Upper and lower bounds on continuous-time computation

Manuel Lameiras Campagnolo[1] and Cristopher Moore[2,3,4]

[1] D.M./I.S.A., Universidade Técnica de Lisboa, Tapada da Ajuda, 1349-017 Lisboa, Portugal mlc@math.isa.utl.pt
[2] Computer Science Department, University of New Mexico, Albuquerque NM 87131
[3] Physics and Astronomy Department, University of New Mexico, Albuquerque NM 87131
[4] Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, New Mexico 87501 moore@santafe.edu

Abstract. We consider various extensions and modifications of Shannon's General Purpose Analog Computer, which is a model of computation by differential equations in continuous time. We show that several classical computation classes have natural analog counterparts, including the primitive recursive functions, the elementary functions, the levels of the Grzegorczyk hierarchy, and the arithmetical and analytical hierarchies.

Key words: Continuous-time computation, differential equations, recursion theory, dynamical systems, elementary functions, Grzegorczyk hierarchy, primitive recursive functions, computable functions, arithmetical and analytical hierarchies.

1 Introduction

The theory of analog computation, where the internal states of a computer are continuous rather than discrete, has enjoyed a recent resurgence of interest. This stems partly from a wider program of exploring alternative approaches to computation, such as quantum and DNA computation; partly as an idealization of numerical algorithms where real numbers can be thought of as quantities in themselves, rather than as strings of digits; and partly from a desire to use the tools of computation theory to better classify the variety of continuous dynamical systems we see in the world (or at least in its classical idealization).

However, in most recent work on analog computation (e.g. [BSS89,Mee93,Sie98,Moo98]) time is still discrete. Just as in standard computation theory, the machines are updated with each tick of a clock. If we are to make the states of a computer continuous, it makes sense to consider making its progress in time continuous too. While a few efforts have been made in the direction of studying computation by continuous-time dynamical systems [Moo90,Moo96,Orp97b,Orp97a,SF98,Bou99,Bou99b,CMC99,CMC00,BSF00], no particular set of definitions has become widely accepted, and the various models do not seem to be equivalent to each other. Thus analog computation has not yet experienced the unification that digital computation did through Turing's work in 1936.

In this paper, we take as our starting point Shannon's General Purpose Analog Computer (GPAC), a natural model of continuous-time computation defined in terms of differential equations. By extending it with various operators and oracles, we show that a number of classical computation classes have natural analog counterparts, including the primitive recursive and elementary functions, the levels of the Grzegorczyk hierarchy, and (if some physically unreasonable operators are allowed) the arithmetical and analytical hierarchies. We review recent results on these extensions, place them in a unified framework, and suggest directions for future research.

The paper is organized as follows. In Section 2 we review the standard computation classes over the natural numbers. In Section 3 we review Shannon's GPAC, and in Section 4 we show that a simple extension of it can compute all primitive recursive functions. In Section 5 we restrict the GPAC to linear differential equations, and show that this allows us to compute exactly the elementary functions, or the levels of the Grzegorczyk hierarchy if we allow a certain number of nonlinear differential equations as well. In Section 6 we show that allowing zero-finding on the reals yields much higher classes in the arithmetical and analytical hierarchies, and in Section 7 we conclude.

2 Recursive function classes over $\mathbb{N}$

In classical recursive function theory, where the inputs and outputs of functions are in the natural numbers $\mathbb{N}$, computation classes are often defined as the smallest set containing a basis of initial functions and closed under certain operations, which take one or more functions in the class and create new ones. Thus the set consists of all those functions that can be generated from the initial ones by applying these operations a finite number of times. Typical operations include (here $\boldsymbol{x}$ represents a vector of variables, which may be absent):

1. *Composition:* Given f and g, define $(f \circ g)(\boldsymbol{x}) = f(g(\boldsymbol{x}))$.
2. *Primitive recursion:* Given f and g of the appropriate arity, define h such that $h(\boldsymbol{x}, 0) = f(\boldsymbol{x})$ and $h(\boldsymbol{x}, y+1) = g(\boldsymbol{x}, y, h(\boldsymbol{x}, y))$.
3. *Iteration:* Given f, define h such that $h(\boldsymbol{x}, y) = f^{[y]}(\boldsymbol{x})$, where $f^{[0]}(\boldsymbol{x}) = \boldsymbol{x}$ and $f^{[y+1]}(\boldsymbol{x}) = f(f^{[y]}(\boldsymbol{x}))$.
4. *Limited recursion:* Given f, g and b, define h as in primitive recursion but only on the condition that $h(\boldsymbol{x}, y) \leq b(\boldsymbol{x}, y)$. Thus h is only allowed to grow as fast as another function already in the class.
5. *Bounded sum:* Given $f(\boldsymbol{x}, y)$, define $h(\boldsymbol{x}, y) = \sum_{z<y} f(\boldsymbol{x}, z)$.
6. *Bounded product:* Given $f(\boldsymbol{x}, y)$, define $h(\boldsymbol{x}, y) = \prod_{z<y} f(\boldsymbol{x}, z)$.
7. *Minimization* or *Zero-finding:* Given $f(\boldsymbol{x}, y)$, define $h(\boldsymbol{x}) = \mu_y f(\boldsymbol{x}, y)$ as the smallest y such that $f(\boldsymbol{x}, y) = 0$ provided that $f(\boldsymbol{x}, z)$ is defined for all $z \leq y$. If no such y exists, h is undefined.
8. *Bounded minimization:* Given $f(\boldsymbol{x}, y)$, define $h(\boldsymbol{x}, y_{\max})$ as the smallest $y < y_{\max}$ such that $f(\boldsymbol{x}, y) = 0$ and as $y_{\max}$ if no such y exists.

Note that minimization is the only one of these that can create a partial function; all the others yield total functions when applied to total functions. In bounded minimization, we only check for zeroes less than $y_{\max}$, and return $y_{\max}$ if we fail to find any.

Along with these operations, we will start with basis functions such as

1. The zero function, $\mathcal{O}(x) = 0$
2. The successor function, $\mathcal{S}(x) = x + 1$
3. The projections, $U_i^n(x_1, \ldots, x_n) = x_i$
4. Addition
5. Multiplication
6. Cut-off subtraction, $x \dot{-} y = x - y$ if $x \geq y$ and 0 if $x < y$

Then by starting with various basis sets and demanding closure under various properties, we can define the following classical complexity classes:

1. The *elementary* functions $\mathcal{E}$ are those that can be generated from zero, successor, projections, addition, and cut-off subtraction, using composition, bounded sum, and bounded product.
2. The *primitive recursive* functions $\mathcal{PR}$ are those that can be generated from zero, successor, and projections using composition and primitive recursion. We get the same class if we replace primitive recursion with iteration.
3. The *partial recursive* functions are those that can be generated from zero, successor, and projections using composition, primitive recursion and minimization.
4. The *recursive* functions are the partial recursive functions that are total.

