ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CSE**lab**
Computational Science and Engineering Laboratory

# Physics Informed Neural Networks for Identification and Forecasting of Chaotic Dynamics

Bachelor Thesis

Zador Pataki

Summer 2020

Supervisors: Prof. Dr. Petros Koumoutsakos
Pantelis Vlachas

Computational Science and Engineering Laboratory, ETH Zürich

"I hereby declare that I am the author of the written work at hand, which is original and written in my own words. Exceptional content is explicitly cited as such and listed in the Bibliography."

# Abstract

In this work we investigate the ability of physics informed neural networks — data-driven neural networks which incorporate laws of physics generally in the form of partial differential equations — to infer the solutions of and to identify a number of nonlinear partial differential equations. In particular, in both continuous- and discrete-time cases, we solve problems of inference and identification of partial differential equations with different levels of complexity that exhibit distinct ranges of nonlinear phenomena. A relatively simple physics informed neural network framework was created that can be easily adjusted and applied to solve any of the mentioned problems concerning different partial differential equations. Highlighting its data efficiency and robustness, the framework was effectively applied to solve problems of inference and identification of second order partial differential equations; the Burgers and the Nonlinear Schrodinger equation. On the other hand when handling the Kuramoto-Sivashinsky equation, a fourth order partial differential equation known for its chaotic dynamic behaviour, the adjusted framework was found not to be as accurate as traditional numerical methods.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In the field of machine learning, neural networks (NNS) — conventionally driven by empirical data — have become a popular tool applied across many domains, including in science. In scientific domains, however, incorporating constraints other than that of empirical data (e.g. conservation laws) has proven to be an effective method, as it makes it possible to generate relatively accurate solutions even in cases of limited data [1], which would otherwise present many difficulties; a particularly interesting feature due to the fact that data is often hard to acquire. Deep NNs (DNNs) have been introduced as a method which incorporates multiple NN layers between the input and output layers. They have been found to produce results which have, in some cases, surpassed human expert performances [2,3], and recently have also been used to explore the scientific domain in machine learning.

Physics informed NNs (PINNs) are NNs which to some degree incorporate laws of physics generally in the form of partial differential equations (PDEs), and have existed as a method in machine learning for over two decades [4]. Recently, however, with the availability of automatic differentiation in machine learning libraries simplifying their application, PINNs have attracted a new wave of interest inspiring further research. They have been used in a range of applications, such as for discovering general physical concepts [5, 6] and for predicting [7,8] and extracting quantitative information of fluid flows [9] using experimental data. Beyond the pursuit of applications, researchers have also developed new approaches to further explore PINNs, such as the quantification of uncertainty in PINNs by employing arbitrary polynomial chaos [10], or physics informed model developments with no experimental data provided at all [11].

In a paper by Zhang et al. [12], a summary of the most significant advantages of PINN applications were presented, where the authors discuss PINNs incorporating DNNs. One key feature of such PINNs that the authors highlighted is that with the use of PINNs "we can truly predict the state of the system, unlike the DNNs driven solely by data that can interpolate accurately only

within the training domain." In this work, we try to explore such features of PINNs in a number of examples incorporating various PDEs occurring in physics.

We implement PINNs incorporating DNNs in two main problems: (i) a problem of inference, where we have full knowledge of the PDEs in question, and using only initial and boundary information we attempt to predict the corresponding solutions; (ii) a problem of identification, where only the general form of the PDEs are known and the goal is to predict the unknown coefficients using data of the PDEs' solution provided across the entire spatio-temporal domains. For each of the two problems two different algorithms are presented: firstly each problem is treated in continuous-time and secondly in discrete-time resulting in four different algorithms, which can be easily modified to be applied to different PDEs.

By enforcing the physical concepts behind PDEs in loss functions, I wanted to develop algorithms to solve the problems mentioned earlier applied to the Burgers equation — according to a work by Raissi et al. [6] — and then adjust these algorithms to apply them to more complex examples of the Nonlinear Schrodinger equation and the Kuramoto-Sivashinsky equation. The Kuramoto-Sivashinsky equation stands out among the other PDEs treated in this paper, both for its higher level of complexity and for its chaotic dynamic behaviour. Such behaviours are present in natural systems, as noted by Vlachas et al. [13] and have hindered the forecasting and the understanding of such systems. However, with recent developments in machine learning, researchers have taken steps to overcome these obstacles.

The Kuramoto-Sivashinsky equation is a prototypical dynamical system developed by Yoshiki Kuramoto and Gregory Sivashinsky to model diffusive instabilities in a laminar flame front. Raissi et al. solved an identification problem of the Kuramoto-Sivashinsky equation in discrete-time with a high level of accuracy, relying on Gaussian processes [1]. To our knowledge, however, such problems of the Kuramoto-Sivashinsky equation are yet to be solved using PINNs and so this work aims to investigate the effectiveness of PINNs when applied to PDEs with chaotic dynamics.

# Chapter 2

# Methods

In this work, 1D nonlinear PDEs were considered of the general form

$$u_t + \mathcal{N}[u;\ \lambda] = 0,\ x \in \Omega,\ t \in [0,\ T], \tag{2.1}$$

where $u_t$ denotes the time derivative of $u(t, x)$, which denotes the solution of a given PDE, $\mathcal{N}[\cdot;\ \lambda]$ is a non-linear operator parametrized by $\lambda$, and $\Omega$ is a subset of $\mathbb{R}^D$ [6]. Two main problems were considered, where the goal of the first problem (inference) was to find the solution $u(t, x)$ given initial and boundary information and the coefficients $\lambda$, while the goal of the second problem (identification) was to find the coefficients $\lambda$ given some information on the PDE solution $u(t, x)$.

As opposed to conventional problems solved with NNs, the above mentioned problems were solved using minimal amounts of data. PINNs were trained to both fit the data provided and to model the inherent characteristics of the PDEs, where, for the latter, no additional data was required. The architectures of the PINNs treated in this paper consist of a DNN predicting solutions of the PDEs and the calculation of the partial derivatives of the predictions which are then incorporated in physics informing functions introduced below. Depending on the particular problem, the output of the PINNs includes the predictions of the solutions of the PDEs, its partial derivatives and furthermore the physics informing function. The training scheme of these PINNs greatly depends on enforcing both the target solutions and the physical information of the PDEs in a loss function.

In the continuous-time cases, the problems were tackled by creating physics informing functions $f(t, x)$ for each PDE of the general form:

$$f(t, x) = u_t + \mathcal{N}[u;\ \lambda]\,. \tag{2.2}$$

The reader should note that the function above is equal to the left hand side of PDE (2.1), and therefore in a circular way, the model begins to learn the

characteristics of the PDEs at hand as $f(t, x)$ converges towards zero. In the case of discrete-time methods, on the other hand, only a pair of time-steps were considered. By applying a general form of Runge-Kutta methods with $q$ stages to the general equation (2.1), the solutions at two time-steps $n$ and $n + 1$ were related in the following manner:

$$\begin{aligned} u^{n+c_i} &= u^n - \Delta t \sum_{j=1}^{q} a_{ij} \mathcal{N}\left[u^{n+c_j}; \lambda\right], \ i \in \{1, ..., q\}, \\ u^{n+1} &= u^n - \Delta t \sum_{j=1}^{q} b_j \mathcal{N}\left[u^{n+c_j}; \lambda\right], \end{aligned} \tag{2.3}$$

where $u^n$ and $u^{n+1}$ are the solutions to the PDE at time-steps $n$ and $n + 1$ respectively, while $\{a_{ij}, b_j, c_j\}$ are elements of the Butcher's Tableau used to specify the corresponding Runge-Kutta method. As opposed to the continuous-time case, where the physics informing function (2.2) encapsulates the PDE characteristics of the entire spatial and temporal domain, in the discrete case, we define the physics informing function as:

$$f(n, c_j) = -\mathcal{N}\left[u^{n+c_j}; \lambda\right], \tag{2.4}$$

providing only enough information of the PDE to relate the solutions $u^n$ and $u^{n+1}$, i.e. to make the time step comparison possible.

For building the PINN framework applied to the Burgers equation (see chapter 1), the L-BFGS optimizer was used to optimize over the case specific loss functions. The mentioned optimizer is a full-batch gradient based optimization algorithm, which Raissi et al. [6] claimed was more appropriate when treating cases with smaller data-sets as opposed to otherwise more computationally efficient mini-batch settings which can readily employ modern variants of the stochastic gradient descent. Solving the inference and the identification problems requires different methods depending on whether the continuous-time or the discrete-time cases are considered, and so it is worth discussing the methods used for each of the four combinations of inference/identification problems and continuous-/discrete-time cases individually although all methods have similar approaches.

## 2.1 Case specific methods in continuous-time

### 2.1.1 Continuous-time inference methods

In the case of continuous-time inference, the loss function of the training scheme incorporates the error relating to initial data, boundary conditions and that of the PDE underlying the physics informing function (2.2). In the case of inference, the reader should note, the PDEs in question are known and so the coefficients $\lambda$ of the just now mentioned function are known constants. In the following examples, the continuous-time methods for solving both the Burgers equation and the Nonlinear Schrodinger equation will be presented in detail.

#### Continuous-time inference of the Burgers equation

The Burgers equation is a fundamental PDE, with its 1D case being of the form (2.1). The reader will note that the Burgers equation is an example used in all combinations of continuous-/discrete-time cases and inference/identification problems in this paper. Due to the fact that it is a notably easier equation to apply in comparison to the other differential equations presented, PINN frameworks were first created to solve the above mentioned problems and were later adjusted and applied to the Nonlinear Schrodinger equation and the Kuramoto-Sivashinsky equation.

The Burgers equation along with Dirichlet boundary conditions read as follow:

$$\begin{aligned} &u_t + uu_x - (0.01/\pi)u_{xx} = 0, \;\; x \in [-1, 1], \;\; t \in [0, 1], \\ &u(0, x) = -\sin(\pi x), \\ &u(t, -1) = u(t, 1) = 0, \end{aligned} \tag{2.5}$$

such that the corresponding physics informing function can be extracted:

$$f = u_t + uu_x - (0.01/\pi)u_{xx}. \tag{2.6}$$

The following equation is the loss function, which the L-BFGS optimizer attempts to minimize and thereby trains the DNN model in this example. It is a linear combination of mean squared errors (MSEs) relating to the initial and boundary conditions presented in (2.5) and the physics informing function (2.6):

$$\mathcal{L} = MSE_0 + MSE_b + MSE_f \tag{2.7}$$

where

$$MSE_0 = \frac{1}{N_0}\sum_{i=1}^{N_0}|u\left(t_0^i, x_0^i\right) - u_0^i|^2$$
$$MSE_b = \frac{1}{N_b}\sum_{i=1}^{N_b}\left(|u^i\left(t_b^i, -1\right)|^2 + |u^i\left(t_b^i, 1\right)|^2\right)$$
$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}|f\left(t_f^i, x_f^i\right)|^2.$$

Here, $\{x_0^i, u_0^i\}_{i=1}^{N_0}$ denotes the initial data, $\{t_b^i\}_{i=1}^{N_b}$ corresponds to the points on the boundary, and $\{t_f^i, x_f^i\}_{i=1}^{N_f}$ represents the points on $f(t,x)$ [6].

