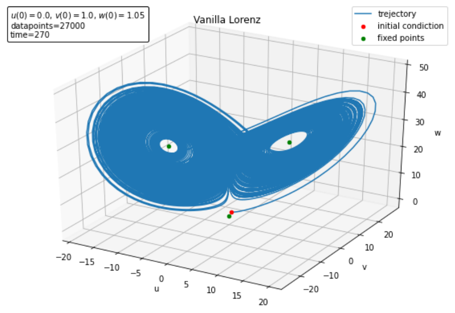
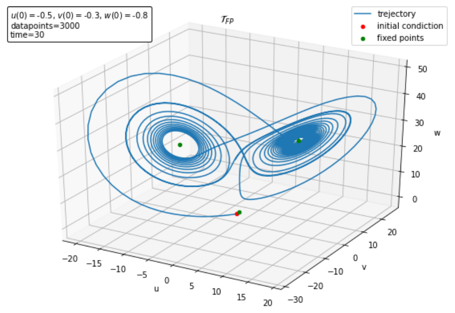
Entropy and Information in Lorenz Trajectories

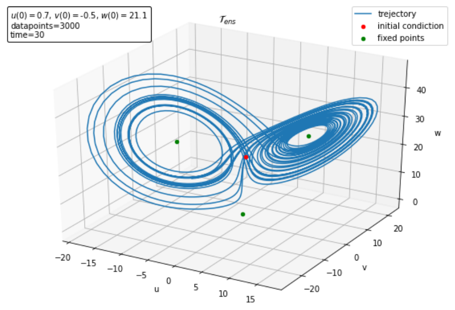
December 2021

1 Entropy in Lorenz Trajectories

The following text is going to show, using different functions, why different trajectories in Lorenz Model seem to contain more information than others. Namely the performance of a LSTM trained with a dataset of trajectories with initial condition close to fixed points are better then others.







The idea behind this phenomena that we observe is that close to the fixed points the dynamics is far from the chaotic dynamics. We would like to measure the information that the different trajectories contain.

1.1 Shannon Entropy

The first function that we can use is the Shannon Entropy [1948]. This function is well known in Information Theory. The main idea of this function is that it can measure the amount of information needed to reproduce the input (that is an event sampled from a distribution of probability). Namely, the minimum number of bits that you can use to store an events from a distribution.

Given X a random variable, with possible outcomes $x_1, x_2, ..., x_N$, with probabilities $p(x_1), p(x_2), ..., p(x_N)$, we can define the Shannon Entropy:

$$\begin{aligned} I_{sh} &= I_{sh}(p(\mathbf{x})) \\ I_{sh} &= -\sum_{i=1}^{N} p(x_i)\log_2(p(x_i)) \end{aligned} \tag{1}$$

1.1.1 Technical Issues

Lorenz's systems trajectory is a timeseries with 'float64' numbers.

A classic Shannon Entropy algorithm has to compute the relative frequency of each number in the array to give an approximated value of $p(x_1)$. A known issue [[1]] when you have to use float64 values is that it is almost impossible to find two identical values in a timeseries.

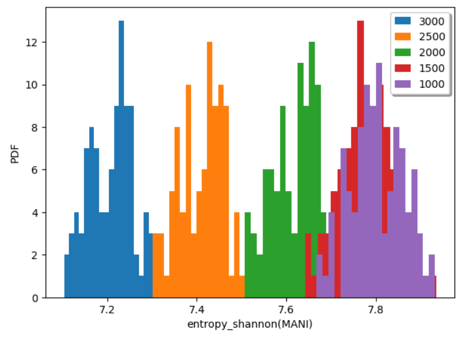
Indeed the function "torch.load()"[[2] creates a vector of 'float64' arrays with a precision of 16 decimals. That's why to compute entropy is useful to use another definition of equality.

