How much can analog and hybrid systems be proved (super-)Turing*

Olivier Bournez
LORIA/INRIA
615 Rue du Jardin Botanique
54602 Villers lès Nancy
FRANCE
email: bournez@loria.fr

October 14, 2008

Abstract

Church thesis and its variants say roughly that all reasonable models of computation do not have more power than Turing Machines. In a contrapositive way, they say that any model with super-Turing power must have something unreasonable.

Our aim is to discuss how much theoretical computer science can quantify this, by considering several classes of continuous time dynamical systems, and by studying how much they can be proved Turing or super-Turing.

1 Introduction

One major result of the twentieth century is Kurt Gödel incompleteness theorem [23], demonstrating that no proof system can capture our reasoning about natural numbers. The original arguments in [23] are based on an informal notion of deduction. A few time after Gödel's paper, Alan Turing proposed in [54], a model of machine able to capture formal deductions in any deduction system.

Actually, what Turing outlined a proof is: What can be calculated by a human working mechanically with paper and pencil in a finite number of steps (in particular this covers deduction in formal systems) is computable by a Turing Machine [17, 22, 54].

*This work was partially supported by French Ministry of Research through ANR Project SOGEA.

It was soon discovered[1] that the power of Turing machines can be proved to be equal to several other formalisms that have been introduced, including the lambda calculus from Alonzo Church [15], and the recursive functions from Stephen Kleene [27].

These considerations leaded to Church-Turing's thesis: "What is effectively calculable is computable". In that thesis "calculable" refers to some intuitively given notion, whereas "computable" means "computable by a Turing machine" [17, 22, 41].

Following Jack Copeland [17], the original thesis refers to a notion of calculation, where calculation is intended in the sense that it can be[2] realized by a human computing mechanically with paper and pencil, and is often confused with the following thesis (called Thesis M in [22]): "What can be calculated by a machine is computable" [17].

Here the notion of machine still refers to some intuitively given notion of machine, with the constraint that the machine is "intended to conform to the physical laws (if not to the resource constraints) of the actual world" [17], otherwise the thesis is known to be false: see e.g. the surveys [18, 41] or the examples to follow.

One close variant of this thesis, also discussed in [17], is the following: "Any process that can be given a mathematical description can be simulated by a Turing machine". Once again (and actually for the same counter-examples) if the process abstracts from the issue whether it could exist in the actual world, the thesis is known to be false [17].

The three theses are independent:

- first thesis has to do with computations realizable by humans working mechanically with paper and pencils [17],
- second has to do with physic of the actual world [17, 40, 51, 57],
- third has to do with our models of the physic of the actual world [17, 40, 51, 57].

We believe that each thesis has actually to do with convictions, since none of them is truly provable, as each of them is referring to some informal notions or to the actual world of which we do not have a model[3]. There have been however several tentatives of proofs in literature, relying on more "basic" hypotheses about the involved notions: see e.g. [8, 22].

If we take each thesis in a contrapositive way, they mean that any system that computes something not computable by a Turing machine involves something, call it a "resource", that is either non-calculable by a mechanical method, or by

[1] As observed in [16], this might be considered as not so surprising, since both formalisms have been explicitly design to solve Hilbert's Entscheidungsproblem (whether an arbitrary formula of the predicate calculus can be decided to be a tautology).

[2] At least in principle.

[3] Note that as soon as we believe in the existence of concepts like integers, Gödel's theorem says we can't have a model.

a physical machine, or by a model of a physical machine. Call such a resource "non-reasonable".

Our aim is not to argue in favor or against each of the theses, but to try do discuss on what makes a resource "non-reasonable".

Actually, as soon as we talk about Turing machines, we are dealing with something that can be considered as non-reasonable: a Turing machine involves an infinite tape, and hence something infinite. Infiniteness is a formal notion, and hence a first measure of the complexity of a resource.

But, this is not the only resource that can be considered as non-reasonable, and not the only possible measure. Similarly to what is argumented by José Félix Costa and Jerzy Mycka in [39], what is missing is a clear and well understood way to measure complexity of the reasonableness of resources.

In this paper we discuss the power of several models of continuous time dynamical systems with respect to the power of Turing machines. We consider several variants of systems, according to some hypotheses made about their "reasonableness", and we try to compare their power with Turing machines, from a computability and complexity point of view.

We would like to say that our motivation and discussion about measuring the complexity of involved resources is very close[4] to one of the motivation of José Félix Costa and Jerzy Mycka for studying analog computations in their series of papers (see for example [34, 35, 36, 37, 38]), expressed explicitly in [39].

We also add that we do not claim that the considered models have any physical reality. We mainly focus on these models, since they are models that have already been considered and proposed in literature, about (idealized abstract) concrete systems of our world, and since we think that they are informative about critical "non-reasonable" resources in our world. Most of them are clearly unrealistic, or involve non-computable things, but we think that, even if we believe the[5] theses true, refusing to discuss about such models is only refusing to talk about the reasons why we think that the theses should be true.

Several papers, in particular from people mainly advocating against hypercomputations, have argued that discussing some systems in some physical theory able of do hyper-computations helps to understand weakness of the physical models of our world [2, 50, 52, 53]. Our aim is in some sense a parallel computer scientist point of view: Discussing theoretical models able to do hypercomputations, helps to understand weakness of theoretical computer science models.

2 Mathematical preliminaries

Let $\mathbb{N}$, $\mathbb{Q}$, $\mathbb{R}$, denote the set of natural integers, the set of rational numbers, and the set of real numbers respectively. Given $x \in \mathbb{R}^n$, we write $\mathbf{x}$ to emphasize

[4] Even if we think that our own general goal is more about trying to understand the relations between models, and their power, and we do not think so strongly that the toolbox of analysis could help to solve classical discrete problems [39].

[5] Or a subset of the theses.

that x is a vector. $\|.\|$ will denote the sup norm.

An open (respectively closed) half space, is the set of points $\mathbf{x} \in X$, satisfying $\mathbf{a}.\mathbf{x} < b$ (resp. $\mathbf{a}.\mathbf{x} \leq b$) for some $\mathbf{a} \in \mathbb{R}^d$, $b \in \mathbb{R}$ where . stands for inner product. It is said rational if furthermore $\mathbf{a} \in \mathbb{Q}^d$, $b \in \mathbb{Q}$.

A polyhedron P is any boolean combination (intersections, unions) of open or closed half spaces. It is said rational if the half spaces are.

