2011 年度冬の LA シンポジウム [15]

滑らかな常微分方程式の計算量

太田浩行* 河村彰星†
マルチン・ツィーグラー‡ カルステン・レースニク§

平成 24 年 1 月 31 日

概要

常微分方程式 $h(0) = 0,\ h'(t) = g(t, h(t))$ の解 h の計算量と, 関数 g の計算量及び制限の関係は, 常微分方程式を数値的に解くことの本質的な難しさを表しているとして調べられている. 本稿では河村が 2010 年の論文の中で Lipschitz 条件を満たす多項式時間計算可能な関数 g の常微分方程式の解が **PSPACE** 困難たりうるという結果を示すために用いた手法を, 微分可能な g に拡張する. これにより多項式時間計算可能で 1 回連続微分可能な関数の常微分方程式が, **PSPACE** 困難な解を持ちうることがわかる. また任意の k について, 多項式時間計算可能で k 回微分可能な関数の常微分方程式は, 本稿で定義される計算量 **DIVP(log)** について困難な解を持ちうることを示す. **DIVP(log)** = **PSPACE** であるかどうかは未解決である.

1 導入

1.1 計算可能解析

計算可能解析 (Computable Analysis) [10] では計算可能性理論や計算量理論の視点から解析学を扱う. 「計算可能な実数」や「多項式時間計算可能な実関数」といった概念を定義し, 解析学に現れる様々な実数や実関数の本質的な難しさを分析する.

計算可能解析における「機械が実関数を計算する」ことの定義を導入する.

実関数 $f\colon \mathbf{R} \to \mathbf{R}$ を計算するといっても, 実数は有限の文字列で表されないため, 機械がその完全な値を読み書きすることはできない. そこで実数を近似値の列で表現する. 有理数の列 $(r_n)_n$ が x を表現するとは, $(r_n)_n$ が x へ速く収束すること, すなわち $|r_n - x| \leq 2^{-n}$ を満たすこととする. 数列は $n \in \mathbf{N}$ を $r_n \in \mathbf{Q}$ へ移す関数と考えることもでき, そのような関数または数列を実数の名と呼ぶ.

* 東京大学, hota@is.s.u-tokyo.ac.jp
† 東京大学
‡ Martin Ziegler, ダルムシュタット工科大学
§ Carsten Rösnick, ダルムシュタット工科大学

表 1.1　多項式時間計算可能関数 g の常微分方程式 (1.1) の解 h の計算量

制限	上界	下界
—	—	計算不可能たりうる [9]
h が g の唯一解	計算可能 [1]	任意の時間がかかりうる [5, 8]
g が Lipschitz 条件を満たす	多項式領域計算可能	**PSPACE** 困難になりうる [4]
g が $(0,1)$ 回連続微分可能	多項式領域計算可能	**PSPACE** 困難になりうる [本稿定理 1.1]
g が $(0,k)$ 回連続微分可能	多項式領域計算可能	**DIVP(log)** 困難たりうる [本稿定理 1.2]
g が解析的	多項式時間計算可能 [7, 3]	—

実関数を計算するモデルとしては神託チューリング機械を用いる. ある機械が関数 $f\colon [0,1] \to \mathbf{R}$ を計算するとは, 入力となる実数 x の名を神託として与えられ, 求める精度 n を入力として与えられたとき, 有理数 s_n で $|s_n - f(x)| \le 2^{-n}$ を満たすものを出力することとする. この神託機械の資源を制限することで, 実関数が多項式時間計算可能, 或いは多項式領域計算可能であることを定義できる (2 節).

1.2　問題と関連研究

連続実関数 $g\colon [0,1] \times \mathbf{R} \to \mathbf{R}$ に対して次の常微分方程式を考える.

$$h(0) = 0, \qquad h'(t) = g(t, h(t)) \quad (t \in [0,1]) \tag{1.1}$$

本稿では g が多項式時間計算可能であるとき, 解 h がどれほど複雑でありうるかを考える.

g に他に何の制限も設けない場合, 解 h (一般に一意でない) は計算不能たりうるため, 様々な制限のもと h の計算量が研究されている (表 1.1). この表では下に向うにつれて左列の条件が強まっている. Lipschitz 条件とは解 h が一意であるための十分条件であり, これが満たされるときには, 解 h は多項式領域計算可能であり, **PSPACE** 困難 (2 節で定義) でありうることがわかっている. つまり上界と下界が一致しているといえる (詳しくは河村 [4]). 一方で g が解析的であるとき, 解 h も解析的となり, このとき h は多項式時間計算可能である.