A number of our results regard the class $\mathcal{E}$ of elementary functions, which was introduced by Kálmar [Kál43]. For example, multiplication and exponentiation over $\mathbb{N}$ are both in $\mathcal{E}$, since they can be written as bounded sums and products respectively: $xy = \sum_{z<y} x$ and $x^y = \prod_{z<y} x$. Since $\mathcal{E}$ is closed under composition, for each m the m-times iterated exponential $\exp^{[m]}(x)$ is in $\mathcal{E}$, where $\exp^{[0]}(x) = x$ and $\exp^{[m+1]}(x) = 2^{\exp^{[m]}(x)}$. In fact, these are the fastest-growing functions in $\mathcal{E}$, in the sense that no elementary function can grow faster than $\exp^{[m]}$ for some fixed m. The following bound will be useful to us below [Cut80]:

Proposition 1 *If $f \in \mathcal{E}$, there is a number m such that, for all $\boldsymbol{x}$, $f(\boldsymbol{x}) \leq \exp^{[m]}(\|\boldsymbol{x}\|)$ where $\|\boldsymbol{x}\| = \max_i x_i$.*

The elementary functions also correspond to a natural time-complexity class:

Proposition 2 *The elementary functions are exactly the functions computable by a Turing machine in elementary time, or equivalently in time bounded by $\exp^{[m]}(|\boldsymbol{x}|)$ for some fixed m.*

The class $\mathcal{E}$ is therefore very large, and many would argue that it contains all *practically* computable functions. It includes, for instance, the connectives of propositional calculus, functions for coding and decoding sequences of natural numbers such as the prime numbers and factorizations, and most of the useful number-theoretic and metamathematical functions. It is also closed under limited recursion and bounded minimization [Cut80,Ros84].

However, $\mathcal{E}$ does not contain all recursive functions, or even all primitive recursive ones. For instance, Proposition 1 shows that it does not contain the iterated exponential $\exp^{[m]}(x)$ where the number of iterations m is a variable, since any function in $\mathcal{E}$ has an upper bound where m is fixed. To include such functions, we need to include the higher levels of the *Grzegorczyk hierarchy* [Grz53,Ros84]. This hierarchy was used as an early stratification of the primitive recursive functions according to their computational complexity:

Definition 3 (The Grzegorczyk hierarchy) *Let $\mathcal{E}^0$ denote the smallest class containing zero, the successor function, and the projections, and which is closed under composition and limited recursion. Let $\mathcal{E}^{n+1}$ be defined similarly, except with the function E_n added to the list of initial functions, where E_n is defined as follows:*

$$\begin{aligned} E_0(x,y) &= x+y \\ E_1(x) &= x^2+2 \\ E_{n+1}(x) &= E_n^{[x]}(2) \end{aligned}$$

where by $f^{[x]}$ we mean f iterated x times.

The functions E_n are, essentially, repeated iterations of the successor function, and each one grows qualitatively more quickly than the previous one. $E_1(x)$ grows quadratically, and composing it with itself produces functions that grow as fast as any polynomial. $E_2(x)$ grows roughly as 2^{2^x}, and composing it yields functions as large as $\exp^{[m]}$ for any fixed m. $E_3(x)$ grows roughly as $\exp^{[2x]}(2)$, and so on. (These somewhat awkward definitions of E_0 and E_1 are the historical ones.)

We will use the fact that for $n \geq 3$, we can replace limited recursion in the definition of $\mathcal{E}^n$ with bounded sum and bounded product [Ros84]:

Proposition 4 *For $n \geq 3$, $\mathcal{E}^n$ is the smallest class containing zero, successor, the projections, cut-off subtraction, and E_{n-1}, which is closed under composition, bounded sum, and bounded product.*

One consequence of this is that the elementary functions are simply the third level of the Grzegorczyk hierarchy [Ros84], i.e. $\mathcal{E} = \mathcal{E}^3$. Moreover, the union of all the levels of the Grzegorczyk hierarchy is simply the class $\mathcal{PR}$ of primitive recursive functions:

Proposition 5 $\mathcal{PR} = \cup_n \mathcal{E}^n$.

It is known that the class of primitive recursive functions can be defined using iteration instead of primitive recursion [Odi89, p.72]. This means that iteration cannot be used freely in the Grzegorczyk hierarchy. Rather, as the definitions suggest, iteration moves a function one level up. As a matter of fact, iteration of E_{n-1} for a *fixed* number of times gives a bound on any function in $\mathcal{E}^n$, but unbounded iteration of E_{n-1} defines E_n and generates precisely $\mathcal{E}^{n+1}$. In this sense, the Grzegorczyk hierarchy stratifies the primitive recursive functions according to how many levels of iteration are needed to define them, or equivalently how many nested FOR-loops are required to compute them in a simplified programming language.

3 Differential equations, differentially algebraic functions and Shannon's GPAC

An ordinary differential equation of order n is an equation of the form

$$F(x, \boldsymbol{y}(x), \boldsymbol{y}'(x), \ldots \boldsymbol{y}^{(n)}(x)) = 0.$$

If F is a polynomial this equation is called *differentially algebraic* (d.a.) and its solutions are called differentially algebraic functions. The set of d.a. functions includes the polynomials, e^x, and trigonometric functions, as well as sums, products, compositions and solutions of differential equations formed from these such as $f' = \sin f$. Examples of functions which are not d.a. include Euler's Γ function and Riemann's ζ function [Rub89b]

The General Purpose Analog Computer (GPAC) is a simple model of a computer evolving in continuous time. It was originally defined as a mathematical model of an analog device, the Differential Analyser, the fundamental principles of which were described first by Lord Kelvin in 1876 [Kel76] and later by Vannevar Bush [Bow96]. The outputs are generated from the inputs by means of a dependence defined by a finite directed graph (not necessarily acyclic) where each node is either an adder, a unit that outputs the sum of its inputs, or an integrator, a unit with two inputs u and v that outputs the Riemann-Stieltjes integral $\int u\,dv$. These components are used to form circuits like the one in Figure 1, which calculates the function $\sin t$.

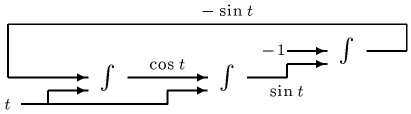

Fig. 1. A simple GPAC circuit that calculates $\sin t$. Its initial conditions are $\sin(0) = 0$ and $\cos(0) = 1$. The output w of the integrator unit $\int$ obeys $dw = u\,dv$ where u and v are its upper and lower inputs respectively.

Shannon [Sha41] showed that the class of functions generable in this abstract model is the set of solutions of systems of the following system of quasilinear differential equations,

$$A(\boldsymbol{x}, \boldsymbol{y})\,\boldsymbol{y}' = B(\boldsymbol{x}, \boldsymbol{y}), \tag{1}$$

satisfying some initial condition $\boldsymbol{y}(\boldsymbol{x}_0) = \boldsymbol{y}_0$. Here A and B are $n \times n$ and $m \times m$ matrices linear in 1 and the variables $x_1, ..., x_m$, $y_1, ..., y_n$, and $\boldsymbol{y}'$ is the $n \times m$ matrix of the derivatives of $\boldsymbol{y}$ with respect to $\boldsymbol{x}$. Later, Pour-El [PE74] made this definition more precise by requiring the solution to be unique for all initial values belonging to a closed set with non-empty interior called the *domain of generation* of the initial condition. We call the set of such solutions of (1) the class of GPAC-computable functions.

