Reservoir Computing on Non-Linear Schrödinger Equation

Progress Report 27 January 2022

A. N. Hasmi[a,*],

[a] *Khalifa University, Abu Dhabi, UAE, 127788*

Abstract

Abstract, should normally be not longer than 200 words.

Keywords:

1. Introduction

The success of machine learning especially deep learning have prompted a new interest in scientific community to apply the machine learning method to solve challenges in physical problems modeled by partial differential equations involving spatiotemporal variables. Depending on the approaches used, there are several perks including deep learning/data-driven approaches to physical problems which includes : (1) ability to incorporate measurement data in the model predictions [10] (2) Speed up computations by modelling computation expensive task and using machine learning approach as closure model to describe missing dynamics [14].

These years, there are many approaches to apply machine learning to solve partial differential equations, a non-exhaustive approach to the methods includes Physics Informed Neural Network [17, 12] and Reservoir Computing [10, 4, 5, 7]. Reservoir Computing is a model-free approach, in the sense that the mathematical equations are not supplied as an input for the model. The strength of Reservoir computing is the learning process only need to be done for the output layer, simplifying the training process and computation.

This research tries to apply the reservoir computing approach to the non-linear Schrödinger equations. NLS is an ubiquitous equation in physics that models many phenomena likes optics, ocean waves with modulating amplitude, rogue waves etc. In particular, we would like to train the phenomena of the occurence of rogue waves using the NLS equation.

2. Reservoir Computing

Reservoir computing can be taught as a subset of Recurrent Neural Networks. The main principle in reservoir computing is the separation between internal states (reservoir) and the readout learning. As opposed to the backward propagation method which trains both the states and output parameters, the training in reservoir computing only applies to the readout, simplifying the learning while still capable to solve tasks (almost always) better compared with RNN [8, 2]. In this research, we follows the adjustment for spatialtemporal models [10], which split the spatial data into overlapping domains and train multiple reservoirs for each data chunk. The reasoning of splitting the data is that the exchange of information between two very far seperated spatial grid is not immediate, thus creating reservoir state modelling a connection between faraway points is not physical and also a waste of computation effort. Thus by partitioning the grid while maintaining exchange of information between reservoir, render the size of reservoir and the amount of training data becomes manageable.

*Corresponding author

Email addresses: 100060615@ku.ac.ae (A. N. Hasmi), abrari@lecturer.itk.ac.id (A. N. Hasmi)

To be precise let measurement data consist of spatiotemporal data $\mathbf{U} = (u_{ij})$, with temporal index $1 \leq i \leq N_\tau$, and spatial index $1 \leq j < N_\xi$. The periodic spatial grid was splitted into non-overlapping regions $\mathbf{s}^{(k)} = \left(s_{ij}^{(k)}\right)$, with $1 \leq j \leq n_p, 1 \leq k \leq M$ with n_p is the number of nodes at each partition and M is the number of partition. Remark that $M \times n_p = N_\xi$. Associated with each partition $\mathbf{s}^{(k)}$,there is an overlapping domain $\mathbf{v}^{(k)}$ with overlap buffer regions l at each most left and right nodes , thus $\mathbf{v}^{(k)}$ has n_p+2l spatial grid. Furthermore, the data is temporally split into training and testing set. The overlapping domain serve as input for the each reservoir enabling change of information between neighboring reservoir, while the non-overlapping domain will act as the output.

Following the formalism in [6], we divide the steps in reservoir computing into three steps: (1) listening, (2) training, (3) prediction.

The listening process correspond to constructing reservoir and evolve them based on input and previous states. Specifically let $r_i^{(k)} \in \mathbb{R}^M$ with $M \gg n_p + 2l$ denote the state of reservoir k at time index i. The reservoir dynamics is given by:

$$r_{i+1}^{(k)} = \tanh\left(W^{(k)} r_i^{(k)} + W_{in}^{(k)} v_i^{(k)}\right) \tag{2.1}$$

The matrix $W^{(k)}$ is an adjecency matrix which usually set to ensure "Echo States Property"[5]. In a simplified term, this condition express that the effect of previous state and previous input at future state should vanish gradually at future time. Practically, even though not always neccessary but the echo state property property is satisfied by scaling $W^{(k)}$, such that $W^{(k)}$ are contractive i.e. the spectral radius of matrix $W^{(k)}$ given by ρ should satisfies $\rho < 1$. Furthermore, the adjecency matrix is set to be sparsely connected and random to ensure that the reservoir is rich enough for learning phase. The input coupling matrix $W_{in}^{(k)}$ is a sparse matrix whose entries magnitude not more than σ. Furthermore, the approximate number of non-zero element at each row is fixed.

