Iteration, Inequalities, and Differentiability in Analog Computers

Manuel Lameiras Campagnolo[1,2], Cristopher Moore[2], and José Félix Costa[3]

[1] Departamento de Matemática, Instituto Superior de Agronomia, Lisbon University of Technology, Tapada da Ajuda, 1399 Lisboa Cx, Portugal `mlc@math.isa.utl.pt`
[2] Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, New Mexico 87501 `moore@santafe.edu`
[3] D.M./I.S.T., Universidade Técnica de Lisboa, Av. Rovisco Pais, 1049-001 Lisboa, Portugal `fgc@math.ist.utl.pt`

Abstract. Shannon's General Purpose Analog Computer (GPAC) is an elegant model of analog computation in continuous time. In this paper, we consider whether the set $\mathcal{G}$ of GPAC-computable functions is closed under iteration, that is, whether for any function $f(x) \in \mathcal{G}$ there is a function $F(x,t) \in \mathcal{G}$ such that $F(x,t) = f^t(x)$ for non-negative integers t. We show that $\mathcal{G}$ is not closed under iteration, but a simple extension of it is. In particular, if we relax the definition of the GPAC slightly to include unique solutions to boundary value problems, or equivalently if we allow functions $x^k\theta(x)$ that sense inequalities in a differentiable way, the resulting class, which we call $\mathcal{G} + \theta_k$, is closed under iteration. Furthermore, $\mathcal{G} + \theta_k$ includes all primitive recursive functions, and has the additional closure property that if $T(x)$ is in $\mathcal{G}+\theta_k$, then any function of x computable by a Turing machine in $T(x)$ time is also.

Key words: Analog computation, recursion theory, iteration, differentially algebraic functions, primitive recursive functions

1 Introduction

There has been a recent resurgence of interest in analog computation, the theory of computers whose states are continuous rather than discrete (see for instance [BSS89,Meer93,SS94,Sie98,BCSS98,Moo98]). However, in most of these models, time is still discrete; just as in classical computation, 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 this direction, studying computation by continuous-time dynamical systems [Moo90,Moo96,Orp97,Orp97a,SF98], no particular set of definitions 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 go back to the roots of analog computation theory by starting with Claude Shannon's General Purpose Analog Computer (GPAC). This was defined as a mathematical model of an analog device, the Differential Analyser, the fundamental principles of which were described by Lord Kelvin in 1876 [Tho76]. The Differential Analyser was developed at MIT under the supervision of Vannevar Bush and was indeed built in 1931, and rebuilt, with important improvements, in 1941. The Differential Analyser's input was the rotation of one or more drive shafts and its output was the rotation of one or more output shafts. The main units were gear boxes and mechanical friction wheel integrators, the latter invented by the Italian scientist Tito Gonella in 1825 [Bow96].

Just as polynomial operations are basic to the Blum-Shub-Smale model of analog computation [BSS89,BCSS98], polynomial differential equations are basic to the GPAC. Shannon [Sha41] showed that the GPAC generates exactly the *differentially algebraic* functions, which are unique solutions of polynomial differential equations. This set of functions includes simple functions like e^x and $\sin x$ as well as sums, products, and compositions of these, and solutions to differential equations formed from them such as $f' = \sin f$. Pour-El [PE74] made this proof rigorous by introducing the crucial notion of the *domain of generation*, thus showing that the differentially algebraic functions are precisely equivalent to GPAC.

Rubel [Rub93] proposed the Extended Analog Computer (EAC), which computes all functions computed by the GPAC but also produces the solutions of a broad class of Dirichlet boundary-value problems for partial differential equations. Rubel stresses that the EAC is a conceptual computer and that it is not known if it can be realized by actual physical, chemical or biological devices.

The gamma function $\Gamma(x)$ is computable by the EAC but not by the GPAC, since it is not differentially algebraic [Ost25,Rub89b].

Another extension of the GPAC, the set of $\mathbb{R}$-recursive functions, was proposed by Moore [Moo96]. Here we include a zero-finding operator analogous to the minimization operator μ of classical recursion theory. In the presence of a liberal semantics that allows functions to be composed with other functions even when they are undefined, this permits contraction of infinite computations into finite intervals, and renders the arithmetical and analytical hierarchies computable through a series of limit processes similar to those used by Bournez in [Bou99]. However, such an operator is clearly unphysical, except when the function in question is smooth enough for zeroes to be found in some reasonable way.

The μ-hierarchy stratifies the class of $\mathbb{R}$-recursive functions according to the number of nested uses of the zero-finding operator. Moore calls the lowest level M_0, where μ is not used at all, the "primitive $\mathbb{R}$-recursive functions." In this paper, we will further restrict our definition of integration by requiring functions and their derivatives to be bounded in the interval on which they are defined, and we will show below that the resulting subset $\mathcal{G}$ of M_0 coincides with the set of GPAC-computable functions.[1]

We propose here a new extension of $\mathcal{G}$. We keep the operators of the GPAC the same — integration and composition — but add piecewise-analytic basis functions such as $\theta_k(x) = x^k\theta(x)$ where $\theta(x)$ is the Heaviside step function $\theta(x) = 1$ for $x \geq 0$ and $\theta(x) = 0$ for $x < 0$. Allowing these functions can be thought of as allowing our analog computer to measure inequalities in a $(k-1)$-times differentiable way. By adding these to the basis set, we get a class $\mathcal{G} + \theta_k$ for each k. These functions are unique solutions of differential equations such as $xy' = ky$ if we define two boundary conditions rather than just an initial condition, which is a slightly weaker definition of uniqueness than that used by Pour-El to define GPAC-computability.

