**ESN - Lorenz63 - Summary**

**Global goal** – analyze how well a particular class of machine learning models (Echo State

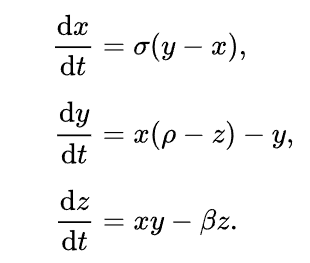
Networks (ESNs) aka Reservoir Computing) reproduces extreme events in prototype chaotic weather/climate models.

**Intermediate goal** – check whether ESN is able to reproduce extreme events on **Lorenz63** model.

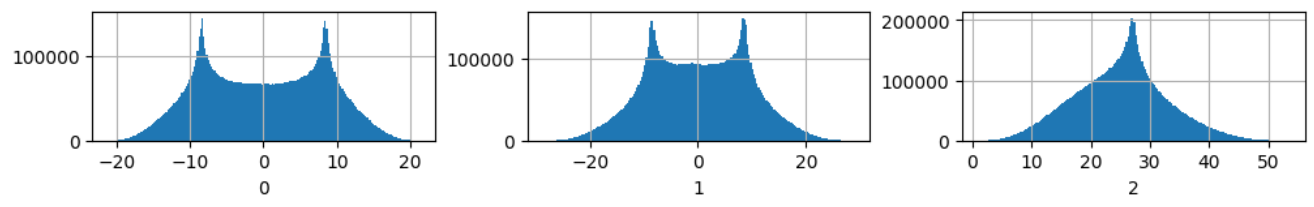
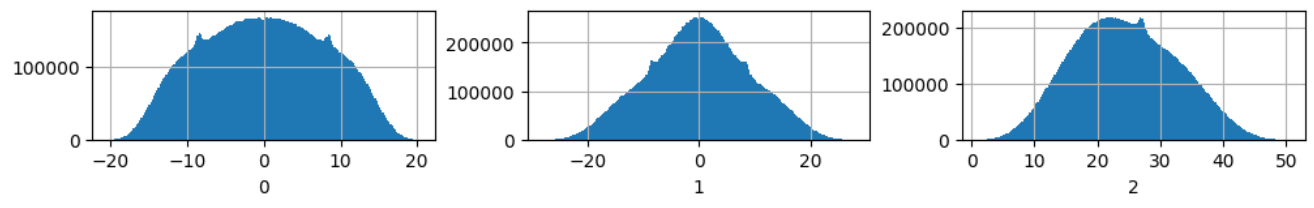
**Plan**:

1. Generate training data using Lorenz63 model.
2. Examine “original” ESN and explain our concerns.
3. Suggest “author’s” ESN, that covers previous concerns, examine the model and evaluating methods.
4. Result summarization and future plans

**Step 1. Generate training data using Lorenz63 model**

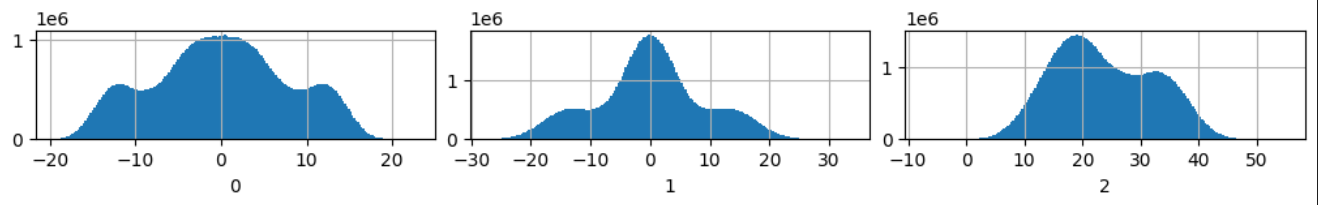
It is crucial to pay attention to generation process of the training data. Lorenz63 has 3 independent variables (x, y, z; or 0, 1, 2 correspondently). We generated data using the equations on pic 1, where .

