Logical Methods in Computer Science
Vol. 10(1:6)2014, pp. 1–15
www.lmcs-online.org

Submitted Jan. 1, 2013
Published Feb. 11, 2014

COMPUTATIONAL COMPLEXITY OF SMOOTH DIFFERENTIAL EQUATIONS *

AKITOSHI KAWAMURA, HIROYUKI OTA, CARSTEN RÖSNICK, AND MARTIN ZIEGLER

Abstract. The computational complexity of the solution h to the ordinary differential equation $h(0) = 0$, $h'(t) = g(t, h(t))$ under various assumptions on the function g has been investigated. Kawamura showed in 2010 that the solution h can be PSPACE-hard even if g is assumed to be Lipschitz continuous and polynomial-time computable. We place further requirements on the smoothness of g and obtain the following results: the solution h can still be PSPACE-hard if g is assumed to be of class C^1; for each $k \geq 2$, the solution h can be hard for the counting hierarchy even if g is of class C^k.

1. Introduction

Let $g\colon [0,1] \times \mathbf{R} \to \mathbf{R}$ be continuous and consider the differential equation

$$h(0) = 0, \qquad Dh(t) = g(t, h(t)) \quad t \in [0,1], \tag{1.1}$$

where Dh denotes the derivative of h. How complex can the solution h be, assuming that g is polynomial-time computable? Here, polynomial-time computability and other notions of complexity are from the field of *Computable Analysis* [10, 20] and measure how hard it is to approximate real functions with specified precision (Section 3).

If we make no assumption on g other than being polynomial-time computable, the solution h (which is not unique in general) can be non-computable. Table 1 summarizes known results about the complexity of h under various assumptions (that get stronger as we go down the table). In particular, as the third row says, if g is (globally) Lipschitz continuous, then the (unique) solution h is known to be polynomial-space computable but still can be PSPACE-hard [4]. In this paper, we study the complexity of h when we put stronger assumptions about the smoothness of g.

In numerical analysis, knowledge about smoothness of the input function (such as being differentiable enough times) often helps to apply certain algorithms or simplify their analysis. However, to our knowledge, this casual understanding that smoothness is good has not been rigorously substantiated in terms of computational complexity theory. This motivates us to

2012 ACM CCS: [**Theory of computation**]: Computational complexity and cryptography—Problems, reductions and completeness; [**Mathematics of computing**]: Mathematical analysis—Numerical analysis—Interval arithmetic; Mathematical analysis—Differential equations—Ordinary differential equations.

Key words and phrases: computable analysis, counting hierarchy, differential equations.

* Some of the results in this paper have been reported at conferences [21, 8]. This work was supported in part by: *Kakenhi* (Grants-in-Aid for Scientific Research, Japan) 23700009 and 24106002; *Marie Curie International Research Staff Exchange Scheme Fellowship* 294962, the 7th European Community Framework Programme; and *German Research Foundation* with project DFG Zi 1009/4-1.

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.2168/LMCS-10(1:6)2014

Table 1: Complexity of the solution h of (1.1) assuming g is polynomial-time computable

Assumptions	Upper bounds	Lower bounds
—	—	can be all non-computable [15]
h is the unique solution	computable [2]	can take arbitrarily long time [9, 12]
the Lipschitz condition	polynomial-space [9]	can be PSPACE-hard [4]
g is of class $\mathrm{C}^{(\infty,1)}$	polynomial-space	can be PSPACE-hard (Theorem 1)
g is of class $\mathrm{C}^{(\infty,k)}$ (for each constant k)	polynomial-space	can be CH-hard (Theorem 2)
g is analytic	polynomial-time [13, 11, 3]	—

ask whether, for our differential equation (1.1), smoothness really reduces the complexity of the solution.

At one extreme is the case where g is analytic: h is then polynomial-time computable (the last row of the table) by an argument based on Taylor series[1] (this does not necessarily mean that computing the values of h from those of g is easy; see the last paragraph of Section 5.2). Thus our interest is in the cases between Lipschitz and analytic (the fourth and fifth rows). We say that g is of class $\mathrm{C}^{(i,j)}$ if the partial derivative $D^{(n,m)}g$ (often also denoted $\partial^{n+m}g(t,y)/\partial t^n \partial y^m$) exists and is continuous for all $n \le i$ and $m \le j$;[2] it is said to be of class $\mathrm{C}^{(\infty,j)}$ if it is of class $\mathrm{C}^{(i,j)}$ for all $i \in \mathbf{N}$.

We will show that adding continuous differentiability does not break the PSPACE-completeness that we knew from [4] for the Lipschitz continuous case:

Theorem 1. There exists a polynomial-time computable function $g\colon [0,1]\times[-1,1]\to\mathbf{R}$ of class $\mathrm{C}^{(\infty,1)}$ such that the equation (1.1) has a PSPACE-hard solution $h\colon [0,1]\to\mathbf{R}$.

The complexity notions (computability and hardness) in this and the following theorems will be explained in Section 3. When g is more than once differentiable, we did not quite succeed in proving that h is PSPACE-hard in the same sense, but we will prove it CH-hard, where $\mathsf{CH}\subseteq\mathsf{PSPACE}$ is the counting hierarchy (see Section 2):

Theorem 2. Let k be a positive integer. There is a polynomial-time computable function $g\colon [0,1]\times[-1,1]\to\mathbf{R}$ of class $\mathrm{C}^{(\infty,k)}$ such that the equation (1.1) has a CH-hard solution $h\colon [0,1]\to\mathbf{R}$.

In Theorems 1 and 2, we said $g\colon [0,1]\times[-1,1]\to\mathbf{R}$ instead of $g\colon [0,1]\times\mathbf{R}\to\mathbf{R}$, because the notion of polynomial-time computability of real functions in this paper is defined only when the domain is a bounded closed region.[3] This makes the equation (1.1) ill-defined

[1] As shown by Müller [13] and Ko and Friedman [11], polynomial-time computability of an analytic function on a compact interval is equivalent to that of its Taylor sequence at a point (although the latter is a local property, polynomial-time computability on the whole interval is implied by analytic continuation; see [13, Corollary 4.5] or [3, Theorem 11]). This implies the polynomial-time computability of h, since we can efficiently compute the Taylor sequence of h from that of g.

[2] Another common terminology (which we used in the abstract) is to say that g is of class C^k if it is of class $\mathrm{C}^{(i,j)}$ for all i, j with $i+j\le k$.

[3] Although we could extend our definition to functions with unbounded domain [6, Section 4.1], the results in Table 1 do not hold as they are, because polynomial-time compubable functions g, such as $g(t,y)=y+1$,

in case h ever takes a value outside $[-1,1]$. By saying that h is a solution in Theorem 1, we are also claiming that $h(t) \in [-1,1]$ for all $t \in [0,1]$. This is no essential restriction, because any pair of functions $g\colon [0,1] \times \mathbf{R} \to \mathbf{R}$ and $h\colon [0,1] \to \mathbf{R}$ satisfying the equation could be scaled down in an appropriate way (without affecting the computational complexity) to make h stay in $[-1,1]$. In any case, since we are making stronger assumptions on g than Lipschitz continuity, the solution h, if it exists, is unique.

