Parallel Reservoir Computing on Non-Linear Schrödinger Equation

Progress Report 26 February 2022

A. N. Hasmi[a,*],

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

Abstract

We present a simulation of higher order rogue waves simulation using parallel reservoir computing. The training data is a simulation of rogue waves using non-linear Schrödinger equations. Fifth order breather was present in the training data, and we intentionally hides the peak and test our simulation comparing the prediction at breather's peak. We found that the solution of reservoir computing was vary accross realizations. Furthermore, improving the number of internal nodes didn't improve the prediction result. Quadratic and cubic non-linear transformation was employed during the training phase, and the cubic non-linear transformation shown worse performance than the quadratic counterpart.

Keywords: Reservoir Computing, Nonlinear Schrödinger, Rogue Waves

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 [1] (2) Speed up computations by modelling computation expensive task and using machine learning approach as closure model to describe missing dynamics [2].

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(PINN) [3, 4, 5] and Reservoir Computing [1, 6, 7, 8, 9, 10]. Several studies attempted to solve Non linear Schrödinger equation and another integrable equations using data-driven methods. Authors in [5] proposed two stage physics informed neural network for solving integrable wave equations, in which the first stage construct a PINN as usual, and the second stage augment the integrability conditions and difference with first stage training result as additional constraint to the PINN loss function. Another variant of PINN was done by enforcing the $\mathcal{PT}$symmetry condition to the Neural Network layers[11]. The attractive feature of PINN is that the method is meshless, in other word, the training data from PINN is random sampled from an interval of space and time. Even though the PINN method might produce a nice generalization, the training process requires that the (sampled) breather should be enclosed in the training interval data. In a loose term, this mean that PINN is suitable for "interpolation" from the data and might not perform well to extrapolate a phenomena beyond the training data.

Reservoir computing is a specialized version of recurrent neural network, contain an internal states called as reservoir with fixed but random parameters and an output layer which will be trained based on training data. Surprisingly despite the simplied learning compared to RNN, RC has shown superiority in learning tasks compared with RNN [12, 13, 14] especially in modelling dynamical systems. An extension of RC for partial differential equation was presented in [1] for Kuramoto-Sivashinsky (KS) equation, an inclusion of knowledge based model was presented in [6, 8]. Sensitivity of the spectral radius of adjacency matrix was investigated in [7], and Bayesian optimization was employed to select the best hyperparameters for each ESN realization [15, 9] .

This research tries to apply reservoir computing approach to the non-linear Schrödinger equations using reservoir computing. 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

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

rogue waves modelled by the NLS equation. To the best of our knowledge, only [7] which attempted to simulate NLS using Reservoir Computing, in which the authors simulated periodic Akhmadiev breather, Kutnezov-Ma solitons and second order periodic breather collision. The authors concluded that the second order breather collision is more unpredictable compared to the first order breather simulation. Our research is different by the fact that we use higher order breather as a training data (fifth order), and our training data is not only for one period, thus the RC has not yet encounter breather's peak. Furthermore, we employ parallel reservoir approach [1, 8] which has shown superiority in simulating the spatio-temporal problem.

This report is organized into four sections. This section present a short literature review and motivation about the studies. Section 2 present a brief introduction about parallel reservoir computing. An explanation about training data, typical result of simulation and impact of several hyperparameters are discussed in section 3. Finally section 4 provides a conclusion and an overview of further research direction.

2. Theory

2.1. 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, . In this research, we follows the adjustment for spatialtemporal models [1], 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.

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 [16], 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"[12]. 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:

$$y_i^{(k)} = W_{out}^{(k)} \omega(r_i^{(k)}) \tag{2.2}$$

minimized a cost function. The function $\omega : \mathbb{R}^M \to \mathbb{R}^M$ is a (predefined) nonlinear modification aimed to futher enhance the nonlinearity feature of the reservoir[14, 1]. An example of nonlinear transformation which proposed in [1] is:

$$\omega_1(r) = \begin{cases} r_j & j \text{ is odd} \\ r_j^2 & j \text{ is even} \end{cases} \tag{2.3}$$

Similarly, [14] examined different non-linear transformations, which are inspired by the non-linearity present in the simulated equation. They conclude that the non-linear transformation enhance the predictive capability of the simulations, however they did not give conclusive result about which non-linearity transformation should be used, as all the transformations give comparable result.