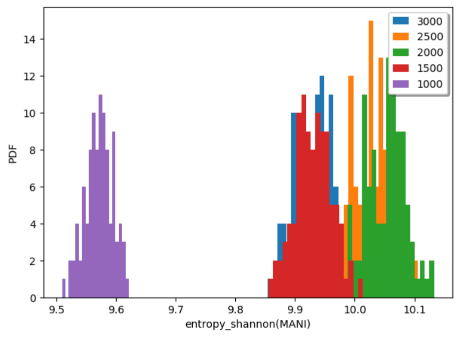
$$\begin{aligned} x(t_a) &= a_0, a_1 a_2 \dots a_{16} \\ x(t_b) &= b_0, b_1 b_2 \dots b_{16} \end{aligned} \tag{2}$$

We say that $x(t_a) = x(t_b)$ with precision p if :

$$a_0 = b_0,\ a_1 = b_1,\ \dots,\ a_p = b_p \tag{3}$$

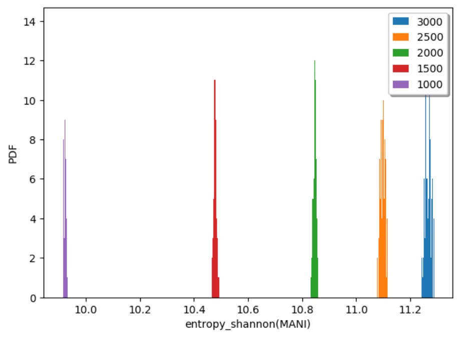
Following this idea we can make different plots of Shannon Entropy of the trajectories close to fixed points. With $p = 1, 2, 3, 4, 16$

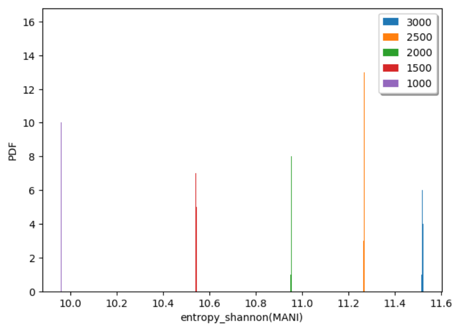


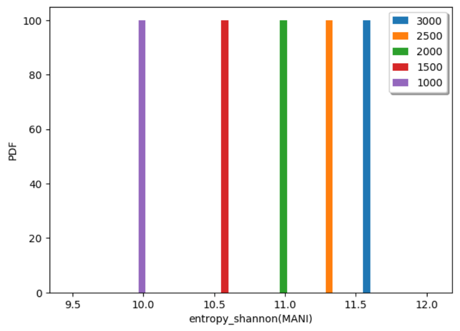


[1]https://stackoverflow.com/questions/18106975/compare-float-and-float64-in-python/18107037

