Effective Data-Driven Models for Chaotic and Turbulent Dynamics

by
Xiaoqian Chen

A Dissertation submitted to the Department of Mathematics,

College of Natural Sciences and Mathematics

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in Mathematics

Chair of Committee: Timofeyev Ilya

Committee Member: Ott William

Committee Member: Mang Andreas

Committee Member: Balu Nadiga

University of Houston
May 2021

DEDICATION/EPIGRAPH

I dedicated this dissertation to my family. I also would like to thank the committee members who were very generous with their expertise and precious time. A special thanks to Dr. Ilya Timofeyev, the committee chairman, for his countless hours of reflecting, reading, encouraging, and most of all patience throughout the entire process. I would like to thank Dr. William Ott, Dr. Andreas Mang, and Dr. Balu Nadiga for scientific discussions and agreeing to serve on my committee. I am also grateful to my colleagues Jeric Alcala and Wilfredo Molina for the cherished time spent together and for various discussions. My appreciation also goes out to my family for nursing me with affections and love. Finally, I would like to express thanks to my fluffy cat, Nity, who has been a constant source of support and accompanied me during challenges of graduate school. I am truly thankful for having you in my life.

ABSTRACT

Recent advances in computing algorithms and hardware have rekindled interest in developing high accuracy, low-cost reduced models for simulating complex physical systems. Such data-driven models allow us to overcome several difficult practical issues, such as (a) extreme computational requirements of direct numerical simulations of complex partial differential equations exhibiting multiple temporal and spatial scales; (b) insufficient data or unknown parameters characterizing sub-processes; and (c) no knowledge of the governing subsystem of equations.

In the case of (a)(b), the form of the dynamics is often at least partially known and it is possible to parametrically estimate a reasonable expression that describes the time evolution of the dynamical process. In the first part of this dissertation, we present the simulated least absolute shrinkage and selection operator (SLASSO), a statistical regression method, to select essential parameters from an assumed efficient stochastic model and obtain corresponding parameter estimations. Our developed approach is particularly suitable for data generated by large-scale variables in multiscale systems. The SLASSO estimator overcomes the issue of a large discretization error for data-sets sub-sampled with a large observational time step. In contrast with traditional Maximum Likelihood Estimators and LASSO estimators for stochastic differential equations, the SLASSO approach is able to correctly select the model that fits the data sub-sampled with a large observational time step. We illustrate this approach on the multiscale additive triad model and the discretized Burgers model. Statistical properties of the slow/large scale variable are reproduced well for the additive triad model. However, the SLASSO estimator only produces adequate results when one has good knowledge for the mathematical expression of the dynamics. In addition, similar to many other parametric approaches, the SLASSO estimator is prone to overfitting.

In the case of (a)(b)(c), the goal is to find a reduced model that replaces expensive direct numerical simulations of complex partial differential equations at fine spatial and temporal scales. In addition, it is often desirable to use observational data to infer a reduced model. To this end, we construct a reduced model using machine-learned surrogates that efficiently and accurately forecast trajectories and ensemble properties of the underlying system. In particular, we develop Reservoir Computing for accurate prediction of dynamical systems. Such models need to be trained on a

reasonably large dataset generated by "true" dynamics, but later can be used for a fast trajectory forecast in an initial-value problem. We apply our approach to the one-dimensional shallow water equations, which is a classical model in fluid dynamics. We assume that we only know a finite set of samples of this system without knowledge of the mathematical structure of the shallow water systems.

We introduce a data-driven approach: Echo State Network (ESN). This network can be trained by a fast, stable and simple training algorithm via regularized linear regression. The ESN is able to efficiently and accurately forecast future system states of the shallow water equations (SWE). Moreover, we demonstrate that the ESN outperforms polynomial regression and is robust with respect to perturbations of the initial data. We illustrate the performance of our algorithms through extensive experiments. We also introduce the transfer learning method which is a fast and effective technique for utilizing similarity between different SWE trajectories while also taking into account their differences.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

Recent advances in computational algorithms and hardware have rekindled interest in developing high accuracy, low-cost surrogate models for simulating physical systems. Solving complex partial differential equations (PDEs) representing multiscale processes is computationally prohibitive for many practical applications. The idea is to replace expensive direct numerical simulations (DNS) of complex PDEs with fine-scale space-time resolution with machine-learned surrogates that efficiently and accurately capture future system coarse states. Many strategies for derivation and of reduced order models have been developed, but many important questions still remain unanswered. In this dissertation, we address two questions: estimation of stochastic differential equations from temporal multiscale data and applicability of machine learning techniques for forecasting trajectories of partial differential equations.

When the form of the right-hand side of the underlying equations is at least partially known, such as the functional form of the PDEs or ODEs (up to some unknown parameter values), one can employ parametric techniques for estimating the right-hand side of the equations. Knowledge of the functional form of the right-hand side of the equations is a particularly reasonable assumption for datasets generated by direct numerical simulations of PDEs. In such cases, one might have a good idea about the functional form of the right-hand side for some coarse variables, but parameter values remain unknown.

First, we develop an extension of the maximum likelihood estimator for parametric estimation of right-hand sides of stochastic differential equations from multiscale data. In particular, we develop the simulated least absolute shrinkage and selection operator (SLASSO), a statistical regression method, to select essential variables from an assumed model and obtain appropriate parameter values. We develop our approach in the context of *Indirect Observability* when the available data was generated by an approximating process. In our case, we consider multiscale system and assume that only slow variables are observed. However, since data is generated by a multiscale model, available data has a reasonably strong signature of fast degrees of freedom at small time steps.

Thus, we develop the SLASSO estimator, which can overcome the difficulty of using the multiscale data and correctly estimate the effective model for slow variables.

Another popular approach that emerged recently is completely data-driven: it assumes that Only observational data are available without knowledge of the mathematical structure of the dynamical systems. In the past few years, there have been rapid algorithmic advances in machine learning, particularly in data-driven modeling. Many approaches have been proposed with the ultimate goal of predicting the trajectories of nonlinear dynamical systems. Popular approaches in Machine Learning for trajectory prediction are recurrent neural networks (RNNs) (e.g. Reservoir Computing) and Long Short-Term Memory networks (LSTM). Most machine learning approaches work as a "blackbox", where the exact form of the equations is not utilized directly. However, there is sufficient evidence that these approaches perform very well for many dynamical systems.

1.1 Model Selection under *Indirect Observability*

In many practical problems, it is desirable to model the dynamical features of an observed dataset by carefully choosing effective stochastic models. In particular, selecting and estimating parameters of a system of stochastic differential equations from a discrete dataset of observations has been a very popular technique in the applied dynamical model analysis. The goal of such approaches is to estimate an effective model that reproduces key quantities of interest such as mean, variance, correlation function, etc.

A considerable amount of existing work [80, 87, 100, 88, 20, 52] addresses the issue of estimating effective stochastic models under *Direct Observability* when the data used for estimation is assumed to be generated by the model itself. In this dissertation, we address the question of estimation of effective stochastic models from stationary time-series under *Indirect Observability*.

Recently, *Indirect Observability* [7, 64, 66, 17] and "inexact computing" [63] received increased attention. *Indirect Observability* occurs in many practical situations when there is a mismatch between the available data and the dynamics to be estimated. In particular, the available data can be generated by a high-dimensional and/or multiscale system, and the estimated effective model

describes the homogenized dynamics of large-scale variables. Also, available data might correspond to a slow subset of dynamical variables, and the fast variables may not be available for estimation.

There are many examples where indirect observability can potentially play an important role. For example, for turbulent dynamical systems with the assumption of scale separation, a reduced stochastic process can be obtained through homogenization in the limit of large time-scale separation between slow and fast degrees of freedom [58, 59, 29, 28, 21, 56, 45].

In [48] authors solve the Navier-Stokes equations of moist convection on a high-resolution grid inside each coarse column in simulations of large-scale atmospheric circulation. Reduced stochastic models for the triad dynamics and truncated Burgers–Hopf equations are derived in [57] using the homogenization technique. Effective stochastic models for the time evolution of spatial averages in finite-difference discretizations of inviscid Burgers-Hopf equation and one-dimensional shallow water equations are derived in [1, 96, 22]. Typically, reduced equations are considerably lower-dimensional compared with the full dynamics. In addition, the absence of fast variables in the reduced models allows taking larger time step in simulations. Therefore, reduced models can be typically simulated much faster while preserving some essential statistical characteristics of the large-scale variables in the full system. In the examples above, the main goal is to construct effective models for large-scale/slow variables. However, reduced models are often known up to certain constants and parametric estimation is required to determine those constants from data. In such cases, data would be generated by the full high-dimensional dynamics, but estimation is carried out for only a subset of dynamic variables.

In [7, 8], the authors formalize the *Indirect Observability* framework in the context of estimating reduced equations for slow dynamics using moment estimators and multiscale slow/fast data. *Indirect Observability* happens naturally when the nature of the observed process is unknown or too complex to use in numerical/analytical calculations; instead, it is desirable to approximate this process by a suitable stochastic process X_t with matching statistical features, such as molecular dynamics and financial mathematics [7, 9, 6, 62, 89, 49]. In finance, market microstructure noise can introduce discrepancies between the data and the model in volatility estimation from

financial time series, while the realized volatility is often applied for estimating the theoretically limiting return's variance [2, 97]. The climate system is assumed to be in statistical equilibrium and the large-scaled reduced system is derived using averaging or homogenization with respect to the equilibrium measure of fast variables [29, 18, 28, 21]. In *Indirect Observability* framework, the observable process is denoted as Y_t^ε, where ε is the scale-separation parameter in multiscale systems. The limiting process of interest X_t is not observed directly; instead the parameters of the stochastic model for X_t are inferred from the data sub-sampled from Y_t^ε. In [7], Y_t^ε is also referred to as an approximating process. The *Indirect Observability* context is defined, when the data to be fitted by a parameterized stochastic model (SM) is not generated by SM itself, but by another stochastic model approximating SM. An important task is to develop a practical approach to select essential parameters and compute parameter estimators, when the data to be fitted by a parameterized stochastic model X_t is generated by another approximating stochastic model Y_t^ε. *Indirect observability* studies the behavior of estimators as $\varepsilon \to 0$, but in many practical situations ε is finite and relatively large.

When the form of the dynamics is at least partially known or a reasonable expression of the dynamical process is assumed, correct model selection is crucial in the subsequent step of estimation. The least absolute shrinkage and selection operator (LASSO) [80] is a useful and well-studied approach to the problem of model selection, which is a shrinkage estimation method by adding a L^1 penalty on parameters to the standard residual sum of squares subject. One of the major advantages of this approach is the simultaneous execution of both parameter estimation and variable selection [80, 25]. De Gregorio and Iacus (2012) [20] applied the LASSO estimator to parametric estimation of SDEs. They point out that the classical LASSO penalty must be modified due to different convergence rates for the drift and diffusion coefficients. The adaptive LASSO estimator for SDEs with the tuning parameters suggested in [100] is unbiased with oracle properties. Lindström and Höök (2018) [52] generalize the method in [20] by imposing weaker restrictions on the data by numerically computing conditional moments of the Kolmogorov Backward (KB) equation.

Under *Indirect Observability* from multiscale data, the time step of estimation should be sufficiently large to avoid the strong influence of small-scale effects. Ideally, one would like to know the transition density, which could allow one to compute the maximum likelihood estimator and then the LASSO estimator. When estimation time step is relatively short, a Gaussian approximation is often used (easily derived from the Euler-Maruyama discretization). However, when the observational time step is relatively large, more accurate approximations for the transition density are needed. Several simulation methods for generating diffusion bridges based on the discrete-time approximation of the diffusion process have been developed [67, 14, 24]. In [12, 51], the authors proposed a sequential Monte Carlo method for simulating diffusion bridges with a re-sampling scheme guided by the empirical distribution of backward paths.

In the first part of this dissertation, we combine simulated likelihood techniques with the adaptive LASSO type estimator for selecting the correct model under *Indirect Observability*. Chapter 2 introduces the simulated LASSO-type estimator (SLASSO) for discretely sampled diffusion processes. Chapter 3 contains applications of this approach to the additive triad model and Burgers model.

1.2 Machine Learning for Predicting Nonlinear Dynamics

When the dynamic equations are unknown or too complex to utilize and only observational data is available, models learned from data offer a viable alternative to analytical or semi-analytical approximations. Data-driven models can help to overcome extreme computational requirements such as (a) numerically integrating coupled, high-dimensional, non-linear partial differential equations exhibiting multiple temporal and spatial scales; (b) insufficient data or unknown parameters characterizing sub-processes; and, in some cases, (c) no knowledge of the governing subsystem of equations. In the past decade, there has been an explosion of activity related to machine learning, including the use of machine learning techniques to address computational challenges outlined above.

In particular, various machine learning techniques have been proposed to accelerate simulations

and predictions of nonlinear dynamical systems [50, 34, 82, 23, 91, 72, 16, 86]. Some of the most popular approaches for time-series analysis and prediction utilize sequential models such as artificial neural networks (ANNs) [34, 82, 23, 72, 73, 71, 13], recurrent neural networks (RNNs) [60, 85, 95, 99, 32, 54, 53, 65], and gated recurrent units (GRU) [91, 30]. ANNs are computational models that mimic biological neural networks. They are represented by a network of neuron-like processing units interconnected via synapse-like weighted links. RNNs are suited for dynamic (temporal) data processing, as they can embed temporal dependence of the inputs into their dynamical behavior. In other words, RNNs are capable of representing dynamical systems driven by sequential inputs owing to their feedback connections. In particular, recent studies demonstrated promising results in using sequential machine learning models to build data-driven parameterizations for modeling atmospheric and oceanic processes [71, 30, 13] and dynamical systems [65].

Vlachas (2020) [86] and Chattopadhyay (2020) [16] surprisingly found that the echo state network (ESN) [41], a specialized simple type of RNNs, outperform LSTM and ANNs in prediction tasks involving chaotic dynamical systems. In particular, ESN is applied in the context of weather prediction [5, 61]. Hence, we consider ESN as a tool in turbulence predictability studies. ESN [41, 42] belongs to a family of Reservoir Computing methods where states of the reservoir represent activations of its constituent neurons (represented as time-dependent dynamic variables). The methods for obtaining effective reservoirs, which have been summarized in [54], are categorized into task-independent generic guidelines and task-dependent reservoir adjustments. In the ESN approach, the reservoir connectivity matrix is generated at random and only output weights are trained using observational data. These weights can be computed easily using ridge regression which is considerably faster than back-propagation. In addition, ESN does not suffer from the vanishing and exploding gradients problem. The key disadvantage of ESN as compared to other RNNs, is weaker expressive power, but it could be improved by augmenting reservoir states with ad-hoc nonlinear combinations [65]. ESN is originally an RNN-based framework and is therefore suitable for temporal/sequential information processing [43]. In the case of physical implementation, ESN has been successfully applied for simple function approximation [37], system identification [44], and

direct adaptive control [92].

ESN has rapidly become popular among more practitioners due to two main advantages. First, ESN is easy to train since we only adapt linear readouts from this dynamical system for a specific task rather than tackling the more difficult problem of training the recurrent network itself. Second, ESN is good at temporal processing. In ESN, the state space can be viewed as a temporal feature space and the readout mapping from the state space as a kernel machine operating in that feature space. Hence, ESN is popular for all fields sharing important properties like long-range dependencies and high performance variances across subjects, for example, human-robot interaction. Human-robot interaction comprises a broad spectrum of tasks such as, natural language processing [83, 69], robotic domain including navigation [4, 19, 38] and gestures recognition [31, 81]. In addition, ESN assisted living applications such as blood pressure estimation [10, 27]. Moreover, ESN can be combined with other deep learning methods to improve the performance of the original ones, such as generative adversarial networks [3], convolutional neural networks [84] and reinforcement learning [15].

In the second part of this dissertation, we present the application of the ESN to the task of predicting trajectories of complex dynamical systems. In particular, we focus on predicting trajectories of the one-dimensional shallow water equations. Chapter 4 describes the basic concept and prediction algorithms of Echo State Network. Shallow water equations and some of its properties are discussed in Chapter 5. Chapters 6 and 7 present the application of the ESN to the one-dimensional shallow water equations.

2 Simulated LASSO-type Estimator

2.1 Stochastic Models

Let $X_t \in \mathbb{R}^d$ be a d-dimensional stochastic process which is the solution of the multivariate stochastic differential equation

$$dX_t = \mu(\alpha, X_t)dt + \sigma(\beta, X_t)dW_t, \quad X_0 = x_0. \tag{2.1}$$

We also assume that the process X_t is a limiting process in the sense that there is a process Y_t^ε such that $Y_t^\varepsilon \to X_t$ as $\varepsilon \to 0$. Under *indirect observability* the data is generated by Y_t^ε, but our goal is to estimate the effective model for X_t.

We also assume that equation (2.1) involves unknown drift and diffusion coefficients which are defined parametrically, i.e.,

$\mu : \Theta_p \times \mathbb{R}^d \to \mathbb{R}^d$ is drift function,

$\sigma : \Theta_q \times \mathbb{R}^d \to \mathbb{R}^d$ is diffusion function,

$\alpha = (\alpha_1, \ldots, \alpha_p)^{'} \in \Theta_p \subset \mathbb{R}^{d \times p}, p \geq 1$ is a parametric matrix in the drift function,

$\beta = (\beta_1, \ldots, \beta_q)^{'} \in \Theta_q \subset \mathbb{R}^{d \times q}, q \geq 1$ is a parametric matrix in the diffusion function.

$W_t \in \mathbb{R}^d$ is a standard Brownian motion.

We denote the whole parameters set as $\theta = (\alpha, \beta) \in \Theta_p \times \Theta_q$. We also denote the unknown true value by $\theta_0 = (\alpha_0, \beta_0)$. Please note that true parameters θ_0 may have some zero components.

We consider discretely sampled data $\{U_n\} = \{Y_{n\Delta}^\varepsilon, n = 1, \ldots, N\}$. Here we consider equidistant sampling with $t_{i+1} - t_i = \Delta$ and under indirect observability Δ has to be large to avoid influence of multiscale effects. We proceed by deriving the simulated-LASSO (SLASSO) estimator for the SDEs (2.1) and treating the data $\{U_n\}$ as if it was generated by (2.1).

2.2 Simulated Maximum Likelihood Estimation

The LASSO method usually consists of adding a L^1 regularization term to the cost function. For parametric estimation of SDEs this means adding L^1 penalty terms on the parameters. Thus, in

our context, LASSO estimation implies combining the maximum likelihood approach plus penalty terms for parameters.

Let $p(X_{t_i}|X_{t_{i-1}}, \theta)$ represent the conditional probability density of X_{t_i} given $X_{t_{i-1}}$ evaluated at time t_{i-1} for a given set of parameters θ. Since a continuous time diffusion is a Markov process, Markov property carries over to any discrete sub-sample from the continuous time path. Next, assume that there is a discrete sample $U = \{X_{t_i}, i = 0, \ldots, n\}$ and the initial condition X_{t_0} is fixed. Then the log-likelihood of discrete observations has the following simple form

$$\mathbb{L}(X, \theta) = \frac{1}{n} \sum_{i=1}^{n} \log p(X_{t_i}|X_{t_{i-1}}, \theta). \tag{2.2}$$

When the transition density $p(X_{t_i}|X_{t_{i-1}}, \theta)$ is known, log-likelihood calculation and its maximization with respect to θ for a given set of data discretely observed from (2.1) is relatively straightforward.

Various approximations of the transition density are typically necessary, since the transition density cannot be computed in a closed-form except for a handful of cases. For example, one commonly used approximation is based on the discretization of the underlying SDEs (2.1). Euler-Maruyama is the lowest-order approximation of (2.1) and is commonly used for this purpose. For instance, the likelihood based on the Euler-Maruyama scheme has been used by [94, 33, 47] to estimate stochastic differential equations. This method approximates the drift and diffusion terms by constant functions over the interval $[t_{i-1}, t_i]$. The explicit approximate form of the transition density is obtained by discretizing the continuous-time stochastic differential equations (2.1) with Euler-Maruyama scheme

$$X_{t_i} - X_{t_{i-1}} \approx \mu(\alpha, X_{t_{i-1}})\Delta + \sigma(\beta, X_{t_{i-1}})\sqrt{\Delta}Z \tag{2.3}$$

where $Z \sim N(0, 1)$.

The discrete process in (2.3) leads to the approximate transition density

$$p(X_{t_i}|X_{t_{i-1}},\theta) \approx \phi\left(X_{t_i};\, X_{t_{i-1}} + \mu(\alpha; X_{t_{i-1}})\Delta;\, \sigma(\beta, X_{t_{i-1}})\sqrt{\Delta}\right) \tag{2.4}$$

where $\phi(x; m; v^2)$ is the density of a normal random variable with mean m and variance v^2 evaluated at x.

The log-likelihood function $\mathbb{L} : \mathbb{R}^{(n+1)\times d} \times \Theta \to \mathbb{R}$, is then

$$\begin{aligned}\mathbb{L}(X,\theta) = \frac{1}{2n}\sum_{i=1}^{n}\Big[&\log\left(2\pi \det(\Sigma_{i-1}(\beta))\right)\\ &+\frac{1}{\Delta}(\Delta X_i - \mu_{i-1}(\alpha)\Delta)'\Sigma_{i-1}^{-1}(\beta)(\Delta X_i - \mu_{i-1}(\alpha)\Delta)\Big],\end{aligned} \tag{2.5}$$

where $\Delta X_i = X_{t_i} - X_{t_{i-1}}$, $\Sigma_i(\beta) = \sigma(\beta, X_{t_i})\sigma(\beta, X_{t_i})'$ and $\mu_i(\alpha) = \mu(\alpha, X_{t_i})$. The error of the Euler-Maruyama method is $O(\sqrt{\Delta})$ in the sense of strong convergence. Therefore, approximations in (2.3) and (2.5) are accurate if the time step length Δ is sufficiently small.

Simulated maximum likelihood estimation (SMLE) is used to approximate the transition density $p(X_{t_i}|X_{t_{i-1}},\theta)$ at the expense of more computational cost. This is particularly useful when the sampling time step Δ is large and the approximation in (2.4) and the resulting likelihood function (2.5) are no longer accurate. The basic idea is partitioning the interval $[t_{i-1}, t_i]$ into M sub-intervals $t_{i-1} = \tau_0 < \tau_1 < \ldots < \tau_M = t_i$, such that the first order approximation (2.5) is sufficiently accurate on each sub-interval with length $\dfrac{\Delta}{M}$. The random variables $X_{\tau_1}, \ldots, X_{\tau_{M-1}}$ are unobserved and must be integrated out. Because the process is Markov, the SMLE approximation is based on the following transition density:

$$p^{(M)}(X_{t_i}|X_{t_{i-1}},\theta) = \int \prod_{m=1}^{M} p(X_{\tau_m}|X_{\tau_{m-1}},\theta)d\lambda(X_{\tau_1},\ldots,X_{\tau_{M-1}}), \tag{2.6}$$

where λ denotes the Lebesgue measure on $\mathbb{R}$. In Bally and Talay (1995) [11], $p^{(M)}(X_{t_i}|X_{t_{i-1}},\theta)$ bias is shown to be of order $O(1/M)$.

Importance sampling can be used to estimate the integral in (2.6). In particular, we can rewrite

the integral in (2.6)

$$p^{(M)}(X_{t_i}|X_{t_{i-1}},\theta) = \int \prod_{m=1}^{M} \frac{p(X_{\tau_m}|X_{\tau_{m-1}},\theta)}{q(X_{\tau_1},\ldots,X_{\tau_{M-1}})} \times q(X_{\tau_1},\ldots,X_{\tau_{M-1}})d\lambda(X_{\tau_1},\ldots,X_{\tau_{M-1}}) \quad (2.7)$$
$$= E_q\left[\prod_{m=1}^{M} \frac{p(X_{\tau_m}|X_{\tau_{m-1}},\theta)}{q(X_{\tau_1},\ldots,X_{\tau_{M-1}})}\right],$$

where E_q refers to the expectation with respect to the importance sampler $q(\cdot)$ and $q(u_1,\ldots,u_{M-1})$ denotes the probability density on $\mathbb{R}^{M-1}$. In practice, the density $q(u_1,\ldots,u_{M-1})$ can be approximated by a product of conditional Gaussian densities based on the Euler-Maruyama discretization of the sampler. This approach suffers from the usual problems with non-parametric density estimation: a slower convergence rate (in the number of simulations) and the curse of dimensionality. If the integral in (2.6) were available in closed-form we could choose M large enough to reduce the discretization error to a sufficiently small level and approximate the log-likelihood accordingly. As the integral is not available in a closed-form, we use a classical Monte-Carlo estimator for this expectation, calculated based on K independent and identically distributed (IID) draws $(X_{\tau_1},\ldots,X_{\tau_{M-1}}) \sim q(\cdot)$,

$$p^{(M,K)}(X_{t_i}|X_{t_{i-1}},\theta) = \frac{1}{K}\sum_{i=1}^{K} p^{(M)}(X_{t_i}|X_{t_{i-1}},\theta). \quad (2.8)$$

The construction of the estimated log-likelihood is a straightforward application of the Markov property similar to that given in (2.2) as

$$\mathbb{L}^{(M,K)}(X,\theta) = \frac{1}{n}\sum_{i=1}^{n} \log p^{(M,K)}(X_{t_i}|X_{t_{i-1}},\theta). \quad (2.9)$$

Selecting a good importance sampling density $q(\cdot)$ is critical. Pederson (1955) [68] suggests simulating $X(\cdot)$ sequentially using the Euler approximation which is criticized for inefficiency. Durham and Gallant (2002) [24] propose two more efficient Monte-Carlo samplers, which benefit from simulating the $X(\cdot)$ values conditionally on both the initial and terminal points, instead of only the

initial point as the Pederson method. Other simulated Monte-Carlo estimators are mainly based on these two sampling methods. Here, we choose the modified Brownian bridge sampler from Durham and Gallant [24].

The modified Brownian bridge sampler is introduced based on recursion

$$X_{\tau_{m+1}} = X_{\tau_m} + \frac{X_{\tau_M} - X_{\tau_m}}{\tau_M - \tau_m}\delta\tau + \sqrt{\frac{M-m-1}{M-m}}\,\sigma(\beta, X_{\tau_m})\,\sqrt{\delta\tau}\,Z_m, \quad m = 0, \ldots, M-1 \tag{2.10}$$

where $\tau_0 = t_{i-1}$, $\tau_M = t_i$, $X_{\tau_0} = X_{t_{i-1}}$, $X_{\tau_M} = X_{t_i}$, and $Z_m \overset{i.i.d.}{\sim} N(0;1)$. Here we assume uniform time step for the sampler $\delta\tau = \tau_k - \tau_{k-1} = \Delta/M$. This sampler is analyzed in [24], and it is demonstrated that the modified diffusion term (multiplication by $\sqrt{(M-m-1)/(M-m)}$) results in a much better performance. Stramer and Yan (2007) [78] show that when diffusion σ is constant, the modified Brownian bridge is also exactly a Brownian bridge and they also show that $K = M^2$ is computationally optimal for a fixed large amount of computer time.

The conditional density $q(\cdot)$ in (2.7) is computed as a product density

$$q(u_1, \ldots, u_{M-1}) = \phi(X_{\tau_{M-1}}|X_{\tau_{M-2}}) \times \ldots \times \phi(X_{\tau_{k+1}}|X_{\tau_k}) \times \ldots \times \phi(X_{\tau_1}|X_{\tau_0}),$$

where $\phi(X_{\tau_{k+1}}|X_{\tau_k}) \sim N(m_k, v_k^2)$ is the density on the normal distribution computed from the Euler-Maruyama discretization of the Bridge sampler in (2.10), i.e., the mean and variance are

$$m_k = X_{\tau_m} + \frac{X_{\tau_M} - X_{\tau_m}}{\tau_M - \tau_m}\delta\tau, \qquad v_k^2 = \frac{M-m-1}{M-m}\,\sigma^2(\beta, X_{\tau_m})\,\delta\tau.$$

Finally, we define $\tilde{\theta} : \mathbb{R}^{(n+1)\times d} \to \Theta$ be the simulated maximum likelihood estimator (SMLE) of $\theta \in \Theta$ based on (2.9), that is

$$\tilde{\theta} = (\tilde{\alpha}, \tilde{\beta}) = \text{argmin}_\theta\, -\mathbb{L}^{(M,K)}(X, \theta). \tag{2.11}$$

2.3 Simulated Monte Carlo LASSO Estimator by Least Squares Approximation

The important property stating that the correct parameters are set to zero by the LASSO method under the true data generating model is called the oracle property. Also, as argued by Fan and Li (2001) [25], a good selection procedure should have the so-called oracle properties:

1. consistently estimates null parameters as zero and vice versa;

2. has the optimal estimation rate and converges to a Gaussian random variable $N(0, \Sigma)$, where Σ is the covariance matrix of the true subset model.

As shown by Zou (2006) [100], since the classical LASSO estimator uses the same amount of shrinkage for each parameter, the classical LASSO estimation cannot be as efficient as the oracle, and the selection results could be inconsistent, whereas its adaptive version has the oracle properties. Intuitively, if larger amounts of shrinkage are applied to the near-zero coefficients while smaller amounts are used for the non-zero ones, an estimator with improved efficiency can be obtained.

The Simulated Monte-Carlo LASSO objective function is given by

$$\mathcal{F}(\theta)_{MLE} = -\mathbb{L}^{(M,K)}(X, \theta) + \sum_{j=1}^{p+q} \lambda_j |\theta_j|, \tag{2.12}$$

where $\mathbb{L}^{(M,K)}(X, \theta)$ is the simulated Monte-Carlo log-likelihood function (2.9) and λ_j are real positive values representing an adaptive amount of shrinkage for each element of parameters θ. The LASSO estimator is the minimizer of the objective function (2.12). Usually, it is a nonlinear optimization problem under L_1 constraints, which might be numerically challenging to solve. In addition, direct optimization of the cost function (2.12) can be extremely computationally costly for data sampled with a large time step Δ, since simulated likelihood needs to be computed for every value of parameters θ.

Least Squares Approximation. Here we introduce a simpler and computationally cheaper approach based on the approximation of the cost function in (2.12). In particular, using the idea

of Wang and Leng (2007) [87], we can consider a least squares approximation (LSA) of the likelihood function $\mathbb{L}^{(M,K)}(X,\theta)$ in (2.12). LSA can transfer many different types of LASSO objective functions into their asymptotically equivalent least squares problems. Thereafter, the standard asymptotic theory can be established and the optimization solution can be found at a lower computational cost. A standard argument proceeds as follows. First, compute the simulated maximum likelihood estimator $\tilde{\theta}$ by solving the by solving the optimization problem (2.11). Next, consider the Taylor series expansion of the likelihood function at the maximum likelihood estimator $\tilde{\theta}$

$$\mathbb{L}(\theta) \approx \mathbb{L}(\tilde{\theta}) + \dot{\mathbb{L}}(\tilde{\theta})(\theta - \tilde{\theta}) + \frac{1}{2}(\theta - \tilde{\theta})'\ddot{\mathbb{L}}(\tilde{\theta})(\theta - \tilde{\theta}), \tag{2.13}$$

where $\dot{\mathbb{L}}$ and $\ddot{\mathbb{L}}$ are the first- and second-order derivatives of the loss function $\mathbb{L}$. Because $\tilde{\theta}$ is the minimizer of $\mathbb{L}$, we know that $\dot{\mathbb{L}}(\tilde{\theta})= 0$. Thus the approximation (2.13) can be simplified to

$$\mathbb{L}(\theta) \approx \mathbb{L}(\tilde{\theta}) + \frac{1}{2}(\theta - \tilde{\theta})'\ddot{\mathbb{L}}(\tilde{\theta})(\theta - \tilde{\theta}). \tag{2.14}$$

The analytical optimization solution of $\mathbb{L}(\theta)$ above does not change if we ignore the constant $\mathbb{L}(\tilde{\theta})$ and the coefficient $1/2$. Thus, the objective function (2.14) can be further simplified to $(\theta - \tilde{\theta})'\ddot{\mathbb{L}}(\tilde{\theta})(\theta - \tilde{\theta})$ in the optimization problem. Finally, we define the LASSO-type objective function given the MLE estimator $\tilde{\theta}$

$$\mathcal{F}(\theta)_{QMLE} = -(\theta - \tilde{\theta})'\ddot{\mathbb{L}}(\tilde{\theta})(\theta - \tilde{\theta}) + \sum_{j=1}^{p+q} \lambda_j |\theta_j|. \tag{2.15}$$

With least squares approximation, (2.15) leads to a minimum distance criterion plus the penalty terms. It is much easier to solve numerically than (2.12); nevertheless, the optimization solutions of the two objective functions are equivalent in some neighborhood of $\tilde{\theta}$. It is worth emphasizing that (2.15) is a convex optimization problem and its global minimizer can be efficiently computed. In addition, the hessian matrix $\ddot{\mathbb{L}}(\tilde{\theta})$ here is constant, which also accelerates computations.

Selecting Penalty Parameters. The theoretical and practical implications of SLASSO

method rely on the specification of the penalty parameters λ_j. Although the asymptotic properties of the LASSO-type estimator (2.15) have been established in Wang and Leng (2007) [87] in the context of regression and linear models, the extension to discretely observed diffusion processes is nontrivial because estimators for the drift and diffusion parameters have two different rates of convergence [20]. Next, we discuss how different rates of convergence for the drift and diffusion parameters affect the selection of penalty parameters λ_j in the objective function (2.15). Many authors utilize constant penalty parameters which are independent of θ (e.g., [80]). Here we follow approach suggested in [87, 100, 20] and define penalty parameters which depend on the SMLE estimator $\tilde{\theta}$

$$\lambda_j = \lambda_0 |\tilde{\alpha}_j|^{-\delta_1}, \qquad \gamma_k = \gamma_0 |\tilde{\beta}_k|^{-\delta_2}, \tag{2.16}$$

where $\tilde{\theta} = (\tilde{\alpha}, \tilde{\beta})$ is the SMLE estimator (2.11), and $\tilde{\alpha}$ and $\tilde{\beta}$ are elements of the parameter vector for the of drift and diffusion, respectively (recall equation (2.1)). It was determined empirically [100, 20] that parameters $\delta_1, \delta_2 \geq 0$ should be taken in the range $\{0.5, 1, 2\}$.

De Gregorio and Iacus (2012) [20] analyzed LASSO estimations for discretely sampled diffusion processes and considered estimation regime $\Delta \to 0$, $n\Delta \to \infty$, $n\Delta^2 \to 0$, $n \to \infty$. Their asymptotic framework is called rapidly increasing design and the condition $n\Delta^2 \to 0$ as $n \to \infty$ means that Δ shrinks to zero slowly. In [20] appropriate conditions were established for estimators to be consistent and correctly estimate the proposed model (probability that the correct parameters are set to zero tends to one). These criteria translate to our framework as follows:

$$\mathcal{C}_1 : \qquad \frac{\lambda_0}{\sqrt{n\Delta}} \to 0; \quad (n\Delta)^{\frac{\delta_1 - 1}{2}} \lambda_0 \to \infty, \tag{2.17}$$

$$\mathcal{C}_2 : \qquad \frac{\gamma_0}{\sqrt{n}} \to 0; \quad (n)^{\frac{\delta_2 - 1}{2}} \gamma_0 \to \infty, \tag{2.18}$$

as $n \to \infty$. As the sample size grows, the weights for near-zero coefficient predictors get inflated (to infinity), whereas the weights for nonzero coefficient predictors converge to a finite constant. Thus, we can simultaneously (asymptotically) estimate large coefficients and small thresholds.

It was demonstrated that the sub-sampling time step Δ plays a crucial role for the estimation under *Indirect Observability* [7, 46, 17, 66, 6, 7, 8, 9].

Therefore, instead of concentrating on the behavior of estimators for varying size of the data, n, we pay particular attention to the sub-sampling rate. Penalization terms in (2.17), (2.18) are defined in terms of Δ, so that

$$(\lambda_0, \gamma_0) = C_p(\sqrt{\Delta}, 1). \tag{2.19}$$

We use the following typical values in computations $\delta_1 = \delta_2 = \{0.5, 1\}$ and $C_p = \{0.01, 0.001\}$.

Given the SMLE estimator $\tilde{\theta}$, the simulated LASSO-type objective function for parameters set $\theta = (\alpha, \beta)$ is defined with tuning penalty (2.16) in the drift and diffusion coefficients, that is

$$\mathcal{F}_{QMLE}(\theta) = (\theta - \tilde{\theta})' \ddot{\mathbb{L}}(\tilde{\theta})(\theta - \tilde{\theta}) + \sum_{j=1}^{p} \lambda_j |\alpha_j| + \sum_{k=1}^{q} \gamma_k |\beta_k|, \tag{2.20}$$

with penalty parameters in (2.16), (2.19). The corresponding adaptive LASSO-type estimator $\hat{\theta} : \mathbb{R}^{(n+1)\times d} \to \Theta$ is defined as the solution of the optimization problem

$$\hat{\theta} = (\hat{\alpha}, \hat{\beta}) = \underset{\theta}{\operatorname{argmin}}\, \mathcal{F}_{QMLE}(\theta). \tag{2.21}$$

De Gregorio and Iacus (2012) [20] proved that under the rapidly increasing sampling design the adaptive LASSO-type estimator $\hat{\theta}$ has good behavior in the oracle sense.

3 SLASSO Application

3.1 SLASSO Application to the Additive Triad Model

In this section, we apply our SLASSO estimator to a particular multiscale example, namely the additive triad model.

3.1.1 Additive Triad Model

The model we consider is the stochastically forced additive triad. This system is a low-dimensional model that has nonlinear interactions reminiscent of those occurring between the Fourier modes of a fluid flow. It is stochastically forced to mimic the 3 interactions with further unresolved modes. The system has three variables, one slow mode x_t and two fast modes y_t and z_t. The fast dynamics are dominated by two independent Ornstein-Uhlenbeck processes. The dynamical equations for this triad are defined as

$$\begin{aligned} dx &= A_1 yz dt, \\ dy &= A_2 xz dt - \gamma_2 y dt + s_2 dW_1(t), \\ dz &= A_3 xy dt - \gamma_3 z dt + s_3 dW_2(t), \end{aligned} \tag{3.1}$$

with $A_1 + A_2 + A_3 = 0$, so that the energy $(x_t^2 + y_t^2 + z_t^2)$ is conserved by the nonlinear interactions. γ_i, s_i are known parameters with $\gamma_2, \gamma_3 > 0$. W_1, W_2 are independent Brownian motions, and $\varepsilon > 0$ is the scale separation parameter. In this example $Y_t^\varepsilon \equiv x_t$, i.e., the data for estimation of the reduced model is generated by the first component of equation (3.1).

Homogenization for the Additive Triad

For a homogenization technique to the construction of triad model parameterizations, we refer to [57]. To study the convergence of the triad model in the limit of infinite separation of time-scale,

we need to introduce ε into the equations (3.1)

$$\begin{aligned} dx &= A_1 yz dt, \\ dy &= A_2 xz dt - \frac{\gamma_2}{\varepsilon} y dt + \frac{s_2}{\sqrt{\varepsilon}} dW_1(t), \\ dz &= A_3 xy dt - \frac{\gamma_3}{\varepsilon} z dt + \frac{s_3}{\sqrt{\varepsilon}} dW_2(t). \end{aligned}$$

On the time-scale t, when increasing the time-scale separation $\frac{1}{\varepsilon} \to \infty$, we have trivial dynamics of the averaged equations

$$\bar{x} = A_1 < y, z >_{\rho_{OU}} = 0, \tag{3.2}$$

where ρ_{OU} is the Gaussian invariant measure of the fast Ornstein-Uhlenbeck process generated by taking $A_i = 0$. In the homogenization setting, one looks at the convergence of the distribution of paths on a longer time-scale. The time is scaled to the diffusive time-scale εt and on this longer diffusive time-scale deviations from the averaged dynamics develop.

Diffusive scaling corresponds to the additive triad system

$$\begin{aligned} dx &= \frac{1}{\varepsilon} A_1 yz dt, \\ dy &= \frac{1}{\varepsilon} A_2 xz dt - \frac{\gamma_2}{\varepsilon^2} y dt + \frac{s_2}{\varepsilon} dW_1(t), \\ dz &= \frac{1}{\varepsilon} A_3 xy dt - \frac{\gamma_3}{\varepsilon^2} z dt + \frac{s_3}{\varepsilon} dW_2(t). \end{aligned}$$

By expanding the backward Kolmogorov equation for the slow-fast system in orders of ε, a Kolmogorov equation for the slow variables can be derived. In case of the additive triad (3.1), the dynamical equation corresponding to the Kolmogorov equation for the slow variables is a one-dimensional Ornstein-Uhlenbeck process [57]

$$dX_t = \alpha X_t dt + \beta dW(t), \tag{3.3}$$

where

$$\alpha = \frac{A_1}{2(\gamma_2+\gamma_3)}\left(\frac{A_2 s_3^2}{\gamma_3}+\frac{A_3 s_2^2}{\gamma_2}\right), \qquad \beta = \frac{A_1 s_2 s_3}{\sqrt{2\gamma_2\gamma_3(\gamma_2+\gamma_3)}}. \tag{3.4}$$

The effective equation (3.3) is then compared to the original model (3.1) on the time-scale t. The stationary distribution for the triad is a product measure and the stationary marginal distribution of x_t is a Gaussian exactly matching the stationary distribution (also Gaussian) of the limiting Ornstein-Uhlenbeck process X_t. The convergence of the correlation function and the kurtosis (the third moment quantifying departures from Gaussian distributions) are confirmed numerically [57]. Auto time-correlation function and kurtosis function are defined as

$$ACF(\tau) = \frac{\mathbb{E}[x_{t+\tau}x_t]}{\mathbb{E}[x_t^2]} \tag{3.5}$$

and

$$KUR(\tau) = \frac{\mathbb{E}[x_{t+\tau}^2 x_t^2]}{(\mathbb{E}[x_t^2])^2 + 2(\mathbb{E}[x_{t+\tau}]\mathbb{E}[x_t])^2}, \tag{3.6}$$

respectively.

Under the *Indirect Observability* we use the data generated by x_t in the triad model. Our goal is to recover the effective equation (3.3). The values of the "true" drift and diffusion parameters (3.4) in the limiting reduced model are only used to test the performance of the SLASSO estimator.

Numerical Method for the Triad Model

In order to approximate the solution, we assume $[0, T]$ is divided into equal N intervals of length $\Delta = \frac{T}{N}$. Triad equations (3.1) are integrated numerically with the time step δt and data is sub-sampled with time step Δ. The third order Runge-Kutta method is applied for the integration of

drift part $\mu(\alpha, v_t)$, where $v(t) = \{x_t, y_t, z_t\}$, i.e.,

$$\begin{aligned} K_1 &= \mu(\alpha, v(t), \\ K_2 &= \mu(\alpha, v(t) + 0.5K_1\delta t), \\ K_3 &= \mu(\alpha, v(t) + 0.75K_2\delta t), \\ K(t) &= \frac{1}{9}(2K_1 + 3K_2 + 4K_3). \end{aligned} \tag{3.7}$$

Then the data are updated according to the Euler scheme,

$$v(t + \delta t) = v(t) + K(t)\delta t + \sum_{j=2}^{3} \sigma_j \delta W_j. \tag{3.8}$$

We use $\delta t = 0.001$ in all simulations.

3.1.2 SLASSO Estimator

In this section, we describe our estimation procedure. We assume that the exact expression of the effective model is unknown. Therefore, we assume a polynomial form and additive and multiplicative noises and our goal is to use LASSO-type techniques to perform model selection. In particular, we assume the following form

$$du_t = (\theta_1 u_t + \theta_2 u_t^2 + \theta_3 u_t^3)dt + \theta_4 dW_1 + \theta_5 u_t dW_2, \tag{3.9}$$

where W_1, W_2 are independent Brownian motions.

Compared with the analytical limiting solution (3.3) of the additive triad model, the general solution (3.9) includes essential non-null parameters (θ_1, θ_4) and unessential null parameters $(\theta_2 = \theta_3 = \theta_5 = 0)$. The penalization terms are set in terms of Δ, such that $(\lambda_0, \gamma_0) = C_p(\sqrt{\Delta}, 1)$, $C_p = \{0.01, 0.001\}$ and $\delta_1 = \delta_2 = \{0.5, 1\}$.

SLASSO Optimization Procedure

Our goal is to perform SLASSO estimator to efficiently classify essential variables which are included in the expression of the effective model. The simulated LASSO-type estimator is based on the assumption that the polynomial expression of the effective model is the correct format and includes the "true" effective model as a subset. Notice that the SLASSO formalism is not limited by polynomial models, but our choice of effective model is motivated by some recent results on the derivation of effective models for fluid dynamics [58, 21, 28, 96].

For the effective model in (3.9) and data generated by $x(t)$ in the triad model (3.1), SLASSO is expected to select an effective model with $\theta_2 = \theta_3 = \theta_5 = 0$ and estimates the essential parameters. Correctly estimated values of essential parameters should be close to the analytically derived drift and diffusion in (3.3). Thus, the true parameters (3.10) in the limiting reduced model can be used to evaluate the performance of SLASSO estimation.

When there are a large number of redundant parameters, it is computationally infeasible to perform model selection in one step, since the power (probability) of selecting zero-value variables may be decentralized. In such cases we can perform LASSO estimation sequentially, eliminating non-essential parameters in multiple steps. However, in the triad model the number of non-essential parameters is relatively low (three) and the model selection can be performed in one step. Often the solution of the optimization procedure can result in some coefficients estimated exactly as zero when using box constrained optimization with zero lower bound in the optimizer [20, 87]. This problem occurs because in practice the boundary and sign of coefficients are unknown for most systems. This results maybe in a lack of generality for box constrained optimization. Therefore, we use the quasi-newton optimizer without any constraint to solve the SLASSO problem.

Summary of the SLASSO optimization procedure. Our implementation of the SLASSO model selection and parameter estimation is organized in the following three steps:

- **Step 1:** Compute the Simulated Maximum Likelihood Estimation (SMLE) $\tilde{\theta}$:

$$\begin{aligned}\tilde{\theta} = (\tilde{\alpha}, \tilde{\beta}) &= \text{argmin}_\theta \, -\mathbb{L}^{(M,K)}(X, \theta) \\ &= \text{argmin}_\theta \, \frac{1}{n}\sum_{i=1}^{n} \log p^{(M,K)}(X_{t_i}|X_{t_{i-1}}, \theta) \\ &= \text{argmin}_\theta \, \frac{1}{n}\sum_{i=1}^{n} \log \frac{1}{K}\sum_{i=1}^{K} E_q\left[\frac{p(X_{\tau_m}|X_{\tau_{m-1}}, \theta)}{q(X_{\tau_1}, \ldots, X_{\tau_{M-1}})}\right],\end{aligned}$$

  where the modified Brownian bridge (2.10) sampler is defined by Durham and Gallant [24] :

$$X_{\tau_m} = X_{\tau_{m-1}} + \frac{X_{\tau_M} - X_{\tau_{m-1}}}{M - m + 1} + \sqrt{\frac{M-m}{M-m+1}}\sqrt{\frac{\Delta t}{M}}\sigma(\beta, X_{\tau_{m-1}})Z_m, \quad m = 1, \ldots, M,$$

  with $M = \Delta/\delta t_{bridge}$. The goal of this step is to obtain a very good quadratic approximation for the maximum likelihood part and, thus, ensure faster convergence.

- **Step 2:** Use unconstrained quasi-Newton optimizer to solve the SLASSO problem (2.20). Given the SMLE estimator $\tilde{\theta}$, the simulated LASSO-type objective function for the parameter set $\theta = (\alpha, \beta)$ is defined with tuning penalty (2.16) in the drift and diffusion coefficients, i.e,

$$\hat{\theta} = (\hat{\alpha}, \hat{\beta}) = \underset{\theta}{\text{argmin}}(\theta - \tilde{\theta})'\ddot{\mathbb{L}}(\tilde{\theta})(\theta - \tilde{\theta}) + \sum_{j=1}^{p}\lambda_j|\alpha_j| + \sum_{k=1}^{q}\gamma_k|\beta_k|,$$

  where the tuning parameters are suggested in Zou (2006) [100]

$$\lambda_j = \lambda_0|\tilde{\alpha}_j|^{-\delta_1}, \qquad \gamma_k = \gamma_0|\tilde{\beta}_k|^{-\delta_2}.$$

  Here $\tilde{\alpha}_j$ and $\tilde{\beta}_k$ are the elements of SMLE estimator of drift and diffusion parametric matrix, respectively. For conscious thinking, we always consider both weak and strong penalty, such that $(\lambda_0, \gamma_0) = C_p(\sqrt{\Delta t}, 1)$, $C_p \in \{0.01, 0.001\}$, $\delta_1 = \delta_2 \in \{0.5, 1\}$.

- **Step 3:** Choose variables with high zero-value percentage to remove from the general expression of the limiting reduced model (3.3). Repeat steps (1 and 2) until there are no zero-value

parameters recognized by the SLASSO estimator.

We consider the behavior of SLASSO estimators when discrete data are sub-sampled from trajectories generated by the triad model (3.1). We analyze the behavior of estimators numerically as the sub-sampling step Δ is varied. Our goal is to analyze numerically the relationship between the time-scale of fast variables in the triad and errors in the estimation procedure.

3.1.3 Numerical Results

In this section, we perform a numerical investigation of the performance of our SLASSO estimator under indirect observability. We consider the triad model with nonlinear interaction coefficients

$$A_1 = -1, \quad A_2 = 0.75, \quad A_3 = 0.25,$$

and

$$\gamma_2 = \gamma_3 = 2r, \quad s_2 = s_3 = 3r, \quad r = 1, 2, 3, 5.$$

The corresponding limiting reduced model (3.3) for the slow variable X_t with the above parameter setting is

$$dX_t = -0.5625\ X_t\ dt + 1.5910\sqrt{r}\ dW_t. \tag{3.10}$$

We would like to emphasize that the theoretical limiting reduced model (3.10) is only used to test the performance of SLASSO estimation.

The parameter r influences the time-scale separation between the slow variable x_t and fast variables $\{y_t, z_t\}$ in the triad model. We define the time-scale of slow variable x_t as the area under the graph of the correlation function, i.e.,

$$\tau_x = \int_0^\infty ACF(s)\, ds. \tag{3.11}$$

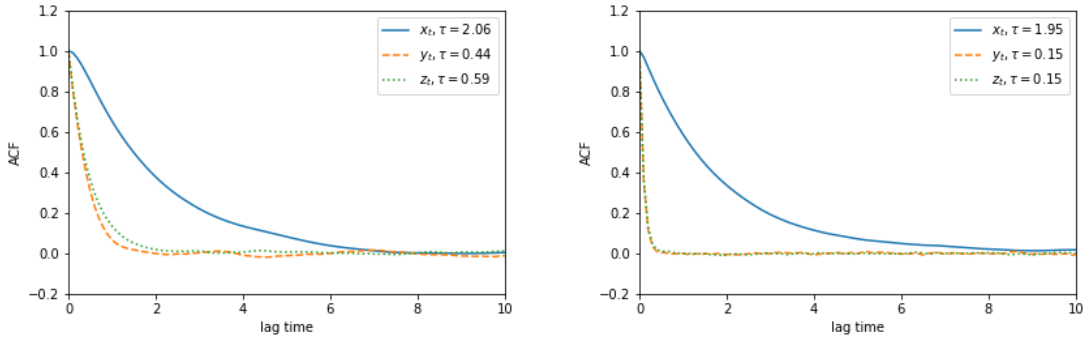
For exponential autocorrelation functions $ACF(s) = e^{-\gamma s}$, the time-scale is equivalent to $\tau_x = \gamma^{-1}$.

Figure 1 and Table **??** show time-scale separation for the slow variable x_t and fast variables y_t

Table 1: Time-scale for the slow variable x_t and fast variables y_t and z_t in the simulations of the triad model with $r = 1, 2, 3, 5$. Last column shows the time-scale separation.