The above function illustrates that there is a multiplicity of domains at which the predictions of the PDE solutions are required in order to calculate the total loss. Hence, in a single training step different data-sets need to pass through the PINN. For this example the PINN architecture was constructed such that it takes two-dimensional inputs (one spatial and one temporal dimension) and provides two one-dimensional outputs. The first being the prediction $u(t,x)$ and the second being the prediction $f(t,x)$, where the latter was calculated through calculating the corresponding partial derivatives using an automatic differentiation function. The required network outputs used in (2.7) were generated in this manner by passing boundary data from both boundaries, initial data and data across the entire spatio-temporal domain individually. Similar approaches were taken for all other examples in this paper.

### Continuous-time inference of the Nonlinear Schrodinger equation

The Nonlinear Schrodinger equation is a nonlinear version of the Schrodinger equation in theoretical physics. It stands out from the other equations discussed in this paper in that its solution is complex and, therefore, two-dimensional.

The 1D Nonlinear Schrodinger Equation along with periodic boundary conditions reads as

$$\begin{aligned} &ih_t + 0.5h_{xx} + |h|^2 h = 0,\ x \in [-5,5],\ t \in [0, \pi/2], \\ &h(0,x) = 2\,\mathrm{sech}(x), \\ &h(t,-5) = h(t,5) \\ &h_x(t,-5) = h_x(t,5), \end{aligned} \tag{2.8}$$

where $h(t,x) = u(t,x) + iv(t,x)$ is the complex valued solution, which is treated in the two-dimensional form $h(t,x) = [u(t,x)\ v(t,x)]$. Analog to the previous example the resulting physics informing function reads as

$$f = ih_t + 0.5h_{xx} + |h|^2 h, \tag{2.9}$$

or more specifically in the forms it was applied:

$$f_u = u_t + 0.5v_{xx} + (u^2 + v^2)v$$
$$f_v = v_t - 0.5v_{xx} - (u^2 + v^2)u$$

and based on the initial and boundary conditions presented in (2.8) the loss function was constructed as the linear combination of the corresponding MSEs:

$$\mathcal{L} = MSE_0 + MSE_b + MSE_{b,x} + MSE_f \tag{2.10}$$

where

$$MSE_0 = \frac{1}{N_0}\sum_{i=1}^{N_0}\left(|u\left(t_0^i, x_0^i\right) - u_0^i|^2 + |v\left(t_0^i, x_0^i\right) - v_0^i|^2\right),$$

$$MSE_b = \frac{1}{N_b}\sum_{i=1}^{N_b}\left(|u^i\left(t_b^i, -1\right) - u^i\left(t_b^i, 1\right)|^2 + |v^i\left(t_b^i, -1\right) - v^i\left(t_b^i, 1\right)|^2\right),$$

$$MSE_{b,x} = \frac{1}{N_b}\sum_{i=1}^{N_b}\left(|u_x^i\left(t_b^i, -1\right) - u_x^i\left(t_b^i, 1\right)|^2 + |v_x^i\left(t_b^i, -1\right) - v_x^i\left(t_b^i, 1\right)|^2\right),$$

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left(|f_u\left(t_f^i, x_f^i\right)|^2 + |f_v\left(t_f^i, x_f^i\right)|^2\right).$$

As opposed to the example of Burgers equation, due to the higher level of complexity of the PDE and the more demanding boundary conditions (boundary conditions depend not only on outputs $h$ but also on $h_x$), instead of two one-dimensional outputs, each forward propagation of the PINN was set up to generate three two-dimensional outputs: $[u(t,x)\ v(t,x)]$, $[u_x(t,x)\ v_x(t,x)]$ and $[f_u(t,x)\ f_v(t,x)]$. Analogously to the example of the Burgers equation, however, four sets of data needed to be passed individually through the PINN in order to generate the required outputs of (2.10).

### 2.1.2 Continuous-time identification methods

In contrast to the case of inference, in continuous-time identification, the loss functions incorporate the MSEs relating to target-data spread across the entire spatio-temporal domain and that of the physics informing function (2.2). The former relates to the solution of PDE at hand — treated as being only partially known — while the latter includes the corresponding unknown coefficients $\lambda$

as parameters. In the following examples, the continuous-time method for identifying the parameters of the 1D Burgers equation and the 1D Kuramoto-Sivashinsky equation are presented.

### Continuous-time identification of the Burgers equation

The general form of the Burgers equation reads as

$$u_t + \lambda_1 u u_x - \lambda_2 u_{xx} = 0 \tag{2.11}$$

where its unknown coefficients have been set as learnable parameters $\lambda = (\lambda_1, \lambda_2)$. By assigning these parameters as parameters of the PINN used to train the model for this problem, the PDE coefficients are treated equivalently to the parameters of the layers of the DNN, and, as a result, they can be learned equivalently.

From equation (2.11) we can read out the physics informing function as before, only that in this case it is parameterized as follows:

$$f = u_t + \lambda_1 u u_x - \lambda_2 u_{xx} \tag{2.12}$$

The solutions of the unknown PDE was provided across the entire spatio-temporal domain[1] as opposed to in the cases of inference, and was used as a target data-set to solve the problem. The training process consisted of simultaneously trying to fit the model to the target data and also trying to converge the physics informing function (2.12) to zero by modifying the parameters $\lambda$. To tackle this problem, a relatively simple loss function was created, summarizing what was mentioned above, which reads as:

$$\mathcal{L} = MSE_u + MSE_f \tag{2.13}$$

where

$$MSE_u = \frac{1}{N}\sum_{i=1}^{N} |u(t_u^i, x_u^i) - u^i|^2$$

$$MSE_f = \frac{1}{N}\sum_{i=1}^{N} |f(t_u^i, x_u^i)|^2$$

In comparison to the training process of the inference problems, the application of the above described method of PDE identification is relatively simple.

[1]The reader should note that we used the Burgers equation mentioned in section 2.1.1 in all examples involving the Burgers equation including in this one

The outputs needed to calculate the loss (2.13) can be generated by passing a single input-set spread across the entire spatio-temporal domain through the PINN, which — like in the previous example of the Burgers equation — takes a two-dimensional input, and outputs only two one-dimensional outputs; the predictions $u(t, x)$ and $f(t, x)$. In other words, in a single training step, only a single forward and backward propagation is required, suggesting faster training speeds.

### Continuous-time identification of the Kuramoto-Sivashinsky equation

The Kuramoto-Sivashinsky equation is a fourth-order nonlinear PDE used to model the diffusive instabilities in laminar flame front and is known for its chaotic behaviour. In this identification problem we treated the 1D Kuramoto-Sivashinsky Equation of the form

$$u_t + uu_x + u_{xx} + u_{xxxx} = 0, \tag{2.14}$$

where the coefficients of the PDE were set as unknown parameters $\lambda = (\lambda_1, \lambda_2, \lambda_3$, resulting in the following partially known PDE:

$$u_t + \lambda_1 uu_x + \lambda_2 u_{xx} + \lambda_3 u_{xxxx} = 0, \tag{2.15}$$

from which the physics informing function according to (2.2) results:

$$f(t, x) = u_t + \lambda_1 uu_x + \lambda_2 u_{xx} + \lambda_3 u_{xxxx}. \tag{2.16}$$

Target-data spread across the spatio-temporal domain was generated for an example of the PDE (2.14) with periodic boundary conditions. The goal of this example was to identify the mentioned unknown parameters. The method for solving this problem is analogous in many ways to the previous example of identification, including that the PINNs used are identical (except for the case specific modifications of the physics informing function) and the optimization occurs over the same loss function (2.13). However, since this example is of a fourth-order PDE while the other examples mentioned are of second-order, the higher complexity of the PDE solutions suggest lower accuracy in prediction.

It was found, in fact, that the same application was not capable of generating representative identifications, and to study this behaviour, different optimization methods were implemented. In the first stage, the DNN architecture was modified by increasing the number of neurons in each layer, and the entire

provided solution data-set of $6.4 \cdot 10^4$ points was used as target data, in order to assist the training process as much as possible. Here the number of neurons per layer were limited by the memory of the available GPU. Later the L-BFGS optimizer was replaced with the Adam optimizer incorporating a simple step-based learning rate scheduler, which we suspected would help the model fit the target date. Finally, we combined both the Adam optimizer and the L-BFGS optimizer, aiming to fine tune the model using the latter optimizer after the fitting of the former optimizer was completed.

## 2.2 Case specific methods in discrete-time

### 2.2.1 Discrete-time inference methods

For the discrete-time cases, the approaches presented to solve the problems are vastly different as was already implied earlier. In the case of discrete-time inference, instead of predicting the solution of the PDEs in the entire spatio-temporal domain, the goal instead is to predict the solution of the PDE in question at some time-step $n+1$ given the solution at the previous time-step $n$ and given boundary conditions. Following the work of Raissi et al. [6], (2.3) was rewritten individually for both the inference and the identification cases, to make it applicable.

