

Bit Complexity of Initial Value Problems

Akitoshi Kawamura

Department of Computer Science

University of Toronto

10 King's College Road, Toronto, ON M5S 3G4 Canada

Abstract. How complex could the solution be to an initial value problem given by a polynomial-time computable function? This question can be given a natural and precise sense in computational complexity theory. We answer the question by several taxonomical results, known and new, stating that more conditions on the problem make the solution simpler. In particular, Lipschitz continuity alone may leave the solution polynomial-space complete, but analyticity makes the solution polynomial-time computable.

1 Introduction

Complexity theory classifies discrete problems by the amount of computational resources needed for a Turing machine to solve them. Polynomial-time computability defined there is often identified with the intuitive notion of a problem being “tractable” or “efficiently solvable” on a digital computer in the real world [22].

Despite its discrete nature, the theory can be applied [5, 12, 26] to problems involving real numbers, providing a formal account of what can be done numerically by a digital computer when validated precision is required. We will review the definitions of computability and polynomial-time computability of real functions in this framework in Section 2.

We will then turn to the initial value problem

$$h(0) = 0, \quad h'(t) = g(t, h(t)), \quad 0 \leq t \leq 1, \quad (1)$$

and present some results stating how complex the solution h can be in comparison to g . More precisely, suppose that g is polynomial-time computable. Then

- (A) it may be the case that there are infinitely many solutions h , all of which are non-computable [9, 20];
- (B) if we assume that the equation has a unique solution h , then it is computable but can take arbitrarily long time to compute [9, 16, 20];

1991 *Mathematics Subject Classification.* 34A99, 65L05, 65Y20, 68Q17.

The author was supported in part by the Nakajima Foundation and by the Natural Sciences and Engineering Research Council of Canada.

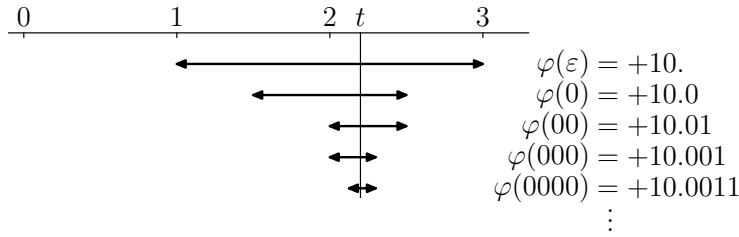


Figure 1 A string-to-string function φ represents $t \in \mathbf{R}$ by giving a sequence of approximations of t with increasing precisions.

- (C) if we assume that g is Lipschitz continuous, then the (unique) solution h is polynomial-space computable [9] but can be polynomial-space complete [8] (Section 3 of this note);
- (D) if we further assume that g is analytic, then the (unique) solution h is also polynomial-time computable (proved in Section 4).

2 Computational complexity of real functions

This section formulates computation over the real numbers. Our model of computation reflects the fact that digital computers can only work on bit strings and not directly on real numbers. It is thus consistent with the classical theory of computation. We assume that the reader has a casual knowledge of the basic concepts of complexity theory, including Turing machines and polynomial time/space computation [22]. The computability notion for real functions equivalent to the one discussed below dates back at least to Grzegorzcyk [5] and has been studied extensively in the field of *computable analysis*, or *recursive analysis*. Study on polynomial-time computability seems to have been started by Ko and Friedman [13].

2.1 Representating a real number. Turing machines work on strings. The input and output must be encoded in some way. For example, we encode integers into strings by the binary notation in order to discuss computability of functions from integers to integers. We can even encode dyadic rational numbers (that is, multiples of possibly negative powers of 2) by interpreting strings (over the alphabet $\{0, 1, +, -, \cdot\}$) of form

$$s a_k \dots a_0 . a_{-1} \dots b_{-m}, \quad (2)$$

where $s \in \{+, -\}$, $a_k, \dots, a_{-m} \in \{0, 1\}$ and either $a_k = 1$ or $k = 0$, in the obvious way.

Real numbers, however, cannot be encoded into strings, because there are uncountably many of them. Instead, we use string-to-string functions to represent real numbers as the limit of a sequence of rational numbers. Let \mathbf{D}_m be the set of strings of form (2). Each $d \in \mathbf{D}_m$ thus denotes a multiple of 2^{-m} . Write $[d]$ for the closed interval of length $2 \cdot 2^{-m}$ centred at this number. We say that a real number t is *represented* by a string-to-string function φ , or that φ is a *name* of t , if for each $m \in \mathbf{N}$, we have $\varphi(0^m) \in \mathbf{D}_m$ and $t \in [\varphi(0^m)]$ (Figure 1). Thus, each $\varphi(0^m)$ gives an approximation of t to precision 2^{-m} using a dyadic rational (and we do not care about the values of φ at strings containing symbols other than 0).