pic

The smaller we take integration step (dt), the more accurate will be generated data according to math theory. We tried various dt, and it so happened that dt influences value distribution of the variables – pic 2, 3, 4.

pic dt = 0.005

pic dt = 0.01 (integration step)

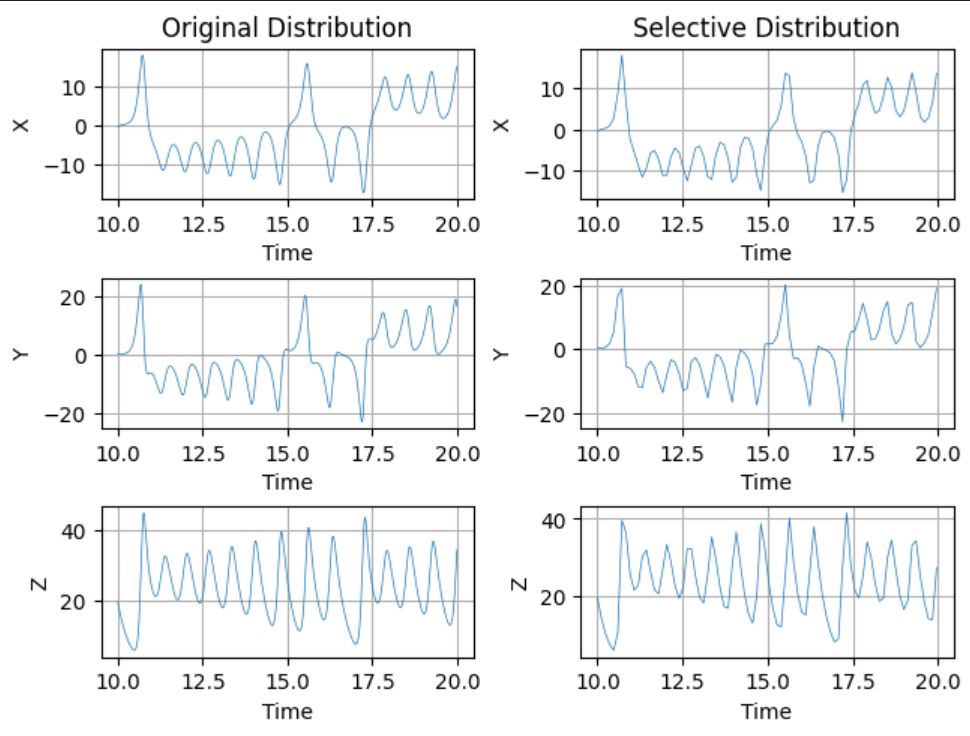


pic dt = 0.001

If we took dt < 0.001 value distribution did not change. So, we stopped on dt = 0.001.

There is one distinctive feature of chaotic Lorenz63 system - it has stable attractor. The choice of starting point for generation new long trajectory (more than 1m points) does not influence the shape of attractor and its statistic properties, like mean and variance of every variable. We will need them further for data preprocessing and evaluation of predicted trajectories.

|  |  |  |  |
| --- | --- | --- | --- |
| Lorenz63 attractor | Var 0 | Var 1 | Var 2 |
| mean | -1.707e-3 | -1.704e-3 | 2.368e+01 |
| variance | 63.237 | 81.679 | 73.676 |

Also, if we generate data with small integration step (0.001), differential equation systems, like Lorenz63, have very similar values of neighbor points. We think that training process of recurrent networks is more efficient, when the values of such points are not the same. So, we decided to take every 100th point, so that the graphs of a variables against time – var(t) – with all points and with only every 100th point seemed to be the same.

pic Original distribution – graph var(t) of all generated points.

Selective distribution – var(t) with every 100th point from original distribution

Overall, we took 10m points generated with dt = 0.001 step and selected each 100th point for ESN training.