	τ_{x_t}	τ_{y_t}	τ_{z_t}	$\tau_{x_t} / \max(\tau_{y_t}, \tau_{z_t})$
$r = 1$	2.06	0.44	0.59	3.49
$r = 2$	1.93	0.28	0.31	6.23
$r = 3$	1.89	0.21	0.23	8.22
$r = 5$	1.95	0.15	0.15	13.00

and z_t in the triad model with $r = 1, 2, 3, 5$. As expected, the time-scale separation increases as r becomes larger. In particular, for $r = 1$, the x_t is about 4 times slower than y_t and z_t and for $r = 5$ the x_t is about 13 times slower than the fast variables. We simulate $750,000$ trajectories of additive



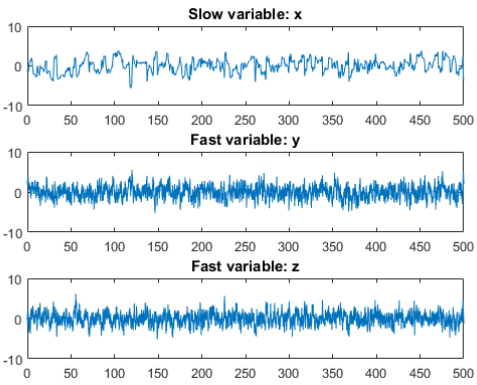
(a) r=1 (b) r=5

Figure 1: Auto-correlation function for the slow variable x_t and fast variables y_t, z_t in additive triad model (3.1); Left: $r = 1$. Right: $r = 5$.

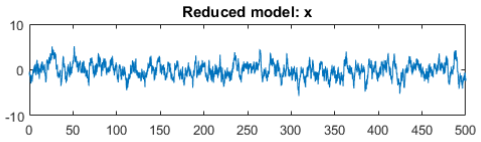
triad model and reduced model and pick sample trajectories with data size $N = 5,000$ and 100 Monte Carlo samples. We use a Monte-Carlo approach to estimate the probability that a certain parameter is zero. When data is generated by the slow variable in the triad model, the SLASSO estimator is capable to perform the model selection in one step. We would like to comment that in some other cases (there are a amount of redundant parameters), it might be necessary to proceed sequentially.

Figure 2 shows one particular trajectory of the additive triad model and the corresponding reduced stochastic model (3.10) with $r = 1$. It is clear that variable x_t is relatively slow compared with variables y_t and z_t. Under the *Indirect Observability* framework, only the variable x_t in

Figure 2 is observable. Our goal is select the correct effective parametrized stochastic model for the observed data; this effective stochastic model should agree with the limiting model in (3.3).



(a) additive triad model



(b) Reduced model

Figure 2: Left: sample path of the additive triad model in (3.1) with $r = 1$; Right: sample path of the corresponding effective model (3.3).

Table 2 reports the Monte Carlo mean and standard deviations of SMLE and SLASSO estimates for the data generated by the slow variable x_t in the additive triad model with varied sub-sampling step $\Delta = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.7\}$ and 100 Monte Carlo realisation. The SLASSO estimation with different penalization setting produce consistent results that correctly set unessential variable$(\theta_2, \theta_3, \theta_5)$ to zero in the general assumed expression (3.9). In this work, the SMLE discretization time step for the Brownian bridge is always taken as $\delta t_{bridge} = 0.05$.

Performance of the SLASSO estimator depends considerably on the sub-sampling time step of additive triad model, Δ. This is directly related to the scale-separation between the slow (x_t) and fast (y_t, z_t) variables and the effect of fast-scale variables on the estimation procedure. Table 2 demonstrates that at $\Delta = 0.7$ the SLASSO estimator is still affected by the multiscale effects generated by the fast variables. In particular, the time-scale of fast variables in the simulations with $r = 1$ is $\max\{\tau_y, \tau_z\} \approx 0.6$. Therefore, it is not surprising that the SLASSO estimation results in large errors even for $\Delta = 0.7$. Note that there are two parts of error here, (i) the difference of the triad model x_t and the limiting process X_t, (ii) the simulation error from SMLE. An experience from [57], the approximation error declines as the time-scale separation becomes large.

Table 2: SLASSO model selection result for additive triad model experiment with $r = 1$. We present parameter values computed by SMLE (see section 3.1.2) after the model selection; standard deviations from 100 Monte-Carlo trajectories are presented in round brackets below the estimated value, and relative errors are presented in square brackets next to the estimated value. Relative errors are computed using the "true" parameter values of the limiting process X_t. There are two parts of error here, the approximation error between the triad model x_t and the limiting process X_t and the simulation error from SMLE.

	θ_1	θ_2	θ_3	θ_4	θ_5	$\theta i^*(\theta_{i^*} = 0)$
True	-0.5625	0	0	1.5910	0	
$\Delta = 0.1$	-0.0963 [83%]	-	-	0.6633 [58%]	-	$\theta_2, \theta_3, \theta_5$
	(0.0111)	-	-	(0.0220)	-	
$\Delta = 0.2$	-0.1739 [69%]	-	-	0.8828 [45%]	-	$\theta_2, \theta_3, \theta_5$
	(0.0127)	-	-	(0.0265)	-	
$\Delta = 0.3$	-0.2336 [58%]	-	-	1.0144 [36%]	-	$\theta_2, \theta_3, \theta_5$
	(0.0130)	-	-	(0.0279)	-	
$\Delta = 0.4$	-0.2848 [49%]	-	-	1.1209 [30%]	-	$\theta_2, \theta_3, \theta_5$
	(0.0144)	-	-	(0.0240)	-	
$\Delta = 0.5$	-0.3203 [43%]	-	-	1.1983 [25%]	-	$\theta_2, \theta_3, \theta_5$
	(0.0156)	-	-	(0.0262)	-	
$\Delta = 0.7$	-0.3703 [34%]	-	-	1.2841 [19%]	-	$\theta_2, \theta_3, \theta_5$
	(0.0189)	-	-	(0.0305)	-	

Comparing with LASSO: By setting $M = K = 1$ in (2.11), there is no Brownian bridge sampler in the likelihood function, which is the classical MLE. To present the importance of the Brownian bridge sampler, we set $M = K = 1$ and utilize LASSO to select effective model for the additive triad model experiment with $r = 1$. Table 3 shows that LASSO cannot select the correct model for the additive triad model due to large discretization error when $\Delta \geq 0.3$. In addition, there are relatively larger error on both estimated drift and diffusion coefficients, comparing with SLASSO result in Table 2. The simulation error gets worse when enlarging the time step Δ.

To study the performance of the SLASSO estimator in the regime with larger time-scale separation, we consider three more experiments for additive triad model with r =2, 3 and 5. In particular, for $r = 5$ the time-scale separation is equal to 13, as indicated by Table **??**. Table 4 shows results of SLASSO estimation from the data generated by the triad model with $r = 2, 3, 5$. We can see that estimation results with $\Delta = 0.7$ improve considerably as the time-scale separation increases. This is an indication that the SLASSO estimator becomes less sensitive to the multiscale effects as

Table 3: LASSO (no Brownian bridge sampler) model selection result for additive triad model experiment with $r = 1$. We present parameter values computed by MLE after the model selection; standard deviations from 100 Monte-Carlo trajectories are presented in round brackets below the estimated value, and relative errors are presented in square brackets next to the estimated value. Relative errors are computed using the "true" parameter values of the limiting process X_t. There are two parts of error here, the approximation error between the triad model x_t and the limiting process X_t and the simulation error from SMLE.

	θ_1	θ_2	θ_3	θ_4	θ_5	$\theta i^*(\theta_{i^*} = 0)$
True	-0.5625	0	0	1.5910	0	
$\Delta = 0.1$	-0.0984[82%]	-	-	0.6648[58%]	-	$\theta_2, \theta_3, \theta_5$
	(0.0110)	-	-	(0.0239)	-	
$\Delta = 0.2$	-0.1718[69%]	-	-	0.8684[45%]	-	$\theta_2, \theta_3, \theta_5$
	(0.0135)	-	-	(0.0236)	-	
$\Delta = 0.3$	-0.2263[60%]	-	-	0.9589[40%]	-0.1675	θ_2, θ_3
	(0.0128)	-	-	(0.0284)	(0.0514)	
$\Delta = 0.4$	-0.2256[60%]	-	-	0.9527[40%]	-0.1657	θ_2, θ_3
	(0.0126)	-	-	(0.0320)	(0.0520)	
$\Delta = 0.5$	-0.2958[47%]	-	-	1.0344[35%]	-0.2639	θ_2, θ_3
	(0.0133)	-	-	(0.0215)	(0.0372)	
$\Delta = 0.7$	-0.3259[42%]	-	-	1.0439[34%]	-0.3123	θ_2, θ_3
	(0.0124)	-	-	(0.0219)	(0.0260)	

r increases.

Figure 3 presents estimators for the drift and diffusion coefficients as means of the 100 Monte-Carlo realizations. We also plot error bars from the Monte-Carlo simulations. We can see that the model selection is consistent with the form of the reduced model (3.3), and only parameters θ_1 and θ_4 have non-zero values after the model selection. We can see that using larger sub-sampling time step Δ clearly reduces the estimation errors, but even for $\Delta = 1$ and $r = 3$ relative errors are not negligible. The relative error of drift and diffusion coefficients further reduce to 6% and 4% respectively for $\Delta = 1$ and $r = 5$.

To check the statistical performance of the model produced by the SLASSO estimation procedure, time auto-correlation and kurtosis function are plotted in Figure 4. We can see that the performance of the effective SLASSO model improves as r increases. In particular, we obtain good agreement between the triad data and the statistical properties of the estimated model with $r = 5$. Table 5 presents one-point moments for the numerical simulation of the additive triad and statis-

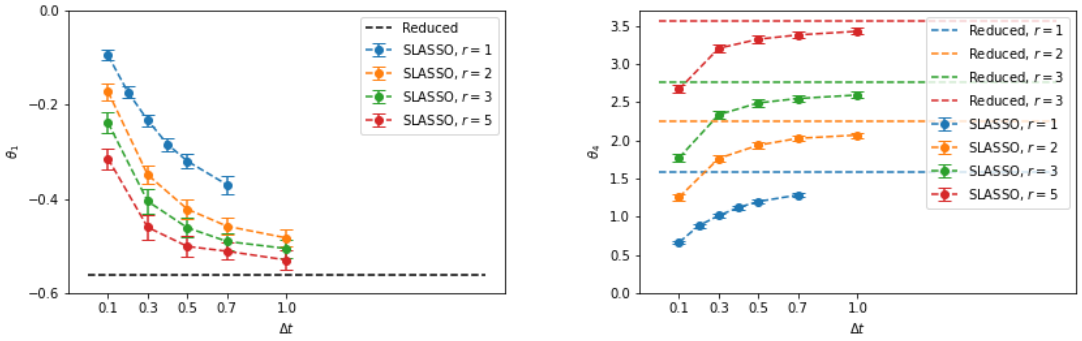
Table 4: SLASSO model selection result for additive triad model experiment with $r = 2, 3, 5$. We present parameter values computed by SMLE (see section 3.1.2) after the model selection; standard deviations from 100 Monte-Carlo trajectories are presented in round brackets below the estimated value, and relative errors are presented in square brackets next to the estimated value. Relative errors are computed using the "true" parameter values of the limiting process X_t. There are two parts of error here, that the approximation error between the triad model x_t and the limiting process X_t, and the simulation error from SMLE.

	r=2		r=3		r=5	
	θ_1	θ_4	θ_1	θ_4	θ_1	θ_4
True	-0.5625	2.2500	-0.5625	2.7557	-0.5625	3.5576
$\Delta = 0.1$	-0.1733 [69%] (0.0175)	1.2527 [44%] (0.0412)	-0.2390 [58%] (0.0216)	1.7690[36%] (0.0508)	-0.3150 [44%] (0.0235)	2.6665[25%] (0.0702)
$\Delta = 0.3$	-0.3493 [38%] (0.0193)	1.7620 [22%] (0.0377)	-0.4054 [28%] (0.0258)	2.3338 [15%] (0.0441)	-0.4605 [18%] (0.0248)	3.2031[10%] (0.0425)
$\Delta = 0.5$	-0.4221 [25%] (0.0202)	1.9356 [14%] (0.0402)	-0.4614 [18%] (0.0212)	2.4882 [10%] (0.0503)	-0.5010 [11%] (0.0229)	3.3216 [7%] (0.0553)
$\Delta = 0.7$	-0.4582 [19%] (0.0176)	2.0245 [10%] (0.0350)	-0.4904 [13%] (0.0181)	2.5458 [8%] (0.0416)	-0.5112 [9%] (0.0200)	3.3776 [5%] (0.0538)
$\Delta = 1.0$	-0.4831 [14%] (0.0181)	2.0676 [8%] (0.0351)	-0.5056 [10%] (0.0201)	2.5917 [6%] (0.0432)	-0.5298 [6%] (0.0211)	3.4250 [4%] (0.0435)

tical properties of estimated model with $\Delta = 0.7$. Overall, there is a very good agreement between the triad model and the estimated model for all three values $r = 1, 2, 3$. Therefore, we can conclude that the estimation procedure results in an effective model which reproduces the one-point statistical properties of the data very well, but the two-point statistics have some discrepancies due to multiscale effects.

Conclusions

We constructed the new type of estimator by combining the simulated maximum likelihood (SML) technique for using larger sub-sampling time step Δ, with LASSO approach for model selection. The SML technique allows using larger sub-sampling time step and, thus, reduce the influence of multiscale effects on estimation. Since SML is very computationally expensive, we use least squares approximation (LSA) of the likelihood function in the LASSO step, which considerably reduces the computational cost of the LASSO step. In addition, different scaling of the penalty terms for the estimation of the drift and diffusion coefficients produce adequate results.



(a) θ_1 (b) θ_4

Figure 3: Estimators $\hat{\theta}_1$ and $\hat{\theta}_4$ computed using SLASSO estimation and averaged over 100 Monte-Carlo realizations. We also plot the error bars using the standard deviation for these estimators in the Monte-Carlo simulations.

These are the main outcomes:

- The simulated LASSO-type estimator (SLASSO) performs model selection very well; this method is able to select the correct expression for the effective model for the slow variable x_t in the additive triad model.

- Large sub-sampling time step Δ is required for correct parametric estimation of the effective model. The estimation procedure can yield large errors (approximately 34%) when the sub-sampling time step is approximately equal to the time-scale of fast variables (e.g. estimation with $\Delta = 0.7$ and $r = 1$). Estimation errors decay quickly as the time-scale separation in increases.

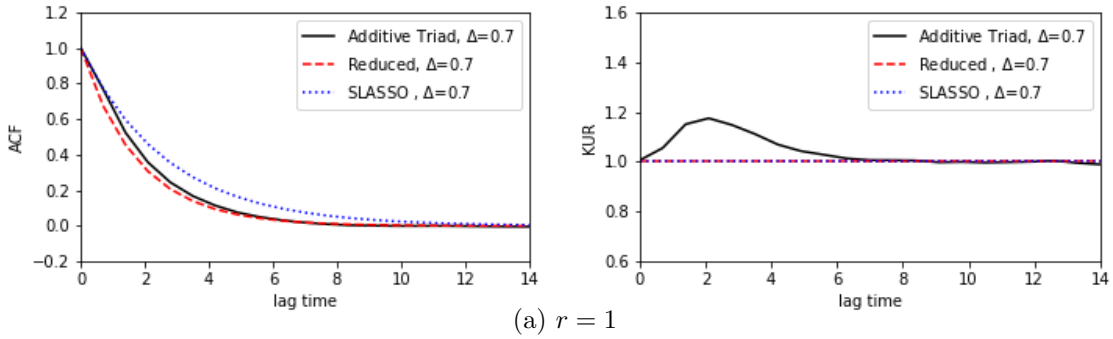
- There are two sources of error in the estimation problem - (i) truncation errors due to large Δ, and (ii) estimation errors due to multiscale effects present in the data. Minimizing truncation errors requires $\Delta \to 0$, and minimizing influence of multiscale effects requires $\Delta \to \infty$. Therefore, there should be a balance between these terms.

- Statistical properties of the slow variable are reproduced well for larger values of r. In particular, one-point statistics are reproduced well for $r = 1, 2, 3, 5$. Two-point statistics have some discrepancies for a small value of $r = 1$. These discrepancies decrease as r increases.

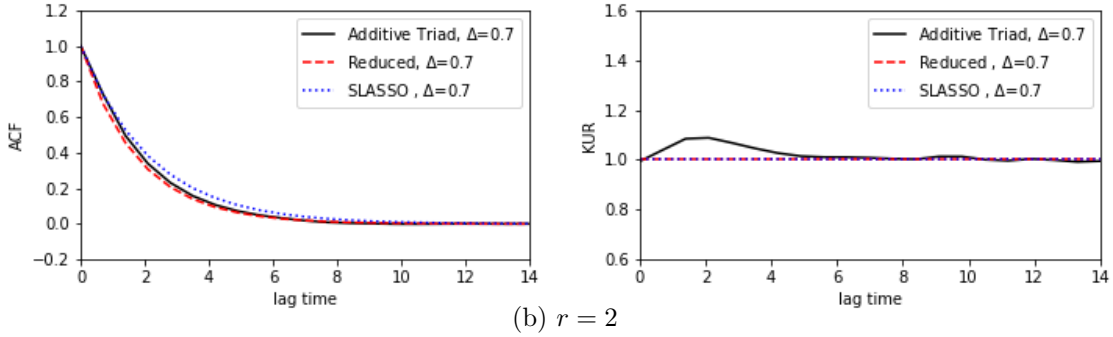
Table 5: One-point statistics(mean, variance, skewness, kurtosis, correlation time) of slow variable x_t in the additive triad model (3.1), the reduced model (3.3) and estimated SLASSO model (3.9). Statistics of the additive triad model is computed by numerical solution. The analytical statistics of the reduced model and estimated SLASSO model are presented. All result are from observation with $\Delta = 0.7$.

		Mean	Variance	Skewness	Kurtosis	Correlation Time
$r = 1$	Additive Triad	0.00	2.26	-0.01	3.00	2.06
	Reduced	0.00	2.25	0.00	3.00	1.78
	SLASSO	0.00	2.23	0.00	3.00	2.70
$r = 2$	Additive Triad	-0.01	4.55	0.01	2.98	1.93
	Reduced	0.00	4.50	0.00	3.00	1.78
	SLASSO	0.00	4.47	0.00	3.00	2.18
$r = 3$	Additive Triad	0.04	6.78	0.04	3.02	1.89
	Reduced	0.00	6.75	0.00	3.00	1.78
	SLASSO	0.00	6.60	0.00	3.00	2.03
$r = 5$	Additive Triad	0.01	11.14	0.04	3.01	1.95
	Reduced	0.00	11.25	0.00	3.00	1.78
	SLASSO	0.00	11.16	0.00	3.00	1.96

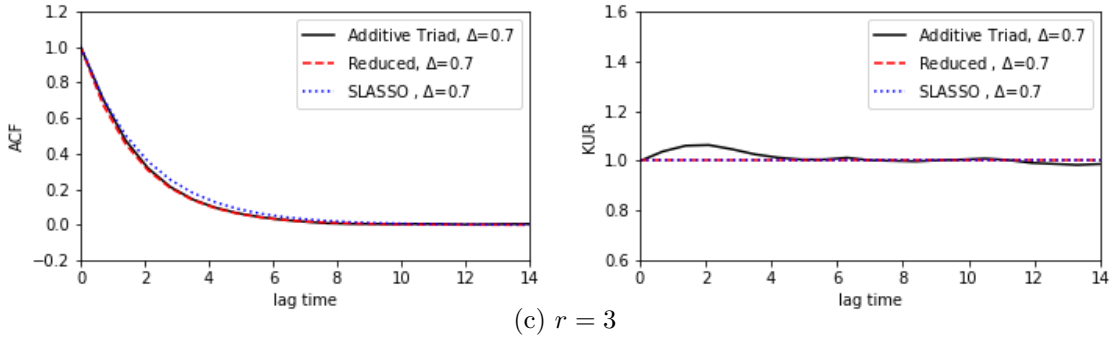
This indicates that these discrepancies are due to multiscale effects and the estimation of the parameter for the drift is most sensitive to the multiscale effects in the data.



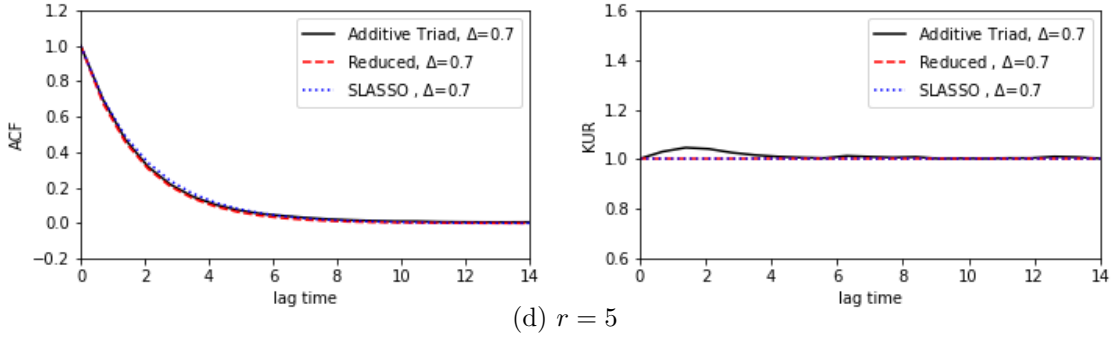
(a) $r = 1$



(b) $r = 2$



(c) $r = 3$



(d) $r = 5$

Figure 4: Comparison of the auto-correlation function (ACF) and kurtosis (KUR) in the simulations of the additive triad model and statistical properties of the estimated SLASSO model. All result are from observation with $\Delta = 0.7$. The statistics are computed for numerical solution for the additive triad model and analytical solution for the reduced model and estimated SLASSO model.

3.2 SLASSO Application to the Burgers Model

In this section, we apply the SLASSO estimator to a multi-dimensional example, namely, the forced Burgers model.

3.2.1 Forced Burgers Model

We consider a finite-difference approximation of the forced Burgers model with white-noise forcing in physical space

$$\frac{\partial}{\partial t}u + \frac{1}{2}\frac{\partial}{\partial x}u^2 = \nu u + \sigma\dot{W},$$

where W is a delta-correlated (both in space and time) Gaussian white noise. This system is a relatively high dimensional model that has interactions on a wide range of spatial scales. In particular, we consider the finite-difference approximation

$$\frac{d}{dt}u_i = -\frac{F_{i-1/2} - F_{i+1/2}}{\Delta x} + \nu u_i + \sigma\dot{W}_i. \tag{3.12}$$

Here, $u_i(t) \in \mathbb{R}^m$ denotes the value of the function $u(x,t)$ at the center of the discrete cell i, $(i \in s = \{0,\ldots,m-1\})$ of width $\Delta x = L/m$. W_i are independent Brownian motions, snd $F_{i-1/2}$ and $F_{i+1/2}$ denote fluxes at the boundaries of cell i. These fluxes are defined using values from neighboring cells,

$$F_{i-1/2} = \frac{1}{6}\left(u_i^2 + u_i u_{i-1} + u_{i-1}^2\right), \qquad F_{i+1/2} = \frac{1}{6}\left(u_{i+1}^2 + u_{i+1}u_i + u_i^2\right). \tag{3.13}$$

The direct expression of the forced Burgers model is written as

$$du_i = -\frac{1}{6\Delta x}(u_{i-1}^2 + u_{i-1}u_i - u_i u_{i+1} - u_{i+1}^2)dt + \nu u_i dt + \sigma dW_i, \quad i = 0,\ldots,m-1. \tag{3.14}$$

Equation above corresponds to a high-dimensional stochastic differential equation with additive noise.

Numerical Method for the Forced Burgers Model

The forced Burgers equations (3.14) are integrated numerically with the time step δt and data is sampled with time step Δ. We use an integration scheme similar to the operator splitting approach, where the deterministic part (drift) is integrated with a higher-order numerical method to avoid numerical instabilities. In particular, the third-order Runge-Kutta method is applied for the integration of the drift part. Then the diffusion part is added using the Euler scheme.

The numerical scheme can be represented as follows. If we define the drift part as

$$\mu_i(u) = -\frac{1}{6\Delta x}(u_{i-1}^2 + u_{i-1}u_i - u_i u_{i+1} - u_{i+1}^2) + \nu u_i,$$

then the Runger-Kutta scheme is defined as

$$\begin{aligned} K_1 &= \mu\left(u(t)\right), \\ K_2 &= \mu\left(u(t) + 0.5K_1\delta t\right), \\ K_3 &= \mu\left(u(t) + 0.75K_2\delta t\right), \\ K(t) &= \frac{1}{9}(2K_1 + 3K_2 + 4K_3), \end{aligned} \tag{3.15}$$

with $u(t), \mu \in \mathbb{R}^m$. Then the solution is updated according to the following scheme

$$u(t+\delta t) = u(t) + K(t)\delta t + \sigma \delta W, \tag{3.16}$$

where $\delta W \in \mathbb{R}^M$ is a vector of independent Gaussian random variables with mean zero and variance δt. We use $\delta t = 0.001$ in all simulations.

Local Averaged Variables for the Forced Burgers Model

Derivation and estimation of effective models is a well-known problem for many applications, and in fluid dynamics in particular. Here we follow the approach of Large-Eddy Simulations and define an

effective model for spatial averages. The goal is to reduce the spacial dimension of the problem and obtain an effective model which can be integrated on a coarser spatial grid. To this end, following the volume-balance procedure of [74, 1], we average over $n \equiv n_x$ neighboring, fixed in space, "fine" grid cells of the original width Δx. We define a coarse grid with a spacing of $n\Delta x$ and cell index set $S = \{0, \ldots, M-1\}$, where $M = m/n$ is the total number of coarse cells. Let $X_I, I \in S$ denote the averaged values of u_i. Then, X_I can be computed by applying a "box" filter in physical space. This filtering can be written simply as the arithmetic mean over n neighboring grid cells

$$X_I = \frac{1}{n} \sum_{k=nI}^{n(I+1)-1} u_k. \tag{3.17}$$

Next, we split u_i into a mean ub_I and a deviation y_i

$$u_i = X_I + y_i \tag{3.18}$$

The index I denotes the coarse cell in which the fine cell i is located. We also refer to X_I and y_i as resolved and unresolved modes, respectively. Then equations for averages and fluctuations become

$$\frac{d}{dt} X_I = -\frac{F_{n(I+1)-1/2} - F_{nI-1/2}}{n\Delta x} - \nu X_I + \frac{\sigma}{\sqrt{n}} \dot{W}_I, \tag{3.19}$$

$$\frac{d}{dt} y_i = -\frac{F_{i+1/2} - F_{i-1/2}}{\Delta x} + \frac{F_{n(I+1)-1/2} - F_{nI-1/2}}{n\Delta x}. \tag{3.20}$$

Since we define resolved modes as averages, a certain scale separation exists between the resolved and unresolved modes. This scale separation is controlled by the size of the averaging window n. When the averaging window is small, X_I is close to u_i. As n becomes larger, the scale separation between X_I and y_i increases. Equations (3.19) and (3.20) are equivalent to the full model (3.14). The goal here is to obtain an effective closed-form equation for the resolved modes X_I and eliminate the unresolved degrees of freedom y_i. This is motivated by the fact that in many realistic applications the unresolved degrees of freedom y_i cannot be observed explicitly. Thus, the observational dataset is generated by the resolved variables X_I in the full model (3.19), (3.20) and the goal is

to use this data to estimate the effective model for X_I. Since the effective estimated model for X_I is not known, we refer to this estimation context as *Indirect Observability*, i.e., the model for X_I is estimated from data generated by an approximate model. However, for $n = 1$ (no averaging), data are generated by the full model (3.14), which means the estimation procedure should recover the model (3.14). In this case, the estimated model is known, we refer to this context as *Direct Observability*. Thus, by varying the parameter n we can compare our estimation results under *Direct* and *Indirect Observability*.

3.2.2 Numerical Results

In this experiment, the parameters in the forced Burgers model (3.14) setting is

$$L = 100, \quad m = 512, \quad \nu = -0.1, \quad \sigma = 0.2. \tag{3.21}$$

We simulate numerically the full model $\delta t = 0.001$ and save observations with sub-sampling time-step $\Delta = 0.1$. We compare estimation results with three time-steps $\Delta t = 0.1, 0.2, 0.3$.

Figure 6 shows time-correlation function for the full variable u and slow variable X_I with averaging window $n_x = 2, 3, 4$. When the averaging window is very small, i.e., $n = 2$, correlation time of slow variable X_I is close to the correlation time of u_i in the full model. The gap between correlation times increases as n becomes larger.

The exact expression of the full Burgers model is known, but the effective model for the resolved variable X_I is unknown. When the averaging window is small, statistical behavior of the resolved variables X_I should be close to the full model. Therefore, we assume that the effective model for the resolved variables ub_I has functional form similar to the full Burgers model. In particular, we assume a polynomial form with linear and quadratic interactions, and additive and multiplicative noises. Our goal is to use SLASSO estimator to perform model selection and parameter estimation.

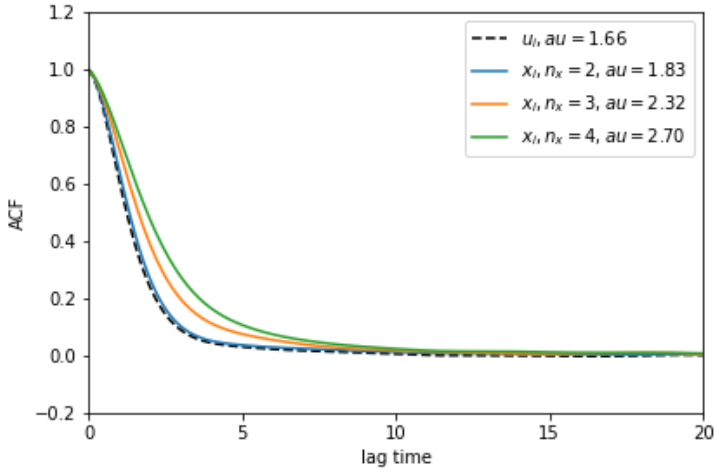


Figure 5: Auto-correlation function for the full variable u and slow variable X_I with averaging window $n = 2, 3, 4$.

Thus, we assume the following form for the estimated model

$$\begin{aligned}\frac{d}{dt}v_i =& \theta_1 v_{i-1}^2 + \theta_2 v_{i-1}v_i + \theta_3 v_{i-1}v_{i+1} + \theta_4 v_i^2 + \theta_5 v_i v_{i+1} + \theta_6 v_{i+1}^2 \\ &+ \theta_7 v_{i-1} + \theta_8 v_i + \theta_9 v_{i+1} + \theta_{10}\dot{W}_{1,i} + \theta_{11} v_i \dot{W}_{2,i}.\end{aligned} \tag{3.22}$$

Here $W_{1,i}$ and $W_{2,i}$ are independent Brownian motion. We know that if the data is generated by variables u_i in the full Burgers model 3.14, then estimation should result in 6 essential non-null parameters (θ_1, θ_2, θ_5, θ_6, θ_8, θ_{10}) and 5 unessential null parameters ($\theta_3 = \theta_4 = \theta_7 = \theta_9 = \theta_{11} = 0$).

The simulated LASSO-type estimator is based on the assumption that the polynomial expression of the effective model is the correct format and includes the "true" effective model as a subset. Our goal is to perform SLASSO estimator to efficiently classify essential variables which are included in the expression of the effective model. The SLASSO optimization procedure is discussed in section 2. It is computationally infeasible to perform model selection in one step when many redundant parameters, since the power (probability) of selecting zero-value variables may be decentralized. Thus, we perform SLASSO estimation sequentially for Burgers experiment, eliminating non-essential parameters in multiple steps. We consider performance of the estimation procedure with both weak

and strong penalty, such that $(\lambda_0, \gamma_0) = C_p(\sqrt{\Delta t}, 1)$, $C_p \in \{0.01, 0.001\}$, $\delta_1 = \delta_2 \in \{0.5, 1\}$.

Table 6: SLASSO model selection result for the Burgers Model. We present parameter values computed by SMLE (see section 3.1.2) after the model selection; standard deviations computed from 100 Monte-Carlo trajectories are presented in round brackets below the estimated value.

		θ_1	θ_2	θ_5	θ_6	θ_8	θ_{10}	θ_{11}	$\theta_{i^*} = 0$
Data	Burgers model	-0.8517	-0.8517	0.8517	0.8517	-0.1	0.2	0	
u	$\Delta t = 0.1$	-0.8453	-0.8545	0.8404	0.8484	-0.1188	0.2000	-	$\theta_3, \theta_4, \theta_7, \theta_9, \theta_{11}$
		(0.0264)	(0.0515)	(0.0501)	(0.0264)	(0.0198)	(0.0019)	-	
	$\Delta t = 0.2$	-0.8484	-0.8442	0.8466	0.8474	-0.1245	0.2007	-	$\theta_3, \theta_4, \theta_7, \theta_9, \theta_{11}$
		(0.0198)	(0.0380)	(0.0338)	(0.0191)	(0.0153)	(0.0020)	-	
	$\Delta t = 0.3$	-0.8530	-0.8431	0.8396	0.8544	-0.1236	0.2026	-	$\theta_3, \theta_4, \theta_7, \theta_9, \theta_{11}$
		(0.0185)	(0.0277)	(0.0253)	(0.0180)	(0.0103)	(0.0021)	-	
$X_I, n = 2$	$\Delta t = 0.1$	-0.4371	-0.4327	0.4351	0.4380	-0.1247	0.1512	0.1748	$\theta_3, \theta_4, \theta_7, \theta_9$
		(0.0526)	(0.1053)	(0.1208)	(0.0592)	(0.0247)	(0.0026)	(0.0199)	
	$\Delta t = 0.2$	-0.4356	-0.4268	0.4223	0.4399	-0.1443	0.1600	0.2426	$\theta_3, \theta_4, \theta_7, \theta_9$
		(0.0456)	(0.0747)	(0.0822)	(0.0530)	(0.0158)	(0.0025)	(0.0178)	
	$\Delta t = 0.3$	-0.4354	-0.4139	0.4139	0.4322	-0.1604	0.1696	0.2952	$\theta_3, \theta_4, \theta_7, \theta_9$
		(0.0404)	(0.0604)	(0.0643)	(0.0386)	(0.0146)	(0.0027)	(0.0161)	
$X_I, n = 3$	$\Delta t = 0.1$	-0.2929	-0.2733	0.3040	0.2977	-0.1222	0.1238	0.1199	$\theta_3, \theta_4, \theta_7, \theta_9$
		(0.0724)	(0.1000)	(0.1067)	(0.0740)	(0.0147)	(0.0020)	(0.0226)	
	$\Delta t = 0.2$	-0.3004	-0.2819	0.2852	0.2864	-0.1400	0.1307	0.1646	$\theta_3, \theta_4, \theta_7, \theta_9$
		(0.0607)	(0.0781)	(0.0738)	(0.0521)	(0.0122)	(0.0027)	(0.0302)	
	$\Delta t = 0.3$	-0.2913	-0.2777	0.2730	0.2894	-0.1566	0.1397	0.1980	$\theta_3, \theta_4, \theta_7, \theta_9$
		(0.0419)	(0.0570)	(0.0596)	(0.0429)	(0.0121)	(0.0031)	(0.0216)	
$X_I, n = 4$	$\Delta t = 0.1$	-0.2105	-0.2337	0.2186	0.2050	-0.1180	0.1061	0.0861	$\theta_3, \theta_4, \theta_7, \theta_9$
		(0.0801)	(0.1029)	(0.1086)	(0.0814)	(0.0172)	(0.0016)	(0.0302)	
	$\Delta t = 0.2$	-0.2129	-0.2014	0.2033	0.1987	-0.1343	0.1119	0.1185	$\theta_3, \theta_4, \theta_7, \theta_9$
		(0.0671)	(0.0802)	(0.0740)	(0.0535)	(0.0125)	(0.0016)	(0.0293)	
	$\Delta t = 0.3$	-0.2179	-0.2156	0.2221	0.2188	-0.1466	0.1180	0.1512	$\theta_3, \theta_4, \theta_7, \theta_9$
		(0.0470)	(0.0615)	(0.0703)	(0.0528)	(0.0104)	(0.0020)	(0.0237)	

Table 6 reports the Monte Carlo mean and standard deviations of SMLE and SLASSO estimates for the data generated by the full Burgers model and resolved variables X_I in the Burgers model with varied time-step of estimation $\Delta = 0.1, 0.2, 0.3$ and 100 Monte Carlo realization. SLASSO has capability to select all essential variables $(\theta_1, \theta_2, \theta_5, \theta_6, \theta_8, \theta_{10})$ correctly for the full Burgers model. Moreover, the model selection is consistent for three different estimation time-steps considered here. Simulated Maximum Likelihood Estimator correctly estimates values of parameters for $n = 1$ for all three estimation time-steps. For $n > 1$ we can see that the standard deviations for estimated parameters become considerably larger compared to $n = 1$. Nonlinear interaction coefficients θ_1, θ_2, θ_5, and θ_6 decrease as n becomes larger, but the standard deviation remains relatively large, indicating that there is a lot of uncertainty in the estimation of these coefficients. This is

an indication that the functional form of the effective model in (3.22) may not be sufficient to reproduce the behavior of resolved variables X_I for larger n.

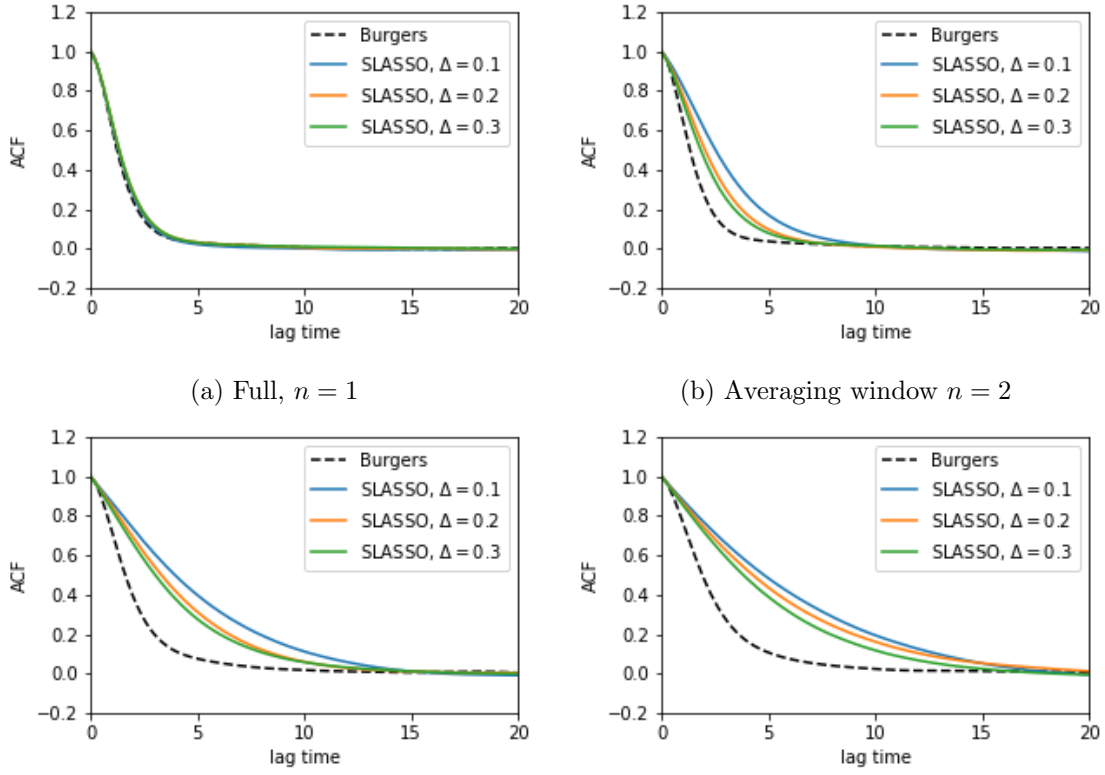
To check the accuracy of the estimation procedure, we regenerate time-series with the model (3.22) and coefficients given in Table 6. We then compare the statistical behavior of dynamic variables in the regenerated dataset and statistical behavior of the same variables in the original dataset. We set $\theta_1 = \theta_2 = \theta_5 = \theta_6$ equal to the mean of these four coefficients to avoid blow-up of solutions and guarantee stationarity of trajectories for the simulations of (3.22) Similar to the full Burgers model, we use the third-order Runge-Kutta method for the drift part and Euler discretization for the diffusion in (3.22).

Table 7: One-point statistics (mean, variance, skewness, kurtosis, correlation time) of resolved variables X_t in the Burgers model (3.14), and estimated SLASSO model (3.22).

		Mean	Variance	Skewness	Kurtosis	Correlation Time
$n = 1$	Burgers	0.00	0.20	0.01	3.02	0.61
	SLASSO, $\Delta = 0.1$	0.00	0.17	0.02	3.00	0.62
	SLASSO, $\Delta = 0.2$	0.00	0.16	0.02	3.00	0.55
	SLASSO, $\Delta = 0.3$	0.00	0.17	-0.01	3.01	0.53
$n = 2$	Burgers	0.00	0.10	0.01	3.02	0.55
	SLASSO, $\Delta = 0.1$	0.00	0.12	-0.02	2.99	0.33
	SLASSO, $\Delta = 0.2$	0.00	0.20	0.00	3.00	0.39
	SLASSO, $\Delta = 0.3$	0.00	0.27	0.01	3.00	0.41
$n = 3$	Burgers	0.00	0.07	0.00	3.01	0.43
	SLASSO, $\Delta = 0.1$	0.00	0.06	-0.02	2.96	0.20
	SLASSO, $\Delta = 0.2$	0.00	0.10	0.03	3.01	0.24
	SLASSO, $\Delta = 0.3$	0.00	0.12	0.01	2.97	0.25
$n = 4$	Burgers	0.00	0.05	-0.01	3.00	0.37
	SLASSO, $\Delta = 0.1$	0.00	0.03	0.00	2.96	0.17
	SLASSO, $\Delta = 0.2$	0.00	0.05	0.00	2.97	0.18
	SLASSO, $\Delta = 0.3$	0.00	0.08	0.07	2.98	0.20

Table 7 presents statistical features (mean, variance, skewness, kurtosis, and correlation time) for resolved variables X_I computed from the full Burgers model and the corresponding estimated SLASSO effective model. Overall, there is a good agreement between the Burgers model and the estimated model for four one-point moments, especially for $\Delta = 0.1$. However, there is a big discrepancy between the correlation times of resolved variables with $n > 1$ in the full Burgers and

estimated SLASSO models. In addition, we also observe that discrepancy for the the variance and kurtosis of resolved variables with $n > 1$ increases as Δ becomes larger.



(a) Full, $n = 1$

(b) Averaging window $n = 2$

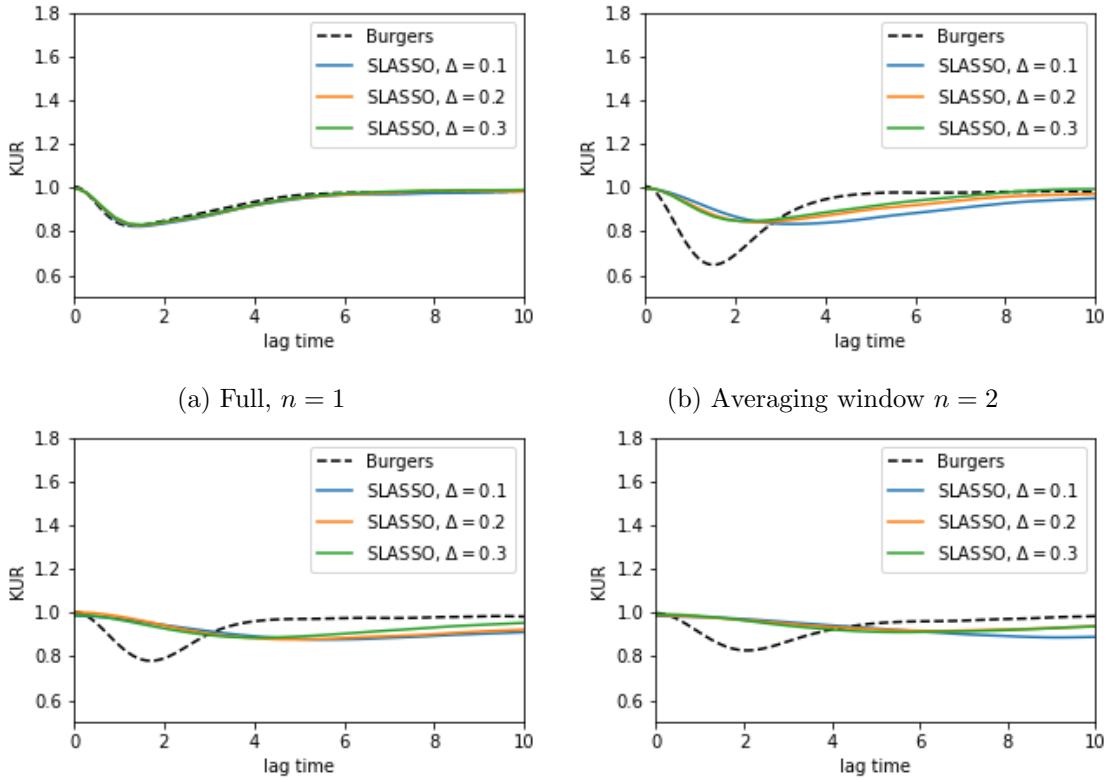
(c) Averaging window $n = 3$

(d) Averaging window $n = 4$

Figure 6: Comparison of the time auto-correlation function (ACF) of resolved variables X_I in the simulations of the estimated effective model and full Burgers model.

To further check the statistical performance of the effective model (3.22) with parameters in 6, the time auto-correlation function (ACF) and kurtosis function (KUR) are presented in Figures 6 and 7, respectively. The black dot-dashed line is auto-correlation of X_I computed from the full Burgers model. The solid lines are the auto-correlation functions computed from the regenerated trajectories using the effective model (3.22). The agreement is excellent under the *Direct Observability* ($n = 1$). The ACF function of the full Burgers model and SLASSO model are almost overlapping. However, there is a considerable discrepancy between the correlation function computed from the full Burgers and estimated effective models. In particular, the correlation function for the estimated effective models exhibits slower decay rate compared to the correlation function

computed from the full Burgers equation. The kurtosis measures deviations from Gaussianity, since for a Gaussian process it is exactly equal to one. Similar to the behavior of correlation functions, kurtosis for the estimated effective model with $n = 1$ overlaps with the kurtosis computed from the full Burgers model. However, we observe considerable discrepancies for the kurtosis in the full Burgers model and estimated model for $n > 1$.



(a) Full, $n = 1$

(b) Averaging window $n = 2$

(c) Averaging window $n = 3$

(d) Averaging window $n = 4$

Figure 7: Comparison of the kurtosis function (KUR) of resolved variables X_I in the simulations of the estimated effective model and full Burgers model.

Conclusions

In this section we utilized Simulated LASSO and Simulated Maximum Likelihood Estimators to compute effective models for resolved variables defined as "box" averages of fine-scale variables. The main outcomes are

- Our results demonstrate that the SLASSO algorithm is able to find null components (parameters equal to zero) for a range of observational time-steps when $n = 1$. This is the case when the correct effective model is among all possible models given by the general functional form of the estimated model (3.22).

- SLASSO and SMLE fail to select the correct model and/or correct parameter values for $n > 1$. This implies that the estimated effective model (3.22) does not contain an appropriate effective stochastic model for the resolved variables X_I with $n > 1$.

- We also observe that the agreement between the statistical behavior of resolved variables X_I in the full model and estimated model (3.22) improved for larger Δ. This is consistent with results for the triad model reported in section 3. This implies that the effect of unresolved variables is reduced for larger values of Δ.

4 Reservoir Computing

In this chapter, we describe the basic concept and prediction algorithms for Reservoir Computing, specifically the Echo State Network. We introduce the background material for the Reservoir Computing and use the L^2 relative error as a measure of prediction performance. We introduce the main parameters of the reservoir and discuss reservoir construction and also introduce various other machine learning terminology.

4.1 Background

The Echo State Network (ESN) [41, 42] is a particular case of Recurrent Neural Networks (RNNs), where the readout is linear. RNNs represent a large and varied class of computational models that are designed by more or less detailed analogy with biological brain modules. RNNs are used for a variety of scientific purposes. In particular, they appear as one of the most promising tools for nonlinear time series processing applications. It can be shown that RNNs are universal approximators of dynamical systems under fairly mild and general assumptions [75]. However, RNNs are difficult to train by gradient-descent-based methods. Training RNNs mainly suffers from expensive computation, vanishing or exploding gradients, and difficult optimization problems.

To remedy the fundamental problems with RNNs (in particular their slow and difficult training process), a fundamentally new approach to RNNs design and training was proposed independently by Wolfgang Maass in 2001 under the name of Liquid State Machines [55] and by Herbert Jaeger under the name of ESN [41]. Generation and training of ESNs is systematically summarized in [42, 54]. The essential component of the approach proposed in those papers is related to the training of RNNs using back-propagation [77]. This component is a particular approach for constructing RNNs and it is currently referred to as *Reservoir Computing*.

Reservoir Computing avoids the shortcomings of gradient-descent training for complex RNNs by setting up the structure of RNNs in the following way:

- A RNN is randomly created and remains unchanged during training. This RNN is called the

reservoir. It is passively excited by the input signal and maintains in its state a nonlinear transformation of the input history.

- The desired output signal is generated as a linear combination of the neuron's signals from the reservoir (excited by input). The weights of this linear combination are obtained by linear regression, using the training signal as a target.

The reservoir consists of many inter-connected neurons (possibly with connection loops) that are represented by time dependent variables. Thus, these neurons are triggered by the input signal and provide a dynamical memory. The information processing capacity of the reservoir is maximal, if the states are maximally de-correlated for the given input. Then the output layer can optimally combine these states to readout the desired output. Reservoir Computing is derived and theoretically justified by regarding learning as a constraint optimization problem [77]. In other words, the role of the reservoir is to nonlinearly transform sequential inputs into a high-dimensional space. Therefore, instead of RNNs, other nonlinear dynamical systems can be used as reservoirs. A motivation for physical implementation of reservoirs is to realize fast information processing devices with low learning cost.

Reservoir Computing received increasing attention in recent years. An overview of recent advances of Reservoir Computing is presented in [79], that discusses various details of Reservoir Computing including the advantage of modeling accuracy, modeling capacity and computational efficiency. In particular, our work is closely related to the application of Reservoir Computing to model-free prediction of the evolution of the state of dynamical systems [53, 65, 32, 99, 16, 86]. In our work we concentrate on a particular model of geophysical fluid dynamics.

4.2 Echo State Network Architecture

In this section, we introduce general formal definition of an ESN [54, 65]. An Echo State Network is a specialized simple type of RNNs which consists of the input layer, the recurrent hidden layer (the reservoir), and the output layer. Here we list the basic properties of the ESN

- Connections in the reservoir are generated randomly and not modified during training.
- Connections are encoded by a *adjacency matrix* A. A is generated as a sparse random matrix.
- In order to maintain high modeling capability, a relatively high-dimensional reservoir is used in the hidden layer of the ESN.
- Nonlinear transformation is introduced to update the state of the reservoir in time; this nonlinear transformation encodes the memory effect of the reservoir.
- Only the linear output layer is modified during training.

The connection of reservoirs represents as the adjacency matrix $A \in \mathbb{R}^{D\times D}$ with "echo state property" (ESP) [93]. The ESP is a condition of asymptotic state convergence of the reservoir network, under the influence of driving input. The ESP is connected to algebraic properties of the reservoir weight matrix A, and to properties of the driving input. It is a rather subtle mathematical concept. The ESP can be violated if the spectral radius of the adjacency matrix exceeds unity. Conversely, under rather general conditions, the ESP is obtained most of the time when the spectral radius of the adjacency matrix is smaller than unity. Neither does a spectral radius below unity generally ensure the ESP, nor does a spectral radius above unity generally destroy it. In numerous applications, a spectral radius well above unity serves best, depending on the nature of the driving input and the nature of the desired readout signal. The wide-spread practice of scaling the spectral radius below unity, thus, leads to an under-exploitation of the learning and modeling capacities of reservoirs.

