The $P \neq NP$ conjecture in the context of real and complex analysis

Jerzy Mycka * José Félix Costa †

October 31, 2005

Abstract

In this paper we aim at an analog characterization of the classical $P \neq NP$ conjecture of Structural Complexity. We consider functions over continuous real and complex valued variables. Subclasses of functions can be defined using Laplace transforms adapted to continuous-time computation, introducing analog classes $DAnalog$ and $NAnalog$. We then show that if $DAnalog \neq NAnalog$ then $P \neq NP$.

1 Introduction and motivation

We have been working towards recursive definitions of computational classes of functions over $\mathbb{R}$. The first presentation of such a theory, analogous to Kleene's classical theory of recursive functions over $\mathbb{N}$, was attempted by Cristopher Moore [9]. Real recursive functions are generated by a fundamental operator, called differential recursion. The other fundamental operator is the taking of infinite limits, introduced in [10]. In [5] one of the authors, along with Campagnolo and Moore show that a linear form of the differential recursion scheme gives rise to an analog characterization of the class of (Kalmar's) elementary functions and to a general analog characterization of the Grzegorczyk hierarchy. In [4] and [6] it is shown 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 [11] we finally show how to capture higher computational classes through limit operators. Manuel Campagnolo showed also in [3] that other computational complexity classes can be captured through appropriate structured differential schemata or by adding simple or bounded integration. Recently, Olivier Bournez and Emmanuel Hainry proved in [2] that a specific kind of limit operator together with differential recursion makes the class of real recursive functions an exact extension to the real numbers (we can say in the sense of Computable Analysis) of the classical recursive functions.

*Institute of Mathematics, University of Maria Curie-Sklodowska, Lublin, Poland, Jerzy.Mycka@umcs.lublin.pl

†Department of Mathematics, I.S.T., Universidade Técnica de Lisboa, Lisboa, Portugal, fgc@math.ist.utl.pt

We believe that the theory of real recursive functions, with infinite limits [11], has enough ingredients to allow a good 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: in this paper we are much involved in the definition of some analog classes, and to find one analytic representation of the conjecture $P \neq NP$. The main goal will be to connect this open problem with problems of Analysis.

We introduce in this paper the classes $DAnalog$ and $NAnalog$, which are subclasses of real recursive functions defined using the Laplace transform. Let us point out that their names do not indicate time or space complexity, however we will connect $DAnalog$ with PF and $NAnalog$ with NPF (classes PF and NPF will be introduced shortly: they stand for functions computable in polynomial time by deterministic and non-deterministic Turing machines, respectively). Let us add that it is not our purpose to build such classes, which strictly inherit the properties of the classical complexity classes; hence, for example, the use of limits in $DAnalog$ is not forbidden for us. We want to find the classes, which are interesting from a point of view of analog computation and such that some relations between these classes in the analog world are analogous to relations between their counterparts in the discrete case.

2 Growth of functions and subexponentiality

We start by recalling that the computational complexity of functions over the non-negative integers is connected with their rate of growth. Let us look at this definition (see [15]).

Definition 1 *A function h of arity n is defined from a function f of arity n and a function g of arity $(n+2)$, by polynomially bounded primitive recursion, if there are polynomials p and q, and a function t of arity $(n+1)$, such that* [1]

$$h(x_1, \ldots, x_n) \;=\; t(x_1, \ldots, x_n, p(\sum_{i=1}^{n} \lfloor \log(x_i + 1) \rfloor)),$$

where

$$t(x_1, \ldots, x_n, 0) = f(x_1, \ldots, x_n),$$

$$t(x_1, \ldots, x_n, y+1) \;=\; g(x_1, \ldots, x_n, y, t(x_1, \ldots, x_n, y)),$$

such that the following condition holds: for all $y \leq p(\sum_{i=1}^{n} \lfloor \log(x_i + 1) \rfloor)$, we have $\lfloor log(t(x_1, \ldots, x_n, y)) \rfloor \;\leq\; q(\sum_{i=1}^{n} \lfloor \log(x_i + 1) \rfloor)$. [2]

The function log can here be used to measure the size of the inputs represented by bit sequences: if x is one of the inputs, then it can be expressed in $O(\lfloor \log(x+1) \rfloor)$ bits. In this section we could have used, as usual, $|x|$ for the

[1] Function log is of base 2 over $\mathbb{N}$ and of base e over $\mathbb{R}$. In what follows, $\lfloor \log(x_1 + 1) \rfloor + \ldots + \lfloor \log(x_n + 1) \rfloor$ denotes the length of the whole input bit string.

[2] Some linear factor is needed to encode the successive pairing of the inputs in binary.

size of x. Since the size of x is approximately given by $\lfloor \log(x+1) \rfloor$, we will use this intuition as a guideline for next sections, while working with real-valued functions. The size of all inputs taken together is $O(\sum_{i=1}^{n} \lfloor \log(x_i+1) \rfloor)$.

Let us recall that the main model of computation, the Turing machine, has an obvious correspondence with the class of recursive functions. Hence, we can distinguish within the set of recursive functions a subset PF corresponding to deterministic Turing machines working in polynomial time.

A partial function f is said to be *computable* by a deterministic Turing machine M if (a) M accepts the domain of f and (b) if $\langle x_1, \ldots, x_n \rangle \in \mathrm{dom(f)}$, then the accepting computation writes in the output tape the value $f(x_1, \ldots, x_n)$.

Definition 2 *PF is the class of partial functions that can be computed in polynomial time by deterministic Turing machines, i.e., by deterministic Turing machines clocked with polynomials.*

We adopted the definition of partial recursive function given in [1]. Note that for functions in PF we can always check the domain, hence the halting problem in this case is decidable. This problem arises because we use partial functions; partial functions are simpler to handle the non-determinism. We have an inductive definition of the total functions in PF, provided by Buss in 1986 (see, e.g., [15], p. 172):

Proposition 3 *The class of total recursive functions computable in deterministic polynomial time is inductively defined from the basic functions $Z = \lambda n.0$ and $S = \lambda n.n+1$, the projections, basic functions $\lambda n.\ 2n$, $\lambda n.\ 2n+1$, $\lambda n.\ \lfloor \frac{n}{2} \rfloor$, the characteristic function δ of 'equality to 0', by the operations of composition, definition by cases, and polynomially bounded primitive recursion.* □

The intuition behind this characterization of the total functions in PF is the following: a Turing machine clocked in polynomial time p can write at most $p(|x|)$ bits in the output tape. This number is bounded by $2^{|x|^k}$, for some k.