Whether smoothness of the input function reduces the complexity of the output has been studied for operators other than solving differential equations, and the following negative results are known. The integral of a polynomial-time computable real function can be #P-hard, and this does not change by restricting the input to C^∞ (infinitely differentiable) functions [10, Theorem 5.33]. Similarly, the function obtained by maximization from a polynomial-time computable real function can be NP-hard, and this is still so even if the input function is restricted to C^∞ [10, Theorem 3.7][4]. (Restricting to analytic inputs renders the output polynomial-time computable, again by the argument based on Taylor series.) In contrast, for the differential equation we only have Theorem 2 for each k, and do not have any hardness result when g is assumed to be infinitely differentiable.

Theorems 1 and 2 are about the complexity of each solution h. We will also discuss the complexity of the final value $h(1)$ and the complexity of the operator that maps g to h; see Sections 5.1 and 5.2.

Notation. Let $\mathbf{N}$, $\mathbf{Z}$, $\mathbf{Q}$, $\mathbf{R}$ denote the set of natural numbers, integers, rational numbers and real numbers, respectively.

We assume that any polynomial is increasing, since it does not change the meaning of polynomial-time computable or polynomial-space computable.

Let A and B be bounded closed intervals in $\mathbf{R}$. We write $|f| = \sup_{x\in A} f(x)$ for $f\colon A \to \mathbf{R}$. A function $f\colon A \to \mathbf{R}$ is of *class* C^i (or *i-times continuously differentiable*) if all the derivatives $Df, D^2f, \ldots, D^if$ exist and are continuous.

For a function g of two variables, we write D_1g and D_2g for the derivatives of g with respect to the first and the second variable, respectively, when they exist. A function $g\colon A \times B \to \mathbf{R}$ is of *class* $\mathrm{C}^{(i,j)}$ if for each $m \in \{0,\ldots,i\}$ and $n \in \{0,\ldots j\}$, the derivative $D_1^m D_2^n g$ exists and is continuous. This derivative $D_1^m D_2^n g$ is then written $D^{(m,n)}g$ (and is known to equal $D_{e_1} \ldots D_{e_{m+n}} g$ for any sequence $e_1 \ldots e_{m+n}$ of m 1s and n 2s). A function g is of *class* $\mathrm{C}^{(\infty,j)}$ if it is of class $\mathrm{C}^{(i,j)}$ for all $i \in \mathbf{N}$.

2. The Counting Hierarchy

We assume that the reader is familiar with the basics of complexity theory, such as the classes P and PSPACE and the notions of reduction and hardness [19]. We briefly review the class CH.

---

could yield functions h, such as $h(t) = \exp t - 1$, that grow too fast to be polynomial-time (or even polynomial-space) computable. Bournez, Graça and Pouly [1, Theorem 2] report that the statement about the analytic case holds true if we restrict the growth of h appropriately.

[4]In the last part of the proof of this fact in the book [10, Theorem 3.7], the definition of f needs to be replaced by, e.g.,

$$f(x) = \begin{cases} u_s & \text{if not } R(s,t), \\ u_s + 2^{-(p(n)+2n+1)\cdot n} \cdot h_1(2^{p(n)+2n+1}(x - y_{s,t})) & \text{if } R(s,t). \end{cases}$$

The counting hierarchy CH is defined analogously to the polynomial-time hierarchy PH [19, Kapitel 10.4] using oracle machines corresponding to the class PP [19, Kapitel 3.3] instead of NP: thus, $\mathsf{CH} = \bigcup_{n\in\mathbf{N}} \mathsf{C}_n\mathsf{P}$, where each level $\mathsf{C}_n\mathsf{P}$ is defined inductively by $\mathsf{C}_0\mathsf{P} = \mathsf{P}$ and $\mathsf{C}_{n+1}\mathsf{P} = \mathsf{PP}^{\mathsf{C}_n\mathsf{P}}$. We leave the details of this definition to the original papers [18, 14, 17]. All we need for our purpose is the fact (Lemma 3 below) that $\mathsf{C}_n\mathsf{P}$ is characterized by the following complete problem $\mathsf{C}_nB_{\mathrm{be}}$: given n lists $X_1, \ldots, X_n$ of propositional variables, a propositional formula $\phi(X_1, \ldots, X_n)$ with all free variables listed, and numbers $m_1, \ldots, m_n$ in binary, determine whether

$$\mathsf{C}^{m_n} X_n \ldots \mathsf{C}^{m_1} X_1 \phi(X_1, \ldots, X_n) \tag{2.1}$$

is true. Here, C^m is the *counting quantifier*: for every formula $\phi(X)$ with the list X of l free variables, we write

$$\mathsf{C}^m X \phi(X) \iff \sum_{x\in\{0,1\}^l} \phi(x) \geq m, \tag{2.2}$$

where formula ϕ is identified with the function $\phi\colon \{0,1\}^l \to \{0,1\}$ such that $\phi(X) = 1$ when $\phi(X)$ is true.

Lemma 3 ([18, Theorem 7])**.** For every $n \geq 1$, the problem $\mathsf{C}_nB_{\mathrm{be}}$ is $\mathsf{C}_n\mathsf{P}$-complete.

By this, we mean that any problem A in $\mathsf{C}_n\mathsf{P}$ reduces to $\mathsf{C}_nB_{\mathrm{be}}$ via some polynomial-time function F in the sense that $v \in A$ if and only if $F(v) \in \mathsf{C}_nB_{\mathrm{be}}$.

Note that this is analogous to the polynomial hierarchy PH, whose nth level Σ^p_n has the complete problem that asks for the value of a formula of the form $\exists X_n \forall X_{n-1} \exists X_{n-2} \ldots \phi(X_1, \ldots, X_n)$ with n alternations of universal and existential quentifiers. Note also that the counting quantifier generalizes the universal and existential quentifiers, and hence $\mathsf{PH} \subseteq \mathsf{CH}$ – in fact, it is known [16] that PH is contained already in P^{PP} and thus in the second level of the counting hierarchy.

3. Computational Complexity of Real Functions

We review some complexity notions from Computable Analysis [10, 20]. We start by fixing an encoding of real numbers by string functions.

Definition 4. A function $\phi\colon \{0,1\}^* \to \{0,1\}^*$ is a *name* of a real number x if for all $n \in \mathbf{N}$, $\phi(0^n)$ is the binary representation of $\lfloor x \cdot 2^n \rfloor$ or $\lceil x \cdot 2^n \rceil$, where $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ mean rounding down and up to the nearest integer.

In effect, a name of a real number x takes each string 0^n to an approximation of x with precision 2^{-n}.

We use *oracle Turing machines* (henceforth just *machines*) to work on these names (Fig. 1). Let M be a machine and ϕ be a function from strings to strings. We write $M^\phi(0^n)$ for the output string when M is given ϕ as oracle and string 0^n as input. Thus we also regard M^ϕ as a function from strings to strings.

