A new conceptual framework for analog computation

Jerzy Mycka

Institute of Mathematics
University of Maria Curie-Sklodowska
Lublin, Poland

José Félix Costa *

Department of Mathematics, I.S.T.
Universidade Técnica de Lisboa
Lisboa, Portugal

Abstract

In this paper we show how to explore the classical theory of computability using the tools of Analysis: a differential scheme is substituted for the classical recurrence scheme and a limit operator is substituted for the classical minimalization. We show that most relevant problems of computability over the non negative integers can be dealt with over the reals: elementary functions are computable, Turing machines can be simulated, the hierarchy of non computable functions be represented (being the classical halting problem solvable in some level). The most typical concepts in Analysis become natural in this framework. The most relevant question is posed: can we solve open problems of classical computability and computational complexity using, in the Popper saying, the toolbox of Analysis?

1 Introduction and motivation

The theory of analog computation, where the internal states of a computer are continuous rather than discrete, has enjoyed a recent resurgence of interest. This stems partly from a wider program of exploring alternative approaches

* Corresponding author
Email addresses: Jerzy.Mycka@umcs.lublin.pl (Jerzy Mycka), fgc@math.ist.utl.pt (José Félix Costa).

to computation, such as neural and quantum computation; partly as an idealization of numerical algorithms where real numbers can be thought of as entities in themselves, rather than as strings of digits [2,38]; and partly from a desire to use the tools of computation theory to better classify the variety of continuous dynamical systems that model our world (or at least its classical idealization) [5,20,37].

In most recent work on analog computation, e.g. [2,18,21,36], time is still discrete. Just as in standard computation theory, the machines are updated with each tick of a clock. If we are to make the state of a computer evolve in a continuum it makes sense to consider making its progress in time continuous too. While a few efforts have been made in the direction of studying computation by continuous-time dynamical systems [19,20,28,27,37,3,1], no particular set of definitions has become widely accepted. Thus analog computation has not yet experienced the unification that digital computation did through Turing's work in 1936.

In this work we go back to the roots of analog computation theory by starting with Claude Shannon's so-called General Purpose Analog Computer (GPAC). [1] 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 [16]. 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 [4]. From the early 1940's, the differential analysers at Manchester, Philadelphia, Boston, Oslo and Gothenburg, among others, were used to solve problems in engineering, atomic theory, astrophysics, and ballistics, until they were dismantled in the 1950s and 1960s following the advent of electronic analog computers and digital computers [4,15].

In the 1940s, two different views of the brain and the computer were equally important. One was the analog technology and theory that had emerged before the war. The other was the digital technology and theory that was to become the main paradigm of computation. [2] The outcome of the contest between these two competing views derived from technological and epistemological arguments. While digital technology was improving dramatically, the technology of analog machines had already reached a significant level of development. In

[1] In spite of being called "general", which distinguish it from special purpose analog computing devices, the GPAC is not a uniform model, in the sense of von Neumann.
[2] For example, students at MIT could at that time learn both about differential analysers and electronic circuits for binary arithmetic [25].

particular, digital technology offered a more effective way to control the precision of calculations. But the epistemological discussion was, at the time, equally relevant. For the supporters of the analog computer, the digital model — which can only process information transformed and coded in binary — wouldn't be suitable to represent certain kinds of continuous variation that help determine brain function. With analog machines, on the contrary, there would be few or no steps between natural objects and the work and structure of computation (cf. [25,14]). The 1942–52 Macy Conferences in cybernetics helped to validate digital theory and logic as legitimate ways to think about the brain and the machine [25]. In particular, those conferences helped made the McCulloch-Pitts digital model of the brain [17] a very influential paradigm. The McCulloch-Pitts model's descriptive strength led von Neumann, among others, to seek identities between the brain and specific kinds of electrical circuitry [14].

As we mentioned before, the first main paradigm of analog computation was Shannon's GPAC. Just as polynomial operations are basic to the Blum-Shub-Smale (BSS) model of analog computation [2], polynomial differential equations are basic to the GPAC. Shannon [35] showed that the GPAC generates the *differentially algebraic* functions, which are unique solutions of polynomial differential equations with arbitrary real coefficients. 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 [30] made this proof rigorous by introducing the crucial notion of the *domain of generation*. However, it is known that, even if the boundary condition is computable by the GPAC, the Dirichlet problem on the disk cannot, in general, be solved by the GPAC [34]. Moreover, the gamma function Γ is not computable by the GPAC, since it is not differentially algebraic [29,33].

Rubel [34] proposed the Extended Analog Computer (EAC). This model has the same computational power as the GPAC but also produces the solutions of a broad class of Dirichlet boundary-value problems for partial differential equations. However, 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. It is not even known if it can compute all analytic functions, in which case it would be too broad to be interesting as a model of computation.

The first presentation of a Theory of Recursive Functions over the Reals was attempted by Cris Moore [20]. Real recursive functions are generated by a fundamental operator, called differential recursion. The other fundamental operator is the taking of limits [23]. Between 1996, since Moore's seminal paper, and 2002 we have been working with the single concept of differential recursion. In [9] we show that a linearizarion of the differential recursion scheme gives rise

to an analog characterization of the class of (Kalmar's) elementary functions. This strong result gave the first hint that classical open problems may be lifted to the analog realm. In [8] and [12] we show that the GPAC is not closed under iteration and that a subclass of real recursive functions coincides with the class of GPAC-computable functions. In [23] we finally show how to capture higher computational classes through the limit operator. Manuel Campagnolo in [6] showed also that other computational complexity classes can be captured through appropriate structured differential schemata or adding simple (bounded) integration.

Analog characterization of classical computational complexity classes is not, as we now see it, a strong motivation to keep analog computation alive. Instead, we oversee two main directions for future research: (a) first, we may try to answer the question *what is the intrinsic computational complexity associated to continuous dynamic systems, specified by means of differential equations?* The second major line of research is *can we solve open problems of classical computability or computational (structural) complexity using the toolbox of Analysis?* This is indeed our hope. Indeed we think that our approach can only survive, and master relevance to the future if solutions to old problems can elegantly be found.

We try to show that our framework is versatile: from a carefull and not so complex definition of the (countable) set of recursive functions over the reals we show by means of the toolbox of Analysis that: (a) Laplace transform can be used to quickly obtain useful real recursive functions and to measure their rate of growth, (b) the embedding of Turing machines into continuous time recursive functions is trivial, (c) a (limit) hierarchy of real recursive functions exist to classify hardness of functions.

