Exploring Robustness of Physics-Informed Neural Networks

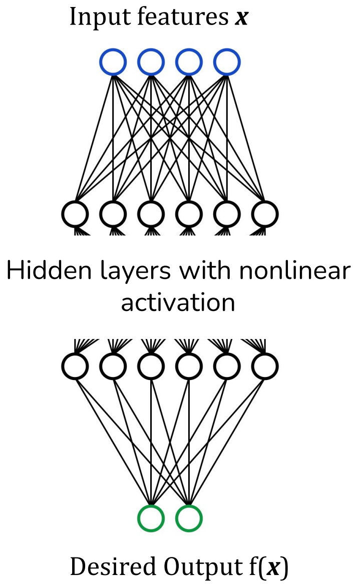
arxiv: 2110.13330
github: https://github.com/CVC-Lab/RobustPINNs

Avik Roy

November 10, 2021

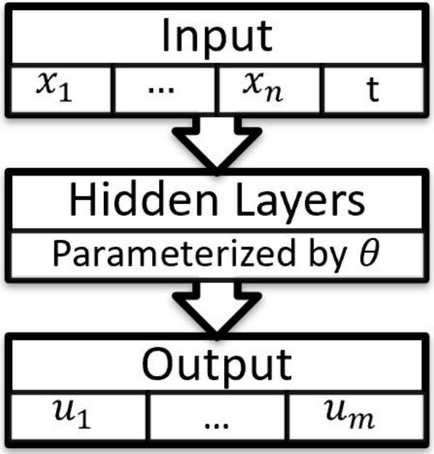
Neural Networks are Function Estimators

- Neural Networks are universal function approximators
- A network can approximate well-behaved nonlinear functions with arbitrary accuracy when equipped with
  - sufficiently large number of nodes and hidden layers
  - nonlinear activations
  - a large training dataset
- A NN can become a reliable surrogate for a nonlinear function


Input features $\boldsymbol{x}$
Hidden layers with nonlinear activation
Desired Output f($\boldsymbol{x}$)

Physics-Informed Neural Network (PINN)

- Can we learn a function when the physics of a function is known?
- Evolution of physical fields often described by partial differential equations - can we use NNs to find their solutions?
- PINN: A neural network trained to approximate spatio-temporal evolution of a set of complex fields



Physics-Informed Neural Network (PINN)

- PINNs can solve a set of coupled PDEs when
    - The PDEs are known to be *uniquely solvable*
    - The spatio-temporal boundary conditions are known
- The parameters (θ) are optimized to enforce the physics by evaluating the gradients of the NN surrogate of the fields and enforcing the physics
- A potentially powerful tool for learning physical systems where data is expensive, so training must depend on small datasets
- Exploits Autograd functionality of modern ML libraries to construct loss functions for training MLPs

Formulation of PINNs - The Setup

- The physics (i.e. the PDE):

Collection of Fields described by the PDE

$$\mathcal{N}\left[u(\vec{x}), f(\vec{x})\right] = 0$$

Analytically known source functions

- A set of initial/boundary conditions:

$$\mathcal{B}\left[u(\vec{x} \in \partial D)\right] = 0$$

Domain boundary

- A DNN surrogate of the solution

$$\tilde{u}(\vec{x}) = \mathbf{NN}_\theta\left(\vec{x}; \underbrace{\mathcal{U}_B, \mathcal{U}_C, \mathcal{U}_D}\right)$$

Training Datasets

The Training Datasets

- The dataset:
  - Boundary points: A collection of measurement points on the domain boundary and known physical measurements at those points

$$\mathcal{U}_B = \{(\vec{x}_i^b, \mathcal{B}[u(\vec{x}_i^b)])_{i=1}^{N_b}\}$$

  - Collocation points: A collection of large number of points within the domain and known source function values at those points

$$\mathcal{U}_C = \{(\vec{x}_i^c, f(\vec{x}_i^c))_{i=1}^{N_c}\}$$

  - Data points (optional): Any set of additional measurements of the fields

$$\mathcal{U}_D = \{(\vec{x}_i^d, u(\vec{x}_i^d))_{i=1}^{N_d}\}$$

Formulation of PINNs - The Training

- The Loss function: $\mathcal{L}_{PINN} = \alpha_{BC}\mathcal{L}_{BC} + \alpha_{PDE}\mathcal{L}_{PDE} + \alpha_D\mathcal{L}_D$