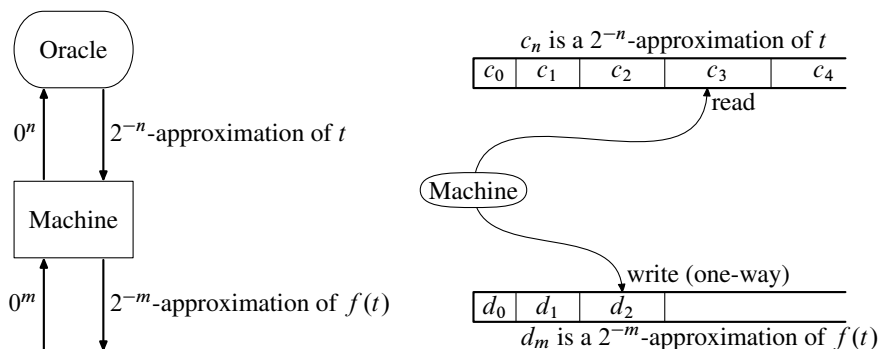


Figure 2 To compute a real function f , the machine should output an approximation of $f(t)$ to given precision m by consulting the oracle for approximations of t to any precision n it desires (left). An alternative picture (right) is that the machine converts any stream that encodes improving approximations of t to a stream that encodes improving approximations of $f(t)$. By a 2^{-n} -approximation of t , we mean a string $d \in \mathbf{D}_n$ with $t \in [d]$.

2.2 Computing a real function. Now we define how a machine works on such names φ to compute a real function. We use *oracle Turing machines* [22, Definition 9.17] (henceforth just *machines*) as the model of computation¹. In addition to the input, output and work tapes, the machine has a query tape and can consult an external oracle φ by entering a distinguished query state with a string v written on the query tape; this string is then replaced by the answer $\varphi(v)$ in one step.

Definition 1 A machine *computes* a function $f: [0, 1] \rightarrow \mathbf{R}$ if, given (any function representing) any $t \in [0, 1]$ as oracle, it computes (some function representing) $f(t)$.

Thus, computation of f can be thought of as a Turing reduction [22, Section 9.2] of $f(t)$ to t (Figure 2, left). A little thought shows that it can equivalently be visualized as a Turing machine that, given on the input tape an infinite string approximating t , writes approximations of $f(t)$ endlessly on the one-way output tape (Figure 2, right).

A machine runs in *polynomial time* if there is a polynomial $p: \mathbf{N} \rightarrow \mathbf{N}$ such that, for any input string u , it halts within $p(|u|)$ steps regardless of the oracle. In the left picture in Figure 2, this means that the machine must halt within $p(m)$ steps; in particular, it never learns t to precision more than $2^{-p(m)}$, because it takes n steps just to write down the query 0^n . In the picture on the right, the machine is required to commit to each output d_m within time $p(m)$. Analogously for *polynomial space*². Polynomial time implies polynomial space, because writing a symbol takes one step.

¹The book [22] regards an oracle as a predicate on strings, not as a string-to-string function. We chose function oracles just for simple presentation; for the discussion in this paper, we could have formulated the definitions using the predicate oracles by encoding names into predicates.

²For polynomial space, we count also the query tape in. A machine thus can only make queries polynomially long in the length of the string input. The definition in Ko's book [10, Section 7.2.1] contains a comment to the contrary, but the subsequent theorems in the chapter build on the definition that does charge the query tape.

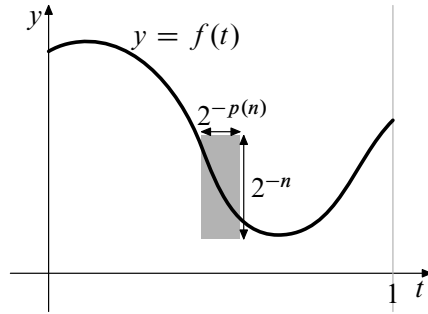


Figure 3 A function $p: \mathbf{N} \rightarrow \mathbf{N}$ is a modulus of continuity of a real function f if $|f(t_0) - f(t_1)| \leq 2^{-p(n)}$ whenever $|t_0 - t_1| \leq 2^{-n}$.

A function $f: [0, 1] \rightarrow \mathbf{R}$ is (*polynomial-time/space*) *computable* if f is computed by some machine (that runs in polynomial time/space). As mentioned above, when writing each approximation of the value, the machine knows the argument only to some finite precision (which is polynomial-time bounded in the case of polynomial-space computation). As a result, all computable functions are continuous, and all polynomial-space computable functions have polynomial *modulus of continuity* (Figure 3). In fact, it is not hard to see the following.

Theorem 2 *A function $f: [0, 1] \rightarrow \mathbf{R}$ is polynomial-time (resp. -space) computable if and only if there are a polynomial p and a polynomial-time (resp. -space) computable string-to-string function g such that for any $n \in \mathbf{N}$, $t \in [0, 1]$ and $d \in \mathbf{D}_{p(n)}$ with $t \in [d]$, we have $g(d) \in \mathbf{D}_n$ and $f(t) \in [g(d)]$.*

The above definitions of (polynomial-time/space) computability can be straightforwardly generalized to functions on any compact subset of \mathbf{R}^m (by considering the machine that takes m oracles).

