ELSEVIER

Physica D 120 (1998) 214–235

PHYSICA D

Analog computation with dynamical systems

Hava T. Siegelmann [a,*], Shmuel Fishman [b,1]

[a] *Faculty of Industrial Engineering and Management, Technion, Haifa 32000, Israel*
[b] *Department of Physics, Technion, Haifa 32000, Israel*

Abstract

Physical systems exhibit various levels of complexity: their long term dynamics may converge to fixed points or exhibit complex chaotic behavior. This paper presents a theory that enables to interpret natural processes as special purpose analog computers. Since physical systems are naturally described in continuous time, a definition of computational complexity for continuous time systems is required. In analogy with the classical discrete theory we develop fundamentals of computational complexity for dynamical systems, discrete or continuous in time, on the basis of an intrinsic time scale of the system. Dissipative dynamical systems are classified into the computational complexity classes P_d, Co-RP_d, NP_d and EXP_d, corresponding to their standard counterparts, according to the complexity of their long term behavior. The complexity of chaotic attractors relative to regular ones leads to the conjecture $P_d \neq NP_d$. Continuous time flows have been proven useful in solving various practical problems. Our theory provides the tools for an algorithmic analysis of such flows. As an example we analyze the continuous Hopfield network.

PACS: 05.45; 89.80
Keywords: Theory of analog computation; Dynamical systems

1. Introduction

Digital computers can be viewed as *dynamical systems*. A dynamical system is defined by a set of equations that describe the evolution of a vector of variables called the state of the system [1–6]. It is a map of a phase space onto itself. Considering the configuration of a Turing machine (the mathematical abstraction of a digital computer) [7] as a state in some space, the update rules describing the behavior of the machine constitute a dynamical system. The input for the machine is the initial condition for the dynamical system. The series of consecutive configurations during the computation process define a trajectory in the space of Turing machine configurations. This trajectory may converge to a halting configuration from which the output can be read. The behavior of *physical systems* can also be modeled by dynamical systems. A physical system is prepared in some initial state from which it evolves in its phase space according to some equations of motion. Many physical systems are dissipative and hence converge to attractors; the attractor may be a simple fixed point, a limit cycle or it may be chaotic with a fractal dimension. This correspondence between computers and physical systems, through the common description by dynamical systems prompts us to view the evolution of physical systems as a process of computation. Analyzing this process and its complexity can contribute to the understanding of computation in nature.

* Corresponding author. E-mail: iehava@ie.technion.ac.il.
[1] E-mail: fishman@physics.technion.ac.il.

0167-2789/98/$19.00
PII: S0167-2789(98)00057-8

In the theory of computation problems are classified by the growth of the computation time as a function of the size of the input (number of bits) [7]. In particular, if the time is polynomial in the size of the input the problem is said to belong to the class P, while if it is exponential, it belongs to the class EXP. There is also the class NP of nondeterministic polynomial problems, where if the answer is guessed, it can be verified in polynomial time. The complexity of the dynamics of dissipative systems is characterized by the properties of their attractors. The convergence to the attractors is usually exponential. The complexity of an attractor is quantified by its Kolmogorov–Sinai (KS) entropy and its Lyapunov exponents [1,5,6]. These essentially measure the sensitivity to changes in initial conditions, and therefore can be viewed as a measure of the degree of chaos. For stable fixed points and limit cycles the largest Lyapunov exponent (and the KS entropy) are non-positive, while they are positive for chaotic attractors. This is a manifestation of the fact that chaotic attractors exhibit a much higher degree of complexity. In analogy with the standard computational complexity classes (P, NP, EXP, etc.) we define computational complexity classes (P_d, NP_d, EXP_d, etc.) that are relevant for physical dynamical systems. In computer science it is an open question whether $P = NP$. For the corresponding classes P_d and NP_d it is here conjectured that $\mathrm{P}_d \neq \mathrm{NP}_d$.

In this work we have a more ambitious aim than developing a fundamental philosophical view of the physical world as performing processes of computation. We develop a theory of computation for alternative nondigital hardware. Whereas the state space of digital computers is *discrete*, physical systems are described by differential equations or maps in a *continuous* phase space, and are thus analog computers. There seems to be an algorithmic advantage in using analog computers for specific problems. Many of the discrete problems stated in classical computer science are solved by imposing a combinatorial structure on the problem. This search for a solution may be highly time-consuming. A prime example is the linear programming problem, with its polyheder structure of solutions [8]. A search in this space via the simplex algorithm has exponential worst case behavior. On the other hand, interior point algorithms such as Karmarkar's polynomial time algorithm approach the solution from the inside of the continuous polyder. This exemplifies the power of this approach of using a continuous rather than discrete phase space. Using our theory, it is claimed that some flow based on the principle of the Karmarkar algorithm have optimal performance [9]. This paper is aimed to provide the theoretical background for an algorithmic and complexity analysis of computation in continuous phase space and continuous time.

We focus on dissipative classical dynamical systems [1,5,6]. This helps in making our theory become realizable by classical (as opposed to quantum) small-scale physical systems (since there dissipation is usually not negligible). Such systems contract phase space volume, and are thus characterized by flow to attractors. We interpret the attractor to which a system flows as the output, the initial condition plays the role of the input, and the flow is the process of computation. What determines the output is the location of the initial condition relative to the basin boundaries of the various attractors. These boundaries may be fractal even if the attractors are regular [10–12]. Generally, there is no way to predict analytically the basins and their boundaries, and thus the general evolution is unpredictable and can be very complicated.

A similar view of the process of computation as a flow from an initial condition to an attractor has been taken by a number of researchers. Brockett introduced a set of ODEs that perform various tasks such as sorting and solving linear programming problems [13]. In the comprehensive book of Helmke and Moore [14] one can find numerous other applications and references, among them, the state of the art in dynamical systems for linear programming. The Hopfield neural network is of particular interest because it is related to NP-complete problems, and because it provides a natural interface between discrete and continuous computation [15,16]. Our theory is, to some extent, a continuation of their work, in that it provides a natural framework for the complexity analysis of continuous time algorithms. Furthermore, we allow for attractors which are not fixed points or limit cycles.

We distinguish our work from the following three related lines of work. First, in an effort to bound the computational power of continuous systems, one line of work concentrated on performing step by step simulations of discrete computational models (see Section 2.3 and e.g. [17]). Whereas this line does not express the inherent computational capabilities of continuous systems, we take continuous systems as is, and interpret their evolution as a process of computation.

The second line of work uses attractor systems to model neurodynamics, and in particular associative memory (see Section 2.4). Our theory can be used to analyze some of these models in terms of computational complexity. Attractor models of awareness, feeling and related activities of the brain (see [18]) are beyond the scope of this paper.

Third, analog computation can be utilized to test possible theoretical limitations of the "physical Church–Turing thesis" [19] that states that the computational capabilities of any physical device cannot exceed (in idealization) that of a Turing machine [20]. If a device that computes problems that cannot be computed by the Turing machine (and therefore digital computers) is found, it will challenge the "physical Church–Turing thesis" and will therefore be of great interest. Some theoretical analog models of computation have the capability of computing beyond the Turing limit [21,22], but no realizable super-Turing system has been noted. We do not suggest the current work as providing a step towards the identification of super-Turing natural systems. We rather concentrate on perceiving physical systems as readily available special purpose computers. Various other related work is discussed in Section 2.

The outline of the paper is as follows: In Section 2 we characterize analog computation; there we also discuss previous related work and its relation to this paper. Section 3 presents the continuous systems which are studied in the framework of our model. We interpret the dynamics of these systems as computing an output from an input, and postulate their characteristic time scale as the basis for a measure of time complexity. We argue that this notion of complexity is well-defined. Section 4 compares continuous systems with their time discretization. The notion of time complexity introduced in this paper is compared with the classical definition for discrete time maps. The two following sections build on the foundations of Section 3 and interpret different scenarios of dynamical evolution as complexity classes. We define there the computational complexity classes P_d, Co-RP_d, NP_d, and EXP_d according to the attractor approached. Section 5 focuses on regular attractors; it provides two different methodologies for verifying the attractor, a deterministic one and a probabilistic one, with the associated classes P_d and Co-RP_d. Section 5.2 demonstrates our theory in an analysis of systems having a Lyapunov functional, concentrating on gradient flows and the continuous time Hopfield neural network. Section 6 considers chaotic attractors: These are typically isolated but may also be intermingled or exhibit a crisis. Identifying points in the basin of attraction of an isolated chaotic attractor is shown to be in NP_d; the other two cases are harder, and can only be dealt with probabilistically. We close with Section 7 where we summarize our results and present a list of open questions.

2. Analog computation

Our theory belongs to the field of analog computation, and in particular it is related to models of analog neurodynamics. The main property that differentiates analog from digital computation models is the use of a continuous state space. It is worth noting that analog computation may be considered a coarse grained theory for digital computation, the way hydrodynamics is a continuum theory for atomic systems. There is no formal list of characteristics of analog models of computation which is agreed upon. In what follows we propose one possible formalization of the concept of an analog computer, motivated by the view of physical systems as performing computation.

(i) *Real constants:* Analog systems can be characterized by the existence of real constants that influence the macroscopic behavior of the system. For example, planetary motion is used to measure

time with very high precision although we know the gravitational constant G only to a few digits. The planets, of course, evolve according to the exact value of G, irrespective of its measurement by humans. In contrast, in digital computation all constants are in principle accessible to the programmer. Other physical constants that may have real values are Planck's constant, the charge of the electron, various wavelengths, masses of particles, and so forth.

(ii) *Continuity in the dynamics:* The motion generated by the system is locally continuous. Computationally, we say that exact comparisons of the form "if $x \geq 0$ compute one thing and if $x < 0$ continue in another computation path" or "tests for 0" are not allowed.

(iii) *Continuous time update:* Classical physical systems are described by differential equations. Therefore we focus on systems with continuous time update. We do not view this as a basic requirement from a model of analog computation.

(iv) *Discrete input–outputs:* Classical complexity theory measures computational resources as a function of the length of the input. In order to create an approach which can be related to the classical one, it is advised that the input and output be discrete. Discreteness of the output of physical systems is dictated by the limited precision of the measurement tools used to probe the continuous phase space.

The theory we present has the above features and the additional simplifying requirement of a finite dimensionality (unlike analog cellular automata for example, which evolve in an infinite-dimensional space, see Section 2.1). Thus we focus on ODEs and not on PDEs, since for the former more results are available.