- Boundary Loss: $$\mathcal{L}_{BC} = \frac{1}{N_b}\sum_i \left|\mathcal{B}[\tilde{u}(\vec{x}_i^b)]\right|^2$$

- Physics Loss: $$\mathcal{L}_{PDE} = \frac{1}{N_c}\sum_i \left|\mathcal{N}[\tilde{u}(\vec{x}_i^c), f(\vec{x}_i^c)]\right|^2$$

- Data Loss: $$\mathcal{L}_D = \frac{1}{N_d}\sum_i \left|\tilde{u}(\vec{x}_i^d) - u(\vec{x}_i^d)\right|^2$$

- The parameters of the DNN are obtained from loss minimization:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}_{PINN}$$

An Example Problem: Nonlinear Schrödinger Equation

- Spatio-temporal evolution of 1D complex field $h(x, t) = u(x, t) + iv(x,t)$

$$\mathcal{N} := i\frac{\partial h}{\partial t} + \frac{1}{2}\frac{\partial^2 h}{\partial x^2} + |h|^2 h = 0$$

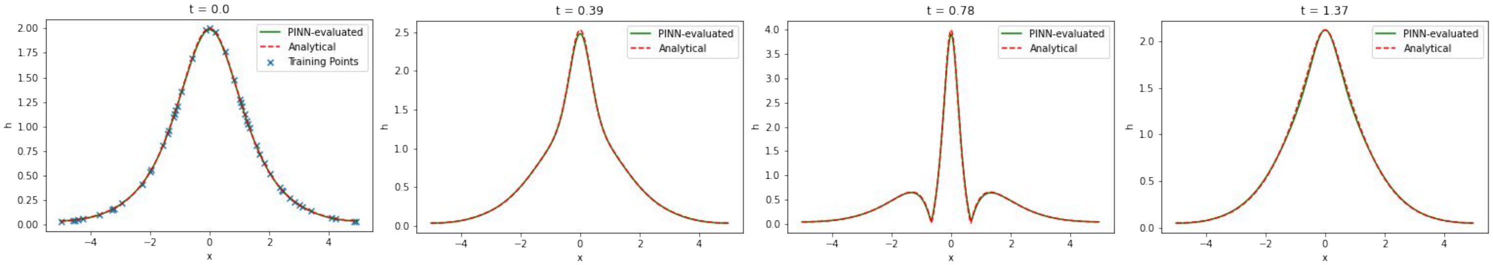
$$(x, t) \epsilon [-5, 5] \times [0, \pi/2]$$

- ***h*(*x*, *t*)** may represent traveling EM field in optical fibers or planar waveguides
- Cauchy Boundary Conditions:
  - Field evaluation on a sample of points on initial timeslice: $h(x_i, 0) = 2\ \mathrm{sech}(x_i) + \varepsilon_\mathrm{i}$
  - Periodic boundary conditions: $h(+5, t_j) = h(-5, t_j)$ and $h_x(+5, t_j) = h_x(-5, t_j)$
- ε_i represents a complex corruption error, when enabled each component is drawn from zero-mean Gaussian distributions

