Some recent developments on Shannon's General Purpose Analog Computer

Daniel Silva Graça
CLC and DM/FCT, Universidade do Algarve, C. Gambelas,
8000-062 Faro, Portugal

April 28, 2004

Abstract

This paper revisits one of the first models of analog computation, the General Purpose Analog Computer (GPAC). In particular, we restrict our attention to the improved model presented in [11] and we show that it can be further refined. With this we prove the following: (i) the previous model can be simplified; (ii) it admits extensions having close connections with the class of smooth continuous time dynamical systems. As a consequence, we conclude that some of these extensions achieve Turing universality. Finally, it is shown that if we introduce a new notion of computability for the GPAC, based on ideas from computable analysis, then one can compute transcendentally transcendental functions such as the Gamma function or Riemann's Zeta function.

1 Introduction

In this paper we explore a particular model of analog computation, the General Purpose Analog Computer (GPAC). The GPAC was introduced in 1941 by Shannon [30] as a mathematical model of an analog device, the Differential Analyzer [5]. This device was one of the most popular analog computers in the 1930s and was intended to solve numerical problems, especially differential equations [3]. In short, a (mechanical) differential analyzer may be seen as a set of interconnected shafts, each of which representing one of the quantities involved in the computation. Although the reader might feel uncomfortable with this approach, based on technologically obsolete computing devices, we believe there is much to explore. Quoting James Nyce [21]: *"Because digital computers and computation have been so successful, they have influenced how we think about both computers as machines and computation as a process - so much so, it is difficult today to reconstruct what analog computing was all about... It is a history in which digital machines can do things 'better' and 'faster' than other machines... However, what is at stake here are not matters of speed or precision.*

Rather, it is an argument about what can be rendered and understood through a machine that does computation."

Indeed, since the pioneering work of Turing, the notion of computability has been pretty much settled. Every 'computable' entity is supposed to have a suitable symbolic representation and a computation is considered as a sequence of 'elementary steps' on these representations. Furthermore, this sequence of 'elementary steps' should be defined in a manner such that it could be performed by any human following a 'rule of thumb.' This remains a highly consensual approach, with some reserves [32], [9], [31], that can be found in the computable analysis literature [25], [14], [33].

On the other side, a different philosophy underlies analog computers. Here we don't have a notion of 'algorithm' and there is no need to translate quantities into appropriate symbolic forms. Moreover, we do not expect an analog computation to be possibly carried out by an human following a 'rule of thumb.' In an analog computer, variables are represented by physical quantities on which the operations are performed. The computation is carried out by some *"physical system that obeys the same mathematical relations that control the physical or technical phenomenon under investigation"* [26, p. 49]. This procedure is in some sense much more natural to the physicist and to the engineer.[1] As General Electric put it in 1952: *"A virtue of the analog computer is that its basic design concepts are usually easy to recognize. What goes on inside is understandable since it is an analog of the real thing"* whereas *"the digital type computer is a product of pure logic. It cannot be described as similar to something with which we are familiar"* [10], cited from [23, p. 39].

Therefore, although digital computers had long ago superseded their analog counterpart due, to a large extent, to the spectacular development of digital technology, we still believe that these old-fashioned analog devices might bring some fresh air to the theory of computation. Quoting again James Nyce [21]: *"Analog machines ... offer us a way to reconsider what we have come to take for granted - how we model and think about objects in the world"* and, in particular, what we understand and mean by computation.

This paper focuses primarily on 3 objectives. The first one is to simplify the submodel of the GPAC presented in [11]. Indeed, the general GPAC presented by Shannon (and also by Pour-El [24]) has some problems that can be solved by the FF-GPAC model introduced by Graça and Costa in [11]. The present paper shows that this model can be further simplified (to the PGPAC model). This point is covered by Sections 2, 3, 4, and partially by Section 6.

The second objective is to extend a result from [11, Corollary 1], where it proved that the class of FF-GPAC computable functions is exactly given by the class of dynamical systems of the form $\mathbf{y}' = \mathbf{p}(\mathbf{y}, t)$, where $\mathbf{p}$ is a vector of polynomials. In Section 5 we prove that these links can be generalized if we add new types of units to the PGPAC model, thereby obtaining the IC model. It is shown that every class of IC computable functions corresponds to a specific

[1] A similar philosophy can also be found, to a certain extent, in more recent models such as artificial neural networks [18], [1]. As Haykin says [12, p. 25]: *"The design of a neural network is based directly on real-life data, with the data set being permitted to speak for itself."*

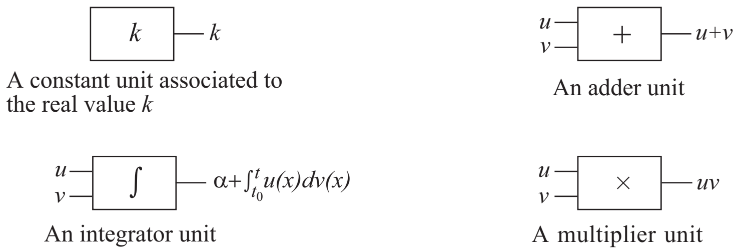

Figure 1: Different types of units used in a GPAC.

class of continuous time dynamical systems and, reciprocally, given a smooth continuous time dynamical system, it can be described as the output of some IC. As a corollary, it will be shown that there are classes of ICs that are Turing universal.

Moreover, and in the spirit of objective 1, we also add new types of units to the FF-GPAC to generalize it to the FIC model, and then prove results establishing links between the IC and FIC models. This is done in Section 6. A schema of relations between the models described in this paper is provided in Fig. 5 (some of the relations are proved in what follows).

Finally, and as a third objective, we show that some of the mathematical limitations pointed out in the literature [27] are not inherent to the GPAC but rather on the underlying notion of computability. In particular, we will show that the Gamma function and Riemann's Zeta function can indeed be computed by a GPAC if we redefine the notion of 'computable function by a GPAC' in a way that it matches more closely the notion of computability from computable analysis. This is done in Section 7.

2 Preliminaries

Unlike the approach in computable analysis [25], [14], [33], the GPAC is not directly based on the Turing machine, neither on some effective procedures. The model basically consists of circuits composed of 'black boxes' as indicated in Fig. 1 (the so-called *analog units*. These are not the units originally used by Shannon, but they are equivalent. Note that integrators compute Riemman-Stieltjes integrals). It is required that inputs (and outputs) can never be interconnected. It is also required that each input is connected to, at most, one output.

The inputs $x_1, ..., x_k$ of a GPAC are applied to every input of a unit that is not connected to the output of some other unit. Then an output for the GPAC will consist of outputs of some units and/or some inputs of the GPAC. Notice the existence of a parameter α in the integrator unit. This corresponds to an initial setting that will settle the output for the integrator. Although these definitions can be made more precise (e.g. a circuit can be seen as a labelled graph), this yields, in our opinion, unnecessary complications. So, we rather prefer to use a naive approach to circuit based models.