The following fundamental result [Sha41,PE74,LR87] establishes that the GPAC-computable functions essentially coincide with the differentially algebraic ones:

Proposition 6 (Shannon, Pour-El, Lipshitz, Rubel) *Let I and J be closed intervals of $\mathbb{R}$. If y is GPAC-computable on I then there is a closed subinterval $I' \subset I$ and a polynomial $P(x, y, y', ..., y^{(n)})$ such that $P = 0$ on I'. If $y(x)$ is the unique solution of $P(x, y, y', ..., y^{(n)}) = 0$ satisfying a certain initial solution on J then there is a closed subinterval $J' \subset J$ on which $y(x)$ is GPAC-computable.*

We will use $\mathcal{G}$ to denote the class of GPAC-computable functions, or equivalently the class of d.a. functions.

Now we show that $\mathcal{G}$ lacks an important closure property: it is not closed under iteration. The proof relies on a result of differential algebra on the iterated exponential function $\exp^{[n]}(x)$ defined by $\exp^{[0]}(x) = x$ and $\exp^{[n]}(x) = e^{\exp^{[n-1]}(x)}$. The following lemma follows from a more general theorem of Babakhanian [Bab73]:

Lemma 7 *For $n \geq 0$, $\exp^{[n]}(x)$ satisfies no non-trivial algebraic differential equation of order less than n.*

Proposition 6, Lemma 7 and our previous remarks are combined in [CMC99] to prove that:

Proposition 8 *The class $\mathcal{G}$ is not closed under iteration. Specifically, there is no GPAC-computable function $F(x, n)$ of two variables that matches the iterated exponential $\exp^{[n]}(x)$ for integer values of n.*

In the next section, we will show that while the GPAC is not closed under iteration, a natural extension of it is, and that this extension therefore includes all the primitive recursive functions.

4 Extending the GPAC

In analogy with oracles in classical computation theory, we can ask what functions become GPAC-computable if we add one or more additional basis functions φ. In terms of Shannon's circuit model, what things become GPAC-computable when we have "black boxes" that compute φ, which we can plug in to our circuit along with integrators and adders? We will refer to the resulting class as $\mathcal{G} + \varphi$.

One such extension explored in [CMC99] is the family of functions $\theta_k(x) = x^k \theta(x)$, where $\theta(x)$ is the Heaviside step function

$$\theta(x) = \begin{cases} 1 \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases}$$

For each k, we can think of $\theta_k(x)$ as a $(k-1)$-times differentiable way of testing whether $x \geq 0$. We claim that this is a physically realistic way to allow our computer to sense inequalities without introducing discontinuities.

In addition, we can show that allowing those functions is equivalent to relaxing slightly the definition of GPAC by solving first-order differential equations with two boundary values instead of just an initial condition. Thus $\mathcal{G} + \theta_k$ is a natural extension of the GPAC. Specifically,

Definition 9 *The function $y = f(x)$ belongs to the class θ if it is the unique solution on $I = [x_1, x_2] \subset \mathbb{R}$ of the differential equation $(a_0 + a_1 x + a_2 y)\, y' = b_0 + b_1 x + b_2 y$ with boundary values $y(x_1) = y_1$ and $y(x_2) = y_2$.*

For instance, the differential equation $xy' = 2y$ with boundary values $y(1) = 1$ and $y(-1) = 0$ has a unique solution on $I = [-1, 1]$, namely $y = x^2\theta(x)$. Then, we can prove [CMC99] that adding any function in θ to the set of basis functions of $\mathcal{G}$ is equivalent to adding θ_k for some k:

Proposition 10 *For any $\varphi \in \theta - \mathcal{G}$ there is a k such that $\mathcal{G} + \varphi = \mathcal{G} + \theta_k$.*

The main property of $\mathcal{G} + \theta_k$ we show here is the following:

Proposition 11 *$\mathcal{G} + \theta_k$ is closed under iteration for any $k > 1$. That is, if f of arity n belongs to $\mathcal{G} + \theta_k$ then there exists a function F of arity $n+1$ also in $\mathcal{G} + \theta_k$, such that $F(\boldsymbol{x}, t) = f^t(\boldsymbol{x})$ for $t \in \mathbb{N}$.*

The proof is constructive. To iterate a function we use a pair of "clock" functions to control the evolution of two "simulation" variables, similar to the approach in [Bra95,Moo96]. Both simulation variables have the same value $\boldsymbol{x}$ at $t = 0$. The first variable is iterated during half of a unit period while the second remains constant (its derivative is kept at zero by the corresponding clock function). Then, the first variable remains steady during the following half unit period and the second variable is brought up to match it. Therefore, at time $t = 1$ both variables have the same value $f(\boldsymbol{x})$. This process is repeated until the desired number of iterations is obtained.

If we denote the simulation variables by y_1 and y_2, and the clock functions by $\theta_k(\sin 2\pi t)$ and $\theta_k(-\sin 2\pi t)$, then the function that iterates f is the unique solution of:

$$\begin{aligned} |\cos \pi t|^{k+1}\, y_1' &= -2\pi(y_1 - f(y_2))\, \theta_k(\sin 2\pi t)\, \theta_k(t) \\ |\sin \pi t|^{k+1}\, y_2' &= -2\pi(y_2 - y_1)\, \theta_k(-\sin 2\pi t)\, \theta_k(t) \end{aligned} \tag{2}$$

where $|x|^k$ can defined in $\mathcal{G} + \theta_k$ as $|x|^k = \theta_k(x) + \theta_k(-x)$.

For general k, the proof that $y_1(t) = f^{[t]}(\boldsymbol{x})$ relies on the local behavior of Equation (2) in the neighborhood of $x = t$ and $x = t + 1$ for $t \in \mathbb{N}$. For instance, as $t \to 1$ from below, (2) becomes

$$\epsilon\, y_1' = -2^{k+1}(y_1 - f(y_2))$$

to first order in $\epsilon = 1 - t$. The solution of this is

$$y_1(\epsilon) = C\epsilon^{2^{k+1}} + f(y_2)$$

for constant C, and y_1 rapidly approaches $f(y_2)$ no matter where it starts on the real line. Similarly, y_2 rapidly approaches y_1 as $t \to 2$, and so on, so for any integer $t > 1$, $y_1(t) = y_2(t) = f^{[t]}(\boldsymbol{x})$. This shows that $F(\boldsymbol{x}, t) = y_1(t)$ can be defined in $\mathcal{G} + \theta_k$, so $\mathcal{G} + \theta_k$ is closed under iteration. Details can be found in [CMC99].

