On the algorithmic complexity of static structures

J. Ratsaby[†] and J. Chaskalovic[‡]

November 7, 2010

[†]Department of Electrical and Electronics Engineering, [‡]Department of Mathematics and Computer Science, Ariel University Center, Ariel 40700, ISRAEL and IJLRDA, University Pierre and Marie Curie - Paris VI, FRANCE

[†]ratsaby@ariel.ac.il, [‡]jch@ariel.ac.il

Abstract

From the theory of algorithmic information introduced by Chaitin [4] it is known that the more complex a system that is acting on a random binary input sequence the more it can deform the stochastic properties of the sequence. It is an open question as to whether such a relationship exists in real complex systems, for instance, systems that are governed by physical laws. The paper provides a first indication that this is true for a system comprised of a static structure described by hyperbolic partial differential equations and is subjected to an external random input force. The system deforms the randomness of an input force sequence in proportion to its algorithmic complexity. We demonstrate this by numerical analysis of a one-dimensional vibrating elastic solid (the system) on which we apply a maximally-random force sequence (input). The level of complexity of the system is controlled via external parameters. The output response is the field of displacements observed at several positions on the body. The algorithmic complexity and stochasticity of the resulting output displacement sequence is measured and compared against the complexity of the system. The results show that the higher the system complexity the more random-deficient the output sequence.

1 Introduction

Measuring the complexity of static structures has been traditionally a problem in the systems analysis literature. The world is full of dynamic complex systems that consist of simple components but which yet have a complex deterministic behavior. Examples of such systems are cellular automata, ecosystems, social systems, sensor networks (consisting of many simple processing units). A complex system is one whose static structure is intricate and requires a long description. For dynamic systems the notion of complexity is usually attributed to the high level of unpredictability of their behavior. While a complex system does not necessarily have to be stochastic it still takes a considerable amount of analysis and computations to determine its behavior (the connection between randomness and complexity will be discussed later). Many problems require the analysis of the complexity of processes and systems, for instance, in control and manufacturing engineering [3] one studies how complexity effects the tradeoff between hardware cost (which rises as the system becomes more complicated) versus improved system performance. The research field of complex systems studies different measures to quantify the behavioral complexity of a system. In computer science, the notion of computational complexity serves as a measure of how difficult it is to compute a solution for a given problem. Computations take time and complexity here means the time rate of growth to solve the problem. Yet another related kind of complexity measure (studied in theoretical computer science) is the so-called algorithmic (or Kolmogorov) complexity which measures how long a computer program (on some generic computational machine) needs to be in order that it produces a complete description of an object. Interestingly, the theory says that if we consider as an object a system that can process input information (available as a binary sequence of high entropy) and produces as output another sequence then the amount of randomness in the output sequence is inversely proportional to the algorithmic complexity of the system.

While this has been traditionally studied in the context of Turing machines and randomness, it is unknown whether there is such a connection between complexity and randomness for more general systems, for instance, those governed by physical laws. As an example of a real system, consider the object to be a bridge. The bridge has some finite descriptive complexity consisting of all the information contained in the engineering design documents. These documents can be put into a single computer file that can be represented by a finite binary string z. This binary sequence has an algorithmic complexity which is defined as the length of the shortest computer program that can generate the sequence. This is defined as the Kolmogorov complexity $K(z)$ of the string z (see [6]). Now consider a *random* input force sequence applied at one of the two ends of the bridge, for instance, suppose there is a person jumping up and down sporadically on the bridge at its entrance (position 0). Denote by x the binary sequence representing this up/down symbols over some fixed time-interval $[0, T]$. Intuitively, being that x is random makes its complexity $K(x)$ maximal and hence close to its actual length $\ell(x)$ since there is no redundancy in the patterns of x that can be used to compress it significantly below its length. Now consider an observer which measures the displacements on the bridge at its other end (position L). He records this over the time interval $[0, T]$ and compares it to a fixed threshold thereby producing a binary output sequence y consisting of up/down symbols that represent the displacement of the bridge at position L. This sequence has a finite algorithmic complexity $K(y)$.

In this paper we show that for such a physical system, the system complexity $K(z)$, the output complexity $K(y)$ and its level of randomness are all related and there exist statistically significant correlations between them.

Ratsaby [8] introduced a quantitative definition of the information content of a static structure (a solid) and explained its relationship to the stability and symmetry of the solid. His model is based on concepts of the theory of algorithmic information and randomness. He modeled a solid as a selection rule of a finite algorithmic complexity which acts on an incoming random sequence of particles in the surroundings. This selection mechanism is intrinsically connected to the solid's complex non-linear structure (partly a consequence of its internal atomic vibrations) and its intricate time-response to external stimulus. As postulated in [8], a simple solid is one whose information content is small. Its selection behavior is of low complexity since it can be described by a more concise time-response model (shorter computer program). The solid's stability over time is explained in [8] by using the stochastic property of the frequency stability of a random sequence. Accordingly, the physical stability of the system (solid) is intrinsically and inversely proportional to the ability of the solid to deform the input sequence and make it less random, i.e., more random-deficient.

The current paper (which is the full version of an earlier paper [9]) presents first evidence that suggests that the model of [8] holds. We choose to simulate a solid structure which consists of a one-dimensional vibrating solid-beam to which we apply a random input force sequence and observe the displacement of the beam at its other end for a finite interval of time. We determine empirically the relationship between the algorithmic complexity of the structure to the stochasticity of the output response.

The paper is organized as follows: in section 2 we give a brief introduction to the main concepts of the area of algorithmic complexity and randomness. In section 3 we state concisely the aim of the paper. In section 4 we develop the equations that describe the solid deformations and compute the numerical equations needed to produce the computer simulation of the solid's response to external forces. In section 5 we state the experimental setup, results and analysis. In section 6 we state the conclusions.

2 Background

In order to give the formal definition of Kolmogorov complexity we need to mention a few basic definitions from the theory of computability [12]. Let $\mathbb{N}$ denote the set of natural numbers. A function f of n variables defined on $\mathbb{N}^n$ is *total* if for all $x \in \mathbb{N}^n$, $f(x) \neq \infty$. A *partial* function f of n variables defined on the extended natural numbers $\overline{\mathbb{N}}^n = (\mathbb{N} \cup \{\infty\})^n$ has as an image the set $\overline{\mathbb{N}}$. Partial means that there exist some input $x \in \overline{\mathbb{N}}^n$ for which $f(x) = \infty$. Unless otherwise stated we assume that any input variable x is in $\overline{\mathbb{N}}^n$. A register machine (RM) [11] is a simple abstract computer which serves as model for mathematical calculation (register machines coincide with Turing machines and Church's λ-definable functions [5]). It is specified by a program $\mathcal{P} =< c_0, c_1, \ldots, c_{h-1} >$ consisting of a finite sequence of commands c_i. The commands operate on registers $R_1, R_2, \ldots$, each capable of storing an arbitrary natural number. There are three possible commands: the clear command, "$R_i \leftarrow 0$", the increment command "$R_i \leftarrow R_i + 1$" and the conditional jump command "Jump to k if $R_i = R_j$" which means go to position k in the program and perform command c_k. The following is an example of a program which copies the value in register R_i into R_j:

```
$c_0 : R_j \leftarrow 0$
$c_1 :$ Jump to 4 if $R_i = R_j$
$c_2 : R_j \leftarrow R_j + 1$
$c_3 :$ Jump to 1 if $R_1 = R_1$
$c_4 :$
```

Note that command c_3 simply is an unconditional jump to command c_1. An RM-program $\mathcal{P}$ computes a (partial) function $f(x_1, x_2, \ldots, x_n)$ by placing in register $R_1, \ldots, R_n$ the input values $x_1, \ldots, x_n$ and setting all other registers to 0. Then, execution starts with the first command c_0. If $\mathcal{P}$ halts then the final value of R_1 contains the value $f(x_1, x_2, \ldots, x_n)$. If $\mathcal{P}$ fails to halt then $f(x_1, \ldots, x_n) = \infty$. If f is a partial function on n variables we say that f is RM-computable if f is computed by some RM program. The renowned Church-Turing thesis ([10]) claims that every algorithmically computable function is RM-computable. Note that computability does not imply the existence of an efficient algorithm or program to compute the function, for instance, there may not always be a program that takes an amount of time polynomial in the number of inputs n to complete the computation.