Intuitively, the ESP states that the reservoir will asymptotically wash out any information from initial conditions. The ESP is guaranteed for *tanh* neuron reservoirs, if the spectral radius of the reservoir adjacency matrix A is below unity. The auto-feedback nature of RNN enable the reservoir states $r(t)$ to reflect traces of the past input history. It is fundamental to impose constraints on their initialization, to ensure dynamical stability in applications.

The schematic architecture of the ESN is shown in Figure 8. Both A and W_{in} are fixed and do not change during training. In our notation, N_x and D are used to indicate the input dimension

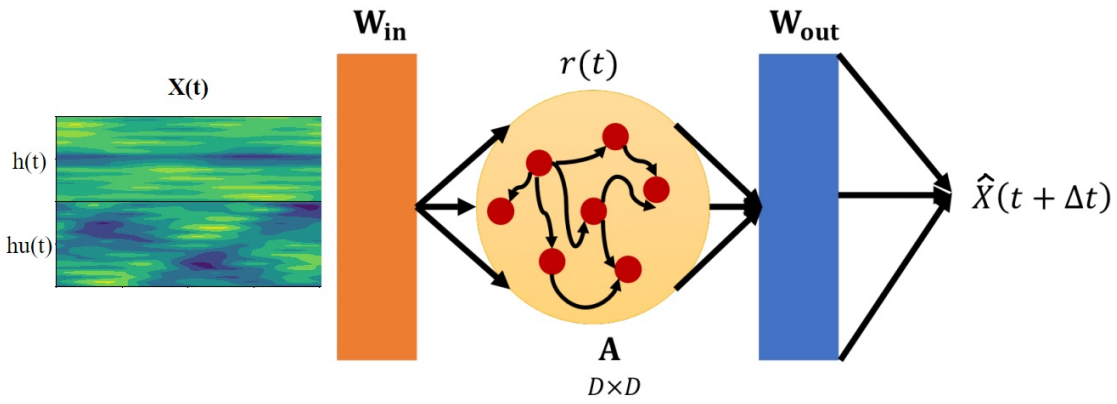


Figure 8: The schematic architecture of an ESN. Inputs $X(t) \in \mathbb{R}^{N_x}$ are fed into the reservoir through input connectivity matrix $W_{in} \in \mathbb{R}^{N_x \times D}$. The reservoir has hidden state $r(t) \in \mathbb{R}^D$, and recurrent connections are given by $A \in \mathbb{R}^{D \times D}$. Output $\hat{X}(t + \Delta t)$ is generated by a linear transformation $W_{out} \in \mathbb{R}^{D \times N_x}$ of reservoir states $r(t)$. Reservoir states $r(t)$ is updated in time using a nonlinear update function. In autonomous mode, predictions $\hat{X}(t + \Delta t)$ are fed back as inputs to the next time step in order to predict multiple time steps into the future. Note that W_{in} and A are fixed matrices. Only W_{out} is trained.

and the reservoir size, respectively. $W_{in} \in \mathbb{R}^{N_x \times D}$ represents random all-to-all connections between input units and the reservoir. The reservoir is driven by the sequential inputs. Each component signal $r(t + \Delta t)$ is a nonlinear transform of the driving input $X(t)$ and the previous state of the reservoir $r(t)$. During training, only weights of the output layer $W_{out} \in \mathbb{R}^{D \times N_x}$ are updated as the linear regression weights of outputs $X(t + \Delta t)$ on the reservoir states $r(t + \Delta t)$. There are three main parts of ESN, that are the input layer, the reservoir, and the output layer. Assume that we have input dataset $X \in \mathbb{R}^{N_x \times T}$, and each input vector $X(t) \in \mathbb{R}^{N_x}$ at each time step t.

Input-to-Reservoir Matrix W_{in} is usually generated from a particular distribution before the training. Here we use $W_{in} \sim Uniform[-\beta_1, \beta_1]$, where β_1 is a small parameter. The scaling of W_{in} and shifting of the input depends on how much nonlinearity of the processing unit the task needs: if the inputs are close to zero, the tanh neurons tend to operate with activations close to zero, where they are essentially linear, while inputs far from zero tend to drive them more towards saturation where they exhibit more nonlinearity. The shift of the input may help to overcome undesired consequences of the symmetry around zero of the tanh-neurons with respect to the sign of the signals. In this dissertation, we take $\beta_1 = 0.01$.

Adjacency Matrix A of the Reservoir. The adjacency matrix of the reservoir $A \in \mathbb{R}^{D\times D}$ is initialized randomly before the training and it does not change during the entire training process. As discussed previously, there are certain conditions on A which increase the possibility of the reservoirs having the echo state property. In particular, it has been observed empirically that the echo state property is very likely for any input if the spectral radius of A is smaller than unity. The connectivity of the reservoir neurons is represented by the adjacency matrix A of size $D \times D$ whose values are drawn from a highly spare uniform distribution with zero mean value. We ensure that the spectral radius of A is less than unity by first dividing the matrix A by its largest eigenvalue and further multiplying it by a scalar ($\beta_2 \leq 1$).

$$A = \beta_2 \frac{W_0}{|\omega_{max}|}, \tag{4.1}$$

where W_0 is a sparse matrix (usually less than 10% of connections), ω_{max} is the largest eigenvalue of W_0, and β_2 is a scaling parameter of the adjacency matrix. Thus, the spectral radius of A is

$$\rho(A) = \beta_2.$$

The optimal value of $\rho(A)$ should be set depending on the desired amount of memory and non-linearity for a particular task. As a rule of thumb discussed in [42, 41, 93], $\rho(A)$ should be close to 1 for tasks that require long memory, and $\rho(A) \ll 1$ for tasks where a long memory might be harmful. Larger $\rho(A)$ also have the effect of driving signals $X(t)$ into more nonlinear regions of tanh, similarly to larger W_{in}. Thus, scaling of both W_{in} and A have a similar effect on the degree of non-linearity of the ESN, while their difference determines the length of the memory effect.

Reservoir States $r(t)$. When the training data is input into the ESN, it will also activate the internal dynamics of the reservoir. The internal dynamical states of the reservoir is updated according to the equation

$$r(t + \Delta t) = F(r(t), X(t)) = f\left(A\, r(t) + W_{in} X(t)\right), \tag{4.2}$$

where $r(t) \in \mathbb{R}^D$ is the reservoir state, $f : \mathbb{R}^D \to \mathbb{R}^D$ is the element-wise application of a non-linear activation function (usually the logistic sigmoid or the tanh function). Before training, the reservoir state is initialized with zero, i.e., $r(0) = \vec{0}$.

Nonlinear Transformation of Reservoir States $r(t)$. A nonlinear transformation

$$\tilde{r}(t) = \psi\left(r(t)\right) \tag{4.3}$$

is often introduced in the prediction step of the ESN to increase the span of nonlinear features of the reservoir (see e.g. [65, 16]). Here $\psi : \mathbb{R}^D \to \mathbb{R}^D$. Usually function ψ is chosen as a simple element-wise nonlinearity. It was demonstrated in [65, 16] that the nonlinear transformation ψ increases the expression power of the dynamics reservoir. Thus, the reservoir update and prediction formulas become

$$r(t + \Delta t) = f\left(A\, r(t) + W_{in} X(t)\right), \tag{4.4}$$

$$\hat{X}(t + \Delta t) = W_{out}\psi(r(t)) \equiv W_{out}\tilde{r}(t). \tag{4.5}$$

A particular example of the nonlinear transformation ψ is

$$\tilde{r}_j(t) = \psi(r) = \begin{cases} r_j^2(t) & \text{if } j \text{ is odd} \\ r_j(t) & \text{if } j \text{ is even.} \end{cases}$$

Additional examples are presented in section 6.3.

Reservoir-to-Output Matrix W_{out}. Only the weights of the output-to-reservoir layer $W_{out} \in \mathbb{R}^{D \times N_x}$ are updated during training. The desired output weights W_{out} are the linear regression weights of the desired outputs with L^2 regularization

$$W_{out} = \underset{W}{\operatorname{argmin}}\, ||W\tilde{r} - X||_2^2 + \lambda ||W||_2^2, \tag{4.6}$$

where $W \in \mathbb{R}^{D \times N_x}$ and λ is the L^2 regularization (ridge regression) parameter. This is also known as Tikhonov regularization [35]. It can be easily shown that W_{out} can be computed explicitly for the Tikhonov regularization as

$$W_{out} = (\tilde{r}\tilde{r}' + \lambda I)^{-1}\tilde{r}X. \tag{4.7}$$

Here, $(\cdot)'$ is the transpose and $(\cdot)^{-1}$ is the inverse operation. This implies that training of the ESN can be done quite fast compared to other Neural Networks.

Reservoir Computing methods differ from other, more "traditional", ANN designs and learning techniques since Reservoir Computing approach makes a conceptual and computational separation between a dynamic reservoir (an RNNs as a nonlinear temporal expansion function) and a recurrence-free (linear in Echo State Network) readout that produces the desired output from the expansion. In this context reservoir state $r(t)$ expands the input history $X(t), X(t-\Delta t), \ldots$ into a rich enough reservoir state space, while readout combines the neuron signals $r(t)$ into the desired output $X(t+\Delta t)$. Since only the non-temporal readout part has to be learned, the training process is relatively simple and fast.

4.3 Prediction Setting

In this section, we describe the prediction setting for the ESN. Such setting arises if a closed form dynamic equations for predicting future states of a nonlinear system is not available or too costly to integrate numerically in real time. We also assume that sufficient training data for the evolution of the state of the system of interest can be observed or generated. Then, we can use Neural Networks and ESN in particular to predict the future state of the system.

Generating Prediction Trajectory. Given an initial condition $X^{true}(T_0)$, our goal is to use ESN to generate further predictions with a time step Δt, i.e., we want to generate time-series $X^{pred}(T_0 + \Delta t)$. In practice, initial time $T_0 = 0$ if no observational or numerical dynamic data is available. On the other hand, if some part of the trajectory $X^{true}(t)$, $t \in [0, T_0]$ (with $T_0 > 0$) is already known, then we can consider $X^{true}(T_0)$ as an initial condition and generate predicted

trajectory $X^{pred}(t)$ for $t > T_0$.

We would like to point out that the internal state of the reservoir, $r(T_0)$, may or may not be known. In particular, if $T_0 = 0$, then we do not have any prior information about the trajectory and we consider $r(0) = \vec{0}$. On the other hand, if some prior information is available, then we can use equation (4.2) to pre-compute the state of the reservoir where we use $X^{true}(t)$, $t \in [0, T_0]$ as an input to the ESN in (4.2).

We can use equation (4.5) to generate predictions starting with $X^{true}(T_0)$. In particular, we use $X^{true}(T_0)$ and $r(T_0)$ as an input into the ESN in (4.2), thus, updating internal states and computing $r(T_0 + \Delta t)$. Next, since W_{out} has been obtained by training, we use equation (4.5) to generate $\hat{X}(T_0+\Delta t)$ and set $X^{pred}(T_0+\Delta t) = \hat{X}(T_0+\Delta t)$. Thus, one time-stepping update consists of both, generating the prediction using (4.5) and reservoir update (4.2) and can be summarized as follows

$$X^{pred}(t + \Delta t) = W_{out}\tilde{r}(t), \tag{4.8}$$

$$r(t + \Delta t) = f\left(A\, r(t) + W_{in}X^{pred}(t)\right). \tag{4.9}$$

We then proceed recursively by using (4.9) with $(X^{pred}(T_0 + k\Delta t), r(T_0 + k\Delta t))$ to generate $(X^{pred}(T_0 + (k + 1)\Delta t), r(T_0 + (k + 1)\Delta t))$.

Prediction Error: The L^2 relative error is used here to quantify the performance of the ESN. In particular, we generate an ensemble of "true" $X_i^{true}(t)$ (with $i = 1, \ldots, N$, where N is ensemble size) trajectories using a suitable numerical scheme and compare them with predicted values $X_i^{pred}(t)$. Then, the L^2 relative error for each trajectory [16] is computed as follows

$$e_{L^2}^{(i)}(t) = \frac{||X_i^{true}(t) - X_i^{pred}(t)||_{L^2}}{< ||X_i^{true}(t)||_{L^2} >}. \tag{4.10}$$

Here $|| \cdot ||_{L^2}$ is the L^2 norm and $< \cdot >$ is averaging over the time interval for the prediction. In

addition, we define the prediction error over N samples as

$$E_{L^2}(t) = \frac{1}{N}\sum_{i=1}^{N} e_{L^2}^{(i)}(t). \tag{4.11}$$

In-sample Prediction. We train the ESN network on the time interval $[0, T_0]$ using the training dataset. In-sample prediction starts by continuing a trajectory from the training dataset further, i.e., using $X^{true}(T_0)$ in training dataset as an initial condition for future prediction. Thus, we generate predicted values $X^{pred}(T_0 + k\Delta t)$ and compare them with with $X^{true}(T_0 + k\Delta t)$ in training dataset. Since ESN already has been trained on the same trajectory $X^{true}(t)$ for $t \in [0, T_0]$, reservoir states $r(t)$ have been updated using (4.2) and we use reservoir states at time T_0 to predict $X^{pred}(T_0 + \Delta t)$. We then continue generating predictions $X^{pred}(T_0 + k\Delta t)$ as discussed earlier in this section.

Out-of-sample Prediction. We perform out-of-sample validation to realistically test the forecasting performance of the ESN. This means that we train the ESN on the training dataset, but testing the prediction utility on different (testing) data. Thus, we use the initial condition $X^{true}(T_0)$ in training dataset as an initial condition and utilize ESN to generate values $X^{pred}(T_0 + k\Delta t)$ describing the trajectory. We also assume that we do not have any prior knowledge of the testing trajectory, and therefore, the reservoir states are taken as $r(T_0) = \vec{0}$. Similar to the in-state prediction, we use (4.9) to generate $X^{pred}(T_0+k\Delta t)$ and compute the L^2 error between the predicted trajectory and $X^{true}(T_0 + k\Delta t)$ in testing dataset.

Out-of-sample Prediction with Reservoir Warm-up. Internal reservoir states $r(t)$ represent the long-term memory of the reservoir from the previous internal states and input trajectory. In other words, the reservoir state present the long-term memory from historical inputs. The echo state property states that the effect of this long-term memory of future predictions should vanish gradually as time lag increases. In particular, the effect of the reservoir state $r(T_0)$ should decay gradually for future predictions $X^{pred}(T_0 + k\Delta t), r(T_0 + k\Delta t)$, as lag $k\Delta t$ increases. However, for relatively small values of k this effect can be considerable. Thus, we can use the *reservoir warm-up*

to improve the prediction utility, especially for small lags k. For $T_0 > 0$ the reservoir warm-up requires several prior value of the "true" trajectory $X^{true}(T_0 - q\Delta t)$ with $q = 1, \ldots, Q$. The parameter $Q\Delta t$ is the warm-up size. Then, we use equation (4.2) with $X^{true}(T_0 - q\Delta t)$, $q = Q, \ldots, 1$ to warm-up the internal reservoir states and carry out prediction starting at time T_0 with $r(T_0) \neq 0$. This allows us to obtain a more realistic internal reservoir state $r(T_0)$. The optimal warm-up size depends on the amount of memory the given task requires. Warm-up size is also related to the spectral radius. Typically, for models with long memory, the spectral radius of the connectivity matrix is $\rho(A) \approx 1$ and such models require a relatively large warm-up size $Q\Delta t$. For models with weak long-term dependence the spectral radius is typically small, i.e., $\rho(A) \ll 1$ and the warm-up size can be taken also small as $Q = 2$ or even $Q = 1$.

5 Shallow Water Equations

This chapter discusses the decaying turbulence, one-dimensional shallow water equations. The content is arranged as follows. Shallow-water equations, domain geometry, and numerical method for generating solutions and training dataset are introduced in section 5.1. First, we consider a standard test with a hill topography in the middle of the domain. The numerical fluxes are computed with centered discretization by the energy conservative scheme [26], which is well-balanced and energy-conserved. The computation of Lyapunov exponents is discussed in section 5.2. The chaotic characterization of the one-dimensional shallow water equation is discussed with respect to different perturbations of the background solution.

5.1 One-dimensional Shallow Water Equations

The Shallow Water Equations (SWE) are a usual model to describe fluid flow in rivers, channels, estuaries or coastal areas. The main assumption of the shallow water model is that the horizontal length scale is much greater than the depth scale. Thus, one can get rid of the vertical dimension by averaging the mass and momentum conservation equations over the depth.

In one dimension of space, the shallow water equations are defined on the space domain $\Omega = (0, L)$ and the time interval $t = (0, T)$:

$$\partial_t h + \partial_x(hu) = 0, \tag{5.1}$$

$$\partial_t(hu) + \partial_x(hu^2 + \frac{1}{2}gh^2) + gh\partial_x z - \nu\partial_{xx}(hu) = 0, \tag{5.2}$$

where t and x denote time and space respectively, $h(x,t)$ is the water height, $u(x,t)$ is the fluid velocity, g is the gravitational constant, $\nu > 0$ is viscosity constant, $z(x)$ is the bottom topography, which does not depend on time. We use periodic boundary conditions $h(x,t) = h(x+L,t)$, $u(x,t) = u(x+L,t)$.

There is no analytical solution to this problem, therefore a reference solution is computed using a energy conservative scheme [26] on a fine grid. It is a second-order accurate approximation of

the one-dimensional shallow water system. The scheme is well-balanced, energy-conserving, and preserves the "lake at rest" equilibrium solution [26].

In our work, numerical solution of the SWE is obtained on an equipartitioned time-space grid. In particular, the time interval is divided into N_t time steps of length Δt and for all $n = \{0, \ldots, N_t\}$, $t^n := n\Delta t$. The domain is divided into N_x cells of length Δx. The left end, center, and right end of the i-th cell are denoted by $x_{i-\frac{1}{2}}$, x_i, and $x_{i+\frac{1}{2}}$, respectively. The water height h and topography z are discretized at the center of the cells, whereas velocity u is discretized at the interfaces between the cells.

Let we set $M := \{1, \ldots, N_x\}$. The mass conservation equation and momentum balance equation are discretized with an energy conservative (when $\nu = 0$) scheme. For $i \in M$ the discrete scheme reads

$$\frac{d}{dt}h_i = -\frac{1}{\Delta x}\left(\bar{h}_{i+\frac{1}{2}}\bar{u}_{i+\frac{1}{2}} - \bar{h}_{i-\frac{1}{2}}\bar{u}_{i-\frac{1}{2}}\right), \tag{5.3}$$

$$\begin{aligned}\frac{d}{dt}h_i u_i = &-\frac{1}{\Delta x}\left(\bar{h}_{i+\frac{1}{2}}\bar{u}^2_{i+\frac{1}{2}} + \frac{g}{2}\bar{h}^2_{i+\frac{1}{2}} - \bar{h}_{i-\frac{1}{2}}\bar{u}^2_{i-\frac{1}{2}} - \frac{g}{2}\bar{h}^2_{i-\frac{1}{2}}\right) \\ &- \frac{g}{2\Delta x}\left(\bar{h}_{i+\frac{1}{2}}\tilde{z}_{i+\frac{1}{2}} + \bar{h}_{i-\frac{1}{2}}\tilde{z}_{i-\frac{1}{2}}\right) \\ &+ \frac{\nu}{(\Delta x)^2}\left(\tilde{h}_{i+\frac{1}{2}}\tilde{u}_{i+\frac{1}{2}} - \tilde{h}_{i-\frac{1}{2}}\tilde{u}_{i-\frac{1}{2}}\right),\end{aligned} \tag{5.4}$$

where

$$\tilde{a}_{i+\frac{1}{2}} = a_{i+1} - a_i, \qquad \bar{a}_{i+\frac{1}{2}} = a_{i+1} + a_i, \tag{5.5}$$

denote the jump and the average of a quantity a across the interface $x_{i+1/2}$, respectively.

The Courant–Friedrichs–Lewy (CFL) condition is a condition for the stability of numerical methods that model convection or wave phenomena. The time step Δt is determined by a standard CFL condition. The water height remains non-negative [98] at time t_{n+1} for a CFL number of $\frac{u\Delta t}{\Delta x} \leq 0.45$.

We present a standard numerical experiment considered in many papers, including [36]. The

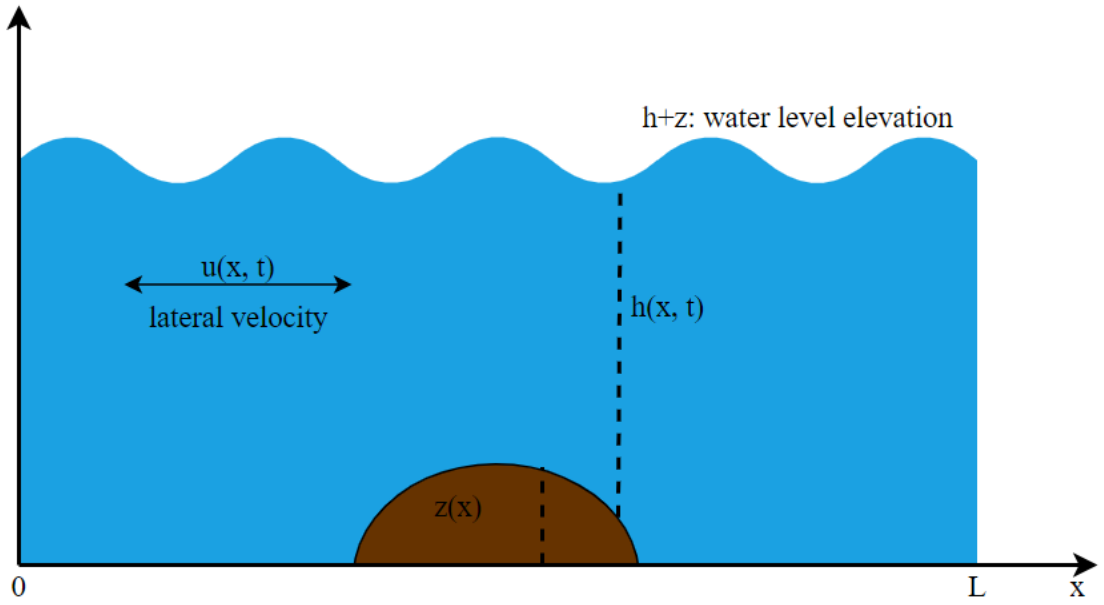


Figure 9: Schematic representation for the physical domain and dependent variables for the one-dimensional shallow water model.

bottom topography is a parabolic "bump" in the middle of the domain $\Omega := [0, L]$, with height TH and width TW.

$$z(x) = \begin{cases} TH\left(1 - \left(\dfrac{x - 0.5L}{0.5TW}\right)^2\right) & \text{if } |x - 0.5L| \le 0.5TW \\ 0 & \text{otherwise.} \end{cases} \tag{5.6}$$

Here, we study the creation of water waves starting from a flat surface. In particular, we choose the initial state with a flat initial water level and constant initial velocity

$$h(x, t = 0) + z(x) = H_0, \qquad u(x, t = 0) = U_0. \tag{5.7}$$

We also consider various perturbations of this initial state.

Table 8: Parameters for one-dimensional shallow water equations.

L	g	ν	H_0	U_0	TH	TW	Δx	Δt
40	32	0.50	4.0	2.5	0.48	8.0	0.1	0.0005

We consider the setup summarized in Table 8. The numerical solution is computed with the energy conservative scheme on a fine grid with $\Delta x = 0.1$ and $\Delta t = 0.0005$. The domain $L = 40$, gravitational constant $g = 32$ and viscosity constant $\nu = 0.5$. The parameters TH and TW correspond to the topography height and width, respectively. Figure 10 presents a 3D plot of the water height $h(x,t) + z(x)$ and snapshots of the water height, momentum and velocity.

Perturbation is a very small disturbance of the background state. One of the goals here is to study how these disturbances propagate in time. The $a\%$ perturbation level of water heights means the initial water level $h + z(t = 0)$ ranges form $(1 - a\%) \times H_0$ to $(1 + a\%) \times H_0$ with mean value H_0, which is the water level for the background state. For example, $h+z(x, t = 0) = 4+0.2\times\sin(2\pi x/L)$ is a example of 5% perturbation level on initial water heights in the first experiment. We generate perturbations of the background state (5.7) by considering the following initial conditions

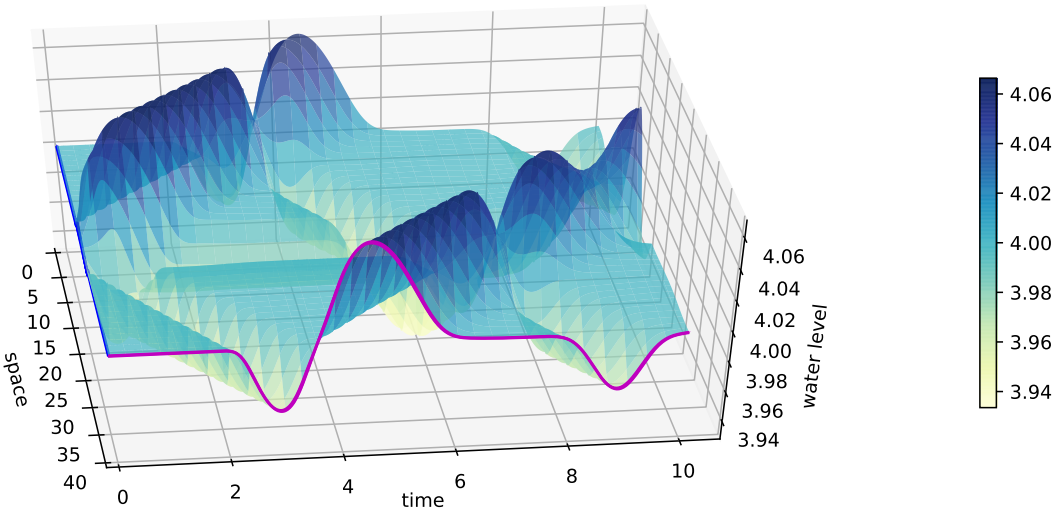
$$h(x,0) + z(x) = \begin{cases} H_0 \pm aH_0 \sin(2k\pi(x - \phi_1 L)/L), & x \in [\phi_1 L, \phi_2 L], \\ H_0, & \text{otherwise}, \end{cases} \tag{5.8}$$

$$u(x,0) = U_0.$$

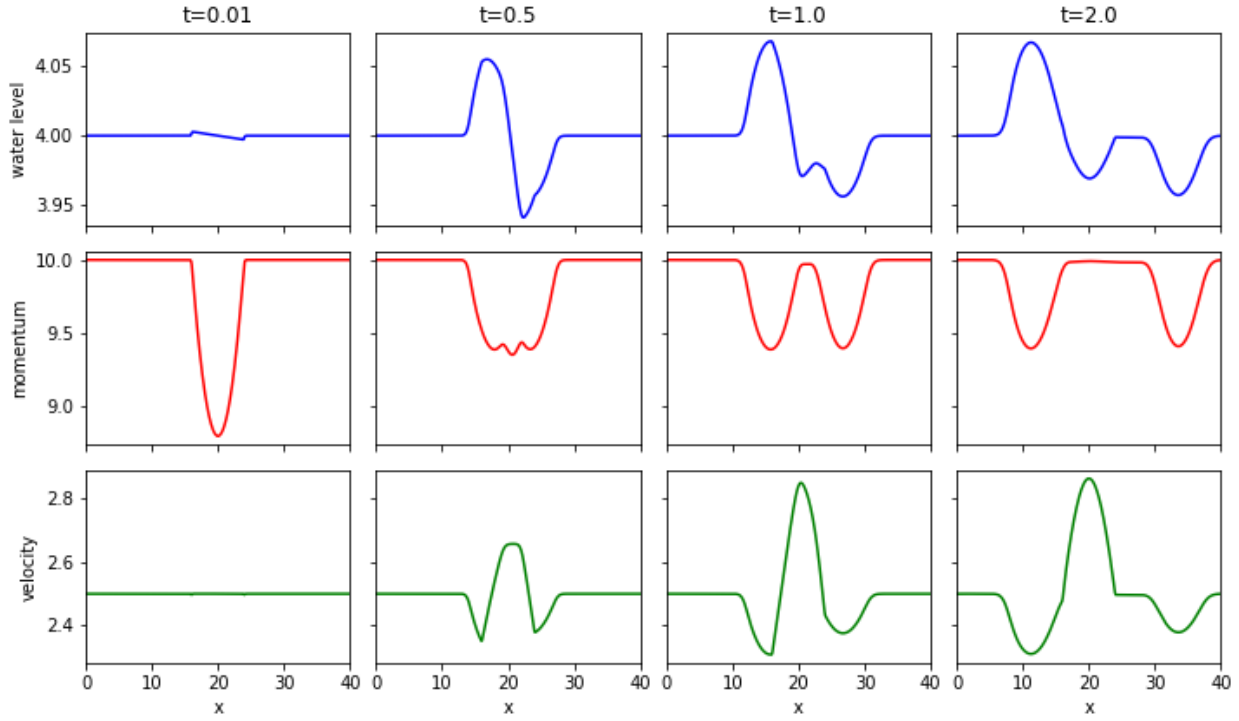
Here, a is the perturbation level. Typically, we randomly choose 1%- 10% perturbation level, such that $a \in [0.01, 0.1]$. The frequency of the perturbation is determined by $k = \{1, 2, \ldots, 10\}$, which is also chosen at random. Parameters ϕ_1 and ϕ_2 control the shift and position of the perturbation. In particular, we first generate an integer $b \in Uniform[1, \ldots, k]$ which controls the fraction of the domain where the perturbation is non-zero, then $\phi_1 \sim Uniform[0, 1 - b/k]$ and $\phi_2 - \phi_1 = \dfrac{b}{k}$. The velocity is taken as constant U_0 for all simulations in the chapter 6.

5.2 Lyapunov Exponent

Lyapunov exponents of a dynamical system quantify the rate of separation (or compression) of trajectories with nearby initial conditions along with various directions in the phase space. Quantitatively, two trajectories in phase space with initial (small) separation vector δX_0 diverge at a rate



(a) 3D plot



(b) Snapshots

Figure 10: Simulations of the 1D shallow water equation with initial conditions (5.7). Part a - water height $h(x,t)+z(x)$. Part b - snapshots the water level $h(x,t)+z(x)$ (blue line), momentum $hu(x,t)$(red line) and velocity $u(x,t)$(green line) at time $t = \{0.01, 0.5, 1.0, 2.0\}$.

given by $|\delta X(t)| \approx e^{\lambda t}|\delta X_0|$. This expression is valid if the dynamics of $X(t)$ is well-approximated by a linear system. If $X(t) \in \mathbb{R}^n$, then the rate λ can be different for vectors δX_0 oriented along different directions. The maximal (or largest) Lyapunov exponent is the largest separation rate over all possible directions. In particular, the maximal Lyapunov exponent plays an important since it characterizes the overall separation (or compression) rate of nearby trajectories and determines the notion of predictability for a dynamical system. A strictly positive maximal Lyapunov exponent is often considered as a definition of deterministic chaos.

For each sample $X_i(t)$, we compute the Lyapunov exponent

$$\lambda_i(t) = \frac{1}{t} \ln \frac{||X_i(t) - X_0(t)||_{L^2}}{||X_i(0) - X_0(0)||_{L^2}}, \tag{5.9}$$

where $X_0(t)$ is the background state (5.7). The trajectory Maximal Lyapunov exponent is $\lambda_i^{max} = \max_t \lambda_i(t)$. Expression (5.9) is easily interpretable for short times when the dynamics of the SWE is well-approximated by a linear system. For larger times, nonlinear effects become important, but $\lambda_i(t)$ can still be understood as the rate of separation of two trajectories. In addition, to better understand the separation of trajectories for all dynamic variables in the SWE, we take $X(t)$ as different combinations of dependent variables. In particular, we compute Lyapunov exponents for four cases - (i) $X(t) \equiv \{h(x_j, t)\}$, (ii) $X(t) \equiv \{hu(x_j, t)\}$, (iii) $X(t) \equiv \{hu(x_j, t), h(x_j, t)\}$, (iv) $X(t) \equiv \{u(x_j, t)\}$, where in all four cases $j = 1, \ldots, N_x$.

Lyapunov Exponent for the First Experiment. Next, we examine the behavior of finite-time Lyapunov exponents for the SWE with parameter settings (Table 8) for the background state (5.7) and nearby trajectories generated with the following initial conditions

$$h(x,0) + z(x) = \begin{cases} 4 \pm 4a \, \sin(2k\pi(x - \phi_1 L)/L), & x \in [\phi_1 L, \phi_2 L], \\ 4, & \text{otherwise}, \end{cases} \tag{5.10}$$

$$u(x,0) = 2.5.$$

Here, a is the perturbation level taken in the range 1% - 10%, k determines the frequency of the perturbation in the range $k = \{1, \ldots, 10\}$. Parameters ϕ_1 and ϕ_2 determine the shift of the perturbation and the domain where the perturbation is non-zero. In particular, these parameters are computed as follows

$$b \sim Uniform[1, \ldots, k], \quad \phi_1 \in [0, 1 - \frac{b}{k}], \quad \phi_2 = \phi_1 + \frac{b}{k}.$$

The perturbation is non-zero in the whole domain $[0, L]$ when $b = k$, and perturbation "occupies" only a fraction b/k of the domain if $b < k$.

Figures 11, 13, and 15 depict the numerical results of computing $\lambda_i(t)$ for some particular initial conditions given as perturbations of the background state (5.7). In particular, Figure 11 depicts the influence of the frequency of perturbation, Figure 13 illustrates the influence of the magnitude of perturbation, and Figure 13 shows the influence of the localized position of perturbation. In addition, Figures 12, 14, and 16 depict snapshots of the numerical solution for the corresponding perturbations in Figures 11, 13, and 15, respectively.

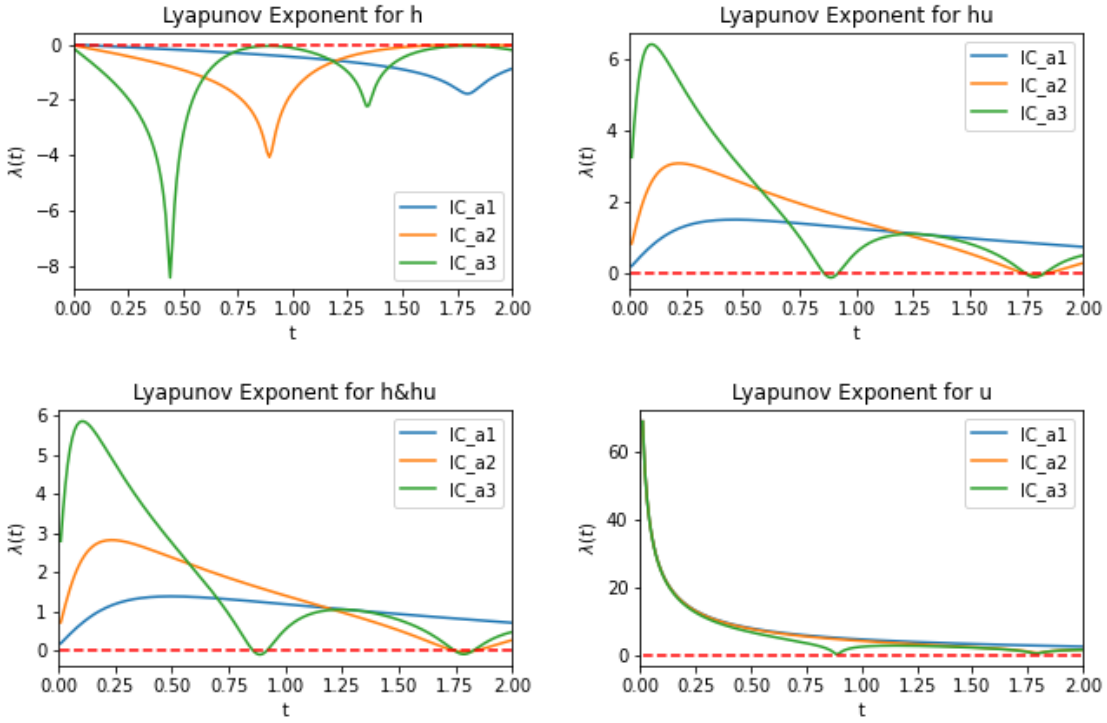


Figure 11: Numerical computation of Lyapunov exponents using (5.9) for initial conditions $h(x,0)+z(x) = 4 + 0.04\sin(k\pi x/L)$, and $u(x,0) = 2.5$; Blue line - $IC_{a1} : k = 2$, Orange line - $IC_{a2} : k = 4$, green line - $IC_{a3} : k = 8$. black line - "flat" IC in (5.7).

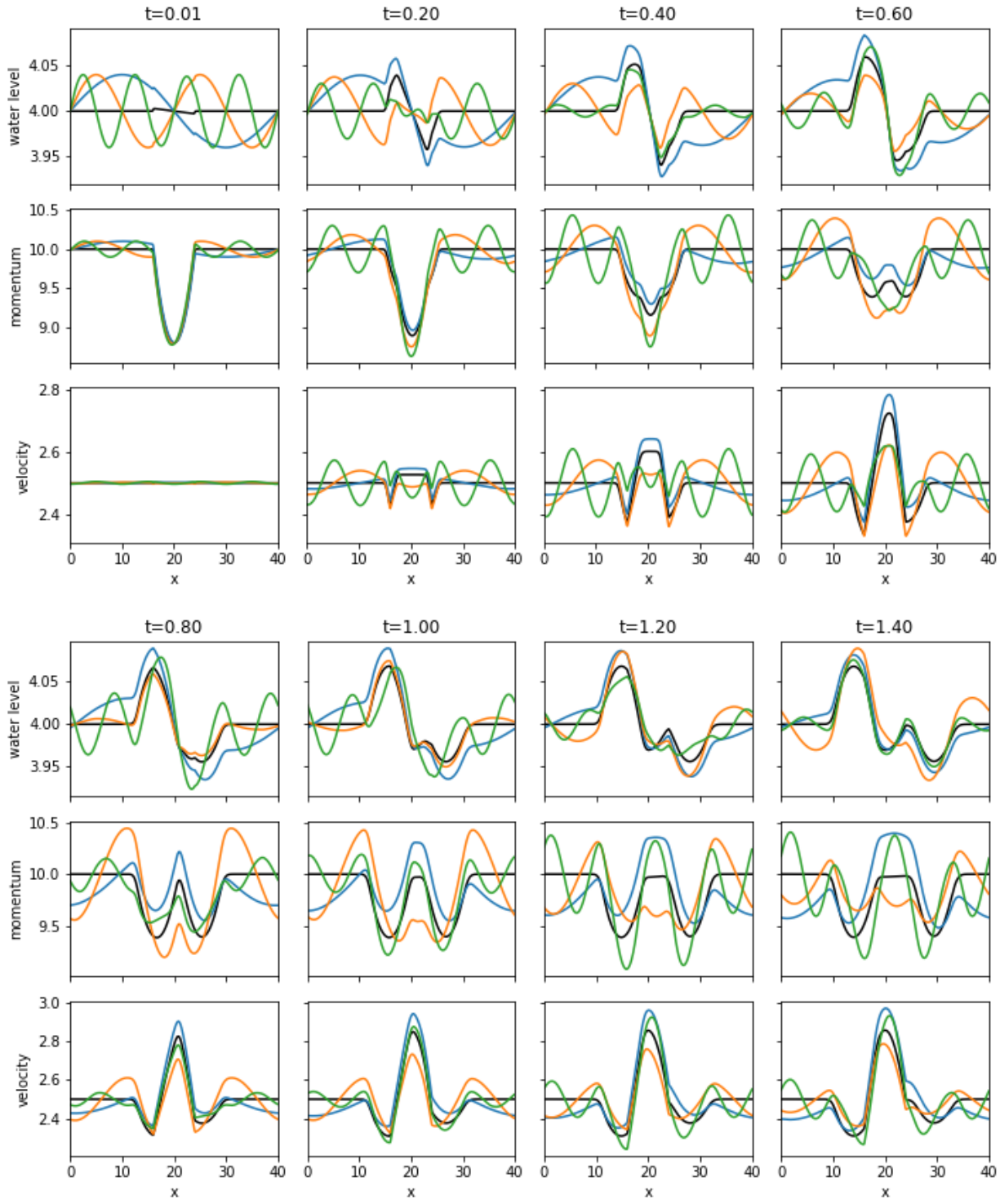


Figure 12: Snapshots of solutions corresponding to computation of Lyapunov exponent in Figure 11. Initial conditions are $h(x,0) + z(x) = 4 + 0.04\sin(k\pi x/L)$, and $u(x,0) = 2.5$; Blue line - $IC_{a1} : k = 2$, orange line - $IC_{a2} : k = 4$, green line - $IC_{a3} : k = 8$. black line - "flat" IC in (5.7).

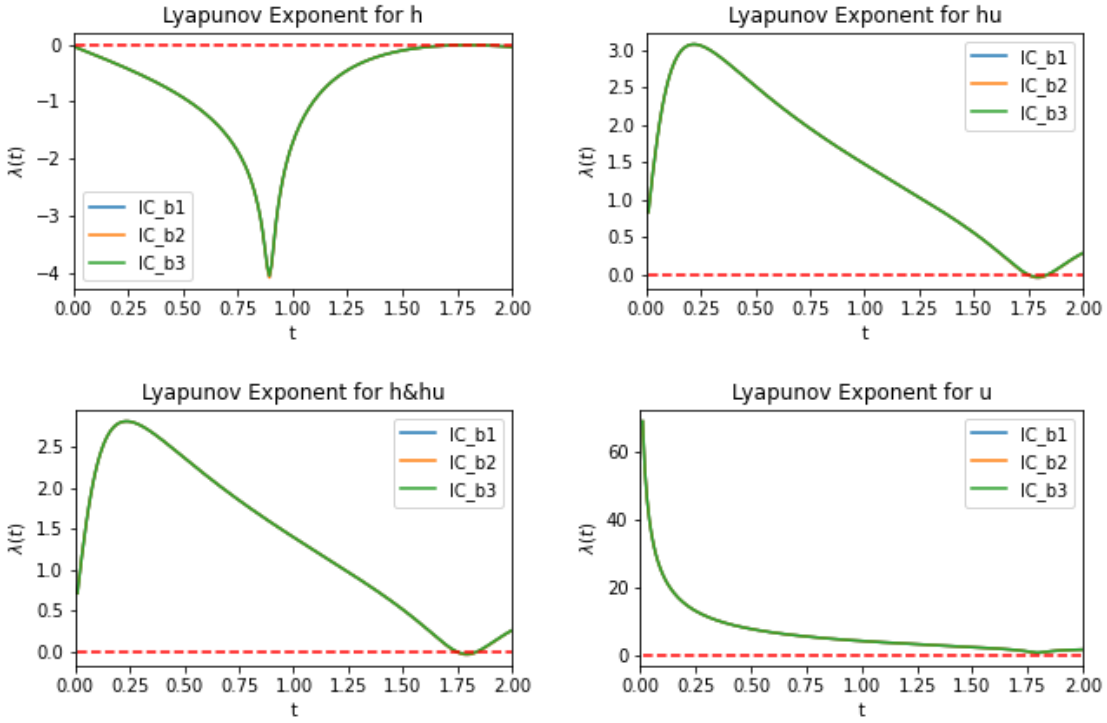


Figure 13: Numerical computation of Lyapunov exponents using (5.9) for initial conditions $h(x,0)+z(x) = 4+a\sin(4\pi x/L)$, and $u(x,0) = 2.5$; Blue line - $IC_{b1} : a = 0.04$, orange line - $IC_{b2} : a = 0.08$, green line - $IC_{b3} : a = 0.2$, black line - "flat" IC in (5.7).

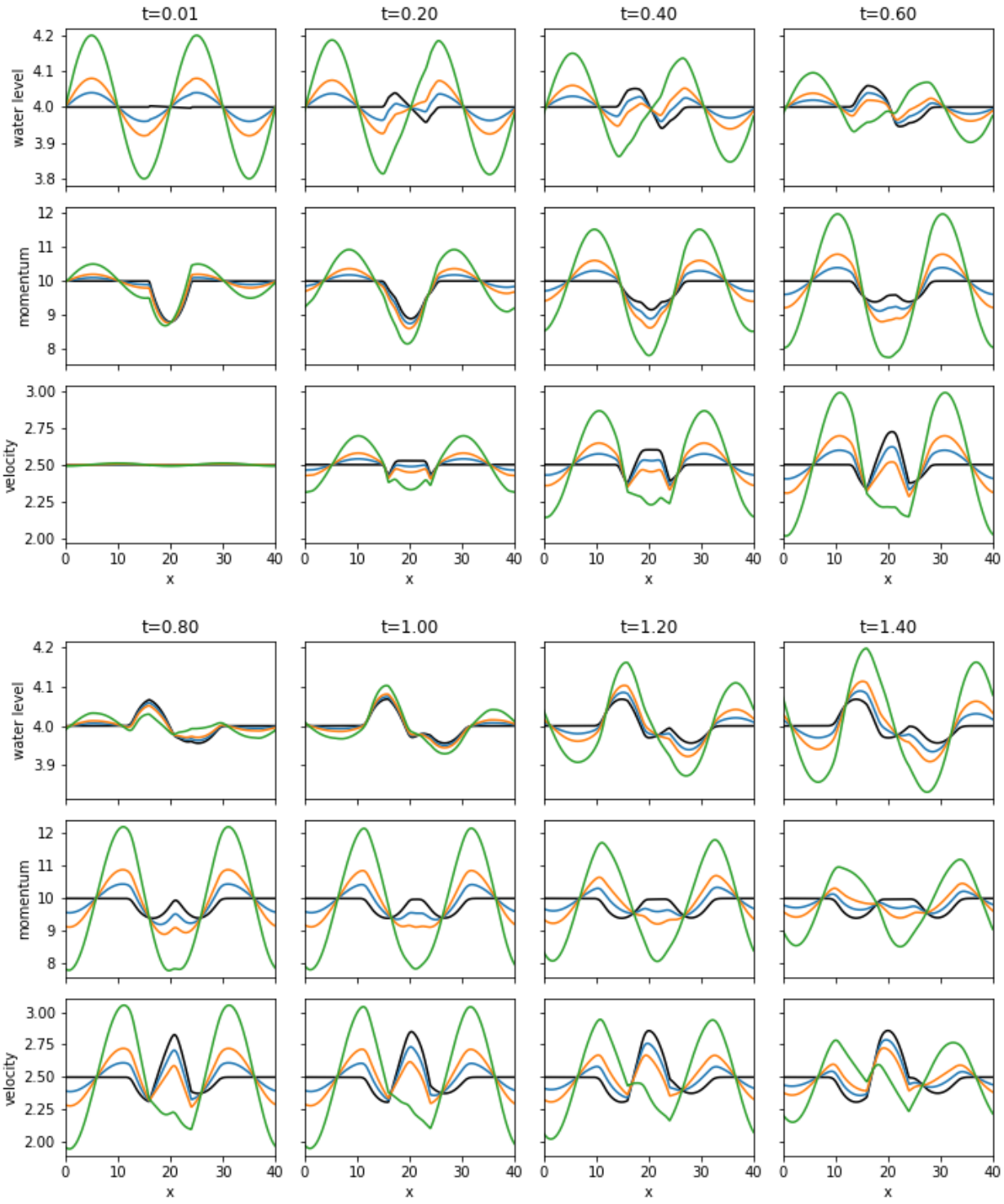


Figure 14: Snapshots of solutions corresponding to computation of Lyapunov exponent in Figure 13. Initial conditions are $h(x,0)+z(x) = 4+a\sin(4\pi x/L)$, and $u(x,0) = 2.5$; Blue line - $IC_{b1} : a = 0.04$, orange line - $IC_{b2} : a = 0.08$, green line - $IC_{b3} : a = 0.2$, black line - "flat" IC in (5.7).

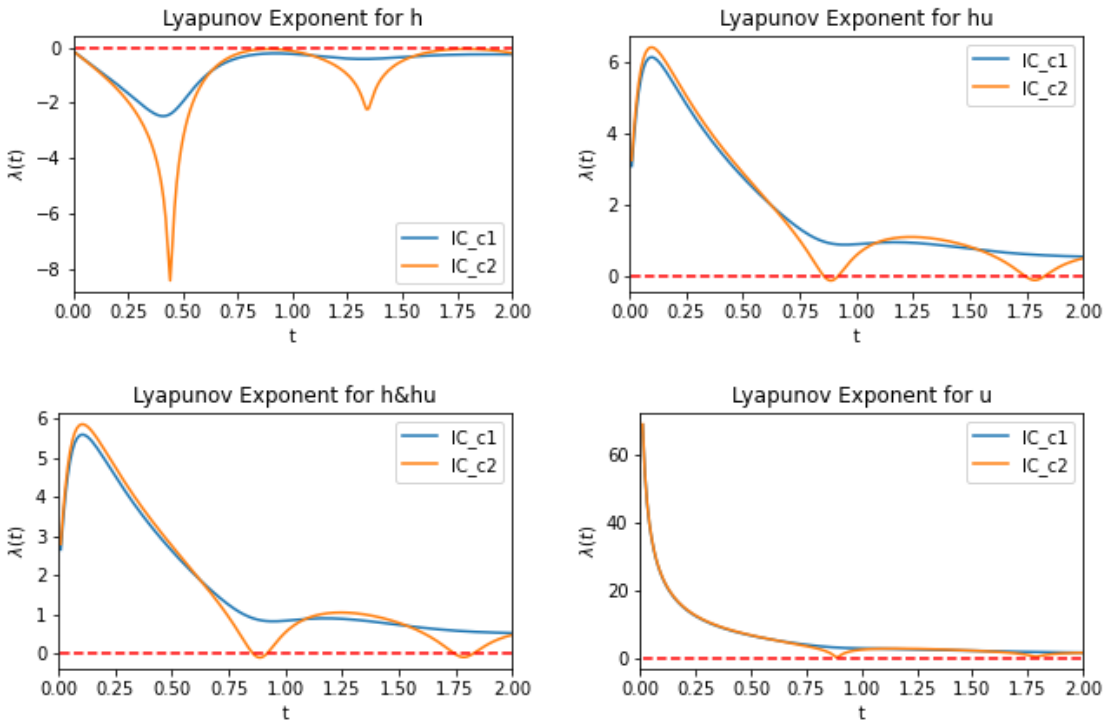


Figure 15: Numerical computation of Lyapunov exponents using (5.9) for initial conditions with perturbations in a different part of the domain; Blue line - $IC_{c1} : h(x,0)+z(x) = \{4+0.04\sin(8\pi(x-0.1L)/L)$ if $x \in [0.1L, 0.6L]$, 4 otherwise$\}$ and $u(x,0) = 2.5$, orange line - $IC_{c2} : h(x,0) + z(x) = 4 + 0.04\sin(8\pi x/L)$ and $u(x,0) = 2.5$.