We choose not to extend the definition to functions f on non-compact sets, because we are less sure that such extension leads to the “right” notion. If we extended Definition 1 in the straightforward way, the identity function on \mathbf{R} would not be polynomial-time computable, because even the length of the first approximation is unbounded. Weihrauch’s definition [26, Chapter 7] makes the identity function polynomial-time computable by using a slightly different output convention, but still precludes, say, the sine function on \mathbf{R} . Hoover [6] allows the running time to depend in a certain way on the input real number. The sine function is polynomial-time computable according to this model. The three definitions coincide when we restrict attention, as we do in this paper, to functions on compact intervals, or when we discuss computability only (and not polynomial-time/space computability).

2.3 Examples. Many familiar continuous functions are computable. Take binary addition $+$: $[0, 1] \times [0, 1] \rightarrow \mathbf{R}$ for example. Suppose that we (the machine) want to compute the sum of real numbers t_0 and t_1 given as oracles. If the input string given to us is 0^m , we are required to find a (string representing) dyadic rational $d \in \mathbf{D}_m$ with $t_0 + t_1 \in [d]$. We do this by first issuing the query 0^{m+2} to both of the oracles, thus obtaining dyadic rationals $d_0, d_1 \in \mathbf{D}_{m+2}$ with $t_0 \in [d_0]$ and $t_1 \in [d_1]$. Then we add d_0 and d_1 as dyadic rationals, and let d be the sum

rounded to the nearest point in \mathbf{D}_m . We have

$$d - (t_0 + t_1) \leq |d - (d_0 + d_1)| + |d_0 - t_0| + |d_1 - t_1| \leq \frac{1}{2^{m+1}} + \frac{1}{2^{m+2}} + \frac{1}{2^{m+2}} = \frac{1}{2^m}, \quad (3)$$

as required. Moreover, this computation can be done in time polynomial in m .

We also note that the limit of a computable sequence that converges fast enough is computable. For example, the sine function restricted to $[0, 1]$ is polynomial-time computable, because an approximation of

$$\sin t = t - \frac{t^3}{3!} + \frac{t^5}{5!} - \frac{t^7}{7!} + \dots \quad (4)$$

to precision 2^{-m} can be found by approximating the sum of the first $m + 1$ terms to precision 2^{-m-1} (which can be done in polynomial time), because the sum of the remaining terms never exceeds 2^{-m-1} . Similar argument will be used repeatedly without describing the details.

3 Lipschitz continuous initial value problems

The precise statements of (A) and (B) in Section 1 are in Ko's paper [9], where he uses previous results [16, 20] and modifies them to make g polynomial-time computable. These results suggest that we need an assumption stronger than uniqueness in order to control the complexity of the solution. A simple sufficient condition for unique solution h of (1) is that g be *Lipschitz continuous* (along its second argument), which is to say,

$$|g(t, y_0) - g(t, y_1)| \leq L \cdot |y_0 - y_1|, \quad t \in [0, 1], y_0, y_1 \in \mathbf{R} \quad (5)$$

for some constant L independent of y_0, y_1 and t [25]. In this section, we ask how complex h can be, assuming that g is Lipschitz continuous and polynomial-time computable. An upper bound was given by Ko [9, Section 4] through a careful analysis of the Euler method³.

Theorem 3 *Let $g: [0, 1] \times [-1, 1] \rightarrow \mathbf{R}$ be Lipschitz continuous and assume that $h: [0, 1] \rightarrow \mathbf{R}$ takes values in $[-1, 1]$ and satisfies (1). If g is polynomial-time computable, then h is polynomial-space computable.*

It is an unproven but widely believed conjecture in complexity theory that the class **PSPACE** of polynomial-space computable predicates (which we regard as functions from the set Σ^* of strings to $\{0, 1\}$) is properly bigger than the class **P** of polynomial-time computable predicates. Since $\mathbf{P} = \mathbf{PSPACE}$ would imply that all polynomial-space computable real functions are polynomial-time computable [9, Lemma 2.2], it would also imply by Theorem 3 that the operator solving Lipschitz continuous initial value problems preserves polynomial-time computability. To state the converse, we define what it means for a polynomial-space computable real function to be the “hardest” among problems solvable in polynomial space.

Definition 4 Let $f: [0, 1] \rightarrow \mathbf{R}$ be polynomial-space computable. We say that f is *polynomial-space complete* if there are p and g as in Theorem 2 such that any predicate $A: \Sigma^* \rightarrow \{0, 1\}$ in **PSPACE** reduces to g in the sense that $A = R \circ g \circ s$ for some predicate $R: \Sigma^* \rightarrow \{0, 1\}$ in **P** and some polynomial-time computable function $s: \Sigma^* \rightarrow \Sigma^*$ (Figure 4).