About the hierarchy of limits, we may add further topics. We show that we can embed the entire arithmetical hierarchy within the limit hierarchy up to some finite level (up to a finite number of limit operations), where the analytic hierarchy starts to be implemented. The use of limits gives rise to uncomputable functions, e. g., at some level we get the halting problem solved. To these previous aspects, we should add the impact of a further one (d): in the basis of the limit hierarchy we can still find a set of functions over the reals indeed computable by physical means, theoretically by Claude Shannon's GPAC and practically by the Differential Analyser of Vannevar Bush. Hence, in the basis we have truly computable functions in the physical sense. Is the GPAC the ultimate limit of computability? Nobody really knows, but we can add that Rubel's improved the GPAC in the 90's building up the conceptual EAC, in a such a way that some limits are still physical realizable in some sense. Can we envisage engines with a greater power? These are indeed conceived in a few theoretical experiences [40,13], although they require an unbounded amount of energy, and for that reason, not for theoretical reasons, they are

not implementable.

As another further point (e) we show that several constructs of Analysis are usable within real recursive functions to show (1) property "function f is polynomial" is (semi)decidable, (2) property "function f is of exponential order" is (semi)decidable, (3) property "function f is everywhere 0" is (semi)decidable, (4) property "function f is in C^0" is (semi)decidable.

A final remark (f) helps the reader to understand that computable numbers can be thought as entire computable structures, indivisible entities [20] or computable by digits (as in the classical way), using continued fractions. Strong uncompressible numbers like Chaitin's halting probability are computable in very precise levels of the limit hierarchy.

Now we finish by recalling to the reader Moore's seminal paper [20] published in 1996. Herein, we try to reformulate many of his constructs that failed to have a strong foundational basis. Reimplementation of the arithmetical hierarchy, and analytical hierarchy by means of continued fractions, are included in this paper in his honour, just to say that after all every construct in [20] can still stand up on top of our hierarchy of limits.

The basic rule of Popper is that *the scientist should specify in advance the conditions under which he will abandon his claims, his framework of research.* This is the fundamental rule of a game called Science. We think that our model is refutable, in the sense of Popper, if within a reasonable time we will not produce such a strong result showing that the toolbox of Analysis is indeed useful to solve problems in classical computability and complexity.

Let us add a few comments about the purpose and context of this article. Our paper is the full version of the short conference paper [24]. The current paper includes full proofs of statements not in [23], and only a few statements taken without proofs from it for understanding and completion.

We also note that since one year ago (since the date of our submission), other researchers have been working towards analog models based on [20]. We also made clear in [23] that our proposal was done to show that the idea of theory of computation over the reals could be settled without ill-defined operators or without unclear assumptions. For example, the major part of paper [20] is built using the operator called η. This operator can be seen as a minimalization over the product of two factors $f \times g$. When the search presumably ends, f is equal to 0 but g is infinite. Author of [20] postulated that it gives 0. We don't agree with this mathematical formulation, since it does not work in pure mathematical analysis.

Paper [22] shows that minimalization from [20] can be derived from the concept of limit. It does not show, e.g., that the η-operator can be derived from the

limit concept in [22]. In general, paper [22] does not help to solve unsound formulation of [20].

Paper [23] introduces an alternative formulation of Moore's ideas form [20], but it does not explore all aspects of this theory in the new framework. Current paper explores all aspects of the seminal paper [20] using the new, mathematically correct, formulation.

2 Recursive functions over the reals with bounded differential recursion and infinite limits

We give a new definition of real recursive functions, which is a derivative of the original definition found in [20]. However it is invented to avoid problems involved in the latter. It is important to see that the following definition is based on the vector operations (a variation of Moore's definition).

Definition 1 *The set of real recursive vectors is generated from the real recursive scalars* $0, 1, -1$ *and the real recursive projections* $I^i_n(x_1, \ldots, x_n) = x_i, 1 \leq i \leq n, n > 0$*, by the operators:*

(1) composition: if f *is a real recursive vector with* n k*-ary components and* g *is a real recursive vector with* k m*-ary components, then the vector with* n m*-ary components* $(1 \leq i \leq n)$

$$\lambda x_1 \ldots x_m . f_i(g_1(x_1, \ldots, x_m), \ldots, g_k(x_1, \ldots, x_m))$$

is real recursive.

(2) differential recursion: if f *is a real recursive vector with* n k*-ary components and* g *is a real recursive vector with* n $k + n + 1$*-ary components, then the vector* h *of* n $k + 1$*-ary components which is the solution of the Cauchy problem for* $1 \leq i \leq n$

$$h_i(x_1, \ldots, x_k, 0) = f_i(x_1, \ldots, x_k),$$

$$\partial_y h_i(x_1, \ldots, x_k, y) = g_i(x_1, \ldots, x_k, y, h_1(x_1, \ldots, x_k, y), \ldots, h_n(x_1, \ldots, x_k, y))$$

is real recursive whenever h *is of the class* C^1 *on the largest interval containing* 0 *in which a unique solution exists.*

(3) infinite limits: if f *is a real recursive vector with* n $k+1$*-ary components, then the vectors* h, h', h'' *with* n k*-ary components* $(1 \leq i \leq n)$

$$h_i(x_1, \ldots, x_k) = \lim_{y \to \infty} f_i(x_1, \ldots, x_k, y),$$

$$h'_i(x_1, \ldots, x_k) = \liminf_{y \to \infty} f_i(x_1, \ldots, x_k, y),$$

$$h_i''(x_1,\ldots,x_k) = \limsup_{y\to\infty} f_i(x_1,\ldots,x_k,y)$$

are real recursive, whenever these limits are defined for all $1 \le i \le n$.[3]

(4) Arbitrary real recursive vectors can be defined by assembling scalar real recursive components of the same arity.

(5) If f *is a real recursive vector, than each of its components is a real recursive scalar.*

The first important remark to the above definition is connected with a cardinality of the set of real recursive functions. Because every function has at least one finite syntactical description, hence the number of real recursive functions is countable. In this way we can observe that the system of functions given by our definition is constructive and not too large (not all real functions are captured by it, and in fact an uncountable number of real functions are left outside).

Let us discuss carefully the details of the definition. For differential recursion we restrict a domain to an interval of continuity. This will preserve the analiticity of functions in the process of defining. Moreover, this operator gives the same class C^k for a defined function as the given functions are from. This eliminates the possibility of defining such functions as $\lambda x.|x|$ without the limit operator.

Constant functions $0_n, 1_n, -1_n$ which are n-ary can be derived from unary constant functions by means of projections. For example $1_k(x_1,\ldots,x_k) = 1$ can be defined as $1_1(I_k^1(x_1,\ldots,x_k)) = 1$. Unary constant functions can be derived by differential recursions: $0(0) = 0, \partial_y 0(y) = I_2^2(y, 0(y))$; $u(0) = c, \partial_y u(y) = 0(I_2^1(y, u(y)))$, where $c = 1, -1$.

We excluded here the possibility of operations on undefined functions: our functions are strict in the meaning that for undefined arguments they are also undefined. But to obtain some interesting functions (like the mentioned η-function) we should improve the power of our system by an addition of the operators of infinite limits. Let us point out that introducing of infinite limits gets discontinuous functions.

