Analog Computation with Continuous ODEs*

Michael S. Branicky†
Dept. of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract

We demonstrate simple, low-dimensional systems of ODEs that can simulate arbitrary finite automata, push-down automata, and Turing machines. We conclude that there are systems of ODEs in $\mathbb{R}^3$ with continuous vector fields possessing the power of universal computation. Further, such computations can be made robust to small errors in coding of the input or measurement of the output. As such, they represent physically realizable computation.

We make precise what we mean by "simulation" of digital machines by continuous dynamical systems. We also discuss elements that a more comprehensive ODE-based model of analog computation should contain. The "axioms" of such a model are based on considerations from physics.

1 Introduction

Lately, there has been a renewed general interest in analog computation (e.g., neural networks). Hardware implementations of analog computation schemes are also making a comeback due to advances in technology [23]. Theoretically, there has been increased interest in continuous realizations of discrete computations ([10, 11, 12, 36] and the references therein). Another recent focus has been on complexity theory of computational machines based on the reals (see [3, 31, 34] and their references). It remains to see whether these will threaten the currently entrenched notion of computational complexity (as in, e.g., [17, 20]).

In this paper, we will focus on the computational power of continuous-time dynamical systems whose evolution is governed by ordinary differential equations (ODEs). We demonstrate simple, low-dimensional systems of ODEs that can simulate arbitrary finite automata (FA), push-down automata (PDA), and Turing machines (TMs). We conclude that there are systems of ODEs in $\mathbb{R}^3$ with continuous vector fields (continuous ODEs) possessing the power of universal computation. Related in spirit is the work of [29], in which partial differential equations mimicking cellular automata are given.

It is well-known that discrete-time dynamical systems possess the power of universal computation (see, e.g., [13, 25, 33]). It is also easy to construct systems of *discontinuous* ODEs possessing the ability to simulate arbitrary TMs [1, 7]. Finally, it is a relatively trivial observation that there are systems of continuous ODEs which possess the power of universal computation—just write down the differential equations governing your favorite personal computer or workstation. However, this requires a system of ODEs with a (potentially) infinite number of states.

Our work here (which follows [5, 7]) shows that it is not necessary to have a potentially infinite number of states nor to resort to discontinuous vector fields. Further, such computations can be made robust to small errors in coding of the input or measurement of the output. As such, they represent more "physically realizable" computation.

The paper is organized as follows. First, we make precise what we mean by "simulation" of digital machines by continuous dynamical systems. The notions are somewhat technical, but intuitive. Next, we present our simulation results, demonstrating the computational power of simple, low-dimensional systems of ODEs. While stated in terms of our definitions of simulation, these results are quite intuitive and continue to hold under alternate, reasonable definitions of simulation. Finally, we discuss elements that a more comprehensive ODE-based model of analog computation should contain. The "axioms" of such a model are based on considerations from physics [16] and previous work on analog computation [9, 35].

*Work supported by the Army Research Office and the Center for Intelligent Control Systems under grants DAAL03-92-G-0164 and DAAL03-92-G-0115.

†E-mail: branicky@mit.edu.

0-8186-6715-X/94 $4.00

2 Notions of Simulation

2.1 Preliminaries

First, we will develop a little notation. For more information, the reader is referred to [18, 30]. Throughout, let $\mathbb{R}$, $\mathbb{R}_+$, $\mathbb{Z}$, and $\mathbb{Z}_+$ denote the reals, nonnegative reals, integers, and nonnegative integers, resp.

Definition 1 *A* continuous (resp. discrete) dynamical system *in the metric space X is a function $q = f(p,t)$ which assigns to each point p of the space X and to each $t \in \mathbb{R}^+$* (resp. $\mathbb{Z}^+$) *a definite point $q \in X$ and possesses the following three properties:*

- *Initial condition: $f(p,0) = p$ for any point $p \in X$.*
- *Continuity on both arguments.*
- *Semi-group property:*

$$f(f(p,t_1),t_2) = f(p,t_1+t_2)$$

for any point $p \in X$ and any t_1 and t_2 in $\mathbb{R}^+$ (resp. $\mathbb{Z}^+$).

Technically, in the mathematical literature of dynamical systems, such systems are referred to as semi-dynamical systems, with the term dynamical system reserved for those which the semigroups $\mathbb{R}^+$, $\mathbb{Z}^+$ above may be replaced by the groups $\mathbb{R}$, $\mathbb{Z}$. However, the more "popular" notion of dynamical system in math and engineering requires only the semigroup property [14, 22]. We use the word *reversible* to distinguish the group from semigroup case.

In this paper the continuous dynamical systems dealt with are ODEs (with unique, global solutions). Throughout, we will use the shorthand *continuous* (resp. *Lipschitz*) *ODEs* to denote ODEs with continuous (resp. Lipschitz continuous) vector fields, that is, ODEs whose right-hand sides are continuous (resp. Lipschitz continuous).

We use the shorthand $[X,S,f]$ to denote the dynamical system f defined on X over the semigroup S. The set $f(p,S) = \{f(p,i) : i \in S\}$ is called the *(positive) orbit* or *trajectory of the point p*. A *fixed point* of $[X,S,f]$ is a point p such that $f(p,s) = p$ for all $s \in S$. A set $A \subset X$ is *invariant with respect to f*, or simply *invariant*, if $f(A,s) \subset A$ for all $s \in S$. Two dynamical systems $[X,S,f]$, $[Y,S,F]$ are said to be *equivalent* (also *topologically equivalent* or *isomorphic*) if there exists a homeomorphism $\psi : X \to Y$ such that

$$\psi(f(p,s)) = F(\psi(p),s)$$

for all $p \in X$ and $s \in S$.

If the mapping ψ is only continuous, then $[X,S,f]$ is said to be *homomorphic* to $[Y,S,F]$. The simplest example of a homomorphism is the situation when the space Y consists of one fixed point. Homomorphisms preserve trajectories, segments of trajectories, fixed points, periodic points, and invariant sets.

