# Classification of Chaotic Behaviors in Jerky Dynamical Systems

**Tianyi Wang**
*Wolfram High School Summer Camp*
*Bentley University*
*175 Forest Street*
*Waltham, MA 02452, USA*

Differential equations are widely used to model systems that change over time, some of which exhibit chaotic behaviors. This paper proposes two new methods to classify these behaviors that are utilized by a supervised machine learning algorithm. Dissipative chaotic systems, in contrast to conservative chaotic systems, seem to follow a certain visual pattern. Also, the machine learning program written in the Wolfram Language is utilized to classify chaotic behavior with an accuracy around 99.1±1.1%.

*Keywords*: chaotic systems; jerky dynamics; differential equations; dynamical systems; phase space; supervised machine learning

## 1. Introduction

Differential equations that involve third derivatives, also known as "jerks," are called third-order differential equations. It has been shown that a third-order differential equation can always be written in a system of three differential equations of the first order [1, pp. 62]. If such a system has time as its independent variable, then it is called a jerky dynamical system. The main goal of this paper is to classify such systems and identify global chaotic behaviors of such systems. In this paper, the classification of jerky dynamical systems is based on the behavior of the trajectories in the phase space.

A jerky dynamical system can be visualized in a phase space, a three-dimensional vector field that shows all possible states of the system. A trajectory of the vector field can be drawn to indicate the evolution of the system with certain initial conditions. Trajectories of different systems can have different behaviors. If the system is convergent, the trajectory gets arbitrarily close to a certain point in the phase space. If the system is divergent, the trajectory can escape to infinity or can be chaotic (meaning initially close points on the trajectory can quickly evolve into very different states or are extremely sensitive to initial conditions), getting attracted to a certain region. Although there are cases where trajectories exhibit local chaotic

https://doi.org/10.25088/ComplexSystems.30.1.93

behavior before they escape, this paper will focus on the systems that exhibit global chaotic behavior. Since the subjects of interest are systems that exhibit global chaotic behaviors, these systems must be divergent, but the trajectories do not escape to infinity.

Furthermore, any dynamical systems, including chaotic ones, can be classified into two categories: conservative or dissipative. Conservative systems are systems in which the phase space volume is preserved. In contrast, phase space volume contracts along a trajectory in a dissipative system [2, p. 540]. Therefore, it is possible to categorize chaotic jerky dynamical systems more specifically into conservative systems or dissipative systems. However, due to the complex behavior of chaotic systems in the phase space, it is very hard for computers to classify them.

In this paper, two methods are proposed to detect systems that are potentially chaotic, based on Cauchy's completeness theorem (Section 2). Both methods are then utilized in a supervised machine learning function to distinguish conservative and dissipative systems. We discovered a prevailing visual pattern for dissipative systems (Section 3), and the classifiers we trained reach more than 99% accuracy (Section 4). These classifiers can then be used to classify any dynamical systems with relatively high accuracy (Section 5).

## 2. Theoretical Background and Methodology

This section introduces the theoretical methods that we use to identify whether a system is convergent or divergent and whether a system is conservative or dissipative. The motivation of the methods we use to identify convergent systems comes from Cauchy's completeness theorem from literature [3, Chapter 3, pp. 52–53], which we will first introduce here.

**Theorem 1 (Cauchy's Completeness Theorem).** In any metric space $X$, every convergent sequence is a Cauchy sequence.

With Theorem 1, we can then derive the condition for a trajectory in phase space to converge. We first make a rigorous definition for the notation we will be using.

**Definition 1.** Consider a system of three differential equations:

$$\dot{x} = f_1(x, y, z)$$
$$\dot{y} = f_2(x, y, z)$$
$$\dot{z} = f_3(x, y, z)$$

where $f_i : \mathbb{R}^3 \to \mathbb{R}$ are smooth, with $t$ being the independent variable. Define $\theta : \mathbb{R} \to \mathbb{R}^3$ as a function that takes some input $t$ and gives the position of the trajectory in $\mathbb{R}^3$ at $t$. Assume an arbitrary fixed incre-

ment in time $\Delta t > 0$, $t_{i+1} = t_i + \Delta t$. Let $\{t_1, t_2, \ldots\}$ be any sequence such that $\lim_{i\to\infty} t_i = \infty$. Define $d(\theta(t_{i+1}), \theta(t_i)) = d(\theta_{i+1}, \theta_i) = d_i$ to be the Euclidean distance between points $\theta(t_i)$ and $\theta(t_{i+1})$.

Then we define what it means for a trajectory in phase space to converge.

**Definition 2**. The trajectory of a system $\theta(t)$ in a phase space converges to a point $P$ if for all $\epsilon > 0$, there exists $T_\epsilon$ such that $\theta(t)$ is within $\epsilon$ of $P$ for all $t > T_\epsilon$.

With the given definitions, together with Theorem 1, we now derive two corollaries that are later utilized in generating visual data associated with each system. They are also the guidelines we will use in labeling training data for the supervised machine learning system.

**Corollary 1**. The trajectory of a system $\theta(t)$ converges to $P$ if $\lim_{i\to\infty} d_i = 0$.

*Proof*. If $\{\theta_i\}$ converges in phase space, then according to Theorem 1, $\{\theta_i\}$ is a Cauchy sequence. From the definition of a Cauchy sequence, we know that $d(\theta_i, \theta_j) < \epsilon$ for $t_i, t_j > N$ for some $N$. From Definition 2, we know that if we choose $N = T_{\epsilon/2}$, then $d(\theta_i, \theta_j) < \epsilon$ holds since $d(\theta_i, \theta_j) \le d(\theta_i, p) + d(p, \theta_j) < 2(\epsilon/2) = \epsilon$ holds in $\mathcal{X}$. Finally, since $t_{i+1} > t_i > T_{\epsilon/2}$, then $d_i = d(\theta_{i+1}, \theta_i) < \epsilon$ for all $t > T_{\epsilon/2}$. This implies $\lim_{i\to\infty} d_i = 0$, which concludes the proof. □

**Corollary 2**. Consider the infinite sequence of Euclidean distance of a system $D = \{d_1, d_2, \ldots, d_n, \ldots\}$. Define $\Delta_i = d_{i+1} - d_i$. Let the distance difference sequence $\delta = \{\Delta_1, \Delta_2, \ldots, \Delta_n, \ldots\}$. The system converges if the corresponding sequence $\delta$ is convergent and diverges if $\delta$ is divergent.

