Towards Model Discovery Using Domain Decomposition and PINNs

Saha, Tirtho S.; Heinlein, Alexander; Reisch, Cordula

DOI
10.1016/j.ifacol.2025.03.008

Publication date
2025

Document Version
Final published version

Published in
IFAC-PapersOnline

Citation (APA)
Saha, T. S., Heinlein, A., & Reisch, C. (2025). Towards Model Discovery Using Domain Decomposition and PINNs. *IFAC-PapersOnline*, *59*(1), 37-42. https://doi.org/10.1016/j.ifacol.2025.03.008

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 59-1 (2025) 37–42

Towards Model Discovery Using Domain Decomposition and PINNs

Tirtho S. Saha [*] **Alexander Heinlein** [**] **Cordula Reisch** [***]

[*] *Institute of Mechanics and Computational Mechanics, Leibniz University Hannover, Germany (e-mail: tirtho.saha@ibnm.uni-hannover.de)*
[**] *Delft Institute of Applied Mathematics, Delft University of Technology, The Netherlands (e-mail: a.heinlein@tudelft.nl).*
[***] *Institute for Partial Differential Equations, TU Braunschweig, Germany (e-mail: c.reisch@tu-bs.de)*

Abstract: We enhance machine learning algorithms for learning model parameters in complex systems represented by differential equations with domain decomposition methods. The study evaluates the performance of two approaches, namely (vanilla) Physics-Informed Neural Networks (PINNs) and Finite Basis Physics-Informed Neural Networks (FBPINNs), in learning the dynamics of test models with a quasi-stationary longtime behavior. We test the approaches for data sets in different dynamical regions and with varying noise level. As results, the FBPINN approach better captures the overall dynamical behavior compared to the vanilla PINN approach, even in cases with data only from a time domain with quasi-stationary dynamics.

Keywords: Nonlinear system identification, Neural networks, Modeling and parameter identification, Quasi-stationary dynamics, Domain decomposition.

1. INTRODUCTION

Mathematical modeling of biological processes is inherently complex due to the intricate and often only partially understood mechanisms involved. Additionally, biological processes exhibit different behaviors on different temporal and spatial scales. Some processes may take a long time, and often data are available only from certain stages, while data for other stages are unavailable. The inverse problem of determining important mechanisms and their parameters based on data is therefore complicated. This study addresses these challenges by leveraging data-driven machine learning approaches to identify abstract mechanisms and determine parameter values that represent the known biological mechanisms from observed data.

We focus on two toy models: the saturated growth model, which captures population growth dynamics, and the competition model, which examines interactions between two species, including scenarios of coexistence and survival (Murray (2007)). These models were tested on synthetic data from different time intervals, such as a dynamical and stationary phase, and the total time domain, with varying noise levels. We employ two different approaches: physics-informed neural networks (PINNs, Raissi et al. (2019)) using the SciANN library (Haghighat and Juanes (2021)), and domain decomposition-based PINNs using finite basis PINNs (FBPINNs, Moseley et al. (2023)); for the application of FBPINNs to (systems) of ordinary differential equations (ODEs) and partial differential equations (PDEs), we also refer to Heinlein et al. (2024), resp. Penwarden et al. (2023). The aim is to compare the ability of the methods in learning the parameters of the dynamical system in cases where the data is limited to certain time intervals. Problems like this occur when dealing with (biological) problems where only stationary data is available that can be interpreted as the result of a dynamical process in advance of the measurement.

In this paper, we decompose the domain into dynamic and quasi-stationary domain, to better capture the dynamical behavior of the system. Up to our knowledge, the application of the domain decomposition approach is new in the field of parameter estimation, in particular for differential equations with data from quasi-stationary dynamics.

2. COMPUTATIONAL METHODS

We start with introducing vanilla PINNs and the more sophisticated FBPINNs with domain decomposition.