The usual cost function is:

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

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.

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 [17] which provide implementations of the reservoir computing including the non-linear transformation.

3. Result and Discussion

3.1. Simulation Setup

We trained a set of reservoirs to learn a simulation which governed by the NLS equation:

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

with the amplitude envelope $\psi(\xi, \tau)$ is a function of propagation distance ξ and co-moving time τ. A well known analytic solution of NLs is given by Akhmadiev breather given by

$$\psi(\xi, t) = \left[1 + \frac{2(1-2a)\cosh(\lambda t) + i\lambda \sinh(\lambda t)}{\sqrt{2a}\cos(\Omega x) - \cosh(\lambda t)}\right] e^{it}$$

the behaviour of the solution is governed by a single parameter $a \in (0, 0.5)$, another parameters are related with a, namely the fundamental wave number $\Omega = 2\sqrt{1-2a}$ and growth factor $\lambda = \sqrt{8a(1-2a)}$. Based on this fundamental solution, higher order maximal intensity breather can be constructed by choosing particular $a_2, a_3 \ldots, a_k$ such that under particular phase and amplitude ,the breathers collides at particular time and position resulting at a higher peak with the procedure given by [18]. We aimed at simulating the construction of 5^{th} order breather on a uniform background describing a highly non-linear situation. The objective of this simulation is to investigate under what conditions the Reservoir Computing can learn and simulate the maximum peak amplitude 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 [19] implemented in Julia library the NonlinearSchrodinger.jl [20]. A brief explanation of the split-step method is given in Appendix. The initial condition was prepared according to [18] which generate the initial condition for maximum intensity soliton via Darboux Transformation. The modulation parameter used in generating the training set is $a_1 = 0.4802$ in which there exists a unique maximum intensity soliton up to fifth order. The length of the spatial domain is one spatial period $[-L/2, L/2]$ with $L = 2\pi/\Omega$ and 256 spatial nodes. The simulation was started from $t = 0$, with $\Delta t = 5 \cdot 10^{-4}$

