Direct Computing of Entropy from Time Series

Mehran Ebrahimi Kahrizsangi

A Thesis

in

The Department

of

Mathematics and Statistics

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Science (Mathematics) at
Concordia University
Montreal, Quebec, Canada

June 2003

Your file Votre référence
ISBN: 0-612-83855-2
Our file Notre référence
ISBN: 0-612-83855-2

Abstract

Direct Computing of Entropy from Time Series

Mehran Ebrahimi Kahrizsangi

Measure Theoretic Entropy and its important properties are studied. We introduce a method to compute entropy of a dynamical system directly from the definition. The computational approach is discussed in detail and is presented in several sections: 1-Partitioning and Scaling Data; 2-Sequencing and Compactification; 3-Probabilities and Information, 4-Entropy Estimation. Also, we apply the same method in two dimensions. A model for filtering entropy based on skew products is given, and we apply our computational results to verify this model.

Acknowledgments

Many people have contributed to my education through their guidance and support throughout my graduate school years at Concordia University. I especially want to thank my supervisors, Professors Abraham Boyarsky and Pawel Góra who accepted me as their student and were always there to give support.

I also acknowledge Professors Galia Dafni, Chantal David, and Robert Raphael. Their encouragement in the first year of my graduate studies at Concordia University served as a source of inspiration.

I would like to thank my friends Wael Bahsoun and Shafiqul Islam, who were notable friends and colleagues, kindly offering me their advice, both scientific and personal, throughout my graduate school years at Concordia University.

Personal friends in Canada, as well as friends who have been in touch with me remotely from Iran and the US, provided me with stimulation and enjoyment critical for me to maintain a healthy balance of work and play. They all continue to be dedicated and supportive friends.

I deeply thank my parents and family for their understanding and support throughout my entire educational process.

Finally, I am greatly thankful to Concordia University Mathematics and Statistics Department, Concordia University Graduate Awards Office, and Institut des Sciences Mathmatiques (ISM) for their support during my studies at Concordia University. Their generous financial assistance made it possible for me to continue my studies in Canada.

Contents

List of Figures

List of Tables

Chapter 1

Preliminaries

1.1 Introduction

The material in this chapter consists of fundamental concepts which will be used throughout this thesis. It is mostly taken from from [5] and [19].

1.2 Measure Spaces

Definition 1.2.1 A family $\mathcal{B}$ of subsets of X is called a *σ-algebra* if and only if:

(a) $X \in \mathcal{B}$;

(b) for any $B \in \mathcal{B}, X\backslash B \in \mathcal{B}$;

(c) if $B_n \in \mathcal{B}$ for $n = 1, 2, \ldots$, then $\bigcup_{n=1}^{\infty} B_n \in \mathcal{B}$.

Elements of $\mathcal{B}$ are called *measurable sets.*

A σ-algebra which is a subset of a σ-algebra $\mathcal{B}$ is called a *sub-σ-algebra* of $\mathcal{B}$.

For a family $\mathcal{F}$ of subsets of X there exists a smallest σ-algebra, $\mathcal{B}$, containing $\mathcal{F}$. In this case we say that $\mathcal{F}$ generates $\mathcal{B}$ and write $\mathcal{B} = \sigma(\mathcal{F})$.

Definition 1.2.2 A function $\mu : \mathcal{B} \to \mathrm{R}^+$ is called a *measure* on $\mathcal{B}$ if and only if:

(a) $\mu(\emptyset) = 0$;

(b) for any sequence $\{B_n\}$ of disjoint measurable sets, $B_n \in \mathcal{B}, n = 1, 2, \ldots$,

$$\mu(\bigcup_{n=1}^{\infty} B_n) = \sum_{n=1}^{\infty} \mu(B_n).$$

The triplet $(X, \mathcal{B}, \mu)$ is called a *measure space.* If $\mu(X) = 1$, we say it is a *normalized measure space* or a *probability space.*

Definition 1.2.3 A family $\mathcal{A}$ of subsets of X is called an *algebra* if:

(a) $X \in \mathcal{A}$;

(b) for any $A \in \mathcal{A}, X \backslash A \in \mathcal{A}$;

(c) for any $A_1, A_2 \in \mathcal{A}, A_1 \bigcup A_2 \in \mathcal{A}$.

An algebra which is a subset of an algebra $\mathcal{B}$ is called a *sub-algebra* of $\mathcal{B}$.

It is clear that intersection of any collection of algebras (σ-algebras) is also an algebra (σ-algebra).

Definition 1.2.4 A family $\mathcal{P}$ of subsets of X is called a *π-system* if and only if for any $A, B \in \mathcal{P}$, their intersection $A \cap B \in \mathcal{P}$.

Definition 1.2.5 Let X be a topological space. Let $\mathcal{D}$ denotes the family of open sets of X. Then the σ-algebra $\mathcal{B} = \sigma(\mathcal{D})$ is called the *Borel σ-algebra* of X and its elements are called the *Borel subsets* of X.

1.3 Direct Product Measure

In this section we define direct product measure.

Theorem 1.1 *Hahn-Kolmogorov Extension ([19])*
Given a set X, and an algebra $\mathcal{A}$ of subsets of X, let $\mu : \mathcal{A} \to \mathbb{R}^+$ be a function satisfying: $\mu(X) = 1$; $\mu(\bigcup_n A_n) = \sum_n \mu(A_n)$ whenever $\forall n\ A_n \in \mathcal{A}, \bigcup_n A_n \in \mathcal{A}$, and the sets A_n are mutually disjoint. Then there is a unique probability measure $\overline{\mu}$ defined on the sigma-algebra generated by $\mathcal{A}$ such that: $\overline{\mu}(A) = \mu(A)$ whenever $A \in \mathcal{A}$.

Definition 1.3.1 (Direct product measure)
Let $(X_i, \mathcal{B}_i, \mu_i)$, $i \in \mathbb{Z}$ be probability spaces. Their *product* is defined as follows:

(a) $X = \prod_{i=-\infty}^{\infty} X_i$.

(b) Let $n_1 < n_2 < \cdots < n_r$ be integers, and $A_{ni} \in \mathcal{B}_{n_i}$, $i = 1, \ldots, r$. we define a *measurable rectangle* to be a set of the form

$$\{(x_j) \in X : x_{n_i} \in A_{n_i}, 1 \leq i \leq r\}.$$

Let $\mathcal{A}$ be a collection of all finite unions of measurable rectangles. $\mathcal{A}$ is an algebra: (1), (3) of Definition 1.2.3 are obvious; to show (2) observe that

$$X \setminus \{(x_j) \mid x_{n_i} \in A_{n_i}, 1 \leq i \leq r\} = \bigcup_{i=1}^{r} \{(x_j) \mid x_{n_i} \in X_{n_i} \setminus A_{n_i}\} \in \mathcal{A},$$

and that $\mathcal{A}$ is closed under finite intersections. Let $\mathcal{B}$ be the smallest σ-algebra generated by $\mathcal{A}$.

(c) Each element of $\mathcal{A}$ can be written as a finite union of disjoint measurable rectangles so that we define

$$\mu(\{(x_j) \mid x_{n_i} \in A_{n_i}, 1 \leq i \leq r\}) = \prod_{i=1}^{r} \mu_{n_i}(A_{n_i})$$

and then extend μ to $\mathcal{A}$ in the obvious manner. The conditions of the Hahn-Kolmogorov Theorem can be shown to be satisfied, and thus we can extend μ to $\mathcal{B}$. Hence we obtain a probability space.

1.4 Measure-Preserving Transformations

Suppose $(X_1, \mathcal{B}_1, \mu_1)$, and $(X_2, \mathcal{B}_2, \mu_2)$ are probability spaces.

Definition 1.4.1 A transformation $T : X_1 \to X_2$ is called *measurable* if $T^{-1}(\mathcal{B}_2) \subseteq \mathcal{B}_1$, i.e., if $B \in \mathcal{B}_2 \Longrightarrow T^{-1}(B) \in \mathcal{B}_1$, where $T^{-1}(B) = \{x \in X_1 : T(x) \in B\}$.

Definition 1.4.2 We say the measurable transformation $T : X_1 \to X_2$ is *measure-preserving* if $\mu_1(T^{-1}(B)) = \mu_2(B)$ for any $B \in \mathcal{B}_2$.

Definition 1.4.3 Let $(X, \mathcal{B}, \mu)$ be a probability space and let: $T : X \to X$ be a measure-preserving transformation. The quadruple $(X, \mathcal{B}, \mu, T)$ is called a *dynamical system.*

In practice it would be difficult to check, using Definition 1.4.2, whether a given transformation is measure-preserving or not since usually one does not have an explicit knowledge of all the elements of a σ-algebra $\mathcal{B}$. However, we often do have an explicit knowledge of a π-system $\mathcal{P}$ generating $\mathcal{B}$. For example, when X is the unit interval, the family $\mathcal{P}$ of all intervals is a π-system.

The following result is therefore very useful for checking whether a transformation is measure-preserving or not.

Theorem 1.2 *Let $(X, \mathcal{B}, \mu)$ be a probability space and let $T : X \to X$ be measurable. Let $\mathcal{P}$ be a π-system (Definition 1.2.4) that generates $\mathcal{B}$. If $\mu(T^{-1}(A)) = \mu(A)$ for any $A \in \mathcal{P}$, then T is measure-preserving. (See [5])*

A few examples of measure-preserving transformations are presented below.

Example 1.4.1 The identity transformation on $(X, \mathcal{B}, \mu)$, for any measure μ, is measure preserving.

Example 1.4.2 Let $X = [0, 1]$, $\mathcal{B}$=Borel σ-algebra of $[0, 1]$ and λ=Lebesgue measure on $[0, 1]$. Let $T : X \to X$ be defined by $T(x) = rx(\text{mod } 1)$, where r is a positive integer greater than or equal to 2. Then T is measure-preserving.

Proof. Let $[a,b] \subset [0,1]$ be a subinterval of $[0,1]$. Its preimage $T^{-1}([a,b])$ consists of r disjoint intervals $I_1, \dots, I_r$ and $\lambda(I_i) = \frac{1}{r}(b-a)$ for $i = 1, \dots, r$. Thus, $\lambda(T^{-1}([a,b])) = \lambda([a,b])$. Since the family $\mathcal{P} = \{[a,b] \subset [0,1]\}$ is a π-system generating $\mathcal{B}$, Theorem 1.2 implies that T is measure-preserving. ■

Example 1.4.3 Let $(X, \mathcal{B}, \mu)$ be as in Example 1.4.2. Define $T : X \to X$ be $T(x) = x + \alpha \pmod 1$, where $\alpha > 0$. Then T preserves Lebesgue measure.

Proof. As in Example 1.4.2 it is enough to show that $\lambda(T^{-1}([a,b])) = \lambda([a,b])$ for any subinterval $[a,b] \subset [0,1]$. The preimage $T^{-1}([a,b])$ consists of one or two disjoint intervals and $\lambda(T^{-1}([a,b])) = \lambda([a,b])$. A more natural way to view this example is to interpret it as a rotation of the circle. Then T-invariance of λ is obvious. ■

Example 1.4.4 (Two sided $(p_0, \dots, p_{k-1})$-shift)
Let $A = \{0, \dots, k-1\}$, give measure p_i to i such that $\sum_{i=0}^{k-1} p_i = 1$. We let $X = \prod_{-\infty}^{\infty} A$ together with the direct product measure we introduced in Definition 1.3.1.

Define $T : X \to X$ by $T(\{x_i\}) = \{y_i\}$, where $y_i = x_{i+1}$. T preserves the measure of each measurable rectangle, and thus, it preserves the measure of sets which are finite union of disjoint measurable rectangles. Also, the set of measurable rectangles form a π-system, therefore by Theorem 1.2, T is measure-preserving. We call T the *two-sided* $(p_0, \dots, p_{k-1})$*-shift.*

Definition 1.4.4 Let $(X, \mathcal{B}, \mu)$ be a probability space and let: $T : X \to X$ be a measure-preserving transformation. We say T is *ergodic* if for any $B \in \mathcal{B}$, such that $T^{-1}B = B$, $\mu(B) = 0$ or $\mu(X \setminus B) = 0$.

Chapter 2

Measure-Theoretic Entropy

2.1 Introduction

Entropy measures the amount of uncertainty in an experiment. It also measures the information in the experiment, or the amount one learns from the outcome of an experiment. Loosely speaking, the entropy of a transformation is a measure of randomness of its orbit structure. For a periodic system the entropy will be zero, it will be a positive number for a chaotic map, and for random maps it is infinity. ([17], [4])

In 1958 Kolmogorov [10] introduced the concept of entropy into ergodic theory. The notion of entropy now used is slighty different from that used by Kolmogorov. The improvement was made by Sinai [16].

In this chapter we define measure-theoretic entropy and present its useful properties and theorems, which are mostly taken from [19], [4], and [18].

2.2 Entropy of a Partition

Throughout, $(X, \mathcal{B}, \mu)$ will denote a probability space.

Definition 2.2.1 A *partition* of $(X, \mathcal{B}, \mu)$ is a collection of disjoint elements of $\mathcal{B}$

whose union is X.

We are interested in finite partitions, which will be denoted by $\alpha = \{A_1, A_2, \ldots, A_k\}$.

Remark 2.1 If α is a finite partition of a probability space $(X, \mathcal{B}, \mu)$, then the collection of all elements of β which are unions of elements of α is a finite sub-σ-algebra of $\mathcal{B}$. Conversely if A is a finite sub-σ-algebra of $\mathcal{B}$, say $A = \{A_i : i = 1, \ldots, n\}$, then the non-empty sets of the form $B_1 \cap \cdots \cap B_n$ where $B_i = A_i$ or $X \setminus A_i$, form a finite partition of $(X, \mathcal{B}, \mu)$. Thus, we have a one-to-one correspondence between finite partitions and finite sub-σ-algebras of $\mathcal{B}$.

Note that any finite sub-σ-algebra of $\mathcal{B}$ is in fact a finite subalgebra of $\mathcal{B}$, and vice versa. In this text we will explain our theory based on finite partitions, though the theorems will work in the same way if we use the corresponding finite sub-σ-algebras (or finite subalgebras), instead of finite partitions on their own.

Definition 2.2.2 Let α, β be two finite partitions. β is called *finer* than α, denoted by $\alpha \leq \beta$, when each element of α is a union of elements of β.

Definition 2.2.3 Given finite partitions $\alpha = \{A_1, A_2, \ldots, A_n\}$, $\beta = \{B_1, B_2, \ldots, B_k\}$ we define their *join* or their *least common refinement* as:

$$\alpha \vee \beta = \{A_i \cap B_j : 1 \leq i \leq n, 1 \leq j \leq k\}.$$

Note that $\alpha \vee \beta$ is again a finite partition. The *join* of finitely many partitions $\alpha_1, \alpha_2, \ldots, \alpha_n$ is defined by $\bigvee_{i=1}^{n} \alpha_i = \alpha_1 \vee \alpha_2 \cdots \vee \alpha_n$.