[2]Used to analyze the entropy of the trajectories

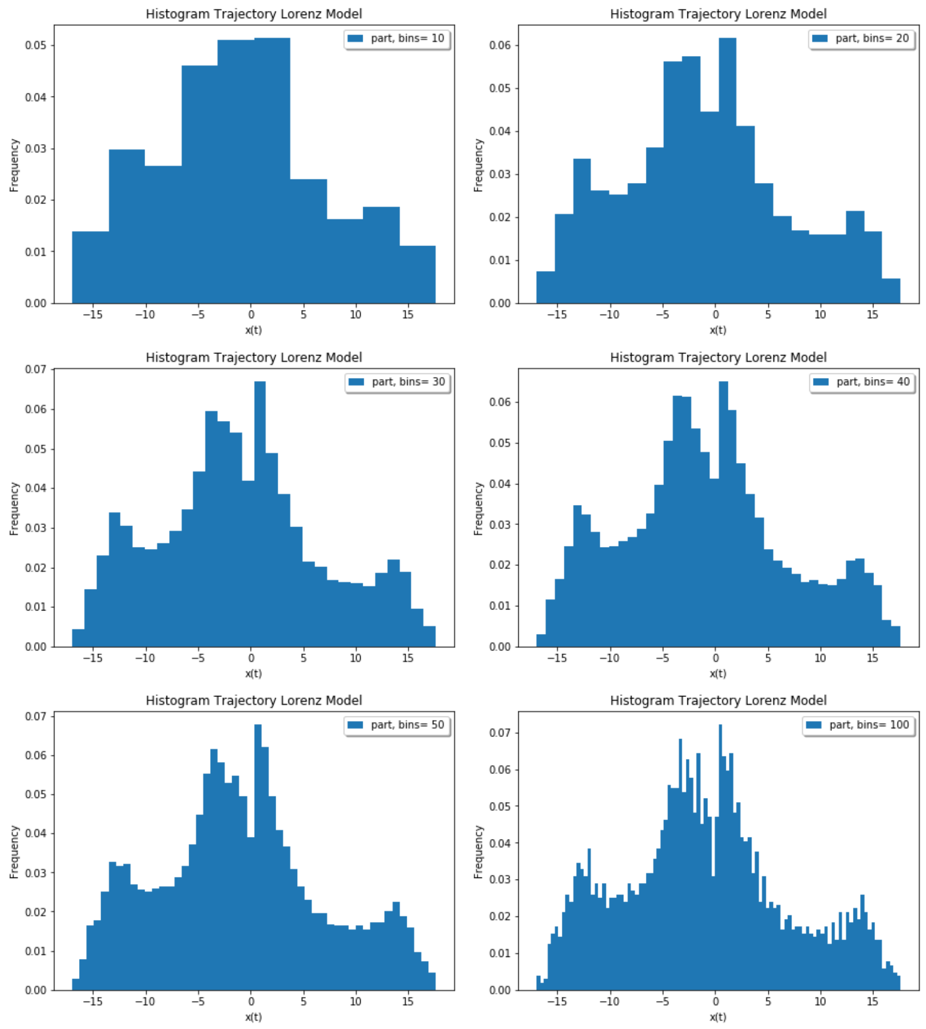






We can see that in the first two plots the histograms shows to have a higher standard deviation than the others, this happens because the probability to have two identical number decrease with precision. The last plot shows this problem, every number in the timeseries is unique: that's why we have histograms with a very small standard deviation, namely the entropy is the same for each trajectory.

1.1.2 Histograms



1.1.3 Infomatic implementation of Shannon Entropy

To evaluate the entropy we use this library [3].

```python
def shannon_entropy(time_series):
    """Return the Shannon Entropy of the sample data.
    Args:
        time_series: Vector or string of the sample data
    Returns:
        The Shannon Entropy as float value
    """

```

[3]https://github.com/nikdon/pyEntropy/blob/master/pyentrp/entropy.py

```python
    # Check if string
    if not isinstance(time_series, str):
        time_series = list(time_series)

    # Create a frequency data
    data_set = list(set(time_series))
    freq_list = []
    for entry in data_set:
        counter = 0.
        for i in time_series:
            if i == entry:
                counter += 1
        freq_list.append(float(counter) / len(time_series))

    # Shannon entropy
    ent = 0.0
    for freq in freq_list:
        ent += freq * np.log2(freq)
    ent = -ent
    return ent
```

1.2 Theory svd entropy

Single Value Decomposition is a factorization of a matrix. This method has been used in different applications. In the context of Deep Learning it is used in Principal Component Analysis [1936] that is useful in the dimensionality reduction. The key point behind this is to find the useful features that you need in the training process, namely the features that explain the variance of your signal. We can find online the documentation of the function svd-entropy [4].

$$S_{svd} = S(\mathbf{x}(t), order, delay) \tag{4}$$

We denote with **x** the dataset that we want to study.

$$\begin{aligned} \mathbf{x}(t_0) &= x_0, \\ \mathbf{x}(t) &= (x_1, x_2, x_3, \dots, x_N) \end{aligned} \tag{5}$$

This function creates a matrix Y with the dataset.

$$\mathbf{y}(i) = (x_i, x_{i+\text{delay}}, ..., x_{i+(\text{order}-1)\text{delay}}) \tag{6}$$

$$Y = \begin{pmatrix} \mathbf{y}(1) \\ \mathbf{y}(2) \\ . \\ . \\ . \\ \mathbf{y}(N-(ord-1)del) \end{pmatrix}$$

[4] https://raphaelvallat.com/entropy/build/html/generated/entropy.svd_entropy.html

If we use $order = 3$ and $delay = 1$ we obtain the following matrix[5]:

$$Y = \begin{pmatrix} x_1 & x_2 & x_3 \\ x_2 & x_3 & x_4 \\ . & . & . \\ . & . & . \\ . & . & . \\ . & x_{N-1} & x_N \end{pmatrix}$$