Definition 1 (Dynamical systems)

- *A (homogeneous inputless) continuous time dynamical system $\mathcal{H}$ is given by $X \subset \mathbb{R}^d$, and some function $f : X \to X$.*
- *A trajectory of $\mathcal{H}$ starting from $\boldsymbol{x}_0 \in X$, is a solution of differential equation $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$, $\boldsymbol{x}(0) = \boldsymbol{x}_0$: that is a continuous and derivable function $\phi : \mathbb{R}^+ \to X$, with $\phi(0) = \boldsymbol{x}_0$, and $\frac{d\phi}{dt}(t) = f(\phi(t))$ for all t.*

Given some property of functions, we will say that a dynamical system has this property if the corresponding function f has. For example, derivable continuous time dynamical systems denote the class of continuous time dynamical systems $\mathcal{H} = (X, f)$ where f is derivable.

There are several ways to evaluate the complexity of function f: one of them is to talk about its smoothness: a function $f : X \subset \mathbb{R}^d \to \mathbb{R}^d$ is said of class $\mathcal{C}^\infty$, if it is r-times continuously differentiable on X, for all $r \in \mathbb{N}$. Functions of class $\mathcal{C}^\infty$ include analytic functions.

One other possibility is to talk about its computational properties in recursive analysis model: see [56] for an up-to-date monograph presentation of recursive analysis from a computability point of view, or [28] for a presentation from a complexity theory point of view.

Following Ker-I Ko [28], let $\nu_{\mathbb{Q}} : \mathbb{N} \to \mathbb{Q}$ be the following representation[6] of dyadic rational numbers by integers: $\nu_{\mathbb{Q}}(\langle p, q, q\rangle) \mapsto \frac{p-q}{2^r}$, where $\langle ., ., .\rangle : \mathbb{N}^3 \to \mathbb{N}$ is a polynomial time computable bijection.

A sequence of integers $(x_i) \in \mathbb{N}^{\mathbb{N}}$ converges quickly toward x (denoted by $(x_i) \rightsquigarrow x$) if the following holds for all i: $|\nu_{\mathbb{Q}}(x_i) - x| < 2^{-i}$.

A point $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ is said computable (denoted by $\mathbf{x} \in \mathcal{R}ec(\mathbb{R})$) if for all j, there is a computable sequence $(x_i) \in \mathbb{N}^{\mathbb{N}}$ with $(x_i) \rightsquigarrow x_j$. It is said polynomial time computable (denoted by $\mathbf{x} \in P(\mathbb{R})$) if the corresponding sequences are.

A function $f : X \subset \mathbb{R}^d \to \mathbb{R}$, where X is compact, is said computable (denoted by $f \in \mathcal{R}ec(\mathbb{R})$), if there exists some d-oracle Turing machine M, such that for all $\mathbf{x} = (x_1, \ldots, x_d) \in X$, for all sequences $(x_i^j) \rightsquigarrow x_j$, M taking as oracles these d sequences, computes a sequence (x'_i) with $(x'_i) \rightsquigarrow f(\mathbf{x})$. A function $f : X \subset \mathbb{R}^d \to \mathbb{R}^d$, where X is compact, is said computable if all its projections are. It is said polynomial time computable (denoted by $f \in P(\mathbb{R})$), if furthermore the involved oracle Turing machines work in polynomial time.

Other alternatives to measure the complexity of a given function exist. One of them, that we will not discuss, has been initiated by [32], and consists in

[6] Many other natural representations of rational numbers can be chosen and provide the same class of computable functions: see [28, 56].

discussing its membership in algebraically defined classes of functions generated by a finite set of basic functions, and closed by some simple operators: see for e.g. [12, 13, 14, 25, 35, 36].

In this paper, we will consider dynamical systems as recognizers of languages: Σ will denote alphabet $\{0,1\}$. Σ^* will denote words over this alphabet.

Two (very classical) encodings of words into real numbers will play some important role in what follows:

- ν_X is the function that maps Σ^* to $[0,1]$ as follows: word $w = w_1 \dots w_n \in \{0,1\}^*$ is mapped to $\nu_X(w) = \sum_{i=1}^{n} \frac{(2w_i+1)}{4^i}$.
- $\nu_{\mathbb{N}}$ is the function that maps Σ^* to $\mathbb{N}$ as follows: word $w = w_1 \dots w_n \in \{0,1\}^*$ is mapped to $\nu_{\mathbb{N}}(w) = \sum_{i=1}^{n} (2w_i + 1)4^i$.

We can now define.

Definition 2 (Dynamical Systems as Language Recognizers) *Let $\mathcal{H}$ be a continuous time dynamical system over space X. We will consider two cases: the case $X = [-1,1]^d$ (compact case), or $X = \mathbb{R}^d$ (unrestricted case). Consider $\nu = \nu_X$ for the compact case, $\nu = \nu_{\mathbb{N}}$ for the unrestricted case. Let V_{accept} be the set of $\boldsymbol{x} \in X$ with $\|\boldsymbol{x}\| \leq 1/4$. Let $V_{compute}$ be the set of $\boldsymbol{x} \in X$ with $\|\boldsymbol{x}\| \geq 1/2$.*

We will say that $\mathcal{H}$ computes some language $L \subset \Sigma^*$, over alphabet $\Sigma = \{0,1\}$, if the following holds: for all $w \in \Sigma^*$, $w \in L$ iff the trajectory of $\mathcal{H}$ starting from $(\nu(w), 0, \cdots, 0, 1)$ reaches V_{accept}.

For robustness reasons, we assume that, for any $w \notin L$, the corresponding trajectory stay forever in $V_{compute}$.

Given some notion of time associated to trajectories, we will say that L is recognized in time T, if furthermore when the trajectory reaches V_{accept}, trajectory has a time bounded above by T. It is said accepted in time $f : \mathbb{N} \to \mathbb{N}$ if furthermore $T \leq f(|w|)$, for all w, where $|w|$ stands for the length of w.

3 A Toy Example

We are going to discuss the Piecewise Constant Derivative (PCD) model that has been introduced by Eugene Asarin, Oded Maler and Amir Pnueli in [5], as a simple model for hybrid systems. It has later on been discussed in several papers such as [3, 4, 10].

A hybrid system is a system that combines continuous evolutions with discrete transitions. Such models appear as soon as one tries to model some systems where a discrete system, such as a computer, evolves in a continuous environment: See e.g. [1].