そこで本稿ではこの隔たりを埋めるため, 滑らかな関数, つまり微分可能な g について h の計算量がどれほどになりうるかを調べ, 以下の結果を得た.

定理 1.1. 多項式時間計算可能かつ $(\infty, 1)$ 回連続的微分可能な実関数 $g\colon [0,1] \times [-1,1] \to \mathbf{R}$ であって, 常微分方程式 (1.1) が **PSPACE** 困難な解 $h\colon [0,1] \to \mathbf{R}$ をもつものが存在する.

定理 1.2. 任意の自然数 $k \geq 2$ に対して, 多項式時間計算可能かつ (∞, k) 回連続的微分可能な実関数 $g\colon [0,1] \times [-1,1] \to \mathbf{R}$ であって, 常微分方程式 (1.1) が **DIVP(log)** 困難な解 $h\colon [0,1] \to \mathbf{R}$ をもつものが存在する.

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 1.1, 1.2 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.

DIVP(log) とは本稿 2 節で定義する計算量クラスであり, **DIVP(log)** $\subseteq$ **PSPACE** であるが, **DIVP(log)** $=$ **PSPACE** か否かはわからない.

二変数関数 g が (i,j) 階連続微分可能であるとは, 第一変数について i 回, 第二変数について j 回微分可能であり, その導関数が連続であることと定義する. この定義は一般的な多変数関数における k 階連続的微分可能の定義 (任意の k 階導関数が存在し, それらがすべて連続) とは異なる. (i,j) 回連続微分可能であれば $\min\{i,j\}$ 回連続微分可能であるため, 定理 1.1, 1.2 はそれぞれ 1 回連続微分可能, k 回連続微分可能と置き換えても成り立つ.

また定理 1.2 において任意の k に対して (∞, k) 階微分可能な関数を考えているが, 一つの関数が任意の k に対して k 階微分であることを求めているわけではない. つまり g が無限回微分可能であると制限しているわけではない. 無限回微分可能な関数に対する常微分方程式の計算量は今後の課題である.

1.3 差分方程式

定理 1.1 と定理 1.2 の証明では, まず滑らかな実関数の常微分方程式によってある種の「離散版」常微分方程式を模倣できることを示し, その離散版の方程式が **PSPACE** 困難ないし **DIVP(log)** 困難であることを示す. この節ではその離散版の方程式である「差分方程式」を定義する.

$[n] = \{0, \ldots, n-1\}$ と書く. 関数 $G\colon [P] \times [Q] \times [R] \to \{-1, 0, 1\}$ に対して, 関数 $H\colon [P+1] \times [Q+1] \to [R]$ が任意の $i \in [P]$, $T \in [Q]$ について以下を満たすとき, H を G の差分方程式の解と呼ぶ.

$$H(i,0) = H(0,T) = 0 \tag{1.2}$$

$$H(i+1, T+1) = H(i+1, T) + G(i, T, H(i,T)) \tag{1.3}$$

P, Q, R をそれぞれ行数, 列数, 欄の大きさと呼ぶ. G と H が常微分方程式の g と h に対応し, $H(i,0) = 0$ と言う条件が $h(0) = 0$, 式 (1.3) と同値である $H(i+1, T+1) - H(i+1, T) = G(i, T, H(i,T))$ と言う条件が $h'(t) = g(t, h(t))$ と対応する.