The theory of recursive functions puts in a formal way the imprecise notion of effective computations, i.e., computations that are RM-computable. We briefly review a few needed basic definitions. An *initial* function is one of the following three functions: the all-zero function $Z(x) = 0$ for all x, the successor function $S(x) = x+1$, the projection function $I_{n,i}(x_1, \ldots, x_n) = x_i$. A function f is *partial recursive* if it can be obtained from initial functions g, h by finitely many applications of any of the following three operators: (1) *composition* ($f(x) = g(h(x))$), (2) *primitive recursion* ($f(x, 0) = g(x)$ or $f(x, y+1) = h(x, y, f(x, y))$ where we allow $n = 0$ so x could be missing as an argument), (3) *minimization* ($f(x) = \text{argmin}_y\{g(x, y) = 0\}$ where argmin is also denoted by μ operator and it returns the least number b such that $g(x, b) = 0$ or $f(x) = \infty$ if no such b exists). Note that the value of the minimization operator may not exist in which case a program for it will not halt; this means the function is partial recursive. It is known that all computable functions are partial recursive and every partial recursive function is computable. A partial recursive function whose value for any input x is well-defined (not infinity) is called *total recursive* function. Let us give a few examples. First, every one of the initial functions is a total recursive function. Let us denote the predecessor function by $pd(x) = x - 1$ if $x > 0$ else 0. We can define $f(x) = pd(x)$ by primitive recursion as follows: first choose as the initial functions $g(x) = Z(x)$, $h(x, z) = I_{2,1}(x, z) = x$. Then write $f(0) = g(x)$, $f(x + 1) = h(x, f(x))$. Hence the function pd is primitive recursive. As another example, denote by $(x - y)_+ = x - y$ if

$x \geq y$ else 0. We show that the function $f(x, y) = (x - y)_+$ is obtained using primitive recursion based on the predecessor function pd as follows: first, let $g(x) = x$, $h(x, y, z) = pd(z)$. Thus we may write $f(x, 0) = g(x)$ and $f(x, y+1) = (x - (y+1))_+ = (x - y - 1)_+ = h(x, y, (x - y)_+)$ and hence $(x - y)_+$ is primitive recursive. It is known that every primitive recursive function (i.e., one which is obtained from initial functions using only the first two operators) is total recursive and is computable. But the converse is not always true, i.e, *not* every computable function is primitive recursive since some computable functions are not total. In fact, not every total computable function (i.e., one whose program halts on every input) is primitive recursive. Let us show an example of a computable function which is not total. First let us define the function $square(x)$ which returns the square of x and is primitive recursive since it can be defined in terms of the primitive recursive function $mul(x, y)$ which returns the product of x and y. Now consider the function $squareroot(x)$ which computes the square root of a number x and is defined as follows (we use pseudo-code):

```
$squareroot(x)${
initialize $t \leftarrow 0$
while ($square(t) \neq x$) do {
$t \leftarrow t + 1$
}
return $t$
}
```

Clearly, this program may not halt on all inputs x. In fact it only halts whenever the input x is a perfect square. Hence the function $squareroot(x)$ is not total, but rather partial recursive.

Based on the unique-factorization theorem for natural numbers (see for instance, [12]) which states that every number can be represented as a product of prime numbers raised to some unique sequence of natural number exponents, it is possible to represent any RM program $\mathcal{P}$ by a unique number z. For any natural number z, denote by $\{z\}$ the program $\mathcal{P}$ whose unique number is z or the empty program if no such program exists. Denote by $\{z\}_n(x)$ the function of n variables $x = (x_1, \ldots, x_n)$ defined by a program whose unique number is z. We can now introduce the notion of a universal function. A function $\Phi_n(z, x) = \{z\}_n(x)$ is called universal since it codes (indexes) every computable function of n variables $x = (x_1, x_2, \ldots, x_n)$ as z varies. The universal function Φ_n is partial recursive (and hence computable) for $n = 1, 2, \ldots$. A program computing the function Φ_n is called an interpreter. For instance, the function Φ_1 is universal for the set of all computable functions of one variable. Thus if we define $\phi_i(x) = \Phi_1(i, x)$ then $\phi_0, \phi_1, \ldots$ is an enumeration of *all* partial computable functions of one variable. It is known that there is no computable universal function for the set of all *total* computable functions of one variable.

We now proceed to introduce the formal definition of Kolmogorov complexity. Kolmogorov [6] proposed to measure the conditional complexity of a finite object x given a finite object y by the length of the shortest program $\mathcal{P}$

which reconstructs x given y as input. In a formal context, an object is represented by a unique finite binary sequence which is in the set of all finite binary sequence $\mathcal{B}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$ where ϵ denotes the empty string. There is a correspondence between finite binary sequences and natural numbers. It is a one-to-one mapping from $\mathcal{B}^*$ onto the natural numbers $\mathbb{N}$ and is defined by: $(\epsilon, 0), (0, 1), (1, 2), (00, 3), (01, 4), (10, 5), (11, 6), \ldots$ where the first element of a pair is the binary sequence and the second element is the corresponding natural number. Formally, a binary sequence $a = (a_n \ldots a_1 a_0)$ is represented by the natural number $x = 2^{n+1} - 1 + \sum_{i=0}^{n} a_i 2^i$. We consider both elements of a pair as the same object. With this mapping in place it is clear that computable functions of one variable over $\mathbb{N}$ correspond to computable functions over $\mathcal{B}^*$.

Let y be a finite binary string with a corresponding natural number y'. Consider a program $\{y'\}$ that simulates an RM on a universal machine Φ, i.e., $\Phi_1(y', p') = \{y'\}_1(p')$ for any $p' \in \overline{\mathbb{N}}$. Let x be any binary string and let p be a program with a unique natural number p' such that $\Phi_1(y', p') = x$. We henceforth use the simpler notation and denote by $\Phi(y, p) \equiv \Phi_1(y', p')$ where y and p are the finite binary strings that correspond to the natural numbers y' and p'. We can now state a formal definition of the Kolmogorov complexity of an object x given object y:

$$K_\Phi(x|y) = \min\{\ell(p) : \Phi(y, p) = x\} \tag{1}$$

where $\ell(\mathcal{P})$ is the length of the program p (note that there may be many programs that produce x given y and we are only interested in the minimal one). In other words Φ is a universal partial recursive function acting as an interpreter (or description method) that can emulate *any* RM. The complexity of x is taken with respect to this universal machine which when provided with an input binary string y (that constitutes an index to a specific RM which is represented by the unique natural number y') and an input program p to run on this RM it then outputs a binary sequence corresponding to the object x. Thus the Kolmogorov complexity of x given y as defined in (1) is the length of the shortest program that generates x based on Φ given y as input. Note that when y equals the empty binary sequence, this gives the unconditional Kolmogorov complexity $K(x)$. The so-called Invariance Theorem states that there is a Φ such that the complexity $K_\Phi(x|y)$ is no more than a constant away from the complexity based on *any* other universal partial recursive function Ψ, i.e., $K_\Phi(x|y) \leq K_\Psi(x|y) + const_\Psi$ where the constant is independent of x and y. This implies that for any pair of universal functions Ψ and Ψ' the following holds: $|K_\Psi(x|y) - K_{\Psi'}(x|y)| \leq const_{\Psi,\Psi'}$. Thus the notion of Kolmogorov complexity is absolute in the sense that the choice of universal machine only effects the complexity of x up to a constant independent of x. In order to appreciate this consider two programming languages LISP and FORTRAN each with its corresponding interpreter which when given a string y it produces a specific RM program (partial recursive function) from a given input program p written in LISP or FORTRAN, respectively, that outputs

x. Both language interpreters will output x and the invariance theorem says that the shortest LISP program and the shortest FORTRAN program that output x are approximately equal in length to within an additive constant that does not depend on x.