Therefore, to test whether the system is convergent or divergent, take an infinite number of sample points on the trajectory in the phase space that corresponds to successive increase in time by a constant increment. Next, calculate the Euclidean distances between two successive points. These distances are elements of an infinite sequence $D = \{d_1, d_2, \ldots, d_n, \ldots\}$. If $D$ is convergent, then the system is also convergent. If $D$ is divergent, then the system is also divergent. Figure 1 illustrates this method.

This method, which we will be using, is a direct result from Corollary 1. We call this method the Euclidean distance test (EDT). Figure 2 is an example of what a convergent and a divergent system may look like.

https://doi.org/10.25088/ComplexSystems.30.1.93

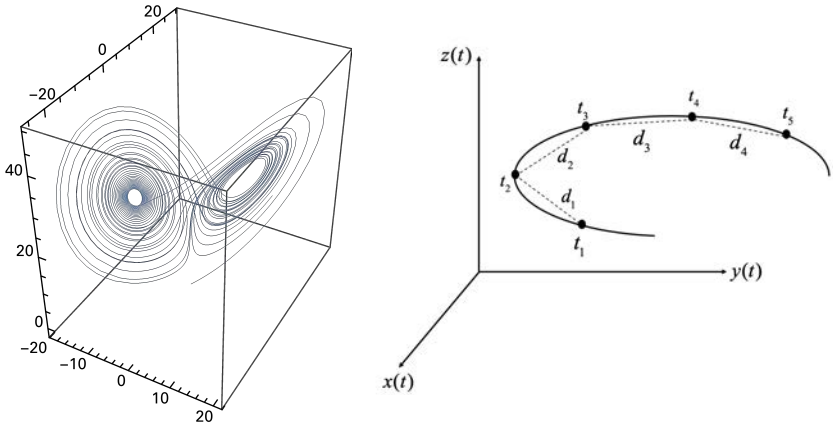



**Figure 1.** (a) An example of a chaotic trajectory in a phase space and (b) an illustration of the technique of taking sample points in phase space.

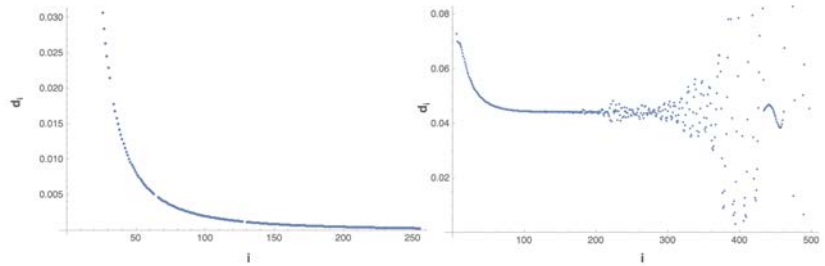



**Figure 2.** Plots of two Euclidean distance sequences. (a) The sequence converges, meaning the corresponding system converges in the phase space. (b) The sequence diverges, meaning the corresponding system diverges in the phase space.

From Corollary 2, we know that we can also generate the sequence $\delta = \{\Delta_1, \Delta_2, \ldots, \Delta_n, \ldots\}$ and test its convergence. This method tests whether the successive difference of Euclidean distances converges. We call it the distance difference test (DDT).

After being able to identify and differentiate convergent and divergent systems, we now introduce a theorem from literature that will help us identify whether a chaotic system is conservative or dissipative [4, Chapter 6, p. 158].

**Theorem 2.** For a dynamical system

$$\begin{cases} \dot{x} = f_1(x, y, z) \\ \dot{y} = f_2(x, y, z) \\ \dot{z} = f_3(x, y, z), \end{cases}$$

where $f_i : \mathbb{R}^3 \to \mathbb{R}$ are differentiable functions, the system is conservative if

$$\frac{\partial f_1}{\partial x} + \frac{\partial f_2}{\partial y} + \frac{\partial f_3}{\partial z} = 0.$$

Theorem 2 will serve as a theoretical guideline that helps us label the category of a system when labeling the training data for the classifier. There will be exceptions to Theorem 2, which we will mention in the next section.

## 3. Discoveries of Visual Patterns Associated with Conservative and Dissipative Systems.

In this section we present the computational methods we use to generate visual data associated with conservative and dissipative systems.

Since both EDT and DDT are valid indicators of whether the system is convergent or divergent, they will be used to train the classifier function later in this paper. The results will be compared to see which test is more effective. The following Wolfram Language function calculates the Euclidean distance of successive points.

```
detectStableFixedEnding[f_, df_, ddf_, (*lasttime_:1000,*)t0_] :=
    Module[{lengths},
        length = EuclideanDistance[{f, df, ddf} /. t → t0, {f, df, ddf} /. t → t0 - 1]];
```

A sequence is generated with the function detectStableFixedEnding. Because it is impossible to generate a sequence with infinitely many terms on a computer, we compute one thousand terms to gain relatively accurate results without going beyond the computational power. The function vectorl calculates the sequence Euclidean distance, and disdiffper calculates the difference in Euclidean distances divided by the distance of the sample points from the origin of the coordinate system, considering that points very far from the origin can have a greater change in distance.

```
vectorl = Table[detectStableFixedEnding[y, Dy, DDy, n], {n, 1, 1000, 1}];
disdiffper = Table[(Abs[vectorl[[n + 1]] - vectorl[[n]]]) / Norm[{y, Dy, DDy} /. t → n],
    {n, 1, Length[vectorl] - 1}];
```

https://doi.org/10.25088/ComplexSystems.30.1.93

By solving differential equations numerically and calculating the Euclidean distances sequence (or distance differences sequence), it becomes possible to visualize the behavior of trajectories in the phase space. For convergent trajectories (Figure 2(a)), the sequences go to zero. For trajectories that escape, the sequences go to infinity. Some sequences that oscillate in finite ranges (Figure 2(b)) may exhibit global chaotic behaviors. Therefore, their chaotic behavior can be confirmed by the phase space plot.