---

[3] These concepts are defined in the completion of the real numbers $R \cup \{-\infty, +\infty\}$. Let the function f be defined on a metric space S and assume real values. If $x_0 \in S$ and $O(x_0,\epsilon)$ is a neighbourhood of x_0, then we define (see [11]) $\limsup_{x\to x_0} f(x) = \lim_{\epsilon\to 0}[\sup_{x\in O(x_0,\epsilon)} f(x)]$ and $\liminf_{x\to x_0} f(x) = \lim_{\epsilon\to 0}[\inf_{x\in O(x_0,\epsilon)} f(x)]$. In infinity we have then $\limsup_{y\to\infty} f(x) = \lim_{y\to\infty}[\sup_{x>y} f(x)]$, $\liminf_{y\to\infty} f(x) = \lim_{y\to\infty}[\inf_{x>y} f(x)]$. Because $\lambda y.[\sup_{x>y} f(x)]$ is a nonincreasing function and $\lambda y.[\inf_{x>y} f(x)]$ is a nondecreasing function thus $\lim_{y\to\infty}[\sup_{x>y} f(x)] = \inf_y[\sup_{x>y} f(x)]$, $\lim_{y\to\infty}[\inf_{x>y} f(x)] = \sup_y[\inf_{x>y} f(x)]$. If $\lim_{x\to\infty} f(x)$ exists, then $\liminf_{x\to\infty} f(x) = \lim_{x\to\infty} f(x) = \limsup_{x\to\infty} f(x)$.

We should also remember that in some cases we can use limits in some real point. This is possible by tranformation them into infinite limits. For example, $\lim_{y\to\frac{\pi}{2}} \sin xy$ can be written as $\lim_{y\to\infty} \sin x(\arctan y)$.

Following [39] (chapter 3), consider a real recursive function f such that: (i) f is continuous on $[0,\infty)$ except possibly for a finite number of jump discontinuities in every finite subinterval; (ii) there is a positive number M such that $|f(t)| \leq Me^{kt}$ for all $t \geq 0$. Then we say that f belongs to the class L_k. Additionally let $L = \bigcup_{k>0} L_k$.

Proposition 2 *If $f \in L_k$, then the Laplace transform $L(f)(x)$ exists for $x > k$ and it is real recursive.*

Proof. From the condition (ii) we have $|f(t)|e^{-kt} \leq M$. Now we can proceed in the following way:

$$\int_0^y |f(t)|e^{-xt}dt = \int_0^y |f(t)|e^{-kt}e^{-(x-k)t}dt \leq \int_0^y Me^{-(x-k)t}dt.$$

Now to check an existance of the Laplace transform it is sufficient to take $\lim_{y\to\infty} \int_0^y Me^{-(x-k)t}dt$ which is defined only if $x > k$, and in this case is given by $\frac{M}{x-k}$. □

If the Laplace transform of f exists, then f is said to be of exponential order: it exists for x greater than some real number k. If the Laplace transform of f exists, and $L(f)(s)$ is defined for $s > 0$, then f is of subexponential order, hence, dominated by some polynomial.

We can also present the proposition, which connects inverse Laplace transform with real recursive functions.

Proposition 3 *Let F, G be Laplace transforms of some real recursive functions. Then inverse Laplace transforms $L^{-1}(FG), L^{-1}(F+G)$ are also real recursive functions.*

The above proposition is a consequence of properties of inverse Laplace transform, namely convolution and linearity.

To illustrate further this transformation let us point out that if f is a $n+1$-ary real recursive function, then its derivative $\partial y f(x_1, \ldots, x_n, y)$ is a real recursive function. This result can be obtained directly by limits or for some functions by properties of Laplace transform. For example, let $f(x) = x^n, n \geq 0$, then $L(f)(s) = \frac{n!}{s^{n+1}}$. By the known property of Laplace transform we have: $L(\partial_x f(x))(s) = sL(f(x)) - f(0) = \frac{n!}{s^n}$, using repeatedly the convolution operation for $\frac{1}{s}$ we get $L^{-1}(\frac{n!}{s^n})(x) = nx^{n-1}$. Derivatives are physical realizable: the class of differential algebraic functions is closed under derivatives, mak-

ing a large class of derivatives physical realizable. Since the extended analog computer also is close to physical implementation, the larger class of EAC-computable functions [34] are also closed under derivatives.

Let us give without a proof some examples of functions generated with the definition of real recursive functions.

Proposition 4 *The functions* $+, \times, -, \exp, \sin, \cos, \lambda x.\frac{1}{x}, /, \ln, \lambda xy.x^y$ *are real recursive functions.*

Only as an example let us present the construction of addition $+(x,0) = I_1^1(x) = x$, $\partial_y +(x,y) = 1_3(x,y,+(x,y))$. Let us mention that $\lambda x.\frac{1}{x}$ can be defined as a real recursive function for positive values by Laplace transform.

We can construct also other special real recursive functions.

Proposition 5 *The Kronecker* δ *function, the signum function, and absolute value are real recursive functions. The Heaviside* Θ *function (equal to 1 if* $x \geq 0$*, otherwise 0), the binary maximum* max*, the square-wave function, and the floor function* $\lambda x.\lfloor x \rfloor$ *are all real recursive too.*

Here we should use infinite limits, for example $\delta(x) = \liminf_{y\to\infty}(\frac{1}{1+x^2})^y$, $sgn(x) = \liminf_{y\to\infty} \frac{2}{1+\exp(-xy)} - 1$. $\Theta(x-c)$ can be obtained as the inverse Laplace transform for $\frac{e^{-cs}}{s}$.

Because the set of natural numbers can be defined by real recursive functions, hence we can extend the definition of real recursive numbers for functions with a domain in $N^k \times R^n, k, n \geq 0$, by the following method: *a function* $f : N^k \times R^n \to R^m, n \geq 0$ *is called real recursive if* $f(n_1, \ldots, n_k, y_1, \ldots, y_n) = F(n_1, \ldots, n_k, y_1, \ldots, y_n)$*, for some real recursive function* $F : R^{k+n} \to R^m$.

We gave the general definition of real recursive functions. For proper analysis of functions it is important to control the domain and singularities of functions. We can postulate new operators which may check the points: are they in the domain of some functions or not. For any function $f : R^{n+1} \to R$ let

$$\eta_y f(\bar{x}, y) = \begin{cases} 1 \text{ if } \lim_{y\to\infty} f(\bar{x}, y) \text{ is defined,} \\ 0 \text{ otherwise.} \end{cases}$$

In the analogous way we can define $\eta_y^i f(\bar{x}, y), \eta_y^s f(\bar{x}, y)$ equal to 1 if $\liminf_{y\to\infty} f(\bar{x}, y)$ is defined (resp. $\limsup$), otherwise equal 0.

