Santa Fe Institute
2016 Complex Systems Summer School

Introduction to Nonlinear Time Series Analysis

1 The Lorenz System

The Lorenz system is defined by a system of three ODEs:

$$\begin{aligned}\dot{x} &= S(y-x)\\ \dot{y} &= x(R-z)-y\\ \dot{z} &= xy-Bz.\end{aligned}$$

Here x, y and z are state-space variables, S,R and B are parameters. There are different naming conventions for these parameters; many people for example use a or σ instead of S.

Use TISEAN's `lorenz` command to generate a 50,000-point trajectory of the Lorenz system with parameter values $R = 15$, $S = 16$, and $B = 4$:

```
lorenz -l50000 -R15 -S16 -B4 -x0 -o lorenzR15.dat
```

The `-o` option directs TISEAN to place its output in the filename that follows it. If you just type `-o` without a filename, TISEAN puts the output in `lorenz.dat`. The `-x0` option tells TISEAN not to discard any points from the beginning of the trajectory; we've chosen that here because we want you to see the transient.

Repeat this with $R = 45$ and plot both trajectories. If you're using gnuplot, try:

```
plot "lorenzR15.dat" using 1:2
```

which plots an $x - y$ projection of this 3D trajectory. To plot other projections, try `using 1:3` and `using 2:3` instead. You can plot in 3D with:

```
splot "lorenzR15.dat" with lines
```

2 Power Spectra

The power spectrum of a signal tells you how much power is present in that signal as a function of frequency. The spectrum of a pure tone—middle C on a piano—will be a single spike on such a plot, for instance.

Use TISEAN's `spectrum` command to compute the power spectra of the trajectories from problem 1:

```
spectrum lorenzR15.dat -c# -o lorenzR15.spectrum
spectrum lorenzR45.dat -c# -o lorenzR45.spectrum
```

The `-c#` option tells TISEAN to read the whole file, not just the first column (which it will do by default).

Make a semilog plot of these results: that is, the log of the power versus the frequency. If you're using gnuplot, you can do this by typing `set logscale y` before typing the `plot` command.

Compare and contrast the two spectra; relate your observations back to what you know about the two state-space trajectories and their properties.

3 Delay-Coordinate Embedding and Lyapunov Exponents

Download the following "real" dataset called amplitude.dat:

`http://tuvalu.santafe.edu/~jgarland/amplitude.dat`

In this problem, you'll use TISEAN's `lyap_k` tool to embed this trajectory and compute its maximal Lyapunov exponent λ_1.

Begin by downloading this data. Next, read up on `lyap_k` on the TISEAN manual[1]. You'll notice lots of discussion of caveats regarding interpretation, data, and parameter values. *This is the main challenge in using nonlinear time-series analysis tools:* these algorithms are designed to work on a limited amount of noisy, finite-precision data. Doing so involves approximations to the full formal mathematics, each of which involves some parameters: iteration limits, εs that specify scales, and something called the Theiler window that helps the algorithms avoid doubling back on themselves in bad ways, among other things. The Kantz & Schreiber book describes all of this in a lot more detail, and you should spend some serious time with it if you plan to use these tools in your work.

[1]This is under "General Manual" on their website—`www.mpipks-dresden.mpg.de/∼tisean`—then "Lyapunov Exponents."

The first step in the analysis of scalar time-series data from a nonlinear dynamical system is to embed it, and the first step in that process is to estimate the delay τ. One way to do this is to use TISEAN's `mutual` command, plot the results, and look for the first minimum in the curve. The second step is to estimate the embedding dimension m, which can be accomplished, for instance, with TISEAN's `false_nearest` command—e.g., the m value where the number of false near neighbors falls to 10%.

After embedding the data with correct values for these parameters, one can compute the Lyapunov exponents, fractal dimensions, etc. for the system. Choosing correct parameter values for the TISEAN tools that perform these calculations, as mentioned above, can be a challenge; when you use any of them, you should **at the very least** explore values other than the defaults and see if your results change.