Many sequences of Euclidean distance for different systems were generated in an attempt to search for visual patterns. The sequence is visualized by the Wolfram Language function ListPlot. A series of comparisons between conservative and dissipative systems suggests the following observational result: conservative chaotic systems exhibit diverse behaviors that are hard to predict, whereas dissipative chaotic systems usually exhibit behavior that follows a certain visual pattern. A description of such a pattern would be "scattering," as demonstrated in Figure 3. The ListPlot function visualizes dissipative jerky systems that were discovered before [5].

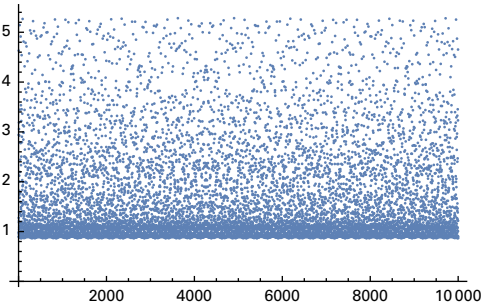



**Figure 3.** Classical behavior of a dissipative system. The sequence of Euclidean distance points creates a scattering visual pattern.

Other dissipative chaotic systems exhibit similar behavior. Figure 4 shows three more examples generated by EDT using systems from [5]. Interestingly, they exhibit some lower bounds.

This pattern also seems to be true for dissipative chaotic systems of higher dimension, for example, four-dimensional systems. Therefore, this visual pattern is likely to be an indication that the corresponding system is dissipative. Conservative chaotic systems, on the other hand, exhibit various behaviors that are difficult to classify (Figures 5 and 6).

Also, a chaotic system can exhibit a visual pattern that is extremely similar to the behavior of a dissipative chaotic system. A well-known

system is the double-pendulum system. However, since it is a two-dimensional motion, using a two-dimensional coordinate system is sufficient to describe its motion. Therefore, the Euclidean distance needs to change from three-dimensional distance to two-dimensional distance. It is important to point out that reducing one dimension will still produce similar results, since the phase space is only projected from a three-dimensional space onto a two-dimensional space.

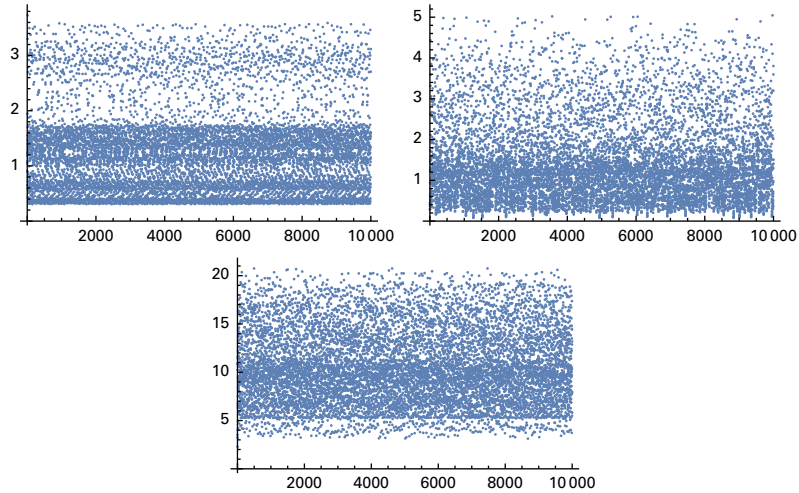



**Figure 4.** Three more examples of dissipative behavior generated by EDT using systems from [5].

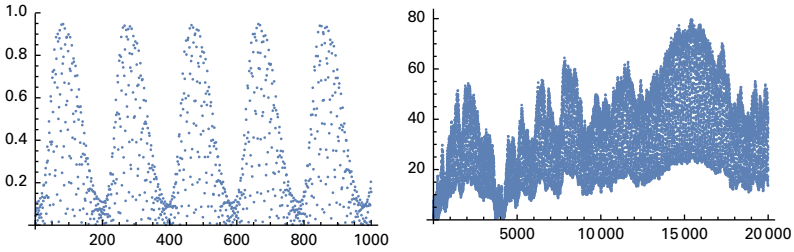



**Figure 5.** Euclidean distance plots of two conservative chaotic examples. (a) corresponds to system $\dddot{x} + 14\dot{x} = \sin(x) - 5x^3$ with initial conditions $x(1) = \dot{x}(1) = \ddot{x}(1) = 0.2$. (b) corresponds to system $\dddot{x} + 0.1\dot{x} + \sin(x) = 0.5\cos(t)$ with initial conditions $x(0) = \dot{x}(0) = \ddot{x}(0) = 0.3$.

