Predicting Time Series of Complex Systems

David Rojas
NASA
Hampton, VA
lempira@gmail.com

Lukas Kroc
Cornell University
Ithaca, NY
kroc@cs.cornell.edu

Marko Thaler
University of Ljubljana
Slovenia
marko.thaler@fs.uni-lj.si

1 Introduction

Scientists have a predilection for finding patterns and establishing relationships in the world around them. One way of doing this is simply by taking measurements of a quantifiable aspect (e.g. temperature, distance, voltage, etc) of the observable system. Such investigations were labor-intensive and highly susceptible to human and measurement error. Once the data was collected, the scientist had to infer a model of the phenomena from the data. The mathematical tools we had to describe phenomena were based on the assumption that the underlying behavior could be described by a line, or it could be made to if you squint hard enough. This is not true, however, about nonlinear systems, whose behavior, before the advent of nonlinear tools and methods, could be characterized only as erratic or chaotic. Technology has enabled us to gather copious amount of accurate data about an observable system. We are able to capture subtle changes in dynamics in short intervals of time. It has also motivated us to refine (or overhaul as the case may be) methods for characterizing the behavior. In this paper we explore one such method, time delay embedding, and its application to two chaotic systems.

The main task of time delay embedding (TDE) is to take a time series data set of observed scalar quantities and reconstruct the multidimensional phase-space of the observable system. We must first keep track of all the different spaces we are talking about. Suppose you have a system that you are observing that is governed by a set of recondite differential equations. The configuration space is where the solutions can exist and the solution manifold is where the solutions do exist. You, however, can only take measurements of one state variable, therefore the observable space is the one-dimensional set of points and reconstructed space is embedded observables. The central theorem in TDE, attributed to Takens and Mane [1], states that scalar quantities from a time series data (observable space) set can be embedded in multivariate phase-space (reconstructed-space), and with the proper choice of embedding dimension and time delay one obtains a reconstructed-space that is topologically similar to the

unobserved phase-space. Finding the appropriate choice for the parameters will be discussed later.

Suppose you have a finite time series $x_1, x_2, ..., x_n$, you can obtain the time delayed vector (lag vector) with the following expression;

$$y(t) = [x(t), x(t-\tau), x(t-2\tau), ..., x(t-(d-1)\tau]$$

where τ is the time delay parameter and d is the embedding dimension. This set of vectors is the reconstructed space.The ultimate goal of the method is to predict future data points. Section 3 compares several prediction methods as applied to the Lorenz system. We will see that the methods ability to predict depend on how far into the future you wish to predict and how many data points are available. Section 4 applies the method to predicting the trajectory of a double pendulum. We will see that implementation issues can also be sources of error.

2 Data Preparation

Sometimes the data we collect (or recieve) is noisy or we don't have enough of it. In these cases some preprocessing of the data will help make the data more tractable for the TDE method.

Filtering delay vectors We can apply a series transformations to the input time series in order to filter out some of the noise. In the method suggested by Sauer[2] of low pass embedding, we apply a discrete Fourier Transform to the raw time series, remove upper half of the frequency spectrum and transform the remaining data back to the time domain. This method is contingent upon the amount of data you have and the sampling rate, and thus influences how much of transformed data is not used. If the rate is high enough, then the lower half of the spectrum will give a good approximation of the signal with less noise. The purpose for low pass embedding is to increase the accuracy of the attractor representation by ensuring that high-frequency noise is filtered out. This method does not fair well, however, when the fourier spectrum is broadband and the noise is indistinguishable from the signal.

Interpolating the data points A common problem may be that you do not have enough data points due to undersampling or availability, in which case interpolation is a way to fill unsampled space. The reason for doing this is because you need nearest neighbor points to accurately calculate the embedding dimension as well as to predict dynamics near the present state vector. In general, there are many methods for interpolating data. You can interpolate between points or sets of points, or you can get a polynomial with autoregression and periodically sample the polynomial.