As an example, in Figure 2 we iterate the exponential function, which as we pointed out in Proposition 8 cannot be done in $\mathcal{G}$. Note that this is a numerical integration of Eq. (2) using a standard package (Mathematica), so this system of differential equations actually works in practice.

Let us set the convention that $\mathcal{G} + \theta_k$ contains a function on $\mathbb{N}$ if it contains some extension of it to $\mathbb{R}$. Since $\mathcal{G} + \theta_k$ contains zero, successor, and projections, and is closed under composition and iteration, it follows that:

Proposition 12 *$\mathcal{G} + \theta_k$ contains all primitive recursive functions.*

In fact, it is known that flows in three dimensions, or iterated functions in two, can simulate arbitrary Turing machines. In two dimensions, these functions can be infinitely differentiable [Moo90], piecewise-linear [Moo90,KCG94], or closed-form analytic and composed of a finite number of trigonometric terms [KM99]. (In [KM99] a simulation in one dimension is achieved at the cost of an exponential slowdown.) However, Proposition 12 is in some sense more elegant than these constructions, since it uses the operators of recursion theory directly instead of relying on a particular simulation or encoding of a Turing machine.

Furthermore, since for any Turing machine $\mathcal{M}$, the function $F(\boldsymbol{x}, t)$ that gives the output of $\mathcal{M}$ on input $\boldsymbol{x}$ after t steps is primitive recursive, and since $\mathcal{G} + \theta_k$ is closed under composition, we can say that $\mathcal{G} + \theta$ is *closed under time complexity* in the following sense:

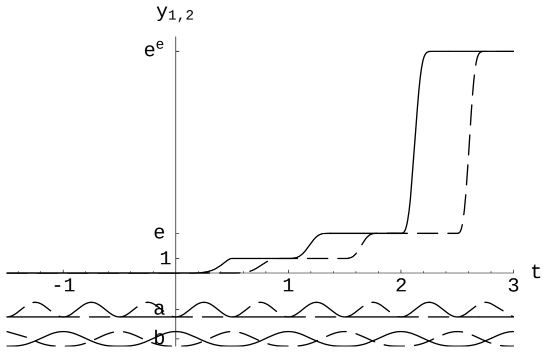

Fig. 2. A numerical integration on $[-1.5, 3]$ of the system of equations (2) for iterating the exponential function. Here $k = 2$. The values of y_1 and y_2 at $t = 0, 1, 2, 3$ are 0, 1, e, and e^e respectively. On the graph below we show (a) the clock functions $\theta_2(\sin(2\pi t))$, $\theta_2(\sin(-2\pi t))$ and (b) the functions $|\cos \pi t|^3$, $|\sin \pi t|^3$. Note that the term $\theta_k(t)$ on the right of Equation (2) assures that $y_1(t) = y_2(t) = 0$ for all $t < 0$ and, therefore, that the solution is unique on $\mathbb{R}$.

Proposition 13 *If a Turing machine $\mathcal{M}$ computes the function $h(\boldsymbol{x})$ in time bounded by $T(\boldsymbol{x})$, with T in $\mathcal{G}+\theta_k$, then h belongs to $\mathcal{G} + \theta_k$.*

Since any function computable in primitive recursive time is primitive recursive, Proposition 13 alone does not show that $\mathcal{G} + \theta_k$ contains any non-primitive recursive functions on the integers. However, if $\mathcal{G} + \theta_k$ contains a function such as the Ackermann function which grows more quickly than any primitive recursive function, this proposition shows that $\mathcal{G} + \theta_k$ contains many other non-primitive recursive functions as well.

It is believed, but not known [Hay96], that all differentially algebraic functions on the complex plane are bounded by some elementary function, i.e. $\exp^{[n]}(x)$ for some n, whenever they are defined for all $x > 0$. For real solutions of d.a. equations the conjecture is known to be false due to a theorem of Vijayaraghavan [Vij32,BBV37,Ban75]. However, the examples of d.a. functions that grow arbitrarily quickly are solutions of equations whose parameters are defined by limit processes, and this gives rise to non-primitive recursive constants. If we restrict ourselves to a model where the GPAC only has access to rational constants in its initial conditions and parameters, we believe the following is true:

Conjecture 14 *Functions $f(x)$ in $\mathcal{G} + \theta_k$ have primitive recursive upper bounds whenever they are defined for all $x > 0$, if the parameters and initial values of their defining differential equations are rational.*

We might try proving this conjecture by using numerical integration to approximate GPAC-computable functions with recursive ones. However, strictly speaking this approximation only works when a bound on the derivatives is known *a priori* [VSD86] or on arbitrarily small domains [Rub89]. If this conjecture is false, then Proposition 13 shows that $\mathcal{G} + \theta_k$ contains a wide variety of non-primitive recursive functions.

We close this section by noting that since all functions in $\mathcal{G} + \theta_k$ are $(k-1)$-times continously differentiable, $\mathcal{G} + \theta_k$ is a near-minimal departure from analyticity. In fact, if we wish to sense inequalities in an infinitely-differentiable way, we can add a C^∞ function such as $\theta_\infty(x) = e^{-1/x}\theta(x)$ to $\mathcal{G}$ and get the same results. The most general version of Proposition 11 is the following:

Proposition 15 *If $\varphi(x)$ has the property that it coincides with an analytic function $f(x)$ over an open interval (a, b), but that $\int_b^c (\varphi(x) - f(x))\, dx \neq 0$ for some $c > b$, then $\mathcal{G} + \varphi$ is closed under iteration and contains all the primitive recursive functions.*

We prove this by replacing $\theta_k(x)$ with $\varphi(x + b) - f(x + b)$, and we leave the details to the reader. Thus any departure from analyticity over an open interval creates a system powerful enough to contain all of $\mathcal{PR}$.

5 Linear differential equations, elementary functions and the Grzegorczyk hierarchy

In this section, we show that restricting the kind of differential equations we allow the GPAC to solve yields various subclasses of the primitive recursive functions: namely, the elementary functions $\mathcal{E}$ and the levels $\mathcal{E}^n$ of the Grzegorczyk hierarchy.

Let us first look at the special case of linear differential equations. If a first-order ordinary differential equation can be written as

$$\boldsymbol{y}'(x) = A(x)\,\boldsymbol{y}(x) + \boldsymbol{b}(x), \tag{3}$$

where $A(x)$ is a $n \times n$ matrix whose entries are functions of x, and $\boldsymbol{b}(x)$ is a vector of functions of x, then it is called a first-order linear differential equation. If $\boldsymbol{b}(x) = 0$ we say that the system is homogeneous. We can reduce a non-homogeneous system to a homogeneous one by introducing an auxiliary variable.