³Ko's original statement [9, Theorem 1] assumes a weaker (and more complicated) condition than Lipschitz continuity (and is thus stronger).

$$\begin{array}{ccc}
\Sigma^* & \xrightarrow{A} & \{0, 1\} \\
s \downarrow & & \uparrow R \\
\Sigma^* & \xrightarrow{g} & \Sigma^*
\end{array}$$

Figure 4 We say that A reduces to g if we can compute $A(u)$ by first applying a polynomial-time algorithm s to u to compute the input to g and then giving its output to a polynomial-time algorithm R .

It is easy to see that this definition is equivalent to Ko's [11, Definition 2.2]. Kawamura [8] proved the following in answer to Ko's question [9].

Theorem 5 *There are a Lipschitz continuous, polynomial-time computable function $g: [0, 1] \times [-1, 1] \rightarrow \mathbf{R}$ and a polynomial-space complete function $h: [0, 1] \rightarrow \mathbf{R}$ that takes values in $[-1, 1]$ and satisfies (1).*

This implies that unless $\mathbf{P} = \mathbf{PSPACE}$, solving Lipschitz continuous initial value problems does not preserve polynomial-time computability.

4 Analytic initial value problems

In this section, we use subscripts to denote the components of any vector $\mu \in \mathbf{N}^m$ or $x \in \mathbf{R}^m$: thus $\mu = (\mu_0, \dots, \mu_{m-1})$ and $x = (x_0, \dots, x_{m-1})$. For any $\mu, \nu \in \mathbf{N}^m$ and $x \in \mathbf{R}^m$, we write

$$|\mu| = \sum_{i=0}^{m-1} \mu_i, \quad \mu! = \prod_{i=0}^{m-1} \mu_i!, \quad \binom{\mu}{\nu} = \frac{\mu!}{\nu! (\mu - \nu)!}, \quad x^\mu = \prod_{i=0}^{m-1} x_i^{\mu_i}. \quad (6)$$

For $\varepsilon > 0$, write $B(x, \varepsilon)$ for the open cube $(x_0 - \varepsilon, x_0 + \varepsilon) \times \dots \times (x_{m-1} - \varepsilon, x_{m-1} + \varepsilon)$.

A function $f: D \rightarrow \mathbf{R}$ on an open set $D \subseteq \mathbf{R}^m$ is said to be *analytic* at $\hat{x} \in D$ if there are an open neighbourhood $V \subseteq D$ of \hat{x} and real numbers a_μ , one for each $\mu \in \mathbf{N}^m$, such that for all $x \in V$, the series

$$\sum_{\mu \in \mathbf{N}^m} a_\mu (x - \hat{x})^\mu \quad (7)$$

converges to $f(x)$. It is easy to see [15, Proposition 2.1.7] that if (some serial rearrangement of) the sum (7) converges, so does the sum of $|a_\mu| \cdot r^\mu$ over $\mu \in \mathbf{N}^m$ for any $r \in [0, |x_0 - \hat{x}_0|) \times \dots \times [0, |x_{m-1} - \hat{x}_{m-1}|)$. The above definition therefore makes sense regardless of the order of summation, and the maximal such open neighbourhood V equals the interior of the set of all x for which $|a_\mu| \cdot |(x - \hat{x})^\mu|$ is bounded. We call this maximal V the *domain of convergence*, and a_μ the *Taylor coefficients*, of f at \hat{x} . A real function on set $K \subseteq \mathbf{R}^m$ is said to be *analytic* if it can be extended to a function on some open set $D \supseteq K$ that is analytic at every point in D .

If g in the initial value problem (1) is analytic, then so is the solution h (which is known to be unique) by the Cauchy–Kowalewsky Theorem [15, Section 2.4]. The goal of this section is to show the following.

Theorem 6 *Let $g: [0, 1] \times [-1, 1] \rightarrow \mathbf{R}$ and $h: [0, 1] \rightarrow \mathbf{R}$ be analytic functions. Assume that h takes values in $[-1, 1]$ and satisfies (1). If g is polynomial-time computable, then so is h .*

4.1 Taylor coefficients of analytic functions. A (multi-dimensional) sequence $(x_\mu)_{\mu \in \mathbf{N}^m}$ of real numbers is said to be *polynomial-time computable* if there is a polynomial-time machine that, given n and (each component of) μ in unary notation as inputs, outputs some $d \in \mathbf{D}_n$ with $x_\mu \in [d]$. We will show in Theorem 11 that a real analytic function on a compact set $K \subseteq \mathbf{R}^m$ is polynomial-time computable if and only if the sequence of its Taylor coefficients at a rational point, or equivalently the sequence of derivatives, is. A similar fact is pointed out by Ko and Friedman [14] for $m = 1$. Generalization to $m > 1$ is not hard, but we present it here for the sake of completeness and simpler proof. We begin by showing that series (7) is easy to compute on a compact subset of the domain of convergence.