Definition 2.2.4 Suppose $T : X \to X$ is a measure-preserving transformation. If $\alpha = \{A_1, A_2, \ldots, A_k\}$, then by $T^{-n}\alpha$ we mean the partition

$$\{T^{-n}A_1, T^{-n}A_2, \ldots, T^{-n}A_k\}.$$

Proposition 2.1 *Let α, β be two finite partitions of $(X, \mathcal{B}, \mu)$. Then $\alpha \leq \beta$ if and only if $\beta = \alpha \vee \beta$.*

Proof. The result is easily proved from Definitions 2.2.3 and 2.2.2. ■

Proposition 2.2 *Let α, β be two finite partitions of $(X, \mathcal{B}, \mu)$ and $T : X \to X$ be a measure-preserving transformation. Then, for any positive integer n:*

(a) $T^{-n}(\alpha \vee \beta) = T^{-n}\alpha \vee T^{-n}\beta$.

(b) If $\alpha \leq \beta$, then $T^{-n}\alpha \leq T^{-n}\beta$.

Proof.

(a)

$$\begin{aligned} T^{-n}(\alpha \vee \beta) &= T^{-n}\{A \cap B \mid A \in \alpha,\, B \in \beta\} \\ &= \{T^{-n}(A \cap B) \mid A \in \alpha,\, B \in \beta\} \\ &= \{T^{-n}A \cap T^{-n}B \mid A \in \alpha,\, B \in \beta\} \\ &= T^{-n}\alpha \vee\ T^{-n}\beta. \end{aligned}$$

(b) Let $T^{-n}(A) \in T^{-n}\alpha$. Since $\alpha \leq \beta$ there is a collection $\gamma \subseteq \beta$ such that

$$A = \bigcup\{B \mid B \in \gamma\}.$$

Then

$$\begin{aligned} T^{-n}A &= T^{-n}(\bigcup\{B \mid B \in \gamma\}) \\ &= \bigcup\{T^{-n}B \mid B \in \gamma\} \\ &= \bigcup\{T^{-n}B \mid T^{-n}B \in T^{-n}\gamma\}. \end{aligned}$$

Therefore, $T^{-n}\alpha \leq T^{-n}\beta$.

■

Definition 2.2.5 Let $\alpha = \{A_1, A_2, \ldots, A_n\}$ be a finite partition of $(X, \mathcal{B}, \mu)$. Then the *entropy of the partition* α is defined by

$$H(\alpha) = -\sum_{i=1}^{n} \mu(A_i) \log(\mu(A_i)). \tag{2.1}$$

We use $0 \cdot \log 0 = 0$ in the calculation if $\mu(A_t) = 0$ for some t. Also, by convention log will be computed in base 2.

This means that if $A_1, A_2, \dots, A_n$ denote the outcomes of an experiment, then $H(\alpha)$ measures the *uncertainty* removed or the *information gained* by performing the experiment. The fact that uncertainty and information should be measured by the same function is reasonable because:

information gained=uncertainity removed.

$H(\alpha)$ is a measure of the uncertainty about which A_i a general point of X will belong to. The following examples will clarify this idea.

Example 2.2.1 If $\alpha = \{X\}$, then $H(\alpha) = 0$.
In this example it is certain that any $x \in X$ can only belong to X. Here α represents the outcomes of a *certain* experiment and therefore there is no uncertainty about the outcome.

Example 2.2.2 Let $X = \{1,2,3,4,5,6\}$ and μ be the probability measure, where $\mu(\{i\}) = \frac{1}{6}$ for $1 \leq i \leq 6$. The partition $\alpha = \{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\}\}$ has maximal information, since knowing which element of α contains the outcome conveys all the available information.

Using Equation 2.1

$$H(\alpha) = -\sum_{i=1}^{6} \mu(\{i\}) \log(\mu(\{i\})) = -\sum_{i=1}^{6} \frac{1}{6} \log(\frac{1}{6}) = \log 6.$$

Example 2.2.3 For the space defined in Example 2.2.2 if we take $A_1 = \{1,3,5\}$ and $A_2 = \{2,4,6\}$, the entropy of the partition $\beta = \{A_1, A_2\}$ is:

$$H(\beta) = -\sum_{i=1}^{2} \mu(A_i) \log(\mu(A_i)) = -\sum_{i=1}^{2} \frac{1}{2} \log(\frac{1}{2}) = \log 2.$$

Entropy of this partition is smaller than $\log 6$ computed for the partition in the previous example.

Example 2.2.4 If $\alpha = \{A_1, A_2, \dots, A_n\}$ where $\mu(A_i) = \frac{1}{n}$ for $1 \leq i \leq n$, then

$$H(\alpha) = -\sum_{i=1}^{n} \mu(A_i) \log(\mu(A_i)) = -\sum_{i=1}^{n} \frac{1}{n} \log(\frac{1}{n}) = \log n.$$

Thus, we gain a lot of information if n is large, i.e., since all the elements of α have equal measures there is much uncertainty about which A_i a point will belong to.

It could be asked: "What is the intuitive idea behind the definition of entropy given in Definition 2.2.5?". A reasonable answer to this question can be formulated as follows.

Suppose we have a set of possible events whose probabilities of occurrence are $p_1, p_2, \ldots, p_n$. These probabilities are known but that is all we know concerning which event will occur. Can we find a measure of how uncertain we are of an outcome?

If there is such a measure, say $H_n(p_1, p_2, \ldots, p_n)$, defined for any positive integer n with $\sum_{i=1}^n p_i = 1$ it is *reasonable* to require of it the following properties:

(a) For any positive integer n

$$H_{n+1}(p_1, \ldots, p_n, 0) = H_n(p_1, \ldots, p_n);$$

(b) continuity and symmetry: For any positive integer n, the function $H_n(p_1, p_2, \ldots, p_n)$ is continuous and symmetric with respect to all its arguments;

(c) extremal property: when all the events are equally likely the uncertainty must have its largest value, i.e.,

$$\max H_n(p_1, \ldots, p_n) = H_n(\frac{1}{n}, \ldots, \frac{1}{n});$$

(d) additivity: If $q_{i_j} \geq 0$, for $1 \leq j \leq m$ and $1 \leq i \leq n$, where $p_i = \sum_{j=1}^m q_{i_j}$ and $\sum_{i=1}^n p_i = 1$, the following additive property holds

$$H_{nm}(q_{1_1}, \ldots, q_{n_m}) = H_n(p_1, \ldots, p_n) + \sum_{i=1}^n p_i H_m(\frac{q_{i_1}}{p_i}, \ldots, \frac{q_{i_m}}{p_i}).$$

In [9] the following result is established.

Theorem 2.1 *Let us consider the sequence of non-negative functions $H_1(1)$, $H_2(p_1,p_2),\dots,$ $H_n(p_1,\dots,p_n)$ defined on the set $\{(p_1,\dots,p_n) \mid p_i \geq 0, \sum_{i=1}^n p_i = 1\}$, and assume H_n satisfies the above four assumptions. Then H_n is of the form:*

$$H_n(p_1,\dots,p_n) = -k\sum_{i=1}^{n} p_i \log p_i$$

where k is a positive constant.

If we assume $k=1$, the quantity of the form $H_n(p_1,\dots,p_n) = -\sum_{i=1}^n p_i \log p_i$ will be provided that corresponds to Definition 2.2.5. This theorem and the assumptions required for its proof are not necessary for the presentation of this thesis, and are presented only to reveal the intuition behind the definitions.

2.3 Conditional Entropy

Definition 2.3.1 Let $\alpha = \{A_1, A_2, \dots, A_n\}$ and $\beta = \{B_1, B_2, \dots, B_k\}$ be finite partitions of a probability space $(X, \mathcal{B}, \mu)$. Then the *conditional entropy of the partition α knowing β* is defined by

$$H(\alpha \mid \beta) = -\sum_{i,j} \mu(A_i \cap B_j) \cdot \log(\frac{\mu(A_i \cap B_j)}{\mu(B_j)}), \tag{2.2}$$

omitting the j-terms where $\mu(B_j)$=0.
$H(\alpha \mid \beta)$ measures the average information obtained from performing the experiment associated with α given the outcome of the experiment associated with β.

Example 2.3.1 Let X be the space defined in Examples 2.2.2 and 2.2.3 and the partitions α and β be defined by

$$\alpha = \{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\}\}$$
$$\beta = \{\{1,3,5\},\{2,4,6\}\}$$

then by Definition 2.3.1:

$$H(\alpha \mid \beta) = -\sum_{i,j} \mu(A_i \cap B_j) \cdot \log(\frac{\mu(A_i \cap B_j)}{\mu(B_j)}) = \log 3.$$

Note that knowing β, i.e., knowing if an unknown element of X is odd or even, reduces the uncertainty about this unknown element performing experiment α. Note that $H(\alpha \mid \beta) = \log 3$ is less than $H(\alpha) = \log 6$ computed in Example 2.2.2 .
Also:

$$H(\beta \mid \alpha) = -\sum_{i,j} \mu(A_i \cap B_j) \cdot \log(\frac{\mu(A_i \cap B_j)}{\mu(A_j)}) = 0.$$

It makes sense because knowing β contains no information if we already know the result of the experiment α, therefore $H(\beta \mid \alpha) = 0$.

Example 2.3.2 Let $\beta = \{X\}$ in the probability space $(X, \mathcal{B}, \mu)$. Then for any partition α we get $H(\alpha \mid \beta) = H(\alpha)$. It can simply be concluded from definition as below:

$$\begin{aligned} H(\alpha \mid \beta) &= -\sum_{i,j} \mu(A_i \cap B_j) \cdot \log(\frac{\mu(A_i \cap B_j)}{\mu(B_j)}) \\ &= -\sum_{i} \mu(A_i \cap X) \cdot \log(\frac{\mu(A_i \cap X)}{\mu(X)}) \\ &= -\sum_{i} \mu(A_i) \log(\mu(A_i)) \\ &= H(\alpha). \end{aligned}$$

It can be interpreted that because β represents the outcome of a trivial experiment, one gains no information from knowledge of it.

2.4 Properties of $H(\alpha)$ and $H(\alpha \mid \beta)$

In this section we prove some useful properties of $H(\alpha)$ and $H(\alpha \mid \beta)$.

Theorem 2.2 *Let α, β, and γ be finite partitions of a probability space $(X, \mathcal{B}, \mu)$ and $T : X \to X$ be a measure-preserving transformation. Then the following statements hold.*

(1) $0 \leq H(\alpha)$

(2) $H(\alpha) \leq \log n$ if $\alpha = \{A_1, A_2, \ldots, A_n\}$

(3) $0 \leq H(\alpha \mid \beta)$

(4) $H(\alpha \vee \beta \mid \gamma) = H(\alpha \mid \gamma) + H(\beta \mid \alpha \vee \gamma)$

(5) $H(\alpha \vee \beta) = H(\alpha) + H(\beta \mid \alpha)$

(6) if $\alpha \leq \beta$, then $H(\alpha \mid \gamma) \leq H(\beta \mid \gamma)$

(7) if $\alpha \leq \beta$, then $H(\alpha) \leq H(\beta)$

(8) if $\beta \leq \gamma$, then $H(\alpha \mid \beta) \geq H(\alpha \mid \gamma)$

(9) $H(\alpha \mid \gamma) \leq H(\alpha)$

(10) $H(\alpha \vee \beta \mid \gamma) \leq H(\alpha \mid \gamma) + H(\beta \mid \gamma)$

(11) $H(\alpha \vee \beta) \leq H(\alpha) + H(\beta)$

(12) $H(T^{-1}\alpha \mid T^{-1}\beta) = H(\alpha \mid \beta)$

(13) $H(T^{-1}\alpha) = H(\alpha)$

Proof.

(1) Let $\Phi : [0, \infty) \to \mathbb{R}$ be defined by

$$\Phi(x) = \begin{cases} 0 & \text{if } x = 0 \\ x \log x & \text{if } x \neq 0. \end{cases}$$

For any $x \in [0,1]$, $\Phi(x) \leq 0$. $(X, \mathcal{B}, \mu)$ is a probability space so $\mu(A_i) \in [0,1]$ for $1 \leq i \leq n$ and therefore $\Phi(\mu(A_i)) \leq 0$, so $\sum_{i=1}^{n} \Phi(\mu(A_i)) \leq 0$. Now by definition:

$$H(\alpha) = -\sum_{i=1}^{n} \mu(A_i) \log(\mu(A_i)) = -\sum_{i=1}^{n} \Phi(\mu(A_i)) \geq 0.$$

(2) We first show that function Φ is convex, i.e., $\Phi(\alpha x + \beta y) \leq \alpha\Phi(x) + \beta\Phi(y)$ if $x, y \in [0, \infty)$, $\alpha, \beta \geq 0$, and $\alpha + \beta = 1$.

$$\Phi'(x) = \log e + \log x$$
$$\Phi''(x) = \frac{\log e}{x} > 0 \text{ on } (0, \infty).$$

Suppose $y > x$; by the mean value theorem there exist z, and w such that,

$$\Phi(y) - \Phi(\alpha x + \beta y) = \Phi'(z)\alpha(y - x) \text{ where } \alpha x + \beta y < z < y \text{ and,}$$
$$\Phi(\alpha x + \beta y) - \Phi(x) = \Phi'(w)\beta(y - x) \text{ where } x < w < \alpha x + \beta y.$$

Since $\Phi'' > 0$ we have $\Phi'(z) \geq \Phi'(w)$, therefore

$$\begin{aligned} \beta(\Phi(y) - \Phi(\alpha x + \beta y)) &= \Phi'(z)\alpha\beta(y - x) \\ &\geq \Phi'(w)\alpha\beta(y - x) \\ &= \alpha(\Phi(\alpha x + \beta y) - \Phi(x)). \end{aligned}$$

Therefore $\Phi(\alpha x + \beta y) \leq \alpha\Phi(x) + \beta\Phi(y)$ if $x, y > 0$, and hence, also if $x, y \geq 0$ by continuity of Φ. So we proved Φ is convex.

It can be seen by induction on n that for Φ convex,

$$\Phi(\sum_{i=1}^{n} \alpha_i x_i) \leq \sum_{i=1}^{n} \alpha_i \Phi(x_i) \tag{2.3}$$

if $x_i \in [0, \infty), \alpha_i \geq 0,$ and $\sum_{i=1}^{n} \alpha_i = 1$.