In what follows we summarize previous work in the field of analog computation and point out the novelty of our work. We begin with discrete-time models, go through hybrid continuous-discrete models, and reach continuous time computation. We close this section with a mention of relevant work in the field of analog neurodynamics.

2.1. Discrete time models

Two well-known analog computation models are the BSS model of computation over the real numbers [23] and the SiSo analog recurrent neural network [21,24]. Smale was the first to insist on a computational model in which the operations are done on the values irrespective of their radix two representation. Together with Blum and Shub they introduced a mathematical model (the BSS model) of computation that includes real state values and real constants; this model does not have the continuity feature. Siegelmann and Sontag suggested another model that is based on neural networks. Unlike the BSS model, their model has continuous dynamics, and it has a finite number of registers (neurons). Both models allow computational capabilities which are richer than those of the classical standard digital computer (super-Turing). Some work was done on comparing these models and on interleaving them [25–27].

The generalized shift map (GS) was discussed by Moore [28]. He shows it to be computationally universal, and claims that this model is physically realizable. This is in contrast to his model [29], which he views as unphysical, and the dynamical recognizers [30]. An extension of the GS to include real constants was suggested in [22]. This analog version of the GS has super-Turing computational power as well.

Cellular automata (CAs) are a computational model which is an infinite lattice of discrete variables with a local homogeneous transition rule. CAs contain Turing machines as a special case, and are thus computationally universal [31]. When the variables are reals the machine is sometimes called an analog cellular automaton, or a coupled map lattice (CML) [32]. It can be thought of as a generalization of both the BSS and SiSo models. The BSS model is equivalent to a Turing machine with real valued cells, and hence is easily simulated by a CML. A SiSo network can be simulated by a CML, where every cell computes as a neuron and the neighborhood structure is big enough to include the finite size of the simulated network. As a result even a finite coupled map lattice is computationally stronger than a universal Turing machine. CAs and coupled map lattices are used in modeling a broad

class of physical phenomena [33]. Coupled map lattices can be considered time and space discretizations of PDEs, which are thus computationally universal as well [34].

2.2. Hybrid models

Motivation for some of the work on hybrid computation is derived from the realization that the functionality of controllers is not fully described in terms of discrete dynamics. Hybrid systems combine discrete and continuous dynamics, usually represented by ODEs that are governed by finite automata. Among famous hybrid models are the works by Tavernini [35], Back et al. [36], Nerode and Kohn [37], Brockett [38–40], and Branicky [41,42]; more can be found in [43–45]. The main interest in hybrid systems stems from their practicality – such as in "stepper motor" and in transmission [40] as well as their stabilizing properties [46].

2.3. Continuous time models

A fundamental question is the computability with differential equations. ODEs were used to simulate various discrete time models, thus providing lower bounds on their computational power. Brockett [40] demonstrated how to simulate finite automata by ODEs; Branicky [17] generalized this result to simulate Turing machines, thus proving their computational universality. These systems retain the discrete nature of the simulated system, in that they follow the computation of a given discrete map step by step via a continuous equation. In the present paper, on the other hand, we characterize the computation of realizable continuous systems rather than simulating discrete maps with differential equations, in an attempt to investigate the power of continuous phase space systems. This philosophy can be applied to solve concrete problems in an efficient manner. In [9] we analyze the dynamical system for linear programming introduced by Faybusovich [47] and compare it with optimal algorithms. This system is related to Brockett's "bracket flows" [13]. Numerous other problems solvable by dynamical systems are shown in the book of Helmke and Moore [14].

Another line of work regarded that the so-called "general purpose analog computer" was dominated by Shannon [48], Pour-El [49] and Rubel [50–52]. Despite the similar title, that "analog computer" is very different from our approach. It describes a mathematical abstraction that consists of a finite number of boxes with plenty of feedback. The boxes are of five types: one produces the running time, one produces any real constant, one is an adder, one is a multiplier and the last and crucial one is an integrator. Although it was originally suggested as a general computational machine to handle computational physics, it was found too weak to solve even some very standard problems there (such as solving the Dirichlect problem and generating Euler's gamma function); their general purpose analog computer produces only solutions of initial-value problems for algebraic ordinary differential equations [52]. The book [53] describes a particular interesting view of computability in analysis, differential equations, and Banach spaces. The questions raised in these seminal works inspired many researchers, including us, to focus on continuous differential equations as a computational device.

2.4. Analog neurodynamics

In the area of neural networks, intelligence is explored from the viewpoint of dynamical systems. These describe the evolution of neural activations with time. The neural activity is described by a trajectory in the state space of a large dimension, say $\mathbb{R}^N$ where N is the number of neurons. As in our theory, the trajectory can be thought of as describing a computation, or equivalently solving a computational problem. The initial state is equivalent to the input of a computational problem, the evolution along the trajectory is the computation process, and the attractor describes the solution. Consequently, the computational capabilities of dynamical systems that are relevant to neurodynamics – nonlinear dissipative systems – are of a great interest.

Content addressable memory allows to recall a stored word without referring to its physical location;

associative memory allows for recal based on partial or partially erroneous information. Both types of memory are typically modeled by dissipative dynamical systems for which an energy (Lyapunov) functional is defined and patterns reside in the location of local minima of the energy. The most popular model is probably the Hopfield network [16,54,55]. Other related models are contained in the works by Amari, Anderson, Kohonen, Nakano, Willshaw, Little, Atiya and Abu-Mostafa, and Morita [56–66]. The meaningful attractors of these networks – where information is stored – are all simple: either stable fixed points or limit cycles. There are various models of neural activity that suggest the role of chaos. Yet, they do not consider the resulting chaotic systems in computational terms.

Here we present a computational view of dissipative dynamical systems. The attractors can be of any kind, and hence the richness of this theory. Our interface is motivated to some extent by the understanding and interpretation of computation in neural networks, and thus includes the previous works as special cases.

3. Our model: Computation in dynamical systems

We consider autonomous dynamical systems defined by a set of ODEs

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{F}(\mathbf{x}(t)), \tag{3.1}$$

where $\mathbf{x}(t)$ is a d-dimensional vector and $\mathbf{F}$ is a d-dimensional vector function of $\mathbf{x}$. We concentrate on ODE's but the concepts we propose can be applied to maps as well. Discrete time maps

$$\mathbf{x}_{n+1} = \mathbf{T}(\mathbf{x}_n) \tag{3.2}$$

can be associated with ODEs in various ways such as the Poincaré map [1] or by the stroboscopic map (see Section 4). Our point of departure is dissipative dynamical systems of classical physics. In dissipative dynamical systems all points found in a given volume in phase space at one time move in such a way that at a later time they occupy a smaller volume. As a consequence dissipative systems are characterized by attractors.

For simplicity, we consider only dissipative dynamical systems with the additional feature that

(∗) Convergence to attractors is exponentially fast.

This requirement does not limit the range of applicability of our theory much because such systems are abundant in nature.

To make statements regarding to what is "likely" or "typical" scenario, one has to define a measure on the space of dynamical systems, which is of infinite dimension. In order to explore such a situation Hunt et al. [67,68] introduced the term "prevalent" that is a generalization of "almost everywhere" to infinite dimensional spaces. The term corresponding to "measure zero" is "shy". It is known that for almost every differentiable map all the fixed points (and periodic orbits) are hyperbolic (namely, none of the stability exponents is of unit modulus). This assures that a situation – where a Lyapunov exponent at a fixed point vanishes (stability exponent of unit magnitude) and consequently the convergence is not exponential – is "shy" corresponding to vanishing probability measure. A similar theorem was proved for continuous flows [67]. Here we assume exponentially fast convergence to chaotic attractors as well (as is the prevalent case for fixed points). We base it on the conjecture that usually (in the sense of prevalence) the sets of points on the attractor for which the Lyapunov exponent is non-negative in directions transverse to the attractor, is of measure zero [69]. (For the discussion of transverse Lyapunov exponents see e.g. [70].) As for the type of attractors, it is believed that systems with chaotic attractors are neither prevalent nor shy, but we are not aware of any decisive relevant rigorous result; our theory allows for all types of attractors.

A computing device maps an input into an output; the internal evolution, or its trace by observers, is regarded as the computation. For dissipative dynamical systems the initial condition corresponds to the input and the system evolves until approaching an attractor which represents the output. Because the actual convergence to an attractor takes an infinite time, we do not require complete convergence but rather agree

that the computation is complete when an ϵ-vicinity of the attractor is approached and the system is confined there. The time it takes to converge to an ϵ-vicinity of the attractor increases with decreasing ϵ.

Comment. Deciding to stop the computation in an ϵ-vicinity of an attractor may sound obscure because the ϵ-vicinity may contain several attractors that can be resolved only for smaller values of ϵ. This is actually not a problem but rather a manifestation of the richness of the corresponding computation. As the resolution is increased, new and more refined results are found.

The precision parameter ϵ also serves to maintain the input and output in a finite form, enabling comparison with the classical computation theory. The finiteness/discreteness of the input and output is a crucial requirement in the theory of computing, assuring that the power of models is based purely on the internal structure rather than on higher input/output precision. This has to be emphasized in the case of dynamical systems where the phase space is continuous.

We are now ready to define computation in dissipative dynamical systems. The input is the initial condition specified with precision ϵ. The output is a description of the attractor with ϵ precision. The convergence time t_c is the time so that for all $t > t_c$ the distance from the attractor is less than ϵ. For classical computers the convergence time is the time until halting, and is identified with the computation time. This is not the case for dynamical systems, where the attractor has to be verified and distinguished from saddle points for example. The *total computation time*, t_t, is thus the sum of the convergence time t_c and the *verification time* t_v:

$$t_t = t_c + t_v. \tag{3.3}$$

We want to define the complexity of the computational process in terms of the total computation time t_t. Because complexity is measured in numbers without units, we must express t_t as a multiple of some time unit inherent to the system. In discrete time computational models, the total computation time is quantified by *the number of steps* until halting. Discretization of a continuous process, e.g. by the Poincaré map, does not give a faithful estimate of the time complexity of the continuous process. A similar disadvantage occurs for a measure which is based on counting the number of grid cells traversed by the trajectory. Our suggestion is to introduce the concept of a *characteristic time scale* that is defined by the rate of convergence of the underlying physical system.