The reader should also remark that each output of a unit can be obtained by solving a set of equations. For instance, if $\mathcal{U}$ is a GPAC consisting of only one adder with inputs x_1 and x_2, then the output of the adder will be the solution of the equation $y = x_1 + x_2$. In general, if we want to determinate the output of some GPAC with n units, we have to solve a set of n equations.

Of course, the solution of a system of equations may not be unique or can even not exist (and it is not difficult to find examples - cf. [11]). Furthermore, in [11] it is also shown that Pour-El's characterization for the GPAC [24] still have some deficiencies. Therefore, in the next section, we restrict Shannon's model in order to avoid these problems.

Another important question (already reported in [24], [29]) is what happens if we allow other types of black boxes beside those indicated in Fig. 1. An answer for this question will be supplied in Section 5. This approach will also enable us to present close connections with the class of C^1 continuous time dynamical systems defined in $\mathbb{R}^n$.

3 The basic model

In this section we introduce one of the basic models that will be used in this paper. It is essentially a restricted version of Shannon's GPAC. In Section 6 it will be shown that this model is equivalent to the FF-GPAC model presented in [11].

For the matters of our work, it is only necessary to consider one input for this model. We will usually refer to this input as the 'time.' However, when considering circuits without integrators, we admit that they might have more than one input.

The model presented in this section is based in the following ideas: First, construct acyclic circuits that compute polynomials (*polynomial circuits*) by using the following units from Fig. 1: constant units, adders, and multipliers.[2] We assume that a polynomial circuit may have no units at all computing, in this case, the identity.[3] Second, use these circuits as building blocks for more complex GPACs that we call polynomial GPACs (PGPAC for short). A PGPAC is constructed in the following manner. Take n integrators $\mathcal{I}_1, ..., \mathcal{I}_n$. Then use polynomial circuits such that the following three conditions hold:

1. Each input of a polynomial circuit is the input of the PGPAC or the output of an integrator;

2. Each integrand input of an integrator is an output of a polynomial circuit;

[2]Notice that multipliers could be replaced by integrators and adders (cf. [24, p. 11]). However, this is not very relevant to our results.

[3]Note that the identity can also be computed by a polynomial circuit consisting of one multiplier and one constant unit associated to the value 1 (just build a circuit that multiplies the input by 1).

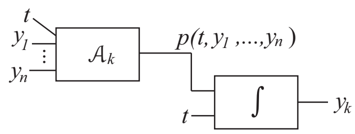

Figure 2: Schema of inputs and outputs for the integrator $\mathcal{I}_k$ in a PGPAC. p denotes a polynomial and y_i denotes the output of $\mathcal{I}_i$.

3. Each variable of integration input of an integrator is the input of the PGPAC.

Formally, a polynomial circuit is defined as follows.

Definition 1 *A* polynomial circuit *is an acyclic GPAC built only with adders, constants units, and multipliers.*

We assume that polynomial circuits may have several inputs. The proof of the following lemma will be left to the reader.

Lemma 2 *If $x_1, ..., x_n$ are the inputs of a polynomial circuit, then the output of the circuit will be $y = p(x_1, ..., x_n)$, where p is a polynomial. Reciprocally, if $y = p(x_1, ..., x_n)$, where p is a polynomial, then there is a polynomial circuit with inputs $x_1, ..., x_n$, and output y.*

Definition 3 *Consider a GPAC $\mathcal{U}$ with n integrators $\mathcal{I}_1, ..., \mathcal{I}_n$, and one input t. Suppose that to each integrator $\mathcal{I}_i$, $i = 1, ..., n$, we can associate a polynomial circuit $\mathcal{A}_i$ with the property that the integrand input of $\mathcal{I}_i$ is connected to an output of $\mathcal{A}_i$. Suppose that each input of $\mathcal{A}_i$ is connected to the output of an integrator or to the input t. Suppose also that the variable of integration input of each integrator is connected to the input t. In these conditions we say that $\mathcal{U}$ is a polynomial GPAC (PGPAC) with input t. (cf. Fig. 2)*

A concrete example of a PGPAC is presented in Fig. 3.

4 Properties of the model

The following theorems are taken from [11]. Notice that the PGPAC and the model used in [11] are apparently different, but their equivalence is shown in Corollary 17.

Theorem 4 *Suppose that $\mathcal{U}$ is a PGPAC with one input t, defined on an interval $[t_0, t_f)$, where t_f may possibly be ∞. Then there exists an interval $[t_0, t^*)$ (with $t^* \leq t_f$) where each output exists and is unique. Moreover, if $t^* < t_f$, then there exists an integrator with output y such that $y(t)$ is unbounded as $t \to t^*$.*

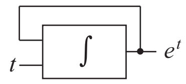

Figure 3: Example of a PGPAC that computes the exponential function exp. We suppose the initial output of the integrator to be $y(0) = 1$ (note that exp is the solution of $y(x) = \int y(x)dx$, $y(0) = 1$).

Theorem 5 *If y is generated on some non-trivial interval I by a PGPAC with n integrators and one input t, then there is a nonzero polynomial p with real coefficients such that*

$$p\left(t, y, y', ..., y^{(n)}\right) = 0, \qquad on\ I. \tag{1}$$

Definition 6 *The unary function y is differentially algebraic if there exists a nonzero polynomial p with real coefficients such that (1) holds. If y is not differentially algebraic, then we say that y is transcendentally transcendental.*

Theorem 7 *Suppose that y is differentially algebraic on some non-trivial interval I. Then there is a closed subinterval $I' \subseteq I$ with non-empty interior such that y can be generated by a PGPAC on I'.*

The last two theorems assert a classical result on the literature about the GPAC [30], [24], [17]: unary functions generated by (P)GPACs are, in essence, differentially algebraic functions.

This result indicates that a large class of functions, such as polynomials, trigonometric functions, elliptic functions, etc., can actually be generated by a PGPAC. As a corollary, some functions such as the Gamma function,

$$\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt, \tag{2}$$

cannot be generated because they are not differentially algebraic functions [28, Theorem 4].

5 An extension of the model

In this section we extend the PGPAC model to the IC model by allowing the use of new units instead of constant units, adders, and multipliers as indicated in Fig. 1. Moreover, and as a generalization of the result presented in [11, Corollary 1], we show that not only the PGPAC can be related with a particular class of dynamical systems, but also every class of ICs can be put into correspondence with some class of continuous time dynamical systems and vice versa.

Henceforth, consider $\mathcal{F}$ to be a set constituted by C^1 functions of the form $f_i : \mathbb{R}^{k_i} \to \mathbb{R}$, defined on an open domain, where $k_i \in \mathbb{N}$ and $i \in I$. We now present a generalization for polynomial circuits.

