Physics-informed neural networks with unknown measurement noise

Philipp Pilar
Department of Information Technology
Uppsala University, Sweden
philipp.pilar@it.uu.se

Niklas Wahlström
Department of Information Technology
Uppsala University, Sweden
niklas.wahlstrom@it.uu.se

Abstract

Physics-informed neural networks (PINNs) constitute a flexible approach to both finding solutions and identifying parameters of partial differential equations. Most works on the topic assume noiseless data, or data contaminated by weak Gaussian noise. We show that the standard PINN framework breaks down in case of non-Gaussian noise. We give a way of resolving this fundamental issue and we propose to jointly train an energy-based model (EBM) to learn the correct noise distribution. We illustrate the improved performance of our approach using multiple examples.

1 Introduction

While the idea of using neural networks to solve partial differential equations (PDEs) dates back to the work of Lagaris et al. [1], the field has received renewed attention [2, 3, 4, 5] due to the seminal work of Raissi et al. [6], in which they introduced physics-informed neural networks (PINNs). The main advantage of PINNs over traditional solvers lies in their flexibility, especially when considering the inverse problem [3]: as neural networks, they have the capacity for universal function approximation, they are mesh-free, and they can directly be applied to very different kinds of PDEs, without the need for a custom solver. In the inverse problem, PDE parameters are learned from data and the question as to the effect of noisy data on the quality of the estimates arises naturally.

In this work, we consider the inverse problem for the case of measurements contaminated with non-Gaussian noise of unknown form. The least-squares loss, which is commonly employed as data loss in PINNs, is known to perform poorly in this case [7, 8]. We give a way of mitigating this issue by suitably modelling the noise distribution. A high-level illustration of our method is given in Fig. 1: we employ an energy-based model (EBM) to learn the noise probability density function (PDF) jointly with the PINN. This PDF is then utilized to estimate the likelihood of the measurements under our model, which in turn serves as data loss.

Non-Gaussian measurement noise of unknown form can appear in a variety of applications [9]. Measurements of geomagnetic fields may exhibit asymmetric, long-tailed noise [7], data in astrophysics or seismology may be contaminated with impulsive noise [10], and not accounted for systematic errors in the measurement procedure may give rise to bias in the noise [11]. In addition to solving the PDE correctly, the PINN-EBM identifies the noise distribution. In that way, it could, for example, detect previously unrecognized systematic errors in the data.

Related work

Some research has been done on PINNs in case of noisy measurements, although typically only Gaussian noise is considered. In Yang et al. [12], the framework of Bayesian neural networks [13] is combined with the PINN framework, in order to obtain uncertainty estimates when training PINNs on noisy data. Bajaj et al. [14] introduce the GP-smoothed PINN, where a Gaussian process (GP)

Machine Learning and the Physical Sciences Workshop, NeurIPS 2023.

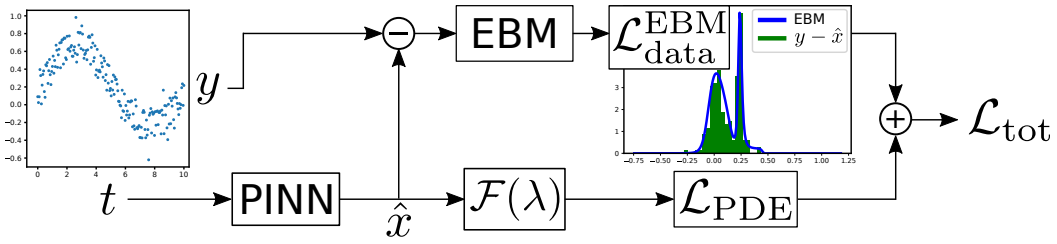

Figure 1: The PINN-EBM approach: the inputs t are passed through the PINN to obtain the PINN predictions $\hat{x}$. The differential operator $\mathcal{F}(\lambda)$ is then applied to $\hat{x}$ to obtain the PDE residuals, which are subsequently utilized to calculate the PDE loss $\mathcal{L}_{\text{PDE}}$. At the same time, the residuals between the noisy measurements y and $\hat{x}$ are formed, serving as noise estimates. The EBM is then trained on these estimates in order to learn the noise PDF, which can in turn be utilized to compute the likelihood of the measurements serving as data loss $\mathcal{L}_{\text{data}}^{\text{EBM}}$. Finally, the two loss terms are combined to form the total loss $\mathcal{L}_{\text{tot}}$. Both PINN and EBM, as well as the PDE parameters λ, can be trained by backpropagating $\mathcal{L}_{\text{tot}}$.