We next define the characteristic time scale for our class of exponentially converging dissipative systems. For simplicity we assume first that the stable nonwandering set is a fixed point $\mathbf{x}^*$. Around this point one can linearize the system

$$\frac{d\mathbf{x}}{dt} = \mathbf{F}(\mathbf{x}(t))$$

to obtain

$$\dot{\delta}\mathbf{x} = M\delta\mathbf{x} \tag{3.4}$$

where $\delta\mathbf{x} = \mathbf{x} - \mathbf{x}^*$, and M is the *stability matrix* defined by

$$M_{ij} = \left.\frac{\partial F_i}{\partial x_j}\right|_{\mathbf{x}=\mathbf{x}^*}. \tag{3.5}$$

The eigenvalues of the stability matrix are called the Lyapunov exponents. Let us denote the eigenvalues of M by λ_i and their real parts by $\mathrm{Re}(\lambda_i)$ (the fixed point is assumed to be stable, hence all $\mathrm{Re}(\lambda_i)$ are negative). Let λ_1 be the eigenvalue with negative real part which is smallest in absolute value, and let $\lambda = |\mathrm{Re}(\lambda_1)|$. The rate of convergence is determined by λ. In the vicinity of the fixed point

$$|\mathbf{x}(t) - \mathbf{x}^*| \sim e^{-\lambda t}, \tag{3.6}$$

leading to the definition of the characteristic time $\tau_{ch} = 1/\lambda$. During the time $\tau_{ch} \log 2$ an additional bit of the attractor is computed. Therefore τ_{ch} is the characteristic time scale for the convergence in the linear regime. (If the matrix cannot be diagonalized but can only be transformed to a Jordan form, the leading exponential behavior is similar.)

Later we justify our choice of τ_{ch} as the characteristic time for the whole computation t_t. The resulting complexity measure has the following invariance property: if the vector field $\mathbf{F}$ is multiplied by a constant a the computation time t_t changes, whereas the complexity of the process should remain unchanged.

And indeed, this multiplication scales τ_{ch} by the same amount, a, leaving the complexity unaffected.[2] A time scale such as τ_{ch} exists for all types of attractors (fixed point, limit cycle or chaotic) as long as exponential convergence is assured, as is the case for the dynamical systems we consider in our model.

We have defined the convergence time t_c as the duration required to flow from the initial condition to the ϵ-vicinity of the attractor. In general, t_c is the sum of three contributions:

$$t_c = t_\epsilon + t_B + t_f, \tag{3.7}$$

where t_ϵ is the time of flow in the linear regime, close to the attractor; t_B is the time required to leave the vicinity of the basin boundary (due to the placement of the initial condition); and t_f is the time of flow outside the linear regime and the vicinity of the boundary. By (3.6), the time it takes to flow in the linear regime from a distance δ from the fixed point to its ϵ-vicinity is

$$t_\epsilon = \frac{1}{\lambda}\left|\log\frac{\epsilon}{\delta}\right|. \tag{3.8}$$

We next regard flow near the boundary. Assume the initial point is in a narrow region of width η in the vicinity of the basin boundary. A corresponding trajectory may flow for some time along the boundary and is repelled in a time of the order

$$t_B \sim \frac{1}{\tilde{\lambda}}|\log\eta|. \tag{3.9}$$

Here $\tilde{\lambda}$ is obtained from a linearization in a specific boundary area: $\tilde{\lambda} = \min \mathrm{Re}(\lambda_i)$ where the minimization is over the eigenvalues with positive real part (corresponding to repelling directions). The above holds for a specific saddle point. A general bound for t_B is obtained by choosing $1/\tilde{\lambda}$ as the minimum among all saddle points. For a fixed initial condition and asymptotically small ϵ, t_ϵ dominates over t_B. However, for initial conditions where ϵ is not significantly smaller than η this is not necessarily the case. Here t_B cannot be ignored and we have to take into account the characteristic time for flow near the boundary $1/\tilde{\lambda}$. We do not know how to deal with such initial conditions, but note that these constitute a very small subset of the space of possible initial conditions: The volume within a distance η from the boundary is $\sim \eta^{d-d_b}$ where d is the dimension of the phase space and d_b is the dimension of the boundary (that may be fractal).

The value of t_f is fixed for an initial condition, and hence is asymptotically dominated by t_ϵ. (In Section 5.2 we calculate t_f for a particular class of systems.) Therefore for initial conditions which are not close to the boundary we can approximate

$$t_c \sim t_\epsilon = O(|\log\epsilon|), \tag{3.10}$$

and choose τ_{ch} as the time scale for the complete convergence process.

3.1. Computational complexity for continuous systems?

An objection may be raised to the notion of computational complexity for continuous time systems.[3] One argument is that the computation time is arbitrary since the time variable t can be replaced by another variable s where $t = g(s)$, such that the derivative

$$\frac{ds}{dt} \equiv f(t) > 0. \tag{3.11}$$

Substituting (3.11) into (3.1) we can find that

$$\frac{d\mathbf{x}}{ds} = \frac{\mathbf{F}(\mathbf{x})}{f(g(s))} \equiv \mathbf{F}_s(\mathbf{x}, s). \tag{3.12}$$

For example, if $ds/dt = e^t$ then the exponential convergence to the attractor is lost. We do not believe this is a valid objection to the definition of computation time for continuous time systems for the following reason. Physical time of a given laboratory system has a meaning that is well-defined (relative to other physical systems), irrespective of our definition of the parameter s. A similar change of the time variable can be defined also for discrete time computers. That is,

[2] For functions $\mathbf{F}$ that are sufficiently smooth, in every neighborhood (in function space) of the trivial vector field $\mathbf{F}(\mathbf{x}) = 0$, there are flows $\mathbf{x}(t)$ that are chaotic, see [71,72]. Multiplying $\mathbf{F}$ by a constant a indeed does not affect the complexity of the function, but the limit $a \to 0$ should not be taken since in this limit the system is clearly regular.

[3] We are grateful to C. Moore and J. Crutchfield for pointing out to us.

one can speed up the computation by defining a new discrete time step of computation that allows for $f(t)$ operations rather than one only. We also note that in such cases the system (3.12) is nonautonomous and hence is outside of our scope.

Another possible objection can be raised: for a given system with exponential convergence to a fixed point one can find a system with faster convergence. For example in the linear regime, the equation

$$\frac{dx}{dt} = -\lambda x, \tag{3.13}$$

with $\lambda > 0$, has the same fixed point and can be replaced by

$$\frac{dx}{dt} = -\lambda \frac{x}{|x|}, \tag{3.14}$$

that reduces to

$$\frac{d|x|}{dt} = -\lambda, \tag{3.15}$$

which converges to the origin in finite time. Our response is that such a replacement requires the exact full knowledge of the fixed points and the ability to design the analog system (3.14) according to the exact knowledge. If for a given initial condition the fixed point is known beforehand, there is no need for performing the computation.

3.2. *Nonexponentially convergent systems*

The dynamical systems that are at the focus of this paper converge exponentially to attractors. The notion of characteristic time is not assured to exist for other dynamical systems. Assume one of the eigenvalues of the stability matrix (3.5) vanishes for the attractor. Along the direction of the associated eigenvector (using the Taylor expansion) the deviation from the fixed point satisfies

$$\dot{y} = \beta y^m, \tag{3.16}$$

where $m \geq 2$ is the order of the first nonvanishing derivative at the fixed point in this direction. The solution is

$$y^{1-m} = -\beta(m-1)t + \text{const.} \tag{3.17}$$

leading to a power law time dependence of the distance from the fixed point.

In particular, let y_1 and y_2 be the values at time t_1 and $t_2 = t_1 + \Delta t$, respectively. For large t_1 and t_2, the ratio is

$$\frac{y_1}{y_2} = \left(\frac{t_1}{t_2}\right)^{1/(1-m)} = \left(1 + \frac{\Delta t}{t_1}\right)^{1/(m-1)}, \tag{3.18}$$

that depends explicitly on t_1 and there is no fixed rate of convergence for the fixed point and hence no characteristic time scale for computation. Note the difference between this power law and exponential convergence of the form $\delta x = \delta x_0 e^{-\lambda t}$ where the ratio

$$\frac{y_1}{y_2} = e^{\lambda \Delta t} \tag{3.19}$$

depends only on the time difference Δt (and not on t_1) and λ is the convergence rate.

4. Maps versus continuous time systems

Continuous time systems are natural for the description of the dynamics of physical objects, but there are situations where the dynamics is modeled by maps [1,3,4,6]. Therefore, the scope of the theory should be extended to include maps. Furthermore, the inclusion of maps facilitates a relatively easy comparison with the concepts of standard digital computation. Maps and systems that evolve continuously in time exhibit many similar dynamical properties, in particular attractors of similar nature. Moreover, each continuous system (3.1) can be associated with the discrete *stroboscopic map*. There the position of the system in phase space is monitored at intervals of length τ, namely at times $t = n\tau$, where n is an integer. This discretization results in the map:

$$\mathbf{T}(\mathbf{x}_n) = \mathbf{x}_n + \int_{n\tau}^{(n+1)\tau} \mathbf{F}(\mathbf{x}(t))\, dt. \tag{4.1}$$

For an autonomous system the value of $\mathbf{x}(t)$ in the integral depends only on the last $\mathbf{x_n}$, and therefore the integral depends only on $\mathbf{x}_n$. For small τ, the map $\mathbf{T}$ can be approximated by Taylor expanding $\mathbf{F}$ around

$\mathbf{F}(\mathbf{x}_n)$, and integrating the resulting expansion term by term with respect to time leading to:

$$T_i(\mathbf{x}_n) = (x_n)_i + F_i(\mathbf{x}_n)\tau + \sum_j \frac{\partial F_i(\mathbf{x}_n)}{\partial x_j} F_j(\mathbf{x}_n)\frac{\tau^2}{2} + \cdots \quad (4.2)$$

If the series is convergent, the map (4.2) is identical to the underlying continuous system. If only a finite number of terms is taken it constitutes an approximation to the continuous system, that improves as τ decreases. Although for sufficiently small τ the map is a good approximation of the continuous system, in the course of the dynamics these two systems will usually flow apart due to the accumulation of differences. The limit of $\tau \to 0$ and the infinite time limit do not necessarily commute and one should study specifically which dynamical invariants (such as attractors) of the stroboscopic map, where a finite number of terms in (4.2) is taken, approach the ones of the continuous system in the limit $\tau \to 0$.