The fundamental existence theorem for differential equations guarantees the existence and uniqueness of a solution in a certain neighborhood of an initial condition for the system $\boldsymbol{y}' = f(\boldsymbol{y})$ when f is Lipshitz. For linear differential equations, we can strengthen this to global existence whenever $A(x)$ is continuous, and establish a bound on $\boldsymbol{y}$ that depends on $\|A(x)\|$:

Proposition 16 ([Arn96]) *If $A(x)$ is defined and continuous on an interval $I = [a, b]$ where $a \leq 0 \leq b$, then the solution of a homogeneous linear differential equation with initial condition $\boldsymbol{y}(0) = \boldsymbol{y}_0$ is defined and unique on I. Furthermore, if $A(x)$ is increasing then this solution satisfies*

$$\|\boldsymbol{y}(x)\| \leq \|\boldsymbol{y}_0\|\, e^{\|A(x)\|\,x}. \tag{4}$$

Given functions f and g, we can form the function h such that $h(\boldsymbol{x}, 0) = f(\boldsymbol{x})$ and $\partial_y h(\boldsymbol{x}, y) = g(\boldsymbol{x}, y)\, h(\boldsymbol{x}, y)$. We call this operation *linear integration*, and write $h = f + \int gh\, dy$ as shorthand. Then we can define an analog class $\mathcal{L}$ which is closed under composition and linear integration. As before, we cam define classes $\mathcal{L} + \varphi$ by allowing additional basis functions φ as well. Specifically, we will consider the class $\mathcal{L} + \theta_k$:

Definition 17 *A function $h : \mathbb{R}^m \to \mathbb{R}^n$ belongs to $\mathcal{L} + \theta_k$ if its components can be inductively defined from the constants 0, 1, -1, and π, the projections, and θ_k, using composition and linear integration.*

The reader will note that we are including π as a fundamental constant. We will need this for Lemma 21. We have not found a way to derive π from linear differential equations alone; perhaps the reader can find a way to do this, or a proof that we cannot. (Since π can easily be generated in $\mathcal{G}$, we have $\mathcal{L} + \theta_k \subseteq \mathcal{G} + \theta_k$.)

We wish to show that for any $k > 2$, $\mathcal{L} + \theta_k$ is an analog characterization of the elementary functions. First, note that by Proposition 16 all functions in $\mathcal{L} + \theta_k$ are total. In addition, their growth is bounded by a finitely iterated exponential, $\exp^{[m]}$ for some m. The following is proved in [CMC00], using the fact that if f and g are bounded by a finite tower of exponentials then their composition and linear integration $h = f + \int gh\, dy$ as well:

Proposition 18 *Let h be a function in $\mathcal{L} + \theta_k$ of arity m. Then there is a constant d and constants A, B, C, D such that, for all $\boldsymbol{x} \in \mathbb{R}^m$,*

$$\begin{aligned} \|h(\boldsymbol{x})\| &\leq A \exp^{[d]}(B\|\boldsymbol{x}\|) \\ \|\partial_{x_i} h(\boldsymbol{x})\| &\leq C \exp^{[d]}(D\|\boldsymbol{x}\|) \text{ for all } i = 1, \ldots, m \end{aligned}$$

where $\|\boldsymbol{x}\| = \max_i |x_i|$.

Note the analogy with Proposition 1 for elementary functions. In fact, we will now show that the relationship between $\mathcal{E}$ and $\mathcal{L} + \theta_k$ is very tight: all functions in $\mathcal{L} + \theta_k$ can be approximated by elementary functions, and all elementary functions have extensions to the reals in $\mathcal{L} + \theta_k$.

We say that a function over the reals is computable if it fulfills Grzegorczyk and Lacombe's, or equivalently, Pour-El and Richards' definition of computable continuous real function [Grz55,Grz57,Lac55,PR89]. Furthermore, we say that it is elementary computable if the corresponding functional is elementary, according to the definition proposed by Grzegorczyk or Zhou [Grz55,Zho97]. Conversely, as in the previous section we say that $\mathcal{L} + \theta_k$ contains a function on $\mathbb{N}$ if it contains some extension of it to the reals.

First, it is possible to approximate effectively any function in $\mathcal{L} + \theta_k$ in elementary time. Proposition 2 implies then that the discrete approximation is an elementary function as well. The constructive inductive proof is given in [CMC00] and is based on numerical techniques to integrate any function definable in $\mathcal{L} + \theta_k$. The elementary bound on the time complexity of numerical integration follows from Proposition 18. Thus:

Proposition 19 *If f belongs to $\mathcal{L} + \theta_k$ for any $k > 2$, then f is elementarily computable.*

Moreover, we can approximate any $\mathcal{L}+\theta_k$ function that sends integers to integers to error less than $1/2$ and obtain its value exactly in elementary time:

Proposition 20 *If a function $f \in \mathcal{L}+\theta_k$ is an extension of a function $\tilde{f}:\mathbb{N}\to\mathbb{N}$, then $\tilde{f}$ is elementary.*

We can also show the converse of this, i.e. that $\mathcal{L}+\theta_k$ contains all elementary functions, or rather, extensions of them to the reals.

First, we show that $\mathcal{L}+\theta_k$ contains (extensions to the reals of) the basis functions of $\mathcal{E}$. Successor and addition are easy to generate in $\mathcal{L}$. So are $\sin x$, $\cos x$ and e^x, since each of these are solutions of simple linear differential equations, and arbitrarily rational constants as shown in [CMC00]. With θ_k we can define cut-off subtraction $x \dot{-} y$ as follows. We first define a function $s(z)$ such that $s(z)=0$ when $z \le 0$ and $s(z)=1$ when $z \ge 1$, for all $z \in \mathbb{Z}$. This can be done in $\mathcal{L}+\theta_k$ by setting $s(0)=0$ and $\partial_z s(z) = c_k\theta_k(z(1-z))$, where $c_k = 1/\int_0^1 z^k(1-z)^k\,dz$ is a rational constant depending on k. Then $x \dot{-} y = (x-y)\,s(x-y)$ is an extension to the reals of cut-off subtraction.

Now, we just have to show that $\mathcal{L}+\theta_k$ has the same closure properties as $\mathcal{E}$, namely the ability to form bounded sums and products.

Lemma 21 *Let f be a function on $\mathbb{N}$ and let g be the function on $\mathbb{N}$ defined from f by bounded sum or bounded product. If f has an extension to the reals in $\mathcal{L}+\theta_k$ then g does also.*

First of all, for any $f \in \mathcal{L}+\theta_k$ there is a function $F \in \mathcal{L}+\theta_k$ that matches f on the integers, and whose values are constant on the interval $[j, j+1/2]$ for integer j [CMC00]. Then the bounded sum of f is then easily defined in $\mathcal{L}+\theta_k$ by linear integration. Simply write $g(0)=0$ and $g'(t) = c_k F(t)\,\theta_k(\sin 2\pi t)$, where c_k is a constant definable in $\mathcal{L}+\theta_k$. Then $g(t) = \sum_{z<n} f(z)$ whenever $t \in [n-1/2, n]$.