Lemma 7 *Suppose that a function $f: D \rightarrow \mathbf{R}$ on an open set $D \subseteq \mathbf{R}^m$ is analytic at $\hat{x} \in D \cap \mathbf{Q}^m$, with domain of convergence $V \subseteq D$ and Taylor coefficients $(a_\mu)_{\mu \in \mathbf{N}^m}$. If the sequence $(a_\mu)_{\mu \in \mathbf{N}^m}$ is polynomial-time computable, so is the restriction of f to any compact subset K of V .*

Proof Fix any $r \in \mathbf{R}^m$ with $\hat{x} + r \in K$. We will show that there is a compact set K_r containing both \hat{x} and $\hat{x} + r$ in its interior such that the restriction of f to K_r is polynomial-time computable. This suffices because K is compact.

Since V is open, $\hat{x} + r/(1 - \varepsilon)^2 \in V$ for some $\varepsilon \in (0, 1)$. As mentioned above, $|a_\mu| \cdot |r/(1 - \varepsilon)^2|^\mu$ is bounded, say by M . Let K_r be the hyperrectangle with vertices $(\hat{x}_0 \pm |r_0|/(1 - \varepsilon), \dots, \hat{x}_{m-1} \pm |r_{m-1}|/(1 - \varepsilon))$. Then $|a_\mu| \cdot |(x - \hat{x})^\mu| \leq M(1 - \varepsilon)^{|\mu|}$ for each $x \in K_r$. Hence, (7) converges fast: the sum (7) differs from the partial sum over $\mu_0, \dots, \mu_{m-1} < N$ by at most

$$\sum_{\mu \in \mathbf{N}^m \setminus \{0, \dots, N-1\}^m} M(1 - \varepsilon)^{|\mu|} = M \cdot \frac{1 - (1 - (1 - \varepsilon)^N)^m}{\varepsilon^m} \leq \frac{Mm}{\varepsilon^m} (1 - \varepsilon)^N, \quad (8)$$

which is bounded by 2^{-n} by making N only polynomially large in n . Thus, (7) is approximated by (approximately) adding up polynomially many terms. This can be done in time if $(a_\mu)_{\mu \in \mathbf{N}^m}$ is polynomial-time computable and \hat{x} is rational. \square

Now we consider the other direction: computing the Taylor coefficients a_μ from the values of f near \hat{x} . If f is analytic at \hat{x} , then each derivative $D^\mu f(\hat{x})$ of f at \hat{x} of order μ exists and equals $a_\mu \mu!$ [15, Remark 2.2.4]. Since $\mu!$ has length polynomial in $|\mu|$ and can be multiplied easily, $(a_\mu)_{\mu \in \mathbf{N}^m}$ is polynomial-time computable if and only if $(D^\mu f(\hat{x}))_{\mu \in \mathbf{N}^m}$ is. Therefore, we will consider how to compute the sequence of derivatives.

The following lemma allows us to approximate the k th derivative of a unary function f at \hat{x} by using the values of f at $k + 1$ points near \hat{x} .

Lemma 8 *Let $n, k, A \in \mathbf{N}$ and $B > kA2^n$. If a real function f on an open interval (u, v) is $k + 1$ times differentiable and $|D^{k+1} f(x)| \leq A$ for all $x \in (u, v)$, then for all \hat{x} with $u < \hat{x} < v - k/B$, the value*

$$B \sum_{i=0}^k (-1)^i \binom{k}{i} f\left(\hat{x} + \frac{i}{B}\right) \quad (9)$$

differs from $D^k f(\hat{x})$ by at most 2^{-n} .

Proof Fix $n, k, A, B, (u, v), f$ and \hat{x} as assumed. Consider the polynomial

$$P(X) = B \sum_{i=0}^k \frac{(-1)^i}{i! (k-i)!} f\left(\hat{x} + \frac{i}{B}\right) \prod_{j \neq i} \left(X - \hat{x} - \frac{j}{B}\right). \quad (10)$$

This P is called the *Lagrange interpolating polynomial*. It agrees with f at $k + 1$ points $\hat{x}, \hat{x} + 1/B, \dots, \hat{x} + k/B$. The Mean Value Theorem yields inductively on $j = 0, \dots, k$ that $D^j P$ agrees with $D^j f$ at (at least) $k + 1 - j$ distinct points between \hat{x} and $\hat{x} + k/B$. In particular, $D^k P(\xi) = D^k f(\xi)$ for some $\xi \in (\hat{x}, \hat{x} + k/B)$. Hence,

$$\begin{aligned} |D^k f(\hat{x}) - D^k P(\xi)| &= |D^k f(\hat{x}) - D^k f(\xi)| \\ &\leq \int_{\hat{x}}^{\xi} D^{k+1} f(X) dX \leq \int_{\hat{x}}^{\xi} A dX \leq \frac{kA}{B} < 2^{-n}. \end{aligned} \quad (11)$$

Calculating the leading coefficient in (10), we see that $D^k P(\xi)$ equals (9). \square

Observe that perturbing each $f(\hat{x} + i/B)$ by ε affects (9) by at most $\varepsilon \cdot 2^\mu \cdot B$. We use this to prove polynomial-time computability of the Taylor coefficients.

