Multiscale Analysis of Complex Time Series: Integration of Chaos and Random Fractal Theory, and Beyond
by Jianbo Gao, Yinhe Cao, Wen-wen Tung and Jing Hu
Copyright © 2007 John Wiley & Sons. Inc.

APPENDIX C

COMPLEXITY MEASURES

The term complexity has many different meanings. Deterministic complexity, for example, measures the randomness of the data. It is a nondecreasing function of the entropy rate h_μ (e.g., Shannon entropy or Kolmogorov-Sinai (KS) entropy). See Fig. C.1 (left). The famous Kolmogorov-Chaitin complexity and the Lempel-Ziv (LZ) complexity both belong to this category. The structural complexity, on the other hand, is maximized for neither high nor low randomness (Fig. C.1 (right)). Complexity is a vast field, with many excellent schemes as well as numerous misleading results published. For details, we refer readers to books by Badii and Politi [19], Cover and Thomas (chapter 7) [86], Chaitin [67], Rissanen [369], Shaw [397], Watanabe [467], Bar-Yam [27], and a number of classic papers [18,20,39,40,66,90–96,102,132,133,142,185,202,212,231,267–270,283,286,287,301,377,408,420,464,465,477,480–482].

In Chapters 11, 13, and 15, we discussed a number of complexity measures. In this appendix, we describe three more: the finite-size Lyapunov exponent (FSLE), the LZ complexity, and the permutation entropy (PE). The latter two were used in Chapter 15 to characterize EEG data.

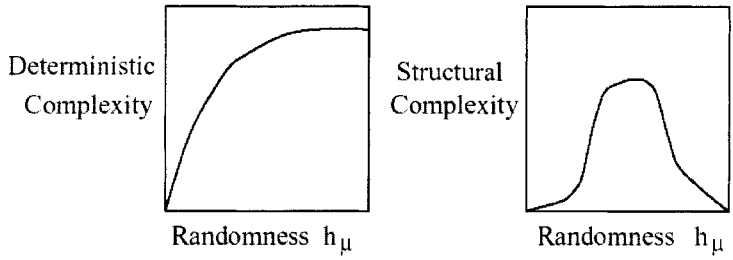

Figure C.1. Deterministic vs. structural complexity.

C.1 FSLE

There exist a number of variants of the FSLE [16]. We describe the most popular one here.

Recall that Wolf et al.'s algorithm for calculating the largest Lyapunov exponent is used to monitor the exponential growth of a distance between a reference and a perturbed trajectory. For a given dataset, the growth rate, however, may not be truly exponential. When this is the case, the growth rate then depends on the actual distance between the reference and the perturbed trajectory. The FSLE is a simple way to quantify this possible dependence. Quantitatively, it is defined through the following steps:

1. One first constructs a suitable phase space from the time series data and then chooses a norm to define distance in the phase space.
2. One then introduces a series of scales, $\epsilon_n = r^n \epsilon_0$, $n = 1, \cdots, P$, where $1 < r \leq 2$.
3. One monitors, on average, how soon a small perturbation of size ϵ_i $i = 0, \cdots, P-1$, grows to the size ϵ_{i+1}. Denote this average time by $T_r(\epsilon_i)$. When $r = 2$, $T_r(\epsilon_i)$ is the doubling time. When $r \neq 2$, $T_r(\epsilon_i)$ is called the r-fold time.
4. Assuming that
$$\epsilon_{i+1} = r\epsilon_i = \epsilon_i e^{T_r(\epsilon_i)\lambda(\epsilon_i)}, \tag{C.1}$$
one obtains the FSLE at scale ϵ_i, denoted by $\lambda(\epsilon_i)$, to be
$$\lambda(\epsilon_i) = \frac{\ln r}{T_r(\epsilon_i)}. \tag{C.2}$$

We now make a few comments:

- Equation (C.1) may be considered as a fitting using exponential function. This fitting is often acceptable, so long as $1 < r \leq 2$, even though the actual growth from scale ϵ_i to scale $\epsilon_{i+1} = r\epsilon_i$ may not be truly exponential.