For our purposes, we define a non-deterministic Turing machine in a way similar to which it is used in the probabilistic computational model. We use the notion of a time constructible function: a function is time-constructible if there exist a Turing machine such that for each input x of length n it halts in an accepting state after $f(n)$ steps of computation. We impose the following conditions on non-deterministic machines (see [1] for the probabilistic Turing machine): (a) every step of a computation can be made in exactly two possible ways, which are considered different even if there is no difference in the corresponding actions (this distinction corresponds to two different bit guesses), (b) the machine is clocked by some time-constructible function and the number of steps in each computation is exactly the number of steps allowed by the clock; if a final state is reached before this number of steps, then the computation is continued, doing nothing up to this number of steps, (c) every computation ends in a final state, which can be either *accept* or *reject*, (d) if the machine computes a function, then all accepting computations write down to the output tape the

value of the function (see [1], chapter 2, for a comparison). It is irrelevant what the machine writes in the output tape if it reaches a rejecting state.

A partial function f is said to be *computable* by a non-deterministic Turing machine M if (a) M accepts the domain of f and (b) if $\langle x_1, \ldots, x_n \rangle \in \mathrm{dom(f)}$, then any accepting computation writes on the output tape the value $f(x_1, \ldots, x_n)$.

Definition 4 *NPF is the class of partial functions that can be computed in polynomial time by non-deterministic Turing machines (i.e., by non-deterministic Turing machines clocked with polynomials).*

Definition 5 *We define the class $\exists PF$ as follows: for a function $f : \mathbb{N}^n \to \mathbb{N}$ in $\exists PF$ there exists a function $F : \mathbb{N}^{n+1} \to \mathbb{N}$ in PF and a polynomial $p : \mathbb{N}^n \to \mathbb{N}$ such that (a) $\langle x_1, \ldots, x_n \rangle \in \mathrm{dom(f)}$ if and only if there exists a number k such that $|k| \leq p(|x_1|, \ldots, |x_n|)$ and $\langle x_1, \ldots, x_n, k \rangle \in \mathrm{dom(F)}$ and (b) $f(x_1, \ldots, x_n)$ is defined and $f(x_1, \ldots, x_n) = y$ if and only if there exists a number k such that $|k| \leq p(|x_1|, \ldots, |x_n|)$, $F(x_1, \ldots, x_n, k)$ is defined and $F(x_1, \ldots, x_n, k) = y$, and, for all such $|k| \leq p(|x_1|, \ldots, |x_n|)$ such that $\langle x_1, \ldots, x_n, k \rangle \in \mathrm{dom(F)}$, we have $F(x_1, \ldots, x_n, k) = y$.*

With the above definitions the following fact can be easily proved.

Proposition 6 *The class NPF coincides with $\exists PF$.*

From the last proposition we also have that *the function f has domain in NP if and only if the function F has domain in P*, fulfilling the classical theorem $NP = \exists P$: a set A is in NP if and only if there exists a polynomial p and a set B in P such that $x \in A$ if and only if $\exists |z| \leq p(|z|)$ $(\langle x, z \rangle \in B)$.

A characteristic function of a set A will be the partial function c_A defined by the expression: $c_A(x) = 1$ if $x \in A$, otherwise it is $\perp$. For sets in P this is like defining a total function since the domain of a characteristic function is always decidable (the value $\perp$ is then interpreted as 0).

The following statements, although they are about classical computational complexity, they are (to our knowledge) not found elsewhere for the definitions as given here. Let us start with an obvious result.

Proposition 7 *A set is in P if and only if its characteristic function is in PF. A set is in NP if and only if its characteristic function is in NPF.*

We can observe that, of course $PF \subseteq NPF$. Now let us connect classes of functions and sets.

Proposition 8 *$NP \subseteq P$ if and only if $NPF \subseteq PF$.*

The above proposition can be proved by a simple construction, as a consequence of the two main propositions in this section we can state immediately:

Proposition 9 *$P = NP$ if and only if $PF = NPF$ if and only if $NPF = \exists PF$.*

We know now that whenever the conjecture $P \neq NP$ is in context, then it can be replaced by the problem $PF \neq NPF$. In our framework of computation over $\mathbb{R}$, we deal with functions rather then sets. Hence, we translate a problem about sets (classes of sets) into a problem about functions (classes of functions).

Before considering the next concepts, we should strongly stress that *functions in PF or in NPF can be considered total*. The reason is obvious: whenever a function f is undefined at x we can give to $f(x)$ value 0 (remember that the domain of f is decidable). However, in the particular case of a function f in PF, f can always be considered algorithmically total in PF:

$$\text{new[f](x)} \;=\; \begin{cases} f(x)+1 & \text{if } x \in \text{dom(f)}; \\ 0 & \text{otherwise.} \end{cases}$$

Since for f in PF, dom(f) is in P, the function new[f](x) is in PF.

It is also interesting to observe that NPF is the class of functions of the form $\lambda x_1 \ldots x_n$.if $\langle x_1, \ldots, x_n \rangle \in A$ then $F(x_1 \ldots x_n)$, where $A \in NP$ and $F \in PF$.

We end this Section with the following definition:

Definition 10 *A* quasi-polynomial *(sometimes called weak exponential) is a function from the class of functions*

$$\{2^{\log(x+1)^k} : k \in \mathbb{N}\}.$$

The class of quasi-polynomials is closed under composition.

3 Recursive functions over $\mathbb{R}$ and over $\mathbb{C}$

We have introduced in [11] the concept of *a real recursive function* (modifying the original definition of R-recursive functions in [9]) and the corresponding class $REC(\mathbb{R})$; here we will define the concept of *complex recursive function* and the corresponding class $REC(\mathbb{C})$, needed to work with the Laplace transform.

Definition 11 *$REC(\mathbb{C})$ The class $REC(\mathbb{C})$ of complex recursive vector functions is generated from the complex recursive scalars 0, 1, i, and the complex recursive projections $I_n^j(z_1, \ldots, z_n) = z_j$, $1 \leq j \leq n$, $n > 0$, by the following operators:*

Composition: *if f is a complex recursive vector function with n k-ary components and g is a complex recursive vector function with k m-ary components, then the complex vector function with n m-ary components, $1 \leq j \leq n$,*

$$\lambda x_1 \ldots \lambda x_m.\ f_j(g_1(z_1, \ldots, z_m), \ldots, g_k(z_1, \ldots, z_m))$$

is complex recursive.

Differential recursion: *if f is a complex recursive vector function with n k-ary components and g is a complex recursive vector function with n $(k+n+1)$-ary components, then the complex vector function h of n $(k+1)$-ary components which is the solution of the Cauchy problem, $1 \leq j \leq n$,*

$$h_j(z_1, \ldots, z_k, 0) = f_j(z_1, \ldots, z_k),$$

$$\partial_z h_j(z_1, \ldots, z_k, z) = g_j(z_1, \ldots, z_k, z, h_1(z_1, \ldots, z_k, z), \ldots, h_n(z_1, \ldots, z_k, z))$$

is complex recursive if, for all scalar components, their real and imaginary parts are of the class C^1 *on the largest interval containing* $(0, 0)$ *in which a unique solution exists.*

Infinite limits: *if* f *is a complex recursive vector function with* n $(k+1)$*-ary components, then the complex vector functions* h, h_{inf}, h_{sup} *with* n k*-ary components,* $1 \leq j \leq n$,

$$h_j(z_1, \ldots, z_k) = \lim_{z \to \infty + i0} f_j(z_1, \ldots, z_k, z),$$

$$h_j^{inf}(z_1, \ldots, z_k) = \liminf_{z \to \infty + i0} f_j(z_1, \ldots, z_k, z),$$

and finally

$$h_j^{sup}(z_1, \ldots, z_k) = \limsup_{z \to \infty + i0} f_j(z_1, \ldots, z_k, z)$$

are complex recursive vector functions.