The learning phase involve in finding output matrix $W_{out}^{(k)}$ such that the readout defined by:

$$y_i^{(k)} = W_{out}^{(k)} \hat{r}_i^{(k)} \tag{2.2}$$

minimized a cost function. The modified state $\hat{r}_i^{(k)}$ is a nonlinear modification aimed to futher enhance the nonlinearity feature of the reservoirChattopadhyay et al. [2], Pathak et al. [10]. Let $r_{ij}^{(k)}$ is the j-th element of vector $r_{ij}^{(k)}$, then the j-th element of vector $\hat{r}_{ij}^{(k)}$ is defined as:

$$\hat{r}_{ij}^{(k)} = \begin{cases} r_{ij}^{(k)} & j \text{ is odd} \\ \left(r_{ij}^{(k)}\right)^2 & j \text{ is even} \end{cases} \tag{2.3}$$

The usual cost function is:

$$E^{(k)} = \min_{W_{out}^k} \sum_{i=1}^{N_{train}} \left\| W_{out}^{(k)} \hat{r}_i^{(k)} - s_i^{(k)} \right\|_2^2 + \beta \left\| W_{out}^{(k)} \right\|^2 \tag{2.4}$$

In essential, the training can be done by using any method to solve linear least square problem. By defining the $\mathbf{R}^{(k)}$ and $\mathbf{S}^{(k)}$ as a matrix whose i-th column is vector $\hat{r}_i^{(k)}$ and $s_i^{(k)}$ respectively, then the training process is equivalent to solve the following matrix equation:

$$\left(\mathbf{R}^{(k)} \mathbf{R}^{(k)*} + \beta \mathbf{I}\right) W^{(k)*} = \mathbf{R}^{(k)} \mathbf{S}^*$$

with notation $\mathbf{R}^*$ denotes the conjugate transpose of matrix $\mathbf{R}$, and $\mathbf{I}$ is the identity matrix with suitable dimension.

Parameter	Explanation	Value
ρ	Spectral Radius of W	0.6
σ	maximum elementwise magnitude of W_{in}	0.1
β	Regularization Parameter	10^{-4}
M	Number of reservoirs	20
l	Number of overlap grid	2
D	Node in reservoir	5000

Table 1: Hyperparameter for Lorenz 96

During the prediction phase, the input vector $v_i^{(k)}$ in equation 2.1 is replaced by the recent available prediction from the readout (equation 2.2) using the output matrix obtained during the training phase. However, as the vector $v_i^{(k)}$ consist of overlapping domain, then the output should be constructed by combining information from the neighbouring reservoir $y_i^{(k-1)}, y_i^{(k)} y_i^{(k+1)}$. Calling this prediction as $\tilde{v}_i^{(k)}$, then the steps in prediction phase are:

1. Compute $y_i^{(k)} = W_{out}^k \hat{r}_i^{(k)}$
2. Construct $\tilde{v}_i^{(k)}$ based on $y_i^{(k-1)}, y_i^{(k)} y_i^{(k+1)}$
3. Update the reservoir states: $r_{i+1}^{(k)} = \tanh\left(W^{(k)} r_i^{(k)} + W_{in}^{(k)} \tilde{v}_i^{(k)}\right)$

The implementation of the reservoir computing is based on the code Martinuzzi [9] which provide implementations of the reservoir computing including the non-linear transformation.

3. Result and Discussion

3.1. Verification: Lorenz 96 Equation

The Lorentz 96 system is a coupled ordinary differential euqation which is used as initial model for atmospheric dynamics or testing a machine learning algorithm for spatiotemporal chaotic system Pathak and Ott [11], Chattopadhyay et al. [2]. The model is defined as coupled ordinary differential equations:

$$\frac{dX_j}{dt} = -X_j + X_{j-1}X_{j+1} - X_{j-1}X_{j-2} + F$$

with the spatial variables X_j, $1 \leq j \leq N$,$X_{j+N} = X_j$.

The training data is generated by using 4th order Runge Kutta time integrator with fixed time step $\Delta t = 0.01$ and choosing $F = 8$. The number of spatial grid N is 40 with the initial condition at every grid being a random number between 0 and 1. The largest Lyapunov exponent for this system is $\Lambda = 1.4$. The number of training data is N_{train}= 80000. The hyperparameter used in reservoir computing is given in Table 1 which are quite similar with the hyperparameter in similar problem used in Pathak and Ott [11]. The spatially averaged error at time index i is given by:

$$e_i = \frac{\|\tilde{u}_i - u_i\|}{\|u_i\|}$$

Figure 3.1 shows a comparison between the test data and the prediction by Reservoir Computing. The second panel shows the prediction by reservoir computing, while the third panel shows the difference between the prediction and the "real" test data (shown in first panel). The fourth panel plots the error of the solutions over time. Comparing our result with the result presented in Pathak and Ott [11](Figure 7) , reveals that our result lies between the realizable RMS error in the computation. A more robust comparison can be achieved by doing predictions starting from different states of attractor, which might be our future work.

