Computational Complexity of Smooth Differential Equations

Akitoshi Kawamura[1], Hiroyuki Ota[1], Carsten Rösnick[2], and Martin Ziegler[2]

[1] University of Tokyo
[2] TU Darmstadt

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 in hope of understanding the intrinsic hardness of solving the equation numerically. Kawamura showed in 2010 that the solution h can be PSPACE-hard even if g is assumed to be Lipschitz continuous. We place further requirements on the smoothness of g and obtain the following results: the solution h is still PSPACE-hard if g is assumed to be continuously differentiable; for each $k \geq 2$, the solution h is hard for the counting hierarchy if g is assumed to be k-times continuously differentiable.

1 Introduction

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

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

where Dh denotes the derivative of h. How complex can the solution h be, assuming that g is polynomial-time computable? Here, the polynomial-time computability and other notions of complexity are from the field of *Computable Analysis* [14] and measure how hard it is to approximate real functions with specified precisions (Section 2).

If we put no assumption on g other than being polynomial-time computable, the solution h (which is not unique in general) can be non-computable. Table 1.1 summarizes known results about the complexity of h under various assumptions on g, with the assumptions getting stronger as we go down. In particular, if g is (globally) Lipschitz continuous, then the (unique) solution h is known to be polynomial-space computable but still can be PSPACE-hard [3]. 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) is often beneficial in applying certain algorithms or simplifying 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 ask whether,

Table 1.1. The complexity of the solution h of (1.1) assuming g is polynomial-time computable.

Assumptions	Upper bounds	Lower bounds
—	—	can be all non-computable [11]
h is the unique solution	computable [1]	can take arbitrarily long time [6, **?**]
the Lipschitz condition	polynomial-space [6]	can be PSPACE-hard [3]
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 any constant k)	polynomial-space	can be CH-hard (Theorem 2)
g is analytic	polynomial-time [10, **?**]	—

for our differential equation (1.1), smoothness helps to reduce the complexity of the solution.

At the extreme is the case where g is analytic: as the last row of the table shows, h can then be shown to be polynomial-time computable by the Taylor series methods. Thus our interest is in the cases between Lipschitz and analytic (the fourth and fifth rows in the table). We say that g is of class $\mathrm{C}^{(i,j)}$ if the partial derivative $D^{(i,j)}g$ (often also denoted $\partial^{i+j}g(t,y)/\partial t^i \partial y^j$) exists and is continuous[3]; it is said to be of class $\mathrm{C}^{(\infty,j)}$ if it is of class $\mathrm{C}^{(i,j)}$ for all $i \in \mathbf{N}$.

Theorem 1. *There is 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* $h\colon [0,1] \to \mathbf{R}$.

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* $h\colon [0,1] \to \mathbf{R}$*, where* CH $\subseteq$ PSPACE *is the Counting Hierarchy (see Section 3.2).*

ここで $g\colon [0,1]\times\mathbf{R} \to \mathbf{R}$ でなく $g\colon [0,1]\times[-1,1] \to \mathbf{R}$ と書いたのは, 本稿では実関数の多項式時間計算可能性を, 定義域が有界閉領域のときにのみ定義するからである. このため h が区間 $[-1,1]$ の外に値を取ることがあると方程式 (1.1) が意味をなさなくなるが, 定理 1 において h が解であるというのは, 任意の $t \in [0,1]$ について $h(t) \in [-1,1]$ が満たされることも含めて述べている. なお両定理とも Lipschitz 条件よりも強い仮定を置いているため, そのような h は g に対して, 存在すれば唯一である.

本稿のように対象を滑らかな関数に制限することによる計算量の変化について, 常微分方程式以外の問題では次のような否定的な結果がある. 多項式時間計算可能な関数から積分により得られる関数は, もとの関数を無限回微分可能なもの

[3] Another common terminology is to say that g is of class C^k if it is of class $\mathrm{C}^{(i,j)}$ for all i, j with $i + j \leq k$.

に限ってもなお一般の場合と同じく #P 困難である. [7, 定理 5.33]. 最大化でも同様に, 無限回微分可能な関数に限っても一般の場合と同じく NP 困難である [7, 定理 3.7][4] (なお対象を解析的な関数に限ると, やはり級数を用いた議論により, これらは多項式時間計算可能になる). 一方常微分方程式については, 定理 2 は各 k についてそれぞれ成立つが, g が (∞, ∞) 回微分可能であると仮定したときの h の計算量については依然不明である.

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

Let A and B be bounded closed interval in $\mathbf{R}$. We denote $|f|$ as $\sup_{x\in A} f(x)$ where $f\colon A \to \mathbf{R}$.