2.1 Physics Informed Neural Networks(PINNs)

In contrast to purely data-driven approaches, PINNs are trained by using a combination of labeled training data and available prior knowledge about the problem (Raissi et al. (2019), Lagaris et al. (1998), Dissanayake and Phan-Thien (1994)). In the forward problem setup, the physical law(s) are known and encoded in a PDE, but the solution of the PDE is unknown. Let us consider a differential equation of the general form:

$$\begin{aligned} \mathcal{D}[u(x)] &= f(x), \quad x \in \Omega \subset \mathbb{R}^d, \\ \mathcal{B}_k[u(x)] &= g_k(x), \quad x \in \Gamma_k \subset \partial\Omega, \end{aligned} \tag{1}$$

where $\mathcal{D}[u(x)]$ is some differential operator with $u(x)$ as the solution, and $\mathcal{B}_k[\cdot]$ is a boundary operator including as well the initial conditions, which ensure uniqueness of the solution. The input x could be spatial and/or temporal, where d is the dimension of the domain. Equation (1) can represent many differential equation problems, including linear and nonlinear problems, ODEs and PDEs, time-dependent and stationary problems, and problems with an initial value, Dirichlet and Neumann boundary problems.

To solve the differential equation (1), PINNs use a neural network (NN) to directly approximate the solution, i.e.,

$$u^{\text{PINN}}(x;\theta) \approx u(x), \tag{2}$$

where x is the input to the network and θ are the trainable parameters of the NN model. The proposed general loss function from Raissi et al. (2019) to train the PINNs model combines two influences,

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{PDE}}(\theta) + \mathcal{L}_{\text{BC}}(\theta). \tag{3}$$

The PDE based loss function is

$$\mathcal{L}_{\text{PDE}}(\theta) = \frac{\lambda_{\text{phy}}}{N_I} \sum_{i=1}^{N_I} (\mathcal{D}[u^{\text{PINN}}(x_i;\theta)] - f(x_i))^2 \tag{4}$$

with $\{x_i\}_{i=1}^{N_I}$ being a set of collocation points sampled within Ω and λ_{phy} being a weight. The boundary condition loss function is

$$\mathcal{L}_{\text{BC}}(\theta) = \sum_{k=1}^{N_k} \frac{\lambda_B^k}{N_B^k} \sum_{i=1}^{N_B^k} (\mathcal{B}_k[u^{\text{PINN}}(x_i^k;\theta)] - g_k(x_i^k))^2, \tag{5}$$

where $\{x_j^k\}_{j=1}^{N_B^k}$ is a set of points sampled along each boundary condition (BC) and λ_B^k is a weight. The weights λ_{phy} and λ_B^k are chosen so that the individual terms in the loss function (3) contribute in a balanced manner. Finding an appropriate choice of λ_{phy} and λ_B^k leading to the best result is usually challenging and problem-dependent.

An alternative to using separate boundary condition loss terms $\mathcal{L}_{\text{BC}}(\theta)$ is to hard constrain the solution to satisfy the boundary condition exactly, which we do in this work. This approach involves directly incorporating the boundary conditions into the neural network architecture to inherently satisfy the boundary condition, thereby removing the need for the boundary condition residual term in the loss function. The hard constraining modifies the NN ansatz in (2) as follows:

$$\tilde{u}^{\text{PINN}}(x;\theta) = \mathcal{C}[u^{\text{PINN}}(x;\theta)] \approx u(x), \tag{6}$$

where $\mathcal{C}$ is the constraining operator applied to the output of the NN model. Consequently, the loss function of the NN becomes:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{PDE}}(\theta) = \frac{\lambda_{\text{phy}}}{N_I} \sum_{i=1}^{N_I} (\mathcal{D}[\tilde{u}^{\text{PINN}}(x_i;\theta)] - f(x_i))^2. \tag{7}$$

In many real-world scenarios, the objective is not only to solve a forward problem but also to address an inverse problem. An inverse problem involves estimating unknown parameters or initial conditions based on observed data, with the governing equations explicitly defined or partially known. In this case, the differential and/or boundary operators $\mathcal{D}$, respectively $\mathcal{B}_k$, may depend on a set of additional parameters $P = (p_1, \ldots, p_{N_P})$. Hence, solving the inverse problem does not only involve finding the network parameters θ but also the parameters P. Given available synthetic or real data and other prior knowledge, the inverse problem reads

$$\min_{\theta,P} \mathcal{L}(\theta,P) = \mathcal{L}_{\text{PDE}}(\theta,P) + \mathcal{L}_{\text{data}}(\theta,P) + \mathcal{L}_{\text{par}}(\theta,P), \tag{8}$$

where the loss function in (7) of the PINNs approach is complemented by the data loss

$$\mathcal{L}_{\text{data}}(\theta,P) = \frac{\lambda_{\text{data}}}{N_D} \sum_{i=1}^{N_D} (\tilde{u}^{\text{PINN}}(x_i;\theta) - u_{\text{data}}(x_i))^2 \tag{9}$$

and an optional parameter-induced loss function

$$\mathcal{L}_{\text{par}}(\theta,P) = \lambda_{\text{param}} \sum_{i=1}^{N_P} (\max\{0, p_{i,\min} - p_i, p_i - p_{i,\max}\})^2. \tag{10}$$