#### Discrete-time inference of the Burgers equation

For the case of discrete-time inference, (2.3) was rewritten and equivalently re-expressed in the following form:

$$\begin{aligned} u^n &= u_i^n, \ i \in \{1, \dots, q\}, \\ u^n &= u_{q+1}^n, \end{aligned} \tag{2.17}$$

where

$$\begin{aligned} u_i^n &:= u^{n+c_i} + \Delta t \sum_{j=1}^{q} a_{ij} \mathcal{N}[u^{n+c_j}], \ i \in \{1, \dots, q\}, \\ u_{q+1}^n &:= u^{n+1} + \Delta t \sum_{j=1}^{q} b_j \mathcal{N}[u^{n+c_j}]. \end{aligned}$$

For this example, the problem of (2.5) was treated again, where, based on the general equation (2.4) the following physics informing function for the discrete Burgers equation results:

$$f(n, c_j) = -\mathcal{N}[u^{n+c_j}] = -u^{n+c_j} u_x^{n+c_j} + (0.01/\pi) u_{xx}^{n+c_j}. \tag{2.18}$$

Based on the above mentioned approach and by incorporating the relationships defined in (2.18) and the boundary conditions defined in (2.5), the loss function used to train the model was created as the linear combination of the sum of squared errors (SSEs) and reads as

$$\mathcal{L} = SSE_n + SSE_b \tag{2.19}$$

where

$$SSE_n = \sum_{j=1}^{q+1} \sum_{i=1}^{N_n} |u_j^n(x^{n,i}) - u^{n,i}|^2,$$
$$SSE_b = \sum_{i=1}^{q} \left( |u^{n+c_i}(-1)|^2 + |u^{n+c_i}(1)|^2 \right) + |u^{n+1}(-1)|^2 + |u^{n+1}(1)|^2.$$

Here, $\{x^{n,i}, u^{n,i}\}_{i=1}^{N_n}$ corresponds to the data at time $t^n$. The DNN in the PINN takes the one-dimensional inputs $x^{n,i}$ and outputs the prediction of the solution $u^{n+1}$ and all of the corresponding elements of the Runge-Kutta scheme outputs $u^{n+c_i}$. On top of this, by calculating the spatial derivatives of the mentioned output and by implementing the physics informing function (2.18), the PINN also outputs the solution $u^n$ in a manner suggested by (2.3). Building on this, the loss function, in words, enforces the boundary conditions in $SSE_b$ of the predicted solutions $u^{n+1}$, and also relates the said outputs to the inputs in $SSE_n$ hence (2.3).

Although less straight-forward, this approach is similar to the corresponding case in continuous-time in that, in one training step, the PINN takes two sets of inputs (on the boundary and across the spatial domain in this case), and outputs two one-dimensional outputs. The first output being the prediction of the DNN and the second being a physics informed prediction.

### 2.2.2 Discrete-time identification methods

The goal of discrete-time identification is identical to that of continuous-time in that we search for the unknown coefficients of a PDE given the solutions of the PDE, by setting them as parameters of the PINN. Similarly to the previous case of discrete-time inference, we look only at two time steps $n$ and $n+1$ related by the Runge-Kutta Scheme and the physics informing function (2.4).

**Discrete-time identification of the Burgers equation**

Like in the corresponding case of inference, we rewrite and equivalently re-express (2.3) as follows:

$$\begin{aligned} u^n &= u_i^n,\ i \in \{1, \dots, q\}, \\ u^{n+1} &= u_i^{n+1},\ i \in \{1, \dots, q\}, \end{aligned} \tag{2.20}$$

where

$$\begin{aligned} u_i^n &:= u^{n+c_i} + \Delta t \sum_{j=i}^{q} a_{ij} \mathcal{N}[u^{n+c_j}; \lambda]\ i \in \{1, \dots, q\}, \\ u_i^{n+1} &:= u^{n+c_i} + \Delta t \sum_{j=i}^{q} (a_{ij} - b_j) \mathcal{N}[u^{n+c_j}; \lambda],\ i \in \{1, \dots, q\}. \end{aligned}$$

In this example, the coefficients which are to be learned are from the Burgers equation of the general form (2.11), from which, according to (2.2), we extract the following physics informing function analogous to the previous example:

$$f(n, c_j) = -\mathcal{N}[u^{n+c_j}; \lambda] = -\lambda_1 u^{n+c_j} u_x^{n+c_j} + \lambda_2 u_{xx}^{n+c_j}. \tag{2.21}$$

Based on the physics informed relationships in (2.20) and the physics informing functions (2.21), the loss function used for training was constructed as follows:

$$\mathcal{L} = SSE_n + SSE_{n+1}, \tag{2.22}$$

where

$$\begin{aligned} SSE_n &= \sum_{j=1}^{q} \sum_{i=1}^{N_n} |u_j^n(x^{n,i}) - u^{n,i}|^2, \\ SSE_{n+1} &= \sum_{j=1}^{q} \sum_{i=1}^{N_{n+1}} |u_j^{n+1}(x^{n+1,i}) - u^{n+1,i}|^2. \end{aligned}$$

Where the discrete-time identification method stands out from the other methods in this paper is that every element of the loss function is physics informed. Per training step, the forward and backwards propagation through the PINN occurs twice. The PINN takes two-dimensional inputs of points across the spatial domain at times $t^n$ and $t^{n+1}$ and predicts the solutions $u^{n+1}$ and $u^n$ respectively. From these predictions and using the relationships presented in (2.20) and the physics informing function (2.21), the PINN finally predicts the solutions $u^n$ and $u^{n+1}$ at the corresponding input times. The final pair of

predictions are the individual outputs of the PINN which are compared to the corresponding target solutions, hence the loss function (2.22). Analog to the continuous-time case, the unknown coefficients are included as parameters in the PINN, and are learned in parallel to the training of the NN layers.

### Discrete-time identification of the Kuramoto-Sivashinsky equation

Given solutions to the 1D Kuramoto-Sivashinsky equation, the goal in this example is to identify the unknown coefficients of the partially known PDE (2.15). The target-data of this problem was provided in a work by Raissi et al. [1]. Analog to the previous example, to tackle solving this problem, we handle only two time steps $n$ and $n+1$ and corresponding target data $u^n$ and $u^{n+1}$. By setting up the PINNs to predict the solutions of the PDE at the input time step while incorporating the corresponding physics informing function, each output of the PINN is physics informed. As a result, each output is a function of the unknown coefficients $\lambda = (\lambda_1, \lambda_2, \lambda_3)$, which are set as parameters of the PINN, and are modified in parallel to the training of the DNN layers.

Like in the previous example, we treat equation (2.3) in the form (2.20), and the loss function used (2.22) is identical. Where these two approaches differ, however, is in the case specific modifications based on the PDE, where the physics informing function in this case reads as

$$f(n, c_j) = -\mathcal{N}[u^{n+c_j}; \lambda] = -\lambda_1 u^{n+c_j} u_x^{n+c_j} - \lambda_2 u_{xx}^{n+c_j} - \lambda_3 u_{xxxx}^{n+c_j}. \tag{2.23}$$

Hence the previous identification example of the Kuramoto-Sivashinsky equation, it was also suspected here that the PINN implemented would generate less accurate results than in the corresponding case of the Burgers equation. To assist the training of the models, the number of data points used as target data and the number of neurons in each fully connected layer was increased, however, unlike in the case of the continuous-time identification problem of the same PDE, a multi-optimizer test was not performed.

# Chapter 3

# Execution of the Experiments and Results

For the sake of simplicity and to make different approaches and examples comparable, in the areas of the methods which were not case specific, i.e. other than the loss functions and the physics informing of the PINNs, the approaches were made as similar as possible. Most importantly, the DNN architecture in each PINN was chosen to be equivalent across all examples (with the exception of the output dimensions), modified only if the predictions were unrepresentative of the targets. A four-layer-deep architecture was implemented with 100 neurons in each layer, which was selected after a number of stages of trial and error. Furthermore, in each case, the L-BFGS optimizing algorithm implemented was set to have the same settings in each case, including its termination tolerance[1] and learning rate. Again, the optimizer was only modified in a case where the predictions otherwise were not representative of the target. Where the individual applications varied, however, is the amount of training data used to train the models. In the upcoming sections, the results of the previously mentioned examples are presented.

## 3.1 Results of examples in continuous-time

### 3.1.1 Continuous-time inference results

#### Continuous-time inference of the Burgers equation

The goal of the experiment presented in this example was to predict the solution of the Burgers equation in (2.5). A plot of the target solution of this

[1]The termination tolerance determines when the optimization process is terminated. The termination can occur at first order of optimality or based on the rate at which function values/parameters change.

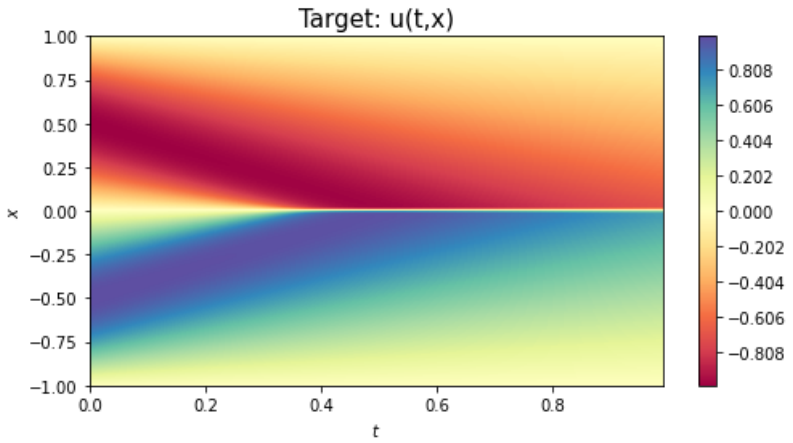


**Figure 3.1** Solution of the Burgers Equation

problem is presented in Fig. 3.1. As described in the methodology section, we wanted to solve this problem by fitting the initial and boundary conditions and by informing the DNN implemented using the physics informing function (2.6). To do this, a target data-set of 150 solutions, $u(t, x)$, was used consisting of solutions at 50 points at initial time, $t = 0$, randomly selected across the spatial domain, as well as solutions lying at 50 points on either boundary of $x$, randomly selected across the temporal domain. Furthermore, $2.56 \cdot 10^4$ points (resulting from the multiplication of 100 temporal against 256 spatial points) were used as an input for the predictions of the physics informing function, spread evenly across the entire spatio-temporal domain. The training process which simultaneously trained across the boundary and initial time target data and the physics informed loss was set to stop learning once it reached the default termination tolerance set by the L-BFGS algorithm, after which a final prediction was made using the entire set of points spread across the entire domain mentioned above (see Fig. 3.1).

A plot of the result of this prediction is presented in Fig. 3.2 A, where we note that the target data used is highlighted using black crosses, each corresponding to a single data point, spread across the mentioned domains. Three snapshots of the predictions are also presented at selected times marked in black dashed lines and were plotted against the exact solutions, in order to provide a more direct illustration of the goodness of the results. These three plots are presented in Fig. 3.2 A, B and C. Finally, the absolute and relative errors of this final prediction was calculated according to the L2-norm and were found to be $1.44 \cdot 10^{-4}$ and $2.68 \cdot 10^{-2}\%$ respectively.