Defined in this way $\eta_y f(\bar{x}, y)$ is a characteristic function for the set of such $\bar{x}$ that $\lim_{y\to\infty} f(\bar{x}, y)$ is well defined (without singularities). Analogously the functions $\eta_y^i f(\bar{x}, y)$, $\eta_y^s f(\bar{x}, y)$ play the same role, respectively, for $\liminf_{y\to\infty} f(\bar{x}, y)$ and $\limsup_{y\to\infty} f(\bar{x}, y)$. The problem arises whether such operators

are real recursive operators. If the answer to the question, whether we can define them by standard operators, is yes, we may patch any partial function to total one. For example, let the function f be total and $F_{\text{total}}(\bar{x}) = \lim_{y\to\infty}(\eta_y f(\bar{x},y))f(\bar{x},y)$, $F(\bar{x}) = \lim_{y\to\infty} f(\bar{x},y)$. Then $F_{\text{total}}(\bar{x})$ is total and if $F(\bar{x})$ is defined, then $F_{\text{total}}(\bar{x}) = F(\bar{x})$ otherwise $F_{\text{total}}(\bar{x}) = 0$.

The key problem in our investigation of the operators η, η^i, η^s is a question: is the class of real recursive functions closed under them. This would be true if functions obtained by these operators from real recursive functions can be constructed as real recursive functions.

The below proposition is cited after [23].

Proposition 6 *The functions $\eta_y g, \eta^i_y g, \eta^s_y g$ are total real recursive functions if g is total real recursive function.*

An iteration of a given function plays the important role in computability theory. For given function $h(x)$ we can built the consecutive values:

$$x, h(x), h(h(x)), \ldots, \underbrace{h(\ldots(h(x))\ldots)}_{k}, \ldots.$$

They are usually denoted by $h^k(x)$. We can present the result about a possibility of such construction in the set of real recursive functions.

Proposition 7 *Let $h : R \to R$ be a real recursive function. Then the function $H : N \times R \to R$, $H(n,x) = h^n(x)$ is real recursive too.*

Proof. The iteration h^n can be given in the following method, $f, g : R \to R$:

$$f(0) = g(0) = x,$$

$$\partial_t g(t) = (h(f(t)) - f(t)) \sin^2 \frac{\pi t}{2} \Theta(\sin \pi t),$$

$$\partial_t f(t) = \frac{1}{2r(t)}(g(t) - f(t))\Theta(-\sin \pi t),$$

where $r(0) = 0, \partial_x r(x) = 2\sin^2 \pi x \Theta(\sin \pi x) - \frac{1}{2}$. The n-th iteration of the function h satisfies the following equation: $h^n(x) = f(2n) = g(2n)$ for natural n. □

The above result can be easily extended for vectors.

As a corollary we can obtain the fact that for a real recursive function $f : R^{n+1} \to R$ its finite product $F_1(n,\bar{x}) = \prod_{i=0}^{n} f(i,\bar{x})$ and finite sum $F_2(n,\bar{x}) = \sum_{i=0}^{n} f(i,\bar{x})$ are real recursive. Let us present only for a demonstration the construction for a product by iterations: if the function $t_f : R^{n+2} \to R^{n+2}$ is

defined in the following way $t_f(\bar{x}, y, i) = (\bar{x}, yf(\bar{x}, i), i+1)$, then $\prod_{i=0}^{n} f(\bar{x}, i) = I_3^2(t_f^n(\bar{x}, 1, 0))$.

Now we use another notion of the classical theory of computability in the analog context. A set $S \subset R$ is called a real recursive set if it has a real recursive characteristic function.

Proposition 8 *The sets of integer numbers Z, natural numbers N, rational numbers Q, the set of all algebraic reals are real recursive sets.*

Proof. For Z it is sufficient to use $\delta(sin\pi x)$ as a characteristic function χ_Z. Then $\chi_N(x) = \chi_Z(x)\Theta(x)$.

For Q a construction is more troublesome. Let us start with an auxiliary function $f(n, x) = \prod_{i=1}^{n}(1 - \chi_Z(xi))$. Such a function is equal to 0 iff x is of the form $\frac{p}{q}, p, q \in Z$, where $1 \leq q \leq n$, otherwise 1. Going to infinity we can define

$$\chi_Q(x) = 1 - \lim_{z \to \infty} f(\lfloor z \rfloor, x).$$

Let us use a letter A as a symbol of the set of algebraic number. If $x \in A$, then there exists such a polynomial P of some degree n with natural coefficients $a_0, \ldots, a_n$ and vector of natural numbers $i_0, \ldots, i_n$ (describing signs of the coefficients of P) such that $\sum_{j=0}^{n} a_j(-1)^{i_j}x^j = 0$. We would like encode these two vectors into two natural numbers a, i. The known result (see [26]) says us that there exists such a (natural) primitive recursive function β that $\beta(a, 0) = n, \beta(a, j) = a_{j-1}, 1 \leq j \leq n+1$, $\beta(i, 0) = n, \beta(i, j) = i_{j-1}, 1 \leq j \leq n+1$. Then the value of P for x is given as $P(x, a, i) = \sum_{j=0}^{\beta(a,0)} \beta(a, j-1)(-1)^{\beta(i,j-1)}x^j$. Because natural primitive recursive functions are real recursive (see [8]), hence P is a real recursive function. Let us extended this function into

$$P'(x, y, z) = \begin{cases} 1 & y \text{ or } z \text{ are not natural numbers or } \beta(y, 0) \neq \beta(z, 0), \\ P(x, y, z) & \text{otherwise.} \end{cases}$$

Then x is an algebraic number only in this case iff there exist such y, z that $P'(x, y, z) = 0$. This condition can be checked by the following construction: let $p(x, y, w) = |P'(x, y, w)| + 1$ for $P'(x, y, z) \neq 0$, otherwise 0; then

$$p'(x, z) = \prod_{k=0}^{\lfloor z \rfloor} \prod_{j=0}^{\lfloor z \rfloor} p(x, k, j)$$

will be equal to zero only in this case if there exist such a polynomial P encoded by $k, j \leq \lfloor z \rfloor$, that its value in x is equal to 0. Now to find a characteristic function of A suffices to write:

$$\chi_A(x) = \begin{cases} 1 \ \eta_z p'(x, z) = 1 \text{ and } \lim_{z \to \infty} p'(x, z) = 0, \\ 0 \text{ otherwise.} \end{cases} \quad \square$$

3 η-Hierarchy

Here we aproach a new problem. Are there different levels of difficulty in a computation if it goes beyond the Turing computability? The natural measure of a function's difficulty can be join with the degree of (dis)continuity. The above considerations lead us to the conception of η-hierarchy which describe the level of nesting limits in the definition of a given function.

We should start with the notion of syntactic n-ary descriptions of real recursive vectors. Let us introduce some kind of symbols called basics descriptors for all basic real recursive functions. The combination of such descriptions for given real recursive functions will form a new description of another function. Let us start with basic functions: i_k^j is a k-ary description for projection I_k^j for all $1 \leq j \leq k$; $1_k, \bar{1}_k, 0_k$ are k-ary descriptions for constants $1, -1, 0$ used with k variables. We must add also operator symbols (descriptors) for all introduced operators: dr - for a differential recursion, c - for a composition, l, ls, li for a respective kind of limits ($\lim, \limsup, \liminf$).