Note that these dynamical systems are *autonomous*, that is, their transition maps do not depend explicitly on absolute time. (The time-varying case is subsumed by adding another state variable for time.) Thus, we will sometimes abuse notation below by saying that the function f, meaning $f(\cdot,1)$, is a "discrete dynamical system defined on X."

2.2 Definitions of Simulation

In dynamical systems, simulation is captured by the notions of topological equivalence and homomorphism [14, 18, 30]. One can extend these notions to systems with inputs and outputs by also allowing memoryless, continuous encoding of inputs, outputs, and initial conditions.

In computer science, simulation is based on the notion of "machines that perform the same computation." This can be made more precise, but is not reviewed here [4, 24].

Other notions of simulation (for discrete dynamical systems) appear in [21]. All these notions, however, are "homogeneous," comparing continuous systems with continuous ones or discrete with discrete. One that encompasses simulation of a discrete dynamical system by a continuous dynamical system is required here.

One notion that associates discrete and continuous dynamical systems is global section [30]. The set $S_X \subset X$ is a *global section* of the continuous dynamical system $[X,\mathbb{R}^+,f]$ if there exists a $t_0 \in \mathbb{R}^+$ such that

$$S_X = \{f(P,kt_0) \mid k \in \mathbb{Z}^+\},$$

where P is a set containing precisely one point from each of the trajectories $f(p,\mathbb{R}^+)$, $p \in X$.

We will develop several definitions of simulation of a discrete dynamical system by a continuous one below. These will be aided by some preliminary definitions.

Definition 2 *A sequence $t_0,t_1,t_2,\ldots$ of reals is* realizable (resp. equi-realizable) *if*

- $t_0 < t_1 < t_2 < \cdots$,
- $\inf_i(t_{i+1}-t_i) > 0$ (resp. $t_{i+1}-t_i = \Delta > 0$).

A sequence will be called lively *if* $\sup_i(t_{i+1}-t_i) < \infty$. *A sequence that is both realizable and lively will be called* effective.

A sequence of intervals $[\tau_0, \tau_0']$, $[\tau_1, \tau_1']$, $[\tau_2, \tau_2']$, ... *is* realizable (resp. lively, effective) *if* $\{\tau_i, \tau_i'\}$ *is realizable* (resp. *lively, effective*).

Note that an equi-realizable sequence is effective. The definition of realizable appeared in the work of [28].

Using the definition of global section for guidance, we define

Definition 3 (S-simulation) *A continuous dynamical system* $[X, \mathbb{R}^+, f]$ simulates via section *or* S-simulates *a discrete dynamical system* $[Y, \mathbb{Z}^+, F]$ *if there exist a continuous surjective partial function* $\psi : X \to Y$ *and an equi-realizable sequence* $\{t_k\}$ *such that for all* $x \in \psi^{-1}(Y)$ *and all* $k \in \mathbb{Z}^+$

$$\psi(f(x, t_k)) = F(\psi(x), k).$$

Note that surjectivity implies that for each $y \in Y$ there exists $x \in \psi^{-1}(y)$ such that the equation holds. Here, continuous partial function means the map from $\psi^{-1}(Y)$ (as a subspace of X) to Y is continuous.

Intuitively, the set $V \equiv \psi^{-1}(Y)$ may be thought of as the set of "valid" states; the set $X \backslash V$ as the "don't care" states. In dynamical systems, V may be a Poincaré section; $X \backslash V$ the set of points for which the corresponding Poincaré map is not defined [18, 19]. In computer science and electrical engineering, V may be the set of circuit voltages corresponding to a logical 0 or 1; $X \backslash V$ the voltages for which the logical output is not defined.

S-simulation is a strong notion of simulation. For instance, compare it with topological equivalence. Typically, though, the homogeneous notions of simulation do not expect time to be parameterized the same (up to a constant) for both systems. For example, a universal Turing machine, U, may take several steps to simulate a single step of any given Turing machine, M. Moreover, the number of such U steps to simulate an M step may change from M step to M step. Some of the notions of simulation defined in [21] also allow this generality. Further, the definition of topological equivalence of vector fields (different than for dynamical systems, see [18]) is such that parameterization of time need not be preserved. Thus, following the definitions in [21] one formulates

Definition 4 (P-simulation) *A continuous dynamical system* $[X, \mathbb{R}^+, f]$ simulates via points *or* P-simulates *a discrete dynamical system* $[Y, \mathbb{Z}^+, F]$ *if there exists a continuous surjective partial function* $\psi : X \to Y$ *such that for all* $x \in \psi^{-1}(Y)$ *there is realizable sequence* $\{t_k\}$ *such that*

$$\psi(f(x, t_k)) = F(\psi(x), k).$$

One readily checks that S-simulation implies P-simulation. This is a weak notion. For instance, consider the case where Y is finite, $|Y| = N$. Suppose $[X, \mathbb{R}^+, f]$ has a point p such that $|f(p, \mathbb{R}^+)| \geq N$ and $p = f(p, t_0)$ for some $t_0 > 0$. That is, the orbit at point p is periodic and contains more than N points. Clearly, one may associate N distinct points in $f(p, \mathbb{R}^+)$ with the points in Y, so that $[X, \mathbb{R}^+, f]$ P-simulates $[Y, \mathbb{Z}^+, F]$. This weakness persists even if Y is infinite. For example, the simple harmonic oscillator defined on the unit circle, $X = S^1$:

$$\begin{aligned} \dot{x_1} &= x_2, \\ \dot{x_2} &= -x_1, \end{aligned}$$

along with $\psi(x) = x_1$ P-simulates every dynamical system of the form $[[-1, 1], \mathbb{Z}^+, F]$. These arguments also show the weakness of some of the definitions in [21]. Finally, this same example shows P-simulation does not imply S-simulation: the harmonic oscillator above cannot S-simulate any $[[-1, 1], \mathbb{Z}^+, F]$ for which 0 is a fixed point and 1 is not a fixed point.