Defining the bounded product $g_n = \prod_{j<n} f_j$ of f in $\mathcal{L}+\theta_k$ is more difficult. We can approximate the iteration $g_{j+1} = g_j f_j$ using synchronized clock functions as in proof of Proposition 11. However, since the model we propose here only allows linear integration, the simulated functions cannot coincide exactly with the bounded product. Nevertheless, we can define a sufficiently close approximation because f and g have bounded growth by Proposition 18. Then since f and g have integer values, the accumulated error on $[0,n]$ resulting from this approximation can be removed with a suitable continuous step function ϕ definable in $\mathcal{L}+\theta_k$. The function ϕ is such that $\phi(t)=j$ if $t \in [j-1/4, j+1/4]$ for all integer j and so, ϕ returns the integer closest to t as long as the error is $1/4$ or less.

If we define a two-component function $\boldsymbol{y}(\tau,t)$ where $y_1(\tau,0)=y_2(\tau,0)=1$,

$$\begin{aligned} \partial_t y_1 &= (y_2 F(t) - y_1)\, c_k\theta_k(\sin 2\pi t)\,\beta(\tau) \\ \partial_t y_2 &= (y_1 - y_2)\, c_k\theta_k(-\sin 2\pi t)\,\beta(\tau) \end{aligned} \tag{5}$$

and $\beta(\tau)$ is an increasing function of τ, then $g_n = \phi(y_1(n,n))$. We can show that if β grows fast enough (roughly as fast as the bound on f given in Proposition 18), then by setting $\tau = n$ we can make the approximation error $|y_1(n,n) - g_n|$ as small as we like, and then remove it with ϕ. Note that the system 5 is linear in y_1 and y_2. Details are given in [CMC00].

We illustrate this construction in Figure 3. We approximate the bounded product of the identity function, i.e. the factorial $(n-1)! = \prod_{j<n} j$. As before, we numerically integrated Equation (5) using a standard package.

We do not know whether $\mathcal{L}+\theta_k$ is closed under bounded product for functions with real, rather than integer, values. We conjecture that it is not, but we have no proof of this. In any case, we have proved that:

Proposition 22 *If f is an elementary function, then $\mathcal{L}+\theta_k$ contains an extension of f to the reals.*

Taken together, Propositions 19, 20 and 22 show that the analog class $\mathcal{L}+\theta_k$ corresponds to the elementary functions in a natural way. It is interesting that linear integration alone gives extensions to the reals of all elementary functions, since these are all the functions that can be computed by any practically conceivable digital device. In terms of dynamical systems, $\mathcal{L}+\theta_k$ corresponds to cascades of finite depth, each level of which depends linearly on its own variables and the output of the level before it. We find it surprising that such systems, as opposed to highly non-linear ones, have so much computational power.

Next, we will extend the above results to the higher levels of the Grzegorczyk hierarchy, $\mathcal{E}^n$ for $n \ge 3$, by allowing the GPAC to solve a certain number of nonlinear differential equations.

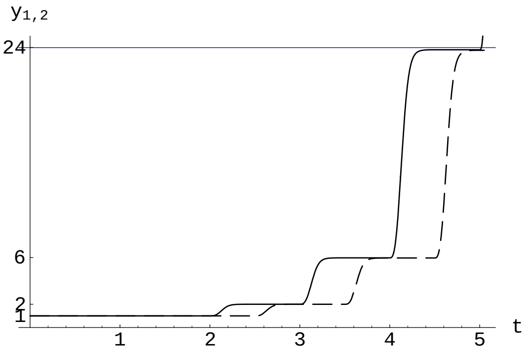

Fig. 3. A numerical integration of Equation (5), where f is a $\mathcal{L}+\theta_k$ function such that $f(0)=1$ and $f(x)=x$ for $x\geq 1$. Here, $k=2$. We obtain an approximation of an extension to the reals of the factorial function. In this example, where we chose a small $\tau<4$, the approximation is just sufficient to remove the error with ϕ and obtain exactly $\prod_{n<5} n = 4! = \phi(y_1(5))$.

Definition 23 (The hierarchy $\mathcal{G}_n+\theta_k$) *Let $\mathcal{G}_3+\theta_k=\mathcal{L}+\theta_k$ be the smallest class containing the constants 0, 1, -1, and π, the projections, and θ_k, which is closed under composition and linear integration. For $n\geq 3$, $\mathcal{G}_{n+1}+\theta_k$ is defined as the class which contains $\mathcal{G}_n+\theta_k$ and solutions of Equation (2) applied to functions f in $\mathcal{G}_n+\theta_k$, and which is closed under composition and linear integration.*

Note that $\mathcal{L}+\theta_k$ contains $\exp^{[2]}(x)$, which grows roughly as fast as E_2 as noted in Section 2. Since $\mathcal{G}_{n+1}+\theta_k$ contains iterations of functions in $\mathcal{G}_n+\theta_k$, it contains at least one function that grows roughly as E_n. From Proposition 4 and using techniques similar to the proofs of Propositions 19, 20 and 22 we can then show (see [CMC00] for details) that:

Proposition 24 *The following correspondences exist between $\mathcal{G}_n+\theta_k$ and the levels of the Grzegorczyk hierarchy, $\mathcal{E}^n$ for all $n\geq 3$:*

1. *Any function in $\mathcal{G}_n+\theta_k$ is computable in $\mathcal{E}^n$.*
2. *If $f\in\mathcal{G}_n+\theta_k$ is an extension to the reals of some $\tilde{f}$ on $\mathbb{N}$, then $\tilde{f}\in\mathcal{E}^n$.*
3. *Conversely, if $f\in\mathcal{E}^n$ then some extension of it to the reals is in $\mathcal{G}_n+\theta_k$.*

A few remarks are in order. First, since we only have to solve Equation (2) $n-3$ times to obtain a function that grows as fast as E_{n-1}, this analog model contains exactly the nth level of the Grzegorczyk hierarchy if it is allowed to solve $n-3$ non-linear differential equations of the form (2).

Secondly, notice that Proposition 24 implies that $\cup_n(\mathcal{G}_n+\theta_k)$ includes all primitive recursive functions since, as mentioned in Proposition 5, $\cup_n\mathcal{E}^n=\mathcal{PR}$.

Finally, instead of allowing the GPAC to solve Equation (2), we can keep everything linear and define $\mathcal{G}_n+\theta_k$ by adding a new basis function which is an extension to the reals of E_{n-1}. While this produces a smaller set of functions on $\mathbb{R}$, it produces extensions to $\mathbb{R}$ of the same set of functions on $\mathbb{N}$ as the class defined here [CMC00].

6 Zero-finding on the reals

In [Moo96] another definition of analog computation is proposed, the $\mathbb{R}$-*recursive* functions. These are the functions that can be generated from the constants 0 and 1 from composition, integration of differential equations of the form $h(\boldsymbol{x},0)=f(\boldsymbol{x})$ and $\partial_y h(\boldsymbol{x},y)=g(h(\boldsymbol{x},y),\boldsymbol{x},y)$ on whatever interval the result $h=f+\int g\,dy$ is unique and well-defined, and the following minimization or zero-finding operator:

Definition 25 (Zero-finding) *If f is $\mathbb{R}$-recursive, then $h(\boldsymbol{x})=\mu_y f(\boldsymbol{x},y)=\inf\{y\in\mathbb{R}\,|\,f(\boldsymbol{x},y)=0\}$ is $\mathbb{R}$-recursive whenever it is well-defined, where the infimum is defined to find the zero of $f(\boldsymbol{x},\cdot)$ closest to the origin, that is, to minimize $|y|$. If both $+y$ and $-y$ satisfy this condition we return the negative one by convention.*

To what extent is this the correct extension of recursion theory to the reals? Integration of differential equations seems to be the closest continuous analog to primitive recursion: we define $h(y + dy)$, rather than $h(y+1)$, in terms of $h(y)$. This definition of zero-finding also seems fairly intuitive; however, it is hard to imagine how a physical process could locate a zero of a function unless it is differentiable, or at least continuous.