Definition 5. Let A be a bounded closed interval of $\mathbf{R}$. A machine M *computes* a real function $f\colon A \to \mathbf{R}$ if for any $x \in A$ and any name ϕ_x of it, M^{ϕ_x} is a name of $f(x)$.

Computation of a function $f\colon A \to \mathbf{R}$ on a two-dimensional bounded closed region $A \subseteq \mathbf{R}^2$ is defined similarly using machines with two oracles.

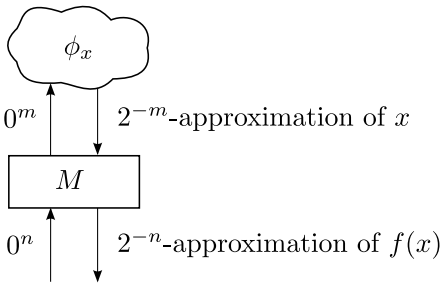

Figure 1: A machine M computing a real function f

A real function is (*polynomial-time*) *computable* if there exists some machine that computes it (in polynomial time). Polynomial-time computability of a real function f means that for any $n \in \mathbf{N}$, an approximation of $f(x)$ with error bound 2^{-n} is computable in time polynomial in n independent of the real number x.

By the time the machine outputs the approximation of $f(x)$ of precision 2^{-n}, it knows x only with some precision 2^{-m}. This implies that all computable real functions are continuous. If the machine runs in polynomial time, this m is bounded polynomially in n. This implies (3.2) in the following lemma, which characterizes polynomial-time real functions by the usual polynomial-time computability of string functions without using oracle machines.

Lemma 6. A real function is polynomial-time computable if and only if there exist a polynomial-time computable function $\phi\colon (\mathbf{Q}\cap[0,1])\times\{0,1\}^* \to \mathbf{Q}$ and polynomial $p\colon \mathbf{N}\to\mathbf{N}$ such that for all $d \in \mathbf{Q}\cap[0,1]$ and $n \in \mathbf{N}$,

$$|\phi(d, 0^n) - f(d)| \le 2^{-n}, \tag{3.1}$$

and for all $x, y \in [0,1]$, $n \in \mathbf{N}$,

$$|x - y| \le 2^{-p(n)} \Rightarrow |f(x) - f(y)| \le 2^{-n}, \tag{3.2}$$

where each rational number is written as a fraction whose numerator and denominator are integers in binary.

To speak about hardness of a real function, we define the notion of a language to it. A language $L \subseteq \{0,1\}^*$ is identified with the function $L\colon \{0,1\}^* \to \{0,1\}$ such that $L(u) = 1$ when $u \in L$.

Definition 7. A language L *reduces* to a function $f\colon [0,1] \to \mathbf{R}$ if there exist a polynomial-time function S and a polynomial-time oracle Turing machine M (Fig. 2) such that for any string u,

(i) $S(u,\cdot)$ is a name of a real number x_u, and
(ii) for any name ϕ of $f(x_u)$, we have that $M^\phi(u)$ accepts if and only if $u \in L$.

This reduction may superficially look stronger (and hence the reducibility weaker) than the one in Kawamura [4] in that M can make multiple queries adaptively, but it is not hard to see that this makes no difference.

For a complexity class C, a function f is *C-hard* if all languages in C reduce to f.

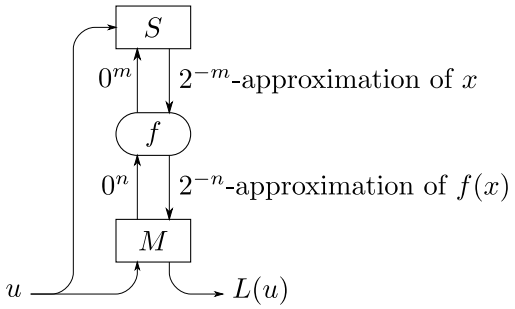

Figure 2: Reduction from a language L to a function $f\colon [0,1] \to \mathbf{R}$

4. Proof of the Main Theorems

The proofs of Theorems 1 and 2 proceed as follows. In Section 4.1, we define *difference equations*, a discrete version of the differential equations. We show that these equations with certain restrictions are PSPACE- and CH-hard. In Section 4.2, we show that these classes of difference equations can be simulated by families of differential equations satisfying certain uniform bounds on higher-order derivatives. In Section 4.3, we prove Theorems 1 and 2 by putting these families of functions together to obtain one differential equation with the desired smoothness ($\mathrm{C}^{(\infty,1)}$ and $\mathrm{C}^{(\infty,k)}$).

The idea of simulating a discrete system with limited feedback by differential equations was essentially already present in the proof of the Lipschitz version of these results [4]. We look more closely at this limitation on feedback, and express it as a restriction on the *height* of the difference equation. We show that a stronger height restriction allows the difference equation to be simulated by smoother differential equations (see the proof of Lemma 10 and the discussion after it), leading to the CH-hardness for $\mathrm{C}^{(\infty,k)}$ functions.

4.1. Difference equations.

In this section, we define difference equations, which are a discrete version of differential equations. Then we show the PSPACE- and CH-hardness of families of difference equations with different height restrictions.

Let $[n]$ denote $\{0,\dots,n-1\}$. Let $G\colon [P]\times[Q]\times[R] \to \{-1,0,1\}$ and $H\colon [P+1]\times[Q+1] \to [R]$. We say that H is the solution of the *difference equation* given by G if for all $i \in [P]$ and $T \in [Q]$ (Fig. 3),

$$H(i,0) = H(0,T) = 0, \tag{4.1}$$

$$H(i+1,T+1) - H(i+1,T) = G(i,T,H(i,T)). \tag{4.2}$$

We call P, Q and R the *height*, *width* and *cell size* of the difference equation. The equations (4.1) and (4.2) are similar to the initial condition $h(0) = 0$ and the equation $Dh(t) = g(t,h(t))$ in (1.1), respectively. In Section 4.2, we will use this similarity to simulate difference equations by differential equations.

We view a family of difference equations as a computing system by regarding the value of the bottom right cell (the gray cell in Fig. 3) as the output. A family $(G_u)_u$ of functions $G_u\colon [P_u]\times[Q_u]\times[R_u] \to \{-1,0,1\}$ *recognizes* a language L if for each u, the difference equation given by G_u has a solution H_u and $H_u(P_u,Q_u) = L(u)$. A family $(G_u)_u$ is *uniform* if the height, width and cell size of G_u are polynomial-time computable from u (in particular, they must be bounded by $2^{p(|u|)}$, for some polynomial p) and $G_u(i,T,Y)$ is polynomial-time

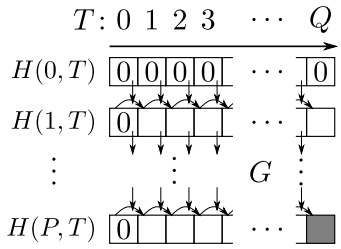

Figure 3: The solution H of the difference equation given by G

computable from (u, i, T, Y). A family $(G_u)_u$ has *polynomial height* (and *logarithmic height*, respectively) if the height P_u is bounded by $|u|^{\mathrm{O}(1)}$ (and by $\mathrm{O}(\log |u|)$, respectively). With this terminology, the key lemma in [4, Lemma 4.7] can be written as follows:

Lemma 8. There exists a PSPACE-hard language L that is recognized by some uniform family of functions with polynomial height[5].