以下では文字列 u ごとに差分方程式 G_u を一つ定めた族 $(G_u)_u$ を考える. 言語 L がこの族 $(G_u)_u$ によって認識されるとは, 各 u に対して G_u の段数と列数, 解をそれぞれ P_u, Q_u, H_u と

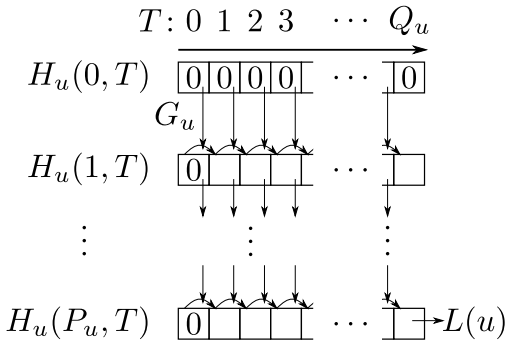

図 1.1　差分方程式と認識される言語

したとき, $H_u(P_u, Q_u) = L(u)$ をみたすこととする [表 1.3]. ここで言語 $L \subseteq \{0,1\}^*$ は関数 $L\colon \{0,1\}^* \to \{0,1\}$ と同一視し, $u \in L$ のとき $L(u) = 1$ としている.

$(G_u)_u$ が**一様**であるとは, 各 u について G_u の段数, 列数及び欄の大きさが $|u|$ の多項式の指数 $(2^{\mathrm{poly}(|u|)})$ で抑えられ, かつ与えられた (u, i, T, Y) から多項式時間で $G_u(i, T, Y)$ が計算できることと定義する.

G_u の段数がさらに $|u|$ の多項式, 対数で抑えられるとき, 族 $(G_u)_u$ はそれぞれ**多項式段**, **対数段**であるという. 多項式段, 対数段の一様関数族によって認識される言語のクラスをそれぞれ **DIVP(poly)**, **DIVP(log)** と名づける. この記法を使うと河村の論文では次が示されている.

補題 1.3 (補題 4.7. [4])**.** $\mathbf{DIVP(poly)} = \mathbf{PSPACE}$.

$\mathbf{DIVP(log)} \subseteq \mathbf{DIVP(poly)} = \mathbf{PSPACE}$ であるが, $\mathbf{DIVP(log)} = \mathbf{PSPACE}$ か否かは未解決である. また **DIVP(log)** の計算量の下限として $\sharp\mathbf{P}$ が考えられる. **DIVP(log)** が決定問題であるのに対し, $\sharp\mathbf{P}$ は関数問題であるため単純には比較できない. そこで一様な $(G_u)_u$ の 2 段の差分方程式によって定義される関数問題のクラスを考える. 最後の欄の値は $H_u(1, Q_u) = \sum_{t<Q_u} G_u(0, t, 0)$. u について多項式時間関数 $G_u(0, t, 0)$ の和になっているため, 差分方程式の関数問題クラスは数え上げ問題である $\sharp\mathbf{P}$ を含む.

2　準備

2.1　表記

自然数の集合を $\mathbf{N}$, 整数の集合を $\mathbf{Z}$, 実数の集合を $\mathbf{R}$, 有理数の集合を $\mathbf{Q}$, $\{0\}^* = \{0^n \mid n \in \mathbf{N}\}$ で表す.

$A \subset \mathbf{R}$ とする. 一変数関数 $f\colon A \to \mathbf{R}$ が i 回微分可能であるとき, その i 階導関数を $\mathcal{D}^{(i)}f$ と表記する.

二変数関数 $g\colon A \times B \to \mathbf{R}$ が (i, j) 回連続微分可能であるとき, 第一変数について i 階, 第二変数について j 階の導関数はその微分の順序によらず等しい [11]. その導関数を $\mathcal{D}^{(i,j)}g$ で表す.

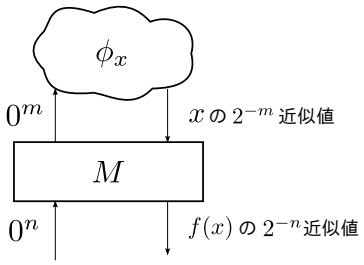

図 2.1　実関数を計算する機械

実関数 $f\colon A \to \mathbf{R}$ に対して $|f| = \sup_{x\in A} f(x)$ と書く.

2.2　実数の名

実数は有限な文字列に符号化できない. そこで文字列から文字列への関数に符号化する.