The next step is to write this matrix using the SVD decomposition.

$$Y = U\Sigma V^* \tag{7}$$

Where U is an unitary matrix, V is also an unitary matrix, and Σ is a diagonal matrix. Now we denotes the eigenvalues of the matrix Σ with σ_i. Now we can compute the average eigenvalues:

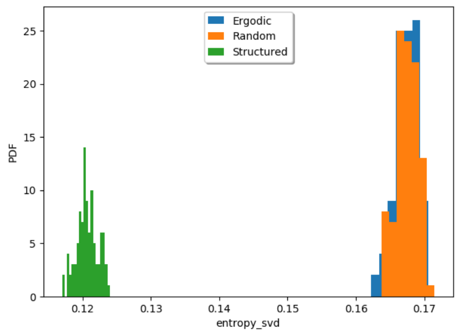
$$\bar{\sigma}_k = \frac{\sigma_k}{\sum_j^M \sigma_j} \tag{8}$$

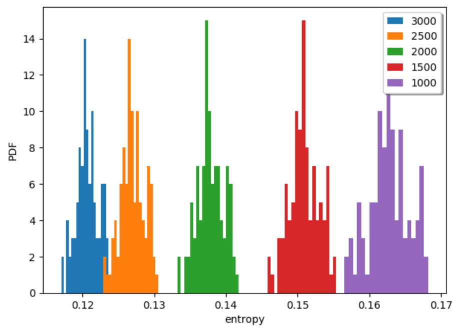
Where M is the number of eigenvalues.

After this we can compute the SVD Entropy:

$$S_{svd} = -\sum_k^M \bar{\sigma}_k log_2(\bar{\sigma}_k) \tag{9}$$

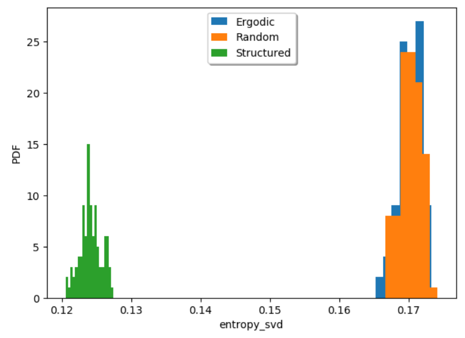
1.2.1 SVD with order 3

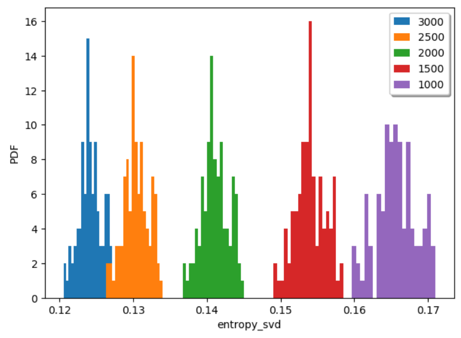




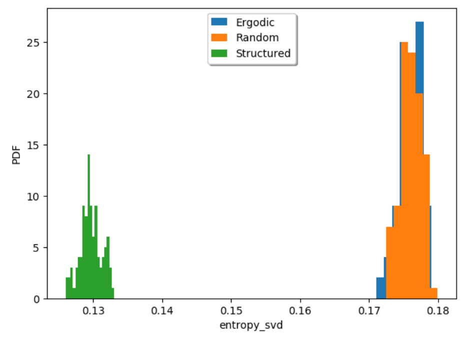
[5]To choose the delay and the order we can use the mutual information [1986] and to find the embedding dimension we can use the False Nearest Neighbours algorithm 2015, even if the qualitative behaviour of the results does not change even if we use different values of order and delay