https://doi.org/10.25088/ComplexSystems.30.1.93

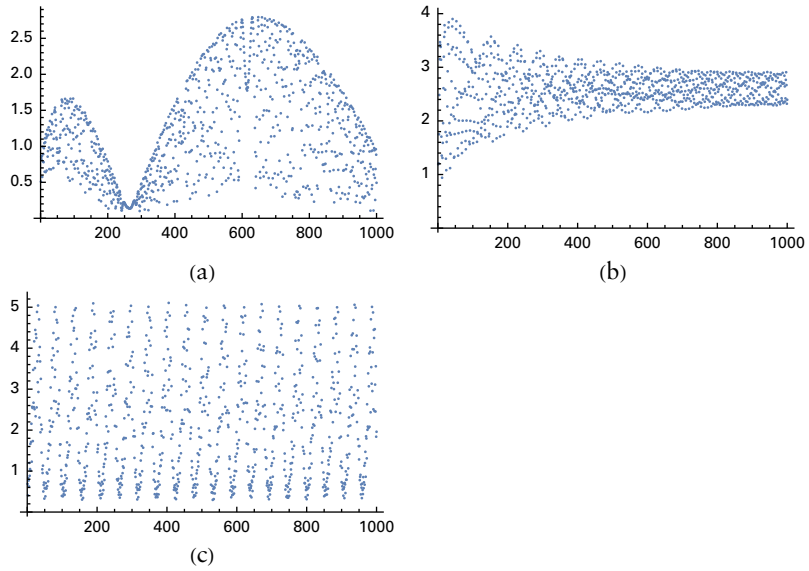



**Figure 6.** Three more conservative systems. (a) is the plot of the system from [6], (b) is the system from [7] and (c) is the system from [4].

One very important exception to Theorem 2 is the Nose–Hoover system, a system used to describe particles interacting with an external system in a way that the energy is conserved, which is defined as follows:

$$\begin{cases} \dot{x} = y \\ \dot{y} = yz - x \\ \dot{z} = 3 - y^2. \end{cases} \tag{1}$$

The Nose–Hoover system has divergence $z$, suggesting that the system is dissipative. However, it has been shown computationally that the Nose–Hoover system is actually conservative [8, p. 2]. Due to its nonzero divergence, the Nose–Hoover system displays "scattering" behavior similar to most dissipative systems. Several modifications of equation (1) are done to make the Nose–Hoover system dissipative [8, pp. 2, 5]. Nonintuitively, these dissipative versions of the Nose–Hoover system exhibit visual patterns that are completely different from the scattering behavior. These unconventional dissipative systems have the following forms:

$$\begin{cases} \dot{x} = y - bx \\ \dot{y} = yz - x \\ \dot{z} = 3 - y^2 \end{cases} \tag{2}$$

$$\begin{cases} \dot{x} = y \\ \dot{y} = yz - x \\ \dot{z} = 3 - y^2 - bz \end{cases} \tag{3}$$

where $b \in \mathbb{R}$. Each of equations (2) and (3) exhibits completely different behavior depending on the value of $b$ (Figures 7 and 8), which suggests that not all dissipative systems confirm the scattering pattern.

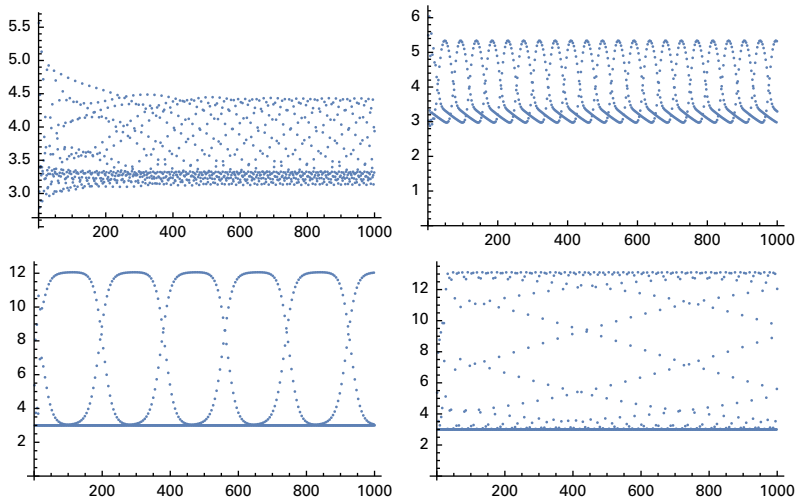



**Figure 7.** Euclidean distance sequence plot of equation (2) with different values of $b$. From left to right, $b$ = 1, 1.5, 4.5, 5, respectively.

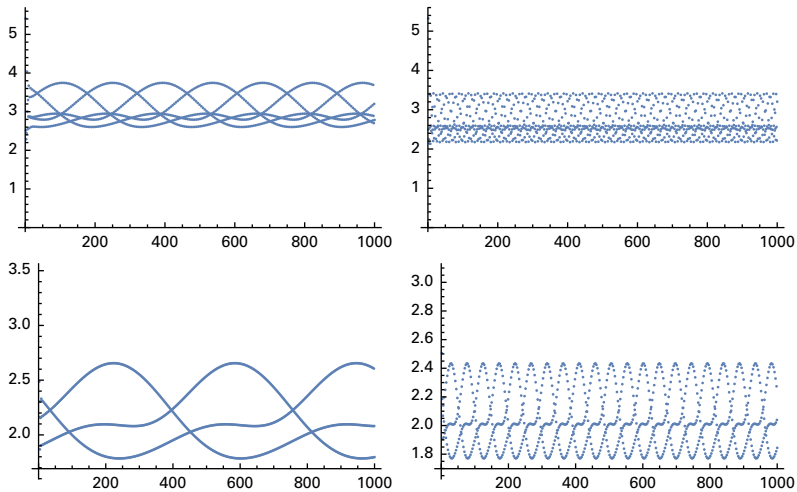



**Figure 8.** Euclidean distance sequence plot of equation (3) with different values of $b$. From left to right, $b$ = 1, 1.5, 3, 4, respectively.

https://doi.org/10.25088/ComplexSystems.30.1.93

In conclusion, in most cases, if a scatter plot of a Euclidean distance sequence exhibits a scattering pattern, then the corresponding system is likely to be dissipative. For plots that have more diverse behavior, the corresponding system is likely to be conservative.