Iteration is a basic operation in recursion theory. If a function $f(x)$ is computable, so is $F(x,t) = f^t(x)$, the t'th iterate of f on x. We will ask whether these analog classes are closed under iteration, in the sense that if $f(x)$ is in the class, then so is some $F(x,t)$ that equals $f^t(x)$ when t is restricted to the natural numbers. Our main result is that $\mathcal{G} + \theta_k$ is closed under iteration for any $k > 1$, but $\mathcal{G}$ is not.

[1] It is erroneously stated in [Moo96] that all of M_0 is GPAC-computable; this is false since M_0 contains non-analytic functions like $\sqrt{x^2} = |x|$. Bounding the derivatives prevents such functions.

We will start by recalling the theory of $\mathbb{R}$-recursive functions [Moo96] and establishing the equivalence between GPAC and $\mathcal{G}$, the class of primitive $\mathbb{R}$-recursive functions whose derivatives are bounded on the interval on which they are defined. Then, we relax our notion of uniqueness and show how the functions θ_k can be defined using boundary value problems. Adding these functions to the GPAC gives the classes $\mathcal{G} + \theta_k$.

We then define the iteration functional and show that $\mathcal{G}$ is not closed under it. In particular, the iterated exponential function is not in $\mathcal{G}$. In $\mathcal{G} + \theta_k$, on the other hand, we can build "clock functions" such as those used in [Bra95,Moo96] to show that $\mathcal{G} + \theta_k$ is closed under iteration for any $k > 1$. It then follows that $\mathcal{G} + \theta_k$ includes all primitive recursive functions. Furthermore, $\mathcal{G} + \theta_k$ is closed under time complexity, in the sense that if $T(x)$ is in $\mathcal{G} + \theta_k$, then so is any function computable by a Turing machine in $T(x)$ steps. Finally, we end with some open questions, such as whether $\mathcal{G} + \theta_k$ includes the Ackermann function.

2 GPAC and $\mathbb{R}$-recursive functions

The general-purpose analog computer (GPAC) is a general model of a computer evolving in continuous time. 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 $\sin t$.

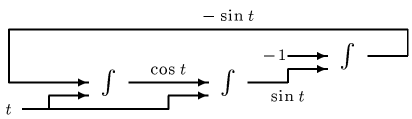

Fig. 1. A simple GPAC circuit that calculates $\sin t$. Its initial conditions are $\sin(0) = 0$ and $\cos(0) = 1$. The output w of the integrator unit $\int$ obeys $dw = u\,dv$ where u and v are its upper and lower inputs.

Shannon [Sha41] showed that the class of functions generable in this abstract model is the set of solutions of a certain class of systems of quasilinear differential equations. Later, Pour-El [PE74] made this definition more precise, by requiring the uniqueness of the solution of the system for all initial values belonging to a closed set with non-empty interior called the *domain of generation* of the initial condition. We give here the general definition of a GPAC-computability for functions of several variables.

Definition 1 (Shannon, Pour-El). *A real-valued function $y : \mathbb{R}^m \to \mathbb{R}$ of m independent variables $\boldsymbol{x} = (x_1, ..., x_m)$ is GPAC-computable on a closed subset D of $\mathbb{R}^m$ if there exists a vector-valued function $\boldsymbol{y}(\boldsymbol{x}) = (y_1(\boldsymbol{x}), ..., y_n(\boldsymbol{x}))$ for some n, and an initial condition $\boldsymbol{y}(\boldsymbol{x}_0) = \boldsymbol{y}_0$ where $\boldsymbol{x}_0 \in D$, such that:*

1. *$y(\boldsymbol{x}) = y_1(\boldsymbol{x})$.*
2. *$\boldsymbol{y} = (y_1, ..., y_n)$ is the unique solution on D of a system of partial differential equations of the form*
$$A(\boldsymbol{x}, \boldsymbol{y})\, \boldsymbol{y}' = B(\boldsymbol{x}, \boldsymbol{y}) \tag{1}$$
satisfying the initial condition $\boldsymbol{y}(\boldsymbol{x}_0) = \boldsymbol{y}_0$, where A and B are $n \times n$ and $n \times m$ matrices respectively and $\boldsymbol{y}'$ is the $n \times m$ matrix of the derivatives of $\boldsymbol{y}$ with respect to $\boldsymbol{x}$. Furthermore, A and B must be linear in 1 and the variables $x_1, ..., x_m$, $y_1, ..., y_n$.
3. *$(\boldsymbol{x}_0, \boldsymbol{y}_0)$ has a* domain of generation*, that is, the solution to (1) remains unique under sufficiently small perturbations of the initial condition.*

We say that a vector-valued function $\boldsymbol{y} : \mathbb{R}^m \to \mathbb{R}^k$ for $k > 1$ is GPAC-computable if each of its components are.

Here $y_2, \ldots, y_n$ are additional variables representing the computer's internal states, and $y = y_1$ is its output. Note that the above definition implies that if $y(\boldsymbol{x})$ is GPAC-computable, then restricting any subset of the variables x_i results in a GPAC-computable function of the remaining variables, since A and B are then linear in 1 and the remaining variables. In particular, if we restrict all the variables but one, the resulting function of one variable is GPAC-computable. We will use this fact for the proof of proposition 8.

The following fundamental result [Sha41,PE74,LR87] establishes, for functions of one variable, a relationship between GPAC-computability and the class of differentially algebraic functions, that is, solutions of polynomial differential equations. We use $y^{(n)}$ to denote the n'th derivative of y.