Thus, P-simulation need not correspond to an intuitive notion of simulation. The reason is that one wants, roughly, homeomorphisms from orbits to orbits, not from points to points. As mentioned in Section 2.1, this is achieved with continuous dynamical systems. However, this is not possible with nontrivial nonhomogeneous systems since a discrete orbit with more than one point is a (countable) disconnected set and any non-constant continuous orbit is an (uncountable) connected set. Thus, there exist homeomorphisms between discrete and continuous orbits only when both are constant.

If X is connected and Y is a discrete topological space, this situation exists even with points, i.e., the only continuous functions from X to Y are constant functions [27]. One way to remedy this is simply to place topologies on X and Y other than their usual topologies, so that continuous maps are possible [6, 28]. There are several ways to accomplish this. One approach is to use so-called small topologies on X. Another is to append a single element $\{\perp\}$ to Y, which stands for "don't care" or "continue," and topologize $Y' = Y \cup \{\perp\}$. For more information and other approaches see [6, 28].

Here—and with a view towards simulating systems defined on discrete topological spaces—we strengthen the definition of P-simulation in two ways. First, we require that the "simulated state" be valid on some neighborhood and for at least some minimal time period. Physically, this allows one to use "imprecise sampling" to obtain discrete data, providing a robustness

that is lacking in the definition of P-simulation. Second, we require that the "readout times" are exactly those for which $x(t) \in \psi^{-1}(Y)$.

Definition 5 (I-simulation) *A continuous dynamical system* $[X, \mathbb{R}^+, f]$ simulates via intervals *or* I-simulates *a discrete dynamical system* $[Y, \mathbb{Z}^+, F]$ *if there exist a continuous surjective partial function* $\psi : X \to Y$ *and* $\epsilon > 0$ *such that* $V \equiv \psi^{-1}(Y)$ *is open and for all* $x \in V$ *the set* $T = \{t \in \mathbb{R}^+ \mid f(x,t) \in V\}$ *is a realizable sequence of intervals* (τ_k, τ_k') *with*

$$\psi(f(x,t_k)) = F(\psi(x),k),$$

for all $t_k \in (\tau_k, \tau_k')$.

Clearly I-simulation implies P-simulation. However, S-simulation and I-simulation are independent notions.

The extra requirement that $\psi^{-1}(Y)$ be open implies that the inverse images of open sets in Y are open in X (and not just in $\psi^{-1}(Y)$ as before). This is probably too strong a requirement in the case of a general topological space Y. However, in the case of Y a discrete topological space, it has the desirable effect that $\psi^{-1}(y)$ is open for all $y \in Y$.

One might also have required an output map that is zero (or any distinguished output value) on the complement of T and non-zero otherwise. This amounts to, in the case of a universal Turing machine simulating a machine M, the existence of a distinguished state meaning "a step of the simulated machine is not yet completed." Here, it is related to the appending of a symbol $\{\perp\}$ to Y as above and extending $\psi : X \to Y' = Y \cup \{\perp\}$ by defining $\psi(x) = \{\perp\}$ if $x \in X \backslash \psi^{-1}(Y)$ [1, 6, 28]. In this case, the requirements on ψ may be replaced by requiring ψ to be continuous from X to Y' (in a suitable topology) after extension. Finally, if X is a metric space one could introduce a "robust" version of I-simulation by requiring the inverse image of $y \in Y$ to contain a ball with at least some minimum diameter.

Below, "simulation" is a generic term, meaning I-simulation, S-simulation, or both. SI-simulation denotes S-simulation and I-simulation. If a machine is equivalent, or simulates one that is equivalent, to a universal Turing machine, one says it has *the power of universal computation.*

3 Simulation of FA, PDA, and TMs

In this section we concentrate on general simulation results and the simulation capabilities of continuous ODEs.

We first construct low-dimensional discrete dynamical systems in $\mathbb{Z}^n$ that are equivalent to finite automata (FA), pushdown automata (PDA), and Turing machines (TMs). Later, we give some general results for continuous ODEs in $\mathbb{R}^{2n+1}$ simulating discrete dynamical systems in $\mathbb{Z}^n$. Combining allows us to conclude simulation of arbitrary FA, PDA, and TMs. By simulating a universal TM, one obtains continuous ODEs with the power of universal computation.

3.1 Discrete Dynamical Systems Equivalent to FA, PDA, and TMs

We start by showing that every TM is equivalent to a discrete dynamical system in $\mathbb{Z}^2$ and then consider systems equivalent to PDA and FA. Later, we refine these results to discrete dynamical systems in $\mathbb{Z}$ equivalent to TMs, PDA, and FA.

The FA, PDA, and TMs considered here are deterministic. Thus their transition functions naturally give rise to discrete dynamical systems. These are defined on state spaces of input strings and states; input strings, states, and stacks; and states, tape head positions, and tapes, respectively.

Here, the states, input strings, stacks, and tape configurations of automata and Turing machines are taken in the discrete topology; $\mathbb{Z}^n$ as a topological or normed space is considered as a subspace of $\mathbb{R}^n$ (in particular, it has the discrete topology).

An *inputless* FA (resp. PDA) is one whose input alphabet is empty, i.e., one whose transition function depends solely on its state (resp. state and top stack symbol). See [20] for precise definitions of FA, PDA, and TM.

Proposition 6 *1. Every TM is equivalent to a discrete dynamical system in* $\mathbb{Z}^2$.

2. There is a discrete dynamical system in $\mathbb{Z}^2$ *with the power of universal computation.*

3. Every FA and inputless PDA is equivalent to a discrete dynamical system in $\mathbb{Z}$. *Every PDA is equivalent to a discrete dynamical system in* $\mathbb{Z}^2$.

Proof

1. Assume that the tape alphabet is $\Gamma = \{\gamma_0, \gamma_1, \ldots, \gamma_{m-2}\}$, $m \geq 2$, with γ_0 the blank symbol; and that the set of states is $Q = \{q_0, \ldots, q_{n-1}\}$, $n \geq 1$. Define $p = \max\{m, n\}$.

   As is customary, the one-sided infinite tape is stored in two stacks, with the state stored on the top of the right stack. The coding used is p-ary.