A function $f\colon A \to \mathbf{R}$ is *i-times continuously differentiable* if there exist the derivatives $Df, D^2 f, \ldots, D^i f$ and all of them are continuous. We write C^k_A for the set of k-times continuously differentiable functions from A to $\mathbf{R}$, and abbreviate it as C^k if the domain A is obvious. A function $g\colon A \times B \to \mathbf{R}$ is *(i, j)-times continuously differentiable* if for each $n \in \{0, \ldots, i\}$ and $m \in \{0, \ldots j\}$, there exists the derivative $D_1^n D_2^m g$ and it is continuous, where $D_1 g$ is the derivative of a two variable function g in the direction of the first variable and $D_2 g$ is the derivative in the direction of the second variable. A function g is *(∞, j)-times continuously differentiable* if g is (i, j)-times continuously differentiable for all $i \in \mathbf{N}$. We write $\mathrm{C}^{(i,j)}[A \times B]$ for the set of (i, j)-times continuously differentiable functions from $A \times B$ to $\mathbf{R}$, and abbreviate it as $\mathrm{C}^{(i,j)}$ if the domain is obvious. When a function g is in $\mathrm{C}^{(i,j)}[A\times B]$, we write $D^{(i,j)}g$ for the derivative $D_1^i D_2^j g$.

2 Computational Complexity of Real Functions

2.1 Computation of Real Function

We start by fixing a way to encode real numbers by functions from strings to strings.

Definition 3. *A function $\phi\colon \{0\}^* \to \{0, 1\}^*$ 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 receives 0^n and returns an approximation of x with precision 2^{-n}.

We use *oracle Turing machines* (henceforth just machines) to work on names of real numbers (Figure 2.1).

[4] ただし葛 [7, 定理 3.7] の証明において関数 f を

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

に修正する必要がある.

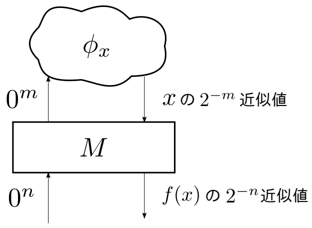

Fig. 2.1. A machine M computing a real function f.

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 4. *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 $f(x)$.*

When A is a bounded closed interval of $\mathbf{R}^2$, we define a machine computing $f\colon A \to \mathbf{R}$ in a similar way using machines with two oracles. A real function is (*polynomial-time*) *computable* if there exists some machine that computes it (in polynomial time).

When a machine M computes a real function f, for each demanded precision 2^{-n}, precision 2^{-m}. Computable real functions are continuous. Giving all approximations of functions at rational points an relation between n and m, we can characterize (polynomial-time) computability of real function without oracle Turing machines

Lemma 5. *A real function is (polynomial-time) computable if and only if there exist a (polynomial-time) computable function $\phi\colon (\mathbf{Q} \cap [0,1]) \times \{0\}^* \to \mathbf{Q}$ and polynomial $p\colon \mathbf{N} \to \mathbf{N}$ such that for all $d \in \mathbf{Q} \cap [0,1]$ and $n \in \mathbf{N}$,*

$$|\phi(d, 0^n) - f(d)| \leq 2^{-n}, \tag{2.1}$$

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

$$|x - y| \leq 2^{-p(m)} \Rightarrow |f(x) - f(y)| \leq 2^{-m}. \tag{2.2}$$

where each rational number in $\mathbf{Q}$ is represented by a pair of integers in binary representation.

2.2 Reduction and Hardness

A language $L \subseteq \{0,1\}^*$ is identified with the function $L\colon \{0,1\}^* \to \{0,1\}$ such that $L(u) = 1$ if and only if $u \in L$.

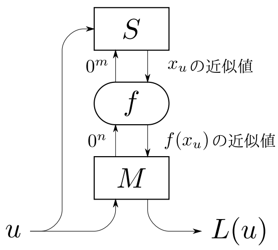

Fig. 2.2. Reduction from a language L to a function $f\colon [0,1] \to \mathbf{R}$

Definition 6 (Reduction). *A Language L reduces to a real function $f\colon [0,1] \to \mathbf{R}$ if there exists a polynomial-time function S and a polynomial-time oracle Turing machine M (Figure 2.2) such that for any string u:*

(i) $S(u, \cdot)$ is a name of x_u;
(ii) $M^{\phi}(u)$ accepts if and only if $u \in L$ for any name ϕ of $f(x_u)$.

As a matter of form this definition is different from that by Kawamura but they have same power as reduction. Let C be a complexity class, a function f is *C-hard* if for all language in C is reducible to f.

3 Proof of The Theorems

The proofs of Theorems 1 and 2 proceed as follows. In Section 3.1, we define *difference equations*, a discrete version of the differential equations. In Section 3.2, we show the PSPACE- and CH-hardness of difference equations with certain restrictions. In Section 3.3, we show that these classes of difference equations are simulable by certain families of differential equations given by $\mathrm{C}^{(\infty,1)}$ and $\mathrm{C}^{(\infty,k)}$ functions. In Section 3.4, we put these families of functions together into one real function to obtain the smooth differential equations stated in the theorems.

The idea of simulating difference equations with differential equations is essentially from the proof of the Lipschitz version [3]. In this paper we focus on the structure of difference equations to analyze precisely the effect of smoothness assumptions. Consequently we show differential equations given by $\mathrm{C}^{(\infty,k)}$ functions can simulate difference equations of restricted *height*, and this leads to the proof of CH-hardness.