Although it is not explicitly recognized as such in [Moo96], the definition of $\mathbb{R}$-recursive relies on another operator, namely the assumption that $x \cdot 0 = 0$ even when x is undefined. While this can be justified by defining

$$x \cdot y = \int_0^y x \, dy$$

it actually deserves to be thought of as an operator in its own right, since it can convert partial functions into total ones. We can then combine μ with a "compression trick" to search over the integers to see if a function over $\mathbb{N}$ has any zeroes. This allows us to solve the Halting Problem; however, in physical terms, it corresponds to having our device run faster and faster, until it accomplishes an infinite amount of computation in finite time. This would require an infinite amount of energy, infinite forces, or both. Thus the *physical Church-Turing thesis*, that physically feasible devices can only compute recursive functions, remains intact.

In fact, by iterating this construction, we can compute functions in any level of the *arithmetical hierarchy* Σ^0_ω, of which the recursive and partial recursive functions are just the 0th and 1st levels respectively, Σ^0_0 and Σ^0_1. Sets in the jth level of this hierarchy can be defined with j alternating quantifiers over the set of integers, $\exists$ and $\forall$, applied to recursive predicates [Odi89]. We note that Bournez uses a similar recursion to show that systems with piecewise-constant derivatives can compute various levels of the hyperarithmetical hierarchy [Bou99,Bou99b].

If we quantify over functions instead of integers we define another hierarchy of even larger classes called the *analytical hierarchy* Σ^1_ω [Odi89]. These functions are $\mathbb{R}$-recursive as well, since we can encode sequences of integers as continued fractions and then search over the reals for a zero [Moo96].

Since this μ-operator is unphysical, in [Moo96] we stratify the class of $\mathbb{R}$-recursive functions according to how many nested uses of the μ-operator are needed to define a given function. Define M_j as the set of functions definable from the constants 0, 1, -1 with composition, integration, and j or fewer nested uses of μ. (We allow -1 as fundamental since otherwise we would have to define it as $\mu_y[y + 1]$. This way, $\mathbb{Z}$ and $\mathbb{Q}$ are contained in M_0.) We call this the μ-hierarchy.

For functions over $\mathbb{R}$ we believe that the μ-hierarchy is distinct. For instance, the characteristic function $\chi_\mathbb{Q}$ of the rationals is in M_2 but not in M_1. The recursive and partial recursive functions over $\mathbb{N}$ have extensions to the reals in M_2 and M_3 respectively. For higher j, M_j contains various levels of the analytical hierarchy as shown in Table 1, but we have no upper bounds for these classes.

However, in the classical setting, Kleene showed that any partial recursive function can be written in the form $h(\boldsymbol{x}, y) = U(\mu_y T(\boldsymbol{x}, y))$, where U and T are primitive recursive functions. Moreover, U and T can be elementary, or taken from an even smaller class [Odi89,Cut80,Ros84]. Thus the set of partial recursive functions can be defined even if we only allow one use of the zero-finding operator, and the μ-hierarchy collapses to its first level. Since the class $\mathcal{L} + \theta_k$ discussed in the previous section includes the elementary functions, this also means that combining a single use of μ with linear integration gives, at a minimum, the partial recursive functions.

If μ is not used at all we get M_0, the "primitive $\mathbb{R}$-recursive functions." These include the differentially algebraic functions $\mathcal{G}$ discussed above, as well as constants such as e and π; however, since the definition of integration in [Moo96] is somewhat more liberal than that of the GPAC where we require the solution to be unique for a domain of generation with a non-empty interior, M_0 also includes functions with discontinuous derivatives like $|x| = \sqrt{x^2}$ and the sawtooth function $\sin^{-1}(\sin x)$.

7 Conclusion

We have explored a variety of models of analog computation inspired by Shannon's GPAC. By allowing various basis functions and operators, we obtain analog counterparts of various function classes familiar from classical recursion theory. We summarize these in Table 1.

There are many open questions waiting to be addressed. These include:

1. Can we obtain upper bounds on classes like $\mathcal{G} + \theta_k$ and M_j, that so far we only have lower bounds for?
2. Is there more physical version of the μ operator which nonetheless extends $\mathcal{G}$ in a non-trivial way?
3. Do lower-level classes like P and NP have natural analog counterparts?

basis functions	operators	recursive classes
$0, 1, U_i^n$	$\circ, \int, x \cdot 0 = 0, (\mathbf{4j+3}).\boldsymbol{\mu}$	$M_{4j+3} \supseteq \Sigma_j^1, \Pi_j^1$
$0, 1, U_i^n$	$\circ, \int, x \cdot 0 = 0, \mathbf{3}.\boldsymbol{\mu}$	$M_3 \supseteq \Sigma_1^0$
$0, 1, U_i^n$	$\circ, \int, \mathbf{2}.\boldsymbol{\mu}$	$M_2 \supseteq \Sigma_0^0$
$0, 1, U_i^n, \theta_k$	$\circ, \int_{\mathbf{linear}}, \mathbf{1}.\boldsymbol{\mu}$	$\mathcal{L} + \theta_k + \mu \supseteq \Sigma_1^0$
$0, 1, -1, U_i^n, \theta_k$	$\circ, \int$	$\mathcal{G} + \theta_k \supseteq \mathcal{PR}$
$0, 1, -1, U_i^n, \theta_k$	$\circ, \int_{\mathbf{linear}}, (\mathbf{n-3}).\int_{\mathbf{2}}$	$\mathcal{G}_n + \theta_k = \mathcal{E}^n, n \geq 3$
$0, 1, -1, \pi, U_i^n, \theta_k$	$\circ, \int_{\mathbf{linear}}$	$\mathcal{L} + \theta_k = \mathcal{E}$
$0, 1, -1, U_i^n$	$\circ, \int$	$\mathcal{G} =$ d.a. functions

Table 1. Summary of the main results of the paper. The operations in the definitions of the recursive classes on the reals are denoted by: $\circ$ for composition, $\int_{\mathbf{linear}}$ for linear integration, $\int_{\mathbf{2}}$ for integrating equations of the form (2) as in definition 23, $\int$ for unrestricted integration and $\boldsymbol{\mu}$ for zero-finding on the reals. A number before an operation, as in $\mathbf{n}.\boldsymbol{\mu}$, means that the operator can be applied at most n times.

We look forward to addressing these with the enthusiastic reader.