In particular, suppose the TM is in configuration C, with tape

$$\mathcal{T} = \gamma_{i_1}, \ldots, \gamma_{i_{N-1}}, \gamma_{i_N}, \gamma_{i_{N+1}}, \ldots,$$

head positioned at cell N, and internal state q_j. Encode the configuration C in the integers

$$T_L = f_1(C) = \sum_{k=0}^{N-1} p^k i_{N-k} + p^N(m-1),$$

$$T_R = f_2(C) = j + \sum_{k=1}^{\infty} p^k i_{N+k}.$$

The second sum is finite since only finitely many tape cells are non-blank. The integer $(m-1)$ is an end-of-tape marker. The TM is assumed to halt on moving off the left of the tape, so that $(m-1, T_R)$ in $\mathbb{Z}^2$ is a fixed point for all valid T_R. On all other valid configurations, C, define transition function G in $\mathbb{Z}^2$ by

$$G(f_1(C), f_2(C)) = (f_1(C'), f_2(C')),$$

where C' is the configuration resulting when the next move function of the TM is applied to configuration C.

2. Use part 1 with any universal TM.

3. The inputless cases are immediate from part 1. For the cases with input, note that we encode the input string in an integer like the left part of the tape of a TM above, the results following.

□

Note that one can perform the above encodings of TMs, FA, and PDA with $[0, p]$ replacing $\mathbb{Z}$. Merely replace p by p^{-1} in the formulas. The important thing added is compactness, and other encodings, e.g., with $[0, 1]$ replacing $\mathbb{Z}$, follow similarly. There is a problem using these encodings since two distinct tapes may have the same encoding, e.g., $3, 2, 0^\omega$ and $3, 1^\omega$. One can get around this by "separating" each tape encoding by replacing p with $2p$ and using $2i$ for the ith symbol. Namely, the tape of length N, $\mathcal{T} = \gamma_{i_1}, \ldots, \gamma_{i_N}$, is encoded as $\sum_{k=0}^{N} (2p)^{-k} 2i_k$. Such "Cantor encodings" were used in [31]. We still do not use such encodings here, however, since later we want to ensure a minimum distance between any two tape encodings.

Finally, a wholly different approach is to use encodings inspired by those in [13]. Suppose we are given an arbitrary TM, T. Let q, h, l, and r be integer codings of its state, position of its read-write head, the parts of the tape on the left and on the right of its head, respectively. A configuration of T is encoded in the integer $2^q 3^h 5^l 7^r$.

More generally, any discrete dynamical system in $\mathbb{Z}^n$ is equivalent to one in $\mathbb{Z}$ by using such encodings, viz., by associating $(i_1, i_2, \ldots, i_n)$ with $p_1^{i_1} p_2^{i_2} \ldots p_n^{i_n}$, where p_i is the ith prime.

We could have used such constructions instead of those in Proposition 6. However, we retain them since their transition functions have properties which those arising from the "prime encodings" do not (cf. Section 3.2). In any case, we conclude

Proposition 7 *Every TM, PDA, inputless PDA, FA, and inputless FA is equivalent to a discrete dynamical system in $\mathbb{Z}$. There is a discrete dynamical system in $\mathbb{Z}$ with the power of universal computation.*

It is important to note that one can extend the transition functions in $\mathbb{Z}^n$ above to functions taking $\mathbb{R}^n$ to $\mathbb{R}^n$. We may extend any function $f : A \subset \mathbb{Z}^n \to \mathbb{R}^m$ in such a manner, by first extending arbitrarily to domain $\mathbb{Z}^n$ and then using linear interpolation. Here is an example, used below:

Example 1 *A continuous* mod *function may be defined by $x \bmod_C m$ equals*

$$\begin{cases} (\lfloor x \rfloor \bmod m) + x - \lfloor x \rfloor, & 0 \le \lfloor x \rfloor \bmod m < m-1, \\ (m-1)(\lfloor x \rfloor + 1 - x), & \lfloor x \rfloor \bmod m = m-1. \end{cases}$$

Later results require extensions that are robust to small input errors. That is, one would like to obtain the integer-valued result on a neighborhood of each integer in the domain. For instance, one may define a continuous nearest integer function, $[\cdot]_C : \mathbb{R} \to \mathbb{R}$, that is robust in this manner as follows:

$$[x]_C \equiv \begin{cases} i, & i - 1/3 < x \le i + 1/3, \\ 3x - 2i - 1, & i + 1/3 < x \le i + 2/3. \end{cases}$$

More generally, define $\Pi : \mathbb{R}^n \to \mathbb{R}^n$, by

$$\Pi(x) = [[x_1]_C, \ldots, [x_n]_C].$$

Then given any function $f : \mathbb{R}^n \to \mathbb{R}^m$, with $f(\mathbb{Z}^n) \subset \mathbb{Z}^m$, we can define a "robust version" by using the function $f \circ \Pi$.

Thus, given $[A, \mathbb{Z}^+, F]$, $A \subset \mathbb{Z}^n$, its transition function may be extended to a continuous function from $\mathbb{R}^n$ to $\mathbb{R}^n$ which is constant in a neighborhood of each point in A. Such a remark is actually a byproduct of a more general result needed below [27, p. 216]:

Fact 1 *Any continuous function $f : A \to \mathbb{R}^m$, A a closed subset of $\mathbb{R}^n$, may be extended to a continuous map $\tilde{f} : \mathbb{R}^n \to \mathbb{R}^m$.*

Throughout the rest of this section we use continuous extensions as in the fact above, the notation $\tilde{f}$ always denoting such an extension of f.

3.2 Continuous Dynamical Systems Simulating TMs, PDA, and FA

It is possible to construct smooth systems of ODEs with inputs that "simulate" finite automata. For instance, in [10] Brockett used a system of his so-called double-bracket equations (also see [11]) to "simulate" the step-by-step behavior of a FA. This was done by coding the input symbols of the FA in a function of time that is the "control input" to a system of double-bracket equations. Specifically, if the input alphabet is $I = \{u_1, \ldots, u_m\}$, the input string $u_{i_0}, u_{i_1}, u_{i_2}, \ldots$ is encoded in a time function, $u(t)$, that is i_k on the intervals $[2kT, (2k+1)T]$ and zero otherwise. (In this paper, we encode the full input string in the initial condition of our simulations.)