Now if we put $\alpha_i = \frac{1}{n}$ and $x_i = \mu(A_i)$, $1 \leq i \leq n$ in Equation 2.3, then:

$$\begin{aligned}
&\Phi(\sum_{i=1}^{n} \frac{1}{n}\mu(A_i)) \leq \sum_{i=1}^{n} \frac{1}{n}\Phi(\mu(A_i))\\
\Longrightarrow\ &\Phi(\frac{1}{n}\sum_{i=1}^{n} \mu(A_i)) \leq \frac{1}{n}\sum_{i=1}^{n} \mu(A_i)\log(\mu(A_i))\\
\Longrightarrow\ &\Phi(\frac{1}{n}) \leq \frac{1}{n}\sum_{i=1}^{n} \mu(A_i)\log(\mu(A_i))\\
\Longrightarrow\ &\frac{1}{n}\log(\frac{1}{n}) \leq \frac{1}{n}\sum_{i=1}^{n} \mu(A_i)\log(\mu(A_i))\\
\Longrightarrow\ &-\log(\frac{1}{n}) \geq -\sum_{i=1}^{n} \mu(A_i)\log(\mu(A_i)) = H(\alpha)\\
\Longrightarrow\ &-\log(\frac{1}{n}) \geq H(\alpha)\\
\Longrightarrow\ &\log n \geq H(\alpha).
\end{aligned}$$

(3) $A_i \cap B_j \subseteq B_j$, so by properties of probability measure:

$$\begin{aligned}
&0 \leq \mu(A_i \cap B_j) \leq \mu(B_j)\\
\Longrightarrow\ &0 \leq \frac{\mu(A_i \cap B_j)}{\mu(B_j)} \leq 1 \text{ (if } \mu(B_j) \neq 0)\\
\Longrightarrow\ &\log(\frac{\mu(A_i \cap B_j)}{\mu(B_j)}) \leq 0\\
\Longrightarrow\ &\mu(A_i \cap B_j)\log(\frac{\mu(A_i \cap B_j)}{\mu(B_j)}) \leq 0\\
\Longrightarrow\ &\sum_{i,j} \mu(A_i \cap B_j)\log(\frac{\mu(A_i \cap B_j)}{\mu(B_j)}) \leq 0\\
\Longrightarrow\ &H(\alpha \mid \beta) = -\sum_{i,j} \mu(A_i \cap B_j)\log(\frac{\mu(A_i \cap B_j)}{\mu(B_j)}) \geq 0.
\end{aligned}$$

(4) Let $\alpha = \{A_i\}_{i \in I}$, $\beta = \{B_j\}_{j \in J}$,and $\gamma = \{C_k\}_{k \in K}$, where I, J, K are finite sets. Also, without the loss of generality, assume that all elements of each of these partitions have strictly positive measure. (Since if $\alpha = \{A_1, A_2, \ldots, A_n\}$ with $\mu(A_i) > 0$, $1 \leq i \leq r$ and $\mu(A_i) = 0$, $r < i \leq n$, we can replace α by

$\{A_1, \ldots, A_{r-1}, \bigcup_{i=r}^n A_i\}$.) Note:

$$\frac{\mu(A_i \cap B_j \cap C_k)}{\mu(C_k)} = \frac{\mu(A_i \cap B_j \cap C_k)}{\mu(A_i \cap C_k)} \cdot \frac{\mu(A_i \cap C_k)}{\mu(C_k)},$$

unless $\mu(A_i \cap C_k) = 0$ and then the left hand side is zero. Also, directly using Definitions 2.2.3 and 2.3.1 we get

$$H(\beta \mid \alpha \vee \gamma) = -\sum_{i,j,k} \mu(A_i \cap B_j \cap C_k) \log(\frac{\mu(A_i \cap B_j \cap C_k)}{\mu(A_i \cap C_k)}),$$

and also

$$H(\alpha \vee \beta \mid \gamma) = -\sum_{i,j,k} \mu(A_i \cap B_j \cap C_k) \log(\frac{\mu(A_i \cap B_j \cap C_k)}{\mu(C_k)}).$$

Therefore

$$\begin{aligned} H(\alpha \vee \beta \mid \gamma) &= -\sum_{i,j,k} \mu(A_i \cap B_j \cap C_k) \log(\frac{\mu(A_i \cap B_j \cap C_k)}{\mu(C_k)}) \\ &= -\sum_{i,j,k} \mu(A_i \cap B_j \cap C_k) \log(\frac{\mu(A_i \cap B_j \cap C_k)}{\mu(A_i \cap C_k)} \cdot \frac{\mu(A_i \cap C_k)}{\mu(C_k)}) \\ &= -\sum_{i,j,k} \mu(A_i \cap B_j \cap C_k) \log(\frac{\mu(A_i \cap B_j \cap C_k)}{\mu(A_i \cap C_k)}) \\ &\quad -\sum_{i,j,k} \mu(A_i \cap B_j \cap C_k) \log(\frac{\mu(A_i \cap C_k)}{\mu(C_k)}) \\ &= H(\beta \mid \alpha \vee \gamma) - \sum_{i,j,k} \mu(A_i \cap B_j \cap C_k) \log(\frac{\mu(A_i \cap C_k)}{\mu(C_k)}) \\ &= H(\beta \mid \alpha \vee \gamma) - \sum_{i,k} \mu(A_i \cap C_k) \log(\frac{\mu(A_i \cap C_k)}{\mu(C_k)}) \\ &= H(\beta \mid \alpha \vee \gamma) + H(\alpha \mid \gamma). \end{aligned}$$

(5) If we put $\gamma = \{X\}$ in (4), we get:

$$\begin{aligned} &H(\alpha \vee \beta) = H(\alpha \vee \beta \mid \{X\}) = H(\alpha \vee \beta \mid \gamma) \\ &= H(\alpha \mid \gamma) + H(\beta \mid \alpha \vee \gamma) \\ &= H(\alpha \mid \{X\}) + H(\beta \mid \alpha \vee \{X\}) \\ &= H(\alpha) + H(\beta \mid \alpha). \end{aligned}$$

(6) Note that if $\alpha \leq \beta$, then $\beta = \alpha \vee \beta$ by Proposition 2.1. Then we use (4) as below:

$$H(\beta \mid \gamma) = H(\alpha \vee \beta \mid \gamma) = H(\alpha \mid \gamma) + H(\beta \mid \alpha \vee \gamma) \geq H(\alpha \mid \gamma).$$

(7) If $\alpha \leq \beta$ by (6) $H(\alpha \mid \gamma) \leq H(\beta \mid \gamma)$ so if we put $\gamma = \{X\}$, we get:

$$H(\alpha \mid \{X\}) \leq H(\beta \mid \{X\})$$

so

$$H(\alpha) \leq H(\beta).$$

(8) Fix i, j and let

$$\alpha_k = \frac{\mu(C_k \cap B_j)}{\mu(B_j)}, \quad x_k = \frac{\mu(A_i \cap C_k)}{\mu(C_k)}$$

in the equation

$$\Phi(\sum_{i=1}^{n} \alpha_i x_i) \leq \sum_{i=1}^{n} \alpha_i \Phi(x_i),$$

as we had in proof of (2). Therefore:

$$\Phi(\sum_k \frac{\mu(C_k \cap B_j)}{\mu(B_j)} \cdot \frac{\mu(A_i \cap C_k)}{\mu(C_k)}) \leq \sum_k (\frac{\mu(C_k \cap B_j)}{\mu(B_j)} \cdot \Phi(\frac{\mu(A_i \cap C_k)}{\mu(C_k)})). \tag{2.4}$$

Also, since $\beta \leq \gamma$

$$\sum_k \frac{\mu(C_k \cap B_j)}{\mu(B_j)} \cdot \frac{\mu(A_i \cap C_k)}{\mu(C_k)} = \frac{\mu(A_i \cap B_j)}{\mu(B_j)}. \tag{2.5}$$

Therefore, by Equations 2.4, 2.5

$$\Phi(\frac{\mu(A_i \cap B_j)}{\mu(B_j)}) \leq \sum_k (\frac{\mu(C_k \cap B_j)}{\mu(B_j)} \cdot \Phi(\frac{\mu(A_i \cap C_k)}{\mu(C_k)})),$$

or equivalently

$$\frac{\mu(A_i \cap B_j)}{\mu(B_j)} \log(\frac{\mu(A_i \cap B_j)}{\mu(B_j)}) \leq \sum_k (\frac{\mu(C_k \cap B_j)}{\mu(B_j)} \cdot \Phi(\frac{\mu(A_i \cap C_k)}{\mu(C_k)})).$$

Thus, multiplying both sides by $\mu(B_j)$ we obtain

$$\mu(A_i \cap B_j)\log(\frac{\mu(A_i \cap B_j)}{\mu(B_j)}) \leq \sum_k (\mu(C_k \cap B_j) \cdot \Phi(\frac{\mu(A_i \cap C_k)}{\mu(C_k)})).$$

Now take the sum over i, j and get

$$\sum_{i,j} \mu(A_i \cap B_j)\log(\frac{\mu(A_i \cap B_j)}{\mu(B_j)}) \leq \sum_{i,j}\sum_k (\mu(C_k \cap B_j) \cdot \Phi(\frac{\mu(A_i \cap C_k)}{\mu(C_k)})).$$

Therefore,

$$\sum_{i,j} \mu(A_i \cap B_j)\log(\frac{\mu(A_i \cap B_j)}{\mu(B_j)}) \leq \sum_{i,j,k} (\mu(C_k \cap B_j) \cdot \Phi(\frac{\mu(A_i \cap C_k)}{\mu(C_k)})),$$

and by the fact that $\{B_j\}$ form a partition

$$\sum_{i,j} \mu(A_i \cap B_j)\log(\frac{\mu(A_i \cap B_j)}{\mu(B_j)}) \leq \sum_{i,k} (\mu(C_k) \cdot \Phi(\frac{\mu(A_i \cap C_k)}{\mu(C_k)})).$$

Therefore,

$$\sum_{i,j} \mu(A_i \cap B_j)\log(\frac{\mu(A_i \cap B_j)}{\mu(B_j)}) \leq \sum_{i,k} (\mu(C_k) \cdot \frac{\mu(A_i \cap C_k)}{\mu(C_k)}\log(\frac{\mu(A_i \cap C_k)}{\mu(C_k)})),$$

and

$$\sum_{i,j} \mu(A_i \cap B_j)\log(\frac{\mu(A_i \cap B_j)}{\mu(B_j)}) \leq \sum_{i,k} (\mu(A_i \cap C_k)\log(\frac{\mu(A_i \cap C_k)}{\mu(C_k)})),$$

which gives

$$-\sum_{i,j} \mu(A_i \cap B_j)\log(\frac{\mu(A_i \cap B_j)}{\mu(B_j)}) \geq -\sum_{i,k} (\mu(A_i \cap C_k)\log(\frac{\mu(A_i \cap C_k)}{\mu(C_k)})),$$

or

$$H(\alpha \mid \beta) \geq H(\alpha \mid \gamma).$$

(9) If we take $\beta = \{X\}$, then for any partition γ, $\beta \leq \gamma$. Therefore, by (8) we get $H(\alpha \mid \{X\}) \geq H(\alpha \mid \gamma)$, but $H(\alpha \mid \{X\}) = H(\alpha)$ so $H(\alpha) \geq H(\alpha \mid \gamma)$.

(10) $\gamma \leq \alpha \vee \gamma$ so by (8)

$$H(\beta \mid \alpha \vee \gamma) \leq H(\beta \mid \gamma).$$

Adding $H(\alpha \mid \gamma)$ to both sides we obtain

$$H(\alpha \mid \gamma) + H(\beta \mid \alpha \vee \gamma) \leq H(\alpha \mid \gamma) + H(\beta \mid \gamma).$$

By (4) we can change the lefthand side as

$$H(\alpha \vee \beta \mid \gamma) \leq H(\alpha \mid \gamma) + H(\beta \mid \gamma).$$

(11) If we put $\gamma = \{X\}$ in (10), we get

$$H(\alpha \vee \beta \mid \{X\}) \leq H(\alpha \mid \{X\}) + H(\beta \mid \{X\}).$$

Therefore,

$$H(\alpha \vee \beta) \leq H(\alpha) + H(\beta).$$

(12) T is a measure-preserving transformation therefore

$$\mu(A_i \cap B_j) \log(\frac{\mu(A_i \cap B_j)}{\mu(B_j)}) = \mu(T^{-1}(A_i \cap B_j)) \log(\frac{\mu(T^{-1}(A_i \cap B_j))}{\mu(T^{-1}(B_j))}).$$

Now we take the sum over i, j, and get

$$\sum_{i,j} \mu(A_i \cap B_j) \log(\frac{\mu(A_i \cap B_j)}{\mu(B_j)}) =$$

$$\sum_{i,j} \mu(T^{-1}(A_i \cap B_j)) \log(\frac{\mu(T^{-1}(A_i \cap B_j))}{\mu(T^{-1}(B_j))}),$$

and therefore by definition

$$H(\alpha \mid \beta) = H(T^{-1}\alpha \mid T^{-1}\beta).$$

(13) T is a measure-preserving transformation therefore

$$\mu(A_i) \log(\mu(A_i)) = \mu(T^{-1}A_i) \log(\mu(T^{-1}A_i)),$$

so

$$\sum_i \mu(A_i)\log(\mu(A_i)) = \sum_i \mu(T^{-1}A_i)\log(\mu(T^{-1}A_i)),$$

and hence by definition

$$H(\alpha) = H(T^{-1}\alpha).$$

■

2.5 Entropy of a Measure Preserving Transformation

Lemma 2.1 *If $\{a_n\}_{n\geq 1}$ satisfies $a_n \geq 0$, and $\forall n, m$ $a_{n+m} \leq a_n + a_m$, then $\lim_{n\to\infty} \frac{a_n}{n}$ exists and equals $\inf_n \frac{a_n}{n}$.*

Proof. Fix $m > 0$. For each $j > 0$, $j = km + n$ where $0 \leq n < m$. Then

$$\frac{a_j}{j} = \frac{a_{n+km}}{n+km} \leq \frac{a_n}{km} + \frac{a_{km}}{km} \leq \frac{a_n}{km} + \frac{ka_m}{km} = \frac{a_n}{km} + \frac{a_m}{m}.$$

As $j \to \infty$, then $k \to \infty$ so

$$\limsup \frac{a_j}{j} \leq \frac{a_m}{m},$$

and therefore

$$\limsup \frac{a_j}{j} \leq \inf \frac{a_m}{m},$$

but

$$\inf \frac{a_m}{m} \leq \liminf \frac{a_j}{j}.$$

Thus, $\lim \frac{a_j}{j}$ exists and equals to $\inf \frac{a_j}{j}$. ■

Theorem 2.3 *Let α be a finite partition of $(X, \mathcal{B}, \mu)$ and T be a measure-preserving transformation. Then the $\lim_{n\to\infty} \frac{1}{n} H(\bigvee_{i=0}^{n-1} T^{-i}\alpha)$ exists.*

Proof. Let $a_n = H(\bigvee_{i=0}^{n-1} T^{-i}\alpha) \geq 0$.

$$\begin{aligned} a_{n+m} &= H(\bigvee_{i=0}^{n+m-1} T^{-i}\alpha) \\ &\leq H(\bigvee_{i=0}^{n-1} T^{-i}\alpha) + H(\bigvee_{i=n}^{n+m-1} T^{-i}\alpha) \quad \text{(by (11) of Theorem 2.2)} \\ &= a_n + H(\bigvee_{i=0}^{m-1} T^{-i}\alpha) \quad \text{(by (13) of Theorem 2.2)} \\ &= a_n + a_m. \end{aligned}$$

We now can apply Lemma 2.1 and, therefore, $\lim_{n\to\infty} \frac{1}{n} H(\bigvee_{i=0}^{n-1} T^{-i}\alpha)$ exists. ■

Definition 2.5.1 Let α be a finite partition of $(X, \mathcal{B}, \mu)$ and T a transformation which preserves μ. The *entropy of the transformation T with respect to the partition α* is defined by

$$h(T, \alpha) = \lim_{n\to\infty} \frac{1}{n} H(\alpha \vee T^{-1}\alpha \vee \cdots \vee T^{-(n-1)}\alpha) = \lim_{n\to\infty} \frac{1}{n} H(\bigvee_{i=0}^{n-1} T^{-i}\alpha).$$

Note that in Theorem 2.3 we showed that this limit always exists.

This definition means that if we think of an application of T as a passage of one day (as a unit) of time, then $\bigvee_{i=0}^{n-1} T^{-i}\alpha$ represents the combined experiment of performing the original experiment represented by α on n consecutive days. $h(T, \alpha)$ is then the average of information per day that one gets from performing the original experiment daily forever.