Kawamura [4] obtained the hardness result in the third row of Table 1 by simulating the difference equations of Lemma 8 by Lipschitz continuous differential equations. Likewise, Theorem 1 follows from Lemma 8 by a modified construction that keeps the function in class $\mathrm{C}^{(\infty,1)}$ (Sections 4.2 and 4.3).

We show further that difference equations restricted to logarithmic height can be simulated by $\mathrm{C}^{(\infty,k)}$ functions for each k (Sections 4.2 and 4.3). Theorem 2 follows from this simulation and the following lemma.

Lemma 9. There exists a CH-hard language L that is recognized by some uniform family of functions with logarithmic height.

Proof. We define the desired language L by

$$\langle 0^{2^n}, w\rangle \in L \iff w \in \mathsf{C}_n B_{\mathrm{be}}, \tag{4.3}$$

using the $\mathsf{C}_n\mathsf{P}$-hard language $\mathsf{C}_n B_{\mathrm{be}}$ from Lemma 3. Then L is obviously CH-hard.

We construct a logarithmic-height uniform function family $(G_u)_u$ recognizing L. We will describe how to construct G_u for a string u of the form $\langle 0^{2^n}, \langle \phi(X_1, \ldots, X_n), m_1, \ldots, m_n\rangle\rangle$, where ϕ is a formula, and X_1, …, X_n are lists of variables with lengths m_1, …, m_n, respectively.

We write $l_i = |X_i|$ and $s_i = \sum_{j=1}^{i}(l_j + 1)$. For each $i \in \{0, \ldots, n\}$ and $x_{i+1} \in \{0,1\}^{l_{i+1}}, \ldots, x_n \in \{0,1\}^{l_n}$, we write $\phi_i(x_{i+1}, \ldots, x_n)$ for the truth value (1 or 0) of $\mathsf{C}^{m_i} X_i \ldots \mathsf{C}^{m_1} X_1 \phi(X_1, \ldots, X_i, x_{i+1}, \ldots, x_n)$. Note that $\phi_0(x_1, \ldots, x_n) = \phi(x_1, \ldots, x_n)$, $\phi_n() = L(u)$, and the relation between ϕ_{i-1} and ϕ_i is given by

$$\phi_i(x_{i+1}, \ldots, x_n) = C^{m_i}\Big(\sum_{x_i \in \{0,1\}^{l_i}} \phi_{i-1}(x_i, x_{i+1}, \ldots, x_n)\Big), \tag{4.4}$$

where we define $C^m \colon \mathbf{N} \to \{0,1\}$ by

$$C^m(k) = \begin{cases} 1 & \text{if } k \geq m, \\ 0 & \text{if } k < m. \end{cases} \tag{4.5}$$

[5]In fact, this language L is in PSPACE, because a uniform family with polynomial height can be simulated in polynomial space.

For $T \in \mathbf{N}$, we write T_i for the ith digit of T written in binary (T_0 being the least significant digit), and $T_{[i,j]}$ for the string $T_{j-1}T_{j-2}\cdots T_{i+1}T_i$.

For each $(i,T,Y) \in [n+1] \times [2^{s_n}+1] \times [2^{|u|}]$, we define $G_u(i,T,Y)$ as follows. The first row is given by

$$G_u(0,T,Y) = (-1)^{T_{s_1}}\phi(T_{[1,s_1]}, T_{[s_1+1,s_2]}, \ldots, T_{[s_{n-1}+1,s_n]}), \tag{4.6}$$

and for $i \neq 0$, we define

$$G_u(i,T,Y) = \begin{cases} (-1)^{T_{s_{i+1}}} C^{m_i}(Y) & \text{if } T_{[1,s_i+1]} = 10\cdots 0, \\ 0 & \text{otherwise.} \end{cases} \tag{4.7}$$

This family $(G_u)_u$ is clearly uniform, and its height $n+1$ is logarithmic in $|u|$.

Let H_u be the solution (as defined in (4.1) and (4.2)) of the difference equation given by G_u. We prove by induction on i that $H_u(i,T) \in [2^{l_i}+1]$ for all T, and that

$$G_u(i,V,H_u(i,V)) = (-1)^{V_{s_{i+1}}}\phi_i(V_{[s_i+1,s_{i+1}]}, \ldots, V_{[s_{n-1}+1,s_n]}) \tag{4.8}$$

if $V_{[1,s_i+1]} = 10\cdots 0$ (otherwise it is immediate from the definition that $G_u(i,V,H_u(n,V)) = 0$). For $i = 0$, the claims follow from (4.6). For $i > 0$, suppose that the induction hypothesis

$$G_u(i-1,V,H_u(i-1,V)) = (-1)^{V_{s_i}}\phi_{i-1}(V_{[s_{i-1}+1,s_i]}, \ldots, V_{[s_{n-1}+1,s_n]}) \tag{4.9}$$

holds. Since H_u is the solution of the difference equation given by G_u, we have

$$H_u(i,T) = \sum_{V=0}^{T-1} G_u(i-1,V,H_u(i-1,V)). \tag{4.10}$$

Since the assumption (4.9) implies that flipping the bit V_{s_i} of any V reverses the sign of $G_u(i-1,V,H_u(i-1,V))$, most of the summands in (4.10) cancel out. The only nonzero terms that can survive are the ones corresponding to those V that satisfy $V_{[1,s_{i-1}+1]} = 10\cdots 0$ and lie between the numbers whose binary representations are $T_{s_n}\ldots T_{s_i+1}00\ldots 0$ and $T_{s_n}\ldots T_{s_i+1}01\ldots 1$. There are at most 2^{l_i} such terms, and each of them is 0 or 1, so $H_u(i,T) \in [2^{l_i}+1]$. In particular, if $V_{[1,s_i+1]} = 10\cdots 0$, we have

$$H_u(i,V) = \sum_{x\in\{0,1\}^{l_i}} \phi_{i-1}(x, V_{[s_i+1,s_{i+1}]}, \ldots, V_{[s_{n-1}+1,s_n]}). \tag{4.11}$$

By this and (4.4), we have

$$C^{m_i}(H_u(i,V)) = \phi_i(V_{[s_i+1,s_{i+1}]}, \ldots, V_{[s_{n-1}+1,s_n]}). \tag{4.12}$$

By this and (4.7), we have (4.8), completing the induction step.

By substituting n for i and 2^{s_n} for V in (4.8), we get $G_u(n,2^{s_n},H_u(n,2^{s_n})) = \phi_n() = L(u)$. Hence $H_u(n+1,2^{s_n}+1) = L(u)$. □

Note that for CH-hardness, we could have defined L using a faster growing function than 2^n in (4.3). This would allow the difference equation in Lemma 9 to have height smaller than logarithmic. We stated Lemma 9 with logarithmic height, simply because this was the highest possible difference equations that we were able to simulate by $\mathrm{C}^{(\infty,k)}$ functions (in the proof of Lemma 10 below).