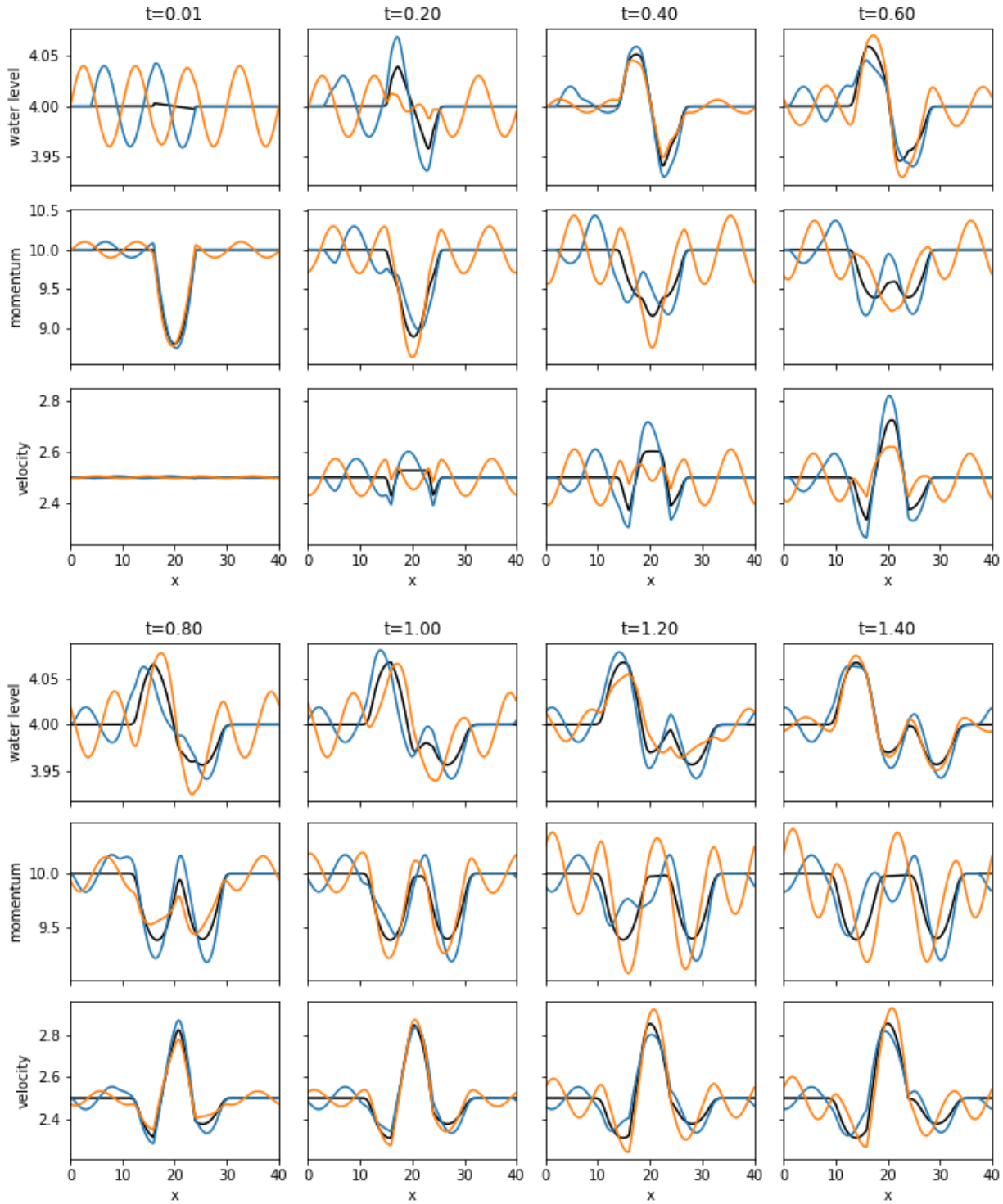


Figure 16: Snapshots of solutions corresponding to computation of Lyapunov exponent in Figure 15. initial conditions with perturbations in a different part of the domain; Blue line - IC_{c1} : $h(x,0)+z(x) = \{4+0.04\sin(8\pi(x-0.1L)/L)$ if $x \in [0.1L, 0.6L]$, 4 otherwise$\}$ and $u(x,0) = 2.5$, orange line - IC_{c2} : $h(x,0)+z(x) = 4+0.04\sin(8\pi x/L)$ and $u(x,0) = 2.5$.

6 ESN Performance and ESN Parameter Setting for the SWE

In this chapter, we analyze numerically various configurations and parameter settings of the ESN with respect to its prediction skill for the solution of the shallow-water equations. In particular, we analyze importance of the following features and parameters - (i) reservoir warm-up in section 6.2; (ii) including additional nonlinear transformation in reservoir construction in section 6.3; (iii) spectral radius of the connectivity matrix in section 6.4; (iv) reservoir size in section 6.4. This study allows us to select the optimal construction for the ESN which is used for predicting the solution of the SWE. The main goal is to understand the performance of the ESN and choose the "best" ESN architecture. We also present some predictions for particular trajectories. We use data from the first experiment in Table 8 to carry out these tests.

Recall, that the prediction setting for the Reservoir Computing is introduced in section 4.3 and L^2 relative error (4.11) of prediction is used as a measure of ESN performance. Time point T_0 is defined as starting time of prediction. We carry out two sets of experiments when $T_0 = 0$ and $T_0 = 10$. The choice $T_0 = 10$ is to allow nonlinear effects to develop so that the solution looks like a more realistic solution of the SWE. Only the initial condition $X_i(T_0)$ is used as an initial input data into the ESN, and future time-instances are predicted recursively as discussed in section 4.3. Therefore, prediction errors accumulate over time. To assess the overall performance of the ESN, all $N = 100$ trajectories are predicted and the L^2 relative prediction error is averaged over 100 trajectories.

6.1 Training and Testing Dataset

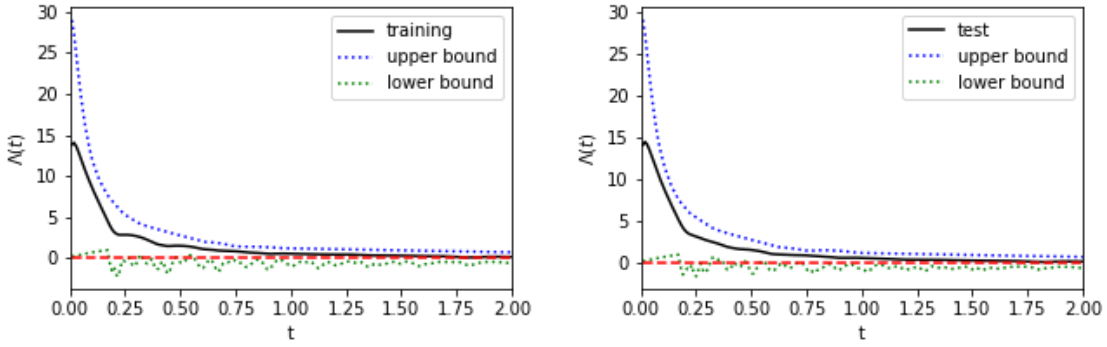
Two datasets are generated for the numerical experiments with Reservoir Computing. These two datasets are generated in a completely similar manner using initial conditions (5.10). We generate $N = 100$ trajectories $X_i(x,t)$ with $i = 1, \ldots, N$ for both training and testing data. Each trajectory includes both, the total water height and momentum, i.e., $X_i(x,t) \equiv \{h(x,t) + z(x), u(x,t)h(x,t)\}$ and discretized with $N_x = L/\Delta x$ spatial points, i.e., $X_i(t) = \{h_i(x_j,t) + z(x_j), u_i(x_j,t)h_i(x_j,t)\}$

with $j = 1, \ldots, N_x$. Both datasets are generated on time interval $[0, 20]$ with a discrete time step $\Delta = 0.01$. However, only training data on the time-interval $[0, 10]$ is used for training. The training dataset with $t \in [10, 20]$ is used for in-sample prediction, as discussed in section 4.3. Testing dataset is used for out-of-sample testing the predictive utility of the ESN in various regimes. In particular, we test ESN out-of-sample prediction utility on time-intervals $[T_0, T_0 + 10]$ with $T_0 = 0,\ 5,\ 10$. "True" solution $X_i(T_0)$ is used as an initial input into the reservoir and solution at a later time is obtained recursively as discussed in section 4.3. For $T_0 = 5,\ 10$ we also compare the out-of-sample prediction with and without the reservoir warm-up. We would like to emphasize that there are no duplicates in the training and test dataset.

Averaged Lyapunov Exponent for Training and Testing Dataset. The averaged Lyapunov exponent $\Lambda(t)$ over a dataset with N trajectories is defined as sample mean of individual Lyapunov exponents

$$\Lambda(t) = \frac{1}{N} \sum_{i=1}^{N} \lambda_i(t). \tag{6.1}$$

Similarly, the maximal Lyapunov exponent is $\Lambda_{max} = \max_t \Lambda(t)$ and the Lyapunov time is $LT = 1/\Lambda_{max}$. Figure 17 depicts averaged Lyapunov exponent $\Lambda(t)$ in (6.1) computed with respect to



(a) Lyapunov exponent for training dataset (b) Lyapunov exponent for testing dataset

Figure 17: Averaged Lyapunov exponent $\Lambda(t)$ in (6.1) and upper and lower bounds over all individual trajectories for the training data (left) and testing data (right).

water level $h+z$ and momentum hu over 100 samples for training testing datasets. We also present upper and lower bounds $\min_i \lambda_i(t)$ and $\max_i \lambda_i(t)$ over all trajectories in the two datasets, i.e., over

$i = 1, \ldots, N$. For both datasets, there is a short period ($t \leq 0.1$) when the averaged Lyapunov exponent is increasing followed by a long-term downward trend. The largest $\Lambda(t)$ is approximately 15 for both datasets and the upper bound for both datasets is approximately equal to 30. The behavior of averaged Lyapunov exponents is very similar for the training and testing datasets. Therefore, both datasets are comparable with respect to how fast (on average) trajectories diverge from the background state (5.7).

6.2 Importance of Reservoir Warm-up Before the First Prediction Step

The prediction task begins with the initial condition being used as an input into the ESN and the goal of the prediction task is to use outputs of the ESN recursively to forecast future values of the trajectory. A forecast can begin at $T_0 = 0$ (only one snapshot of trajectory is known) or at $T_0 > 0$ (some small initial part of the trajectory is known). If the forecast starts with initial conditions at $T_0 = 0$, the reservoir states are initialized with zero for out-of-sample prediction, i.e., $r(0) = \vec{0}$. If prediction task starts with initial conditions $T_0 > 0$, we can use reservoir states $r(T_0)$ obtained during training for the in-sample prediction. We also discuss how to obtain more practical values of the reservoir states for out-of-sample predictions starting with $T_0 > 0$.

The echo state property states that the effect of a previous state and a previous input on a future state should vanish gradually as time passes, and not persist or even get amplified. In other words, the reservoir states present the long-term memory from the historical inputs and this memory should decay in time.

We analyze the dependence of the ESN performance on the reservoir state for training and testing data in Figure 18. The testing dataset is generated out of 100 distinct samples are generated with small (1%-10%) perturbation of the background state using (5.10). The "in-sample" and "out-of-sample" in Figure 18 refer to predictions of trajectories from training and testing datasets, respectively. Figure 18 shows the ESN performance for the in-sample and out-of-sample predictions at times $T_0 = 0$ and $T_0 = 10$. We also present the in-sample prediction starting at time $T = 10$ when the memory of reservoir at first prediction time step T_0 is ignored (i.e., $r(T_0) = \vec{0}$). We can

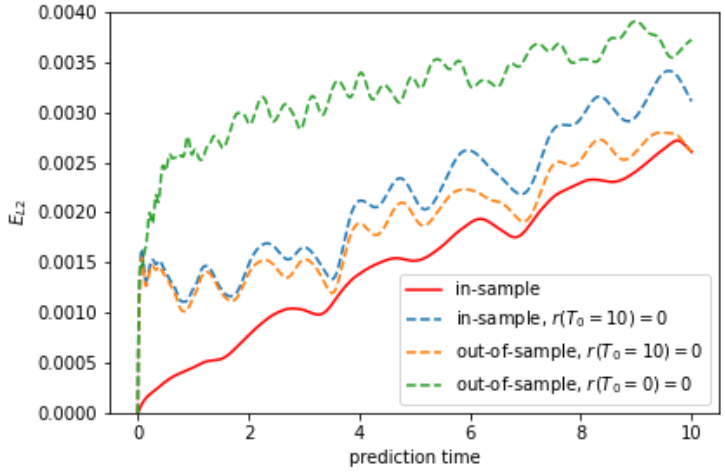


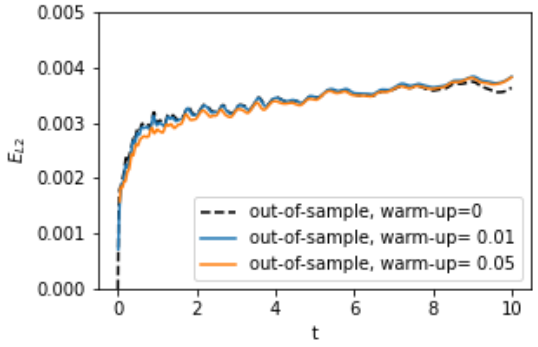
Figure 18: Performance of ESN with/without reservoir. In-sample prediction (red solid line) start with trained reservoir states, while the rest three dotted lines start with zero-value reservoir states. In-sample prediction and Out-of-sample prediction are (red, blue) and (orange, green) respectively. All results averaged over 100 samples (1% - 10% perturbation) with such ESN setting: $D = 5000$, $f = \tanh$, 100 training samples with 1% - 10% perturbation, training time $t = [0, 10]$.

see that the initial reservoir state makes a considerable difference during the first few prediction steps. In particular, prediction errors jump sharply at $T_0 + \Delta t$ when the reservoir state $r(T_0) = \vec{0}$ (green, blue, orange curves in Figure 18). For the in-sample prediction task, there is an obvious gap between trained $r(T_0)$ (red solid line) and zero-value reservoir state $r(T_0) = \vec{0}$ (blue dashed line). We also observe that there is no difference between the in-sample and out-of-sample prediction starting at time $T_0 = 10$ when $r(T_0) = \vec{0}$. Finally, we also can notice that out-of-sample prediction errors are larger for predictions starting with $T_0 = 0$ (green line) than $T_0 = 10$ (orange line). This probably implies that the initial conditions considered at $T_0 = 0$ do not represent a generic solution of the SWE equation. As expected, prediction errors accumulate over time. Overall, we can conclude that the memory of the reservoir is important, especially for the ongoing SWE.

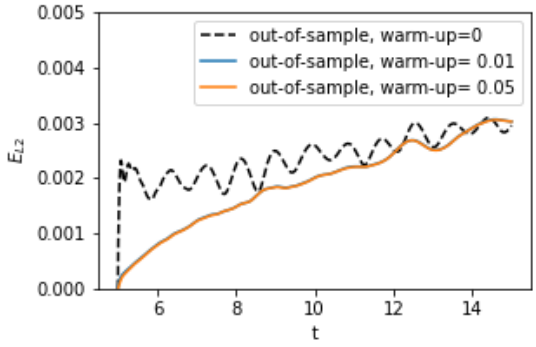
Predictions with Reservoir Warm-Up. Since the utility of predictions depends on the reservoir state, we can "warm-up" the reservoir with running several iterations of the reservoir update (4.2) with inputs prior to T_0. This allows us to obtain more realistic reservoir state $r(T_0)$. To warm the reservoir, we use (4.2) with the "true" values of the trajectory $\{X^{true}(t), t = T_0 - q\Delta t, \dots T_0 - \Delta t\}$. We denote $Q = q\Delta t$ as the *warm-up size*. Recall, that the testing dataset is

sampled with the time step $\Delta = 0.01$. In practice, if "true" snapshots of the solution are not known prior to T_0, then we would have to run the fully resolved shallow-water numerical solver to warm up the reservoir. However, such simulations would be very short but provide considerable benefits by improving the accuracy of predictions.

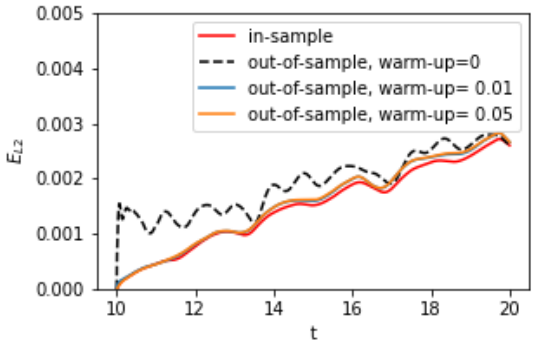
We compare no warm-up $Q = 1$, $Q = 0.01$ warm-up, and $Q = 0.05$ warm-up size for out-of-sample predictions in Figure 19. We consider three different prediction intervals. One prediction interval starts at time $T_0 = 0$ and two other intervals start at times $T_0 = 5$ and $T_0 = 10$. We would like to point out that sub-sampling time step is $\Delta = 0.01$, therefore, warm sizes of $Q = 0.01$ and $Q = 0.05$ correspond to just one and five iterations of (4.2), respectively. Figure 19 illustrates that reservoir warm-up improves utility of prediction starting at $T_0 = 5$ and $T_0 = 10$ (subplots (b) and (c)), but does not improve prediction starting with $T_0 = 0$. Predictions with sizes $Q = 0.01$ and $Q = 0.05$ almost overlap on all three plots in Figure 19. Therefore, we can conclude that warm-up size $Q = 0.01$ (or $q = 1$) is sufficient to improve the performance of the ESN and longer warp-up sizes do not bring further improvements. The sufficiency of a short warm-up size $Q = 0.01$ indicates that the reservoir has a very weak long-term dependence. Equivalently, the influence of each new input sample $X(t)$ is amplified quickly within $\tau = 0.01$. In addition, we note that it is more difficult to predict the dynamics of the SWE with an "artificial" initial conditions starting at $T_0 = 0$. For later times, a natural dynamics of the SWE develops and the reservoir is more efficient for predictions starting with $T_0 > 0$. This is not surprising, since the ESN has been trained primarily on "natural" SWE dynamics. Overall, we can conclude that it is beneficial to warm up the reservoir for a short warm-up time $Q = 0.01$.



(a) out-of-sample prediction, t=[0, 10]



(b) out-of-sample prediction, t=[5, 15]



(c) out-of-sample prediction, t=[10, 20]

Figure 19: Out-of-sample prediction with warming up reservoir. In (a), the black dotted line is prediction start at time $t = 0$ with zero-value reservoir and colored solid lines are prediction start at time t =warm-up with warm states by inputting ESN with $X^{true}(t)$, t =[0, warm-up]. In (b)(c), all prediction start at time $t = 5$ and $t = 10$ respectively. The blue and orange lines are overlapped in (b)(c). All results averaged over 100 samples (1% - 10% perturbation) with such ESN setting: $D = 5000$, $f = \tanh$, $\psi = \psi_1$, 100 training samples with 1% - 10% perturbation, training time $t = [0, 10]$.

6.3 Nonlinearity ψ and f in the Reservoir

In [65, 16], the authors demonstrated that a nonlinear transformation ψ between r and $\tilde{r}$ (the columns of the matrix $\tilde{r}$ should be chosen as nonlinear combinations of the columns of the reservoir state matrix r),

$$\tilde{r}(t) = \psi\left(r(t)\right),$$

is essential for skillful predictions, because this nonlinear transformation increases the expression power reservoir dynamics. Both [65] and [16] suggest a simple "odd squared" transformation, where every odd element is squared while even elements remain unchanged.

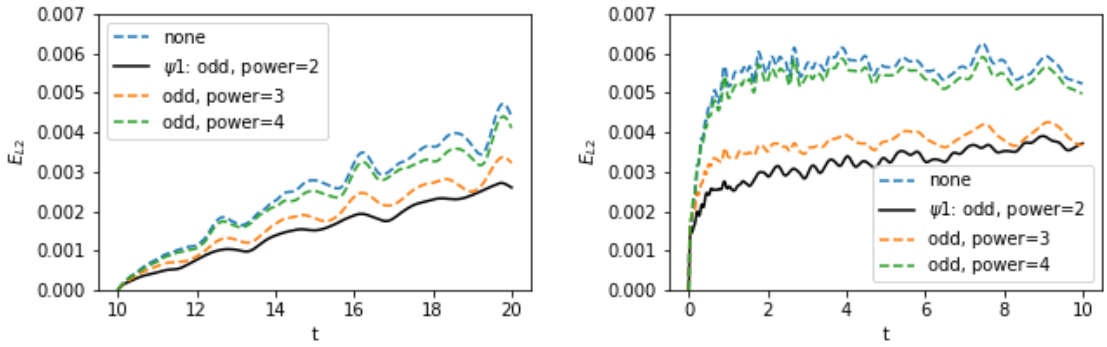
In order to check the degree of the polynomial transformation for the odd elements, we consider quadratic, cubic, and quartic of the odd elements, while keeping the original even elements in the reservoir, i.e., we consider the transformation

$$\psi_1(r) = \begin{cases} r_j^n & \text{if } j \text{ is odd,} \\ r_j & \text{if } j \text{ is even ,} \end{cases} \tag{6.2}$$

with $n = 1$, 2, and 3. Figure 20 shows that the nonlinear transformation in reservoir status improves the ESN prediction. Figure 20 also demonstrates that the quadratic function has the best performance in both in-sample and out-of-sample tests. We also checked fractional powers, such as power$= \dfrac{1}{2}$, as well as tanh function, but ESN with these nonlinear transformations have worse forecasting skills than basic reservoir without any nonlinear transformation. Thus, we can conclude that the quadratic transformation improves the skill of predictions in our case.

We perform additional test of quadratic transformations where we compare the following three transformations

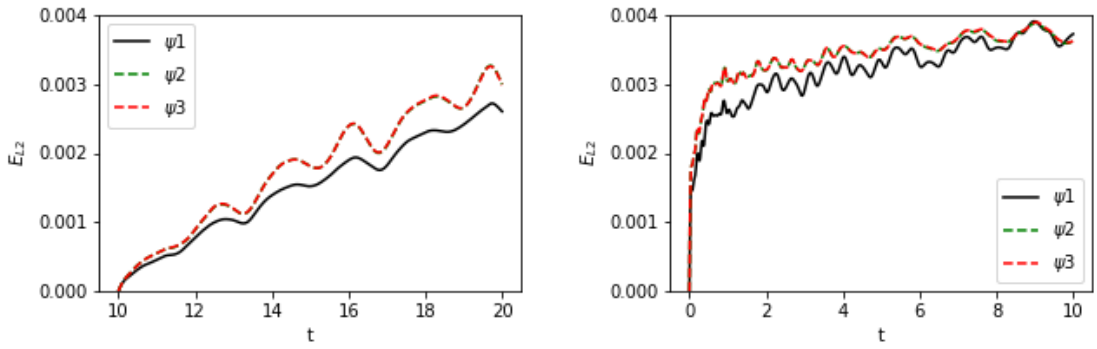
$$\psi_1(r) = \begin{cases} r_j^2, & j \text{ odd,} \\ r_j, & j \text{ even ;} \end{cases} \quad \psi_2(r) = \begin{cases} r_j, \times r_{t,j+1} & j \text{ odd,} \\ r_j, & j \text{ even;} \end{cases} \quad \psi_3(r) = \begin{cases} r_{j-1}, \times r_{t,j+1} & j \text{ odd,} \\ r_j, & j \text{ even.} \end{cases}$$



(a) in-sample prediction (b) out-of-sample prediction

Figure 20: ESN performance with various power of polynomial transformation algorithms (quadratic, cubic and quartic) in the odd elements (black, orange and green lines, respectively) and no transformation (blue dotted line). All results averaged over 100 samples (1% - 10% perturbation) with such ESN setting: $D = 5000$, $f = \tanh$, 100 training samples with 1% - 10% perturbation, training time $t = [0, 10]$.

ψ_1 is $n = 2$ case in Figure 20. Figure 21 depicts a comparison of ESN prediction with the three nonlinear transformations above. The nonlinearity ψ_1 has the best performance in both in-sample and out-of-sample tests, while ψ_2 and ψ_3 moderately degraded performance. Hence, we choose ψ_1 as a nonlinear transformation in reservoir status before readout.



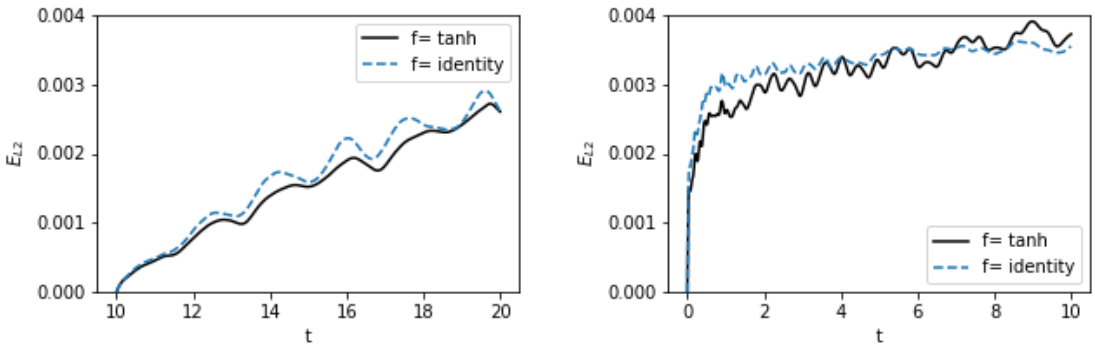
(a) in-sample prediction (b) out-of-sample prediction

Figure 21: ESN performance with three kinds of "odd squared" transformation in reservoirs. All results averaged over 100 samples (1% - 10% perturbation) with such ESN setting: $D = 5000$, $f = \tanh$, 100 training samples with 1% - 10% perturbation, training time $t = [0, 10]$.

After we fixed ψ_1 with $n = 2$ as the nonlinear transformation of reservoir states, we also examine the effect of changing the unit activation function $f = \tanh$, which is applied to units at

every update (4.9), in Figure 22. The nonlinear activation function (compounded throughout each time step) is supposed to provide a critical nonlinearity to most machine learning architecture, including ESN. However, we discovered that substituting the nonlinear $f = \tanh$ with the identity function $f(x) = x$ (i.e., removing the activation function $f = \tanh$) has a minor effect on overall in-sample performance. However, since the nonlinearity $f = \tanh$ slightly improves ESN prediction ability, we keep the nonlinear activation

$$r(t + \Delta t) = f\left(Ar(t) + W_{in}X(t)\right).$$



(a) in-sample prediction

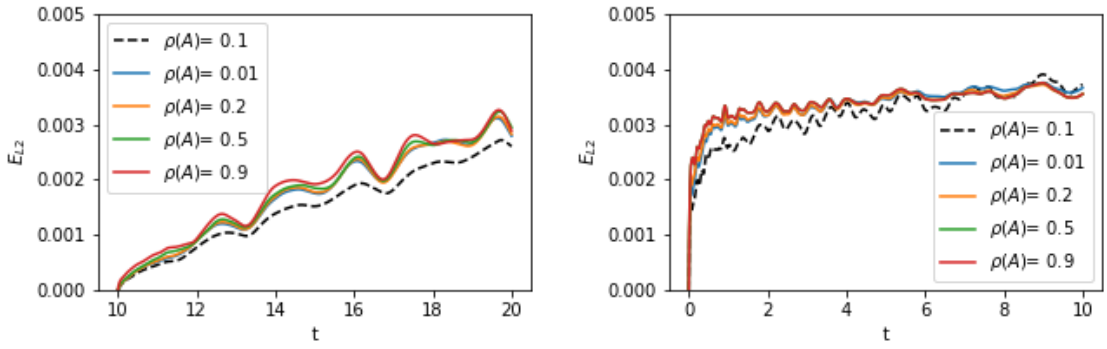
(b) out-of-sample prediction

Figure 22: ESN performance with/without unit activation function $f = \tanh$. Errors are averaged over 100 in-sample predictions (1%-10% perturbation) with ESN setting: $D = 5000$, $\psi = \psi_1$, 100 training samples with (1%-10%) perturbation, training time $t = [0, 10]$.

In summary, ψ_1 with $n = 2$ transformation of reservoir states and activation function $f = \tanh$ improve the ESN predictions. In particular, ψ_1 seems to be more important for improving the utility of predictions. Thus, the nonlinear transformation ψ_1 and the activation function $f = \tanh$ are used in the implementation of the ESN for the rest of the dissertation.

6.4 Connectivity Matrix A and Size of the Reservoir D

Spectral Radius of the Connectivity Matrix. The connectivity of the neurons in the reservoir is represented by the adjacency matrix A of size $D \times D$ whose values are drawn from a highly spare uniform distribution. Figure 23 depicts the performance of the ESN with connectivity matrices with different spectral radius. We can see that the performance difference is rather small, but ESN with $\rho(A) = 0.1$ has the best performance. The spectral radius condition $\rho(A) \leq 0.1$ corresponds to a low spectral radius, which intuitively implies that the dependence of the reservoir state $r(t)$ on past inputs $X_{t-\tau}$ decays exponentially quickly in lag time τ. ESN for SWE must have a low spectral radius condition, which is confirmed in Figure 23. $\beta_2 = \rho(A) = 0.01$ is too small for SWE and any $\rho(A) \geq 0.1$ downgrades the ESN performance. Therefore, we take $\rho(A) = 0.1$ in formula (4.1) when constructing the connectivity matrix A.



(a) in-sample prediction

(b) out-of-sample prediction

Figure 23: ESN performance with various spectral radius condition $\rho(A)$. All results averaged over 100 samples (1% - 10% perturbation) with ESN setting: $D = 5000$, $f = \tanh$, $\psi = \psi_1$, 100 training samples with 1% - 10% perturbation, training time $t = [0, 10]$.

Reservoir Size. In the sense of statistical learning theory, increasing the reservoir size D is the most direct way of increasing the model capacity. However, increasing D also leads to a higher computational load. The unifying theme throughout all applications is to use a fixed RNNs as a random nonlinear excitable medium, whose high-dimensional dynamical "echo" response to a driving input is used as a non-orthogonal signal basis to reconstruct the desired output by a linear combination, minimizing some error criteria.

Figure 24 shows that the scaled prediction error E slightly declines as D is increased from 800 to 5000, and then barely changes as D is doubled in size to 10000. Training the ESN with $D = 10000$ versus $D = 5000$ comes with a higher computational cost, while barely any improvement in accuracy is gained. Thus, concepts from inexact computing can be used to choose D such that precision is traded for large savings in computational resources, which can be then reinvested into more simulations, higher resolutions for critical processes, etc.

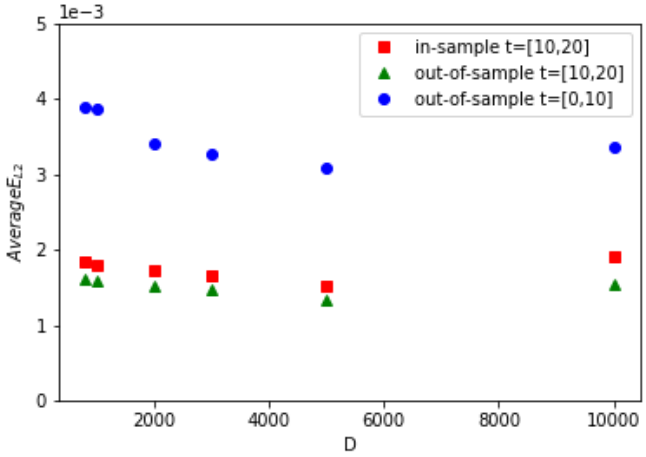


Figure 24: The scaling of average prediction error for in-sample at $t = [10, 20]$ (red squared), out-of-sample at $t = [10, 20]$ (green triangle) and out-of-sample at $t = [0, 10]$ (blue circle), as the reservoirs size D is changed from $D = 800$ to 10000. All results averaged over 100 samples (1% - 10% perturbation) with ESN setting: $f = \tanh$, $\psi = \psi_1$, 100 training samples with 1% - 10% perturbation, training time $t = [0, 10]$.

Conclusion. We can see that the performance of the ESN is not very sensitive to the changes in the spectral radius of the adjacency matrix A and reservoir size D. The reservoir adjacency matrix A with a low spectral radius condition $\rho(A) \approx 0.1$ works best for the SWE. The reservoir size D is fairly inconsequential once it exceeds a critical number of neurons, endowing the model with sufficient capacity.

The experiments of this chapter suggest ESN settings for predicting SWE trajectories. ESN has $D = 5000$ reservoir neurons, where the nonlinear transformation $\psi = \psi_1$ with $n = 2$ and unit activation function $f = \tanh$ are used to enhance expression power. The connectivity matrix of reservoir A is chosen sparsely and then re-scaled by (4.1) with $\beta_2 = 0.1$. The input-to-reservoir

matrix W_{in} is scaled with $\beta_1 = 0.01$. Only the weights W_{out} is trained with L^2 regularization strength $\lambda = 10^{-5}$. Numerical tests in this chapter show that such ESN is able to predict SWE trajectories with high accuracy (averaged L^2 relative error less than 0.4%) until 10 Model Time Unit (MTU), given the true value $X^{true}(T_0)$ of the initial condition for the SWE trajectory or the first two values, $X^{true}(T_0)$ and $X^{true}(T_0 + \Delta)$, to carry out the reservoir warm-up. ESN settings are preserved for the rest of the dissertation unless noted otherwise.

7 SWE Experiment Results

In this chapter, the ESN is used for out-of-sample prediction. We utilize exactly the same ESN to predict solutions with different initial conditions. We would like to emphasize that W_{out} is fixed after training and not retrained. The ESN settings are (see chapter 5)

$$D = 5000, \quad \rho(A) = 0.1, \quad f = \tanh, \quad \psi = \psi_1.$$

The dataset for training ESN consists of 100 trajectories generated with initial conditions as small perturbation (see equation (5.10)) of the background state. Perturbation is only added to the total water height. The perturbation level for the initial condition is in the range 1% - 5% with wave numbers $k = 1, \ldots, 7$. The initial velocity is not perturbed. Numerical solution is saved with $\Delta x = 0.1$, $\Delta = 0.1$ and $t = [0, 30]$. Weights W_{in} and A are generated at random as discussed previously in chapters 4 and 6. The weights W_{out} are saved after training and not modified afterwards.

Recall the prediction setting for the ESN introduced in section 4.3. Thus, the "true" initial condition $X^{true}(x, 0)$ in testing dataset is used as an initial input into the ESN, and consecutive predictions are generated without relying on the "true" trajectory. Similar to other experiments L^2 relative error (4.11) is used to measure the utility of prediction.

The maximal Lyapunov exponent (MLE) λ_i^{max} quantifies the rate of exponential growth (or compression) for small initial perturbations. Larger λ_i^{max} means that near-by trajectories separate faster, which implies that it should be more difficult to predict individual trajectories for larger λ_i^{max}. However, for the SWE equation considered here, the viscosity term $\nu(hu)_{xx}$ plays a stabilizing role and might "kill" high-frequency perturbations fast. Recall, that the averaged Lyapunov time for the training and testing datasets is approximately 0.06 (see section 6.1 and Figure 17), but there are also trajectories with very small Lyapunov exponents $\lambda = O(1)$.

7.1 Prediction of Trajectories with Different Initial Perturbations

Here we describe our numerical results for predicting individual trajectories with different initial conditions. Most initial conditions are generated as small perturbations of the background state (5.10) and, thus are consistent with the training dataset.

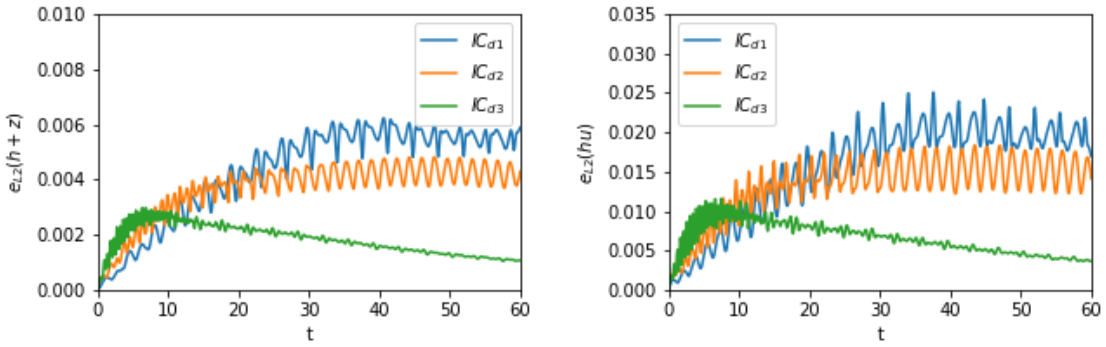
Initial Perturbation with Different Frequency

In this set of experimental results we fix perturbation at 5% and consider initial conditions for SWE with different frequencies of perturbation

$$\begin{aligned} IC_{d1}: \; & h(x,0)+z(x) = 4+0.2\sin(2\pi x/L), \quad u(x,0)=2.5, \\ IC_{d2}: \; & h(x,0)+z(x) = 4+0.2\sin(4\pi x/L), \quad u(x,0)=2.5, \\ IC_{d3}: \; & h(x,0)+z(x) = 4+0.2\sin(8\pi x/L), \quad u(x,0)=2.5. \end{aligned} \tag{7.1}$$

The corresponding maximal Lyapunov exponents (5.9) are computed with respect to the full state vector $\{hu(x_j,t), h(x_j,t)\}$. The maximal Lyapunov exponents are $\lambda_i^{max} = 1.38$ (IC_{d1}), 2.67 (IC_{d2}) and 5.64 (IC_{d3}). Prediction errors for initial conditions (7.1) are depicted in Figure 25. Figure 25 demonstrates that ESN has slightly better performance for predicting the initial conditions with bigger MLE. We observe that numerical errors for higher-frequency initial perturbations are larger on the time interval $[0, 10]$, but after time $t = 10$ the L^2 errors for the high-frequency initial perturbation IC_{d3} decays sharply. It is possible that viscosity has a stronger effect on the trajectory with a higher-frequency initial perturbation IC_{d3} and, thus, solution decays faster after $t = 10$.

Figures 26 and 27 depict predictions of the ESN for the initial conditions in (7.1). Overall, the ESN predicts solutions with initial conditions in (7.1) very well. Therefore, we can conclude that ESN has a high utility of predictions for initial conditions of the form (7.1).



(a) water level

(b) momentum

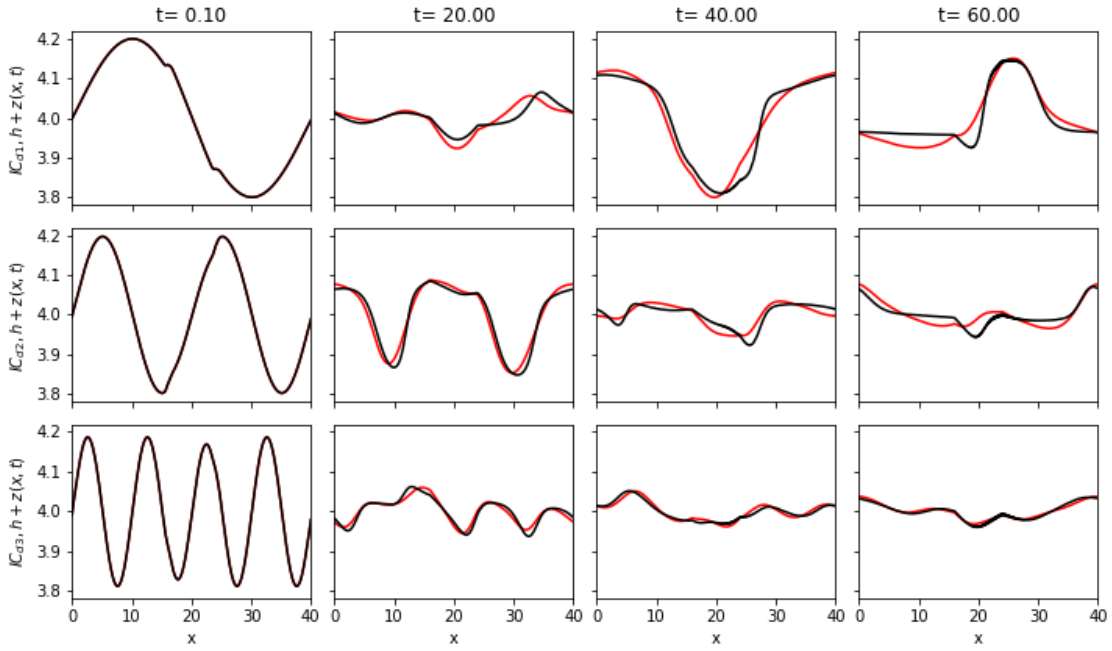
Figure 25: L^2 relative errors (4.11) in ESN predictions of the total water height $h(x,t)+z(t)$ (left) and momentum $h(x,t)u(x,t)$ (right) with initial conditions IC_{d1} (blue), IC_{d2} (orange) and IC_{d3} (green) in (7.1).

Initial Perturbation with Different Amplitude

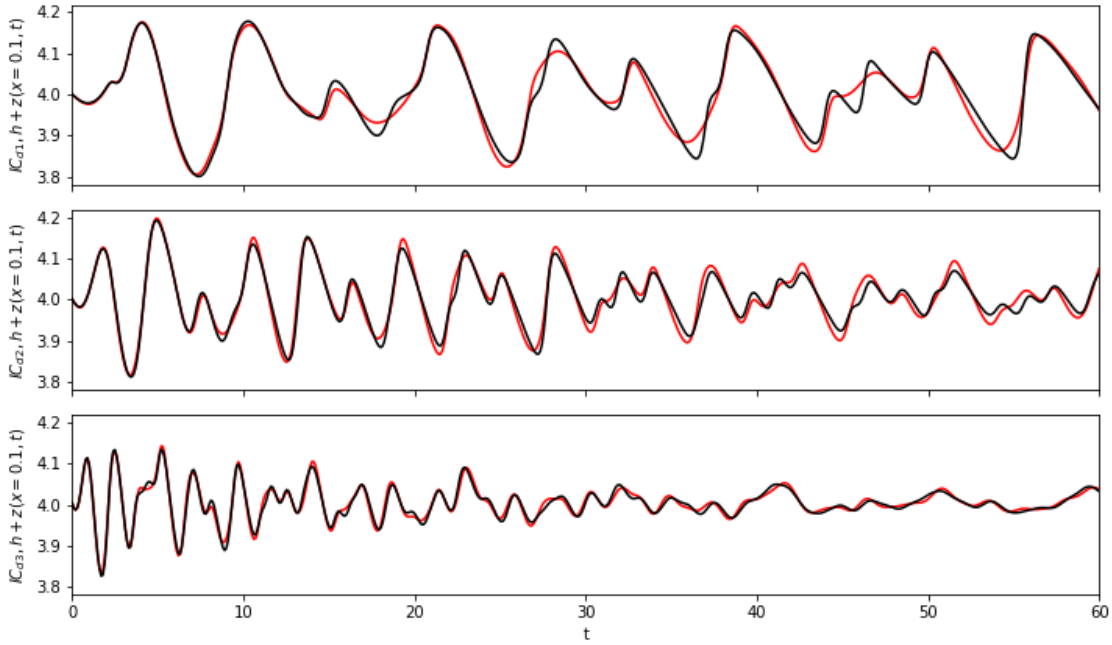
In this section, we investigate how the perturbation level affects the utility of predictions. In this experiment, the $p\%$ perturbation level is defined as the extreme value of perturbation compared with the background state. We change the perturbation level from 1% to 10% by considering the following initial conditions

$$\begin{aligned} IC_{e1}: \ & h(x,0)+z(x) = 4+0.04\sin(2\pi x/L), \quad u(x,0)=2.5, \\ IC_{e2}: \ & h(x,0)+z(x) = 4+0.2\sin(2\pi x/L), \quad u(x,0)=2.5, \\ IC_{e3}: \ & h(x,0)+z(x) = 4+0.4\sin(2\pi x/L), \quad u(x,0)=2.5. \end{aligned} \tag{7.2}$$

The maximal Lyapunov exponent is approximately 1.38 for all perturbation levels. As shown in Figure 28, it is clear that perturbation level affects the utility of the ESN prediction. Initially, for $t \in [0,20]$, prediction errors are roughly proportional to the initial perturbation level. However, for $t > 20$ prediction errors saturate. Also, prediction errors are relatively small reaching 1.5% for the initial perturbation IC_{e3} with 10%. Figure 29 and 30 present the snapshots and a trajectory of predicted water level and momentum.

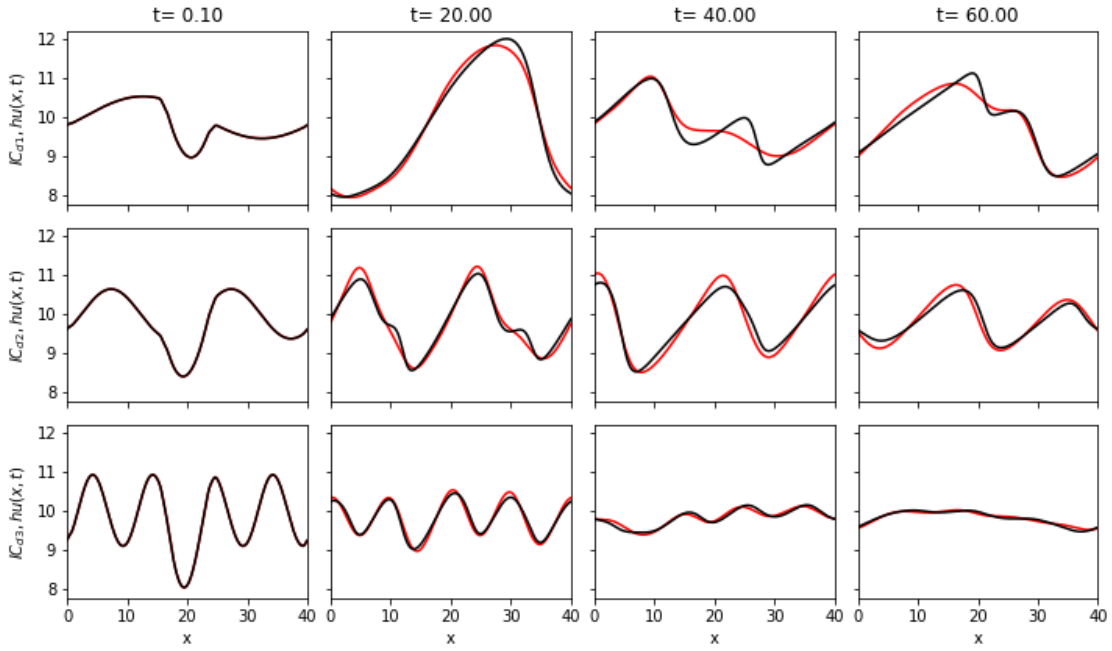


(a) Snapshots of the water level $h(x,t)+z(x)$ for $t = 0.1$, 20, 40, 60 with initial conditions IC_{d1} (top), IC_{d2} (middle) and IC_{d3} (bottom) in (7.1).

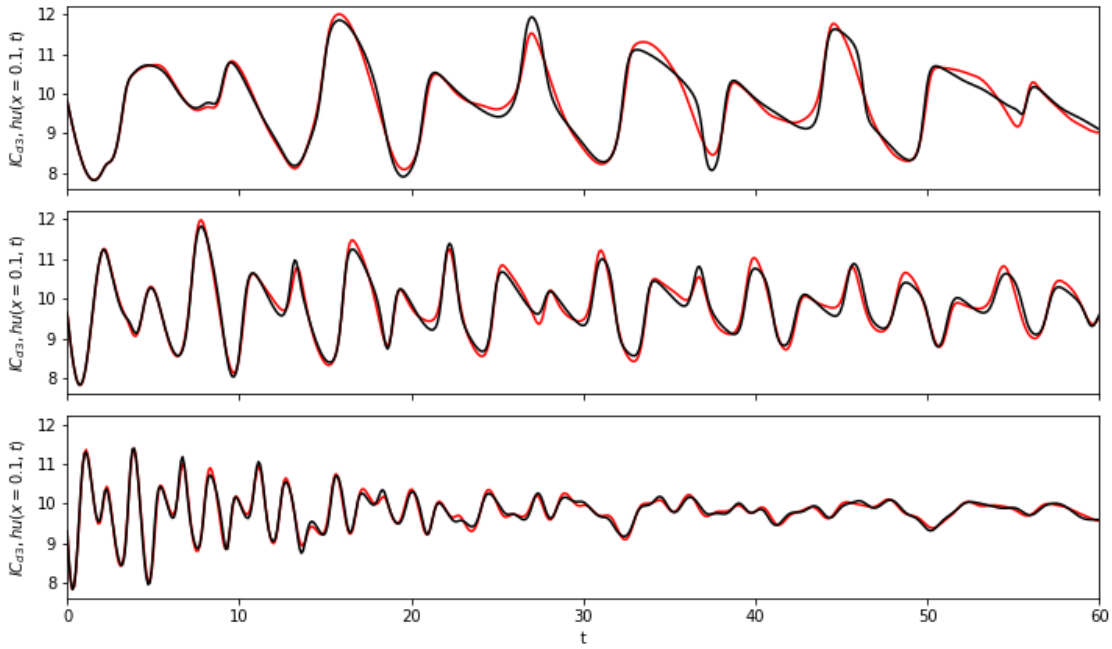


(b) Time-series of the water level $h(x,t)+z(x)$ for $x = 0.1$ with initial conditions IC_{d1} (top), IC_{d2} (middle) and IC_{d3} (bottom) in (7.1).

Figure 26: ESN prediction of the SWE solutions with initial conditions (7.1). Comparison of the water level in direct numerical simulations of the SWE (black line) and predictions of the ESN (red line).

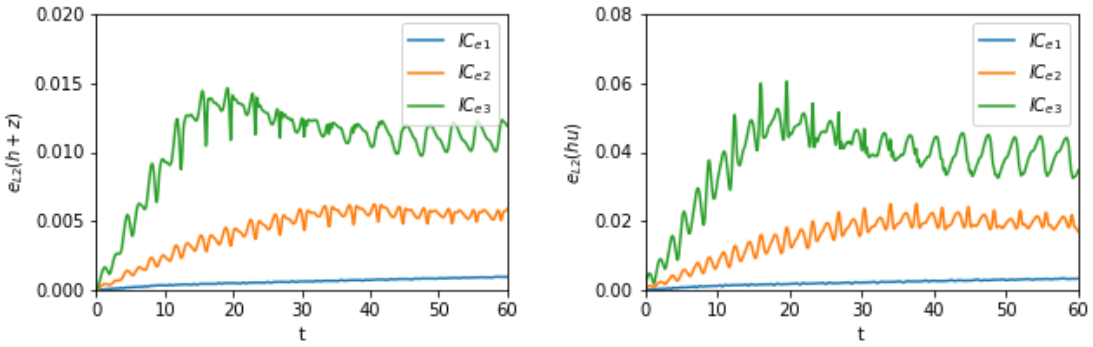


(a) Snapshots of the momentum $u(x,t)h(x,t)$ for $t = 0.1$, 20, 40, 60 with initial conditions IC_{d1} (top), IC_{d2} (middle) and IC_{d3} (bottom) in (7.1).



(b) Time-series of the momentum $u(x,t)h(x,t)$ for $x = 0.1$ with initial conditions IC_{d1} (top), IC_{d2} (middle) and IC_{d3} (bottom) in (7.1).

Figure 27: ESN prediction of the SWE solutions with initial conditions (7.1). Comparison of the momentum in direct numerical simulations of the SWE (black line) and predictions of the ESN (red line).



(a) water level

(b) momentum

Figure 28: L^2 relative errors (4.11) in ESN predictions of the total water height $h(x,t)+z(t)$ (left) and momentum $h(x,t)u(x,t)$ (right) with initial conditions IC_{e1} (blue), IC_{e2} (orange) and IC_{e3} (green) in (7.2).

Initial Perturbation with Multiple Frequencies

Here we study how initial conditions with multiple frequencies affect the ESN prediction of SWE solutions. We consider mixed-frequency perturbations

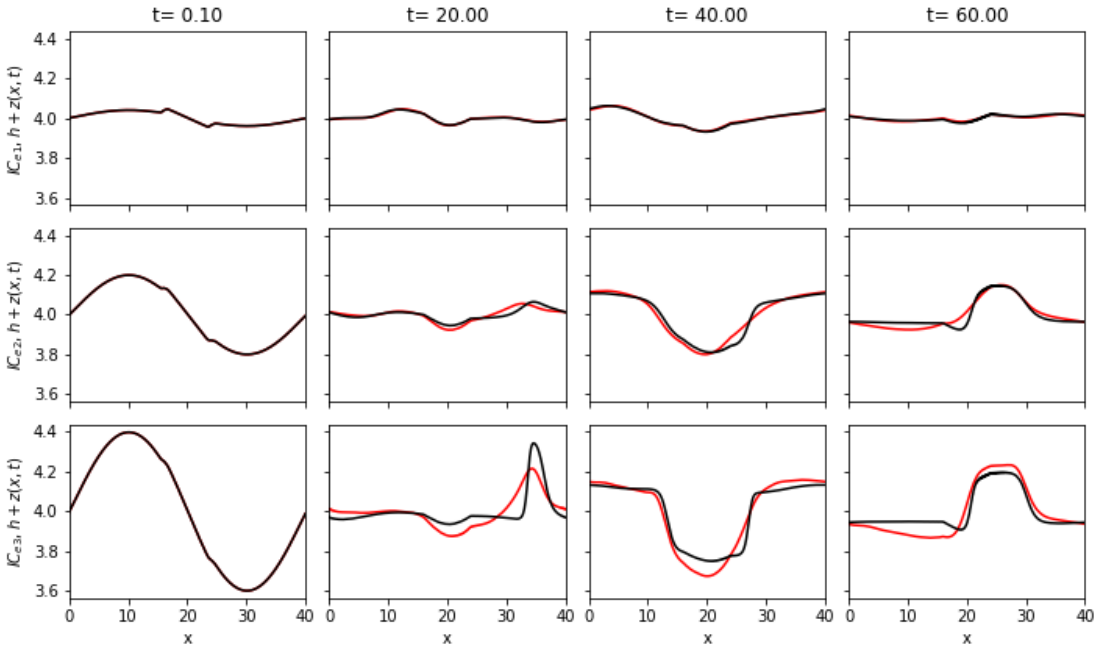
$$\begin{aligned} &u(x,0) = 2.5, \\ &IC_{f1}: \ h(x,0)+z(x) = 4 + 0.2\sin(2\pi x/L), \\ &IC_{f2}: \ h(x,0)+z(x) = 4 + 0.2\sin(2\pi x/L) - 0.08 \times sin(6\pi x/L), \\ &IC_{f3}: \ h(x,0)+z(x) = 4 + 0.4\sin(2\pi x/L) - 0.08 \times sin(6\pi x/L) + 0.04 \times sin(12\pi x/L). \end{aligned} \tag{7.3}$$

Here IC_{f2} is a 2-level (5% and 2%) mixed perturbation example and IC_{f3} is a 3-level (5%, 2% and 1%) mixed perturbation example.