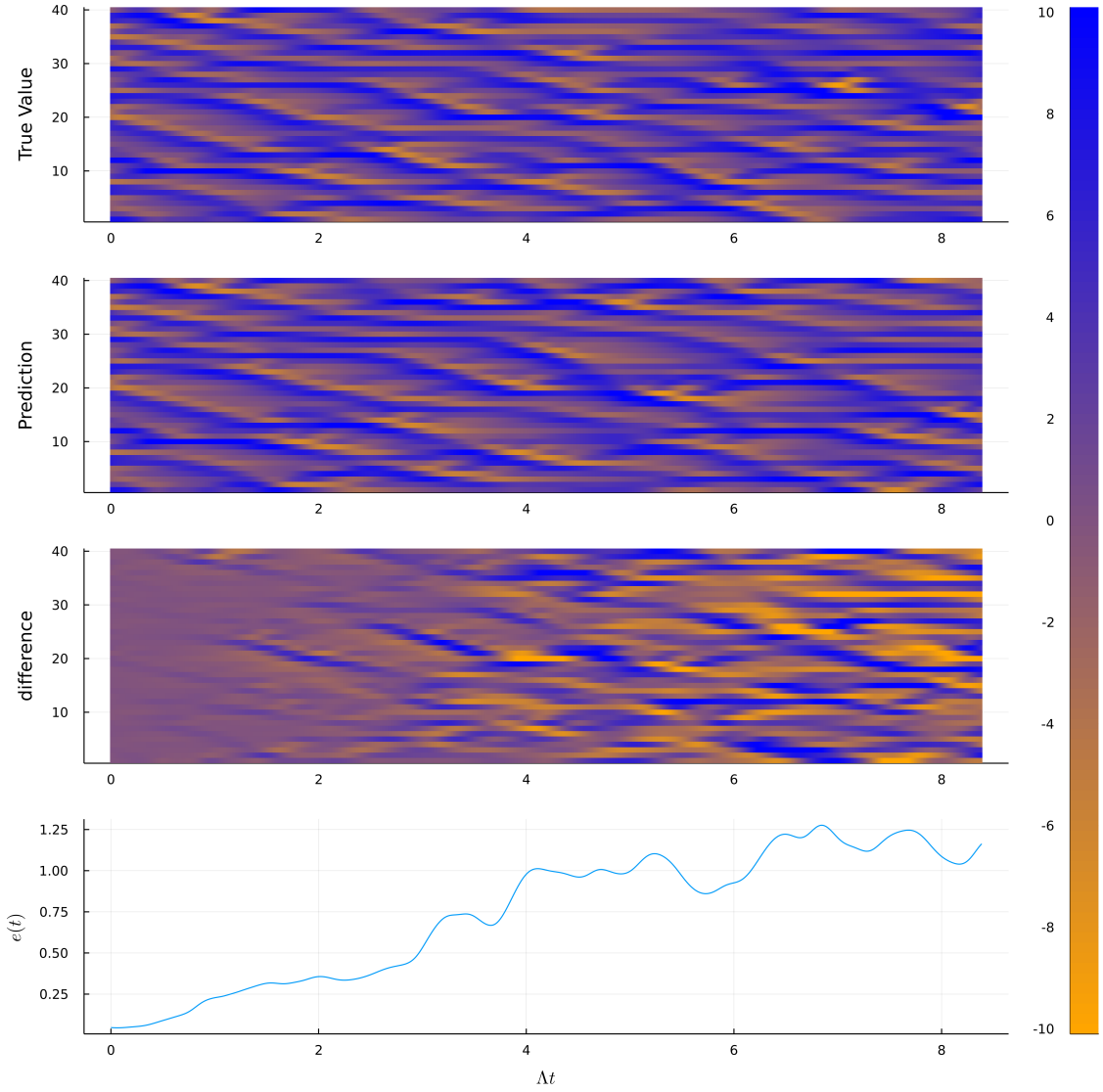

Figure 3.1: Prediction of Lorentz 96

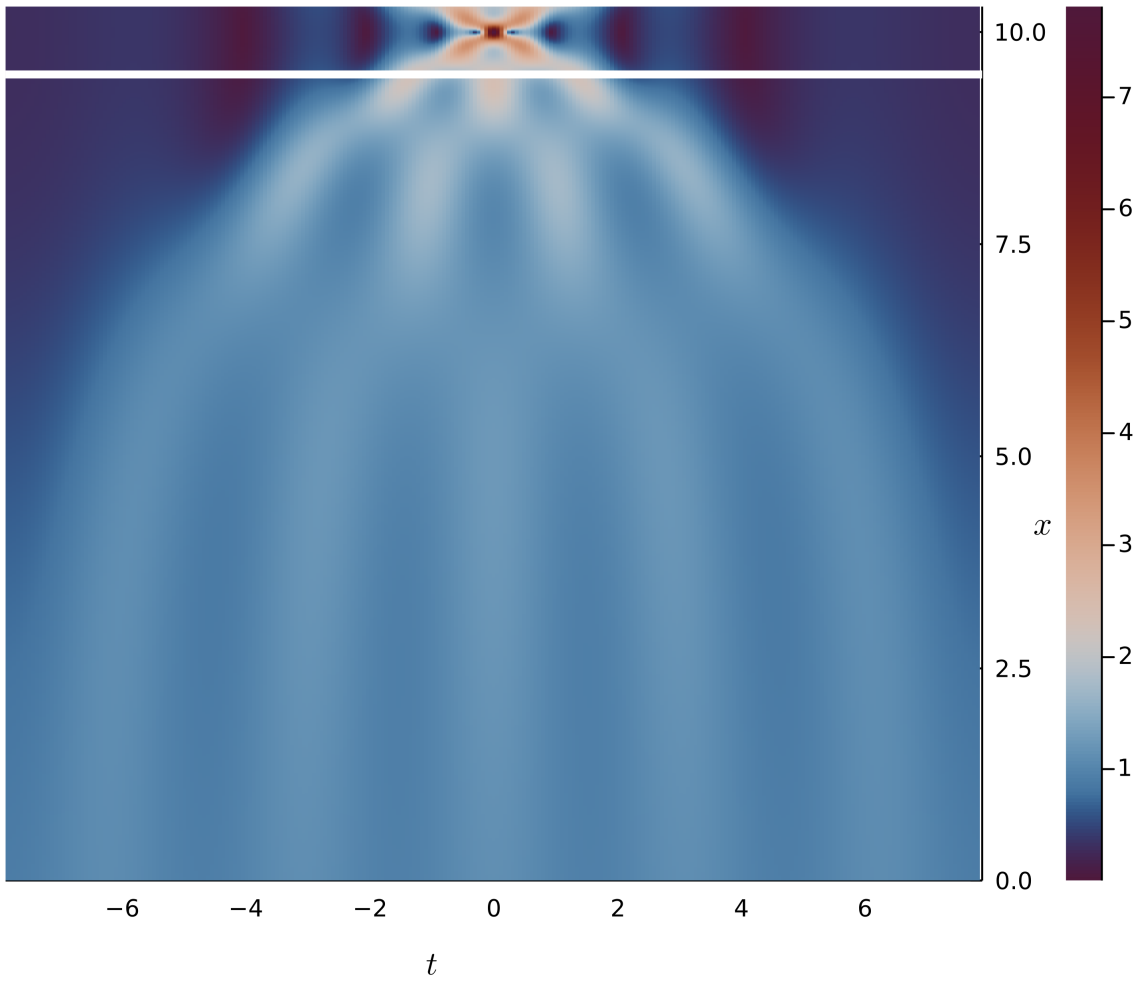

Figure 3.2: Fifth order breather simulation by NLS

3.2. Non-Linear Schrödinger

We train reservoir to learn the solution of NLS equation given by:

$$i\frac{\partial\psi}{\partial\tau} + \frac{1}{2}\frac{\partial^2\psi}{\partial\xi^2} + |\psi|^2\psi = 0$$

with the amplitude envelope $\psi(\xi, \tau)$ is a function of propagation distance ξ and co-moving time τ. We are aimed at simulating the construction of 5^{th} order breather on a uniform background. This test case represents an important modulation instability which is modelling the formation of a rogue wave. The objective of this simulation is to investigate under what conditions the Reservoir Computing can learn and simulate the maximum peak of a breather in NLS. For that purpose, we intentionally hide the peak of thr breather from the training data and preserve it as a part of test set.

The training and test data is generated by using 6th order split-step method Yoshida [16] implemented in Julia library the NonlinearSchrodinger.jl [1]. A brief explanation of the split-step method is given in Appendix. The initial condition was prepared according to Chin et al. [3] which generate the initial condition for maximum intensity soliton via Darboux Transformation. The modulation parameter used in generating the training set is $a = 0.4802$ in which there exists a unique maximum intensity soliton up to fifth order. The length of the spatial domain is one period $[-L/2, L/2]$ with $L = \pi/\sqrt{1-2a}$ and 256 spatial nodes. The simulation was started from $t = 0$, with $\Delta t = 0.01$ and the peaks was achieved after