Definition 8 *A $\mathcal{F}$-circuit is a circuit built like a polynomial circuit, but using only units associated to functions in $\mathcal{F}$.*

Similarly to polynomial circuits, we consider that $\mathcal{F}$-circuits might have several inputs. Next, we introduce the main definition of the section.

Definition 9 *An $\mathcal{F}$-integrating circuit ($\mathcal{F}$-IC) with one input is a circuit built like a PGPAC, where $\mathcal{F}$-circuits are used instead of polynomial circuits.*

As an example, if $\mathcal{F}$ is constituted by the constant functions, the binary sum, and the binary product, then the class of $\mathcal{F}$-ICs will correspond to the class of PGPACs. Next, we present some useful notation.

Definition 10 *Let χ be a set of functions and OP a collection of operators. Then $[OP; \chi]$ denotes the smallest set of functions containing χ and closed under the operations of OP. The set $[OP; \chi]$ is called a* function algebra.

The notation will not be very rigorous (e.g. $[OP, G; \chi]$ means $[OP \cup \{G\}; \chi]$, $[OP; \chi_1, \chi_2]$ means $[OP; \chi_1 \cup \chi_2]$, etc.), but the context will be enough to clarify all situations. We shall consider the following functions and operators.

1. The projections. Let A be a set. For each $n, i \in \mathbb{N}$, where $1 \leq i \leq n$, $U_i^n : A^n \to A$ is called projection (over A) and is defined by $U_i^n(x_1, ..., x_n) = x_i$;
2. The constant functions. For each $k \in \mathbb{R}$, the image of $c_k : \to \mathbb{R}$ is the value k;
3. Composition: Suppose that g is an p-ary function, with $p \geq 1$, and that $f_1, ..., f_p$ are n-ary functions. Then the composition operator applied to these functions by that order yields the n-ary function h given by $h(\mathbf{x}) = g(f_1(\mathbf{x}), ..., f_p(\mathbf{x}))$. We write $h = C(g; f_1, ..., f_p)$.

We set the following notation

$$U = \{U_i^n : n, i \in \mathbb{N} \text{ and } 1 \leq i \leq n\},$$
$$C_{\mathbb{R}} = \{c_k : k \in \mathbb{R}\}.$$

If we relax the notation and take $\{C_{\mathbb{R}}, +, \times\}$ as the set $C_{\mathbb{R}} \cup \{+, \times\}$, then a PGPAC is simply an $\{C_{\mathbb{R}}, +, \times\}$-IC.

Theorem 11 *Let $\mathcal{U}$ be an $\mathcal{F}$-IC with n integrators and one input t. Then there exist n $(n+1)$-ary functions $h_1, ..., h_n \in [C; U, \mathcal{F}]$ such that $(\psi_1, ..., \psi_j)$ is an output of $\mathcal{U}$ if and only if there exist n unary functions $y_1, ..., y_n$ such that:*

1. *$\partial_t y_i = h_i(t, y_1, ..., y_n)$ and $y_i(t_0) = \alpha_i$, where $\alpha_i \in \mathbb{R}$;*
2. *There exist j $(n+1)$-ary functions $g_1, ..., g_j \in [C; U, \mathcal{F}]$ such that $\psi_i = g_i(t, y_1, ..., y_n)$, for $i = 1, ..., j$.*

Proof. It is possible to show that if $y_1, ..., y_n$ are the outputs of the integrators, then the output of each $\mathcal{F}$-circuit is given by $f(t, y_1, ..., y_n)$, where $f \in [C; U, \mathcal{F}]$ (simply generalize Lemma 2). Therefore, each output y_i of an integrator satisfies

$$y_i(t) = \alpha_i + \int_{t_0}^{t} h_i(t, y_1(t), ..., y_n(t))dt, \tag{3}$$

where $\alpha_i \in \mathbb{R}$ and $h_i \in [C; U, \mathcal{F}]$. Part 1 of the theorem follows by differentiating equation (3). Part 2 of the theorem follows from the fact that each output is the input t, the output of some integrator, or a single output of a $\mathcal{F}$-circuit.

Reciprocally, if conditions 1 and 2 are satisfied, then it is not difficult to construct an $\mathcal{F}$-IC $\mathcal{U}$ with input t, n integrators, and output $(\psi_1, ..., \psi_j)$. ■

A similar formalism to the one presented in the previous theorem was already introduced by Pour-El, but for a different model [24]. Indeed, this model uses a system of ODEs of the form $Ay' = b$, where A does not have to be invertible.

The following corollary was already reported in [11, Corollary 1], but for the FF-GPAC model presented there. A similar result was also proved by Pour-El [24, Theorem 4], but it was only explicitly stated (as far as we known), for a special case, in [7, Proposition 2].

Corollary 12 *The function y is generated by a PGPAC if and only if it is a component of the solution $\mathbf{y} = (y_1, ..., y_n)$ of $\mathbf{y}' = \mathbf{p}(\mathbf{y}, t)$, where $\mathbf{p}$ is a vector of polynomials.*

Proof. The PGPAC uses as basic functions elements of $\{C_{\mathbb{R}}, +, \times\}$. But $[C; U, C_{\mathbb{R}}, +, \times]$ is the set of all polynomials. Then part 1 of Theorem 11 gives us $y_i' = p_i(t, y_1, ..., y_n)$, where p_i is a polynomial. Moreover, by using part 2 of that theorem, we conclude that each g_i is a polynomial. Hence, it can be written as $g_i' = q_i$, where q_i is a polynomial. Therefore, for the special case of polynomials, part 1 and 2 of Theorem 11 can be condensed in a single system $\mathbf{y}' = \mathbf{p}(\mathbf{y}, t)$, where t is the input. ■

Notice that the previous theorem provides a very pleasant characterization of the computational power of an $\mathcal{F}$-IC in terms of continuous time dynamical systems. Indeed, it is known [13, p. 160] that a C^1 continuous time dynamical system, working on the Euclidean space S, is equivalent to a system of ordinary differential equations (ODEs)

$$\mathbf{x}' = \mathbf{f}(\mathbf{x}), \tag{4}$$

where $\mathbf{x}$ is a unary function with $\mathbf{x}(t) \in S$.[4] Hence, Theorem 11 says us two things: (i) By adding different types of units one gets, in general, more complex dynamical systems and more computational power. The exact characterization of this power is given by the theorem; (ii) Given a C^1 continuous time dynamical system, working in $\mathbb{R}^n$, one can associate it an ODE (4) that, by its turn, can be simulated by some $\mathcal{F}$-IC (just take $\mathcal{F} = \{f_1, ..., f_n\}$, where $f_1, ..., f_n$ designate the various components of $\mathbf{f}$).

[4] Notice that $\mathbf{x}' = \mathbf{f}(\mathbf{x}, t)$ can be reduced to (4) by taking $x_{n+1}' = 1$.