## 4. Supervised Machine Learning Classifier for Conservative and Dissipative Chaotic Systems

The Nose–Hoover system is a very important exception, suggesting that conservative chaotic systems can behave in a way that is similar to dissipative systems. And, though less frequently, dissipative systems can have more complicated behavior than just the usual scattering visual pattern. Though it is reasonable to believe that most dissipative systems confirm the scattering pattern, such exceptions make it difficult to conclusively identify a dissipative system just by looking at the ListPlot graph. Also, since conservative systems exhibit much more diverse behaviors, it is even harder to identify them. Therefore, it would appear that developing a machine learning algorithm to classify chaotic dynamical systems is more convenient and more accurate.

The Wolfram Language function Classify can deal with this task for a relatively small learning sample size. Since the learning sample size in this research is typically no more than 1000, it is sufficient to use Classify without constructing a neural network. The first step is to start from the chaotic conservative and dissipative systems that were already discovered. These systems would be the initial samples. Then, by changing the initial conditions of these systems, we can get a different sample of the same kind (conservative or dissipative). Changing initial conditions by different amounts will serve to obtain more samples for the training set of the classifier function.

The second step is replication, a process in which the initial samples are used to generate more chaotic systems of the same kinds. The idea is to change the initial conditions of the samples by a little bit. Chaotic systems are extremely sensitive to initial conditions, which means a slight change in initial conditions might totally destroy a chaotic system because the numerical values that cause a system to be chaotic are very rare. Numerical search for second-order polynomial jerk functions has been done before [2, p. 538]. The result shows that for randomly chosen initial conditions, the most common situation is that the trajectory escapes to infinity, and the second most common case is that the trajectory converges to a certain point. The odds of finding a chaotic solution are around one in $10^4$. The numerical method for replicating samples in this research is the following: given a chaotic system with initial conditions $(x_0, y_0, z_0)$, or sometimes $(\dot{x}_0, \dot{y}_0, \dot{z}_0)$, change the initial conditions by $\epsilon$ amount, where $\epsilon \ll 1$

but still several orders of magnitude bigger than the smallest machine precision. It was experimentally found that for all the samples chosen from different references for this study [1–9], $\epsilon = 2 \cdot 10^{-10}$ is a safe range in which the chaotic systems remain chaotic. Therefore, we chose step size $\Delta = 10^{-10}$, so that for three initial conditions $(x_0, y_0, z_0)$, each initial condition can increase by an amount of 0, $10^{-10}$ or $2 \cdot 10^{-10}$. Therefore, for a single chaotic sample, 27 samples of the same kind are replicated. The following code demonstrates one example of how this replication process is achieved, which generates a numerical list of Euclidean distances.

```
Tablefunc = Table[ func = NDSolveValue[{
 x'[t] == y[t] – 5 x[t],
 y'[t] == y[t] * z[t] – x[t],
 z'[t] == 3 – y[t] ^ 2,
 x[0] == 0 + e1,
 y[0] == 5 + e2,
 z[0] == 0 + e3
 }, {x[t], y[t], z[t]}, {t, 0, 1000}],
 {e1, 0, 2 * 10 ^ –10, 10 ^ –10},
 {e2, 0, 2 * 10 ^ –10, 10 ^ –10}, {e3, 0, 2 * 10 ^ –10, 10 ^ –10}];
FlattenTablefunc = Flatten[Tablefunc, 2];
MakeVectorFromFunc2[q_] :=
 Table[detectStableFixedEnding[
 f_, df_, ddf_, (*lasttime_:1000,*)t0_] := Module[{lengths},
        length = EuclideanDistance[{f, df, ddf} /. t → t0, {f, df, ddf} /. t → t0 – 1]];
vectorl = Table[detectStableFixedEnding[Part[Flatten[q, 2][[x]], 1],
 Part[Flatten[q, 2][[x]], 2], Part[Flatten[q, 2][[x]], 3], n], {n, 1, 1000, 1}],
 {x, 1, Length[FlattenTablefunc]}];
```

Also, it is necessary to pay special attention to the Nose–Hoover system, since both the conservative Nose–Hoover system and the dissipative Nose–Hoover system behave very differently from the conventional examples, making them good learning samples for the classifier function. It is clear from the last section that both equations (2) and (3) exhibit dramatically different behavior for different values of $b$. Therefore, it is necessary to consider a different value of $b$ as an additional factor besides the initial condition for the Nose–Hoover sample. The value of $b$ was set to be able to vary from 0.75 to 5, with a step size of 0.75. Coupled with the allowed variations for initial conditions, the additional variation in the $b$ value produces 56 samples of the same kind for equations (2) and (3). It is worth noticing that equation (1) was also considered as a learning sample for the classifier to classify conservative systems more accurately, but since its behavior is independent of the value of $b$, this will only produce 27 learning samples for the conservative category. The following code generates Nose–Hoover samples from equation (2). Since there is one more level of structure due to $b$ values, the table is flattened to its third level rather than its second.

```
Tablefunc = Table[func = NDSolveValue[{
 x'[t] == y[t] - b * x[t],
 y'[t] == y[t] * z[t] - x[t],
 z'[t] == 3 - y[t] ^ 2,
 x[0] == 0 + e1,
 y[0] == 5 + e2,
 z[0] == 0 + e3
 }, {x[t], y[t], z[t]}, {t, 0, 1000}],
 {e1, 1 * 10 ^ -10, 2 * 10 ^ -10, 10 ^ -10},
 {e2, 0, 1 * 10 ^ -10, 10 ^ -10}, {e3, 0, 1 * 10 ^ -10, 10 ^ -10}, {b, 0.5, 5, 0.75}];
FlattenTablefunc = Flatten[Tablefunc, 3];
MakeVectorFromFunc2[q_, l_] :=
Table[detectStableFixedEnding[
 f_, df_, ddf_, (*lasttime_:1000,*)t0_] := Module[{lengths},
        length = EuclideanDistance[{f, df, ddf} /. t → t0, {f, df, ddf} /. t → t0 - 1]];
vectorl = Table[detectStableFixedEnding[Part[Flatten[q, l][[x]], 1],
 Part[Flatten[q, l][[x]], 2], Part[Flatten[q, l][[x]], 3], n], {n, 1, 1000, 1}],
 {x, 1, Length[FlattenTablefunc]}];
```

The third step is bias elimination. There are some samples that exist in a very small range, making it possible for the classifier to put plots of small ranges into one category. But this classification is not correct because the goal is to classify based on visual patterns instead of numerical values. An easy solution to this issue is to multiply every sample by a random real number. This can be easily accomplished with the Wolfram Language function RandomReal. Each sample is randomly scaled by a factor no bigger than 1000.