We now proceed to discuss how complexity relates to randomness. Let us extend the notation $\mathcal{B}^*$ introduced above for the set of all finite binary sequences and denote by $\mathcal{B}_n$ the set of all finite binary sequences of length n. An admissible *selection rule* R [13] is a partial recursive function on $\mathcal{B}^*$ that picks certain bits from a binary sequence x. Let $R(x)$ denote the selected subsequence. By $K(R|n)$ we mean the length of the shortest program computing the subsequence $R(x)$ given n. Kolmogorov introduced a notion of randomness deficiency $\delta(x|n)$ of a finite sequence $x \in \mathcal{B}_n$ where $\delta(x|n) = n - K(x|n)$ and $K(x|n)$ is the Kolmogorov complexity of x not accounting for its length n, i.e., it is a measure of complexity of the information that codes only the specific pattern of 0s and 1s in x without the bits that encode the length of x (which is $\log n$ bits). Randomness deficiency measures the opposite of chaoticity of a sequence. The more regular the sequence the less complex (chaotic) and the higher its deficiency. An infinitely long binary sequence is regarded random if it satisfies the principle of stability of the frequency of 1s for any of its subsequences that are obtained by an admissible selection rule [7**?**].

In [6] it was shown that the stochasticity of a finite binary sequence x may be precisely expressed by the deviation of the frequency of ones from some $0 < p < 1$, for any subsequence of x selected by an admissible selection rule R of finite complexity $K(R|n)$. The chaoticity of x is the opposite of its randomness deficiency, i.e., it is large if its Kolmogorov complexity is close to its length n. The works of [1, 2, 6, 13] relate this chaoticity to stochasticity. In [1, 2] it is shown that chaoticity implies stochasticity. This can be seen from the following relationship (with$p = \frac{1}{2}$):

$$\left|\nu(R(x)) - \frac{1}{2}\right| \leq c\sqrt{\frac{\delta(x|n) + K(R|n) + 2\log K(R|n)}{\ell(R(x))}} \tag{2}$$

where for a binary sequence s, we denote by $\nu(s) = \frac{\#(s)}{\ell(s)}$ the frequency of 1s in s where $\#(s)$ denotes the number of 1s in s, and $\ell(R(x))$ is the length of the subsequence selected by R, $c > 0$ is some absolute constant. From this we see that as the chaoticity of x grows (randomness deficiency decreases) the stochasticity of the selected subsequence grows (bias from $\frac{1}{2}$ decreases). The information content of the selection rule, namely $K(R|n)$, has a direct effect on this relationship: the lower $K(R|n)$ the stronger the stability (smaller deviation of the frequency of 1s from $\frac{1}{2}$).

3 Aim of the paper

In this paper we provide first evidence that the basic notion of randomness and its relationship to complexity (as discussed in the previous section) underlie the behavior of physical systems. This supports the ideas introduced in [8]. We focus on a system composed of a vibrating elastic solid (described by the classical equations of solid mechanics) and its interaction with a random input force. We show that as a result of this interaction, the deformation of the solid over time can be described as an output sequence whose stochastic and algorithmic properties follow those of an output subsequence selected by a selection rule of a finite complexity. Based on a large sample of computer-generated simulations of such solids we provide statistically significant results that show that the complexity of the system inversely affects the complexity of the solid deformations (observed output) and its stochasticity agrees with the theory (2). The next section describes the solid's mechanical equations.

4 The solid's equations

The solid consists of an elastic homogeneous and one-dimensional beam of length L. Let us denote by x the position on the beam so that $0 \leq x \leq L$ and by $\overrightarrow{x}$ the unit vector on the x-axis. Denote by $\overrightarrow{f} = f(x,t)\overrightarrow{x}$ a force applied at time t on position x in the direction of $\overrightarrow{x}$. We define by$\overrightarrow{u} = u(x,t)\overrightarrow{x}$ the displacement at time t on x. The classical equation which describes the field of displacements u at a specific position and time when a force f is applied is as follows:

$$\left(\frac{\partial^2 u}{\partial t^2} - \frac{E}{\rho}\frac{\partial^2 u}{\partial x^2}\right)(x,t) \quad = \quad f(x,t), (0 < x < L, t > 0), \qquad (3)$$

where E is Young's modulus (the ratio of stress to corresponding strain when the beam behaves elastically), and ρ is the mass density. We impose the following boundary conditions:

$$u(0,t) = u(L,t) = 0,\ \forall t > 0, \tag{4}$$

i.e., the beam is fixed at its two ends so the only displacements is due to internal elasticity stresses of the material. Let $u_0(x), u_1(x)$ be two given functions that satisfy $u_0(0) = u_0(L) = 0$. As initial conditions we set the following,

$$u(x,0) = u_0(x), 0 < x < L, \tag{5}$$

$$\frac{\partial u}{\partial t}(x,0) = u_1(x), 0 < x < L. \tag{6}$$

Equations (3-6) represent the model that describes the deformations of the elastic solid. In order to simulate the response of the solid to external forces we use the following numerical approximation. This is performed by introducing a regular mesh of the $[0, L]$ interval with a constant step $\triangle x$ such that $N + 2$ equally spaced points are distributed on $[0, L]$. Specifically, we have the following mesh: $x_0 = 0$, $x_i = x_{i-1} + \triangle x$, $1 \leq i \leq N + 1$, $x_{N+1} = L$ and $\triangle x = \frac{L}{N+1}$.

Similarly, if time t belongs to the interval $[0, T]$, we introduce $M + 1$ discrete time points $t_0 = 0$, $t_n = n\triangle t$, $1 \leq n \leq M$, where $\triangle t = \frac{T}{M}$. Let us introduce the approximation sequence $\widetilde{u}(j,n)$, $1 \leq j \leq N$ and $1 \leq n \leq M$ such that $\widetilde{u}(j,n) \approx u(x_j,t_n)$, where u is solution of (3-6).

Let us also denote by $\widetilde{f}(j,n) \equiv f(x_j,t_n)$. We consider the following finite differences scheme to get an approximation of (3-6):

$$\begin{aligned}
\widetilde{u}(j,n+1) &= 2\tilde{u}(j,n) - \widetilde{u}(j,n-1) + (\triangle t)^2 \widetilde{f}(j,n) \\
&\quad + \left(\frac{\triangle t}{\triangle x}\right)^2 \frac{E}{\rho} \left[\widetilde{u}(j+1,n) - 2\widetilde{u}(j,n) + \widetilde{u}(j-1,n)\right], \quad (7) \\
\widetilde{u}(0,n) &= \widetilde{u}(N+1,n) = 0, \quad (8) \\
\widetilde{u}(j,0) &= u_0(x_j), \quad (9) \\
\widetilde{u}(j,1) &= u_0(x_j) + (\triangle t) u_1(x_j) \quad (10)
\end{aligned}$$

provided that the following CFL stability condition on the solid's parameters is satisfied,

$$\sqrt{\frac{E}{\rho}} \frac{\triangle t}{\triangle x} \leq 1.$$

In the next section we describe the experimental setup and results produced by (7-10).

5 Experimental results

We performed a series of experiments which consisted of several hundreds simulation trials of the response of a vibrating solid (henceforth called a *system*) to an input force sequence. We used the numerical equations of section 4 as the solid's model. As a choice of parameters we took $L = 20$, $T = 70$, $E = 0.7$, $\rho = 0.4$, $N = 30$, $M = 200$.