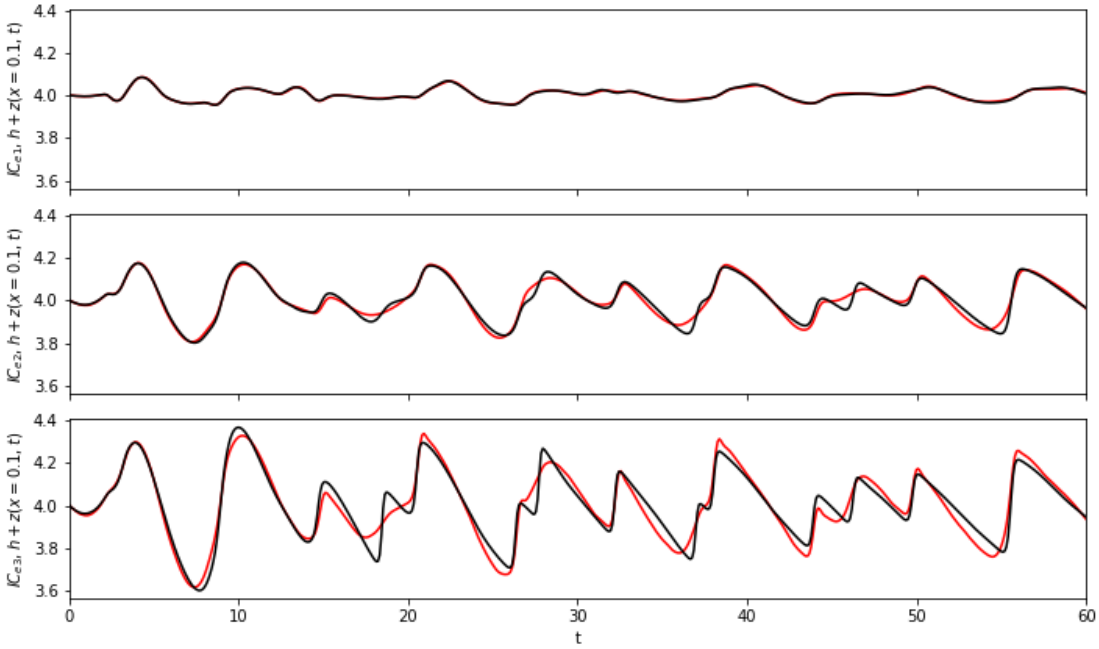
Figure 31 shows that adding higher-frequency perturbations to the initial conditions has a minor effect on the ESN performance.

7.1.1 Conclusion

In summary, ESN has the high utility of prediction for initial conditions close to the background state (5.7) with perturbation on the water height level. We would like to point out that, since

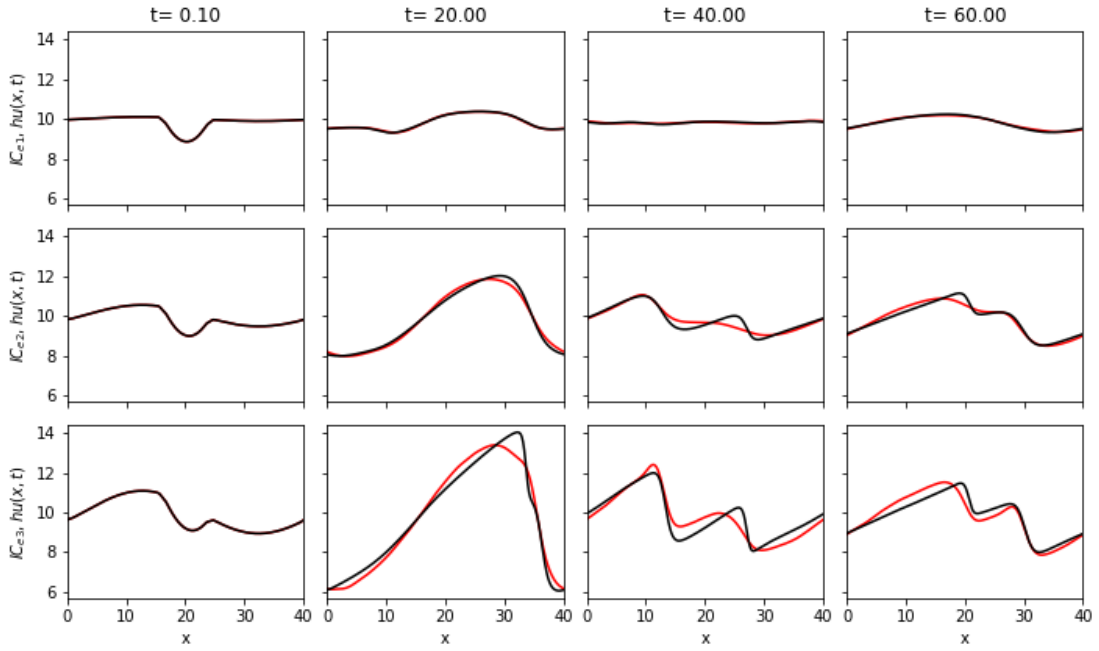


(a) Snapshots of the water level $h(x,t) + z(x)$ for $t = 0.1$, 20, 40, 60 with initial conditions IC_{e1} (top), IC_{e2} (middle) and IC_{e3} (bottom) in (7.2).

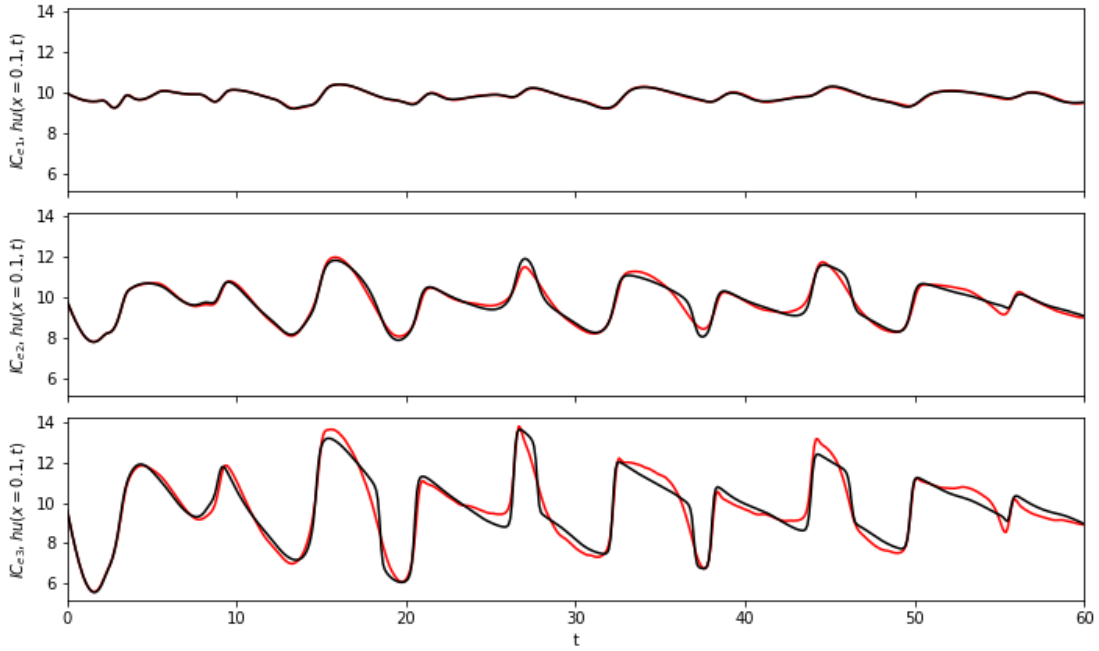


(b) Time-series of the water level $h(x,t) + z(x)$ for $x = 0.1$ with initial conditions IC_{e1} (top), IC_{e2} (middle) and IC_{e3} (bottom) in (7.2).

Figure 29: ESN prediction of the SWE solutions with initial conditions (7.2). Comparison of the water level in direct numerical simulations of the SWE (black line) and predictions of the ESN (red line).

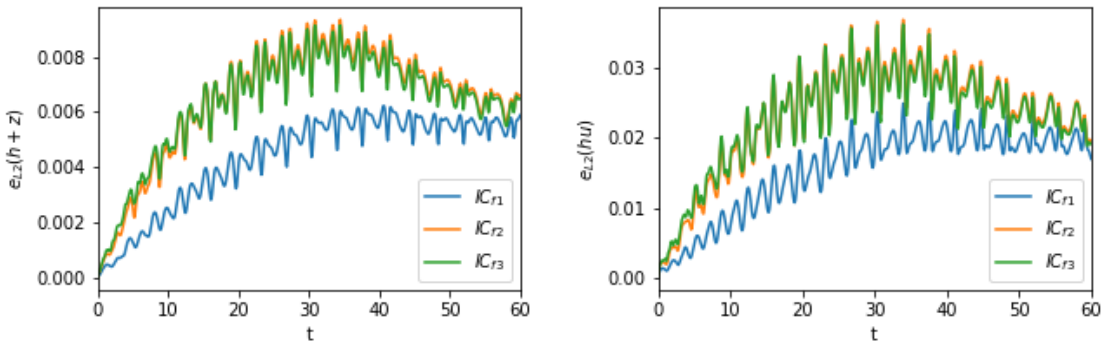


(a) Snapshots of the momentum $u(x,t)h(x,t)$ for $t = 0.1$, 20, 40, 60 with initial conditions IC_{e1} (top), IC_{e2} (middle) and IC_{e3} (bottom) in (7.2).



(b) Time-series of the momentum $u(x,t)h(x,t)$ for $x = 0.1$ with initial conditions IC_{e1} (top), IC_{e2} (middle) and IC_{e3} (bottom) in (7.2).

Figure 30: ESN prediction of the SWE solutions with initial conditions (7.2). Comparison of the momentum in direct numerical simulations of the SWE (black line) and predictions of the ESN (red line).

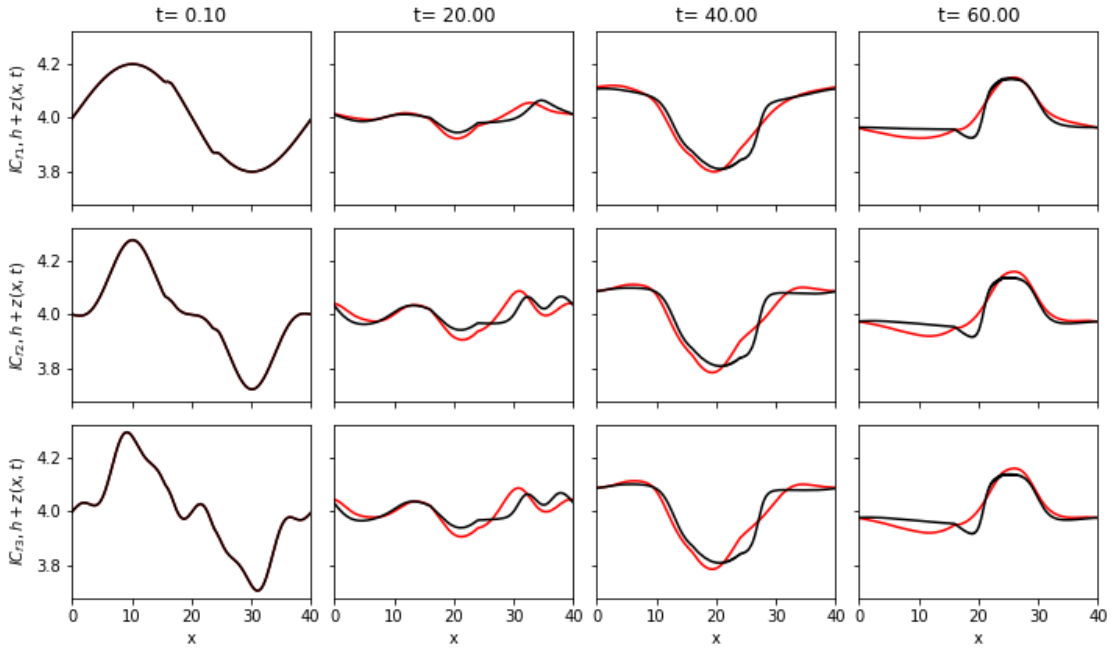


(a) water level

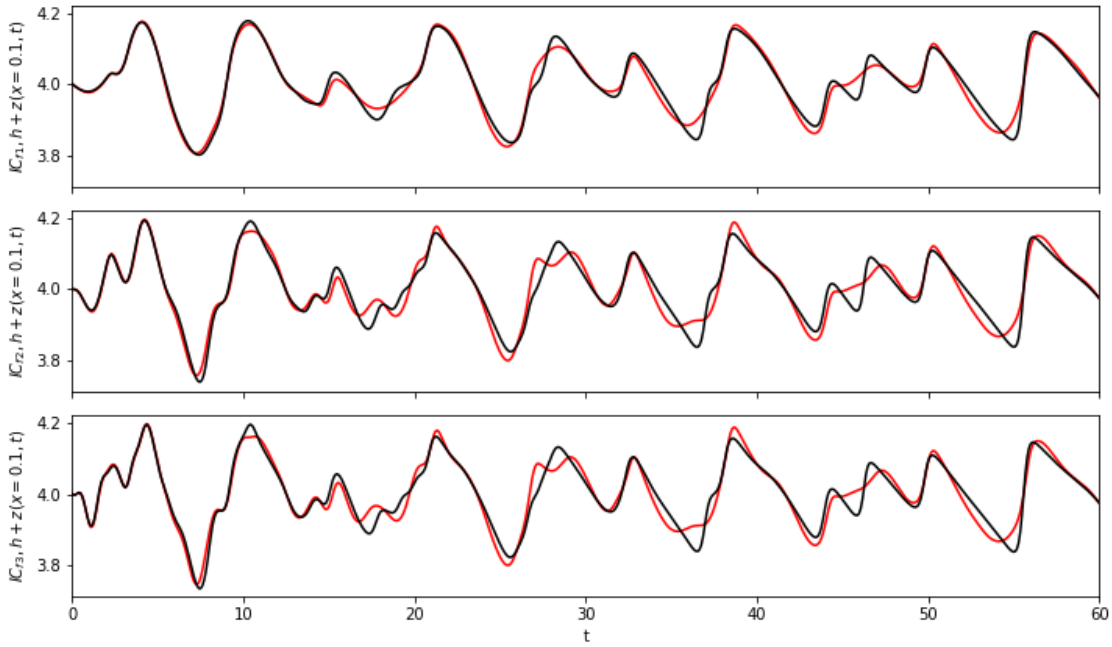
(b) momentum

Figure 31: L^2 relative errors (4.11) in ESN predictions of the total water height $h(x,t)+z(t)$ (left) and momentum $h(x,t)u(x,t)$ (right) with initial conditions IC_{f1} (blue), IC_{f2} (orange) and IC_{f3} (green) in (7.3).

perturbations are constructed as trigonometric functions, the averaged water level is preserved by perturbations and is identical for all initial conditions considered in this section. The water level at rest is always equal to H_0.

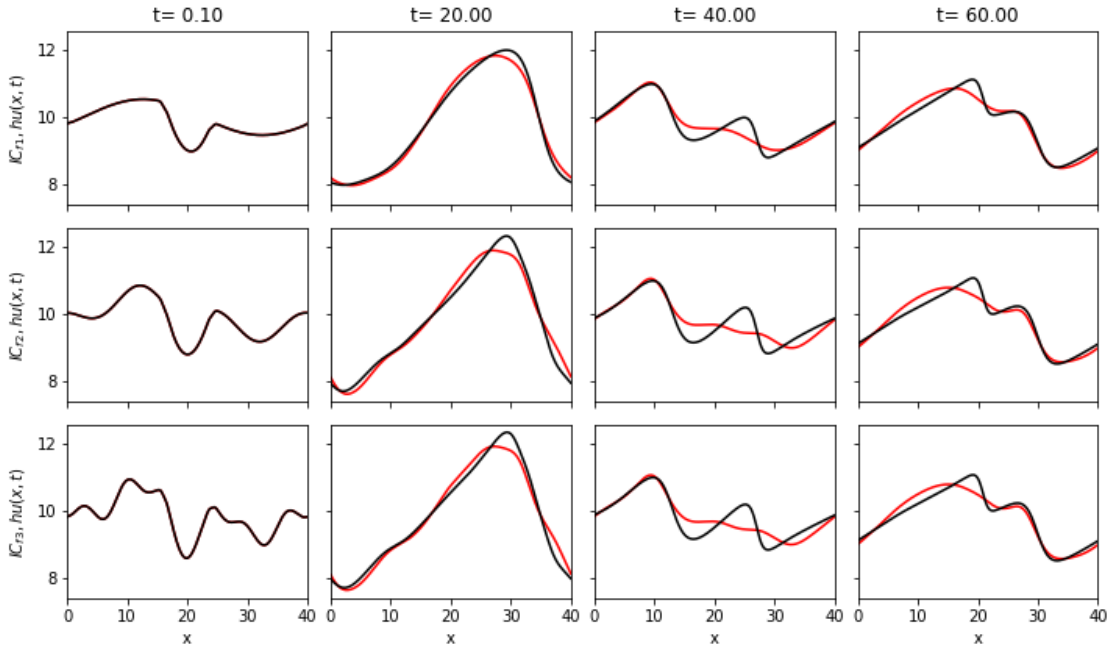


(a) Snapshots of the water level $h(x,t) + z(x)$ for $t = 0.1$, 20, 40, 60 with initial conditions IC_{f1} (top), IC_{f2} (middle) and IC_{f3} (bottom) in (7.3).

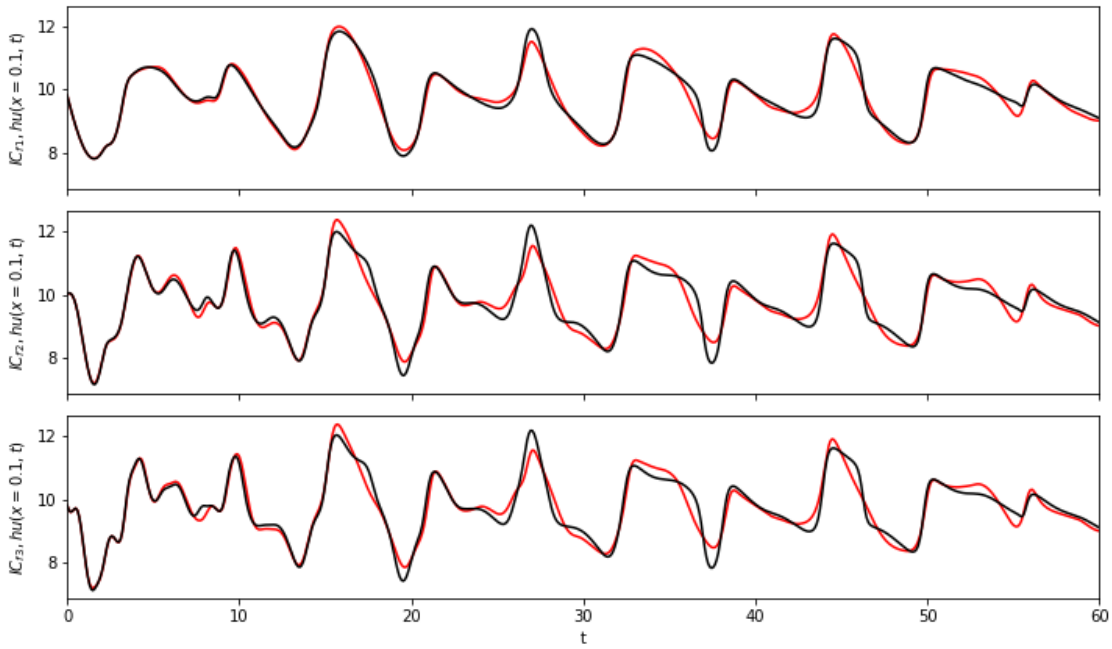


(b) Time-series of the water level $h(x,t) + z(x)$ for $x = 0.1$ with initial conditions IC_{f1} (top), IC_{f2} (middle) and IC_{f3} (bottom) in (7.3).

Figure 32: ESN prediction of the SWE solutions with initial conditions (7.3). Comparison of the water level in direct numerical simulations of the SWE (black line) and predictions of the ESN (red line).



(a) Snapshots of the momentum $u(x,t)h(x,t)$ for $t = 0.1$, 20, 40, 60 with initial conditions IC_{f1} (top), IC_{f2} (middle) and IC_{f3} (bottom) in (7.3).



(b) Time-series of the momentum $u(x,t)h(x,t)$ for $x = 0.1$ with initial conditions IC_{f1} (top), IC_{f2} (middle) and IC_{f3} (bottom) in (7.3).

Figure 33: ESN prediction of the SWE solutions with initial conditions (7.3). Comparison of the momentum in direct numerical simulations of the SWE (black line) and predictions of the ESN (red line).

7.2 Predictions with Perturbation on the Velocity and Different Initial Water Level and Velocity

In this section, we consider more severe testing datasets. In particular, we consider initial conditions with perturbations of the velocity and initial conditions which correspond to a different mean water level and mean velocity. We generate 5 datasets, but TEST 1 is identical to the examples from section 7.1 and is included here only for reference. Regimes for generating initial conditions for each dataset are summarized in Table 9. All datasets consist of 20 trajectories (solutions of the SWE).

Table 9: Parameters of testing datasets for ESN with different mean water height H_0, mean velocity U_0, and different perturbation levels around the base state (5.7).

	H_0	pert. for water level	U_0	pert. for velocity
TEST 1	4.0	1%-5%	2.5	0
TEST 2	4.0	1%-5%	2.5	1%-5%
TEST 3	4.0	1%-5%	2.5	5%-10%
TEST 4	4.0	1%-5%	2.375	1%-5%
TEST 5	3.8	1%-5%	2.5	1%-5%

TEST 1: This is the dataset similar to the testing dataset in section 7.1. Therefore, we have very high confidence the ESN has a high utility of prediction for this dataset. We include TEST1 for reference.

TEST 2: We also consider perturbations of the velocity, but the averaged velocity is preserved and is equal to $U_0 = 2.5$. In particular, we consider solutions of the SWE with the following initial conditions

$$h(x,0) + z(x) = \begin{cases} H_0 \pm aH_0 \sin(2k\pi(x - \phi_1 L)/L), & x \in [\phi_1 L, \phi_2 L], \\ H_0, \quad \text{otherwise}, \end{cases} \tag{7.4}$$

$$u(x,0) = U_0 \pm U_0 d \sin(2p\pi x/L + w) \quad x \in [0, L].$$

Here a and d are perturbation levels for the water level and velocity, respectively. Both a and d are

uniform random variables within 1%-5%. Frequency for the perturbation to the water level is in the range $k = \{1, \ldots, 10\}$. Frequency for the perturbation to the velocity is in the range $p = \{1, \ldots, 4\}$ and shift is $w \in [0, 2\pi]$. Both k and p are generated as uniform discrete random variables.

TEST 3: This dataset is similar to the dataset generated for TEST 2, except the perturbation level d of velocity is larger and ranges from 5% to 10% in (7.4).

TEST 4 and **TEST 5** consider trajectories with the following initial conditions

$$h(x,0) + z(x) = \begin{cases} H_0(1+s_H) \pm aH_0 \sin(2k\pi(x-\phi_1 L)/L), & x \in [\phi_1 L, \phi_2 L], \\ H_0, \quad \text{otherwise}, \end{cases} \tag{7.5}$$

$$u(x,0) = U_0(1+s_U) \pm U_0 d \sin(2p\pi x/L + w) \quad x \in [0, L].$$

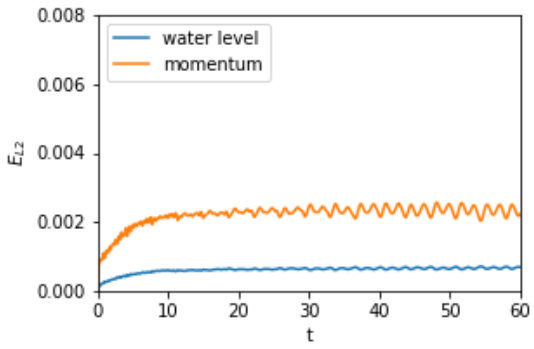
Here, $s_H, s_U \in [-1, 1]$ correspond to changing the mean of initial water level and velocity, respectively. Parameters s_H and s_U are fixed for all trajectories in each dataset. All other parameters are identical to the TEST 3 dataset and initial conditions in (7.4).

TEST 4: Velocity is shifted down by 5%, i.e., $s_U = -5\%$ and $U_0 = 2.5 \times (1 - 5\%) = 2.375$. Initial water level is generated with $s_H = 0$ as in (7.5).

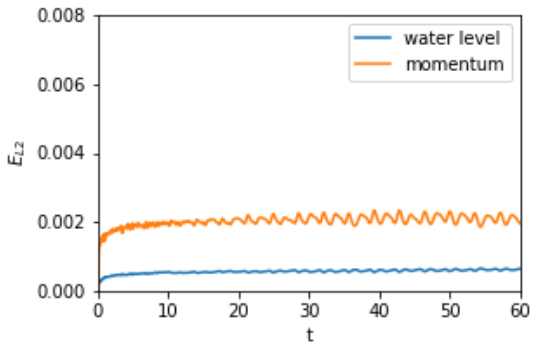
TEST 5: Water level is shifted down, i.e., $s_H = -5\%$ and $H_0 = 4.0 \times (1 - 5\%) = 3.8$. Initial velocity is generated with $s_U = 0$ as in (7.5).

Numerical Results

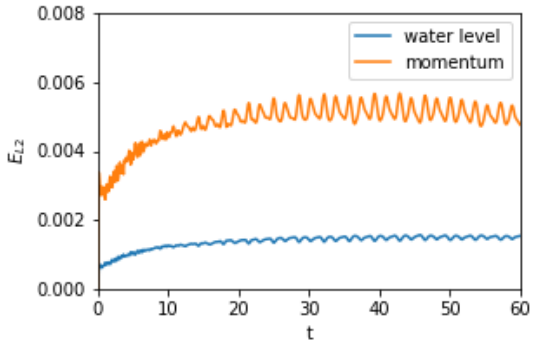
Figure 34 shows the averaged prediction error for the five testing datasets. The initial conditions for TEST 1 only have perturbation on initial water level and initial conditions for TEST 2 have both 1%-5% perturbation on the water level and velocity. ESN predicts trajectories from both datasets with high accuracy. We would like to remind, that the ESN was trained on a dataset with a "flat" initial condition for the velocity (5.10). Nevertheless, ESN is able to predict very well SWE trajectories with initial conditions with extra perturbation on velocity. In TEST 3 the velocity perturbation level is increased up to 10%. As shown in Figure 34 (c), ESN still has good



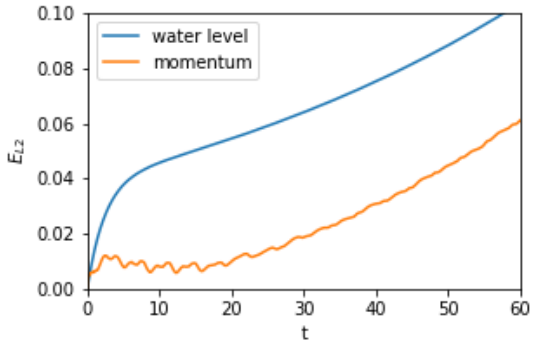
(a) TEST 1



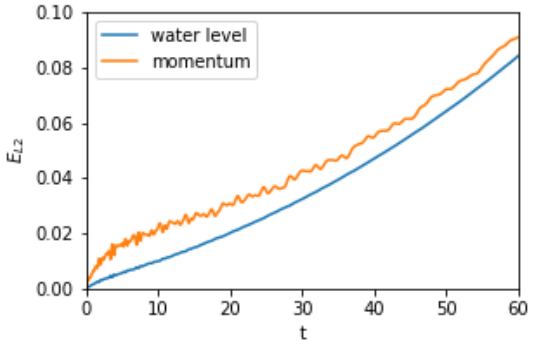
(b) TEST 2



(c) TEST 3



(d) TEST 4



(e) TEST 5

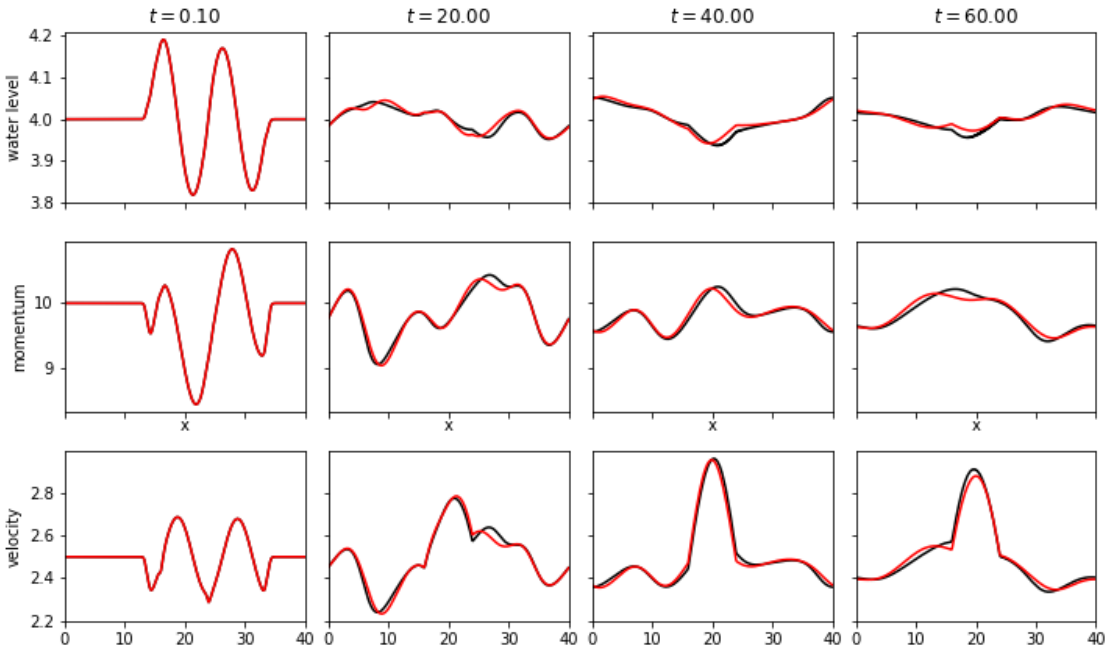
Figure 34: Averaged L^2 relative errors in ESN predictions of TEST 1-5. The total water height $h(x,t) + z(t)$ (blue) and momentum $h(x,t)u(x,t)$ (orange) for five test dataset. All results are predicted by same ESN: $D = 5000$, $\rho(A) = 0.1$, $f = tanh$, $\psi = \psi 1$, 100 training samples with 1% - 5% perturbation at t=[0,30].

prediction performance, even though prediction is slightly larger compared with TEST 1 and TEST 2 (depicted in Figure 34 (a) and (b)).

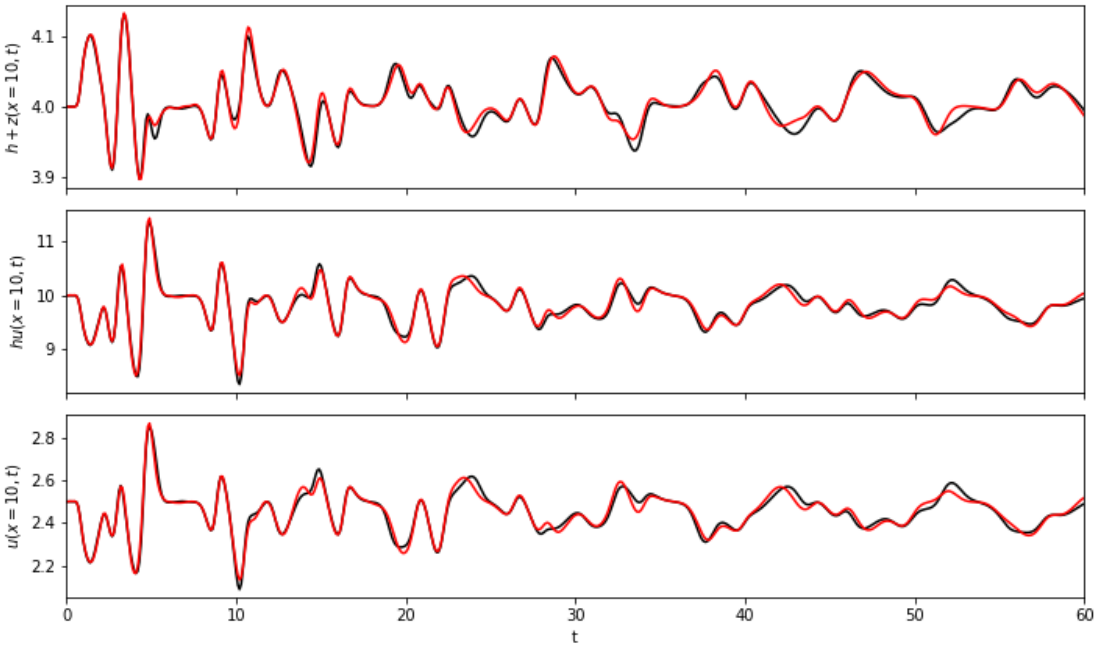
Figure 34 demonstrates that the utility of prediction deteriorates considerably when initial conditions are not consistent with the training dataset - either when the mean velocity U_0 or the mean water level H_0 are shifted, which corresponds to (d) and (e), respectively. Prediction errors increase significantly compared to TEST 2.

7.2.1 Conclusion

ESN is robust in predicting out-of-sample SWE trajectories which have the same mean water level and velocity. Perturbations can be quite large and can reach up to 10%, as shown in TEST 1, TEST 2, and TEST 3. However, ESN doesn't have the capability to predict SWE trajectories when the initial velocity or water level is shifted from the trained mean level. Figures 35 - 39 depict one particular trajectory from each testing dataset and corresponding prediction of the ESN. These figures confirm our conclusion. Moreover, ESN does not have the conservation of mass and conservation of momentum properties, as shown in Figures 38 and 39. In simulations depicted in Figures 38 and 39 the initial mean water level is $H_0 = 4$ and $H_0 = 3.8$, respectively. We can see that predictions of the ESN for the water height are considerably lower by time $t = 20$ for TEST 4 (Figure 38) and time $t = 40$ for TEST 5 (Figure 39).

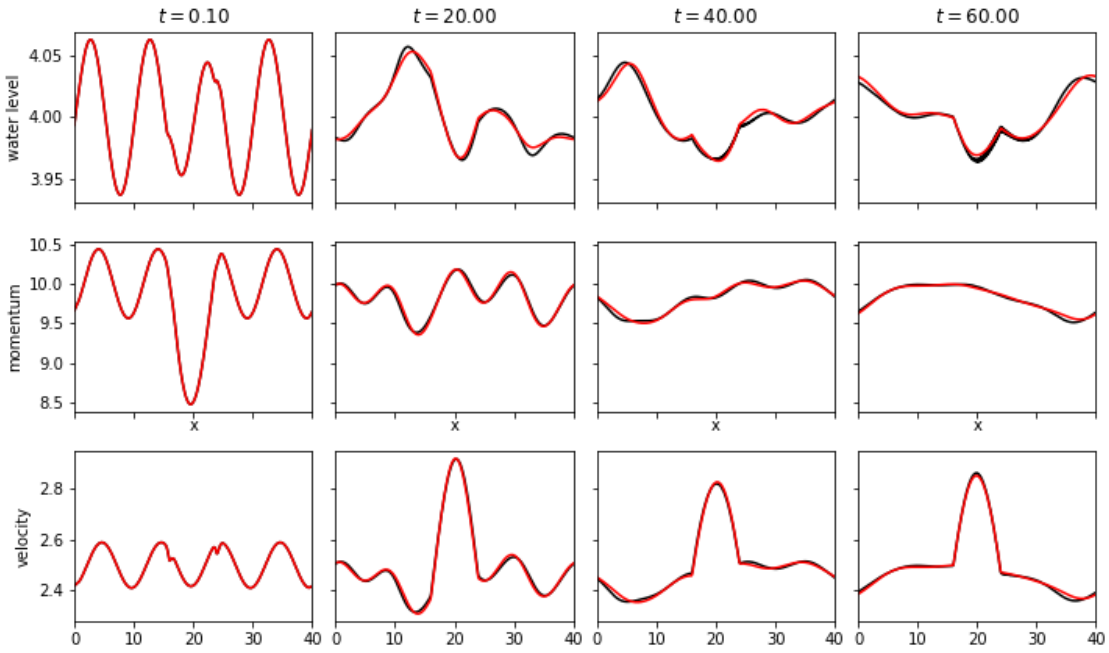


(a) Snapshots for $t = 0.1$, 20, 40, 60

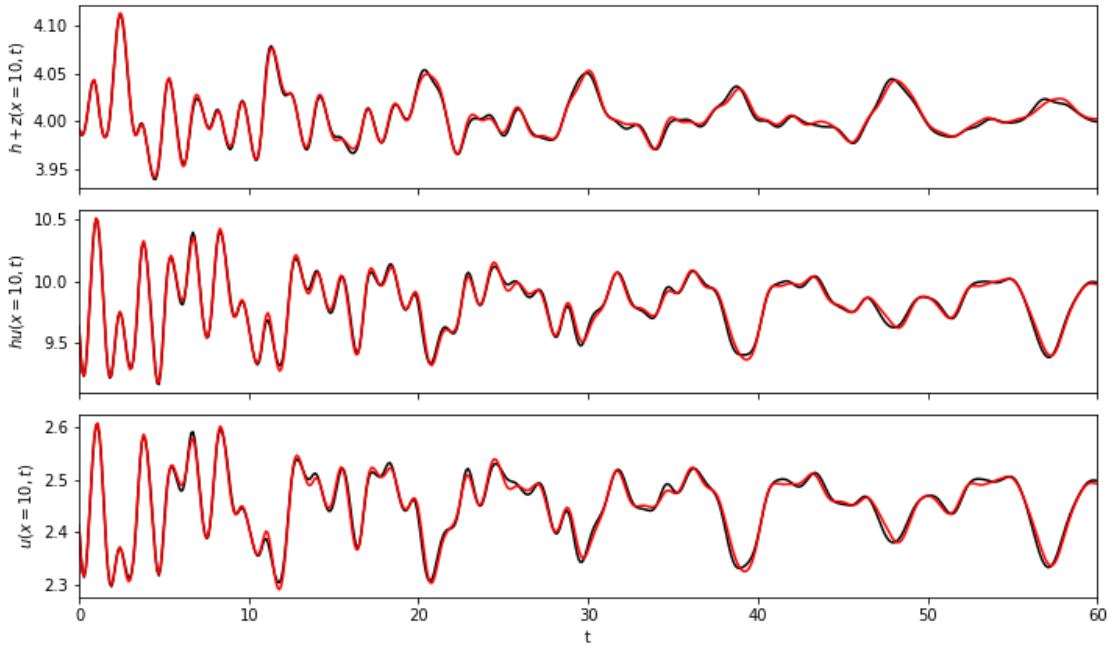


(b) Time-series for $x = 10$

Figure 35: Simulations for one sample in TEST 1. Comparison of direct numerical simulations of the SWE (black line) and predictions of the ESN (red line).

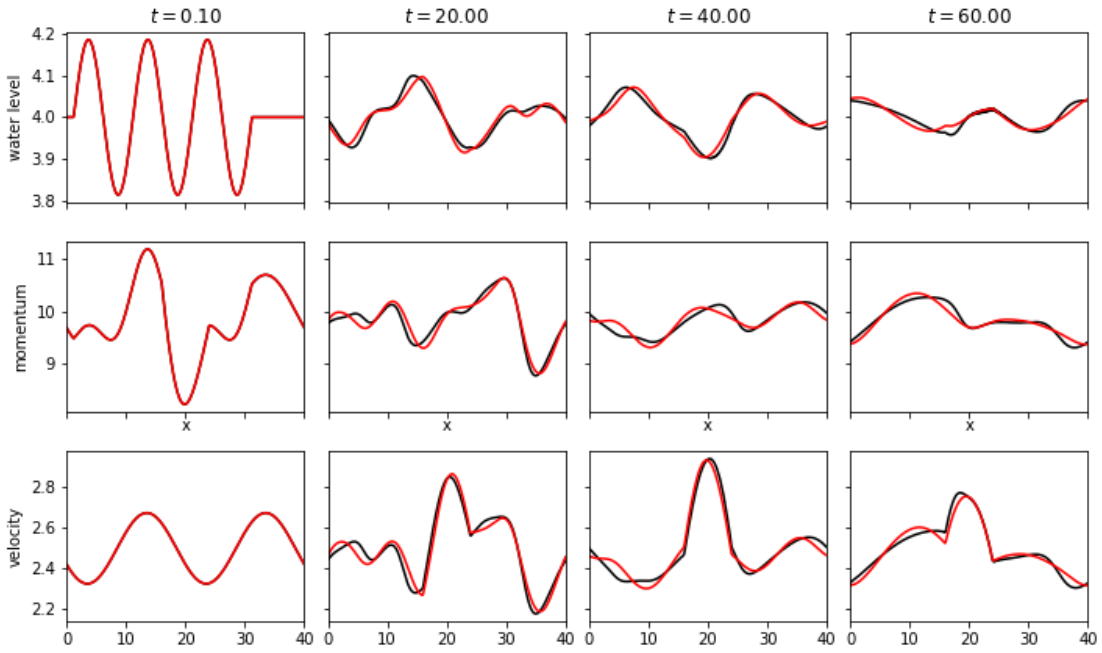


(a) Snapshots for $t = 0.1$, 20, 40, 60

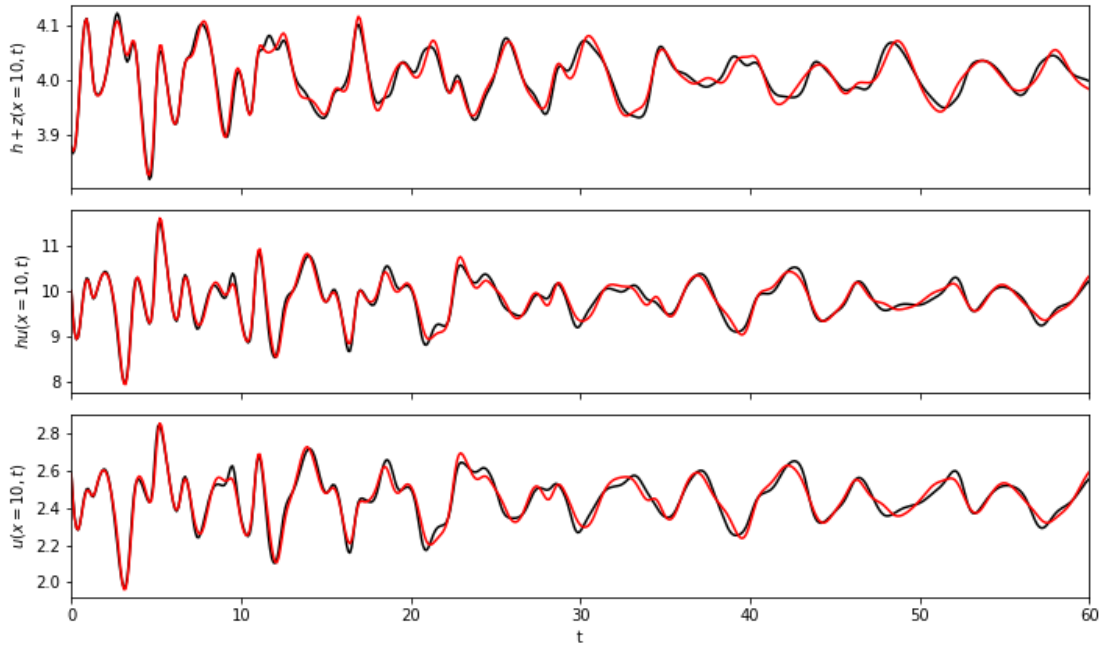


(b) Time-series for $x = 10$

Figure 36: Simulations for one sample in TEST 2. Comparison of direct numerical simulations of the SWE (black line) and predictions of the ESN (red line).

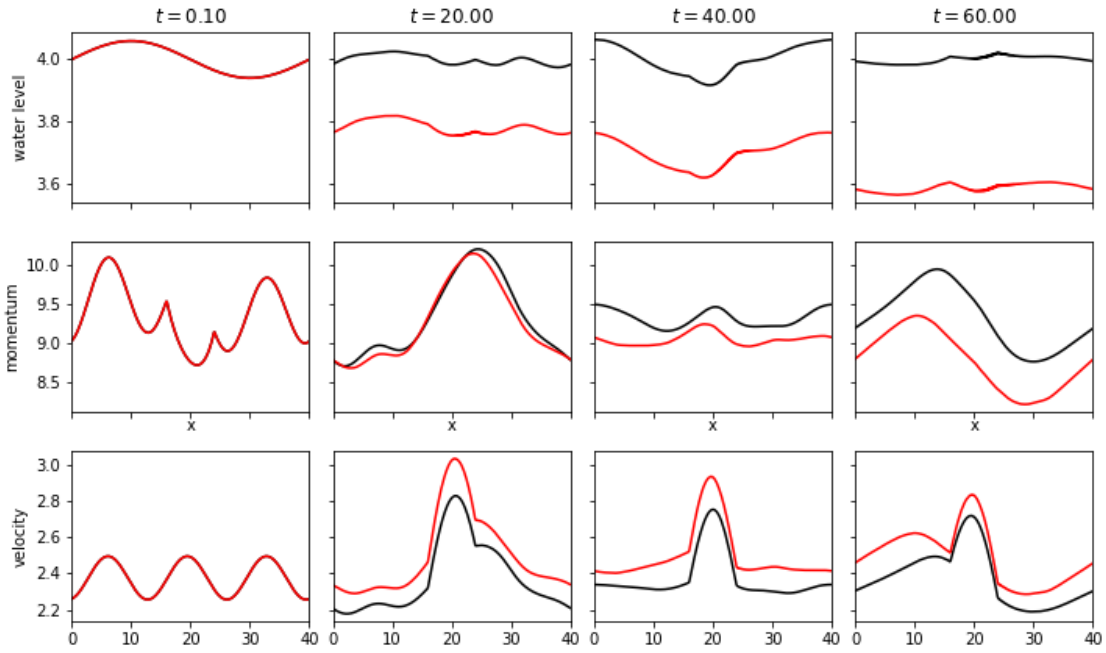


(a) Snapshots for $t = 0.1$, 20, 40, 60

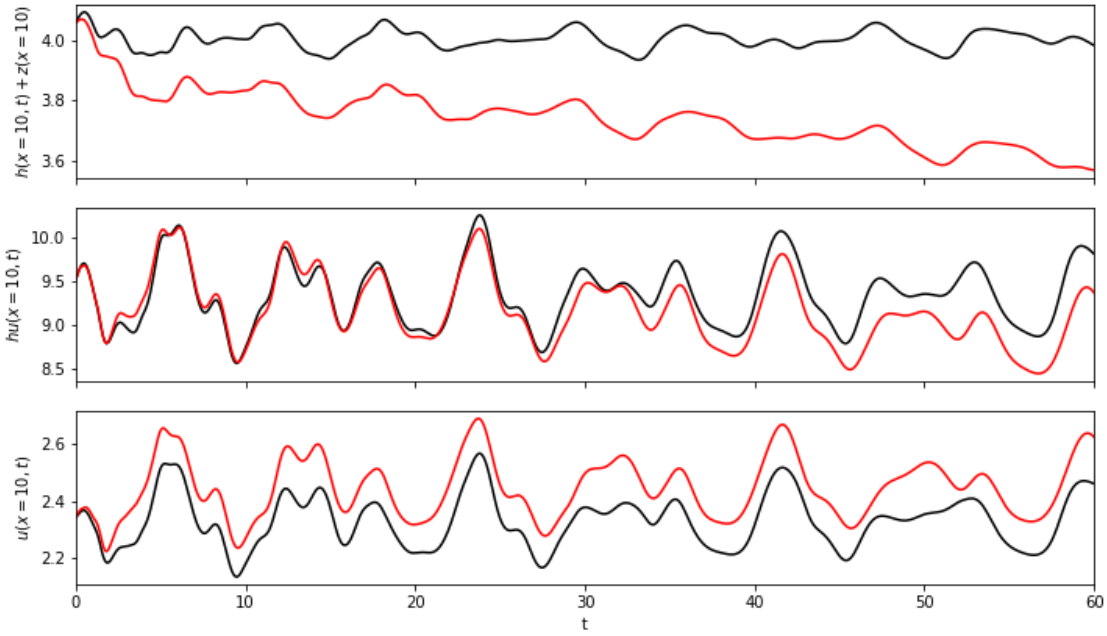


(b) Time-series for $x = 10$

Figure 37: Simulations for one sample in TEST 3. Comparison of direct numerical simulations of the SWE (black line) and predictions of the ESN (red line).