From a theoretical computer science point of view, one interest of the hybrid systems models, is that they generalize both discrete time transition systems and continuous time dynamical systems.

Definition 3 (PCD System [4]) *A (rational) piecewise-constant derivative (PCD) system is a continuous time dynamical system $\mathcal{H}$, defined by differential equation $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ on $X \subset \mathbb{R}^d$, where $f : X \to \mathbb{R}^d$, can be represented by the formula*

$$f(\boldsymbol{x}) = \boldsymbol{c}_i \text{ for } \boldsymbol{x} \in P_i,\ i = 1, \ldots, n$$

where $c_i \in \mathbb{Q}^d$, and the P_i constitutes a partition of X into rational polyhedra.

A trajectory of $\mathcal{H}$ starting from some $\mathbf{x}_0 \in X$, is a solution of the differential equation $\dot{\mathbf{x}} = f(\mathbf{x})$ with initial condition $\mathbf{x}(0) = \mathbf{x}_0$: that is a continuous function $\phi : \mathbb{R}^+ \to X$ such that $\phi(0) = \mathbf{x}_0$, and for every t, $f(\phi(t))$ is equal to the *right derivative* of $\phi(t)$.

In other words, a PCD system consists of partitioning the space into convex polyhedral sets ("regions"), and assigning a constant derivative $\mathbf{c}$ ("slope") to all the points sharing the same region. The trajectories of such systems are broken lines, with the breakpoints occurring on the boundaries of the regions [5]: see Figure 1.

Figure 1: A PCD system in dimension 2.

Eugene Asarin, Oded Maler, and Amir Pnueli have proved that PCD systems can simulate Turing machines, as soon as we suppose the dimension $d \geq 3$ [5] (observe that conversely any language computed by a rational PCD system $\mathcal{H}$ is clearly recursively enumerable).

Theorem 1 (PCD systems = Turing [5]) *1. Any recursively enumerable set L is computed by a (rational) PCD system $\mathcal{H}$ over $[-1, 1]^3$.*

2. This does not hold over $[-1, 1]^2$, nor $\mathbb{R}^2$, in the general case.

The trick used in [5] has already been seen is several other contexts (see e.g. [29, 49]): the current state of a Turing machine at some time t, given by some internal state $q \in Q$, and some tape $w_{-m}w_{-m+1} \ldots w_0 w_1 \ldots w_n$, with the head in front of cell w_0, is encoded into two real numbers (x_1^t, x_2^t) by

$x_1^t = q + \nu_X(w_0 w_1 \ldots, w_n)$, $x_2^t = \nu_X(w_{-1} w_{-2} \ldots w_{-m})$. Computing the encoding (x_1^{t+1}, x_2^{t+1}) of the state of the machine at time $t+1$ reduces in doing multiplications by 4, divisions by 4, as well as additions, depending on the current scanned symbols of the simulated Turing machine, that can be read easily by testing the membership of x_1^t and x_2^t in some simple intervals. Each such operation and test can be implemented by regions of PCD systems. The point is then just to build "paths" that bring the output of the regions that computes (x_1^{t+1}, x_2^{t+1}) from (x_1^t, x_2^t) to their input, so that the whole system computes sequence (x_1^t, x_2^t) for all t.

4 On Imposing Smoothness

It can be objected that piecewise constant derivative systems involve discontinuous functions, and hence something non-reasonable, and hence that Theorem 1 do not deal with "realistic" functions.

Actually, it can be reinforced as follows (see [31] for a proof) (observe that an alternate proof obtained by "smoothing" previous PCD system construction is proposed in [11]).

Theorem 2 (Smooth Systems $\geq$ Turing [31]) *Any recursively enumerable set L is computed by a $\mathcal{C}^\infty$ (and $\mathcal{R}ec(\mathbb{R})$) continuous time dynamical system $\mathcal{H}$ over $[-1,1]^3$.*

It is known that there exist differential equations, with computable coefficients, with computable initial conditions, that cannot be numerically solved via deterministic methods by a digital computer. One example was provided by Marian Pour-El and Ian Richards in [43]: there exists a polynomial-time computable function $f : [0,1] \times [-1,1] \to \mathbb{R}$ such that the equation $\frac{dx}{dt} = f(t,x)$ defined by f does not have a computable solution y on $[0,\delta]$, for any $\delta > 0$.

Same authors later on expanded their result to show that wave equation (which is a partial equation), even with computable initial data, can have a unique solution which is not computable [44].

However, if an ordinary differential equation over a compact has a unique solution, then it must be computable: see e.g. [28]. This holds has soon as f is twice continuously differentiable.

Remark: However, note that even if the solution of an ordinary differential equation is unique, the complexity of the computable solution $y(t,x)$ has no fixed complexity bounds: For any recursive real number a between 0 and 1, there exists a polynomial-time computable function $f : [0,1] \times [-1,1]$ such that $y(x) = ax^2$ is the unique solution of $\frac{dx}{dt} = f(t,x)$ [28, 30].

From these considerations, we get.

Corollary 1 (Smooth and Computable Systems = Turing) *Continuous time dynamical systems of $\mathcal{C}^\infty \cap \mathcal{R}ec(\mathbb{R})$ over $[-1,1]^d$ have precisely the power of Turing machines: they recognize precisely recursively enumerable sets.*

It is conjectured in [33] that no analytic map on a compact, and finite-dimensional space, can simulate a Turing machine, through a reasonable input and output encoding. The question whether we can suppose the continuous time dynamical system analytic in previous corollary is *à priori* distinct. However, if we believe the conjecture true, a negative answer would be surprising since most known undecidability results (putting aside results obtained by a pure diagonalization), rely on the simulation of a Turing machine[7].

If the constraint of bounded space is relaxed, it has been recently obtained by Daniel Graça, Manuel Campagnolo and Jorge Buescu that Turing machines can be simulated by analytic maps (furthermore in an error-robust manner) [24].

Theorem 3 (Non-Compact Analytic Systems $\geq$ Turing [24]) *Any recursively enumerable set L is computed by an analytic (and $\mathcal{R}ec(\mathbb{R})$) continuous time dynamical system $\mathcal{H}$ over $\mathbb{R}^7$.*

5 On Relaxing Rationality

Suppose that we relax the hypothesis that the c_i and the polyhedra P_i are rational in Definition 3. If no constraint is put on the involved real constants, it has been proved in [11] that any language $L \subset \Sigma^*$ is computed by some (non-rational) PCD system.