In [10], Brockett was interested in the capabilities of his double-bracket equations. However, the resulting "simulations" of FA happen to behave poorly with respect to our definitions of simulation. Nevertheless, the key idea of his simulations of FA is that the input coding, $u(t)$, is used in such a way that it alternately switches between two different systems of double-bracket equations. The effect of the computation is that on alternating segments of time one computes the next state, then copies it, respectively. Then, the process may be repeated. In the most general sense, one has simulated the C program:

$$\text{for}(k = 0; ; k{+}{+}) \{\, h_{k+1} = F(x_k, u_k); \quad x_{k+1} = h_{k+1}; \}$$

It is not hard to see that one could use the same approach as that in [10] but more well-behaved systems of ODEs to simulate the step-by-step behavior of FA. Consider a FA with transition function δ, states $Q = \{q_1, \ldots, q_n\}$, and input alphabet I as above. Code state q_i as i and consider the first two equations of Eq. (2) below. Choose $\beta = n$ and replace, respectively, S_1, S_2, and G with $h_+(u(t))$, $h_-(u(t) - 1)$, and

$$D : \{1, \ldots, n\} \times \{1, \ldots, m\} \to \{1, \ldots, n\},$$

defined by $D(i, j) = k$ if $\delta(q_i, u_j) = q_k$. The result is that any FA may be SI-simulated by a system of ODEs in $\mathbb{R}^2$ *with input.* This was announced in [5].

It is also possible to simulate FA, PDA, and TMs using discontinuous vector fields [1]. However, the simulations used there are not robust to small errors.

In the present paper we concentrate on more general results and on coding the input as part of the initial state instead of a function of time. In addition, we do not allow access to precise timing signals or discontinuous vector fields, which considerably increases one's capabilities [7].

Above, we used an exact timing pulse or "clock" to precisely switch between two different vector fields in order to simulate discrete dynamical systems in $\mathbb{R}^n$. Even if one does not have access to a precise clock or allow discontinuous vector fields, one can still simulate discrete dynamical systems defined on subsets of $\mathbb{Z}^n$. (In fact, the theorem we prove below is not specific to $\mathbb{Z}^n$.) Again, the essential idea behind the simulations is to alternately switch between two different vector fields. However, when simulating systems in $\mathbb{Z}^n$, if one uses "robust versions" of their transition functions, and chooses well-behaved ODEs, it is not necessary to precisely time these switches using an exact clock. Indeed, we can use continuous functions to switch among vector fields.

It is still convenient to ensure, however, that only one vector field is active (non-zero) at any given time. To this end, we define the following.

Definition 8 (Inexact Clock) *Here, we give a Lipschitz continuous differential equation with output which implements an inexact clock.*

Define $\dot{\tau}(t) = 1/T$, *initialized at* $\tau(0) = 0$. *Now, define*

$$S_{1,2}(\tau) = h_{\pm}[\sin(\pi\tau)],$$

where

$$h_+(r) = \begin{cases} 0, & r \le \delta/2, \\ 2r/\delta, & \delta/2 < r \le \delta, \\ 1, & \delta < r, \end{cases}$$

$h_-(r) = h_+(-r)$, *and* $0 < \delta < \sqrt{2}/2$.

What is key is that the inexact clock above does not require discontinuous vector fields, discontinuous functions, or discrete dynamics. Thus, one can switch between two different systems of ODEs with (Lipschitz) continuous functions of the state of another (Lipschitz) ODE. This is why $2n + 1$ dimensional ODEs are used below to simulate an n-dimensional discrete dynamical system.

We also need the following technical definitions:

Definition 9 (Nondegeneracy, finite gain) *A function* $f : \mathbb{R}^n \to \mathbb{R}^n$, *is* nondegenerate (resp. finite gain) *if there exist constants* $\beta \ge 0$, $M \ge 0$, *such that*

$$\|x\| \le M\|f(x)\| + \beta, \qquad (\text{resp. } \|f(x)\| \le M\|x\| + \beta),$$

for all $x \in X$.

Now we are ready for our main simulation result:

Theorem 10 *Every discrete dynamical system F defined on $Y \subset \mathbb{Z}^n$*

1. *can be SI-simulated by a system of continuous ODEs in $\mathbb{R}^{2n+1}$.*
2. *such that $F(\cdot,1)$ is finite gain and nondegenerate can be I-simulated by a system of continuous ODEs in $\mathbb{R}^{2n+1}$.*
3. *such that Y is bounded can be SI-simulated by a system of Lipschitz ODEs in $\mathbb{R}^{2n+1}$.*

Proof Let $G \equiv F(\cdot,1)$ and $0 < \epsilon < 1/3$. $S_{1,2}$ and δ are as in the preceding example. For each $y \in Y$, define the set H_y to be those (x,z,τ) such that all the following hold:

$$\begin{aligned} \|x-y\|_\infty < \epsilon, \qquad & \|z-y\|_\infty < \epsilon, \\ \sin(\pi\tau) < \delta/2, \qquad & \tau \bmod_C 2 < 1/2. \end{aligned}$$

Set $\psi(x,z,\tau) = \Pi(z) = y$ if $(x,z,\tau) \in H_y$. Note that the $\psi^{-1}(y) = H_y$ are open and disjoint.

Initialize $x(0), z(0), \tau(0)$ in $\psi^{-1}(y)$, $y \in Y$.

1. Choose

$$\begin{aligned} \dot{x} &= -\epsilon^{-2}[x - \tilde{G}(\Pi(z))]^3 S_1(\tau), \\ \dot{z} &= -\epsilon^{-2}[z - \Pi(x)]^3 S_2(\tau), \\ \dot{\tau} &= 1. \end{aligned}$$

   It is straightforward to verify $\Pi(z(2k)) = G^k(y)$, $k \in \mathbb{Z}^+$, and the interval constraint.