For chaotic systems, the qualitative nature of the attractors is sensitive to infinitesimal changes of parameters. Since τ can be treated as a parameter, even in the limit $\tau \to 0$, the attractors for $\tau \neq 0$ may be different from these at $\tau = 0$. This is the situation even for the logistic map [1,73] where arbitrarily small perturbations may lead to the replacement of the chaotic attractor by a periodic orbit. Consequently one cannot be assured that a map of the form (4.2) taken to any arbitrary order and for arbitrary small τ will converge to the same attractor as the corresponding continuous system. Therefore the results of the computation may be different for the two systems.

Hyperbolic attractors, on the other hand [2], remain stable under small perturbations and therefore the map (4.2) and the corresponding continuous systems lead to the same computations, provided the initial conditions are sufficiently far from the basin boundary. This requirement is necessary since the basin boundaries do not necessarily approach each other in the limit $\tau \to 0$.

In Section 5.2 it is stated that the computation in both systems is equivalent if a Lyapunov functional exists.

The case of convergence to a stable fixed point can be studied in detail. For sufficiently small τ, stable fixed points of the continuous flow remain stable under time discretization. For small τ, Eq. (4.2) can be approximated as

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \tau\mathbf{F}(\mathbf{x}_n), \quad (4.3)$$

which can be recognized as the Euler approximation. The lowest nontrivial order of (4.2) was taken to obtain this map. The fixed points of (4.3) and the original continuous flow are identical, since at the fixed points of the continuous system $\mathbf{F}(\mathbf{x}^*) = 0$. (It is not clear, however, that the basins of attraction of the fixed points in the two systems are similar or even approach each other in the limit $\tau \to 0$.)

We next compare the convergence time of the continuous time equation with its Euler approximation. The tangent map (the map of the small deviations $\delta\mathbf{x}_n$ from the fixed point) is

$$\delta\mathbf{x}_{n+1} = \delta\mathbf{x}_n + \tau M\,\delta\mathbf{x}_n$$

or equivalently

$$\delta\mathbf{x}_{n+1} = M_T\,\delta\mathbf{x}_n,$$

where

$$M_T = 1 + \tau M \quad (4.4)$$

and M is given by (3.5). The eigenvalues of M_T are

$$\Lambda_i = 1 + \tau\lambda_i, \quad (4.5)$$

where λ_i are the eigenvalues of M. Similarly to the continuous case, the convergence rate is determined by the largest eigenvalue $\Lambda = 1 - \tau\lambda$ where $\lambda > 0$. The distance from the fixed point decreases with the number of iterations n as $\Lambda^n = (1 - \tau\lambda)^n$ and in the limit $\tau \to 0$ it approaches the result found for the continuous system $e^{-\lambda\tau n}$. The convergence to the fixed point is faster for the map than for the continuous system since $|\log(1 - \tau\lambda)| > \tau\lambda$ for $\lambda\tau < 1$. For an unstable fixed point, on the other hand, the escape rate is faster for the continuous system than for the corresponding map. Note that stable fixed points of the continuous system remain stable for the corresponding map (4.3) for values of τ such that $\tau|\mathrm{Re}(\lambda_i)| < 2$.

Maps have two different measures of time complexity. One corresponds to the notion of the characteristic time of convergence, introduced for continuous time systems as $\tau_{ch} = 1/\lambda$. This is the characteristic number of steps $n_{ch} = 1/|\log \Lambda|$, where Λ is the largest eigenvalue (that is smaller than unity) and the rate of convergence is $|\log \Lambda|$. For maps which are obtained as a discretization of continuous flows, τ_{ch} and n_{ch} are related by

$$\tau_{ch} = n_{ch}\tau. \tag{4.6}$$

Note that n_{ch} does not have to be an integer. The second (classical) complexity measure is the number of steps, and thus the classical characteristic time is unity.

We next calculate the time required for reaching an ϵ-vicinity of an attractor for general maps. In analogy with the definition of the time t_ϵ from Section 3, we define n_ϵ as the corresponding number of steps. Thus we have that

$$\epsilon \approx \delta e^{-n_\epsilon/n_{ch}} \tag{4.7}$$

for ϵ and δ ($\epsilon < \delta$) in the linear regime. This yields

$$n_\epsilon \approx \left\lceil n_{ch} \left| \log \frac{\epsilon}{\delta} \right| \right\rceil. \tag{4.8}$$

As in the case of continuous computation, n_{ch} is a property of the system, while n_ϵ depends on the required precision.

5. Computing regular attractors

In the following two sections we discuss the complexity of computation with exponentially convergent dissipative dynamical systems. In this section we consider systems which flow to either fixed points or limit cycles, and in Section 6 systems with chaotic attractors are discussed. We recall that the computation time is the sum of the time required for convergence and the verification time. Since we consider systems with exponential convergence, the convergence time is $O(|\log \epsilon|)$ for all types of attractors. What makes the difference is the verification procedure. It characterizes the type of computation (e.g. deterministic, probabilistic or nondeterministic) and its efficiency.

For simple attractors we propose two types of verification procedures. One requires an external device computing linear equations; this defines the deterministic efficient class P_d. The second computes purely with the dynamical system, with no external tools. Under such setup a probabilistic process enables efficient computation, giving rise to the class Co-RP_d. The following section describes the two verification procedures which define the corresponding complexity classes P_d and Co-RP_d. Section 5.2 illustrates our theory with an analysis of systems with a Lyapunov functional, and considers in particular gradient flows and the Hopfield network.

5.1. Verifying a regular attractor: P_d and Co-RP_d

Assume first an attracting fixed point. When the phase space velocity $d\mathbf{x}/dt$ is found to be smaller than ϵ (in appropriate units) for some time, it is likely that the system has reached the ϵ-vicinity of a fixed point. Then we want to verify that this point is indeed an attractor and is not an unstable hyperbolic point.

5.1.1. Preliminaries: P and Co-RP

In the field of computational complexity problems are partitioned into complexity classes according to the difficulty of solving them. The standard model of computation is the Turing machine, and the difficulty of solving a problem is quantified by the resources required to solve it on a Turing machine. The class P consists of all problems/functions/languages which can be computed using polynomial (in the size of the input represented in binary) time resource. Co-RP is a class of decision problems defined relative to a modified version of the Turing machine. (In a decision problem, a computation is a decision to accept or reject the input.)

Definition 1 ([74], volume I). A probabilistic Turing machine *is a machine that computes as follows*:

(i) Every step of the computation can be designated in two possible ways, with probability p to choose one computation path and probability $1 - p$ to choose the alternative.

(ii) The number of steps in each computation is exactly the same.
(iii) Every computation ends with *reject* or *accept*.

The *error probability* of a probabilistic Turing machine $\mathcal{M}$ is the real value $e_{\mathcal{M}}(w)$ defined by the ratio of computations on input w resulting in the wrong answer to the total number of computations on w.

BPP is defined as the class of decision problems computed by polynomial time probabilistic Turing machines whose error probabilities are bounded above by some positive constant $c < \frac{1}{2}$. RP is a subset of BPP, where the error can occur on positive instances only. Co-RP is another subset of BPP, defined as the class of problems where the bounded error may occur on negative instances only. It is an open question in computer science whether any of these three probabilistic classes are strictly stronger than P. Their relation with the class NP (to be defined in the next section) is also unknown.

We next describe the two paradigms for verification.

5.1.2. Probabilistic verification

We want to verify convergence of a trajectory $\mathbf{x}(t)$. We start the verification at a point $\tilde{\mathbf{x}}$ on the trajectory, a point that is in the ϵ vicinity of a suspected fixed point $\mathbf{x}^*$. We create k new points $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_k$ by adding to $\tilde{\mathbf{x}}$ independent noise of amplitude up to ϵ. Now the system is called repetitively on each of these points; each time it is stopped after time t_v to be specified later. Consequently a cluster of initial conditions in a sphere of radius ϵ around the fixed point is generated. If the fixed point is stable, the sphere shrinks, while if it is a saddle point, it gets stretched in the unstable direction at the rate $e^{\mathrm{Re}(\lambda_1)t}$, where λ_1 is the dominating Lyapunov exponent. After a time of the order $\tilde{t_v} = (1/\mathrm{Re}(\lambda_1))|\log(\epsilon/\delta)|$ it reaches a distance of order $\delta > \epsilon$ from the fixed point. (It is assumed here that both ϵ and δ are in the linear regime; δ can be taken as 1000ϵ.) For a hyperbolic saddle point, there is a finite probability $p(\tilde{t_v})$ to find one of the k trajectories at a distance of order δ from the fixed point (say between $\delta/2$ and δ). If one of the trajectories is found at such a distance we conclude that the fixed point is unstable and continue the computation from $\tilde{\mathbf{x}}$ until it runs away to another fixed point. The probability that none of the trajectories reaches the distance of order δ from an unstable fixed point is $(1-p)^k$, that is exponentially small in k. Note that $t_v = O(\log \epsilon|)$ and therefore the verification time t_v is polynomial.

We next demonstrate that in many cases this problem can be indeed associated with the class Co-RP. This occurs when stable fixed points are verified with no error in polynomial time. For unstable fixed points the above procedure may give the erroneous result that the fixed point is stable. This can occur with a probability bounded by $(1-p)^k$. There is still the case that the initial point is on the stable manifold of an unstable hyperbolic point. Such a point should be verified positively as the output. However, our verification algorithm will give a negative result regarding the convergence with probability $1-(1-p)^k$, which increases with k, and cannot be bounded from above by a number smaller than $1/2$. Noticing that such initial conditions are of measure zero, it is reasonable to ignore the latter case, resulting in a Co-RP computation. There are specific situations where this probabilistic verification will not work. The classification of these is under exploration. [4]

5.1.3. Deterministic verification

This deterministic method has the disadvantage of requiring an external tool for computing linear equations and the eigenvalues of the stability matrix.