Here, λ_{data} represents the weight for the data residual in the loss function, and N_D is the number of data points used for the training. In general, the data points $u_{\text{data}}(x_i)$ could consist of measurements, observations, or synthetic data derived from simulations. Additionally, λ_{param} is the weight for the parameter constraints term in the loss function, and N_P is the number of parameters. $(p_{i,\min}, p_{i,\max})$ are the `min` and `max` the values of the i-th parameter. The range for the values of $p_{i,\min}$ and $p_{i,\max}$ depends on known values, and in the specific cases of this work, all parameters are considered non-negative, so the lower threshold is known.

Often, if more terms are included in the loss function, the complexity of the training increases due to the higher number of interactions between terms and the additional constraints imposed on the optimization process.

2.2 Domain Decomposition-Based PINNs(FBPINNs)

Vanilla PINN approaches show a spectral bias, meaning that they can learn the low frequency components more easily than the high-frequency components of the solution. To address this issue, Moseley et al. (2023) observed that domain decomposition-based PINN architectures can learn multiscale components of the solution.

The domain decomposition-based PINN approach defines an approximate solution similar to those given in (2) and (6), as it works well with both soft and hard boundary constraints setups. However, it differs in terms of the network architecture, as it employs as many neural networks as the number of subdomains chosen. The global network produces the output

$$u^{\text{FBPINN}}(x;\theta) = \sum_{j=1}^{J} \omega_j(x) \cdot \text{unnorm} \circ u_j^{\text{sub}}(x;\theta_j) \circ \text{norm}_j(x), \tag{11}$$

where the term $u^{\text{FBPINN}}(x;\theta)$ represents the collective sum of the output of all subdomains. The normalization term norm_j adjusts the input variable x to the range of $[-1,1]$ in each dimension over the subdomain before it is input to the individual neural network $u_j^{\text{sub}}(x;\theta_j)$ at j-th subdomain Ω_j. Then comes the output unnormalisation unnorm term, which ensures that the output stays within the range $[-1,1]$ in each neural network. Finally, the outputs are multiplied by the window function $\omega_j(x)$, which is smooth, differentiable, and zero outside the subdomain, confines the network's solution locally. Moreover, the choice of the subdomains enhances the learning of specific frequencies fitting to the subdomain size.

For a hyperrectangular subdomain the window functions are defined from a partition of unity as

$$\sum_{j=1}^{J} \omega_j \equiv 1 \quad \text{on } \Omega, \qquad \text{supp}(\omega_j) \subset \overline{\Omega}_j, \tag{12}$$

with

$$\omega_j(x) = (H(x - x_{j,\min}) \cdot H(x_{j,\max} - x)) \cdot \frac{1}{4}\left(1 + \cos\left(\pi \frac{x - \mu_j}{\sigma_j}\right)\right)^2, \tag{13}$$

where ω_j is the window function in the j-th subdomain after domain decomposition, and H is the Heaviside step function that ensures the solution is zero outside of each subdomain. The interval $(x_{j,\min}, x_{j,\max})$ represents the left and right overlapping region for the subdomain j. μ_j and σ_j represent the center and half-width of each subdomain, respectively. The cosine function ensures the solution is smooth within the interval and has a continuous first derivative. See Fig. 1 for a one-dimensional example with two subdomains and corresponding window functions.