Hence, one concludes that there is a tight relationship between ICs and C^1 continuous time dynamical systems working in $\mathbb{R}^n$. Another interesting result is given by the following corollary:

Corollary 13 *For each Turing machine M, there is a class $\mathcal{F}$ constituted by C^∞ functions such that M can be simulated by an $\mathcal{F}$-IC.*

Proof. This is an immediate consequence of Theorem 5.7, Corollary 5.8, and the comments following them in [4], where it is stated that every Turing machine can be simulated by a system of smooth ODEs. Then using Theorem 11, one concludes the result. ∎

A review of notions of simulation can be found in [4]. In essence, the previous theorem can be described as follows. Given a Turing machine M, one can encode each configuration into an element of $\mathbb{Z}^2$. Hence, each Turing machine M is equivalent to a discrete time dynamical system $(\mathbb{Z}^2, f)$, where $f(\mathbf{x}, t) \in \mathbb{Z}^2$ gives the state reached from $\mathbf{x} \in \mathbb{Z}^2$ after $t \in \mathbb{N}$ time steps [4, Proposition 5.1]. Then one can prove that there is a system of smooth ODEs associated to a continuous time dynamical system $(\mathbb{R}^4, F)$, where $F(\mathbf{x}, t) \in \mathbb{R}^4$ gives the state reached from $\mathbf{x} \in \mathbb{R}^4$ after time $t \in \mathbb{R}_0^+$, with the following property: there exists an $\varepsilon > 0$ such that for each $\mathbf{x} \in \mathbb{N}^2$, the first two components of $F(\mathbf{x}, \mathbf{x}, t)$ are equal to $f(\mathbf{x}, k)$, where $t \in [2k - \varepsilon, 2k + \varepsilon]$ and $k \in \mathbb{N}$. In this manner one can say that $(\mathbb{R}^4, F)$ simulates $(\mathbb{Z}^2, f)$ and, therefore, $(\mathbb{R}^4, F)$ simulates the Turing machine M. The existence of ε is important since one should have some robustness to "imprecise time sampling."[5] Moreover, the point $\mathbf{x}$ encodes the initial configuration of Turing machine M, including its input, and is used as an initial state for the dynamical system $(\mathbb{R}^4, F)$ and, hence, as an initial condition to the associated system of ODEs. Therefore, the initial input of M is represented in the $\mathcal{F}$-IC through the initial settings of some integrators. It is also important to remark that, in this case, these initial setting would take integer values.

In Section 6, we present a concrete example of a class $\mathcal{F}$ with the power of Turing universality.

6 Feedforward ICs

Although ICs might appear natural in the context of the theory of dynamical systems (cf. Theorem 11), they usually are not flexible enough for many applications. In fact, the restriction involving the variable of integration input for integrators may be very limiting when composing ICs. For instance, if one wants to compute $\exp(\exp(t))$, one could compose two circuits of those presented in Fig. 3. But then the resulting circuit would no longer be an IC because the variable of integration input of an integrator would be the output of the other

[5] The reader might feel, with reason, that one should also let F to be robust to small perturbations. This would certainly provide a more natural notion of simulation. However, if we consider F to be analytic, and if we do not introduce some kind of bound a priori, it seems very hard to avoid an accumulation of errors that will compromise the computation. So, an usual procedure is to allow the use of "exact computation" (eg. [30], [15], [22]).

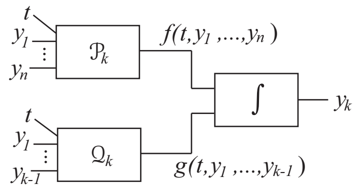

Figure 4: Schema of inputs and outputs for the integrator $\mathcal{I}_k$ in the $\mathcal{F}$-FIC $\mathcal{U}$. Here $f, g \in [C; U, \mathcal{F}]$. y_i denotes the output of $\mathcal{I}_i$.

integrator. So, an extension of the IC model is desirable. However, this extension should not have the problems referred at Section 2 (e.g., outputs should exist and be unique). Therefore, we extend the FF-GPAC model presented in [11] to obtain the FIC model. Then we show that the IC and FIC models can be related and, as a corollary, we prove that FF-GPACs and PGPACs are equivalent.

Definition 14 *A $\mathcal{F}$-feedforward IC ($\mathcal{F}$-FIC) with one input is a circuit that uses n integrators $\mathcal{I}_1, ..., \mathcal{I}_n$ and $2n$ $\mathcal{F}$-circuits $\mathcal{P}_1, ..., \mathcal{P}_n, \mathcal{Q}_1, ..., \mathcal{Q}_n$ satisfying the following conditions:*

1. *Each input of $\mathcal{P}_i$ is the input of the $\mathcal{F}$-FIC or the output of an integrator, for $i = 1, ..., n$;*
2. *Each input of $\mathcal{Q}_i$ is the input of the $\mathcal{F}$-FIC or the output of an integrator $\mathcal{I}_j$, for $j < i$ and $i = 1, ..., n$;*
3. *The integrand input of $\mathcal{I}_i$ is an output of $\mathcal{P}_i$, for $i = 1, ..., n$;*
4. *The variable of integration input of $\mathcal{I}_i$ is an output of $\mathcal{Q}_i$, for $i = 1, ..., n$.*

This is sketched in Fig. 4. When $\mathcal{F} = \{C_{\mathbb{R}}, +, \times\}$, one obtains the FF-GPAC model presented in [11, Definition 3]. From Definitions 9 and 14, it is immediate to conclude the following:

Lemma 15 *Let $\mathcal{U}$ be an $\mathcal{F}$-IC. Then $\mathcal{U}$ is also a $\mathcal{F}$-FIC.*

A kind of converse for the previous theorem can also be proved:

Theorem 16 *Let $\mathcal{F}$ be a set of C^1 functions and let $\mathcal{U}$ be a $\mathcal{F}$-FIC. Then there exists an $\{\mathcal{F}, \mathcal{F}', c_{-1}, +, \times\}$-IC that generates the same outputs of $\mathcal{U}$, where $\mathcal{F}'$ is the class constituted by the partial derivatives of all elements from $\mathcal{F}$.*