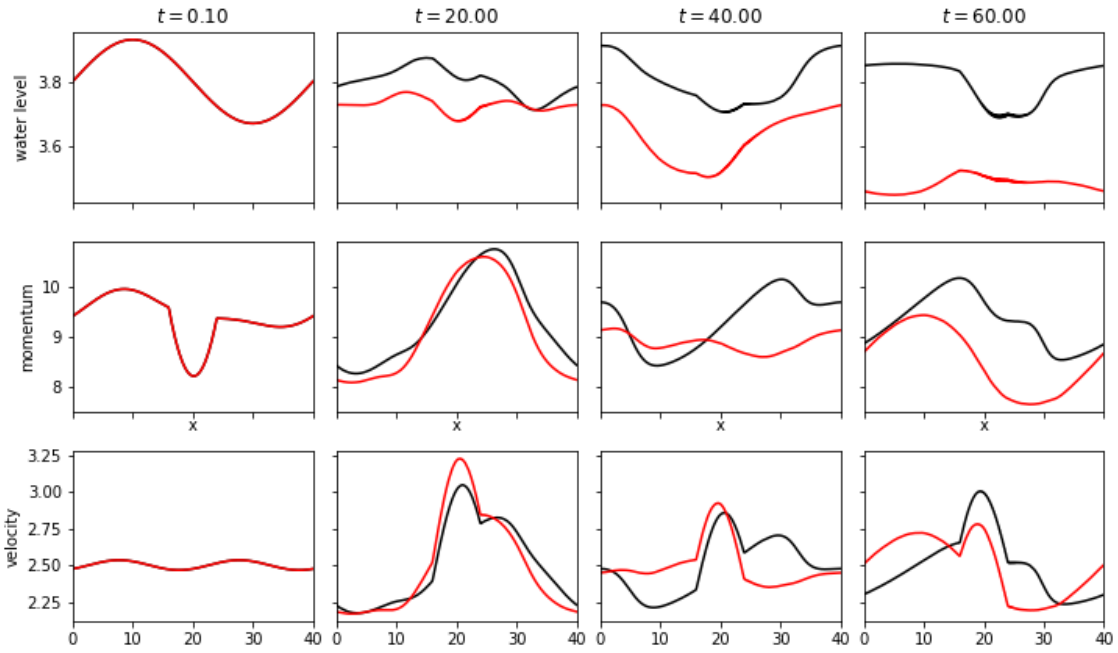


(a) Snapshots for $t = 0.1$, 20, 40, 60

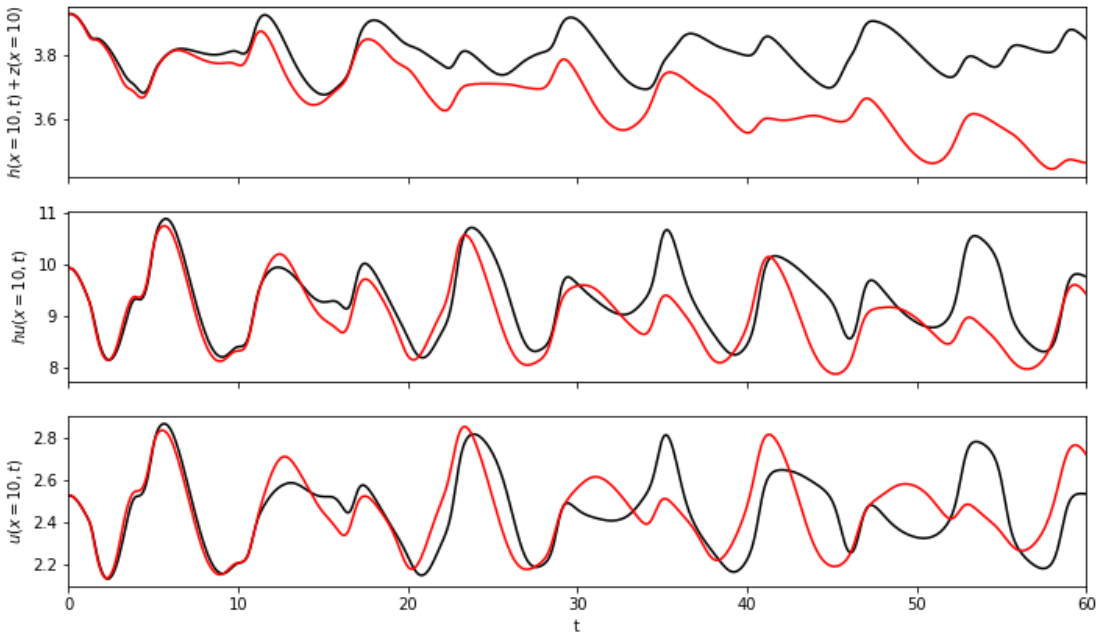


(b) Time-series for $x = 10$

Figure 38: Simulations for one sample in TEST 4 ($s_H = 0, s_U = -5\%$). Comparison of direct numerical simulations of the SWE (black line) and predictions of the ESN (red line).



(a) Snapshots for $t = 0.1$, 20, 40, 60



(b) Time-series for $x = 10$

Figure 39: Simulations for one sample in TEST 5 ($s_H = -5\%, s_U = 0$). Comparison of direct numerical simulations of the SWE (black line) and predictions of the ESN (red line).

7.3 Transfer Learning

Transfer learning is a machine learning method to tackle "the problem of retaining and applying the knowledge learned in one or more tasks to efficiently develop an effective hypodissertation for a new task" [76]. The idea of transfer learning is to utilize similarity between different tasks. Modeling fluid dynamics, in particular turbulence, is of great importance for a wide variety of engineering applications. We saw in the previous section that SWE has a high utility of prediction for initial conditions which are consistent with training data (same mean velocity U_0 and mean water level H_0), but does not predict trajectories that are inconsistent with the training data (different U_0 or H_0). One approach for predicting trajectories with different U_0 or H_0 would be to train a new ESN and generate a new W_{out}. However, such training needs a long time turbulence data with high resolution and high accuracy. In most applications, the generation of such turbulence data is computationally very costly. To overcome this problem, we employ transferring Reservoir Computing [39].

Conventional Method. Results in the previous section are generated using the Conventional Prediction Method. In this approach, the prediction of SWE trajectories by ESN is obtained without transfer learning. In this setup, there exists a well-trained ESN to predict SWE trajectories. The essential matrices W_{in}, A and W_{out} of ESN are not modified and are used for predicting SWE trajectories. We demonstrated that conventional method has high prediction accuracy for trajectories with initial conditions consistent with the training dataset, i.e., initial conditions that having the same initial mean water level and momentum as the training dataset.

Transfer Learning Formula for ESN [39]. We also demonstrated that the conventional method fails for trajectories with initial conditions not consistent with the training dataset. However, it might be possible to utilize transfer learning to "reuse" this ESN to predict similar SWE, where the initial mean water level and/or velocity are different from the original training dataset.

Assume that there are two training data $\mathcal{D}$ and $\mathcal{D}^*$, commonly referred to in the literature as *source* and *target* domain data, respectively. Also, assume that we already trained ESN on the previously existing *source* data $\mathcal{D}$, i.e., we obtained weights W_{out} by minimizing the mean squared

error with L^2 regularization, given *source* training data $\mathcal{D}$. Next, there is a new (usually relatively small size) *target* training data $\mathcal{D}^*$, that is similar to data $\mathcal{D}$, but also different in some sense.

We "reuse" the trained weights W_{out} and modify them with a correction δW, such that the new weights $W_{out} + \delta W$ are more suitable to represent the new data, $\mathcal{D}^*$. The correction is determined solely by using the original weights, W_{out}, and the *target* training data $\mathcal{D}^*$. Similar to the conventional method, the correct weights δW are determined by an optimization procedure

$$\delta W = \underset{\delta W}{\operatorname{argmin}} \, ||(W_{out} + \delta W)\tilde{r^*} - X^*||_2^2 + \alpha||\delta W||_2^2. \tag{7.6}$$

Here $\delta W \in \mathbb{R}^{D \times N}$ and $\tilde{r^*}(t)$ is the reservoir states of ESN driven by the input signal $X^*(t) \in \mathcal{D}^*$. The parameter $\alpha \geq 0$ is similar to L^2 regularization and defines as the **transfer rate**.

The corresponding analytical solution for the correction is

$$\delta W = (r^*(r^*)' + \alpha I)^{-1}(r^* X^* - r^*(r^*)' W_{out}), \tag{7.7}$$

where $(\cdot)'$ is the transpose and $(\cdot)^{-1}$ is the inverse operation.

When the transfer rate is zero, $\alpha = 0$, the above formula reduces to the conventional training method, which is just supervised learning by using the target training data $\mathcal{D}^*$ only. There is no knowledge transfer from source domain for $\alpha = 0$. On the other hand, in the limit of large transfer rate, $\alpha \to \infty$, we obtain $\delta W \to 0$, since there is a strong penalty on δW. Thus, for $\alpha \to \infty$ the above formula implies reusing weights W_{out} without any correction, i.e., $\delta W = 0$. Thus, there is no knowledge transfer from target domain for $\alpha \gg 1$.

In the paper [40], the authors take the Lorenz equations as an example to demonstrate that optimizing the transfer rate is essential, which leads to more accurate inference than the conventional method by an order of magnitude. The optimal transfer rate is in the range of $[0.01, 1]$ for the inference problem of the Lorenz chaos. The experience to find the optimal transfer rate is by parameter tuning. We observe the performance significantly improves with the transfer learning in a optimal range, by trying various $\alpha \in (0, \infty)$.

In conclusion, the transfer learning constitutes a one-parameter family of learning method which connect the conventional ESN ($\alpha = 0$) and the transfer method ($\alpha \gg 1$).

7.3.1 Velocity Shift

In this section, we apply the transfer learning method to predicting SWE trajectories with initial conditions in TEST 4. We would like to point out that the straightforward conventional method failed for this testing dataset (section 7.2). To apply transfer learning approach, we use the SWE model to generate one short simulation trajectory on time interval $[0, 2]$. This short trajectory is the target training data $\mathcal{D}^*$ for computing correction δW. We use transfer rate $\alpha = 0.01$. In fact, weight correction δW is updated by using one of the trajectories in TEST 4 testing dataset. W_{out} is previously trained (same as in sections 7.1 and 7.2). Then the corrected weights $W_{out} + \delta W$ is fixed and is used for predicting other trajectories in TEST 4 starting from initial conditions at $T_0 = 0$. To fairly compare results with transfer learning, we adopt this approach for all numerical results in this section, i.e., given a testing dataset with a particular mean velocity U_0, only one short SWE trajectory corresponding to this value of U_0 is used to compute weights correction δW. All other trajectories in this testing dataset are predicted using the ESN model with updated weights $W_{out} + \delta W$.

We generate four additional testing datasets which are summarized in Table 10. The mean water height for all datasets is $H_0 = 4$. Initial conditions are generated using (7.5) with $s_H = 0$.

TEST 4: mean velocity is $U_0 = 2.375$ corresponds to a shift of -5%.

TEST 6: mean velocity is $U_0 = 2.25$ corresponds to a shift of -10%.

TEST 7: mean velocity is $U_0 = 2.75$ corresponds to a shift of +10%.

TEST 8: mean velocity is $U_0 = 2$ corresponds to a shift of -20%.

TEST 9: mean velocity is $U_0 = 3$ corresponds to a shift of +20%.

Averaged errors of prediction are depicted in Figure 40. The performance of the transferring ESN is good in all five tests. Although correction δW is only computed using a small dataset

Table 10: Parameters of testing datasets for ESN with fixed mean water height $H_0 = 4$ and different mean velocity U_0.

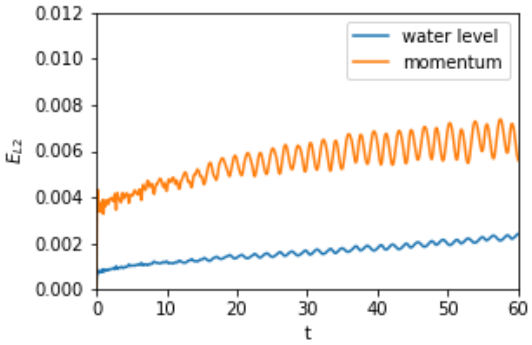
	size	H_0	pert. on water level	U_0	pert. on velocity
TEST 4	20	4.0	1%-5%	2.375	1%-5%
TEST 6	20	4.0	1%-5%	2.25	1%-5%
TEST 7	20	4.0	1%-5%	2.75	1%-5%
TEST 8	20	4.0	1%-5%	2	1%-5%
TEST 9	20	4.0	1%-5%	3	1%-5%

target $\mathcal{D}^*$ which consists of 20 snapshots of one solution of the SWE generated on time-interval $[0, 2]$ with sampling time step $\Delta = 0.1$. Overall, the transfer learning performs very well. When the available training data size is highly limited (i.e., $\mathcal{D}^*$ is a single short trajectory), the transfer learning is the most effective method for the inference task for $s_U \neq 0$. Figure 40 (b)(c) show that the prediction error of the inference task $s_U \leq 10\%$ are less than 2% for the water level. This is comparable (c.f. with Figure 34(a)(b)(c)) with the conventional method given a large training dataset $\mathcal{D}$ with $H_0 = 4$, $U_0 = 2.5$ (no mean velocity shift). When the similarity between the training domain and inference task becomes weaker, such as $|s_U| = 20\%$ in Figure 40(d)(e), the ESN with transfer learning performs worse and where prediction errors almost double. Nevertheless, even in cases of large mean velocity shift $|s_U| = 20\%$, the ESN performs much better compared with the conventional method trained on inconsistent data without the velocity shift or the conventional method trained on the small size of velocity shift (c.f. Figure 41).

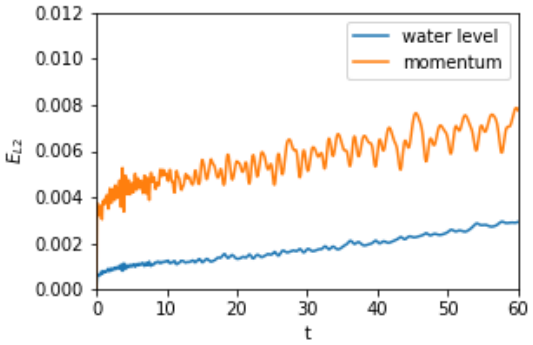
Figures 42 - 45 depict prediction result with transfer learning method for some randomly chosen trajectories from test datasets. In particular, Figure 42 is for TEST 4, where the inference task is with 5% smaller initial mean velocity. Figures 43 and 44 depict trajectories for the inference task is with 10% smaller (TEST 6) and larger (TEST 7) initial mean velocity, respectively. Figures 45 and 46 depict trajectories for the inference task is with 20% smaller (TEST 8) and larger (TEST 9) initial mean velocity, respectively. Since ESN without transfer learning performs very poorly and trajectories diverge for the ESN predictions with a positive shift of the mean velocity, Figures 45 and 46 only show the numerical simulation of SWE and ESN prediction with transfer learning. For SWE trajectories with increased initial mean velocity (Figures 45 and 46) the dynamics of the SWE

appear to be "more chaotic" and at time $t = 60$ the profile of the water level is highly non-trivial. We observe that the ESN prediction does not obey the mass conservation and predictions of the ESN for the water level are consistently lower than the DNS of the SWE.

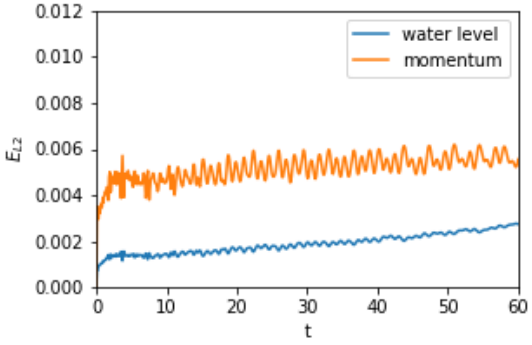
Summary. The transfer learning method is extremely effective for predicting trajectories with initial conditions generated with a "shifted" mean velocity. The accuracy is considerably improved using a small target dataset $\mathcal{D}^*$.



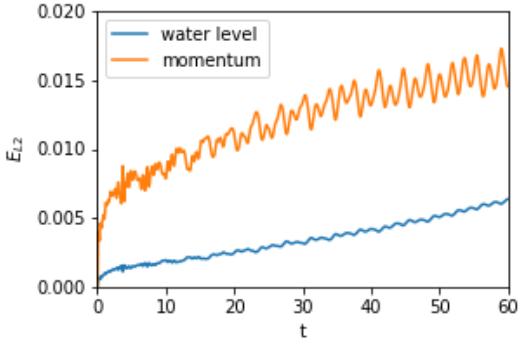
(a) Transfer Learning, TEST 4 ($s_U = -5\%$)



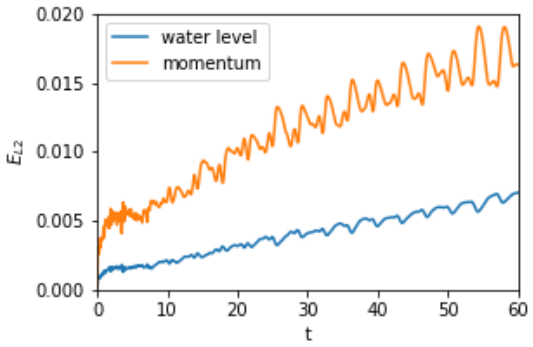
(b) Transfer Learning, TEST 6 ($s_U = -10\%$)



(c) Transfer Learning, TEST 7 ($s_U = 10\%$)

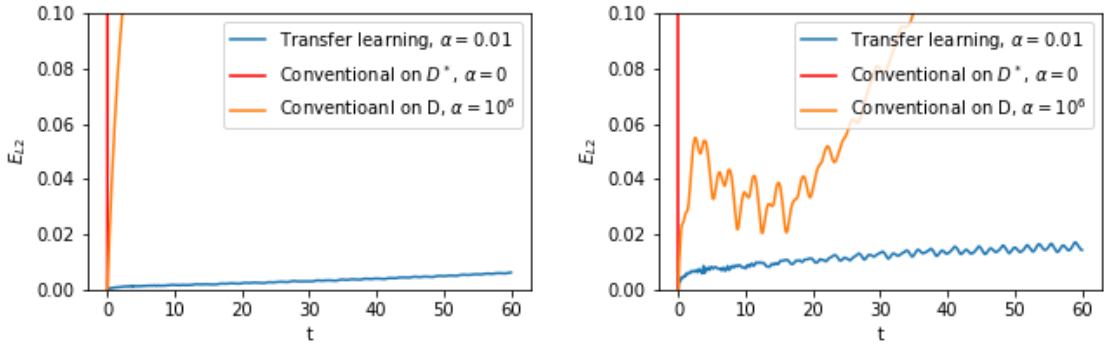


(d) Transfer Learning, TEST 8 ($s_U = -20\%$)



(e) Transfer Learning, TEST 9 ($s_U = 20\%$)

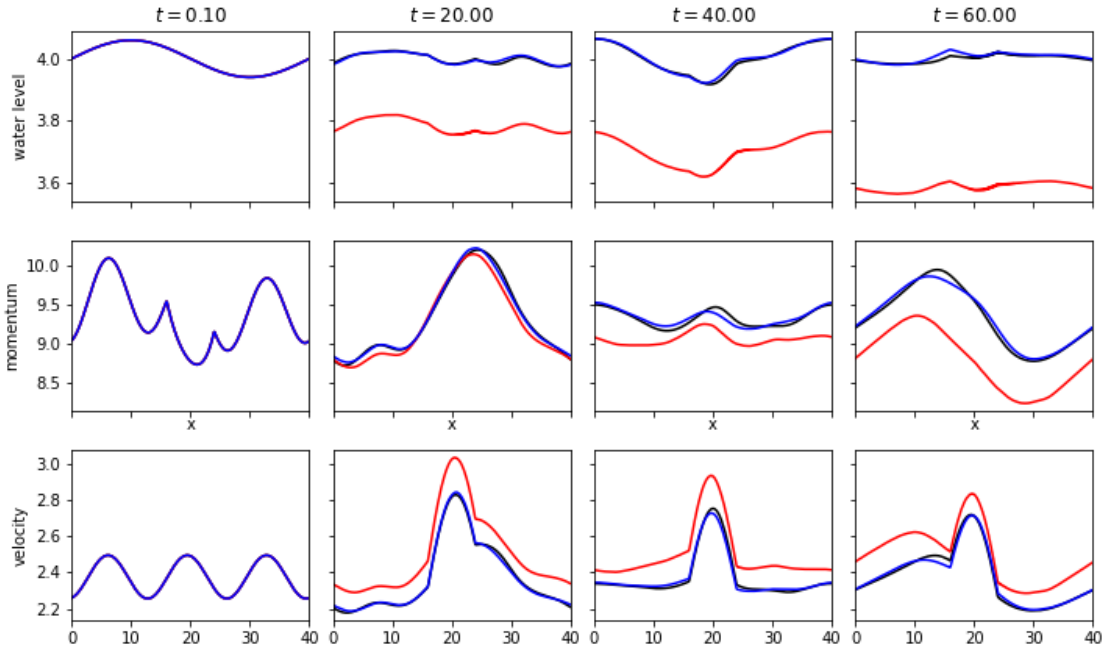
Figure 40: Averaged L^2 relative errors in ESN predictions of TEST 4 - TEST 9 using Transfer Learning. The total water height $h(x,t) + z(t)$ (blue) and momentum $h(x,t)u(x,t)$ (orange).



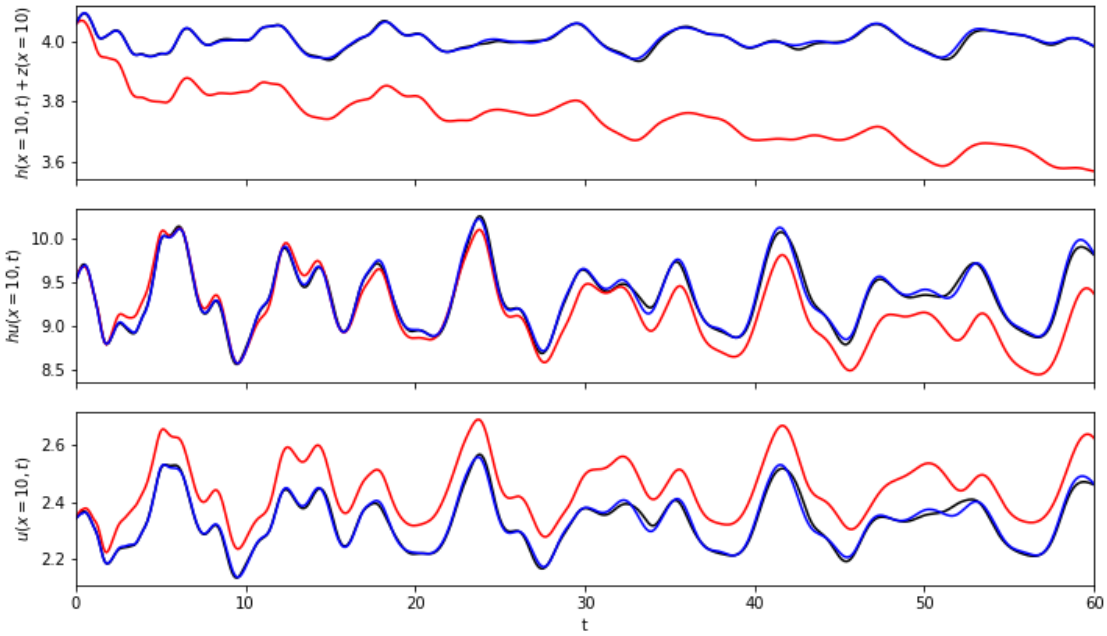
(a) water level

(b) momentum

Figure 41: Comparison of averaged L^2 relative errors in ESN predictions of TEST 8 with/without Transfer Learning for the the total water height $h(x,t) + z(t)$ (Left) and momentum $h(x,t)u(x,t)$ (Right). blue - ESN with transfer learning $\alpha = 0.01$, Red - $\alpha = 0$, same as conventional method with the target dataset $\mathcal{D}^*$, Orange - $\alpha = 10^6$, same as conventional method with the source domain $\mathcal{D}$.

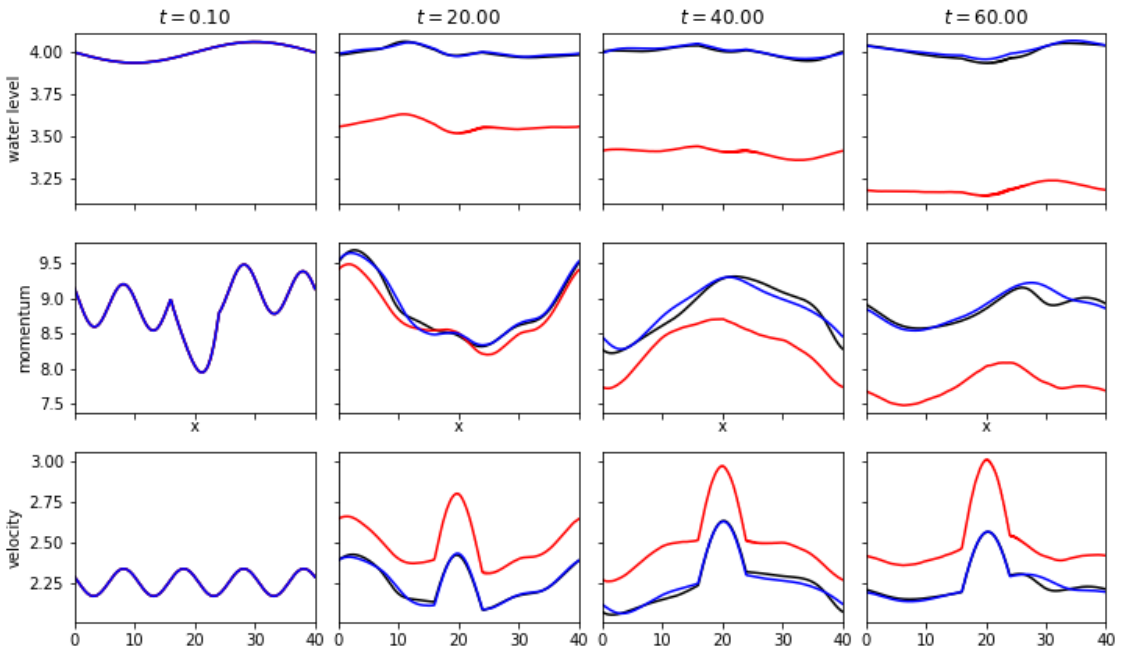


(a) Snapshots for $t = 0.1$, 20, 40, 60

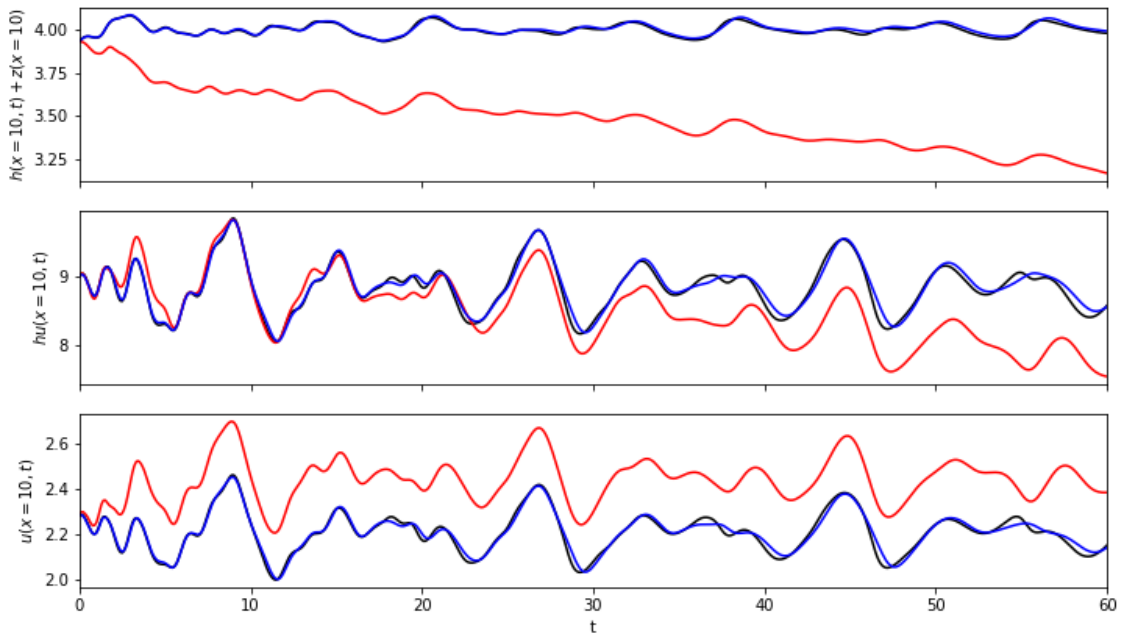


(b) Time-series for $x = 10$

Figure 42: Simulations for same TEST 4 ($s_U = -5\%$) sample in Figure 38. Comparison of direct numerical simulations of the SWE (black line), predictions of the ESN with conventional method (red line) and with Transfer Learning (blue line).

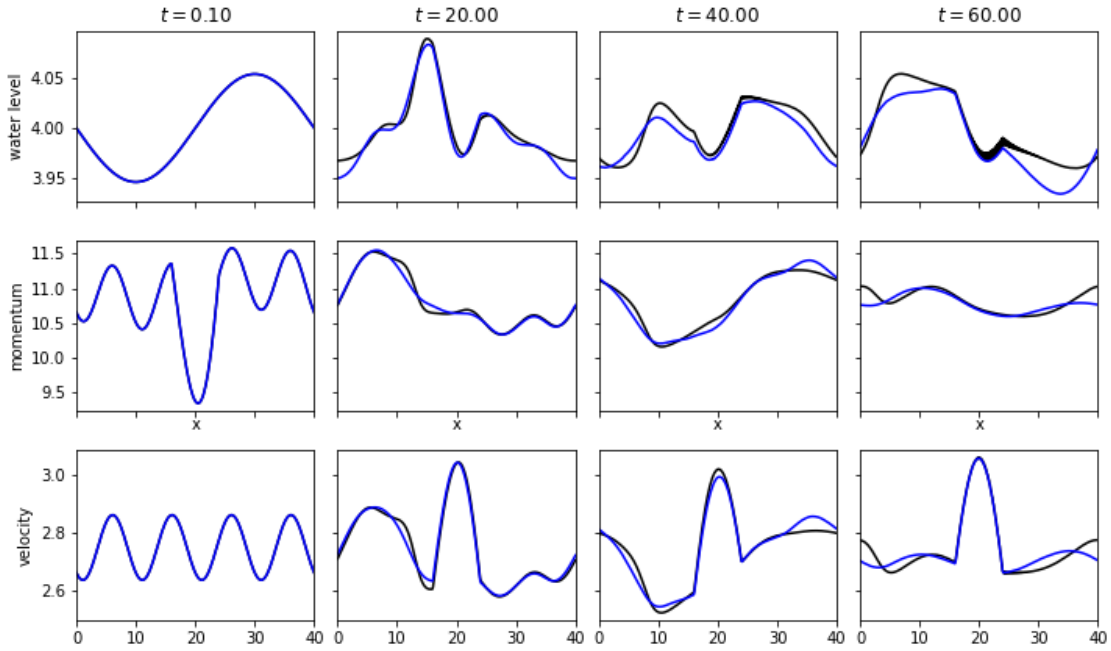


(a) Snapshots for $t = 0.1$, 20, 40, 60

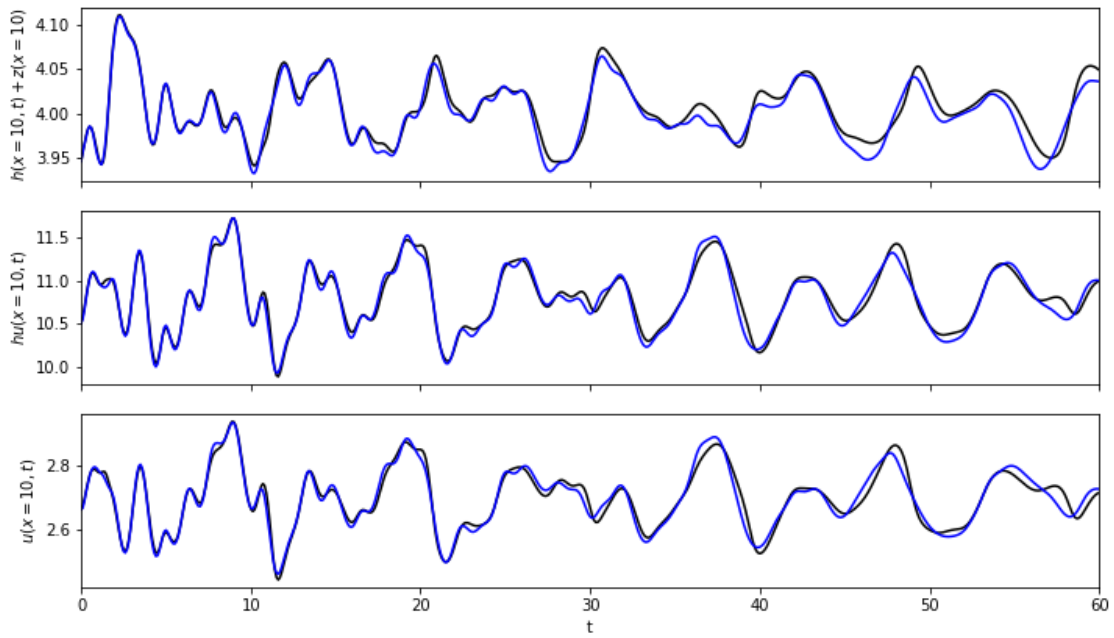


(b) Time-series for $x = 10$

Figure 43: Simulations for one sample in TEST 6 ($s_U = -10\%$) with transfer learning method. Comparison of direct numerical simulations of the SWE (black line), predictions of the ESN with conventional method (red line) and with Transfer Learning (blue line).

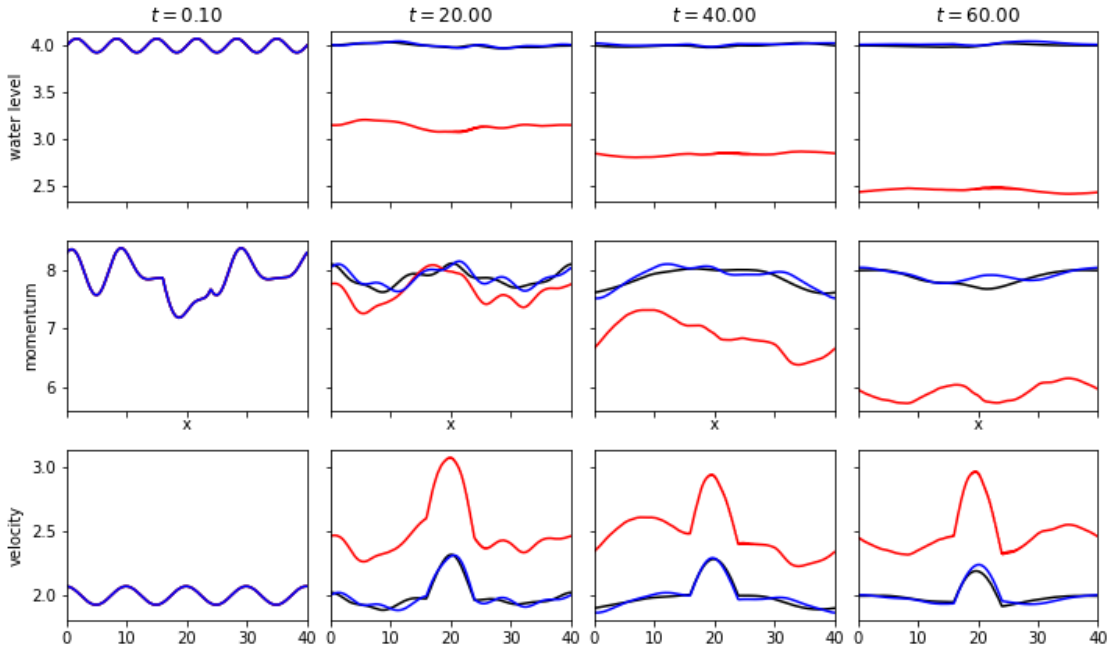


(a) Snapshots for $t = 0.1$, 20, 40, 60

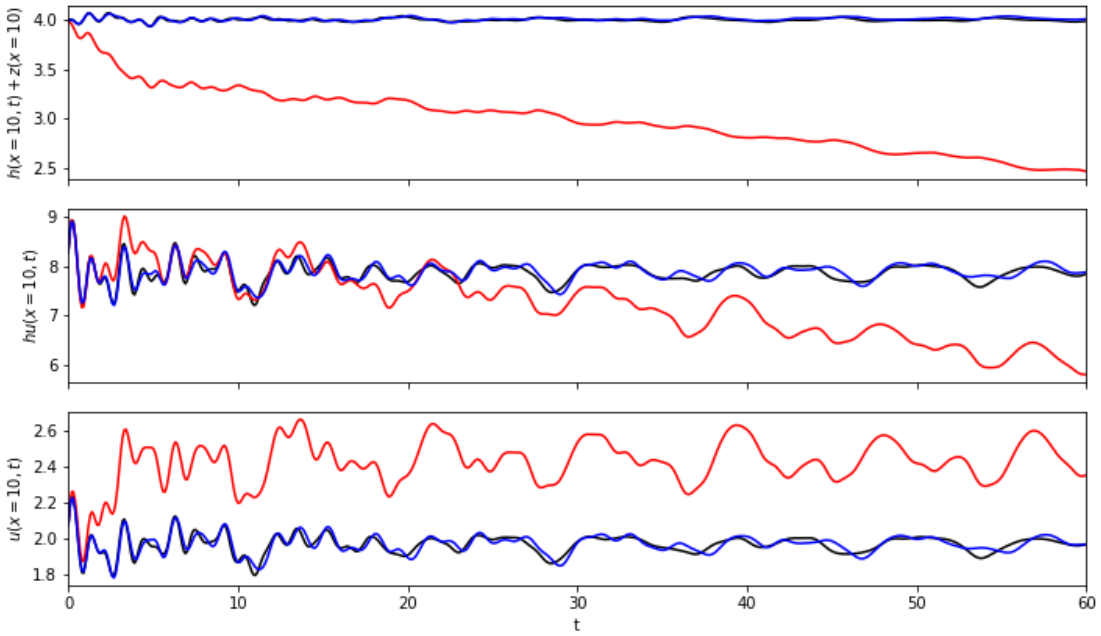


(b) Time-series for $x = 10$

Figure 44: Simulations for one sample in TEST 7 ($s_U = 10\%$) with transfer learning method. Comparison of direct numerical simulations of the SWE (black line), predictions of the ESN with Transfer Learning (blue line).

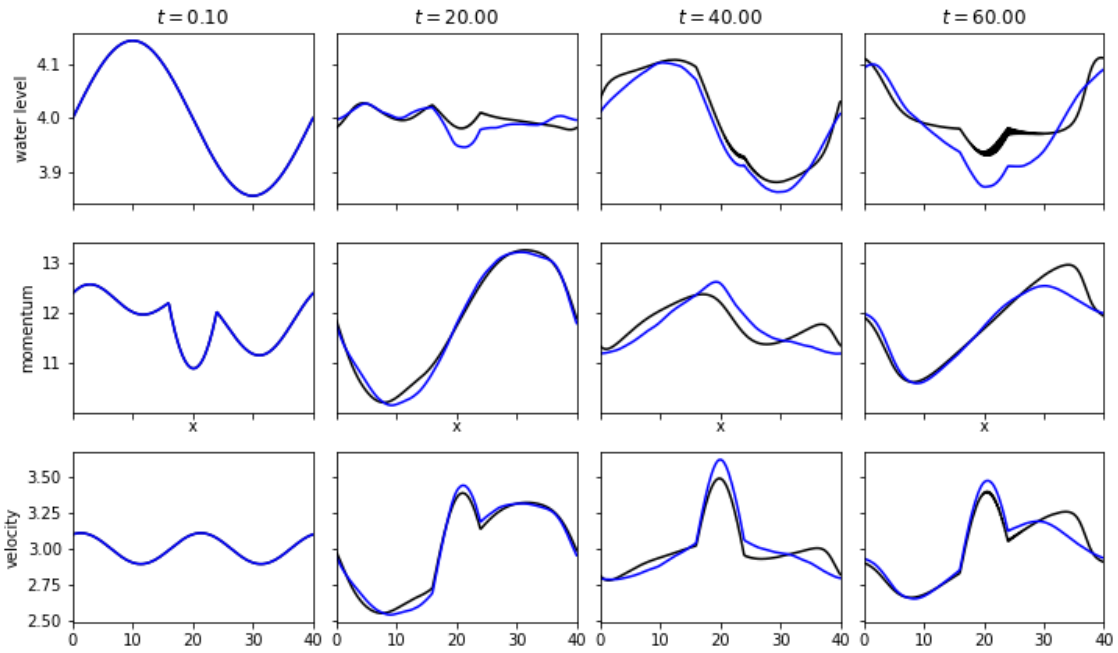


(a) Snapshots for $t = 0.1$, 20, 40, 60

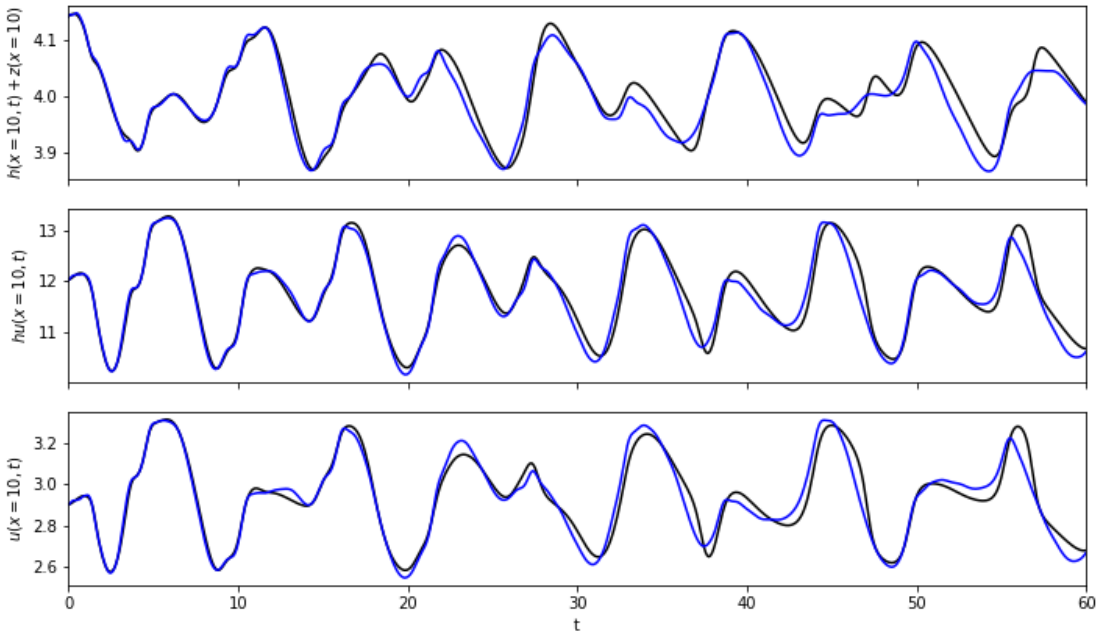


(b) Time-series for $x = 10$

Figure 45: Simulations for one sample in TEST 8 ($s_U = -20\%$) with transfer learning method. Comparison of direct numerical simulations of the SWE (black line), predictions of the ESN with conventional method (red line) and with Transfer Learning (blue line).



(a) Snapshots for $t = 0.1$, 20, 40, 60



(b) Time-series for $x = 10$

Figure 46: Simulations for one sample in TEST 9 ($s_U = 20\%$) with transfer learning method. Comparison of direct numerical simulations of the SWE (black line), predictions of the ESN with Transfer Learning (blue line).

7.3.2 Water Height Shift

In this section, we apply the transfer learning method to SWE testing dataset TEST 5 where the initial mean water level is shifted down 5%. Conventional ESN predictions failed for this testing dataset (section 7.2). We use the same transfer learning, as in section 7.3.1, i.e., $\alpha = 0.01$. For each "shifted" initial mean water level we generate target dataset $\mathcal{D}^*$ as a single SWE trajectory on $[0, 2]$ sampled with time step $\Delta = 0.1$. W_{out} is still same trained weights as the sections 7.1 and 7.2. Corrected weights $W_{out} + \delta W$ are used to test all other trajectories in TEST 5 dataset. In addition, we generate 3 testing datasets with different initial mean water levels with parameters summarized in Table 11. Initial conditions in these datasets are generated using (7.5) with $s_U = 0$ and different values of s_H, that are summarized in Table 11. The testing datasets used in this section are

TEST 5: mean water level is $H_0 = 3.8$ corresponds to a shift of -5%.

TEST 10: mean water level is $H_0 = 4.2$ corresponds to a shift of +5%.

TEST 11: mean water level is $H_0 = 3.92$ corresponds to a shift of -2%.

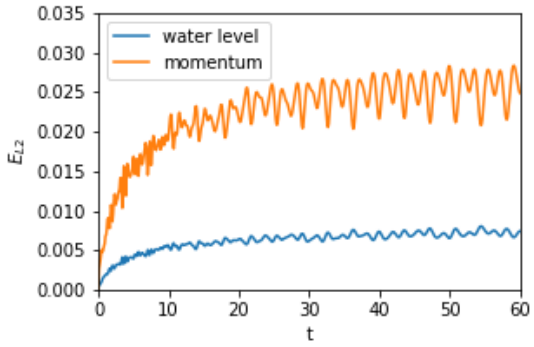
TEST 12: mean water level is $H_0 = 4.08$ corresponds to a shift of +2%.

Table 11: Parameters of testing datasets for ESN with fixed mean water height $U_0 = 2.5$ and different mean velocity H_0.

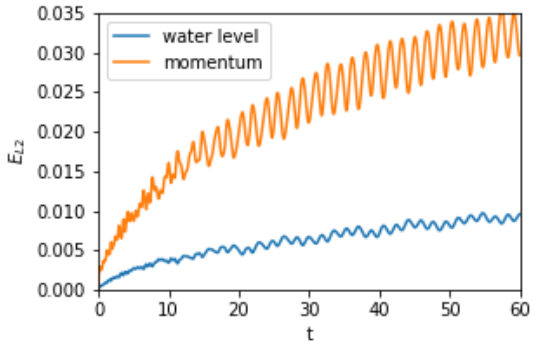
	size	H_0	pert. on water level	U_0	pert. on velocity
TEST 5	20	3.8	1%-5%	2.5	1%-5%
TEST 10	20	4.2	1%-5%	2.5	1%-5%
TEST 11	20	3.92	1%-5%	2.5	1%-5%
TEST 12	20	4.08	1%-5%	2.5	1%-5%

Averaged L^2 errors for all the four datasets are presented in Figure 47. In particular, compare Figures 39(e) (ESN prediction without transfer learning) and 47(a) (ESN prediction with transfer learning). We can see that transfer learning results in a considerable improvement of the ESN prediction. However, the performance of the ESN with transfer learning is still not ideal for $|s_H| = 5\%$, especially for $t \geq 20$ since errors for the water level prediction can reach 2.5% - 3% (Figures 47(a)(b)). It is difficult to improve performance of the ESN. We tried to use longer

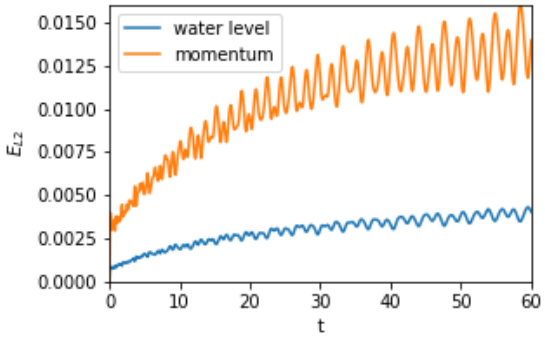
time-intervals in the target dataset $\mathcal{D}^*$ for computing correction. In particular, we tried to generate $\mathcal{D}^*$ by using a longer SWE trajectories on $[0, 10]$ and $[0, 30]$, but larger target data $\mathcal{D}^*$ did not lead to a considerable improvement (as shown in Figure 48). Our results indicate that predicting the SWE trajectories with "shifts" in the mean initial water level is more challenging compared to the prediction of trajectories with different mean velocity. Figures 49 and 50 depict snapshots for $|s_H| = 5\%$.



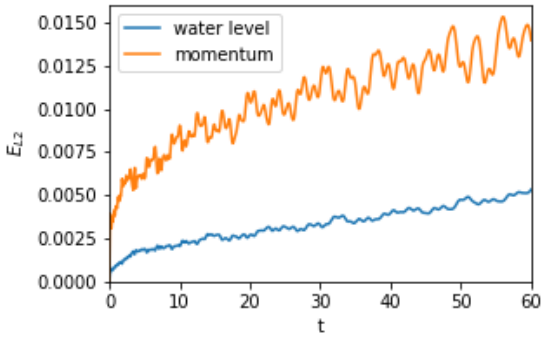
(a) Transfer Learning, TEST 5 ($s_H = -5\%$)



(b) Transfer Learning, TEST 10 ($s_H = 5\%$)



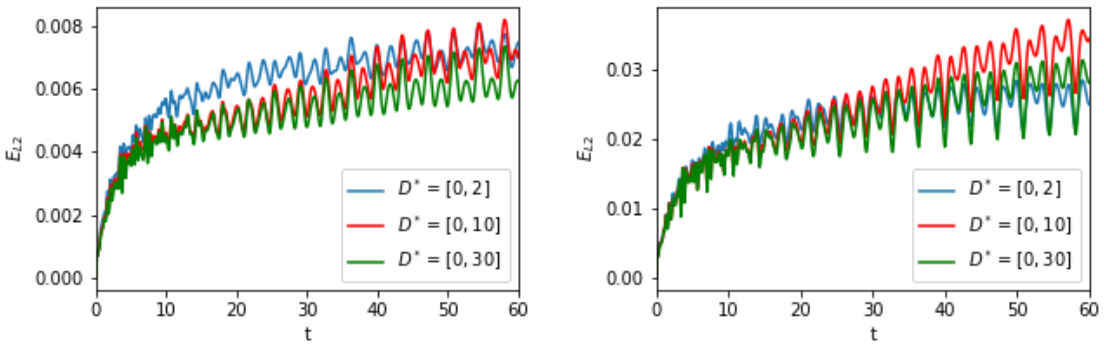
(c) Transfer Learning, TEST 11 ($s_H = -2\%$)



(d) Transfer Learning, TEST 12 ($s_H = 2\%$)

Figure 47: Averaged L^2 relative errors (4.11) in ESN predictions of TEST 5 and TEST 10-12. The total water height $h(x,t) + z(t)$ (blue) and momentum $h(x,t)u(x,t)$ (orange) using Transfer Learning.

Performance of the ESN prediction with transfer learning improves considerably for smaller "shifts" of the initial mean water level. In particular, averaged errors are approximately 1.5% for longer times for $|s_H| = 2\%$ (Figures 47(c)(d)). Corresponding time-series are depicted in Figures 51 and 52.



(a) water level

(b) momentum

Figure 48: Comparison of averaged L^2 relative errors in ESN predictions of TEST 5 using different time-intervals for generating the target dataset $\mathcal{D}^*$. The bigger size of the target dataset $\mathcal{D}^*$ only slightly improves the performance. The total water height $h(x,t) + z(t)$ (Left) and momentum $h(x,t)u(x,t)$ (Right).

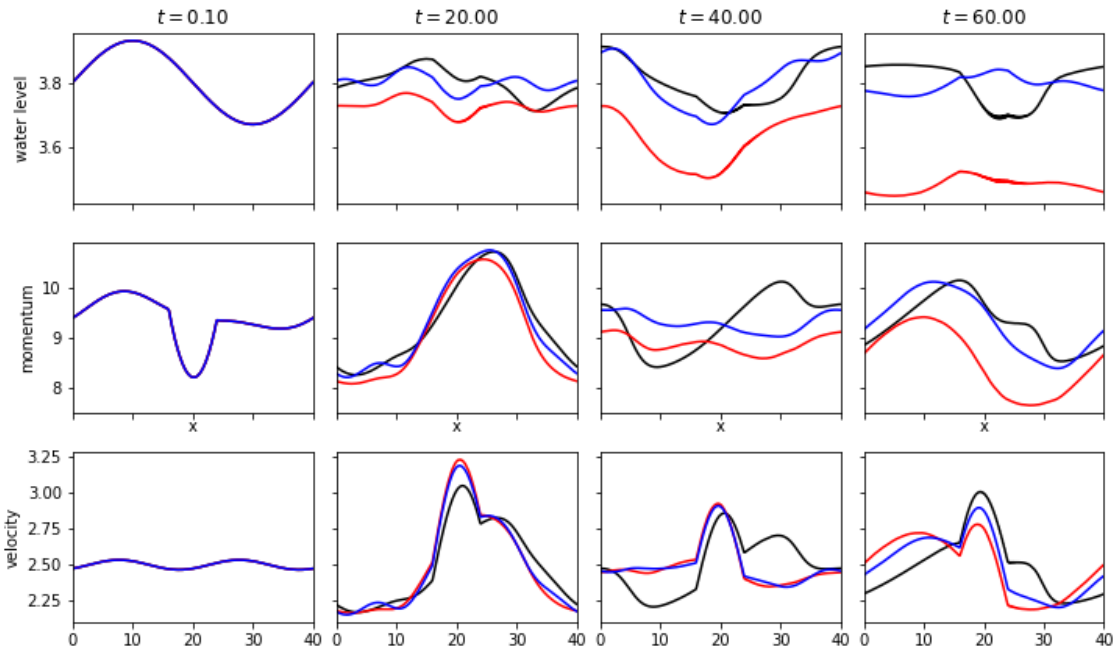
If we consider trajectories in TEST 2 dataset, then the maximum water level difference (highest water level - lowest water level) is 0.4116 and the maximum velocity difference (largest velocity - smallest velocity) is 1.3334. This corresponds to approximately 10% of the mean water level $H_0 = 4$, and more than 50% of initial mean velocity mean $U_0 = 2.5$. Therefore, it is likely that larger "shifts" of the initial mean water level are stronger perturbations of the background state (5.7) since the source dataset does not contain many time-instances with $|s_H| = 5\%$ increase/decrease of the water level.