Solution obtained from a PINN: Error-free data

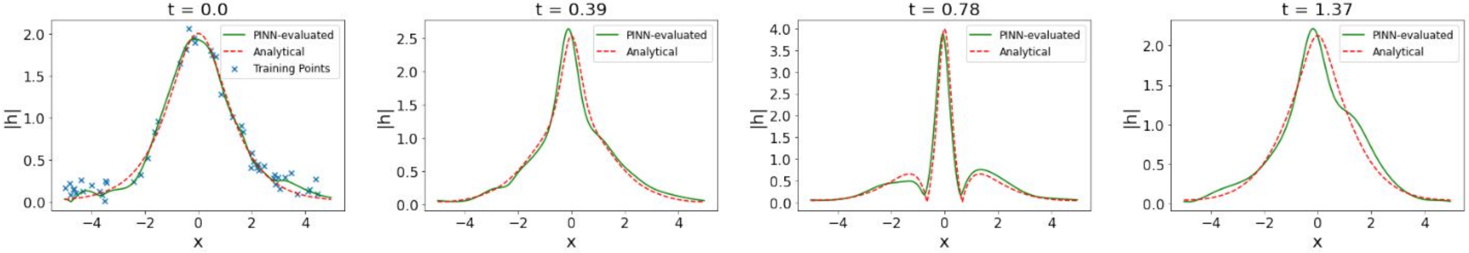
- Training with error free data: $\varepsilon_i = 0$
- The NN is a simple MLP with 6 hidden layers with 70 nodes per hidden layer
- 50 points taken on initial timeslice and 50 more for the periodic boundary conditions
- 20000 randomly points chosen points within the space-time grid to enforce physics
- Iterated for 50k times with Adam optimizer with a learning rate of 1e-3

$|h(x, t)|$



Error Propagation in PINNs: $\varepsilon_i \neq 0$

- Data collected on domain boundary can be subject to noise, errors in measurement, or systematic uncertainties
- Choosing each component of ε_i from a zero mean Gaussian distribution with a standard deviation of 0.1 and training with the same architecture
- Overfitted PINNs tend to propagate errors


t = 0.0

t = 0.39

t = 0.78

t = 1.37

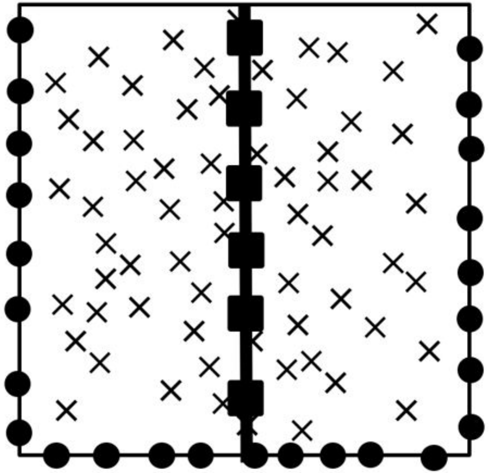
Overfitting PINNs

- PINNs fail to self-correct when initial dataset is corrupted
- For a PINN to work we need $N_c >> N_b$
- The number of parameters for a PINN architecture is much larger than the number of training points - leads to overfitting
- PINN converges to a local minima of the loss function where physics is obeyed and the field overfits on the domain boundary
- The PINN dynamically propagates the overfitted field over the entire domain

Regularization of PINNs-I: continuity conservation

- Can we regularize PINNs using physics inspired regularization?
- One variant of PINN is called conservative PINNs-
  - Divide the domain into smaller subdomains
  - Train a PINN for each subdomain
  - Apply functional and flux continuity on subdomain interfaces

$$\mathcal{L}_{cPINN} = \sum_{j=1}^{d} \mathcal{L}^{j}_{PINN} + \alpha_I \mathcal{L}^{j}_{I}$$



- ● Boundary points
- ■ Interface points
- × Collocation points

The cPINN Loss Function

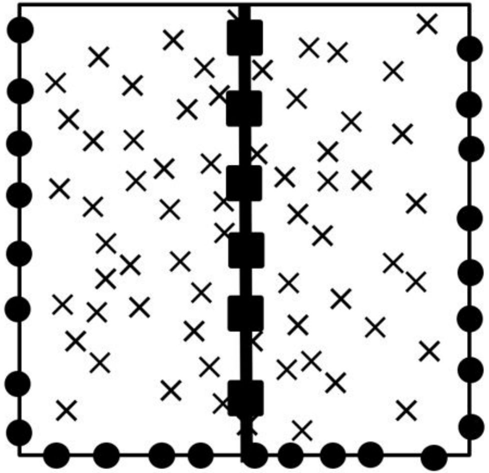
$$\mathcal{L}_{cPINN} = \sum_{j=1}^{d} \mathcal{L}^{j}_{PINN} + \alpha_I \mathcal{L}^{j}_{I}$$