Proof. If $y_1, ..., y_n$ are the outputs of the integrators of $\mathcal{U}$, then the output of each $\mathcal{F}$-circuit is given by $f(t, y_1, ..., y_n)$, where $f \in [C; U, \mathcal{F}]$. Therefore, each output y_i of an integrator satisfies

$$y_i(t) = \alpha_i + \int_{t_0}^{t} h_i(t, y_1(t), ..., y_n(t))\ dg_i(t, y_1(t), ..., y_{i-1}(t))$$

where $\alpha_i \in \mathbb{R}$ and $g_i, h_i \in [C; U, \mathcal{F}]$. Differentiating the last equation, one gets

$$y_i' = h_i(t, y_1, ..., y_n) \sum_{j=0}^{i-1} \frac{\partial g_i}{\partial y_j} y_j',$$

with $y_0 = t$. This can be rewritten as

$$\mathbf{A}\mathbf{y}' = \mathbf{b},$$

with

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -h_2 \frac{\partial g_2}{\partial y_1} & 1 & 0 & \cdots & 0 \\ -h_3 \frac{\partial g_3}{\partial y_1} & -h_3 \frac{\partial g_3}{\partial y_2} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -h_n \frac{\partial g_n}{\partial y_1} & -h_n \frac{\partial g_n}{\partial y_2} & -h_n \frac{\partial g_n}{\partial y_3} & \cdots & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} h_1 \frac{\partial g_1}{\partial t} \\ h_2 \frac{\partial g_2}{\partial t} \\ h_3 \frac{\partial g_3}{\partial t} \\ \vdots \\ h_n \frac{\partial g_n}{\partial t} \end{bmatrix}.$$

Because $\det(\mathbf{A}) = 1$, $\mathbf{A}$ is invertible and

$$\mathbf{y}' = \mathbf{A}^{-1}\mathbf{b}.$$

Notice that each member of $\mathbf{A}^{-1}$ can be obtained from elements of $\mathbf{A}$ using only products and sums, and the same happens for $\mathbf{A}^{-1}\mathbf{b}$. Using the rule of differentiation for composite functions, one can easily conclude that each component of $\mathbf{A}^{-1}\mathbf{b}$ belongs to $[C; U, \mathcal{F}, \mathcal{F}', c_{-1}, +, \times]$. Using Theorem 11, one infers that every output of $\mathcal{U}$ is also an output of an $\{\mathcal{F}, \mathcal{F}', c_{-1}, +, \times\}$-IC. ■

Corollary 17 *A function f is generated by a FIC using constant units, adders, and multipliers (cf. Fig. 1) if and only if it is generated by a PGPAC. That is, the class of functions generated by FF-GPACs is exactly the class of functions generated by PGPACs.*

Proof. PGPACs are $\{C_{\mathbb{R}}, +, \times\}$-ICs. Then, by Lemma 15, one only has to show that if f is generated by a $\{C_{\mathbb{R}}, +, \times\}$-FIC, it is also generated by an $\{C_{\mathbb{R}}, +, \times\}$-IC. But the partial derivatives of the functions in $C_{\mathbb{R}} \cup \{+, \times\}$ lie in $C_{\mathbb{R}} \cup U \cup \{+, \times\}$. Hence, by Theorem 16, f is also generated by an $\{C_{\mathbb{R}}, +, \times\}$-IC. ■

Fig. 5 provides the essential connections between the different models referred to in this work. With the use of the FIC model, one can prove some interesting results. Notice that, for the following theorem, g is only generated by a $\mathcal{F}$-circuit (and not by a $\mathcal{F}$-FIC or an $\mathcal{F}$-IC). This is the main difference between Theorems 18 and 19.

Theorem 18 *Let $g \in [C; U, \mathcal{F}]$ be a k-ary function and let $f_1, ..., f_k$ be unary functions generated by some $\mathcal{F}$-FICs ($\mathcal{F}$-ICs) $\mathcal{U}_1, ..., \mathcal{U}_k$, respectively. Then the composition of g with $f_1, ..., f_k$ is also generated by a $\mathcal{F}$-FIC ($\mathcal{F}$-IC).*

$$\begin{array}{ccccc}
\text{PGPAC} & \overset{\text{Cor. 17}}{=} & \text{FF-GPAC} & \subseteq & \text{Shannon's GPAC} \\
\Vert\ {\scriptstyle \text{Def. 9}} & & \Vert\ {\scriptstyle \text{[11, Def.3]}} & & \\
\{C_{\mathbb{R}},+,\times\}\text{-IC} & = & \{C_{\mathbb{R}},+,\times\}\text{-FIC} & & \\
\rotatebox{-90}{$\subseteq$} & & \rotatebox{-90}{$\subseteq$} & & \\
\mathcal{F}\text{-IC} & \overset{\text{Lemma 15}}{\subseteq} & \mathcal{F}\text{-FIC} & & \\
\rotatebox{-90}{$\subseteq$} & & \Vert & & \\
\{\mathcal{F},\mathcal{F}'\}\text{-IC} & \overset{\text{Thm. 16}}{\supseteq} & \mathcal{F}\text{-FIC} & & \\
\textit{cf. Fig. 2} & & \textit{cf. Fig. 4} & &
\end{array}$$

Figure 5: Schema of relations between the models described in this work. Here $\{C_{\mathbb{R}},+,\times\} \subseteq \mathcal{F}$ and $\mathcal{F}'$ is the set of partial derivatives of $\mathcal{F}$. Fig. 2 and Fig. 4 suggest the structure of circuits in the first and second columns, respectively.

Proof. g can be computed by a $\mathcal{F}$-circuit. Connecting the inputs of this circuit with the outputs of $\mathcal{U}_1,...,\mathcal{U}_k$, one gets a composite circuit $\mathcal{U}$ that is a $\mathcal{F}$-FIC ($\mathcal{F}$-IC, respectively) computing $C(g; f_1,...,f_k)$. ■

Theorem 19 *Let $f : \mathbb{R} \to \mathbb{R}^n$ and $g : \mathbb{R} \to \mathbb{R}$ be unary functions generated by some $\mathcal{F}$-FICs. Then $g \circ f$ is also generated by a $\mathcal{F}$-FIC.*

Proof. Suppose that f and g are generated by $\mathcal{F}$-FICs $\mathcal{U}_1,\mathcal{U}_2$, respectively. Then link the output of $\mathcal{U}_1$ to the input of $\mathcal{U}_2$. The resulting circuit will be a $\mathcal{F}$-FIC generating $g \circ f$. ■

Notice that the previous theorem does not apply, in general, to $\mathcal{F}$-ICs. For the following result, consider the function θ_k defined by $\theta_k(x) = x^k$ if $x \geq 0$ and $\theta_k(x) = 0$ if $x < 0$ ($k \in \mathbb{N}$). This function can be seen [7] as a C^{k-1} version of Heaviside's step function $\theta(x)$, where $\theta(x) = 1$ for $x \geq 0$ and $\theta(x) = 0$ for $x < 0$. Because we assumed in the beginning of Section 5 that $\mathcal{F}$ is constituted by C^1 functions, we only consider functions θ_k for $k \geq 2$.