Proposition 1 (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 condition on J then there is a closed subinterval $J' \subset J$ on which $y(x)$ is GPAC-computable.*

Next we recall recursion theory on the reals [Moo96], which is defined in analogy with classical recursion theory. A function $h : D \subset \mathbb{R}^m \to \mathbb{R}^n$ is $\mathbb{R}$-recursive if it can be inductively defined from the projections $U_i(\boldsymbol{x}) = x_i$, the constants 0 and 1, and the following operators.[2]

- *Composition*: if a p-ary function f and functions $g_1, \ldots, g_p$ of the same arity are $\mathbb{R}$-recursive, then $h(\boldsymbol{x}) = f(g_1(\boldsymbol{x}), ..., g_p(\boldsymbol{x}))$ is $\mathbb{R}$-recursive.
- *Integration*: if f and g are $\mathbb{R}$-recursive then the function h satisfying the equations $h(\boldsymbol{x}, 0) = f(\boldsymbol{x})$ and $\partial_y h(\boldsymbol{x}, y) = g(\boldsymbol{x}, y, h(\boldsymbol{x}, y))$ is $\mathbb{R}$-recursive, defined on the largest interval containing 0 on which it is finite and unique.
- *μ-recursion (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.

Clearly, this definition is intended as a continuous analog of classical recursion theory [Odi89], replacing primitive recursion and zero-finding on $\mathbb{N}$ with integration and zero-finding on $\mathbb{R}$.

The class of $\mathbb{R}$-recursive functions is very large. It contains many traditionally uncomputable functions, such as the characteristic functions of sets in the arithmetical and analytical hierarchies [Moo96,Odi89]. However, we can stratify this class by counting the number of nested uses of the μ-operator: 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.

Unlike the classical case in which one μ suffices, we believe that the continuous μ-hierarchy is distinct. For instance, the characteristic function $\chi_{\mathbb{Q}}$ of the rationals is in M_2 but not in M_1 [Moo96]. If μ is not used at all we get M_0, the "primitive $\mathbb{R}$-recursive functions." M_0 contains most common functions such as $x + y$, xy, e^x, $\sin x$, the inverses of these when defined, and constants such as e and π. However, M_0 also contains some functions with discontinuous derivatives, such as $|x| = \sqrt{x^2}$ and the sawtooth function $\sin^{-1}(\sin x)$.

To restrict M_0 further, and to make it more physically realistic, we require that functions defined by integration only be defined on the largest interval containing 0 on which they and their derivatives are bounded. This corresponds

[2] Strictly speaking, the projection and identity functions can be defined by integrating unit vectors.

to the physical requirement of bounded energy in an analog device. We call this operator *bounded integration*, and call the resulting class $\mathcal{G}$. To make this last definition more precise, we define $\mathcal{G}$ recursively as the smallest class of functions defined on some domain $D \subset \mathbb{R}^m \to \mathbb{R}^n$ containing 0, 1, -1 and the projections, and which is closed under composition and bounded integration. Functions in $\mathcal{G}$ are analytic, since constants and projections are analytic and composition and bounded integration preserve analyticity.

Next, we establish the equivalence between GPAC and $\mathcal{G}$ for functions of several variables on their domains. We also notice that those models are equivalent to the class of dynamical systems of the form $\boldsymbol{y}' = R(\boldsymbol{x}, \boldsymbol{y})$ where R is rational, that is, a quotient of two polynomials. This was shown in the context of control theory by Wang and Sontag [WS92].

Proposition 2. *Let $\boldsymbol{y} : D \subset \mathbb{R}^m \to \mathbb{R}^n$, with D closed and bounded. The following propositions are equivalent.*

1. *$\boldsymbol{y}$ is GPAC-computable,*
2. *$\boldsymbol{y}$ is the unique flow of a dynamical system $\boldsymbol{y}' = R(\boldsymbol{x}, \boldsymbol{y})$, where R is a matrix of rational functions,*
3. *$\boldsymbol{y}$ belongs to $\mathcal{G}$.*

Proof. ($1 \Rightarrow 2$) This is given in the proof of theorem 2 in [PE74].

($2 \Rightarrow 3$) Both polynomials and the function $f(x, y) = x/y$ are definable in $\mathcal{G}$ where the latter is defined either for $y > 0$ or $y < 0$ [Moo96]. By composition, these give us any rational function away from its singularities, and $\boldsymbol{y}$ such that $\boldsymbol{y}' = R(\boldsymbol{x}, \boldsymbol{y})$ is definable in $\mathcal{G}$ by integration.

($3 \Rightarrow 1$) The projections and the constants 0 and 1 are clearly GPAC-computable. Since functions in $\mathcal{G}$ are defined from simpler ones by composition and integration, we just have to show that GPAC-computability is preserved under both these operators.

For composition, Shannon [Sha41, theorems IV and VII] showed that if two functions f and g are GPAC-computable then their composition $h = f \circ g$ is also. In terms of circuits like those shown in figures 1 and 2, we simply plug the outputs of one function into the inputs of another.

For integration, we use the diagram in figure 2. Here we combine an integrator and an adder to match the definition of integration used in [Moo96]. $\square$

In the rest of the paper, we will use $\mathcal{G}$ interchangeably for the GPAC-computable functions, the differentially algebraic functions, and the subset of M_0 formed by bounded integration. In the next section, we will consider a natural extension of $\mathcal{G}$.

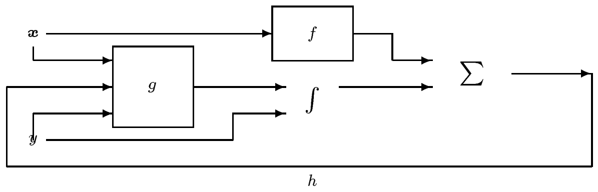

Fig. 2. A GPAC circuit for the definition of integration used in [Moo96], where $h(\boldsymbol{x}, 0) = f(\boldsymbol{x})$ and $\partial_y h(\boldsymbol{x}, y) = g(\boldsymbol{x}, y, h)$.