定義 2.1 (実数の名)**.** 関数 $\phi\colon \{0\}^* \to \mathbf{Z}$ が実数 $x \in [0,1]$ の名であるとは, $\phi(0^n) = \lfloor x\cdot 2^n \rfloor$ または $\phi(0^n) = \lceil x\cdot 2^n \rceil$ を満たすこと.

ここで $\lfloor\cdot\rfloor, \lceil\cdot\rceil$ とはそれぞれ整数への切り捨て関数と切り上げ関数である. つまり実質的には実数 x の名は, サイズ n の入力を受け取ると, 精度 n 桁の x の近似値を返す. 以下では ϕ の値を二進数で表すことにし, ϕ を文字列から文字列への関数として扱う.

2.3　計算可能実関数, 多項式時間実関数

実数を受け取り実数を返す関数を機械が計算するとはどういうことか定義しよう. 実数自体が関数として符号化されているため, それを読み書きする機構として, 神託チューリング機械 (以下単に機械という) を使う [図 2.3].

機械 M に, 文字列から文字列への関数 ϕ を神託として与え, 文字列 0^n を入力として与えたとき, 出力される文字列を $M^\phi(0^n)$ で表す. つまり M^ϕ をやはり文字列から文字列への関数とみる.

定義 2.2. A を $\mathbf{R}$ の有界閉区間とする. 機械 M が実関数 $f\colon A \to \mathbf{R}$ を計算するとは, 任意の実数 $x \in A$, 任意の x の名 ϕ_x に対して, M^{ϕ_x} が $f(x)$ の名であること.

A が $\mathbf{R}^2$ の有界閉領域であるときにも, 神託を二つ取る機械を考えて同様に定義する.

計算可能な実関数は Grzegorczyk によって初めて形式的に定義され, [2].

ある実関数が計算可能であるとは, その関数を計算する神託機械が存在することである. 同様に, ある実関数が多項式時間計算可能であるとは, その関数を計算する多項式時間神託機械が存在することである.

文字列 u で添字づけられた実関数 $f_u\colon A \to \mathbf{R}$ の族 $(f_u)_u$ を機械 M が計算するとは, 任意の実数 $x \in A$, 任意の x の名 ϕ_x に対して, 文字列 v を $M^{\phi_x}(u,v)$ へ移す関数が, $f_u(x)$ の名であるこ

とをいう. 実関数族が多項式時間計算可能であるとは, その実関数族を計算する多項式時間神託機械が存在することである.

神託機械 M で f を計算するとき, 求める精度 n に対して, x の近似値に必要な精度 m が定まるため, 計算可能な関数は連続である. また n と m の対応関係と有理数における近似値を与えることで, 計算可能実関数や多項式時間計算可能実関数に対して, 神託機械を用いない同値な特徴付けが可能である.

補題 2.3. 実関数 $f\colon [0,1] \to \mathbf{R}$ に対して, $\phi_f\colon (\mathbf{Q} \cap [0,1]) \times \{0\}^* \to \mathbf{Q}$, $m_f\colon \mathbf{N} \to \mathbf{N}$ は

$$|\phi_f(d, 0^n) - f(d)| \leq 2^{-n} \qquad (d \in (\mathbf{Q} \cap [0,1]), \quad n \in \mathbf{N}) \tag{2.1}$$

$$|x - y| \leq 2^{-p_f(m)} \Rightarrow |f(x) - f(y)| \leq 2^{-m} \qquad (x, y \in [0,1], \quad m \in \mathbf{N}) \tag{2.2}$$

をみたす関数とする.

- f が計算可能であることは, 計算可能な ϕ_f, m_f が存在することと同値である.
- f が多項式時間計算可能であることは, 多項式時間計算可能な ϕ_f, 多項式 m_f が存在することと同値である.

2.4 困難性

実関数の計算量の下界を述べるために, 困難性を定義する.

