

June 2002

Barcelona Short Course

Daniel Kaplan, Macalester College

Lag Embedding, Poincaré Maps and Sections

Objectives

This lab demonstrates two important techniques in nonlinear time series analysis: processing data sampled from a continuous-time system to produce a discrete-time representation via the Poincaré section and Poincaré map; and constructing a multi-dimensional representation of a single signal using lag embedding. Some issues to examine are how the choice of cut in a Poincaré section affects the results and how the choice of embedding lag affects the gross shape of the “attractor.”

Data Sets

Several synthetic and real-world data sets are provided:

Rosler The full x, y, z state of the Rossler system is sampled by the program `makerosler`. This integrates the differential equations of the Rossler system for whatever duration and whatever initial condition you specify. You can set the time between samples. For example,
`>> [t,x,y,z] = makerosler(1000, [1 0 0], .1);`
will integrate the Rossler equations starting at an initial condition $x = 1, y = 0, z = 0$ and sampling every 0.1 second.¹

Several plots of the x, y, z data are easy to make: A time series plot of one variable:

```
>> plot(t,x)
```

The trajectory in the true state space

```
>> plot3(x,y,z)
```

You can use the rotation tool in the Figure window (the button with an arrow chasing it's own tail in the Figure window): press the button and then click and drag in the Figure window to change the perspective on the three dimensional plot.

To make an animation of the trajectory, use

```
>> comet3(x,y,z)
```

You can rotate this perspective using the rotation tool while the animation is in progress.

Lorenz Gives the full x, y, z state of the Lorenz system.

```
>> [t,x,y,z] = makelorenz(100, [2 0 20], .01);
```

Normal sinus rhythm The file `nsr.dat` contains an intracellular electrogram from normal sinus rhythm, provided by Nitish Thakor at Johns Hopkins University. You load this time series with the command

```
>> load nsr.dat
```

which creates a variable `nsr`. You can plot it as a time series with

```
>> plot(nsr)
```

or, say, the first 1000 points with

```
>> plot(nsr(1:1000))
```

Since this is a single-dimensional measurement, you can't plot out a “state space” — first, you'll have to construct the space.

¹“Second” should really read “time unit.” There are no physical units specified in the Rossler equations.

Ventricular fibrillation The file `vf.dat` contains an intracellular electrogram during ventricular fibrillation, again provided by Nitish Thakor. Load the data with the command

```
>> load vf.dat
```

which will create a variable `vf`.

Measles Month-by-month measurements of the number of measles cases in various cities are in the files `balt.dat`, `newyork.dat`, `detroit.dat`.

Tools

This lab is mainly about the visualization of time series and trajectories in real or reconstructed state spaces. In later labs we will deal with analysis and quantitative characterization of these trajectories.

The fundamental tools for visualization are, of course, plotting tools. We've already seen some of these: `plot`, `plot3`, `comet3`.

Another convenient plotting tool is

`scatplot(x,tau)` takes a scalar time series and makes a delay plot, a plot of $x(t + \tau)$ vs $x(t)$. By default, τ is 1. For example:

```
>> scatplot(x)
```

or

```
>> scatplot(x,20)
```

Sections

To create a Poincaré section, the following tools can be used:

`psection [t,p] = psection(traj,crossing level, plane, times)` is used to make a classical cutting-plane Poincaré section. `traj` is the collection of time series describing the trajectory. The returned variables are: `t`, the times at which the plane was cut; `p`, the position in the trajectory space of each pass (in a negative direction) through the plane. It is a matrix where there is one variable in each column. (It is very boring to make a Poincaré section of a single variable! Try it sometime.) The other arguments, shown in *italics*, are optional. The “crossing level” and “plane” describe where the cutting plane is located. By default, the plane cuts the first variable at the mean of that variable. For example, taking `x`, `y`, and `z` to be from the Rossler system, try

```
>> [tt,pp] = psection([x y z]);
```

Note that the three signals have been combined into one matrix to be handed off to `psection` and that `psection` returns two variables, the times of the crossings `tt` and the position `pp` in the full space of the crossings. There is one row for each crossing. Since the default cutting plane is the first variable, you'll see that the first column is constant. The other columns contain the information you want.