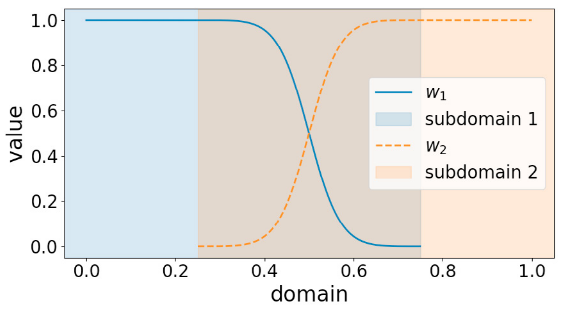

Fig. 1. One-dimensional example of window functions (13) for two overlapping subdomains. The light blue and light orange regions represent the respective subdomain intervals, while the combined light brown region highlights the overlap between the two subdomains.

The loss function for the FBPINN method is calculated for the given ansatz defined in (6), with the network output provided in (11), similar to the PINN loss function in (8) giving

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{PDE}}(\theta) + \mathcal{L}_{\text{data}}(\theta) + \mathcal{L}_{\text{par}}(\theta), \tag{14}$$

with the loss functions in (7), (9) and (10), where the PINN solution $\tilde{u}^{\text{PINN}}$ is replaced by the FBPINN solution (11).

These two computational approaches will be used in the following for determining in an inverse problem setting the parameters of ordinary differential equations with available data in certain time domains. The domain decomposition approach therefore applies to the time domain, giving more weight to time domains with more data or, in perspective, time domains with higher quality data.

3. MATHEMATICAL MODELS

We introduce two differential equation models with a non-trivial solution behavior. Those models will be investigated with the methods of Sec. 2.

3.1 Saturated growth model

In the saturated growth model (e.g. Murray (2007)), we consider a population of one species u with the carrying capacity C. The saturated growth model is

$$\frac{du}{dt} = u(C - u) \quad \text{with} \quad u(0) = u_0, \tag{15}$$

where u represents the population size, and $C > 0$ is the carrying capacity. This model captures the dynamics of a population undergoing saturated growth, such as the growth of a virus population in liver tissue. The solution tends towards C for initial values $u_0 > 0$, representing the saturation of growth as the population reaches its maximum capacity. The growth rate is moderated by the term $(C - u)$, which implies that the population growth rate decreases as it approaches the carrying capacity C. Fig. 2(a) shows the solution of the saturated growth model for an initial value $u(0) = u_0 > 0$ and $C = 1$.

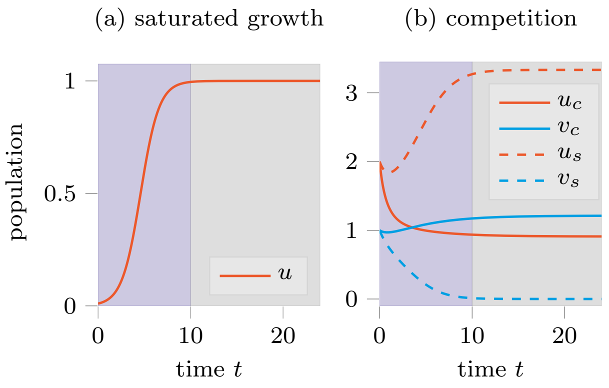

Fig. 2. Solutions of (a) the saturated growth model (15) and (b) the competition model (16) for parameters with coexistence (u_c, v_c) or single-survival (u_s, v_s). The dynamic time frame is shaded in blue, the quasi-stationary in gray.

3.2 Competition Model

The second model is a Lotka-Volterra competition model (e.g., Murray (2007)), which served as a test model in Reisch and Burmester (2023) concerning the possibility of model discovery. The dynamics of two species with inter- and inner-species competition is given by

$$\begin{aligned} \frac{du}{dt} &= u(1 - a_1 u - a_2 v) = f_1(u, v), \\ \frac{dv}{dt} &= rv(1 - b_1 u - b_2 v) = f_2(u, v), \\ \text{with} \quad & u(0) = u_0 > 0; \quad v(0) = v_0 > 0, \end{aligned} \tag{16}$$

where a_1, a_2, b_1, b_2 and r are all positive coefficients, and the species u and v compete for shared resources. There are two non-trivial long-time behaviors depending on the parameter values. Either, the system tends toward a coexistence steady state $(u_c^*, v_c^*) \neq (0, 0)$ or one species vanishes, i.e., either $(u_s^* \neq 0, v_s^* = 0)$ or $(u_s^* = 0, v_s^* \neq 0)$.