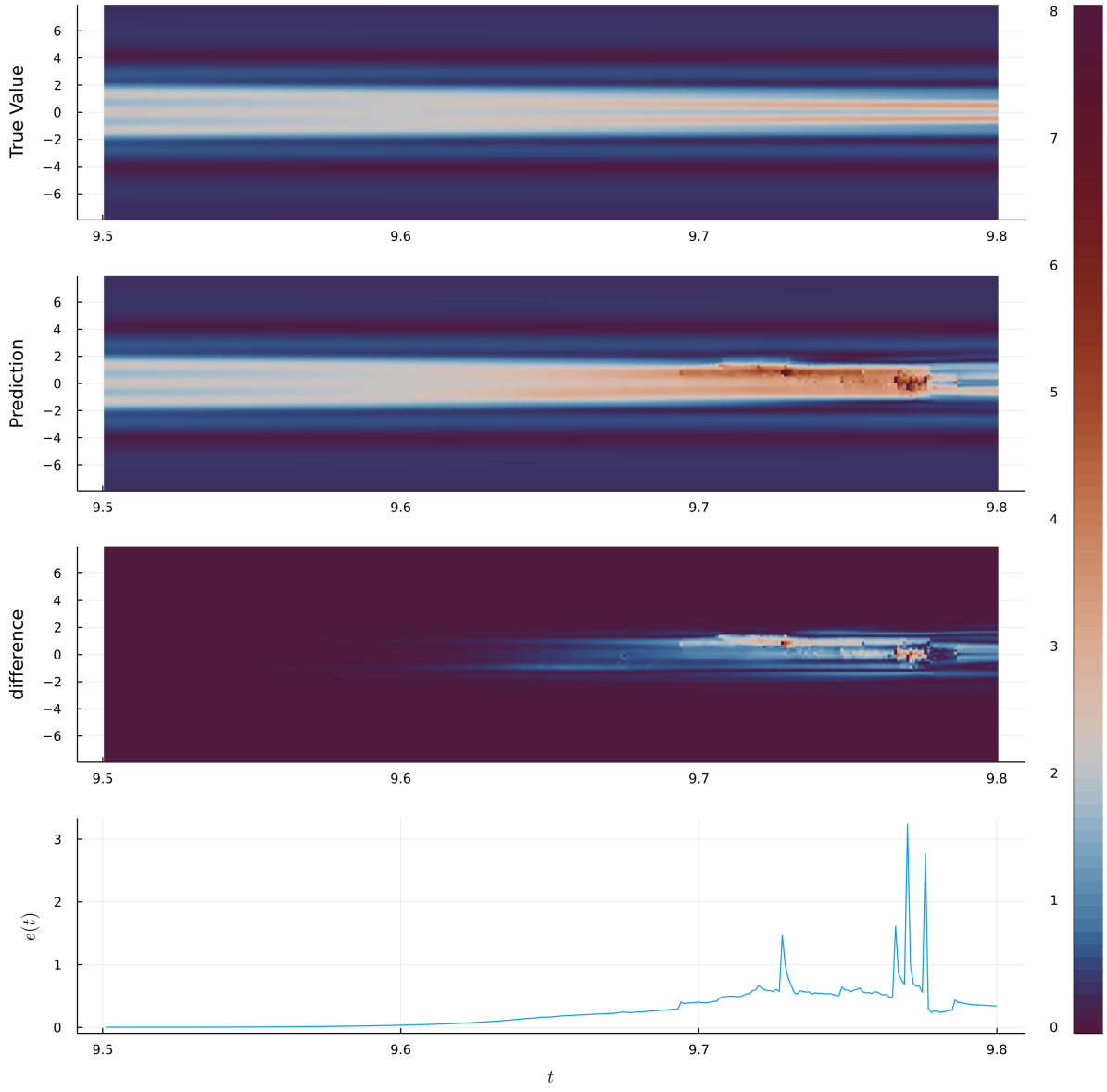

Figure 3.3: NLS simulation

$t = 10$. Figure 3.2 shows the training and the test set. The white line separates between the training and test set at $t = 9.5$.

The hyperparameters used in the prediction of NLS is given in Table 2 which are decided in reference to the simulation of Kuramoto-Sivashinsky equations using RC [11].

Figure 3.3 depict a result of our Reservoir Computing Simulation. The prediction yields an accurate result up to 0.1 seconds from the start of the prediction phase. Following that we observe gradual overestimation of $|\psi|$ until "breathers" were generated at $t = 9.728$,9.666, 9.77 which are ahead of the expected true breather. The magnitude of RC prediction resulted in an overestimation of the breather peak more than two and six times the expected height which can be observed in detail in 3.4 . In the figure, we also presented a profile of the magnitude of RC prediction one time step after the peak breather.

The error accumulates and beginning at $t = 9.6$s the error is too significant and yield a totally wrong prediction. The unreliability of the result after $t = 9.6s$ was also observed in calculation of norm error and energy error in the RC result shown in 3.5. The norm error and energy error was defined as the differences between norm and energy at particular times with the norm and energy at the beginning of the prediction

Parameter	Explanation	Value
ρ	Spectral Radius of W	0.6
σ	maximum elementwise magnitude of W_{in}	0.1
β	Regularization Parameter	10^{-4}
M	Number of reservoirs	32
l	Number of overlap grid	6
D	Nodes in reservoir	4000

Table 2: Hyperparameter for Nonlinear Schrödinger

stage.

3.3. Future Work

The verification stage by replicating the simulation of the Lorenz 96 actuator gives confidence in the code for Reservoir Computing. To obtain a robust estimation of the accuracy of the RC prediction, one can run the prediction stage over several intervals as done in Pathak and Ott [11], Chattopadhyay et al. [2].

