ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Seminar for Applied Mathematics SAM

On generalization error estimates of physics informed neural networks for approximating dispersive PDEs

B. Genming and U. Koley and S. Mishra and R. Molinaro

Research Report No. 2021-13
April 2021

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

On generalization error estimates of physics informed neural networks for approximating dispersive PDEs

Genming Bai * Ujjwal Koley † Siddhartha Mishra * Roberto Molinaro *

April 13, 2021

* Seminar for Applied Mathematics (SAM), D-Math
ETH Zürich, Rämistrasse 101.
gbai@student.ethz.ch, siddhartha.mishra@sam.math.ethz.ch,
roberto.molinaro@sam.math.ethz.ch

† Centre for Applicable Mathematics, Tata Institute of Fundamental Research
P.O. Box 6503, GKVK Post Office, Bangalore 560065, India
ujjwal@math.tifrbng.res.in

Abstract

Physics informed neural networks (PINNs) have recently been widely used for robust and accurate approximation of PDEs. We provide rigorous upper bounds on the generalization error of PINNs approximating solutions of the forward problem for several dispersive PDEs.

1 Introduction

Our aim is to show that PINNs approximate a wide variety of PDEs including dispersive PDEs. We explore this framework to dispersive PDEs like KdV type equations, BO type equations, and Camassa-Holm type equations.

2 Mathematical framework for PINNs

In this section, following [**?**], we recapitulate some of the relevant mathematical tools to be used in the subsequent analysis.

2.1 The underlying abstract PDE

Let X, Y be separable Banach spaces with norms $\|\cdot\|_X$ and $\|\cdot\|_Y$, respectively. For definiteness, we set $Y = L^p(\mathbb{D};\mathbb{R}^m)$ and $X = W^{s,q}(\mathbb{D};\mathbb{R}^m)$, for $m \geqslant 1$, $1 \leqslant p, q < \infty$ and $s \geqslant 0$, with $\mathbb{D} \subset \mathbb{R}^{\bar{d}}$, for some $\bar{d} \geqslant 1$. In particular, we will also consider space-time domains with $\mathbb{D} = (0,T) \times D \subset \mathbb{R}^d$ with $d \geqslant 1$. In this case $\bar{d} = d + 1$. Let $X^* \subset X$ and $Y^* \subset Y$ be closed subspaces with norms $\|\cdot\|_{X^*}$ and $\|\cdot\|_{Y^*}$, respectively.

We start by considering the following abstract formulation of our underlying PDE:

$$\mathcal{D}(\mathbf{u}) = \mathbf{f}. \tag{2.1}$$

Here, the *differential operator* is a mapping, $\mathcal{D} : X^* \mapsto Y^*$ and the *input* $\mathbf{f} \in Y^*$, such that

$$\begin{aligned} &(H1): \quad \|\mathcal{D}(\mathbf{u})\|_{Y^*} < +\infty, \quad \forall\ \mathbf{u} \in X^*, \text{ with } \|\mathbf{u}\|_{X^*} < +\infty. \\ &(H2): \quad \|\mathbf{f}\|_{Y^*} < +\infty. \end{aligned} \tag{2.2}$$

Moreover, we assume that for all $\mathbf{f} \in Y^*$, there exists a unique $\mathbf{u} \in X^*$ such that (2.1) holds.

2.2 Quadrature rules

In the following section, we need to consider approximating integrals of functions. Hence, we need an abstract formulation for quadrature. To this end, we consider a mapping $g : \mathbb{D} \mapsto \mathbb{R}^m$, such that $g \in Z^* \subset Y^*$. We are interested in approximating the integral,

$$\overline{g} := \int_{\mathbb{D}} g(y)dy,$$

with dy denoting the $\bar{d}$-dimensional Lebesgue measure. In order to approximate the above integral by a quadrature rule, we need the quadrature points $y_i \in \mathbb{D}$ for $1 \leqslant i \leqslant N$, for some $N \in \mathbb{N}$ as well as weights w_i, with $w_i \in \mathbb{R}_+$. Then a quadrature is defined by,

$$\overline{g}_N := \sum_{i=1}^{N} w_i g(y_i), \tag{2.3}$$

for weights w_i and quadrature points y_i. We further assume that the quadrature error is bounded as,

$$\left|\overline{g} - \overline{g}_N\right| \leqslant C_{quad}\left(\|g\|_{Z^*}, \bar{d}\right) N^{-\alpha}, \tag{2.4}$$

for some $\alpha > 0$.

2.3 PINNs

In this section, we will describe physics-informed neural networks (PINNs). We start with a description of neural networks which form the basis of PINNs.

2.3.1 Neural Networks.

Given an input $y \in \mathbb{D}$, a feedforward neural network (also termed as a multi-layer perceptron), shown in figure 1, transforms it to an output, through a layer of units (neurons) which compose of either affine-linear maps between units (in successive layers) or scalar non-linear activation functions within units, resulting in the representation,

$$\mathbf{u}_\theta(y) = C_K \circ \sigma \circ C_{K-1} \ldots\ldots\ldots \circ \sigma \circ C_2 \circ \sigma \circ C_1(y). \tag{2.5}$$

Here, $\circ$ refers to the composition of functions and σ is a scalar (non-linear) activation function. Popular choices for the activation function σ in (2.5) include the sigmoid function, the hyperbolic tangent function and the $ReLU$ function.

For any $1 \leqslant k \leqslant K$, we define

$$C_k z_k = W_k z_k + b_k, \quad \text{for } W_k \in \mathbb{R}^{d_{k+1} \times d_k}, z_k \in \mathbb{R}^{d_k}, b_k \in \mathbb{R}^{d_{k+1}}. \tag{2.6}$$

For consistency of notation, we set $d_1 = \bar{d}$ and $d_K = m$.

Our neural network (2.5) consists of an input layer, an output layer and $(K-1)$ hidden layers for some $1 < K \in \mathbb{N}$. The k-th hidden layer (with d_k neurons) is given an input vector $z_k \in \mathbb{R}^{d_k}$ and transforms it first by an affine linear map C_k (2.6) and then by a nonlinear (component wise) activation σ. A straightforward addition shows that our network contains $\left(\bar{d} + m + \sum_{k=2}^{K-1} d_k\right)$ neurons. We also denote,

$$\theta = \{W_k, b_k\},\ \theta_W = \{W_k\} \quad \forall\ 1 \leqslant k \leqslant K, \tag{2.7}$$

to be the concatenated set of (tunable) weights for our network. It is straightforward to check that $\theta \in \Theta \subset \mathbb{R}^M$ with

$$M = \sum_{k=1}^{K-1} (d_k + 1) d_{k+1}. \tag{2.8}$$

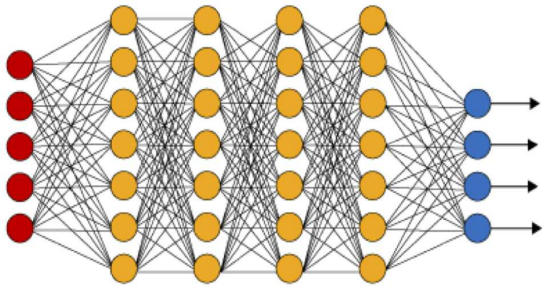


Figure 1: An illustration of a (fully connected) deep neural network. The red neurons represent the inputs to the network and the blue neurons denote the output layer. They are connected by hidden layers with yellow neurons. Each hidden unit (neuron) is connected by affine linear maps between units in different layers and then with nonlinear (scalar) activation functions within units.

2.3.2 Training PINNs: Loss functions and optimization

The neural network $\mathbf{u}_\theta$ (2.5) depends on the tuning parameter $\theta \in \Theta$ of weights and biases. Within the standard paradigm of deep learning [9], one *trains* the network by finding tuning parameters θ such that the loss (error, mismatch, regret) between the neural network and the underlying target is minimized. Here, our target is the solution $\mathbf{u} \in X^*$ of the abstract PDE (2.1) and we wish to find the tuning parameters θ such that the resulting neural network $\mathbf{u}_\theta$ approximates $\mathbf{u}$.

Following standard practice of machine learning, one obtains training data $\mathbf{u}(y)$, for all $y \in \mathcal{S}$, with training set $\mathcal{S} \subset \mathbb{D}$ and then minimizes a loss function of the form $\sum_{\mathcal{S}} \|\mathbf{u}(y) - \mathbf{u}_\theta(y)\|_X$ to find the neural network approximation for $\mathbf{u}$. However, obtaining this training data requires possibly expensive numerical simulations of the underlying PDE (2.1). In order to circumvent this issue, the authors of [17] suggest a different strategy. An abstract paraphrasing of this strategy runs as follows: we assume that for every $\theta \in \Theta$, the neural network $\mathbf{u}_\theta \in X^*$ and $\|\mathbf{u}_\theta\|_{X^*} < +\infty$. We define the following *residual*:

$$\mathcal{R}_\theta = \mathcal{R}(\mathbf{u}_\theta) := \mathcal{D}(\mathbf{u}_\theta) - \mathbf{f}. \tag{2.9}$$

By assumptions (H1),(H2) (cf. (2.2)), we see that $\mathcal{R} \in Y^*$ and $\|\mathcal{R}\|_{Y^*} < +\infty$ for all $\theta \in \Theta$. Note that $\mathcal{R}(\mathbf{u}) = \mathcal{D}(\mathbf{u}) - \mathbf{f} \equiv 0$, for the solution $\mathbf{u}$ of the PDE (2.1). Hence, the term *residual* is justified for (2.9).

The strategy of PINNs, following [17], is to minimize the *residual* (2.9), over the admissible set of tuning parameters $\theta \in \Theta$ i.e

$$\text{Find } \theta^* \in \Theta: \quad \theta^* = \arg\min_{\theta \in \Theta} \|\mathcal{R}_\theta\|_Y. \tag{2.10}$$

Realizing that $Y = L^p(\mathbb{D})$ for some $1 \leqslant p < \infty$, we can equivalently minimize,

$$\text{Find } \theta^* \in \Theta: \quad \theta^* = \arg\min_{\theta \in \Theta} \|\mathcal{R}_\theta\|^p_{L^p(\mathbb{D})} = \arg\min_{\theta \in \Theta} \int_{\mathbb{D}} |\mathcal{R}_\theta(y)|^p dy. \tag{2.11}$$

As it will not be possible to evaluate the integral in (2.11) exactly, we need to approximate it numerically by a quadrature rule. To this end, we use the quadrature rules (2.3) discussed earlier and select the *training set* $\mathcal{S} = \{y_n\}$ with $y_n \in \mathbb{D}$ for all $1 \leqslant n \leqslant N$ as the quadrature points for the quadrature rule (2.3) and consider the following *loss function*:

$$J(\theta) := \sum_{n=1}^{N} w_n |\mathcal{R}_\theta(y_n)|^p = \sum_{n=1}^{N} w_n \left|\mathcal{D}(\mathbf{u}_\theta(y_n)) - \mathbf{f}(y_n)\right|^p. \tag{2.12}$$

It is common in machine learning [9] to regularize the minimization problem for the loss function i.e we seek to find,

$$\theta^* = \arg\min_{\theta \in \Theta} \left(J(\theta) + \lambda_{reg} J_{reg}(\theta)\right). \tag{2.13}$$

Here, $J_{reg} : \Theta \to \mathbb{R}$ is a *regularization* (penalization) term. A popular choice is to set $J_{reg}(\theta) = \|\theta_W\|_q^q$ for either $q = 1$ (to induce sparsity) or $q = 2$. The parameter $0 \leqslant \lambda_{reg} \ll 1$ balances the regularization term with the actual loss J (2.12).

The proposed algorithm for computing this PINN is given below,

Algorithm 2.1. Finding a physics informed neural network to approximate the solution of the PDE (2.1).

Inputs: *Underlying domain* $\mathbb{D}$*, differential operator* $\mathcal{D}$ *and input source term* $\mathbf{f}$ *for the PDE* (2.1)*, quadrature points and weights for the quadrature rule* (2.3)*, non-convex gradient based optimization algorithms.*

Goal: *Find PINN* $\mathbf{u}^* = \mathbf{u}_{\theta^*}$ *for approximating the PDE* (2.1).

Step 1: *Choose the training set* $\mathcal{S} = \{y_n\}$ *for* $y_n \in \mathbb{D}$*, for all* $1 \leqslant n \leqslant N$ *such that* $\{y_n\}$ *are quadrature points for the underlying quadrature rule* (2.3).

Step 2: *For an initial value of the weight vector* $\overline{\theta} \in \Theta$*, evaluate the neural network* $\mathbf{u}_{\overline{\theta}}$ (2.5)*, the PDE residual* (2.9)*, the loss function* (2.13) *and its gradients to initialize the underlying optimization algorithm.*

Step 3: *Run the optimization algorithm till an approximate local minimum* θ^* *of* (2.13) *is reached. The map* $\mathbf{u}^* = \mathbf{u}_{\theta^*}$ *is the desired PINN for approximating the solution* $\mathbf{u}$ *of the PDE* (2.1).

3 Korteweg de-Vries & Kawahara equations

3.1 The underlying PDEs

We consider a generalized nonlinear dispersive equation, known as Kawahara equation, which has a form of the Korteweg de-Vries equation with an additional fifth order derivative term given by

$$\begin{aligned} u_t + uu_x + \alpha u_{xxx} - \beta u_{xxxxx} = 0, \quad \forall x \in (0,1),\, t \in (0,T), \\ u(x,0) = \bar{u}(x), \quad \forall x \in (0,1), \\ u(0,t) = h_1(t),\, u(1,t) = h_2(t),\, u_x(0,t) = h_3(t),\, u_x(1,t) = h_4(t),\, u_{xx}(1,t) = h_5(t),\ \forall\, t \in (0,T). \end{aligned} \tag{3.1}$$

Here α, β are non-negative real constants. Note that if $\beta = 0$, then the above equation is called Korteweg de-Vries equation, and if $\beta \neq 0$, then the above equation is called Kawahara equation. It is well known that KdV equation plays a pivotal role in the modeling of shallow water waves, and in particular, the one-dimensional waves of small but finite amplitude in dispersive systems can be described by the KdV equation. However, under certain circumstances, the coefficient of the third order derivative in the KdV equation may become very small or even zero. In such a scenario, one has to take account of the higher order effect of dispersion in order to balance the nonlinear effect, and consider generalized nonlinear dispersive equation, known as Kawahara equation. For more specific physical background of Kawahara equation, we refer to the work by Hunter and Scheurle [12].

For the sake of simplicity it will be assumed $\alpha = \beta = 1$ in the upcoming analysis, since their values are not relevant in the present setting, and needless to mention that the analysis also works for the case $\beta = 0$ (KdV case). Regarding the existence of solutions to (3.1), we closely follow the work by Faminskii & Larkin [7], and report the following result.