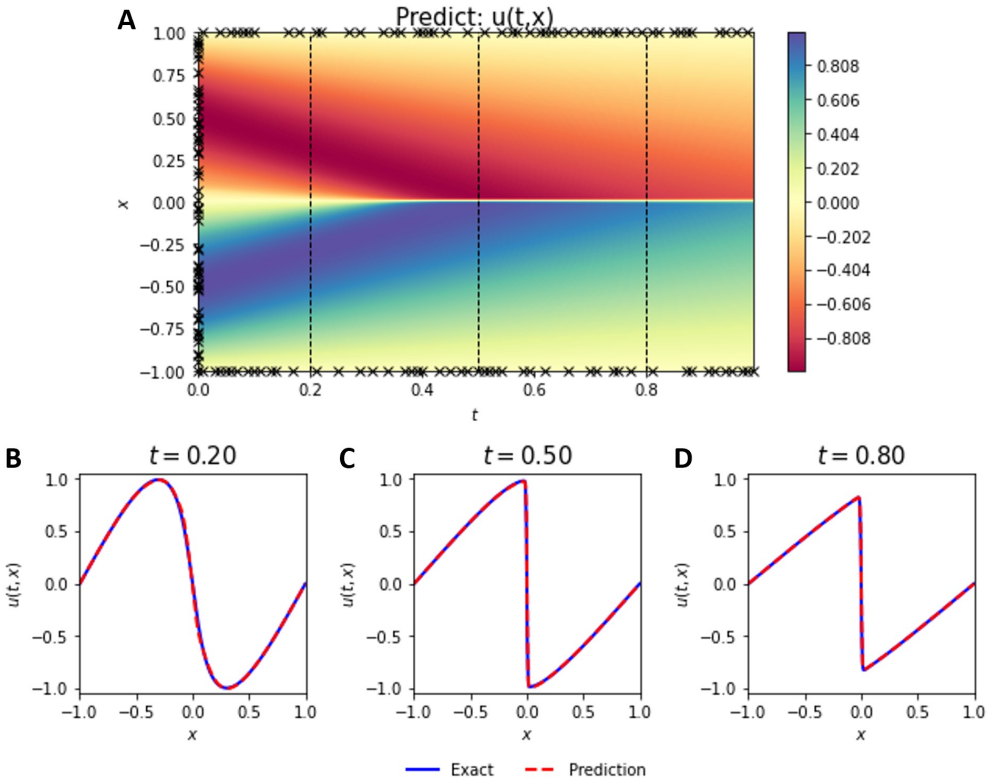


**Figure 3.2** Predictions of the Continuous-time Burgers Equation Inference Problem

A: Plot of the prediction of the solution of the Burgers equation with the initial time and boundary data points marked in black crosses, and the times at which snapshots were taken marked with dashed black lines; B: The prediction of the solution of the Burgers equation plotted against the exact solution of the Burgers equation at time $t$=0.20; C: The prediction of the solution of the Burgers equation plotted against the exact solution of the Burgers equation at time $t$=0.50; D: The prediction of the solution of the Burgers equation plotted against the exact solution of the Burgers equation at time $t$=0.80.

### Continuous-time inference of the Non-linear Schrodinger equation

Analogous to the previous example, given boundary and initial time target data, the solution across the entire domain of the Non-linear Schrodinger equation (2.8) was searched. The mentioned equation has a complex solution $h(t,x) = u(t,x) + iv(t,x)$. For the sake of simplicity all illustrations present only the magnitude of the solutions, $h(t,x)$. The domain was made up of 256 spatial against 201 temporal points, resulting (upon multiplying together) in a total of 51456 evenly spaced points on the basis of which the target solution data was assigned. A plot of the magnitude of the target solution can be found in Fig. 3.3. A data-set of size 150 was used to enforce the initial and boundary conditions, with 50 corresponding to randomly selected points on ei-

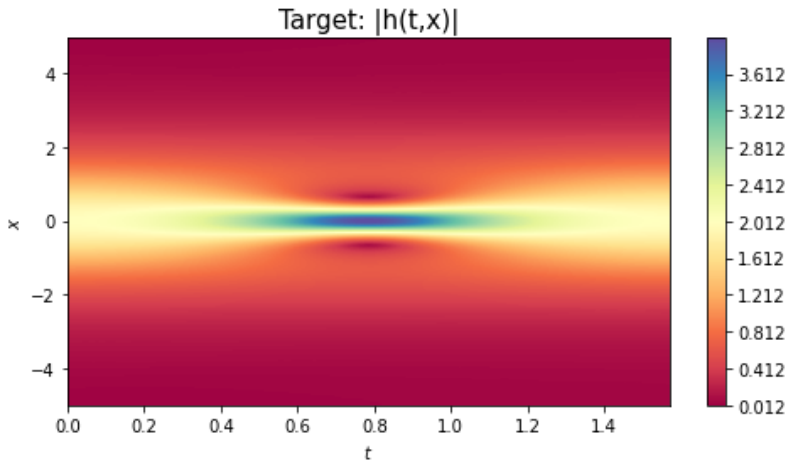


**Figure 3.3** Solution of the Nonlinear Schrodinger Equation

ther boundary and another 50 corresponding to randomly selected initial data points. Note that the provided boundary information does not include the solution to the PDE on the boundaries but depends only on periodic boundary conditions. As a result no boundary target data was used, instead the target data-set was only of a size of 50 points at initial time. The mentioned target data-set is two-dimensional[2] as opposed to the previous example where it is one-dimensional; the first dimension relating to the real and the second relating to the complex targets. The entire 201 by 256 grid of points was used as inputs for the physics informed loss.

The training process lasted in accordance with the default termination tolerance of the L-BFGS optimizing algorithm, and the resulting trained model was used to finally make a prediction of the solution of the Nonlinear Schrodinger Equation. The resulting prediction is plotted in Fig. 3.4 A, with the target data used marked with black crosses. At three snapshots, the prediction is also plotted against the exact solution at the given times in Fig. 3.4 B, C and D. The times at which these snapshots were taken are also marked on Fig 3.4 A using black dashed lines. Errors of the prediction were calculated according to the L2-norm resulting in an absolute error of $7.29 \cdot 10^{-5}$ and a relative error of $9.80 \cdot 10^{-3}\%$.

[2]Note that we consider the target data-set as two dimensional because we treat the complex solution of the PDE in the form $h(t,x) = [u(t,x)\ v(t,x)]$ as already explained in the methodology

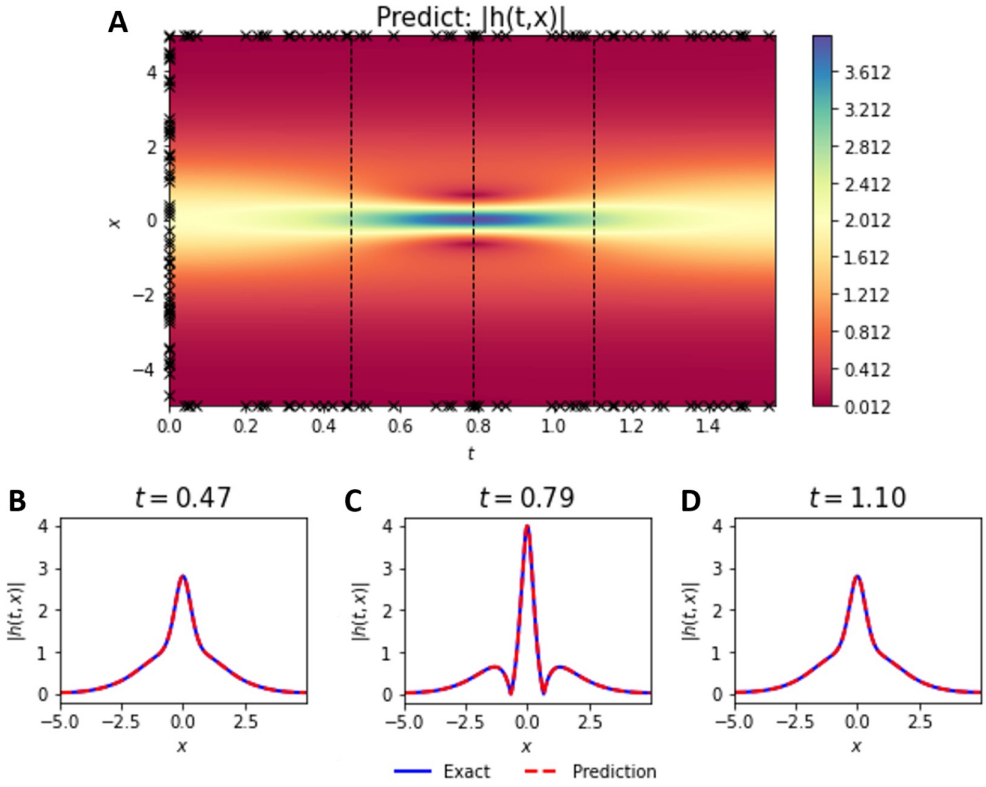


**Figure 3.4** Predictions of the Continuous-time Nonlinear Schrodinger Equation Inference Problem

A: Plot of the magnitude of the prediction of the solution of the Nonlinear Schrodinger equation with the initial time and boundary data points marked in black crosses, and the times at which snapshots were taken marked with dashed black lines; B: The magnitude of the prediction of the solution of the Nonlinear Schrodinger equation plotted against the magnitude of the exact solution of the Nonlinear Schrodinger equation at time $t$=0.47; C: The magnitude of the prediction of the solution of the Nonlinear Schrodinger equation plotted against the magnitude of the exact solution of the Nonlinear Schrodinger equation at time $t$=0.79; D: The magnitude of the prediction of the solution of the Nonlinear Schrodinger equation plotted against the magnitude of the exact solution of the Nonlinear Schrodinger equation at time $t$=1.10.

### 3.1.2 Continuous-time identification results

The continuous-time identification problems are inherently different in approach to the inference problems, in that the used target data-set, spread across the entire domain, is far greater and the PDEs are only partially known. The greatest obstacle, as a result, is not fitting the model to the solutions, but to identify the PDEs from relatively uninformed predictions. Here, what is meant by uninformed is that fitting the model to the target solution can not rely on the smoothing characteristics of inferring PINNs, hence why larger target data-sets are required.

### Continuous-time identification of the Burgers equation

Given the general form of the Burgers equation (2.11) — which is the basis for the physics informing function (2.12) — and target data $u(t, x)$ spread across the entire domain, the goal was to predict the coefficients $\lambda_1$ and $\lambda_2$, which have target values of 1 and $0.01/\pi$ respectively.

Analogous to the earlier described Burgers equation examples, the domain was split into 100 temporal against 256 spatial points, resulting in a high resolution grid of a total of $2.56 \cdot 10^4$ points. Although this grid was used to generate plots in the entire domain, the input points used for both the target data loss and the physics informed loss was restricted to a set of $2 \cdot 10^3$ points, randomly spread across the domain. Solutions of the PDE were assigned to each of these points as target data. The solution of the PDE in question has already been presented in Fig. 3.1.