The prediction of Reservoir Computing to the simulation of the NLS equation yield a good prediction until $t = 0.1$. Several adjustments of hyperparameter is a future research direction including but not limited to shifting the start of the prediction stage, increasing the number of training data, varying the spectral radius, number of reservoirs, regularization parameter etc. Furthermore, we will try to change the non-linearity transformation as done in Chattopadhyay et al. [2] for instance by changing the equation 2.3 by cubic non-linearity which is the nonlinearity presents in the NLS equation.

From the computational point of view, we would like to improve our existing Julia code by (1) Harnessing the parallel computation especially as we use multiply reservoirs which are highly parallel to each other (2) Test whether exploiting the sparseness of adjacent matrix W and input to reservoir coupling matrix W_{in} will improve the computational time.

4. Conclusion

The prediction of reservoir computing for solving 5th order breather of NLS with a hyperparameter shows that the (provisional) prediction time is about 0.1 seconds. Further research needs to be done by varying the hyperparameter used in reservoir computing and closely observing the impact of these to the prediction result.

Appendix A. NLS Training Data Generation

The Nonlinear Schrodinger equation was solved using 6th order split-step method [15]. The superiority of split-step method compared to variants of finite difference methods to discretize the nonlinear Schrödinger equation was investigated in [13]. The split-step method considers the NLS equations consist of two differential operators:

$$\mathcal{N} = |\psi|^2 \psi \qquad \mathcal{D} = \frac{1}{2}\frac{\partial^2 \psi}{\partial \xi^2}$$

The NLS can be rewritten as the following:

$$\frac{\partial \psi}{\partial \tau} = i\mathcal{D} + i\mathcal{N} \tag{A.1}$$

Let $e^{i\mathcal{D}\tau}\psi_0$ the exact solution of the initial value problem:

$$\frac{\partial \psi}{\partial \tau} = i\mathcal{D}$$

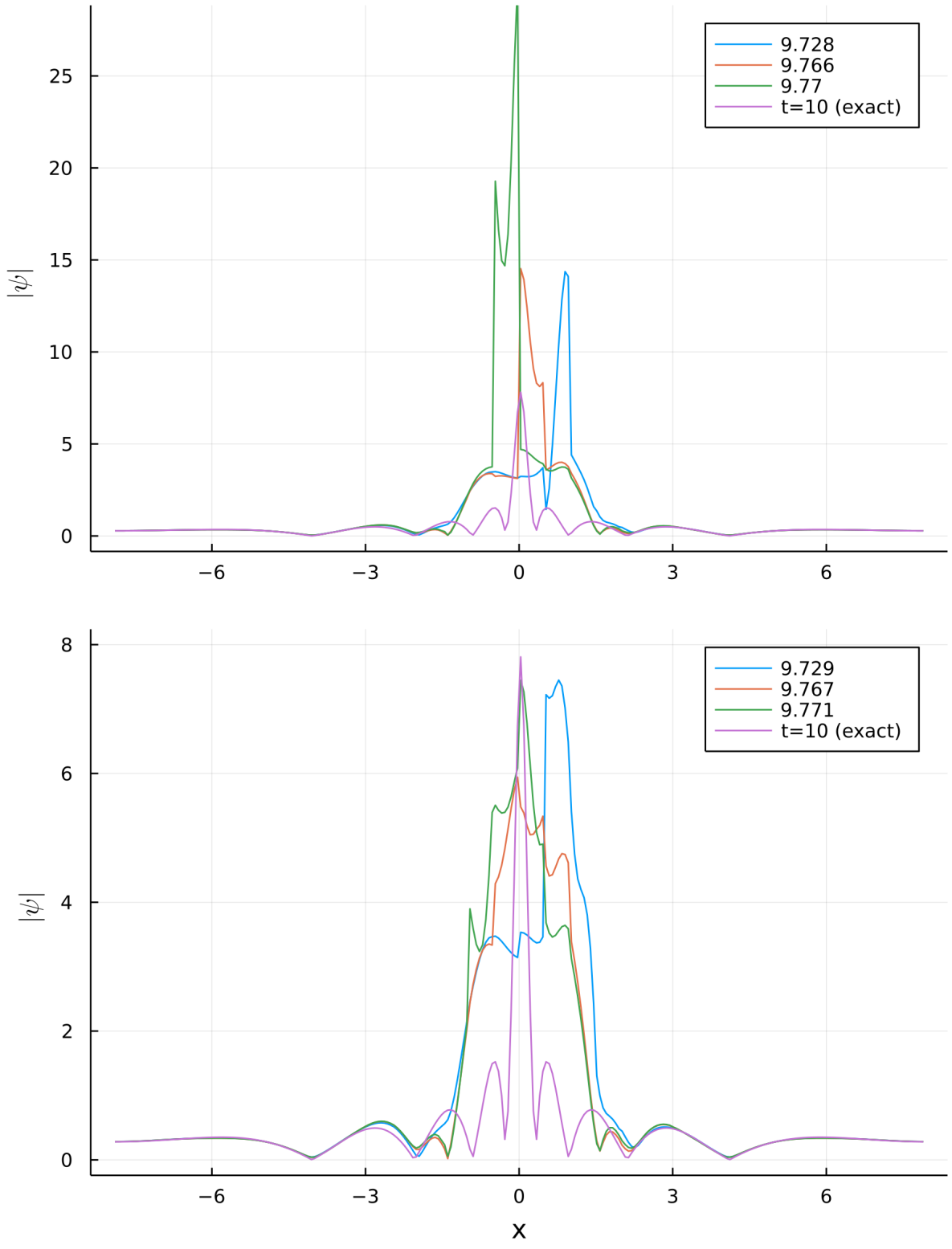

Figure 3.4: Comparison of magnitude at different time prediction

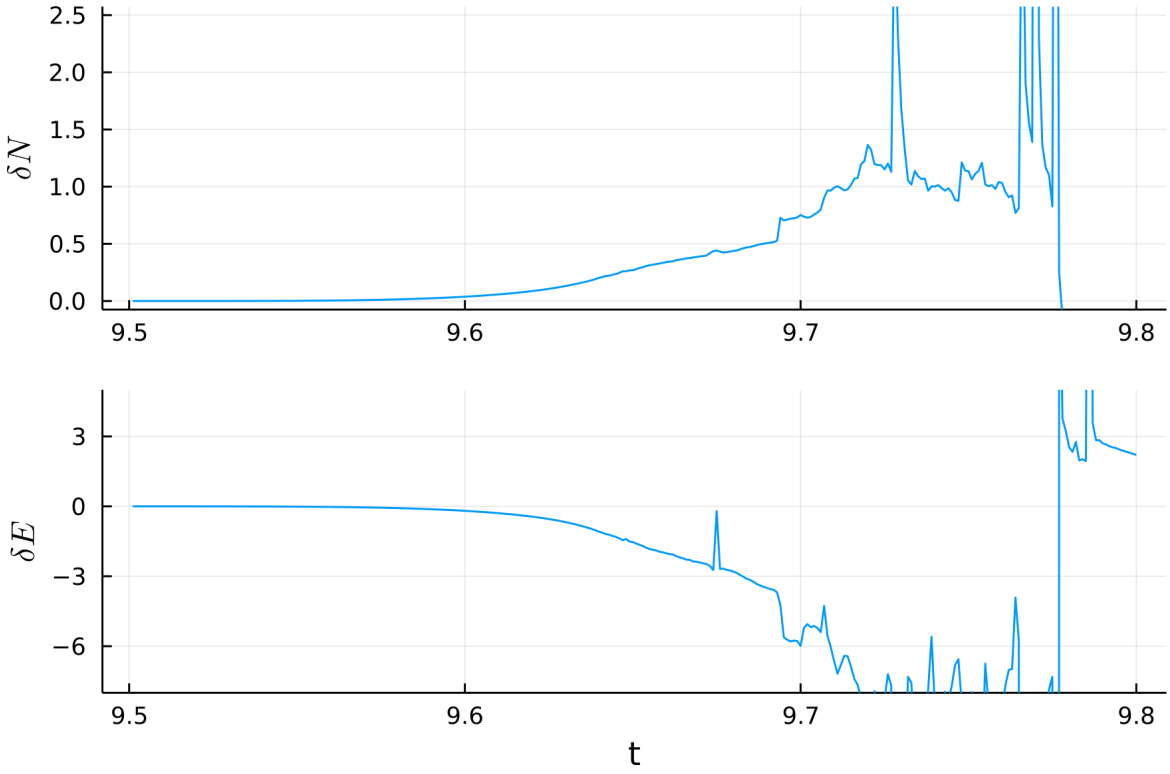

Figure 3.5: Comparison of norm error and energy error

The numerical implementation of the solution of the dispersive operator is performed in the Fourier space. Specifically, let $\mathcal{F}, \mathcal{F}^{-1}$ denote the Fourier transform and inverse Fourier transform respectively, then:

$$e^{i\mathcal{D}\tau}\psi_0 = \mathcal{F}^{-1}\left\{e^{-i\omega^2\tau/2}\mathcal{F}\left\{\psi_0\right\}\right\}$$

while $e^{i\mathcal{N}\tau}\psi_0$ the exact solution of:

$$\frac{\partial\psi}{\partial\tau} = i\mathcal{N}$$

which can be computed straightforwardly as:

$$e^{i\mathcal{N}\tau}\psi_0 = e^{i\tau|\psi|^2}\psi_0$$

The solution of the original problem is given by $e^{i\mathcal{D}\tau+i\mathcal{N}\tau}\psi_0$. The idea of the split-step method is approximate the solution of NLS as a composition of the non-linear and the dispersion operator:

$$e^{i\mathcal{D}\tau+i\mathcal{N}\tau} = e^{id_m\mathcal{D}\tau}e^{ic_m\mathcal{N}\tau}\cdots e^{ic_1\mathcal{N}\tau}e^{id_0\mathcal{D}\tau}e^{ic_0\mathcal{N}\tau} + O(\tau^k)$$