Theorem 3.1. *For any integer* $k \geq 0$*,* $l = 1$ *or* 2*, define the spaces*

$$\mathcal{X}_k((0,1)\times(0,T)) := \left\{ u : \partial_t^n u \in C([0,T]; H^{5(k-n)}(0,1)) \cap L^2((0,T); H^{5(k-n)+1}(0,1)) \right\},$$

$$\mathcal{B}_k^l(0,T) := \prod_{j=0}^{l} H^{k+(2-j)/5}(0,T).$$

Let $\bar{u} \in H^{5k}(0,1)$*, boundary data* $(h_1, h_3) \in \mathcal{B}_k^1(0,T)$*, and* $(h_2, h_4, h_5) \in \mathcal{B}_k^2(0,T)$ *satisfy the natural compatibility conditions. Then there exists a unique solution* $u \in \mathcal{X}_k$*, and the flow map in Lipschitz continuous on any ball in the corresponding norm.*

Remark 3.2. Since a function which is contained in Sobolev spaces of arbitrary order is in fact smooth, thus to obtain classical solution of (3.1) we may choose intial and boundary data from appropriate Sobolev spaces. ∎

Remark 3.3. We remark that all the upcoming calculations are valid for more general dispersive equations of the following form: For $l \in \mathbb{N}$

$$u_t + (-1)^{l+1}\partial_x^{2l+1}u + uu_x = f,$$

with appropriate boundary conditions. ■

3.2 PINNs

We first specify the training set $\mathcal{S}$, and define appropriate residuals to run the algorithm 2.1. In what follows, we begin with the description of the training set $\mathcal{S}$.

3.2.1 Training Set.

Let us define the space-time domain $\Omega_T = (0,1)\times(0,T)$, and divide the training set $\mathcal{S} = \mathcal{S}_{int} \cup \mathcal{S}_{sb} \cup \mathcal{S}_{tb}$ of the abstract PINNs algorithm 2.1 into the following three subsets,

(a) Interior training points $\mathcal{S}_{int} = \{y_n\}$ for $1 \leqslant n \leqslant N_{int}$, with each $y_n = (x_n, t_n) \in \Omega_T$. We use low-discrepancy Sobol points as training points.

(b) Spatial boundary training points $\mathcal{S}_{sb} = (0,t_n) \cup (1,t_n)$ for $1 \leqslant n \leqslant N_{sb}$, and the points t_n chosen as low-discrepancy Sobol points.

(c) Temportal boundary training points $\mathcal{S}_{tb} = \{x_n\}$, with $1 \leqslant n \leqslant N_{tb}$ and each $x_n \in (0,1)$, chosen as low-discrepancy Sobol points.

3.2.2 Residuals

To define residuals for the neural network $u_\theta \in C^k([0,T]\times[0,1])$, defined by (2.5), with $\theta \in \Theta$ as the set of tuning parameters, we use the hyperbolic tangent tanh activation function, i.e., $\sigma = \tanh$. With this setting, we define the following residuals

(a) Interior Residual given by,

$$\mathcal{R}_{int,\theta}(x,t) := \partial_t u_\theta(x,t) + u_\theta(u_\theta)_x(x,t) + (u_\theta)_{xxx}(x,t) - (u_\theta)_{xxxxx}(x,t). \tag{3.2}$$

Note that the above residual is well-defined and $\mathcal{R}_{int,\theta} \in C^{k-5}([0,T]\times[0,1])$ for every $\theta \in \Theta$.

(b) Spatial boundary Residual given by,

$$\begin{aligned}
&\mathcal{R}_{sb1,\theta}(0,t) := u_\theta(0,t) - h_1(t), \quad \forall t \in (0,T),\\
&\mathcal{R}_{sb2,\theta}(1,t) := u_\theta(1,t) - h_2(t), \quad \forall t \in (0,T),\\
&\mathcal{R}_{sb3,\theta}(0,t) := (u_\theta)_x(0,t) - h_3(t), \quad \forall t \in (0,T),\\
&\mathcal{R}_{sb4,\theta}(1,t) := (u_\theta)_x(1,t) - h_4(t), \quad \forall t \in (0,T),\\
&\mathcal{R}_{sb5,\theta}(1,t) := (u_\theta)_{xx}(1,t) - h_5(t), \quad \forall t \in (0,T).
\end{aligned} \tag{3.3}$$

Given the fact that the neural network and boundary data are smooth, above residuals are well-defined.

(c) Temporal boundary Residual given by,

$$\mathcal{R}_{tb,\theta}(x) := u_\theta(x,0) - \bar{u}(x), \quad \forall x \in (0,1). \tag{3.4}$$

Again the above quantity is well-defined and $\mathcal{R}_{tb,\theta} \in C^k((0,1))$, as both the initial data and the neural network are smooth.

3.2.3 Loss function

We set the following loss function

$$J(\theta) := \sum_{n=1}^{N_{tb}} w_n^{tb} |\mathcal{R}_{tb,\theta}(x_n)|^2 + \sum_{n=1}^{N_{sb}} \sum_{i=1}^{5} w_n^{sb} |\mathcal{R}_{sbi,\theta}(t_n)|^2 + \lambda \sum_{n=1}^{N_{int}} w_n^{int} |\mathcal{R}_{int,\theta}(x_n, t_n)|^2. \tag{3.5}$$

Here the residuals are defined by (3.4), (3.3), (3.2), w_n^{tb} are the N_{tb} quadrature weights corresponding to the temporal boundary training points $\mathcal{S}_{tb}$, w_n^{sb} are the N_{sb} quadrature weights corresponding to the spatial boundary training points $\mathcal{S}_{sb}$ and w_n^{int} are the N_{int} quadrature weights corresponding to the interior training points $\mathcal{S}_{int}$. Furthermore, λ is a hyperparameter for balancing the residuals, on account of the PDE and the initial and boundary data, respectively.

3.3 Estimate on the generalization error

We are interested in estimating the following generalization error for the PINN $u^* = u_{\theta^*}$ with loss function (3.5), for approximating the solution of (3.1):

$$\mathcal{E}_G := \left(\int_0^T \int_0^1 |u(x,t) - u^*(x,t)|^2 dx dt \right)^{\frac{1}{2}}. \tag{3.6}$$

We are going to estimate the generalization error in terms of the *training error* that we define as,

$$\mathcal{E}_T^2 := \underbrace{\sum_{n=1}^{N_{tb}} w_n^{tb} |\mathcal{R}_{tb,\theta^*}(x_n)|^2}_{(\mathcal{E}_T^{tb})^2} + \underbrace{\sum_{n=1}^{N_{sb}} \sum_{i=1}^{5} w_n^{sb} |\mathcal{R}_{sbi,\theta^*}(t_n)|^2}_{(\mathcal{E}_T^{sb})^2} + \underbrace{\lambda \sum_{n=1}^{N_{int}} w_n^{int} |\mathcal{R}_{int,\theta^*}(x_n, t_n)|^2}_{(\mathcal{E}_T^{int})^2}. \tag{3.7}$$

Note that the training error can be readily computed *a posteriori* from the loss function (3.5).

We also need the following assumptions on the quadrature error. For any function $g \in C^k(\Omega)$, the quadrature rule corresponding to quadrature weights w_n^{tb} at points $x_n \in \mathcal{S}_{tb}$, with $1 \leqslant n \leqslant N_{tb}$, satisfies

$$\left| \int_\Omega g(x) dx - \sum_{n=1}^{N_{tb}} w_n^{tb} g(x_n) \right| \leqslant C_{quad}^{tb} (\|g\|_{C^k}) N_{tb}^{-\alpha_{tb}}. \tag{3.8}$$

For any function $g \in C^k(\partial\Omega \times [0,T])$, the quadrature rule corresponding to quadrature weights w_n^{sb} at points $(x_n, t_n) \in \mathcal{S}_{sb}$, with $1 \leqslant n \leqslant N_{sb}$, satisfies

$$\left| \int_0^T \int_{\partial\Omega} g(x,t) ds(x) dt - \sum_{n=1}^{N_{sb}} w_n^{sb} g(x_n, t_n) \right| \leqslant C_{quad}^{sb} (\|g\|_{C^k}) N_{sb}^{-\alpha_{sb}}. \tag{3.9}$$

Finally, for any function $g \in C^\ell(\Omega \times [0,T])$, the quadrature rule corresponding to quadrature weights w_n^{int} at points $(x_n, t_n) \in \mathcal{S}_{int}$, with $1 \leqslant n \leqslant N_{int}$, satisfies

$$\left| \int_0^T \int_\Omega g(x,t) dx dt - \sum_{n=1}^{N_{int}} w_n^{int} g(x_n, t_n) \right| \leqslant C_{quad}^{int} (\|g\|_{C^\ell}) N_{int}^{-\alpha_{int}}. \tag{3.10}$$

In the above, $\alpha_{int}, \alpha_{sb}, \alpha_{tb} > 0$ and in principle, different order quadrature rules can be used. We estimate the generalization error for the PINN in the following,

Theorem 3.4. *Let $u \in C^k([0,1]\times[0,T])$ be the unique classical solution of the Korteweg de-Vries & Kawahara euqation* (3.1)*. Let $u^* = u_{\theta^*}$ be a PINN generated by algorithm 2.1, corresponding to loss function* (2.13), (3.5)*. Then the generalization error* (3.6) *can be estimated as,*

$$\begin{aligned}\varepsilon_G \leqslant\ & C_1\big(\varepsilon_T^{tb} + \varepsilon_T^{int} + C_2(\varepsilon_T^{sb}) + C_3(\varepsilon_T^{sb})^{1/2} \\ & + (C_{quad}^{tb})^{1/2} N_{tb}^{-\alpha_{tb}/2} + (C_{quad}^{int})^{1/2} N_{int}^{-\alpha_{int}/2} + C_2(C_{quad}^{sb})^{1/2} N_{sb}^{-\alpha_{sb}/2} + C_3(C_{quad}^{sb})^{1/4} N_{sb}^{-\alpha_{sb}/4}\big)\end{aligned} \tag{3.11}$$

where

$$\begin{aligned} C_1 &= \sqrt{T + 2C_4 T^2 e^{2C_4 T}}, \quad C_2 = \sqrt{\|u\|_{C_t^0 C_x^0} + 1} \\ C_3 &= \sqrt{10(\|u^*\|_{C_t^0 C_x^4} + \|u\|_{C_t^0 C_x^4})T^{1/2}}, \quad C_4 = \|u^*\|_{C_t^0 C_x^1} + \frac{1}{2}\|u\|_{C_t^0 C_x^1} + \frac{1}{2}\end{aligned} \tag{3.12}$$

and $C_{quad}^{tb} = C_{quad}^{tb}(\|\mathcal{R}_{tb,\theta^*}\|_{C^k})$, $C_{quad}^{sb} = C_{quad}^{sb}(\sum_{i=1}^{5}\|\mathcal{R}_{sbi,\theta^*}\|_{C^{k-2}})$ and $C_{quad}^{int} = C_{quad}^{int}(\|\mathcal{R}_{int,\theta^*}\|_{C^{k-5}})$ are the constants defined by the quadrature error (3.8), (3.9), (3.10)*, respectively.*

Proof. It is easy to see that the error $\hat{u} : u^* - u$ satisfies the following equation,

$$\begin{aligned} \hat{u}_t + \hat{u}_{xxx} - \hat{u}_{xxxxx} + u^* u_x^* - u u_x &= \mathcal{R}_{int}, \quad \forall x \in (0,1)\ t \in (0,T), \\ \hat{u}(x,0) &= \mathcal{R}_{tb}(x), \quad \forall x \in (0,1), \\ \hat{u}(0,t) &= \mathcal{R}_{sb1}(0,t), \quad t \in (0,T), \\ \hat{u}(1,t) &= \mathcal{R}_{sb2}(1,t), \quad t \in (0,T), \\ \hat{u}_x(0,t) &= \mathcal{R}_{sb3}(0,t), \quad t \in (0,T), \\ \hat{u}_x(1,t) &= \mathcal{R}_{sb4}(1,t), \quad t \in (0,T), \\ \hat{u}_{xx}(1,t) &= \mathcal{R}_{sb5}(1,t), \quad t \in (0,T). \end{aligned} \tag{3.13}$$

Here, we have denoted $\mathcal{R}_{int} = \mathcal{R}_{int,\theta^*}$ for notational convenience and analogously for the residuals $\mathcal{R}_{tb}, \mathcal{R}_{sb}$. Note that

$$u^* u_x^* - u u_x = \hat{u}\hat{u}_x + u\hat{u}_x + \hat{u}u_x;\ \hat{u}\hat{u}_{xxx} = (\hat{u}\hat{u}_{xx})_x - \frac{1}{2}(\hat{u}_x^2)_x,\ \hat{u}\hat{u}_{xxxxx} = (\hat{u}\hat{u}_{xxxx})_x - (\hat{u}_x\hat{u}_{xxx})_x + \frac{1}{2}(\hat{u}_{xx}^2)_x.$$

Multiplying both sides of the PDE (3.13) with $\hat{u}$, integrating over the domain and integrating by parts yields,

$$\begin{aligned} \frac{1}{2}\frac{d}{dt}\int_0^1 \hat{u}^2\,dx &= -\int_0^1 \hat{u}\hat{u}_{xxx}\,dx + \int_0^1 \hat{u}\hat{u}_{xxxxx}\,dx - \int_0^1 \hat{u}(\hat{u}\hat{u}_x - u\hat{u}_x + u_x\hat{u})\,dx + \int_0^1 \hat{u}\mathcal{R}_{int}\,dx \\ &\leqslant -\hat{u}_{xx}\hat{u}\big|_0^1 + \frac{1}{2}\,(\hat{u}_x)^2\big|_1 + \hat{u}_{xxxx}\hat{u}\big|_0^1 - \hat{u}_{xxx}\hat{u}_x\big|_0^1 + \frac{1}{2}\,(\hat{u}_{xx})^2\big|_1 \\ &\quad - \int_0^1 \hat{u}^2\hat{u}_x\,dx - \Big(\frac{1}{2}\hat{u}^2 u\Big|_0^1 - \frac{1}{2}\int_0^1 \hat{u}^2 u_x\,dx\Big) - \int_0^1 \hat{u}^2 u_x\,dx + \int_0^1 \hat{u}\mathcal{R}_{int}\,dx \\ &\leqslant \|\hat{u}\|_{C_x^4}(|\mathcal{R}_{sb1}| + |\mathcal{R}_{sb2}| + |\mathcal{R}_{sb3}| + |\mathcal{R}_{sb4}|) + \frac{1}{2}(\mathcal{R}_{sb3}^2 + \mathcal{R}_{sb5}^2) \\ &\quad + (\|u^*\|_{C_x^1} + \frac{1}{2}\|u\|_{C_x^1})\int_0^1 \hat{u}^2\,dx + \frac{1}{2}\|u\|_{C_x^0}(\mathcal{R}_{sb1}^2 + \mathcal{R}_{sb2}^2) + \frac{1}{2}\int_0^1 \mathcal{R}_{int}^2\,dx + \frac{1}{2}\int_0^1 \hat{u}^2\,dx \\ &\leqslant (\|u^*\|_{C_t^0 C_x^4} + \|u\|_{C_t^0 C_x^4})(|\mathcal{R}_{sb1}| + |\mathcal{R}_{sb2}| + |\mathcal{R}_{sb3}| + |\mathcal{R}_{sb4}|) \\ &\quad + \frac{1}{2}(\mathcal{R}_{sb3}^2 + \mathcal{R}_{sb5}^2) + \frac{1}{2}\|u\|_{C_t^0 C_x^0}(\mathcal{R}_{sb1}^2 + \mathcal{R}_{sb2}^2) + \frac{1}{2}\int_0^1 \mathcal{R}_{int}^2\,dx \\ &\quad + (\|u^*\|_{C_t^0 C_x^1} + \frac{1}{2}\|u\|_{C_t^0 C_x^1} + \frac{1}{2})\int_0^1 \hat{u}^2\,dx \\ &=: C_1(\sum_{i=1}^5 |\mathcal{R}_{sbi}|) + C_2(\sum_{i=1}^5 \mathcal{R}_{sbi}^2) + \frac{1}{2}\int_0^1 \mathcal{R}_{int}^2\,dx + C_3\int_0^1 \hat{u}^2\,dx \end{aligned} \tag{3.14}$$