**Step 2.** **Examine “original” ESN, and explain our concerns.**

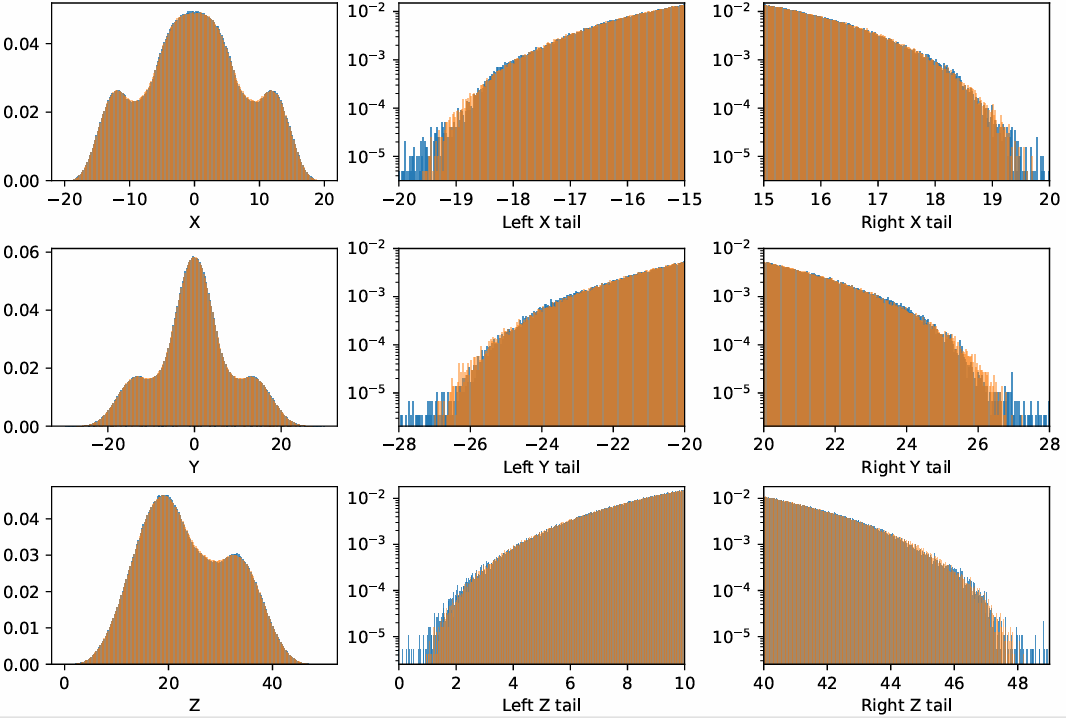
The design of most simple ESN can be found in this article (<https://arxiv.org/pdf/2012.02974>). The simple ESN (we are going to call as “original” ESN), is characterized by a single trainable output layer. The architecture of our “original” ESN is described below:

* Win(3, N) – random matrix uniformly generated form [-w,w] segment
* Reservoir(N, N) – random matrix generated as described in article
* (hidden) states[i] = tanh(states[i-1] @ reservoir + input\_data[i] @ Win)
* Wout(N, 3) – trainable matrix (with regularization)
* Output \_data = states[i] @ Wout

For such ESN we need big reservoir – N x N matrix – approximately N = 1000, in order to generate long trajectory with proper value distributions. The more N, the less probability for ESN to collapse during prediction.

It is well-known, that ESN models can reproduce attractors of chaotic systems. However, we had a more specific question: is ESN able to reproduce extreme events in Lorenz63 attractor – tails of value distribution of the variables?

We decided to check whether original ESN is able to do that.

And that is true, for instance pic 6.