3.1 Difference Equations

In this section, we define difference equations, a discrete version of differential equations, and show the PSPACE- and CH-hardness of families of difference equations with 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 $T \in [Q]$(Figure 3.1),

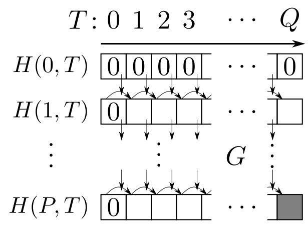

Fig. 3.1. The solution H of the difference equation given by G

$$H(i, 0) = H(0, T) = 0, \tag{3.1}$$

$$H(i + 1, T + 1) - H(i + 1, T) = G(i, T, H(i, T)). \tag{3.2}$$

We call P, Q and R the *height*, *width*, *cell size* of the difference equation. The equations (3.1) and (3.2) are similar to the initial condition $h(0) = 0$ and the equation $Dh(t) = g(t, h(t))$ in (1.1). In Section 3.3, we will simulate difference equations by differential equations using this similarity.

We use a family of difference equations as a computing system by interpreting the value of the bottom right cell (the gray cell in Figure 3.1) 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 solution H_u of the difference equation given by G_u exists and $H_u(P_u, Q_u) = L(u)$. A family $(G_u)_u$ is *uniform* if the height and width and cell size of G_u are polynomial-time computable from u and $G_u(i, T, Y)$ is polynomial-time computable from (u, i, T, Y). Note that height, width and cell size of a uniform G_u is bounded by $2^{p(|u|)}$ where p is some polynomial. A family $(G_u)_u$ has *polynomial height* if the height P_u is bounded by some polynomial $p(|u|)$. A family $(G_u)_u$ has *logarithmic height* if the height P_u is bounded by $c \log |u| + d$ with some constants c and d. In terms of these definition, [3, Lemma 4.7], proved by Kawamura to show the PSPACE-hardness of the Lipschitz version, can be written in the following form:

Lemma 7. *There exists a* PSPACE*-hard language L that is recognized by some uniform family of functions with polynomial height*[5]*.*

Kawamura obtained the result in the third row in Table 1.1 by simulating the difference equations of Lemma 7 by Lipschitz-continuous differential equations. Likewise, Theorem 1 follows from Lemma 7, by a modified construction that keeps the function in class $\mathrm{C}^{(\infty,1)}$ (Sections 3.3 and 3.4).

We show further that $\mathrm{C}^{(\infty,k)}$ functions, for any k, can simulate difference equations restricted to have logarithmic height (Sections 3.3 and 3.4). Theorem 2 follows from this simulation and the following lemma.

[5] The language class recognized by difference equations is closed under Karp reduction and the language class recognized by uniform families with polynomial height coincides PSPACE.

Lemma 8. *There exists a* CH*-hard language* L *such that it is recognized by some uniform family of functions with logarithmic height.*

The definition of the counting hierarchy CH, its connection to difference equations and the proof of Lemma 8 will be presented in Section 3.2.

3.2 The Counting Hierarchy and Difference Equations of Logarithmic Height

The polynomial hierarchy PH is defined using non-deterministic polynomial-time oracle Turing machines:

$$\Sigma_0^p = \mathsf{P}, \qquad \Sigma_{n+1}^p = \mathsf{NP}^{\Sigma_n^p}, \qquad \mathsf{PH} = \bigcup_n \Sigma_n^p. \tag{3.3}$$

In the same way, the counting hierarchy CH [13] is defined using probabilistic polynomial-time oracle Turing machines[6]:

$$\mathsf{C}_0\mathsf{P} = \mathsf{P}, \qquad \mathsf{C}_{n+1}\mathsf{P} = \mathsf{PP}^{\mathsf{C}_n\mathsf{P}}, \qquad \mathsf{CH} = \bigcup_n \mathsf{C}_n\mathsf{P}. \tag{3.4}$$

It is known that $\mathsf{PH} \subseteq \mathsf{CH} \subseteq \mathsf{PSPACE}$, but we do not know whether $\mathsf{PH} = \mathsf{PSPACE}$.

Each level of the counting hierarchy has a complete problem defined as follows. For every formula $\phi(X)$ with the list X of l free propositional variables, we write

$$\mathsf{C}^m X\phi(X) \longleftrightarrow \sum_{X\in\{0,1\}^l} \phi(X) \geq m, \tag{3.5}$$

where $\phi(X)$ is identified with the function $\phi\colon \{0,1\}^l \to \{0,1\}$ such that $\phi(X) = 1$ if and only if $\phi(X)$ is true. This "counting quantifier" C^m generalizes the usual quantifiers $\exists$ and $\forall$, because $\mathsf{C}^1 = \exists$ and $\mathsf{C}^{2^l} = \forall$. For lists $X_1, \ldots, X_n$ of variables and a formula $\phi(X_1, \ldots, X_n)$ with all free variables listed, we define

$$\langle \phi(X_1, \ldots, X_n), m_1, \ldots, m_n \rangle \in \mathsf{C}_n B_{be} \longleftrightarrow \mathsf{C}^{m_1} X_1 \cdots \mathsf{C}^{m_n} X_n \phi(X_1, \ldots, X_n). \tag{3.6}$$

Lemma 9 ([13, Theorem 7]). *For every* $n \geq 1$*, the problem* $\mathsf{C}_n B_{be}$ *is* $\mathsf{C}_n\mathsf{P}$*-complete.*

We define a problem $\mathsf{C}_{\log} B_{be}$ by

$$\langle 0^{2^n}, u \rangle \in \mathsf{C}_{\log} B_{be} \longleftrightarrow u \in \mathsf{C}_n B_{be}. \tag{3.7}$$

We show that $\mathsf{C}_{\log} B_{be}$ is CH-hard and recognized by a logarithmic-height uniform function family, as required in Lemma 8.