The final step before training the classifier is to extract the test set. A test set is used to measure the accuracy of the classifier. All the unbiased learning samples, conservative and dissipative, are combined into one list. After shuffling the sample list and making conservative and dissipative samples be evenly distributed, 100 and 200 samples are extracted from the list as test sets for two testing rounds.

By using these four steps (summarized in Figure 9), we start from some initial samples, then we duplicate them to generate more learning examples, eliminate bias and extract a test set from the learning

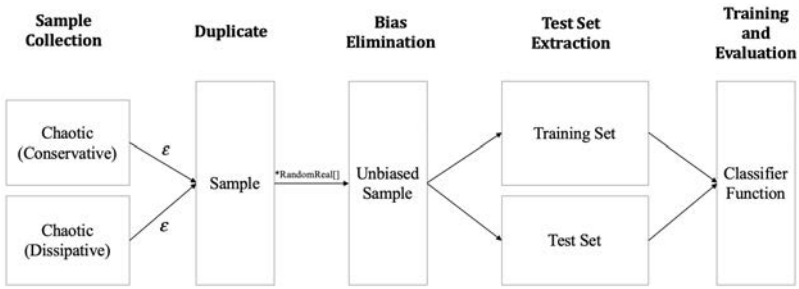



**Figure 9.** The diagram of the procedure from collecting samples to training a classifier.

samples. Then the learning samples are used to train the classifier and the test set is used to measure the accuracy of the classifier.

In the following code, D1 and C1 are lists that contain labeled samples of dissipative systems and conservative systems, respectively. The test (100 elements) and train are extracted from the shuffled total sample collection tt. The training set is used to train the classifier cl, and the accuracy of the classifier is examined by using the test set and ClassifierMeasurements to generate a ConfusionMatrixPlot.

```
t = Flatten[{D1, C1}];
tt = RandomSample[t];
test = Take[tt, {1, 100}];
train = Take[tt, {101, 517}];
cl = Classify[train];
ClassifierMeasurements[cl, test, "ConfusionMatrixPlot"]
```

The samples used in the classifier above were obtained by EDT. In Section 2, it was established that DDT is also valid for obtaining samples. The procedures for collecting samples are very similar, except that the function for generating sequences needs to be slightly modified by adding a command for calculating disdiffper, as shown in the following.

```
MakeVectorFromFunc2[q_] :=
 Table[
 detectStableFixedEnding[f_,
  df_, ddf_, (*lasttime_:1000,*)t0_] := Module[{lengths},
  length = EuclideanDistance[{f, df, ddf} /. t → t0, {f, df, ddf} /. t → t0 – 1]];
vectorl = Table[detectStableFixedEnding[Part[Flatten[q, 2][[x]], 1],
  Part[Flatten[q, 2][[x]], 2], Part[Flatten[q, 2][[x]], 3], n], {n, 1, 1000, 1}];
 disdiffper = Table[(Abs[vectorl[[n + 1]] – vectorl[[n]]]) /
  Norm[{Part[Flatten[q, 2][[x]], 1], Part[Flatten[q, 2][[x]], 2],
   Part[Flatten[q, 2][[x]], 3]} /. t → n], {n, 1, Length[vectorl] – 1}],
 {x, 1, Length[FlattenTablefunc]}];
```

The results suggest that both methods of training a classifier worked reasonably well (Figures 10 and 11). Even though the accuracy report said that DDT is slightly better than EDT, it is important to keep in mind that the process of obtaining samples involves randomly scaling the plot. This means that each time the training set and the testing set are different from the training set and the test set obtained last time, which may cause variations in accuracies and the number of misclassified samples. Nonetheless, it has been demonstrated that EDT and DDT are useful algorithms to classify conservative and dissipative systems computationally.

https://doi.org/10.25088/ComplexSystems.30.1.93

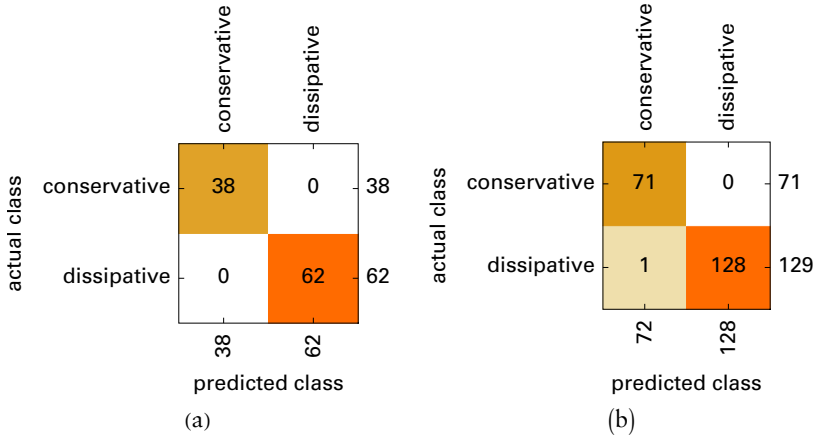



**Figure 10.** The accuracy of the classifier is measured by a confusion matrix plot. The classifier is trained using samples obtained by EDT. In the confusion matrix, diagonal entries indicate the number of elements in the test set that are classified correctly, whereas the nondiagonal entries indicate samples that are misclassified. The matrix on the left is the result of testing 100 elements after feeding 417 learning samples. The matrix on the right is the result of testing 200 elements after feeding 317 learning samples. There is only one misclassified element by the latter training method. According to the information report of the classifier, the accuracies are (98.2±2.5)% and (99.2±2.4)%.