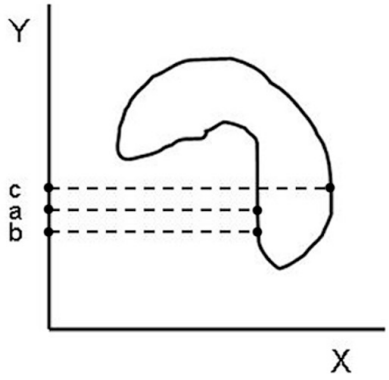

Figure 1: Point c is a false neighbor of a that is due to projecting onto the y axis

2.1 Finding the Right Dimension

When we take data from one state variable of a system we project the state-space dynamics onto a single axis. Problems can arise namely, overlapping of orbits. Since we know that this can not occur in a chaotic attractor, we must find an embedding dimension that will unfold the attractor so that no overlaps remain. We have seen that Takens comments on the topology of the reconstructed-space and the state-space, but it also produces the upper bound $d_E = 2n + 1$ (n is the dimension of the system) on the embedding dimension. It, however, says nothing about the minimum dimensions required to unfold the attractor. In a comparative study by Cellucci et al [7], global false nearest neighbors was shown to be the best method for finding the lower bound. Figure 1 False Nearest Neighbors

Global False Nearest Neighbors The basic idea behind this method is to remove all the overlaps in the orbits by testing whether the embedding dimension has removed all false neighbors. A false neighbor of a point in one that exists because of the projection onto a lower dimension, and is not due to the dynamics of attractor. A simple example can be seen in 1. Points B and C are equidistant to point A but only A and B are near each other in the trajectory. Point C is a false neighbor of A, and is produced by projection the attractor onto the y axis. Embedding into a higher dimension would reveal that C is indeed a false neighbor. The general procedure is as follows: Given the time delayed vector

$$y(t) = [x(t), x(t-\tau), x(t-2\tau), ..., x(t-(d-1)\tau)]$$

Compute the nearest neighbor NN vector

$$y^{NN}(t) = [x^{NN}(t), x^{NN}(t-\tau), x^{NN}(t-2\tau), ..., x^{NN}(t-(d-1)\tau)]$$

Compute the Euclidean distance for the vectors in d dimensions

$$[S_d(t)]^2 = \sum_{k=1}^{d} [x(t-(k-1)\tau) - x^{NN}(t-(k-1)\tau)]^2$$

Now, compute the Euclidean distance for the vectors in d+1 dimensions. The extra coordinates in the vectors at $x(t-d\tau)$ and $x^{NN}(t-d\tau)$

$$[S_{d+1}(t)]^2 = \sum_{k=1}^{d+1} [x(t-(k-1)\tau) - x^{NN}(t-(k-1)\tau)]^2$$

If we take the difference of the distance we get the expression, which is difference of the extra coordinates squared.

$$[x(t-d\tau) - x^{NN}(t-d\tau)]^2$$

Taking the ratio of the above expression with respect to the original distance $S_d(t)$, we get a nice test for false neighbors.

$$\frac{\left|x(t-(k-1)\tau) - x^{NN}(t-(k-1)\tau)\right|}{S_d(t)}$$

Note that this must be done for each point in the data set, and for sets with many points this can become computationally intensive.

2.2 Finding the Right Embedding Delay

If one is observing a system and takes photographs at periodic points in time, one can ascertain something about how the system is changing. If the snapshots are too frequent, the photographs will probably look indistinguishable from each other, providing little information about what has changed. However, if the time between the snapshots is too large, then one can only guess the dynamics that lead the system to such a state. One faces a similar problem when trying to find the appropriate time delay for a time series embedding. If the delay is too small then the reconstructed data vector will be highly correlated and we will learn little about how the system has changed. If the delay is too large, the reconstructed vector will be uncorrelated and thus statistically random. To find the correct delay we follow the procedure described by Abarbanel[3], and treat the chaotic dynamical system as an information source.