You'll duck all of those issues in this problem, though, and run `lyap_k` with parameter values that have been established to work well for this data set: $3 \leq m \leq 6$, a delay of 8, a Theiler window of 100, a minimum epsilon of 0.1, and 500 iterations:

```
lyap_k amplitude.dat -d8 -m3 -M6 -t100 -r.1 -s500 -o
```

Some versions of TISEAN have a slightly different syntax for specifying the minimum and maximum embedding dimensions:

```
lyap_k amplitude.dat -d8 -M3,6 -t100 -r.1 -s500  -o
```

The `-o` option tells TISEAN to put the output in a file with the same name as the data file, but with the suffix "lyap". You can re-route the output to another file if you want; just specify the name of that file after the `-o`.

Plot the results. If you're using gnuplot, try

```
plot "amplitude.dat.lyap" with linespoints
```

Does your curve have a "scaling region"—a region where there's a clear diagonal line? The slope of this line, if it exists, is an estimate of the maximal Lyapunov exponent λ_1.

Spoiler Alert!! Let's say we know that the dominant Lyapunov exponent for this particular data is 0.015 ± 0.002. If we plot this line along with amplitude.dat.lyap we would see that this line has the same slope as the scaling region. For example in gnuplot:

```
plot "./amplitude.dat.lyap",0.015*x-4.8
```

4 Fractal Dimension

The correlation dimension d_{corr} is one member of the (large) family of fractal dimensions, many of which are covered in Chapter 6 of Kantz & Schreiber.

Correlation dimension is defined in terms of the correlation integral, which can be approximated by the correlation sum. Informally, the correlation sum counts the number of pairs $(\vec{x}(i), \vec{x}(j))$ in a given set of vectors that are at most ε apart. Formally, the correlation sum is defined as:

$$C(\varepsilon) = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \Theta(\varepsilon - ||\vec{x}(i) - \vec{x}(j)||), \quad \vec{x}(i) \in \mathbb{R}^m$$

where m is the embedding dimension, N is the number of points in the trajectory and $\Theta(x)$ is the Heaviside step function:

$$\Theta(x) = \begin{cases} 1 & : x > 0 \\ 0 & : x \leq 0 \end{cases}$$

As $N \to \infty$ we expect the correlation sum $C(\varepsilon)$ to scale like a power law, $C(\varepsilon) \propto \varepsilon^D$, where D is the correlation dimension defined by

$$d(N, \varepsilon) = \frac{\partial \ln C(\varepsilon, N)}{\partial \ln \varepsilon} \qquad D = \lim_{\varepsilon \to 0} \lim_{N \to \infty} d(N, \varepsilon)$$

That $N \to \infty$ requires an infinite amount of data, however, which is obviously not the case in practice. Therein lies the challenge of building and using algorithms like TISEAN's `d2`, which calculates the correlation sum of a trajectory.

Run the `d2` command on the amplitude dataset, embedded with $\tau = 8$ and a Theiler window of 100:

```
d2 amplitude.dat -d8 -t100 -o
```

This command will generate three files with the same name as the data set and different suffixes. We are interested in the ".c2" file. The first column in this file is ε and the second is an estimate of the correlation sum $C(\varepsilon)$ for that particular ε. The file will have several sections; these are the estimates for different embedding dimensions. The default range is $m = [1 - 10]$; you can specify otherwise with the `-m` and `-M` options to `d2` if you wish.

Make a log-log plot of ε versus $C(\varepsilon)$ and look for the scaling region in the plot. The slope of that region, if it exists, is an estimate of the correlation dimension of the data set.

Spoiler Alert!! Let's say we know that the Correlation dimension for this data set is 2.1 ± 0.05. If we plot a log-log line with slope 2.1 along with amplitude.dat.c2 we would see that this line has the same slope as the scaling region. For example in gnuplot:

```
plot 'amplitude.dat.c2',.001*x**2.1
```

5 Installing TISEAN on Your Machine

TISEAN is a useful time-series analysis package that you may find helpful in your own research. The purpose of this section is to help you install this package on your own machine and get started using it.