PINN loss Interface loss

$$\mathcal{L}_I^j = \frac{1}{N_{Ij}} \sum_{i=1}^{N_{Ij}} \left(\left| \tilde{u}_j(\vec{x}_i^j) - \tilde{u}_{j+1}(\vec{x}_i^j) \right|^2 + \left| \nabla \tilde{u}_j(\vec{x}_i^j) \cdot \mathbf{n_i^j} - \nabla \tilde{u}_{j+1}(\vec{x}_i^j) \cdot \mathbf{n_i^{j+1}} \right|^2 \right)$$

Functional continuity

Flux continuity



● Boundary points

■ Interface points

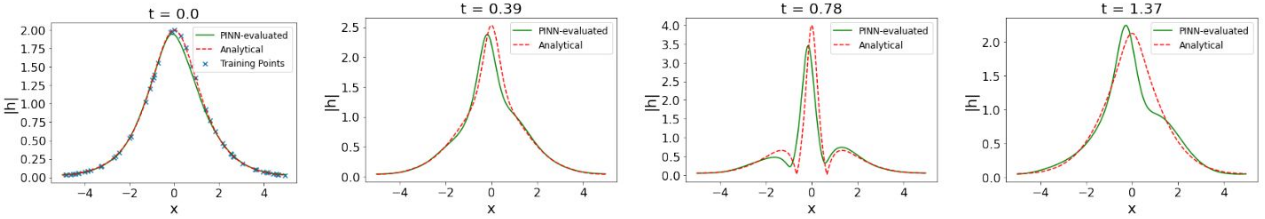
× Collocation points

The interface loss is a regularizer of the PINN loss function. How does it change the PINN’s behavior?

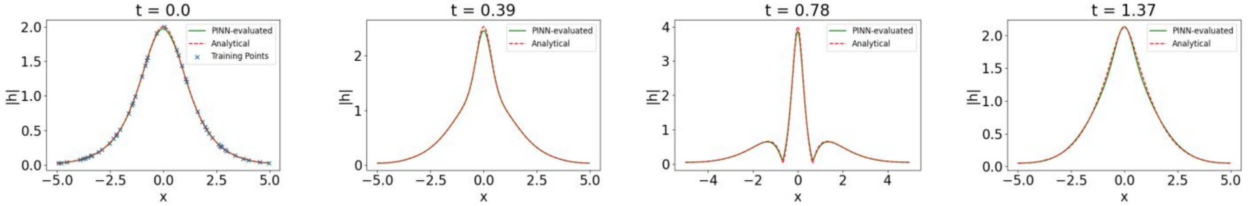
Performance of cPINNs

- cPINNs’ performance depends on the choice of subdomain boundaries
- cPINNs with two and three equal spatial subdomains with $\varepsilon_i = 0$