The training process in accordance with the presented method in subsection 2.1.1 — simultaneously training the parameters of the DNN layers and the PDE parameters themselves — lasted until the L-BFGS methods termination tolerance was reached. After this, the entire grid of points was passed through the network in order to generate illustrations of the corresponding prediction to highlight how well the model was able to fit the solutions. A plot of the fitted solution in the entire domain is presented in Fig. 3.5 A, where the training data points are marked in black crosses (made small to maintain a relatively clear image of the fitted solution), and furthermore three snapshots were taken at times marked with thick black dashed lines. These snapshots Fig. 3.5 B, C and D are plotted against the exact solutions at given times.

The resulting parameters were found to be $\lambda_1 = 0.97$ and $\lambda_2 = 4.62 \cdot 10^{-3}$ $= 0.015/\pi$, with relative errors of 3.15% and 45.26% respectively. The relative error of the second parameter is noticeably large, however, when considering its absolute error of $1.4 \cdot 10^{-3}$, it is in fact less than that of the absolute error of the first parameter which is 0.032. Considering that the loss functions used to train the model's function are on an absolute error basis, these errors are justified and the estimated coefficients for $\lambda_1$ and $\lambda_2$ are to be considered acceptable for the model.

### Continuous-time identification of the Kuramoto-Sivashinsky equation

Given the general form of the 1D Kuramoto-Sivashinsky equation (2.5), the goal was to identify the unknown parameter $\lambda = (\lambda_1, \lambda_2, \lambda_3)$, by incorporating the physics informing function (2.6) relating to the PDE. The plot of the target solution is presented in Fig. 3.6.

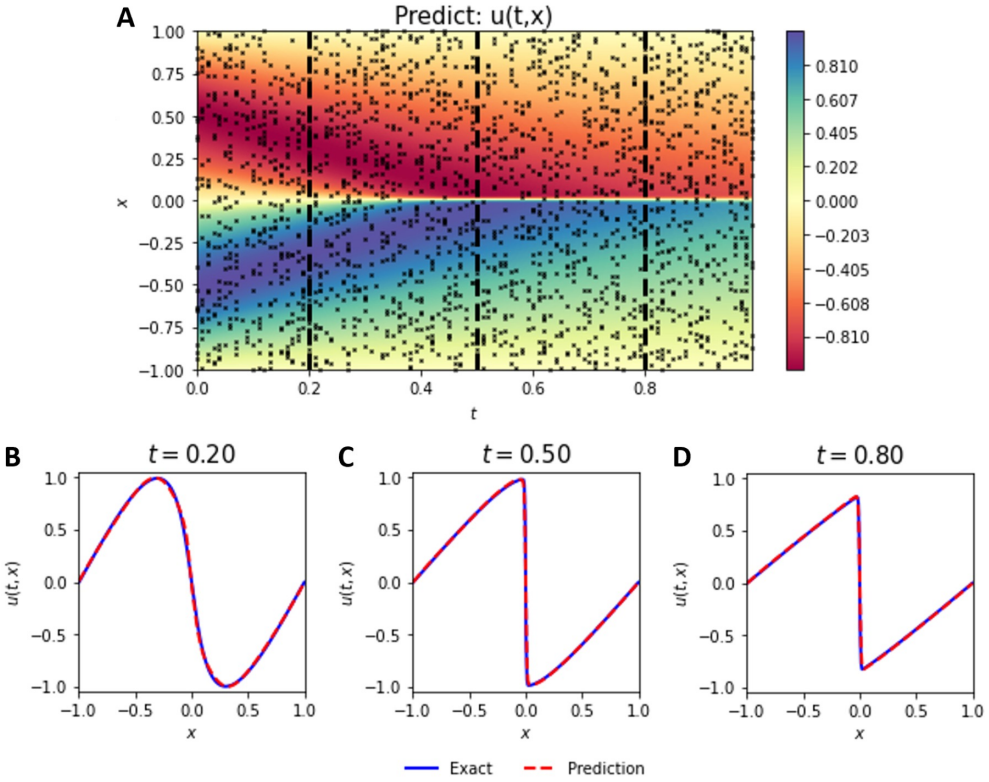


**Figure 3.5** Predictions of the Continuous-time Burgers Equation Identification Problem

A: Plot of the model's fit of the solution of the Burgers equation with the with the target data points of the solution marked with tiny black crosses, and the times at which snapshots were taken marked with thick dashed black lines; B: The model's fit of the solution of the Burgers equation plotted against the exact solution of the Burgers equation at time $t$=0.20; C: The model's fit of the solution of the Burgers equation plotted against the exact solution of the Burgers equation at time $t$=0.50; D: The model's fit of the solution of the Burgers equation plotted against the exact solution of the Burgers equation at time $t$=0.80.

It quickly became clear that when using small data-sets, the proposed method — which was used to solve the same problem in the Burgers equation — failed in fitting the model to the target set and the parameters relating to the unknown coefficients did not develop in a reasonable manner. To explore this behaviour, the training process was expanded and assisted as much as possible. The target data-set was set to include the solutions of the PDE corresponding to every single point in the grid, i.e. for the grid made up of 1000 temporal points against 64 spatial points, a target data-set of size $6.4 \cdot 10^4$ was used. Moreover, the fully connected layers were increased to include 200 neurons in one experiment and 400 neurons in another. In this section, the focus is on the case of using 400 neurons, as it allowed the model to fit the target

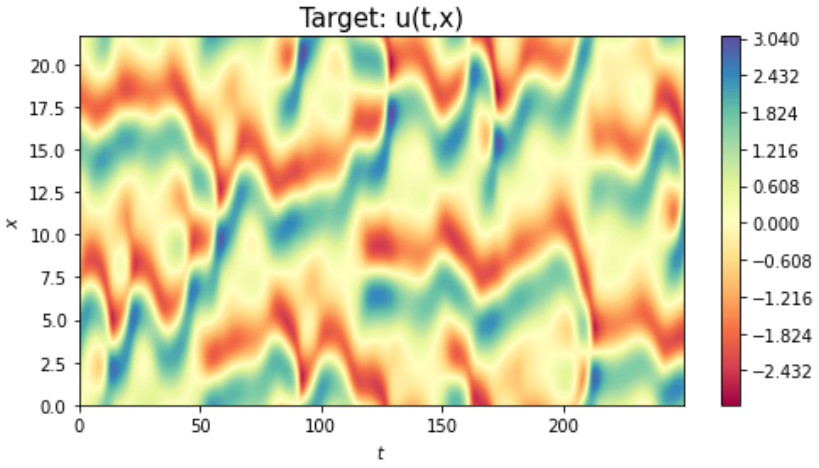


**Figure 3.6** Solution of the Continuous-time Kuramoto-Sivashinsky Equation

with a higher level of accuracy[3]. For interested readers, more on the details of the experiment findings including plots can be found in the Appendix of this paper.

With this new set-up, the model was capable of fitting the target solution to a higher degree, as one would expect, however when compared to the level of fit present in the other examples of this paper, the fit was still very far from ideal. The found absolute and relate errors according to the L2-norm are 0.62 and 64.3% respectively. The predicted coefficients were found to be $\lambda_1 = 0.008$, $\lambda_2 = 0.009$ and $\lambda_3 = 0.001$. However, due to the great fitting errors, this was no surprise.

Following this observation, a new method was implemented to increase the fitting accuracy. The training scheme was altered by first replacing the L-BFGS optimizer with a simple Adam optimizer with a step-based learning rate scheduler with a batch size of 64, set to stop learning after $5 \cdot 10^4$ iterations, i.e. after 50 epochs. It was found that although this method did increase the fitting accuracy, resulting in absolute and relative errors of 0.122 and 12.5%, according to the L2-norm, the predictions of the unknown PDE coefficients were not sufficiently reasonable ($\lambda_1 = 5.28 \cdot 10^{-2}$, $\lambda_2 = -1.19 \cdot 10^{-2}$ *and* $\lambda_3 = -0.03 \cdot 10^{-2}$).

As a final test, the two optimizers were combined in the following manner: first the model is trained using the previously defined Adam scheme, after which the L-BFGS optimizing scheme is applied according to the method used in the other examples in this paper. The idea behind incorporating the L-BFGS

[3] 400 neurons per layer was the limit at which these experiments could be performed governed by the memory of the GPU implemented

optimizing scheme after the Adams scheme was to both fine tune the fitting of the model and to see if in the case of a relatively good fit, the L-BFGS method would be able to extract a more accurate estimation of the unknown coefficients of the model. Note that for this particular experiment, the training process was manually terminated after over 6 hours of training, when it was clear that the model was not developing in a meaningful manner.

A plot of the fitted solution in the entire domain is presented in Fig. 3.7 A, where the dashed lines represent times at which snapshots were taken. The mentioned snapshots of the fitted predictions are plotted against the target solutions at the corresponding times in Fig. 3.7 B, C and D. It is apparent that the model was able to fit the target to a relatively high degree, comparable to that of the corresponding example treating the Burgers equation. The absolute and relative errors according to the L2-norm were found to by 0.009 and 9.30% respectively. Despite this, although the parameters of the PINN corresponding to the unknown coefficients did present a slightly greater divergence from their initial values of zero than in the other experiments, the predicted coefficients: $\lambda_1 = 0.093$, $\lambda_2 = -0.055$ *and* $\lambda_3 = -0.001$ were still not reasonably representative of the target.

## 3.2 Results of examples in discrete-time

### 3.2.1 Discrete-time inference results

#### Discrete-time inference of the Burgers equation

As opposed to the continuous-time case, the goal here was to infer the solution of the Burgers equation presented in (2.5) at a specific time $t^{n+1}$, given the boundary conditions presented and the solution to the problem at time $t^n$. In this example, the time-steps we looked at were $t^n = 0.10$ and $t^{n+1} = 0.90$, which — according to the method presented in section 2.2.1 — resulted in a time-step size of $\Delta t = 0.8$ where the general form of the Runge-Kutta method was applied with $q = 500$ stages. For the physics informed loss, we used a data-set of 250 points randomly spaced across the spatial domain at the first time step $n$ to which the corresponding solutions were assigned, while for the boundary loss, the target data-set was limited to one point on either boundary at time step $n + 1$. The training process was according to the L-BFGS algorithm, with the default termination tolerances.