Theorem 20 *For each fixed $k \geq 2$ and every Turing machine M, there is a $\{C_{\mathbb{R}},+,\times,\theta_k\}$-FIC $\mathcal{U}$ simulating it. The initial input of M corresponds to integer initial settings for $\mathcal{U}$.*

Proof. We only give a sketch of the proof. In particular, it is sufficient to check that Branicky's simulation of Turing machines [4, Proposition 5.7] can be implemented by $\{C_{\mathbb{R}},+,\times,\theta_k\}$-FICs. Using the same arguments and notation as Branicky (we suppose that the reader is referring to the proof of [4, Proposition 5.7]), one just have to take $\bar{F}$ as the function presented by Koiran and Moore in [16, Theorem 2],[6] Π as the function defined by $\Pi(x) = s(x+1/2)$, where s is a function presented by Campagnolo in [6, p. 7] and $S_1(t) = \theta_k(\sin(\pi t))$, $S_2(t) = \theta_k(-\sin(\pi t))$. It is a straightforward exercise to see that the entire construction can be implemented in a $\{C_{\mathbb{R}}, +, \times, \theta_k\}$-FIC (use Theorem 11 and Lemma 15). ■

It is important to remark two facts: (i) This simulation is done in the sense indicated after Corollary 13, but with a slight modification (substitute $[2k-\varepsilon, 2k+\varepsilon]$ by $[2k, 2k+\varepsilon]$); (ii) This simulation does not enable us to simulate Type-2 machines [33]. Indeed, the construction presented in [16, theorem 2] only allows the encoding of a tape with a finite number of non-blank symbols.

Corollary 21 *For each fixed $k \geq 2$ and for every Turing machine M, there is an $\{C_{\mathbb{R}}, +, \times, \theta_k\}$-IC $\mathcal{U}$ simulating it. The initial input of M corresponds to integer initial settings for $\mathcal{U}$.*

Proof. M can be simulated by a $\{C_{\mathbb{R}}, +, \times, \theta_{k+1}\}$-FIC. Hence, by Theorem 16, M can be simulated by an $\{C_{\mathbb{R}}, +, \times, \theta_{k+1}, \theta'_{k+1}\}$-IC. But $\theta'_{k+1}(t) = (k+1)\theta_k(t)$ and $\theta_{k+1}(t) = t\theta_k(t)$. Therefore, M can be simulated by an $\{C_{\mathbb{R}}, +, \times, \theta_k\}$-IC. ■

7 Γ is GPAC-computable

In this section we show that the Gamma function can be computed by a PGPAC. This might seem contrary to reason in virtue of Theorem 5, since Γ is not differentially algebraic. However, we can achieve computability of this function by *changing our notion of computability for the GPAC.*

Indeed, it is a classical result in computable analysis that Γ is computable (cf. [25]). So, it might seem that the PGPAC is a less powerful model because it cannot compute Γ. However, in [7, pp. 657-658] it is referred that this comparison is based on two non-equivalent definitions of computability and, therefore, different arguments are needed. In fact, we will prove in this section that if we redefine our notion of GPAC-computability in a manner that it matches more closely the philosophy underlying computable analysis, then one can compute the Gamma function as well as Riemann's Zeta function.

Remark that within the traditional framework, outputs of a GPAC are usually provided in real time, i.e., once some input t is presented to the circuit, the output $f(t)$ is immediately updated. Therefore, computations take 'time 0' to carry out (cf. Fig. 6). But this is not what happens with computable analysis. Indeed, one of the basic concepts of this theory is that f is computable if for each x one can approximate $f(x)$ to any extent, in an effective manner, probably using the information encoded in the input x.

So, we introduce a similar notion of computability for the GPAC as follows. Let $\|\cdot\|_\infty$ be the sup-norm defined in $\mathbb{R}^n$ by $\|(x_1, ..., x_n)\|_\infty = \max\{|x_1|, ..., |x_n|\}$.

[6] Notice that for $n = 2, 3, ...$, the term $\sin(nx)$ is divisible by $\sin(x)$ (this can be proved by induction) and then the function h_p presented in this paper consists of a multipolynomial with terms $\sin(x)$ and $\cos(x)$. This argument was pointed out by José Félix Costa.

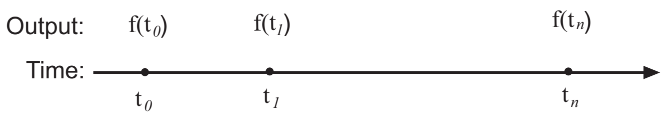

Figure 6: A GPAC computes in 'real time.'

Definition 22 *A function $f : \mathbb{R}^n \to \mathbb{R}^k$ is generated by a GPAC via approximations if there exists some GPAC $\mathcal{U}$ with input t and at least n integrators admitting initial settings $x_1, ..., x_n$, such that*

$$\|f(x_1, ..., x_n) - g(x_1, ..., x_n, t)\|_\infty \leq \varepsilon(x_1, ..., x_n, t), \tag{5}$$

with $\lim_{t\to\infty} \varepsilon(x_1, ..., x_n, t) = 0$, where $g(x_1, ..., x_n, t)$ is a vector constituted by k outputs of $\mathcal{U}$ and ε is one output of $\mathcal{U}$.

Therefore, the definition basically says that one can approximate up to any preassigned precision the value of $f(x)$ with the help of a GPAC. In fact, one just have to wait the necessary time to achieve the preassigned precision δ. Moreover, in each instant t, one has an upper bound for the error δ, that is given by output ε from the GPAC (this is intended to be the GPAC equivalent for "efective convergence" from computable analysis. More details are provided below). Similarly, one can define generability via approximations for PGPACs, $\mathcal{F}$-ICs, etc. It is also worthwhile to remark that, due to the connections between PGPAC-computable functions and $\mathbb{R}$-recursive functions presented in [11], if f is generated by a PGPAC via approximations, then f belongs to the class H_1 presented in [20].[7] Moreover, f is also computable by Rubel's Extended Analog Computer [29].

Notice that t represents the time, that might not necessarily correspond to physical time. It is also important to remark that this approach is natural from the dynamical systems point of view. We present some input via initial setting, thereby determining the complete behavior of the system for this input. Then we consider another input, the time, that allows us to observe the evolution of the system (that is seen as a process of computation).

The reader that is familiar to computable analysis may ask why we did not use, for example, the function $1/2^t$ instead of ε in (5). This is due to a more general problem concerning analog computation. For instance, if t is the time input of $\mathcal{U}$, then one can obtain a circuit calculating the exponential function e^{-t} (cf. Fig. 3) and can link it to the time input of $\mathcal{U}$. In this manner one can exponentially accelerate the computation. Moreover, if instead of a circuit that generates e^t one uses a circuit generating $\tan(t)$, then one can approximate $f(x)$ to any extent only by using values of t in $[0, \pi/2)$. This is in some sense similar to the "compression trick" presented by Moore in [19], where infinite computations

[7]Provided we allow the use of all reals and not only the constants $-1, 0, 1$.

can be carried out "within finite time." Therefore, one can always accelerate the computation process in order to capture the bound $1/2^t$.