Acknowledgements. We thank José Félix Costa for his collaboration on several results described in this paper, and Christopher Pollett and Ilias Kastanas for helpful discussions. This work was partially supported by grants from the Fundação para a Ciência e Tecnologia (PRAXIS XXI/BD/18304/98) and the Luso-American Development Foundation (754/98). MLC also thanks the Santa Fe Institute for hosting a visit that made this work possible.

References

[Arn96] V. I. Arnold. *Equations Différentielles Ordinaires.* Editions Mir, 5 ème edition, 1996.
[Bab73] A. Babakhanian. Exponentials in differentially algebraic extension fields. *Duke Math. J.*, 40:455–458, 1973.
[Ban75] S. Bank. Some results on analytic and meromorphic solutions of algebraic differential equations. *Advances in mathematics*, 15:41–62, 1975.
[BBV37] S. Bose, N. Basu and T. Vijayaraghavan. A simple example for a theorem of Vijayaraghavan. *J. London Math. Soc.*, 12:250–252, 1937.
[Bou99] O. Bournez. Achilles and the tortoise climbing up the hyper-arithmetical hierarchy. *Theoretical Computer Science*, 210(1):21–71, 1999.
[Bou99b] O. Bournez. *Complexité algorithmique des systèmes dynamiques continus et hybrides.* PhD thesis, École Normale Supérieure de Lyon, 1999.
[Bow96] M.D. Bowles. U.S. technological enthusiasm and the British technological skepticism in the age of the analog brain. *IEEE Annals of the History of Computing*, 18(4):5–15, 1996.
[Bra95] M. S. Branicky. Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoretical Computer Science*, 138(1), 1995.
[BSF00] A. Ben-Hur, H. Siegelmann, and S. Fishman. A theory of complexity for continuous time systems. To appear in *Journal of Complexity.*
[BSS89] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completnes, recursive functions and universal machines. *Bull. Amer. Math. Soc.*, 21:1–46, 1989.
[CMC99] M.L. Campagnolo, C. Moore, and J.F. Costa. Iteration, inequalities, and differentiability in analog computers. To appear in *Journal of Complexity.*
[CMC00] M.L. Campagnolo, C. Moore, and J.F. Costa. An analog characterization of the subrecursive functions. In P. Kornerup, editor, *Proc. of the 4th Conference on Real Numbers and Computers*, pages 91–109. Odense University, 2000.
[Cut80] N. J. Cutland. *Computability: an introduction to recursive function theory.* Cambridge University Press, 1980.
[Grz53] A. Grzegorczyk. Some classes of recursive functions. *Rosprawy Matematyzne*, 4, 1953. Math. Inst. of the Polish Academy of Sciences.
[Grz55] A. Grzegorczyk. Computable functionals. *Fund. Math.*, 42:168–202, 1955.
[Grz57] A. Grzegorczyk. On the definition of computable real continuous functions. *Fund. Math.*, 44:61–71, 1957.
[Hay96] H.K. Hayman. The growth of solutions of algebraic differential equations. *Rend. Mat. Acc. Lincei.*, 7:67–73, 1996.
[Kál43] L. Kálmar. Egyzzerü példa eldönthetetlen aritmetikai problémára. *Mate és Fizikai Lapok*, 50:1–23, 1943.
[KCG94] P. Koiran, M. Cosnard, and M. Garzon. Computability with low-dimensional dynamical systems. *Theoretical Computer Science*, 132:113–128, 1994.

[Kel76] W. Thomson (Lord Kelvin). On an instrument for calculating the integral of the product of two given functions. *Proc. Royal Society of London*, 24:266–268, 1876.

[KM99] P. Koiran and C. Moore. Closed-form analytic maps in one or two dimensions can simulate Turing machines. *Theoretical Computer Science*, 210:217–223, 1999.

[Lac55] D. Lacombe. Extension de la notion de fonction récursive aux fonctions d'une ou plusieurs variables réelles I. *C. R. Acad. Sci. Paris*, 240:2478–2480, 1955.

[LR87] L. Lipshitz and L. A. Rubel. A differentially algebraic replacement theorem, and analog computation. *Proceedings of the A.M.S.*, 99(2):367–372, 1987.

[Mee93] K. Meer. Real number models under various sets of operations. *Journal of Complexity*, 9:366–372, 1993.

[Moo90] C. Moore. Unpredictability and undecidability in dynamical systems. *Physical Review Letters*, 64:2354–2357, 1990.

[Moo96] C. Moore. Recursion theory on the reals and continuous-time computation. *Theoretical Computer Science*, 162:23–44, 1996.

[Moo98] C. Moore. Dynamical recognizers: real-time language recognition by analog computers. *Theoretical Computer Science*, 201:99–136, 1998.

[Odi89] P. Odifreddi. *Classical Recursion Theory.* Elsevier, 1989.

[Orp97a] P. Orponen. On the computational power of continuous time neural networks. In *Proc. SOFSEM'97, the 24th Seminar on Current Trends in Theory and Practice of Informatics*, Lecture Notes in Computer Science, pages 86–103. Springer-Verlag, 1997.

[Orp97b] P. Orponen. A survey of continuous-time computation theory. In D.-Z. Du and K.-I Ko, editors, *Advances in Algorithms, Languages, and Complexity*, pages 209–224. Kluwer Academic Publishers, Dordrecht, 1997.

[PE74] M. B. Pour-El. Abtract computability and its relation to the general purpose analog computer. *Trans. Amer. Math. Soc.*, 199:1–28, 1974.

[PR89] M. B. Pour-El and J. I. Richards. *Computability in Analysis and Physics.* Springer-Verlag, 1989.

[Ros84] H. E. Rose. *Subrecursion: functions and hierarchies.* Clarendon Press, 1984.

[Rub89] L. A. Rubel. Digital simulation of analog computation and Church's thesis. *The Journal of Symbolic Logic*, 54(3):1011–1017, 1989.

[Rub89b] L. A. Rubel. A survey of transcendentally transcendental functions. *Amer. Math. Monthly*, 96:777–788, 1989.

[SF98] H. T. Siegelmann and S. Fishman. Analog computation with dynamical systems. *Physica D*, 120:214–235, 1998.

[Sha41] C. Shannon. Mathematical theory of the differential analyser. *J. Math. Phys. MIT*, 20:337–354, 1941.

[Sie98] H. Siegelmann. *Neural Netwoks and Analog Computation: Beyond the Turing Limit.* Birkhauser, 1998.

[Vij32] T. Vijayaraghavan. Sur la croissance des fonctions définies par les équations différentielles. *C. R. Acad. Sci. Paris*, 194:827–829, 1932.

[VSD86] A. Vergis, K. Steiglitz, and B. Dickinson. The complexity of analog computation. *Mathematics and computers in simulation*, 28:91–113, 1986.

[Zho97] Q. Zhou. Subclasses of computable real functions. In T. Jiang and D. T. Lee, editors, *Computing and Combinatorics*, Lecture Notes in Computer Science, pages 156–165. Springer-Verlag, 1997.