We believe that restricting to discrete polynomial time yields more interesting results: the discrete time of a trajectory is the number of regions crossed by the trajectory. Formally,

Definition 4 (Discrete Time) *To any trajectory $\phi : \mathbb{R}^+ \to X$ of a PCD system $\mathcal{H}$, associate the set T_ϕ of the time $t_i \geq 0$ at which the direction of ϕ change: the left derivative of ϕ in t_i does not exist, or is distinct from its right derivative. We say that ϕ has discrete time n, if T_ϕ contains n elements.*

Note that there is also an other natural notion of time for continuous time dynamical systems.

Definition 5 (Continuous Time) *To any trajectory $\phi : \mathbb{R}^+ \to X$ of a continuous time dynamical system (for e.g. a PCD system), the continuous time of the trajectory $\phi(t)$ is the variable t.*

Example: The trajectory of Figure 1 has discrete time 9. If we suppose that the norm of the speed vectors are 1, its continuous time is equal to its length.

Recall (see e.g. [7, 42]) that a family of boolean circuits $\mathcal{C} = (C_i)_{i\in\mathbb{N}}$, with C_i with i inputs and 1 output, recognizes a language $L \subset \Sigma^*$, iff for all $w \in \Sigma^*$, $w \in L$ if and only if $C_{|w|}$ accepts w.

[7] Or of models like two-counters machines that simulate Turing machines, or of problems like post-correspondence-problems that encode the simulation of (non-deterministic) Turing machines.

Definition 6 (Class $\mathbb{P}$) *A language $L \subset \Sigma^*$ is in $\mathbb{P}$ iff L is recognized by a family of circuits of polynomial size: there exists some polynomial p, with $size(C_n) = p(n)$ for all n.*

Class $\mathbb{P}$ is also known as $P/poly$, since it corresponds to polynomial time with a polynomial advice [7]. It is known to contain some non-computable sets, as well as to correspond to sets recognizable in polynomial time with a tally oracle [7]. It has been characterized as a natural class to characterize the computational power of several continuous space and time dynamical systems [46, 47, 48].

Class $\mathbb{P}$ corresponds to non-uniform polynomial time, since it consists in relaxing second condition in next characterization of polynomial time.

Proposition 1 (P versus $\mathbb{P}$ (see e.g. [42])) *A language $L \subset \Sigma^*$ is recognized in polynomial time by a Turing machine, iff*

1. *it is in $\mathbb{P}$;*
2. *the function that maps 1^n to the encoding of circuit C_n is computable in polynomial time.*

The following results are established in [11] (observe that languages recognized in polynomial time by rational PCD systems correspond precisely to P, that is polynomial time for Turing machines).

Theorem 4 (P(PCD Systems) = $\mathbb{P}$ [11])

- *Any $L \in \mathbb{P}$ is computed in polynomial discrete time by some (possibly non-rational) PCD system $\mathcal{H}$ over $[-1, 1]^3$.*
- *Any language L computed by some (possibly non-rational) PCD system $\mathcal{H}$ in polynomial discrete time is in $\mathbb{P}$.*

It is known that $\mathbb{P}$ contains some non-computable sets, and hence, PCD systems with non-rational coefficients are stronger than classical Turing Machines [11].

The extra-power comes from the power given by non-computable constants: this can actually be proved as follows: given some PCD system $\mathcal{H}$, we write $Constant(\mathcal{H})$ for the finitely many constants $\alpha_1, \ldots, \alpha_m$ involved in the description of the polyhedra P_i, as well as the finitely many constants $\beta_1, \ldots, \beta_m$ involved in the coordinates of the vectors $\mathbf{c}_i$, as well as all the finitely many products $\alpha_i\beta_j$.

Definition 7 (Computable PCD Systems) *A PCD system is said to have computable constants (denoted by $\mathcal{H} \in \mathcal{R}ec(\mathbb{R})$) if $Constant(\mathcal{H}) \subset \mathcal{R}ec(\mathbb{R})$.*

We will say that a language belongs to P/rec, if it belongs to $\mathbb{P}$, and the function that maps[8] 1^n to the encoding of circuit C_n is computable (observe that we

[8] or maps n, that would give the same definition.

do not say polynomial time computable, otherwise, this definition would correspond to polynomial time). Since circuit value (see e.g. [42]) is recursive, P/rec is a subset of recursive sets. Since there exist some functions non-computable in polynomial time, polynomial time is strictly included in P/rec, in turn strictly included in $\mathbb{P}$.

Theorem 5 (P(Computable PCD Systems)= P/rec) *We have*

- *Any language $L \in P/rec$ is computed in polynomial discrete time by some (possibly non-rational) PCD system $\mathcal{H}$ with computable constants over $[-1,1]^3$.*
- *Any language L computed by some (possibly non-rational) PCD system $\mathcal{H}$ with computable constants in polynomial discrete time is in P/rec.*

Proof: First item follows from the constructions of [11], observing that the constant encoding the advice used in [11] is actually computable in the sense of recursive analysis, as soon as the advice (or equivalently the family of circuits) is.

Now, for second item, we know that the language recognized by $\mathcal{H}$ is recognized by a family of circuits $(C_n)_{n\in\mathbb{N}}$ of polynomial size $p(n)$. Given n, by enumerating the finitely many circuits of size $p(n)$, one can compute such a circuit C_n as soon as one can test effectively whether a given circuit C agrees with $\mathcal{H}$ on the finitely many words w of length n.

From the dynamics of PCD systems, testing a given circuit C against $\mathcal{H}$ on some word w can be done effectively as soon as one has a way, given some row matrix L with rational coefficients and an rational b, to tell effectively whether $L\gamma \geq b$ (respectively $L\gamma > b$) or not, where $\gamma = (\gamma_1, \ldots, \gamma_l)$ is the vector of the constants of $Constant(\mathcal{H})$.

Replacing some constants by their expression if needed, we can furthermore assume that $\gamma_1, \ldots, \gamma_l$ are linearly independent over rational numbers (recall that $\mathcal{H}$, and hence γ is fixed).

Now, each test $L\gamma \geq b$ (respectively $L\gamma > b$) can be done as follows: Approximate $L\gamma$ by $x_n \in \mathbb{Q}$ with precision 2^{-n}, for increasing n until $n = n_0$ with $\|b - x_{n_0}\| > 2^{-n_0}$. Such an n_0 must exist, since $L\gamma \neq b$, otherwise $\gamma_1, \ldots, \gamma_l$ would not be linearly independent over rational numbers. Now, answer $L\gamma \geq b$ (resp. $L\gamma > b$) iff $x_{n_0} - 2^{-n_0} > b$.