A system consists of a solid whose length is divided into 31 positions, $0, 1, \ldots, 30$. A force sequence $\widetilde{f}(15, n)$ is applied at position 15 while for all remaining positions the applied force is of zero magnitude. The non-zero force sequence $\widetilde{f}(15, n)$ makes the solid vibrate *a priori* hence we call the system a *vibrating solid.* This force sequence consists of a series of ternary values $-1, 0, +1$ scaled by a constant of 30. The length of the sequence is 200 and the symbols are obtained sequentially by a repeated series of random draws using the random variable $\mathcal{F}$ with the following probability distribution: let $0 < p \leq 1$, then $\mathcal{F}$ takes the value 0 with probability $1-p$, the value $+1$ with probability $\frac{p}{2}$, and -1 with probability $\frac{p}{2}$. The complexity of the sequence is controlled by the choice of p. We used a different p for different trials by randomly picking its value and using it as the parameter value p of the distribution of the random variable $\mathcal{F}$.

To the system we apply an input force sequence $\widetilde{I}(1, n)$ at position 1 consisting of 200 randomly drawn binary values $+1$ and -1 each with probability $1/2$ and scaled by a constant 10. Note that this input force is applied to a vibrating solid (as mentioned above). As the output of the system, we observe the displacement sequences at five positions $\widetilde{u}(N-5, n)$, $\ldots$, $\widetilde{u}(N-1, n)$, $1 \leq n \leq 200$ and convert their values a from real to ternary $V(a)$ using the following rule: given $a \in \mathbb{R}$ then $V(a) = +1$, 0 and -1 if $a > \tau$, $|a| \leq \tau$

and $a < -\tau$, respectively, with $\tau = 0.1$. We then append these five ternary sequences together to form a single ternary output sequence of length 1000 (henceforth this is called the *output sequence*). We also consider the subsequence of this output sequence which consists only of the values $+1$, -1, i.e., without the zeros (we call this the *output subsequence*).

As an estimate of the complexity $K(x)$ of a sequence x we use a standard compression algorithm (Gzip, which is a variation of the algorithm of [14]) to compress x. The length of the resulting compressed version of x is used as an approximation of $K(x)$. Henceforth, when we say system complexity we mean the length of the compressed version of the sequence consisting of all applied forces appended sequentially into one string with $31 \cdot 200 = 6200$ ternary symbols (in our experiments, all but the $\widetilde{f}(15, n)$ force are just all-zeros hence in this 6200-long string approximately only 200 bits contain information). The output complexity is the length of the compressed version of the ternary output sequence.

Let M denote the ratio of the compressed length divided by the uncompressed length of the system and let O denote this ratio for the output sequence. A large M (or O) means that the compressed length is larger hence the complexity of the system (or output sequence) is larger. We sometime simply refer to M and O as the system and output complexity, respectively. Figure 1 displays two sets of trials.

In each trial of set (a) a random input force sequence was applied at position 1 (as described above). In each trial of set (b) *no* input sequence

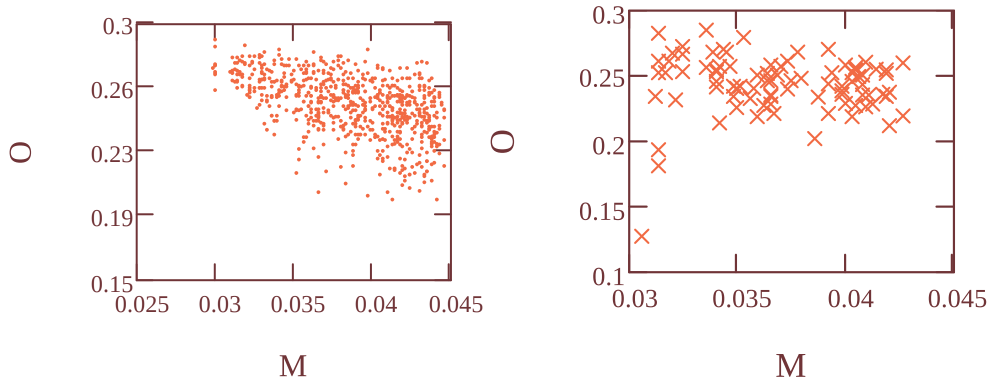

Figure 1: Output's complexity O versus the complexity M, (a) with random force input, (b) with no input

was applied. As is seen, the resulting behavior is clearly different in each of the two sets of trials. With an input present, as the complexity M increases there appears to be a decreasing trend in the value of O and an increase in the spread, i.e., the range of possible values of O. With no input, both O and its spread of values are basically constant with respect to M.

In Figure 2 we plot the frequency of 1s in the output subsequence (this is the number of 1s divided by the number of non-zero symbols in the output sequence).

As can be seen, with an increase in the complexity there appears to be an increase in the spread of possible frequency values. Before we further discuss these results we proceed to perform the statistical tests.

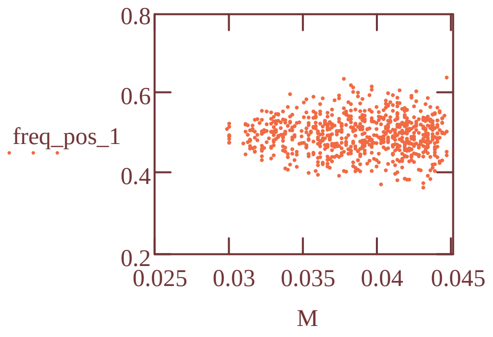

Figure 2: Output frequeny of 1s versus M

5.1 Analysis

In order to test the significance of these results we estimate the output complexity O as a function of the complexity M. Denote by X and Y the random variables corresponding to M and O, respectively. Let the underlying conditional probability distribution function be $P(Y|X)$ with marginals $P(X)$, $P(Y)$. As a sample we use the set of trials of Figure 1(a), denoted by $S = \{(x_i, y_i)\}_{i=1}^N$ with cardinality $N = 723$, and do linear regression in order to estimate Y with dependence on X. Figure 3 shows the resulting estimate,

$$\hat{Y}(X) = 0.341 - 2.284\,X, \tag{11}$$

surrounded by the 95% confidence limits for the regression line, i.e., the actual regression line of the population falls within the limits defined by the two curved dashed lines.