[15] is utilized to ameliorate noisy initial value data and make the PINN training more robust in this situation. In Chen et al. [16], the PINN-SR method is introduced which can be employed to determine governing equations from scarce and noisy data. In contrast to these works, we give a way of taking into account unknown, non-Gaussian noise in the PINN and provide a training procedure for this case.

While EBMs are commonly employed for classification [17] and image generation [18], they have also been successfully applied to regression problems; applications include object detection [19] and visual tracking [20]. In Gustafsson et al. [21], different methods for training EBMs for regression are discussed. In our work we also consider regression tasks and to the best of our knowledge, our paper is the first to combine EBMs and PINNs. The standard EBM approach to regression would not take the physical knowledge in form of the differential equation into account.

Background

PINNs The PINN-framework [6] can be used as an alternate approach to solving PDEs. A neural network is employed to parameterize the numerical solution $\hat{x}(t) = \hat{x}(t|\theta_{\text{PINN}})$ of the PDE, where θ_{PINN} denotes the network parameters. The PDE is defined via the differential operator $\mathcal{F}(\lambda)$, which may contain unknown parameters λ. We are given a dataset $\mathcal{D}_d = \{t_d^i, y_d^i\}_{i=1}^{N_d}$ of N_d (noisy) measurements of the PINN solution and choose another set of N_c so-called collocation points $\mathcal{D}_c = \{t_c^i\}_{i=1}^{N_c}$. When training the PINN, two losses enter into the loss function: the data loss, evaluating the fit of the PINN prediction with the data, and the PDE loss, a measure of the fulfillment of the PDE by the PINN solution:

$$\mathcal{L}_{\text{data}}(\hat{x}, \{t_d, y_d\}_{\text{mb}}) = \frac{1}{N_d'} \sum_{i=1}^{N_d'} (\hat{x}(t_d^i) - y_d^i)^2, \qquad \mathcal{L}_{\text{PDE}}(\mathcal{F}, \hat{x}, \{t_c\}_{\text{mb}}) = \frac{1}{N_c'} \sum_{i=1}^{N_c'} f(t_c^i)^2, \quad (1)$$

where the $f(t) = \mathcal{F}(\lambda)\hat{x}(t) \overset{!}{=} 0$ denote the PDE residuals, and N_d' and N_c' the number of data and collocation points in the current minibatch (mb), respectively. The PINN is then trained by minimizing the total loss $\mathcal{L}_{\text{tot}} = \mathcal{L}_{\text{data}} + \omega \mathcal{L}_{\text{PDE}}$ with respect to the parameters θ_{PINN} and λ. Unless stated otherwise, the weighting factor $\omega = 1$.

EBMs EBMs constitute a powerful method of learning probability densities from data [17]. The EBM can retain all of the flexibility of a neural network, via the following parametrization: first, we introduce $\hat{h}(t, y) = \hat{h}(t, y|\theta_{\text{EBM}})$ which denotes the (scalar) output of the neural network with weights θ_{EBM}. Then the probability $p(y|t, \hat{h}) = \frac{e^{\hat{h}(t,y)}}{Z(t,\hat{h})}$, where the partition function $Z(t, \hat{h}) = \int e^{\hat{h}(t,\tilde{y})} d\tilde{y}$. To train the network, we choose the approach of directly minimizing the negative log-likelihood (NLL) [21], which can be written as

$$\text{NLL}(\{t_d, y_d\}_{\text{mb}}, \hat{h}) = -\log \left(\prod_{i=1}^{N_d'} p(y_d^i | t_d^i, \hat{h}) \right) = \sum_{i=1}^{N_d'} \log Z(t_d^i, \hat{h}) - \hat{h}(t_d^i, y_d^i). \quad (2)$$

In our case, the evaluation of the partition function to high accuracy remains tractable by utilizing numerical integration, since the noise distribution is one-dimensional (note that this does not constitute a restriction on the dimensionality of the PDE).