4.2. **Simulating difference equations by real functions.** We show that certain families of smooth differential equations can simulate PSPACE- and CH-hard difference equations from the previous section.

Before stating Lemmas 10 and 11, we extend the definition of polynomial-time computability of real functions to that of *families* of real functions. A machine M *computes* a family $(f_u)_u$ of functions $f_u\colon A \to \mathbf{R}$ indexed by strings u if for any $x \in A$ and any name ϕ_x of x, the function taking v to $M^{\phi_x}(u, v)$ is a name of $f_u(x)$. We say a family of real functions $(f_u)_u$ is polynomial-time if there is a polynomial-time machine computing $(f_u)_u$.

Lemma 10. There exist a CH-hard language L and a polynomial μ, such that for any $k \geq 1$ and polynomial γ, there are a polynomial ρ and families $(g_u)_u$, $(h_u)_u$ of real functions such that $(g_u)_u$ is polynomial-time computable and for any string u:

(i) $g_u\colon [0,1] \times [-1,1] \to \mathbf{R}$, $h_u\colon [0,1] \to [-1,1]$;
(ii) $h_u(0) = 0$ and $Dh_u(t) = g_u(t, h_u(t))$ for all $t \in [0,1]$;
(iii) g_u is of class $\mathrm{C}^{(\infty,k)}$;
(iv) $D^{(i,0)}g_u(0,y) = D^{(i,0)}g_u(1,y) = 0$ for all $i \in \mathbf{N}$ and $y \in [-1,1]$;
(v) $\left|D^{(i,j)}g_u(t,y)\right| \leq 2^{\mu(i,|u|)-\gamma(|u|)}$ for all $i \in \mathbf{N}$ and $j \in \{0,\dots,k\}$;
(vi) $h_u(1) = 2^{-\rho(|u|)}L(u)$.

Lemma 11. There exist a PSPACE-hard language L and a polynomial μ, such that for any polynomial γ, there are a polynomial ρ and families $(g_u)_u$, $(h_u)_u$ of real functions such that $(g_u)_u$ is polynomial-time computable and for any string u satisfying (i)–(vi) of Lemma 10 with $k = 1$.

In Lemmas 10 and 11, we have the new conditions (iii)–(v) about the smoothness and the derivatives of g_u that were not present in [4, Lemma 4.1]. To satisfy these conditions, we construct g_u using the smooth function f in following lemma.

Lemma 12 ([10, Lemma 3.6])**.** There exist a polynomial-time function $f\colon [0,1] \to \mathbf{R}$ of class C^∞ and a polynomial s such that

- $f(0) = 0$ and $f(1) = 1$;
- $D^n f(0) = D^n f(1) = 0$ for all $n \geq 1$;
- f is strictly increasing;
- $D^n f$ is polynomial-time computable for all $n \geq 1$;
- $|D^n f| \leq 2^{s(n)}$ for all $n \geq 1$.

Although the existence of the polynomial s was not explicitly stated in [10, Lemma 3.6], it can be proved easily.

We will prove Lemma 10 using Lemma 9 as follows. Let $(G_u)_u$ be a family of functions obtained by Lemma 9, and let $(H_u)_u$ be the family of the solutions of the difference equations given by $(G_u)_u$. We construct h_u and g_u from H_u and G_u so that $h_u(T/2^{q(|u|)}) = \sum_{i=0}^{p(|u|)} H_u(i,T)/B^{d_u(i)}$ for each $T = 0, \dots, 2^{q(|u|)}$ and $Dh_u(t) = g_u(t, h_u(t))$. The polynomial-time computability of $(g_u)_u$ follows from that of $(G_u)_u$. We omit the analogous and easier proof of Lemma 11.

Proof of Lemma 10. Let L and $(G_u)_u$ be as in Lemma 9, and let H_u be the solutions of the difference equations given by G_u. Let f and s be as in Lemma 12.

We may assume that $G_u\colon [p(|u|)] \times [2^{q(|u|)}] \times [2^{r(|u|)}] \to \{-1,0,1\}$ for some $p, q, r\colon \mathbf{N} \to \mathbf{N}$, where p has logarithmic growth and q and r are polynomials. We may also assume, similarly

to the beginning of the proof of [4, Lemma 4.1], that

$$H_u(i, 2^{q(|u|)}) = \begin{cases} L(u) & \text{if } i = p(|u|), \\ 0 & \text{if } i < p(|u|), \end{cases} \tag{4.13}$$

$$G_u(i, T, Y) \neq 0 \to i = j_u(T) \tag{4.14}$$

for some $j_u \colon [2^{q(|u|)}] \to [p(|u|)]$.

We construct families of real functions $(g_u)_u$ and $(h_u)_u$ that simulate G_u and H_u in the sense that $h_u(T/2^{q(|u|)}) = \sum_{i=0}^{p(|u|)} H_u(i, T)/B^{d_u(i)}$, where the constant B and the (increasing) function $d_u \colon [p(|u|) + 1] \to \mathbf{N}$ are defined by

$$B = 2^{\gamma(|u|)+r(|u|)+s(k)+k+3}, \qquad d_u(i) = \begin{cases} (k+1)^i & \text{if } i < p(|u|), \\ \sigma(|u|) & \text{if } i = p(|u|), \end{cases} \tag{4.15}$$

where σ is a polynomial satisfying $(k+1)^{p(x)} \leq \sigma(x)$ (which exists because p is logarithmic). Thus, the value $H_u(i, T) \in [2^{r(|u|)}]$ will be stored in the $d_u(i)$th digit of the base-B expansion of the real number $h_u(T/2^{q(|u|)})$. This exponential spacing described by d_u will be needed when we bound the derivative $D^{(i,j)}g_u$ in (4.22) below.

For each $(t, y) \in [0, 1] \times [-1, 1]$, there exist unique $N \in \mathbf{N}$, $\theta \in [0, 1)$, $Y \in \mathbf{Z}$ and $\eta \in [-1/4, 3/4)$ such that $t = (T + \theta)2^{-q(|u|)}$ and $y = (Y + \eta)B^{-d_u(j_u(T))}$. Using f and a polynomial s of Lemma 12, we define $\tilde{g}_{u,Y} \colon [0, 1] \to \mathbf{R}$, $g_u \colon [0, 1] \times [-1, 1] \to \mathbf{R}$ and $h_u \colon [0, 1] \to [-1, 1]$ by

$$\tilde{g}_{u,Y}(t) = \frac{2^{q(|u|)} D f(\theta)}{B^{d_u(j_u(T)+1)}} G_u\big(j_u(T), T, Y \bmod 2^{r(|u|)}\big), \tag{4.16}$$

$$g_u(t, y) = \begin{cases} \tilde{g}_{u,Y}(t) & \text{if } \eta \leq \frac{1}{4}, \\ (1 - f(\frac{4\eta-1}{2}))\tilde{g}_{u,Y}(t) + f(\frac{4\eta-1}{2})\tilde{g}_{u,Y+1}(t) & \text{if } \eta > \frac{1}{4}, \end{cases} \tag{4.17}$$

$$h_u(t) = \sum_{i=0}^{p(|u|)} \frac{H_u(i, T)}{B^{d_u(i)}} + \frac{f(\theta)}{B^{d_u(j_u(T)+1)}} G_u\big(j_u(T), T, H_u(j_u(T), T)\big). \tag{4.18}$$

We will verify that $(g_u)_u$ and $(h_u)_u$ defined above satisfy all the conditions stated in Lemma 10. Polynomial-time computability of $(g_u)_u$ can be verified using Lemma 6. The condition (i) is immediate, and (ii) follows from the relation between G_u and H_u (by a similar argument to [4, Lemma 4.1]).