7.3.3 Conclusion

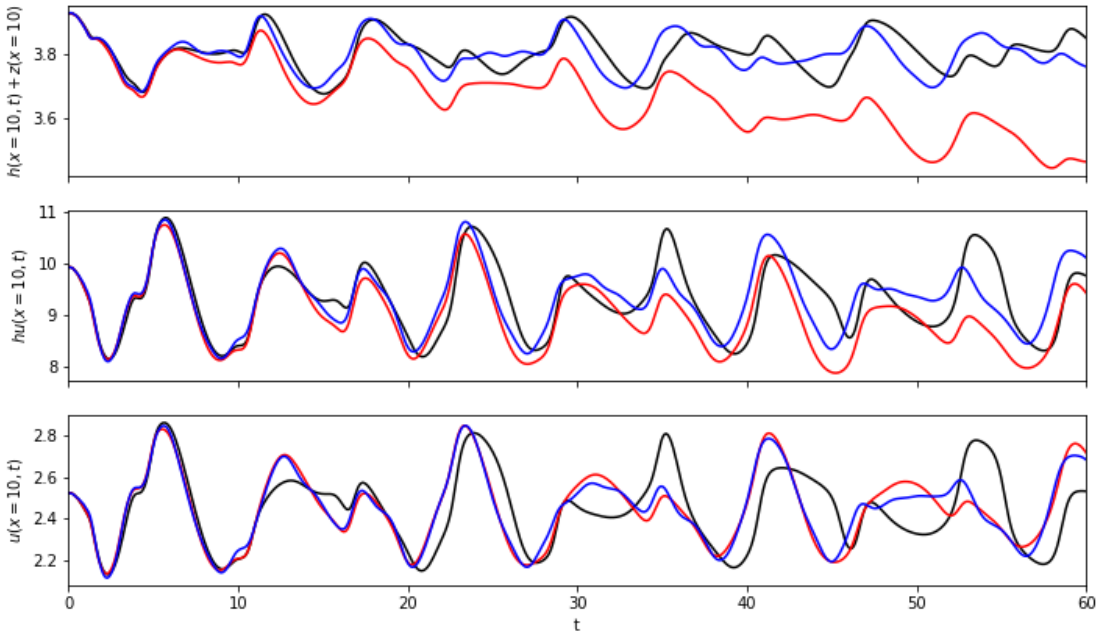
ESN with transfer learning has much better performance for predicting SWE trajectories with "shifted" initial mean water level or mean initial velocity compared with the conventional ESN. In addition, we can also conclude that transfer learning is essential for reproducing ensemble properties with such "shifted" initial conditions. Although the target dataset $\mathcal{D}^*$ is rather small, transfer learning makes a considerable difference in predicting individual trajectories and ensemble properties.

The dynamics of the SWE seem to be more sensitive to the changes in the magnitude of the

water level than the magnitude of the velocity. Therefore, ESN with transfer learning does better predicting trajectories with "shifted" initial mean velocity than trajectories with "shifted" initial mean water level. Thus, ESN with transfer learning can predict with high accuracy properties of an ensemble of solution with "shifted" mean water level and mean velocity up to $\pm 2\%$ and $\pm 5\%$, respectively. The corresponding errors for the "shifted" initial mean water height and velocity are 3%-4% and about 2%, respectively.

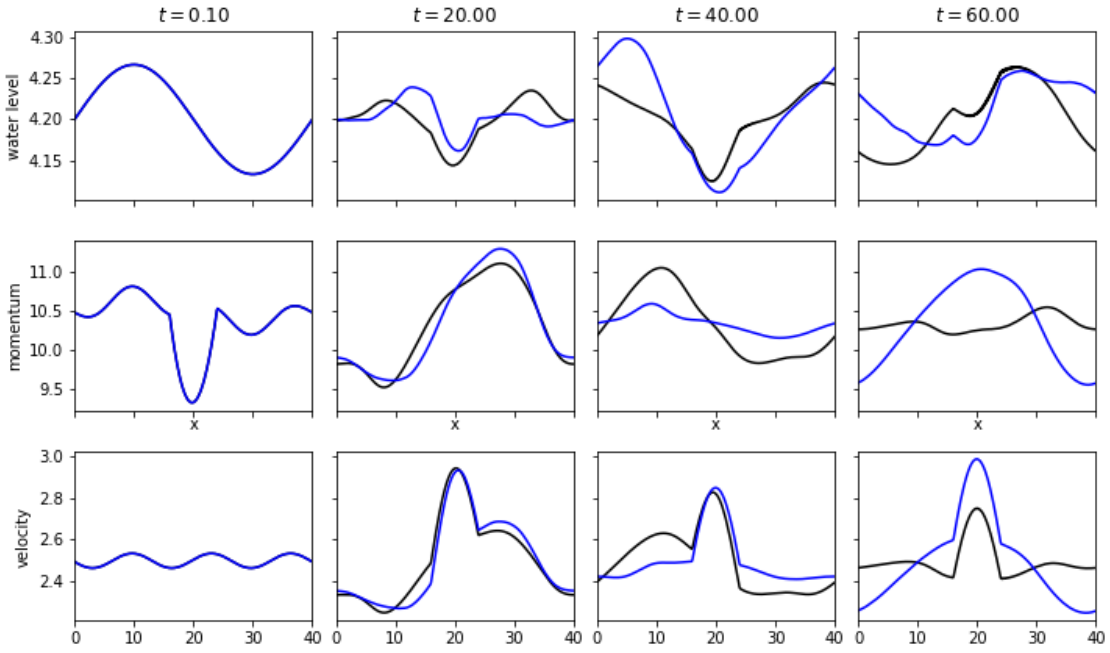


(a) Snapshots for $t = 0.1$, 20, 40, 60

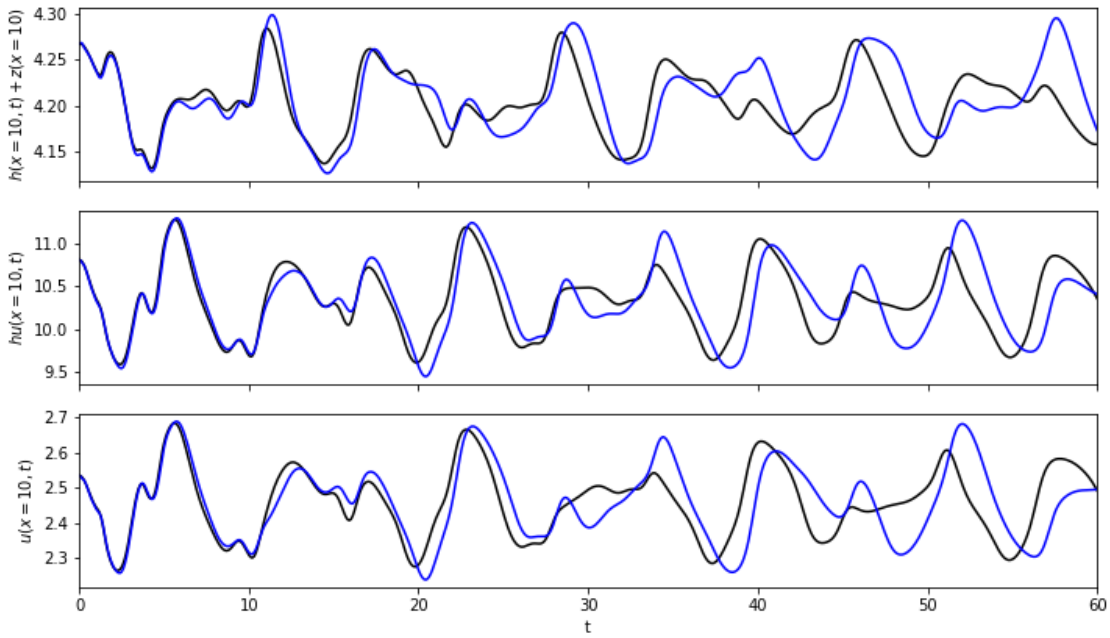


(b) Time-series for $x = 10$

Figure 49: Simulations for TEST 5 ($s_H = -5\%$) (see Figure 39) with transfer learning method. Comparison of direct numerical simulations of the SWE (black line), predictions of the ESN with conventional method (red line) and with Transfer Learning (blue line).

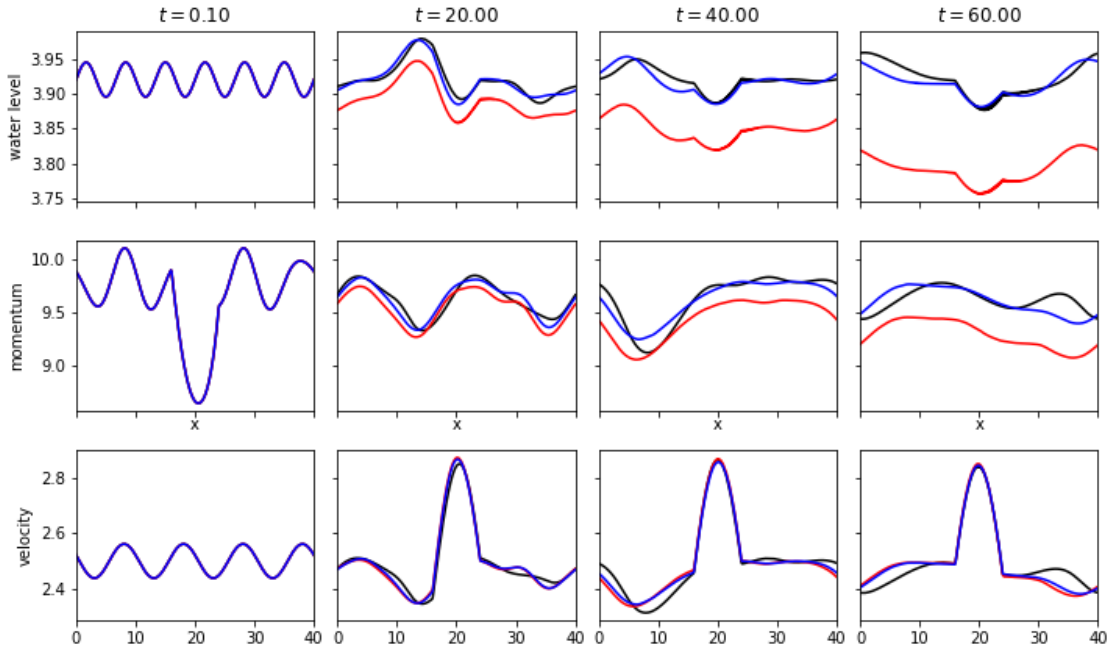


(a) Snapshots for $t = 0.1$, 20, 40, 60

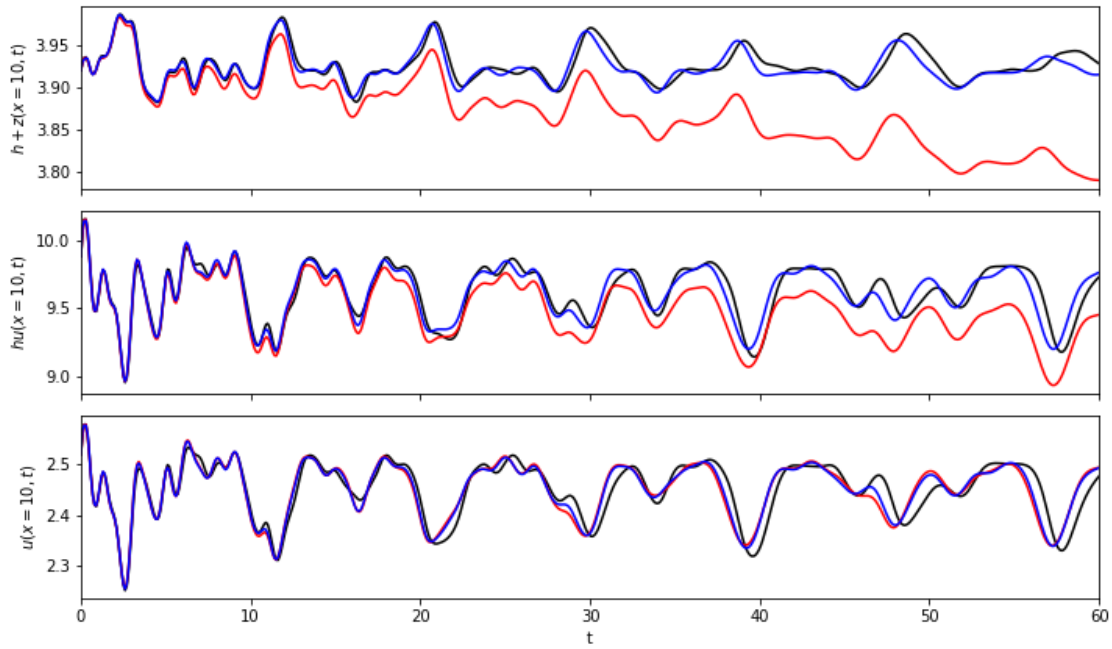


(b) Time-series for $x = 10$

Figure 50: Simulations for TEST 10 ($s_H = 5\%$) with transfer learning method. Comparison of direct numerical simulations of the SWE (black line), predictions of the ESN with Transfer Learning (blue line).

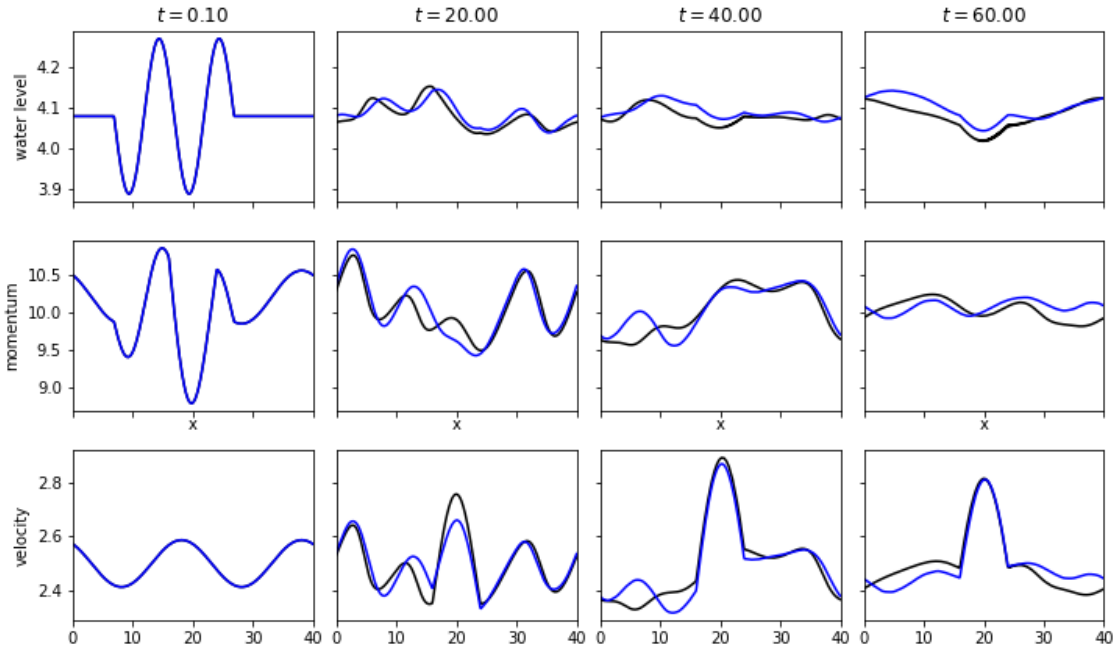


(a) Snapshots for $t = 0.1$, 20, 40, 60

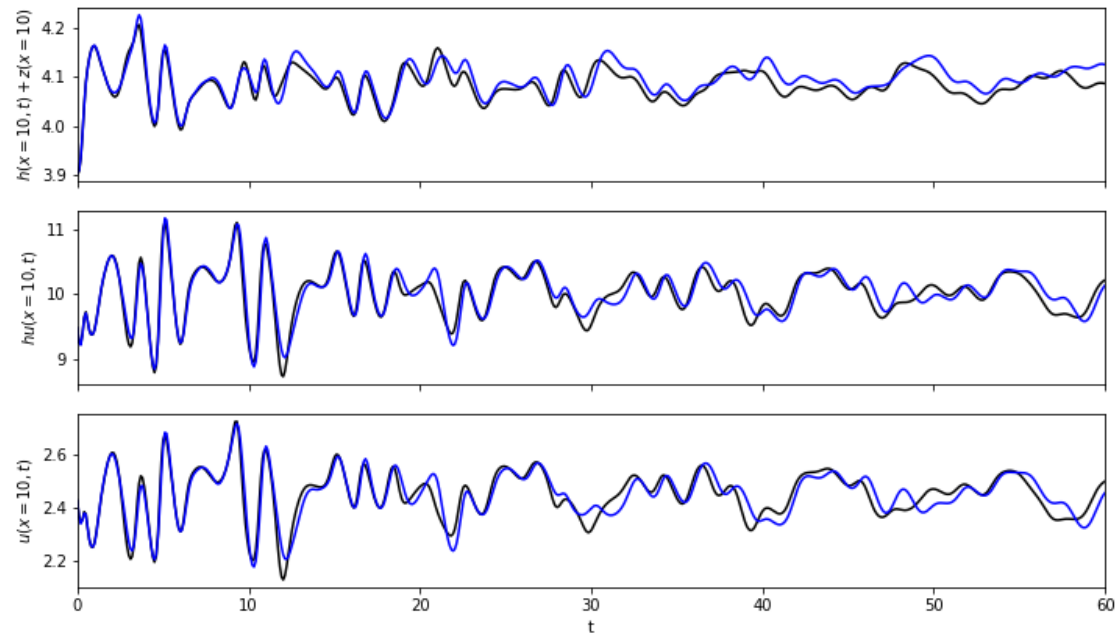


(b) Time-series for $x = 10$

Figure 51: Simulations for TEST 11 ($s_H = -2\%$) with transfer learning method. Comparison of direct numerical simulations of the SWE (black line), predictions of the ESN with conventional method (red line) and with Transfer Learning (blue line).



(a) Snapshots for $t = 0.1$, 20, 40, 60



(b) Time-series for $x = 10$

Figure 52: Simulations for TEST 12 ($s_H = 2\%$) with transfer learning method. Comparison of direct numerical simulations of the SWE (black line), predictions of the ESN with Transfer Learning (blue line).

7.4 Ensemble Simulations

There are many ensembles-based numerical methods, such as data-assimilation, uncertainty quantification, inverse problems. Direct numerical implementation of such methods is often extremely costly, since in order to generate ensembles of a reasonable size, it is necessary to solve the initial-value problem many times. Thus, there is always a balance between the sampling error (number of ensemble members) and model error (accuracy of the numerical solver). However, the accuracy of individual solutions is not the main emphasis in such methods. Instead, it is important to reproduce ensemble properties such as ensemble mean, variance, and covariance, higher moments, etc. Thus, reduced-order models can play a particularly important role in such applications, since they sacrifice the accuracy of individual solutions in favor of faster computations and, thus, producing more ensemble members and reducing the sampling error.

In this section, we investigate how well the ESN methodology reproduces properties of an ensemble of SWE solutions. To this end, we generate many trajectories with different initial conditions using (5.8) and compare the mean and variance of the ensemble of "true" solution with ensemble generated using predictions of the ESN. In particular, we perform two numerical experiments. In the first experiment we generate an ensemble of solutions with initial conditions in (5.10) and test how well ESN is able to reproduce statistical properties of this ensemble. In the second experiment, we generate an ensemble of solutions with initial conditions with "shifted" mean water level and mean initial velocity and utilize ESN with transfer learning to predict ensemble properties.

In each ensemble experiment we generate $N = 100$ trajectories $X_i(x,t)$ with $i = 1, \ldots, N$. Each ensemble member, $X_i(x,t)$, consists of two vectors, the water height and the momentum, i.e., $X_i(x,t) = \{h_i(x,t) + z(x), h_i(x,t)u_i(x,t)\}$. If we denote $v_i(x,t) = h_i(x,t) + z(x)$, then we define the mean and standard deviation/ variance of the ensemble for water height as

$$M(x,t) = \frac{1}{N}\sum_{i=1}^{N} v_i(x,t), \quad STD(x,t) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\big(M(x,t) - v_i(x,t)\big)^2}.$$

We use similar formulas for computing ensemble mean and standard variance for the momentum with $v_i(x,t) = h_i(x,t)u_i(x,t)$. In this section we compare ensemble mean and variance computed by performing direct numerical simulations of the SWE and computed by using the predictions of the ESN.

7.4.1 First Experiment

This experiment generates an ensemble of solutions of the SWE with the same mean initial water level, H_0 and initial mean velocity U_0. In particular, we generate the following ensemble of solutions:

1. Generate $N = 100$ trajectories with initial conditions (5.10) for ensemble simulation. The perturbations level a is set to 1%; the initial velocity is constant $U_0 = 2.5$.

$$h(x,0) + z(x) = \begin{cases} 4 \pm 4a \sin(2k\pi(x - \phi_1 L)/L), & x \in [\phi_1 L, \phi_2 L], \\ 4, \quad \text{otherwise}, \end{cases}$$

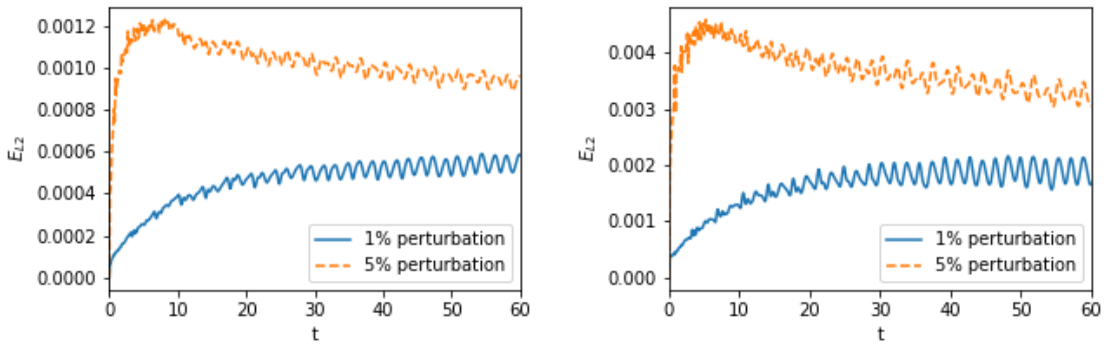
$$u(x,0) = 2.5.$$

2. Use previously trained ESN from chapter 6 (same as in sections 7.1 and 7.2) to predict these 100 out-of-sample solutions starting with $T_0 = 0$; no transfer learning is used here.

3. Use predictions of the ESN for individual trajectories to compute the ensemble mean and variance for water level $h(x,t) + z(x)$ and momentum $hu(x,t)$.

4. Generate another ensemble (both direct numerical simulations and predictions of the ESN) by repeating steps 1-3 with $a = 5\%$ perturbation level in initial conditions (5.10).

Figure 53 shows the averaged prediction error of ensemble simulations. Clearly, the initial perturbation level has an impact on the magnitude of the prediction error. We would like to point out that the averaged L^2 error is computed for individual trajectories. We can see that prediction errors are small (less than 1%). Thus, we can see that ESN has a very good utility of prediction and we expect that ESN reproduces the statistical properties of the ensemble very well.

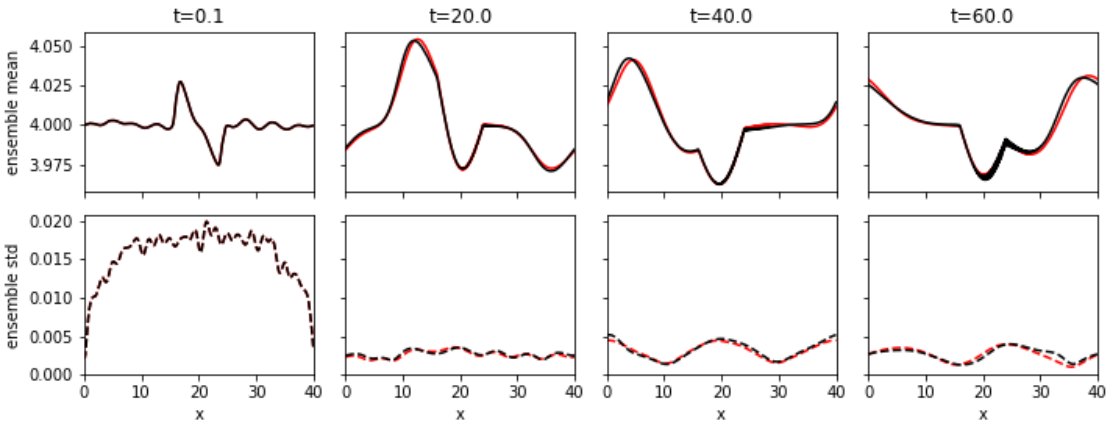


(a) water level

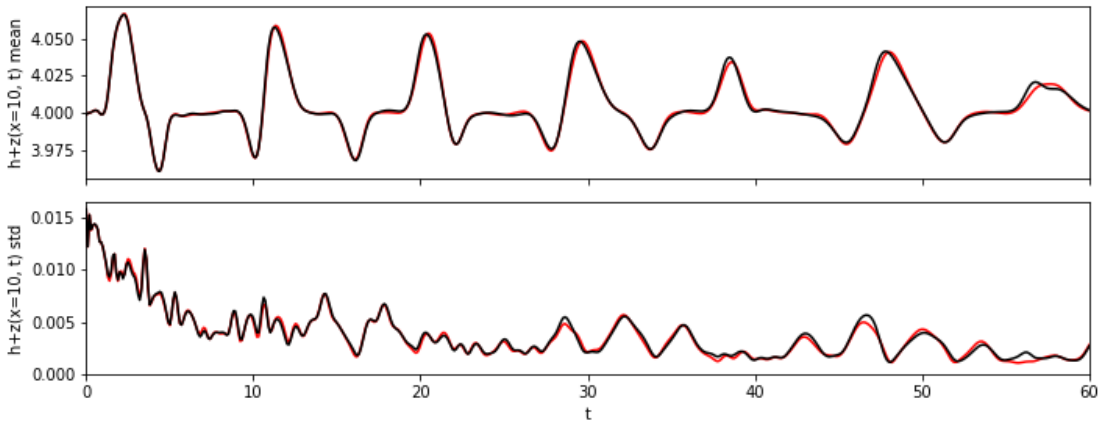
(b) momentum

Figure 53: Averaged L^2 relative errors of individual solutions in the first ensemble simulation with initial conditions (5.10) with $a = 1\%$ and $a = 5\%$. Left - total water height $h(x,t) + z(x)$. Right - momentum $h(x,t)u(x,t)$.

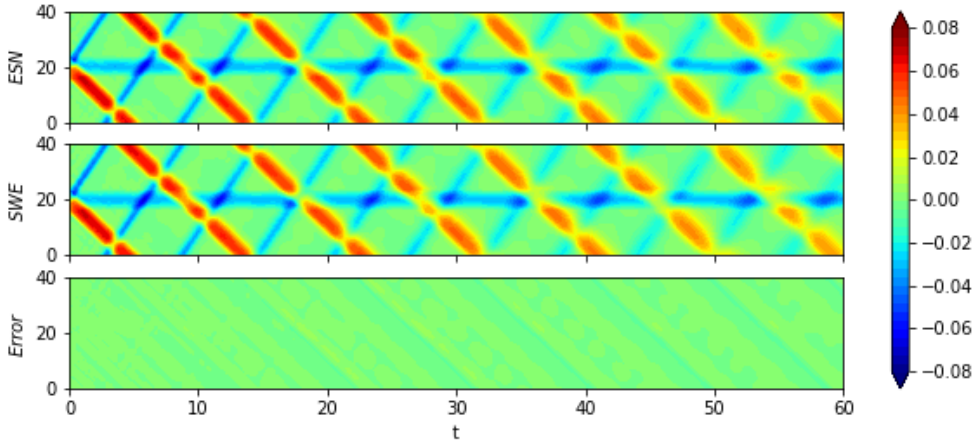
Figures 54 and 55 show results of predicting ensemble mean and standard deviation for water level and momentum for the ensemble with $a = 1\%$ initial perturbation. Figures 56 and 57 show results of predicting ensemble mean and standard deviation for water level and momentum for the ensemble with $a = 5\%$ initial perturbation. First, we observe that standard variance of the ensemble becomes very small for larger times. This is known as "ensemble collapse". This is a direct consequence of including the dissipative term $(hu)_{xx}$ in the equations. Since there is dissipation, energy decays in time and all solutions converge to a flat steady state.



(a) Snapshots of the mean (top) and standard variance (bottom) for $t = 0.1$, 20, 40, 60

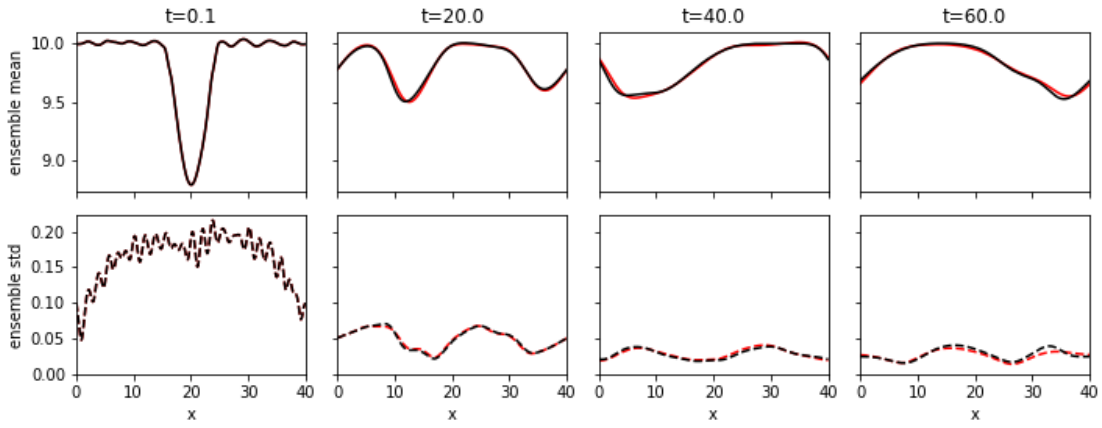


(b) Time-series for the mean (top) and standard variance (bottom) at $x = 10$

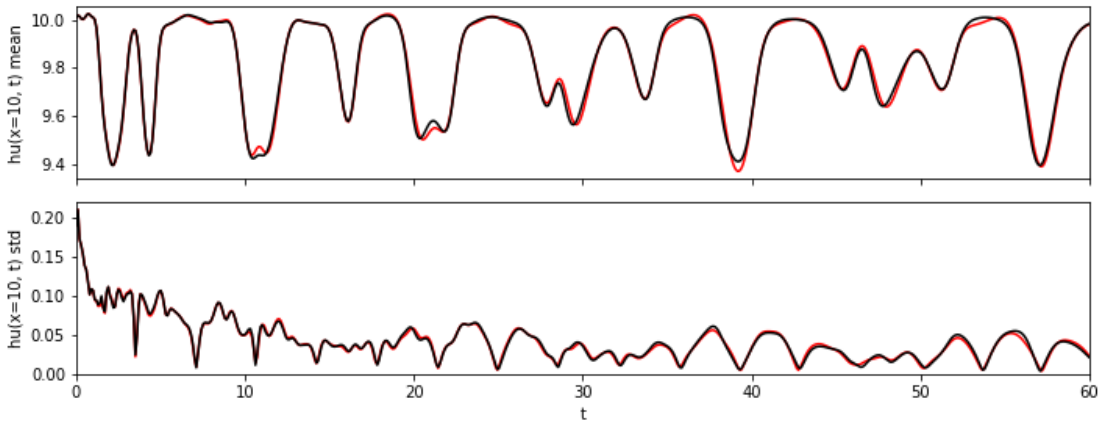


(c) Ensemble Mean $M(x,t) - 4.0$; Top - ESN, Middle - DNS SWE, Bottom - Abs. Error

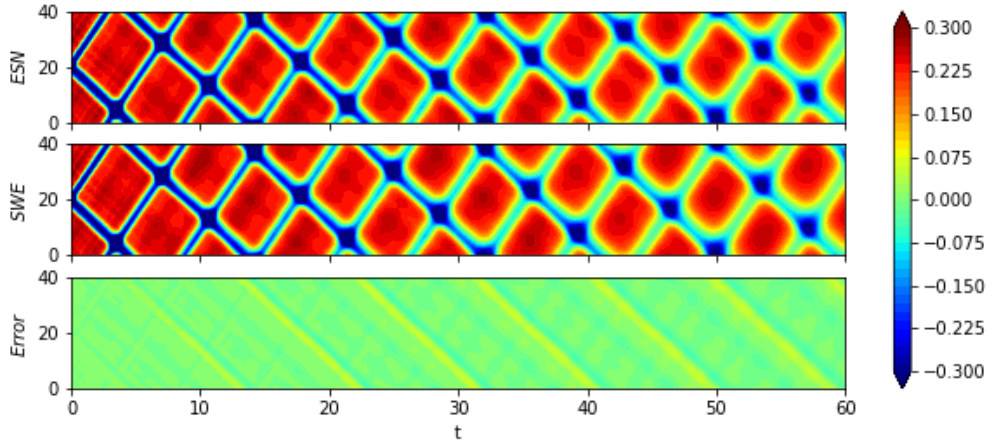
Figure 54: First ensemble experiment with $a = 1\%$. Comparison of statistical properties of the ensemble for the total water level $h(x,t) + z(x)$ using direct numerical simulations of the SWE (black line) and predictions of the ESN (red line).



(a) Snapshots of the mean (top) and standard variance (bottom) for $t = 0.1$, 20, 40, 60

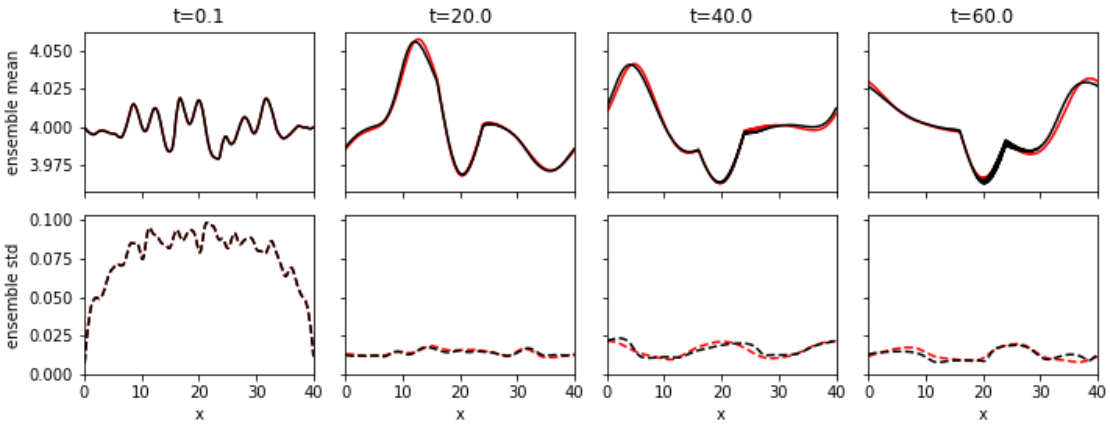


(b) Time-series for the mean (top) and standard variance (bottom) at $x = 10$

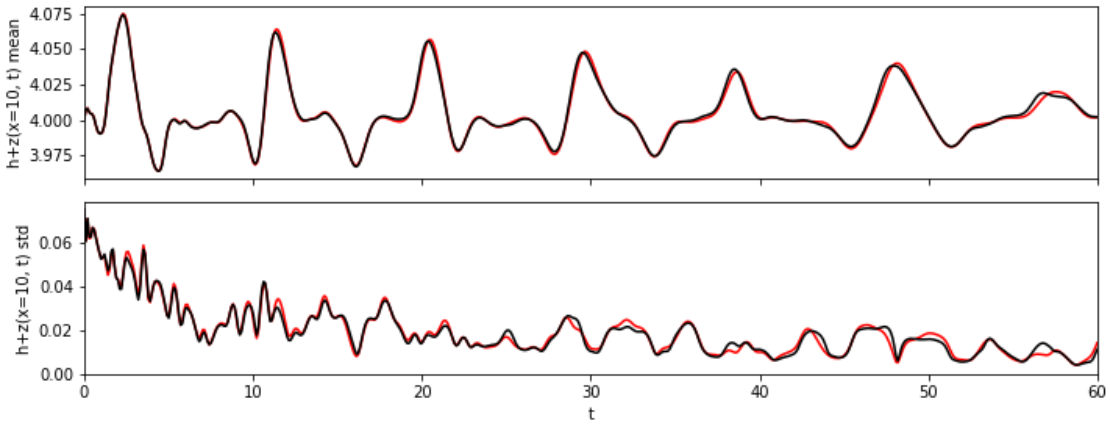


(c) Ensemble Mean $M(x,t) - 9.75$; Top - ESN, Middle - DNS SWE, Bottom - Abs. Error

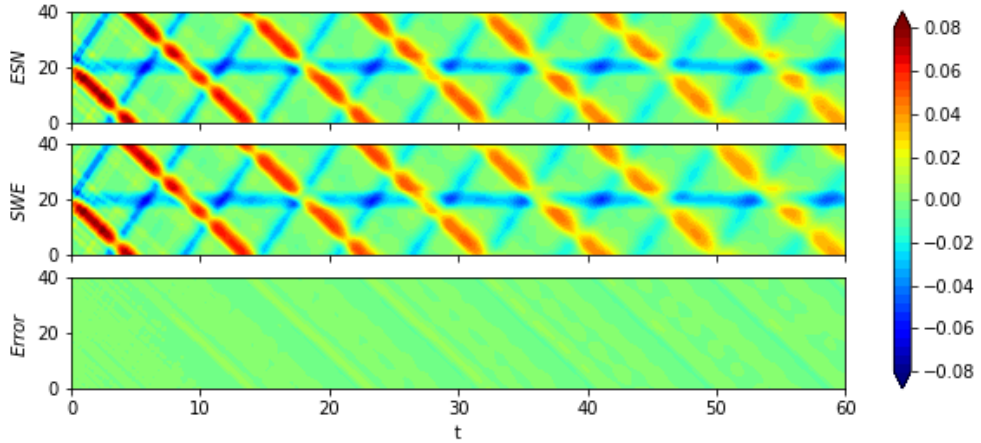
Figure 55: First ensemble experiment with $a = 1\%$. Comparison of statistical properties of the ensemble for the momentum $h(x,t)u(x,t)$ using direct numerical simulations of the SWE (black line) and predictions of the ESN (red line).



(a) Snapshots of the mean (top) and standard variance (bottom) for $t = 0.1$, 20, 40, 60

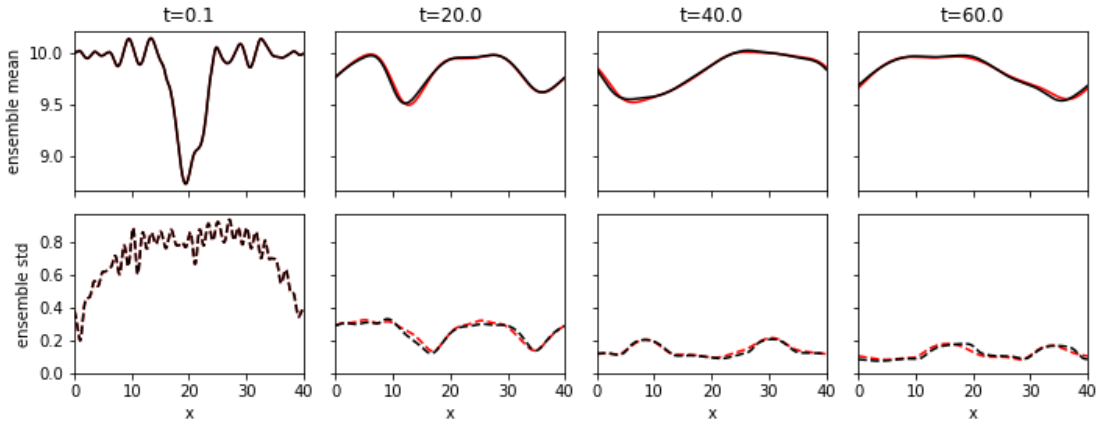


(b) Time-series for the mean (top) and standard variance (bottom) at $x = 10$

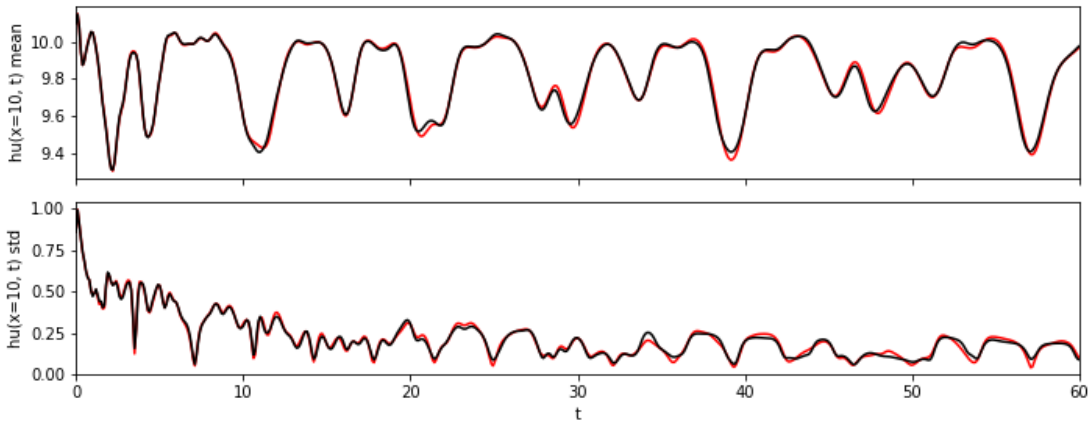


(c) Ensemble Mean $M(x,t) - 4.0$; Top - ESN, Middle - DNS SWE, Bottom - Abs. Error

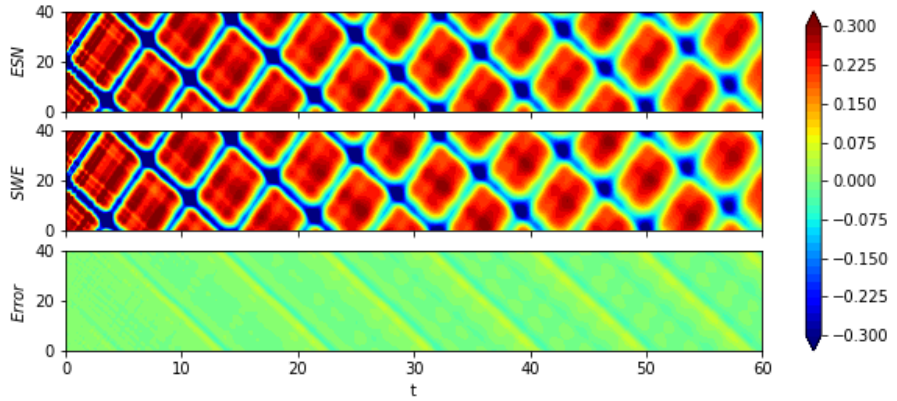
Figure 56: First ensemble experiment with $a = 5\%$. Comparison of statistical properties of the ensemble for the total water level $h(x,t) + z(x)$ using direct numerical simulations of the SWE (black line) and predictions of the ESN (red line).



(a) Snapshots of the mean (top) and standard variance (bottom) for $t = 0.1$, 20, 40, 60



(b) Time-series for the mean (top) and standard variance (bottom) at $x = 10$



(c) Ensemble Mean $M(x,t) - 9.75$; Top - ESN, Middle - DNS SWE, Bottom - Abs. Error

Figure 57: First ensemble experiment with $a = 5\%$. Comparison of statistical properties of the ensemble for the momentum $h(x,t)u(x,t)$ using direct numerical simulations of the SWE (black line) and predictions of the ESN (red line).

7.4.2 Second Experiment

In this ensemble experiment, we consider initial conditions with "shifted" mean initial water level and mean initial velocity. We demonstrated in section 7.3 that in this case transfer learning is essential for accurate prediction of trajectories. Thus, we include incorporate transfer learning into ESN ensemble predictions. We perform numerical integration of the SWE equations with parameters in Table 8 and sample the solution with sampling time step $\Delta = 0.1$. Other parameters of the second ensemble experiment are as follows:

1. We generate $N = 100$ trajectories $X_i(x,t)$ with $i = 1, \ldots, N$ using initial conditions:

$$\begin{aligned} h(x,0) + z(x) &= H_0(1 \pm s_H) \pm a\, H_0 \, \sin(2k\pi x/L + \omega_1) \quad x \in [0, L], \\ u(x,0) &= U_0(1 \pm s_U) \pm d\, U_0 \, \sin(2p\pi x/L + \omega_2) \quad x \in [0, L], \end{aligned} \tag{7.8}$$

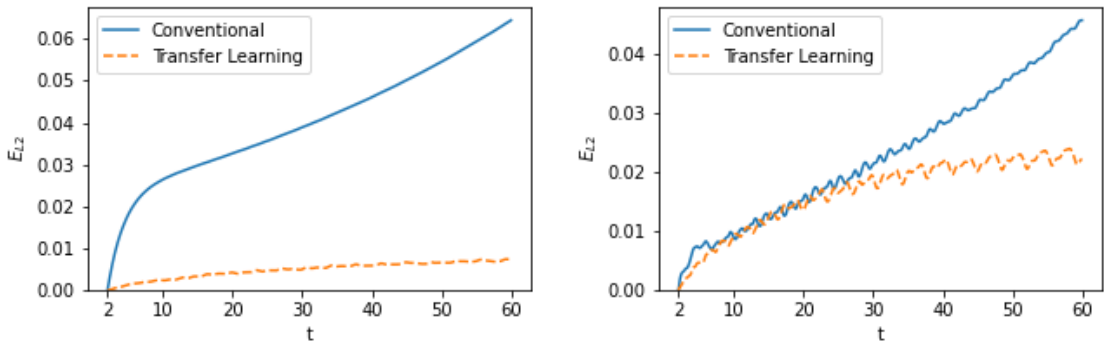
   where shifts for water level and velocity are $s_H \sim Uniform[0.01, 0.02]$ and $s_U \sim Uniform[0.01, 0.05]$, respectively. Parameters a and d are perturbation levels for the water level and velocity, respectively. Both a, $d \ \sim Uniform[0.01, 0.05]$. Frequency for the perturbation are discrete uniform random variables with $k, p \sim Uniform(1, 2, 3)$ and $\omega_1, \omega_2 \sim Uniform[0, 2\pi]$.

2. We use the previously trained ESN with the settings from chapter 6 (same as in sections 7.1 and 7.2) to perform prediction of trajectories on time interval $[2, 60]$, i.e., prediction starts with time $T_0 = 2$. Prediction time $T_0 = 2$ is chosen to be consistent with the transfer learning approach discussed below.

3. We use the previously trained ESN from sections 7.1 and 7.2 and apply transfer learning approach (same parameters as in section 7.3; transfer rate $\alpha = 0.01$) on time interval $[0, 2]$. Then we use ESN with transfer learning method to predict trajectories and ensemble properties on time interval $[2, 60]$.

Figure 58 shows the averaged prediction error of ensemble simulations. Similar to results presented in section 7.3 transfer learning leads to a considerable improvement of trajectory prediction

for the water height by the ESN. Figure 58 also demonstrates that ESN predictions for the momentum are less affected by the application of transfer learning, especially for shorter times. This figure provides a strong indication that ESN with transfer learning should be able to predict ensemble properties quite well.



(a) water level (b) momentum

Figure 58: Averaged L^2 relative errors of individual solutions in the second ensemble simulation with initial conditions (7.8) without (conventional) and with transfer learning. Left - total water height $h(x,t)+z(x)$. Right - momentum $h(x,t)u(x,t)$.

Figures 59 and 60 depict comparison of the ensemble mean and std. deviation prediction for the water height between the direct numerical simulations of the SWE and simulations of the ESN without transfer learning and with transfer learning, respectively. We can see that transfer learning leads to a significant improvement for both, the mean and the standard deviation in the ensemble simulations of the water height. Standard deviation in the ensemble predictions of the ESN without transfer learning increases (almost) linearly in time (Figure 59(b) bottom plot), leading to a completely incorrect predictions for the ensemble spread by the time $t = 20$ (about 75% relative errors). Relative errors for the standard deviation are even larger at time $t = 40$ (about 120% relative errors) and $t = 60$ (more than 200% relative errors). Therefore, we can conclude that transfer learning is essential in reproducing the ensemble properties using ESN predictions.

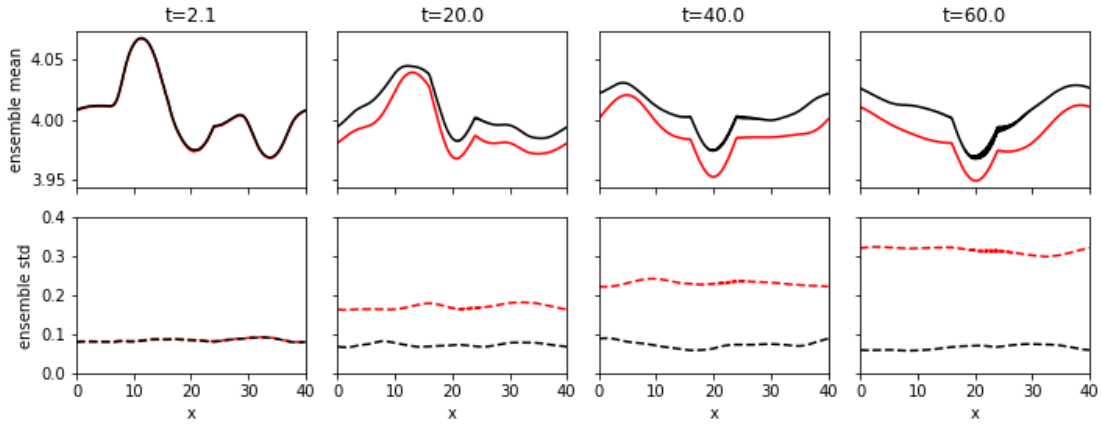
Figures 61 and 62 depict comparison of the ensemble mean and std. deviation prediction for the momentum. Similar to the previous Figure, we compare ensemble properties in direct numerical simulations of the SWE and simulations of the ESN without (Figure 61) and with (Figure 62)

transfer learning. Comparison of Figures 61 and 62 indicates that momentum is less sensitive to the transfer learning. However, using transfer learning leads to significant improvement for the ensemble prediction for the momentum. Similar to the ensemble predictions for the water level, variance of the ensemble for the momentum (c.f. Figures 61(a) and 62(a) bottom snapshots for std.deviation at times $t = 40$ and $t = 60$) is affected strongly. Without transfer learning errors in the variance can reach up 25% at time $t = 60$ (Figure 61(a)). Therefore, transfer learning leads to a significant improvement to the ensemble predictions for the momentum as well.