Fig. 2(b) illustrates the competition models featuring coexistence and single-survival scenarios. As depicted, in coexistence scenarios, two species survive together within shared environments. Conversely, in survival scenarios, one

Table 1. Parameter values of the competition model (16) in the two settings.

parameter	r	a_1	a_2	b_1	b_2
coexistence	0.5	0.7	0.3	0.3	0.6
single-survival	0.5	0.3	0.6	0.7	0.3

species gradually dominates the other over time. Tab. 1 gives the parameter values used in the simulations.

3.3 Model discovery and parameter estimation

The three models, saturated growth for one species and competition between two species with two parameter settings, serve as test models for a model discovery or, by restricting the mechanisms beforehand, for a parameter estimation in varying data scenarios. The introduced PINN approaches solve the inverse problem of finding parameter values by including the parameter to learn in the underlying differential equation that contributes in $\mathcal{L}_{\mathrm{PDE}}$. In this study, we provide the terms in the ODE that are included in the model used for data generation. In future works, we plan to implement a sparse choice of some mechanism terms, like in Brunton et al. (2016).

The parameters we want to learn are C for the saturated growth model and for the competition model the parameters in Tab. 1. We want to compare the ability of determining the model parameters by using data either from the dynamical time interval $[0, 10]$, the quasi-stationary time interval $[10, 24]$, or in the whole time interval $[0, 24]$; the dynamical and quasi-stationary time intervals are shown in Fig. 2. We will investigate both approaches, vanilla PINN and FBPINN, in all data settings.

Our expectations for the computational results are based on analytical properties of the stationary points: In the saturated growth model (15), the nontrivial stationary point $u^\star = C$ gives directly the parameter that we want to estimate. Our first hypothesis therefore is that both computational approaches should be able to reproduce a good estimation of the parameter, independent of the time frame employed for data collection. Besides, the solution of the neural network should be more precise when giving data from the dynamical time domain rather than from the quasi-stationary because the dynamics lead to the stationary states, but there are multiple dynamics tending towards the same stationary state.

The identification of the parameters in the competition model is much more challenging, firstly because there are more parameters to identify, and secondly because the stationary states do not depend on all parameters. More precisely, the coexistence stationary state is given by

$$(u_c^\star, v_c^\star) = \left(\frac{a_2 - b_2}{a_2 b_1 - a_1 b_2}, \frac{a_1 - b_1}{a_2 b_1 - a_1 b_2}\right),$$

so it is independent of r and in numerical simulations we have two stationary state values for determining four dependent parameters. The stationary state in the single-survival parameter setting is $(u_s^\star, v_s^\star) = (1/a_1, 0)$, and hence, independent of a_2, b_1, b_2, r. By knowing only the longtime behavior of the solution, it is therefore hard to determine the parameter values except for a_1.

Our hypotheses for the competition model take this into account: We expect that learning a_1 in the single-survival setting is feasible, even when only data in the quasi-stationary domain is available. On the other hand, determining any parameter in the coexistence case is hard for taking only data in the quasi-stationary domain.

Table 2. Learned values C for the saturated growth model (15), the true value is $C = 1$.

	$[0, 24]$	$[0, 10]$	$[10, 24]$
PINN	1.0042	0.9916	1.0097
FBPINN	0.9917	0.9915	0.9995

So far, these hypotheses on learning the parameters in the different models and settings are valid for both methods of Sec. 2. We investigate now how the domain decomposition with overlapping domains affect (i) the parameter estimation and (ii) the quality of the learned NN solution, both compared to vanilla PINNs.

4. RESULTS

We start with testing our hypotheses on the general parameter estimation problem and then compare more detailed the outcomes of vanilla PINNs and FBPINNs. In both methods, we employ hard enforcement boundary constraints. For the FBPINN approach, we choose the domain decomposition based on the dynamic and quasi-stationary time domains in Fig. 2.

4.1 Model accuracy and parameter learning

Our first hypothesis is that the parameter C in the saturated growth model is easy to learn for any of the algorithms and independent of the data region used. This hypothesis is confirmed by test cases that we run with the hyperparameters in Tab. 3. The learned parameters C for the three temporal domains and the two approaches, vanilla PINN and FBPINN, are given in Tab. 2.