Now the collection of descriptors of real recursive vectors can be inductively defined as follows: $i_n^j, 1_n, \bar{1}_n, 0_n$ are n-ary descriptions of I_n^j, $1 \leq j \leq n \in N$, $f(x_1, \ldots, x_n) = 1$, $f(x_1, \ldots, x_n) = -1$,$f(x_1, \ldots, x_n) = 0$ for all $(x_1, \ldots, x_n) \in R^n$, $n \in N$, respectively. If $\langle h \rangle = \langle h_1, \ldots, h_m \rangle$ is a k-ary description of the real recursive vector h and $\langle g \rangle = \langle g_1, \ldots, g_k \rangle$ is a n-ary description of the real recursive vector g, then $c(\langle h \rangle, \langle g \rangle)$ is a n-ary description of the composition of h and g. For differential recursion we can write: if $\langle h \rangle = \langle h_1, \ldots, h_n \rangle$ is a k-ary description of the real recursive vector h and $\langle g \rangle = \langle g_1, \ldots, g_n \rangle$ is a $k+n+1$-ary description of the real recursive vector g, then $dr(\langle h \rangle, \langle g \rangle)$ is a $k+1$-ary description of the solution of the Cauchy problem for h, g (if such a solution exists). Finally, if $\langle h \rangle = \langle h_1, \ldots, h_m \rangle$ is a $n+1$-ary description of the real recursive vector h, then $l(\langle h \rangle)$, $li(\langle h \rangle)$, $ls(\langle h \rangle)$ is a n-ary description of an apropriate infinite limit (respectively $\lim, \liminf, \limsup$) of h (if such limits exist).

Now we can find the η-number for a description of some function f.

Definition 9 *For a given n-ary description s of a vector f let $E_i^k(s)$ (the η-number with respect to i-th variable of the k-component) be defined as follows:*

(1) $E_i^1(0_n) = E_i^1(1_n) = E_i^1(\bar{1}_n) = 0$;
(2) $E_i^m(c(\langle h \rangle, \langle g \rangle)) = \max_{1 \leq j \leq k}(E_j^m(\langle h \rangle) + E_i^j(\langle g_j \rangle))$, *where h is a n components k-ary vector and g is a k-components m-ary vector;*
(3) for a differential recursion we distinguish two cases:
- $i \leq k$:
  $E_i^j(dr(\langle f \rangle, \langle g \rangle)) =$
  $\max(E_i^1(\langle f_1 \rangle), \ldots, E_i^1(\langle f_n \rangle), E_i^1(\langle g_1 \rangle), \ldots, E_i^1(\langle g_n \rangle),$

$E^1_{k+1}(\langle g_1\rangle), \ldots, E^1_{k+1}(\langle g_n\rangle))$

- $i = k+1$*:*
  $E^j_i(dr(\langle f\rangle, \langle g\rangle)) =$
  $\max_{0\leq m\leq n}(\max(E^1_{k+m+1}(\langle g_1\rangle), \ldots, E^1_{k+m+1}(\langle g_n\rangle)))$
  where f *is a* n *components* k*-ary vector and* g *is a* n *components* $k+n+1$*-ary vector;*

(4) $E^k_i(l(\langle h\rangle)) = E^k_i(li(\langle h\rangle)) = E^k_i(ls(\langle h\rangle)) = \max(E^k_i(\langle h\rangle), E^k_{n+1}(\langle h\rangle)) + 1$, *where* h *is a* k *components* $n+1$*-ary vector.*

For the n-ary description s of m components we can define now $E(\langle h\rangle) = \max_k \max_i E^k_i(\langle h\rangle)$ for $1 \leq i \leq n, 1 \leq k \leq m$. Now we can deal with the η-number for a real recursive functions where $\eta(f)$ can be defined as the minimum of $E(\langle f\rangle)$ for all possible descriptions of the function f. We are ready to conclude with definition of η-hierarchy as a family of $H_j = \{f : \eta(f) \leq j\}$. It will be comfortable to think about the η-hierarchy as the measure of the difficulty of real recursive functions. If $f \in H_j$, then j is the number of nested (non-parallel) η needed to patch the function f to the total function.

Let us start with recalling of some real recursive functions from previous propositions.

Example 10 *From the constructions given in Propositions 4, 5 we have* $+$, $\times$, $-$, $\exp$, $\sin, \cos$, $\lambda x.\frac{1}{x}$, $/$, $\ln$, $\lambda xy.x^y$ *are in* H_0*, the Kronecker* δ *function, the signum function and absolute value are in* H_1*. The Heaviside function* Θ*, the binary maximum* $\max$*, the square-wave function and the floor function are in* H_1*.*

To see that our framework is strongly supported by one such classical theory of computation (Shannon's Theory of Analog Computation), we add physical realizability to the basis of recursive functions over the reals.

Proposition 11 *A subclass of* H_0 *coincides with the functions GPAC-computable.*

Detailed proof of this statement can be found in [12].

Let us give here the examples of some functions which have the important significance in mathematics and can be expressed in terms of real recursiveness. Let us point out that Rubel showed in [33] incompleteness of the GPAC-computable functions proving the Euler's Γ-function and the Riemann ζ-function are not GPAC-computable.

Example 12 *The Euler's* Γ*-function and the Riemann* ζ*-function are real recursive functions from the class* H_1*.*

Let us recall that Laplace transform of $t^x, x > -1$, is equal to $\frac{\Gamma(x+1)}{s^{t+1}}$, hence

$\Gamma(y)$ for $y > 0$ is real recursive and (because Laplace transform uses only one limit) in H_1. Let us add that Marion Pour-El (see [30]) proved that Γ is not GPAC-computable so its class is most probably strictly H_1. The following equation stands $\zeta(x) = \frac{1}{\Gamma(x)} \int_0^\infty \frac{t^{x-1}}{1-\exp(-t)} dt$, for $x > 0$, where the right side can be defined simply by real recursive operators using the previous results. It is clear from the form of the expression $\frac{1}{\Gamma(x)} \int_0^\infty \frac{t^{x-1}}{1-\exp(-t)} dt$ that ζ is also in H_1.

We can also add the corollaries of the constructions used in Propositions 7, 8.

Proposition 13 *For given function f from the class $H_i, i \geq 0$, its iteration $F(n, x) = f^n(x)$ is in the class $H_{\max(1,i)}$.*

By the class of some set we understand the class of its characteristic function.