Then integrating the above inequality over $[0, \bar{T}]$ for any $\bar{T} \leqslant T$, using Cauchy-Schwarz, and Gronwall's inequality we obtain

$$\begin{aligned}
&\int_0^1 \hat{u}(x,\bar{T})^2\,dx \\
&\leqslant \int_0^1 \mathcal{R}_{tb}^2\,dx + 2C_1T^{1/2}\sum_{i=1}^5(\int_0^T \mathcal{R}_{sbi}^2\,dt)^{1/2} + 2C_2\sum_{i=1}^5(\int_0^T \mathcal{R}_{sbi}^2\,dt) + \int_0^T\int_0^1 \mathcal{R}_{int}^2\,dxdt + 2C_3\int_0^{\bar{T}}\int_0^1 \hat{u}^2\,dxdt \\
&\leqslant (1+2C_3Te^{2C_3T})\Big(\int_0^1 \mathcal{R}_{tb}^2\,dx + 10C_1T^{1/2}(\sum_{i=1}^5\int_0^T \mathcal{R}_{sbi}^2\,dt)^{1/2} + 2C_2\sum_{i=1}^5(\int_0^T \mathcal{R}_{sbi}^2\,dt) + \int_0^T\int_0^1 \mathcal{R}_{int}^2\,dxdt\Big)
\end{aligned} \tag{3.15}$$

Then integrate (3.15) in time yields

$$\begin{aligned}
\varepsilon_G^2 := \int_0^T\int_0^1 \hat{u}(x,T)^2\,dxdT \leqslant (T+2C_3T^2e^{2C_3T})\Big(&\int_0^1 \mathcal{R}_{tb}^2\,dx + 10C_1T^{1/2}(\sum_{i=1}^5\int_0^T \mathcal{R}_{sbi}^2\,dt)^{1/2} \\
&+ 2C_2\sum_{i=1}^5(\int_0^T \mathcal{R}_{sbi}^2\,dt) + \int_0^T\int_0^1 \mathcal{R}_{int}^2\,dxdt\Big)
\end{aligned} \tag{3.16}$$

with

$$C_1 = \|u\|_{C_t^0C_x^4} + \|u^*\|_{C_t^0C_x^4}, \quad C_2 = \frac{1}{2}\|u\|_{C_t^0C_x^0} + \frac{1}{2}, \quad C_3 = \|u^*\|_{C_t^0C_x^1} + \frac{1}{2}\|u\|_{C_t^0C_x^1} + \frac{1}{2} \tag{3.17}$$

Finally, applying the estimates (3.8), (3.9), (3.10) on the quadrature error, and definition of training errors (3.7), yields the desired inequality (3.11).

□

3.4 Numerical experiments

3.4.1 Implementation

The PINNs algorithm 2.1 has been implemented within the PyTorch framework [23] and the code can be downloaded from **https://github.com/baigm11/DispersivePinns**. As is well documented [**?**, **?**, **?**], the coding and implementation of PINNs is extremely simple. Only a few lines of Python code suffice for this purpose. All the numerical experiments were performed on a single GeForce GTX1080 GPU.

The PINNs algorithm has the following hyperparameters, the number of hidden layers $K-1$, the width of each hidden layer $d_k := \bar{d}$ in (2.5), the specific activation function A, the parameter λ in the loss function (3.5), the regularization parameter λ_{reg} in the cumulative loss function (2.13) and the specific gradient descent algorithm for approximating the optimization problem (2.13). We use the hyperbolic tangent tanh activation function, thus ensuring that all the smoothness hypothesis on the resulting neural networks, as required in all bounds on generalization error below, are satisfied. Moreover, we use the second-order LBFGS method [8] as the optimizer. We follow the ensemble training procedure of [**?**] in order to choose the remaining hyperparameters. To this end, we consider a range of values, shown in Table 1, for the number of hidden layers, the depth of each hidden layer, the parameter λ and the regularization parameter λ_{reg}. For each configuration in the ensemble, the resulting model is retrained (in parallel) n_θ times with different random starting values of the trainable weights in the optimization algorithm and the one yielding the smallest value of the training loss is selected.

3.4.2 KdV equation

We look at single soliton and double soliton solution as the test cases of KdV equation. Fortunately we have solutions of closed-form for both cases [10]. In single soliton case, we have the following exact solution

$$u(x,t) = 9\big(1-\tanh^2(\sqrt{3/2}(x-3t))\big) \tag{3.18}$$

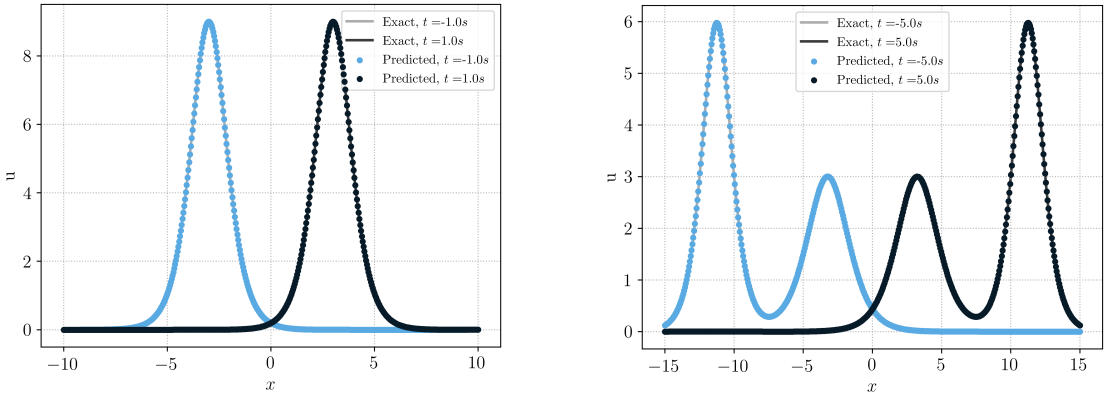
	$K-1$	d	q	λ_{reg}	λ	n_θ
KdV Equation	4, 8	20, 24, 28	2	0	0.1, 1, 10	5
Kawahara Equation	4, 8, 12	20, 24, 28, 32	2	0	0.1, 1, 10	5
CH Equation	4, 8	20, 24, 28	2	0	0.1, 1, 10	5
BO Equation, Single Soliton	4, 8, 12	20, 24, 28, 32	2	0	0.1, 1, 10	5
BO Equations, Double Soliton	4, 8	20, 24, 28	2	0	0.1, 1, 10	5

Table 1: Hyperparameter configurations employed in the ensemble training of PINNs.

which represents a single bump moving to the right with speed 3 with initial peak at $x = 0$. In double soliton case, we have the exact solution

$$u(x,t) = 6(b-a)\frac{b\,\mathrm{csch}^2(\sqrt{b/2}(x-2bt)) + a\,\mathrm{sech}^2(\sqrt{a/2}(x-2at))}{\left(\sqrt{a}\tanh(\sqrt{a/2}(x-2at)) - \sqrt{b}\tanh(\sqrt{b/2}(x-2bt))\right)^2} \tag{3.19}$$

for any real numbers a and b where we have used $a = 0.5$ and $b = 1$ in the numerical experiment. (3.19) represents two solitary waves that "collide" at $t = 0$ and separate for $t > 0$. For large $|t|$, $u(\cdot, t)$ is close to a sum of two solitary waves at different locations. The results of best performance among ensemble training are presented in Table 2 and figure 2. We use low-discrepancy Sobol points as training points. Training is relatively easy for KdV equation. We only need very few training points to get a very low relative error. The time for LBFGS optimizer to converge is around 1min and 10min for single and double soliton respectively which are very cheap. Our results outperform a lot what are reported in [6, 10] using convergent finite difference methods in the sense of relative error.



(a) Single soliton

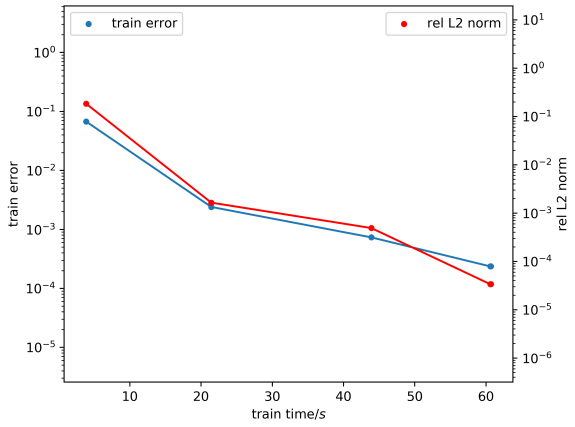
(b) Double soliton

Figure 2: Exact and PINN solution to single and double soliton test case of KdV equation

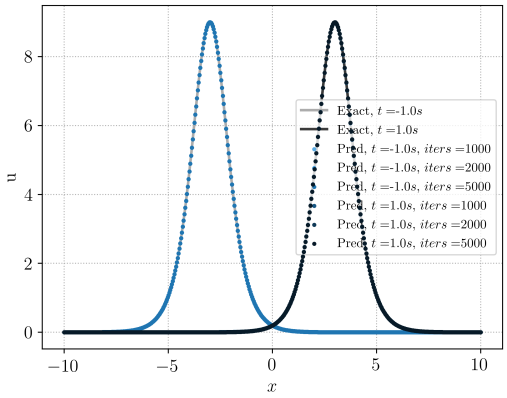
	N_{int}	N_{sb}	N_{tb}	$K-1$	d	λ	$\mathcal{E}_T$	$\mathcal{E}_G^r$
Single Soliton	2048	512	512	4	20	0.1	0.000236	0.00338%
Double Soliton	4096	1024	1024	4	32	1	0.000713	0.059%

Table 2: Best performing *Neural Network* configurations for the Single Soliton and Double Soliton problem. Low-discrepancy Sobol points are used for every reported numerical example.

The profiles and data at early iterations for KdV single and double soliton cases are summarized in figure 3, Table 3 and figure 4, Table 4 respectively. Since there is always a time boundary residual term,



(a) Single soliton



(b) Single soliton

Figure 3: (a) is the plot of train error and relative error versus train time; (b) is the plots at initial and final time for different train iterations

the initial plot of PINNs should be very closed to the exact initial data and what matters is the plot of PINNs at final time. We see that in both cases only a few iterations are enough to get a very satisfactory relative generalization error.

max_iters	**training time**/s	ε_T	ε_G^r
100	4	6.75e-02	1.84e-01
500	21	2.41e-03	1.65e-03
1000	44	7.34e-04	4.92e-04
2000	61	2.36e-04	3.38e-05

Table 3: Results for Single Soliton of KdV equation with different training iterations

max_iters	**training time**/s	ε_T	ε_G^r
100	9	1.21e-01	4.82e-01
500	48	2.60e-02	1.30e-01
1000	95	7.00e-03	4.32e-02
2000	159	2.54e-03	1.11e-02
5000	436	7.89e-04	6.50e-04
10000	499	7.13e-04	5.88e-04

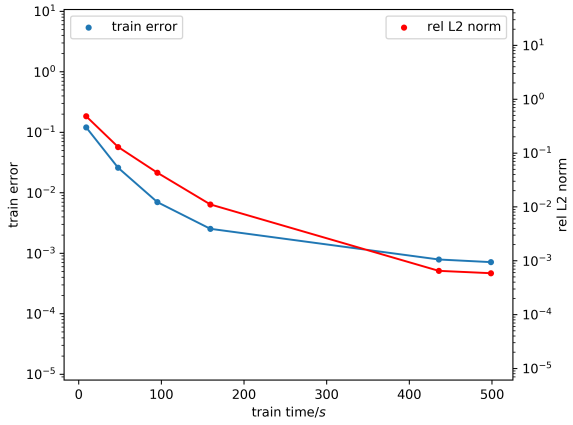
Table 4: Results for Double Soliton of KdV equation with different training iterations

3.4.3 Kawahara equation

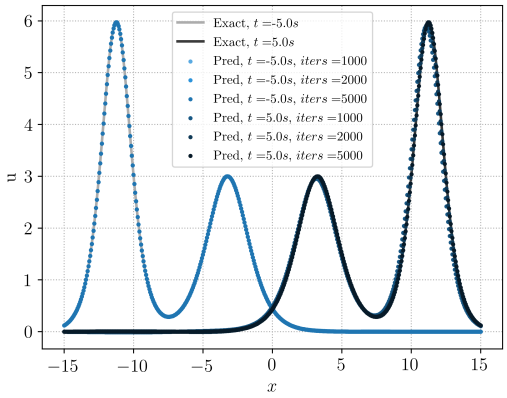
Following the numerical experiments in [3, 14, 15], we consider a Kawahara-type equation which differs from Kawahara equation in a first-order term u_x.

$$u_t + u_x + uu_x + u_{xxx} - u_{xxxxx} = 0 \tag{3.20}$$

This first-order term u_x is harmless and we can easily derive a similar a posterior bound on generalization error. Different from KdV equation, for this Kawahara-type equation we only have closed-form solution in

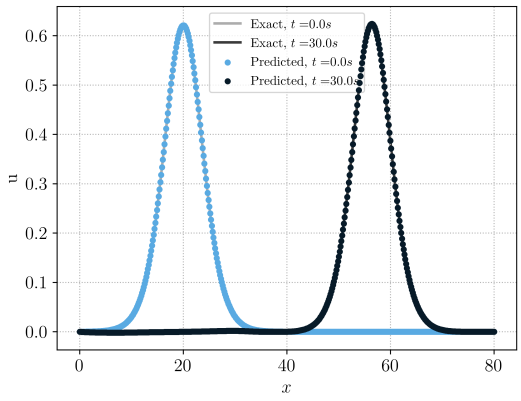


(a) Double soliton

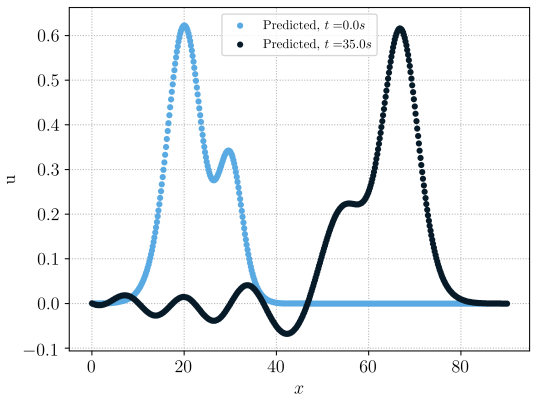


(b) Double soliton

Figure 4: (a) is the plot of train error and relative error versus train time; (b) is the plots at initial and final time for different train iterations