Next, we check our hypotheses on the competition model. Firstly, we anticipated that, in the coexistence case, determining the parameters from the quasi-stationary time domain is difficult due to an underdetermined algebraic system for the parameters depending on the steady states. Fig. 3 shows these problems with false estimates in the quasi-stationary setting but acceptable estimates in the dynamical or full time domain. Both approaches perform qualitatively the same in this case.

The second hypothesis for the competition model states that in the single-survival parameter setting the estimation of a_1 is possible even in the quasi-stationary time domain, while it is not possible to determine the other parameters in this time domain exactly. Fig. 3 supports this hypothesis, and again, both methods perform qualitatively similarly. Consequently, the parameter estimation is a task that does not improve by FBPINNs, which was expected from the nature of the problem.

However, the effect of learned parameters from different approaches can be observed in Fig. 4 by the energy plots using the Lyapunov function

$$\begin{aligned}\phi = &- a_1 b_2 r(b_1 u + a_2 v) + a_1 a_2 b_1 b_2 r u v \\ &+ \frac{r}{2} a_1 b_2 (a_1 b_1 u^2 + a_2 b_2 v^2).\end{aligned} \tag{17}$$

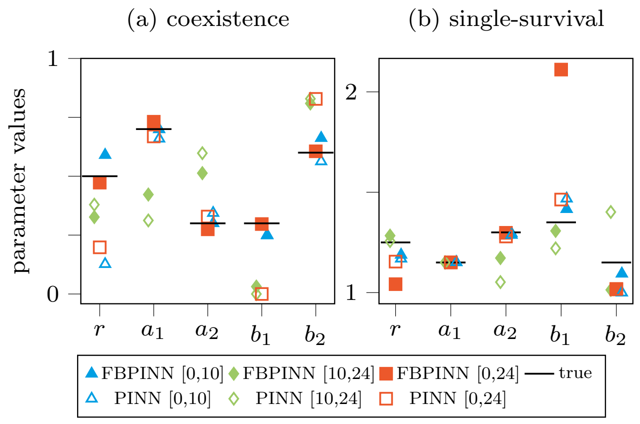

Fig. 3. Learned parameters in the competition model. The values $[a, b]$ give the time domain of data used.

Even though the differences of the learned parameters in the time interval $[0, 24]$ are relatively small, the energy plots given by the Lyapunov function based on the learned parameters differ rather crucial. While the learned parameters with FBPINN give an energy functional that strongly resembles the energy functional of the ground truth parameters, the vanilla PINN energy functional is qualitatively different. This has an effect on variations of the initial conditions: While the solutions will still tend towards a point close to the true stationary state in the FBPINN case, the dynamics with the learned parameters from vanilla PINN may be totally different.

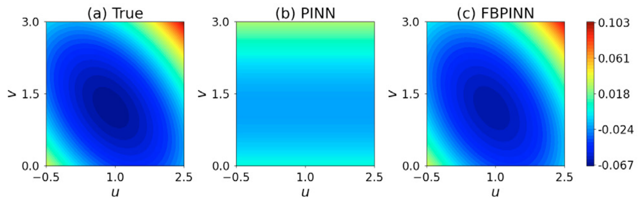

Fig. 4. Energy plots for the competition model (16) in the coexistence setting with data in $[0, 24]$.

Next, we want to compare the second outcome of the PINN approaches, a learned solution of the NN. Fig. 5 shows the mean squared error of the NN output and the numerical solution of (15), resp. (16), as ground truth. The results of the comparison in Fig. 5 show a higher accuracy of the FBPINN compared to vanilla PINN. This difference is very prominent as well for the quasi-stationary time domains, where the parameter estimation failed in both parameter settings of the competition model.

Based on this impression, we dive deeper into the differences of the solutions for one case. The difference of the MSE is largest for the competition models. Therefore, we compare the time-resolved solutions of the competition model in the coexistence case, see Fig. 6. The differences in the models depend, of course, on the data time domains. The solution of the vanilla PINN in the dynamical time domain $[0, 10]$ has a larger error for larger time, while the error for the solution with quasi-stationary data from $[10, 24]$ has a larger error for small time. The FBPINN solution in the whole time domain shows a surprising oscillatory behavior for larger time. A reason for this behavior might be overfitting of the included noise. Therefore, the oscillations only occur when data in the quasi-stationary time domain is available. A more sophisticated hyperparameter tuning may reduce this effect.