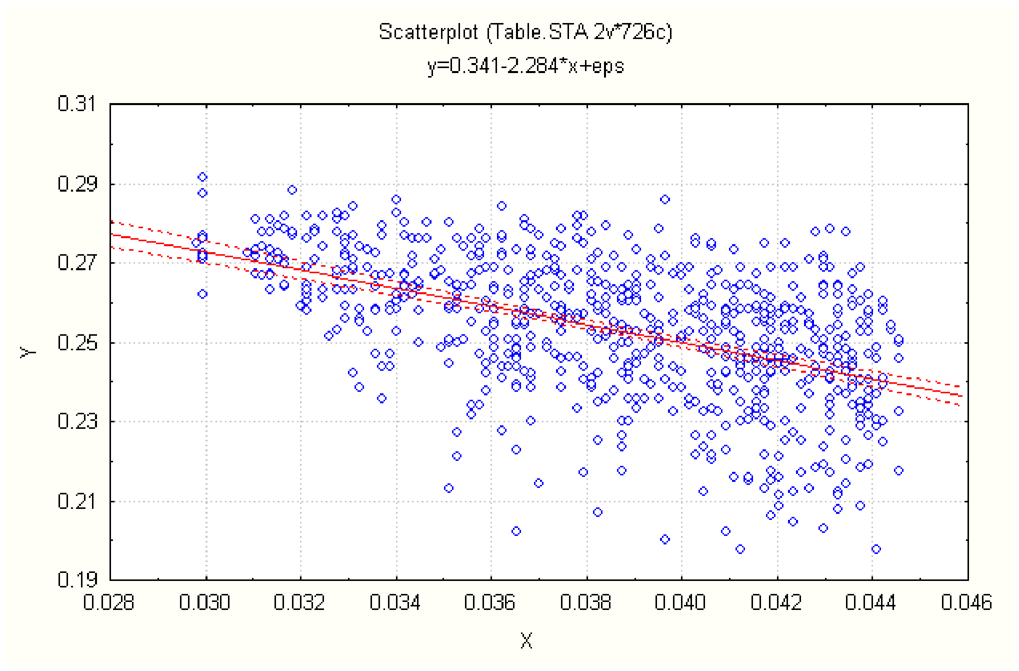

Figure 3: Estimate for output complexity Y as a function of complexity X

The following summarizes the accuracy of this linear regression estimate: $R^2 = .246187335$ is the coefficient of determination which measures the reduction in total variation of Y due to X and is defined as $R^2 = 1-(SS_R/SS)$ with $SS_R = \sum_i \left(y_i - \hat{Y}(x_i)\right)^2$ being the sum of squares of the residuals, $SS = \sum_i (y_i - \bar{y})^2$ the total variation and $\bar{y} = \frac{1}{N}\sum_i y_i$. The square root R is the coefficient of correlation between the independent variable X and dependent variable Y. The standard error $SE = .015128505$ where $SE = \sqrt{\frac{1}{N}SS_R}$. Dividing SS_R and SS by their degrees of freedom and taking their ratio $F = SS/SS_R$ as an overall F test gives $F(1, 724) = 236.4508$ which amounts to a p-value less than 0.000000. Thus with very high confidence the residual variance differs from the total variation hence the linear estimate $\hat{Y}(X)$

explains well the variation of Y. The Durbin-Watson d value is 1.994 which implies that the assumptions on the residuals being uncorrelated and normally distributed are met.

Next, from Figure 1(a) it is evident that as the complexity X increases the spread of the output complexity Y increases. To quantify this assertion let us represent this spread by the random variable

$$Z(X,Y) = Y - \min_{y:P(y|X)>0} y. \tag{12}$$

As we have done above for Y we now estimate Z with dependence on X (the model is shown only the value of X and asked to predict Z). We form the following sample (based on S),

$$\zeta = \{(x_i, z_i)\}_{i=1}^{N}, \; z_i = y_i - \min_{x_j \in NN(x_i,k)} y_j \tag{13}$$

where $NN(x,k)$ denotes the set of k nearest sample point x_j to x satisfying $x_j \leq x$. Figure 4 shows the resulting estimate equation (based on $k = 7$),

$$\hat{Z}(X) = -0.028 + 1.35\,X$$

for the regression line. This verifies the increase in the value of Z (i.e., in the spread of the output complexity Y) as the complexity X increases. The following summarizes the accuracy of this regression estimate: $R^2 = .098$, the F-ratio is $F(1,724) = 79.548$ with a p-level smaller than .000000. The standard error of the estimate is $SE = .015413$ with a Durbin-Watson $d = 1.938$. Thus the estimator $\hat{Z}(X)$ accurately captures the variability of Z, i.e., the spread of output complexity Y.

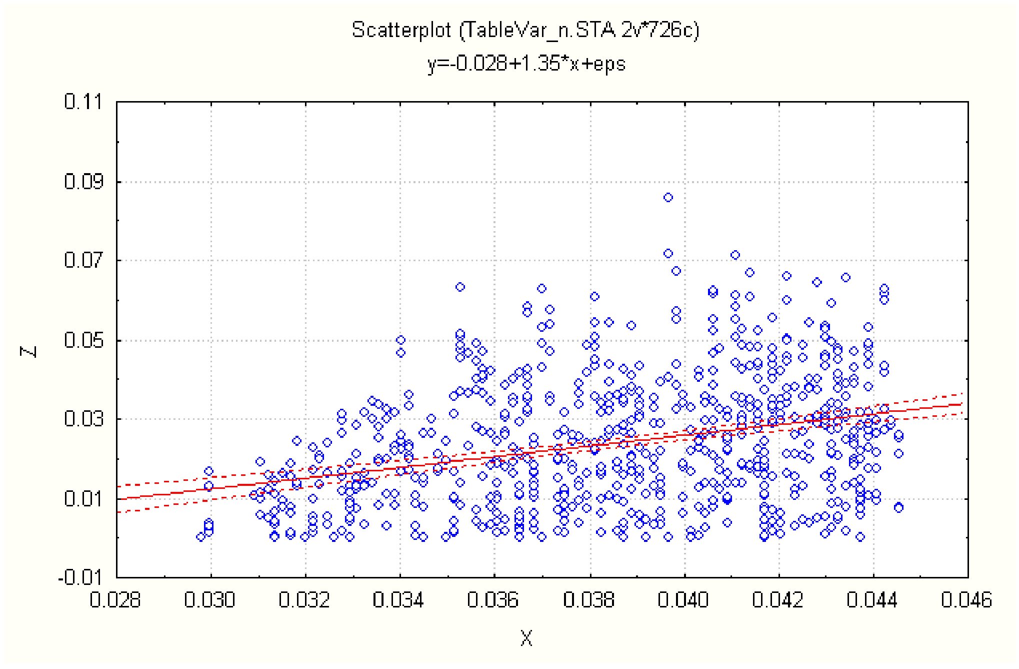

Figure 4: Estimate of the spread in output complexity Z as a function of the system's complexity X

As mentioned above, Figure 1(b) shows that when no input is present the behavior of the output complexity is almost unaffected by the system's complexity. To test this, we take the set of trials used in Figure 1(b) and study the correlation between the output complexity Y and the system complexity X. As shown in Figure 5 there is hardly any correlation between them and the slope of the regression is almost zero.

We already commented on the increasing spread of possible frequency

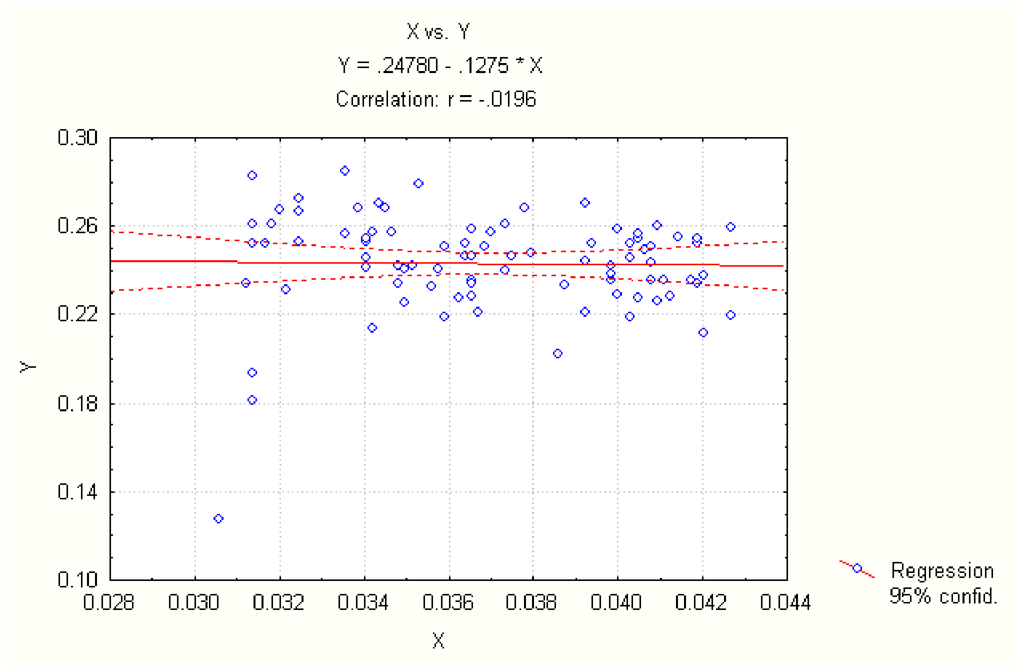

Figure 5: The no-input scatter plot of output complexity Y versus system's complexity X

values of the output subsequence (Figure 2) as the system's complexity increases (with a random input sequence being applied). Denote by Y the probability of having a +1 appear in the output subsequence and let X be the system's complexity. Let us define the following random variable

$$W(X) = \max_{y:P(y|X)>0} y - \min_{y:P(y|X)>0} y$$

to represent the spread in the possible values of the probability of +1. We form the following sample (based on S),

$$\zeta' = \{(x_i, w_i)\}_{i=1}^{N}, \; w_i = \max_{x_j \in NN(x_i,k)} y_j - \min_{x_j \in NN(x_i,k)} y_j \tag{14}$$

with $k = 7$. Figure 6 shows (on the top scatter plot with red x symbols) the frequency of 1s in the output subsequence versus the system complexity X. The bottom plot (with blue $\triangle$) shows the sample ζ' with the w component on the same vertical axis.