3 Extending GPAC with boundary value problems

3.1 The class θ

We will extend $\mathcal{G}$ with a set of functions $\theta_k(x) = x^k\theta(x)$, where $\theta(x)$ is the Heaviside step function

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

Each $\theta_k(x)$ can be interpreted as a function which checks inequalities such as $x \geq 0$ in a differentiable way, since θ_k is $(k-1)$-times differentiable. We will show in this section that allowing those functions is equivalent to relaxing slightly the definition of GPAC by considering a two point boundary value problem for equation (1), instead of just an initial condition. In this section we will consider the case where y is a function of one variable and A and B are scalars, so $A(x, y)\, y' = B(x, y)$ where A and B are linear in 1, x, and y.

Definition 2. *The function y belongs to the class θ if it is the unique solution on $I = [x_1, x_2] \subset \mathbb{R}$ of*

$$(a_0 + a_1 x + a_2 y)\, y' = b_0 + b_1 x + b_2 y \tag{2}$$

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 $y = 0$ for $x < 0$ and $y = x^2$ for $x \geq 0$, that is, $y = x^2\theta(x)$. If, instead, the boundary values are $y(1) = y(-1) = 1$ then the solution is $y = x^2$ and is in $\mathcal{G}$.

Note that (2) defines a rational flow,

$$y' = \frac{b_0 + b_1 x + b_2 y}{a_0 + a_1 x + a_2 y} \tag{3}$$

Such a flow may have singularities (x_0, y_0), defined by

$$a_0 + a_1 x_0 + a_2 y_0 = b_0 + b_1 x_0 + b_2 y_0 = 0 \tag{4}$$

At such a point, y' is undefined and the flow can branch into many different trajectories. Since the lines $a_0 + a_1 x_0 + a_2 y_0 = 0$ and $b_0 + b_1 x_0 + b_2 y_0 = 0$ typically cross at one point, this singularity is usually unique.[3] Thus we can say that functions in θ fulfill a condition similar to, but somewhat weaker than, the domain of generation of Pour-El. Note that functions in θ are differentiable and continuous on their domains.

Next, we show that y is piecewise GPAC-computable on I:

Proposition 3. *Let y be a function in θ defined on $I = [x_1, x_2]$ from equation (2) with boundary values $y(x_1) = y_1$ and $y(x_2) = y_2$. Let (x_0, y_0) be the singularity defined above. Then, either y is GPAC-computable on I or there are two GPAC-computable functions, f_1 and f_2, such that y can be written as $y(x) = f_1(x)$ if $x_1 \leq x \leq x_0$ and $y(x) = f_2(x)$ if $x_0 \leq x \leq x_2$, with $f_1(x_0) = f_2(x_0) = y_0$.*

Proof. We consider two cases. If $x_0 \notin I$ or $y(x_0) \neq y_0$, then the flow's trajectory will not pass through the singularity (x_0, y_0). Then y is defined uniquely by the initial condition $y(x_1) = y_1$ and is therefore in $\mathcal{G}$.

Now suppose that $x_0 \in I$ and $y(x_0) = y_0$. Since the singularity (x_0, y_0) is unique, equation (2) with initial condition $y(x_1) = y_1$ has a unique and therefore GPAC-computable solution $\tilde{f}_1$ that coincides with y on $[x_1, x_0)$. Now, since y is continuous at x_0, the limit $\lim_{x \to x_0^-} \tilde{f}_1(x)$ exists and equals y_0. Let f_1 be an extension of $\tilde{f}_1$ on x_0 such that $f_1(x_0) = y_0$. Then, f_1 is defined on $[x_1, x_0]$. Moreover, since $y'(x_0)$ exists and is finite, the left derivative of f_1 at x_0 exists and is finite too: its value is also $y'(x_0)$. Therefore, f_1 is the unique solution of (2) with initial condition $y(x_1) = y_1$ on $[x_1, x_0]$ and is GPAC-computable. The proof that f_2 is GPAC-computable on $[x_0, x_2]$ is similar. □

The last results show in particular that each branch of a function y in θ is analytic and, therefore, it can be written as one power series $f_1(x) = \sum_i \lambda_i^{(1)} (x - x_0)^i$ on $[x_1, x_0]$ and another power series $f_2(x) = \sum_i \lambda_i^{(2)} (x - x_0)^i$ on $[x_0, x_2]$, with $y'(a) = \lambda_1^{(1)} = \lambda_1^{(2)}$. Consequently, $y(x)$ is continuously differentiable.

Let's look closer at the class θ and see which functions belong to it. If y belongs to θ then it must satisfy the rational flow in equation (3). If we transform

[3] If these lines are parallel, we have a rational flow such as $y' = (x+y)/(x+y+1)$ and the solution is in $\mathcal{G}$ wherever it is defined. If they coincide, then $a_0 + a_1 x + a_2 y = C(b_0 + b_1 x + b_2 y)$ for constant C, and the solution of $y' = C$ is trivially in $\mathcal{G}$.

our variables to $x - x_0$ and $y - y_0$ we obtain an equivalent equation where $a_0 = b_0 = 0$ and the singularity is at $(0,0)$. Functions in θ then satisfy an equation of the form

$$(a_1 x + a_2 y)\, y' = b_1 x + b_2 y \tag{5}$$

Consider now the interval $[0, \epsilon]$ for small ϵ. We saw that it is possible to write $y(x) = \sum_i \lambda_i x^i$ on $[0, \epsilon]$ since each branch of y is analytic. Then, equation (5) turns into

$$\lambda_1 + 2\lambda_2 x + ... = \frac{(b_1 + b_2\lambda_1)x + b_2\lambda_2 x^2 + ...}{(a_1 + a_2\lambda_1)x + a_2\lambda_2 x^2 + ...}$$

on $(0, \epsilon]$ (note that $y(0) = 0$ in the new variables). Taking the limit $x \to 0^+$ of both sides we get

$$\lambda_1(a_1 + a_2\lambda_1) = b_1 + b_2\lambda_1 \tag{6}$$

Since $\lambda_1 = y'(0)$ exists and is real, the discriminant of (6) must fulfill $r = (b_2 - a_1)^2 + 4a_2 b_1 \geq 0$. From now on we'll just consider the case $r > 0$.