In this case the space being cut is three-dimensional, and the cut itself is two-dimensional. You can plot the places where the trajectory passes through the cutting plane with

```
>> plot(pp(:,2), pp(:,3), 'r')
```

(If you had used another cutting plane than the default one, you would have had to plot the points differently.) For example, try

```
>> scatplot(pp(:,2))
```

The Poincaré map is the relationship between the position of one pass through the plane with the next pass through the plane. Plotting this out fully would require 4 dimensions. For the Rossler data — but not for all data generally — the passes lie practically on a one-dimensional curve. We can exploit this structure to create a Poincaré map as a two dimensional plot. Try

```
>> scatplot(pp(:,2))
```

dmax(data,timescale,thresh) finds local maxima in a scalar time series.² The **timescale** sets the meaning of “local.” For an oscillatory signal, it should be set to be about somewhat less than the time between peaks — the precise value usually isn’t critical. The optional argument **thresh** is used to discard local maxima that are not large enough to be of interest to you. **dmax** returns the times and heights of the maxima detected. By plotting these out along with the original signal, you can adjust **timescale** as appropriate. For example,

```
>> [tt,aa] = dmax(balt,8);
```

The time scale has been set to 8, since this is somewhat less than the once-a-year peaks expected in measles.

```
>> plot(balt); hold on; plot(tt,aa,'o');hold off
```

The plot suggests that some of the small maxima have been missed. You can change the time scale to pick up these peaks if you like.

Embedding

Lag embedding is the fundamental technique in nonlinear time series analysis. This takes a scalar signal $x(t)$ and turns it into a vector signal; at each time t the vector is $(x(t), x(t-\tau), \dots, x(t-(p-1)\tau))$. The parameter p is called the embedding dimension, the parameter τ is the lag.

lagembed(x,p,tau) takes a vector **x** and returns a matrix that is the lag-embedding of embedding dimension **p** and lag **tau**, both of which should be integers.³ Try

```
>> traj = lagembed(balt,4,2);
```

Note that **lagembed** doesn’t create any new information; it just re-organizes existing information. Each column of the matrix **traj** is the original signal shifted a bit in time (and with a bit cut off the ends of the signal).

Finally, a small program that keeps a “sub-space” of a collection of data stored as a matrix:

keepPC(data,nkeep) keeps the **nkeep** largest principal components of the cloud of **data**.

Questions

1. Embed the time series


```
>> simple = [1:10]';
```

 (making sure it’s a column vector) for various embedding lags and dimensions. By looking at the matrix produced by **lagembed** you should be able to see exactly what is going on in lag embedding.
2. Plot the Lorenz data in the original x, y, z coordinates to see the shape of the attractor in the actual state space. Now use lag embedding on just a single coordinate to produce a reconstructed model of the attractor. Since we’re visualizing the attractor, we’re pretty much restricted to using a dimension of $p = 2$, so you can use **scatplot** to form and plot the embedding in one command. You might also want to use **lagembed** with a dimension of $p = 3$ along with **plot3**. Note that **plot3** takes three vectors as arguments, so you will have to do something like:


```
>> foo = embed(x,3,10);
>> plot3(foo(:,1), foo(:,2), foo(:,3));
```

Find an embedding lag that gives the most faithful-looking reconstruction of the trajectory; compare the reconstructed trajectory to the original $[x \ y \ z]$. Try lags that are very short, lags that are a fraction of the typical period of oscillation of the signal, and lags that are longer than the period of oscillation. Try embeddings of all three state variables individually.

²Although **dmax** may not seem to be a Poincaré section, you might want to think of it as a section in the space constructed from the signal and its first derivative: the maximum of the signal corresponds to the plane where the derivative is zero.

³In principle, τ does not need to be an integer, but here we are dealing with sampled data.

The program `acf(data)` computes the autocorrelation function of a scalar signal. One rule of thumb for selecting an embedding lag is to pick the lag that corresponds to the first zero-crossing of the autocorrelation function or, for signals without regular oscillations, the lag at which the autocorrelation function falls to about $\frac{1}{e} \approx 0.37$. (You can use `acf(data,maxlag)` to restrict the plot to lags less than `maxlag`.)

3. Construct a Poincaré section of the Rossler data and generate a Poincaré map from this. How do the results depend on the way in which the Poincaré section is taken, that is, on the position and orientation of the cutting plane?

Create a Poincaré map of the Lorenz data. It is not as easy as it might seem to create a map that shows a simple structure. It's hard to find a good Poincaré cutting plane. Try both the classical Poincaré cut (`psection`) and local maxima of one signal.

4. Create lag embeddings of the real-world signals, picking the embedding lag to produce a “chaotic-looking” trajectory. Experiment with ways to take Poincaré sections of these reconstructions. For the cardiac data, a typical Poincaré section is measuring the reconstructed model state once per “beat.” For the measles data, the section is measuring things once per outbreak.
5. The use of principal components allows one to use a rather high embedding dimension and project this down into a lower-dimensional space while retaining much of the (linear) information in the signal. Try using a very short embedding lag — much shorter than you found in (1) — and a high embedding dimension. The product of the embedding dimension and embedding lag is called the “embedding window” and reflects the amount of the signal that is incorporated in each point in the embedding. Use `keepPC` to keep the two or three largest principal components of the embedding. Plot these out and compare them qualitatively to the results you got using the best embedding lag you found in (1).