(a) Single soliton



(b) Double soliton

Figure 5: (a) is the plot of exact and PINN solution of single soliton test case of KdV equation; (b) is the PINN solution of additive initial data. The backgroud plot is the exact solution 3.21 of fast wave alone.

single soliton case. As before, we use low-discrepancy Sobol points as training points. For single soliton case, we have the exact solution

$$u(x,t) = \frac{105}{169}\text{sech}^4\Big(\frac{1}{2\sqrt{13}}(x - \frac{205}{169}t - x_0)\Big) \tag{3.21}$$

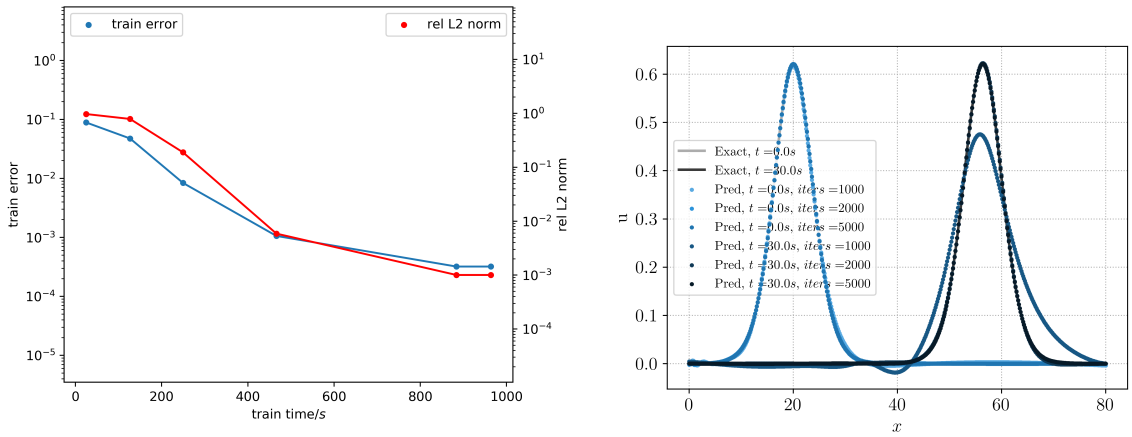
This represents a single bump moving to the right with speed $\frac{205}{169}$ with inital peak at $x = x_0$. For double soliton case, unfortunately there's no closed-form solution. Following the construction of initial data in [3] and [24], we'll use the following initial data

$$u(x,0) := u_1(x,0) + u_2(x,0) = \sum_{i=1}^{2} d_i^2 \text{sech}^4\Big(\frac{\sqrt{\mu d_i}}{4}(x - x_i)\Big), \quad \mu = \sqrt{\frac{16}{105}} \tag{3.22}$$

We choose $d_1 = \sqrt{\frac{105}{169}}$ and $d_2 = \sqrt{\frac{105}{338}}$ in our experiments. At $t = 0$, this initial data is the sum of two solitary waves that are located at x_1 and x_2. Notice that fast wave u_1 is exactly the single soliton solution (3.21) and slow wave u_2 is initially half height of u_1. We expect the profile of evolution is similar to double soliton as KdV equation. Since there's an advective nonlinear term in Kawahara equation, the additive

	N_{int}	N_{sb}	N_{tb}	$K-1$	d	λ	$\mathcal{E}_T$	$\mathcal{E}_G^r$
Single Soliton	2048	512	512	4	24	10	0.000321	0.101%
Double Soliton	16384	4096	4096	4	32	10	0.000665	no exact solution

Table 5: Best performing *Neural Network* configurations for the Single Soliton, Double Soliton, Wave Generation and Anti-wave Generation. Low-discrepancy Sobol points are used for every reported numerical example.



(a) Single soliton

(b) Single soliton

Figure 6: (a) is the plot of train error and relative error versus train time; (b) is the plots at initial and final time for different train iterations

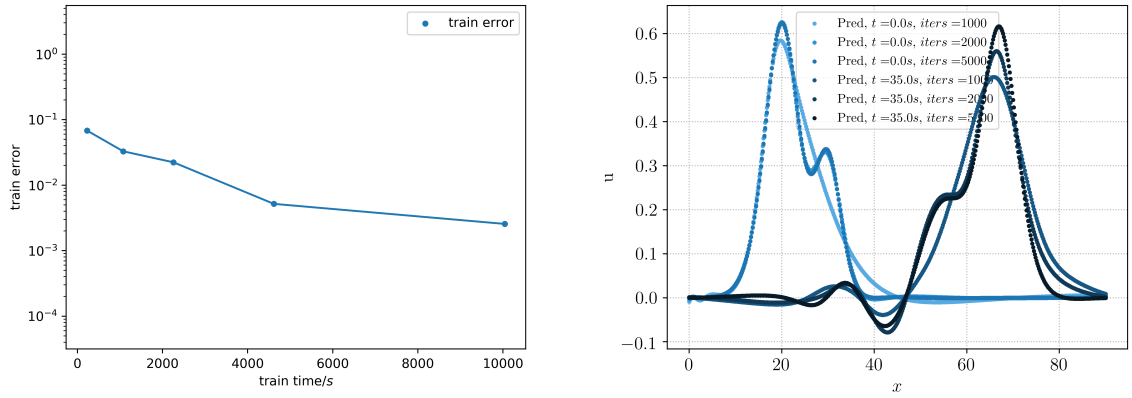
structure in initial data should undergo a complicated interaction through this nonlinear term. Indeed our PINN experiment, see figure 5, shows it's far from double soliton where peaks collide elastically. The height and shape of fast wave u_1 will be preserved while the slow wave u_2 will break down into smaller waves in the tail. Also the speed of fast wave u_1 is a little bit faster than the case where u_1 evolves alone as single soliton (3.21).

We summarize the results and plots in Table 5 and figure 5 respectively. Because time complexity grows exponential w.r.t. the order of PDEs and Kawahara equation is of 5-th order, it's computationally very consuming compared with KdV equation. In double soliton case, usually it takes several hours for LBFGS optimizer to converge. It's interesting to notice 15min is enough for LBFGS optimizer to converge in single soliton case with configurations in Table 5. Intuitively, the moving pattern of single soliton is very simple, thus PINN can be trained to capture this transitional pattern in a short time.

The profiles and data at early iterations for Kawahara single and double soliton cases are summarized in figure 6, Table 6 and figure 7, Table 7 respectively. And all other hyperparameters are the same as in the Table 5. We see that PINN for single soliton case is easy to train: $10min$ is enough to make generalization error ε_G^r less than 0.1%. Double soliton case is much more time consuming through, where a large part of time is used to resolve the dispersed peaks. From figure 7(b), we see in the final profiles the peaks are getting more resolved as the increase of iterations.

max_iters	**training time**/s	ε_T	ε_G^r
100	25	8.89e-02	9.70e-01
500	127	4.76e-02	7.86e-01
1000	249	8.40e-03	1.89e-01
2000	466	1.06e-03	5.88e-03
5000	964	3.21e-04	1.01e-03

Table 6: Results for Single Soliton of Kawahara equation with different training iterations



(a) Double soliton

(b) Double soliton

Figure 7: (a) is the plot of train error versus train time; (b) is the plots at initial and final time for different train iterations

max_iters	**training time**/s	ε_T
100	226	6.82e-02
500	1077	3.28e-02
1000	2255	2.23e-02
2000	4616	5.18e-03
5000	10043	2.56e-03

Table 7: Results for Double Soliton of Kawahara equation with different training iterations

4 Camassa-Holm equation

4.1 The underlying PDE

In this section, we consider the following initial-boundary value problem of the one-dimensional Camassa-Holm equation on a compact interval

$$\begin{aligned} u_t - u_{txx} + 3uu_x + 2\kappa u_x &= 2u_x u_{xx} + uu_{xxx}, \quad \forall x \in (0,1),\ t \in [0,T], \\ u(x,0) &= u_0(x), \quad \forall x \in (0,1). \\ u(0,t) = u_{xx}(0,t) &= u(1,t) = u_{xx}(1,t) = 0, \quad \forall t \in [0,T]. \end{aligned} \tag{4.1}$$

Here, κ is a real constant. This equation models the unidirectional propagation of shallow water waves over a flat bottom, with u representing the fluid velocity. The most compelling feature of the above equation is that it is completely integrable for all values of κ. Much of the attention in literature has been given to the special case $\kappa = 0$, which plays an important role in the modeling of nonlinear dispersive waves in hyperelastic rods [2]. Regarding the existence of solutions, we report following result which is in the spirit of Kwek et. al. [16], but slightly differs and is reminiscent of the results from [16].

Theorem 4.1. *Let* $\mathcal{X} := \{u \in H^4(0,1) : u(0) = u_{xx}(0) = u(1) = u_{xx}(1) = 0\}$. *Then for every* $u_0 \in \mathcal{X}$, *the problem* (4.1) *has a unique solution*

$$u \in C([0,T);\mathcal{X}) \cap C^1([0,T);H_0^1(0,1)),$$

for some $T > 0$. *In addition,* $u_{xx} \in C^1([0,T);H_0^1(0,1))$, *and* u *depends continuously on* u_0 *in the* H^4*-norm.*

4.2 PINNs

We first specify the training set $\mathcal{S}$, and define appropriate residuals to run the algorithm 2.1. In what follows, we begin with the description of the training set $\mathcal{S}$.

4.2.1 Training Set

Let us define the space-time domain $\Omega_T = (0,1) \times (0,T)$, and divide the training set $\mathcal{S} = \mathcal{S}_{int} \cup \mathcal{S}_{sb} \cup \mathcal{S}_{tb}$ of the abstract PINNs algorithm 2.1 into the following three subsets,

(a) Interior training points $\mathcal{S}_{int} = \{y_n\}$ for $1 \leqslant n \leqslant N_{int}$, with each $y_n = (x_n, t_n) \in \Omega_T$. We use low-discrepancy Sobol points as training points.

(b) Spatial boundary training points $\mathcal{S}_{sb} = (0, t_n) \cup (1, t_n)$ for $1 \leqslant n \leqslant N_{sb}$, and the points t_n chosen as low discrepancy Sobol points.

(c) Temporal boundary training points $\mathcal{S}_{tb} = \{x_n\}$, with $1 \leqslant n \leqslant N_{tb}$ and each $x_n \in (0,1)$, chosen as low-discrepancy Sobol points.

4.2.2 Residuals

- Interior Residual given by,

$$\begin{aligned} \mathcal{R}_{int,\theta}(x,t) := \partial_t u_\theta(x,t) - \partial_{txx} u_\theta(x,t) + 3u_\theta(x,t)(u_\theta)_x(x,t) + 2\kappa(u_\theta)_x(x,t) \\ - 2(u_\theta)_x(x,t)(u_\theta)_{xx}(x,t) - (u_\theta)(x,t)(u_\theta)_{xxx}(x,t). \end{aligned} \tag{4.2}$$

  Note that the residual is well defined and $\mathcal{R}_{int,\theta} \in C([0,T] \times [0,1])$ for every $\theta \in \Theta$.

- Spatial boundary Residual given by,

$$\begin{aligned} \mathcal{R}_{sb1,\theta}(0,t) &:= u_\theta(0,t), \quad \forall t \in (0,T), \\ \mathcal{R}_{sb2,\theta}(1,t) &:= u_\theta(1,t), \quad \forall t \in (0,T), \\ \mathcal{R}_{sb3,\theta}(0,t) &:= (u_\theta)_{xx}(0,t), \quad \forall t \in (0,T) \\ \mathcal{R}_{sb4,\theta}(1,t) &:= (u_\theta)_{xx}(1,t), \quad \forall t \in (0,T). \end{aligned} \tag{4.3}$$

  Given the fact that the neural network is smooth, this residual is well defined.

- Temporal boundary Residual given by,

$$\mathcal{R}_{tb,\theta}(x) := \left[\Big(u_\theta(x,0) - u_0(x) \Big)^2 + \Big((u_\theta)_x(x,0) - (u_0)_x(x) \Big)^2 \right]^{1/2}, \quad \forall x \in (0,1). \tag{4.4}$$

  Again this quantity is well-defined and $\mathcal{R}_{tb,\theta} \in C^2((0,1))$ as both the initial data and the neural network are smooth.

4.2.3 Loss function

We use the following loss function to train the PINN for approximating the Camassa-Holm equation (4.1),

$$J(\theta) := \sum_{n=1}^{N_{tb}} w_n^{tb} |\mathcal{R}_{tb,\theta}(x_n)|^2 + \sum_{n=1}^{N_{sb}} \sum_{i=1}^{4} w_n^{sb} |\mathcal{R}_{sbi,\theta}(t_n)|^2 + \lambda \sum_{n=1}^{N_{int}} w_n^{int} |\mathcal{R}_{int,\theta}(x_n, t_n)|^2. \tag{4.5}$$

Here w_n^{tb} are the N_{tb} quadrature weights corresponding to the temporal boundary training points $\mathcal{S}_{tb}$, w_n^{sb} are the N_{sb} quadrature weights corresponding to the spatial boundary training points $\mathcal{S}_{sb}$ and w_n^{int} are the N_{int} quadrature weights corresponding to the interior training points $\mathcal{S}_{int}$. Furthermore, λ is a hyperparameter for balancing the residuals, on account of the PDE and the initial and boundary data, respectively.