Average Mutual Information Autocorrelation is a tool that is frequently used to find patterns in time domain signals. It is a useful tool insomuch as the given time series has a nice discrete spectrum (i.e. linear). Spectra obtained from nonlinear sources are often broadband and complicated and thus are not described well by autocorrelation. An alternative method, known as average mutual information, quantifies the answer to the question "How much information can I get about one random variable by observing another?" Here, we

take "information" to mean that as formulated by Claude Shannon. [6] If one takes a set of measurements $A = a_i$ and another set $B = b_j$ then the amount of information (in bits) one obtains about b_j from observing a_i is given by the following expression:

$$log_2 \left[\frac{P_{AB}(a_i, b_j)}{P_A(a_i)P_B(b_j)} \right]$$

In the equation, $P_{AB}(a_i, b_j)$ is joint probability of observations a_i and b_j while $P_A(a_i)$ and $P_B(b_j)$ are individual probabilities. We get the probability densities from a histogram of the observations. From the equation we see that the mutual information of the observations is zero if they are independent because the quantity in brackets becomes 1. Instead of using two different sets of observations we can create the second set by using a time delayed copy of the first. For example, if we are given a set $x(t)$, we can create a second set $x(t + \tau)$. We now can calculate the average mutual information by multiplying the mutual information of a_i,b_j by the joint probability giving the equation:

$$I(\tau) = \sum_t P(x(t), x(t + \tau)) log_2 \left[\frac{P(x(t), x(t + \tau)}{P(x(t))P(x(t + \tau))} \right]$$

The above is an expression for average mutual information as a function of the time delay. As previously mentioned we expect the amount of mutual information to decrease as time delay increases, because the observations become statistically independent. To find the appropriate time delay to use for the embedding one would use the first minimum of $I(\tau)$ [4] [3]. By looking at the average mutual information function for the Lorentz system, we see that the minimum is reached at Timelag of 10 timesteps (0.1s). **??**

3 Prediction Algorithms

There are various methods and algorithms that can be used to predict behavior of a chaotic system [2]. Our main focus is on the Time Delayed Embedding technique as described in Section 1. This section describes in detail the algorithms used, shows their behavior and analyzes their relative strengths and weaknesses. This leads us to a proposed adaptive approach of time series prediction, which is also discussed in this section.

3.1 Overview of Methods Considered

The methods we considered can be described by their two main properties:

- what data points are used to predict new ones: *direct* (where every future data point is predicted using a fixed set of training datapoints) and *iterative* approach (where next data point is predicted using the training and previously predicted datapoints)

- how the new data points are predicted: *return map* (a function from a number of previous datapoints to a new one is used by locally interpolating the available data set) and *nearby trajectory tracking* (evolution of known trajectories near the to-be-predicted point is used, assuming the predicted point will be close to those trajectories even in the future)

The methods are described in more detail bellow. The input sequence is denoted by $x_1, \ldots, x_n$, and prediction function k steps ahead of n is denoted by P_k.

Direct Prediction In direct prediction, a fixed input data point(s) X_{fixed} (the last ones in the input sequence) are chosen to serve as a starting points for prediction of all future values. Size of X_{fixed} depends on dimension of the embedding. All other data points are used to properly interpolate P_k. So the prediction equation is

$$x_{n+k} = P_k(X_{fixed})$$

The function P_k must be constructed accordingly for every desired k. This construction depends on the used method and is described bellow.