Assembling and designating components: *(a) arbitrary complex recursive vector functions can be defined by assembling scalar complex recursive scalar components; (b) if* f *is a complex recursive vector function, then each of its scalar components is a complex recursive scalar function.*

Complex recursive numbers: *arbitrary complex recursive scalar functions of arity* 0 *are called complex recursive numbers.*

In the above definition, we can take limits (either lim, or lim inf, or lim sup) to $\infty + i0$, but a simple change of variables will allow us to take limits to $x_0 + i\infty$, where x_0 is a real recursive number. Limits to $x_0 + i\infty$ are particularly important to compute, e.g., the inverse Laplace transform, Euler's Γ function, Riemann's ζ function, etc.

The restriction of the domain and range in the above definition to $\mathbb{R}$ leads us to the concept of *a real recursive function*, which is presented in detail in [11]. Here we use only slightly informal definition.

Definition 12 *The class* $REC(\mathbb{R})$ *of real recursive vector functions is defined as in Definition 11 with all domains and ranges restricted to* $\mathbb{R}$ *and with the consequent obvious modifications.*

Let us only recall some useful results. First, the functions $x + y$, $x \times y$, $x - y$, e^x, $\sin x$, $\cos x$, $\frac{1}{x}$, x/y, $\log x$, x^y are real recursive functions. Also some special functions are real recursive too: the Kronecker δ function, the signum function, and absolute value, the Heaviside Θ function (equal to 1 if $x \geq 0$, otherwise 0), the binary maximum max, the square-wave function s ($s(x) = 1$ for $x \in [2n, 2n+1]$, $n \in \mathbb{N}$, and $s(x) = 0$ otherwise), and the floor function. Particularly interesting real recursive functions from $REC(\mathbb{R})$ are functions obtained by the iteration functional. We give the following proposition after [9] and [11] (where the proof can be found).

Proposition 13 *If f is a real recursive function of arity one, then the iteration of f, F, is a real recursive function of arity two, such that, for all $n \in \mathbb{N}$, $F(n,x) = f^n(x)$ (i.e., the iteration functional preserves real recursiveness).*

The last result provides a binary real recursive function F such that, on non-negative integers in the second argument, F gives the iteration function $\lambda xn.\ f^n(x)$. If x is a binary representation of some number and p is a polynomial on $|x|$, then $F(x, p(|x|))$ represents the value after a polynomial running of f.

Proposition 14 *If $f : \mathbb{N}^k \to \mathbb{N}$ is a partial recursive function, then there exists a real valued function $F : \mathbb{R}^k \to \mathbb{R}$ such that*

1. *F is a real recursive,*
2. *for all $x_1, \ \ldots\ , \ x_k \in \mathbb{N}$, we have $F(x_1, \ldots, x_k) = f(x_1, \ldots, x_k)$,*
3. *for all $x_i \in [n_i, n_i + 1]$, $n_i \in \mathbb{N}$, $1 \leq i \leq k$, we have $\min(F(n_1, \ldots, n_k),\ F(n_1 + 1, \ldots, n_k + 1)) \leq F(x_1, \ldots, x_k) \leq \max(F(n_1, \ldots, n_k),\ F(n_1 + 1, \ldots, n_k + 1))$.*

Proof. Let us start with the simple observation that the points 1 and 2 of the proposition can be obtained via the simulation of Turing machines by real recursive functions (e.g., such as in [9, 11]).

Let $F' : \mathbb{R}^k \to \mathbb{R}$ be such a function. The third point can be established by a simple transformation: $F(x_1, \ldots, x_k) = F'(\lfloor x_1 \rfloor, \ldots, \lfloor x_k \rfloor)$. □

For a given function $f : N^k \to N$ we will call its counterpart described in the above manner *a canonical extension*.

We recall from classical recursion theory that the recursion scheme can be substituted by iteration if we join to the set of primitive functions the functions to code and decode pairs of numbers (see [14]). We can add the following theorem, which is based on [4].

Proposition 15 *Let $f : \mathbb{N}^k \to \mathbb{N}$ be a primitive recursive function. Then its canonical extension $F : \mathbb{R}^k \to \mathbb{R}$ can be defined by composition and differential recursion from the basic functions $-1,\ 0,\ 1,\ \Theta$, and the projections.*

In [2] is introduced a *minimalization operator* in order to capture, together with composition and differential recursion, *exactly* the whole class of partial recursive functions and no more integer valued functions. For the same reasons, expressed in the paragraph above, here we prefer the concept of limit as being efficient and allowing us to reason about computability and complexity in Analysis in a natural way.

Now let us consider a complex domain. Here the functions $z_1 + z_2$, $z_1 \times z_2$, $z_1 - z_2$, e^z, $\sin z$, $\cos z$, $\sinh z$, $\cosh z$, $\frac{1}{z}$, z_1/z_2, $\log z$, z^w, $\tan^{-1}(z)$ are complex recursive. We present here only a few justifications. Trigonometric functions can be defined by composition, using the formulas $\sin(z) = \frac{e^{iz} - e^{-iz}}{2i}$ and $\cos(z) = \frac{e^{iz} + e^{-iz}}{2}$. In the case of the hyperbolic functions, we use the formulas $\sinh(z) =$

$\frac{e^z-e^{-z}}{2}$ and $\cosh(z) = \frac{e^z+e^{-z}}{2}$. The functions sin and cos can be also obtained directly as follows: $\sin(z) = -i\ \sinh(iz)$ and $\cos(z) = \cosh(iz)$. The rest of definitions can be done in the standard way (see [11]).