However, since ε is computed by the same GPAC as g, this speed-up procedure also speeds-up the upper bound ε, thereby providing a more natural complexity measure for the computation of f. Hence, Definition 22 presents a complexity measure ε "robust to speed-up procedures." Let us now present an interesting result.

Theorem 23 *Function* Γ *is generated by a PGPAC via approximations in* $(0, +\infty)$.

Proof. To prove this result, we rely on Corollary 12. The idea is to generate the function given by $f_x(t) = t^{x-1}e^{-t}$ and to integrate it from 0 to ∞. Note that this integral only converges if $x \in (0, +\infty)$. It is easily seen that f_x is a solution of

$$y' = \frac{(x-1)}{t}y - y, \qquad y(1) = \frac{1}{e}. \tag{6}$$

However, y will not be defined at $t = 0$.[8] So, a few more steps are required to compute Γ. From (2), one gets

$$\Gamma(x) = \int_0^1 t^{x-1}e^{-t}dt + \int_1^\infty t^{x-1}e^{-t}dt.$$

Substituting $w = 1/t$ in the first integral, one has

$$\int_0^1 t^{x-1}e^{-t}dt = \int_1^\infty \left(\frac{1}{w}\right)^{x+1} e^{-\frac{1}{w}}dw.$$

or

$$\Gamma(x) = \int_1^\infty \left(\frac{1}{t}\right)^{x+1} e^{-\frac{1}{t}} + t^{x-1}e^{-t}dt. \tag{7}$$

Here one can consider $t_0 = 1$ and that the computation runs through $t \to \infty$. In this manner, $\frac{1}{t}$ is generated by a PGPAC since it is the solution of

$$y' = -y^2, \qquad y(1) = 1.$$

Using Corollary 12 and equation (6), one concludes that f_x can be generated by a PGPAC, where x is given by a constant unit. Noting that

$$\left(\frac{1}{t}\right)^{x+1} e^{-\frac{1}{t}} = \frac{f_x(t^{-1})}{t^2}$$

one can conclude that the integrand part of the integral given in (7) can be generated by some PGPAC. Using the output of this PGPAC as an input for an integrator (integrand input), and taking t for the other input (variable of

[8]This problem arises for powers like $g_x(t) = t^x$, because it is only defined for $t > 0$ (x may take negative values).

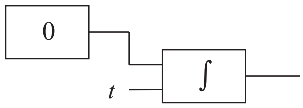

Figure 7: If the integrator has initial setting x, then the output of the integrator will always be x during the whole computation.

integration input), one concludes that the output of this integrator converges to (2) as $t \to \infty$. Call this circuit $\mathcal{U}'$.

We now want to check that (5) is satisfied. By taking Taylor's expansion of $z^{x-1}e^{-z}$ with $z \geq 0$, the reader may verify that

$$z^{x-1}e^{-z} = z^{x-1}\left(\sum_{i=0}^{\infty}\frac{z^i}{i!}\right)^{-1} \leq z^{x-1}\left(\frac{z^k}{k!}\right)^{-1} \leq k!z^{-2},$$

as long as $k \geq x+1$ is an integer and $z \geq 1$. Taking $k = \lceil x+1 \rceil$ and $t \geq 1$, one has

$$\left|\int_1^{\infty} z^{x-1}e^{-z}dz - \int_1^t z^{x-1}e^{-z}dz\right| = \int_t^{\infty} z^{x-1}e^{-z}dz \leq \frac{k!}{t} \leq \frac{(x+2)^{(x+2)}}{t}.$$

Moreover, for $z \geq 0$, $z^{x-1}e^{-z} \leq z^{x-1}$ and

$$\int_0^a z^{x-1}e^{-z}dz \leq \frac{a^x}{x}.$$

Hence, for $t \geq 1$,

$$\left|\int_1^{\infty}\left(\frac{1}{w}\right)^{x+1}e^{-\frac{1}{w}}dw - \int_1^t\left(\frac{1}{w}\right)^{x+1}e^{-\frac{1}{w}}dw\right| = \int_t^{\infty}\left(\frac{1}{w}\right)^{x+1}e^{-\frac{1}{w}}dw =$$
$$= \int_0^{1/t} z^{x-1}e^{-z}dz \leq \frac{t^{-x}}{x}$$

Therefore, at time $t \geq 1$ (note that $t_0 = 1$), the output $\int_1^t \left(\frac{1}{z}\right)^{x+1} e^{-\frac{1}{z}} + z^{x-1}e^{-z}dz$ of $\mathcal{U}'$ will approximate $\Gamma(x)$, with an upper bound for the error given by

$$\frac{t^{-x}}{x} + \frac{(x+2)^{(x+2)}}{t}.$$

This later function converges to 0 as $t \to \infty$ and can also be generated by a PGPAC $\mathcal{U}''$. Now consider the circuit of Fig. 7. Its output has value x, where x is the initial setting of the integrator. Replacing the constant units with output x in $\mathcal{U}'$ and $\mathcal{U}''$ by the circuit of Fig. 7, and putting these circuits in parallel, one gets a PGPAC that generates Γ via approximations ■

Another function known not to be differentially algebraic is Riemann's Zeta function. On the real line, with $x > 1$, it can be defined by

$$\zeta(x) = \frac{1}{\Gamma(x)} \int_0^\infty \frac{u^{x-1}}{e^u - 1} du. \tag{8}$$

Using an argument similar to the one employed for the Gamma function, one can show the following:

Theorem 24 *Function ζ is generated by a PGPAC via approximations in* $(1, +\infty)$.

Notice that the previous results hold even if we restrict the PGPAC to use only units associated to computable values (in the sense of computable analysis).

It is important to mention that all the limiting results concerning the GPAC are only valid when 'real time' computation is used. So, one should investigate to which extent the mathematical limitations of the GPAC presented in [27] also apply to computability via approximations.

8 Conclusion

We have introduced a model (PGPAC) based on Shannon's GPAC and we have shown that this model presents some characteristics that make it more suitable than Shannon's and Pour-El's GPAC. Moreover, we have extended this model, showing that some of these extensions are Turing universal, and also established links with the theory of continuous time dynamical systems.

However, there still are many open questions. For instance, one could ask under which assumptions the PGPAC lead to computable functions, in the sense of computable analysis. In [24] it is presented a very interesting work on these ideas, although for a different model. In particular, Pour-El shows the following. Let f be an analytic differentially algebraic function (functions generated by PGPACs are of this type). Hence, locally, f can be expressed as $f(x) = \sum_{i=0}^{\infty} b_i (x-c)^i$. Then Pour-El proves that the sequence of coefficients $\{b_i\}$ is "essentially computable in a finite number of the b_i's and c."[9] In particular, if c and all the b_i's are computable, then f is computable. In general f is not computable because f might be the constant function c_k, where k is not computable. But even in the case where all units of a GPAC are associated to computable values, it is unknown whether f should have a series expansion with computable coefficients and be therefore computable.