2. Let α and L, β and M *each be one more than* the finite-gain, nondegeneracy constants of G under norm $\|\cdot\|_\infty$. Choose

$$\begin{aligned} \dot{x} &= -2\epsilon^{-1}[x - \tilde{G}(\Pi(z))] S_1(\tau), \\ \dot{z} &= -2\epsilon^{-1}[z - \Pi(x)] S_2(\tau), \\ \dot{\tau} &= 1/[\alpha + \beta + L\|z\|_\infty + M\|x\|_\infty]. \end{aligned} \tag{1}$$

   It is straightforward to verify $\Pi(z(t)) = G^k(y)$ on an interval about the time t_k where $\tau(t_k) = 2k$, $k \in \mathbb{Z}^+$.

3. Let $\beta = \max\{\|i-j\|_\infty \mid i,j \in Y\}$. Choose

$$\begin{aligned} \dot{x} &= -2\beta\epsilon^{-1}[x - \tilde{G}(\Pi(z))] S_1(\tau), \\ \dot{z} &= -2\beta\epsilon^{-1}[z - \Pi(x)] S_2(\tau), \\ \dot{\tau} &= 1. \end{aligned} \tag{2}$$

   It is straightforward to verify $\Pi(z(2k)) = G^k(y)$, $k \in \mathbb{Z}^+$, and the interval constraint. □

Note that nondegeneracy and finite gain of the extension $\tilde{G}$ need not hold for points not in Y. Note also that the simulations above are "robust" in the sense that there is a neighborhood of initial conditions leading to the correct simulations. The import of part 2 of the theorem is that if $G \equiv F(\cdot,1)$ is nondegenerate and may be extended to a Lipschitz function, then the ODEs used in the I-simulation are also Lipschitz.

Note also that the theorem continues to hold for any discrete dynamical system defined on $Y \subset \mathbb{R}^n$ such that there is some minimum separation between any two distinct points of Y.

The discrete dynamical systems equivalent to TMs given by Proposition 6 have transition functions that are both finite gain and nondegenerate. Unfortunately, the transition functions of systems equivalent even to PDA need not be Lipschitz. Consider a PDA which pushes a tape symbol γ on input symbol i_1 and pops γ on input symbol i_2 and test with inputs of the form i_1^{n+1}, $i_1^n i_2$. One may check that the "prime encodings" mentioned earlier lead to transition functions that are neither finite gain nor nondegenerate.

Thus, relating the theorem back to simulation of TMs, PDA, and FA, we have many results, the most striking of which are:

Corollary 11 *Every TM, PDA, and FA can be SI-simulated by a system of continuous ODEs in $\mathbb{R}^3$.*

Every FA (resp. *inputless FA*) *can be I-simulated* (resp. *SI-simulated*) *by a system of Lipschitz continuous ODEs in $\mathbb{R}^3$.*

Using SI-simulation, there is a system of continuous ODEs in $\mathbb{R}^3$ with the power of universal computation.

Proof Everything is immediate from the theorem and Propositions 6 and 7 except that the FA transition function is Lipschitz, which is readily checked. □

All the simulation results for discrete dynamical systems on $\mathbb{Z}$ can be extended from continuous to smooth vector fields by using C^∞ interpolation (with so-called "bump" functions [14]) rather than linear interpolation in extending their transition functions and the functions $[\cdot]_C$ and $h_\pm$, and by replacing $\|\cdot\|_\infty$ with $\|\cdot\|_2$ in Eq. (1).

4 Towards an ODE Model of Analog Computation

Performed computation is a physical activity. Thus, all computational processes are subject to general physical constraints. Such ideas are at the core of

Turing's machines and Church's thesis. We take the following physical computability axioms from [16][1]

P1. The speed of propagation of information is bounded.

P2. The amount of information which can be encoded in the state of a finite system is bounded.

P3. The topology of space-time is locally Euclidean (a limit on circuit connectivity).

For further discussion, see [2, 15, 16, 32, 33]. As a prototypical example of a state which stores information, we consider the voltage on a capacitor.[2] The reader can surely think of many others.

We now discuss a model of analog computation via differential equations, based largely on [35]. We will deal with ODEs with output:

$$\begin{aligned} \dot{X}(t) &= f(X(t)), \qquad X(t_0) = X_0 \\ Y(t) &= Q(X(t)) \end{aligned} \tag{3}$$

where $X(t)$ and $Y(t)$ are elements of $\mathbb{R}^n$ and $\mathbb{R}^p$, respectively. This represents the constraint that

C1. The state and output dimensions are finite for any given computation.

The output function Q is assumed to have the property that its output can be "read out" unambiguously with respect to small changes, i.e., we will assume that $Q(X(t_f), P)$ is constant on some ϵ ball around $X(t_f)$ if the "answer" is computed via the differential equation in time t_f. Thus

C2. Inputs (including initial conditions) can be specified, and outputs can be read only to absolute precision, ϵ.

This has many implications (see [6, 28]).

We also want to only consider equations for which the following hold (from P1, P1, and P2, respectively):

C3. f is locally Lipschitz.

C4. $R = \max_{t_0 \le t \le t_f} \|\ddot{X}(t)\| < \infty$

C5. All states and their derivatives are bounded in infinite time for any given computation.

[1] A subset of Fredkin and Toffoli's axioms, not their numbering.

[2] As considered in circuit theory: a continuous variable despite the fact that charge is carried by discrete electrons.

In terms of our capacitor model, the amount of charge that can be stored—and the rate at which we can store charge—are globally bounded quantities. This assumption, along with our ϵ precision condition implies that the information we can store in a single physical variable is bounded. The fact that these bounds are only for "any given computation" is very important if one is to allow the power of simulating TMs. It also allows us to consider physical digital computers as analog computers. Otherwise, the effect of C1, C2, and C5 would reduce us to a finite automata. It is analogous to the fact that the number of tape cells used in any given TM computation can be bounded. Together, C3 and C5 imply that for each computation there is a nonnegative constant Λ such that $\|f(X_1) - f(X_2)\| \le \Lambda \|X_1 - X_2\|$. Λ is necessarily a measure of resources as scaling it scales the time necessary to complete the computation.