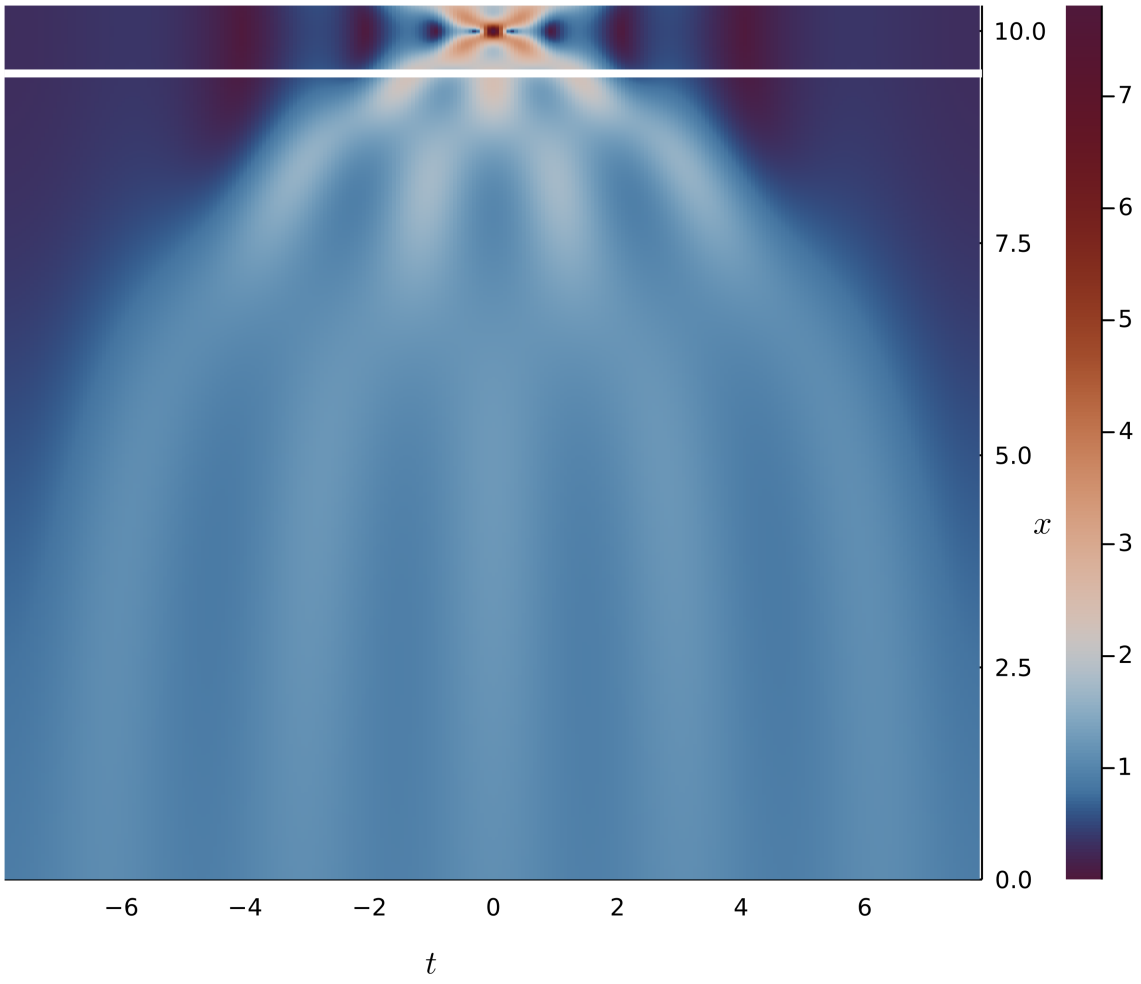

Figure 3.1: Fifth order breather simulation by NLS

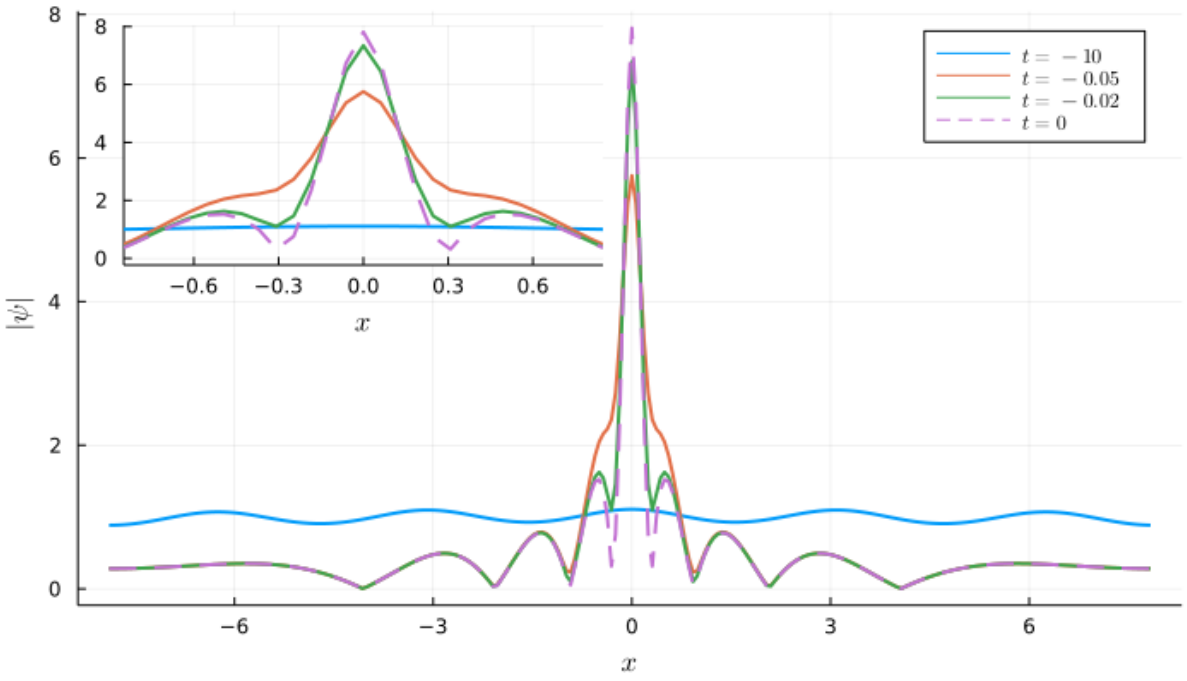

Figure 3.2: A snapshot of training data at $t = -10$ (beginning of simulation), $t = -0.05, t = -0.02$ and $t = 0$ (expected peak). The inset gives a more detailed view at centre of the coordinate.