(a) If you have your own computer with you, follow the directions below to install TISEAN on it. Binaries for Mac/Windows/Linux can be found at:

`http://tuvalu.santafe.edu/~jgarland/tisean.html`

Mac Users This is the most difficult installation and may not work on your machine. Due to the diversity of machines, operating systems and package we will not be able to provide support for the installation of TISEAN on every machine. We have found that the simplest method for installation is through homebrew as all the messy installation steps have been taken care of for you. Details and installation instructions for homebrew are here: `http://brew.sh/`. With this installed open a terminal and type:

```
brew tap homebrew/science
brew install tisean
```

If you prefer to install this differently or prefer to not use homebrew that is fine, but we may not be able to aid you in the installation process. TISEAN is not a requirement of this course and you will not need it to complete any aspect of this course so feel free not to install it.

Linux and Windows Users

Linux and Windows binaries are available at:

`http://tuvalu.santafe.edu/~jgarland/tisean.html` Download the file corresponding to your operating system and decompress this file on your computer.

Using a command prompt/shell, go to the directory in which you unzipped TISEAN, then go to the "bin" folder in that directory.

You can then simply use the command prompt to run TISEAN commands. To test your install, run the following command: for windows users, type `henon -h`; for linux users, type `./henon -h`. Do not simply click on the .exe file—that will run the command with no arguments! Call one of us over if you get an error message.

(b) TISEAN's version of the Hénon map is defined as:

$$\begin{aligned} x_{n+1} &= 1 - Ax_n^2 + By_n \\ y_{n+1} &= x_n \end{aligned}$$

Generate a trajectory of this map using the following TISEAN command:

```
henon -l100000 -A0.8 -B0 -X.01 -Y.01 -o
```

This command sets the parameter values `A` and `B` to 0.8 and 0, respectively, iterates the map 100,000 times from the initial condition $(x_0, y_0) = (0.01, 0.01)$, and puts the resulting trajectory in a file called `henon.dat` in the current directory. Look at the last few lines of this file. Does this map have an attractor for these parameter values? What kind?

If the `henon` command does not work on your computer after following the above procedure,

come and see me!!

(c) Plot the data using your favorite plotting tool (Excel, Matlab, gnuplot, ...). You can find a short gnuplot tutorial at:

```
www.cs.colorado.edu/~lizb/na/gnuplot-info.html
```

Homework

- Vary the `henon` parameters (A and B) in the TISEAN example, plot the attractor (x_n vs y_n), and explore the dynamics:
  - Choose $-1 < B < 1$ and $A > \frac{3}{4}(1-B)^2$. What kind of attractor is this? Does it change if you change the initial conditions?
  - Now fix $B = 0.3$ and vary $0.8 \leq A \leq 1.3$. What happens at $A = 1.06$ and $A = 1.3$?
  - Now fix A but change the sign of B. How does this affect the dynamics?
- Change the TISEAN parameters (e.g. `d, m, M, t,` and `r`) for problems 3 and 4. Do your results change? A lot?
- Add some noise to your Lorenz data from problem 1 using TISEAN's `makenoise` command. Plot the trajectory and its power spectrum; compare the results to those from the noise-free data. Does this make sense?
- In problems 3 and 4 you were given d_{corr} and λ_1 for amplitude.dat. Try to find these two values "independently". Fit lines to the scaling regions of the plots in problems 3 and 4 and use them to compute numerical estimates of λ_1 and d_{corr}. You can do this with a straightedge or with a regression package (e.g., Excel's `trendline`), as you wish. If this was easy for you, do this with the $R = 45$ Lorenz data set you created in problem 1.
- (for experts) Use `mutual` (or `corr`) and `false_nearest` to determine appropriate embedding parameters for the `amplitude.dat` data set.
- Use `delay` to embed the `amplitude.dat` data—either with the τ and m values prescribed in problem 3, or with the values that you determined yourself. Plot the results. Do you recognize this attractor?