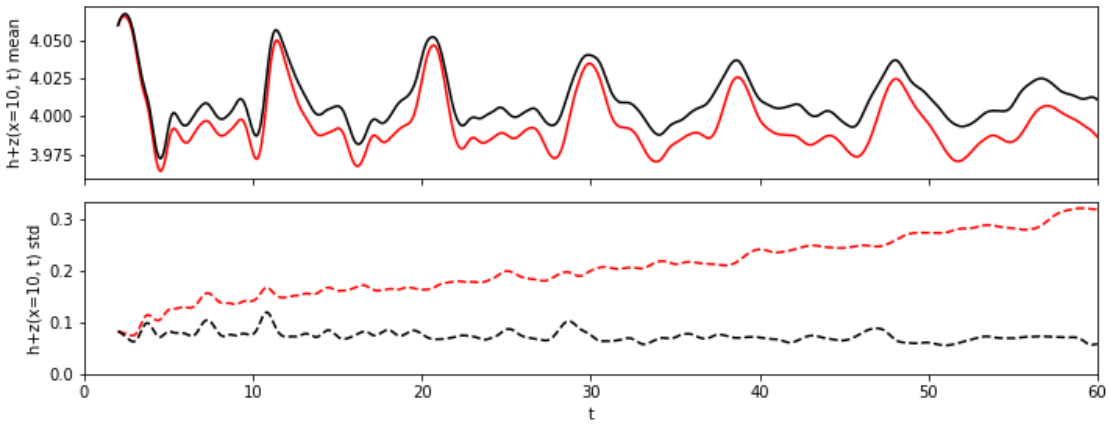
7.4.3 Conclusion

The results in this section indicate that

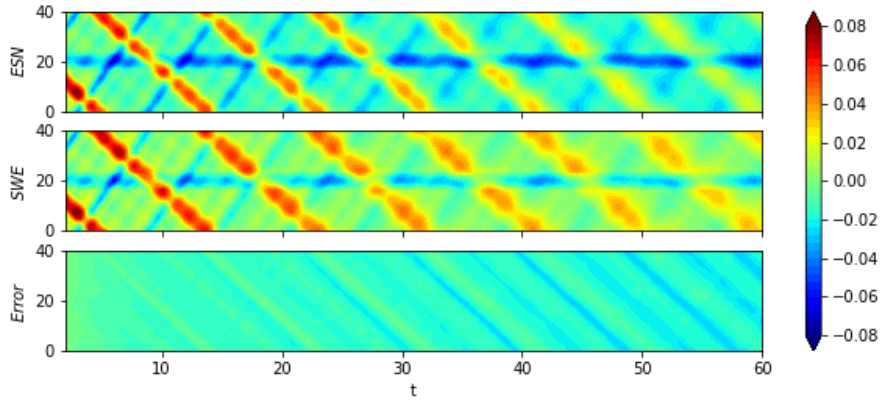
- Transfer learning is essential for predicting ensemble properties correctly; ESN with transfer learning is capable of predicting ensemble properties within a few percent,
- Predictions for ensemble properties of the water height are more challenging; errors for the ensemble properties of the water height can be larger than for the ensemble properties of the momentum,
- Even a short transfer learning on the interval $[0, 2]$ with sampling rate $\Delta = 0.1$ can significantly improve prediction properties of the ESN,
- ESN with transfer learning can significantly accelerate ensemble computations - generating a single trajectory with direct numerical simulations of the SWE takes approximately 21 minutes, and generating one trajectory using ESN with transfer learning takes approximately 1.2 minutes.



(a) Snapshots of the mean (top) and standard variance (bottom) for $t = 2.1$, 20, 40, 60

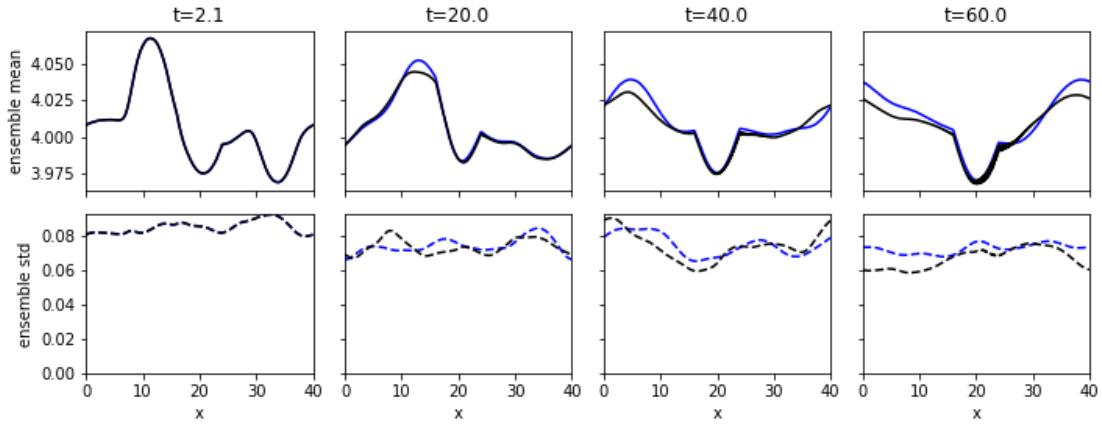


(b) Time-series for the mean (top) and standard variance (bottom) at $x = 10$

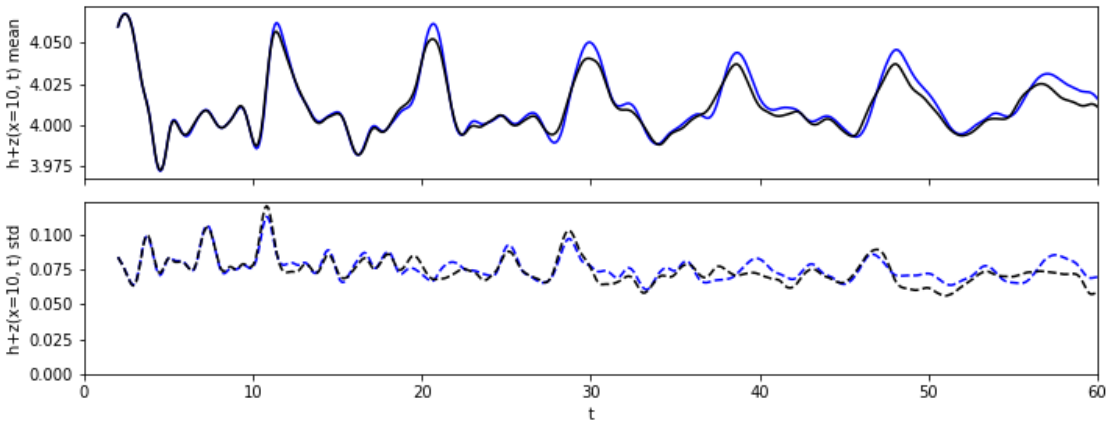


(c) Ensemble Mean $M(x,t) - 4.0$; Top - ESN, Middle - DNS SWE, Bottom - Abs. Error

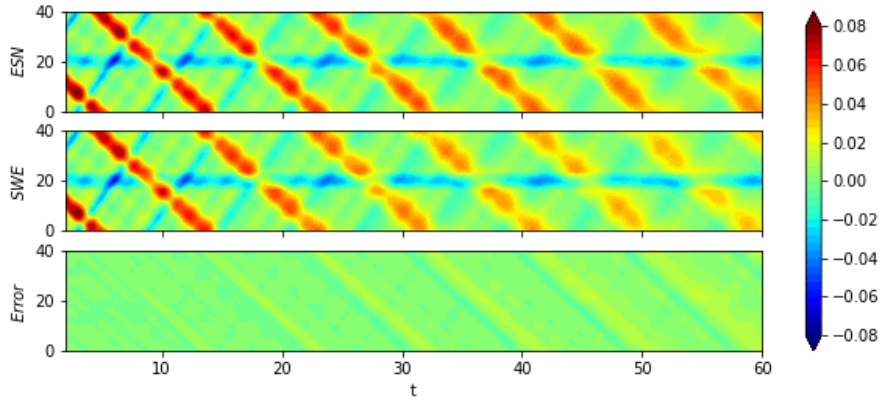
Figure 59: Second ensemble experiment using conventional ESN. Comparison of statistical properties of the ensemble for the total water level $h(x,t) + z(x)$ using direct numerical simulations of the SWE (black line) and predictions of the conventional ESN (red line).



(a) Snapshots of the mean (top) and standard variance (bottom) for $t = 2.1$, 20, 40, 60

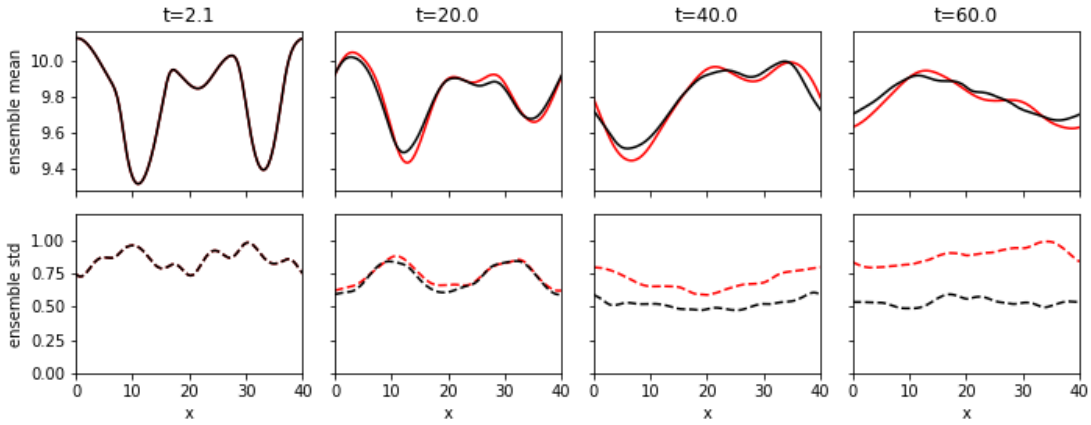


(b) Time-series for the mean (top) and standard variance (bottom) at $x = 10$

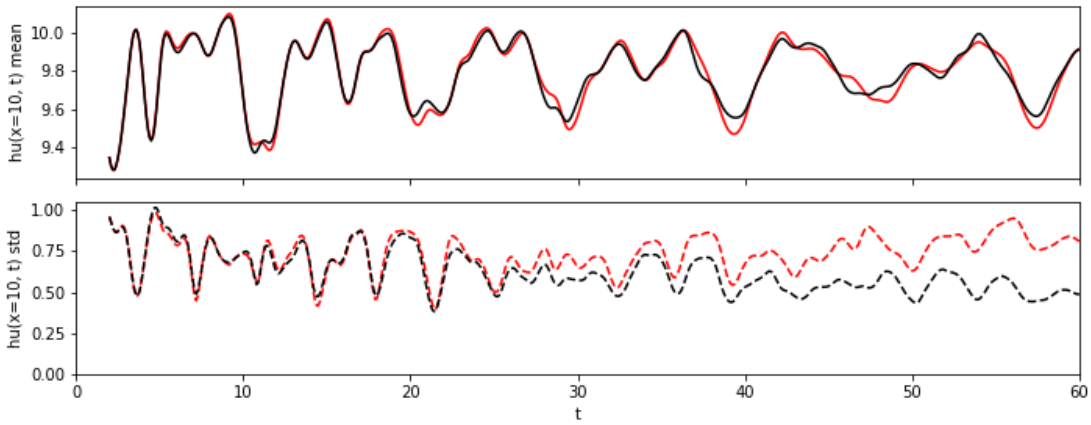


(c) Ensemble Mean $M(x,t) - 4.0$; Top - ESN, Middle - DNS SWE, Bottom - Abs. Error

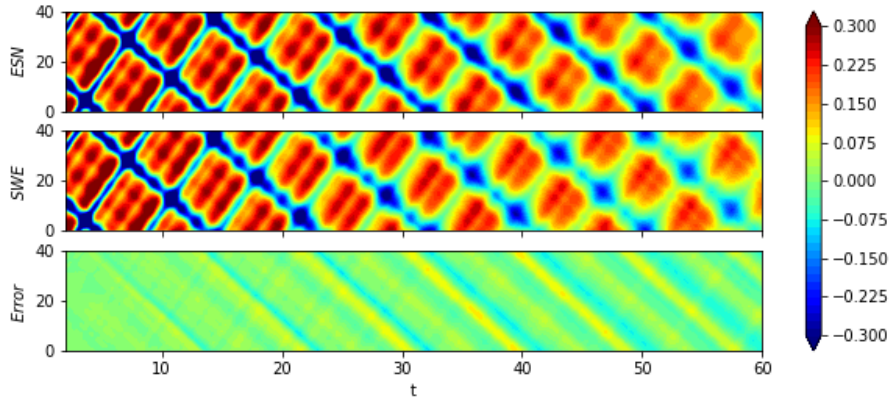
Figure 60: Second ensemble experiment using transferring ESN. Comparison of statistical properties of the ensemble for the total water level $h(x,t) + z(x)$ using direct numerical simulations of the SWE (black line) and predictions of the ESN with transfer learning (blue line).



(a) Snapshots of the mean (top) and standard variance (bottom) for $t = 2.1$, 20, 40, 60

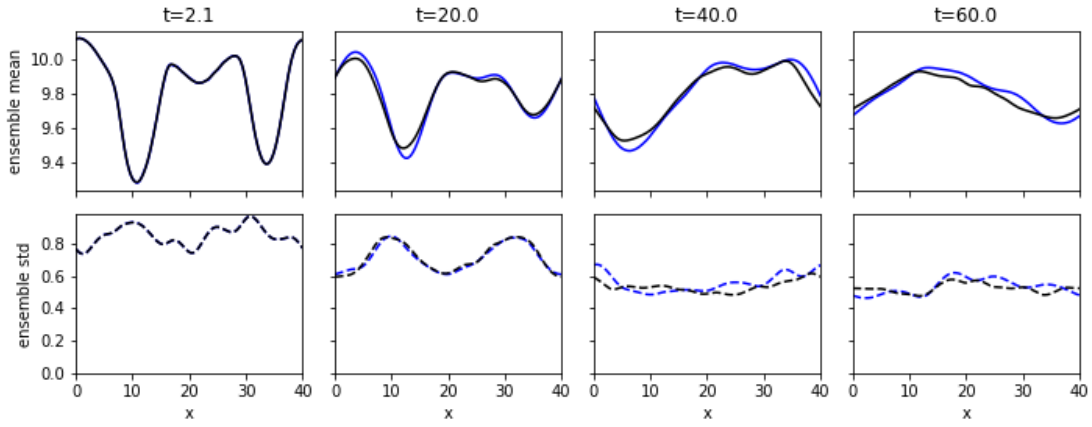


(b) Time-series for the mean (top) and standard variance (bottom) at $x = 10$

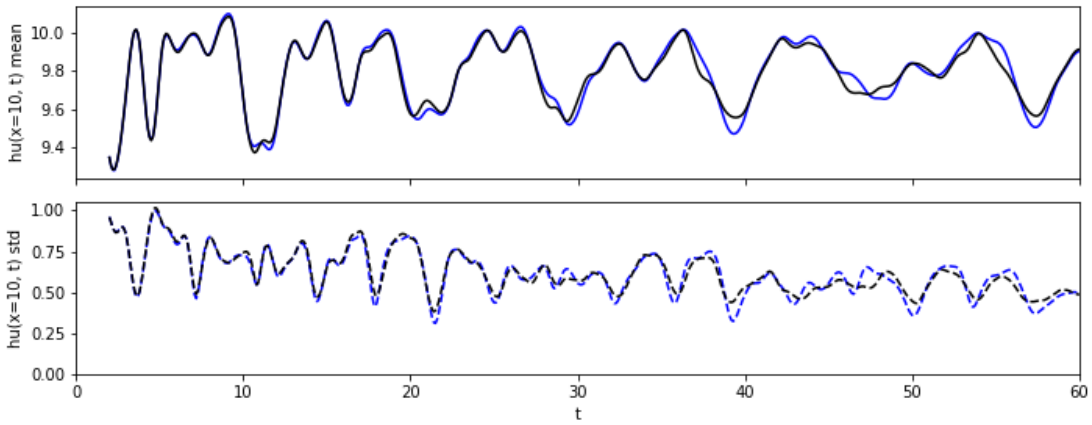


(c) Ensemble Mean $M(x,t) - 9.75$; Top - ESN, Middle - DNS SWE, Bottom - Abs. Error

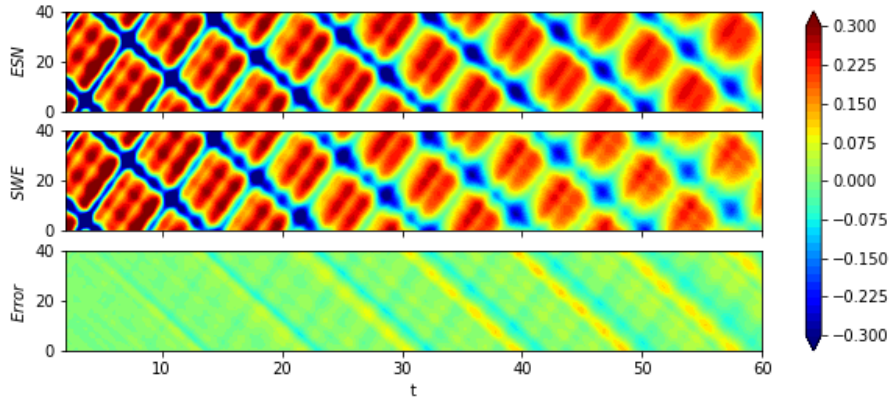
Figure 61: Second ensemble experiment using conventional ESN. Comparison of statistical properties of the ensemble for the momentum $h(x,t)u(x,t)$ using direct numerical simulations of the SWE (black line) and predictions of the conventional ESN (red line).



(a) Snapshots of the mean (top) and standard variance (bottom) for $t = 2.1$, 20, 40, 60



(b) Time-series for the mean (top) and standard variance (bottom) at $x = 10$



(c) Ensemble Mean $M(x,t) - 9.75$; Top - ESN, Middle - DNS SWE, Bottom - Abs. Error

Figure 62: Second ensemble experiment using transferring ESN. Comparison of statistical properties of the ensemble for the momentum $h(x,t)u(x,t)$ using direct numerical simulations of the SWE (black line) and predictions of the ESN with transfer learning (blue line).

7.5 Comparison with Polynomial Regression

In this section, we compare ESN predictions with results from nonlinear polynomial regression. The advantage of ESN is that the reservoir automatically incorporates nonlinear feedback of historical inputs. It has been demonstrated that neural networks can automatically select correct nonlinear features of the input signal. In contrast with the ESN, the nonlinearity has to be specified explicitly in regression methods, including polynomial regression [90, 70]. Recent research [70] suggest that polynomial regression model outperforms ESN in predicting classical Lorenz 63 and Lorenz 96 systems. Although Lorenz systems are chaotic and capture some properties of fluid dynamics, they are far simpler than shallow water models. In addition, Lorenz 63 and Lorenz 96 systems are quadratic, making it relatively simple to guess the form of the nonlinearity. In contrast with the Lorenz systems, SWE have a more complex nonlinearity, since SWE are typically integrated numerically in terms of conservative variables which are water height $h(x,t)$ and momentum $m(x,t) \equiv u(x,t)h(x,t)$ (5.1). Therefore, since one needs to explicitly compute velocity $u(x,t) = m(x,t)/h(x,t)$ to make a forward time step of the SWE equations written in these conservative variables, SWE equations have a nonlinearity h^{-1} in addition to polynomial terms. Thus, SWE equations present a nontrivial test for the performance of polynomial regression. We use training dataset discussed in chapter 6 and predict solutions on $[0, 10]$ with sub-sampling time step $\Delta = 0.01$.

Ridge Regression. Polynomial regression with regularization is a statistical nonlinear regression with either L^1 (LASSO) or L^2 (Ridge) regularization. Here we consider L^2 norm regularization and polynomial features of inputs. Assume that the polynomial expansion of inputs up to p degree, the Ridge Regularized Regression is defined as

$$W_{reg} = \underset{W_{reg}}{\operatorname{argmin}} ||W_{reg} g^{(p)}\left(X(t-1)\right) - X(t)||_2^2 + \lambda ||W_{reg}||_2^2, \tag{7.9}$$

$$X(t) = W_{reg} g^{(p)}\left(X(t-1)\right). \tag{7.10}$$

Here $||\cdot||_2$ is the L^2-norm of a vector and λ is the L^2 regularization parameter. The function

$g^{(p)}\left(X(t-1)\right)$ is a function that performs polynomial expansion of $X(t-1)$ up to a specified order p, such as $g^{(1)}\{x,y\} = \{1,x,y\}$ and $g^{(2)}\{x,y\} = \{1,x,y,x^2,y^2,xy\}$, etc. One can reduce the computational cost of polynomial regression by taking physics into account and considering only various cross-products of neighboring mesh points in the discretization (5.3), (5.4).

The main difference between the polynomial regression and ESN is that polynomial regression W_{reg} only uses an explicitly pre-defined polynomial feature of inputs $g^{(p)}$, instead of highly in-homogeneous reservoir r in ESN. If we consider all polynomial features up to order p, the number of features of $g^{(p)}$ is $\sum_{i=0}^{p} \frac{(2N_x+i-1)!}{i!(2N_x-1)!}$. Therefore, the final computational inference cost is $O\left(2N_x(1+2N_x+\ldots+(2N_x)^p)\right) = O\left((2N_x)^{p+1}\right)$, where N_x is the number of points in the space discretization of the SWE. The inference cost of ESN is $O(D^2)$, where D is the reservoir size.

The polynomial regression is computationally expensive for high-dimensional data since the nonlinearity is completely dense with respect to polynomial features, i.e., one has to consider all possible cross-products. In [70], the authors demonstrated that polynomial regression model with degree $p = 4$ outperforms ESN in predicting classical Lorenz 63 and Lorenz 96 systems. We would like to point out that the dimension of the Lorenz 63 model is 3, and the dimension of the Lorenz 96 model is 8 large-scale variables for regression.

Fully Dense Nonlinear Regression. Comparison between the polynomial regression with $p = 1$, $p = 2$ and the ESN is depicted in Figure Figure 63. The direct numerical simulation of the SWE yields solution with $N_x = 400, \Delta x = 0.1$. The numerical cost of the first order polynomial regression is $O(800^2)$ and the numerical cost of the second order polynomial regression is already up to $O(800^3)$. ESN with $D = 800$ reservoir (black dashed line) and linear regression (blue solid line) have the same inference cost $O(800^2)$. Increasing the degree of polynomial regression to 2 (orange solid line) results in the computational cost of $O(800^3)$. However, quadratic regression has difficultly in achieving accurate estimation of trajectories even with strong regularization $\lambda \in \{0.01, 0.1, 1\}$. Figure 63 demonstrates that ESN has a much better performance achieving much smaller errors. In addition, it is computationally less expensive to train ESN.

Sparse Nonlinear Regression with 5-point stencil. Here we take additional knowledge of

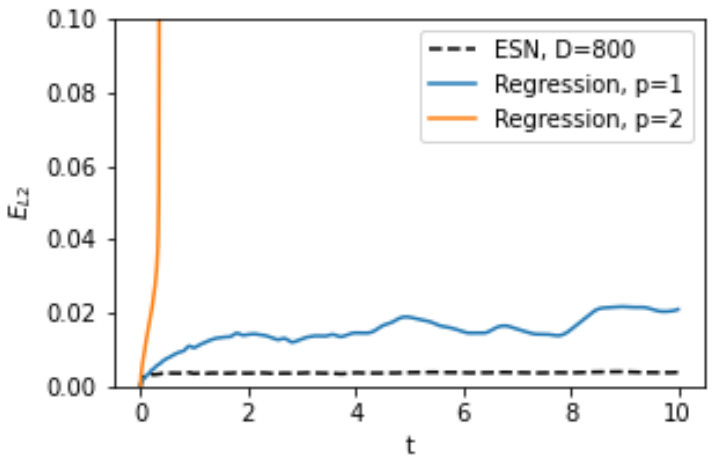


Figure 63: Averaged L^2 relative errors of individual solutions. Comparison between polynomial regression with fully dense nonlinear features with $p = 1$ (orange) and $p = 2$ (blue) and ESN (dashed black). Sampling of SWE solutions with $N_x = 400$, $\Delta x = 0.1$, $\Delta t = 0.01$. Penalty for polynomial regression $\lambda = 10^{-1}$. ESN setting: $D = 800$, $f = \tanh$, $\psi = \psi_1$, 100 training samples with 1% - 10% perturbation at time $t = [0, 10]$. All results averaged over the training dataset (100 samples).

shallow water equation into consideration to reduce the computational complexity of nonlinear regression. Also, reducing the number of parameters for nonlinear regression helps to avoid overfitting when the number of parameters is larger than the amount of data. In particular, since numerical solution of the SWE is obtained on an equipartitioned time-space grid, few neighboring variables at cells $i \pm 1$, $i \pm 2$, etc. are most relevant for predicting the time-evolution of variables in cell i. Thus, for each cell we consider a 5-point stencil for predicting the time-evolution of the water height and momentum, i.e., we use polynomial features of 10 most closet neighbors $\{h(x \pm 2\Delta x, t)$, $h(x \pm \Delta x, t)$, $h(x, t)$, $hu(x \pm 2\Delta x, t)$, $hu(x \pm \Delta x, t)$, $hu(x, t)\}$ for predicting $\{h(x, t + \Delta t)$, $hu(x, t + \Delta t)\}$. Results of the comparison between the sparse polynomial regression and ESN are depicted in Figure 64. The first order ($p = 1$) and second order ($p = 2$) of polynomial regression numerical cost is $O(800^2)$ and $O(6 \times 800^2)$, respectively. There is overfitting for cubic regression with $p = 2$; therefore we do not pursue $p > 2$. It is clear that ESN considerably outperforms polynomial regression.

Sparse Nonlinear Regression with 3-point stencil. Next, we use a 3-point stencil for predicting the time-evolution of the water height and momentum, i.e., we use polynomial features

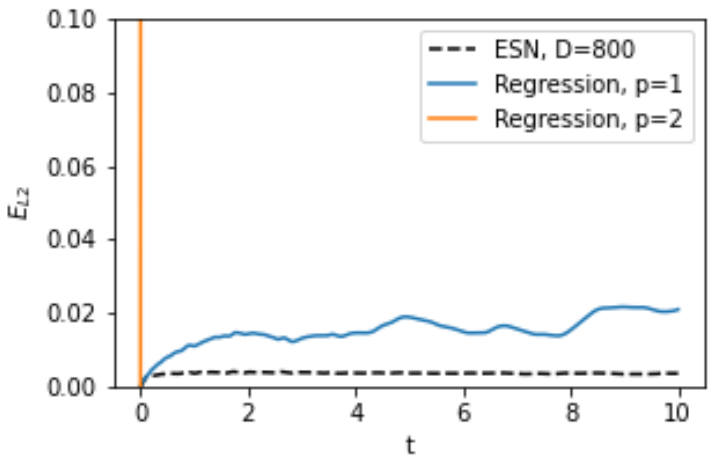


Figure 64: Averaged L^2 relative errors of individual solutions. Comparison between polynomial regression with sparse nonlinear features on 5-point stencil with $p = 1$ (blue), $p = 2$ (orange) and ESN (dashed black). Sampling of SWE solutions with $N_x = 400$, $\Delta x = 0.1$, $\Delta t = 0.01$. Penalty for polynomial regression $\lambda = 10^{-2}$. ESN setting: $D = 800$, $f = \tanh$, $\psi = \psi_1$, 100 training samples with 1% - 10% perturbation at time $t = [0, 10]$. All results averaged over the training dataset (100 samples).

of 6 most closet neighbors $\{h(x \pm \Delta x, t)$, $h(x, t)$, $hu(x \pm \Delta x, t)$, $hu(x, t)\}$ for predicting $\{h(x, t + \Delta t)$, $hu(x, t + \Delta t)\}$. Results of the comparison between the sparse polynomial regression with 3-point stencil and ESN are depicted in Figure 65. It is clear that ESN considerably outperforms polynomial regression in this case as well.

7.5.1 Conclusion

ESN considerably outperforms low-degree polynomial regression for predicting the time-evolution of solutions. As we pointed out early, the SWE has an unusual nonlinearity with $h(x, t)$ in the denominator, and it is possible that polynomial regression is unable to capture this form of nonlinearity. The advantage of the ESN is a quite rich internal dynamics of reservoir, because neurons in the reservoir are only loosely coupled by different activation signals. Overall, ESN seems to appear far superior compared to the traditional polynomial regression.

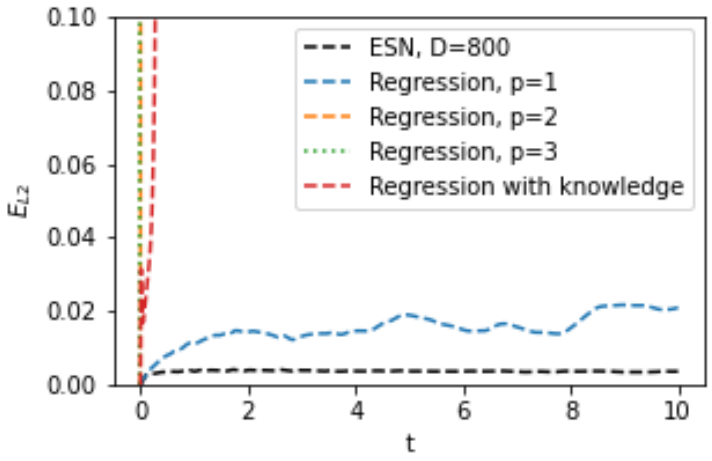


Figure 65: Averaged L^2 relative errors of individual solutions. Comparison between polynomial regression with sparse nonlinear features on 3-point stencil with $p = 1$ (blue), $p = 2$ (orange) $p = 3$ (green), knowledge about SWE(red) and ESN (dashed black). The regression with $p = 2$ (orange) and $p = 3$ (green) are overlapping. The knowledge (red) means regression with polynomial feature $\{h, hu, h^2, hu^2\}$ which are chosen from the SWE expression. Sampling of SWE solutions with $N_x = 400$, $\Delta x = 0.1$, $\Delta t = 0.01$. Penalty for polynomial regression $\lambda = 10^{-2}$. ESN setting: $D = 800$, $f = \tanh$, $\psi = \psi_1$, 100 training samples with 1% - 10% perturbation at time $t = [0, 10]$. All results averaged over the training dataset (100 samples).

7.6 Discussion

In this chapter, we discussed the application of the Echo State Network (ESN) to learning and predicting the dynamics of the shallow water equations (SWE). Although shallow water equations have a rather simple attractor in the regime considered here (everything converges to a "flat" solution), SWE has a complex transient regime with interacting nonlinear waves. This regime can probably be classified as decaying turbulence, since energy in high wave numbers decays to zero as time becomes large. However, the transient regime exhibits complex nonlinear behavior. Finite-time Lyapunov exponents can have positive values for short times, but as time goes towards infinity, all Lyapunov exponents become negative.

We apply the ESN formalism to learn the dynamics of the SWE. Our approach can also be interpreted as "equation learning", since ESN leans the appropriate dynamics which can approximate the dynamics of the SWE for a long time.

There are several conserved quantities for the SWE in the absence of the diffusive $(hu)_{xx}$ term - energy and momentum. In addition, SWE with or without the diffusive term also conserve the total mass. It turns out that these conserved quantities play a crucial role for predicting the dynamics of the SWE.

Our training dataset consists of solution trajectories with initial conditions generated as perturbations of the background state. All solutions at the initial time have the same total water height and momentum. We demonstrated in this chapter that the vanilla ESN has a high utility of prediction for solutions consistent with the training data, i.e., for solutions with the same initial water height and momentum. We would like to point out that as time goes towards infinity, the momentum decays to zero, but this is well-captured by the ESN dynamics. Therefore, we can conclude that ESN is able to learn the dynamics of the SWE very well in this particular regime. We also demonstrated that reservoir warm-up can be an efficient and computationally cheap approach for improving accuracy of predictions.

Next, we tried to utilize the same (vanilla) ESN for predicting trajectories which are not consistent with the training data, i.e., trajectories which initially have "shifted" total water height and/or

momentum. Vanilla ESN does a quite poor job predicting such trajectories. Therefore, we can also conclude that the dynamics of the SWE depends on the large-scale conserved quantities at the initial time. This might be an obvious observation in the context of analysis of partial differential equations, but here we also observe a confirmation of this fact in the context of machine learning. We also observed that ESN performs worse when the mean water height is "shifted", compared to the "shift" in the mean velocity. This is consistent with the dynamics, since the mean water height is a "true" conserved quantity and the final form of the attractor (fixed point) is determined by the initial mean water level. Therefore, the mean water level is very important for the overall short-time and long-time dynamics.

We successfully applied transfer learning to remedy the situation and predict trajectories with initial conditions which are inconsistent with the training dataset. This implies that the "learned dynamics" is probably sufficiently smooth (continuous and maybe even differentiable) with respect to conserved large-scale quantities of initial conditions. Therefore, ESN can quickly "discover" a slightly "shifted" dynamic equations which corresponds to a different mean total water level. It would be interesting if this conclusion can be tackled analytically. Recently, Dr. Patel developed an approach where he treat neural networks as an over-complete basis [70]. Thus, one can think of ESN as a high-dimensional function providing a smooth (with respect to parameters) approximation of the SWE dynamics.

Transfer learning provides an efficient approach for quickly adjusting the ESN to represent an appropriate model which is consistent with large-scale (conserved) quantities of the solution. We demonstrated that trajectories can be generated considerably faster (approximately 20 times) with the transfer learning approach, while retaining the accuracy of ESN predictions.

Finally, we also demonstrated that our transfer learning approach can be very efficient in simulating ensembles of trajectories. Thus, ML methodology developed here can significantly accelerate various ensemble-based methods. Such methods are highly relevant for several important practical problems, such as uncertainty quantification, Bayesian data assimilation, and inverse problems.

Bibliography

[1] Achatz, U., Dolaptchiev, S. I., and Timofeyev, I. Subgrid-scale closure for the inviscid burgers-hopf equation. *Communications in Mathematical Sciences 11*, 3 (2013), 757–777.

[2] Ait-Sahalia, Y., Mykland, P. A., and Zhang, L. How often to sample a continuous-time process in the presence of market microstructure noise. *The Review of Financial Studies 18*, 2 (2005), 351–416.

[3] Akiyama, T., and Tanaka, G. Echo state network with adversarial training. In *International Conference on Artificial Neural Networks* (2019), Springer, pp. 82–88.

[4] Antonelo, E. A., and Schrauwen, B. On learning navigation behaviors for small mobile robots with reservoir computing architectures. *IEEE Transactions on Neural Networks and Learning Systems 26*, 4 (2014), 763–780.

[5] Arcomano, T., Szunyogh, I., Pathak, J., Wikner, A., Hunt, B. R., and Ott, E. A machine learning-based global atmospheric forecast model. *Geophysical Research Letters 47*, 9 (2020), e2020GL087776.

[6] Azencott, R., Beri, A., Jain, A., and Timofeyev, I. Sub-sampling and parametric estimation for multiscale dynamics. *Communications in Mathematical Sciences 11*, 4 (2013), 939–970.

[7] Azencott, R., Beri, A., and Timofeyev, I. Parametric estimation of stationary stochastic processes under indirect observability. *Journal of Statistical Physics 144*, 1 (2011), 150–170.

[8] Azencott, R., Ren, P., and Timofeyev, I. Parametric estimation from approximate data: Non-gaussian diffusions. *Journal of Statistical Physics 161*, 5 (2015), 1276–1298.

[9] Azencott, R., Ren, P., and Timofeyev, I. Realised volatility and parametric estimation of heston sdes. *Finance and Stochastics 24* (2020), 723–755.

[10] Bacciu, D., Barsocchi, P., Chessa, S., Gallicchio, C., and Micheli, A. An experimental characterization of reservoir computing in ambient assisted living applications. *Neural Computing and Applications 24*, 6 (2014), 1451–1464.

[11] Bally, V., and Talay, D. The law of the euler scheme for stochastic differential equations. *Probability Theory and Related Fields 104*, 1 (1996), 43–60.

[12] Beskos, A., Roberts, G., Stuart, A., and Voss, J. Mcmc methods for diffusion bridges. *Stochastics and Dynamics 8*, 03 (2008), 319–350.

[13] Bolton, T., and Zanna, L. Applications of deep learning to ocean data inference and subgrid parameterization. *Journal of Advances in Modeling Earth Systems 11*, 1 (2019), 376–399.

[14] Brandt, M. W., and Santa-Clara, P. Simulated likelihood estimation of diffusions with an application to exchange rate dynamics in incomplete markets. *Journal of Financial Economics 63*, 2 (2002), 161–210.

[15] Chang, H.-H., Song, H., Yi, Y., Zhang, J., He, H., and Liu, L. Distributive dynamic spectrum access through deep reinforcement learning: A reservoir computing-based approach. *IEEE Internet of Things Journal 6*, 2 (2018), 1938–1948.

[16] Chattopadhyay, A., Hassanzadeh, P., and Subramanian, D. Data-driven predictions of a multiscale lorenz 96 chaotic system using machine-learning methods: reservoir computing, artificial neural network, and long short-term memory network. *Nonlinear Processes in Geophysics 27*, 3 (2020), 373–389.

[17] Crommelin, D., and Vanden-Eijnden, E. Diffusion estimation from multiscale data by operator eigenpairs. *Multiscale Modeling & Simulation 9*, 4 (2011), 1588–1623.

[18] Culina, J., Kravtsov, S., and Monahan, A. H. Stochastic parameterization schemes for use in realistic climate models. *Journal of the Atmospheric Sciences 68*, 2 (2011), 284–299.

[19] Dasgupta, S., Wörgötter, F., and Manoonpong, P. Information dynamics based self-adaptive reservoir for delay temporal memory tasks. *Evolving Systems 4*, 4 (2013), 235–249.

[20] De Gregorio, A., and Iacus, S. M. Adaptive lasso-type estimation for multivariate diffusion processes. *Econometric Theory 28*, 4 (2012), 838–860.

[21] Demaeyer, J., and Vannitsem, S. Comparison of stochastic parameterizations in the framework of a coupled ocean–atmosphere model. *Nonlinear Processes in Geophysics 25*, 3 (2018), 605–631.

[22] Dolaptchiev, S. I., Achatz, U., and Timofeyev, I. Stochastic closure for local averages in the finite-difference discretization of the forced burgers equation. *Theoretical and Computational Fluid Dynamics 27*, 3 (2013), 297–317.

[23] Duraisamy, K., Iaccarino, G., and Xiao, H. Turbulence modeling in the age of data. *Annual Review of Fluid Mechanics 51* (2019), 357–377.

[24] Durham, G. B., and Gallant, A. R. Numerical techniques for maximum likelihood estimation of continuous-time diffusion processes. *Journal of Business & Economic Statistics 20*, 3 (2002), 297–338.

[25] Fan, J., and Li, R. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association 96*, 456 (2001), 1348–1360.

[26] Fjordholm, U. S., Mishra, S., and Tadmor, E. Well-balanced and energy stable schemes for the shallow water equations with discontinuous topography. *Journal of Computational Physics 230*, 14 (2011), 5587–5609.

[27] Franco, G., Cerina, L., Gallicchio, C., Micheli, A., and Santambrogio, M. D. Continuous blood pressure estimation through optimized echo state networks. In *International Conference on Artificial Neural Networks* (2019), Springer, pp. 48–61.

[28] Franzke, C., and Majda, A. J. Low-order stochastic mode reduction for a prototype atmospheric gcm. *Journal of the Atmospheric Sciences 63*, 2 (2006), 457–479.

[29] Franzke, C., Majda, A. J., and Vanden-Eijnden, E. Low-order stochastic mode reduction for a realistic barotropic model climate. *Journal of the Atmospheric Sciences 62*, 6 (2005), 1722–1745.

[30] Gagne, D. J., Christensen, H. M., Subramanian, A. C., and Monahan, A. H. Machine learning for stochastic parameterization: Generative adversarial networks in the lorenz'96 model. *Journal of Advances in Modeling Earth Systems 12*, 3 (2020), e2019MS001896.

[31] Gallicchio, C., and Micheli, A. A reservoir computing approach for human gesture recognition from kinect data. In *AI* AAL@ AI* IA* (2016), Citeseer, pp. 33–42.

[32] Gauthier, D. J. Reservoir computing: Harnessing a universal dynamical system. *Physics Review Letter 120*, 2018 (2018), 024102.

[33] Genon-Catalot, V., and Jacod, J. On the estimation of the diffusion coefficient for multi-dimensional diffusion processes. In *Annales de l'IHP Probabilités et Statistiques* (1993), vol. 29, pp. 119–151.

[34] Gentine, P., Pritchard, M., Rasp, S., Reinaudi, G., and Yacalis, G. Could machine learning break the convection parameterization deadlock? *Geophysical Research Letters 45*, 11 (2018), 5742–5751.

[35] Golub, G. H., Hansen, P. C., and O'Leary, D. P. Tikhonov regularization and total least squares. *SIAM Journal on Matrix Analysis and Applications 21*, 1 (1999), 185–194.

[36] Goutal, N. *Proceedings of the 2nd workshop on dam-break wave simulation*. Department Laboratoire National d'Hydraulique, Groupe Hydraulique Fluviale, 1997.

[37] Han, S. I., and Lee, J. M. Precise positioning of nonsmooth dynamic systems using fuzzy wavelet echo state networks and dynamic surface sliding mode control. *IEEE Transactions on Industrial Electronics 60*, 11 (2012), 5124–5136.

[38] Hinaut, X., Petit, M., Pointeau, G., and Dominey, P. F. Exploring the acquisition and production of grammatical constructions through human-robot interaction with echo state networks. *Frontiers in Neurorobotics 8* (2014), 16.

[39] Inubushi, M., and Goto, S. Transferring reservoir computing: Formulation and application to fluid physics. In *International Conference on Artificial Neural Networks* (2019), Springer, pp. 193–199.

[40] Inubushi, M., and Goto, S. Transfer learning for nonlinear dynamics and its application to fluid turbulence. *Physical Review E 102*, 4 (2020), 043301.

[41] Jaeger, H. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report 148*, 34 (2001), 13.

[42] Jaeger, H. Echo state network. *Scholarpedia 2*, 9 (2007), 2330.

[43] Jaeger, H., and Haas, H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science 304*, 5667 (2004), 78–80.

[44] Jaeger, H., Lukoševičius, M., Popovici, D., and Siewert, U. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks 20*, 3 (2007), 335–352.

[45] Jain, A., Timofeyev, I., and Vanden-Eijnden, E. Stochastic mode-reduction in models with conservative fast sub-systems. *ArXiv Preprint ArXiv:1410.3004* (2014).

[46] Kalliadasis, S., Krumscheid, S., and Pavliotis, G. A. A new framework for extracting coarse-grained models from time series with multiscale structure. *Journal of Computational Physics 296* (2015), 314–328.

[47] Kessler, M. Estimation of an ergodic diffusion from discrete observations. *Scandinavian Journal of Statistics 24*, 2 (1997), 211–229.

[48] Khairoutdinov, M. F., and Randall, D. A. A cloud resolving model as a cloud parameterization in the ncar community climate system model: Preliminary results. *Geophysical Research Letters 28*, 18 (2001), 3617–3620.

[49] Klus, S., Nüske, F., Peitz, S., Niemann, J.-H., Clementi, C., and Schütte, C. Data-driven approximation of the koopman generator: Model reduction, system identification, and control. *Physica D: Nonlinear Phenomena 406* (2020), 132416.

[50] Kutz, J. N. Deep learning in fluid dynamics. *Journal of Fluid Mechanics 814* (2017), 1–4.

[51] Lin, M., Chen, R., and Mykland, P. On generating monte carlo samples of continuous diffusion bridges. *Journal of the American Statistical Association 105*, 490 (2010), 820–838.

[52] Lindström, E., and Höök, J. Unbiased adaptive lasso parameter estimation for diffusion processes. *IFAC-PapersOnLine 51*, 15 (2018), 257–262.

[53] Lu, Z., Pathak, J., Hunt, B., Girvan, M., Brockett, R., and Ott, E. Reservoir observers: Model-free inference of unmeasured variables in chaotic systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science 27*, 4 (2017), 041102.

[54] Lukoševičius, M., and Jaeger, H. Reservoir computing approaches to recurrent neural network training. *Computer Science Review 3*, 3 (2009), 127–149.

[55] Maass, W., Natschläger, T., and Markram, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation 14*, 11 (2002), 2531–2560.

[56] Majda, A., Abramov, R. V., and Grote, M. J. *Information theory and stochastics for multiscale nonlinear systems*, vol. 25. American Mathematical Soc., 2005.

[57] Majda, A., Timofeyev, I., and Vanden-Eijnden, E. A priori tests of a stochastic mode reduction strategy. *Physica D: Nonlinear Phenomena 170*, 3-4 (2002), 206–252.

[58] Majda, A. J., Timofeyev, I., and Vanden Eijnden, E. A mathematical framework for stochastic climate models. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences 54*, 8 (2001), 891–974.

[59] Majda, A. J., Timofeyev, I., and Vanden-Eijnden, E. Systematic strategies for stochastic mode reduction in climate. *Journal of the Atmospheric Sciences 60*, 14 (2003), 1705–1722.

[60] Mohan, A., Daniel, D., Chertkov, M., and Livescu, D. Compressed convolutional lstm: An efficient deep learning framework to model high fidelity 3d turbulence. *arXiv preprint arXiv:1903.00033* (2019).

[61] Nadiga, B. Reservoir computing as a tool for climate predictability studies. *Journal of Advances in Modeling Earth Systems* (2021), e2020MS002290.

[62] Nüske, F., Koltai, P., Boninsegna, L., and Clementi, C. Spectral properties of effective dynamics from conditional expectations. *Entropy 23*, 2 (2021), 134.

[63] Palem, K. V. Inexactness and a future of computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 372*, 2018 (2014), 20130281.

[64] Papavasiliou, A., Pavliotis, G., and Stuart, A. Maximum likelihood drift estimation for multiscale diffusions. *Stochastic Processes and their Applications 119*, 10 (2009), 3173–3210.

[65] Pathak, J., Hunt, B., Girvan, M., Lu, Z., and Ott, E. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical Review Letters 120*, 2 (2018), 024102.

[66] Pavliotis, G. A., and Stuart, A. Parameter estimation for multiscale diffusions. *Journal of Statistical Physics 127*, 4 (2007), 741–781.

[67] Pedersen, A. R. Consistency and asymptotic normality of an approximate maximum likelihood estimator for discretely observed diffusion processes. *Bernoulli* (1995), 257–279.

[68] Pedersen, A. R. A new approach to maximum likelihood estimation for stochastic differential equations based on discrete observations. *Scandinavian Journal of Statistics* (1995), 55–71.

[69] Popov, A., Koprinkova-Hristova, P., Simov, K., and Osenova, P. Echo state vs. lstm networks for word sense disambiguation. In *International Conference on Artificial Neural Networks* (2019), Springer, pp. 94–109.

[70] Pyle, R., Jovanovic, N., Subramanian, D., Palem, K. V., and Patel, A. B. Domain-driven models yield better predictions at lower cost than reservoir computers in lorenz systems. *Philosophical Transactions of the Royal Society A 379*, 2194 (2021), 20200246.

[71] Rasp, S., Pritchard, M. S., and Gentine, P. Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences 115*, 39 (2018), 9684–9689.

[72] Reichstein, M., Camps-Valls, G., Stevens, B., Jung, M., Denzler, J., Carvalhais, N., et al. Deep learning and process understanding for data-driven earth system science. *Nature 566*, 7743 (2019), 195–204.

[73] Scher, S., and Messori, G. Generalization properties of feed-forward neural networks trained on lorenz systems. *Nonlinear Processes in Geophysics 26*, 4 (2019), 381–399.

[74] Schumann, U. Subgrid scale model for finite difference simulations of turbulent flows in plane channels and annuli. *Journal of Computational Physics 18*, 4 (1975), 376–404.

[75] Sherstinsky, A. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena 404* (2020), 132306.

[76] Silver, D., Bakir, G., Bennett, K., Caruana, R., Pontil, M., Russell, S., and Tadepalli, P. Best of NIPS 2005: Highlights on the 'inductive transfer: 10 years later' workshop.

[77] Steil, J. J. Backpropagation-decorrelation: online recurrent learning with o (n) complexity. *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541) 2* (2004), 843–848.

[78] Stramer, O., and Yan, J. On simulated likelihood of discretely observed diffusion processes and comparison to closed-form approximation. *Journal of Computational and Graphical Statistics 16*, 3 (2007), 672–691.

[79] Tanaka, G., Yamane, T., Héroux, J. B., Nakane, R., Kanazawa, N., Takeda, S., Numata, H., Nakano, D., and Hirose, A. Recent advances in physical reservoir computing: A review. *Neural Networks 115* (2019), 100–123.

[80] Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological) 58*, 1 (1996), 267–288.

[81] Tietz, S., Jirak, D., and Wermter, S. A reservoir computing framework for continuous gesture recognition. In *International Conference on Artificial Neural Networks* (2019), Springer, pp. 7–18.

[82] Toms, B. A., Kashinath, K., Yang, D., et al. Deep learning for scientific inference from geophysical data: The madden-julian oscillation as a test case. *arXiv preprint arXiv:1902.04621* (2019).

[83] Tong, M. H., Bickett, A. D., Christiansen, E. M., and Cottrell, G. W. Learning grammatical structure with echo state networks. *Neural Networks 20*, 3 (2007), 424–432.

[84] Tong, Z., and Tanaka, G. Reservoir computing with untrained convolutional neural networks for image recognition. In *2018 24th International Conference on Pattern Recognition (ICPR)* (2018), IEEE, pp. 1289–1294.

[85] Vlachas, P. R., Byeon, W., Wan, Z. Y., Sapsis, T. P., and Koumoutsakos, P. 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 (2018), 20170844.

[86] Vlachas, P. R., Pathak, J., Hunt, B. R., Sapsis, T. P., Girvan, M., Ott, E., and Koumoutsakos, P. Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Networks 126* (2020), 191–217.

[87] Wang, H., and Leng, C. Unified lasso estimation by least squares approximation. *Journal of the American Statistical Association 102*, 479 (2007), 1039–1048.

[88] Wang, H., Li, G., and Jiang, G. Robust regression shrinkage and consistent variable selection through the lad-lasso. *Journal of Business & Economic Statistics 25*, 3 (2007), 347–355.

[89] Wang, J., Chmiela, S., Müller, K.-R., Noé, F., and Clementi, C. Ensemble learning of coarse-grained molecular dynamics force fields with a kernel approach. *The Journal of Chemical Physics 152*, 19 (2020), 194106.

[90] Wilks, D. S. Effects of stochastic parametrizations in the lorenz'96 system. *Quarterly Journal of the Royal Meteorological Society: A Journal of the Atmospheric Sciences, Applied Meteorology and Physical Oceanography 131*, 606 (2005), 389–407.

[91] Wu, J.-L., Kashinath, K., Albert, A., Chirila, D., Xiao, H., et al. Enforcing statistical constraints in generative adversarial networks for modeling chaotic dynamical systems. *Journal of Computational Physics 406* (2020), 109209.

[92] Xu, D., Lan, J., and Principe, J. C. Direct adaptive control: an echo state network and genetic algorithm approach. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* (2005), vol. 3, IEEE, pp. 1483–1486.

[93] Yildiz, I. B., Jaeger, H., and Kiebel, S. J. Re-visiting the echo state property. *Neural networks 35* (2012), 1–9.

[94] Yoshida, N. Estimation for diffusion processes from discrete observation. *Journal of Multivariate Analysis 41*, 2 (1992), 220–242.

[95] Yu, R., Zheng, S., Anandkumar, A., and Yue, Y. Long-term forecasting using tensor-train RNNs. *Arxiv* (2017).

[96] Zacharuk, M., Dolaptchiev, S. I., Achatz, U., and Timofeyev, I. Stochastic subgrid-scale parametrization for one-dimensional shallow-water dynamics using stochastic mode reduction. *Quarterly Journal of the Royal Meteorological Society 144*, 715 (2018), 1975–1990.

[97] Zhang, L., Mykland, P. A., and Aït-Sahalia, Y. A tale of two time scales: Determining integrated volatility with noisy high-frequency data. *Journal of the American Statistical Association 100*, 472 (2005), 1394–1411.

[98] Zhang, X., and Shu, C.-W. Maximum-principle-satisfying and positivity-preserving high-order schemes for conservation laws: survey and new developments. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 467*, 2134 (2011), 2752–2776.

[99] Zimmermann, R. S., and Parlitz, U. Observing spatio-temporal dynamics of excitable media using reservoir computing. *Chaos: An Interdisciplinary Journal of Nonlinear Science 28*, 4 (2018), 043118.

[100] Zou, H. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association 101*, 476 (2006), 1418–1429.