Lemma 9 *Assume that a function $f: V \rightarrow \mathbf{R}$ on an open set $V \subseteq \mathbf{R}^m$ is infinitely differentiable and that there is a polynomial $\alpha: \mathbf{N}^m \rightarrow \mathbf{N}$ such that $|D^\mu f(x)| \leq 2^{\alpha(\mu)}$ for all $\mu \in \mathbf{N}^m$ and $x \in V$. Let $K \subseteq V$ be a compact set containing a point $\hat{x} \in \mathbf{Q}^m$ in its interior. If the restriction of f to K is polynomial-time computable, so is the sequence $(D^\mu f(\hat{x}))_{\mu \in \mathbf{N}^m}$.*

Proof Given $\mu \in \mathbf{N}^m$ in unary notation, we can find in polynomial time integers A and B that are big enough to satisfy $B > |\mu|A2^n$, $B(\hat{x}, \mu/B) \subseteq K$ and $A \geq 2^{\alpha(\mu_0, \dots, \mu_{i-1}, \mu_i+1, 0, \dots, 0)}$ for all $i = 0, \dots, m-1$, but yet $B = 2^{n+\beta(\mu)-|\mu|-1}$ for some polynomial β . For each i , Lemma 8 implies that $D^{(\mu_0, \dots, \mu_{i-1}, \mu_i, 0, \dots, 0)} f(x)$ can be approximated to precision $1/2^{n-1}$ in time polynomial in μ and n if we are given approximations of $D^{(\mu_0, \dots, \mu_{i-1}, 0, 0, \dots, 0)} f$ at certain $\mu_i + 1$ points near x to precision $1/(2^n \cdot 2^{\mu_i} \cdot B) \geq 1/2^{2n+\beta(\mu)-1}$. Repeating this inductively for $i = 0, \dots, m-1$, we can approximate $D^\mu f(\hat{x})$ to precision 2^{-n} in polynomial time using approximations of f at certain $(\mu_0+1) \cdots (\mu_{m-1}+1)$ points near \hat{x} to precision $1/2^{2^{m(n+\beta(\mu))-\beta(\mu)-1}}$. \square

If f is analytic at \hat{x} , then it can be shown [15, Proposition 2.2.10] that there are an open neighbourhood V of \hat{x} and constants C and R such that

$$\frac{|D^\mu f(x)|}{\mu!} \leq \frac{C}{R^{|\mu|}}, \quad x \in V, \mu \in \mathbf{N}^m. \quad (12)$$

Thus f (restricted to some small enough V) meets the requirement of Lemma 9.

We prove the converse by extending the local result we saw in Lemma 7.

Lemma 10 *Let $f: K \rightarrow \mathbf{R}$ be an analytic function on a connected compact set $K \subseteq \mathbf{R}^m$ containing $\hat{x} \in \mathbf{Q}^m$. If the sequence $(D^\mu f(\hat{x}))_{\mu \in \mathbf{N}^m}$ is polynomial-time computable, so is f .*

Proof Let \bar{f} be an analytic extension of f to an open set $D \supseteq K$. For each $x \in D$, let $\varepsilon_x > 0$ be so small that $\overline{B(x, 3\varepsilon_x)} \subseteq V_x$, where V_x is the domain of convergence of \bar{f} at x . It is easy to see that $\overline{B(x, \varepsilon_x)} \subseteq V_r$ for every $r \in B(x, \varepsilon_x)$ by the remark following (7).

Let $y \in K$. Since K is connected and compact, there are $x_0, \dots, x_p \in D$ such that $\hat{x} = x_0$, $y \in B(x_p, \varepsilon_{x_p})$ and $B(x_{i-1}, \varepsilon_{x_{i-1}})$ intersects $B(x_i, \varepsilon_{x_i})$ for each $i = 1, \dots, p$. Let $r_i \in B(x_{i-1}, \varepsilon_{x_{i-1}}) \cap B(x_i, \varepsilon_{x_i}) \cap \mathbf{Q}^m$. By the above paragraph, $r_1 \in V_{\hat{x}}$, $r_2 \in V_{r_1}$, \dots , $r_p \in V_{r_{p-1}}$, $y \in V_{r_p}$. The conclusion follows inductively by using Lemmata 7 and 9 at each step. \square

Theorem 11 *Let $f: K \rightarrow \mathbf{R}$ be an analytic function on a compact connected set $K \subseteq \mathbf{R}^m$ containing $\hat{x} \in \mathbf{Q}^m$. Then f is polynomial-time computable if and only if the sequence of Taylor coefficients of f at \hat{x} is.*

Proof By Lemmata 9 and 10. \square

4.2 Solution by power series. Now we return to Theorem 6. Let g and h be as assumed there. There are sequences $(a_{i,j})_{(i,j) \in \mathbf{N}^2}$ and $(b_k)_{k \in \mathbf{N}}$ such that

$$g(t, y) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} a_{i,j} \cdot t^i \cdot y^j, \quad h(t) = \sum_{i=0}^{\infty} b_i \cdot t^i \quad (13)$$

for all t and y sufficiently close to the origin. In view of Theorem 11, it suffices to prove that if $(a_{i,j})_{(i,j) \in \mathbf{N}^2}$ is polynomial-time computable, then so is $(b_k)_{k \in \mathbf{N}}$.