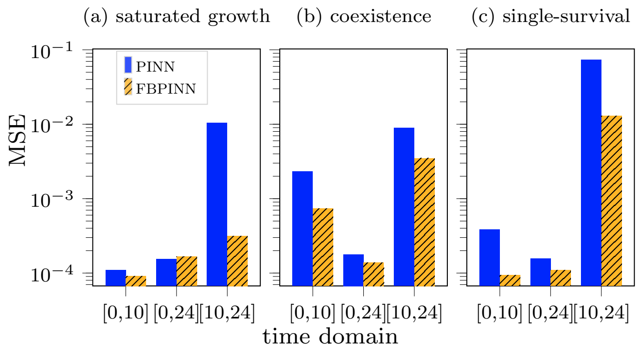

Fig. 5. MSE of the vanilla PINN solution and the FBPINN solution based on data from three time intervals.

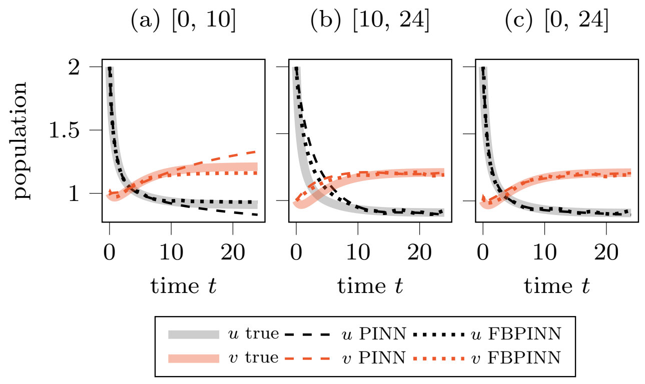

Fig. 6. Comparison of time-dependent PINN and FBPINN solutions for the competition model with coexistence

To our surprise, the overlap size of the subdomains had only a small influence on the solution quality with one exception: In the single survival test case, we found a large MSE for a window overlap into the data region of $wo = 1.001$ and smaller. For the other test cases, a window overlap variation between 1.001 and 2.3 does not affect the solution quality. In all regarded cases, there were still collocation points of the PDE loss in the overlap.

4.2 Loss landscapes with and without domain decomposition

Fig. 7 shows the loss landscapes (Li et al. (2018)) of FBPINNs and PINNs with individual colorbars. The loss landscape of FBPINNs shows less sensitivity to small variations in the trained weights compared to those of vanilla PINNs. The PINN loss landscapes are more convex than the FBPINN loss landscapes for data in $[0, 10]$ and $[0, 24]$. Following Li et al. (2018), this may indicate that the network initialization is more crucial for FBPINNs, where some chaotic regions exist next to well-formed convex regions. Further interpretation of the loss landscapes is challenging because the landscape shows only two random directions of the large parameter space of the NNs.

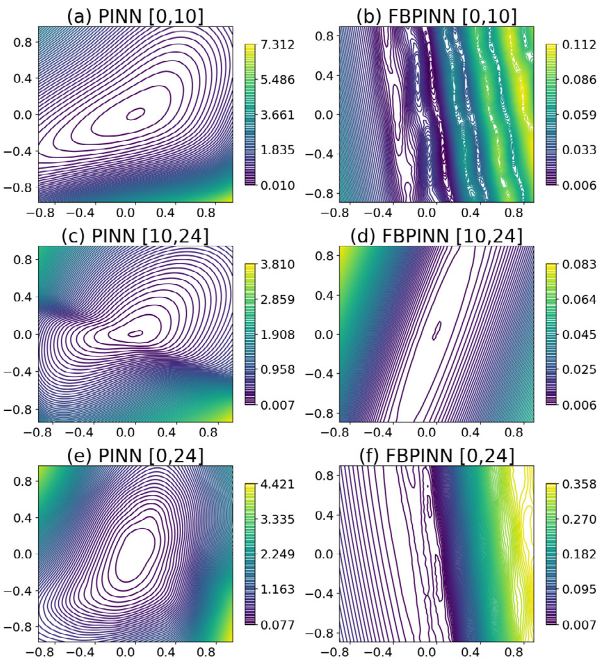

Fig. 7. PINN and FBPINN loss landscapes for the competition model with coexistence in the three data settings.