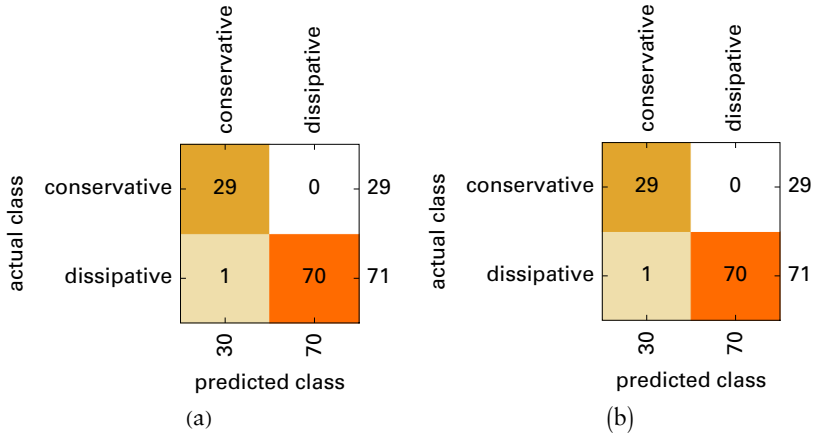


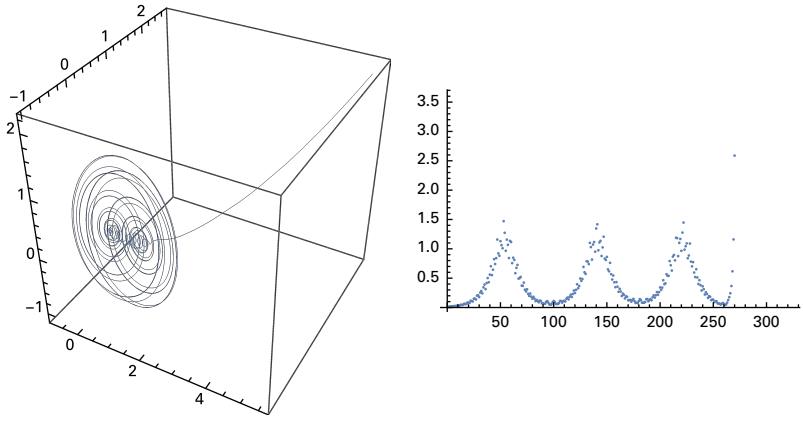

**Figure 11.** The accuracy of the classifier measured by a confusion matrix plot. The classifier is trained using samples obtained by DDT. The matrix on the left is the result of testing 100 elements after feeding 417 learning samples. The matrix on the right is the result of testing 200 elements after feeding 317 learning samples. Both methods misclassified one element. According to the information report of the classifier, the accuracies are (99.4±1.9)% and (99.3±2.4)%.

## 5. Supervised Machine Learning Classifier for Identifying Chaotic Dynamical Systems

Recognizing chaotic behaviors from a collection that contains both chaotic and nonchaotic systems is a much more difficult task for computers. Convergent trajectories are usually easy to identify because their Euclidean distance sequences converge to zero. Systems that have escaping trajectories are generally harder to distinguish from those with chaotic trajectories because they might exhibit locally chaotic behaviors or behaviors that are extremely similar to those of chaotic systems. An example of such systems is given below. The parameters 0.2 and 1 are the result of numerical searches of chaotic solutions to the equation with this particular form:

$$\dddot{x} + \ddot{x} - 0.2x(x-1) = 0 \tag{4}$$

The trajectory of equation (4) exhibits locally chaotic behavior in a very small range and then quickly escapes. The chosen initial condition is $x(0) = 0$, $\dot{x}(0) = -0.011$, $\ddot{x}(0) = 0$. The trajectory and its Euclidean distance sequence are shown in Figure 12.




(a) (b)

**Figure 12.** The trajectory and Euclidean distance sequence of the system in equation (4). (a) the trajectory is attracted to a small region for some finite cycles and then escapes. (b) the Euclidean distance sequence first oscillates and then diverges.

Additionally, there are some trajectories that escape to infinity while rotating periodically, forming a cylindrical path. There are also some trajectories that exhibit global chaotic behavior while expanding to infinity. Such systems, like some members of three-dimensional quadratic systems, have been studied previously [6]. In this classifier, the former are defined as escape trajectories and the latter are defined

as chaotic trajectories. However, since the former class exhibits behaviors that are similar to chaos and the latter class exhibits behaviors that are similar to escaping behaviors, they will create considerable difficulties for the classifier.

In order to duplicate convergent initial samples and escaping initial samples, their initial conditions are added by a random real number instead of a small $\epsilon$ because they are relatively less sensitive to the change of initial conditions compared to chaotic systems. This method of duplication ensures the duplicates will be the same kind as their initial samples. A total number of 776 convergent and escaping samples are generated. They are mixed with chaotic samples obtained in Section 4 for training. Two hundred testing elements are extracted from a total of 1293 samples. Both EDT and DDT are employed in training for comparison. The results suggest that in this task, DDT is considerably better in accuracy than EDT, and observation shows that DDT generates plots that are more distinguishable than EDT (Figures 13 and 14).