Definition 2.5.2 Let T be a measure-preserving transformation on $(X, \mathcal{B}, \mu)$. The *entropy of the transformation T* is defined by $h(T) = \sup_\alpha h(T, \alpha)$ where the sup is taken over all finite partitions α contained in $\mathcal{B}$ of $(X, \mathcal{B}, \mu)$. The entropy $h(T)$ is called the *metric entropy, measure-theoretic entropy, or Kolmogorov-Sinai entropy of T*. $h(T)$ is sometimes denoted by $h(T, \mu)$.

Again if we think of an application of T as a passage of one day (as a unit) of time, $h(T)$ will be the maximum average information per day obtainable by performing a finite experiment.

2.6 Properties of $h(T)$ and $h(T,\alpha)$

In this section we prove important properties of $h(T)$ and $h(T,\alpha)$.

Theorem 2.4 *Let T be a measure-preserving transformation on the probability space $(X,\mathcal{B},\mu)$, and $\alpha=\{A_1,A_2,\dots,A_n\}$, $\beta=\{B_1,B_2,\dots,B_m\}$ be finite partitions of this space. Then the following statements hold:*

(1) $0\le h(T,\alpha)\le H(\alpha)\le \log(n)$;

(2) $h(T,\alpha\vee\beta)\le h(T,\alpha)+h(T,\beta)$;

(3) if $\alpha\le\beta$, then $h(T,\alpha)\le h(T,\beta)$;

(4) $h(T,\alpha)\le h(T,\beta)+H(\alpha\mid\beta)$;

(5) if T is invertible and $m\ge 1$, then $h(T,\alpha)=h(T,\bigvee_{i=-m}^{m}T^{i}\alpha)$.

Proof.

(1) In (1) of Theorem 2.2 we proved $0\le H(\alpha)$. Therefore, $0\le H(\bigvee_{i=0}^{n-1}T^{-i}\alpha)$ and thus, $0\le \lim_{n\to\infty}\frac{1}{n}H(\bigvee_{i=0}^{n-1}T^{-i}\alpha)$ or $0\le h(T,\alpha)$.

To prove $h(T,\alpha)\le H(\alpha)$:

$$\begin{aligned}\frac{1}{n}H\Big(\bigvee_{i=0}^{n-1}T^{-i}\alpha\Big)&\le\frac{1}{n}\sum_{i=0}^{n-1}H(T^{-i}\alpha)\quad\text{(by (11) of Theorem 2.2)}\\&=\frac{1}{n}\sum_{i=0}^{n-1}H(\alpha)\quad\text{(by (13) of Theorem 2.2)}\\&=H(\alpha).\end{aligned}$$

Therefore, $\lim_{n\to\infty}\frac{1}{n}H(\bigvee_{i=0}^{n-1}T^{-i}\alpha)\le H(\alpha)$ and thus, $h(T,\alpha)\le H(\alpha)$.

The inequality $H(\alpha)\le\log(n)$ is already proved in (2) of Theorem 2.2.

(2)

$$H(\bigvee_{i=0}^{n-1} T^{-i}(\alpha \vee \beta)) = H(\bigvee_{i=0}^{n-1} T^{-i}\alpha \vee \bigvee_{i=0}^{n-1} T^{-i}\beta)$$

(by (11) of Theorem 2.2.)

$$H(\bigvee_{i=0}^{n-1} T^{-i}(\alpha \vee \beta)) \leq H(\bigvee_{i=0}^{n-1} T^{-i}\alpha) + H(\bigvee_{i=0}^{n-1} T^{-i}\beta).$$

Therefore, dividing both sides by n

$$\frac{1}{n}H(\bigvee_{i=0}^{n-1} T^{-i}(\alpha \vee \beta)) \leq \frac{1}{n}H(\bigvee_{i=0}^{n-1} T^{-i}\alpha) + \frac{1}{n}H(\bigvee_{i=0}^{n-1} T^{-i}\beta),$$

so

$$\lim_{n\to\infty} \frac{1}{n}H(\bigvee_{i=0}^{n-1} T^{-i}(\alpha \vee \beta)) \leq \lim_{n\to\infty} \frac{1}{n}H(\bigvee_{i=0}^{n-1} T^{-i}\alpha) + \lim_{n\to\infty} \frac{1}{n}H(\bigvee_{i=0}^{n-1} T^{-i}\beta).$$

By the definition

$$h(T, \alpha \vee \beta) \leq h(T, \alpha) + h(T, \beta).$$

(3) If $\alpha \leq \beta$, then $T^{-i}\alpha \leq T^{-i}\beta$. Thus, $\bigvee_{i=0}^{n-1} T^{-i}\alpha \leq \bigvee_{i=0}^{n-1} T^{-i}\beta$.

Now, by (7) of Theorem 2.2

$$H(\bigvee_{i=0}^{n-1} T^{-i}\alpha) \leq H(\bigvee_{i=0}^{n-1} T^{-i}\beta).$$

Then,

$$\frac{1}{n}H(\bigvee_{i=0}^{n-1} T^{-i}\alpha) \leq \frac{1}{n}H(\bigvee_{i=0}^{n-1} T^{-i}\beta).$$

Thus,

$$\lim_{n\to\infty} \frac{1}{n}H(\bigvee_{i=0}^{n-1} T^{-i}\alpha) \leq \lim_{n\to\infty} \frac{1}{n}H(\bigvee_{i=0}^{n-1} T^{-i}\beta).$$

Therefore, by the definition

$$h(T, \alpha) \leq h(T, \beta).$$

(4) By (7) of Theorem 2.2, we can write

$$H(\bigvee_{i=0}^{n-1} T^{-i}\alpha) \leq H((\bigvee_{i=0}^{n-1} T^{-i}\alpha) \vee (\bigvee_{i=0}^{n-1} T^{-i}\beta)),$$

which by (5) of Theorem 2.2

$$= H(\bigvee_{i=0}^{n-1} T^{-i}\beta) + H(\bigvee_{i=0}^{n-1} T^{-i}\alpha \mid \bigvee_{i=0}^{n-1} T^{-i}\beta).$$

Now, by (10) of Theorem 2.2

$$H((\bigvee_{i=0}^{n-1} T^{-i}\alpha) \vee (\bigvee_{i=0}^{n-1} T^{-i}\beta)) \leq \sum_{i=0}^{n-1} H(T^{-i}\alpha \mid (\bigvee_{j=0}^{n-1} T^{-j}\beta)),$$

which by (8) of Theorem 2.2

$$\leq \sum_{i=0}^{n-1} H(T^{-i}\alpha \mid T^{-i}\beta),$$

and applying (12) of Theorem 2.2

$$= nH(\alpha \mid \beta).$$

Thus,

$$H(\bigvee_{i=0}^{n-1} T^{-i}\alpha) \leq H(\bigvee_{i=0}^{n-1} T^{-i}\beta) + nH(\alpha \mid \beta).$$

Now, we divide both sides of this inequality by $\frac{1}{n}$ and take the limit as $n \to \infty$, and obtain

$$h(T, \alpha) \leq h(T, \beta) + H(\alpha \mid \beta).$$

(5)

$$h(T, \bigvee_{i=-m}^{m} T^{i}\alpha) = \lim_{k\to\infty} \frac{1}{k} H(\bigvee_{j=0}^{k-1} T^{-j}(\bigvee_{i=-m}^{m} T^{i}\alpha))$$
$$= \lim_{k\to\infty} \frac{1}{k} H(\bigvee_{i=-m}^{m+k-1} T^{-i}\alpha).$$

Thus, we know

$$h(T, \bigvee_{i=-m}^{m} T^{i}\alpha) = \lim_{k\to\infty} \frac{1}{k} H(\bigvee_{i=-m}^{m+k-1} T^{-i}\alpha). \tag{2.6}$$

On the other hand

$$H(\bigvee_{i=-m}^{m+k-1} T^{-i}\alpha) = H(\bigvee_{i=-m}^{m} T^{-i}\alpha \vee \bigvee_{i=m}^{m+k-1} T^{-i}\alpha).$$

Applying (5) of Theorem 2.2

$$= H(\bigvee_{i=m}^{m+k-1} T^{-i}\alpha) + H((\bigvee_{i=-m}^{m} T^{-i}\alpha) \mid (\bigvee_{i=m}^{m+k-1} T^{-i}\alpha)).$$

Now, by (13) of Theorem 2.2

$$= H(\bigvee_{i=0}^{k-1} T^{-i}\alpha) + H((\bigvee_{i=-m}^{m} T^{-i}\alpha) \mid (\bigvee_{i=m}^{m+k-1} T^{-i}\alpha)).$$

Therefore, we can write

$$\lim_{k\to\infty} \frac{1}{k} H(\bigvee_{i=-m}^{m+k-1} T^{-i}\alpha) =$$

$$\lim_{k\to\infty} (\frac{1}{k} H(\bigvee_{i=0}^{k-1} T^{-i}\alpha) + \frac{1}{k} H((\bigvee_{i=-m}^{m} T^{-i}\alpha) \mid (\bigvee_{i=m}^{m+k-1} T^{-i}\alpha))). \tag{2.7}$$

So, by Equations 2.6 and 2.7, we can write

$$h(T, \bigvee_{i=-m}^{m} T^{i}\alpha) = \lim_{k\to\infty} (\frac{1}{k} H(\bigvee_{i=0}^{k-1} T^{-i}\alpha) + \frac{1}{k} H((\bigvee_{i=-m}^{m} T^{-i}\alpha) \mid (\bigvee_{i=m}^{m+k-1} T^{-i}\alpha))). \tag{2.8}$$

Now, we prove

$$\lim_{k\to\infty} \frac{1}{k} H((\bigvee_{i=-m}^{m} T^{-i}\alpha) \mid (\bigvee_{i=m}^{m+k-1} T^{-i}\alpha)) = 0. \tag{2.9}$$

In fact by (9) of Theorem 2.2

$$\frac{1}{k} H((\bigvee_{i=-m}^{m} T^{-i}\alpha) \mid (\bigvee_{i=m}^{m+k-1} T^{-i}\alpha)) \leq \frac{1}{k} H(\bigvee_{i=-m}^{m} T^{-i}\alpha),$$

and therefore Equation 2.9 holds, if we take the limit of both sides as $k \to \infty$.

Now, by Equations 2.8 and 2.8

$$h(T, \bigvee_{i=-m}^{m} T^{i}\alpha) = \lim_{k\to\infty} \frac{1}{k} H(\bigvee_{i=0}^{k-1} T^{-i}\alpha).$$

Therefore,

$$h(T, \bigvee_{i=-m}^{m} T^{i}\alpha) = h(T, \alpha).$$

■

In the following theorem we prove two properties of $h(T)$.

Theorem 2.5 *Let T be a measure-preserving transformation of the probability space $(X, \mathcal{B}, \mu)$. Then the following statements hold:*

(1) for $m > 0$, $h(T^m) = mh(T)$;

(2) if T is invertible, then $h(T^m) = |m|h(T)$ $\forall m \in Z$.

Proof.

(1) We first show that

$$h(T^m, \bigvee_{i=0}^{m-1} T^{-i}\alpha) = mh(T, \alpha).$$

This holds since

$$\lim_{k\to\infty} \frac{1}{k} H(\bigvee_{j=0}^{k-1} T^{-mj}(\bigvee_{i=0}^{m-1} T^{-i}\alpha)) = \lim_{k\to\infty} \frac{m}{km} H(\bigvee_{i=0}^{km-1} T^{-i}\alpha) = mh(T, \alpha).$$

Thus,

$$\begin{aligned} mh(T) = m \sup_{\alpha \text{ finite}} \{h(T, \alpha)\} &= \sup_{\alpha \text{ finite}} \{h(T^m, \bigvee_{i=0}^{m-1} T^{-i}\alpha)\} \\ &\le \sup_{\beta} h(T^m, \beta) = h(T^m). \end{aligned}$$

Also,

$$h(T^m, \alpha) \le h(T^m, \bigvee_{i=0}^{m-1} T^{-i}\alpha) = mh(T, \alpha),$$

by (3) of Theorem 2.4 and therefore, $h(T^m) \leq mh(T)$. The result follows from these two inequalities.

(2) It suffices to show that $h(T^{-1}) = h(T)$. All we need to show is that $h(T^{-1}, \alpha) = h(T, \alpha)$ for all finite partitions α, but

$$H(\bigvee_{i=0}^{n-1} T^i\alpha) = H(T^{-(n-1)} \bigvee_{i=0}^{n-1} T^i\alpha),$$

and by (13) of Theorem 2.2

$$= H(\bigvee_{j=0}^{n-1} T^{-j}\alpha).$$

■

2.7 Calculation of Entropy

The computation of $h(T)$ is in general a difficult task. However, the computation becomes feasible in those cases when there exists a finite partition α for which $h(T) = h(T, \alpha)$. The Kolmogorov-Sinai theorem which we will study in this section provides a method of computing the entropy in a large number of cases.

In this section we introduce the Kolmogorov-Sinai theorem, and its corollaries. Also we present two examples, which show how to compute the entropy by applying these theorems.

Definition 2.7.1 If $\mathcal{C}$ and $\mathcal{D}$ are sub-σ-algebras of $\mathcal{B}$ where $(X, \mathcal{B}, \mu)$ is a probability space, then we write $\mathcal{C} \leq^\circ \mathcal{D}$ if $\forall\ C \in \mathcal{C}$, $\exists\ D \in \mathcal{D}$ such that $\mu(C \triangle D) = 0$. (Note that $C \triangle D = (C - D) \bigcup (D - C)$.)

If both $\mathcal{C} \leq^\circ \mathcal{D}$ and $\mathcal{D} \leq^\circ \mathcal{C}$, then we write $\mathcal{C} \overset{\circ}{=} \mathcal{D}$.

Let α and β be finite partitions of $(X, \mathcal{B}, \mu)$. Also, assume that the corresponding sub-σ-algebras of α, β are respectively $\mathcal{A}$, $\mathcal{B}$, in the way we introduced them in Remark 2.1. Then we write $\alpha \leq^\circ \beta$ if and only if $\mathcal{A} \leq^\circ \mathcal{B}$.

Definition 2.7.2 If $(X, \mathcal{B}, \mu)$ is a probability space and $\{\alpha_n\}$ a sequence of partitions of this space, that are not necessarily finite, by $\bigvee_n \alpha_n$ we mean the smallest sub-σ-algebra of $\mathcal{B}$ which contains $\bigcup_n \alpha_n$.

Theorem 2.6 *(Kolmogorov-Sinai Theorem) [19]*
Let $T : (X, \mathcal{B}, \mu) \to (X, \mathcal{B}, \mu)$ be an invertible measure-preserving transformation and let α be a finite partition of X contained in $\mathcal{B}$ and $\bigvee_{n=-\infty}^{\infty} T^n\alpha \overset{\circ}{=} \mathcal{B}$. Then $h(T) = h(T, \alpha)$.

Theorem 2.7 *[19] Suppose $T : (X, \mathcal{B}, \mu) \to (X, \mathcal{B}, \mu)$ is a measure-preserving transformation (not necessarily invertible) and α is a finite partition of this space contained in $\mathcal{B}$ with $\bigvee_{n=0}^{\infty} T^n\alpha \overset{\circ}{=} \mathcal{B}$. Then $h(T) = h(T, \alpha)$.*

Entropy can be defined for any at-most-countable partition as follows.

Definition 2.7.3 If $\alpha = \{A_1, A_2, \dots\}$ is an at-most-countable partition of the space $(X, \mathcal{B}, \mu)$ then

$$H(\alpha) = -\sum_i \mu(A_i)\log(\mu(A_i))$$

which may be infinite.