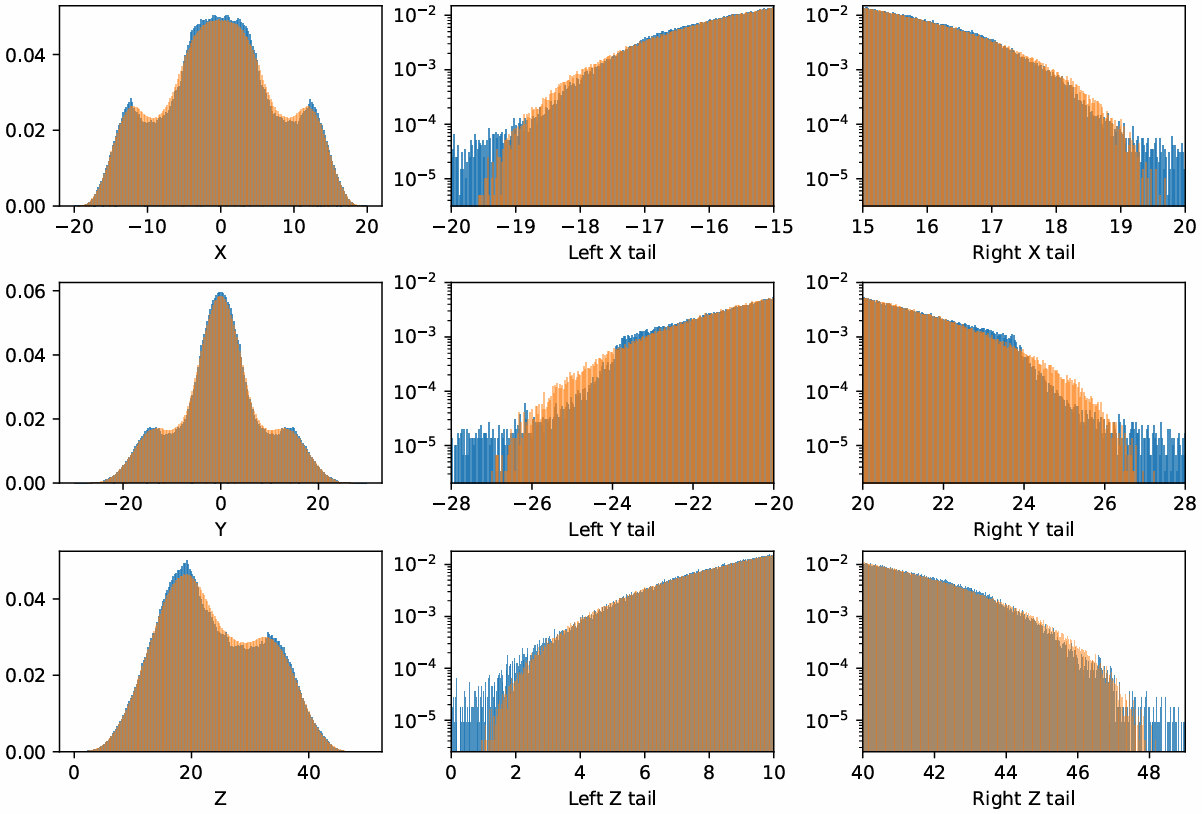
pic Original ESN with N = 3000, sparsity = 50%, spectral radius = 0.5, w = 0.5, regularization = 0.001

*All graphs are with pdf on vertical axis; the tails are shown with logarithmic scale*

*Orange – Lorenz63 distribution; Blue – ESN distribution*

The overall distribution is excellent, the tails are a bit outperformed, but that can be fixed by proper hyper-parameters selection.

However, a problem appears: it takes to a lot of time and memory for training and especially for predicting 10m points with 3000 x 3000 reservoir. If we plan to work with further chaotic systems, that has much more variables, we would need to deal with very resource-intensive process. So, we need to decrease the size of the reservoir as much as we can.

We tried original ESN with various N (from 1000 to 100). The smallest ESN, which does not collapse and has relatively good distribution, has N = 800, but the tails look quite bad.

pic Original ESN with N = 800, sparsity = 50%, spectral radius = 0.5, w = 0.5, regularization = 0.001

All graphs are with pdf on vertical axis; the tails are shown with logarithmic scale

*Orange – Lorenz63 distribution; Blue – ESN distribution*

It means that, while decreasing the size of reservoir, we need somehow maintain its ability to remember patterns of value distribution of the variables. The obvious solution, we decided to use is to increase the number of trainable Wout layers. Our solution will be thoroughly explained in the next step.