In the linear regime, the point $\mathbf{x}^{(n)}$ in the phase-space depends linearly on the previous point of the trajectory. That is,

$$\mathbf{x}^{(n)} = A\mathbf{x}^{(n-1)}.$$

(If the dynamical system is defined by differential equations we will consider its corresponding stroboscopic map (4.1).) The matrix A is of dimension $d \times d$, where d is the dimension of the phase space. It has to be calculated from the values of $\mathbf{x}^{(n)}$ in the linear regime. Thus, d^2 equations are required. It is not sufficient to trace the same one trajectory at different points because the equations resulting in this case would be linearly dependent. To obtain other equations, we thus

[4] We thank Amir Katz for alerting us to this possibility.

add to $\mathbf{x}^{(n)}$ perturbations with ϵ-amplitude, as was done in the probabilistic verification. Repeating this process d times we obtain d linearly independent trajectories which supply us the required equations. Using the external tool we then calculate the elements of the matrix. We repeat the same calculation a few times using a similar algorithm to assure that we find the same elements within a given precision, thus verifying that the motion is indeed in a linear regime.

Having calculated the matrix A, one is able to compute its eigenvalues and decide whether the appropriate fixed point is attracting or repelling. In the case of an attracting fixed point the computation is regarded as completed.

If we want the entries of A within ϵ precision, the x's are to be taken within ϵ_1 precision so that

$$c(A)\epsilon_1 < \epsilon,$$

where $c(A)$ is the condition number of the matrix A [75]. Conservatively, the computation takes then not more than $O((d^4 + \frac{1}{3}(d^4 - 1)d^2) \log^2 \epsilon_1)$ steps, which is polynomial in $\log \epsilon$. The resulting computational class is P_d.

5.1.4. *Verifying limit cycles*

So far it was assumed that the simple attractor in question is a fixed point. If it is a limit cycle, the degree of computational difficulty should be similar to the one of the fixed point, since in both cases the Kolmogorov–Sinai entropy vanishes [3,6]. The details of the calculation may be different. One way to detect a limit cycle is to measure the frequency of the motion and to check that the system approaches some constant frequency. Then one should verify that the system returns to some ϵ-vicinity, with a period corresponding to this frequency. The details of such a calculation are left for further study.

5.2. *Computation for systems with a Lyapunov (energy) functional*

We next calculate explicit bounds on the computation time for simple systems for which a global Lyapunov or energy functional exists. It is a differentiable functional $E(\mathbf{x})$ satisfying

$$\frac{\mathrm{d}E(\mathbf{x}(t))}{\mathrm{d}t} < 0, \tag{5.1}$$

along every trajectory $\mathbf{x}(t)$, except points $\mathbf{x}_i^*$ where this derivative vanishes. If the $\mathbf{x}_i^*$s are isolated then by the Lyapunov theorem [2,76] these fixed points are the attractors of the system, if there is a neighborhood of $\mathbf{x}_i^*$ where E is larger than $E(\mathbf{x}_i^*)$. In this case basin boundaries are smooth. Discretizing with small τ

$$\frac{1}{\tau}[E(\mathbf{x}_{n+1}) - E(\mathbf{x}_n)] = \frac{1}{\tau}[E(\mathbf{x}((n+1)\tau)) - E(\mathbf{x}(n\tau))] + O(\tau) \tag{5.2}$$

by the Taylor expansion, leads to $E(\mathbf{x}_{n+1}) - E(\mathbf{x}_n) < 0$. The fixed points of the continuous systems and of the corresponding discrete maps are identical, since these are the zeros of $\mathbf{F}$. Furthermore, in the limit $\tau \to 0$, the basins of attraction of all fixed points of the two systems approach each other. Hence, so do the results of the computation. (It is worthwhile to note that although the result of the computation is identical for these two types of systems, the trajectories leading from the initial to the final points may be different due to accumulation of differences.)

For these systems we can compute explicit bounds on t_f. Since at the fixed point $\mathrm{d}E/\mathrm{d}t$ vanishes, there is a region around the fixed point where it is smaller than in the rest of the basin of attraction (except near the boundary). Therefore there exists a δ so that when the distance from the fixed point is larger than δ,

$$\left|\frac{\mathrm{d}E}{\mathrm{d}t}\right| > v_\delta, \tag{5.3}$$

where v_δ is a positive number, bounding the rate of change of energy at a point x_δ in the linear regime. Consequently if ΔE is the difference between the maximal and minimal (at the fixed point) values of E, the time of flow t_f from the initial point to the point of distance δ from the fixed point satisfies

$$t_f < \frac{\Delta E}{v_\delta}. \tag{5.4}$$

The exact value v_δ depends on the particular system at hand. We next approximate v_δ for two examples: gradient flow and the Hopfield neural network.

5.2.1. Gradient flow

Gradient flow is defined by

$$F_i \equiv \frac{\mathrm{d}x_i}{\mathrm{d}t} = -\frac{\partial E}{\partial x_i}. \tag{5.5}$$

Using the chain rule

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \sum_i \frac{\partial E}{\partial x_i}\frac{\mathrm{d}x_i}{\mathrm{d}t} \tag{5.6}$$

we obtain that

$$\frac{\mathrm{d}E}{\mathrm{d}t} = -\sum_i F_i^2 = -|\mathbf{F}|^2 \tag{5.7}$$

and v_δ of (5.3) is the value of $|\mathbf{F}(\mathbf{x}_\delta)|^2$ where x_δ is a point in the linear regime of distance δ.

5.2.2. The Hopfield neural network

A more sophisticated example is the Hopfield network [16,54]. This is a particular type of neural network that was inspired by the spin glass model and the modeling of associative memory. It has been practically used to solve some optimization problems, such as small instances of the Traveling Salesman Problem. The continuous version of the Hopfield network [16] is defined by

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} \equiv F_i = -x_i + \sum_j W_{ij}\sigma_j(x_j) \tag{5.8}$$

with symmetric weights $W_{ij} = W_{ji}$ and $\sigma_j(x_j) = \tanh(x_j)$. In what follows we assume a "generalized Hopfield model" with the functions σ_j satisfying:

(i) $|\sigma_j| \le 1$,

(ii) $\partial\sigma_j(x)/\partial x > 0$,

(iii) $\frac{\partial^2\sigma_j(x)}{\partial x^2}$ is negative for positive x, positive for negative x and tends to zero monotonically in the limits $\pm\infty$.

The energy E is defined here by

$$F_i = -\frac{1}{c}\frac{\partial E}{\partial \sigma_i} \tag{5.9}$$

where c is a positive constant, leading to

$$\frac{\mathrm{d}E}{\mathrm{d}t} = -c\sum_i \left(\frac{\partial\sigma_i}{\partial x_i}\right)\left(\frac{\mathrm{d}x_i}{\mathrm{d}t}\right)^2. \tag{5.10}$$

Because of condition (ii), the differential of the energy is indeed negative.

We next estimate a bound on the time of flow. Denote $B_i = \sum_j |W_{ij}|$. Because of condition (i) there are no fixed points in the region $|x_i| > B_i$ since there $\mathrm{d}x_i/\mathrm{d}t < 0$ for positive x_i and $\mathrm{d}x_i/\mathrm{d}t > 0$ for negative x_i. For this reason, after sometime the equations of motion flow in the region $|x_i| < B_i$. In this region, because of condition (iii), $\mathrm{d}\sigma_i/\mathrm{d}x$ takes its minimal value at $|x_i| = B_i$. Denote this value by b_i, and let B_W be the minimal value of these b_i. Then v_δ of (5.3) can be estimated by

$$v_\delta = cB_W|\mathbf{F}(\mathbf{x}_\delta)|^2, \tag{5.11}$$

where $\mathbf{x}_\delta$ is a point in the linear regime of the fixed point.

6. Computation for chaotic systems

In this section we consider strange attractors, and in particular chaotic attractors; usually (though not always) strange attractors are chaotic. There are various options regarding the definition of chaotic attractors [70]; in this paper we follow the definition by Milnor. A is a "Milnor attractor" if it cannot be divided into invariant subsets (for an exact definition see [70]). This definition gives rise to various types of chaotic attractors; for example, the attractors of a system may be intermingled [77–79]. In this case near each point of the attractor there are points belonging to basins of other attractors; under other definitions of chaotic attractors [2,70] such intermingling will be considered as one attractor only. Also the phenomenon of crisis [1,80] may occur; this describes systems in which chaotic attractors change dramatically when a parameter is varied. It is believed, however, that most chaotic attractors are isolated (namely isolated chaotic attractors are prevalent [69]). This is based on the fact that other types of chaotic attractors (intermingled or in crisis) become isolated when a generic perturbation is added to the system. In the case of crisis this perturbation is a change in some parameter. In the case of

intermingled attractors these are symmetry breaking perturbations.

The behavior of a chaotic system can be very rich, exhibiting structures on all scales. These increasingly rich structures are revealed as the resolution is increased. Furthermore, systems where the number of attractors is arbitrarily large can be found [12,81] and thus many possible output responses are possible in the associated computation. These statements will be demonstrated in the following paragraph.

Assume a regular Hamiltonian system with an elliptic fixed point and trajectories $x(n) = \cos 2\pi\omega n$. If ω is rational, the trajectory is periodic, and it is quasi-periodic for irrational ω. If some disturbance is added so that the system is not integrable anymore, by Kolmogorov–Arnold–Moser theorem [2–4,82,83] most of the irrational trajectories around the elliptic fixed point will be only slightly deformed; by the Poincaré–Birkhoff theorem [2–4,84], the rational trajectories will be replaced by a sequence of fixed points that are alternatively unstable hyperbolic and elliptic. Around the unstable hyperbolic points there are two types of manifolds (stable and unstable); these may intersect many times and generate chaotic motion. The elliptic fixed points in the new sequence are again subject to a perturbation and thus a chain of fixed points is generated on a finer scale. This construction can be continued ad infinitum; island chains of arbitrary length are generated in this way. So far a Hamiltonian system was considered. When some small dissipation is introduced this process is truncated. Some of the chaotic parts will turn into chaotic attractors and many elliptic points will become periodic attractors. The period of the longest attractors of this type will be determined by the rate of dissipation. The weaker the dissipation, the longer the surviving orbits will be. Consequently for such a chaotic system we can reach very fine dynamics or equivalently, complex computation.

Where does the complex behavior of the dynamics show itself? The convergence is as fast as in the case of regular attractors, due to the assumption of exponential convergence. The verification, though, is much harder. (In some situations it is possible to define a simply structured *trapping region* [85] that encloses the attracting nonwandering set. In this case there is no need to verify the attractor, and the complexity is low. However, usually such a method is not applicable.)