By structural induction we can also prove the following theorems (the second one implies that $REC(\mathbb{R}) \subseteq REC(\mathbb{C})$ in some sense)

Proposition 16 *If a function $f : \mathbb{C} \to \mathbb{C}$ is a complex recursive function, $f(z) = f_{\Re}(x, y) + if_{\Im}(x, y)$, where $\Re\ z = x$ and $\Im\ z = y$, then $f_{\Re}$ and $f_{\Im}$ are real recursive functions.*

Proposition 17 *If $f : \mathbb{R} \to \mathbb{R}$ is a real recursive function in $REC(\mathbb{R})$, then there exists a function $\hat{f} : \mathbb{C} \to \mathbb{C}$ in $REC(\mathbb{C})$ such that its imaginary part at $x + i0$ vanishes and the following equation holds*

$$f(x) = \hat{f}(x + i0).$$

Let us add a digression about using the Laplace transform and its inverse in the context of real and complex recursive functions. The formulas

$$F(s) = \int_0^\infty f(\xi)\ e^{-s\xi}\ d\xi, f(\xi) = \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} F(s)\ e^{s\xi}\ ds$$

are referred to as the Laplace transform $F = L[f]$ and the inverse Laplace transform $f = L^{-1}[F]$, respectively. The second integral is generally carried out by a contour integration.

Following [16], consider a real recursive function f on the positive real axis such that: (i) f is continuous on $[0, \infty)$ except possibly for a finite number of jump discontinuities in every finite sub-interval; (ii) there is a positive number M such that the absolute value $|f(\xi)| \leq Me^{k\xi}$, for all $\xi \geq 0$, in which case we say that f belongs to the class L_k. Additionally let $L = \bigcup_{k>0} L_k$. Then we can cite the following after [12].

Proposition 18 *If $f \in L_k$ is a real recursive function, then the Laplace transform $L[f](x + iy)$ exists for $x > k$ and it is complex recursive.*

From the following infinite limits of functions defined by differential recursion (but given by simple integration) $\lim_{\tau\to\infty} \int_0^\tau e^{-s\xi}\ f(\xi)\ d\xi$ and $\lim_{\tau\to 0+i\infty} \int_{c-\tau}^{c+\tau} e^{\xi z}\ F(z)\ dz$ we can obtain the following result.

Proposition 19 *The Laplace transform and the inverse Laplace transform are, respectively, real recursive and complex recursive functions, whenever their function arguments are real recursive and complex recursive functions, respectively.*

We are now going to show, how to use the Laplace transform together with the Bromwich contour to get a good understanding of the class of recursive functions over the reals and over the complexes. We also show that almost all real recursive functions obtained and used to our purpose in previous papers (e.g., [9, 5, 6, 11, 12]) can be obtained in a very elegant and ingenious way.

The *guideline* is as follows: taking a real recursive function $f(x)$ we obtain its complex recursive transform $F(s)$, from which we obtain a real recursive function $F(x+i0)$, and then we apply again the machinery or combining two complex recursive functions $F_1(s)$ and $F_2(s)$, we find a new complex recursive function $F(s)$, from which we obtain a real recursive function using the Bromwich contour. In order to use the Bromwich contour, we have to ensure that the given function F of the complex variable s is analytic throughout the finite s plane except for a finite number of isolated singularities.

We start with the Heaviside step function Θ. The Laplace transform is the complex recursive function $L[\Theta(x)](s) = \frac{1}{s}$. From this function we conclude that also $\lambda x.\ \frac{1}{x}$ is also real recursive. Since simple integration of a real recursive function is a real recursive function, integrating $\Theta(x)$ we obtain two new real and one new complex recursive functions:

$$x = \int_0^x \Theta(\xi)d\xi, \quad L[\int_0^x \Theta(\xi)d\xi](s) = \frac{L[\Theta(x)](s)}{s} = \frac{1}{s^2}.$$

Iterating this procedure we get more recursive functions: starting with $\Theta(x)$ we reach two new real and one new complex recursive functions: x^n, $\frac{1}{s^{n+1}}$. From this reasoning it also follows that the function $\lambda x.\ \frac{1}{x^n}$ is real recursive, for all $n > 0$.

In general, when we have a periodic function f such that $f(x) = f(x+a)$, its Laplace transform is given by

$$\frac{\int_0^a\ e^{-s\xi}\ f(\xi)\ d\xi}{1-e^{-as}}.$$

Using the Bromwich contour to transform back the complex recursive function $\lambda s.\ \frac{1}{s^2} \times \tanh s$ we get the saw-tooth function

$$st(x)\ =\ x\Theta(x) + 2\sum_{n=1}^{\infty}\ (-1)^n\ (x-2n)\ \Theta(x-2n).$$

The square-wave and the saw-tooth functions are used in [4, 9, 11] to obtain the iteration of a real recursive function as real recursive function. From the iteration of a real recursive function we can then get the simulation of the execution of a Turing machine.

As a final remark, we note that many periodic discontinuous functions can be obtained using the method just described. However we should keep in our minds the following statement. If f and g both belong to L, and $F(s) = G(s)$ for all (sufficiently) large values of s, then $f(t) = g(t)$ for all values of t where f and g are continuous. The last statement (taken from [16]) shows that the above mentioned functions are defined up to a countable number of discontinuity points, a set of measure zero, that does not interfere with the solutions of differential equations involving such functions.

The consideration of Turing machines in the context of real or complex recursive functions is not of utmost importance, since we already have a characterization of the classes of functions computed in polynomial time by deterministic

and non-deterministic Turing machines using recursive function theory. However, we would like to justify our definition in this aspect too.

Let us recall that the set of possible computations of a non-deterministic Turing machine M on an input string w can be described by a binary tree of computations: the nodes of the tree are configurations of the machine M on input w; the root is the initial configuration, and at any node, its sons are those configurations which can be reached in one move. In this way we can simulate a run of a non-deterministic Turing machine M, giving to M an input word x together with a guess word y, i.e., two real numbers, which code for the input and the guess, respectively. Note that, given a *guess* y, then after retrieval of the last bit $b := b(y)$, the remaining part is given by $r := \frac{y-b(y)}{2}$. This function is analytic if we find an analytic way of determining $b(y)$. We know that, as consequence of our choice of binary alphabet, the last bit of the guess is just

$$b := 1 - \frac{\cos(\pi y) \times (1 + \cos(\pi y))}{2}.$$

We can then look at b as a function of real y. This real function for integer arguments gives b, which is a bit.

Using the analytic map of two dimension introduced in [8] and extending it with the guess we find that there are analytic real recursive functions which can simulate the transition of any non-deterministic Turing machine. All what is needed to justify this statement is the additional component of f_M to use a variable for non-determinism:

$$[f_M(x_1, x_2, y)]_3 := \frac{y - b(y)}{2}.$$

To obtain the function computed by M it is enough to iterate the steps until we reach the final state. We can use limits here: if the final state is found then the sequence $\{f_M^n(x_1, x_2, y)\}_{n\in\mathbb{N}}$ has a limit, otherwise it diverges. Let us add that the halting problem for Turing machines is decidable by a real and a complex recursive function (see [9, 11]). However, it does not mean that all problems are decidable by means of real recursive functions. The full discussion and examples of the problems undecidable by means of real (and complex) recursive functions is given in [13].

4 *DAnalog* and *NAnalog*

A real (or complex) total recursive function is said to be of exponential order if in every step of its construction, its components (the components of their real and imaginary parts in the complex case) are of exponential order. It is said to be of subexponential order if in every step of its construction, its components (the components of their real and imaginary parts in the complex case) are of subexponential order.