Iterative Prediction This approach is complementary to the previous one: k is fixed to a suitable value (time-step of the prediction) and values of X are successively predicted one-by-one at time multiples of k. Data points X_t used to predict the value x_{t+k} are the last values (training or predicted) in the sequence. Their number depends on the dimension of the embedded space. The prediction equation is:

$$x_{t+k} = P_k(X_t)$$

Prediction Using Return Map Here, the prediction function P_k is of the form $R^{d-1} \to R$, where d is the dimension of the time-delayed state space. Every consecutive d-tuple of input points $x_{i_1}, \ldots, x_{i_d}$ is treated as a sample input-output assignment $(x_{i_1}, \ldots, x_{i_{d-1}}) \mapsto x_{i_d}$. The value of x_{i_d} is set according to the value of k: for $k = l\tau$, $x_{i_d} = x_{i_{d-1}+l}$. To evaluate $P_k(X)$ for some $X = (x_{i_1}, \ldots, x_{i_{d-1}})$, one finds a set $\{X_i\}$ of neighboring points to X for which the outcome $P_k(X_i)$ is known, and uses some interpolation technique to find $P_k(X)$ (simplest linear interpolation suffices).

Figure 2 shows a schematic chart of the situation, for $d = 3$.

Prediction Using Nearby Trajectory Tracking This method works using the assumption that trajectories that are nearby a point that we want to predict into the future, will also be close to it in the predicted future. Such trajectories can therefore be used to estimate the future position of the point. So $P_k(X)$ is constructed as follows: for a given point in the state space X, we first find some number of known trajectories that pass through the vicinity of X, and identify a point c_i on each that is closest to X. We then project all c_is into the future (on the known trajectories), and use the resulting values as y values

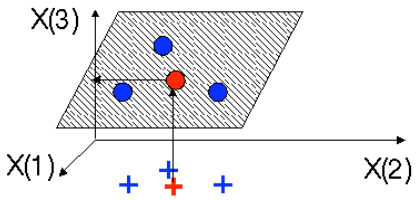

Figure 2: Schema of prediction using Return Map. Blue are known points, red is the one to be predicted

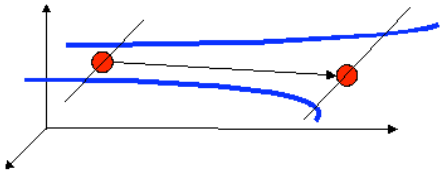

Figure 3: Schema of prediction using Nearby Trajectory Tracking.

for a local interpolation of the value of X in that future. Again, simple linear interpolation can be used. We found that more complicated techniques (like quadratic interpolation) can sometimes produce more accurate predictions but are also less robust.

This approach is depicted in Figure 3

3.2 Strengths and Weaknesses Compared

We compared the afore mentioned methods trying to find out if any one offered better performance than the others. We used the Lorentz System as a testbed for our prediction runs (with particular values of parameters $\sigma = 10$,$\rho = 28$ and $\beta = \frac{8}{3}$):

$$\begin{aligned} x' &= \sigma(y - x) \\ y' &= x(\rho - z) - y \\ z' &= xy - \beta z \end{aligned} \tag{1}$$

We mainly focused on comparing Return map with Iterative prediction and Nearby trajectory tracking with Direct prediction methods. A sample result of such comparison is shown in Figure 4.

In Figure 4, Nearby trajectory tracking seems to perform better than Return map approach. But the main observation was that this is not the case in general. The relative performance very much depends on the particular situation: how detailed information is known about the sequence (time between known datapoints), how far into the future is the prediction supposed to be made (prediction horizon), and on what portion of the chaotic attractor the desired predicted point lies. The more detailed information is known about the

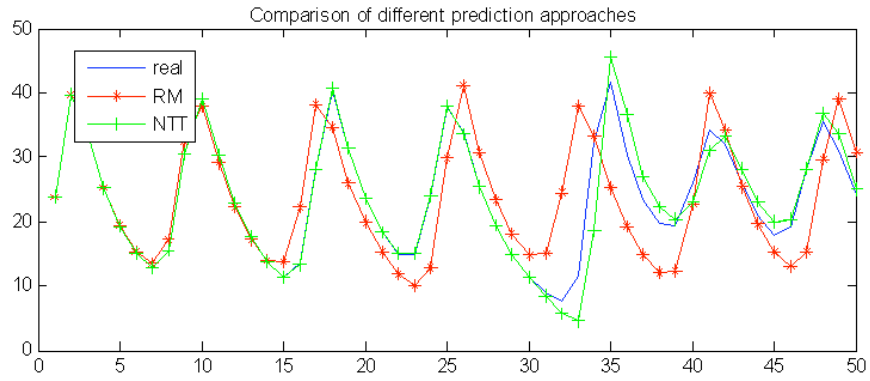

Figure 4: Comparison of different prediction methods on x variable of the Lorentz system. RM is Return map with Iterative prediction method, and NTT is Nearby trajectory racking with Direct prediction.