6.1. Polynomial and exponential classes

We start with isolated attractors. If one has to determine the location of the attractor within the precision ϵ, there is a profound difference between regular and strange attractors. In the first case the computation is polynomial while it may require exponentially long time in the second case. A typical strange attractor is multifractal: its measure on various points is not uniform. Let the measure in a hyper-sphere of radius ϵ on the attractor be [1,86]

$$\mu_i = \epsilon^{\alpha_i}. \tag{6.1}$$

The measure is smallest in regions where $\alpha_i = \alpha_{\max}$. The time it takes to reach the region of the smallest measure is inversely proportional to it. Therefore, the time required to compute the location of the attractor is

$$t_A \sim \epsilon^{-\alpha_{\max}} = \mathrm{e}^{\alpha_{\max}|\log\epsilon|}, \tag{6.2}$$

which is exponential in the number of bits of the required precision. This computation thus belongs to the computational class EXP_d.

If the attractors are known (from previous calculations for example), and this exponential amount of information is kept, the computation time required to determine the attractor is proportional to $|\log\epsilon|$. Otherwise, it is *exponential* in $|\log\epsilon|$.

If we are to classify one initial point, it requires exponential time to recognize the attractor. If, however, we wish to compute starting from exponentially many initial points for a constant (independent of ϵ) number of isolated attractors, the total time will be exponential in $|\log\epsilon|$, which may translate into behavior ranging from exponential growth to exponential decay per input point.

Note that the process described above does not include verifying that the strange set is indeed an attractor rather than a transient. With the exponential (in $|\log\epsilon|$) amount of information, one may sometimes use the probabilistic verification described for fixed

points. In order to have a polynomial algorithm we resort to nondeterminism.

6.2. Nondeterministic computation

We can do better than exponential time for typical (i.e. isolated) chaotic attractors if the computational problem is rephrased as: "Does a given point $\mathbf{x}$ approach an isolated chaotic attractor Φ?", where the attractor is specified by one of its fixed points, Φ. In this case, we abuse the notation, using Φ both to denote the fixed point and its associated attractor. In addition to Φ another real constant ν is provided so that any periodic orbit that passes in the ν neighborhood of Φ is on the attractor as well [1]. Now the problem is defined as follows:

Given an initial point with precision ϵ decide whether this point is attracted to the ϵ vicinity of the attractor Φ.

We next show that this problem is in the nondeterministic class NP_d.

Let us recall the definition of the class NP from classical computation. Assume for example that one has to decide if a natural number n is composite. This problem is hard. However, if the rules are changed, so that in addition to n you are also given two natural numbers n_1 and n_2, with the promise that n is composite if and only if the multiplication $n_1 n_2$ is equal to n. In this setup we call n_1 and n_2 the *certificate* of the "yes" answer to the question whether n is composite. The authentication of the certificate is all that is required in this new framework. Here the authentication process is the multiplication $n_1 n_2$ and comparing it to n: according to the rules, $n \neq n_1 n_2$ implies that the answer is "no". It is not clear that if a decision problem can be authenticated efficiently for any given certificate, it can also be solved efficiently. This is one of the most fundamental open questions in the realm of theoretical computer science, and is known as the P = NP question.

Here we consider the class NP_d rather than NP. The difference is that the authentication is executed with the dynamical system, rather than with a Turing machine.

Assume one is given an initial point $\mathbf{x}$. There is a certificate point $\mathbf{y}$ so that $\mathbf{y}$ is in the ν neighborhood of Φ and $\mathbf{y}$ belongs to a periodic orbit of length $O(|\log\epsilon|)$. These two properties are easy to authenticate, and hence to deduce that $\mathbf{y}$ is on the attractor. Furthermore, we maintain that for this initial point $\mathbf{x}$ the periodic orbit of the point $\mathbf{y}$ is going to pass in the ϵ vicinity of the trajectory of $\mathbf{x}$ in time $O(\log\epsilon)$, provided that $\mathbf{x}$ approaches the attractor associated with the point Φ. If such a point $\mathbf{y}$ is provided, the affirmative answer can be indeed computed polynomially.

The following lemma is required:

Lemma 2. There is a constant ν such that for any initial point $\mathbf{x}$ in the basin of attraction of an isolated chaotic attractor, that is denoted by one of its fixed points Φ, there is a point $\mathbf{y}_x$ so that:

(i) $\mathbf{y}_x$ is in the ν-vicinity of the fixed point Φ.
(ii) $\mathbf{y}_x$ is on a periodic orbit of length $O(|\log\epsilon|)$.
(iii) The periodic orbit of (ii) passes in the ϵ vicinity of the trajectory starting from $\mathbf{x}$ up to $O(|\log\epsilon|)$ many steps (or the corresponding time for the continuous system) on both the trajectories.

The correctness of this lemma will validate the existence of a certificate $\mathbf{y}$ with the properties described above.

We next justify the lemma: The measure of the ν neighborhood $\mathcal{A}$ around Φ is $\mu(\nu) \sim \nu^{\alpha}$ for a constant α. We denote by $\mathcal{B}$ the set of points on the attractor which are in the ϵ vicinity of the trajectory starting at $\mathbf{x}$. The measure of this set is $\mu(\mathcal{B}) \sim \epsilon^{\beta}$. The measure of the set of orbits that start in $\mathcal{A}$ and visit $\mathcal{B}$ after n iterations is thus

$$\mu(T^n(\mathcal{A}) \cap \mathcal{B});$$

because of mixing [4,6], in the limit $n \to \infty$ it approaches $\mu(\mathcal{A})\mu(\mathcal{B})$. Therefore for large n we can approximate

$$\mu(T^n(\mathcal{A}) \cap \mathcal{B}) \approx \mu(\mathcal{A})\mu(\mathcal{B}). \tag{6.3}$$

The number of periodic orbits of length n on a chaotic attractor of topological entropy h_T [1] is

$$N_n \sim \frac{e^{h_T n}}{n}. \tag{6.4}$$

We denote by $N_n(\mathcal{A}, \mathcal{B})$ the approximate number of periodic orbits of length n that pass in both sets $\mathcal{A}$ and $\mathcal{B}$. It is estimated by:

$$N_n(\mathcal{A}, \mathcal{B}) = \mu(T^n(\mathcal{A}) \cap \mathcal{B}) N_n. \tag{6.5}$$

We want

$$N_n(\mathcal{A}, \mathcal{B}) \approx r, \tag{6.6}$$

where r is somewhat larger than unity so that to assure the existence of such an orbit. From Eqs. (6.5) and (6.6) we conclude that

$$\mu(T^n(\mathcal{A}) \cap \mathcal{B}) = \frac{r}{N_n}. \tag{6.7}$$

Taking logs of both sides of Eq. (6.7) and substituting Eq. (6.3), we can approximate

$$\alpha \log \nu + \beta \log \epsilon \approx \log r - \log N_n, \tag{6.8}$$

which is equivalent to

$$\alpha \log \nu + \beta \log \epsilon \approx \log r - h_T n + \log n. \tag{6.9}$$

Because asymptotically $\epsilon \ll \nu$, $r \approx 1$, and $n \gg \log n$, we approximate

$$|\log \epsilon| \approx h_T n/\beta, \tag{6.10}$$

or equivalently $n = \mathrm{O}(|\log \epsilon|)$. That is, the period of the trajectory that passes in both $\mathcal{A}$ and $\mathcal{B}$ is linear in the precision required.

We can now compare the NP_d result of approaching an attractor with the parallel case of a regular attractor, where the fixed point is given. In the case of a fixed point verification of its identity is trivial, while for a chaotic attractor it is required, resulting in NP_d computation. For chaotic attractors the extension of verification of stability as performed for regular attractors is still an open question and is left for future study. The complexity of chaotic attractors as manifest by their KS entropy and their multifractality leads us to conjecture that:

Conjecture 3. $\mathrm{P}_d \neq \mathrm{NP}_d$

6.3. Probabilistic computation and intermingling

For chaotic systems there is also the possibility of intermingled – rather than isolated – attractors. In this case in any δ-vicinity of an attractor there are points belonging to the basin of attraction of another attractor. A deterministic computation with a limited precision initial condition may be erroneous; yet, probabilistic computation may be applicable.

For the existence of intermingled attractors it is required that there is a chaotic attractor in an invariant subset of phase space [77,78]. This results in temporal fluctuations in the Lyapunov exponents of the motion transverse to this subspace. Consequently for such an attractor the motion looks repelling for finite intervals of time. In any arbitrary small vicinity of the attractor there are sets of points that are not attracted to it. The measure of these shrinks as the attractor is approached. Invariant subspaces, which are required for such attractors, are usually found in systems with some symmetry, namely when the equations of motion are invariant under some operation. An example of such a system is a particle (of a unit mass) moving in the two-dimensional potential [77,78]:

$$V(x, y) = (1 - x^2)^2 + (x + \bar{x})y^2,$$

with friction and driving. Assigning $\mathbf{r} = (x, y)$, the equation of motion is given by

$$\frac{\mathrm{d}^2\mathbf{r}}{\mathrm{d}t^2} = -\gamma \frac{\mathrm{d}\mathbf{r}}{\mathrm{d}t} - \left(\mathbf{x_0}\frac{\partial}{\partial x} + \mathbf{y_0}\frac{\partial}{\partial y}\right) V(x, y) + f_0 \sin(\omega t)\mathbf{x_0}, \tag{6.11}$$

where γ, $\bar{x}$, f_0 and ω are parameters, while $\mathbf{x_0}$ and $\mathbf{y_0}$ are unit vectors in the x and y directions, respectively. The phase space is five-dimensional with the coordinates x, y, $\mathrm{d}x/\mathrm{d}t$, $\mathrm{d}y/\mathrm{d}t$ and $(\omega t) \bmod 2\pi$. Because of the $y \Leftrightarrow -y$ symmetry of the potential, the motion is invariant with respect to $(y, \mathrm{d}y/\mathrm{d}t) \Leftrightarrow (-y, -\mathrm{d}y/\mathrm{d}t)$. Consequently $(y = 0, \mathrm{d}y/\mathrm{d}t = 0)$ is an invariant subspace. The motion in this subspace satisfies

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = -\gamma \frac{\mathrm{d}x}{\mathrm{d}t} + 4x(1 - x^2) + f_0 \sin(\omega t). \tag{6.12}$$

For appropriate values of the parameters there is a chaotic attractor A for the motion described by (6.12). For general initial conditions with $(y, \mathrm{d}y/\mathrm{d}t) \neq 0$, the flow is either to A or to $|y| = \infty$. The initial conditions leading to these two attractors are *intermingled*: in any

vicinity of an initial condition from which the flow is to the attractor A, points that flow to $|y| = \infty$ are found. Reaching a particular attractor can never be assured. However, some probabilistic arguments may hold.