Substituting (13) into (1), we get $b_0 = 0$ and

$$\sum_{k=0}^{\infty} (k+1) \cdot b_{k+1} \cdot t^k = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} a_{i,j} \cdot t^i \cdot \left(\sum_{k=0}^{\infty} b_k \cdot t^k \right)^j = \sum_{k=0}^{\infty} \sum_{i=0}^k \sum_{j=0}^{k-i} a_{i,j} \cdot B_{k-i,j} \cdot t^k, \quad (14)$$

where we put

$$B_{s,j} = \sum_{\substack{(k_1, \dots, k_j) \in \mathbf{N}^j \\ k_1 + \dots + k_j = s}} b_{k_1} \cdots b_{k_j} = \begin{cases} 1 & \text{if } j = 0 \text{ and } s = 0, \\ 0 & \text{if } j = 0 \text{ and } s > 0, \\ \sum_{k=0}^s b_k \cdot B_{s-k, j-1} & \text{if } j > 0 \end{cases} \quad (15)$$

for integers $j \geq 0$ and $s \geq j$. Comparing the coefficients of t^k in (14), we get

$$b_{k+1} = \frac{1}{k+1} \cdot \sum_{i=0}^k \sum_{j=0}^{k-i} a_{i,j} \cdot B_{k-i,j}. \quad (16)$$

Note that (15) and (16) give a mutual recurrence defining $B_{s,j}$ and b_j in the order

$$b_0, B_{0,0}, b_1, B_{1,0}, B_{1,1}, b_2, B_{2,0}, B_{2,1}, B_{2,2}, b_3, B_{3,0}, \dots \quad (17)$$

In fact, they allow us to compute easily the r th term of this list with precision $2^{p(r)} \cdot \varepsilon$, for some polynomial p , if all the preceding $r-1$ terms are known to within $\varepsilon > 0$, since $|a_{i,j}|$ and $|b_k|$ are bounded exponentially in their subscripts by (12). Using this inductively, we obtain the approximation of the r th term to precision 2^{-m} in polynomial time in $r+m$ by computing the r' th term, for $r' = 1, \dots, r$, to precision $2^{-m-p(r)-p(r-1)-\dots-p(r'+1)}$.

5 Some reflections on the model

There are at least two other veins of research that, while completely different in philosophy from ours, discuss also “computability” and “complexity” of real functions (that are incomparable to our notions).

One is from the study of *analog models* [2, 21], where changes of physical quantities in the continuous world are interpreted as computation performed by nature or by a device built for the purpose. Naturally, many of these models involve differential equations representing (classical) physical laws. Relation between such analog models and the traditional notion of computation is discussed by several authors [19, 24, 29]. The results mentioned in the present paper may provide further links. For example, it is routine to modify our Theorem 6 to prove that the real primitive recursive functions defined in Moore’s paper [17] (and refined

by Kawamura [7]) are polynomial-time computable on any compact subset of their domains.

The other approach is the algebraic model, such as the one proposed by Blum et al. [1]. This is somewhat closer to our approach in that the machine works with discrete clock ticks. The difference is that they use a modified version of Turing machines that store real numbers and perform operations on them in one step, whereas our bit model, as we explained in Section 2, applies the traditional Turing machine as it is to continuous settings through oracles and representations. The algebraic model may be close to the intuitive picture that numerical analysts have in mind when they write pseudocodes for their algorithms. Since real numbers cannot be stored in its entirety in reality, the reliability of the algorithm when it is implemented on digital computers must be verified outside the model. The theory of information-based complexity [23, 27] uses this algebraic model to discuss complexity of numerical problems. The difference and connection between such algebraic models and the “bit model” that was used in this paper are studied by several authors [3, 4, 26, 28].

As we saw in Section 2, our model takes seriously the fact that digital computers work on bits and thus can only specify a real number to a certain precision. Moreover, our definition of computing a real function requires that the machine yield the value with guaranteed precision. Such a strict notion of computation has not been popular in the past among numerical analysts, because algorithms with guaranteed error bounds often requires considerably more time than standard ones. Recently, however, there is increasing interest in validated methods [18] with a view to shifting the burden of determining the reliability of a numerical solution from the user to the computer.

But there is still a gap between the practice and our formulation: the results (A)–(D) that we saw in this paper address the complexity of the function h itself, rather than the difficulty of “solving the equation” by “computing h from g ,” a task apparently closer to what we would want to do in practice. Such study would require a nice definition of complexity of operators taking real functions to real functions. There is a canonical way to induce computability over the space of continuous functions [26, Chapter 6], and the positive result in (B) can be “effectivized” to the statement that the operator that solves initial value problems with a unique solution is computable in this induced sense. But this argument does not easily apply to polynomial-time/space computability.