---

[6] This characterization, introduced by Torán in [12], is different from Wagner's original one.

*Proof (*Proof of Lemma 8*).* First we prove that $\mathsf{C}_{\log}B_{be}$ is CH-hard. For each problem A in CH, there is a constant n such that $A \in \mathsf{C}_n\mathsf{P}$. From Lemma 9, for each $u \in \{0,1\}^*$ there is a polynomial-time function f_n such that $u \in A \leftrightarrow f_n(u) \in \mathsf{C}_n B_{be}$. So

$$u \in A \longleftrightarrow \langle 0^{2^n}, f_n(u)\rangle \in \mathsf{C}_{\log}B_{be}. \tag{3.8}$$

Since $\langle 0^{2^n}, f_n(\cdot)\rangle$ is polynomial time computable, A is reducible to $\mathsf{C}_{\log}B_{be}$.

Next we construct a logarithmic-height uniform function family $(G_u)_u$ recognizing $\mathsf{C}_{\log}B_{be}$. Let $u = \langle 0^{2^n}, \langle \phi(X_1,\dots,X_n), m_1,\dots,m_n\rangle\rangle$, where $n, m_1,\dots,m_n$ are nonnegative integers and ϕ is a formula. (If u is not of this form, then $u \notin \mathsf{C}_{\log}B_{be}$.)

We write $l_i = |X_i|$ and $s_i = i + \sum_{j=1}^{i} l_j$. For each $i \in \{0,\dots,n\}$ and $Y_{i+1} \in \{0,1\}^{l_{i+1}},\dots,Y_n \in \{0,1\}^{l_n}$, we write $\phi_i(Y_{i+1},\dots,Y_n)$ for the truth value of the subformula $\mathsf{C}^{m_i}X_i \cdots \mathsf{C}^{m_1}X_1\phi(X_1,\dots,X_i,Y_{i+1},\dots,Y_n)$, so that $\phi_0 = \phi$ and $\phi_n() = \mathsf{C}_{\log}B_{be}(u)$. We regard the quantifier C^m as a function from $\mathbf{N}$ to $\{0,1\}$:

$$C^m(x) = \begin{cases} 1 & \text{if } x \geq m, \\ 0 & \text{if } x < m. \end{cases} \tag{3.9}$$

Thus,

$$\phi_{i+1}(Y_{i+2},\dots,Y_n) = C^{m_{i+1}}\left(\sum_{X_{i+1}\in\{0,1\}^{l_i}} \phi_i(X_{i+1},Y_{i+2},\dots,Y_n)\right). \tag{3.10}$$

For $T \in \mathbf{N}$, we write T_i for the ith digit of T written in binary, 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

$$G_u(0,T,Y) = (-1)^{T_{s_1}}\phi(T_{[1,s_1]}, T_{[s_1+1,s_2]},\dots,T_{[s_{n-1}+1,s_n]}), \tag{3.11}$$

and for $i \neq 0$, we define

$$G_u(i,T,Y) = \begin{cases} (-1)^{T_{s_i+1}} C^{m_i}(Y) & \text{if } T_{[1,s_i+1]} = 10\cdots0, \\ 0 & \text{otherwise.} \end{cases} \tag{3.12}$$

Define H_u from G_u by (3.1) and (3.2).

We prove by induction on i that $H_u(i,T) \in [2^{l_i}]$ for all T, and that

$$G_u(i,T,H_u(i,V)) = (-1)^{V_{s_{i+1}}}\phi_i(V_{[s_i+1,s_{i+1}]},\dots,V_{[s_{n-1}+1,s_n]}) \tag{3.13}$$

if $V_{[1,s_i+1]} = 10\cdots0$ (otherwise it is immediate from the definition that $G_u(i,V,H_u(n,V)) = 0$).

For $i = 0$, the claims follows from (3.11). For the induction step, assume (3.13). We have

$$H_u(i+1,T) = \sum_{V=0}^{T-1} G_u(i,V,H_u(i,V)). \tag{3.14}$$

Since the assumption (3.13) implies that flipping the bit $V_{s_{i+1}}$ of any V reverses the sign of $G_u(i, V, H_u(i, V))$, most of the summands in (3.14) cancel out. The terms that is can survive satisfy that $V_{[1,s_i+1]} = 10\cdots0$ and that V is between $\overline{T_{s_n}\dots T_{s_{i+1}+1}00\dots0}$ and $\overline{T_{s_n}\dots T_{s_{i+1}+1}01\dots1}$, where we write $\overline{U}$ for the number represented by string U in binary. Since these terms are 0 or 1, $H_u(i+1, T) \in [2^{l_i}]$. Then if $T_{[1,s_{i+1}+1]} = 10\cdots0$,

$$H_u(i+1, T) = \sum_{X\in\{0,1\}^{l_i}} \phi_i(X, T_{[s_{i+1}+1,s_{i+2}]}, \dots, T_{[s_{n-1}+1,s_n]}). \tag{3.15}$$

By this equation and (3.10),

$$\begin{aligned} G_u(i+1, T, H_u(i+1, T)) &= (-1)^{T_{s_{i+2}}} C^{m_{i+1}}(H_u(i+1, T)) \\ &= (-1)^{T_{s_{i+2}}} \phi_{i+1}(T_{[s_{i+1}+1,s_{i+2}]}, \dots, T_{[s_{n-1}+1,s_n]}), \end{aligned} \tag{3.16}$$

completing the induction steps.