Interesting fact – at some point, when we were checked original ESN, it did not want to predict long trajectory and each time collapsed. The key solution to this problem was my habit to do data standardization always when I work with ML models. This case showed that ESN with only one output layer (without bias vector) is not able to cope with standardization. Even if I just moved initial distribution, so that its mean value is zero, ESN performed worse, than If I did not do that. However, enhanced ESN model from the step 3, benefited from initial data standardization. I guess because it has more Wout layers.

**Step 3. Suggest “author’s” ESN, that covers previous concerns, examine the model and evaluating methods.**

The “author’s” ESN is similar to “original” one, but have some important key features:

* We standardized input data
* We used 3 trainable output layers with 2 nonlinear between them:
  + Linear(N, 2N/3)
  + ReLU()
  + Linear(2N/3, N/3)
  + ReLU()
  + Linear(N/3, 3)
* Because of this, we had to use backpropagation while training
* The loss function was MSE
* The optimizer was Adam with common parameters
* The whole training trajectory was splitted into batches and shuffled (however, training on the sequential (not shuffled) batches tend to have a bit better result)
* Each input batch was trained after pre-warming of hidden states

We used torch framework. We encountered lots of obstacles while doing this part of research. Finally, we hold a small cross search for best hyper-parameters among the length of training batch (batch), input scaling parameter (w), spectral radius of the reservoir (radius), size of the reservoir (N) and number connections per neuron in reservoir (degree). Note: sparsity (of reservoir) = degree / N \* 100%

