation and Motility Effects of TGF-β on Cancer Cells. Comput. Math. Methods Med. **10**, 71–83 (2009). https://doi.org/10.1080/17486700802171993

[65] F.M. Rohrhofer, S. Posch, C. Gößnitzer, B.C. Geiger, Approximating families of sharp solutions to Fisher's equation with physics-informed neural networks. Comput. Phys. Commun. **307**, 109422 (2025). https://doi.org/10.1016/j.cpc.2024.109422

[66] S. Sultan, Z. Zhang, A comparative investigation of a time-dependent mesh method and physics-informed neural networks to analyze the generalized Kolmogorov-Petrovsky-Piskunov equation. Int. J. Numer. Methods Fluids **96**, 651–669 (2024). https://doi.org/10.1002/fld.5259

[67] J. Burgers, in *Advances in Applied Mechanics*, vol. 1, ed. by R. Von Mises, T. Von Kármán (Elsevier, 1948), pp. 171–199. https://doi.org/10.1016/S0065-2156(08)70100-5

[68] H.A. Basha, Burgers' equation: A general nonlinear solution of infiltration and redistribution. Water Resour. Res. **38**, 29–1–29–9 (2002). https://doi.org/10.1029/2001WR000954

[69] P. Yuldashev, S. Ollivier, M. Averiyanov, O. Sapozhnikov, V. Khokhlova, P. Blanc-Benon, Nonlinear propagation of spark-generated *N*-waves in air: Modeling and measurements using acoustical and optical methods. J. Acoust. Soc. Am. **128**, 3321–3333 (2010). https://doi.org/10.1121/1.3505106

[70] S.W. Mason, Adoption of the Transport and Burgers' Equations in Modeling Neurological Shock-Waves in the Human Brain Due to Improvised Explosive Devices (IEDs). Front. Appl. Math. Stat. **4**, 29 (2018). https://doi.org/10.3389/fams.2018.00029

[71] I. Das, D. Das, P. Debnath, S. Nath, M. Chanda, *Physics Informed Neural Networks(PINNs) for Burgers' equation*, in *2024 4th International Conference on Artificial Intelligence and Signal Processing (AISP)* (2024), pp. 1–6. https://doi.org/10.1109/AISP61711.2024.10870712

[72] S. Bein, P. Connor, K. Pedro, P. Schleper, M. Wolf, Refining fast simulation using machine learning. EPJ Web Conf. **295**, 09032 (2024). https://doi.org/10.1051/epjconf/202429509032

[73] R. Rathnakumar, J. Huang, H. Yan, Y. Liu. Bayesian Entropy Neural Networks for Physics-Aware Prediction (2024). Preprint at https://arxiv.org/abs/2407.01015

[74] E. Heiden, M. Macklin, Y. Narang, D. Fox, A. Garg, F. Ramos. DiSECt: A Differentiable Simulation Engine for Autonomous Robotic Cutting (2021). Preprint at https://arxiv.org/abs/2105.12244

[75] S. Kumar, E. Malmi, A. Severyn, Y. Tsvetkov. Controlled Text Generation as Continuous Optimization with Multiple Constraints (2021). Preprint at https://arxiv.org/abs/2108.01850

[76] M. Xia, T. Gao, Z. Zeng, D. Chen. Sheared LLaMA: Accelerating Language Model Pre-training via Structured Pruning (2024). Preprint at https://arxiv.org/abs/2310.06694

[77] V. Pareto, *Manual of political economy (manuale di economia politica)* (Kelley, 1971 (1906)), pp. xii, 504 p. Translated by Ann S. Schwier and Alfred N. Page

[78] I. Das, J.E. Dennis, A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. Struct. Multidiscip. Optim. **14**, 63–69 (1997). https://doi.org/10.1007/BF01197559

[79] S. Boyd, L. Vandenberghe, *Convex Optimization* (Cambridge University Press, 2004)

[80] J. Nocedal, S. Wright, *Numerical Optimization*, 2nd edn. Operations Research and Financial Engineering (Springer, 2006)

[81] I.M. Sobol, On the distribution of points in a cube and the approximate evaluation of integrals. USSR Comput. Math. Math. Phys. **7**, 86–112 (1967). https://doi.org/10.1016/0041-5553(67)90144-9