sequence (more datapoints), the better both methods get. Nearby trajectory tracking is also able to work with fewer datapoints in general. The length of the prediction period is an important consideration: Nearby trajectory tracking is performing better on short-term predictions, but may be totally off after a certain threshold in prediction horizon. This threshold depends on the topology of the attractor in the prediction horizon region: if we predict not-very-far into the future, the trajectories that are nearby the to-be-predicted one at the last-known point, are also nearby at the time we want to predict. But for far-ahead predictions, the nearby trajectories will eventually start going in very different directions (because we deal with a chaotic system), and the assumption of closeness of nearby trajectories is broken. The Return map approach may be better in such scenarios.

Based on the above observations we conjecture that an adaptive approach could be constructed, which would combine the strengths of the individual methods. The main idea is that the Nearby trajectory tracking performs better within a certain prediction time interval *and* we know when it stops performing well. This can be done by looking at how far the nearby trajectories get from each other while the prediction time increases, and stop using the method if they get too far. Then, few steps of the Return map methods can be used to get "around the difficult spot", and Nearby trajectory tracking resumed after that. Our preliminary study shows that this indeed results in better prediction in certain scenarios, but further work needs to be done in this respect.

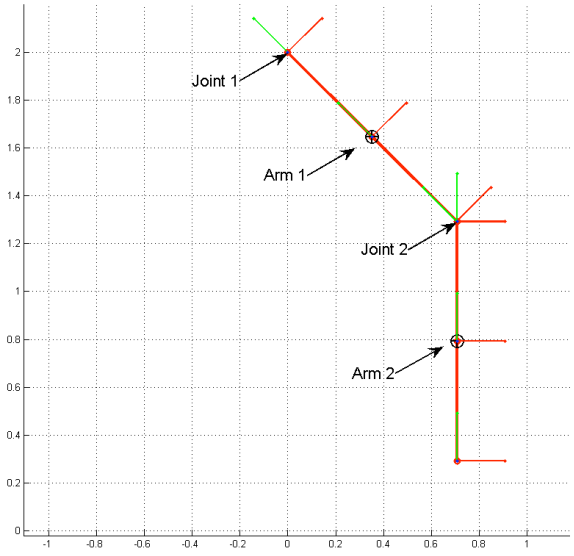

Figure 5: Double pendulum.

4 Double Pendulum

In order to more clearly understand how a physical chaotic system can behave, we have created a model of a frictionless double pendulum. Figure 5 shows the pendulum consisted of two rigid arms connected to each other by a frictionless joint (e.g. bearing). The first arm is connected to a wall by another frictionless joint while the second arm is moving freely.

This kind of system has four state space variables (angles θ_1, θ_2 and angular velocities $\dot{\theta_1}$, $\dot{\theta_2}$ of the first and second arm, respectively). State space variables are connected to each other by coupled second order differential equations [5] that can be solved numerically (e.g. fourth order Runge-Kutta method).

Because double pendulum is a chaotic system, small changes in initial conditions produce highly different behavior over time. State space topology of the system also depends heavily on the selection of initial conditions (see Figure 6). Small initial angles produce more periodic trajectories in the state space, while larger initial angles produce trajectories that diverge more quickly.

4.1 Prediction results

Based on a finite set of state space variables time series (250 seconds, time step = 0.005s), we wanted to analyze the degree of prediction accuracy for a desired prediction horizon (2.5s). On the other hand we also wanted to analyze how different ways of observable encoding affects the prediction accuracy.