4.3 Estimate on the generalization error.

As for the semilinear parabolic equation, we will try to estimate the following generalization error for the PINN $u^* = u_{\theta^*}$, generated through algorithm 2.1, with loss functions (2.13), (4.5), for approximating the solution of the Camassa-Holm equation (4.1):

$$\mathcal{E}_G := \left(\int_0^T \int_0^1 |u(x,t) - u^*(x,t)|^2 dx dt \right)^{\frac{1}{2}}. \tag{4.6}$$

This generalization error will be estimated in terms of the *training error*,

$$\mathcal{E}_T^2 := \lambda \underbrace{\sum_{n=1}^{N_{int}} w_n^{int} |\mathcal{R}_{int,\theta^*}(x_n,t_n)|^2}_{(\mathcal{E}_T^{int})^2} + \underbrace{\sum_{n=1}^{N_{tb}} + w_n^{tb} |\mathcal{R}_{tb,\theta^*}(x_n)|^2}_{(\mathcal{E}_T^{tb})^2} + \underbrace{\sum_{n=1}^{N_{sb}} \sum_{i=1}^{4} w_n^{sb} |\mathcal{R}_{sbi,\theta^*}(t_n)|^2}_{(\mathcal{E}_T^{sb})^2}, \tag{4.7}$$

readily computed from the training loss (4.5) *a posteriori*. We have the following estimate,

Theorem 4.2. *Let $\kappa > 0$ and let $u \in C^3((0,T)\times(0,1))$ be the unique classical solution of Casamma-Holm equation (4.1). Let $u^* = u_{\theta^*}$ be the PINN, generated by algorithm 2.1, with loss function (4.5). Then, the generalization error (4.6) is bounded by,*

$$\begin{aligned} \varepsilon_G \leqslant\ & C_1\big(\varepsilon_T^{tb} + \varepsilon_T^{int} + C_2(\varepsilon_T^{sb}) + C_3(\varepsilon_T^{sb})^{1/2} \\ & + (C_{quad}^{tb})^{1/2} N_{tb}^{-\alpha_{tb}/2} + (C_{quad}^{int})^{1/2} N_{int}^{-\alpha_{int}/2} + C_2 (C_{quad}^{sb})^{1/2} N_{sb}^{-\alpha_{sb}/2} + C_3 (C_{quad}^{sb})^{1/4} N_{sb}^{-\alpha_{sb}/4}\big) \end{aligned} \tag{4.8}$$

where

$$\begin{aligned} C_1 &= \sqrt{T + 2C_4 T^2 e^{2C_4 T}} \\ C_2 &= \sqrt{2(|\kappa| + \|u^*\|_{C_t^0 C_x^2} + \|u\|_{C_t^0 C_x^2})} \\ C_3 &= 2T^{1/4} \sqrt{2\|u^*\|_{C_t^1 C_x^1} + 2\|u\|_{C_t^1 C_x^1} + 2\|u\|_{C_t^0 C_x^1} (\|u^*\|_{C_t^0 C_x^1} + \|u\|_{C_t^0 C_x^1})} \\ C_4 &= \frac{1}{2} + 3\|u^*\|_{C_t^0 C_x^1} + \frac{3}{2}\|u\|_{C_t^0 C_x^3}, \end{aligned} \tag{4.9}$$

and $C_{quad}^{tb} = C_{qaud}^{tb}\left(\|\mathcal{R}_{tb,\theta^*}\|_{C^2}\right)$, $C_{quad}^{int} = C_{qaud}^{int}\left(\|\mathcal{R}_{int,\theta^*}\|_{C^0}\right)$, and $C_{quad}^{sb} = C_{qaud}^{sb}\left(\|\mathcal{R}_{sb,\theta^*}\|_{C^1}\right)$ are the constants associated with the quadrature errors are constants are appear in the bounds on quadrature error (3.8)-(3.10).

Proof. Let $\hat{u} = u^* - u$ be the error with the PINN. From the PDE (4.1) and the definition of the interior residual (4.2), we have the following identities,

$$\hat{u}_t - \hat{u}_{txx} + 2\kappa\hat{u}_x + 3(u^* u_x^* - u u_x) = 2u_x^* u_{xx}^* - 2u_x u_{xx} + u^* u_{xxx}^* - u u_{xxx} + \mathcal{R}_{int} \tag{4.10}$$

Observe that

$$\begin{aligned} u^* u_x^* - u u_x = \hat{u}\hat{u}_x + u\hat{u}_x + \hat{u}u_x; \quad u_x^* u_{xx}^* - u_x u_{xx} = \hat{u}_x\hat{u}_{xx} + u_x\hat{u}_{xx} + \hat{u}_x u_{xx}, \\ u^* u_{xxx}^* - u u_{xxx} = \hat{u}\hat{u}_{xxx} + u\hat{u}_{xxx} + \hat{u}u_{xxx} \end{aligned}$$

Then multiply both side of (4.10) with $\hat{u}$, integrate by part and use the identity (**??**)

$$\frac{1}{2}\frac{d}{dt}\int_0^1 (\hat{u}^2 + (\hat{u}_x)^2)\, dx + \kappa\hat{u}^2\big|_0^1 - \hat{u}\hat{u}_{tx}\big|_0^1 = -3\mathcal{A} + 2\mathcal{B} + \mathcal{C} + \int_0^1 \hat{u} R_{int}\, dx \tag{4.11}$$

where

$$\begin{aligned} \mathcal{A} &:= \int_0^1 \hat{u}(\hat{u}\hat{u}_x + u\hat{u}_x + \hat{u}u_x)\, dx = \int_0^1 (\hat{u}_x + u_x)\hat{u}^2\, dx + \int_0^1 \hat{u}u\hat{u}_x\, dx \\ &= \int_0^1 (\hat{u}_x + u_x)\hat{u}^2\, dx - \frac{1}{2}\int_0^1 u_x\hat{u}^2\, dx + \frac{1}{2}u\hat{u}^2\Big|_0^1 = \int_0^1 (\hat{u}_x + \frac{1}{2}u_x)\hat{u}^2\, dx = \int_0^1 (u_x^* - \frac{1}{2}u_x)\hat{u}^2\, dx \end{aligned} \tag{4.12}$$

We estimate $\mathcal{B}$ as follow

$$\begin{aligned}\mathcal{B} &:= \int_0^1 \hat{u}(\hat{u}_x\hat{u}_{xx} + u_x\hat{u}_{xx} + \hat{u}_x u_{xx})\,dx = -\frac{1}{2}\int_0^1 \hat{u}_{xxx}\hat{u}^2\,dx + \frac{1}{2}\hat{u}^2\hat{u}_{xx}\Big|_0^1 \\ &- \int_0^1 u_x\hat{u}_x^2\,dx + \frac{1}{2}\int_0^1 u_{xxx}\hat{u}^2\,dx - \frac{1}{2}u_{xx}\hat{u}^2\Big|_0^1 + \hat{u}u_x\hat{u}_x|_0^1 - \frac{1}{2}\int_0^1 u_{xxx}\hat{u}^2\,dx + \frac{1}{2}\hat{u}^2u_{xx}\Big|_0^1 \\ &= -\frac{1}{2}\int_0^1 \hat{u}_{xxx}\hat{u}^2\,dx - \int_0^1 u_x\hat{u}_x^2\,dx + \frac{1}{2}\hat{u}^2\hat{u}_{xx}\Big|_0^1 + \hat{u}u_x\hat{u}_x|_0^1\end{aligned} \tag{4.13}$$

where we integrate the second term on the right of first line by part twice. Finally, we estimate $\mathcal{C}$ as follow

$$\begin{aligned}\mathcal{C} &:= \int_0^1 \hat{u}(\hat{u}\hat{u}_{xxx} + u\hat{u}_{xxx} + \hat{u}u_{xxx})\,dx = \int_0^1 (u_{xxx} + \hat{u}_{xxx})\hat{u}^2\,dx + \int_0^1 \hat{u}u\hat{u}_{xxx}\,dx \\ &= \int_0^1 (u_{xxx} + \hat{u}_{xxx})\hat{u}^2\,dx + \frac{3}{2}\int_0^1 u_x\hat{u}_x^2\,dx - \frac{1}{2}\int_0^1 u_{xxx}\hat{u}^2\,dx - \hat{u}u_x\hat{u}_x|_0^1 \\ &= \int_0^1 (u^*_{xxx} - \frac{1}{2}u_{xxx})\hat{u}^2\,dx + \frac{3}{2}\int_0^1 u_x\hat{u}_x^2\,dx - \hat{u}u_x\hat{u}_x|_0^1\end{aligned} \tag{4.14}$$

We integrant the second term on the right of first line by part four times, and discard three vanishing zero boudnary term. For the boundary term in (4.11), we have

$$|\,\hat{u}\hat{u}_{tx}|_0^1\,| \leqslant (\|u^*\|_{C_t^1C_x^1} + \|u\|_{C_t^1C_x^1})(|\mathcal{R}_{sb1}| + |\mathcal{R}_{sb2}|) \tag{4.15}$$

From (4.11)-(4.15), we get

$$\begin{aligned}\frac{1}{2}\frac{d}{dt}\int_0^1 (\hat{u}^2 + (\hat{u}_x)^2)\,dx &= -\kappa\hat{u}^2\big|_0^1 + \hat{u}\hat{u}_{tx}|_0^1 - 3\mathcal{A} + 2\mathcal{B} + \mathcal{C} + \int_0^1 \hat{u}\mathcal{R}_{int}\,dx \\ &= \int_0^1 (-3u^*_x + \frac{3}{2}u_x + \frac{1}{2}u_{xxx})\hat{u}^2\,dx - \frac{1}{2}\int_0^1 u_x\hat{u}_x^2\,dx + \int_0^1 \hat{u}\mathcal{R}_{int}\,dx \\ &- \kappa\hat{u}^2\big|_0^1 + \hat{u}\hat{u}_{tx}|_0^1 + \hat{u}^2\hat{u}_{xx}\big|_0^1 + \hat{u}u_x\hat{u}_x|_0^1 \\ &\leqslant (\frac{1}{2} + 3\|u^*\|_{C_t^0C_x^1} + \frac{3}{2}\|u\|_{C_t^0C_x^3})\int_0^1 \hat{u}^2\,dx + \frac{1}{2}\|u\|_{C_t^0C_x^1}\int_0^1 \hat{u}_x^2\,dx \\ &+ (|\kappa| + \|u^*\|_{C_t^0C_x^2} + \|u\|_{C_t^0C_x^2})(\mathcal{R}^2_{sb1} + \mathcal{R}^2_{sb2}) + \frac{1}{2}\int_0^1 \mathcal{R}^2_{int}\,dx \\ &+ \big(\|u^*\|_{C_t^1C_x^1} + \|u\|_{C_t^1C_x^1} + \|u\|_{C_t^0C_x^1}(\|u^*\|_{C_t^0C_x^1} + \|u\|_{C_t^0C_x^1})\big)(|\mathcal{R}_{sb1}| + |\mathcal{R}_{sb2}|) \\ &=: C_1\sum_{i=1}^4 |\mathcal{R}_{sbi}| + C_2\sum_{i=1}^4 \mathcal{R}^2_{sb,i} + \frac{1}{2}\int_0^1 \mathcal{R}^2_{int}\,dx + C_3\int_0^1 (\hat{u}^2 + \hat{u}_x^2)\,dx\end{aligned} \tag{4.16}$$

Then integrating the above inequality over $[0,\bar{T}]$ for any $\bar{T} \leqslant T$, we obtain

$$\begin{aligned}&\int_0^1 (\hat{u}^2 + \hat{u}_x^2)(x,\bar{T})\,dx \leqslant \int_0^1 \mathcal{R}^2_{tb}\,dx \\ &+ 2C_1T^{1/2}\sum_{i=1}^4 (\int_0^T \mathcal{R}^2_{sbi}\,dt)^{1/2} + 2C_2\sum_{i=1}^4 (\int_0^T \mathcal{R}^2_{sbi}\,dt) + \int_0^T\int_0^1 \mathcal{R}^2_{int}\,dxdt + 2C_3\int_0^T\int_0^1 (\hat{u}^2 + \hat{u}_x^2)\,dxdt \\ &\leqslant (1 + 2C_3Te^{2C_3T})\Big(\int_0^1 \mathcal{R}^2_{tb}\,dx + 8C_1T^{1/2}(\sum_{i=1}^4\int_0^T \mathcal{R}^2_{sbi}\,dt)^{1/2} + 2C_2\sum_{i=1}^4 (\int_0^T \mathcal{R}^2_{sbi}\,dt) + \int_0^T\int_0^1 \mathcal{R}^2_{int}\,dxdt\Big)\end{aligned} \tag{4.17}$$

We can now use Cauchy-Schwarz in the first inequality and Gronwall's inequality. Then integrate (4.17)

over $\bar{T} \in [0, T]$

$$\varepsilon_G^2 := \int_0^T \int_0^1 \hat{u}(x,\bar{T})^2 \, dx d\bar{T} \leqslant \int_0^T \int_0^1 (\hat{u}^2 + \hat{u}_x^2)(x,\bar{T}) \, dx d\bar{T}$$
$$\leqslant (T + 2C_3 T^2 e^{2C_3 T})\Big(\int_0^1 \mathcal{R}_{tb}^2 \, dx + 8C_1 T^{1/2} \Big(\sum_{i=1}^4 \int_0^T \mathcal{R}_{sbi}^2 \, dt\Big)^{1/2} + 2C_2 \sum_{i=1}^4 \Big(\int_0^T \mathcal{R}_{sbi}^2 \, dt\Big) + \int_0^T \int_0^1 \mathcal{R}_{int}^2 \, dx dt \Big) \tag{4.18}$$

with

$$\begin{aligned} C_1 &= \|u^*\|_{C_t^1 C_x^1} + \|u\|_{C_t^1 C_x^1} + \|u\|_{C_t^0 C_x^1} (\|u^*\|_{C_t^0 C_x^1} + \|u\|_{C_t^0 C_x^1}) \\ C_2 &= |\kappa| + \|u^*\|_{C_t^0 C_x^2} + \|u\|_{C_t^0 C_x^2}, \quad C_3 = \frac{1}{2} + 3\|u^*\|_{C_t^0 C_x^1} + \frac{3}{2}\|u\|_{C_t^0 C_x^3} \end{aligned} \tag{4.19}$$

Finally, applying the estimates (3.8), (3.9), (3.10) on the quadrature error, and definition of training errors (4.7), yields the desired inequality (4.8).

□

4.4 Peakon limit

Standard test cases for CH equation($\kappa = 0$) are single and double peakon as in [11]. Peakon differs from soliton in that peakon is singular at its peak. The standard closed-form solution for single peakon is

$$u(x,t) = ce^{|x-ct|} \tag{4.20}$$

and for double peakon is

$$u(x,t) = c_1 e^{|x-c_1 t|} + c_2 e^{|x-c_2 t|} \tag{4.21}$$

Obviously these solutions are at most H_x^1 regular due to the singularities at their peaks. To apply theorem 4.2, we need $u \in C_x^3$ at least. Thus we don't expect a smooth NN can approximate H_x^1 functions nicely. In fact, our experiments also give a negative result on these peakon cases.

Nevertheless, thanks to the so-called peakon limit procedure proposed by Allen Parker in [20–22], we can test our PINN algorithm on generalized CH($\kappa \neq 0$) which does have smooth analytic single and double soliton solution. The linear term $2\kappa u_x$ resolves the singularity of test case (4.20) and (4.21) at the peaks. For convenience, we write the parameter κ as k^2. With parameter k, the exact solution of single soliton is given in [20] by

$$\begin{aligned} u(\theta) &= \frac{2kcp^2}{(1+k^2p^2) + (1-k^2p^2)\cosh\theta} \\ \Theta &= p(x - \tilde{c}t + x_0) \\ \Theta &= \frac{\theta}{k} + p \ln \frac{(1+kp) + (1-kp)e^{\theta}}{(1-kp) + (1+kp)e^{\theta}} \end{aligned} \tag{4.22}$$

where $\tilde{c} = \frac{c}{k} = \frac{2k^2}{1-k^2p^2}$ and p is an additional parameter. To get the exact solution, we need to take the inverse of $\Theta(\theta)$. $\Theta(\theta)$ is invertible if and only if $0 < kp < 1$ which is an additional restraint when choosing p. It's easy to observe that the solution moves to the right with speed $\tilde{c}$ and preserves the shape meanwhile. Moreover when $k \to 0$ and $kp \to 1$, the single soliton wave tends to single peakon wave (4.20) pointwisely and this limiting procedure is termed peakon limit.