まず実関数に言語が還元することを定義する. 言語 $L\colon \{0,1\}^* \to \{0,1\}$ が実関数 $f\colon [0,1] \to \mathbf{R}$ に多項式時間還元可能であるとは, f を計算する機械を使って $L(u)$ を多項式時間で計算可能であることである. つまり f を計算する機械があるとしたとき, 入力 u に対して, 精度 n をこの機械に与え, ある実数 x_u の神託を模倣し, $f(x_u)$ の n 桁近似値から, u が L に属するか否かを多項式時間で計算可能であることである [図 2.4]. 厳密には以下のように定義する.

定義 2.4 (多項式時間還元可能)**.** 言語 L が実関数 $f\colon [0,1] \to \mathbf{R}$ に多項式時間還元可能であるとは, 多項式時間計算可能な関数 R, S, T が存在し, 任意の文字列 u に対して以下を満たすことをいう.

- $S(u, \cdot)$ はある実数 x_u の名
- $f(x_u)$ の任意の名 ϕ に対して

$$L(u) = R(u, \phi(T(u))).$$

以下単に言語が実関数に還元可能といった場合, 多項式時間還元可能を指す. 計算量 C に対して, 関数 f が C 困難であるとは, C に属する任意の言語が f に還元可能であることと定義する.

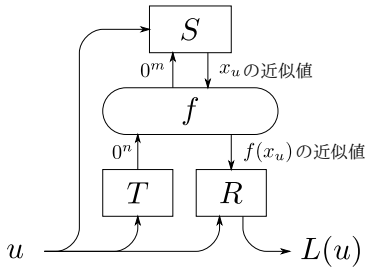

図 2.2　言語 L から関数 f への還元

3　1 回連続微分可能関数と常微分方程式

この節では定理 1.1, つまり $(\infty, 1)$ 階連続微分可能な関数の常微分方程式の解は **PSPACE** 困難でありうることを示す. ただし紙面の都合上, 詳細な証明は省き, 証明の概略を説明するに止める.

証明の流れとして, まず補題 1.3 から補題 3.1 を示したあと, 補題 3.1 から定理 1.1 を示す. 補題 3.1 では補題 1.3 の関数族 $(G_u)_u, (H_u)_u$ を模倣する実関数族 $(g_u)_u, (h_u)_u$ の存在を述べている. 定理 1.1 の証明では $(g_u)_u, (h_u)_u$ から求める g, h を構成する.

3.1　多項式段一様関数族の差分方程式と PSPACE

任意の **PSPACE** に含まれる言語 L について, その差分方程式が L を認識する多項式段一様な関数族 $(G_u)_u$ が存在すること (補題 1.3) は河村の論文のなかで示されているが, その証明の概略を示す.

任意の L ではなく **PSPACE** 完全な言語である QBF を認識する $(G_u)_u$ を構成することにより, 任意の L を認識する多項式段一様な関数族 $(G_u)_u$ が存在することを示す. ここで QBF とは, 文字列 u を $\psi = Q_1 x_1 \cdots Q_n x_n \phi(x_1, \ldots, x_n)$ と解釈したとき $u \in \mathsf{QBF} \Leftrightarrow \psi = T$ によって定義される言語である. ただし Q_i は $\exists$ または $\forall$, $\phi(x_1, \ldots, x_n)$ は x_i 以外の変数を含まない論理式とする.

論理式 $\psi = Q_1 x_1 \cdots Q_n x_n \phi(x_1, \ldots, x_n)$ の値を $\vee, \wedge$ によってラベル付された二分木によって計算することを考える. 量化子 $Q_1 x_1$ を除き x_1 を T と F に置き換えた式をそれぞれ $\psi_T = Q_2 x_2 \cdots Q_n x_n \phi(T, x_2, \ldots, x_n)$, $\psi_F = Q_2 x_2 \cdots Q_n x_n \phi(F, x_2, \ldots, x_n)$ と置くと $Q_1 = \forall$ ならば $\psi = \psi_T \wedge \psi_F$, $Q_1 = \exists$ ならば $\psi = \psi_T \vee \psi_F$. つまり変数の 1 つ少ない 2 つの論理式と量化子によってもとの論理式の値も決まる. これを再帰的に繰り返すことで ψ は計算可能であり, それは深さ n の二分木を葉から根へ値を定めていくことと同じである. この過程は二分木の深さが段数に, 幅が列数に対応する形で多項式段一様な関数による差分方程式で模倣可能であるため, QBF を認識する多項式段一様関数の差分方程式が存在する.