Proposition 14 *The sets of natural and integer numbers are in H_1, the set of rational numbers Q is in H_3, and the set of algebraic numbers is in H_5.*

Proof. All sets except of the set of algebraic numbers are obvious. In this last case we can recall (compare the construction of η in [23]) that for $f \in H_i, i \geq 0$ we have $\eta f \in H_{i+3}$, hence the set of algebraic numbers is in H_5. □

4 The halting problem

Now we can turn to some application of the η operator. We consider a possibility of a process of Turing machines simulation by real recursive functions. Such problems were considered by Moore [20], however his assumptions were connected with a wrong established η-function.

A Turing machine can be given by the following description. It consists of an infinite tape for storing the input, output, and scratch working, and a finite set of internal states. All elements on a tape are strings. Without loss of generality, we can choose some alphabet for these strings, the binary alphabet is a practical choice. The machine works in steps. In one step it scans the symbol from the current position of the tape (under the head of the machine), changes this symbol according to current state of the machine and moves the position of the tape to left or right with a transformation of state. Some states are distinguished as final, when the machine reaches one of them then it stops. Our Turing machine model obey to the following rules (classical constraints): (a) input is finite and (b) output is finite, no matter the length of computation, being it finite or infinite.

Proposition 15 *There are real recursive functions from the class H_1, which can simulate any Turing machine.*

Proof of this proposition can be found in [23]

It can be mentioned that the process of simulation is especially important for universal Turing machines. The results in this area proved in last years [31] give us the interesting restrictions of the size of such machines (for example, there exists a universal Turing machine for 5 states and 5 symbols) what leads us to significant simplicity of the constructed function. It is worth to point out that the analiticity of such function has as a consequence a lower level of the complexity of the simulation.

Let us signal a few important questions concerned to Turing machines. The first problem is known as the halting problem: does the machine M for some input reach the final state? There is not a natural recursive characteristic function of this problem. The method of simulation of Turing machines given above can resolve it in the simple way with real recursive functions.

Proposition 16 *For any Turing machine M, there exists a real recursive function the class H_3 which is the characteristic function of the halting problem for M.*

The proof given in [23] uses a construction of a sequence of configurations. To check whether this sequence is ended by a configuration with some final step or it is infinite the η-function is taken.

Let us turn more deeply into the problems of computation beyond the power of Turing machines (hypercomputation). The problem of infinity which can appear in the sequel of not finishing computation introduced troubles into the computability theory and practice. The first step to improve this situation is directed to change the behaviour of a Turing machine. For this purpose we may use an accelerated Turing machine [10]. Its description is the same as for a standard Turing machine, but a temporal pattern of steps is given. Each subsequent step is performed in half the time of the step before. Such machines could complete an infinity of steps in two time units only. This feature of accelerated Turing machines gives us the power to puzzle out the halting problem by programming the following algorithm: mark the first square on the tape by 0, change it only in the final (last) step to 1, if after 2 time units we have 0 in the distinguished square, then the machine does not halt, otherwise it halts. However, some difficulties arise also in this model. Let us imagine the machine changing value of one square from 1 to 0 and conversely in all steps using only one non final internal state. We can hesitate what is on the tape after all steps (in infinity), because in this case the computation diverges. Let us assume that in (x, y) the number x encodes the right half of tape (including the symbol under the head of the Turing machine) together with the current state and y the left half of tape. The accelerated Turing machine can be simulated in the same way as the standard Turing machine with only

one modification: in the end it is not necessary to have the result (z_x, z_y) with a final state i written in z_x. Hence, the convergent infinite computations and finite computations both give the correct result, however the divergent computations have undefined result.

The above remarks prove that η operator gives us the additional power to standard models of computation by controling the domain of computable functions and machines. Such possibility is an effect of checking in a finite amount of time an infinite number of a computation elements. The standard objection to such extensions of computable systems is their unphysical character. Theory of n-body dynamics and general relativity may provide counterarguments to such a statement. In fact, we know that some results for Newtonian physics [40] or general relativity [13] may be used to harness devices more powerful than a standard Turing machine.

5 Arithmetical hierarchy and computable numbers

We will proceed now with the relations of natural numbers taken from the arithmetical hierarchy. The class $\Sigma_0^0 = \Pi_0^0$ contains only such relations, which have recursive characteristic functions. The upper stages of this hierarchy can be constructed from the lower ones in the following way:

$$\Sigma_{n+1}^0 = \{P : (\exists P' \in \Pi_n^0) P(\bar{m}) \equiv \exists s P'(\bar{m}, s)\},$$

$$\Pi_{n+1}^0 = \{P : (\exists P' \in \Sigma_n^0) P(\bar{m}) \equiv \forall s P'(\bar{m}, s)\},$$

where $P \subseteq N^k, P' \subseteq N^{k+1}, k \geq 1$. To complete our hierarchies we can add the following equation $\Delta_n^0 = \Sigma_n^0 \cap \Pi_n^0, n \geq 0$.

Now let us correlate this infinite hierarchy of sets and relations to the η-hierarchy. We must return to the Turing machine and its simulation by real recursive functions.

From Proposition 15 and from the fact that all natural recursive sets and relations have Turing computable total characteristics we get the following conclusion:

Corollary 17 *Every natural recursive set or relation is in H_2, i.e. $\Sigma_0^0 = \Pi_0^0 \subset H_2$.*

The next element of our investigation has to deal with higher levels of arithmetical hierarchy. For this purpose we need to analyse the method of use of quantifiers.

For every function $f : R^{n+1} \to R$ we can construct such real recursive function

$\rho_f : R^n \to R$ that

$$\rho_f(\bar{x}) = \begin{cases} 1 \ \exists y \in Nf(\bar{x}, y) = 0, \\ 0 \ \forall y \in Nf(\bar{x}, y) \neq 0. \end{cases}$$

To this effect we start with a description of the function $f_c(\bar{x}, y) = 1 - \delta(f(\bar{x}, y))$. This function has the following property $f_c(\bar{x}, y) = 1 \equiv f(\bar{x}, y) \neq 0$, $f_c(\bar{x}, y) = 0 \equiv f(\bar{x}, y) = 0$. It is easy to observe that now

$$\lim_{z \to \infty} \prod_{j=0}^{z} f_c(\bar{x}, j) = \begin{cases} 0 \ \exists y \in Nf(\bar{x}, y) = 0, \\ 1 \ \forall y \in Nf(\bar{x}, y) \neq 0. \end{cases}$$

Hence $\rho_f(\bar{x}) = 1 - \lim_{z \to \infty} \prod_{j=0}^{\lfloor z \rfloor} f_c(\bar{x}, j)$. Finally, by properties of the iteration, we can claim that $\rho_f \in H_{i+2}$.