For double soliton, similar to the single soliton case, we need additional parameters p_1, p_2. Follow the procdure in [21], we define

$$c_i = \frac{2k^3}{1 - k^2 p_i^2}, \quad i = 1, 2 \tag{4.23}$$

and

$$w_i = -p_i c_i, \quad i = 1, 2 \tag{4.24}$$

and

$$A_{12} = \frac{(p_1 - p_2)^2}{(p_1 + p_2)^2} \tag{4.25}$$

for $i = 1, 2$, define

$$\begin{aligned} a_i &= 1 + kp_i \\ b_i &= 1 - kp_i \end{aligned} \tag{4.26}$$

as before, we define θ_i w.r.t. y as

$$\theta_i = p_i(y - c_i t + \alpha_i), \quad i = 1, 2 \tag{4.27}$$

and

$$\begin{aligned} v_{12} &= \frac{4k^3(p_1 - p_2)^2}{(1 - k^2p_1^2)(1 - k^2k_2^2)} \\ b_{12} &= \frac{8k^6(p_1 - p_2)^2(1 - k^4p_1^2p_2^2)}{(1 - k^2p_1^2)^2(1 - k^2p_2^2)^2} \end{aligned} \tag{4.28}$$

then the exact double soliton solution w.r.t. y is given by

$$u(y,t) = k^2 + \frac{2}{k}\frac{w_1^2e^{\theta_1} + w_2^2e^{\theta_2} + b_{12}e^{\theta_1+\theta_2} + A_{12}(w_1^2e^{\theta_1+2\theta_2} + w_2^2e^{2\theta_1+\theta_2})}{rf^2} \tag{4.29}$$

where

$$\begin{aligned} f(y,t) &= 1 + e^{\theta_1} + e^{\theta_2} + A_{12}e^{\theta_1+\theta_2} \\ r(y,t) &= k + \frac{2}{f^2}(c_1p_1^2e^{\theta_1} + c_2p_2^2e^{\theta_2} + v_{12}e^{\theta_1+\theta_2} + A_{12}(c_1p_1^2e^{\theta_1+2\theta 2} + c_2p_2^2e^{2\theta_1+\theta 2})) \end{aligned} \tag{4.30}$$

Finally we have the following relation between x and y

$$x(y,t) = \frac{y}{k} + \ln\frac{a_1a_2 + b_1a_2e^{\theta 1} + b_2a_1e^{\theta 2} + b_1b_2A_{12}e^{\theta_1+\theta_2}}{b_1b_2 + a_1b_2e^{\theta 1} + q_2b_1e^{\theta 2} + a_1a_2A_{12}e^{\theta_1+\theta_2}} + k^2t + \alpha \tag{4.31}$$

where α is the phase parameter. To get the exact solution, we need to take the inverse of $x(y,t)$ w.r.t. y at the training points. $x(y,t)$ is invertible w.r.t. y if and only if $0 < kp_i < 1, i = 1, 2$ which, again, is an additional restraint when choosing p_1, p_2. When $k \to 0$ and $kp_i \to 1, i = 1, 2$, the double soliton wave tends to double peakon wave (4.21) pointwisely.

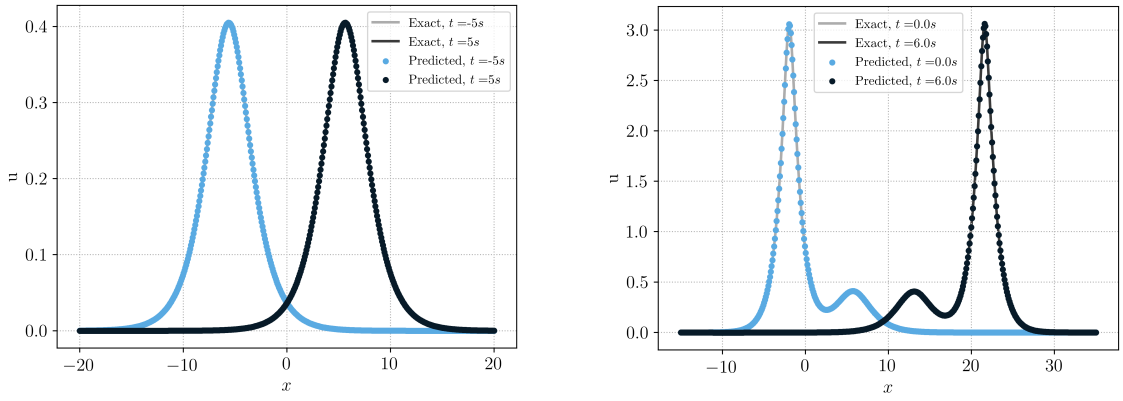
4.5 Numerical experiments

For single soliton case of generalized CH, we use parameters $k = 0.6, p = 1$, and $k = 0, 6, p_1 = 1.5, p_2 = 2$ for double soliton case. Similar to the previous equations, PINN for single soliton is easier to train because of the simple transitional moving pattern. See Table 8 for best performance and figure 8 for corespondent plots. From figure 8(b), we observe a sharp peak of fast soliton and PINN still succeeds to resolve large derivatives there.

	N_{int}	N_{sb}	N_{tb}	$K-1$	d	λ	$\mathcal{E}_T$	$\mathcal{E}_G^r$
Single Soliton	16384	4096	4096	4	20	1	3.70e-06	0.00191%
Double Soliton	16384	4096	4096	8	24	0.1	0.00127	0.186%

Table 8: Best performing *Neural Network* configurations for the Single Soliton and Double Soliton problem. Low-discrepancy Sobol points are used for every reported numerical example.

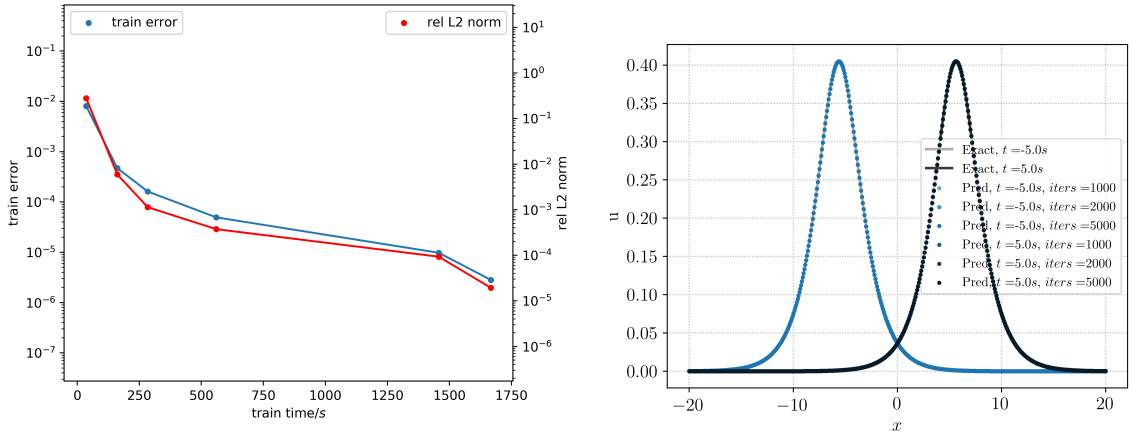
The profiles and data at early iterations for Camassa-Holm single and double soliton cases are summarized in figure 9, Table 9 and figure 10, Table 10 respectively. And all other hyperparameters are the same as in the Table 8. And we use parameter $(k, kp) = (0.6, 1)$ and $(k, kp_1, kp_2) = (0.6, 0.9, 0.6)$ for single and double soliton respectively. As expected, single soliton case only requires $10min$ to be trained very well and double soliton needs around $1h$ to get a comparable relative generalization error.



(a) Single soliton, $k = 0.6, p = 1$

(b) Double soliton, $k = 0.6, p_1 = 1.5, p_2 = 1$

Figure 8: Exact and PINN solution to single and double soliton test case of generalized CH equation



(a) Single soliton

(b) Single soliton

Figure 9: (a) is the plot of train error and relative error versus train time; (b) is the plots at initial and final time for different train iterations

5 Benjamin-Ono Equation

5.1 The underlying PDE

In this section, we consider the following Benjamin-Ono (BO) equation

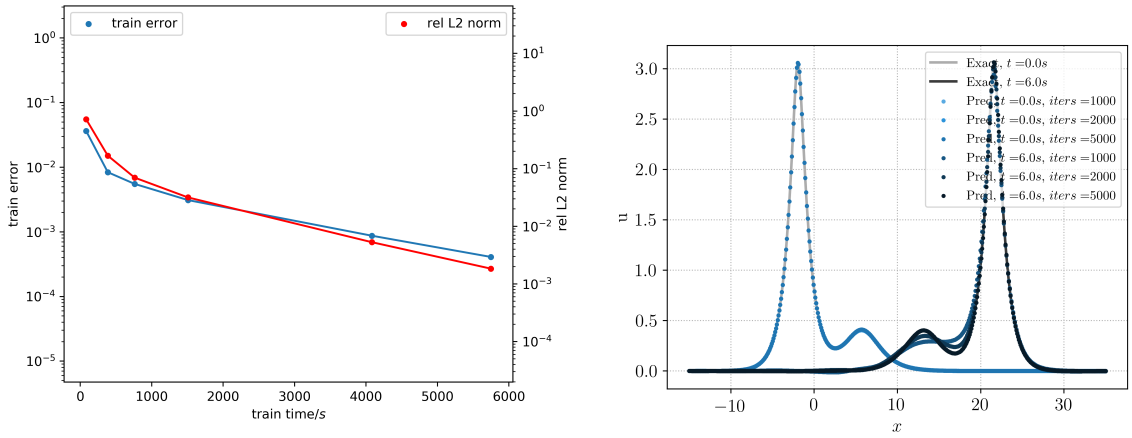
$$\begin{aligned} u_t + uu_x + Hu_{xx} &= 0, \quad x \in \mathbb{R}, \quad t > 0, \\ u(x,0) &= u_0(x), \quad x \in \mathbb{R}, \\ u(x,t) &= u(x+1,t), \quad x \in \mathbb{R}, \quad t > 0. \end{aligned} \tag{5.1}$$

The BO equation was first deduced by Benjamin [1] and Ono [19] as an approximate model for long-crested unidirectional waves at the interface of a two-layer system of incompressible inviscid fluids, one being infinitely deep. Later, it was shown to be a completely integrable system. In the periodic setting, Molinet [18] proved well-posedness in $H^s(\mathbb{T})$ for $s \geq 0$. For operator splitting methods applied to the BO equation, see [5]. Since our analysis heavily depends on the smoothness of the solutions, regarding the existence of smooth solutions, we use the following existence result for BO equation.

Theorem 5.1. *For any $s > 5/3$, let $u_0 \in H^s(0,1)$. Then there exists a global smooth solution to* (5.1)

max_iters	training time/s	ε_T	ε_G^r
100	36	8.08e-03	2.81e-01
500	161	4.71e-04	5.93e-03
1000	284	1.61e-04	1.14e-03
2000	560	4.96e-05	3.75e-04
5000	1457	9.77e-06	9.31e-05
10000	1667	2.83e-06	1.94e-05

Table 9: Results for single Soliton of CH equation with different training iterations



(a) Double soliton

(b) Double soliton

Figure 10: (a) is the plot of train error and relative error versus train time; (b) is the plots at initial and final time for different train iterations

max_iters	training time/s	ε_T	ε_G^r
100	83	3.63e-02	7.19e-01
500	386	8.37e-03	1.68e-01
1000	762	5.52e-03	6.99e-02
2000	1508	3.10e-03	3.17e-02
5000	4083	8.71e-04	5.29e-03
10000	5747	4.09e-04	1.84e-03

Table 10: Results for double soliton of CH equation with different training iterations

such that

$$u \in C(0,T;H^s(0,1)), \quad u_t \in C^1(0,T;H^{s-2}(0,1)).$$

Note that the above result was also used by Kenig, Ponce and Vega [13] to prove uniqueness properties of BO equation. Moreover, the above result ensures that the solutions satisfy the equation (5.1) pointwise for sufficiently smooth initial data.

5.2 PINNs

We first specify the training set $\mathcal{S}$, and define appropriate residuals to run the algorithm 2.1. In what follows, we begin with the description of the training set $\mathcal{S}$.

5.2.1 Training Set.

Let us define the space-time domain $\Omega_T = (0,1) \times (0,T)$, and divide the training set $\mathcal{S} = \mathcal{S}_{int} \cup \mathcal{S}_{sb} \cup \mathcal{S}_{tb}$ of the abstract PINNs algorithm 2.1 into the following three subsets,

(a) Interior training points $\mathcal{S}_{int} = \{y_n\}$ for $1 \leqslant n \leqslant N_{int}$, with each $y_n = (x_n, t_n) \in \Omega_T$. We use low-discrepancy Sobol points as training points.

(b) Spatial boundary training points $\mathcal{S}_{sb} = (0,t_n) \cup (1,t_n)$ for $1 \leqslant n \leqslant N_{sb}$, and the points t_n chosen as low discrepancy Sobol points.

(c) Temporal boundary training points $\mathcal{S}_{tb} = \{x_n\}$, with $1 \leqslant n \leqslant N_{tb}$ and each $x_n \in (0,1)$, chosen as low-discrepancy Sobol points.

5.2.2 Residuals

We define the residual $\mathcal{R}$ in algorithm 2.1, consisting of the following parts,

- *Interior residual* given by,

$$\mathcal{R}_{int,\theta}(x,t) := (u_\theta)_t(x,t) + u_\theta(x,t)(u_\theta)_x(x,t) + H(u_\theta)_{xx}(x,t), \quad (x,t) \in (0,1) \times (0,T), \tag{5.2}$$

- *Spatial boundary Residual* given by,

$$\mathcal{R}_{sb,\theta}(x,t) := u_\theta(x,t) - u_\theta(x+1,t), \quad \forall x \in \mathbb{R},\ t \in (0,T]. \tag{5.3}$$

- *Temporal boundary Residual* given by,

$$\mathcal{R}_{tb,\theta}(x) := u_\theta(x,0) - u_0(x), \quad \forall x \in (0,1). \tag{5.4}$$

As the underlying neural networks have the required regularity, the residuals are well-defined.