Definition 2.7.4 An at-most-countable partition α of the space$(X, \mathcal{B}, \mu)$ is called a *generator* for an invertible measure-preserving transformation T if $\bigvee_{n=-\infty}^{\infty} T^n\alpha \overset{\circ}{=} \mathcal{B}$.

Remark 2.2 As in Theorem 2.6 one can prove that if α is a generator and $H(\alpha) < \infty$, then $h(T) = h(T, \alpha)$. (Note α is not necessarily finite as we defined the generator.) (See [19])

Theorem 2.8 *[19] In the space $(X, \mathcal{B}, \mu)$, let $\{\alpha_n\}$ be an increasing sequence of finite partitions contained in $\mathcal{B}$, i.e.,*

$$\alpha_1 \leq^\circ \alpha_2 \leq^\circ \alpha_3 \leq^\circ \dots$$

and $\bigvee_{n=1}^{\infty} \alpha_n \overset{\circ}{=} \mathcal{B}$. Then $h(T) = \lim_{n\to\infty} h(T, \alpha_n)$.

Example 2.7.1 The two-sided $(p_0, \dots, p_{k-1})$-shift has entropy $-\sum_{i=0}^{k-1} p_i \cdot \log p_i$.

Let $X = \prod_{-\infty}^{\infty}\{0, 1, \dots, k-1\}$ and T be the shift as we introduced in 1.4.4. Let $A_i = \{\{x_k\} : x_0 = i\}$, $0 \le i \le k-1$. Then $\alpha = \{A_0, \dots, A_{k-1}\}$ is a partition of X. By definition of $\mathcal{B}$, $\bigvee_{i=-\infty}^{\infty} T^i\alpha = \mathcal{B}$.

By Kolmogorov-Sinai Theorem 2.6

$$h(T) = \lim_{n\to\infty} \frac{1}{n} H(\alpha \vee T^{-1}\alpha \vee \cdots \vee T^{-(n-1)}\alpha).$$

A typical element of $\alpha \vee T^{-1}\alpha \vee \cdots \vee T^{-(n-1)}\alpha$ is

$$A_{i_0} \cap T^{-1}A_{i_1} \cap \cdots \cap T^{-(n-1)}A_{i_{n-1}} = \{\{x_n\} : x_0 = i_0, x_1 = i_1, \dots, x_{n-1} = i_{n-1}\}$$

which has measure $p_{i_0} \cdot p_{i_1} \cdots p_{i_{n-1}}$. Thus,

$$\begin{aligned} &H(\alpha \vee T^{-1}\alpha \vee \cdots \vee T^{-(n-1)}\alpha) \\ &= - \sum_{i_0,\dots,i_{n-1} \in \{0,1,\dots,k-1\}} (p_{i_0} \cdots p_{i_{n-1}}) \cdot \log(p_{i_0} \cdots p_{i_{n-1}}) \\ &= - \sum_{i_0,\dots,i_{n-1} \in \{0,1,\dots,k-1\}} (p_{i_0} \cdots p_{i_{n-1}})[\log p_{i_0} + \cdots + \log p_{i_{n-1}}] \\ &= -n \sum_{i=0}^{k-1} p_i \cdot \log p_i. \end{aligned}$$

Therefore,

$$h(T) = h(T, \alpha) = \lim_{n\to\infty} \frac{1}{n} \cdot (-n \sum_{i=0}^{k-1} p_i \cdot \log p_i) = -\sum_{i=0}^{k-1} p_i \cdot \log p_i.$$

Example 2.7.2 Let $T : [0, 1) \to [0, 1)$ be given by $T(x) = 2x \pmod 1$. Below we show that $h(T) = \log 2$.

The set of dyadic rationals are dense on the unit interval. The collection of intervals of the form $[a, b)$ with dyadic rational end points generated the class $\mathcal{B}$ of Borel sets of $[0, 1]$.

Let $\alpha = \{[0, \frac{1}{2}), [\frac{1}{2}, 1)\}$, and let $\alpha_n = \bigvee_{i=0}^{n-1} T^{-i}\alpha$. Each α_n is of the form

$$\alpha_n = \{[0, \frac{1}{2^n}), [\frac{1}{2^n}, \frac{2}{2^n}), \dots, [\frac{2^n - 1}{2^n}, 1)\}.$$

Let [a,b) be an arbitrary diadic rational interval where

$$a = \frac{1}{2^{n_1}} + \frac{1}{2^{n_2}} + \cdots + \frac{1}{2^{n_s}}$$

and

$$b = \frac{1}{2^{m_1}} + \frac{1}{2^{m_2}} + \cdots + \frac{1}{2^{m_k}}.$$

Since

$$a = \frac{2^{n_s - n_1} + 2^{n_s - n_2} + \cdots + 1}{2^{n_s}}$$

and

$$b = \frac{2^{m_k - m_1} + 2^{m_k - m_2} + \cdots + 1}{2^{m_k}},$$

[a,b) is a union of elements of α_n for $n = \max\{m_k, n_s\}$. Thus every open set in $[0,1)$ is an at most countable union of elements of $\cup_{n=1}^{\infty}\alpha_n$. Thus $\mathcal{B} \subseteq \vee_{n=1}^{\infty}\alpha_n$. Since every element of α_n is a Borel set $\vee_{n=1}^{\infty}\alpha_n \subseteq \mathcal{B}$. Therefore $\vee_{n=1}^{\infty}\alpha_n = \mathcal{B}$.

Now by Kolmogorov-Sinai Theorem 2.6 $h(T) = h(T,\alpha)$. With properties of Lebesgue measure we have,

$$H(\bigvee_{i=0}^{n-1} T^{-i}\alpha) = -2^n(\frac{1}{2^n}\log(\frac{1}{2^n})) = n\log 2.$$

Then $h(T) = h(T,\alpha) = \lim_{n\to\infty}\frac{1}{n}(n\log 2) = \log 2.$

Chapter 3

Computing Entropy Directly from the Definition

3.1 Introduction

As we stated earlier the computation of $h(T)$ is in general a difficult task. Although the Kolmogorov-Sinai Theorem provides a computational tool to compute the entropy in many cases, finding a generating partition is not always possible. Even if we have a generating partition α for a system, evaluating $h(T, \alpha)$ is not always easy.

There have also been other approaches to numerically evaluate the metric entropy of a system. In this chapter we describe an algorithm developed in [17]. The approach is directly based on the definition of metric entropy. It includes methods of partitioning the data, computing sequential distributions, and compactifying results to reduce memory requirements, which will be discussed in greater depth.

Our method is even feasible when the system is not known explicitly and we just have knowledge of a finite trajectory of a system. That is, we can compute the entropy from time series generated from the system.

In this chapter it is always assumed that $(X, \mathcal{B}, \mu)$ is a probability space and $T : X \to X$ is a measure-preserving transformation of this space.

In our computational examples we will assume that the dynamical system is one dimensional and bounded, i.e., X is a bounded subset of the real line. Later in Chapter 5 we explain how this method can be applied in higher dimensions.

3.2 Entropy Computing Algorithm

Definition 3.2.1 The elements of a partition are called *bins*, and the number of bins in a partition will be called *size of the partition*, or *partition-size*. For example $\alpha_0, \alpha_1, \ldots, \alpha_{k-1}$ are the bins of the partition α of size k defined below:

$$\alpha = \{\alpha_0, \alpha_1, \ldots, \alpha_{k-1}\}.$$

First we go back to the original definition of the metric entropy to be able to present the direct entropy computation method.

By definition, $h(T) = \sup_\alpha h(T, \alpha)$ where the sup is taken over all finite partitions α contained in $\mathcal{B}$, so

$$h(T) = \sup_{\alpha}(\lim_{n\to\infty} \frac{1}{n} H(\alpha \vee T^{-1}\alpha \vee \cdots \vee T^{-(n-1)}\alpha)). \tag{3.1}$$

A typical element of $\alpha \vee T^{-1}\alpha \vee \cdots \vee T^{-(n-1)}\alpha$ is of the form

$$a_0 \cap T^{-1}a_1 \cap \cdots \cap T^{-(n-1)}a_{n-1} \ ,$$

where $a_0, a_1, \ldots, a_{n-1}$ are (not necessarily distinct) bins of α.

Notation 3.1 For a probability space $(X, \mathcal{B}, \mu)$ with a measure-preserving transformation T of this space, and α a partition of this space, where $a_0, \ldots, a_n$ are (not necessarily distinct) bins of α, we denote $p(a_0, a_1, \ldots, a_{n-1})$ as the probability that

$$x \in a_0 \cap T^{-1}a_1 \cap \cdots \cap T^{-(n-1)}a_{n-1},$$

i.e.,

$$\begin{aligned}p(a_0, a_1, \ldots, a_{n-1}) &= \mu(\{x \in X \mid x \in a_0 \cap T^{-1}a_1 \cap \cdots \cap T^{-(n-1)}a_{n-1}\}) \\ &= \mu(\{x \in X \mid x \in a_0, x \in T^{-1}a_1, \ldots, x \in T^{-(n-1)}a_{n-1}\}) \\ &= \mu(\{x \in X \mid x \in a_0, T(x) \in a_1, \ldots, T^{n-1}(x) \in a_{n-1}\}).\end{aligned}$$

In other words, $p(a_0, a_1, \ldots, a_{n-1})$ is the probability that the iterations of a point $x \in X$ visit the bins $a_0, a_1, \ldots, a_{n-1}$ respectively at times $0, 1, \ldots, n-1$, iterated by T at each discrete time interval.

Using notation (3.1) and the definition of the entropy of a partition

$$H(\alpha \vee T^{-1}\alpha \vee \cdots \vee T^{-(n-1)}\alpha) =$$

$$- \sum_{a_0, a_1, \ldots, a_{n-1} \in \alpha} p(a_0, a_1, \ldots, a_{n-1}) \log(p(a_0, a_1, \ldots, a_{n-1})). \tag{3.2}$$

Therefore, by Equations 3.1, 3.2

$$h(T) = \sup_\alpha [\lim_{n\to\infty} (-\frac{1}{n} \sum_{a_0, a_1, \ldots, a_{n-1} \in \alpha} p(a_0, a_1, \ldots, a_{n-1}) \log(p(a_0, a_1, \ldots, a_{n-1})))]. \tag{3.3}$$

For simplicity we define new notations which will be used throughout.

Notation 3.2 We define $I_n(\alpha)$ by

$$I_n(\alpha) = - \sum_{a_0, a_1, \ldots, a_{n-1} \in \alpha} p(a_0, a_1, \ldots, a_{n-1}) \log(p(a_0, a_1, \ldots, a_{n-1})). \tag{3.4}$$

Using Notation 3.2 in Equation 3.3 we can write:

$$h(T) = \sup_\alpha [\lim_{n\to\infty} \frac{I_n(\alpha)}{n}]. \tag{3.5}$$

Also, note that by this new notation we can write

$$h(T, \alpha) = \lim_{n\to\infty} \frac{I_n(\alpha)}{n}. \tag{3.6}$$

Now, we are ready to summarize the method of computing entropy.

Algorithm 3.3 Computing Entropy by Definition.

1- Choose a partition α of X.

2- Compute the probabilities $p(a_0, a_1, \ldots, a_{n-1})$.

3- Compute $I_n(\alpha)$ using the probabilities.

4- Evaluate $\lim_{n\to\infty} \frac{I_n(\alpha)}{n}$, i.e., $h(T, \alpha)$.

5- Choose some new partition α and perform steps 2 to 4 for this new partition.

6- Find the maximum of the values $h(T, \alpha)$ computed for each different partition, which estimates $h(T)$.

3.3 Introducing Symbolic Representation and Considering the Algorithm

In order to make the computation easier, it is desirable to introduce a new representation of our space called *symbolic representation*. We associate with each element of a partition a symbol, e.g., an integer between 0 to $k-1$ inclusively, for partition size of k. In this way it is possible to represent symbolically the orbit of a point by noting in which element of the partition it lies at each iteration of the transformation.

In our estimation we only compute a long but finite orbit of the system, i.e., we only know the values of $x, T(x), T^2(x), \ldots, T^{npt-1}(x)$ where *npt* is a positive integer. Therefore the entropy computing method is only based on the information about this finite orbit. We assume that the probabilistic behavior of the finite orbit $x, T(x), T^2(x), \ldots, T^{npt-1}(x)$ is a good estimate for the probabilistic behavior of an infinite orbit.

In symbolic terms the finite orbit $x, T(x), T^2(x), \ldots, T^{npt-1}(x)$ is in fact a finite sequence of length *npt* of integer numbers, each of them between 0 to k-1 inclusively. From now on by *main sequence* we mean the finite sequence of numbers $x, T(x), T^2(x), \dots, T^{npt-1}(x)$ represented symbolically.

As we said before $p(a_0, a_1, \dots, a_{n-1})$ is the probability that the iterations of a point $x \in X$ visit the bins $a_0, a_1, \dots, a_{n-1}$ respectively at times $0, 1, \dots, n-1$, iterated by T at each discrete time interval. In symbolical terms this probability is equivalent to the appearance of the sequence $a_0, a_1, \dots, a_{n-1}$ in the main sequence. Therefore $p(a_0, a_1, \dots, a_{n-1})$ (also denoted by $p(a_0 a_1 \dots a_{n-1})$) can be computed as the fraction:

$$\frac{\text{number of times the sequence } a_0, a_1, \dots, a_{n-1} \text{ appears in the main sequence}}{npt - n + 1} \tag{3.7}$$

Note that $npt - n + 1$ is the number of all possible appearance times of a sequence of length n in a main-sequence of length npt.

Definition 3.3.1 For main sequence $x, T(x), T^2(x), \dots, T^{npt-1}(x)$ and a positive integer n, the *separation level* (of sequences of length n) is defined by the fraction:

$$\frac{\text{number of distinct sequences of length } n\text{, that appear in the main sequence}}{\text{total possible number of sequences of length } n \text{ in the main sequence}}$$

which is equal to

$$\frac{\text{number of distinct sequences of length } n\text{, that appear in the main sequence}}{npt - n + 1}$$

The following must be noted about the Algorithm 3.3.

(a) We use the symbolic representation to implement this algorithm as we discussed.

(b) It may be asked: What will be the best choice of a partition α for which we get better estimates for h(T)? As we discussed earlier finding a generating partition is not always possible. In absence of any knowledge of the availability of a generating partition for a dynamical system, we can only take finer partitions to better approximate the generating partition. As was pointed out in [17], the use of finer partitions improves the accuracy of the entropy estimation. It must be noted that for a partition of k bins, the maximum possible entropy computed will be $\log k$.

(c) To compute $I_n(\alpha)$ we need to compute the probabilities $p(a_0a_1 \ldots a_{n-1})$ using the symbolic representation and formula of the Equation 3.7, for all combinations of $a_0, a_1, \ldots, a_{n-1}$ from the set of symbols $0, 1, \ldots, k-1$. Therefore we are required to compute k^n different probability vectors, and it is evident that a memory overflow would occur as n grows. In Chapter 5, we introduce a computational approach called sequencing to prevent this overflow problem.