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

Table 1: Hyperparameter for Nonlinear Schrödinger

and the peaks was achieved after , meaning there are 20000 time grids between the start of training data and the peak. Figure 3.2 illustrate the complexity of the result that we want to achieve. The simulations begin with relatively uniform amplitude , toward $t = 0$, a higher order breather wave representing the rogue waves occurs.

The hyperparameters used in the prediction of NLS is given in Table 1 which are decided in reference to the simulation of Kuramoto-Sivashinsky equations using RC [21].

3.2. Performence Metric and General Result

To quantify the error between predictions, a metric should be defined. As we are interested in the prediction at maximum peak, we will only find the difference between prediction and the real solution at this particular point. Precisely, let the exact solution $\psi_{10}(x) = \psi(x, t = 10)$ and prediction at $t = 10$ as $\hat{\psi}$, we assume that both of them are elements in a normed space $(X, \|\cdot\|)$ than the error is defined as:

$$e = \frac{\left\|\hat{\psi}_{10} - \psi_{10}\right\|_{H^1}}{\|\psi_{10}\|_{H^1}} \tag{3.1}$$

With $H^1$is the Sobolev space metric defined as:

$$\|u\|_{H^1} = \left[\int_X |u|^2 \, dx\right]^{\frac{1}{2}} + \left[\int_X \left|\frac{\partial u}{\partial x}\right|^2 dx\right]^{\frac{1}{2}}$$

As our function is periodic, the spatial derivative is computed in the Fourier space, $\frac{\partial u}{\partial x} = \mathcal{F}^{-1}\{ik\mathcal{F}\{u\}\}$.

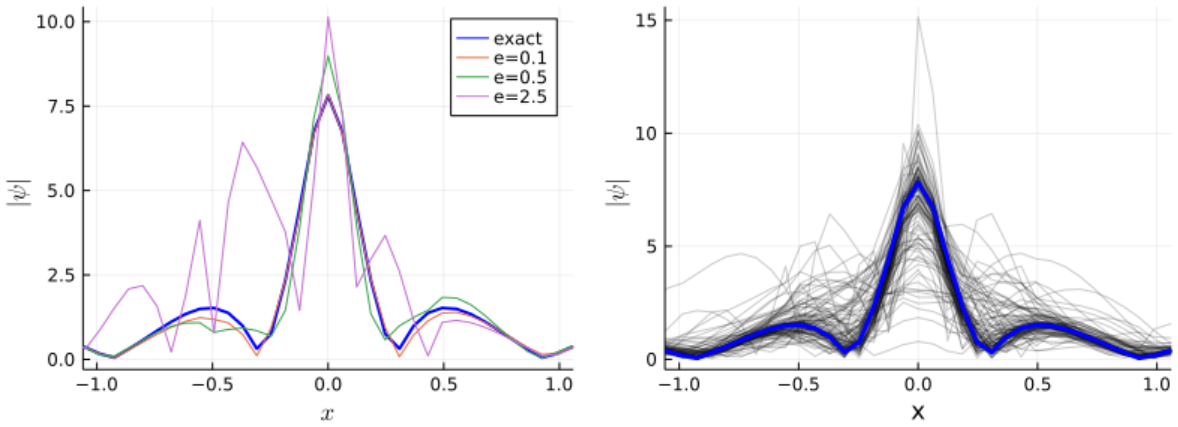

Figure 3.3: Left: Representative result for $e = 0.11$ (red line), $e = 0.88$ (green line) and $e = 2.685$ (purple line) blue line is the test data. Right: blue figure shows the test data while the black lines is an ensemble of 100 realization.