5.2.3 Loss function

We consider the following loss function for training PINNs to approximate the BO equation (5.1),

$$J(\theta) := \sum_{n=1}^{N_{tb}} w_n^{tb} |\mathcal{R}_{tb,\theta}(x_n)|^2 + \sum_{n=1}^{N_{sb}} w_n^{sb} |\mathcal{R}_{sb,\theta}(x_n,t_n)|^2 + \lambda \sum_{n=1}^{N_{int}} w_n^{int} |\mathcal{R}_{int,\theta}(x_n,t_n)|^2. \tag{5.5}$$

Here the residuals are defined by (5.2)-(5.4). w_n^{tb} are the N_{tb} quadrature weights corresponding to the temporal boundary training points $\mathcal{S}_{tb}$, w_n^{sb} are the N_{sb} quadrature weights corresponding to the spatial boundary training points $\mathcal{S}_{sb}$ and w_n^{int} are the N_{int} quadrature weights corresponding to the interior training points $\mathcal{S}_{int}$. Furthermore, λ is a hyperparameter for balancing the residuals, on account of the PDE and the initial and boundary data, respectively.

5.3 Estimate on the generalization error.

We denote the PINN, obtained by the algorithm 2.1, for approximating the BO equation, as $\mathbf{u}^* = \mathbf{u}_{\theta^*}$, with θ^* being a (approximate, local) minimum of the loss function (2.13),(5.5). We consider the following generalization error,

$$\mathcal{E}_G := \left(\int_0^T \int_0^1 \|\mathbf{u}(x,t) - \mathbf{u}^*(x,t)\|^2 dx dt \right)^{\frac{1}{2}}, \tag{5.6}$$

with $\|\cdot\|$ denoting the Euclidean norm in $\mathbb{R}^d$. We will bound the generalization error in terms of the following *training errors*,

$$\mathcal{E}_T^2 := \underbrace{\sum_{n=1}^{N_{tb}} w_n^{tb} |\mathcal{R}_{tb,\theta^*}(x_n)|^2}_{(\mathcal{E}_T^{tb})^2} + \underbrace{\sum_{n=1}^{N_{sb}} w_n^{sb} |\mathcal{R}_{sb,\theta^*}(x_n,t_n)|^2}_{(\mathcal{E}_T^{sb})^2} + \lambda \underbrace{\sum_{n=1}^{N_{int}} w_n^{int} |\mathcal{R}_{int,\theta^*}(x_n,t_n)|^2}_{(\mathcal{E}_T^{int})^2}. \tag{5.7}$$

As in the previous sections, the training errors can be readily computed *a posteriori* from the loss function (5.5).

We have the following bound on the generalization error in terms of the training error,

Theorem 5.2. *Let $u \in C^k([0,1]\times[0,T])$ be the unique classical solution of Benjamin-Ono equation* (5.1). *Let $u^* = u_{\theta^*}$ be the PINN, generated by algorithm 2.1, with loss function* (5.5). *Then, the generalization error* (5.6) *is bounded by,*

$$\begin{aligned} \varepsilon_G \leqslant\ & C_1\big(\varepsilon_T^{tb} + \varepsilon_T^{int} + C_2(\varepsilon_T^{sb})^{1/2} \\ & + (C_{quad}^{tb})^{1/2} N_{tb}^{-\alpha_{tb}/2} + (C_{quad}^{int})^{1/2} N_{int}^{-\alpha_{int}/2} + C_2 (C_{quad}^{sb})^{1/4} N_{sb}^{-\alpha_{sb}/4}\big) \end{aligned} \tag{5.8}$$

where

$$\begin{aligned} C_1 &= \sqrt{T + 2C_3 T^2 e^{2C_3 T}}, \\ C_2 &= T^{1/4} \sqrt{2(\|u^*\|_{C_t^0 C_x^2} + \|u\|_{C_t^0 C_x^2}) + 2\|u\|_{C_t^0 C_x^0}(\|u\|_{C_t^0 C_x^0} + \|u^*\|_{C_t^0 C_x^0})}, \\ C_3 &= \frac{1}{2} + \|u^*\|_{C_t^0 C_x^1} + \frac{1}{2}\|u\|_{C_t^0 C_x^1}, \end{aligned} \tag{5.9}$$

and $C_{quad}^{tb} = C_{qaud}^{tb}\left(\|\mathcal{R}_{tb,\theta^*}\|_{C^k}\right)$, $C_{quad}^{int} = C_{qaud}^{int}\left(\|\mathcal{R}_{int,\theta^*}\|_{C^{k-2}}\right)$, and $C_{quad}^{sb} = C_{qaud}^{sb}\left(\|\mathcal{R}_{sb,\theta^*}\|_{C^k}\right)$ are the constants associated with the quadrature errors (3.8)-(3.10).

Proof. We will drop explicit dependence of all quantities on the parameters θ^* for notational convenience. We denote the difference between the underlying solution u of (5.1) and PINN u^* as $\hat{u} = u^* - u$. Using the PDE (5.1) and the definitions of the residuals (5.2)-(5.4), a straightforward calculation yields the following PDE for the $\hat{u}$,

$$\begin{aligned} \hat{u}_t + H\hat{u}_{xx} + u^* u_x^* - u u_x = \mathcal{R}_u, \quad & \text{a.e. } (x,t) \in (0,1)\times(0,T), \\ \hat{u}(0,t) - \hat{u}(1,t) = \mathcal{R}_{sb}, \quad & t \in (0,T), \\ \hat{u}(x,0) = \mathcal{R}_{tb}, \quad & x \in (0,1). \end{aligned} \tag{5.10}$$

We take a inner product of the equation in (5.10) with the vector $\hat{u}$, and integrate by parts to obtain the term coming from the Hilbert transform

$$\int_0^1 \hat{u} H(\hat{u}_{xx})\, dx = -\int_0^1 \hat{u}_x H(\hat{u}_x)\, dx + H(\hat{u}_x)(1)\hat{u}(1) - H(\hat{u}_x)(0)\hat{u}(0) = H(\hat{u}_x)(1)\hat{u}(1) - H(\hat{u}_x)(0)\hat{u}(0)$$

For the boundary terms:

$$\begin{aligned} \Big[H(\hat{u}_x)(1)\hat{u}(1) - H(\hat{u}_x)(0)\hat{u}(0)\Big] &= \Big[\big(H(\hat{u}_x)(1) - H(\hat{u}_x)(0)\big)\hat{u}(1) + H(\hat{u}_x)(0)(\hat{u}(1) - \hat{u}(0))\Big] \\ &= H(\hat{u}_x)(0)(\hat{u}(1) - \hat{u}(0)) \leq \|H(\hat{u}_x)(0)\|_{C_t^0} |\mathcal{R}_{sb}| \leq (\|u\|_{C_t^0 C_x^2} + \|u^*\|_{C_t^0 C_x^2}) |\mathcal{R}_{sb}| \end{aligned}$$

In the second line, we used the periodicity of u and u^*. For other terms, we can follow arguments given before and get

$$\begin{aligned}
\frac{1}{2}\frac{d}{dt}\int_0^1 \hat{u}^2\,dx &= -\int_0^1 \hat{u}H\hat{u}_{xx}\,dx - \int_0^1 \hat{u}(\hat{u}\hat{u}_x - u\hat{u}_x + u_x\hat{u})\,dx + \int_0^1 \hat{u}\mathcal{R}_{int}\,dx \\
&\leqslant (\|u^*\|_{C_x^2} + \|u\|_{C_x^2})|\mathcal{R}_{sb}| - \int_0^1 (u_x^* - \frac{1}{2}u_x)\hat{u}^2 - \frac{1}{2}u\hat{u}^2\Big|_0^1 + \int_0^1 \hat{u}\mathcal{R}_{int}\,dx \\
&\leqslant (\|u^*\|_{C_x^2} + \|u\|_{C_x^2})|\mathcal{R}_{sb}| \\
&+ (\|u^*\|_{C_x^1} + \frac{1}{2}\|u\|_{C_x^1})\int_0^1 \hat{u}^2\,dx + \|u\|_{C_x^0}(\|u\|_{C_x^0} + \|u^*\|_{C_x^0})|\mathcal{R}_{sb}| \\
&+ \frac{1}{2}\int_0^1 \mathcal{R}_{int}^2\,dx + \frac{1}{2}\int_0^1 \hat{u}^2\,dx \\
&\leqslant \big(\|u^*\|_{C_t^0C_x^2} + \|u\|_{C_t^0C_x^2} + \|u\|_{C_t^0C_x^0}(\|u\|_{C_t^0C_x^0} + \|u^*\|_{C_t^0C_x^0})\big)|\mathcal{R}_{sb}| \\
&+ \frac{1}{2}\int_0^1 \mathcal{R}_{int}^2\,dx + (\frac{1}{2} + \|u^*\|_{C_t^0C_x^1} + \frac{1}{2}\|u\|_{C_t^0C_x^1})\int_0^1 \hat{u}^2\,dx \\
&=: C_1|\mathcal{R}_{sb}| + \frac{1}{2}\int_0^1 \mathcal{R}_{int}^2\,dx + C_2\int_0^1 \hat{u}^2\,dx
\end{aligned} \tag{5.11}$$

Then integrating the above inequality over $[0,\bar{T}]$ for any $\bar{T} \leqslant T$, we obtain

$$\begin{aligned}
\int_0^1 \hat{u}(x,\bar{T})^2\,dx &\leqslant \int_0^1 \mathcal{R}_{tb}^2\,dx + 2C_1T^{1/2}(\int_0^T \mathcal{R}_{sb}^2\,dt)^{1/2} + \int_0^T\int_0^1 \mathcal{R}_{int}^2\,dxdt + 2C_2\int_0^{\bar{T}}\int_0^1 \hat{u}^2\,dxdt \\
&\leqslant (1 + 2C_2Te^{2C_2T})\Big(\int_0^1 \mathcal{R}_{tb}^2\,dx + C_1T^{1/2}(\int_0^T \mathcal{R}_{sb}^2\,dt)^{1/2} + \int_0^T\int_0^1 \mathcal{R}_{int}^2\,dxdt\Big)
\end{aligned} \tag{5.12}$$

We use Cauchy-Schwarz in the first line and Gronwall's inequality in the second line. Then integrate (5.12) over $\bar{T} \in [0,T]$

$$\begin{aligned}
\varepsilon_G^2 &:= \int_0^T\int_0^1 \hat{u}(x,\bar{T})^2\,dxd\bar{T} \\
&\leqslant (T + 2C_2T^2e^{2C_2T})\Big(\int_0^1 \mathcal{R}_{tb}^2\,dx + 2C_1T^{1/2}(\int_0^T \mathcal{R}_{sb}^2\,dt)^{1/2} + \int_0^T\int_0^1 \mathcal{R}_{int}^2\,dxdt\Big)
\end{aligned} \tag{5.13}$$

with

$$C_1 = \|u^*\|_{C_t^0C_x^2} + \|u\|_{C_t^0C_x^2} + \|u\|_{C_t^0C_x^0}(\|u\|_{C_t^0C_x^0} + \|u^*\|_{C_t^0C_x^0}), \quad C_2 = \frac{1}{2} + \|u^*\|_{C_t^0C_x^1} + \frac{1}{2}\|u\|_{C_t^0C_x^1} \tag{5.14}$$

Finally, applying the estimates (3.8), (3.9), (3.10) on the quadrature error, and definition of training errors (5.7), yields the desired inequality (5.8).

□

5.4 Evaluation of singular integral

For test cases, we'll look at single and double soliton as proposed in [4]. We first consider periodic setting in single soliton case, i.e.

$$\begin{aligned}
&u_t + uu_x + H_{per}u_{xx} = 0 \quad \forall x \in \mathbb{T}, t \in (0,T) \\
&u(x,0) = u_0(x) \quad \forall x \in \mathbb{T}
\end{aligned} \tag{5.15}$$

where $\mathbb{T} := \mathbb{R}/2L\mathbb{Z}$. The periodic Hilbert transform is defined by

$$H_{per}u(x) = \text{p.v.}\frac{1}{2L}\int_{-L}^{L} \cot(\frac{\pi}{2L}y)u(x-y)dy \tag{5.16}$$

Obviously, the difficulty lies in the singular integral term. To evaluate interior residual at training points, we need to evaluate Hilbert transform at these training points which, after discretization, is a u_{xx}-related summation. Since the most computationally part is second derivative term u_{xx}, it's naturally for us to search a discretization where required evaluations of u_{xx} are as less as possible. One natural approach is to use Cartesian grid instead of Sobol points as interior training points. Cartesian grid is regular in the sense of equidistant and periodicity, thus it has the potential that evaluations of u_{xx} can be reused such that only evaluations of u_{xx} on the Cartesian grid suffice to compute evaluations of $H_{per}u_{xx}$ on the same grid.

We denote the spatial discretization of Cartesian grid as $\{x_i\}_{i=-N}^{N}$ and additionally require $x_0 = 0$. And we can discretize singular integral term as

$$\begin{aligned} H_{per}u_{xx}(x) &= \text{p.v.}\frac{1}{2L}\int_{-L}^{L}\cot(\frac{\pi}{2L}y)u_{xx}(x-y)dy \\ &\approx \frac{1}{2N}\sum_{j=-N,j\neq 0}^{N}\cot(\frac{\pi}{2L}x_j)u_{xx}(x-x_j) \end{aligned} \tag{5.17}$$

We exclude index $j = 0$ in order to be consistent with the definition of principle value because $x_0 = 0$ is a singularity of $\cot(\frac{\pi}{2L}x_j)$. More importantly, what we need to compute is $H_{per}u_{xx}(x)\big|_{x_i}$ which can be represented as a discrete periodic convolution of $\cot(\frac{\pi}{2L}x_j)$ and $u_{xx}(x)|_{x_j}$

$$\begin{aligned} H_{per}u_{xx}(x_i) &\approx \frac{1}{2N}\sum_{j=-N,j\neq 0}^{N}\cot(\frac{\pi}{2L}x_j)u_{xx}(x_i-x_j) \\ &= \frac{1}{2N}\sum_{j=-N,j\neq 0}^{N}\cot(\frac{\pi}{2L}x_j)u_{xx}(x_{i-j}) \end{aligned} \tag{5.18}$$

This means that to compute $H_{per}u_{xx}(x_i), -N \leqslant i \leqslant N$ we only need to compute $u_{xx}(x_i), -N \leqslant i \leqslant N$. Moreover, discrete periodic convolution (5.18) can be accelerated by Fast Fourier transformation(FFT) from a complexity of $O(N^2)$ to $O(N\log(N))$.

5.5 Numerical experiments

Unlike previous cases, here we need an additional parameter $\Delta := \frac{\Delta t}{\Delta x}$ for Cartesian grid. The larger is Δ, more dense is the grid in spatial direction than temporal direction. Because evaluation of singular integral is the source of leading error, therefore we need to set Δ large to get PINN work. Fortunately, the decrease of grid in temporal direction doesn't hurt the error from space-time integral of $u(x,t)$ since, for two dimension composite trapezoidal rule, the quadrature error is of order $O(\Delta x\Delta t)$ and $\Delta x\Delta t$ is determined by total training points thus is independent of ratio Δ.