Since g_u is constructed by interpolating between the (finitely many) values of G_u using a $\mathrm{C}^{(\infty,\infty)}$ function f, it is of class $\mathrm{C}^{(\infty,\infty)}$ and thus satisfies (iii). Calculating from (4.16), we have

$$D^i \tilde{g}_{u,Y}(t) = \frac{2^{(i+1)q(|u|)} D^{i+1} f(\theta)}{B^{d_u(j_u(T)+1)}} G_u\big(j_u(T), T, Y \bmod 2^{r(|u|)}\big) \tag{4.19}$$

for each $i \in \mathbf{N}$. By this and (4.17), we have

$$D^{(i,0)} g_u(t, y) = \begin{cases} D^i \tilde{g}_{u,Y}(t) & \text{if } \eta \leq \frac{1}{4}, \\ \big(1 - f\big(\frac{4\eta-1}{2}\big)\big) D^i \tilde{g}_{u,Y}(t) + f\big(\frac{4\eta-1}{2}\big) D^i \tilde{g}_{u,Y+1}(t) & \text{if } \frac{1}{4} < \eta \end{cases} \tag{4.20}$$

for each $i \in \mathbf{N}$ and

$$D^{(i,j)}g_u(t,y) = \begin{cases} 0 & \text{if } -\frac{1}{4} < \eta < \frac{1}{4}, \\ \left(2B^{d_u(j_u(T))}\right)^j \cdot D^j f\left(\frac{4\eta-1}{2}\right) \cdot (D^i \tilde{g}_{u,Y+1}(t) - D^i \tilde{g}_{u,Y}(t)) & \text{if } \frac{1}{4} < \eta < \frac{3}{4} \end{cases} \tag{4.21}$$

for each $i \in \mathbf{N}$ and $j \in \{1, \ldots, k\}$. Substituting $t = 0, 1$ ($\theta = 0$) into (4.20), we get $D^{(i,0)}g_u(0,y) = D^{(i,0)}g_u(1,y) = 0$, as required in (iv).

We show that (v) holds with $\mu(x,y) = (x+1)q(y) + s(x+1)$. Note that μ is a polynomial independent of k and γ, and that $|D^i \tilde{g}_{u,Y}(t)| \leq 2^{\mu(i,|u|)}/B^{d_u(j_u(T)+1)}$ by (4.19). Hence,

$$\begin{aligned} |D^{(i,j)}g_u(t,y)| &\leq \left(2B^{d_u(j_u(T))}\right)^k \cdot 2^{s(k)} \cdot 2 \cdot \frac{2^{\mu(i,|u|)}}{B^{d_u(j_u(T)+1)}} = \frac{2^{\mu(i,|u|)} \cdot 2^{s(k)+k+1}}{B^{d_u(j_u(T)+1)-k \cdot d_u(j_u(T))}} \\ &\leq 2^{\mu(i,|u|)} \cdot \frac{2^{s(k)+k+1}}{B} \leq 2^{\mu(i,|u|)-\gamma(|u|)} \end{aligned} \tag{4.22}$$

by (4.20), (4.21) and our choice of B.

We have (vi) with $\rho(x) = \sigma(x) \cdot (\gamma(x) + r(x) + s(k) + k + 3)$, because

$$h_u(1) = \frac{H_u(p(|u|), 2^{q(|u|)})}{B^{d_u(p(|u|))}} = \frac{L(u)}{2^{\sigma(|u|)\cdot(\gamma(|u|)+r(|u|)+s(k)+k+3)}} = 2^{-\rho(|u|)}L(u). \tag{4.23}$$

□

We used the exponential positioning function d_u defined at (4.15), so that we have $d_u(i+1) > k \cdot d_u(i)$ for the second inequality of (4.22) for $k \geq 2$. Because of this, we had to restrict p to be logarithmic, because otherwise the function σ in (4.15) would have to be superpolynomial and so would ρ in (4.23).

The proof of Lemma 11 is analogous, starting with the L and $(G_u)_u$ of Lemma 8. The only difference is that p is now a polynomial and therefore we use $d_u(i) = i$ in (4.15) (and $\sigma = p$ in (4.23)).

4.3. **Proof of the Main Theorems.** Using the function families $(g_u)_u$ and $(h_u)_u$ obtained from Lemmas 10 or 11, we construct the functions g and h in Theorems 1 and 2 as follows. Divide $[0,1)$ into infinitely many subintervals $[l_u^-, l_u^+]$, with midpoints c_u. We construct h by putting a scaled copy of h_u onto $[l_u^-, c_u]$ and putting a horizontally reversed scaled copy of h_u onto $[c_u, l_u^+]$ so that $h(l_u^-) = 0$, $h(c_u) = 2^{-\rho'(|u|)}L(u)$ and $h(l_u^+) = 0$ where ρ' is a polynomial. In the same way, g is constructed from $(g_u)_u$ so that g and h satisfy (1.1). We give the details of the proof of Theorem 2 from Lemma 10, and omit the analogous proof of Theorem 1 from Lemma 11.

Proof of Theorem 2. Let L and μ be as Lemma 10. Define $\lambda(x) = 2x + 2$, $\gamma(x) = \mu(x,x) + x\lambda(x)$ and for each u let $\Lambda_u = 2^{\lambda(|u|)}$, $c_u = 1 - 2^{-|u|} + 2\bar{u} + 1/\Lambda_u$, $l_u^{\mp} = c_u \mp 1/\Lambda_u$, where $\bar{u} \in \{0, \ldots, 2^{|u|} - 1\}$ is the number represented by u in binary notation. Let ρ, $(g_u)_u$, $(h_u)_u$ be as in Lemma 10 corresponding to the above γ.

We define

$$g\left(l_u^{\mp} \pm \frac{t}{\Lambda_u}, \frac{y}{\Lambda_u}\right) = \begin{cases} \pm \sum_{l=0}^{k} \frac{D^{(0,l)} g_u(t,1)}{l!}(y-1)^l & \text{if } 1 < y, \\ \pm g_u(t,y) & \text{if } -1 \le y \le 1, \\ \pm \sum_{l=0}^{k} \frac{D^{(0,l)} g_u(t,-1)}{l!}(y+1)^l & \text{if } 1 < y, \end{cases} \tag{4.24}$$

$$h\left(l_u^{\mp} \pm \frac{t}{\Lambda_u}\right) = \frac{h_u(t)}{\Lambda_u} \tag{4.25}$$

for each string u and $t \in [0,1)$, $y \in [-1,1]$. Let $g(1,y) = 0$ and $h(1) = 0$ for any $y \in [-1,1]$.