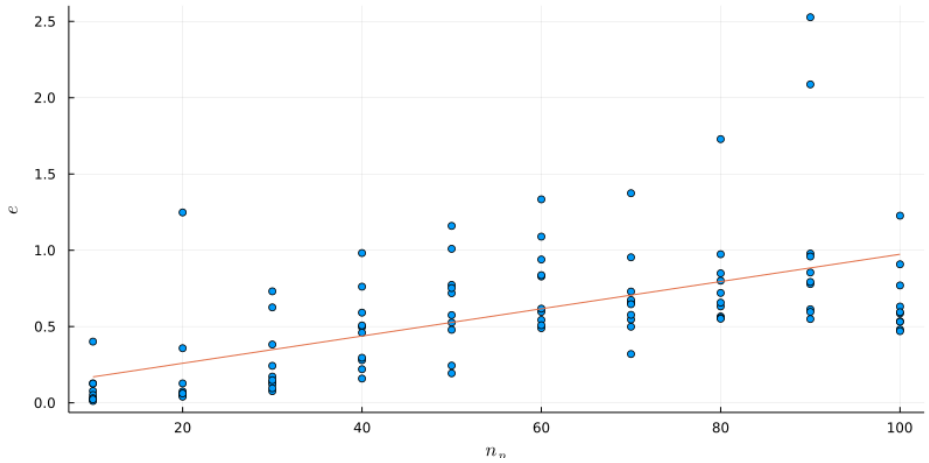

Figure 3.4: Simulation error at $t = 10$ as a function of number of prediction steps (n_p) (x-axis). Each dot represent a realization of RC. Ten realizations are done for each n_p.

In general, the result will vary for different realization. Figure 3.3 (a) shows some of possible results associated with different error. While the (b) part shows an ensemble of result from 100 realizations with the same hyperparameters.

3.3. Prediction Capability

As our aim is to capture the peak of the breather, we want to know how long in advance we can predict the occurance of maximal intensity breather given the training data. Hence, we varied the time between beginning of prediction phase with the occurance of the peak of breather ($t = 10$). In other words, if we want n_p steps in prediction phase, than we started the prediction phase corresponding to $t = 10 - n_p\Delta t$. Notice that in our simulation, the start of training data is fixed ($t = 0$). This means that as n_p increase, the number of training data is slightly decrease, however this decrease is relatively small compared to the overall training data.

We run simulations by varying n_p from 10 to 100 in 10 increment. Ten realizations are done for each n_p, with the hyperparameters as in Table 1 and 200 nodes in each reservoir and using ω_1 as non-linear transformation. Figure 3.4 plots the error defined as 3.1, with the red line is a linear fit of the data. An observation of the result shows an expected trend that the prediction accuracy detoriates as the number of prediction steps increase. However, we note that the error is varied for each realization, and we want to further study about the error at using different simulation hyperparameters. For this, we choose to focus on investigating at $n_p = 40$, as at this n_p, we observe a significant variance of the realized error.

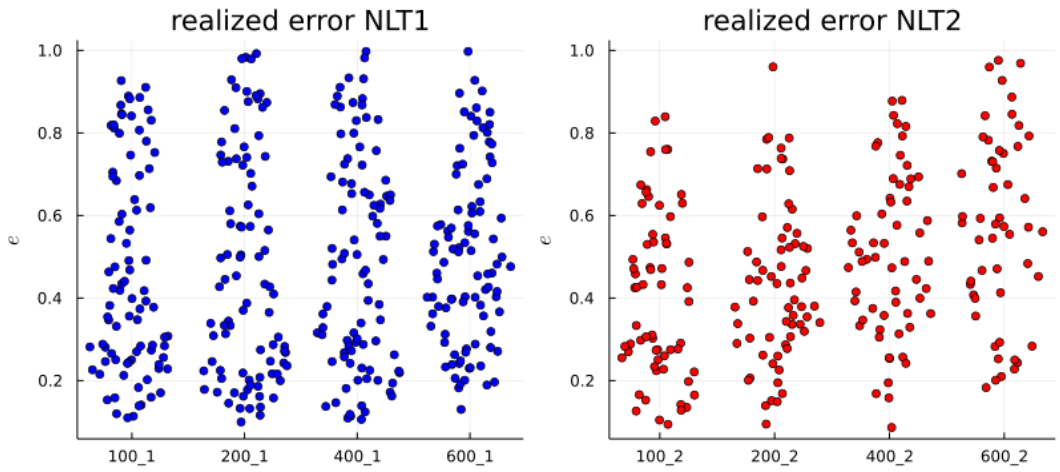

Figure 3.5: The dotplots of error using different internal nodes and non-linear transformations. Each dot corresponds to one ESN realization. The left plot correspond to (2.3) while the right plot represent the result using (3.2)

3.4. Reservoir Nodes and Nonlinear Transformation

In addition to nonlinear transformation (2.3), we introduced a different non-linear transformation which are inspired by the cubic non-linearity of the NLS equaation, given by:

$$\omega_2(r) = \begin{cases} r_j & j \text{ is odd} \\ |r_j|^2 r_j & j \text{ is even} \end{cases} \tag{3.2}$$

The numerical experiments are conducted by varying the reservoir nodes namely 100, 200, 400 and 600 nodes. We presents a result by filtering out the simulations which yield $e > 1$, as these result are totally different quantitative and qualitatively from the expected simulations. The dotplot containing the error of each realization is shown in Figure 3.5. As evident from the figure, increasing the number of reservoir nodes does not correspond to improvement of the simulation result. This is more visible in lower plot of figure 3.6, which shows that there is tendency that the mean error rises as the enlargment of reservoir nodes.