By substituting n for i and 2^{s_n} for T in (3.13), we get $G_u(n, 2^{s_n}, H_u(n, 2^{s_n})) = \phi_n() = \mathsf{C}_{\log}B_{be}(u)$. Hence $H_u(n+1, 2^{s_n}+1) = \mathsf{C}_{\log}B_{be}(u)$.

We show that $(G_u)_u$ is uniform and has logarithmic height. The height $n+1$, the width $2^{s_n}+1$, and the cell size $2^{|u|}$ of G_u are polynomial-time computable from u, and $n+1 \le \log(|0^{2^n}|)+1 \le \log|u|+1$. □

The language class recognized by uniform function families with i rows contains $\mathsf{C}_i\mathsf{P}$ (the ith level of the counting hierarchy) and is contained in $\mathsf{C}_{i+1}\mathsf{P}$. While the class $\mathsf{C}_i\mathsf{P}$ is defined by (3.4) using oracle Turing machines, it is also characterized as those languages Karp-reducible to C_iB_{be}, or as those accepted by a polynomial-time alternating Turing machine extended with "threshold states" and having at most i alternations. Likewise, the language class accepted by uniform function families of logarithmic height coincided with languages Karp-reducible to $\mathsf{C}_{\log}B_{be}$ and with those accepted by an extended alternating Turing machine with logarithmic alternations. Since this class contains CH, we only state CH-hard in Lemma 10 and Theorem 2, but it is not known such class how hard the class is between CH and PSPACE.

3.3 Families of Real Functions Simulating Difference Equations

We show that certain families of smooth differential equations can simulate PSPACE-hard or CH-hard difference equations stated in previous section.

Before stating Lemma 10 and Lemma 11, we extend the definition of polynomial-time computability of real function to 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 polynomials* 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) $g_u : [0,1] \times [-1,1] \to \mathbf{R}$*,* $h_u : [0,1] \to [-1,1]$*;*
(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* (∞, k)*-times continuously differentiable;*
(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* $j \in \{0, \ldots, k\}$*;*
(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* $k = 1$*.*

We will prove Lemma 10 using Lemma 8 as follows. Let a function family $(G_u)_u$ be as in Lemma 8, and let $(H_u)_u$ be the solution of the difference equation given by $(G_u)_u$. We construct h_u and g_u from H_u and G_u such that $h_u(T/2^{q(|u|)}) = \sum_{i=0}^{p(|u|)} H_u(i, T)/B^{d_u(i)}$ for each $T = 0, \ldots, 2^{q(|u|)}$ 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 can prove Lemma 11 from Lemma 7 in the same way.

In Lemma 10, we have the new conditions (iii)–(v) about the smoothness and the derivatives of g_u that were not present in [3, Lemma 4.1]. To satisfy these conditions, we construct g_u using the smooth function f in following lemma.

Lemma 12 ([7, Lemma 3.6]). *There exist a polynomial-time function* $f \in \mathrm{C}^\infty[0,1]$ *and a polynomial* s *such that*

(i) $f(0) = 0$ *and* $f(1) = 1$*;*
(ii) $D^n f(0) = D^n f(1) = 0$ *for all* $n \geq 1$*;*
(iii) f *is strictly increasing;*
(iv) $D^n f$ *is polynomial-time computable for all* $n \geq 1$*;*
(v) $|D^n f| \leq s(n)$ *for all* $n \geq 1$*.*

Although the existence of the polynomial s satisfying the condition (v) is not stated in [7, Lemma 3.6], it can be shown easily.

We only prove Lemma 10 here and omit the analogous and easier proof of Lemma 11.

Proof (Proof of Lemma 10). Let L and $(G_u)_u$ be as in Lemma 8, and let a function family $(H_u)_u$ be the solution of the difference equation given by $(G_u)_u$.

By a similar argument to the beginning of the proof of [3, Lemma 4.1], we may assume that there exist polynomial-time functions p, j_u and polynomials q,

r satisfying the following properties:

$$G_u \colon [p(|u|)] \times [2^{q(|u|)}] \times [2^{r(|u|)}] \to \{-1, 0, 1\}, \tag{3.17}$$

$$H_u(i, 2^{q(|u|)}) = \begin{cases} L(u) & \text{if } i = p(|u|), \\ 0 & \text{if } i < p(|u|), \end{cases} \tag{3.18}$$

$$G_u(i, T, Y) \neq 0 \to i = j_u(T). \tag{3.19}$$

Since G_u has logarithmic height, there exists a polynomial σ such that $(k+1)^{p(x)} \leq \sigma(x)$