□

Since P/rec is included in the set of recursive languages, we get.

Corollary 2 (P(PCD Systems) $\subset$ Recursive) *Any language computed by some (possibly non-rational) PCD system $\mathcal{H}$ with computable constants in polynomial discrete time is recursive.*

Simple generalizations of previous arguments show that this also holds for languages recognized in exponential discrete time.

Observe that previous arguments can be generalized to yield a whole structural complexity of the power of PCD systems according to their constants, similar[9] to the one that was obtained for neural networks in [6].

6 Imposing Smoothness

Following the constructions from [11], one can also impose to the PCD system to be smooth (the discrete time of the PCD system becomes a continuous time).

Theorem 6 ($\mathbb{P} \subset$ P(Smooth Systems) [11]) *Any language $L \in \mathbb{P}$ is computed by a $\mathcal{C}^\infty$ continuous time dynamical system $\mathcal{H}$ in polynomial continuous time over $[-1,1]^3$.*

Theorem 7 ($P/rec \subset$ P(Smooth and Computable Systems)) *Any language $L \in P/rec$ is computed by a $\mathcal{C}^\infty$ continuous time dynamical system $\mathcal{H}$ of $\mathcal{R}ec(\mathbb{R})$ in polynomial continuous time over $[-1,1]^3$.*

Theorem 8 ($P \subset$ P(Smooth and Poly. Computable Systems)) *Any language L recognized in polynomial time by a Turing machine is computed by a $\mathcal{C}^\infty$ continuous time dynamical system $\mathcal{H}$ of $P(\mathbb{R})$ in polynomial continuous time over $[-1,1]^3$.*

Conversely, one natural question is to understand whether it is possible to provide upper bounds on the power of computations of continuous time dynamical systems in polynomial continuous time.

Any sufficiently smooth system defined on a compact domain can be simulated by some numerical method: given some t, and n, one can estimate the position of a given trajectory at some time t with precision 2^{-n}. The point is that usual methods, such as Euler's method work only in a time that is proportional to some exponential in t. This also holds for most known numerical methods of fixed order: see for example the discussion in [50].

One may think that it is an intrinsic limitation of numerical methods, and hence that, potentially, continuous time dynamical systems could do things faster than Turing machines: see for example why Anastasios Vergis, Kenneth Steiglitz, and Bradley Dickinson in [55] avoided to take time as a natural resource in their discussion.

However, Warren Smith has recently demonstrated that it is possible to prove that time can be considered as a reasonable resource under some additional hypotheses: see [50].

Definition 8 (Polynomially Limited Variation (PLV) [50]) *A dynamical system is said to have a Polynomially Limited Variation if it is of class $\mathcal{C}^\infty$, and it is known that in any time interval $0 \leq t \leq T$, the absolute value of each component of f, of each component of $\phi^{(k)}$ for a trajectory ϕ, as well as the absolute*

[9] But different, since the model here is not really equivalent, and is more problematic. Mostly linear precision does not suffice here.

value of each partial derivative of f with respect to any of its arguments, having total differentiation-degree k, is similarly bounded, by bounds of type $(kT)^{O(k)}$.

Using Butcher's Runge-Kutta scheme, with an order taken as linearly dependent of T, Warren Smith proved:

Theorem 9 (PLV Implies Efficient Simulation [50]) *Any dynamical system of $P(\mathbb{R})$ with a Polynomially Limited Variation can be simulated numerically efficiently on a Turing machine: Given some initial condition of a trajectory ϕ in $P(\mathbb{R})$, the value of $\phi(t)$ at any time $0 \leq t \leq T$, can be computed accurate to precision ϵ, for any desired $\epsilon > 0$, in a time that depends only polynomially on T, and $\min(\epsilon, 1)^{-1/\max(1,T)}$.*

Remark: This does not imply that ϕ is in $P(\mathbb{R})$: for example it does not say that $\phi(t)$ can be computed in a time polynomial in $-\log(\epsilon)$. May this be exploited to compute faster using continuous dynamical systems, or can Smith's result be improved?
Remark: The conditions of Definition 8 seem to have some connections with the ideas of José Félix Costa and Jerzy Mycka in several papers about considering that polynomial time for continuous systems must be connected to Laplace transforms: see for e.g. [35, 37]. We think it would be interesting to better understand these relations.

With this result in hand, we claim:

Theorem 10 (P = P(Smooth and Poly. Comput. PLV Systems)) *The languages computed by continuous time dynamical system $\mathcal{H}$ of $P(\mathbb{R})$ with Polynomially Limited Variation in polynomial continuous time over $[-1, 1]^d$ do correspond precisely to languages recognized in polynomial time by Turing machines.*

Proof: If a language L is recognized by a continuous time dynamical system $\mathcal{H}$ of $P(\mathbb{R})$ with a Polynomially Limited Variation in polynomial time, then by simulating $\mathcal{H}$ using Theorem 9 (ϵ is fixed to $1/4$), a Turing machine can recognize L in polynomial time.

Conversely, we know that any language L recognized in polynomial time by a Turing machine can be computed in polynomial time by a PCD system. Using the ideas of [11], it can be smoothed to a $\mathcal{C}^\infty$ system: original regions of the PCD system doing computations are kept intact, and interpolation regions are added. On first regions, since all partial derivatives are 0, it is clear that the conditions of Definition 8 hold. Now, interpolation regions can be build using affine combinations of translations of the (integral of) classical function $g(x) = \exp(-1/x)$ for $x > 0$, 0 for $x \leq 0$, which is $\mathcal{C}^\infty$ on $\mathbb{R}$.

By using triangular inequality, linearity, and reasoning independently on each such region, we only need to prove that for all k, $g^{(k)}(x)$ can be bounded by a bound of type $(k)^{O(k)}$. Since for $x > 0$, $g^{(k)}(x) = P_k(1/x)\exp(-1/x)$, for some polynomial P_k of degree $2k$, using triangular inequality, and developing P_k into its at most $2k + 1$ monomials, we only need to prove that $h(x) = (1/x)^k \exp(-1/x)$ can be bounded by a bound of type $(k)^{O(k)}$.