4.3 Noise Effect

The time-dependent dynamics of the learned solutions in Fig. 6 shows some oscillatory behavior due to noise. Fig. 8 compares the MSE for different noise levels in the data for the competition model with coexistence.

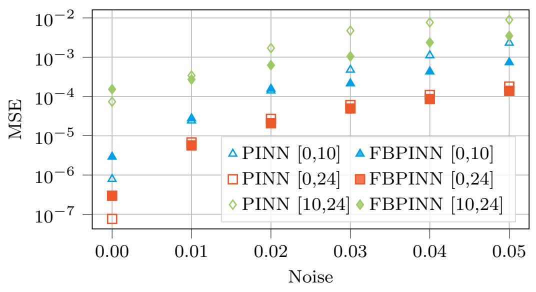

Fig. 8. MSE of solutions for the competition model in the coexistence case with varying noise.

For any non-zero noise level, the FBPINN solution has a smaller MSE than the vanilla PINN solution. In some cases, the difference is in the order of one magnitude. This supports the better performance of FBPINNs.A reason for this observation may be the ability of FBPINN to learn different parts of the solution in a more stable way due to the different subdomains.

5. TRAINING PARAMETERS

The used hyperparameters are given in Tab. 3. The results are robust against changes of nC, wo, wi and the number of layers. The FBPINNs approach uses all the training points in a single training step. Consequently, we set the batch size to be equal to the sum of the number of data points (nD) and the number of collocation points (nC), for the PINNs approach as well. The code is available at github.com/tirtho109/VanillaPINNsVsFBPINNs.

Table 3. Hyperparameters

Parameter	PINN	FBPINN
Hidden Layers (layers)	[5,5,5]	[5,5,5]
Epochs	50000	50000
Activation Function	tanh	tanh
Physics loss weight (λ_{phy})	1.0	1.0
Data loss weights (λ_{data})	1.0	1.0
Optimizer	Adam	Adam
Learning rate	0.001	0.001
Number of collocation points (nC)	200	200
Collocation Sampling	Grid	Grid
Number of data points (nD)	100	100
Noise (μ, σ)	0, 0.05	0, 0.05
Batch Size	300	300
Number of subdomains ($nsub$)	N/A	2
Window overlap data-region (wo)	N/A	1.9
Window overlap no-data-region (wi)	N/A	1.0005
Parameter loss weight (λ_{param})	N/A	1×10^6

REFERENCES

Brunton, S.L., Proctor, J.L., and Kutz, J.N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *PNAS*, 113(15), 3932–3937.

Dissanayake, M. and Phan-Thien, N. (1994). Neural-network-based approximations for solving partial differential equations. *Commun. Numer. Meth. En.*, 10(3), 195–201.

Haghighat, E. and Juanes, R. (2021). SciANN: A keras/tensorflow wrapper for scientific computations and physics-informed deep learning using artificial neural networks. *Comput. Methods Appl. Mech. Eng.*, 373, 113552.

Heinlein, A., Howard, A.A., Beecroft, D., and Stinis, P. (2024). Multifidelity domain decomposition-based physics-informed neural networks and operators for time-dependent problems. ArXiv:2401.07888 [cs, math].

Lagaris, I., Likas, A., and Fotiadis, D. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Netw.*, 9(5), 987–1000.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. (2018). Visualizing the loss landscape of neural nets. *Adv. Neural Inf. Process. Syst.*, 31.

Moseley, B., Markham, A., and Nissen-Meyer, T. (2023). Finite basis physics-informed neural networks: a scalable domain decomposition approach for solving differential equations. *Adv. Comput. Math.*, 49(4), 62.

Murray, J.D. (2007). *Mathematical biology: I. An introduction*, volume 17. Springer.

Penwarden, M., Jagtap, A., Zhe, S., Karniadakis, G., and Kirby, R. (2023). A unified scalable framework for causal sweeping strategies for physics-informed neural networks (PINNs) and their temporal decompositions. *J. Comput. Phys.*, 493, 112464.

Raissi, M., Perdikaris, P., and Karniadakis, G.E. (2019). 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.

Reisch, C. and Burmester, H. (2023). Model selection focusing on longtime behavior of differential equations. ArXiv:2312.05128 [math].