We construct the families of real functions $(g_u)_u$ and $(h_u)_u$ simulating 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 function $d_u \colon [p(|u|)+1] \to \mathbf{N}$ are defined by

$$B = 2^{\gamma(|u|)+r(|u|)+s(k)+k+3}, \qquad d_u(i) = \begin{cases} \sigma(|u|) & \text{if } i = p(|u|), \\ (k+1)^i & \text{if } i < p(|u|). \end{cases} \tag{3.20}$$

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 $\delta_{u,Y} \colon [0,1] \to \mathbf{R}$, $g_u \colon [0,1] \times [-1,1] \to \mathbf{R}$ and $h_u \colon [0,1] \to [-1,1]$ by

$$\delta_{u,Y}(t) = \frac{2^{q(|u|)} Df(\theta)}{B^{d_u(j_u(T)+1)}} G_u\bigl(j_u(T), T, Y \bmod 2^{r(|u|)}\bigr), \tag{3.21}$$

$$g_u(t,y) = \begin{cases} \delta_{u,Y}(t) & \text{if } \eta \leq \frac{1}{4}, \\ (1 - f(\frac{4\eta-1}{2}))\delta_{u,Y}(t) + f(\frac{4\eta-1}{2})\delta_{u,Y+1}(t) & \text{if } \eta > \frac{1}{4}, \end{cases} \tag{3.22}$$

$$h_u(t) = \sum_{i=0}^{p(|u|)} \frac{H_u(i,T)}{B^{d_u(i)}} + \frac{f(\theta)}{B^{d_u(j_u(T)+1)}} G_u\bigl(j_u(T), T, H_u(j_u(T), T)\bigr). \tag{3.23}$$

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 5. The condition (i) is immediate from (3.22) and the condition (ii) is verified by the same argument as [3, Lemma 4.1].

We prove that the condition (iii) holds, which states that $g_u \in \mathrm{C}^{(\infty,k)}$. This condition follows from that $D_2^j D_1^i g_u$ exists and is continuous for each $i \in \mathbf{N}$ and $j \in \{0, \dots, k\}$. We can show it by induction on i and j. For $j = i = 0$, it follows immediately from the definition of g_u. For $i \neq 0$ and $j = 0$, assuming $g_u \in \mathrm{C}^{(i-1,0)}$ by the induction hypothesis, we get

$$D^i \delta_{u,Y}(t) = \frac{2^{(i+1)q(|u|)} D^{i+1} f(\theta)}{B^{d_u(j_u(T)+1)}} G_u\bigl(j_u(T),\ T,\ Y \bmod 2^{r(|u|)}\bigr), N \tag{3.24}$$

$$D_1^i g_u(t,y) = \begin{cases} D^i \delta_{u,Y}(t) & \text{if } \eta \leq \frac{1}{4}, \\ \bigl(1 - f\left(\frac{4\eta-1}{2}\right)\bigr) D^i \delta_{u,Y}(t) + f\left(\frac{4\eta-1}{2}\right) D^i \delta_{u,Y+1}(t) & \text{if } \frac{1}{4} < \eta. \end{cases} \tag{3.25}$$

And it is continuous by the definition of f (Lemma 12). For $i \neq 0$ and $j \neq 0$, assuming $g_u \in \mathrm{C}^{(i,j-1)}$ by the induction hypothesis, we get

$$D_2^j D_1^i g_u(t,y) = \begin{cases} 0 & \text{if } -\frac{1}{4} < \eta < \frac{1}{4}, \\ (2B^{d_u(j_u(T))})^j D^j f(\frac{4\eta-1}{2})(D^i \delta_{u,Y+1}(t) - D^i \delta_{u,Y}(t)) & \text{if } \frac{1}{4} < \eta < \frac{3}{4}. \end{cases} \tag{3.26}$$

and it is continuous. Here we complete the induction step.

Substituting $t = 0, 1$ ($\theta = 0$) into (3.25), we get $D^{(i,0)} g_u(0,y) = D^{(i,0)} g_u(1,y) = 0$, so the condition (iv) holds.

We show that the condition (v) holds with $\mu(x,y) = (x+1)q(y) + s(x+1)$. Note that μ is a polynomial and independent of k and γ. Since $|D^i \delta_{u,Y}(t)| \leq 2^{(i+1)q(|u|)+s(i+1)} B^{-d_u(j_u(|u|)+1)}$ by (3.21), for all $i \in \mathbf{N}$ and $j \in \{0, \dots, k\}$, we have

$$|D^{(i,j)} g_u| \leq 2^k B^{k \cdot j_u(T)} 2^{s(k)} \cdot 2 \cdot \frac{2^{(i+1)q(|u|)+s(i+1)}}{B^{d_u(j_u(|u|)+1)}} \leq \frac{2^{\mu(i,|u|)+s(k)+k+1}}{B} \leq 2^{\mu(i,|u|)-\gamma(|u|)} \tag{3.27}$$

by (3.25), (3.26) and our choice of B.

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

$$h_u(1) = \frac{H_u(p(|u|), 2^{q(|u|)})}{B^{d_u(p(|u|))}} = \frac{L(u)}{2^{\sigma(|u|) \cdot (\gamma(|u|)+r(|u|)+s(k)+k+3)}} = 2^{-\rho(|u|)} L(u). \tag{3.28}$$

$\square$

To prove Lemma 11, let L and $(G_u)_u$ be as Lemma 7, and let $(H_u)_u$ be the solution of the difference equation given by $(G_u)_u$. Define $(g_u)_u$ and $(h_u)_u$ as (3.22) and (3.23) with $d_u(i) = i$. It is shown in the same way as above that they meet all the conditions stated in Lemma 11.