Consider $\mu = 1/((k+1)K)$, with $K = k\ln(k+1)$. For $x \leq \mu$, we have $h(x) \leq (1/\mu)^k \exp(-1/\mu) \leq (k+1)^k \exp(Kk)\exp(-K(k+1)) \leq 1$. Now, we always have $\exp(-1/x) \leq 1$ for $x \geq 0$, so that for $x \geq \mu$, $1/x \leq (k+1)K$, and $h(x) \leq (k+1)^k K^k \leq O((k)^{3k})$.

□

Authorizing general computable functions, we get.

Theorem 11 (P/rec = P(Smooth and Computable PLV Systems)) *The languages computed by a continuous time dynamical system $\mathcal{H}$ of $\mathcal{R}ec(\mathbb{R})$ with Polynomially Limited Variation in polynomial continuous time over $[-1,1]^d$ do correspond precisely to languages of P/rec.*

If non-computable functions are authorized, we get.

Theorem 12 ($\mathbb{P}$ = P(Smooth and PLV Systems)) *The languages computed by a continuous time dynamical system $\mathcal{H}$ with Polynomially Limited Variation in polynomial continuous time over $[-1,1]^d$ do correspond precisely to languages of $\mathbb{P}$.*

Proof: Use same technique as before for converse direction: the only point is to see that the involved functions are in the claimed classes of recursive analysis.

For direct direction, observe that the proof of [50] gives a polynomial number of iterations for Butcher's Runge-Kutta scheme independently of the complexity of the function involved in the differential equation. Transform this polynomially many iterations into a circuit of polynomial size, feeded with sufficient approximations of the function as in [50], but relaxing the hypothesis that these approximations should be computable in polynomial time.

□

7 On Zeno's phenomenon

We now come back to PCD systems. To a finite continuous time, can correspond a non-finite discrete time: Consider for example, the maximal trajectory defined by the PCD system depicted on Figure 2.

This has already been observed in [4], and used to show that any arithmetical set can be recognized by a PCD system.

Actually, with the terminology of Definition 4, to any trajectory $\phi : \mathbb{R}^+ \to X$, is associated the set T_ϕ of the time $t_i \geq 0$ at which the direction of ϕ change. T_ϕ is easily shown to be a well-ordered set. As any well-ordered set, it must be isomorphic to some ordinal. This ordinal is considered as the discrete time of the trajectory in the general case.

Example: In Figure 2, the trajectory going from $(x,0)$ to $(0,0)$ has discrete time ω.

Actually, the discrete time of a finite continuous time trajectory can be bounded above according to the dimension.

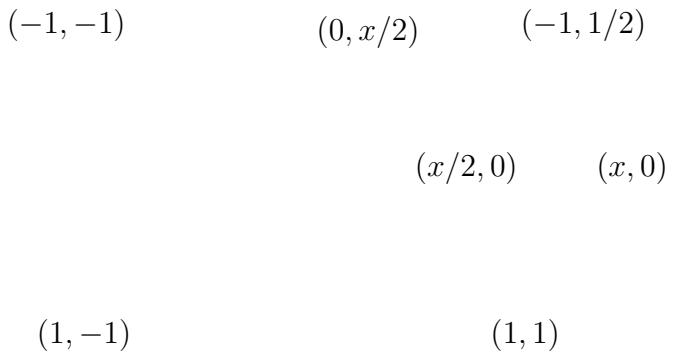

Figure 2: Zeno's paradox: A trajectory of continuous time $5x$ and discrete time ω between point $(x, 0)$ and point $(0, 0)$.

Theorem 13 (Discrete time vs Continuous Time [9, 10]) *Any trajectory ϕ of finite continuous time of a PCD system over $\mathbb{R}^d$, has a discrete time $T_d < \omega^{d-1}$ for $d \geq 3$. For $d = 2$, $T_d \leq \omega$.*

Recall that the hyper-arithmetical hierarchy is an extension of the arithmetical hierarchy to constructive ordinal numbers. It consists of the classes of languages $\Sigma_1, \Sigma_2, \ldots, \Sigma_k, \ldots, \Sigma_\omega, \Sigma_{\omega+1}, \Sigma_{\omega+2}, \ldots, \Sigma_{\omega 2}, \Sigma_{\omega 2+1}, \ldots, \Sigma_{\omega^2}, \ldots$ indexed by the constructive ordinal numbers. It is a strict hierarchy and it satisfies the strict inclusions $\Sigma_\alpha \subset \Sigma_\beta$ whenever $\alpha < \beta$. It can be related to the analytical hierarchy by $\Delta_1^1 = \cup_\beta \Sigma_\beta$: see [45].

Class Σ_1 is defined as the class of the recursively enumerable sets. When k is a constructive ordinal and when the class Σ_k is defined, Σ_{k+1} is defined as the class of the languages that are recursively enumerable in a set in Σ_k. When k is a constructive limit ordinal, $k = lim\, k_i$, and when the classes $(\Sigma_{k_i})_{i\in\mathbb{N}}$ are defined, Σ_k is defined as the class of the languages that are recursively enumerable in some fixed diagonalisation of classes $(\Sigma_{k_i})_i$: see [45] for full details.

It has been proved in [9, 10] that the power of PCD systems in finite continuous time can be characterized as follows (providing an extension of [4] to a full characterization of the power of PCD systems according to their dimension).

Theorem 14 (PCD Systems vs Hyper-Arithmetical Hierarchy [9, 10]) *The power of rational PCD systems in finite continuous time over $[-1, 1]^d$ or $\mathbb{R}^d$ can be characterized as follows:*

- *For $d = 2k + 3$, they recognize precisely the sets of Σ_{ω^k}.*
- *For $d = 2k + 4$, they recognize precisely the sets of Σ_{ω^k+1}.*

From Corollary 1, we see that such super-Turing phenomena for smooth and $\mathcal{R}ec(\mathbb{R})$ systems over $[-1,1]^d$ can not happen: they can always be simulated. It follows that there is no hope to "smooth" the considered systems in the theorem above.

8 On Robustness

Since the proofs of undecidability, or more generally of simulation of Turing machines, often involve to encode the configuration of a Turing machine (or of a two counter automata) into some real numbers, and since this require infinite precision, in the hybrid system verification community, a folklore conjecture appeared saying that this undecidability is due to non-stability, non-robustness, sensitivity to initial values of the systems, and that it never occurs in "real systems" [3, 20].