Theorem 18 *The sets and relations from Σ_i^0, Π_i^0 belong to H_{i+2} for $i \geq 0$.*

Proof. It is clear from the above considerations that if a relation $R \in N^{k+1}$ is in $H_i, i \geq 0$ (which, of course, means its characteristic function $\chi_R \in H_i$), then the relation $P(\bar{x}) = \exists y R(\bar{x}, y)$ has the characteristic function χ_P equal to $\rho_{1-\chi_R}$. Because the natural recursive relations are in H_2, so the function χ_R is at least in H_2. Morever, a normalization of the function f to two values $0, 1$ realized by f_c is not needed in the case of $f = \chi_R$. Hence the relation P is in E_{i+1}. For relation $Q(\bar{x}) = \forall y R(\bar{x}, y)$ it is sufficient to observe that the characteristic function $\chi_Q(\bar{x}) = 1 - \chi_{\exists \neg R}(\bar{x})$ belongs to H_{i+1} too. Using the above results as an inductive step with an additional assumption that natural recursive relations are in H_2 we obtain the thesis of this theorem. □

Of course, as a consequence of the above proposition we have the following statement.

Corollary 19 *Natural recursively enumerable sets are in the class H_3.*

Although we don't know if our limit hierarchy collapses, we indeed do know that the arithmetical hierarchy does collapse within the limit hierarchy since the entire hierarchy of arithmetics is contained in Δ_1^1, and this class is included in some finite level of the limit hierarchy. To better understand this fact we have to recall the paper of one of us [22] that shows how to implement a search (for zeroes) operator with the limit construct and our paper [23] where the analytical hierarchy $\Sigma_\omega^1, \Pi_\omega^1, \Delta_\omega^1$, is roughly implemented.

Let us add that by computable reals (points) we understand values of real recursive functions with an arity 0.

We can prove that all real numbers given by a continued fraction built from real recursive sequences of naturals are real recursive, and conversely that for

a real number its continued fraction expansion can be described by a real recursive function. Continued fractions, together with the search for zeroes' operator, can be used to implement the Analytic Hierarchy in the same way as Cris Moore did it in [20].

Proposition 20 *Let x be a real number given as a continued fraction $x = [x_0, x_1, x_2, \ldots]$:*

$$x = x_0 + \cfrac{1}{x_1 + \cfrac{1}{\ddots}}.$$

Then $\phi(x, n) = x_n$ is a real recursive function. Conversely if $f : R \to R$ is a real recursive function, which maps natural numbers to natural numbers, then $x[f] = [f(0), f(1), f(2), \ldots]$ is a real recursive number.

Proof. For the first part of this proposition it is sufficient to define [4] $\phi(x, n) = \lfloor g^n(x) \rfloor$, where $g(x) = \begin{cases} 0 & x \in Z, \\ \frac{1}{x - \lfloor x \rfloor} & x \notin Z. \end{cases}$

Conversely, for a given f we use the real recursive map

$$t(x, k) = \begin{cases} (\frac{1}{x} + f(k-1), k-1), & k > 0, \\ (x, 0), & k = 0. \end{cases}$$

Now if $T(k) = I_2^1(t^k(f(k), k))$, then

$$T(k) = f(0) + \cfrac{1}{f(1) + \cfrac{1}{\ddots + \frac{1}{f(k)}}}$$

and we can find x as $\lim_{y \to \infty} T(\lfloor y \rfloor)$. □

In this sense e, π are computable reals: $\pi = 3 + [7, 15, 1, 292, 1, \ldots], e = 2 + [1, 2, 1, 1, 4, 1, 1, \ldots, 2n, 1, 1, \ldots]$. Let the continued fraction for x be written $[x_0, x_1, \ldots]$. Then the limiting value of the geometric mean is almost always Khinchin's constant (failing only for a countable number of reals) $K = \lim_{n \to \infty} \sqrt[n]{x_0 \ldots x_n}$, which is also real recursive number as: $K = \lim_{n \to \infty} \prod_{k=1}^{n} (1 + \frac{1}{k(k+2)})^{\frac{\ln k}{\ln 2}}$.

We can also prove that many real numbers which are not computable in Turing sense are real recursive.

Proposition 21 *There exist Turing uncomputable real numbers which are real recursive.*

[4] In the case $n = 0$ we have $g^0(x) = x$.

Proof. Let us choose some function $f : N \to \{0,1\}$ with a graph $G_f(x,y) \equiv y = f(z)$ which belongs to the class $\Delta_j, j > 1$. Then we can construct the number x equal to $\lim_{n\to\infty} \sum_{i=0}^{n} (\frac{3f(i)}{4^i} + \frac{1-f(i)}{4^i})$, which is uncomputable in Turing sense. However in the obvious manner it is real recursive. □

Finally let us mention a real recursive character of a particular Turing uncomputable number, namely Chaitin's constant.

Proposition 22 *Chaitin's Ω constant is a real recursive number.*

Proof. As usual let $\Omega = \sum_p 2^{-|p|}$, where p is a binary representation of halting programs (without inputs) of Turing machine with a property, that no proper prefix of a syntactically correct program is a syntactically correct program.

Let U be some Turing universal machine working on Turing programs without inputs given on a tape by a specific binary coding. This coding has such a property that if a binary sequence w encodes a syntactically correct program, then no proper prefix of w encodes a syntactically correct program. It can be simply obtain, for example by a convention that the beginning of w contains a length of w. Let us assume that $b_n(i)$ is a binary representation of natural number i given by n digits (possibly with zeroes at the beginning).

We can define

$$F(n) = \sum_{i=1}^{n} [\sum_{j=0}^{2^{i+1}-1} H'_U(0, b_{i+1}(j))] 2^{-(i+1)},$$

where H'_U is a real recursive function with such a property that $H'(0,x) = 0$ iff x does not encode a syntactically correct program, or if x encodes a correct program which is not halting; otherwise H' has the value 1. This function can be easily obtained from a characteristic function H_U of the halting problem for U, because checking a syntactical correctness of a program can be done by a natural total recursive function (hence by a real recursive function too). An existance of such a function is guaranteed by Proposition 16. A tape is fullfilled only by a binary sequence $b_{i+1}(j)$ with a length $i+1$. The final step is given as $\Omega = \lim_{n\to\infty} F(n)$. □

6 Conclusions

We introduced a framework of real recursive functions (i.e. an inductive set) in such a way that (a countable set of) functions over the reals exist that simulate arbitrary Turing machines, decide the halting problem, and decide all levels of the arithmetic hierarchy. Such a class of functions includes in a very natural way the elementary functions of Analysis, and real numbers computable in the Turing sense. The main ingredients with regard to Kleene's theory, are

the following closure operators: a scheme of differential recursion substitutes for recursion and the taking of infinite limits substitutes for minimalization.

The main result, that Turing machines can be simulated by differential equations of inductive nature is based on an iteration scheme, and the existence of a few (one is indeed enough) non-analytical functions. The proof is built only to show completeness, and it does not mean (it is not proved that it can not) that the more natural constructions do exist.

Although results are easy to obtain some doubts may arise in an in-depth reading.