Our next goal is to show that all functions in θ belong, under linear transformations, to a simple class of functions. Formally, we will prove that

Proposition 4. *For any function $y(x)$ which solves equation (5) with $r = (b_2 - a_1)^2 + 4a_2 b_1 > 0$, there is an invertible linear transformation $\binom{X}{Y} = T\binom{x}{y}$ with $T_{22} \neq 0$ such that $Y = c^-|X|^\gamma$ if $x \leq 0$ and $Y = c^+|X|^\gamma$ if $x > 0$, for some constants $c^-, c^+ \in \mathbb{R}$ and some $\gamma > 1$.*

Proof. Consider the following linear autonomous system in two dimensions,

$$\frac{dz}{dt} = Az \qquad \text{with} \qquad z = \begin{pmatrix} x \\ y \end{pmatrix} \qquad \text{and} \qquad A = \begin{pmatrix} a_1 & a_2 \\ b_1 & b_2 \end{pmatrix} \tag{7}$$

It is easy to show that the trajectories of $z(t)$ in the (x,y)-plane must solve (5) when $a_1 x + a_2 y \neq 0$. The shape of the trajectory near the origin depends on the eigenvectors and eigenvalues of A. Since the discriminant r of equation (6) is positive by assumption, A has two distinct real eigenvalues $\gamma_1, \gamma_2 = (b_2 + a_1 \pm \sqrt{r})/2$. These must have the same sign, or all trajectories diverge from the origin [HK91] so $a_2 b_1 < b_2 a_1$.

Since A is invertible in non-trivial cases of (5), it can be diagonalized with some linear transformation T. Thus we can convert (7) to $\frac{dw}{dt} = Dw$ where $w = Tz$ and $D = TAT^{-1}$ is the diagonal matrix whose entries are the eigenvalues γ_1, γ_2 of A. The solution of this new system is $w = (c_1 e^{\gamma_1 t}, c_2 e^{\gamma_2 t})$ for arbitrary constants c_1 and c_2. If we write X and Y for w_1 and w_2 respectively, eliminating t gives either $X = 0$ or

$$Y = c\,|X|^\gamma \qquad \text{where} \qquad \gamma = \gamma_2/\gamma_1 \qquad \text{and} \qquad c = c_2/c_1^{\gamma_2/\gamma_1} \tag{8}$$

for each branch of the trajectory coming out of the origin. We can assume that $\gamma > 1$ by switching the two coordinates if necessary, and writing X and Y for w_2 and w_1 instead. Since the two branches must meet in a differentiable way, they are either both vertical or both satisfy (8) for some $\gamma > 1$, possibly with different constants $c = c^+, c^-$ on each side. Finally, for the solution $y(x)$ in the original coordinates to be differentiable at $x = 0$, Y must have a nonzero coefficient in y, i.e. $T_{22} \neq 0$. This is shown in figure 3. □

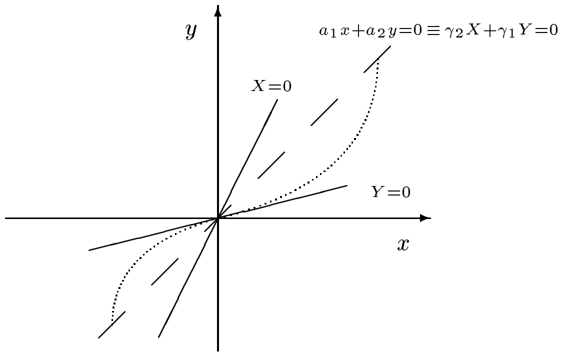

Fig. 3. The relevant directions of the (x, y)-plane for solutions of equation (5). The dotted line shows a function of the form discussed in proposition 4 in the neighborhood of $(0, 0)$.

3.2 The classes $\mathcal{G} + \theta_k$

We have then a class of GPAC-computable functions $\mathcal{G}$ and a class θ that contains some functions which don't belong to $\mathcal{G}$. In analogy with oracles, for any function f we will define the class $\mathcal{G} + f$ as the smallest class of functions containing 0, 1, -1, the projections and f, and which is closed under composition and bounded integration. We can define $\mathcal{G} + S$ for sets of functions S in the same way. In physical terms, these classes represent the functions computable by a GPAC with an expanded set of components, namely integrators, adders, and "black boxes" that compute f.

In this subsection, we discuss the family of classes $\mathcal{G} + \theta_k$, where we adjoin the function $\theta_k(x) = x^k\theta(x)$. First, we show that this family, with the appropriate k, contains any function in the class θ as defined above.

Proposition 5. *If $y(x) \in \theta$ is of the form defined in proposition 4 with exponent γ then $y(x) \in \mathcal{G} + \theta_\gamma$.*

Proof. We know from proposition 4 that each branch of any function $y(x)$ in θ must satisfy a pair of equations of the form

$$\begin{array}{ll} F^-(x,y) = Y - c^-|X|^\gamma = T_{21}x + T_{22}y - c^-|T_{11}x + T_{12}y|^\gamma = 0 & (x < 0) \\ F^+(x,y) = Y - c^+|X|^\gamma = T_{21}x + T_{22}y - c^+|T_{11}x + T_{12}y|^\gamma = 0 & (x \geq 0) \end{array}$$

Next we will define a function $F(x)$ in $\mathcal{G} + \theta_\gamma$ such that $F = F^-$ if $x < 0$ and $F = F^+$ if $x \geq 0$. Wherever y is defined, the sign of X is either the same as that of x or $-x$. In the former case, we let

$$F(x) = T_{21}x + T_{22}y - c^+\theta_\gamma(x) + c^-\theta_\gamma(-x)$$

and in the latter we switch c^+ and c^-.