2 Problem formulation

We consider the following setup, consisting of the differential equation and a measurement equation:

$$\mathcal{F}(\lambda)x(t) = 0, \qquad y(t) = x(t) + \epsilon, \tag{3}$$

where the parametric form of the differential operator $\mathcal{F}(\lambda)$ is given, as well as N_d measurements $y(t)$ of the corresponding solution $x(t)$ contaminated with homogeneous measurement noise ϵ. In this work, we aim to combine the PINN and the EBM framework in order to solve the inverse problem in case of measurements contaminated with non-Gaussian and non-zero mean noise.

Using the least-squares loss will produce wrong solutions in such cases, as biased noise will push the PINN solution systematically away from the correct solution x. A way of mitigating this is to add an offset term θ_0 to the PINN prediction in the data loss, i.e. $\hat{x} \to \hat{x} + \theta_0$ in (1), which is supposed to learn the bias. While consistent with the PINN loss, this solution still ignores the actual shape of the noise distribution.

In order to take the full shape of the noise distribution into account, we train an EBM and employ it to obtain the negative log-likelihood of the data, given both our models. For the training procedure, we then employ the following loss function:

$$\mathcal{L}_{\text{tot}} = \mathcal{L}^{\text{EBM}}_{\text{data}}(\{y_d - \hat{x}(t_d)\}_{\text{mb}}, \hat{h}|\theta_{\text{PINN}}, \theta_{\text{EBM}}) + \omega\mathcal{L}_{\text{PDE}}(\mathcal{F}, \hat{x}, \{t_c\}_{\text{mb}}|\theta_{\text{PINN}}, \lambda), \tag{4}$$

where $\mathcal{L}^{\text{EBM}}_{\text{data}}(\cdot, \hat{h}) = \frac{1}{N'_d}\text{NLL}(\cdot, \hat{h}) = \log Z(\hat{h}) - \frac{1}{N'_d}\sum_{i=1}^{N'_d} \hat{h}(\cdot)$, utilizing (2); note that $\hat{h}$ and hence the NLL are now independent of t, because we consider homogeneous noise. Since we do not know the actual magnitude of the noise at each data point, we use the residuals $y_d - \hat{x}(t_d)$ between the current PINN prediction and the measurements as our best guess. These estimates of the noise values then serve as training data for the EBM. In other words, we employ an unconditional EBM to model the PDF of the residuals. With the loss (4), both models are trained jointly until convergence. Details on the training procedure are given in Appendix A.1.

3 Experimental results

In our experiments, we compare results for the standard PINN, the PINN with learnable offset parameter θ_0, and the combination of PINN and EBM. We refer to these models as PINN, PINN-off and PINN-EBM. We consider homogeneous noise in a variety of shapes: Gaussian noise (G), a uniform distribution (u), and a mixture of three Gaussians (3G). 3G0 refers to the same distribution as 3G, shifted to have zero mean. Plots and exact definitions of these noise distributions can be found in Appendix A.3. We employ two differential equations of increasing complexity:

$$\text{The exponential equation:} \quad \dot{x}(t) - \lambda x(t) = 0 \tag{5}$$

$$\text{The Navier-Stokes equations:} \quad \begin{aligned} &u_{\text{t}} + \lambda_1(uu_{\text{x}} + vu_{\text{y}}) + p_{\text{x}} - \lambda_2(u_{\text{xx}} + u_{\text{yy}}) = 0 \\ &v_{\text{t}} + \lambda_1(uv_{\text{x}} + vv_{\text{y}}) + p_{\text{y}} - \lambda_2(v_{\text{xx}} + v_{\text{yy}}) = 0 \end{aligned} \tag{6}$$

The correct values of the PDE parameters are given by $\lambda = 0.3$, $\lambda_1 = 1$, $\lambda_2 = 0.01$. The results are collected in Table 1, where we consider the following performance metrics: the absolute error in the learned values of the PDE parameters ($|\Delta\lambda|$), the root-mean-square error (RMSE) on the validation data, the log-likelihood (logL) of the validation data according to the models, and the square values of the PDE residuals (f^2) on the training data.