Finally we would also like a constraint on the complexity of the function f. We are trying to avoid the construction of f's so complex that they are just look-up tables for the computations at hand. Thus, we want something like

C6. The f_i must be computable by a polynomial-sized circuit.

This is a constraint on the (circuit) complexity of computing the function f using, say, the adders, multipliers, and constant scalers of analog devices. We will not be more precise than this here, but it is easy to see why such an assumption is desirable. For instance, one could have as the right-hand side functions which effectively compute the final answer one was trying to achieve in the first place. Thus, the complexity of computing the f_i needs to be considered in any complexity theory of analog computation via ODEs.

More concretely, consider the following example. In [11], Brockett notes that the following set of differential equations can, among other things, sort n distinct numbers:

$$\dot{H} = [H, [H, N]].$$

Here, H is a symmetric $n \times n$ matrix with $H(0)$ chosen so that its eigenvalues are the numbers to be sorted, $N = \mathrm{diag}\{1, 2, \ldots, n\}$, and $[A, B] = AB - BA$. These equations perform the computation of sorting numbers in the sense that (except for some initial conditions in a set of measure zero) as $t \to \infty$, $H(t)$ converges to a diagonal matrix whose diagonal entries are the numbers sorted in increasing order (see [11] for more details).

Now we know that sorting n numbers is an $O(n \log n)$ operation. But just computing the right-

hand side of the above ODE takes $O(n^3)$ operations. Comparisons are further hindered by the fact that time of sorting on a digital computer depends on the number of bits needed to represent numbers, while in Brockett's case, one is sorting reals (cf. [3]).

Of particular importance are systems like those we used in Section 3, which (with constant U) globally asymptotically converge to a single value; the prototypical example being $\dot{X}(t) = AX(t) + BU$, where A and B are constant matrices of size $n \times n$ and $n \times m$, respectively, and A is strictly Hurwitz. Here U represents a subset of the state that is constant (for instance, to code inputs) or changes on a much faster time scale than X (as in our inexact clock above). See [9] for a discussion of more models with properties akin to this. The additional constraint of ϵ absolute precision allows models with this property to converge to a quantized answer in finite time. Further, these models are both robust to small noise and modeling errors [8] and will "maintain" the correct answer (up to quantization) for all time after convergence, making the timing of readout and switching noncritical.

5 Conclusions

We defined notions of simulation of a discrete dynamical system by a continuous dynamical system. S-simulation, or simulation via section, was motivated by the definition of global section in dynamical systems [30]. Relaxing this to allow different parameterizations of time we considered P-simulation, which was seen to be weak. To remedy this, we defined I-simulation, or simulation via intervals. Both S-simulation and I-simulation imply P-simulation. S-simulation and I-simulation are independent notions.

We found that one can simulate arbitrary discrete dynamical systems defined on subsets of $\mathbb{Z}^n$ without resorting to discontinuous vector fields, exact timing pulses or "clocks," or infinitely many state variables. Instead, one can use an approximation to an exact clock, implemented with a one-dimensional Lipschitz ODE. The result is that we can perform SI-simulations (resp. I-simulations) using continuous (resp. Lipschitz) ODEs in $\mathbb{R}^{2n+1}$.

Related to our general simulation results is a theorem by N. P. Zhidkov [37] (see also [30, p. 135]), that states if a reversible discrete dynamical system is defined on a compact subset $K \subset \mathbb{R}^n$, then there exists on a subset of $\mathbb{R}^{2n+1}$ a reversible continuous dynamical system that is defined by ODEs and has K as a global section.

As far as computational abilities, we saw that there are systems of continuous ODEs possessing the ability to SI-simulate arbitrary pushdown automata and Turing machines. Finite automata may be SI-simulated with continuous, Lipschitz ODEs. By SI-simulating a universal Turing machine, we concluded that there are ODEs in $\mathbb{R}^3$ with continuous vector fields possessing the power of universal computation. Further, the ODEs simulating these machines may be taken smooth and do not require the machines to be reversible (cf. [26, p. 228]).

The import of S-simulation here is that such simulations take only "linear time" [13]. The import of I-simulation is that the readout times for which the state/tape is valid are non-empty intervals. Indeed, the intervals are at least some minimum length. Also, the simulations were "robust" in the sense that they can tolerate small errors in the coding of the initial conditions. Though not required by our definitions, the sets of "valid" initial conditions contained balls of at least some minimum diameter.

We now turn to some discussion. To demonstrate the computational capabilities of continuous dynamical systems, we first constructed low-dimensional discrete dynamical systems in $\mathbb{Z}^n$ equivalent to Turing machines (TMs), pushdown automata (PDA), and finite automata (FA). It is well-known that certain discrete dynamical systems are equivalent to TMs and possess the power of universal computation (see, e.g., [13, 25, 31]). Our systems were constructed with the goal of simulation by continuous/Lipschitz ODEs in mind. One notes that while it is perhaps a trivial observation that there are systems of (Lipschitz) ODEs with the power of universal computation—just write down the ODEs modeling your personal computer—this requires a system of ODEs with a potentially infinite number of states.

Our simulation of arbitrary Turing machines was announced in [5]. It is now a special case of the current results. These results imply that, in general, questions regarding the dynamical behavior of low-dimensional, well-behaved ODEs are computationally undecidable. See [25, 26] for a discussion of such questions.

The best definition of "simulation" is not apparent. While stated in terms of our definitions of simulation, the simulation results of Section 3 are intuitive and would probably continue to hold under alternate definitions of simulation. Our discussion of a model for analog computation is preliminary in nature. These remain topics of research.

Condition C5 of Section 4 may be achieved by the simulations of Theorem 10 (hence Cororollary 11) by turning off the last equation in certain conditions (e.g., upon halt or input read) or by using an inexact clock based on a variable-rate harmonic oscillator (in S^1 or $\mathbb{R}^2$, e.g., $\dot{\tau} = -\gamma(x,z)\rho$, $\dot{\rho} = \gamma(x,z)\tau$).