Finally, we use the implicit function theorem to show that y can be defined in $\mathcal{G} + \theta_k$. Necessary conditions are fulfilled since $F(x,y)$ is continuously differentiable and $\partial_y F(0,0) = T_{22} \neq 0$. Therefore, $F(x,y) = 0$ defines implicitly a function $y(x)$ in a neighborhood of 0. On that neighborhood, $y(x)$ is definable in $\mathcal{G} + \theta_k$ by integration: $y'(x) = -\partial_x F(x,y(x))/\partial_y F(x,y(x))$, with the initial condition $(x,y) = (0,0)$. □

The classes $\mathcal{G}+\theta_k$ for various k inherit their various degrees of differentiability from θ_k. Thus $\mathcal{G}+\theta_k$ represents the power of a GPAC which can check inequalities in a $(k-1)$-times differentiable way:

Proposition 6. *Any function in $\mathcal{G} + \theta_k$ is $(\lceil k \rceil - 1)$-times differentiable.*

Proof. Composition and bounded integration preserve j-times differentiability for any j, and θ_k is $(\lceil k \rceil - 1)$-times differentiable. □

The classes $\mathcal{G} + \theta_k$ then form a distinct hierarchy:

Proposition 7. *If $1 < j < k$, $\mathcal{G} + \theta_k \subset \mathcal{G} + \theta_j$. Moreover, if $k - j \geq 1$, this inclusion is proper.*

Proof. The function $y = x^a$ satisfies the differential equation $xy' = ay$ and the initial condition $y(1) = 1$. If $a > 1$, its derivative goes to zero at $x = 0$, so x^a is GPAC-computable for $x \geq 0$. Then since $\theta_k(x) = \theta_j(x)^a$ where $a = k/j > 1$, $\theta_k \in \mathcal{G} + \theta_j$ so $\mathcal{G} + \theta_k \subset \mathcal{G} + \theta_j$ follows by definition. If $k - j \geq 1$, this inclusion is proper since anything in $\mathcal{G} + \theta_k$ is $(\lceil k \rceil - 1)$-times differentiable but θ_j is not. □

In the next section, we will compare $\mathcal{G}$ and $\mathcal{G} + \theta_k$ and show that $\mathcal{G}$ lacks an important closure property — closure under iteration — which these extensions $\mathcal{G} + \theta_k$ possess.

4 Iteration and primitive recursive functions

For any function f, we define its *iterate* $F(\boldsymbol{x}, t) = f^t(\boldsymbol{x})$, where $f^t(\boldsymbol{x})$ denotes the result of t successive applications of f on $\boldsymbol{x}$ (note that $f^0(\boldsymbol{x}) = \boldsymbol{x}$). Iteration is a fundamental operation in the classical theory of computation, as the following result makes clear:

Lemma 1. ([Odi89, proposition I.5.9 and remark]) *The class of primitive recursive functions is the smallest class of functions that: 1) contains the zero, successor, and projection functions; 2) is closed under composition; and 3) is closed under iteration.*

In this section, we will prove that $\mathcal{G}$ is not closed under iteration, but that $\mathcal{G} + \theta_k$ is closed under iteration for any $k > 1$. This last result together with the preceding lemma show that the class of primitive recursive functions is contained in $\mathcal{G} + \theta_k$. Here we adopt the convention that a function on $\mathbb{N}$ is in an analog class $\mathcal{C}$ if some extension of it to $\mathbb{R}$ is, i.e. if there is some function $\tilde{f} \in \mathcal{C}$ that matches f on inputs in $\mathbb{N}$.

To prove that $\mathcal{G}$ is not closed under iteration, we use a result of differential algebra regarding the iterated exponential function $\exp_n(x)$ defined by $\exp_0(x) = x$ and $\exp_n(x) = \exp(\exp_{n-1}(x))$. The following lemma is a particular case of a more general theorem of Babakhanian [Bab73, Theorem 2].

Lemma 2. *For each $n \geq 0$, $\exp_n(x)$ satisfies no non-trivial algebraic differential equation of order less than n.*

Then this gives us the following:

Proposition 8. *$\mathcal{G}$ 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 $n \in \mathbb{N}$.*

Proof. If such a function $F(x, n)$ is GPAC-computable, it must satisfy a system of differential equations $A\,\boldsymbol{y}' = B$, where $y_1 = F$, of some finite degree d. As we pointed out after the definition of GPAC-computability in section 2, if we fix n the resulting function $\exp_n$ of x has to satisfy a system of degree less than or equal to d. But lemma 2 says this is impossible for $n > d$, so by making n large enough we obtain a contradiction. □

Now, we show that $\mathcal{G} + \theta_k$ is closed under iteration. To build the iteration 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 an 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 unit period and the second variable is brought up to match it. Therefore, at time $t = 2$ both variables have the same value $f(\boldsymbol{x})$. This process is repeated until the desired number of iterations is obtained.