The plot of the solution to the Burgers equation in the entire spatial and temporal domain is plotted in Fig. 3.6 A, where, with dashed black lines, the time-steps involved in the training process are illustrated. After the training

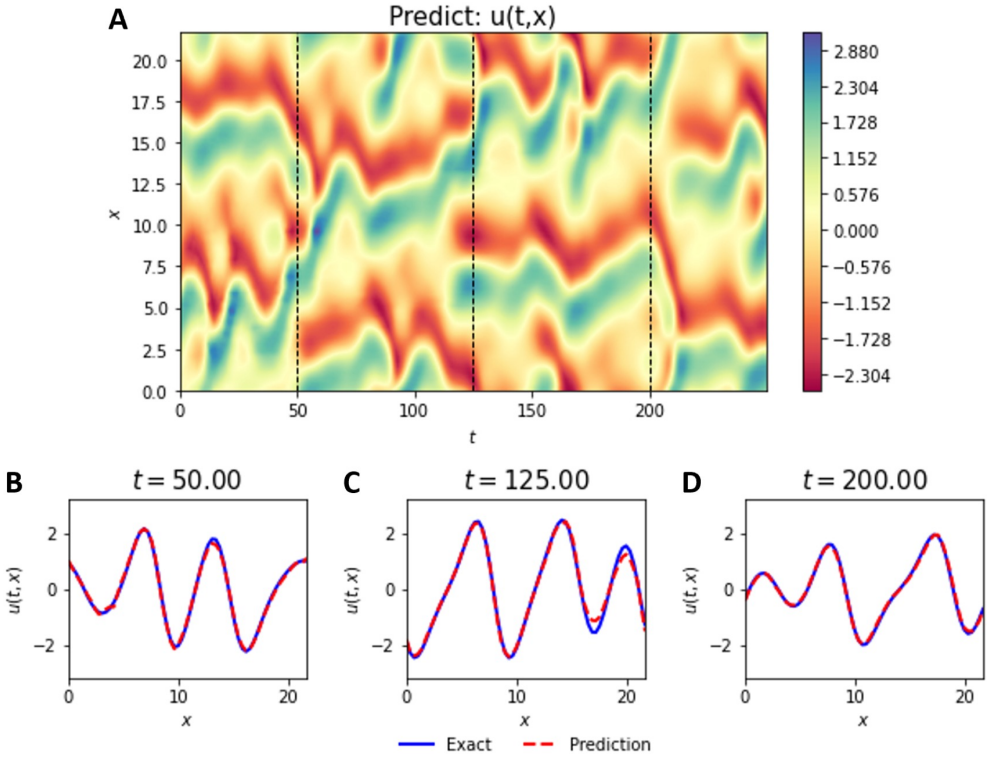


**Figure 3.7** Predictions of the Continuous-time Kuramoto-Sivashinsky Equation Identification Problem

A: Plot of the model's fit of the solution of the Kuramoto-Sivashinsky equation with the times at which snapshots were taken marked with thick dashed black lines; B: The model's fit of the solution of the Kuramoto-Sivashinsky equation plotted against the exact solution of the Kuramoto-Sivashinsky equation at time $t$=50; C: The model's fit of the solution of the Burgers equation plotted against the exact solution of the Burgers equation at time $t$=125; D: The model's fit of the solution of the Burgers equation plotted against the exact solution of the Burgers equation at time $t$=200.

process was completed, a final prediction of the solution at time-step $n+1$ was made. Plots of the previously mentioned time-steps are presented in Fig. 3.6 B and C where the exact solutions are plotted against the provided data and plotted against the predicted solution respectively. The absolute and relative errors of the prediction according to the L2-norm were found to be $1.24 \cdot 10^{-8}$ and $3.13 \cdot 10^{-6}\%$ respectively.

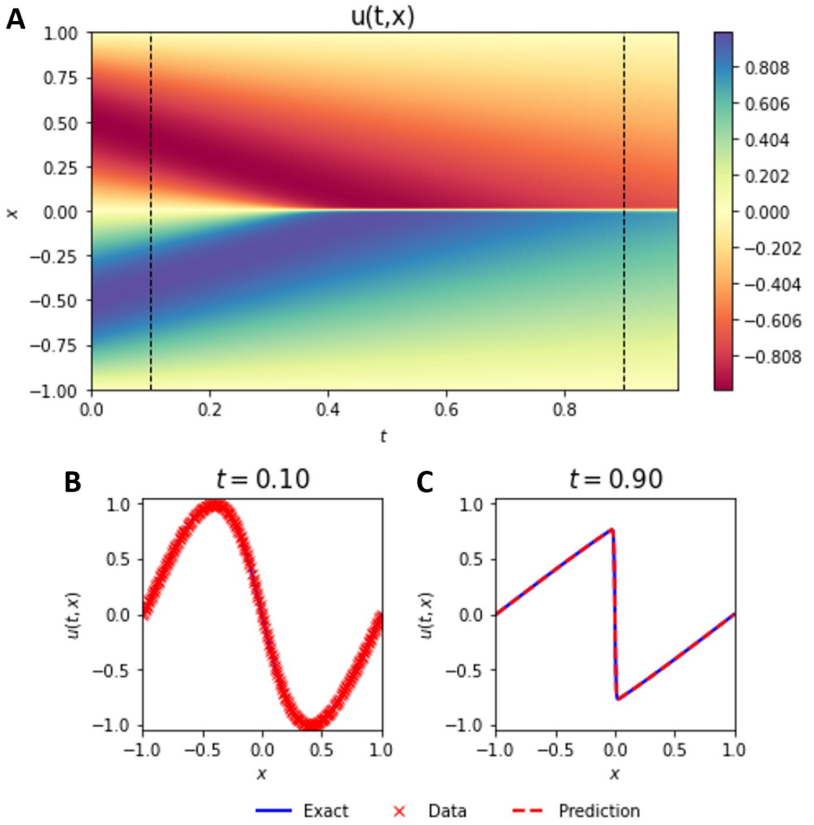


**Figure 3.8** Predictions of the Discrete-time Burgers Equation Inference Problem

A: The solution of the Burgers equation with the times steps that were treated in this discrete-time inference problem marked with dashed black lines; B: The target data points of the solution of the Burgers equation plotted against the exact solution of the Burgers equation at time $t = 0.10$; C: The prediction of the solution of the Burgers equation plotted against the exact solution of the Burgers equation at time $t = 0.90$.

### 3.2.2 Discrete-time identification results

#### Discrete-time identification of the Burgers equation

As in the previous problem of inference, in this example we studied only the times $t^n = 0.10$ and $t^{n+1} = 0.90$ at time-steps $n$ and $n+1$ respectively with the time step size $\Delta t = 0.8$. Using target data at both time-steps randomly spaced across the spatial domain, the goal was to predict the unknown coefficients $\lambda_1$ and $\lambda_2$ of the PDE (2.11), which have correct values of 1 and $0.01/\pi$ respectively. By incorporating these coefficients as parameters in the PINN, these parameters were modified in parallel to the layers of the DNN, using the L-BFGS optimizing algorithm.

The training was performed according to the method description in section 2.2.2, where the general form of the Runge-Kutta method was applied to the PDE with $q = 81$ stages. As opposed to the previous example of inference, in this method, two loss functions (2.22) of the same form were used to train the model. By incorporating the physics informing function (2.21) extracted from the partially known PDE, the first loss function calculated the error between the target data and the PINNs prediction[4] of the PDE solution at time-step n while the second loss function calculated the same at time-step $n + 1$. The target data-sets used had sizes of 199 and 201 respectively, at randomly spaced points across the spatial domain.

After the training was completed, the solutions at the time-steps in question were predicted by the trained model to get an idea of how well the model was able to fit the target solutions. The trained parameters of the unknown coefficients were also extracted from the PINN. The solution of the PDE in the entire domain is plotted in Fig. 3.7 A where the time-steps $n$ and $n + 1$ are marked with dashed black lines. At the corresponding times, the models predictions were plotted against the exact solutions in Fig. 3.7 B and C. The solutions were found to be $\lambda_1 = 0.998$ and $\lambda_2 = 3.70 \cdot 10^{-3} = 0.012/\pi$ with absolute errors of $2.30 \cdot 10^{-3}$ and $0.51 \cdot 10^{-3}$, and relative errors of 0.23% and 16.12% respectively. The difference in relative errors can be overlooked when considering the absolute errors, as already mentioned in the continuous-time identification problem of the same Burgers equation. The result is supported by the fact that the loss functions operate on an absolute error basis.

A final observation, which is highlighted by the provided plots, is the error of the fitting of the solution of the model, such that the comparison between different identification examples can be made easier. According to the L2-norm, the absolute and the relative errors were found to be $5.61 \cdot 10^{-6}$ and $5.48 \cdot 10^{-4}$% respectively.

### Discrete-time identification of the Kuramoto-Sivashinsky equation

The method used to tackle this problem, presented in section 2.2.2, is analogous to the corresponding example of the Burgers equation. The time-steps $n$ and $n + 1$ studied were at the times $t^n = 81.20$ and $t^{n+1} = 81.60$ respectively, with a time step size of $\Delta t = 0.4$. The goal here was to predict the unknown coefficients $\lambda_1$, $\lambda_2$ and $\lambda_3$ from the partially known PDE (2.15), each of which had correct values of 1.

In this example, the Runge-Kutta method was applied to the PDE with $q = 500$ stages and the same loss function (2.22) was used as in the previ-

[4]Note that the PINNs prediction incorporates not only the DNNs prediction but also the physics informing function and therefore the unknown parameters.