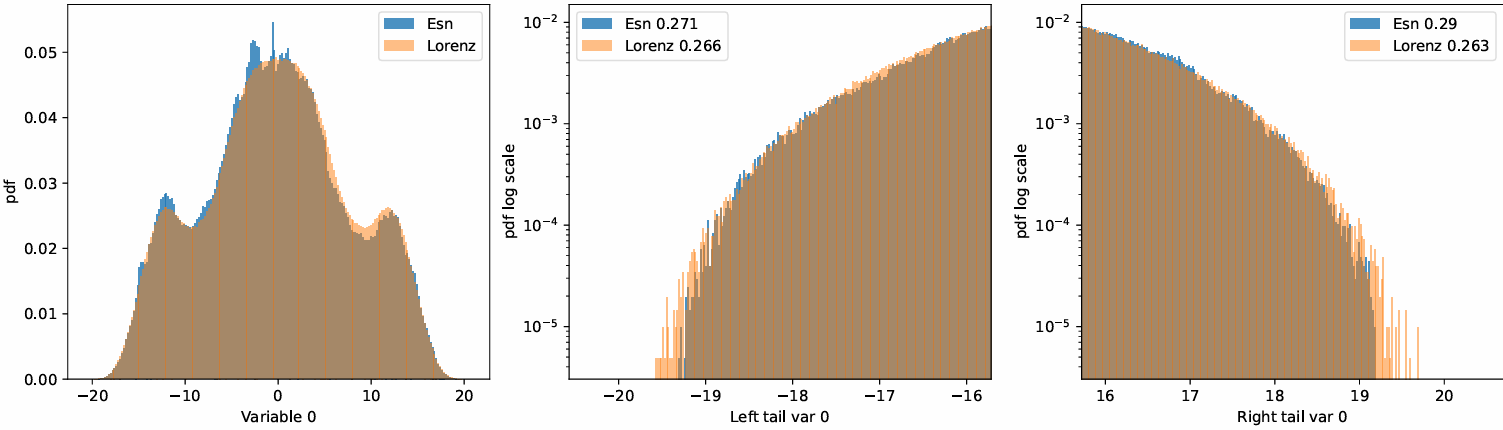
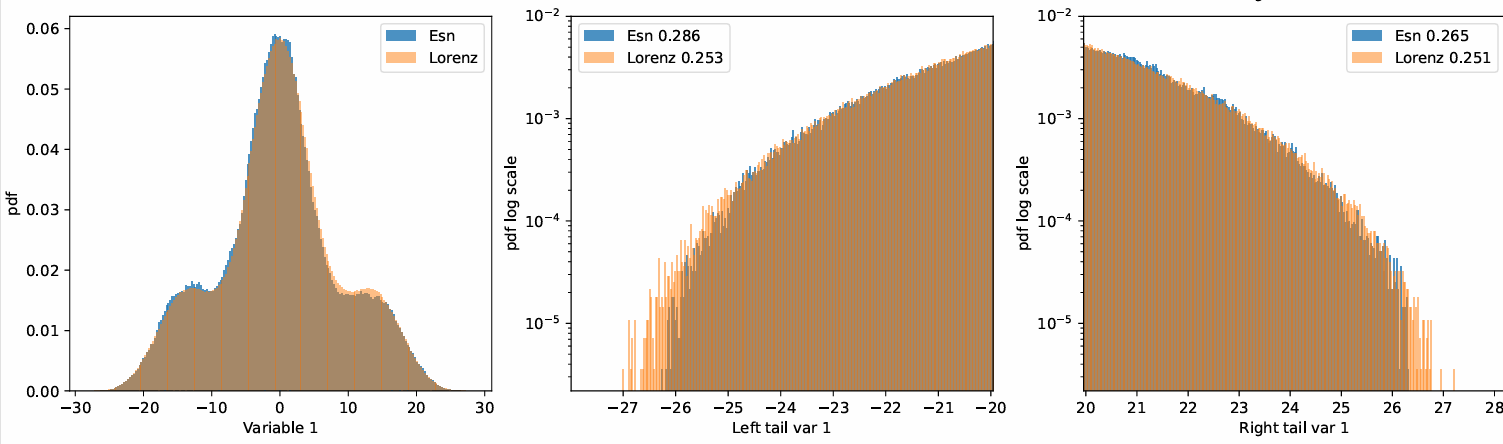
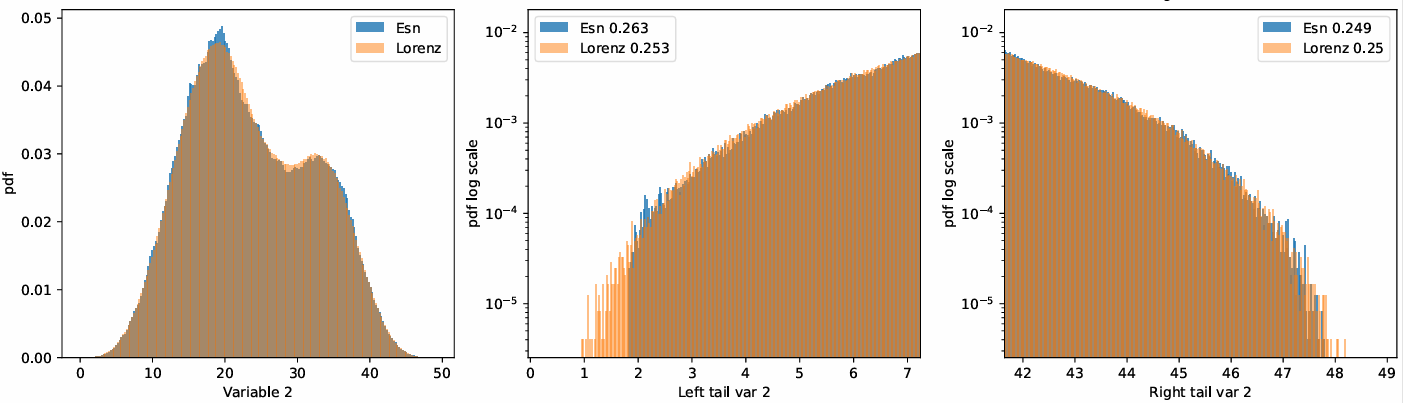
In order to receive the best parameters used Pareto distribution by the following plan:

* We chose the quantile that indicates our tail: left – 0.01, right – 0.99
* Then we fit tail values from Lorenz63 and “author’s” ESN (trained with particular parameters) into pareto distribution and receive the k coefficient. On below histograms, you can see the k value near the corresponding distributions.
* We took MSE of all tails from both Lorenz63 and particular ESN and placed it as an “error” constant.
* We looked for the minimum error value among all ESNs trained with different hyper-parameters.

