

June 2002

Barcelona Short Course

Daniel Kaplan, Macalester College

Dimensions

This lab is about the correlation integral and correlation dimension. Calculating the correlation integral is straightforward once the embedding parameters have been chosen. The correlation dimension is derived from the correlation integral, but there are a number of ways of doing this; the theoretically correct method is impossible in practice. Interpretation of the correlation integral is therefore somewhat difficult.

In this lab we'll look at the correlation dimension for a variety of signals: chaotic, random, and deterministic but with a random component.

Tools

The main program is `c2`, which computes the correlation dimension of a time series. The program `lag` embeds the time series. The mandatory arguments are:

- the time series;
- the embedding dimension;
- the embedding lag;

Some additional arguments are optional:

- the decimation factor. The `c2` program is very slow because it is written in an interpreted computer language. In order to speed things up, the program is arranged to allow you to calculate a sample of the correlation integral. Rather than using every reference point in the time series, a sample of reference points is used. By default the decimation factor is 1, but increasing it will correspondingly decrease the computation time (at the cost of a less precise correlation integral).

USE A LARGE DECIMATION FACTOR at first, decreasing it when you find the computation time is satisfactory. What's large? If your time series is length N , then the computational time goes as $\frac{N^2}{\text{decim}}$. You should set `decim` so that this number is no larger than, say, 100,000. That is, for a time series of length 1000, set `decim` to be at least 10 at first. If you get results that you want to refine, you can lower `decim`. For a time series of length 1000, with `decim` set to 10, you can expect the correlation integral calculation to take some tens of seconds. For a time series of length 10000, `decim` should be set to something like 10000, decreasing it as you decide to invest in a more precise calculation.

- the Theiler correction factor. This factor is intended to reduce the influence of linear correlations on computations of short time series. It's default value is reasonable, so you don't need to worry about it except when you want to see what happens without it.
- the embedding dimensions to report. Again, this is worth worrying about only if computation time is an issue.

The program returns two variables:

`r` a vector of length scales at which the correlation integral has been calculated.

`c` the correlation integral. Each column is the integral at one of the embedding subspaces. That is, the first column is for an embedding dimension of 1, the second column is for an embedding dimension of 2, and so on. (If the `reportdims` variable is set, then the columns are for the corresponding dimension in `reportdims`.)

It's really much easier to use than the above suggests. Let's try it:

```
>> load rossler.dat
>> x = rossler(:,1); % the x component
>> [r,c] = c2(x,3,3,10); % decimation 10
>> plot(r,c); legend('1', '2', '3')
```

The `legend` command helps to label the curves: there's one for embedding dimension 1, one for dimension 2, and so on. (Use the mouse to drag the legend box to another place if it's in the way.)

Another program, `c2embed` is exactly the same as `c2` but it takes a matrix instead of a time series. This is useful if you already have an embedded cloud of points, perhaps from principal components of an embedding or some other source.

If we plot the correlation integral on log-log axes, then the slope is the correlation dimension.

```
>> plot(log(r), log(c)); legend('1', '2', '3');
```

You can see that the slope is not the same everywhere; this is the source of some difficulty with the correlation.

We need to pick part of the curve to calculate the slope. One easy way to do this is to specify the upper and lower bounds on the vertical axis since often the curves appear quite straight over the same vertical range. The program `d2byC` will do this:

```
>> dims = d2byC(log(r), log(c), 6, 8)
ans:      ans:  0.9924
          1.7963
          1.8626
```

To illustrate a somewhat longer calculation, we'll calculate the correlation integral for embedding dimensions up to 10:

```
>> [r,c] = c2(x(1:1000),10,3,10);
>> plot(log(r), log(c)); legend(num2str([1:10]'));
>> xlim([-5 5]) % set the x-axis to a better scale
>> dims = d2byC(log(r), log(c), 5,7)
ans:      ans =
          0.9569
          1.7803
          1.9395
          1.9788
          1.9347
          2.0032
          2.0156
          2.0489
          2.0570
          2.0559
```

Researchers often plot the computed dimension versus the embedding dimension

```
>> plot(1:10, dims, 'x-')
```

and look for "saturation," a leveling-off of the curve. Here the curve levels off at a dimension of slightly more than 2.

Questions

- Compute the correlation integral of a very simple set of points embedded in 1 dimension:
 $\gg x = [1;2;3;4;5;6;7;8;9;10];$
 (Make sure that x is a column vector, that is, a matrix with only one column.) Now compute the correlation dimension of these points (in their 1-dimensional embedding):
 $\gg [r,c] = c2embed(x);$
 $\gg plot(r,c)$
 Explain the result in detail. (Note, here we've plotted r and c on linear axes. In the remainder of the lab, we'll use log-log axes.)
- Compute the correlation dimension of gaussian white noise:
 $\gg noise = drand(1000,1);$
 $\gg mn = lagembed(noise, 10, 1);$
 How does the correlation dimension depend on the embedding dimension?
- Compute the correlation dimension of a periodic sine curve:
 $\gg s = sin([1:1000]/8)';$
 How does this depend on the embedding dimension?
- Take a nonlinear measurement transformation of the sine curve, for instance
 $\gg x = s.^3;$
 or
 $\gg x = sin(3*s);$
 Do the dimensions differ from those of the original sine curve?
- Add a small amount of noise to the sine curve
 $\gg x = s + 0.1*randn(size(s));$
 How does this change the correlation integral?
- Take a length of the Rossler data generated with `makerossler`. Arrange to sample the signal so that there are roughly 25 points per cycle and about 1000 points altogether. What is the dimension? Now generate another such signal from a different initial condition. Add it to the first signal. What is the dimension of the sum?