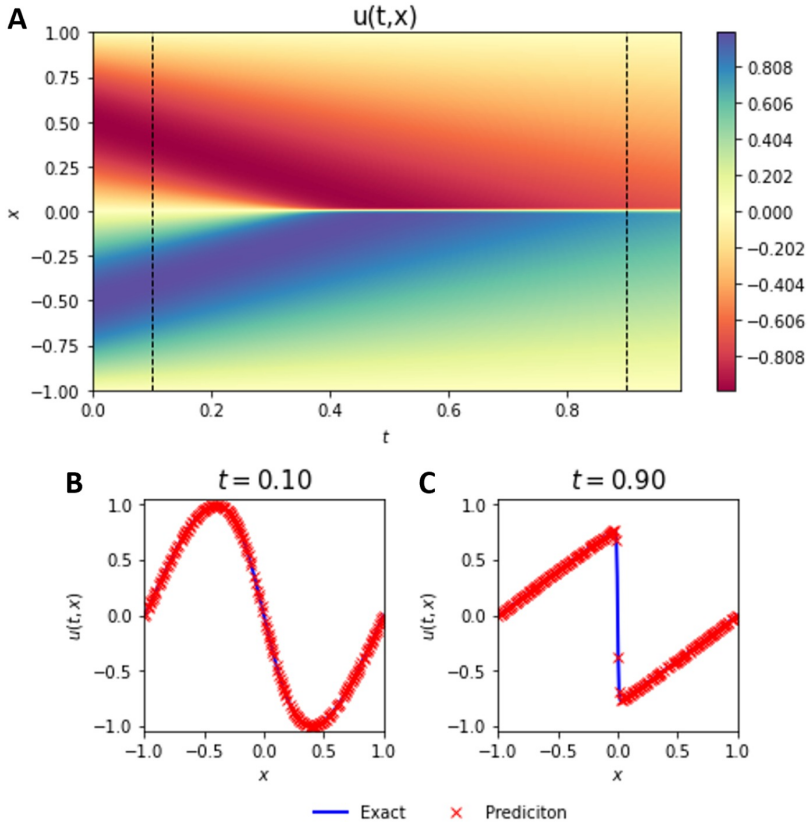


**Figure 3.9** Predictions of the Discrete-time Burgers Equation Identification Problem

A: The solution of the Burgers equation with the times steps that were treated in this discrete-time identification problem marked with dashed black lines; B: Plot of the model's fit of the solution of the Burgers equation plotted against the exact solution of the Burgers equation at time $t$=0.10; C: Plot of the model's fit of the solution of the Burgers equation plotted against the exact solution of the Burgers equation at time $t$=0.90.

ous example, while (2.23) is physics informing function incorporated in the PINN. Like in the example of continuous-time identification of the Kuramoto-Sivashinsky equation, it quickly became clear that the PINN framework struggled to solve the problem, and so both the PINN architecture and the target data-set size was increased. At time steps $n$ and $n+1$, target data-set sizes of 1000 and 1001 were used respectively, while the number of neurons per fully connected layer in the DNN were increased from 100 to 600[5].

After the training was completed using the L-BFGS optimizer, the fit of the solutions at the time steps were plotted and the trained parameters of the

[5]Earlier it was stated that 400 neurons was close to the limit of what the system could bare. Given the slight difference in application, here the number of neurons could be increased even further.

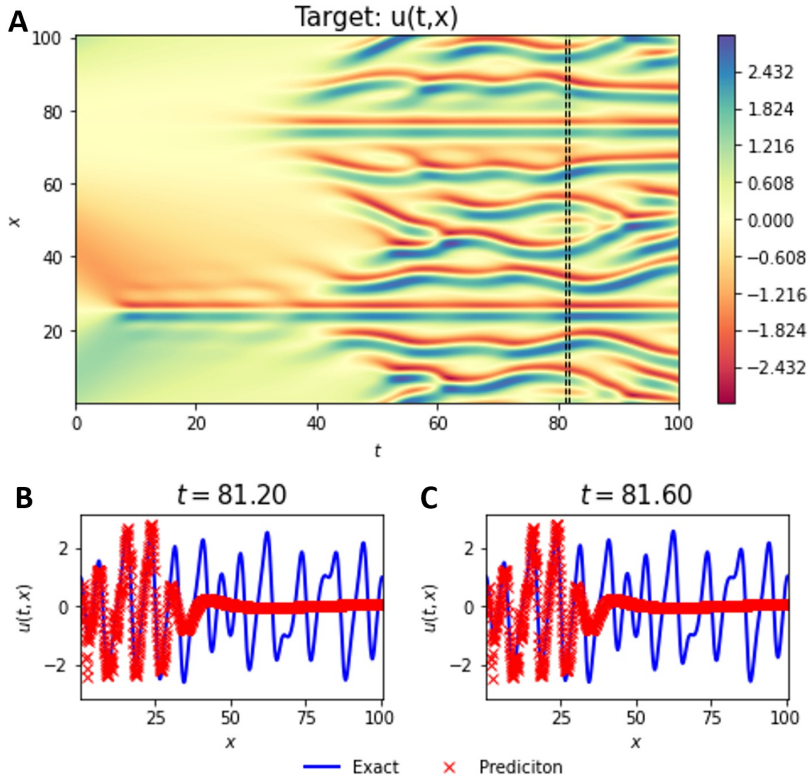


**Figure 3.10** Predictions of the Discrete-time Kuramoto-Sivashinsky Equation Identification Problem

A: The solution of the Kuramoto-Sivashinsky equation with the times steps that were treated in this discrete-time identification problem marked with dashed black lines; B: Plot of the model's fit of the solution of the Kuramoto-Sivashinsky equation plotted against the exact solution of the Burgers equation at time $t$=0.10; C: Plot of the model's fit of the solution of the Kuramoto-Sivashinsky equation plotted against the exact solution of the Burgers equation at time $t$=0.90.

unknown coefficients were extracted from the PINN. A plot of the target solution is plotted in Fig. 3.10 A, where the time-steps treated are marked with dashed black lines. The solution predictions at the corresponding times are plotted against the target solutions in Fig. 3.10 B and C. It was found that the summed absolute and relative errors of fit at times $t^n$ and $t^{n+1}$ were 2.43 and 100.2% respectively, although even from the plots presented it is clear that the model failed to fit the target solution. The resulting coefficients $\lambda_1$, $\lambda_2$ and $\lambda_3$ were found to be non reasonable: 0.058, 0.001 and $2.26 \cdot 10^{-7}$ respectively, although given that the model was not able to fit the solution, this is an expected outcome.

# Chapter 4

# Conclusion

In this paper, a number of applications of PINNs were presented, tackling a range of problems and treating various PDEs modeling physical phenomena with relatively simple architectures. Applied to the 1D Burgers equation, the introduced methods were able to solve problems of inference and identification in both continuous- and discrete-time effectively, making predictions with relatively low errors according to the L2-norm. The PINN frameworks were then adjusted and applied to the continuous-time inference problem of the 1D Nonlinear Schrodinger equation and to both the continuous- and discrete-time identification problems of the 1D Kuramoto-Sivashinsky equation.

By enforcing the initial and boundary information and the physical constraints in the loss function of the training scheme, the solution of the Burgers equation, a second order PDE, was inferred in continuous-time with a relatively high level of accuracy. As presented in section 3.1.1, the relative error of the inferred prediction was found to be only 0.027% (according to the L2-norm). Considering the target data-set size of only 150 points and the prediction relating to over $5 \cdot 10^4$ input points spread across the spatio-temporal domain, this application, where purely data driven methods would struggle, highlights the capabilities of PINNs.

The same method was applied to the Nonlinear Schrodinger equation, which stands out among the other PDEs in this paper in that it has a two-dimensional solution as opposed to just one. Similarly to the case of its counterpart, the solution of the Nonlinear Schrodinger equation was predicted with a relatively high level of accuracy with a relative error of $9.8 \cdot 10^{-3}$% (according to the L2-norm; see section 3.1.1). It is worth noting that for this example of inference, even less data was used (a total of only 50 target data points) and still the resulting error was low.

The example of identification of the Burgers equation is only marginally different to that of inference in continuous-time. By incorporating the unknown coefficients of the PDE as parameters in the PINNs, the loss function used to train the models remains the same. For case of continuous-time identification

of the Burgers equation with target coefficients of 1 and $0.001/\pi$, the predicted coefficients were found to be 0.98 and $0.0015/\pi$. Using a target data set of $2 \cdot 10^3$ points spread randomly across the spatio-temporal domain, the model was able to fit the target PDE to a relatively high degree with a relative error of only 0.07% (according to the L2-norm). As the physics informing function greatly depends on the level of accuracy of the fit of the model, this suggests why the predictions of the coefficients was so successful. The PINN framework was then adjusted and applied to the Kuramoto-Sivashinsky equation. As already mentioned in sections 2.1.2 and 3.1.2, the training needed to be modified since initially the generated model was unable to fit the solution. After introducing a training scheme that implemented both the Adams and L-BFGS optimizer the correct coefficients could still not be extracted at a reasonable level of accuracy, despite that the model was able to fit the solution with a relative error of 0.93% (according to the L2-norm), which was considered acceptable.

Similar results were observed in the discrete-time identification examples. Although the introduced schemes were able to effectively predict the unknown coefficients of the Burgers equation (with absolute errors of $2.30 \cdot 10^{-3}$ and $0.51 \cdot 10^{-3}$ according to the L2-norm), the adjusted framework failed to make reasonable predictions when applied to the Kuramoto-Sivashinsky equation.

The PINN frameworks were effectively applied to the problems of inference and identification when treating second order PDEs, highlighting their ability to solve problems with minimal amounts of data and their ability to make accurate predictions (according to the L2-norm) in settings where their purely data driven counterparts would struggle. Furthermore, the proposed PINN frameworks could be effectively applied to different problems — without much modification — involving the Burgers equation in both continuous- and discrete-time cases and to a problem involving the Nonlinear Schrodinger equation, highlighting the robustness of the applications of DNNs. The same can not be said about the application of the PINNs to the problems concerning the Kuramoto-Sivashinsky equation, however. Being a fourth order PDE, it introduces a much higher level of complexity than the other PDEs treated in this paper. Although PINNs present many interesting features, implementing them in more complex examples, where chaotic dynamics are present, will require further research. We believe that apart from increasing the depth of the network or increasing the sizes of the layers even further, future research should also investigate the effects of more sophisticated NN architectures. Incorporating long short-term memory networks, dimensionality reduction or autoencoders into the architecture of the PINNs might prove to be important for future refinements. In addition, comparing the data-efficiency of PINNs with networks utilized in other works [1, 8, 13, 14, 15] might be an interesting matter of future analysis.

# Chapter 5

# Appendix

The appendix accompanies the continuous-time identification of the Kuramoto-Sivashinsky equation example in section 3.1.2. Here, findings of some experiments are presented which aimed to investigate the behaviour of PINN methods when treating fourth-order PDEs. In each experiment either the fully connected layer sizes or the optimizing schemes were varied and corresponding final prediction illustrations, identified coefficients and errors are compared. Note that although the fully connected layer sizes were modified, the PINN architectures incorporating four-layer-deep NNs were not varied in any other way.

The experiments are as follows:

(A) L-BFGS optimizing scheme with 200 neurons per layer

(B) L-BFGS optimizing scheme with 400 neurons per layer

(C) Adam optimizing scheme with 200 neurons per layer

(D) Adam optimizing scheme with 400 neurons per layer

(E) Adam + L-BFGS optimizing schemes with 400 neurons per layer

The letters assigned to each experiments will be used to describe each experiment from here on for simplicity. Furthermore, note that certain findings of experiments B, D and E have have already been presented in section 3.1.2.