It can be shown similarly to the Lipschitz version [4, Theorem 3.2] that g and h satisfy (1.1) and g is polynomial-time computable. Here we only prove that g is of class $\mathrm{C}^{(\infty,k)}$. We claim that for each $i \in \mathbf{N}$ and $j \in \{0,\dots,k\}$, the derivative $D_1^i D_2^j g$ is given by

$$D_1^i D_2^j g\left(l_u^{\mp} \pm \frac{t}{\Lambda_u}, \frac{y}{\Lambda_u}\right) = \begin{cases} \pm \Lambda_u^{i+j} \sum_{l=j}^{k} \frac{D^{(i,l)} g_u(t,1)}{(l-j)!}(y-1)^l & \text{if } y < -1, \\ \pm \Lambda_u^{i+j} D^{(i,j)} g_u(t,y) & \text{if } -1 \le y \le 1, \\ \pm \Lambda_u^{i+j} \sum_{l=j}^{k} \frac{D^{(i,l)} g_u(t,-1)}{(l-j)!}(y+1)^l & \text{if } 1 < y \end{cases} \tag{4.26}$$

for each $l_u^{\mp} \pm t/\Lambda_u \in [0,1)$ and $y/\Lambda_u \in [-1,1]$, and by $D_1^i D_2^j g(1,y) = 0$. This is verified by induction on $i+j$. The equation (4.26) follows from calculation (note that this means verifying that (4.26) follows from the definition of g when $i = j = 0$; from the induction hypothesis about $D_2^{j-1} g$ when $i = 0$ and $j > 0$; and from the induction hypothesis about $D_1^{i-1} D_2^j g$ when $i > 0$). That $D_1^i D_2^j g(1,y) = 0$ is immediate from the induction hypothesis if $i = 0$. If $i > 0$, the derivative $D_1^i D_2^j g(1,y)$ is by definition the limit

$$\lim_{s \to 1-0} \frac{D_1^{i-1} D_2^j g(1,y) - D_1^{i-1} D_2^j g(s,y)}{1-s}. \tag{4.27}$$

This can be shown to exist and equal 0, by observing that the first term in the numerator is 0 and the second term is bounded, when $s \in [l_u^-, l_u^+]$, by

$$\begin{aligned} |D_1^{i-1} D_2^j g(s,y)| &\le \Lambda_u^{i-1+j} \sum_{l=j}^{k} |D^{(i-1,l)} g_u| \cdot (\Lambda_u + 1)^l \\ &\le \Lambda_u^{i-1+j} \cdot k \cdot 2^{\mu(i-1,|u|)-\gamma(|u|)} \cdot (2\Lambda_u)^k \\ &\le 2^{(i-1+j+k)\lambda(|u|)+2k+\mu(i-1,|u|)-\gamma(|u|)} \le 2^{-2|u|} \le 2^{-|u|+1}(1-s), \end{aligned} \tag{4.28}$$

where the second inequality is from Lemma 10 (v) and the fourth inequality holds for sufficiently large $|u|$ by our choice of γ. The continuity of $D_1^i D_2^j g$ on $[0,1) \times [-1,1]$ follows from (4.26) and Lemma 10 (iv). The continuity on $\{1\} \times [-1,1]$ is verified by estimating $D_1^i D_2^j g$ similarly to (4.28). □

5. Other Versions of the Problem

5.1. **Complexity of the final value.** In the previous section, we considered the complexity of the solution h of the ODE as a real function. Here we discuss the complexity of the final value $h(1)$ and relate it to tally languages (subsets of $\{0\}^*$), as did Ko [9] and Kawamura [3, Theorem 5.1] for the Lipschitz continuous case.

We say that a language L *reduces to* a real number x if there is a polynomial-time oracle Turing machine M such that M^ϕ accepts L for any name ϕ of x. Note that this is the same as the reduction in Definition 7 to a constant function with value x.

Theorem 13. For any tally language $T \in \mathsf{PSPACE}$, there are a polynomial-time computable function $g\colon [0,1] \times [-1,1] \to \mathbf{R}$ of class $\mathrm{C}^{(\infty,1)}$ and a function $h\colon [0,1] \to \mathbf{R}$ satisfying (1.1) such that L reduces to $h(1)$.

Theorem 14. Let k be a positive integer. For any tally language $T \in \mathsf{CH}$, there are a polynomial-time computable function $g\colon [0,1] \times [-1,1] \to \mathbf{R}$ of class $\mathrm{C}^{(\infty,k)}$ and a function $h\colon [0,1] \to \mathbf{R}$ satisfying (1.1) such that L reduces to $h(1)$.

We can prove Theorem 14 from Lemma 10 in the same way as the proof of [3, Theorem 5.1]. We skip the proof of Theorem 13 since it is similar.

Proof. Let T be any tally language in CH and k be any positive integer, and let L and μ be as Lemma 10. Define $\lambda(x) = x + 1$, $\gamma(x) = \mu(x, x) + x\lambda(x)$ and let ρ, $(g_u)_u$, $(h_u)_u$ be as in Lemma 10 corresponding to the γ. Since L is CH-hard, there are a polynomial-time function F such that $T(0^i) = L(F(0^i))$ for all i.

Let $l_n = 1 - 2^n$ and $\bar\rho(n) = \sum_{i=0}^{n-1} \rho(|F(0^i)|)$. Define g and h as follows: when the first variable is in $[0,1)$, let

$$g\left(l_n + \frac{t}{2^{n+1}}, \frac{2m + (-1)^m y}{2^{2n+\gamma(n)+\bar\rho(n)}}\right) = \frac{g_{F(0^n)}(t,y)}{2^{n-1+\gamma(n)+\bar\rho(n)}}, \tag{5.1}$$

$$h\left(l_n + \frac{t}{2^{n+1}}\right) = \frac{h_{F(0^n)}(t)}{2^{2n+\gamma(n)+\bar\rho(n)}} + \sum_{i=0}^{n-1} \frac{T(0^i)}{2^{2i+\gamma(i)+\bar\rho(i+1)}} \tag{5.2}$$

for each $n \in \mathbf{N}$, $t \in [0,1]$, $y \in [-1,1]$ and $m \in \mathbf{Z}$; when the first variable is 1, let

$$g(1, y) = 0, \tag{5.3}$$

$$h(1) = \sum_{n=0}^{\infty} \frac{T(0^n)}{2^{2n+\gamma(n)+\bar\rho(n+1)}}. \tag{5.4}$$

It can be proved similarly to the proof of Theorem 2 that g is polynomial-time computable and of class $\mathrm{C}^{(\infty,k)}$ and that g and h satisfy (1.1). The equation (5.4) implies that T reduces to $h(1)$. □

5.2. **Complexity of operators.** Both Theorems 1 and 2 state the complexity of the solution h under the assumption that g is polynomial-time computable. But how hard is it to "solve" differential equations, i.e., how complex is the operator that takes g to h? To make this question precise, we need to define the complexity of operators taking real functions to real functions.