2 domain cPINN

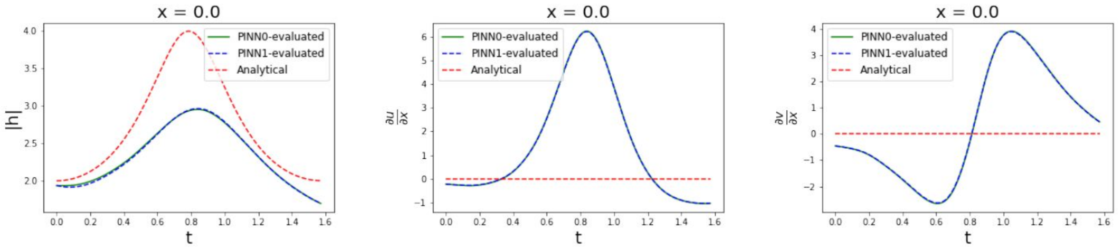


3 domain cPINN

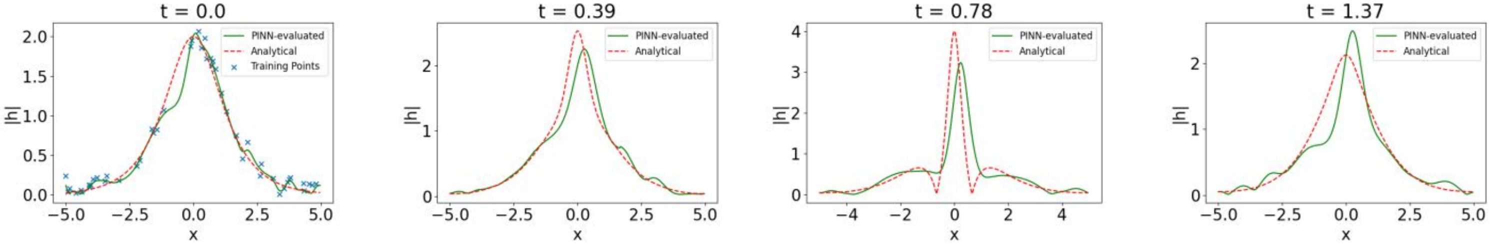


Performance of cPINNs

- cPINN can converge without reaching the solution of the analytical solution



- This behavior of cPINN prohibits it from recovering the intended solution

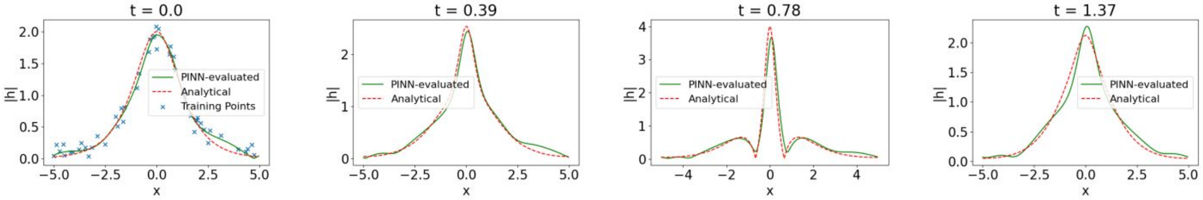


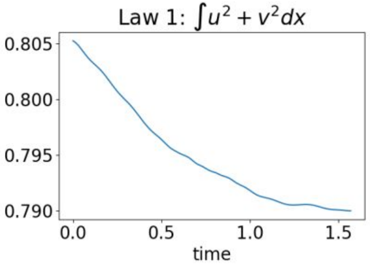
3 domain cPINN, non-zero error

Regularization of PINNs-II:conservation laws

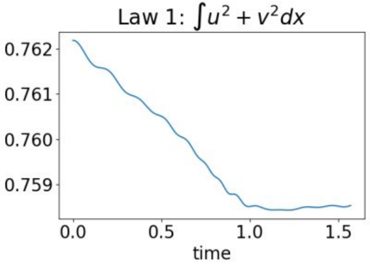
- Conservation laws associated with physical processes can be thought of as regularizers
- One conservation law for nonlinear Schrodinger equation

$$\int |h(x,t)|^2 dx = \int (u(x,t)^2 + v(x,t)^2) dx = C$$





without regularization



with regularization

Introducing Gaussian Processes

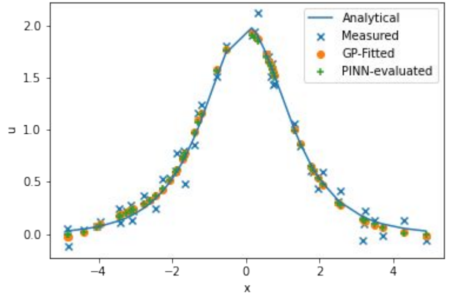
- Consider a physical process **X$_t$** indexed by some continuous variable ***t*** such that for any finite collection of samples **X$_i$ … X$_k$** represent a jointly Gaussian distribution
- In our case, we can treat the real and imaginary components of the h(x, 0) field as Gaussian processes with