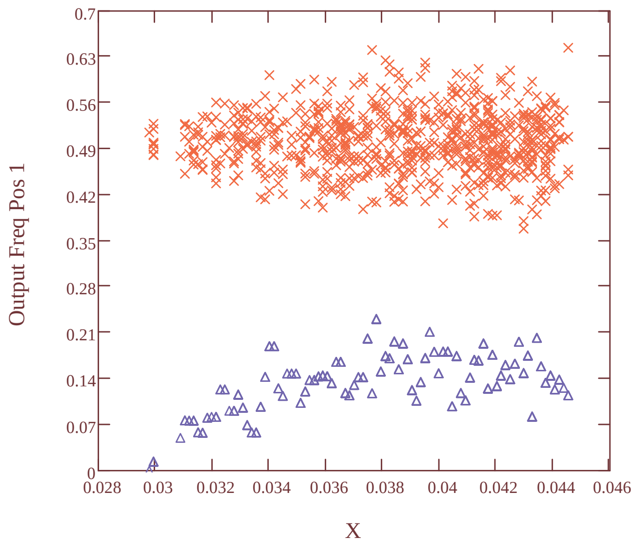

Figure 6: Scatter plot of frequency of 1s in the output subsequence (top cluster of red xs). The corresponding sample ζ' used to estimate the spread W as a function of the system's complexity X (bottom plot of blue $\triangle$)

We estimate W based on ζ' first transforming the w_i values to w_i^2 and then doing linear regression to estimate W^2. Figure 7 shows the resulting estimate equation $-0.023 + 1.083\,X$ for W^2. It follows that the estimate of

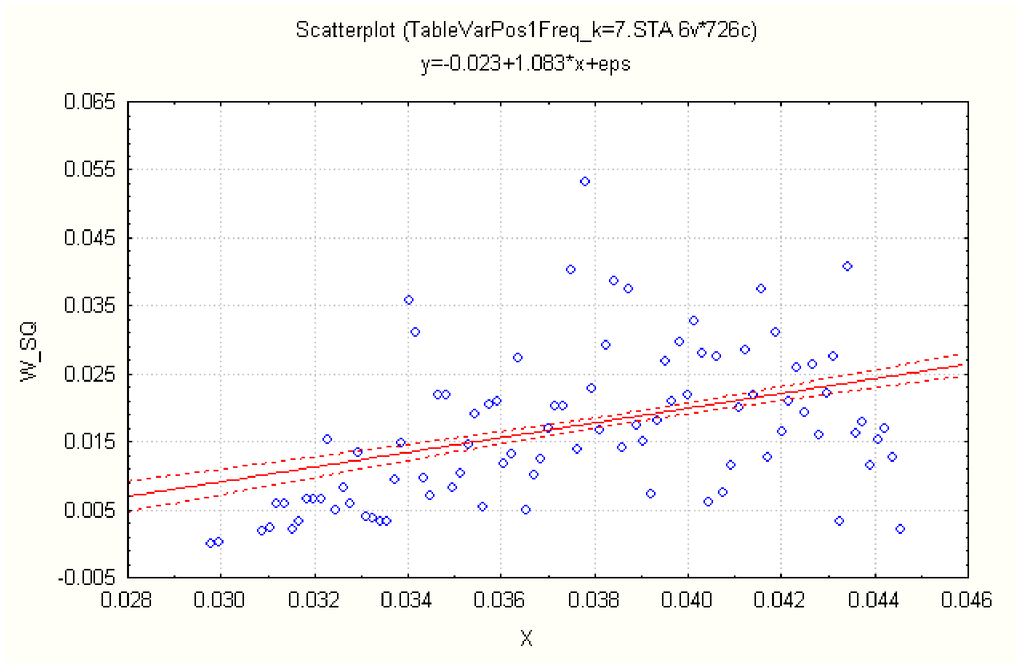

Figure 7: Estimate of the square of the spread W^2 of the output frequency of 1s as a function of the system's complexity X

W is

$$\hat{W}(X) = \sqrt{1.083\,X - 0.023}. \tag{15}$$

This verifies the increasing trend in the spread of values of the frequency of 1s as the system's complexity X increases. The following summarizes the accuracy of this linear regression estimate: $R^2 = .135$, the F-ratio is $F(1,724) = 113.06$ with a p-level smaller than .00000. The standard error of the estimate is $SE = .01037$ and the Durbin-Watson $d = 1.71$.

5.2 Some more details on the simulations

Several additional graphs showing additional details of the above experiments are shown below. Figure 8 shows the observed system description rate M (scatter plot in blue) and the entropy (the minimal expected number of bits per character) used for the system description (red solid curve). They are plotted versus the probability parameter p (in the range $0 < p \leq 1$) used to generate the random force sequence at position 15 of the solid. It follows from the procedure (described above) of generating the system's vibrating force that the entropy of the random variable $\mathcal{F}$ is $H(p) = -(1-p)\log(1-p) - p\log\frac{p}{2}$. As seen from Figure 8, in order to get a higher system complexity one needs to draw a force sequence with a parameter p closer to $1/2$. There is some additional textual information (145 bytes) appended into each of the files that contain the 6200-long ternary string that describes the system. Since M is the ratio of the compressed to uncompressed versions (the actual uncompressed length as reported by the operating system is 6377 bytes) then to get the rate for the number of bits per character used to describe the system (considering just the 145-byte textual information and the 200-byte force sequence at position 15) we multiply M by $6377 \cdot 8$ and divide by 345.

The next series of figures show examples of the actual solid's response (displacement u is shown on the z-axis) over time (y-axis) along the positions of the solid (x-axis). In all, the input sequence is maximally random with probability $1/2$ for $+1$, -1. The magnitude of the input force sequence is 10 and the magnitude of the system's vibrating force sequence is 30. The

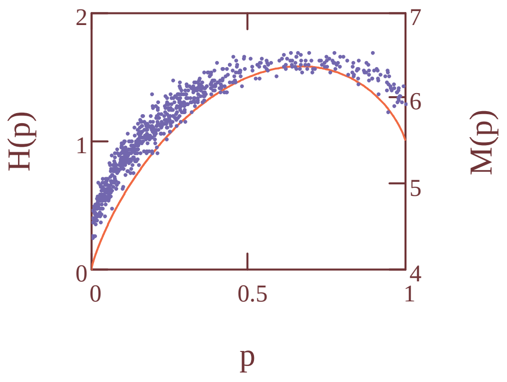

Figure 8: Entropy H (red) v.s. observed system complexity M (blue) as a function of parameter p which represents the probability of having a non-zero (i.e. ± 1) symbol in the system's vibrating force sequence (applied at position 15)

two force sequences are superimposed on the same 3D-plot that displays the displacement response. The output sequence is taken as the concatenation of the string obtained from the displacement at the last five positions (appearing on the plot to be closest to the reader). Figure 9 shows the response to a system of high complexity. Figure 10 shows a trial without a system force. This represents a low-complexity system. Figure 11 is the response of a system of a mid-level complexity. Figures 12 and 13 show the response when no input force is applied.