In practice, the log-likelihood is the most relevant performance metric: contrary to the RMSE and $|\Delta\lambda|$, it can also be calculated when the true solution is not known. While f^2 can also be calculated without knowledge of the true solution, it only measures how well the learned PDE is fulfilled and not necessarily the correct one.

When considering the results for the exponential differential equation, it is apparent that PINN-EBM performs best in case of non-Gaussian noise. PINN-off manages to improve upon PINN in case of biased noise, but is hindered by the high variance of its predictions; we suspect that this may be due

Exponential (5)

noise		PINN-EBM	PINN-off	PINN
3G	$100\|\Delta\lambda\|$	**1.22**±1.12	3.96±3.62	10.22±1.47
	RMSE	**0.29**±0.29	1.07±0.87	4.01±0.28
	logL	**-3.48**±0.97	-7.85±9.00	-8.88±2.75
	$100f^2$	**0.51**±0.30	30.85±11.51	48.83±17.07
u	$100\|\Delta\lambda\|$	**1.27**±0.33	3.25±2.45	12.29±0.66
	RMSE	**0.19**±0.12	0.79±0.53	5.11±0.17
	logL	**-4.75**±1.14	-6.49±6.77	-19.06±3.82
	$100f^2$	**0.72**±0.29	16.25±4.75	37.94±12.56
G	$100\|\Delta\lambda\|$	2.45±1.78	2.02±1.34	**1.08**±0.91
	RMSE	0.60±0.43	0.46±0.27	**0.29**±0.07
	logL	-5.13±2.47	-5.05±3.55	**-4.15**±2.48
	$100f^2$	**0.42**±0.32	10.82±5.97	11.72±6.58
3G0	$100\|\Delta\lambda\|$	**1.17**±1.03	5.28±3.23	2.21±2.03
	RMSE	**0.23**±0.19	1.12±0.69	0.48±0.15
	logL	**-3.73**±1.29	-9.76±11.15	-5.44±3.99
	$100f^2$	**0.61**±0.41	32.20±15.05	34.02±15.53

Navier-Stokes (6)

noise		PINN-EBM	PINN-off	PINN
3G	$100\|\Delta\lambda_1\|$	**1.19**±0.67	2.92±0.47	23.1±0.11
	$100\|\Delta\lambda_2\|$	**0.04**±0.03	0.09±0.05	0.08±0.06
	RMSE	**0.01**±0.00	0.03±0.00	0.19±0.00
	logL	**0.03**±0.08	-0.15±0.07	-0.40±0.30
	$100f^2$	**0.04**±0.00	0.10±0.01	0.18±0.01

Table 1: Results for the differential equations (5)-(6) in case of different noise forms (compare Appendix A.3). The entries in the table are mean values plus-or-minus one standard deviation. 10 runs were performed in case of the exponential differential equation, 5 runs for the Navier-Stokes equations. Bold font highlights best performance.

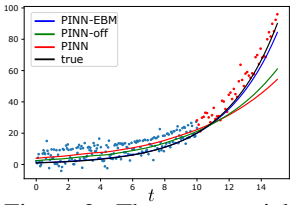

Figure 2: The exponential differential equation (5) with 3G noise. The results for one individual run are shown, where the blue dots represent training data and the red dots validation data.