After the training process, the unknown coefficients of the Kuramoto-Sivashinsky equation were predicted. These predictions are presented in Table 5.1 in their PDE forms, and are compared to the target PDE. As already highlighted in the methodology section, the coefficients are predicted by including them as parameters in the PINNs, and are modified in parallel to the model as it tries to fit the known target solution. For each experiment, the absolute and relative errors of the fit of the model (according to the L2-norm) are provided in

| Target | $u_t + uu_x + u_{xx} + u_{xxxx} = 0$ |
|---|---|
| Experiment A | $u_t + 0.038 \cdot uu_x - 0.006 \cdot u_{xx} - 0.001u_{xxxx} = 0$ |
| Experiment C | $u_t + 0.008 \cdot uu_x + 0.009 \cdot u_{xx} + 0.001 \cdot u_{xxxx} = 0$ |
| Experiment B | $u_t + 0.045 \cdot uu_x - 0.004 \cdot u_{xx} - 5.2 \cdot 10^{-7} \cdot u_{xxxx} = 0$ |
| Experiment D | $u_t + 0.052 \cdot uu_x - 0.012u_{xx} - 3.0 \cdot 10^{-4} \cdot u_{xxxx} = 0$ |
| Experiment E | $u_t + 0.093 \cdot uu_x - 0.055 \cdot u_{xx} - 0.001 \cdot u_{xxxx} = 0$ |

**Table 5.1** Predictions of the individual experiments of the Kuramoto-Sivashinsky continuous-time identification problem

| Experiment | Absolute Fit Error | Relative Fit Error |
|---|---|---|
| A | 0.642 | 65.7 % |
| B | 0.619 | 63.4% |
| C | 0.242 | 24.8 % |
| D | 0.122 | 12.5 % |
| E | 0.009 | 0.93 % |

**Table 5.2** Errors of the individual experiments of the Kuramoto-Sivashinsky continuous-time identification problem

Table 5.2. For each experiment, to illustrate the just now mentioned errors, three snapshots were taken of the fitted models at three different times. The snapshots are marked in Figure 5.1 — illustrating the target solution of the PDE — with dashed black lines. The corresponding snapshots are plotted against the PDE solutions in Figures 5.2 A to E.

In this example, treating a case with a large data-set, it is notable that that the Adam optimization scheme lead to the model fitting the target solutions with far greater accuracies than the L-BFGS scheme[1]. Furthermore, while the L-BFGS optimizing scheme lasted a number of hours, the Adam optimizing scheme lasted no more than tens of minutes. Despite this, more experiments would have to be performed to be able to draw any conclusions, because despite higher fitting accuracies, the Adam optimizing scheme did not lead to better results in the identification of the PDE. The same can be said about the combination of the Adam and the L-BFGS optimizing scheme, although the model was able to fit PDE solutions with even higher accuracies.

[1]Note that in the methodology section 2 of this paper it was claimed that the L-BFGS optimizing scheme is more appropriate for treating cases with smaller data-sets

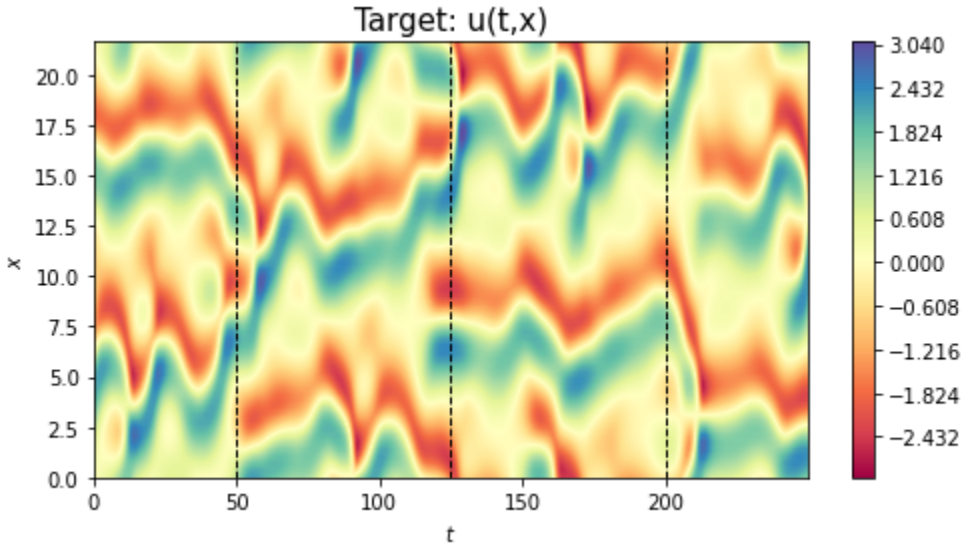


**Figure 5.1** Solution of the Kuramoto-Sivashinsky Equation

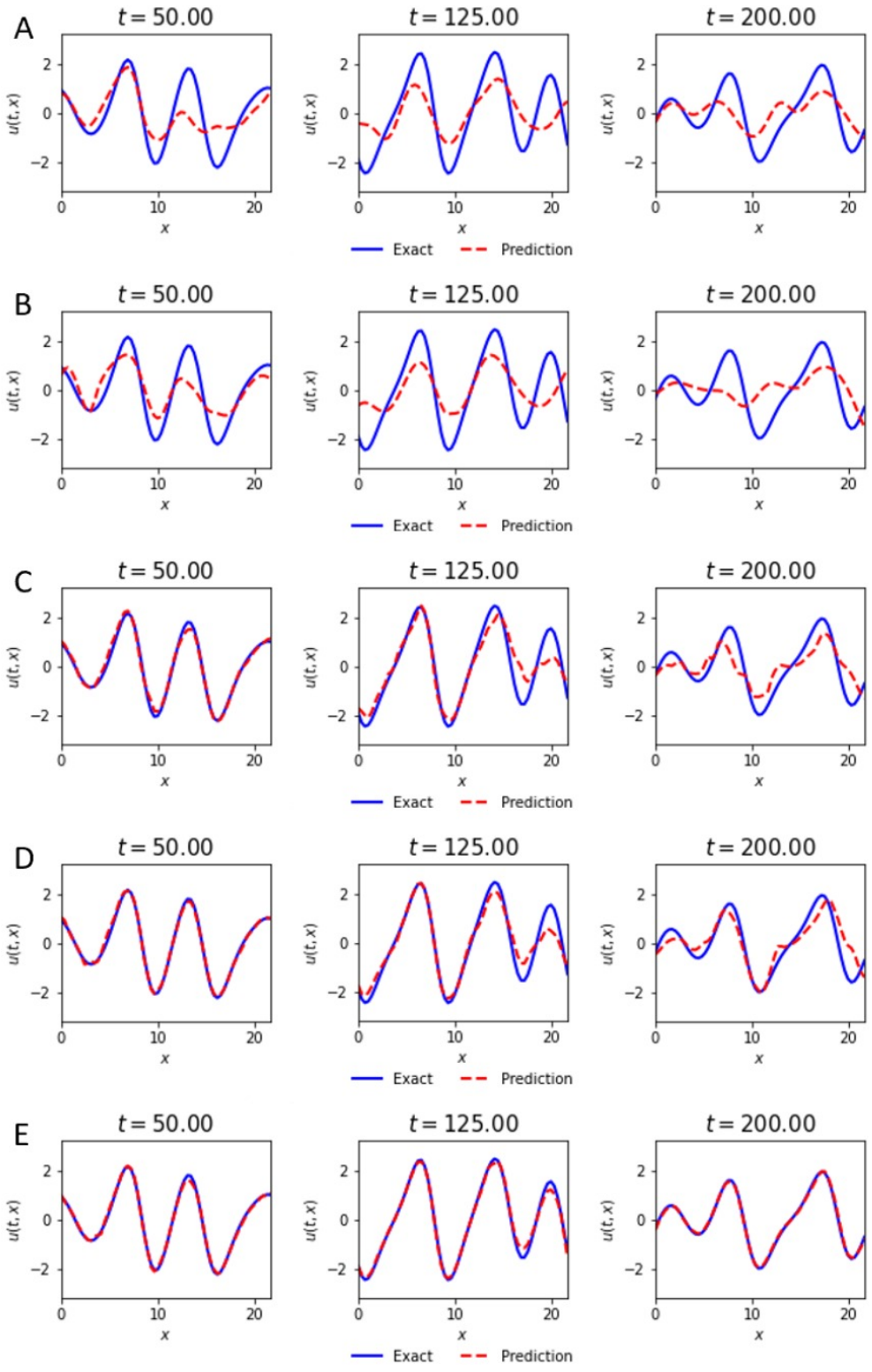


**Figure 5.2** Predictions of the Discrete-time Kuramoto-Sivashinsky Equation Identification Problem

# Bibliography


[1] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.

[2] Fei-Yue Wang, Jun Jason Zhang, Xinhu Zheng, Xiao Wang, Yong Yuan, Xiaoxiao Dai, Jie Zhang, and Liuqing Yang. Where does alphago go: From church-turing thesis to alphago thesis and beyond. *IEEE/CAA Journal of Automatica Sinica*, 3(2):113–120, 2016.

[3] Noam Brown and Tuomas Sandholm. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

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

[5] Raban Iten, Tony Metger, Henrik Wilming, Lídia Del Rio, and Renato Renner. Discovering physical concepts with neural networks. *Physical Review Letters*, 124(1):010508, 2020.

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

[7] Ryan King, Oliver Hennigh, Arvind Mohan, and Michael Chertkov. From deep to physics-informed learning of turbulence: Diagnostics. *arXiv preprint arXiv:1810.07785*, 2018.

[8] N Benjamin Erichson, Michael Muehlebach, and Michael W Mahoney. Physics-informed autoencoders for lyapunov-stable fluid flow prediction. *arXiv preprint arXiv:1905.10866*, 2019.

[9] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

[10] Dongkun Zhang, Lu Lu, Ling Guo, and George Em Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics*, 397:108850, 2019.

[11] Yinhao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, and Paris Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394:56–81, 2019.

[12] Dongkun Zhang, Ling Guo, and George Em Karniadakis. Learning in modal space: Solving time-dependent stochastic pdes using physics-informed neural networks. *SIAM Journal on Scientific Computing*, 42(2):A639–A665, 2020.

[13] Pantelis R Vlachas, Wonmin Byeon, Zhong Y Wan, Themistoklis P Sapsis, and Petros Koumoutsakos. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2213):20170844, 2018.

[14] Zhong Yi Wan, Pantelis Vlachas, Petros Koumoutsakos, and Themistoklis Sapsis. Data-assisted reduced-order modeling of extreme events in complex dynamical systems. *PloS one*, 13(5):e0197704, 2018.

[15] Pantelis R Vlachas, Jaideep Pathak, Brian R Hunt, Themistoklis P Sapsis, Michelle Girvan, Edward Ott, and Petros Koumoutsakos. Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Networks*, 2020.