Prediction method used was a direct Nearby trajectory tracking in state

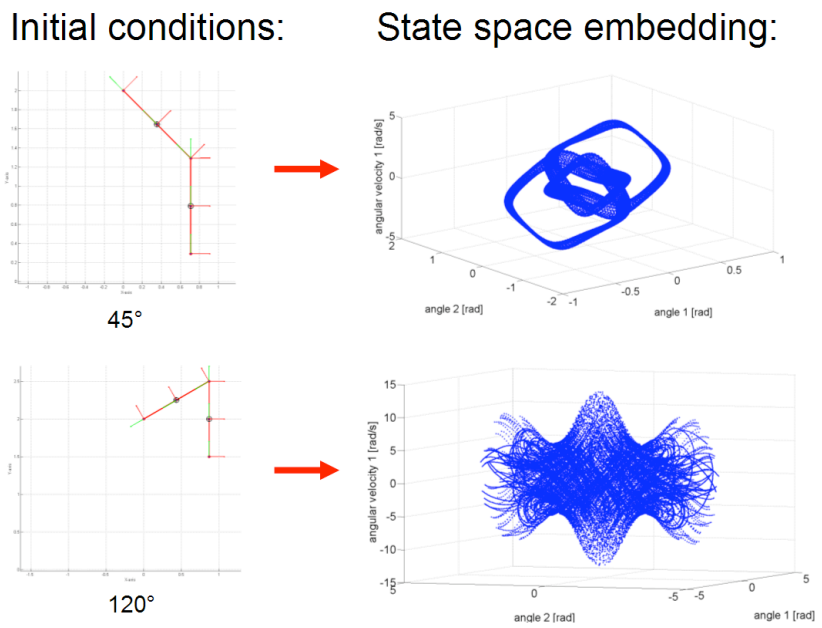

Figure 6: Influence of initial conditions on the state space trajectories.

space. Prediction accuracy was estimated for 300 prediction runs with the same initial conditions (see Figure 6, bottom pictures) and different starting points of prediction. Prediction error of an individual run was defined as the average square distance between predicted and actual angles θ_1 and θ_2 in the prediction horizon.

Based on the prediction accuracies of individual runs we formed a prediction accuracy histogram (see Figure 7). The histogram shows the prediction accuracy is wide spread across a large set of values. This indicates a high influence of starting point on the prediction accuracy. In our case a very limited (e.g. short) number of training data was available which translated to a very sparsely populated trajectory state space. When starting point dictated higher average divergence of nearby trajectory paths in the prediction horizon, higher prediction errors were the consequence.

To better understand the prediction error growth over time Figures 8 and 9 show an example of the predicted and actual angle θ_1 and angular velocity $\dot{\theta}_1$ of the pendulum's first joint. The agreement between predicted and actual time series is very good for the first 800 time steps (4s). After that point the error starts to grow substantially, especially for the angular velocity (Figure 9), and on average never decreases.

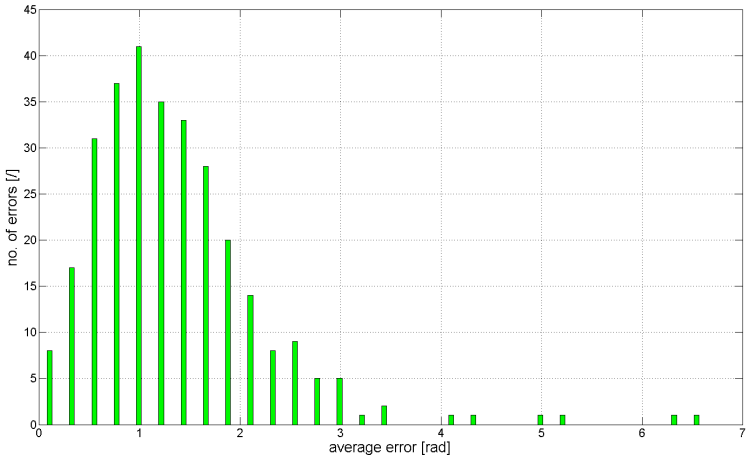

Figure 7: Prediction error histogram.