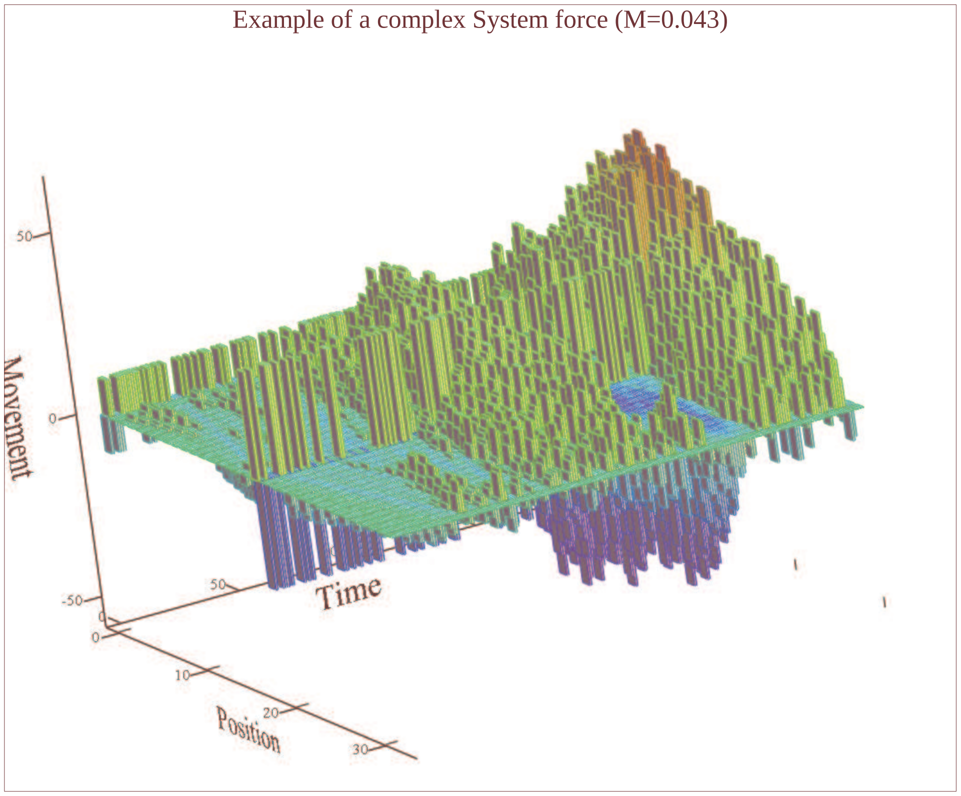

Figure 9: Response to a random input force sequence (applied at position 0). System's vibrating force (at position 15) is of high complexity. Output is seen to be of low complexity.

6 Conclusions

Based on these results, it is clear that when a random input sequence is applied to the vibrating solid (system) the observed output sequence is *not* simply a result of the random vibrating force sequence which is part of the system (applied at position 15) but is a direct consequence of the interaction of the system with an external random input force–when no input is present

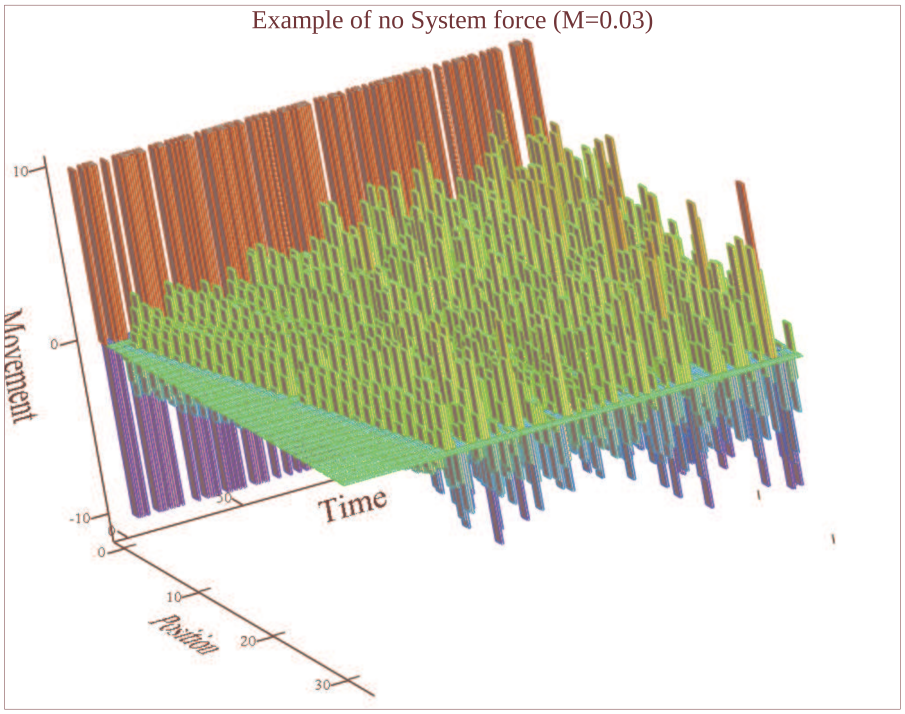

Figure 10: Response to a random input force sequence (applied at position 0). System has *no* vibrating force applied hence is of minimal complexity. The output is seen to be of high complexity.

no significant correlation exists between the system and output complexities. The strong negative correlation between these two complexities (11) suggests that the system deforms the input randomness and produces a less complex output sequence. This agrees with the model introduced in [8] which says that a solid effectively acts as a selection rule picking bits from the input sequence to produce a less random output. This is evident in the significant decrease in the output complexity (Figure 3) and increase in its spread of values (Figure 4) indicating that the possibility of producing a less-complex

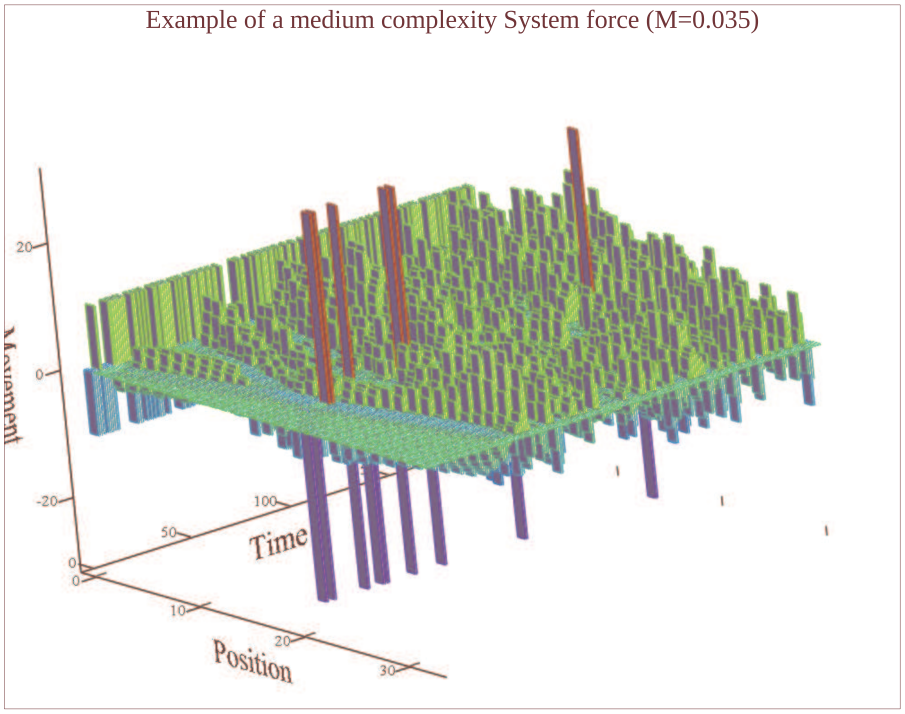

Figure 11: Response to a random input force sequence (applied at position 0). System's vibrating force (at position 15) has a mid-level complexity. Output is seen to be of lower complexity than the example in Figure 10.

output sequence increases as the system's complexity rises.