Proposition 9. *$\mathcal{G} + \theta_k$ 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 $t \in \mathbb{N}$.*

Proof. For simplicity, we will show how to iterate functions of one variable. Our simulation variables will be y_1 and y_2, and our clock functions will be $\theta_k(\sin \pi t)$ and $\theta_k(-\sin \pi t)$. We then have the following system of equations:

$$\begin{aligned} |\cos(\pi t/2)|^{k+1}\, y_1' &= -\pi(y_1 - f(y_2))\, \theta_k(\sin \pi t) \\ |\sin(\pi t/2)|^{k+1}\, y_2' &= -\pi(y_2 - y_1)\, \theta_k(-\sin \pi t) \end{aligned} \tag{9}$$

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

We will prove that $y_1(2t) = y_2(2t) = f^t(x)$ for all integer $t \geq 0$. We will consider the case where $k > 1$ is an odd integer; for even k the proof is slightly more complicated, and for non-integer k equation (9) seems to lack a closed-form solution, although the proof still holds. Suppose our initial conditions are $y_1(0) = y_2(0) = x$. On the interval $[0, 1]$, $y_2'(t) = 0$ because $\theta_k(-\sin \pi t) = 0$. Therefore, y_2 remains constant with value x. The solution for y_1 on this interval is then

$$y_1(t) = f(y_2) + cE \cos^{2^{k+1}}(\pi t/2)$$

where E is a finite expression of the form $\exp(\sum_j \alpha_j \cos(j\pi t))$ depending only on k, and c is a constant such that $y_1(0) = y_2(0) = x$. Thus $y_1(1) = f(y_2)$. A similar argument for y_2 on $[1, 2]$ shows that $y_2(2) = y_1(2) = f(x)$, and so on for y_1 and y_2 on subsequent intervals.

To check the differentiability of y_1 and y_2, note that on $[0, 1]$ the derivative of y_1 is then given by

$$y_1'(t) = \pi c E \frac{\cos^{2^{k+1}}(\pi t/2) \sin^k(\pi t)}{\cos^{k+1}(\pi t/2)}$$

This can be simplified to

$$y_1'(t) = -2^k \pi c E \cos^{2^{k+1}-1}(\pi t/2) \sin^k(\pi t/2).$$

using the relation $\sin 2s = 2 \sin s \cos s$. It is then easy to see that at least the first $k - 1$ right derivatives of y_1 vanish at $t = 0$ and at least the first $k - 1$ left derivatives vanish at $t = 1$. Moreover y_1 is constant on the interval $[1, 2]$ since $\theta_k(\sin \pi t) = 0$, so we conclude that y_1 is $(k-1)$-times differentiable on $[0, 2]$ and on subsequent intervals. The proof for y_2 is similar.

For general k, the proof relies on the local behavior of equation (9) in the neighborhood of $x = 2t$ and $x = 2t + 1$ for $t \in \mathbb{N}$. For instance, as $t \to 1$ from below, (9) 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 $t > 1$, $y_1(2t) = y_2(2t) = f^t(\boldsymbol{x})$. Thus we have shown that $F(\boldsymbol{x}, t) = y_1(2t)$ can be defined in $\mathcal{G} + \theta_k$, so $\mathcal{G} + \theta_k$ is closed under iteration. □

As an example, in figure 4 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 (9) using standard packages, so this system of differential equations actually works in practice.

Using lemma 1 gives the following corollary, again with the convention that $\mathcal{G} + \theta_k$ includes a function on $\mathbb{N}$ if it includes some extension of it to $\mathbb{R}$:

Corollary 1. *$\mathcal{G} + \theta_k$ contains all primitive recursive functions.*

Proof. Since $\mathcal{G} + \theta_k$ contains the zero function $Z(x) = 0$, the successor function $S(x) = x + 1$, and the projections $U_i^n(\boldsymbol{x}) = x_i$, and since it is closed under composition and iteration, it follows from lemma 1 that $\mathcal{G} + \theta_k$ contains all primitive recursive functions. □

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 $\mathcal{G} + \theta_k$ is *closed under time complexity* in the following sense:

Proposition 10. *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 $\mathcal{G} + \theta_k$.*

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

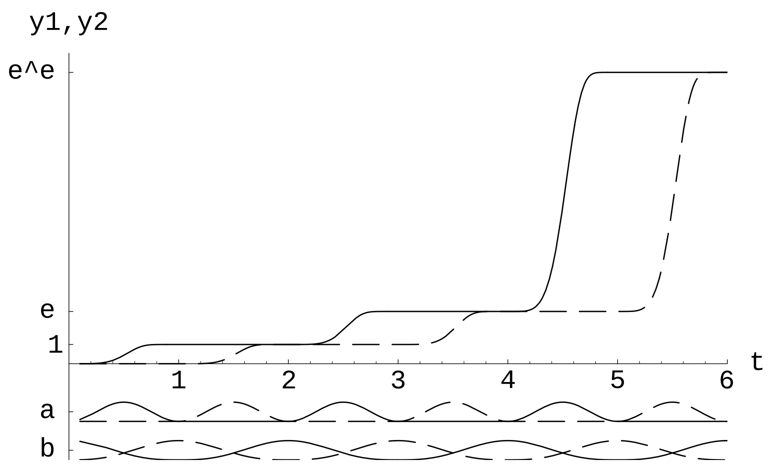

Fig. 4. A numerical integration of the system of equations (9) for iterating the exponential function. Here $k = 2$. The values of y_1 and y_2 at $t = 0, 2, 4, 6$ are 0, 1, e, and e^e respectively. On the graph below we show (a) the clock functions $\theta_2(\sin(\pi t))$, $\theta_2(\sin(-\pi t))$ and (b) the functions $|\cos(\pi t/2)|^3$, $|\sin(\pi t/2)|^3$.

be infinitely differentiable [Moo90], piecewise-linear [Moo90,KCG94], or closed-form analytic and composed of a finite number of trigonometric terms [KM99].[4] Thus there are explicitly definable functions in $\mathcal{G} + \theta_k$, or even $\mathcal{G}$, that can be used to make proposition 10 constructive.