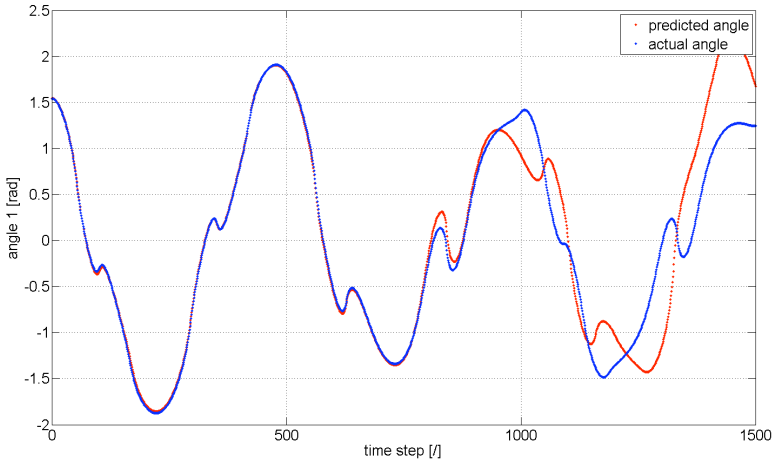

Figure 8: Comparison of predicted and actual angle θ_1.

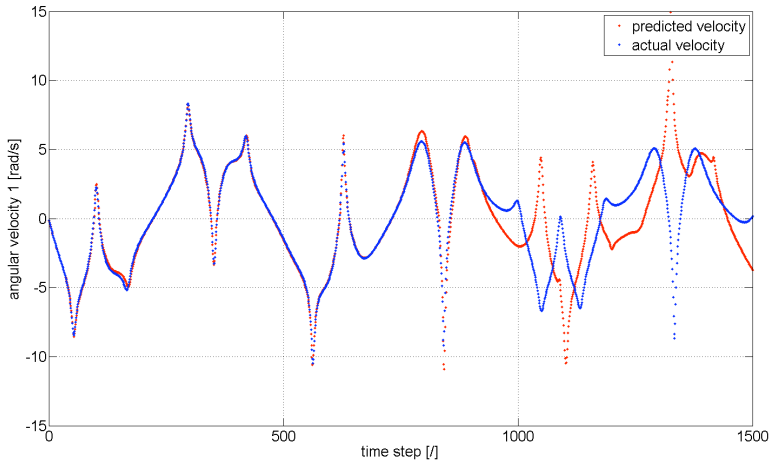

Figure 9: Comparison of predicted and actual angular velocity $\dot{\theta}_1$.

4.2 Observable encoding

Joint angles of the pendulum can be encoded in different ways. This affects the prediction accuracy. We compared relative, absolute and sine-cosine encoding of angles. Relative encoding uses relative angle of one pendulum arm against another. This can sometimes simplify the notation but the prediction error of the first angle is directly transmitted to the second one. Absolute encoding can overcome this deficiency. Both the absolute and relative encoding generate discontinuities in the state space trajectory when individual arms of the pendulum rotate for a full circle. Encoding angles as $0°$ to $360°$ or $-180°$ to $180°$ results in a discontinuity at $360°$ and $0°$ or $-180°$ and $180°$. This type of encoding also causes a discontinuity in state space trajectories which can become a big source of prediction error, especially if one predicts the future state of the pendulum with the use of neighbor trajectories. The prediction results of this method are determined with the use of weighting average. Though, in the case when one neighboring trajectory goes through the discontinuity and the other does not, the result lies somewhere in the middle of the state space and not at its boundaries. One can avoid this discontinuity with the use of sine and cosine encoding.