The best solution has next hyper-parameters with pareto error = 3.59e-04:

* Warm = 100, batch = 400
* w = 0.5, radius = 0.5
* N = 60, degree = 30

The results are awesome (pic 8). Various random states may give slightly different distributions. Here, for instance, the tails are predicted very good, but there are some outliers in the middle of var 0 distribution and mean value for var 0 and var 1 is -0.21, while Lorenz63 has approximately 0 for both var 0 and var 1. Another random state may fix the histogram in the middle of distribution, but less perform at the tails. However, in this research, we are more interested in tails of distributions, so we will consider this example as best.

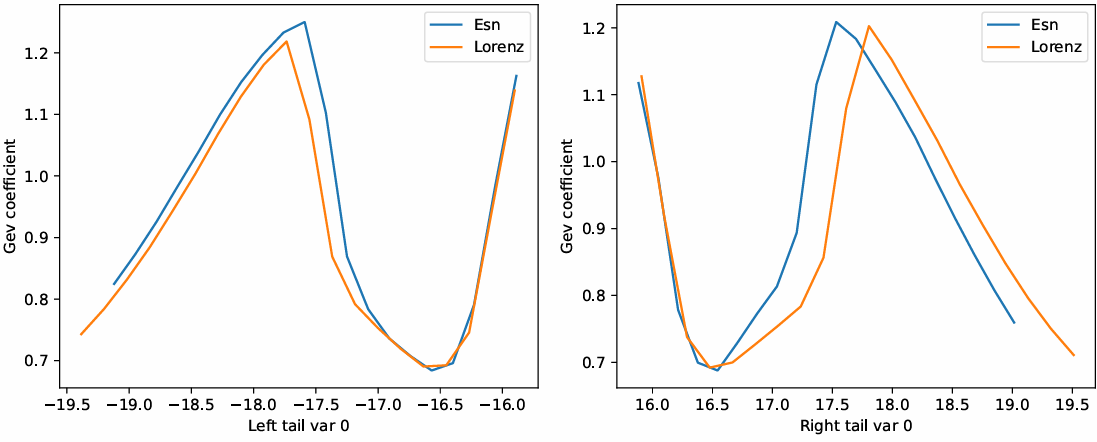


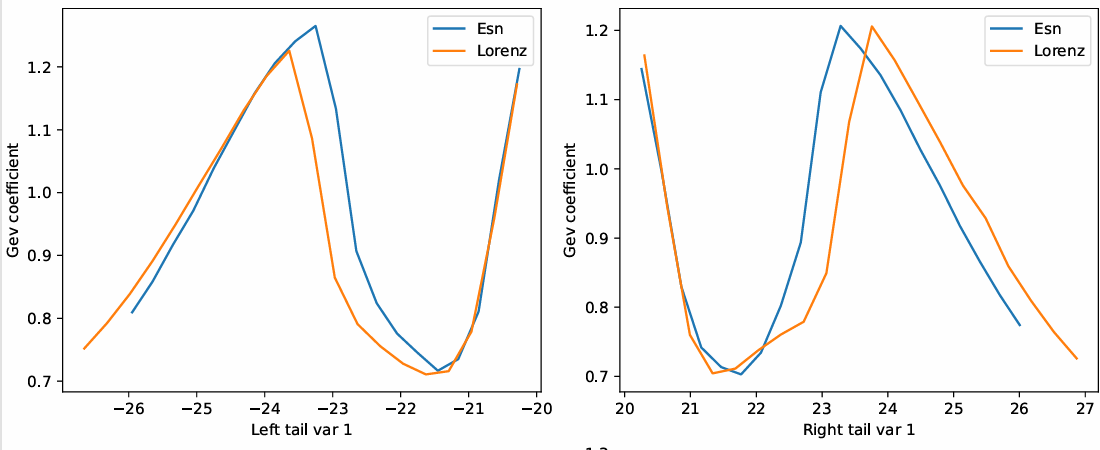
pic Best author’s ESN with N = 60, sparsity = 50%, spectral radius = 0.5, w = 0.5