- One major purpose of defining the FSLE is to identify interesting scales. The algorithm for computing the FSLE involves $P + 1$ scale parameters — the set of scales ϵ_i, $i = 0, \cdots, P - 1$, and a scale parameter needed to perform renormalization in order to use Wolf et al.'s algorithm. Determining these scales a priori through a process of trial and error is often a challenge.
- In order to calculate the r-fold time $T_r(\epsilon_i)$, a large dataset is usually needed to suitably define a reference and a perturbed trajectory and take averages.
- The r-fold time $T_r(\epsilon_i)$ is often connected with the prediction time scale. This interpretation is not suitable for stochastic systems. In fact, for $1/f^{\beta}$ and Levy processes, using Eqs. (15.11), (15.12), and (C.1), one has

$$T_r(\epsilon_i) = \frac{\ln r}{\mu} \epsilon_i^{\frac{1}{\mu}}, \tag{C.3}$$

where $\mu = H$ for $1/f^{\beta}$ processes and $\mu = 1/\alpha$ for Levy flights. In such cases, $T_r(\epsilon_i)$ may not be interpreted as a prediction time in the usual sense.

C.2 LZ COMPLEXITY

The LZ complexity and its derivatives, being easily implementable, very fast, and closely related to the Kolmogorov complexity, have found numerous applications in characterizing the randomness of complex data.

To compute the LZ complexity, a numerical sequence first has to be transformed into a symbolic sequence. The most popular approach is to convert the signal into a $0 - 1$ sequence by comparing the signal with a threshold value S_d [491]. That is, whenever the signal is larger than S_d, one maps the signal to 1; otherwise, one maps it to 0. One good choice of S_d is the median of the signal [317]. When multiple threshold values are used, one may map the numerical sequence to a multisymbol sequence. Note that if the original numerical sequence is a nonstationary random walk–type process, one should analyze the stationary differenced data instead of the original nonstationary data.

After the symbolic sequence is obtained, it can then be parsed to obtain distinct words, and the words can be encoded. Let $L(n)$ denote the length of the encoded sequence for those words. The LZ complexity can be defined as

$$C_{LZ} = \frac{L(n)}{n}. \tag{C.4}$$

Note this is very much in the spirit of the Kolmogorov complexity [267, 268].

There exist many different methods to perform parsing. One popular scheme was proposed by the original authors of the LZ complexity [282, 497]. For convenience, we call this Scheme 1. Another attractive method is described by Cover and Thomas [86], which we shall call Scheme 2. For convenience, we describe them in the context of binary sequences.

• **Scheme 1:** Let $S = s_1 s_2 \cdots s_n$ denote a finite-length 0 – 1 symbolic sequence; $S(i,j)$ denote a substring of S that starts at position i and ends at position j, that is, when $i \leq j$, $S(i,j) = s_i s_{i+1} \cdots s_j$ and when $i > j$, $S(i,j) = \{\}$, the null set; $V(S)$ denote the vocabulary of a sequence S. It is the set of all substrings, or words, $S(i,j)$ of S, (i.e., $S(i,j)$ for $i = 1, 2, \cdots, n$; $j \geq i$). For example, let $S = 001$; we then have $V(S) = \{0, 1, 00, 01, 001\}$. The parsing procedure involves a left-to-right scan of the sequence S. A substring $S(i,j)$ is compared to the vocabulary that is comprised of all substrings of S up to $j-1$, that is, $V(S(1, j-1))$. If $S(i,j)$ is present in $V(S(1,j-1))$, then update $S(i,j)$ and $V(S(1,j-1))$ to $S(i,j+1)$ and $V(S(1,j))$, respectively, and the process repeats. If the substring is not present, then place a dot after $S(j)$ to indicate the end of a new component, update $S(i,j)$ and $V(S(1,j-1))$ to $S(j+1, j+1)$ (the single symbol in the $j+1$ position) and $V(S(1,j))$, respectively, and the process continues. This parsing operation begins with $S(1,1)$ and continues until $j = n$, where n is the length of the symbolic sequence. For example, the sequence 1011010100010 is parsed as $1 \cdot 0 \cdot 11 \cdot 010 \cdot 100 \cdot 010\cdot$. By convention, a dot is placed after the last element of the symbolic sequence. In this example, the number of distinct words is six.

• **Scheme 2:** The sequence $S = s_1 s_2 \cdots$ is sequentially scanned and rewritten as a concatenation $w_1 w_2 \cdots$ of words w_k chosen in such a way that $w_1 = s_1$ and w_{k+1} is the shortest word that has not appeared previously. In other words, w_{k+1} is the extension of some word w_j in the list, $w_{k+1} = w_j s$, where $0 \leq j \leq k$, and s is either 0 or 1. The sequence 1011010100010 in Scheme 1 is parsed as $1 \cdot 0 \cdot 11 \cdot 01 \cdot 010 \cdot 00 \cdot 10\cdot$. Therefore, a total of seven distinct words are obtained. This number is larger than the six of Scheme 1 by one.