Acknowledgements

This work was supported by the Army Research Office and the Center for Intelligent Control Systems under contracts DAAL03-92-G-0164 and DAAL03-92-G-0115. Foremost, thanks to Sanjoy K. Mitter for his guidance and support. The author would also like to thank Eduardo Sontag for an engaging discussion on analog computation and Sandro Zampieri for one on definitions of simulation.

References

[1] E. Asarin and O. Maler On some relations between dynamical systems and transition systems Preprint. Nov. 1993. Submitted to *ICALP '94*.

[2] C.H. Bennett. Computational measures of physical complexity. In *Lectures in the Science of Complexity*, pp. 787–798. Addison-Wesley Longman, 1989.

[3] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers. *Bull. American Math. Soc.*, 21(1):1–46, 1989.

[4] L.S. Bobrow and M.A. Arbib. *Discrete Mathematics.* W. B. Saunders, Philadelphia, 1974.

[5] M.S. Branicky. Equivalence of analog and digital computation. In *Workshop on Continuous Algorithms and Complexity*, Barcelona, Oct. 1993. Centre de Recerca Matematica. Abstract.

[6] M.S. Branicky. Topology of hybrid systems. In *Proc. 32nd IEEE Conf. on Decision and Control*, pp. 2309–2314, San Antonio, TX, Dec. 13–17, 1993.

[7] M.S. Branicky. Universal computation and other capabilities of hybrid and continuous dynamical systems. TR LIDS-P-2218, Lab. Information Decision Systems, MIT, Dec. 1993. (To appear in *Theoretical Computer Science*).

[8] M.S. Branicky. Continuity of ODE Solutions. *Applied Math. Lett.*, (to appear).

[9] R.W. Brockett. Pulse driven dynamical systems. In A. Isidori and T.J. Tarn, eds., *Systems, Models, and Feedback*, pp. 73–79, Birkhauser, Boston, 1992.

[10] R.W. Brockett. Smooth dynamical systems which realize arithmetical and logical operations. In H. Nijmeijer and J.M. Schumacher, eds., *Three Decades of Mathematical Systems Theory*, pp. 19–30. Springer, Berlin, 1989.

[11] R.W. Brockett. Dynamical systems that sort lists, diagonalize matrices, and solve linear programming problems. *Lin. Alg. and Its Appl.*, 146:79–91, 1991.

[12] M.T. Chu. On the continuous realization of iterative processes. *SIAM Review*, 30(3):375–387, 1988.

[13] M. Cosnard, M. Garzon, and P. Koiran. Computability properties of low-dimensional dynamical systems. In *Proc. 10th Symp. Theoretical Aspects Comp. Science*, LNCS 665, Springer, 1993.

[14] R.L. Devaney. *An Introduction to Chaotic Dynamical Systems.* Addison-Wesley, 2nd edition, 1989.

[15] E. Fredkin. Digital mechanics. *Physica D*, 45:254–270, 1990.

[16] E. Fredkin and T. Toffoli. Conservative logic. *Int. J. of Theoretical Physics*, 21(3/4):219–253, 1982.

[17] M.R. Garey and D.S. Johnson. *Computers and Intractability.* W. H. Freeman, New York, 1979.

[18] J. Guckenheimer and P. Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields.* Springer, New York, 1990.

[19] M.W. Hirsch and S. Smale, *Differential Equations, Dynamical Systems, and Linear Algebra*, Academic, San Diego, 1974.

[20] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, MA, 1979.

[21] P. Kůrka. Simulation in dynamical systems and Turing machines. TR, Dept. of Math. Logic and Philosophy of Math., Charles Univ., Praha, Czechia, 1993.

[22] D.G. Luenberger. *Introduction to Dynamic Systems.* John Wiley and Sons, New York, 1979.

[23] C. Mead. *Analog VLSI and neural systems.* Addison-Wesley, Reading, MA, 1989.

[24] M.L. Minsky. *Computation: Finite and Infinite Machines.* Prentice-Hall, Englewood Cliffs, NJ, 1967.

[25] C. Moore. Unpredictability and undecidability in dynamical systems. *Phys. Rev. Lett.*, 64:2354–2357, 1990.

[26] C. Moore, Generalized shifts: unpredictability and undecidability in dynamical systems. *Nonlinearity*, 4:199–230, 1991.

[27] J.R. Munkres, *Topology*, Prentice-Hall, Englewood Cliffs, NJ, 1975.

[28] A. Nerode and W. Kohn. Models for hybrid systems: Automata, topologies, stability. TR 93-11, Mathematical Sciences Institute, Cornell Univ., Mar. 1993.

[29] S. Omohundro. Modelling cellular automata partial differential equations. *Physica D*, 10:128–134, 1984.

[30] K.S. Sibirsky. *Introduction to Topological Dynamics.* Noordhoff, Leyden, Netherlands, 1975.

[31] H.T. Siegelman and E.D. Sontag. Turing computation with neural nets. *Applied Math. Lett.*, 4:77–80, 1991.

[32] T. Toffoli. Physics and computation. *Int. J. of Theoretical Physics*, 21(3/4):165–175, 1982.

[33] T. Toffoli. Cellular automata as an alternative to (rather than an approximation of) differential equations in modelling physics. *Physica D*, 10:117–127, 1984.

[34] J.F. Traub and H. Wozniakowski. Theory and applications of information-based complexity. In *1990 Lectures in Complex Systems*, pp. 163–193. Addison-Wesley, Redwood City, CA, 1991.

[35] A. Vergis, K. Steiglitz, and B. Dickinson. The complexity of analog computation. *Math. and Computers in Simulation*, 28:91–113, 1986.

[36] D.S. Watkins and L. Elsner. Self-similar flows. *Lin. Alg. and Its Appl.*, 110:213–242, 1988.

[37] N.P. Zhidkov, Nekotorie svoistva diskretnikh dinamicheskikh sistem, *Mosk. Gos. Univ. Lomonosova Uch. Zap.*, **163**, *Matematika* 6:31–59, 1952 (Russian).