The selected subsequence consists of ± 1 with zeroes deleted. Being less chaotic, its stochastic level decreases. This is evident in the increase in the square of the spread of values of the frequency of 1's (Figure 7). The higher the system complexity, the higher the spread, i.e., the larger the bias from $1/2$, the more the chance that the output sequence be less chaotic and random. If we divide the compressed length of the system by the length of the output

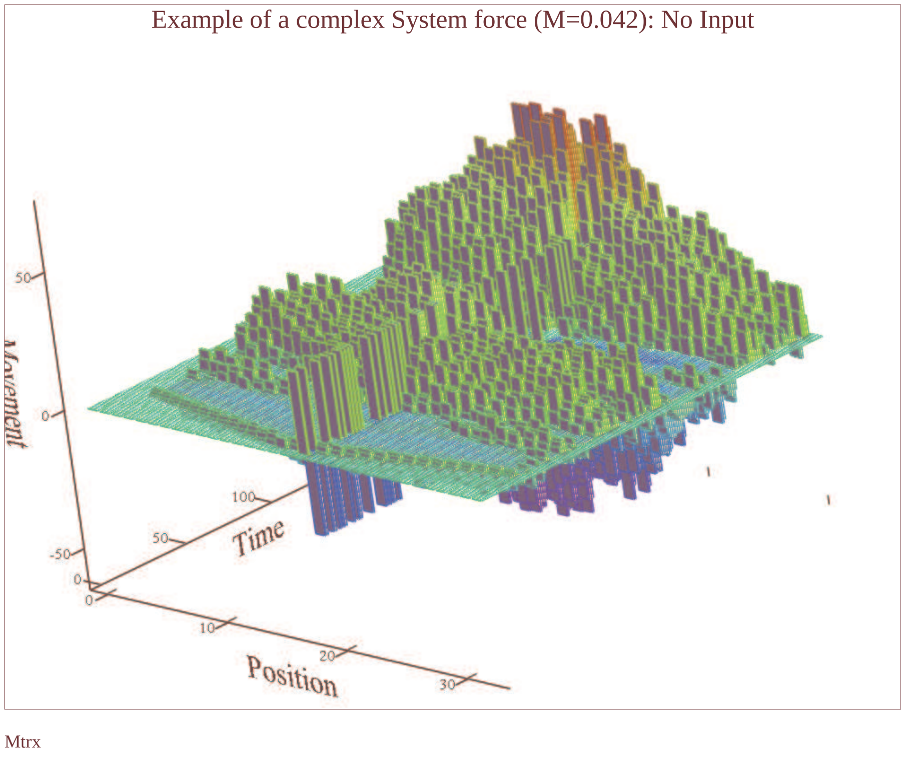

Figure 12: The no-input response to a system's force (at position 15) of high complexity.

(binary) subsequence and denote it by X' then re-estimate W^2 based on X' we obtain the following estimate for W,

$$\hat{W}(X') = \sqrt{-0.03 + 0.173X'}. \tag{16}$$

The $R^2 = .0924$, $SE = .0106$, $F(1,724) = 73.725$ and the p-level less than 0.00000. The Durbin-Watson $d = 1.66$. This estimate of the spread agrees with the rate predicted by the theory (2). To see this, let x be

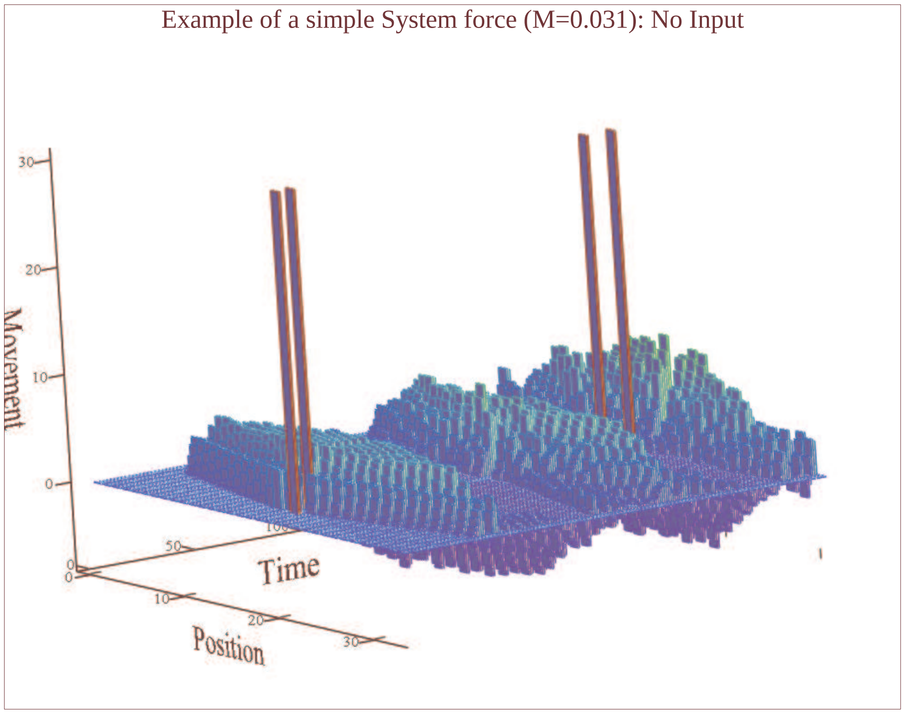

Figure 13: The no-input response to a system's force (at position 15) of low complexity.

the input sequence, let the system be the selection rule R with a system complexity $K(R|n)$, let $R(x)$ be the output subsequence (consisting only of binary values ± 1), let $\nu(R(x))$ be the frequency of 1s in this subsequence and take the deficiency of randomness $\delta(x|n)$ of the input sequence to be zero (since the input sequence is maximally random). Then the theoretical rate of the maximal possible deviation (spread) between $\nu(R(x))$ and $1/2$ is $O(\sqrt{K(R|n)/\ell(R(x))})$. This is the same rate in which the estimate of spread W grows with respect to the X' in (16).

To summarize, the results above imply that a system based on classical equations of mechanics that consists of a vibrating solid subjected to external random input-force acts like an algorithmic selection rule of a finite complexity. It produces an output sequence whose stochastic and chaotic properties are effected by the system's complexity as predicted by the theory of algorithmic randomness.

References

[1] A. E. Asarin. Some properties of Kolmogorov δ random finite sequences. *SIAM Theory of Probability and its Applications*, 32:507–508, 1987.

[2] A. E. Asarin. On some properties of finite objects random in an algorithmic sense. *Soviet Mathematics Doklady*, 36(1):109–112, 1988.

[3] Deshmukh A.V., Talavage J., and Barash M. Complexity in manufacturing systems, part 1: Analysis of static complexity. *IIE Transactions*, 30(7):645–655, 1998.

[4] G. J. Chaitin. Algorithmic information theory. *IBM journal of research and development*, 21:350–359, 1977.

[5] A. Church. A set of postulates for the foundation of logic. *Annals of Mathematics*, 33:346–366, 1932.

[6] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1:1–17, 1965.

[7] A. N. Kolmogorov. On tables of random numbers. *Theoretical Computer Science*, 207(2):387–395, 1998.

[8] J. Ratsaby. An algorithmic complexity interpretation of Lin's third law of information theory. *Entropy*, 10(1):6–14, 2008.

[9] J. Ratsaby and I. Chaskalovic. Random patterns and complexity in static structures. In *D.A. Karras et. al. (Eds.),Proc. Int'l Conf. on Artificial Intelligence and Pattern Recognition (AIPR'09)*, pages 255–261. ISRST, 2009.

[10] Kleene S.C. *Introduction to Metamathematics.* North-Holland, Amsterdam, 1952.

[11] J. C. Shepherdson and H. E. Sturgis. Computability of recursive functions. *Journal of the Association of Computing Machinery*, 10:217–255, 1963.

[12] M. Sipser. *Introduction to the Theory of Computation.* Course Technology, 1997.

[13] V. V. Vyugin. Algorithmic complexity and stochastic properties of finite binary sequences. *The Computer Journal*, 42:294–317, 1999.

[14] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.