Another different path is followed in [8], [2]. Namely, Campagnolo, Costa, and Moore showed that restricted forms of integration lead to a hierarchy of continuous time systems related to the Grzegorczyk hierarchy over the naturals. In some sense, this can be captured by GPACs if we allow weaker forms

[9]The reader might refer to section 1 of [24] to see the exact meaning of this expression. Roughly, essential computability in $b \in \mathbb{R}$ uses standard computability, with the aid of a function $f(n)$ that gives the decimal expansion of b upon to n digits.

of integrations in the integrator units. More recently, Bournez and Hainry generalized this result to the case of the reals. The idea was to introduce a new operator computing restricted versions of limits. It would be interesting if some links could be established between this later work and the contents of Section 7.

For the sake of completeness we mention that Campagnolo et al. also presented a conjecture [7, Conjecture 1], where functions generated by PGPACs that only have access to rational constants in their initial conditions and parameters, are expected to have primitive recursive upper bounds.

Some directions for further research can be pointed out. Let us present some of them.

1. Can a PGPAC simulate a Turing machine? Indeed, all existing simulations of Turing machines always involve some non-analytic function (in this case θ_k). But is this kind of function really necessary? And what about the simulation of Type-2 machines?

2. How can we precisely define a notion of complexity for the models introduced above? And can we present connections between this theory and the theory of dynamical systems?

3. Is it possible to establish connections with computable analysis? In particular, can we restrict the constant units in Fig. 1 to some values (e.g. $-1, 0, 1$) and then relate PGPAC-computability with computability from the computable analysis point of view?

Acknowledgments. The author would like to thank Manuel Campagnolo for many helpful comments and suggestions. In particular, it was M. Campagnolo who realized the potential of Corollary 12 and that foresaw the possibility of establishing connections between GPAC-circuits and dynamical systems. Vasco Brattka also helped to clarify some points concerning computable analysis, especially in aspects regarding the computability of the Gamma function. The author would also like to thank the anonymous referees for their suggestions and remarks.

A special thanks goes to José Félix Costa for introducing the author to the GPAC model. Indeed, it was him who supplied the author with the necessary background to deal with the model and it was under its supervision that some earlier research was done and that the results presented in [11] were obtained. Moreover, all historical references presented throughout the text were provided by Félix Costa.

This work was partially supported by *Fundação para a Ciência e a Tecnologia* (FCT) and FEDER via the Center for Logic and Computation - CLC, and via the project ConTComp POCTI/MAT/45978/2002. The author is also grateful to *Fundação Calouste Gulbenkian* for the support given to this research through the *Programa Gulbenkian de Estímulo à Investigação.*

References

[1] J. A. Anderson. *An Introduction to Neural Networks.* MIT Press, 1995.

[2] O. Bournez and E. Hainry. An analog characterization of computable functions over the real numbers. submitted for publication.

[3] M. D. Bowles. U. S. technological enthusiasm and british technological skepticism in the age of the analog brain. *IEEE Ann. Hist. Comput.*, 18(4):5–15, 1996.

[4] M. S. Branicky. Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoret. Comput. Sci.*, 138(1):67–100, 1995.

[5] V. Bush. The differential analyzer. A new machine for solving differential equations. *J. Franklin Inst.*, 212:447–488, 1931.

[6] M. L. Campagnolo. The complexity of real recursive functions. In C. S. Calude, M. J. Dinneen, and F. Peper, editors, *Unconventional Models of Computation (UMC'02)*, LNCS 2509, pages 1–14. Springer, 2002.

[7] M. L. Campagnolo, C. Moore, and J. F. Costa. Iteration, inequalities, and differentiability in analog computers. *J. Complexity*, 16(4):642–660, 2000.

[8] M. L. Campagnolo, C. Moore, and J. F. Costa. An analog characterization of the Grzegorczyk hierarchy. *J. Complexity*, 18(4):977–1000, 2002.

[9] J. Copeland. Even Turing machines can compute uncomputable functions. In J. Casti, C. Calude, and M. Dinneen, editors, *Unconventional Models of Computation (UMC'98)*, pages 150–164, 1998.

[10] General Electric Management Consultant Services Division. *The Next Step in Management ... an Appraisal of Cybernetics*, 1952.

[11] D. S. Graça and J. F. Costa. Analog computers and recursive functions over the reals. *J. Complexity*, 19(5):644–664, 2003.

[12] S. Haykin. *Neural Networks - A Comprehensive Foundation.* Prentice Hall, 1999.

[13] M. W. Hirsch and S. Smale. *Differential Equations, Dynamical Systems, and Linear Algebra.* Academic Press, 1974.

[14] K.-I Ko. *Computational Complexity of Real Functions.* Birkhäuser, 1991.

[15] P. Koiran, M. Cosnard, and M. Garzon. Computability with low-dimensional dynamical systems. *Theoret. Comput. Sci.*, 132:113–128, 1994.

[16] P. Koiran and C. Moore. Closed-form analytic maps in one and two dimensions can simulate Turing machines. *Theoret. Comput. Sci.*, 210(1):217–223, 1999.

[17] L. Lipshitz and L. A. Rubel. A differentially algebraic replacement theorem, and analog computability. *Proc. Amer. Math. Soc.*, 99(2):367–372, 1987.

[18] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.*, 5:115–133, 1943.

[19] C. Moore. Recursion theory on the reals and continuous-time computation. *Theoret. Comput. Sci.*, 162:23–44, 1996.

[20] J. Mycka and J. F. Costa. Real recursive functions and their hierarchy. submitted for publication.

[21] J. M. Nyce. Guest editor's introduction. *IEEE Ann. Hist. Comput.*, 18:3–4, 1996.

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

[23] L. Owens. Where are we going, Phil Morse? Changing agendas and the rhetoric of obviousness in the transformation of computing at MIT, 1939-1957. *IEEE Ann. Hist. Comput.*, 18:34–41, 1996.

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

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

[26] S. Puchta. On the role of mathematics and mathematical knowledge in the invention of Vannevar Bush's early analog computers. *IEEE Ann. Hist. Comput.*, 18:49–59, 1996.

[27] L. A. Rubel. Some mathematical limitations of the general-purpose analog computer. *Adv. Appl. Math.*, 9:22–34, 1988.

[28] L. A. Rubel. A survey of transcendentally transcendental functions. *Amer. Math. Monthly*, 96(9):777–788, 1989.

[29] L. A. Rubel. The extended analog computer. *Adv. Appl. Math.*, 14:39–50, 1993.

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

[31] H. T. Siegelmann. *Neural Networks and Analog Computation: Beyond the Turing Limit*. Birkhäuser, 1999.

[32] S. Smale. *Mathematical Research Today and Tomorrow*, chapter Theory of computation, pages 59–69. Springer, 1992.

[33] K. Weihrauch. *Computable Analysis: An Introduction*. Springer, 2000.