3.4 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 (Proof of Theorem 2). Let L and μ be as Lemma 11. Define

$$\lambda(x) = 2x + 2, \qquad \gamma(x) = x\mu(x,x) + x\lambda(x). \tag{3.29}$$

and for each u

$$\Lambda_u = 2^{\lambda(|u|)}, \qquad c_u = 1 - \frac{1}{2^{|u|}} + \frac{2\bar{u}+1}{\Lambda_u}, \qquad l_u^{\mp} = c_u \mp \frac{1}{\Lambda_u} \tag{3.30}$$

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

$$g\left(l_u^{\mp} \pm \frac{t}{\Lambda_u}, \frac{y}{\Lambda_u}\right) = \begin{cases} \pm \sum_{l=0}^{k} \dfrac{D^{(0,l)} g_u(t,1)}{l!} (y-1)^l & \text{if } 1 < y, \\ \pm g_u(t,y) & \text{if } -1 \leq y \leq 1, \\ \pm \sum_{l=0}^{k} \dfrac{D^{(0,l)} g_u(t,-1)}{l!} (y+1)^l & \text{if } 1 < y, \end{cases} \tag{3.31}$$

$$h\left(l_u^{\mp} \pm \frac{t}{\Lambda_u}\right) = \frac{h_u(t)}{\Lambda_u} \tag{3.32}$$

for each string u and $t \in [0,1)$, $y \in [-1,1]$. Let $D_1^i g(1,y) = 0$ and $h(1) = 0$ for any $y \in [-1,1]$ and $i \in \mathbf{N}$.

It can be shown similarly to the Lipschitz version [3, Theorem 3.2] that g and h satisfy (1.1) and g is polynomial-time computable. Here we only prove that $g \in \mathrm{C}^{(\infty,k)}[[0,1] \times [-1,1]]$.

We show that $D_1^i D_2^j g$ exists and is continuous for each $i \in \mathbf{N}$ and $j \in \{0, \ldots, k\}$ by induction on i and j. By the induction hypothesis, we can assume that $D_1^n D_2^m g$ exists and is continuous for all $(n,m) \in \{0,\ldots,i\} \times \{0,\ldots,j\}$ except the case $(n,m) = (i,j)$.

We define for each $l_u^{\mp} \pm t/\Lambda_u \in [0,1)$ and $y/\Lambda_u \in [-1,1]$,

$$D_1^i D_2^j g\left(l_u^{\mp} \pm \frac{t}{\Lambda_u}, \frac{y}{\Lambda_u}\right) = \begin{cases} \pm \Lambda_u^{i+j} \sum_{l=j}^{k} \frac{D^{(i,l)} g_u(t,1)}{(l-j)!} (y-1)^l & \text{if } y < -1, \\ \pm \Lambda_u^{i+j} D^{(i,j)} g_u(t,y) & \text{if } -1 \leq y \leq 1, \\ \pm \Lambda_u^{i+j} \sum_{l=j}^{k} \frac{D^{(i,l)} g_u(t,-1)}{(l-j)!} (y+1)^l & \text{if } 1 < y, \end{cases} \tag{3.33}$$

and $D_1^i D_2^j g(0,y) = 0$. For $i = j = 0$, it is equal to the definition of g. We get (3.33) for $i = 0$ and $j \neq 0$ by differentiating $D_2^{j-1} g$ with respect to the second variable, and for $i \neq 0$ and $j \neq 0$ by differentiating $D_1^{i-1} D_2^j g$ with respect to the first variable.

The continuousness of (3.33) on $[0,1) \times [-1,1]$ follows from Lemma 10 (iv). We verify that $D_1^i D_2^j g$ is continuous when the first variable moves arbitrarily close to 1, i.e., $|u|$ goes to infinity. From Lemma 10 (v), we get

$$\begin{aligned} \left| D_1^i D_2^j g\left(l_u^{\mp} \pm \frac{t}{\Lambda_u}, \frac{y}{\Lambda_u}\right)\right| &\leq \Lambda_u^{i+j} \sum_{l=j}^{k} |D^{(i,l)} g_u| (\Lambda_u + 1)^l \\ &\leq \Lambda_u^{i+j} \cdot k \cdot 2^{\mu(i,|u|) - \gamma(|u|)} \cdot (2\Lambda_u)^k \\ &\leq 2^{(i+j+k)\lambda(|u|) + 2k + \mu(i,|u|) - \gamma(|u|)}. \end{aligned} \tag{3.34}$$

By our choice of γ, it converges to 0 when $|u| \to \infty$. Hence $\lim_{t \to 1-0} D_1^i D_2^j g(t,y) = D_1^i D_2^j g(1,y) = 0$. Here we complete the induction steps. $\square$

4　演算子の計算量

定理 1, 2 はいづれも関数 g を多項式時間計算可能と仮定した上で解 h の計算量について述べている. しかし微分方程式を「解く」困難さ, すなわち与えられた g から h を求める演算子の計算量は如何であろうか. この問に答えるにはまず実関数を実関数へ写す演算子の計算量を定義することを要する.