Let y_0 be the distance of a point from an attractor A. The probability to flow from this point to another attractor is proportional to $|y_0|^{\bar{\eta}}$, where $\bar{\eta}$ is a positive exponent [77,78]. The probability decreases with y_0 but does not vanish for any nonvanishing y_0. This suggests a computation process defined by flowing towards the attractor until y_0 is within the required preassigned probability. There is another type of probabilistic behavior in this system: Let a flow starting from some initial point go to the attractor A. Choose a neighboring point at random within a distance ϵ from the first point. With a probability proportional to ϵ^{ϕ} where ϕ is positive, it will not flow to A. (The exponents $\bar{\eta}$ and ϕ describe different probabilities.) A probabilistic computation can be associated with intermingled attractors either by running a deterministic computation or by adding first a small independent perturbation to smooth up the phase space, and only then perform the calculation. The last is maybe more similar to the classical definition of probabilistic computation.

6.4. Strong undecidability and crisis

For chaotic systems computation may be totally undecidable, and no nondeterminism or stochasticity can help. A strong undecidability takes place when "crisis," namely a sudden change in the structure of the attractor with the change of a parameter p, occurs [1]. The value of p where the change takes place is denoted by p_c.

One type of crisis is boundary crisis, where at $p = p_c$ the attractor touches its basin boundary. A system exhibiting such a crisis is the Ikeda–Hamel–Jones–Maloney map [1,80] given by

$$z_{n+1} = C + Bz_n \exp \mathrm{i}\left[\kappa - \frac{p}{1 + |z_n|^2}\right], \tag{6.13}$$

where $z_n = x_n + \mathrm{i}y_n$. At crisis ($p = p_c$), there exists an ϵ vicinity (for an ϵ arbitrary small) of the attractor that contains a hyper-sphere of radius ϵ_1 ($\epsilon_1 < \epsilon$) that flows to another attractor. For p slightly larger than p_c, the chaotic attractor crosses its basin boundary and therefore disappears and is replaced by a chaotic transient [87]. That is, for a time T the trajectories behave as if they move to the chaotic attractor but then suddenly, they "realize" that this is not an attractor and they move off to another distant attractor. The probability to find a transient of time T is

$$P(T) \sim \mathrm{e}^{-T/\langle T\rangle} \tag{6.14}$$

where

$$\langle T\rangle \sim (p - p_c)^{-\bar{\gamma}}, \tag{6.15}$$

where $\bar{\gamma}$ is a positive exponent. The average transient time diverges at p_c. Consequently, for p slightly larger than p_c, the computation will not converge for a very long time.

Another type of crisis is "attractor merging crisis", where strange attractors merge when a parameter p is varied [1]. Again at p_c the attractors touch and then the computation is undecidable. For p slightly larger than p_c, the orbits will spend long time intervals T in the vicinity of one of the "old" attractors. The distribution again satisfies (6.14). The trajectory jumps intermittently between the "old" attractors and computationally it is undecidable. There are examples where the average time scale is

$$\langle T\rangle \sim \mathrm{e}^{\bar{k}/((p-p_c)^{1/2})}, \tag{6.16}$$

which near p_c is much larger than in (6.15). If the distance between the attractors in the case of the "attractor merging crisis", or the distance between the attractor and its basin boundary, in the case of boundary crisis is smaller than ϵ (which is the precision of the computation) the behavior will be similar to a computationally undecidable one.

7. Discussion

Analog computation is useful in a wide range of engineering applications such as control and robotics, and is important for modeling of biological motor control and neural modeling. It is also of importance for fundamental problems in computer science,

like the validity of the physical Church–Turing thesis [19]. It may also be of use in the design of special purpose analog computers. For these reasons a fundamental theory for computation in *continuous space and time* is required. This paper constitutes the first effort (to our knowledge) to provide a natural interface between dynamical systems and computational models with the emphasis on continuous time update. Our theory formulates problems in computational complexity within the framework of dynamical systems. In this field much is "believed" and "conjectured" (e.g. [1,3,5,6,67]), but relatively little is mathematically proven. Hence, we are unable to provide a mathematical theory but rather an instructive understanding that bridges between realizable dynamical systems and the theory of computation.

The main results of the present work are:

- Definition of computation by dynamical systems.
- Definition of computation time that applies to both discrete and continuous time.
- Complexity analysis of the continuous-time Hopfield network.
- For systems with regular attractors two computation frameworks are provided. In the first one, computation is deterministic and in P_d. In the second it is probabilistic and in Co-RP_d (works only under some restrictions).
- If the attractors are chaotic and isolated, polynomial time is not sufficient to trace an attractor, but exponential deterministic time is needed. However, we can naturally define nondeterministic computation in this case and these systems are in NP_d. We conjectured that $P_d \neq NP_d$.
- If the attractors are chaotic and intermingled (with riddled basins) only probabilistic computation is feasible.
- Chaotic systems which exhibit crisis (or chaotic intermittency) demonstrate natural undecidable computation.

In a forthcoming paper the application of this theory to the problem of linear programming and its implications to optimal performance continuous time algorithms will be discussed [9].

There is some difficulty in comparing the classical digital model with our model. For example, the computation time of a problem is not fixed for all inputs, but rather only for all those inputs that are in the same basin of attraction (resulting in the same response). We can still bound the computation time for all inputs by the slowest computation time, but such an approach will remove some useful information.

Interesting open questions are:

- Can one find a deterministic method to verify attractors without introduction of an additional device that solves linear equations?
- Can one find a better lower bound on the computational complexity of verifying an isolated strange attractor?
- What is the complexity of deciding a point close to the boundaries? Or equivalently: we associated the computational class NP_d with the task of recognizing an isolated strange attractor. What is the extra information required to recognize a boundary and in particular its "sides"?
- For systems with intermingled basins of attraction deterministic computation will never converge. The corresponding problems should be decidable by probabilistic computation. What notions of classical complexity theory are applicable in such a case?
- Can one formalize non-uniform classes by our model? In particular, could we realize the super-Turing model suggested in [22]?
- Can one find examples of problems that are non-decidable by other models of computation but are decidable by our model? Are there problems that are more efficiently computed in our model than in the classical theory of computation? Are there such natural questions?
- Our dynamical systems are special purpose computers. Can one define a general purpose analog computer within the above formalism?

Acknowledgements

It is our great pleasure to thank C. Grebogi, B. Hunt, E. Ott, I. Procaccia and J.A. Yorke for informative and stimulating discussions and communications. We thank P. Orponen for initiating our general interest and suggesting our basic approach to the problem.

We thank in particular Asa Ben-Hur who discussed with us all aspects of this work, argued with us on each point until it was justified, clarified, and carefully written. The work was supported in part by the US–Israel Binational Science Foundation (BSF), by the National Science Foundation under grant no. PHY94-07194, by the Fund for Promotion of Research at the Technion and by the Minerva Center for Nonlinear Physics of Complex Systems. It is our great pleasure to thank R.E. Prange and J.A. Yorke for the hospitality at the University of Maryland and the Newton Institute at the University of Cambridge where some of this work was done.

References

[1] E. Ott, Chaos in Dynamical Systems, Cambridge University Press, Cambridge, 1993.
[2] J. Guckenheimer, P. Holmes, Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields, Springer, New York, 1983.
[3] M. Tabor, Chaos and Integrability in Nonlinear Dynamics, Wiley, New York, 1989.
[4] V.I. Arnold, A. Avez, Ergodic Problems of Classical Mechanics, Benjamin, New York, 1968.
[5] P. Bergé, Y. Pomeau, C. Vidal, Order within Chaos, Herman, Paris, 1984.
[6] H.G. Schuster, Deterministic Chaos, Physik-Verlag, Weinheim, 1984.
[7] C. Papadimitriou, Computational Complexity, Addison-Wesley, Reading, MA, 1995.
[8] H. Karloff, Linear Programming, Birkhauser, Basel, 1991.
[9] H.T. Siegelmann, A. Ben-Hur, S. Fishman, A theory of complexity for continuous time dynamics, Theoretical Computer Science, submitted for publication.
[10] S.W. McDonald, C. Grebogi, E. Ott, J.A. Yorke, Fractal basin boundaries, Physica D 17 (1985) 125–153.
[11] G. H. Hsu, E. Ott, C. Grebogi, Strange saddles and the dimensions of their invariant manifolds, Phys. Lett. A 127 (4) (1988) 199–204.
[12] U. Feudel, C. Grebogi, B. Hunt, J.A. Yorke, A map with more than 100 coexisting low period periodic attractors, Phys. Rev. E 54 (1) (1996) 71–81.
[13] R.W. Brockett, Dynamical systems that sort lists, diagonalize matrices and solve linear programming problems, Linear Algebra Applications 146 (1991) 79–91.
[14] U. Helmke, J.B. Moore, Optimization and Dynamical Systems, Springer, London, 1994.
[15] J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, Proc. Nat. Acad. Sci. USA, 79 (1982) 2554.
[16] J. Hertz, A. Krogh, R. Palmer, Introduction to the Theory of Neural Computation, Addison-Wesley, Redwood City, 1991.
[17] M.S. Branicky, Analog computation with continuous ODEs, in: Proceedings of the IEEE Workshop on Physics and Computation, Dallas, TX, 1994, pp. 265–274.
[18] R. Penrose, The Emperor's New Mind, Oxford University Press, Oxford, 1989.
[19] R. Feynman, Quantum mechanical computers, Foundations Phys. 16 (6) (1986) 507–531; originally appeared in: Optics News, Feb. 1985.
[20] R. Penrose, Shadows of the Mind, Oxford University Press, Oxford, 1994.
[21] H.T. Siegelmann, E.D. Sontag, Analog computation via neural networks, Theoret. Compute. Sci. 131 (1994) 331–360.
[22] H.T. Siegelmann, Computation beyond the turing limit, Science, 268 (5210) (1995) 545–548.
[23] L. Blum, M. Shub, S. Smale, On a theory of computation and complexity over the real numbers: NP completeness, recursive functions, and universal machines, Bull. Am. Math. Soc. 21 (1989) 1–46.
[24] H.T. Siegelmann, E.D. Sontag, On computational power of neural networks, J. Comput. Syst. Sci. 50 (1) (1995) 132–150.
[25] J.L. Balcázar, R. Gavaldà, H.T. Siegelmann, E.D. Sontag, Some structural complexity aspects of neural computation, in: IEEE Structure in Complexity Theory Conference, San Diego, CA, May 1993, pp. 253–265.
[26] P. Koiran, A weak version of the Blum, Shub & Smale model, in: Proceedings of the 34th FOCS Symposium, (1993), pp. 486–495.
[27] R. Gavalda, H.T. Siegelmann, Discontinuities in recurrent neural networks, Neural Comput. to appear.
[28] C. Moore, Unpredictability and undecidability in dynamical systems, Phys. Rev. Lett. 64 (1990) 2354–2357.
[29] C. Moore, Recursion theory on the reals and continuous-time computation, Theoret. Comput. Sci. 162 (1996) 23–44.
[30] C. Moore, Dynamical recognizers: Real-time language recognition by analog computers, Theoret. Compute. Sci.
[31] A.R. Smith III, Simple computation-universal cellular spaces, J. ACM 18 (1971) 339.
[32] P. Orponnen, M. Matamala, Universal computation by finite two-dimensional coupled map lattices, in: Proceedings of the Physics and Computation, Boston, MA, November 1996, pp. 243–247.
[33] T. Toffoli, N. Margolus, Cellular Automata Machines, MIT Press, Cambridge, MA, 1987.
[34] S. Omohundro, Modeling cellular automata with partial differential equations, Physica D 10 (1984) 128–134.
[35] L. Tavernini, Differential automata and their discrete simulators, Nonlinear Anal. Theor. Methods Appl. 11 (1987) 665–683.
[36] A. Back, J. Guckenheimer, M. Myers, A dynamical simulation facility for hybrid systems, in: R.L. Grossman, A. Nerode, A.P. Ravn, H. Rischel (Eds.), Hybrid Systems Lecture notes in Computer Science vol. 736, Springer, New York, 1993, pp. 255–267.
[37] A. Nerode, W. Kohn, Models for hybrid systems: Automata, topologies, controllability, observability, in: A.P.