For example, Martin Fränzle writes in [21] "Hence, on simple information-theoretic grounds, the undecidability results thus obtained can be said to be artifacts of an overly idealized formalization. However, while this implies that the particular proof pattern sketched above lacks physical interpretation, it does not yield any insight as to whether the state reachability problem for hybrid systems featuring noise is decidable or no. We conjecture that there is a variety of realistic noise models for which the problem is indeed decidable".

There were several attempts to formalize and prove (or to disprove) this conjecture: it has been proved that small perturbations of the trajectory still yields undecidability [26]. Infinitesimal perturbations of the dynamics for a certain model of hybrid systems has shown to rise to decidability [21]. This has been extended to several models by [3].

Let us look at this latter result: they consider several classes of widely used models of dynamical systems: Turing machines, piecewise affine maps, linear hybrid automata, and piecewise constant derivative systems. For each of them is introduced a notion of "perturbed" dynamics and is studied the computational power of the corresponding perturbed systems. Perturbations are defined for each model using a notion of metrics on the state space. For a given model, given a transition system with a reachability relation R, the idea is to perturb the dynamic by a small ϵ, and then take (as the perturbed dynamics of the system) the limit (intersection) R_ω of the perturbed reachability relations as this ϵ tends to 0. In that setting, a system is said "robust" if its reachability relation does not change under small perturbations of the dynamics, i.e. R_ω is equal to R [3].

Eugene Asarin and Ahmed Bouajjani show:

Theorem 15 (Robustness [3]) *For Turing machines, piecewise affine maps, linear hybrid automata, and piecewise constant derivative systems, the relation R_ω belongs to the class Π^1_0 (it is co-recursively enumerable), and moreover, any Π^1_0 relation can be reduced to a relation R_ω of a perturbed system: any complement of a recursively enumerable set, can be semi-decided by an infinitesimally perturbed system.*

That means that any robust system has its reachability problem decidable.

Corollary 3 (Robustness implies Recursiveness for Some Systems [3]) *For Turing machines, piecewise affine maps, linear hybrid automata, and piecewise constant derivative systems, any language computed by a robust system is recursive.*

We think it worth also investigating which simulations of all the previous sections can be extended to be robust in the sense of [3], and in which sense.

9 Summary

In this paper, we have considered several models with respect to their super-Turing power.

The results can be summarized somehow by the following tables, for compact systems.

- When time is discrete time:

Class	Computability	Complexity
Rational PCD Systems	R.E. languages	P
Non-Rational $\mathcal{R}ec(\mathbb{R})$ PCD Systems	R.E. languages	P/rec
Non-Rational PCD Systems	All Languages	$\mathbb{P}$
PLV $P(\mathbb{R})$ Smooth Systems	R.E. languages	P
PLV $\mathcal{R}ec(\mathbb{R})$ Smooth Systems	R.E. languages	P/rec
PLV Smooth Systems	All Languages	$\mathbb{P}$
$\mathcal{R}ec(\mathbb{R})$ Smooth Systems	R.E. Languages	$\geq P/rec$
Smooth Systems	All Languages	$\geq \mathbb{P}$

- When time is continuous time:

Class	Computability = Complexity
Rational PCD Systems	Σ_{ω^k} in dimension $d = 2k + 3$
	Σ_{ω^k+1} in dimension $d = 2k + 4$

10 Thanks

I would like to thank Manuel Campagnolo, Johanne Cohen, Daniel Graça, Emmanuel Hainry, and Jean-Yves Marion for very interesting discussions and suggestions about this work or parts of this work.

This work has also benefited from several email exchanges with José Félix Costa, and from several past discussions with Eugene Asarin. Some results mentioned are extensions of results obtained in collaboration with Michel Cosnard.

References

[1] *Proceedings of the IEEE, Special Issue on "Hybrid Systems"*, 88(7), July 2000.

[2] Scott Aaronson. NP-complete problems and physical reality. *ACM SIGACT News*, 36(1), 2005.

[3] Eugene Asarin and Ahmed Bouajjani. Perturbed Turing machines and hybrid systems. In *Logic in Computer Science*, pages 269–278, 2001.

[4] Eugene Asarin and Oded Maler. Achilles and the tortoise climbing up the arithmetical hierarchy. *Journal of Computer and System Sciences*, 57(3):389–398, December 1998.

[5] Eugene Asarin, Oded Maler, and Amir Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science*, 138(1):35–65, February 1995.

[6] José L. Balcázar, Ricard Gavaldà, Hava T. Siegelmann, and Eduardo D. Sontag. Some structural complexity aspects of neural computation. In *8th IEEE Conference on Structure in Complexity Theory*, pages 253–256. IEEE Computer Society Press, 1993.

[7] José Luis Balcázar, Josep Diáz, and Joaquim Gabarró. *Structural Complexity I.* EATCS Monographs on Theoretical Computer Science, 1988.

[8] Udi Boker and Nachum Dershowitz. A formalization of the Church-Turing theorem. Submitted, 2005.

[9] Olivier Bournez. Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy. *Theoretical Computer Science*, 210(1):21–71, 6 January 1999.

[10] Olivier Bournez. *Complexité Algorithmique des Systèmes Dynamiques Continus et Hybrides.* PhD thesis, Ecole Normale Supérieure de Lyon, Janvier 1999.

[11] Olivier Bournez and Michel Cosnard. On the computational power of dynamical systems and hybrid systems. *Theoretical Computer Science*, 168(2):417–459, November 1996.

[12] Olivier Bournez and Emmanuel Hainry. Real recursive functions and real extentions of recursive functions. In *Machines, Computations and Universality (MCU'2004)*, volume 3354 of *Lecture Notes in Computer Science*, Spinger-Verlag, September 2004.

[13] Olivier Bournez and Emmanuel Hainry. Elementarily computable functions over the real numbers and $\mathbb{R}$-sub-recursive functions. *Theoretical Computer Science*, 2005. To appear.

[14] Manuel Campagnolo. Continuous time computation with restricted integration capabilities. *Theoretical Computer Science*, 317:147–165, 2004.

[15] Alonzo Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics,*, 58:345–363, 1936. Also in [19].

[16] Carol E. Cleland. The concept of computability. *Theoretical Computer Science*, 317(1–3):209–225, June 2004.

[17] B. Jack Copeland. The Church-Turing thesis. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Fall 2002. Available online at: *http://plato.stanford.edu/entries/church-turing/*.

[18] B. Jack Copeland and Richard Sylvan. Beyond the universal Turing machine. *Australasian Journal of Philosophy*, 77:46–66, January 27 1999.