実数を入出力する関数の計算量を論ずるには, 実数を文字列関数で表した. 即ち $\mathbf{R}$ の各元の名として文字列関数を使ったのであり, その対応を $\mathbf{R}$ の表現という. 同じように実関数を入出力する演算子の計算量を論ずるには, 実関数を文字列関数で表す. つまり連続な実関数 $h\colon [0,1] \to \mathbf{R}$ の空間 $\mathrm{C}[0,1]$ や, Lipschitz 連続な実関数 $g\colon [0,1] \times [-1,1] \to \mathbf{R}$ の空間 $\mathrm{C_L}[[0,1] \times [-1,1]]$ について, 表現を指定すればよい. 演算子の計算可能性や計算量はその表現に依ることになるが, ここでは [5] に従い, $\mathrm{C}[0,1]$ の表現として $\delta_\square$ を, $\mathrm{C_L}[[0,1] \times [-1,1]]$ の表現として $\delta_{\square\mathrm{L}}$ をそれぞれ用いる. $\delta_\square$ は関数空間 $\mathrm{C}[0,1]$ の表現として或る意味で自然な唯一のものであることが判っており [4], また $\delta_{\square\mathrm{L}}$ は $\delta_\square$ に Lipschitz 定数の情報を附加した表現である.

これらの表現では使われる文字列関数の長さが有界でないため, 神託機械の時間・空間を測る方法を二階多項式を使って拡張し, これに基いて多項式空間 **FPSPACE** などの計算量クラスや, 多項式時間 Weihrauch 帰着 $\leq_\mathrm{W}$ などの帰着, その下での困難性を定義する [5]. この枠組を上述の実関数の表現と組合せることで, 本稿の結果も以下の如く構成的な形で述べることができる.

実関数 $g \in \mathrm{C_L}[[0,1] \times [-1,1]]$ を, (1.1) の解 $h \in \mathrm{C}[0,1]$ に対応させる演算子 ODE を考える. ODE は $\mathrm{C_L}[[0,1] \times [-1,1]]$ から $\mathrm{C}[0,1]$ への部分写像である. [5, 定理 4.9] では表 1.1 第三行の証明を構成的に書き直すことで, ODE が $(\delta_{\square\mathrm{L}}, \delta_\square)$-**FPSPACE**-$\leq_\mathrm{W}$ 完全であることが示された. 本稿の定理 1 も同じように構成的に書き直すことができる. 即ち ODE を (∞, k) 回連続微分可能な入力に制限したものを ODE_k と書くと,

Theorem 13. ODE_1 は $(\delta_{\square\mathrm{L}}, \delta_\square)$-**FPSPACE**-$\leq_\mathrm{W}$ 完全.

これを示すには, 定理 1 の証明において関数の構成に使われた情報が入力から容易に得られることを確かめればよく, 新たな技巧を要しないから詳細は省略する. この構成的な主張は非構成的な主張よりも強いものであり [5, 補題 3.7, 3.8], 定理 1 は定理 13 の系として従う.

なお定理 2 も同様に構成的な形で成立ち, 各 $k \in \mathbf{N}$ について ODE_k は $(\delta_{\square\mathrm{L}}, \delta_\square)$-**CH**-$\leq_\mathrm{W}$ 困難であるが, この [5] の枠組における **CH** を定義するには相対化の扱いについて今少しの議論を要するので別稿で扱う.

References

1. E.A. Coddington and N. Levinson. *Theory of Ordinary Differential Equations*. McGraw-Hill, 1955.
2. A. Kawamura. Complexity of initial value problems, 2010. To appear in *Fields Institute Communications*.
3. A. Kawamura. Lipschitz continuous ordinary differential equations are polynomial-space complete. *Computational Complexity*, 19(2):305–332, 2010.

4. A. Kawamura. On function space representations and polynomial-time computability. Dagstuhl Seminar 11411: Computing with Infinite Data, 2011. http://www-imai.is.s.u-tokyo.ac.jp/~kawamura/dagstuhl.pdf.
5. A. Kawamura and S. Cook. Complexity theory for operators in analysis. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, pages 495–502. ACM, 2010.
6. K.I. Ko. On the computational complexity of ordinary differential equations. *Information and Control*, 58(1-3):157–194, 1983.
7. K.I. Ko. *Complexity Theory of Real Functions*. Birkhäuser Boston, 1991.
8. K.I. Ko and H. Friedman. Computing power series in polynomial time. *Advances in Applied Mathematics*, 9(1):40–50, 1988.
9. W. Miller. Recursive function theory and numerical analysis. *Journal of Computer and System Sciences*, 4(5):465–472, 1970.
10. N.T. Müller. Uniform computational complexity of Taylor series. *Automata, Languages and Programming*, pages 435–444, 1987.
11. 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.
12. J. Torán. Complexity classes defined by counting quantifiers. *Journal of the ACM (JACM)*, 38(3):752–773, 1991.
13. K.W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23(3):325–356, 1986.
14. Klaus Weihrauch. *Computable Analysis: An Introduction*. Texts in Theoretical Computer Science. Springer, 2000.