Recall that, in order to discuss complexity of real functions, we used string functions as names of elements in $\mathbf{R}$. Such an encoding is called a *representation* of $\mathbf{R}$. Likewise, we now want to encode real functions by string functions to discuss complexity of real operators. In other words, we need to define representations of the class $\mathrm{C}_{[0,1]}$ of continuous functions $h\colon [0,1] \to \mathbf{R}$ and class $\mathrm{CL}_{[0,1]\times[-1,1]}$ of Lipschitz continuous functions $g\colon [0,1]\times[-1,1] \to \mathbf{R}$. The notions of computability and complexity depend on these representations. Following [6], we use $\delta_\square$ and $\delta_{\square\mathrm{L}}$ as the representations of $\mathrm{C}_{[0,1]}$ and $\mathrm{CL}_{[0,1]\times[-1,1]}$, respectively. It is known that $\delta_\square$ is the canonical representation of $\mathrm{C}_{[0,1]}$ in a certain sense [5], and $\delta_{\square\mathrm{L}}$ is the representation defined by adding to $\delta_\square$ the information on the Lipschitz constant.

Since these representations use string functions whose values have variable lengths, we use *second order polynomials* to bound the amount of resources (time and space) of machines [6], and this leads to the definitions of second-order complexity classes (e.g. **FPSPACE**, polynomial-space computable), reductions (e.g. $\leq_{\mathrm{W}}$, polynomial-time Weihrauch reduction), and hardness. Combining them with the representations of real functions mentioned above, we can restate our theorems in the constructive form as follows.

Let *ODE* be the operator mapping a real function $g \in \mathrm{CL}_{[0,1]\times[-1,1]}$ to the solution $h \in \mathrm{C}_{[0,1]}$ of (1.1). The operator *ODE* is a partial function from $\mathrm{CL}_{[0,1]\times[-1,1]}$ to $\mathrm{C}_{[0,1]}$ (it is partial because the trajectory may fall out of the interval $[-1,1]$, see the paragraph following Theorem 2). In [6, Theorem 4.9], the $(\delta_{\square\mathrm{L}}, \delta_\square)$-**FPSPACE**-$\leq_{\mathrm{W}}$-completeness of *ODE* was proved by modifying the proof of the results in the third row of Table 1. Theorem 1 can be rewritten in a similar way. That is, let $ODE\restriction^{\mathrm{C}^{(\infty,1)}}$ be the operator *ODE* restricted to class $\mathrm{C}^{(\infty,1)}$. Then we have:

Theorem 15. The operator $ODE\restriction^{\mathrm{C}^{(\infty,1)}}$ is $(\delta_{\square\mathrm{L}}, \delta_\square)$-**FPSPACE**-$\leq_{\mathrm{W}}$-complete.

To show this theorem, we need to verify that the information used to construct functions in the proof of Theorem 1 can be obtained easily from the inputs. We omit the proof since it does not require any new technique. Theorem 15 automatically implies Theorem 1 because of [6, Lemmas 4.7 and 4.8] and the $\leq_{\mathrm{W}}$ versions of [6, Lemmas 3.11 and 3.12].

In contrast, the polynomial-time computability in the analytic case (the last row of Table 1) is *not* a consequence of a statement based on $\delta_\square$. It is based on the calculation of the Taylor coefficients, and hence we only know how to convert the Taylor sequence of g at a point to that of h. In other words, the operator *ODE* restricted to the analytic functions is not $(\delta_{\square\mathrm{L}}, \delta_\square)$-**FP**-computable, but $(\delta_{\mathrm{Taylor}}, \delta_{\mathrm{Taylor}})$-**FP**-computable, where δ_{Taylor} is the representation that encodes an analytic function using its Taylor coefficients at a point in a suitable way. More discussion on representations of analytic functions and the complexity of operators on them can be found in [7].

References

[1] O. Bournez, D. Graça, and A. Pouly. Solving analytic differential equations in polynomial time over unbounded domains. In *Proceedings of the 36th International Symposium on Mathematical Foundations of Computer Science* (MFCS), LNCS 6907, 170–181, 2011.

[2] E.A. Coddington and N. Levinson. *Theory of Ordinary Differential Equations*. McGraw-Hill, 1955.

[3] A. Kawamura. Complexity of initial value problems. To appear in *Fields Institute Communications*.

[4] A. Kawamura. Lipschitz continuous ordinary differential equations are polynomial-space complete. *Computational Complexity*, 19(2):305–332, 2010.

[5] A. Kawamura. On function spaces and polynomial-time computability. Dagstuhl Seminar 11411: Computing with Infinite Data, 2011.

[6] A. Kawamura and S. Cook. Complexity theory for operators in analysis. *ACM Transactions on Computation Theory*, 4(2), Article 5, 2012.
[7] A. Kawamura, N. Müller, C. Rösnick, and M. Ziegler. Parameterized uniform complexity in numerics: from smooth to analytic, from NP-hard to polytime. http://arxiv.org/abs/1211.4974.
[8] A. Kawamura, H. Ota, C. Rösnick, and M. Ziegler. Computational complexity of smooth differential equations. In *Proceedings of the 37th International Symposium on Mathematical Foundations of Computer Science* (MFCS), LNCS 7464, 578–589, 2012.
[9] K.I. Ko. On the computational complexity of ordinary differential equations. *Information and Control*, 58(1-3):157–194, 1983.
[10] K.I. Ko. *Complexity Theory of Real Functions*. Birkhäuser Boston, 1991.
[11] K.I. Ko and H. Friedman. Computing power series in polynomial time. *Advances in Applied Mathematics*, 9(1):40–50, 1988.
[12] W. Miller. Recursive function theory and numerical analysis. *Journal of Computer and System Sciences*, 4(5):465–472, 1970.
[13] N.T. Müller. Uniform computational complexity of Taylor series. *Automata, Languages and Programming*, pages 435–444, 1987.
[14] I. Parberry and G. Schnitger. Parallel computation with threshold functions. *Journal of Computer and System Sciences*, 36(3):278–302, 1988.
[15] M.B. Pour-El and I. Richards. A computable ordinary differential equation which possesses no computable solution. *Annals of Mathematical Logic*, 17(1-2):61–90, 1979.
[16] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.
[17] J. Torán. Complexity classes defined by counting quantifiers. *Journal of the ACM*, 38(3):752–773, 1991.
[18] K.W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23(3):325–356, 1986.
[19] I. Wegener. *Komplexitätstheorie: Grenzen der Effizienz von Algorithmen*. Springer, 2003. In German. English translation by R. Pruim: *Complexity Theory: Exploring the Limits of Efficient Algorithms*, Springer, 2005.
[20] K. Weihrauch. *Computable Analysis: An Introduction*. Texts in Theoretical Computer Science. Springer, 2000.
[21] 太田，河村，ツィーグラー，レースニク．滑らかな常微分方程式の計算量．数理解析研究所講究録1799「アルゴリズムと計算理論の新展開」67〜72頁．平成24年．(H. Ota, A. Kawamura, M. Ziegler, and C. Rösnick. Complexity of smooth ordinary differential equations. Presented at the Tenth EATCS/LA Workshop on Theoretical Computer Science, Kyoto, 2012.) In Japanese.