[19] Martin Davis. *The undecidable*. Raven Press, 1965.

[20] John Foy. A dynamical system which must be stable whose stability cannot be proved. *Theoretical Computer Science*, 328:355–361, 2004.

[21] Martin Fränzle. Analysis of hybrid systems: An ounce of realism can save an infinity of states. In Jörg Flum and Mario Rodríguez-Artalejo, editors, *Computer Science Logic (CSL'99)*, volume 1683 of *Lecture Notes in Computer Science*, pages 126–140. Springer Verlag, 1999.

[22] Robin Gandy. Church's thesis and principles for mechanisms. *The Kleene Symposium*, pages 123–148, 1980.

[23] Kurt Gödel. Über formal unentscheidbare Satze der Principia Mathematica und verwandter Systeme I. *Monatschefte fur Mathematik und Physik*, 38:173–198, 1931. English translation in [19].

[24] Daniel Graça, Manuel Campagnolo, and Jorge Buescu. Robust simulations of Turing machines with analytic maps and flows. In B. Cooper, B. Loewe, and L. Torenvliet, editors, *Proceedings of CiE'05, New Computational Paradigms*, volume 3526 of *Lecture Notes in Computer Science*, pages 169–179, Springer-Verlag, 2005.

[25] Daniel S. Graça. Some recent developments on Shannon's general purpose analog computer. *Mathematical Logic Quarterly*, 50(4–5):473–485, 2004.

[26] Thomas Henzinger and Jean-Francois Raskin. Robust undecidability of timed and hybrid systems. *Hybrid systems: computation and control; Second International Workshop, HSCC'99, Berg en Dal, The Netherlands, March 29–31, 1999; proceedings*, 1569, 1999.

[27] Stephen C. Kleene. General recursive functions of natural numbers. *Mathematical Annals*, 112:727–742, 1936. Also in [19].

[28] Ker-I Ko. *Complexity Theory of Real Functions*. Progress in Theoretical Computer Science. Birkhäuser, Boston, 1991.

[29] Pascal Koiran, Michel Cosnard, and Max Garzon. Computability with low-dimensional dynamical systems. *Theoretical Computer Science*, 132(1-2):113–128, September 1994.

[30] Webb Miller. Recursive function theory and numerical analysis. *Journal of Computer and System Sciences*, 4(5):465–472, October 1970.

[31] Christopher Moore. Unpredictability and undecidability in dynamical systems. *Physical Review Letters*, 64(20):2354–2357, May 1990.

[32] Christopher Moore. Recursion theory on the reals and continuous-time computation. *Theoretical Computer Science*, 162(1):23–44, 5 August 1996.

[33] Christopher Moore. Finite-dimensional analog computers: Flows, maps, and recurrent neural networks. In Cristian S. Calude, John L. Casti, and Michael J. Dinneen, editors, *Unconventional Models of Computation (UMC'98)*. Springer, 1998.

[34] Jerzy Mycka. Analog computation beyond the Turing limit. *Applied Mathematics and Computation*, this issue.

[35] Jerzy Mycka and José Félix Costa. The $P \neq NP$ conjecture. Submitted.

[36] Jerzy Mycka and José Félix Costa. Undecidability over continuous-time. Submitted.

[37] Jerzy Mycka and José Félix Costa. The computational power of continuous dynamic systems. In *Machines, Computations and Universality (MCU'2004)*, volume 3354 of *Lecture Notes in Computer Science*, pages 163–174, Springer-Verlag, September 2004.

[38] Jerzy Mycka and José Félix Costa. Real recursive functions and their hierarchy. *Journal of Complexity*, 20(6):835–857, 2004.

[39] Jerzy Mycka and José Félix Costa. What lies beyond the mountains, computational systems beyond the Turing limit. *European Association for Theoretical Computer Science Bulletin*, 85:181–189, 2005.

[40] István Németi and Gyula David. Relativistic computers and the Turing barrier. *Applied Mathematics and Computation*, this issue.

[41] Toby Ord. The many forms of hypercomputation. *Applied Mathematics and Computation*, this issue.

[42] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[43] Marian Boykan Pour-El and J. Ian Richards. A computable ordinary differential equation which possesses no computable solution. *Annals of Mathematical Logic*, 17:61–90, 1979.

[44] Marian Boykan Pour-El and J. Ian Richards. The wave equation with computable inital data such that its unique solution is not computable. *Advances in Mathematics*, 39:215–239, 1981.

[45] Hartley Rogers Jr. *Theory of Recursive Functions and Effective Computability*. MIT Press, April 1987.

[46] Hava T. Siegelmann. Computation beyond the Turing limit. *Science*, 268:545–548, 1995.

[47] Hava T. Siegelmann. *Neural Networks and Analog Computation - Beyond the Turing Limit*. Birkauser, 1999.

[48] Hava T. Siegelmann and Eduardo D. Sontag. Analog computation via neural networks. *Theoretical Computer Science*, 131(2):331–360, September 1994.

[49] Hava T. Siegelmann and Eduardo D. Sontag. On the computational power of neural nets. *Journal of Computer and System Sciences*, 50(1):132–150, February 1995.

[50] Warren D. Smith. Church's thesis meets the N-body problem. *Applied Mathematics and Computation*, this issue.

[51] Warren D. Smith. History of "Church's theses" and a manifesto on converting physics into a rigorous algorithmic discipline. Technical report, NEC Research Institute, 1999. Avalaible on *http://www.math.temple.edu/ wds/homepage/works.html*.

[52] Warren D. Smith. On the uncomputability of hydrodynamics. Technical report, NEC Research Institute, 2003. Avalaible on *http://www.math.temple.edu/ wds/homepage/works.html*.

[53] Warren D. Smith. Three counterexamples refuting Kieu's plan for «quantum adiabatic hypercomputation» and some uncomputable quantum mechanical tasks. *Applied Mathematics and Computation*, this issue.

[54] Alan Turing. On computable numbers, with an application to the "Entscheidungsproblem". 42(2):230–265, 1936.

[55] Anastasios Vergis, Kenneth Steiglitz, and Bradley Dickinson. The complexity of analog computation. *Mathematics and Computers in Simulation*, 28(2):91–113, 1986.

[56] Klaus Weihrauch. *Computable Analysis*. Springer, 2000.

[57] Andrew Chi-Chih Yao. Classical physics and the Church–Turing Thesis. *Journal of the ACM*, 50(1):100–105, January 2003.