A function over non negative integers once embedded into the reals may look artificial in the sense of having plateaux between integer numbers, although remaining (e. g.) C^∞: this happens by means of the iteration scheme. That is a consequence of our way of proving completeness. But, surely, we can provide examples showing that other much more natural functions (coinciding on the integers) exist. Take the Fibonacci sequence. Since an explicit expression does not exist in classical computability over the non negative integers, we provide a recursion scheme: from $f(0) = f(1) = 1, f(n) = f(n-1)+f(n-2)$, we get a new function $g(n) = 2^{f(n)}3^{f(n+1)}$ such that $g(0) = 6, g(n+1) = 2^{(g(n))_1}3^{(g(n))_2}$, where $(\ldots)_1$ and $(\ldots)_2$ are the first and second exponents of prime factorization. The function g is primitive recursive, and so is f, given by $f(n) = (g(n))_1$. The reader is now invited to realize that our view of computability over the reals has much more contents than the embedding of Turing machine simulation in differential schemata. Indeed, we know that a Turing machine exists that computes the n-th Fibonacci number. However, we have it immediately from $f(x) = \frac{1}{\sqrt{5}}(a^{x+1} - b^{x+1})$ where a, b are the golden numbers $\frac{\sqrt{5}+1}{2}$ and $\frac{\sqrt{5}-1}{2}$. This is not just an extension to the reals of some function from N to N, it is a real valued function in its own rights with no "plastic cirurgy" of an arbitrary extension. The function f is now being defined as $f(x) = f_1(x) - f_2(x)$, where $\partial_x f_1(x) = (\ln a) \times f_1(x)$, $\partial_x f_2(x) = (\ln b) \times f_2(x)$, with $f_1(0) = \frac{a}{\sqrt{5}}$ and $f_2(0) = \frac{b}{\sqrt{5}}$.

Assume for simplicity that with limits we can decide whenever a real-valued function is in C^0 for non negative values [5] . We know that the classical problem of knowing if a given unary computable function is everywhere 0 is undecidable: this undecidability result is based on standard methods like reducibility via s-m-n theorem. With the toolbox of Analysis we have different but nevertheless standard methods too: We can take the absolute value of a given such function f, namely $|f(x)|$, and integrate from 0 to infinity; we then have $I(f) = \int_0^\infty |f(x)|dx = 0$ if and only if f is 0 in $[0, \infty)$; $\delta(I(f))$ provides a characteristic to such a problem.

[5] We are ready to give details of such method in the forthcoming paper.

We also showed that analytical tools exist to control the growth of real recursive functions through Laplace transform, e. g., if the Laplace transform is defined all over the positive reals, than the function does not grow faster than a polynomial.

We believe that our most general framework, envolving infinite limits, have enough ingredients to allow the translation of classical computability and classical computational complexity problems into Analysis. We do believe that such translations might be a solution to open problems described in analytic terms: we are now much envolved in the definition of analog classes P and NP.

Acknowledgements We would like to thank Cris Moore for many discussions on the more obscure results presented in his seminal paper [20]. Our paper was design to present a most general framework for the Recursion Theory on the Reals, showing that Moore's original ideas can still be fully implemented in our conceptual scheme.

References

[1] Asa Ben-Hur, Hava T. Siegelmann, and S. Fishman. A theory of complexity for continuous time systems. *Journal of Complexity*, 18(1): 87-103, 2002.

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

[3] O. Bournez. Achilles and the tortoise climbing up the hyper-arithmetical hierarchy. *Theoretical Computer Science*, 210(1):21-71, 1999.

[4] M. D. Bowles. U. S. technological enthusiasm and british technological skepticism in the age of the analog brain. *IEEE Annals of the History of Computing*, 18(4):5-15, 1996.

[5] M. S. Branicky. Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoretical Computer Science*, 138(1):67-100, 1995.

[6] M. L. Campagnolo. Continuous-time computation with restricted integration capabilities. *Theoretical Computer Science*, 317:147–165, 2004.

[7] M. L. Campagnolo. The complexity of real recursive functions. In C. S. Calude, M. J. Dinneen, and F. Peper, (eds), *Unconventional Models of Computation, UMC 2002*, volume 2509 of Lecture Notes in Computer Science, 1-14, Springer--Verlag, 2002.

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

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

[10] B. J. Copeland. Even Turing machines can compute uncomputable functions. In C. S. Calude, J. Casti, and M. J. Dinneen (eds), *Unconventional Models of Computation*, Springer-Verlag, 1998.

[11] *Encyclopaedia of Mathematics, Upper and lower bounds*, Kluwer Academic Publishers, 1993.

[12] D. Graça and J. F. Costa. Analog computers and recursive functions over the reals. *Journal of Complexity*, 19(5): 644-664, 2003.

[13] G. Etesi and I. Nemeti. Non-Turing computations via Malament-Hogarth space--times. *Int. J. Theor. Phys.* 41:341-370, 2002.

[14] S. J. Heims. *John von Neumann and Norbert Wiener: From Mathematics to the Technologies of Life and Death*, MIT Press, 1980.

[15] P. A. Holst. Svein Rosseland and the Oslo Analyser. *IEEE Annals of the History of Computing*, 18(4):16-26, 1996.

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

[17] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics,* 5:115-133, 1943.

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

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

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

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

[22] J. Mycka. μ-Recursion and infinite limits. *Theoretical Computer Science,* 302:123-133, 2003.

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

[24] Jerzy Mycka and José Félix Costa. The computational power of continuous dynamical systems. In *Proceedings of Machines, Computation, Universality, Sankt-Petersburg, 2004*, M. Margenstern, Ed., vol. 3354 of *Lecture Notes in Computer Science*, 163–174, 2005.

[25] J. M. Nyce. Nature's machine: mimesis, the analog computer and the rhetoric of technology. In R. Paton (ed), *Computing with Biological Metaphors*, 414-423, Chapman & Hall, 1994.

[26] P. Odifreddi. *Classical Recursion Theory*, North-Holland, 1989.

[27] P. Orponen. On the computational power of continuous time neural networks. NeuroCOLT Report NC-TR 95-051, Royal Holloway College, Univ. London, Dept. of Computer Science, 1995.

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

[29] A. Ostrowski. Zum Holderschen Satz uber $\Gamma(x)$. *Math. Annalen*, 94: 248-251, 1925.

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

[31] Y. Rogozhin. Small universal Turing machines. Universal machines and computations. *Theoret. Comput. Sci.* 168 (2): 215-240, 1996.

[32] J. F. Ritt and E. Gourin. An assemblage-theoretic proof of the existence of transcendentally transcendental functions. *Bul. Amer. Math. Soc.*, 33: 182-184, 1927.

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

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

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

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

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

[38] J. Traub and A. G. Werschulz. *Complexity and Information*, Cambridge University Press, 1998.

[39] A. Vretblad. *Fourier Analysis and Its Applications*, Springer-Verlag, 2003.

[40] Z. Xia. The existence of noncollision singularities in Newtonian systems. *The Annals of Mathematics*, 135(3):411-468, 1992.