Ravn, R.L. Grossman, A. Nerode, H. Rischel (Eds.), Hybrid Systems, Lecture notes in Computer Science vol. 736, Springer, New York, 1993, pp. 317–356.
[38] R.W. Brockett, Smooth dynamical systems which realize arithmetical and logical operations, in: H. Nijmeijer, L.M. Schumacher (Eds.), Three Decades of Mathematical Systems Theory, Springer, Berlin, 1989, pp. 19–30.
[39] R.W. Brockett, Dynamical systems that sort lists, Linear Algebra Appl. 146 (1991) 79–91.
[40] R.W. Brockett, Hybrid models for motion control systems, in: H.L. Trentelman, J.C. Willems (Eds.), Essays in Control: Perspectives in the Theory and its Applications, Birkhauser, Boston, 1993, pp. 29–53.
[41] M.S. Branicky, Topology of hybrid systems, in: Proceedings of the 32nd IEEE Conference on Decision and Control, San Antonio, TX, 1993, pp. 2309–2314.
[42] M.S. Branicky, Universal computation and other capabilities of hybrid and continuous dynamical systems, Theoret. Comput. Science 138 (1) (1995).
[43] P.J. Antsaklis, M.D. Lemmon, J.A. Stiver, Hybrid systems modeling and event identification, Technical Report ISIS-93-002, University of Notre Dame, 1993.
[44] E. Asarin, O. Maler, On some relations between dynamical systems and transition systems, Proc. ICALP, Lecture Notes in Comp. Sci. 820 (1994) 59–72.
[45] R.L. Grossman, A. Nerode, A.P. Ravn, H. Rischel (Eds.), Hybrid Systems, Lecture notes in Computer Science, vol. 736, Springer, New York, 1993.
[46] Z. Artstein, Examples of stabilization with hybrid feedback, in: T.A. Henzinger Alur, E.D. Sontag (Eds.), Hybrid Systems III: Verification and Control, Lecture notes in Computer Science, vol. 1066, Springer, Berlin, 1996.
[47] L. Faybusovich, Hamiltonian structure of dynamical systems which solve linear programming problems, Physica D 53 (1991) 217–232.
[48] C.E. Shannon, Mathematical theory of the differential analyzer, J. Math. Phys. Massachusetts Instit. Technol. 20 (1941) 337–354.
[49] M.B. Pour-El, Abstract computability and its relation to the general purpose analog computer (some connections between logic, differential equations and analog computers), Trans. Am. Math. Soc. 199 (1974) 1–29.
[50] L.A. Rubel, A universal differential equation, Bull. Am. Math. Soc. 4(3) (1981) 345–349.
[51] L.A. Rubel, Digital simulation of analog computation and Church's thesis, J. Symbolic Logic 54 (3) (1989) 1011–1017.
[52] L.A. Rubel, The extended analog computer, Adv. Appl. Math. 14 (1993) 39–50.
[53] M.B. Pour-El, J.I. Richards, Computability in Analysis and Physics, Springer, New York, 1988.
[54] J.J. Hopfield, Neurons with graded responses have collective computational properties like those of two-state neurons, in: Proc. Nat. Acad. Sci. USA 81 (1984) 3088–3092.
[55] J.J. Hopfield, D.W. Tank, Neural computation of decisions in optimization problems, Biol. Cybernet. 52 (1985) 141–152.
[56] S.-I. Amari, Characteristics of randomly connected threshold-element networks and network systems, Proc. IEEE 59 (1971) 35–47.
[57] J.A. Anderson, A simple neural network generating an interactive memory, Math. Biosci. 14 (1972) 197–220.
[58] J.A. Anderson, A memory storage model utilizing spatial correlation functions, Kybernetik 5 (1968) 113–119.
[59] T. Kohonen, Correlation matrix memories, IEEE Trans. Comput. 21 (1972) 353–359.
[60] K. Nakano, Associatron – a model of associative memory, IEEE Trans. Man, Systems Cybernet. 2 (1972) 380–388.
[61] K. Nakano, J.-I. Nagumo, Information processing using a model of associative memory, in: Proceedings of the Second International Conference on Artificial Intelligence, University Microfilms, High Wycomb, UK, 1971. pp. 101–106.
[62] D.J. Willshaw, O.P. Buneman, H.C. Longuet-Higgins, Non-holographic associative memory, Nature 222 (1969) 960–962.
[63] W.A. Little, The existence of persistent states in the brain, Math. Biosci. 19 (1974) 101–120.
[64] W.A. Little, G.L. Shaw, Analytic study of the memory storage capacity of a neural network, Math. Biosci. 39 (1978) 281–290.
[65] A.F. Atiya, Y.S. Abu-Mostafa, An analog feedback associative memory, IEEE Trans. Neural Networks 4 (1993) 117–126.
[66] M. Morita, Associative memory with nonmonotonic dynamics, Neural Networks 6 (1993) 115–126.
[67] B.R. Hunt, T. Sauer, J.A. Yorke, Prevalence: A translational-invariant "almost every" on infinite dimensional spaces, Bull. Am. Math. Soc. 27 (2) (1992) 217–238.
[68] B.R. Hunt, T. Sauer, J.A. Yorke, Prevalence: An addendum, Bull. Am. Math. Soc. 28 (2) (1993) 306–307.
[69] B.R. Hunt, J.A. Yorke, private communication.
[70] P. Ashwin, J. Buescu, I. Stewart, From attractor to chaotic saddle: A tale of transverse instability, Nonlinearity 9 (1996) 703–737.
[71] D.A. Ruelle, F. Takens, On the nature of turbulence, Commun. Math. Phys. 20 (1971) 167–192.
[72] S. Newhouse, D.A. Ruelle, F. Takens, Occurrence of strange axiom A attractors near quasi periodic flows on T^m, $m \geq 3$, Commun. Math. Phys. 64 (1978) 35–40.
[73] M.J. Feigenbaum, Quantitative universality for a class of nonlinear transformations, J. Stat. Phys. 19 (1978) 25–52.
[74] J.L. Balcázar, J. Díaz, J. Gabarró, Structural Complexity, EATCS Monographs, vols. I and II, Springer, Berlin, 1988–1990; Second ed. for volume I in 1995.
[75] G.H. Golub, C.F. Van Loan, Matrix Computations, The Johns Hopkins University Press, Baltimore, 1989.
[76] M. Hirsch, S. Smale, Differential Equations, Dynamical Systems and Linear Algebra, Academic Press, New York, 1974.
[77] E. Ott, J.C. Sommerer, J.C. Alexander, I. Kan, J.A. Yorke, Scaling behavior of chaotic systems with riddled basins, Phys. Rev. Lett. 71 (25) (1993) 4134–4137.

[78] E. Ott, J.C. Alexander, I. Kan, J.C. Sommerer, J.A. Yorke, The transition to chaotic attractors with riddled basins, Physica D 76 (1994) 384–410.
[79] J.C. Alexander, J.A. Yorke, Z. You, I. Kan, Riddle basins, Int. J. Bifurc. & Chaos 2 (4) (1992) 795–813.
[80] C. Grebogi, E. Ott, F. Romeiras, J.A. Yorke, Critical exponents for crisis-induced intermittency, Phys. Rev. A 36 (1987) 5365–5380.
[81] L. Poon, C. Grebogi, Controlling complexity, Phys. Rev. Lett. 75 (1995) 4023–4026.
[82] V.I. Arnold, Small denominators and problems of stability of motion in classical and celestial mechanics, Russian Math. Surveys 18 (6) (1963) 85.
[83] J. Moser, Stable and Random Motion in Dynamical Systems, Princeton University Press, Princeton, NJ, 1973.
[84] G.D. Birkhoff, On the periodic motion of dynamical systems, Acta Math. 50 (1927) 359.
[85] H.E. Nusse, J.A. Yorke, The structure of basins of attraction and their trapping regions, Ergodic Theory Dynamical Systems 17 (1997) 463–482.
[86] T.C. Halsey, M.H. Jensen, L.P. Kadanoff, I. Procaccia, B.I. Shraiman, Fractal measures and their singularities: The characterization of strange sets, Phys. Rev. A 33 (2) (1986) 1141–1151.
[87] J.C. Sommerer, Determination of crisis parameter values by direct observation of manifold tangencies, Int. J. Bifurc. & Chaos 2 (2) (1992) 383–396.