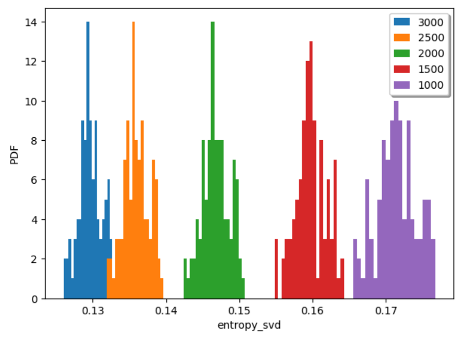
1.2.2 SVD with order 4



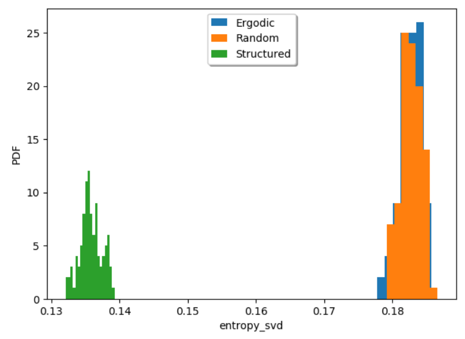


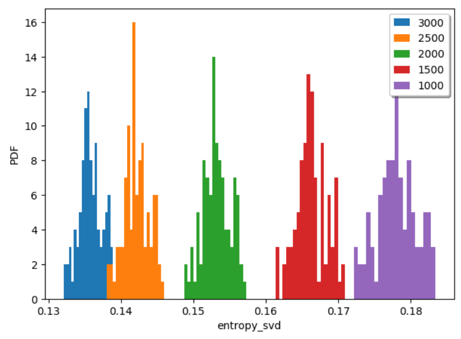
1.2.3 SVD with order 5



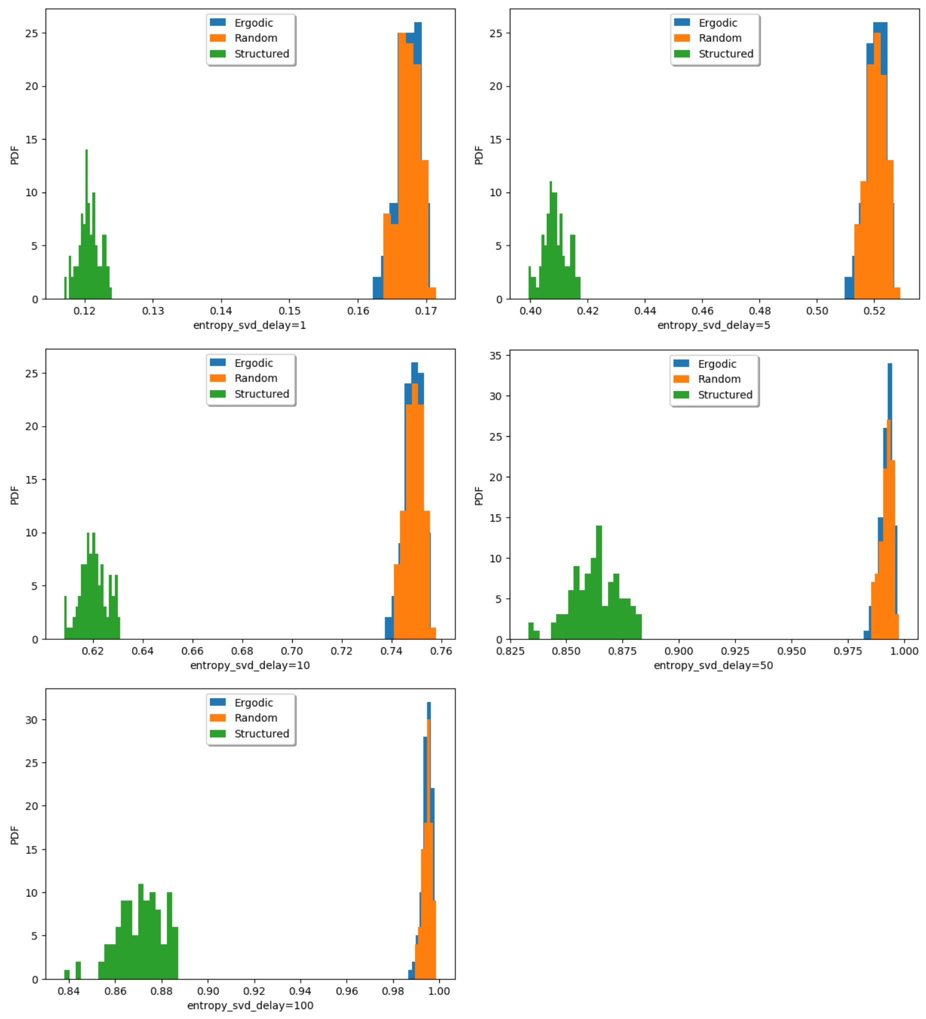


1.2.4 SVD with order 6





1.2.5 SVD time delay



1.3 Fisher Information

Fisher information is used together with svd decomposition.[2006] We can define the Fisher Information in the following way[6]:

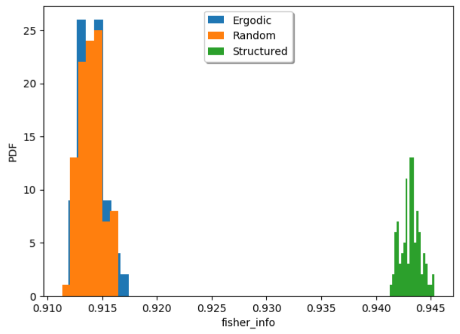
$$I_F = \sum_{k}^{M-1} \frac{(\bar{\sigma}_{k+1} - \bar{\sigma}_k)^2}{\bar{\sigma}_{k-1}} \tag{10}$$

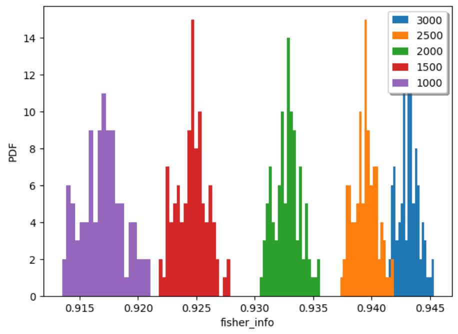
The quantities are exactly the same that we have defined in the previous section. The difference between this function and the svd entropy is that FI behaves

[6] for reference see here:`https://www.mdpi.com/1099-4300/23/11/1424`

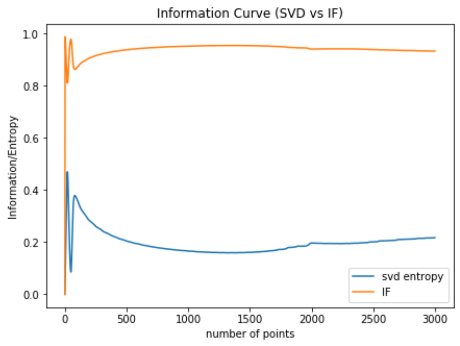
contrary to the entropy. We can see this in the plots.[2021]