(d) To estimate the entropy we need to compute $\lim_{n\to\infty} \frac{I_n(\alpha)}{n}$ for each partition α. In practice we only can compute $\frac{I_n(\alpha)}{n}$ up to a finite integer. The entropy in our case will be estimated by the slope of the line asymptotic to the data on the graph of information $I_n(\alpha)$ against n. Theoretically, for an infinite amount of data, such a line exists. In practice, with a finite amount of data, after a certain number of steps the information $I_n(\alpha)$ approaches its maximum information for the number of sequences under consideration. Therefore, we need to estimate the entropy by determining a linear region on the information graph. It will be discussed that the best region will be after the initial rise and before 0.2 separation-level (of sequences of length n)(See Definition 3.3.1) is reached in the graph of $I_n(\alpha)$ against n.

In Chapter 5 we will discuss these facts in greater depth.

3.4 A Step By Step Example

Before we explain any further details about our method, we present an example.

Example 3.4.1 Consider $T(x) = 2x \pmod 1$ on the space of the $([0,1), \mathcal{B}, \mu)$ where μ is the Lebesgue measure and $\mathcal{B}$ is the σ-algebra of the Lebesgue measurable sets on $[0,1)$. As we showed in Example 2.7.2, $h(T) = \log 2$. In this example we show how one can implement the Algorithm 3.3 to evaluate $h(T)$.

1) Sample of computing $I_n(\alpha)$ with main-sequence length of 50 and Partition-size 4

First we divide $[0, 1)$ into 4 bins of equal size,

$$\alpha = \{[0, \frac{1}{4}), [\frac{1}{4}, \frac{2}{4}), [\frac{2}{4}, \frac{3}{4}), [\frac{3}{4}, 1)\}.$$

Now we begin from a partition which symbolically is represented as $\{0, 1, 2, 3\}$ and number of bins=4.

For example we start with a main-sequence of length 50. We choose a random point in [0,1] and perform 49 iterations of T, i.e., $npt = 50$ and at each step find the corresponding partition element in which the iterations lie in. For a random starting point we have got the following main-sequence of length 50 as a result of computation

12132000132120012001321213320133212132012132001201

Next, we calculate the $I_n(\alpha)$ for $n = 1, 2, \ldots, 6$ using Equation (3.7).

To compute $I_1(\alpha)$, we compute the following probabilities.
$p(0) = \frac{12}{50}$, $p(1) = \frac{15}{50}$, $p(2) = \frac{14}{50}$, and $p(3) = \frac{9}{50}$. Therefore,

$$I_1(\alpha) = -\sum_{a_0 \in \alpha} p(a_0) \log(p(a_0)) \approx 1.974752133.$$

Similarly to compute $I_2(\alpha)$, we need to compute $p(a_0a_1)$ for all $a_0, a_1 \in \alpha$, The partition size is 4 in this case, and there are two variables a_0, and a_1 so totally there will be 4^2 probabilities to be computed. We use Equation 3.7, and compute the probabilities $p(a_0a_1)$. The nonzero probabilities are as below.
$p(00) = \frac{5}{49}$, $p(01) = \frac{7}{49}$, $p(12) = \frac{7}{49}$, $p(13) = \frac{7}{49}$,
$p(20) = \frac{7}{49}$, $p(21) = \frac{7}{49}$, $p(32) = \frac{7}{49}$, $p(33) = \frac{2}{49}$.

Now, we use Equation 3.4 and compute $I_2(\alpha)$:

$$I_2(\alpha) = -\sum_{a_0, a_1 \in \alpha} p(a_0a_1) \log(p(a_0a_1)) \approx 2.930657860.$$

The separation level (see Definition 3.3.1) of sequences of length $n = 2$ will be $\frac{8}{49} = 0.1632653061$.

Table 3.1: Results for Example 3.4.1

n	$I_n(\alpha)$	separation level
1	1.999792548	0.0004000000
2	2.999621310	0.0008000800
3	3.999218215	0.0016003200
4	4.998433746	0.0032009602
5	5.996722438	0.0064025610
6	6.992898709	0.0128064032
7	7.983791036	0.0256153692
8	8.965941542	0.0512358651
9	9.926743303	0.1024819856
10	10.848083900	0.2041837654
11	11.668770160	0.3748748749
12	12.301696270	0.5740314346
13	12.735451060	0.7456948338
14	12.991958510	0.8594172424
15	13.135091170	0.9261966753

2) Increasing main-sequence length to 10000, and computing $I_n(\alpha)$ and separation level

To get a better estimate of the entropy $h(T)$ we increase the length of main-sequence from 50 to 10000 and perform the same computations. It means we start from a longer finite orbit. In Table 3.1 we have computed $I_n(\alpha)$ and *separation level* for n from 1 to 15. Note that we are still working with the initial partition-size= 4.

Figure 3.1 shows the values of $I_n(\alpha)$ plotted against n. It can be seen that the slope of the graph is falling off after $n = 9$ starting at which the separation-level exceeds 0.2.

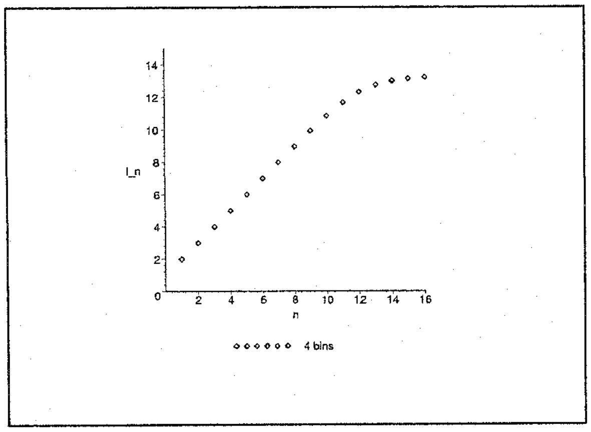

Figure 3.1: Graph of $I_n(\alpha)$ for $n = 1, \ldots, 15$, $npt = 10000$, partition-size$= 4$

To estimate entropy $h(T, \alpha)$ we find the slope of the best line fit to data as an estimate of Entropy. If we use least squares method to find the slope of the best fit line to the points on the graph of Figure 3.1 with separation level< 0.2, i.e., $(I_n(\alpha), n)$ for $n = 1, \ldots, 9$, we get the

$$\text{slope of the best fitting line to data} = .9928395725.$$

Therefore, we estimated $h(T, \alpha) = .9928395725$.

3) Computing over different partition sizes, and entropy estimation.

At this step we perform the same computation as in the previous part we used to find $h(T, \alpha)$, for different partitions α. The new partitions will be of the size $2, 8, 16, 64$. Similar to the case we studies in the previous part of this example, for each partition-size k, we partition the interval $[0, 1]$ into k disjoints parts of the same size. In Figure 3.2, we have plotted $I_n(\alpha)$ against the sequence-size of n, for these different partitions. Each partition size represents a new partition α. Note that again we work with a main sequence of length of 10000.

To compute $h(T, \alpha)$ again we have to find the slope of the best line fitting the data in the linear region of the graph. This region, as we said, could be seen in part of the graph where separation-level is less than 0.2. For each partition size we compute its corresponding slope and we get the following results:

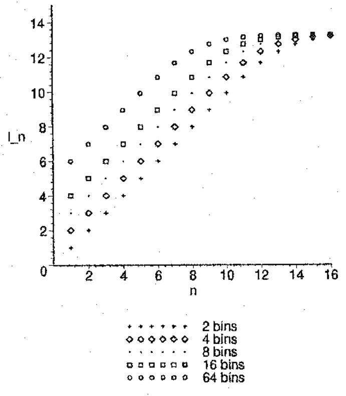

Figure 3.2: Graph of $I_n(\alpha)$ for $n = 1, \ldots, 15$, $npt = 10000$, and different partition-sizes

In cases where α is a partition of size:

2 bins, $h(T, \alpha) = 0.9939654661$;

4 bins, $h(T, \alpha) = 0.9928395725$;

8 bins, $h(T, \alpha) = 0.9913716708$;

16 bins, $h(T, \alpha) = 0.9894412532$;

64 bins, $h(T, \alpha) = 0.9832916560$.

The entropy $h(T)$ is the maximum of the computed values $h(T, \alpha)$.

The maximum over these values is .9939654661. If we compare it to $\log 2$, (the value of $h(T)$ computed theoretically), we get $\log 2 - 0.9939654661 = 0.0060345339$. As we see in this case the method gives a very good approximation of $h(T)$.

Chapter 4

Computational Approach

4.1 Introduction

The calculation of the entropy presents several difficulties for numerical studies. These can be attributed to problems associated with partitioning of the data and keeping track of the sequences. For a point that goes through the sequence of bins $a_0, a_1, \ldots, a_{n-1}$, the sequence can be considered as a point in the Euclidean space of n dimensions by using the bin labels to create an n-vector. Then for an initial partition of size k bins, there are k^n boxes in the equivalent n dimensional space.

For fine partitions it is evident that as n grows a memory overflow would occur even in a large computer if every n dimensional box were stored, so it becomes imperative that some compactification scheme should be used. In fact we have used this scheme in parts (2) and (3) of Example 3.4.1 to compute $I_n(\alpha)$ for different values of n when we increased the value of *npt* from 50 to 10000, though we did not explain how we had performed the computations.

Below we set out our approach in several sections: 1-Partitioning and Scaling Data; 2-Sequencing and Compactification; 3-Probabilities and Information, 4-Entropy Estimation.

We will use the Maple instructions to explain our algorithms in these sections.

From now on we write the variables with the new font to make our Maple codes more readable, e.g., we write k, npt, and n respectively for, k, npt, and n. For arrays (or vectors) the convention has been chosen that when a Greek index is used, e.g., A[α], we are referring to the array itself, and when a Latin index is used, e.g., A[i], we are referring to a particular element of A[α].

4.2 Partitioning and Scaling Data

For computational efficiency, it is imperative that the partitioning of data be accomplished quickly and efficiently. We assume that a trajectory of a dynamical system, i.e., x, $T(x)$, $\dots$, T^{npt-1}, is stored in a vector x[α] of length npt. So elements of x[α] are x[i] ($0 \leq$ i $<$ npt).

We must convert the data into a representation of integer values. Since our systems are assumed to be bounded, the data lies in the interval [x_min,x_max]. We first shift the data to interval [0,x_max-x_min] by subtracting x_min from every element of x[α]. The Maple code corresponding to the shifting process is simply written in the following way:

```
for i from 0 to npt-1 do
 x[i]:= x[i]-x_min:
od:
length_of_domain:=x_max-x_min:
for i from 0 to npt-1 do
 x[i]:= x[i]/length_of_domain:
od:
```

Now we partition the data into a partition of size k. In other words we assign an integer from 0 to k-1 inclusive to each element of the array x[α]. To do so we take an array bin[α] of size npt, which elements (bin[i], $0 \leq$ i $<$ npt) are integers from 0 to k-1 inclusively. This scaling can be implemented in the following way using

Maple's `floor` function which returns the greatest integer less than or equal to a number.

```
for i from 0 to npt-1 do
    bin[i]:= floor(x[i]*k):
od:
```

Array bin[α] exhibits the same relative properties as the initial input data. Conversely, any partition of the original data is equivalent to a partition of the data in bin[α] simply by scaling the partition in the same manner as the original data was scaled.

It is evident that this process has no effect on the calculation of the entropy, since the entropy depends solely on the sequence of bins through which a point moved, and this is equivalent under the identification of the original data with the scaled data. The identification gives an isomorphism between the initial data and the initial partition, and the scaled data in bin[α] and the scaled partition; and the entropy is invariant under isomorphism.

The elements of bin[α] are all taken to be starting points of sequences, so it is clear that to determine the sequence of bins followed by a particular point, it is necessary is to look at the elements of bin[α] following that starting point. To calculate a sequence of probability $p(a_0, a_1, \ldots, a_{n-1})$, we count the number of times the particular sequence of numbers $a_0, a_1, \ldots, a_{n-1}$ appears in bin[α]. This is not as simple as it appears, since it requires keeping track of k^n data points, which soon overwhelms the memory of a computer. The resolution of this problem is discussed below in section 4.3.

4.3 Sequencing and Compactification

For a given partition into k bins, it was mentioned above that a particular sequence of bins $a_0, a_1, \ldots, a_{n-1}$ can be represented by a point in an Euclidean space of n dimensions, i.e., on an integral grid k$\times$ k $\times$ $\ldots\times$ k, where the product is taken n times. This simply means that if p is a point on the n-dimensional grid with coordinates $(a_0, a_1, \ldots, a_{n-1})$, then $a_0, a_1, \ldots, a_{n-1}$ represents a possible sequence of bins through which a point on the space might move in n time steps.

Consequently, it would be a simple matter to tabulate all the sequences of length n on the space, if we could create an n-dimensional array with each dimension of length k and use the elements of bin[α] to index the array and increment the proper positions. Let such array be called Hist[$\alpha_1, \alpha_2, \ldots \alpha_n$], with n arguments, i.e., each of its elements is in the form Hist$[a_0, a_1, \ldots, a_{n-1}]$, where each a_i could vary from 0 to k-1 inclusively.

Then the tabulation is given by incrementing
Hist[bin[i],bin[i+1],...,bin[i+(n-1)]], where i varies over all starting points, which would be from 0 to npt-n inclusively. In other words, in the algorithm, for i from 0 to npt-n we perform the following increment

```
Hist[bin[i],bin[i+1],...,bin[i+(n-1)]]:=
        Hist[bin[i],bin[i+1],...,bin[i+(n-1)]]+1
```

The problem of course is that this requires k to power of n memory positions. Luckily, many of these positions have zero entries (corresponding to no such sequence occurring); in fact, the maximum number of non-zero entries is npt-n+1. This allows us to compactify the information of the array at each stage of computation to minimize the memory requirements.

We will now describe a simple algorithm for compactifying the sequencing information stored in Hist[$\alpha_1, \alpha_2, \ldots \alpha_n$]. While there are probably more efficient approaches, this one has two important benefits. First, it requires no more than a fixed amount of memory and only a two dimensional array with a maximum size of npt $\times$ k. Second, it does not lose the sequencing information, whereas any algorithm which utilized a sorting routine would have to overcome the problem of re-ordering and the subsequent loss of sequences.

The algorithm will be described using a two dimensional histogram array histogram[β, γ]. To begin the sequencing process, create a new vector compress[α] of size npt, and initialize it so that compress[i]=bin[i], for i from 0 to npt-1.

```
for i from 0 to npt-1 do
   compress[i]:= bin[i]:
 od:
```

We also need another variable to keep track of non-zero positions in histogram[β, γ]; call it newcount. Initially we set newcount=k. Note that k was the number of bins. The histogram[β, γ] array can be either dynamically allocated to its working size k*newcount or set to the maximum size of k*npt. The first step is to increment histogram[β, γ] for two-sequences (where we introduce the term n-sequence to mean a sequence of length n) according to the following

```
histogram[bin[i+1],compress[i]]:=histogram[bin[i+1],compress[i]]+1
```
(4.1)

where $0 \leq$ i $\leq$ (npt-2) is the range of i. Every two-sequence increments a particular position in histogram[β, γ], and identical two-sequences increment the same position in histogram[β, γ], so the histogram[β, γ] array represents the distribution of two-sequences.

We now make use of the possibility of interpreting a two-dimensional array as a one-dimensional vector (as is generally done in most computers when storing arrays in memory) by concatenating either over rows or columns.