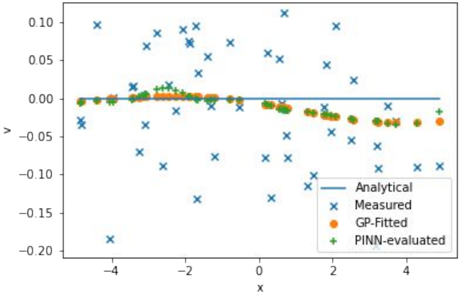
$$\begin{aligned}
&\mathbb{E}(U_i) = 2\text{sech}(x = x_i)\\
&\mathbb{E}(V_i) = 0\\
&\text{Cov}(U_i, U_j) = \text{Cov}(V_i, V_j) = \sigma^2 \delta_{ij}
\end{aligned}$$

- Based on a set of samples observed for a Gaussian process, one can use Gaussian process regression to obtain a joint distribution of any finite subset of the processes
- Pairwise covariance is estimated by a kernel function

Gaussian Process (GP) Based Smoothing

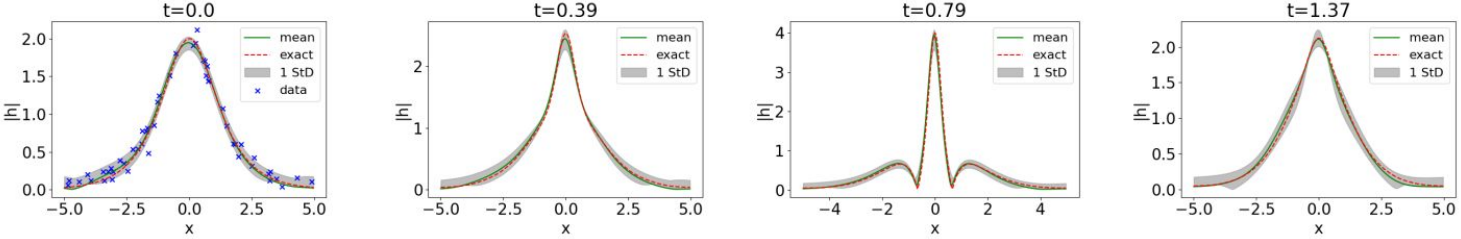
- Applying Gaussian Process based smoothing can suppress error propagation in PINNs
- Instead of using a fixed order polynomial- GPs can prevent underfitting or overfitting in the smoothing process
- Used a RBF + White Noise kernel to fit the data- a useful choice in most cases when the surface data is expected to be sufficiently smooth and errors are uncorrelated





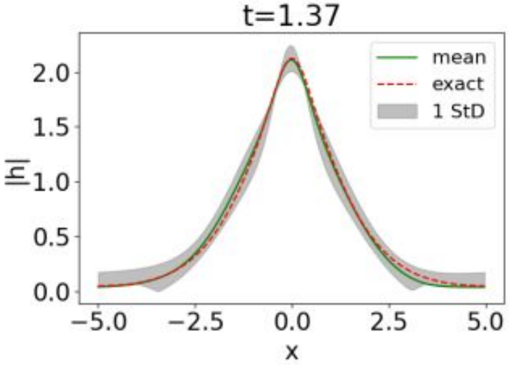
Results from GP-smoothed PINN

- Apply GP Regression (GPR) on initial timeslice based on a cross-validated choice of kernel function
- Use the smoothed evaluation of the field on initial timeslice to train the PINN
- Harness the smoothing power of GPR along with NN’s universal approximator to obtain a robust solution of the PDE



Propagation of Uncertainty

- Error propagation of unregularized PINNs corrupt the estimated lineshape
- GP-smoothed PINN allows recovering the expected lineshape along with a variance estimation for field evolution
- Steps:
  - Train the NN with GP-smoothed initial condition
  - Update the initial condition with +1σ or -1σ band of initial condition lineshape
  - Start with optimized θ and retrain the network to reoptimize them to get θ ± $\delta\theta$ for the updated initial conditions
  - Draw inference from the reoptimized NNs to get the uncertainty bands at later times



$$\tilde{u}(\vec{x}) \pm \delta\tilde{u}(\vec{x}) = \mathbf{NN}_{\theta\pm\delta\theta}\left(\vec{x};\hat{\mathcal{U}}_B^{\pm},\mathcal{U}_C,\mathcal{U}_D\right)$$