In Analysis we have a precise boundary between two different worlds of computable functions. The *subexponential* world has the following property: for

every function f, $L[f](s)$ is defined along the whole positive real axis $\Re\ s\ >\ 0$. The *exponential* world has the weaker property: for every function f, $L[f](s)$ is defined for values of the complex variable s such that $\Re\ s\ \geq\ x_f$, where x_f depends on f.

To consider some functions as subexponential, sometimes we have to make a small shift on the real variable to avoid a discontinuity at the origin. E.g., function $\lambda x.\ \frac{1}{x+\epsilon}$ is subexponential and its transform is $\lambda s.\ e^{\epsilon s} E_1(\epsilon s)$, where E_1 is the exponential integral of degree one, for positive ϵ as small as we want.

Trivially, it can be shown that:

Proposition 20 *Subexponential functions are preserved by differentiation.*

Proof. We will prove that if f is a total function of x, such that its Laplace transform is defined for the all positive real axis, then the function $\bar{f}$ defined by $\bar{f}(x) = \partial_x f(x)$ is subexponential too. We have $L[\lambda x.\ \partial_x f(x)](s)\ =\ s\ F(s) - f(0)$ whenever $L[f]\ =\ F$. This fact completes our proof, since $\lambda s.\ s\ F(s)$ is defined for all the positive real axis, whenever F is defined too in the same open interval. □

Proposition 21 *Subexponential functions are preserved by integrals.*

Proof. We will prove that if f is a total function of x, such that its Laplace transform is defined for the all positive real axis, then the function $\bar{f}$ defined by $\bar{f}(x) = \int_0^x\ f(\xi)\ d\xi$ is subexponential too. Let f be a real valued function of ξ. We have $L[\int_0^x\ f(\xi)\ d\xi](s)\ =\ \frac{F(s)}{s}$ whenever $L[f]\ =\ F$. This fact completes our proof, since $\frac{F(s)}{s}$ is defined for all the positive real axis, whenever F is defined too in the same open interval. □

Proposition 22 *Exponential functions are preserved by limits (i.e. if the limit of a exponential function exists, then it is exponential).*

Proof. We have to prove that if f is a total function of x and y, such that their Laplace transforms with respect to x and y exists, then the function $\bar{f}$ defined by $\bar{f}(x) = \lim_{\upsilon\to\infty}\ f(x,\upsilon)$, if such a limit exists, it is exponential too. Consider f, a real valued function of x and υ, and any infinite divergent sequence υ_n of real numbers; we have, for all such sequences υ_n, assuming that $f_n(x) = f(x,\upsilon_n)$ is measurable,[3]

$$L_x[\lim_{n\to\infty}\ f_n(x))](s)\ =\ \int_0^\infty \lim_{n\to\infty}\ e^{-s\xi}\ f(\xi,\upsilon_n)\ d\xi\ =\ \lim_{n\to\infty}\ \int_0^\infty e^{-s\xi}\ f(\xi,\upsilon_n)\ d\xi$$

$$=\ \lim_{n\to\infty}\ L_x[f_n(x))](s).$$

This means that $L_x[\lim_{\upsilon\to\infty}\ f(x,\upsilon)](s)\ =\ \lim_{\upsilon\to\infty}\ L_x[f(x,\upsilon)](s)$,
i.e., $L_x[\lim_{\upsilon\to\infty}\ f(x,\upsilon)](s) = \lim_{\upsilon\to\infty}\ L_x[f(x,\upsilon)](s) = \lim_{\upsilon\to\infty}\ F_x(s,\upsilon)$ with

[3] A statement that is always true for real recursive functions defined over $\mathbb{R}$.

$L_x[f(x,\upsilon)](s) = F_x(s,\upsilon)$. Using the Bromwich contour we obtain, in similar lines,

$$\frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} \lim_{\upsilon\to\infty} F_x(s,\upsilon)\, e^{s\xi}\, ds = \lim_{\upsilon\to\infty} f(x,\upsilon)$$

that is $\lim_{\upsilon\to\infty} f(x,\upsilon) = L^{-1}[\lim_{\upsilon\to\infty} F_x(s,\upsilon)](x)$, and, since the limit in the left hand expression exists and F_x is defined, the limit in the right hand must also exist, meaning that, $\lim_{\upsilon\to\infty} f(x,\upsilon)$ is exponential too. □

We propose the definition of the classes *DAnalog* and *NAnalog*, which can be interpreted as classes of real recursive functions computed with *quasi-polynomial restrictions* on their values. The definitions are not suggested by the classical counterparts, but they will be strongly motivated. Of course, according to what was said above, the classes *DAnalog* and *NAnalog* will be defined as subclasses of $REC(\mathbb{R})$.

Definition 23 *The class of real recursive functions, designated by DAnalog, is defined as the collection of all real recursive functions f such that:*

1. *$L[f]$ is defined on the whole positive axis;*
2. *f is 0, 1, -1, θ_2 ($\theta_2(x) = x^2\Theta(x)$) or a projection, otherwise f can be given as composition or differential recursion of functions from DAnalog;*[4]
3. *in every step of the construction of f, each component of f can not grow faster than a quasi-polynomial.*

Of course, the last condition implies the first one. In this paper we are concentrated on the restrictions $e^{\log(x)^k}$, but let us stress that we treat the Laplace transform as a fundamental tool in our work, while restrictions can be done in many different ways. Indeed, we consider that conditions 1 and 3 alone induce the class of feasible analog computations.

It is obvious to observe that *DAnalog* is closed under differentiation. Let us add that *DAnalog* is closed for integration too. We have $\phi_1(x) = \int_0^x f(\xi)\, d\xi > \phi_0(x) = f(x)$. Integrating successively we get a infinite non-collapsing chain $\phi_0(x) < \phi_1(x) < \ldots < \phi_n(x) < \ldots$, where, for each pair of functions, the inequality holds almost everywhere: e.g., starting with 1, we will get all the powers n^k, for all k. We also know that the exponential is a fixed point in this hierarchy:

$$e^x = \int_0^x e^{\xi}\, d\xi.$$

However, starting from 1 we will never cover the whole gap between 1 and e^x. If some sub-exponential function f is reached, then the next step will be a function γ_1 such that $\gamma_1(x) < xf(x)$. From here, we get $\gamma_2(x) < x^2 f(x), \ldots$ and so on, a new sub-hierarchy obtained by means of polynomial factors. This means *inter alia* that from a polynomial we will never reach a quasi-polynomial by

[4] The inclusion of function θ_2 in the basis is a consequence of removing the limit operator from the definition of *DAnalog*, see e.g. [4].

successive integrations and that from a quasi-polynomial we will never reach a function not belonging to this class.