The words obtained by Scheme 2 can be readily encoded. One simple way is as follows [86]. Let $c(n)$ denote the number of words in the parsing of the source sequence. For each word, we use $\log_2 c(n)$ bits to describe the location of the prefix to the word and one bit to describe the last bit. For our example, let 000 describe an empty prefix; then the sequence can be described as $(000,1)(000,0)(001,1)(010,1)(100,0)(010,0)(001,0)$. The total length of the encoded sequence is $L(n) = c(n)[\log_2 c(n) + 1]$. Equation (C.4) then becomes

$$C_{LZ} = c(n)[\log_2 c(n) + 1]/n. \tag{C.5}$$

When n is very large, $c(n) \leq n/\log_2 n$ [86, 282]. Replaing $c(n)$ in Eq. (C.5) by $n/\log_2 n$, one obtains

$$C_{LZ} = \frac{c(n)}{n/\log_2 n}. \tag{C.6}$$

The commonly used definition of C_{LZ} takes the same functional form as Eq. (C.6), except that $c(n)$ is obtained by Scheme 1. Typically, $c(n)$ obtained by Scheme 1 is smaller than that obtained by Scheme 2. However, encoding the words obtained by Scheme 1 requires more bits than that obtained by Scheme 2. We surmise that the complexity defined by Eq. (C.4) is similar for both schemes. Indeed, numerically,

we have observed that the functional dependence of C_{LZ} on n (based on Eqs. (C.5) and (C.6)) is similar for both schemes.

Rapp et al. [364] have considered the issue of normalizing the LZ complexity to make it independent of the sequence length. Recently, this issue has been reconsidered by Hu et al. [226], using an analytic approach. Specifically, they derived formulas for the LZ complexity for random equiprobable sequences as well as periodic sequences with an arbitrary period m and proposed a simple formula to perform normalization. It should be emphasized, however, that there may not exist universal normalization schemes that work for complex data with vastly different time scales.

Finally, we note that in the literature, the LZ complexity is sometimes interpreted as characterizing the structure of the data, possibly due to the perception that structure is a more appealing word than randomness. This is inappropriate, since asymptotically, the LZ complexity is equal to the Shannon entropy. This very characteristic of the LZ complexity — characterizing the randomness of the data — also dictates that the LZ complexity alone should not be used for the purpose of distinguishing deterministic chaos from noise.

C.3 PE

PE is introduced in [26] as a convenient means of analyzing a time series. It may be considered as a measure from chaos theory, since embedding vectors are used in the analysis. Using the notations of [61], it can be described as follows. For a given but otherwise arbitrary i, the m number of real values of $X_i = [x(i), x(i+L), \cdots, x(i+(m-1)L)]$ are sorted in $[x(i+(j_1-1)L) \leq x(i+(j_2-1)L) \leq \cdots \leq x(i+(j_m-1)L]$. When an equality occurs, e.g., $x(i+(j_{i1}-1)L) = x(i+(j_{i2}-1)L)$, the quantities x are ordered according to the values of their corresponding j's; that is, if $j_{i1} < j_{i2}$, then we write $x(i+(j_{i1}-1)L) \leq x(i+(j_{i2}-1)L)$. Therefore, the vector X_i is mapped onto $(j_1, j_2, \cdots, j_m)$, which is one of the $m!$ permutations of m distinct symbols $(1, 2, \cdots, m)$. When each such permutation is considered as a symbol, the reconstructed trajectory in the m-dimensional space is represented by a symbol sequence. Let the probability for the $K \leq m!$ distinct symbols be $P_1, P_2, \cdots, P_K$. Then PE, denoted by E_p, for the time series $\{x(i), i = 1, 2, \cdots\}$ is defined as

$$E_p(m) = -\sum_{j=1}^{K} P_j \ln P_j. \tag{C.7}$$

The maximum of $E_P(m)$ is $\ln(m!)$ when $P_j = 1/(m!)$. It is convenient to work with

$$0 \leq E_p = E_p(m)/\ln(m!) \leq 1. \tag{C.8}$$

Thus E_p gives a measure of the departure of the time series under study from a completely random one: the smaller the value of E_p, the more regular the time series is.