For example, if A[α] is an s$\times$t array, then a position in A[α] may be labeled by A[i,j] for $0 \leq$ i $<$ s, $0 \leq$ j $<$ t, or by concatenating over columns as A[j*s+i], or by rows as A[i*t+j]. This property will be used to set up a correspondence between two-sequences and positions in the histogram[β, γ] array where we replace the starting points in compress[α] by their relative positions in histogram[β, γ], labelled by concatenation over rows. To do this, simply make the replacement

```
compress[i]:=compress[i]+newcount*bin[i+1]
```
(4.2)

for all i in the range $0 \leq$ i $\leq$ (npt-2). No information is lost in this process, since all unique two-sequences produce unique values for the associated elements in compress[α]. When this is completed, compress[i] contains the position in histogram[β, γ] corresponding to the two-sequences starting at the associated point bin[i]. Note that none of the zero entries in histogram[β, γ] are reflected in compress[α], so the numbers in compress[α] do not, in general, come from a contiguous interval of integers. In order to minimize the memory requirements, it is necessary to renumber the values of compress[α], so that if there are p unique values, they will be renumbered from 0 to p-1, with their order maintained.

The compression process is fairly simple. First designate a variable, newcount, to count the number of nonzero entries in histogram[β, γ]. Then in one pass through histogram[β, γ], replace the old value of histogram[i,j] with the current value of newcount and then increment newcount.

```
rowlength:=newcount:
newcount:=0:
for i from 0 to k-1 do
 for j from 0 to rowlength-1 do
   if histogram[i,j]>0 then
      histogram[i,j]:=newcount:
      newcount:=newcount+1:
   fi:
 od:
od:
```

At the end of this step, the nonzero entries in histogram[β, γ] represent their relative orders of appearance in histogram[β, γ].

Now the compression can begin. The numbers in compress[α] represent the old bin numbers previously in histogram[β, γ], but we now replace them with the ordered numbers places in histogram[β, γ] in the last step.

```
for i from 0 to npt-2 do
  column:= (compress[i] mod rowlength):
  row:=   floor(compress[i]/rowlength):
  compress[i]:=histogram[row,column]:
od:
```

which results in the elements of compress[α] being numbered from 0 to newcount-1.

At the next stage of the calculation, when we want to consider three-sequences, the working size of histogram[β,γ] need only be k $\times$newcount. The compress[α] vector again holds the correct numbers to allow for use of the histogram[β,γ] incrementing routine (4.1), where the next value in the sequence is selected from bin[α], i.e., bin[i+2]. This process continues recursively, where for n-sequences we use bin[i+(n-1)] and change the range of i to 0 $\leq$ i $\leq$ (npt-n) in (4.1) and (4.2), and in the above compression codes.

Note that if the allocation of histogram[β,γ] is dynamic within the sequencing loop, it is necessary to free the reserved memory space before going on to the next step.

The complete Maple code related to sequencing and compactification is given in the Appendix.

4.4 Probabilities and Information

After going through the n-sequence compression loop, the compress[α] vector contains values that uniquely identify the n-sequences followed by the starting points. Consequently, to calculate sequential probabilities all that must be done is to tabulate the relative frequencies of occurrence of the different sequences. This can be done simply by using the elements of compress[α] as indices to increment positions in a probability vector. So we create a vector, probability[α], of length newcount (since that is the number of unique values in `compress[`α`]` and increment the elements as below:

```
for i from 0 to npt-n do
  probability[compress[i]]:=probability[compress[i]]+1:
od:
```

When the incrementing is completed, it is necessary to normalize the probabilities so the total sum is 1. The normalization will be performed as below:

```
for i from 0 to newcount-1 do
  probability[i]:=probability[i]/(npt-n+1):
od:
```

Once the probabilities are computed, it is an easy task to compute the information. The calculation of the information of `n`-sequences following from the original definition of entropy, will be given in a loop by

```
information:=0:
for i from 0 to newcount-1 do
  information:= information+probability[i]*ln(probability[i]):
od:
```

where `information` is the variable representing information. From the definition of entropy it is evident that the result of the loop calculation above must be multiplied by `(-1)`, so the last step is to take

```
information:=(-1)* information:
```

At this stage the information content of `n`-sequences has been computed. It is now possible to continue on to calculate the information content in `n+1` sequences by keeping `compress[`α`]` and the value of `newcount`, clearing `histogram[`β, γ`]`, and looping back through the operations in the last two Sections.

4.5 Entropy Estimation

All the above calculations were done for a given partition. From the information computations described above, it is possible to plot a graph of information vs sequence length. In theory, for an amount of infinite data, this graph should be asymptotic to a line, the slope of which corresponds to the entropy. However, with finite amounts of data this is not the case and the information graph begins to approach the maximum information state for the number of sequences under consideration. This causes the graph to fall off from the line defining the entropy, approaching the slope=0 state near the maximum information. In certain situations, notably difference equations, there tends to be a noticeable linear region before the falloff toward maximum information.

It has been considered in [17] that the region before 0.2 separation-level is reached will give good experimental results. So in this case we will simply plot a best-fit line through the points in the linear region to estimate the entropy. This corresponds to $\lim_{n\to\infty} \frac{I_n(\alpha)}{n}$. For some systems, where there is not such a distinctive linear region, we use $\lim_{n\to\infty} I_{n+1}(\alpha) - I_n(\alpha)$ to estimate the entropy. In fact in this case we find the difference $I_{n+1}(\alpha) - I_n(\alpha)$, for the maximum value of n for which the separation level < 0.2.

A sample code of entropy estimation using least-squares method is attached in the Appendix.

4.6 Computational Efficiency

The computational requirements of this approach can be calculated easily. Subtracting the minimum of data and the scaling data requires $O($`npt`$)$ operations. The one-dimensional histogramming requires just one pass trough the data, so it is also $O($`npt`$)$. Incrementing the two-dimensional array `histogram[`β, γ`]` is likewise accomplished in $O($`npt`$)$ operations. Then renumbering `compress[`α`]` requires one pass through `histogram[`β, γ`]` which has `k newcount` elements, where `k` is fixed by the choice of the number of initial bins and `newcount` can never grow larger than `npt`, although the calculation should be cut off soon after `newcount`$>$ 0.20$\times$`npt`. For `k`$<<$`npt`, the renumbering is O(`npt`), or for larger `k`, O(`k`$\times$`npt`). Incrementing `probability[`α`]` required `npt` operations and calculating `information` requires `3`$\times$`newcount` operations. Consequently, this implementation is accomplished in O(`npt`) or O(`k`$\times$`npt`) operations.

The memory requirements involve only one large array `histogram[`β,γ`]`, all others being vectors of length $\leq$`npt`. The working size of `histogram[`β,γ`]` is `k`$\times$`newcount` and can not grow beyond its maximum size of `n`$\times$`npt`; actually if the calculation is cut off soon after the 20% separation level has been reached, then the size of `histogram[`β,γ`]` is $\leq$`k`$\times$`npt`$\times$`A`, for `A`$\approx$`0.2`.

Chapter 5

Computational Approach in Two-Dimensions and Filtering Entropy

5.1 Introduction

When dealing with a system modelled by a discrete time, nonlinear difference equation,

$$x_{n+1} = T(x_n), \tag{5.1}$$

the method described in Chapters 3, 4 provides an algorithm for computing metric entropy. However, when the system is contaminated by noise,

$$x_{n+1} = T(x_n) + \xi_n, \tag{5.2}$$

its entropy, in general, is infinite, while that of the underlying deterministic source system is finite. To extract the entropy of the dynamical system from the noisy data, computational techniques have been developed [1], but these are often difficult to implement and their accuracy cannot easily be verified. In [11], there is a conjecture that the metric entropy of the deterministic dynamical system is the difference between the entropy of the noisy system and the entropy of the noise itself. The basic assumption is that the large (often infinite) entropies of the noisy system and the noise itself will cancel to reveal the entropy of the chaotic system. However, there are no proofs in [11] and it is not clear under what conditions, if any, the method applies. Furthermore, the proposed method assumes a special structure for the noise. In [3] a model is proposed in which this conjecture is true in a fairly general setting.

We describe this model below and give examples in which we have implemented the algorithm of [17]. In fact we explained this algorithm in Chapters 3, 4 in the one-dimensional case. In this chapter we need to implement it in the two-dimensional case.

In Section 5.2 we state the notation and formulate the problem with the aid of a skew product representation. Also, it contains the formula for the entropy of the skew product which in this special setting is similar to Adler's natural formula . In Section 5.3, using the entropy formula of Section 5.2, we state that the entropy of the chaotic map T can be filtered from the data of the noisy system. In Section 5.4 we present the method for computing the dynamical entropy from observed data, and we present a computational example that verifies our method.

5.2 Notation and Entropy of Skew Products

Definition 5.2.1 Let $(X, \mathcal{A}, \sigma, \pi)$ be a dynamical system and let $(Z, \mathcal{B}, \phi_x, \mu_x)_{x\in X}$ be a family of dynamical systems such that the function ϕ_x is $X \times Z$ measurable. A *skew product* of σ and $\{\phi_x\}_{x\in X}$ is a transformation $F : X \times Z \to X \times Z$ defined by

$$F(x, z) = (\sigma(x), \phi_x(z)),$$

$x \in X$, $z \in Z$.

In [2], Adler remarked that the natural conjecture for the formula of the entropy of a skew product is:

$$h(F,\nu) = h(\sigma,\pi) + \int_X h(\phi_x,\mu_x)d\pi(x), \tag{5.3}$$

where $h(F,\nu)$, $h(\sigma,\pi)$ and $h(\phi_x,\mu_x)$ are the entropies of F, σ, ϕ_x, respectively.

Now, we define the skew product we will use in the sequel.

Definition 5.2.2 Let $([0,1],\mathcal{A},\tau,\lambda)$ be an ergodic dynamical system, where λ is the Lebesgue measure. Let $([0,1],\mathcal{B},T,\mu)$ be an ergodic dynamical system. We assume that both τ and T are piecewise C^1. We consider the following *perturbation* of T:

$$T_x(z) = T(z) + g(x) \pmod 1,$$

where g is piecewise C^1 and $|g'| \leq M$, $M < \infty$. Consider the family of dynamical systems, $([0,1],\mathcal{B},T_x,)_{x\in[0,1]}$. We assume that $|\tau'| > \sup_x \sup_z |T_x'(z)|$. The *skew product* of τ and T_x is the transformation

$$S : [0,1]\times[0,1] \to [0,1]\times[0,1]$$

defined by

$$S(x,z) = (\tau(x), T_x(z)).$$

We assume that S preserves the two dimensional measure $\nu = \lambda \times \mu_R$ and that the system $([0,1]\times[0,1],\mathcal{A}\times\mathcal{B},S,\nu)$ is ergodic. Then, μ_R satisfies the equation $\mu_R(A) = \int_0^1 \mu_R(T_x^{-1}(A))d\lambda(x)$, for any measurable set $A \subset [0,1]$. In particular, this holds if all T_x preserve the same measure μ_R.

For the second iterate S^2 we have:

$$\begin{aligned} S^2(x,z) &= (\tau^2(x), T_{\tau(x)} \circ T_x(z)) \\ &= (\tau^2(x), T(T(z)+g(x)) + g(\tau(x)). \end{aligned} \tag{5.4}$$

In general,

$$S^n(x,z) = (\tau^n(x), R^n(z)), \tag{5.5}$$

where

$$R^n(z) = T_{\tau^{n-1}(x)} \circ \cdots \circ T_{\tau(x)} \circ T_x(z).$$

The skew product S is our model for a random map

$$R = T_x(z) \text{ with probability } \lambda,$$

i.e., x is chosen according to Lebesgue measure.

The following Theorem, proved in [3], gives a formula similar to Equation (5.3) for this skew product.

Theorem 5.1 *[3] $h(S,\nu) = h(\tau,\lambda) + \int_0^1 \int_0^1 \log|T_x'(z)| d\mu_R(z) d\lambda(x)$. If all T_x preserve the same measure μ_R, then $h(S,\nu) = h(\tau,\lambda) + \int_0^1 h(T_x,\mu_R) d\lambda(x)$.*

5.3 Dynamical Entropy Estimation

In this section, we describe a method to filter the entropy of the chaotic map T from the total entropy of the noisy system. The second component of the skew product $S(x,z)$ is what we observe as a data contaminated with noise

$$S^n(x,z) = (\tau^n(x), T_{\tau^{n-1}(x)} \circ \cdots \circ T_{\tau(x)} \circ T_x(z)). \tag{5.6}$$

Let us define a new skew product whose second component represents only the noise:

Definition 5.3.1 Let $Q(x,z)$ be a transformation from $[0,1] \times [0,1]$ to itself given by

$$Q(x,z) = (\tau(x), I_x(z)),$$

where

$$I_x(z) = I(z) + g(x) \pmod 1,$$

and I is the identity map from the unit interval $[0,1]$ to itself. Then,

$$Q^n(x,z) = (\tau^n(x), I_{\tau^{n-1}(x)} \circ \cdots \circ I_{\tau(x)} \circ I_x(z)). \tag{5.7}$$

Proposition 5.1 $h(Q, \lambda \times \lambda) = h(\tau, \lambda)$.

Proof. Using Theorem 5.1, we have

$$h(Q, \lambda \times \lambda) = h(\tau, \lambda) + \int_0^1 \int_0^1 \log|I_x'(z)| d\lambda(z) d\lambda(x). \tag{5.8}$$

Observe that $I_x'(z) = I'(z) = 1$. Thus, the double integral is equal to 0 and $h(Q, \lambda \times \lambda) = h(\tau, \lambda)$. ■

Definition 5.3.2 We define the entropy of the noisy system by $h_{\text{total}} \equiv h(S, \nu)$, the entropy of the noise by $h_{\text{noise}} \equiv h(Q, \lambda \times \lambda)$ and the dynamical entropy as the difference between the total entropy of the noisy system and the entropy of the noise; i.e.,

$$\begin{aligned} h_{\text{dyn}} &= h_{\text{total}} - h_{\text{noise}} = h(S, \nu) - h(Q, \lambda \times \lambda) \\ &= \int_0^1 \int_0^1 \log|T_x'(z)| d\mu_R(z) d\lambda(x). \end{aligned} \tag{5.9}$$

This definition of the dynamical entropy is based on the definition in [11]. The following Proposition states that that the main conjecture of [11] holds for our model.

Proposition 5.2 *[3] Suppose that $T(z)$ is a piecewise monotonic map which is eventually expanding. Then*

$$h_{dyn} \to h(T, \mu) \qquad \text{as } \sup_x |g(x)| \to 0.$$

5.4 Dynamical Entropy Computation

In general, the entropy of a noisy system and the entropy of the noise itself are very large. We now present an example where we compute the entropy of a chaotic map T and the dynamical entropy h_{dyn} of T contaminated by noise. Since, by Proposition 5.1, $h_{\text{noise}} = h(Q, \lambda \times \lambda) = h(\tau, \lambda)$, we calculate $h(\tau, \lambda)$ rather than $h(Q, \lambda \times \lambda)$. We use our algorithm discussed in Chapters 3,4 to compute $h(S, \nu)$ and $h(\tau, \lambda)$ and then we find $h_{\text{dyn}} = h(S,\nu) - h(\tau,\lambda)$. To compute $h(S,\nu)$ and $h(\tau,\lambda)$, we consider the time series

$$w_{n+1} = T(w_n) + g(x_n) \pmod 1, \tag{5.10}$$

where

$$x_{n+1} = \tau(x_n). \tag{5.11}$$

$h(\tau,\lambda)$ is computed using time series (5.11), and to compute $h(S,\nu)$ we evaluate the entropy of the two-dimensional time series (x_n, w_n) as defined above. In our example we verify that

$$h_{\text{dyn}} \approx h(T,\mu) \quad \text{as } \sup_x |g(x)| \to 0,$$

i.e., $h_{\text{dyn}} \approx h(T,\mu)$ if the effect of the noise on the data is sufficiently small. Note that in the following example we could compute $h(S,\nu)$ and $h(\tau,\lambda)$ directly using the definitions of functions T and τ, but we actually use the time series (5.10) and (5.11) to compute the entropy. We chose this method since in real-world problems T and τ are not necessarily determined explicitly, and we only know the time series (5.10) and (5.11).