Since any function computable in primitive recursive time is primitive recursive, proposition 10 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.

5 Conclusion and open problems

It was already believed that analog computers like Shannon's GPAC are not as powerful as Turing machines, since certain functions that are computable in the sense of recursive analysis (Euler's Γ, for instance) are not GPAC-computable [PE74,Rub89a]. However, this argument is based on two non-equivalent definitions of computability for real functions, one being related to effective convergence of rational sequences [Grz57,PER89], and the other being GPAC-computability. Here we have given a clearer answer to this question by exploring the property of closure under iteration.

We have shown that $\mathcal{G} + \theta_k$ includes all primitive recursive functions. One can ask if it includes non-primitive recursive functions such as the Ackermann function. It is believed, but not known [Hay96], that all differentially algebraic functions are bounded by some elementary function, i.e. $\exp_n(x)$ for some n, whenever they are defined everywhere on the complex plane. To match this conjecture that functions in $\mathcal{G}$ have elementary upper bounds, we suggest the following:

Conjecture 1. Functions $f(x)$ in $\mathcal{G} + \theta_k$ have primitive recursive upper bounds whenever they are defined for all $x > 0$.

We might try proving this conjecture by using numerical integration; for instance, GPAC-computable functions can be approximated by recursive functions. However, strictly speaking this approximation only works when a bound on the derivatives is known *a priori* [VSD86] or on arbitrarily small domains [Rub89a]. If this conjecture is false, then proposition 10 shows that $\mathcal{G} + \theta_k$ contains a wide variety of non-primitive recursive functions.

[4] In [KM99] a simulation in one dimension is achieved, but at the cost of an exponential slowdown.

As we saw, most commonly used functions in mathematics are generable by a GPAC. However, this is not the case for functions like Euler's Γ, Riemann's ζ and solutions to the Dirichlet problem on a disk [PE74,Rub88]. It is known that if the GPAC is extended with a restricted limit operator then Euler's Γ and Riemann's ζ functions become computable [Rub93] and, therefore, $\mathcal{G}$ is strictly included in $\mathcal{G} + \lim$. As we saw, $\mathcal{G} + \theta_k$ is larger than $\mathcal{G}$. It would be interesting to compare $\mathcal{G} + \theta_k + \lim$ with $\mathcal{G} + \theta_k$ or $\mathcal{G} + \lim$.

Acknowledgements. C.M. wishes to thank Olivier Bournez for pointing out a number of problems with the definition of integration in [Moo96], Bruce Litow, Ariel Scolnicov, John Tromp, and Eduardo Sontag for helpful conversations, and Molly Rose and Spootie the Cat for their support. M.L.C. thanks Evelyne Hubert for some remarks concerning real power series solutions of algebraic differential equations. This work was partially supported by FCT PRAXIS XXI/BD/18304/98, FLAD 754/98, JNICT COMBINA PBIC/C/TIT/2527/95.

References

[Bab73] A. Babakhanian. Exponentials in differentially algebraic extension fields. *Duke Math. J.*, 40:455–458, 1973.

[BSS89] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bull. Amer. Math. Soc.* 21:1–46, 1989.

[BCSS98] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation.* Springer-Verlag, 1998.

[Bou99] O. Bournez. Achilles and the tortoise climbing up the hyper-arithmetical hierarchy. *Theoretical Computer Science*, 210(1):21–71, 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.

[Grz57] A. Grzegorczyk. On the definition of computable real continuous functions. *Fund. Math.*, 44:61–71, 1957.

[HK91] J. Hale and H. Koçak. *Dynamics and Bifurcations.* Springer-Verlag, 1991.

[Hay96] W. K. Hayman. The growth of solutions of algebraic differential equations. *Rend. Mat. Acc. Lincei* s.9, v. 7: 67-73, 1996.

[KCG94] P. Koiran, M. Cosnard, and M. Garzon. Computability with low-dimensional dynamical systems. *Theoretical Computer Science*, 132:113–128, 1994.

[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.

[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.

[Meer93] 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.

[Orp97] P. Orponen. A survey of continuous-time computation theory. *Advances in Algorithms, Languages, and Complexity* (D.-Z. Du and K.-I Ko, Eds.), 209-224. Kluwer Academic Publishers, Dordrecht, 1997.

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

[Ost25] A. Ostrowski. Zum Hölderschen satz über $\Gamma(x)$. *Math. Annalen*, 94:248–251, 1925.

[PE74] M. B. Pour-El. Abtract computability and its relation to the general purpose analog computer. *Transactions of A.M.S.*, 199:1–28, 1974.

[PER89] M. B. Pour-El and J. I. Richards. *Computability in Analysis and Physics.* Springer-Verlag, 1989.

[Rub88] L. A. Rubel. Some mathematical limitations of the general-purpose analog computer. *Advances in Applied Mathematics*, 9:22–34, 1988.

[Rub89a] 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.

[Rub93] L. A. Rubel. The extended analog computer. *Advances in Applied Mathematics*, 14:39–50, 1993.

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

[SS94] H. Siegelmann and E. D. Sontag. Analog computation via neural networks. *Theoretical Computer Science* 131:331–360, 1994.

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

[SF98] H. T. Siegelmann and S. Fishman. Analog Computation with Dynamical Systems. *Physica D* 120:214–235, 1998.

[Tho76] W. Thomson. On an instrument for calculating the integral of the product of two given functions. *Proc. Royal Society of London*, 24:266–268, 1876.

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

[WS92] Y. Wang and E. D. Sontag, Algebraic differential equations and rational control systems. *SIAM J. Control and Opt.* 30:1126-1149, 1992.