For periodic single soliton case, we have the exact solution

$$u(x,t) = \frac{2c\delta^2}{1-\sqrt{1-\delta^2}\cos(c\delta(x-ct-x_0))}, \quad \delta = \frac{\pi}{cL} \tag{5.19}$$

where L is the half periodicity. This represents a single bump moving to the right with speed c periodically with initial peak at $x = x_0$. In our experiments, we choose $L = 15$, $c = 0.25$ and $x_0 = 0$. We also observe a critical value of hyperparameter Δ. That is, for $\Delta \lessapprox 5$ the performance is getting better as the increase of Δ and for $\Delta \gtrapprox 6$ PINN easily converges to zero solution in a very short time.

For periodic double soliton case, the closed-form exact solution is very complicated. However if we take the long wave limit $k \to 0$, it's reduced to real line case with a simple expression

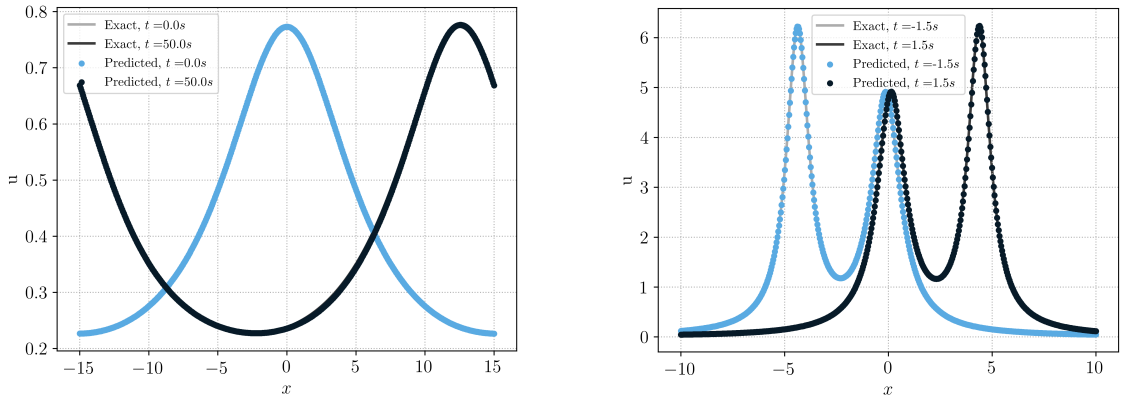
$$u(x,t) = \frac{4c_1c_2(c_1\lambda_1^2 + c_2\lambda_2^2 + (c_1+c_2)^3c_1^{-1}c_2^{-1}(c_1-c_2)^{-2})}{(c_1c_2\lambda_1\lambda_2 - (c_1+c_2)^2(c_1-c_2)^{-2})^2 + (c_1\lambda_1 + c_2\lambda_2)^2} \tag{5.20}$$

where

$$\begin{aligned} \lambda_1 &= x - c_1t \\ \lambda_2 &= x - c_2t \end{aligned} \tag{5.21}$$

This solution represents two waves that "collide" at $t = 0$ and separate for $t > 0$. For large $|t|$, $u(\cdot, t)$ is close to a sum of two one-solitons at different locations. We choose $c_1 = 2$ and $c_2 = 1$ in our experiments. Because singular integral is global, to evaluate Hilbert transform of PINN, we need PINN to be defined on $\mathbb{R}$ which is not the case. Computationally we are always on a bounded domain. Thus we approximate as follows: for computational domain $[-L, L]$, we first extend PINN by zero to $[-5L, 5L]$ and then use a similar discretization as (5.17) and compute discrete periodic convolution of $\frac{1}{\pi x_j}$ and $u_{xx}(x)|_{x_j}$ and finally restrict the result of discrete periodic convolution onto domain $[-L, L]$.

Different from periodic single soliton case, we haven't observe any critical value for $\Delta \leqslant 30$, so one can expect a nicer performance for some $\Delta > 30$. And surprisingly, we also observe that LBFGS optimizer converges more quickly than single soliton case under the same hyperparameter configuration, see best performance in Table 11 and correspondent plots in figure 11.



(a) Single soliton

(b) Double soliton

Figure 11: Exact and PINN solution to single and double soliton of BO equation

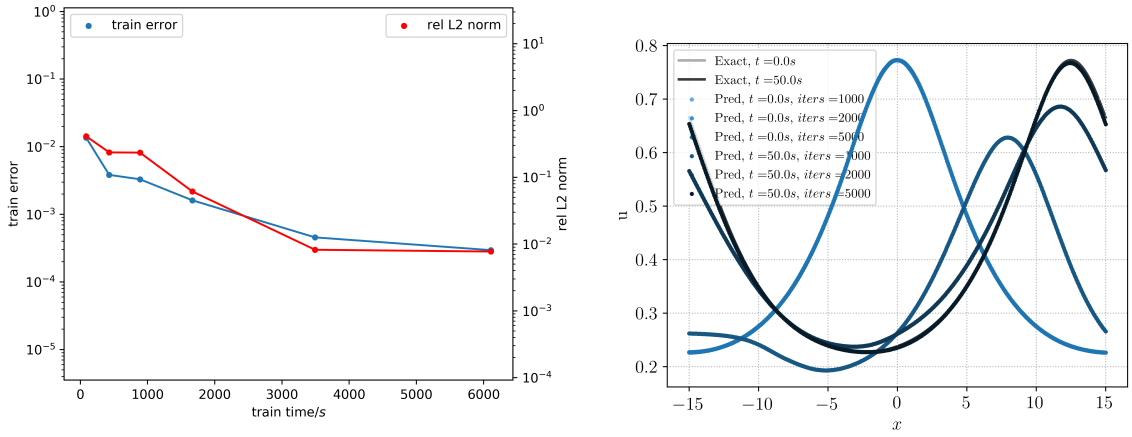
	N_{int}	N_{sb}	N_{tb}	$K-1$	d	λ	$\mathcal{E}_T$	$\mathcal{E}_G^r$	Δ
Single Soliton	32768	8192	8192	12	24	1	0.000296	0.773%	4
Double Soliton	65536	16384	16384	4	20	10	0.00616	0.657%	30

Table 11: Best performing *Neural Network* configurations for the Single Soliton and Double Soliton problem. Low-discrepancy Sobol points are used for all boundary points; Cartesian grids are used for all interior points.

max_iters	**training time**/s	ε_T	ε_G^r
100	87	1.36e-02	4.11e-01
500	430	3.83e-03	2.36e-01
1000	888	3.30e-03	2.34e-01
2000	1667	1.61e-03	6.13e-02
5000	3492	4.56e-04	8.22e-03
10000	6107	2.96e-04	7.73e-03

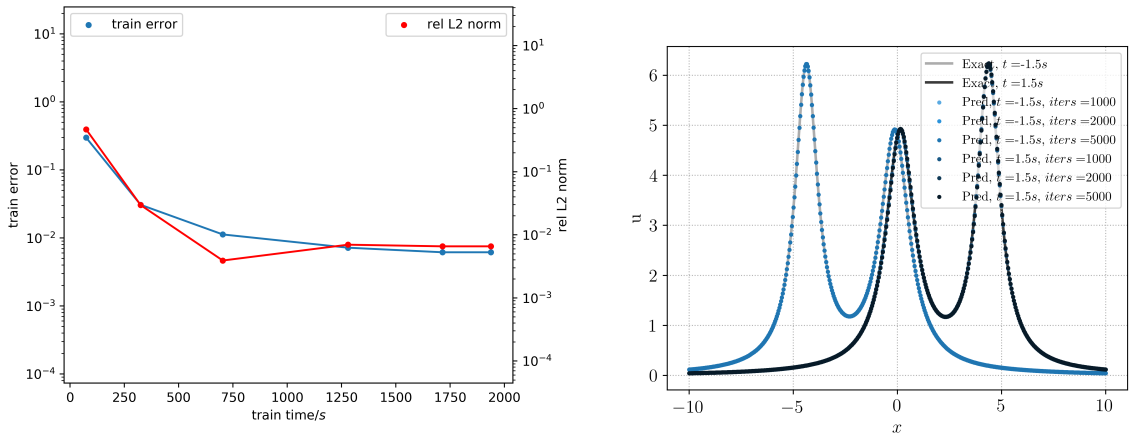
Table 12: Results for single soliton of BO equation with different training iterations

The profiles and data at early iterations for Benjamin-Ono single and double soliton case are summarized in figure 12, Table 12 and figure 13, Table 13 respectively. And all other hyperparameters are the same as in the Table 11.



(a) Single soliton (b) Single soliton

Figure 12: (a) is the plot of train error and relative error versus train time; (b) is the plots at initial and final time for different train iterations



(a) Double soliton (b) Double soliton

Figure 13: (a) is the plot of train error and relative error versus train time; (b) is the plots at initial and final time for different train iterations

max_iters	**training time/s**	ε_T	ε_G^r
100	74	2.98e-01	4.69e-01
500	325	3.07e-02	2.96e-02
1000	703	1.13e-02	3.92e-03
2000	1280	7.19e-03	6.98e-03
5000	1715	6.16e-03	6.57e-03
10000	1937	6.16e-03	6.57e-03

Table 13: Results for double soliton of BO equation with different training iterations

It's interesting to notice that single soliton needs more time to train than double soliton under the same max_iters, which didn't happen for all previous equations. It's partially because the periodic boundary condition of single soliton case. We can see from figure 12(b) and 13(b) that the peak as well as the boundary values of single soliton at early stage are evidently lower than exact solution, thus more iterations and training time are required to resolve the peak and boundary values. In contrast with single soliton, double soliton on the real line resolves its peaks even at small max_iters, so heuristically the training later should be fast since its shape is almost correct. Also we observe that, in figure 13(a) and Table 13, "$max\_iters = 1000$" has the best relative generalization error. It slightly breaks the correlation between training error ε_T and relative generalization error ε_G^r. It's caused by the error coming from the extension of domain and the zero padding.

6 Discussion

In this paper, following the approach proposed in [**?**, **?**, **?**], we proved the rigorous a posterior bound for generalization error of PINN in the context of dispersive PDEs. Besides, we tested our PINN algorithm on several standard test cases and the relative L2 errors of all test cases can be easily reduced to less than 1%. PINN is very easy to implement and our results outperform the convergent finite difference methods a lot as reported in [3, 4, 10, 11] in the sense of relative L2 error and computational time.

PINN neither relies on any reformulation of PDEs, nor needs observations of the unknown solution. In other words, PINNs, as unsupervised learning, can learn directly from the PDE problem. What we need are only the equation imposed with the right boundary conditions. Thus PINN is not ad hoc and can be easily adapted to a large variety of PDEs.

The gist of PINN is to control the difference of solutions through the interior and boundary residuals. The boundary conditions to ensure high-order dispersive PDEs well-posed are usually complicated. For example, Kawahara equation has five spatial boundary conditions. Surprisingly, most of time, they are exactly what we need to bound the generalization error. It's interesting to see how they come into play in the generalization error bound. There's a very subtle connection between the boundedness of generalization error and the regularity theory of dispersive PDEs.

Since our bound on generalization error is of a posterior type, it should be interpreted properly as: *as long as the PINN is trained well, it generalizes well*. However in the context of dispersive PDEs, it's not always trivial to train PINN well. On the one hand, differentiating high-order PDEs needs operations of exponential growth. On the other hand, dispersive term generally results in complicated oscillations and fractal structures in the solution, see figure 5 for example. Such multi-scale features are difficult to be learned by PINN. So a potential direction could be the design of suitable architecture and hyperparameters to make PINN more robust for dispersive problems.

References

[1] T. Benjamin. Internal waves of permanent form in fluid of great depth. *J. Fluid. Mech.*, 29:559–592, 1967.

[2] R. Camassa and D. Holm. An integrable shallow water equation with peaked solitons. *Phys. Rev. Lett*, 71:1661–1664, 1993.

[3] J. C. Ceballos, M. Sepúlveda, and O. P. Vera Villagrán. The korteweg–de vries–kawahara equation in a bounded domain and some numerical results. *Applied Mathematics and Computation*, 190(1):912 – 936, 2007.

[4] R. Dutta, H. Holden, U. Koley, and N. H. Risebro. Convergence of finite difference schemes for the benjamin–ono equation. *Numerische Mathematik*, 134(2):249–274, 2015.

[5] R. Dutta, H. Holden, U. Koley, and N. H. Risebro. Operator splitting for the benjamin-ono equation. *J. Differential Equations*, 259(11):6694–6717, 2015.

[6] R. Dutta, U. Koley, and N. H. Risebro. Convergence of a higher order scheme for the korteweg–de vries equation. *SIAM J. Numer. Anal.*, 53(4):1963–1983, 2015.

[7] A. Faminskii and N. Larkin. Initial-boundary value problems for quasilinear dispersive equations posed on a bounded interval. *Electronic Journal of Differential Equations*, 01:1–20, 2010.

[8] R. Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

[9] I. Goodfellow, Y. Benigo, and A. Courville. Deep learning. *MIT press*, 2016.

[10] H. Holden, U. Koley, and H. Risebro. Convergence of a fully discrete finite difference scheme for the korteweg-de vries equation. *IMA Journal of Numerical Analysis*, 35(3):1047–1077, 2015.

[11] H. Holden and X. Raynaud. Convergence of a finite difference scheme for the camassa–holm equation. *SIAM J. Numerical Analysis*, 44:1655–1680, 01 2006.

[12] J. K. Hunter and J. Scheurle. Existence of perturbed solitary wave solutions to a model equation for water waves. *Physica D.*, 32:253–268, 1988.

[13] C. E. Kenig, G. Ponce, and L. Vega. Uniqueness properties of solutions to the benjamin-ono equation and related models, 2019.

[14] U. Koley. Error estimate for a fully discrete spectral scheme for korteweg-de vries-kawahara equation. *Cent. Eur. J. Math.*, 10(1):173–187, 2012.

[15] U. Koley. Finite difference schemes for the korteweg–de vries–kawahara equation. *Int. J. Numer. Anal. Model.*, 13(3):344–367, 2016.

[16] K.-H. Kwek, H. Gao, W. Zhang, and C. Qu. An initial boundary value problem of camassa–holm equation. *Journal of Mathematical Physics*, 41:8279–8285, 12 2000.

[17] I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 2000.

[18] L. Molinet. Global well-posedness in l^2 for the periodic benjamin–ono equation. *Amer. J. Math.*, 130:635–683, 2008.

[19] H. Ono. Algebraic solitary waves in stratified fluids. *J. Phy. Soc. Japan*, 39(4):1082–1091, 1975.

[20] A. Parker. On the camassa-holm equation and a direct method of solution i. bilinear form and solitary waves. *Proceedings of The Royal Society A: Mathematical, Physical and Engineering Sciences*, 460:2929–2957, 10 2004.

[21] A. Parker. On the camassa–holm equation and a direct method of solution. ii. soliton solutions. *Proceedings of The Royal Society A Mathematical Physical and Engineering Sciences*, 461:3611, 08 2005.

[22] A. Parker. On the camassa–holm equation and a direct method of solution. iii. n-soliton solutions. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering*, 461:3893, 08 2005.

[23] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[24] M. N. Rasoulizadeh and J. Rashidinia. Numerical solution for the kawahara equation using local rbf-fd meshless method. *Journal of King Saud University - Science*, 32(4):2277 – 2283, 2020.