Note that by the Baker-Campbell-Hausdorff formula,if $\mathcal{D},\mathcal{N}$ is commute: $[\mathcal{D},\mathcal{N}] = \mathcal{D}\mathcal{N} - \mathcal{N}\mathcal{D} = 0$ then $c_0 = d_0 = 1$ while $c_1 = c_2 = \cdots c_m = d_1 = d_2 = \cdots d_m = 0$, namely if the operators are commute then $e^{i\mathcal{D}\tau+i\mathcal{N}\tau} = e^{i\mathcal{D}\tau}e^{i\mathcal{N}\tau}$. However, in NLS equation the operator is not commute, hence we have to approximate the exponential operator and retain the solution until order k.

References

[1] Ashour, O.A., 2021. NonlinearSchrodinger: Higher-Order Algorithms and Darboux Transformations for Nonlinear Schr\"odinger Equations URL: `http://arxiv.org/abs/2103.14469`, `arXiv:2103.14469`.

[2] Chattopadhyay, A., Hassanzadeh, P., Subramanian, D., 2020. 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, 373–389. doi:10.5194/npg-27-373-2020.

[3] Chin, S.A., Ashour, O.A., Nikolić, S.N., Belić, M.R., 2016. Maximal intensity higher-order Akhmediev breathers of the nonlinear Schrödinger equation and their systematic generation. Physics Letters, Section A: General, Atomic and Solid State Physics 380, 3625–3629. doi:10.1016/j.physleta.2016.08.038, arXiv:1607.04504.

[4] Doan, N.A., Polifke, W., Magri, L., 2021. Short- And long-term predictions of chaotic flows and extreme events: A physics-constrained reservoir computing approach. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 477. doi:10.1098/rspa.2021.0135.

[5] Jaeger, H., Haas, H., 2004. Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. Science 304, 78–80. doi:10.1126/science.1091277.

[6] Lu, Z., Hunt, B.R., Ott, E., 2018. Attractor reconstruction by machine learning. Chaos 28. doi:10.1063/1.5039508, arXiv:1805.03362.

[7] Lukoševičius, M., Jaeger, H., 2009a. Reservoir computing approaches to recurrent neural network training. Computer Science Review 3, 127–149. doi:10.1016/j.cosrev.2009.03.005.

[8] Lukoševičius, M., Jaeger, H., 2009b. Reservoir computing approaches to recurrent neural network training. Computer Science Review 3, 127–149. doi:10.1016/j.cosrev.2009.03.005.

[9] Martinuzzi, F., 2020. ReservoirComputing.jl. URL: https://github.com/SciML/ReservoirComputing.jl.git.

[10] Pathak, J., Hunt, B., Girvan, M., Lu, Z., Ott, E., 2018. Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach. Physical Review Letters 120, 24102. URL: https://doi.org/10.1103/PhysRevLett.120.024102, doi:10.1103/PhysRevLett.120.024102.

[11] Pathak, J., Ott, E., 2021. Reservoir Computing for Forecasting Large Spatiotemporal Dynamical Systems, in: Reservoir Computing. Springer Nature Singapore, pp. 117–138. doi:10.1145/2967446.2967448.

[12] Raissi, M., Perdikaris, P., Karniadakis, G.E., 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics 378, 686–707. URL: https://doi.org/10.1016/j.jcp.2018.10.045, doi:10.1016/j.jcp.2018.10.045.

[13] Taha, T.R., Ablowitz, M.I., 1984. Analytical and numerical aspects of certain nonlinear evolution equations. II. Numerical, nonlinear Schrödinger equation. Journal of Computational Physics 55, 203–230. doi:10.1016/0021-9991(84)90003-2.

[14] Vinuesa, R., Brunton, S.L., 2021. The Potential of Machine Learning to Enhance Computational Fluid Dynamics , 1–13URL: http://arxiv.org/abs/2110.02085, arXiv:2110.02085.

[15] Weideman, J.A.C., Herbst, B.M., 1986. Split-Step Methods for the Solution of the Nonlinear Schrodinger Equation. SIAM Journal on Numerical Analysis 23, 485–507.

[16] Yoshida, H., 1990. Construction of higher order symplectic integrators. Physics Letters A 150, 262–268. doi:10.1016/0375-9601(90)90092-3.

[17] Zubov, K., McCarthy, Z., Ma, Y., Calisto, F., Pagliarino, V., Azeglio, S., Bottero, L., Luján, E., Sulzer, V., Bharambe, A., Vinchhi, N., Balakrishnan, K., Upadhyay, D., Rackauckas, C., 2021. NeuralPDE: Automating Physics-Informed Neural Networks (PINNs) with Error Approximations , 1–77URL: http://arxiv.org/abs/2107.09443, arXiv:2107.09443.