The property of $DAnalog$ of being closed under differentiation and integration is considered a requirement of a well-defined analog class.

We define an admissible bounded quantification. We start with a class FUN of functions; $\exists\ FUN$ is defined as in the classical framework: a function $f : \mathbb{R}^n \to \mathbb{R}$ is in $\exists\ FUN$ if there exists a function $F : \mathbb{R}^{n+1} \to \mathbb{R}$ in FUN and polynomial $\phi : \mathbb{N}^n \to \mathbb{N}$, such that (a) $\langle x_1, \ldots, x_n\rangle \in$ dom(f) if and only if there exists a number k such that $|k| \leq \phi(||\lfloor x_1\rfloor||, \ldots, ||\lfloor x_n\rfloor||)$ and $\langle x_1, \ldots, x_n, k\rangle \in$ dom(F) and (b) $f(x_1, \ldots, x_n)$ is defined and $f(x_1, \ldots, x_n) = y$ if and only if there exists a number k such that $|k| \leq \phi(||\lfloor x_1\rfloor||, \ldots, ||\lfloor x_n\rfloor||)$, $F(x_1, \ldots, x_n, k)$ is defined and $F(x_1, \ldots, x_n, k) = y$, and, for all such $|k| \leq \phi(||\lfloor x_1\rfloor||, \ldots, ||\lfloor x_1\rfloor||)$ such that $\langle x_1, \ldots, x_n, k\rangle \in$ dom(F), we have $F(x_1, \ldots, x_n, k) = y$.

Definition 24 *The class of real recursive functions, designated by $NAnalog$, is defined as the collection of all real recursive functions obtained from functions in $DAnalog$ by admissible bounded quantification (i.e. $NAnalog = \exists\ DAnalog$).*

Using the same techniques we used in [11] to represent by means of real recursive functions the entire arithmetical and analytical hierarchies, we can define the class $NAnalog$ by analytical means, without quantifiers.

We get the immediate result:

Proposition 25 *$DAnalog \subseteq NAnalog$.*

Proof. This proof follows directly from definitions of $DAnalog$ and $NAnalog$. Each function f of arity n in $DAnalog$ can be seen as a function F of arity $(n+1)$ in $DAnalog$ where the additional variable is introduced by composition with $(n+1)$-ary projections. □

Example 26 *The functions $x + y$, xy, $x - y$, $\frac{1}{x+\epsilon}$, for all $\epsilon \in \mathbb{R}^+$, and $\frac{x}{y+\epsilon}$, for all $\epsilon \in \mathbb{R}$, belong to $DAnalog$. The reference functions $e^{\log(x)^k}$, for all $k \in \mathbb{N}$, is also in $DAnalog$. Also* sin *and* cos *are in $DAnalog$.*

We can give immediately a negative result that the function e^x is not in the class $DAnalog$.

We can add that it is possible to prove that many physical systems are in $DAnalog$, e.g.: RC circuits, RLC circuits, harmonic oscillators.

Now, we continue the characterization of analog classes of feasible analog computation.

Proposition 27 *For every function $f \in PF$, the Laplace transform $L[\hat{f}]$ of its canonical extension $\hat{f}$ is subexponential.*

Proof. Let us consider a function f of arity 1 from PF. From the definition of the class PF, the canonical extension of such a function grows as fast as $e^{\log(x)^k}$, for some positive integer k, up to a multiplicative constant. We then have, with

$y = \log(x)$, $dx = e^y\ dy$, and considering without loss of generality that k is odd, that $L[\hat{f}](s)$ is given by $2\ \int_0^\infty\ e^{-se^y+y^k+1}\ dy$. For all $k \in \mathbb{N}$, we have

$$L[\hat{f}](s) = 2\ \int_0^\infty\ e^{-se^y+y^k+1}\ dy$$

and, for y greater than some y_0,

$$e^{-se^y+y^k+1} < e^{-sy}.$$

Let $M = 2\ \int_0^{y_0}\ e^{-se^y+y^k+1}\ dy$. We have that M is defined and that $\int_{y_0}^\infty\ e^{-sy}\ dy = \frac{e^{-sy_0}}{s}$ is defined for all s. □

We may modify the proof of Proposition 15 to deal with the most interesting cases immediately.

Proposition 28 *For every function $f \in PF$ of arity k, there exists a real recursive function $\hat{f} \in DAnalog$, of the same arity, such that $f(n_1, \ldots, n_k) = \hat{f}(n_1, \ldots, n_k)$, for every $n_1, \ \ldots, \ n_k \in \mathbb{N}$.*

Proof. The proof is given by structural induction. Let us consider the structure of functions of lower arity in PF. If the function is Z, S, I_n^j, for some $n > 0$ and $1 \leq j \leq n$, $\lambda x.\ 2x$, $\lambda x.\ 2x+1$, $\lambda x.\ \lfloor \frac{x}{2} \rfloor$, or δ, then it has its counterpart in $DAnalog$ – this fact can easily be shown (cf. [11], where all these constructions are presented). Similar reasoning applies to basic functions of coding and decoding pairs. The definition by cases is trivial to bound.

Let $h(n) = f(g(n))$, where for f and g in PF we have by induction extensions $\hat{f}$ and $\hat{g}$ in $DAnalog$. Then $\hat{h}(x) = \hat{f}(\hat{g}(x))$ is an adequate representation of h.

The last step is devoted to polynomially bounded recursion. Let f be given by $h(x) = v(x, p(|x|))$, with

$$v(x,0) = f(x), \quad v(x, y+1) = g(x, y, v(x,y)),$$

where for $y \leq p(|x|)$, we have $|v(x,y)| \leq q(|x|)$, where p and q are integer polynomials and $|x|$ the length of x, given by $\lfloor \log(x+1) \rfloor$ as usual.

We know (by an inductive hypothesis) that f and g have their counterparts in $DAnalog$, given by $\hat{f}$ and $\hat{g}$. Primarily, we are interested in the function $\hat{v}$ given by the following iteration relation: $\hat{v}(x,y) = I_3^3(T(y, (x, 0, \hat{f}(x))))$, where $T(n, (x,y,z)) = t^n(x,y,z)$, $t(x,y,z) = (x, y+1, \hat{g}(x,y,z))$. We can see that $\hat{v}$, defined in this way, has the property $\hat{v}(n,y) \leq \max(v(n, \lfloor y \rfloor), v(n, \lfloor y+1 \rfloor))$. Because v is bounded by some polynomial on the length of y, then $\hat{v}$ will be bounded by $e^{\log(x)^k}$, for some $k \in \mathbb{N}$, up to a multiplicative constant. This means that $\hat{h}(x) = \hat{v}(x, p(\log(x)))$ is in $DAnalog$. □

Now by virtue of Definition 24 of $NAnalog$, we have for free the following result:

Proposition 29 *For every function $f \in NPF$ of arity k, there exists a real recursive function $\hat{f} \in NAnalog$, of the same arity, such that $f(n_1, \ldots, n_k) = \hat{f}(n_1, \ldots, n_k)$, for every $n_1, \ \ldots, \ n_k \in \mathbb{N}$.*

Now we change the direction of our consideration. We start from real functions and then we restrict them to the set of non-negative integers.