The negative results still have some implications on numerical computation, because it seems safe to assume that an “efficiently computable” operator, whatever it should mean, cannot take a function that is polynomial-time computable to a function that is not. Thus, the negative result in (C) suggests that no numerical algorithm can solve the initial value problem efficiently (with validated error bound) for all Lipschitz continuous input function g , unless $\mathbf{P} = \mathbf{PSPACE}$.

Acknowledgement. (To be added later)

References

- [1] BLUM, L., CUCKER, F., SHUB, M., AND SMALE, S. *Complexity and Real Computation*. Springer, 1997.
- [2] BOURNEZ, O., AND CAMPAGNOLO, M. L. A survey on continuous time computations. In *New Computational Paradigms*. Springer, 2008, pp. 383–423.

- [3] BRATTKA, V., AND HERTLING, P. Feasible real random access machines. *Journal of Complexity* 14 (1998), 490–526.
- [4] BRAVERMAN, M. On the complexity of real functions. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science* (October 2005), pp. 155–164.
- [5] GRZEGORCZYK, A. Computable functionals. *Fundamenta Mathematicae* 42 (1955), 168–202.
- [6] HOOVER, H. J. Feasible real functions and arithmetic circuits. *SIAM Journal on Computing* 19, 1 (1990), 182–204.
- [7] KAWAMURA, A. Differential recursion. *ACM Transactions on Computational Logic* 10, 3 (2009), 22:1–22.
- [8] KAWAMURA, A. Lipschitz continuous ordinary differential equations are polynomial-space complete. In *Proceedings of the 24th IEEE Conference of Computational Complexity* (July 2009).
- [9] KO, K. On the computational complexity of ordinary differential equations. *Information and Control* 58 (1983), 157–194.
- [10] KO, K. *Computational Complexity of Real Functions*. Birkhäuser Boston, 1991.
- [11] KO, K. On the computational complexity of integral equations. *Annals of Pure and Applied Logic* 58, 3 (November 1992), 201–228.
- [12] KO, K. Polynomial-time computability in analysis. In *Handbook of Recursive Mathematics: Volume 2: Recursive Algebra, Analysis and Combinatorics*, I. L. Ershov et al., Eds., vol. 139 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, December 1998, pp. 1271–1317.
- [13] KO, K., AND FRIEDMAN, H. Computational complexity of real functions. *Theoretical Computer Science* 20, 3 (1982), 323–352.
- [14] KO, K., AND FRIEDMAN, H. Computing power series in polynomial time. *Advances in Applied Mathematics* 9 (1988), 40–50.
- [15] KRANTZ, S. G., AND PARKS, H. R. *A Primer of Real Analytic Functions*, second ed. Birkhäuser Advanced Texts. Birkhäuser Boston, June 2002.
- [16] MILLER, W. Recursive function theory and numerical analysis. *Journal of Computer and System Sciences* 4 (1970), 465–472.
- [17] MOORE, C. Recursion theory on the reals and continuous-time computation. *Theoretical Computer Science* 162 (1996), 23–44.
- [18] NEDIALKOV, N. S., JACKSON, K. R., AND CORLISS, G. F. Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation* 105 (1999), 21–68.
- [19] PENROSE, R. *The Emperor’s New Mind: Concerning Computers, Minds, and the Laws of Physics*. Oxford University Press, Inc., New York, NY, USA, 1989.
- [20] POUR-EL, M. B., AND RICHARDS, I. A computable ordinary differential equation which possesses no computable solution. *Annals of Mathematical Logic* 17, 1–2 (1979), 61–90.
- [21] SHANNON, C. E. Mathematical theory of the Differential Analyzer. *Journal of Mathematics and Physics* 20, 4 (1941), 337–354.
- [22] SIPSER, M. *Introduction to the Theory of Computation*, second ed. Course Technology, 2005.
- [23] TRAUB, J. F., WASILKOWSKI, G. W., AND WOŹNIAKOWSKI, H. *Information-Based Complexity*. Academic Press, 1988.
- [24] VERGIS, A., STEIGLITZ, K., AND DICKINSON, B. The complexity of analog

- computation. *Mathematics and Computers in Simulation* 28, 2 (1986), 91–113.
- [25] WALTER, W. *Ordinary Differential Equations*, vol. 182 of *Graduate Texts in Mathematics, Readings in Mathematics*. Springer, New York, 1998. Translated by Russell Thompson from the sixth edition of *Gewöhnliche Differentialgleichungen*, 1996.
- [26] WEIHRAUCH, K. *Computable Analysis: An Introduction*. Texts in Theoretical Computer Science. Springer, 2000.
- [27] WERSCHULZ, A. G. *The Computational Complexity of Differential and Integral Equations: An Information-Based Approach*. Oxford University Press, 1991.
- [28] WOŹNIAKOWSKI, H. Why does information-based complexity use the real number model. *Theoretical Computer Science* 219 (1999), 451–465.
- [29] YAO, A. C. Classical physics and the Church–Turing Thesis. *Journal of the Association for Computing Machinery* 50, 1 (2003), 100–105.