We would like to further elaborate about the effects of non-linearity transformation. As opposed to the reduction of cases with $e > 1$ with number of reservoir nodes in ω_1, the realized simulations with $e > 1$ surge with the increase number of reservoir as can be observed in the upper plot of figure 3.6. Furthermore, figure 3.5 shows that ω_2 consistenly result in fewer realizations with $e < 0.2$ compared to non linear transformation ω_1. Thus, we conclude that ω_2 is worse than the quadratic ω_1, a conclusion which is counter intuitive as ω_2 is based on the non-linearity present in the NLS equation. By viewing the non-linearity transformation as an "knowledge based model" to the reservoir, the awful result of ω_2 can be attributed as (numerical) instability of feedback system for ESN network which was discussed in [9].

4. Conclusion and Future Work

The study shows that in some realizations, reservoir computing can be used to predict the occurance of rogue wave with reasonable error. However, the performance of the reservoir computing can vary on particular realization realization. The result in this paper shows that increasing the number of reservoir doesn't imply improvement of the prediction result.

To improve the robustness of the result, we might want to optimize the hyperparameters as discussed in [9, 15]. The hyperparameter optimization is based on observation that at each realization of adjecancy matrix A and input coupling matrix W_{in} , we can scale them with maximum eigenvalue ρ and input scaling σ respectively which has high sensitivity to the ESN prediction [7].

This study showed that cubic non-linear transformation has worse performance compared to quadratic transformation. We are interest in an alternative quadratic transformation, namely $|r|\, r$. In addition, the number of reservoirs M can be varied, as probably only fewer M is needed for the ESN predictions. Reference [8] shown that having $M > 1$ indeed improve the accuracy of the result, however, after certain M the gain in the result is not as significant as the additional computational and memory cost. Furthermore, having $M = 1$ simplifies the computation and enabling easier implementation in enforcing a global constraint, i.e. the integrability condition of NLS as done in [5] for PINN.

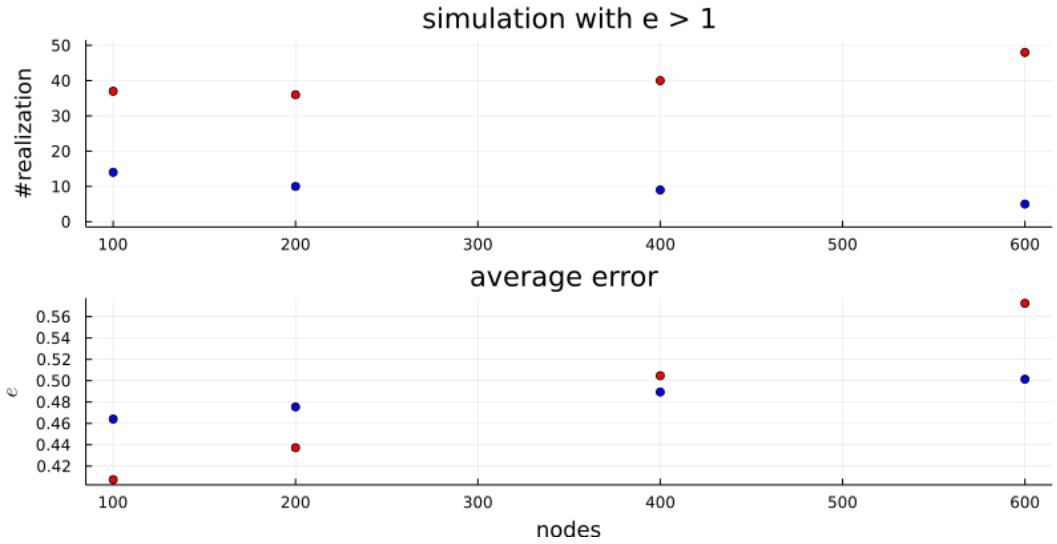

Figure 3.6: Upper plot: The number of realized simulations with $e > 1$. Lower plot: the mean error for simulations with $e \leq 1$. The horizontal axis correspond to the number of nodes in reservoir, the blue color correspond to simulation using ω_1, the red color using ω_2.