Definition 30 *A recursive function $f : \mathbb{N} \to \mathbb{N}$ is said to be an* admissible restriction *of $F : \mathbb{R} \to \mathbb{R}$ if there exists an indexed ordered set $\{\alpha_i\}_{i\in\mathbb{N}}$ such that $f(i) = F(\alpha_i)$, for all $i \in \mathbb{N}$. Then the set $\{\alpha_i\}_{i\in\mathbb{N}}$ is said to be* admissible *for the function F.*

We are interested in admissible bounded indexed ordered sets in the sense that any two contiguous values, α_i and α_{i+1}, are bounded by a polynomial ψ, i.e., $||\lfloor\alpha_{i+1}\rfloor|| < \psi(||\lfloor\alpha_i\rfloor||)$. We can consider functions with many variables. A tuple of indexed ordered sets of real numbers $\{\alpha_i^1\}_{i\in\mathbb{N}}, \ldots, \{\alpha_i^k\}_{i\in\mathbb{N}}$ is said to be *admissible* for a function $F : \mathbb{R}^k \to \mathbb{R}$, of arity k, if $F(\alpha_{i_1}^1, \ldots, \alpha_{i_k}^k) \in \mathbb{N}$, for all $i_1, \ldots, i_k \in \mathbb{N}$, all sequences of real numbers are admissible bounded sets, and all pairs of indexed sets $\{\alpha_i^\mu\}_{i\in\mathbb{N}}$, $\{\alpha_i^\nu\}_{i\in\mathbb{N}}$ are polynomial bounded: there exists polynomial ϕ such that $||\lfloor\alpha_i^\mu\rfloor|| \leq \phi(||\lfloor\alpha_i^\nu\rfloor||)$ and $||\lfloor\alpha_i^\nu\rfloor|| \leq \phi(||\lfloor\alpha_i^\mu\rfloor||)$.

Let us present for the example that for e^x with a sequence $a_n = 2\log n, n \geq 1$, we have the admissible restriction $F(n) = n^2$. Not all functions have admissible restrictions, like $\lambda x.\ e^{-x}$, or just like a constant $\frac{1}{2}$. Real recursive functions that extend functions over the integers have infinite admissible restrictions. For those functions f that have no admissible restrictions we can construct $\lfloor f\rfloor$, which always has admissible restrictions.

We can now formulate and prove the following proposition:

Proposition 31 *Let $H : \mathbb{R}^k \to \mathbb{R}$ be a real recursive function in DAnalog. If an admissible restriction h of H, induced by an admissible set, exists, then h is in PF.*

Proof. Let us consider for simplicity the case of one variable. The statement will be proved by induction in the structure of H. If H is one of the basic functions in *DAnalog*, then its restriction to non-negative integers (e.g., the admissible set $\{n\}_{n\in\mathbb{N}}$) is obviously in PF.

Now consider that H is obtained by composition: $H = G \circ F$. If a set α exists such that $h(i) = H(\alpha_i) \in \mathbb{N}$, for all $i \in \mathbb{N}$, then $H(\alpha_i) = G(F(\alpha_i))$. Hence, the set $\gamma = \{F(\alpha_i)\}_{i\in\mathbb{N}}$ is an admissible set for G. By structural induction $g(i) = G(\gamma_i)$ is in PF and because $h(i) = g(i)$ we conclude that $h \in PF$.

The last part of the proof is devoted to differential recursion. Consider a function H of arity one, defined by $H(0) = F$ ($= 0$, without loss of generality), $\partial_y H(y) = G(y, H(y))$ such that $H(y) \leq e^{\log(y)^k}$, for some k, up to a multiplicative constant. Let α be an admissible set for H. Of course we can rearrange α_i to be strictly monotonic. Then $H(\xi) = \int_0^\xi G(y, H(y))\ dy$. We can divide the right-hand side of this equation into intervals $[\alpha_i, \alpha_{i+1}]$, and use the mean value theorem to obtain $H(\alpha_n) = \sum_{i=0}^{n-1} G(\xi_i, H(\xi_i))$, where $G(\xi_i, H(\xi_i)) = \int_{\alpha_i}^{\alpha_{i+1}} G(y, H(y))\ dy$. Since in *DAnalog* there are functions from L_k, hence the mean value theorem can be used here.

If α is the admissible set for H, then $H(\alpha_i)$ can be presented as a sum of values of some natural function $k(i)$. For $H(\alpha_i) \in \mathbb{N}$, we have $\sum_{i=0}^{n-1} G(\xi_i, H(\xi_i)) =$

$H(\alpha_i) \in \mathbb{N}$. This holds for all $n > 0$, hence $G(\xi_i, H(\xi_i)) \in \mathbb{N}$, for every $i \in \mathbb{N}$. Let $\gamma_i = H(\xi_i)$. We define $k(i) = G(\xi_i, \gamma_i)$. From an inductive hypothesis $k(i)$ is in PF, then we find that h is in PF: G is polynomially bounded and $h(i) = \sum_{j=0}^{i-1} k(j)$, and the sum is polynomially bounded by the condition on G, hence h is also in PF. □

Let the reader see how the last proof is based on an analytic result — the *mean value theorem*. We stress this fact to show how the intermixing computability and analysis provide an analytic language to the theory of computation. Approximation techniques can also be used like in Computable Analysis (see, e.g., [17]). However, these approximation techniques give rise to differential equations having solutions with large plateaux between consecutive integers and fast growing rates close to the integer values of the argument.

Similarly, we can prove that:

Proposition 32 *Let $H : \mathbb{R}^k \to \mathbb{R}$ be a real recursive function in $NAnalog$. If an admissible restriction h of H, induced by an admissible polynomial bounded set, exists, then h is in NPF.*

Proof Every function f of arity n in $NAnalog$ corresponds to a function F of arity $n+1$ in $DAnalog$. Since f comes from F by universal quantification in the last variable of F, all admissible restrictions of f are extensible to admissible restrictions of F. According to Proposition 31, all these extended admissible restrictions, say ζ, lie in PF. But any such ζ is suitable for quantification in the last variable, providing functions ξ in NPF. Reasoning this way we can also suppose that all functions involved grow slower than $\lambda x.\ e^{log(x)^k}$ in all its arguments. □

5 The Conjecture $DAnalog \neq NAnalog$

We now show that the nature of computational classes PF and NPF induces a relation (strict inclusion or equality) between analog classes $DAnalog$ and $NAnalog$, respectively, analogous to the classical open relation between P and NP.