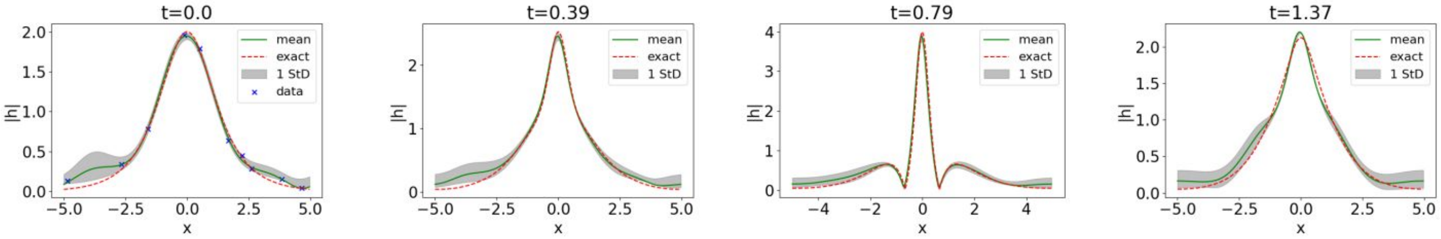
$$\hat{\mathcal{U}}_B^{\pm} = \{(\vec{x}_i^b, \mathcal{B}[\hat{u}(\vec{x}_i^b) \pm \delta\hat{u}(\vec{x}_i^b)])_{i=1}^{N_b}\}$$

$$(\theta \pm \delta\theta)^* = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}_{PINN}(\theta;\hat{\mathcal{U}}_B^{\pm})$$

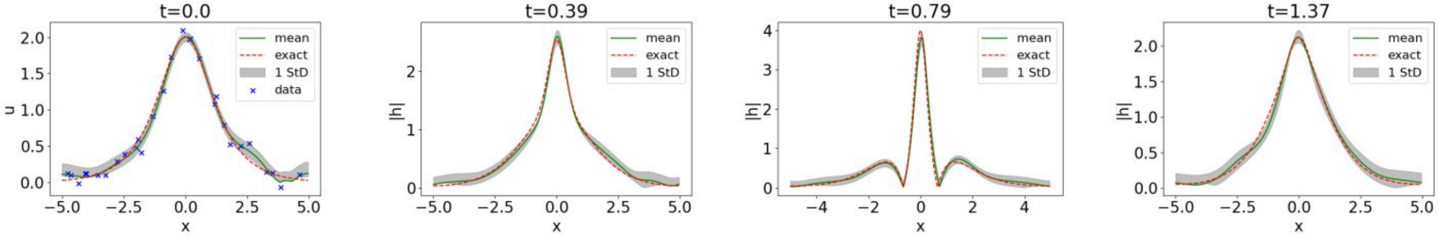
Sparse Gaussian Processes (SGP) for Smoothing

- Do we need all our observations to optimize the Gaussian Process at the initial timeslice?
- Use a sparse subset of the observations to obtain the optimized GPR
- This selection is based on a greedy algorithm
    - Start with a random subset of the observations
    - Get a tentative fit for the GPR
    - Only include observations that are “far enough” from current selection of points
    - The total number of points is bounded by some upper limit, *M*
    - Reoptimize the GPR kernel once all points are selected

Performance of SGP Smoothing



Too few Inducing Points (M = 10), poor fit



Adequate Inducing Points (M = 30, 29 chosen by the algorithm), reasonable fit

Conclusion

- PINNs: powerful tools in the interface of physics and ML
- Making PINNs robust against noises in training data is an important challenge
- Physics inspired regularizers can fall short to auto-correct against error propagation
- GP smoothed PINNs and its sparse variation can prove useful in ensuring robustness
- Our experiments suggest this provides better safeguard against other proposed methods like adversarial uncertainty quantification