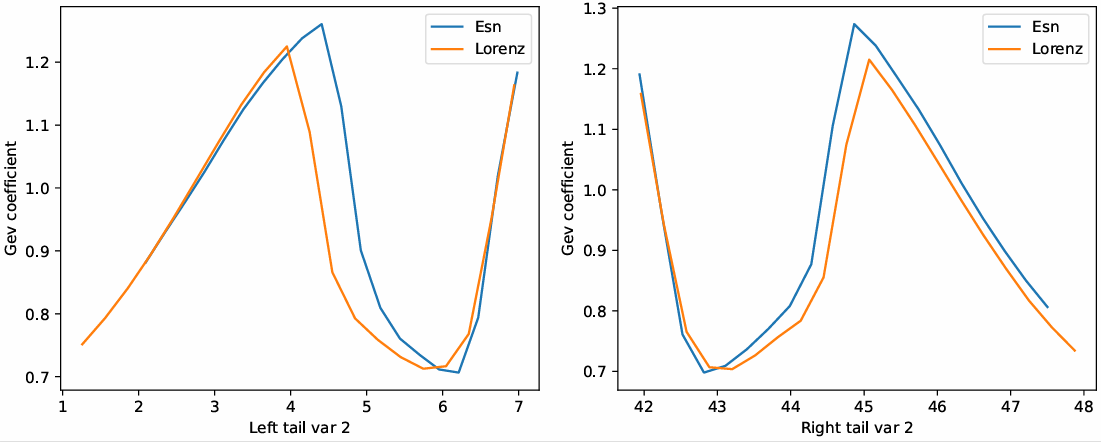
In order to compare our best ESN distribution with Lorenz63 distribution somehow numerically (not only by eye), we decided to use conditional General Extreme Value distribution by following plan:

* We chose the quantile that indicates our tail: left – 0.01, right – 0.99.
* We splitted the section [Lorenz63.min(), quantile\_left] (for left tail) and [quantile\_right, Lorenz63.max()] (for right tail) into 21 equal sections and received 20 conditional values for each tail.
* Then we transform tail values by following formula
* Then we fit tail values into GEV distribution and receive the k coefficient.
* After that, we plot a graph of k(condvalue) for particular left and right tail for all variables.
* Do the same steps with our best ESN solution and compare graphs for corresponding tails.
* Note: nor Lorenz63, nor ESN distribution is dense, because 10m points is too little for such subset to be dense. Because of this, after doing transformation on initial tail, we can receive “outliers” that affect the k coefficient of conditional GEV significantly. We decided to bound the transformed tail on [0, 100] section.

For the best “authors” ESN, we received the next comparison conditional GEV graphs (pic 9). Due to outliers’ elimination, the graphs look smooth in both Lorenz63 and ESN cases. The best ESN line tend to repeat the Lorenz63 trend line with small shifts. We consider this result worthy.



****

****

pic Conditional GEV graphs. Comparison between Lorenz63 and author’s best ESN

**Step 4: Result summarization and future plans**

We conducted diligent research on ESN models throughout autumn semester. We have a pipeline that works perfectly for small chaotic Lorenz63 model. Our next plan is to examine more complicated models, like Lorenz96, Kuramoto–Sivashinsky equation, etc, and investigate further, whether ESN is able to reproduce extreme events in chaotic systems.