Let us indicate the following property, which is implied by Propositions 28 and 29: if $f \in PF$ or $f \in NPF$ then its canonical extension F is bounded by a quasi-polynomial in any step of the construction of F.

If we restrict classes $DAnalog$ and $NAnalog$ to classes of functions that have admissible restrictions - and this will not be loosing generality, but rather a definition strategy that can be restated as we saw before in Section 4 - then Proposition 34 and the main theorem of this section will follow.

Proposition 33 *If a function has an admissible restriction in any step of its construction, and in any step of its construction all its admissible restrictions are quasi-polynomially bounded, then it belongs to $DAnalog$.*

Proof. Suppose that a function f of arity one in these conditions is not in $DAnalog$. Then we can find, in some step of the construction a component g such that, for all $k \in \mathbb{N}$, $g(x) \geq 2^{\log(x)^k}$. The function g has an admissible restriction which is not quasi-polynomially bounded, contrarily to our hypothesis. Thus f is in $DAnalog$. □

Proposition 34 *If $NP \subseteq P$ then $NAnalog \subseteq DAnalog$.*

Proof. If f is a function in $NAnalog$, then all their admissible restrictions along its construction are non-deterministic quasi-polynomially bounded (i.e., they are in NPF), and then, by hypothesis, they are deterministic quasi-polynomially bounded (i.e., they are in PF). We end the proof applying Proposition 33. □

And now, by the simple contraposition we can finally state the main result.

Proposition 35 *If $NAnalog \neq DAnalog$ then $NP \neq P$.*

Hence, considering quasi-polynomially bounded functions over the real numbers, which have admissible restrictions, then an analytic proof that $DAnalog \neq NAnalog$ (in the continuum) will provide a proof that P is a proper subclass of NP.

We now are in a proper situation to ask if there is simple mathematical condition in Complex Analysis to check the conjecture $DAnalog \neq NAnalog$. That is, to express the conjecture $P \neq NP$ and solve it by means of analytical tools. The precedent proposition implies that $P \neq NP$ if, for some function $f \in NAnalog$, which has admissible restrictions, does not exist a new system of differential equations for f in $DAnalog$.

6 Conclusions

In this paper we presented formally only the implication: if $NAnalog \neq DAnalog$ then $NP \neq P$. We leave to the reader the proof of the converse implication: Let $f \in NPF$; then f has a canonical extension in $NAnalog$; if $NAnalog \neq DAnalog$, then this canonical extension is also in $DAnalog$; we conclude that the function f is in PF as we wished to show.

We are now considering Sobolev spaces. We wanted to show here that the consideration of feasible analog circuits made of elastic strings and viscous dashpots ([7]) lead to the Laplace transform as a measure of feasible analog computation. Away of physical considerations, the use of Sobolev spaces will allow the replacement of the Laplace transform by a suitable transform measuring rates of growth adapted to our cases. This direction of research is, at it seems, reasonable and, possibly, most fruitful for a better representation of P in the analog realm.

As usual, we can not close this paper without listing some still open problems:

- Is $REC(\mathbb{C})$ richer than $REC(\mathbb{R})$ under growth constraints? E.g., is the class $DAnalog$ over $\mathbb{C}$ richer than the class $DAnalog$ over $\mathbb{R}$? We believe that the answer is YES, with regard to the different power of Cauchy's problem in real and complex space.

- The simulation of a non-deterministic Turing machine designed to compute a function shows clearly that $NAnalog$ is well conceived: a suitable guess exists such that the value of the function, for a given input, is computed. Can this function be obtained by integration in the complex plane? We think that a deep exploration of this idea would bring more light to $P \neq NP$ conjecture.

- Can the Laplace transformation pair replace the differential recursion scheme in the case of some computational subclasses of real recursive functions? That would be a quite elegant formulation.

Acknowledgments. Thanks to Agnieszka for her support.

We thank to Samson Abramsky from CompLab in Oxford and to Nachum Dershowitz from the School of Computer Science, Tel Aviv University, for their encouragement. Martin Davis was precious to us, whether being encouraging or critical. Martin *namque erit ille mihi semper deus*.[5]

Our friend Professor António Portela from the Technical University of Lisbon pointed to us very interesting physical aspects of the Laplace transform in dissipative systems, being of a mechanical nature, an electric nature, whenever *friction* exists. Part of Section 2 was produced as an answer to many questions raised by Fernando Ferreira from the University of Lisbon.

We also thank to the two referees that helped to make this version of the paper.

To Cris Moore, from New Mexico University, we express our gratitude and admiration for having been the mentor of our work in 1995.

References

[1] José L. Balcázar, Josep Díaz, and Joaquim Gabarró. *Structural Complexity I*, Springer-Verlag, Second Edition, 1995.

[2] Olivier Bournez and Emmanuel Hainry. Real recursive functions and real extensions of recursive functions. Machines, Computation, Universality (MCU 2004), *Lecture Notes in Computer Science*, 3354:116-127, 2005, Springer-Verlag.

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

[5] Vergil, Bucolica I, reference to Augustus who received the title of *god* in the year 29.

[4] Manuel L. Campagnolo, Cristopher Moore, and José Félix Costa. Iteration, inequalities, and differentiability in analog computers. *Journal of Complexity*, 16(4):642-660, 2000.

[5] Manuel L. Campagnolo, Cristopher Moore, and José Félix Costa. An analog characterization of the Grzegorczyk hierarchy. *Journal of Complexity*, 18(4):977–1000, 2002.

[6] Daniel Graça and José Félix Costa. Analog computers and recursive functions over the reals. *Journal of Complexity*, 19(5): 644-664, 2003.

[7] Dorothy E. Hyndman. *Analog and Hybrid Computing*. Pergamon Press, 1970.

[8] Pascal Koiran and Cristopher Moore. Closed-form analytic maps in one and two dimensions can simulate universal Turing machines. *Theoretical Computer Science,* 210(1): 217-223, 1999.

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

[10] Jerzy Mycka. μ-recursion and infinite limits. *Theoretical Computer Science*, 302: 123-133, 2003.

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

[12] Jerzy Mycka and José Félix Costa. The computational power of continuous dynamic systems. Machines, Computation, Universality (MCU 2004), *Lecture Notes in Computer Science*, 3354:164-175, 2005, Springer-Verlag.

[13] Jerzy Mycka and José Félix Costa. Undecidability over continuous-time, *Logic Journal of the IGPL*, Oxford University Press, in print.

[14] Piergiorgio Odifreddi. *Classical Recursion Theory I*, Elsevier, 1992.

[15] Piergiorgio Odifreddi. *Classical Recursion Theory II*, Elsevier, 1999.

[16] Anders Vretblad. *Fourier Analysis and Its Applications*, Graduate Texts in Mathematics 223, Springer-Verlag, 2003.

[17] Klaus Weihrauch. *Computable Analysis, An Introduction*, Texts in Theoretical Computer Science, Springer-Verlag, 2000.