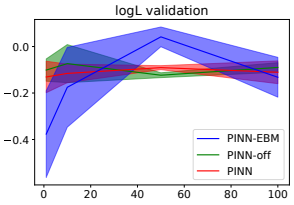

Figure 3: The Navier-Stokes equations with 3G noise. We consider the log-likelihood on the validation data as a function of the parameter ω, weighting the PDE loss. The averages of 5 runs are depicted, plus-or-minus one standard deviation.

to noise values far away from the mean having an outsized effect on the learning of the parameter θ_0 in this model. In case of Gaussian noise, PINN produces the best results; this makes sense, since here it implicitly uses the correct likelihood.

For the Navier-Stokes equations, PINN-EBM also outperforms the other models, with PINN-off performing significantly better than PINN. Here, $\omega = 50$ was chosen for PINN-EBM, according to the logL on the validation data (compare Fig. 3). For PINN and PINN-off, increasing ω did not lead to improved results and led to completely wrong predictions for $\omega > 10$ (not shown).

Additional experiments and results are discussed in the technical report [22].

4 Conclusions and future work

In this work, we demonstrated that the standard PINN fails in case of non-zero mean noise and we proposed the PINN-EBM to resolve this problem; utilizing an EBM to learn the noise distribution allows the PINN to produce good results also in case of non-zero mean and non-Gaussian noise. Employing a simple toy problem and the complex Navier-Stokes equations, we demonstrated the capabilities of our method and showed that it outperforms the standard PINN by a significant margin in case of non-Gaussian noise. In addition to determining the correct PDE solution, the PINN-EBM also allows for the identification of the true noise distribution, which may result in novel insights.

In the future, it would be interesting to investigate combining the PINN-EBM with other improvements to the PINN framework, such as adaptive weighting schemes [23], scheduling approaches [24, 25] or additional loss terms [26]. In principle, our method could also be applied to the problem considered in [14] (see also Section 1): where the standard GP presumably would fail here as well in case of non-Gaussian noise, a representation for the initial conditions could potentially also be learned via PINN-EBM.

Impact Statement

This research may enable PINNs to be trained on real-world data contaminated with non-Gaussian noise. We expect this to be of interest for scientific applications and do not foresee any negative societal impact.

Acknowledgements

The work is financially supported by the Swedish Research Council (VR) via the project *Physics-informed machine learning* (registration number: 2021-04321) and by the *Kjell och Märta Beijer Foundation*. We would also like to thank Fredrik K. Gustafsson for valuable feedback.

References

[1] Isaac E. Lagaris, Aristidis Likas, and Dimitrios I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9 (5):987–1000, 1998.

[2] Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, Jul 2022.

[3] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[4] Stefano Markidis. The old and the new: Can physics-informed deep-learning replace traditional linear solvers? *Frontiers in Big Data*, 4, 2021.

[5] Jan Blechschmidt and Oliver G. Ernst. Three ways to solve partial differential equations with neural networks — a review. *GAMM-Mitteilungen*, 44(2):e202100006, 2021.

[6] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[7] Catherine G. Constable. Parameter estimation in non-Gaussian noise. *Geophysical Journal International*, 94(1):131–142, 1988.

[8] Ayşen D Akkaya and Moti L Tiku. Robust estimation in multiple linear regression model with non-Gaussian noise. *Automatica*, 44(2):407–417, 2008.

[9] Don H Johnson and P Srinivasa Rao. On the existence of Gaussian noise (signal modelling). In *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, pages 1673–1676. IEEE Computer Society, 1991.

[10] Binwei Weng and Kenneth E Barner. Nonlinear system identification in impulsive environments. *IEEE Transactions on signal processing*, 53(7):2588–2594, 2005.

[11] Roger Barlow. Systematic errors: facts and fictions. *arXiv preprint hep-ex/0207026*, 2002.

[12] Liu Yang, Xuhui Meng, and George Em Karniadakis. B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. *Journal of Computational Physics*, 425:109913, 2021.

[13] Ethan Goan and Clinton Fookes. *Bayesian Neural Networks: An Introduction and Survey*, pages 45–87. Springer International Publishing, Cham, 2020.