Appendix A. NLS Training Data Generation

The Nonlinear Schrodinger equation was solved using 6th order split-step method [22]. The superiority of split-step method compared to variants of finite difference methods to discretize the nonlinear Schrödinger equation was investigated in [23]. 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:

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

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] J. Pathak, B. Hunt, M. Girvan, Z. Lu, E. Ott, Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach, Physical Review Letters 120 (2) (2018) 24102. doi:10.1103/PhysRevLett.120.024102.
URL https://doi.org/10.1103/PhysRevLett.120.024102

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

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

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

[5] S. Lin, Y. Chen, A two-stage physics-informed neural network method based on conserved quantities and applications in localized wave solutions (2021) 1–18arXiv:2107.01009.
URL http://arxiv.org/abs/2107.01009

[6] J. Pathak, A. Wikner, R. Fussell, S. Chandra, B. R. Hunt, M. Girvan, E. Ott, Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model, Chaos 28 (4) (2018). arXiv:1803.04779, doi:10.1063/1.5028373.

[7] J. Jiang, Y. C. Lai, Model-free prediction of spatiotemporal dynamical systems with recurrent neural networks: Role of network spectral radius, Physical Review Research 1 (3) (2019) 33056. arXiv:1910.04426, doi:10.1103/PhysRevResearch.1.033056.
URL https://doi.org/10.1103/PhysRevResearch.1.033056

[8] A. Wikner, J. Pathak, B. Hunt, M. Girvan, T. Arcomano, I. Szunyogh, A. Pomerance, E. Ott, Combining machine learning with knowledge-based modeling for scalable forecasting and subgrid-scale closure of large, complex, spatiotemporal systems, Chaos 30 (5) (2020). arXiv:2002.05514, doi:10.1063/5.0005541.

[9] F. Huhn, L. Magri, Gradient-free optimization of chaotic acoustics with reservoir computing 014402 (2021). arXiv:2106.09780, doi:10.1103/physrevfluids.7.014402.
URL http://arxiv.org/abs/2106.09780

[10] N. A. Doan, W. Polifke, L. Magri, 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 (2253) (2021). doi:10.1098/rspa.2021.0135.

[11] W. Zhu, W. Khademi, E. G. Charalampidis, P. G. Kevrekidis, Neural Networks Enforcing Physical Symmetries in Nonlinear Dynamical Lattices: The Case Example of the Ablowitz-Ladik Model, arXiv Nonlinear Sciences 1 (2) (2021). arXiv:2110.04693v1.

[12] H. Jaeger, H. Haas, Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication, Science 304 (5667) (2004) 78–80. doi:10.1126/science.1091277.

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

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

[15] A. Racca, L. Magri, Robust Optimization and Validation of Echo State Networks for learning chaotic dynamics, Neural Networks 142 (2021) 252–268. arXiv:2103.03174, doi:10.1016/j.neunet.2021.05.004.
URL `https://doi.org/10.1016/j.neunet.2021.05.004`

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

[17] F. Martinuzzi, ReservoirComputing.jl (2020).
URL `https://github.com/SciML/ReservoirComputing.jl.git`

[18] S. A. Chin, O. A. Ashour, S. N. Nikolić, M. R. Belić, 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 (43) (2016) 3625–3629. arXiv:1607.04504, doi:10.1016/j.physleta.2016.08.038.

[19] H. Yoshida, Construction of higher order symplectic integrators, Physics Letters A 150 (5-7) (1990) 262–268. doi:10.1016/0375-9601(90)90092-3.

[20] O. A. Ashour, NonlinearSchrodinger: Higher-Order Algorithms and Darboux Transformations for Nonlinear Schrödinger Equations (2021). arXiv:2103.14469.
URL `http://arxiv.org/abs/2103.14469`

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

[22] J. A. C. Weideman, B. M. Herbst, Split-Step Methods for the Solution of the Nonlinear Schrodinger Equation, SIAM Journal on Numerical Analysis 23 (3) (1986) 485–507.

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