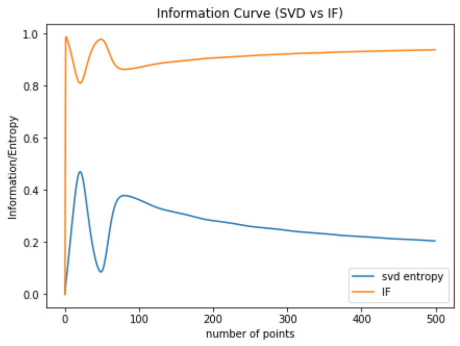
1.3.1 FISHER INFORMATION



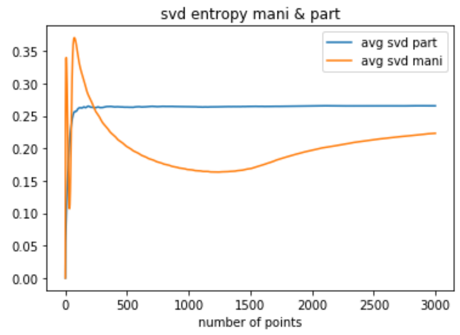


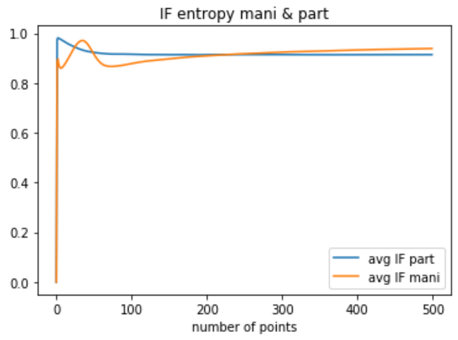
1.4 Growth of Information and Entropy



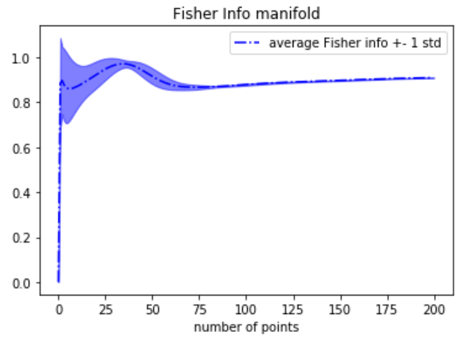


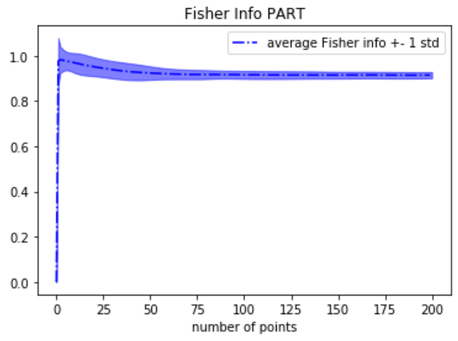
As we can see and as we can expect the plots behave in opposite ways. If we look at this by a statistical point of view, it could be useful to understand the average behaviour of the IF and SVD entropy, and also to understand the fluctuation that we have. We can try to compare the different shapes of the average amount of information (or entropy) of different trajectories when we increase the number of points.

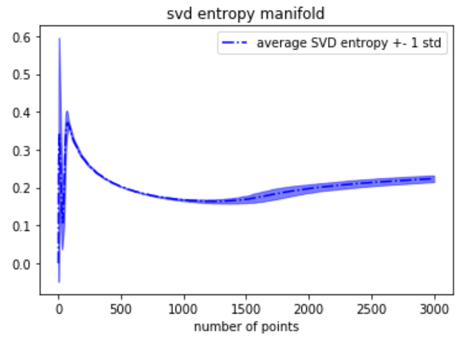


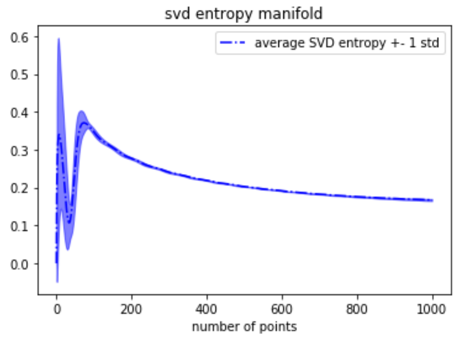


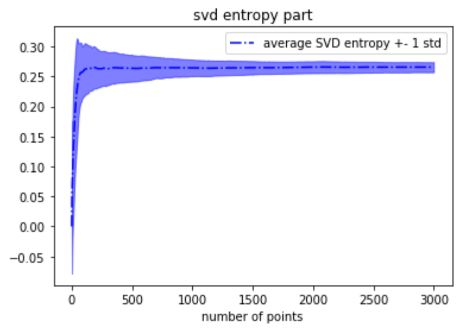
It is interesting to notice that around 300 number of points we can see that the amount of FI is bigger in the manifold data than in the part. We have the same behaviour happens with the svd entropy, but as we expect it is in the opposite direction. We can also notice by looking at these plots that the part trajectories seems to have a more stable variation of information than the manifold. In particular we see that at the very beginning of the curve, namely around 200 points the svd entropy of the manifold dataset increases a lot.











References

Fraser A. M., Swinney H. L., 1986, Phys. Rev. A, 33, 1134

Hotelling H., 1936, Biometrika, 28, 321

Mayer A. L., Pawlowski C. W., Cabezas H., 2006, Ecological Modelling, 195, 72

Raubitzek S., Neubauer T., 2021, Entropy, 23

Shannon C. E., 1948, The Bell System Technical Journal, 27, 379

Tang Y., Krakovská A., Mezeiová K., Budáčová H., 2015, Journal of Complex Systems, 2015, 932750