To statistically analyze the prediction errors of different types of encoding we have used the same 300 run prediction setup as described previously but with different angle encodings. Prediction error distributions with different types of encoding are presented in Figure 10.

In Figure 10 we can see the absolute encoding produces on average lower prediction errors than the relative encoding. Somewhat surprising is the result of sine-cosine encoding which has the highest average error. Though we have

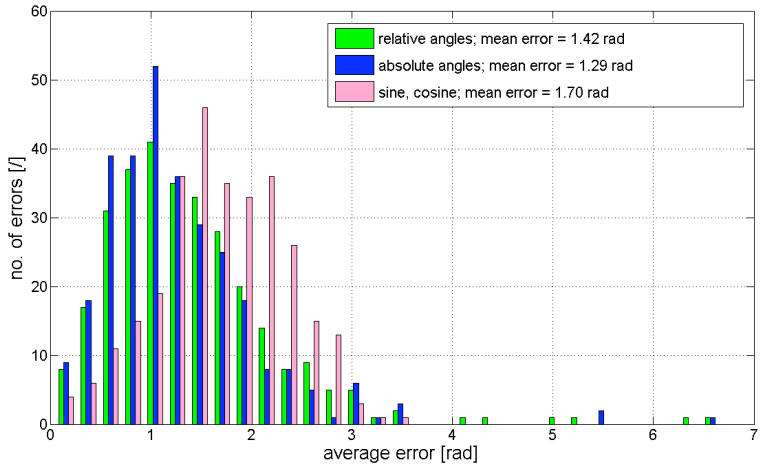

Figure 10: Encoding comparison based on the prediction error histogram.

eliminated the problem of trajectory discontinuities in the state space, we have introduced a new source of prediction error. Through the splitting of one variable in two (e.g. sine and cosine) and not taking into account the connection between both (Eq. 2) we have increased the probability for errors. Sine and cosine are connected to each other through the equation:

$$\sin^2(x) + \cos^2(x) = 1 \tag{2}$$

If we do not take this constraint into account a new source of error is generated. In our case we did not take any prior knowledge (e.g. constraint optimization) into account except for the training time series. This created an additional source of error (e.g. complex angles) that has outweighed all the potential benefits of the proposed solution.

5 Summary

In our work we have looked at ways of predicting chaotic time series using a Time Delayed Embedding technique. We identified various approaches to doing so, and discussed their relative advantages and disadvantages. Two chaotic systems (Lorentz and Double pendulum) were used in the study.

The Lorentz system was used to perform initial comparisons between the methods considered. The Double pendulum example has showed that small number of state space training points can not produce accurate predictions of future states beyond some short prediction horizon. The prediction accuracy is highly influenced through the nearest neighbor trajectory topology in the prediction horizon region of the state space. Through the use of prior knowledge

about the modeled system, intelligent variable encoding can reduce the average prediction error substantially.

The main observation of our study was that there is no single best approach for all scenarios. One approach works best in some situation, while it breaks down in another. Thus we proposed an adaptive strategy which would be able to detect regions where each approach performs best and switch between methods appropriately. More work needs to be done in this direction to fully explore the possibilities of this strategy.

References

[1] F Taken. *Lecture Notes in Math,* 898:366-381, 1981.

[2] Weigend, Andreas S, Gershenfeld, Neil A: *Time Series Prediction: Forecasting the Future and Understanding the Past,* Proceedings Volume 15, Santa Fe Institute Studies in the Sciences of Complexity, 1993.

[3] Abarbanel, Henry D.I. *Analysis of observed chaotic data,* New York: Springer-Verlag, 1996.

[4] A.M. Fraser Physica D 34, 391, 1989.

[5] `http://scienceworld.wolfram.com/physics/DoublePendulum.html`

[6] C.E. Shannon. *A Mathematical Theory of Communication,* Bell System Tech. 1948

[7] C.J. Cellucci *et al. Phys Rev E,* 67:066210-1-13, 2003