A few small changes should be applied to the program in the Appendix to compute $h(S,\nu)$ using the entropy of the two-dimensional time series (x_n, w_n). Before we present our example, we clarify what changes have to be made in the code.

(a) In map definition code we also define definitions of g,τ. For example:

```
> #****************Map Definition*********************#
> t:= x -> frac(2*x):
> x_max:=1:
> x_min:=0:
> randomval:=evalf(rand(1..10^10)/10^10):
> startx:=randomval(): #random point to start with#
> #**************************************************#
> g:= x -> x:
> taw:= x -> epsilon*(frac(13/epsilon(x+epsilon/2))-0.5):
> #**************************************************#
> randomval:=evalf(rand(1..10^10)/10^10):
> startw:=randomval(): #random point to start with#
```

(b) In the partitioning code, when we are iterating data, we compute both of time series `w[i]` and `x[i]`.

```
> #*************Iterating****************************#
> for i from 0 to npt-1 do
>   w[i]:=startw:
>   x[i]:=startx:
>   startw:=evalf( frac(frac(t(startw)+g(startx))+1) ):
>   startx:=evalf(taw(startx)):
> od:
> #**************************************************#
```

(c) We partition the intervals of each dimension into `k` bins. Therefore, we will have `k`×`k` bins in 2-dimensional case. This makes it necessary for us to change the part of the code, related to symbolic representation of data. In two-dimensional case we take `bin[i]` as two-dimensional state of a point `p` at each iteration. Therefore `bin[i]` can take `k^2` different states from `0` to `k^2-1`. The following code shows how we have simply made the appropriate changes.

```
> #*********Symbolic Representation of Data***********#
> for i from 0 to npt-1 do:
> bin[i]:=min(floor(w[i]*k),k-1)+k*min(floor(x[i]*k),k-1);
> od:
> k:=k^2:
```

```
> #**************************************************#
> for i from 0 to npt-1 do
>   compress[i]:= bin[i]:
> od:
> #**************************************************#
> newcount:=k:
> #**************************************************#
```

The rest of the code in sequencing and compactification and entropy computing remains the same.

Example 5.4.1 In this example we verify that

$$h_{\text{dyn}} \approx h(T, \mu) \quad \text{as } \sup_x |g(x)| \to 0,$$

i.e., $h_{\text{dyn}} \approx h(T, \mu)$ if the effect of the noise on the data is sufficiently small. We use the algorithm we discussed to compute the entropy of the noisy system and the entropy of the noise itself from the time series.

We consider the piecewise linear transformation,

$$T(x) = 2x \text{ (mod1)}.$$

As the noise generator we use the map $x \mapsto 13x$ (mod 1), scaled and shifted to act on the interval $[-\varepsilon/2, \varepsilon/2]$:

$$\tau_\varepsilon(x) = \epsilon(\frac{13}{\epsilon}(x + \frac{\epsilon}{2}) \text{ (mod 1)}) - \frac{\epsilon}{2}.$$

Let $g(x) = x$. Using the algorithms we discussed, from time series (5.10) and (5.11) we compute values $h(S, \nu)$ and $h(\tau, \lambda)$, for different values of ϵ close to zero. Note that $-\frac{\epsilon}{2} < x_n < \frac{\epsilon}{2}$ and if $\epsilon \to 0$, $\sup_x |g(x)| = \sup_x |x| \to 0$. Therefore for small values of ϵ we should have $h_{\text{dyn}} \approx \log 2$. Table 5.1 shows the results of our computation for different values of ϵ. As expected (see Proposition 5.2) $h_{\text{dyn}} - h(T, \mu) \approx 0$ for small values of ε. The values of $h(S)$ and $h(\tau)$ in Table 5.1 are estimated by considering time series containing 100000 iterations. We use initial partition size of 15 bins to compute $h(\tau, \lambda)$ in one dimension, and $15 \times 15 = 225$ bins of the initial partition size to compute $h(S)$ in the two dimensional case. To evaluate each entropy value, we compute the slope of the best fit line by using the first (at most 15) points (before the 20% separation limit is reached) on the plot of information against sequence length.

Table 5.1: Results for Example 5.4.1

ϵ	$h(S)$	$h(\tau)$	h_{dyn}	$h_{\text{dyn}} - h(T, \mu)$
0.2	4.998999137	3.699403629	1.299595508	0.2995955076
0.1	5.034553762	3.699403629	1.335150133	0.3351501325
0.06	4.982158785	3.699403629	1.282755156	0.2827551564
0.05	4.958605763	3.699403629	1.259202134	0.2592021341
0.005	4.711355661	3.699403629	1.011952032	0.0119520321
0.00001	4.659583096	3.699403629	.9601794665	-0.0276014906
0.0000001	4.658729063	3.699403629	.9593254343	-0.0406745658
0.000000001	4.658131143	3.699403629	.9587275137	-0.0412724864

We check the precision of our computations at $\epsilon = 0.000000001$, with respect to $\log 2 = 1$. h_{dyn} calculated by this method is equal to 0.9587275137. The percentage error is equal to

$$|\frac{0.9587275137 - \log 2}{\log 2}| = 0.0412724863 \approx 4\%. \tag{5.12}$$

Bibliography

[1] Abarbanel, H.D.I., *Analysis of Observed Chaotic Data*, Springer-Verlag, 1996.

[2] Adler, R., *A note on the entropy of skew product transformations*, Proc. Amer. Math. Soc., (1962), pp. 665-669.

[3] Bahsoun, W., Góra, P., Boyarsky, A., and Ebrahimi, M., *Filtering Entropy*, To Appear in Physica D, (2003).

[4] Billingsly, P. , *Ergodic Theory and Information*, Wiley, New York, (1965).

[5] Boyarsky, A. and Góra, P., *Laws of Chaos: Invariant Measures and Dynamical Systems in One Dimension*, Brikhauser, New York, (1997).

[6] Crutchfield, J.P. and Packard, N.H., *Symbolic Dynamics of Noisy Chaos*, Physica D, Vol. 7, (1983), No. 1-3, pp. 201-223.

[7] Curry, J. H., *On Compupting the Entropy of the Henon Attractor*, Journal of Statistical Physics, Vol. 26, No. 4, (1981).

[8] Denker, M. and Grillenberger, C.,*Ergodic Theory on Compact Spaces* , Lecture Notes in Mathematics, No. 527, Springer-Verlag, Berlin, (1976).

[9] Guiasu, S., *Information Theory with Applications*, McGraw-Hill, (1977).

[10] Kolmogorov, A. N., *A New Metric Transient Dynamical System and Automorphisms of Lebesgue Spaces*, Doklady Akademii Nauk SSSR, Vol. 119, pp. 861-864, (1958); english summary, Math Rev. 21, 386 (1960).

[11] Ostruszka, A., Pakoński, P., Słomczyński, W., Życzkowski K., *Dynamical Entropy for Systems with Stochastic Perturbation*, Phys. Rev. E, Vol.62, (2000), No. 2, pp. 2018-2029.

[12] Peterson, K. E., *Introductory Ergodic Theory*, Lecture Notes, Department of Mathematics, Unversity of North Carolina, (1971).

[13] Reza, F. N., *An Introduction to Information Theory*, McGraw-Hill, (1961).

[14] Rokhlin, V. A., *Lectures on the Entropy Theory of Measure-Preserving Transformations*, Khumsan State Institute of Mathematics of the Academy of Sciences UzSSR, (1965).

[15] Rudin, W., *Real and Complex Analysis*, McGraw-Hill, New York, (1974).

[16] Sinai, Ja. G., *On the Concept of Entropy of a Dynamical System*, Doklady Akademii Nauk SSSR, Vol. 124, pp. 768-771, (1959); English Summary, Math Rev. 21, 386 (1960).

[17] Short, K. M., *Direct Calculation of Metric Entropy from Time Series*, J. Comp. Phy., Vol. 104, (1993), No. 1 , pp. 162-172.

[18] Shanon, C. E. and Weaver,W., *Mathematical Theory of Comunication*, The University of Illinois Press, Urbana, (1964).

[19] Walters, P., *An Introduction to Ergodic Theory*, Springer-Verlag, New York, (1982).

Appendix: Maple Code for Computing Entropy

```
> #****************************************************#
> #                 Preferences
> #****************************************************#
> #****************Map Definition**********************#
> t:= x -> frac(2*x):
> x_max:=1:
> x_min:=0:
> randomval:=evalf(rand(1..10^10)/10^10):
> startx:=randomval(): #random point to start with#
> #****************************************************#
> #***************number of iterations*****************#
> npt:=10000:
>
> #********Partitions Preferences**********************#
> partition_size_list:=[2,8,64]:
> lengt_of_partition_size_list:=3:
>
> #***Sequencing Preferences***************************#
> # pointstarting to find the best fit line
> initial_rise_point:=1:
> # Steps that sequencing process is at most done
>
> maximum_sequence_length:=10;
>
> #***Precision Preferences*************************#
> #precision
> Digits:=10;
>
> #***Plot Preferences********************************#
> maximum_information:=15:
> style_array[0]:=point:
> style_array[1]:=diamond:
> style_array[2]:=box:
> style_array[3]:=cross:
> style_array[4]:=circle:
> #*********Constants*********************************#
>  plotcount:=0:
>  entropies_array_index:=1:
>  points_of_graph_array:=array(0..1000):
>  compress:=array(0..900000):
>  histogram:=array(0..10200,0..900000):
>  probability:=array(0..900000):
>  information_array:=array(0..20000):
>  entropies_array:=array(0..200):
> #*****************************************************#
> #           Partitioning and Iterating
> #*****************************************************#
```

```
>
> #*************Iterating***************************#
> for i from 0 to npt-1 do
>   x[i]:=startx:
>   startx:=evalf(t(startx)):
> od:
>
> for i from 0 to npt-1 do
>    x[i]:= x[i]-x_min:
> od:
>
> length_of_domain:=x_max-x_min:
>
> for i from 0 to npt-1 do
>    x[i]:= x[i]/length_of_domain:
> od:
>
> for partition_index  from 1 to
>  lengt_of_partition_size_list do
> k:=partition_size_list[partition_index]:
> unassign('information_array'):
>
> #*********Symbolic Representation of Data***********#
> for i from 0 to npt-1 do:
> bin[i]:=floor(x[i]*k);
> od:
> #***************************************************#
> for i from 0 to npt-1 do
>   compress[i]:= bin[i]:
> od:
> #*****************************************************#
> newcount:=k:
>
> #*****************************************************#
> #          Sequencing and Compactification
> #*****************************************************#
> for n from 1 to  maximum_sequence_length  do
>
> for i from 0 to (npt) do
>    probability[i]:= 0;
> od:
>
> for i from 0 to npt-n do
>   if type(histogram[bin[i+n-1],compress[i]],integer)
>    then
> histogram[bin[i+n-1],compress[i]]:=
>                  histogram[bin[i+n-1],compress[i]]+1:
>   else
>     histogram[bin[i+n-1],compress[i]]:=1:
>   fi:
> od:
> #*****************************************************#
> for i from 0 to npt-n do
>  compress[i]:=compress[i]+newcount*bin[i+n-1]:
> od:
>
```

```
> rowlength:=newcount:
> newcount:=0:
>
> for i from 0 to k-1 do
>  for j from 0 to rowlength-1 do
>    if  type(histogram[i,j],integer) then
>        histogram[i,j]:=newcount:
>        newcount:=newcount+1:
>    fi:
>  od:
> od:
>
> for i from 0 to npt-n do
>  column:= (compress[i] mod rowlength):
>  row:=   floor(compress[i]/rowlength):
> compress[i]:=histogram[row,column]:
> od:
>
> unassign('histogram'):
>
> #****************************************************#
> #          Probabilities and Information
> #****************************************************#
> for i from 0 to npt-n do
> if type(probability[compress[i]],integer) then
>  probability[compress[i]]:=probability[compress[i]]+1:
> else
>  probability[compress[i]]:=1:
```

```
> fi:
> od:
>
> for i from 0 to newcount-1 do
> probability[i]:=probability[i]/(npt-n+1):
> od:
> #**************************************************#
> information:=0:
> for i from 0 to newcount-1 do
> information:=
> information+probability[i]*ln(probability[i])/ln(2):
> od:
> information:=-information:
> #**************************************************#
> information_array[n]:=evalf(information):
> separation_level:=evalf(newcount/(npt-n+1)):
> print("Information I_n(alpha):", information_array[n]
> ,"  Sequence_length n:", n,
> "   Separation level:",separation_level ):
> if ( separation_level <0.2)  then fall_off_point:=n:
> fi:
> od:
> #**************************************************#
> points_of_graph_array[k]:=[[m,information_array[m]]
> $m=1..maximum_sequence_length];
> separationtable[k]:=fall_off_point;
> print("Fall of point happens at:",fall_off_point,
> "  npt=",npt, "Partition-size k=", k);
```

```
>
> evaluating_points_array[k] :=
> [[m,information_array[m]]$m=1..fall_off_point]:
> print("Array of points of the graph used to evaluate
>  the entropy:", evaluating_points_array[k]);
> unassign('entropy_slope'):unassign('t'):
> #****************************************************#
> #*************Least Squares Method****************#
> E := (entropy_slope,t)->
> sum( ((entropy_slope*evaluating_points_array[k][q][1]+t)-
> evaluating_points_array[k][q][2] )^2,q=
> initial_rise_point..fall_off_point):
> assign(solve({diff(E(entropy_slope,t), entropy_slope)=0,
> diff( E(entropy_slope,t), t)=0})):
> print("Entropy slope:",entropy_slope," Fall off point:",
> fall_off_point, "  Partition size:", k,
> "  Number of points used to evaluate the
> entropy slope:", fall_off_point-initial_rise_point+1);
> entropies_array[entropies_array_index]:=entropy_slope:
> entropies_array_index:=entropies_array_index+1:
> print("******************************");
> unassign('entropy_slope'):unassign('t'):
> unassign('evaluating_points_array'):
> od:
> #****************************************************#
> #         Plotting Information Graphs
> #****************************************************#
> for i from 1 to lengt_of_partition_size_list do:
```

```
> p[i]:=
> [plot([points_of_graph_array[partition_size_list[i]]]
> ,style=point, symbol=style_array[i mod 5],
> color=black,view=[0..maximum_sequence_length,
> 0..maximum_information],labels=["n", "I_n"],
> scaling=constrained,legend=
> [convert(partition_size_list[i],string)])]:
> od:plot_vector:=[[p[count][1]]
> $count=1..lengt_of_partition_size_list]:
> with(plots):display([plot_vector[count][1]]
> $count=1..lengt_of_partition_size_list);
> #**************************************************#
> #         Printing Entropy Computed
> #**************************************************#
> print("Entropy of the map is the maximum of all
> entropy-slopes=",max (entropies_array[h]
> $h=1..lengt_of_partition_size_list) );
```