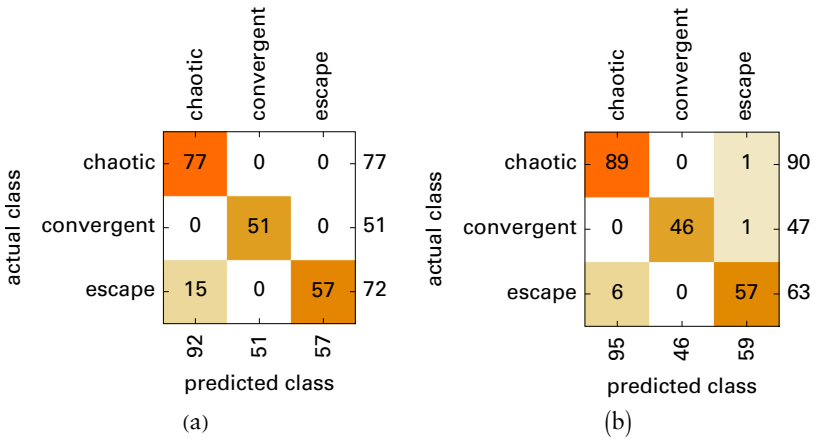



**Figure 13.** The accuracy of the classifier measured by a confusion matrix plot. The classifier is trained using samples obtained by EDT. Two tests are conducted by randomly changing the elements in the test set and the training set. The results suggest that of all misclassifications, the most frequent situation is that the classifier had trouble distinguishing escaping trajectories from chaotic trajectories, which is consistent with the analysis. According to the report of the classifier, the accuracy is (92.5±1.2)%.

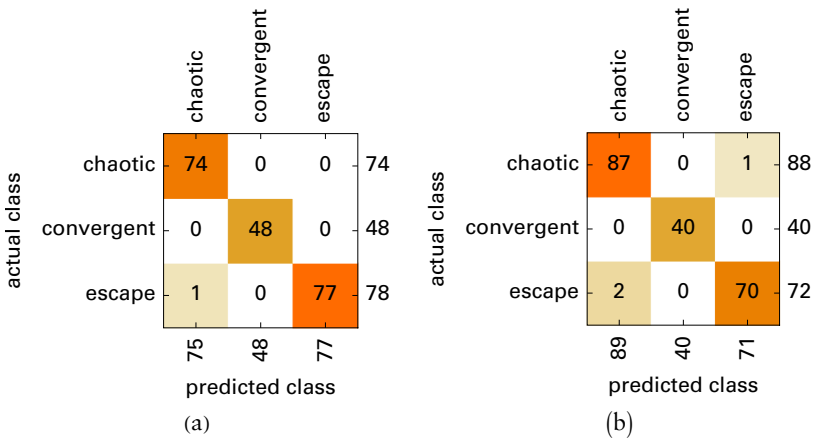



**Figure 14.** The accuracy of the classifier measured by a confusion matrix plot. The classifier is trained using samples obtained by DDT. Two tests are conducted by randomly changing the elements in the test set and the training set. According to the report of the classifier, the accuracy is (99.1±1.1)%.

## 6. Conclusion

In this paper, different behaviors of jerky dynamical systems have been discussed, specifically their behaviors in phase space. Two methods of classifying jerky dynamical systems are proposed, and classifiers are trained using known samples based on these two methods: Euclidean distance test (EDT) and distance difference test (DDT).

By visualizing the behavior of chaotic systems in phase space using the EDT method, it was found that most dissipative systems with zero divergence exhibit a similar visual pattern described as "scattering," whereas conservative systems exhibit much more diverse and complex behaviors, making them more interesting to study.

In the task of differentiating conservative and dissipative chaotic systems, both EDT and DDT demonstrated quite accurate results. In the task of distinguishing convergent, escaping and chaotic systems, DDT has an obvious advantage over EDT. The potential applications of these machine learning algorithms may include searching new chaotic systems and classifying chaotic systems given the behavioral data in the phase space.

https://doi.org/10.25088/ComplexSystems.30.1.93

## Acknowledgments

The author thanks Wolfram Research for holding Wolfram High School Summer Camp in 2019 and the faculty of the camp, including Stephen Wolfram (Wolfram Research), Peter Barendse (Wolfram Research) and Eryn Gillam (MIT) for proposing and perfecting this project. The author gives special thanks to his camp mate, Adrienne Lai, for proofreading this paper and correcting the grammatical mistakes that the author made.

## References

[1] Ö. Umut and S. Yasar, "A Simple Jerky Dynamics, Genesio System," *International Journal of Modern Nonlinear Theory and Application*, **2**(1), 2013 pp. 60–68. doi:10.4236/ijmnta.2013.21007.

[2] J. C. Sprott, "Some Simple Chaotic Jerk Functions," *American Journal of Physics*, **65**(6), 1997 pp. 537–543. doi:10.1119/1.18585.

[3] W. Rudin, *Principles of Mathematical Analysis*, Beijing: China Machine Press, 2019.

[4] B. Hasselblatt and A. Katok, *A First Course in Dynamics: With a Panorama of Recent Developments*, New York: Cambridge University Press, 2003.

[5] S. Vaidyanathan and C. Volos, eds., *Advances and Applications in Chaotic Systems*, Cham, Switzerland: Springer International Publishing, 2016.

[6] J. C. Sprott, "Some Simple Chaotic Flows," *Physical Review E*, **50**(2), 1994 pp. R647–R650. doi:10.1103/PhysRevE.50.R647.

[7] J. Heidel and F. Zhang, "Nonchaotic and Chaotic Behavior in Three-Dimensional Quadratic Systems: Five-One Conservative Cases," *International Journal of Bifurcation and Chaos*, **17**(6), 2007 pp. 2049–2072. doi:10.1142/S021812740701821X.

[8] B. Munmuangsaen, J. C. Sprott, W. J.-C. Thio, A. Buscarino and L. Fortuna, "A Simple Chaotic Flow with a Continuously Adjustable Attractor Dimension," *International Journal of Bifurcation and Chaos*, **25**(12), 2015 1530036. doi:10.1142/S0218127415300360.

[9] G. Gugapriya, K. Rajagopal, A. Karthikeyan and B. Lakshmi, "A Family of Conservative Chaotic Systems with Cyclic Symmetry," *Pramana–Journal of Physics*, **92**(4), 2019 48. doi:10.1007/s12043-019-1719-1.