[14] Chandrajit Bajaj, Luke McLennan, Timothy Andeen, and Avik Roy. Robust learning of physics informed neural networks. *arXiv:2110.13330*, 2021.

[15] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*. MIT press, 2006.

[16] Zhao Chen, Yang Liu, and Hao Sun. Physics-informed learning of governing equations from scarce data. *Nature Communications*, 12(1):6136, Oct 2021.

[17] Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang. A tutorial on energy-based learning. *Predicting structured data*, 2006.

[18] Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[19] Fredrik K. Gustafsson, Martin Danelljan, Goutam Bhat, and Thomas B. Schön. Energy-based models for deep probabilistic regression. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 325–343, Cham, 2020. Springer International Publishing.

[20] Martin Danelljan, Luc Van Gool, and Radu Timofte. Probabilistic regression for visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[21] Fredrik K. Gustafsson, Martin Danelljan, Radu Timofte, and Thomas B. Schön. How to train your energy-based model for regression. *CoRR*, abs/2005.01698, 2020.

[22] Philipp Pilar and Niklas Wahlström. Physics-informed neural networks with unknown measurement noise. *arXiv:2211.15498*, 2022.

[23] Levi McClenny and Ulisses Braga-Neto. Self-adaptive physics-informed neural networks using a soft attention mechanism. *arXiv:2009.04544*, 2020.

[24] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W. Mahoney. Characterizing possible failure modes in physics-informed neural networks. In *Advances in Neural Information Processing Systems*, 2021.

[25] Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality is all you need for training physics-informed neural networks. *arXiv:2203.07404*, 2022.

[26] Jeremy Yu, Lu Lu, Xuhui Meng, and George Em Karniadakis. Gradient-enhanced physics-informed neural networks for forward and inverse pde problems. *Computer Methods in Applied Mechanics and Engineering*, 393:114823, 2022.

Algorithm 1: Training the PINN-EBM

Input : PINN $\hat{x}(\cdot|\theta_{\mathrm{PINN}})$, EBM $\hat{h}(\cdot|\theta_{\mathrm{EBM}})$, data points $\{t_d, y_d\}$, collocation points $\{t_c\}$, differential operator $\mathcal{F}(\lambda)$, weighting factor ω

Output: optimized θ_{PINN}, θ_{EBM}, λ

$i = 0$

while *Training* **do**

  Draw a minibatch of data points $\{t_d, y_d\}_{\mathrm{mb}}$ and of collocation points $\{t_c\}_{\mathrm{mb}}$

  if $i < i_{\mathrm{ebm}}$ **then**

    Calculate data loss $\mathcal{L}_{\mathrm{data}}(\hat{x}, \{t_d, y_d\}_{\mathrm{mb}})$ according to (1)

  else

    if $i = i_{\mathrm{EBM}}$ **then** initialize EBM

    Calculate data loss $\mathcal{L}_{\mathrm{data}}^{\mathrm{EBM}}\left(\{y_d - \hat{x}(t_d)\}_{\mathrm{mb}}, \hat{h}\right)$ from (4)

  end

  if $i >= i_{\mathrm{EBM}}$ **then** $\omega' = \omega$ **else** $\omega' = 1$

  Calculate PDE loss $\mathcal{L}_{\mathrm{PDE}}(\mathcal{F}, \hat{x}, \{t_c\}_{\mathrm{mb}})$ according to (1)

  Compute total loss $\mathcal{L}_{\mathrm{tot}} = \mathcal{L}_{\mathrm{data}} + \omega' \mathcal{L}_{\mathrm{PDE}}$

  Calculate gradient $\nabla_\theta \mathcal{L}_{\mathrm{tot}}$ and update $\theta = \{\theta_{\mathrm{PINN}}, \theta_{\mathrm{EBM}}, \lambda\}$

  $i \mathrel{+}= 1$

end

A Additional details on the experiments

In this appendix, we give additional details on the training procedure and the experiments.

A.1 Training PINN and EBM jointly

In Algorithm 1, the training procedure for the PINN-EBM is summarized. Since both PINN and EBM need to be trained in parallel, the optimization procedure can be very challenging. In our experiments, it proved advantageous to start with the standard PINN loss in order to obtain a solution close to the data, and to only subsequently, after i_{EBM} iterations, switch to the EBM loss, in order to fine-tune the solution. Before the EBM loss is used for the first time, we initialize the EBM by training it for N_{EBM} iterations on the current noise estimates $y_d - \hat{x}(t_d)$, while keeping all parameters except θ_{EBM} fixed. Starting directly with the EBM loss also worked, but it often took significantly longer for the algorithm to converge.

Initializing the EBM only later during the training process is advantageous for two reasons: firstly, the least-squares loss is still a sensible first guess, so initializing the EBM on the residuals stemming from the pretrained network will let it start out with a reasonable form of the likelihood instead of a random one. Secondly, it will give us a better idea on the range in which the residuals between PINN prediction and data lie, allowing for a better normalization of the inputs to the EBM and hence more efficient training.

A.2 Implementation details [1]

The datasets employed for the exponential differential equation 5 contained 200 training points, 50 validation points and 2000 collocation points. For the Navier-Stokes equations 6, 4000 training points, 1000 validation points and 4000 collocation points were used. The collocation points were generated on a uniform grid. For both PINN and EBM, fully-connected neural networks with tanh activation function were used, with 4 layers of width 40 in case of the former, and 3 layers of width 5 for the latter. For the EBM, a dropout layer with factor 0.5 was inserted before the last layer. Both inputs and outputs of the networks were normalized to their expected ranges. In case of the Navier-Stokes experiment, 5 layers of width 30 were used for the PINN. The EBM was initialized after $i_{\mathrm{EBM}} = 4000$ iterations, except for the Navier-Stokes example, where $i_{\mathrm{EBM}} = 10000$ was chosen. The Adam optimizer with learning rate of 2e-3 was used for both PINN and EBM. The

[1] The code for the project is available at https://github.com/ppilar/PINN-EBM.

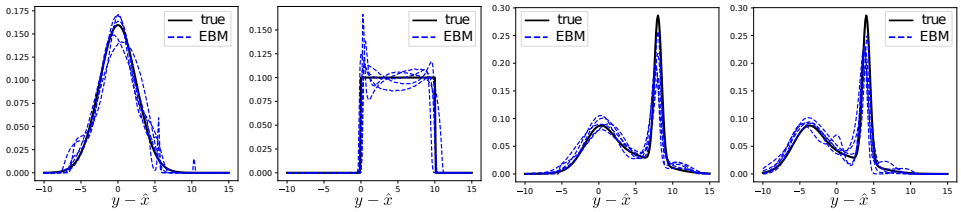

Figure 4: The noise distributions considered in the experiments are depicted. From left to right, we have a Gaussian distribution (G), a uniform distribution (u), a mixture of Gaussians (3G), and the same mixture of Gaussians, shifted to obtain zero mean (3G0). The dashed blue curves give examples of PDFs learned by the EBM during the training process, whereas the black curve gives the true PDF.

batch size for data points was 200 and the batch size for collocation points was 100. Training the PINN-EBM typically took 30-50% longer than the standard PINN.

A.3 Noise

In our experiments, we consider homogeneous noise in a variety of shapes. Amongst them are Gaussian noise $p(x) = \mathcal{N}(x|0, 2.5^2)$, a uniform distribution $p(x) = \mathcal{U}[0, 10]$ and a Gaussian mixture of the form $p(x) = \frac{1}{3}\left(\mathcal{N}(x|0, 2^2) + \mathcal{N}(x|4, 4^2) + \mathcal{N}(x|8, 0.5^2)\right)$. The values given here are scaled to the example of the exponential function (5). For other experiments, the shapes of the curves are retained, but the magnitude of the noise values is rescaled by a factor corresponding to the measurement values of the dataset at hand. In Fig. 4, the different noise distributions are depicted, together with a few examples of the PDFs learned by the EBM during experiments.