3.2 多項式段差分方程式を模倣する関数族

補題 3.1 を述べるため, 実関数族 $(g_u)_u$ ($g_u\colon [0,1]\times[-1,1] \to \mathbf{R}$) が多項式時間計算可能であることを定義する. ある神託機械 M が実関数族 $(g_u)_u$ を計算するとは, 実数 t, y の名 ϕ, ψ を神託として受けとり, 文字列 u, 求める精度 n を入力として受けとったとき, $|M^{\phi,\psi}(u, 0^n) - g_u(t, y)| \leq 2^{-n}$ を満たすことである. そして実関数族が多項式時間計算可能であるとは, その実関数族を計算し, $|u|$ と n の多項式時間で動作する神託機械が存在することと定義する.

補題 3.1. 任意の言語 $L \in \mathbf{PSPACE}$ に対して, 係数のみに i を含む多項式 μ_i が存在して, 任意の多項式 γ に対して, 多項式 ρ, 関数族 $(g_u)_u, (h_u)_u$ で, $(g_u)_u$ は多項式時間計算可能であり, 各二進文字列 u に対して以下を満たすものが存在する.

(i) $g_u\colon [0,1] \times [-1,1] \to \mathbf{R}$, $\quad h_u\colon [0,1] \to [-1,1]$;
(ii) h_u は g_u の常微分方程式 (1.1) の解;
(iii) g_u は $(\infty, 1)$ 階連続微分可能;
(iv) 任意の $i \in \mathbf{N}$, $y \in [-1,1]$ に対して

$$\mathcal{D}^{(i,0)} g_u(0, y) = \mathcal{D}^{(i,0)} g_u(1, y) = 0$$

(v) 任意の $i \in \mathbf{N}$ に対して

$$|\mathcal{D}^{(i,1)} g_u| \leq 2^{\mu_i(|u|) - \gamma(|u|)}, \qquad |\mathcal{D}^{(i,0)} g_u| \leq 2^{\mu_i(|u|) - \gamma(|u|)}$$

(vi) $h_u(1) = 2^{-\rho(|u|)} L(u)$.

この補題より関数族 $(g_u)_u, (h_u)_u$ で h_u は g_u の常微分方程式の解であり, $h_u(1)$ に $L(u)$ の情報を持つものの存在が示される.

条件 (iii) – (v) はすべて定理 1.1 の g が滑らかな関数とするために必要となる条件である. 詳細については定理の証明の際に説明する.

この補題の証明の基本的な流れを説明する. まず任意の言語 $L \in \mathbf{PSPACE}$ に対し, 補題 1.3 を用いて L を認識する差分方程式 $(G_u)_u$ 及びその解 $(H_u)_u$ を得る. そして各 G_u, H_u から, その差分方程式の計算過程を模倣する $(\infty, 1)$ 階連続微分可能な $g_u\colon [0,1] \times [-1,1] \to \mathbf{R}$ とその常微分方程式の解 $h_u\colon [0,1] \to \mathbf{R}$ を構成する. また $(G_u)_u$ の一様性から $(g_u)_u$ の多項式時間計算可能性を示す.

ここまでの証明の流れは基本的に, リプシッツ連続条件の場合の証明とかわらない. 違いは g_u を滑らかな関数にするため, 以下のような滑らかな多項式時間実関数 $f\colon [0,1] \to \mathbf{R}$ を用いて g_u を構成している点である.

補題 3.2 (補題 3.6. [6])**.** 以下を満たす多項式時間無限回微分可能実関数 $f\colon [0,1] \to \mathbf{R}$ が存在する.

(i) $f(0) = 0, \quad f(1) = 1$;
(ii) 任意の $n \geq 1$ で $f^{(n)}(0) = f^{(n)}(1) = 0$;
(iii) f は $[0,1]$ で単調増加;
(iv) 任意の $n \geq 1$ で $f^{(n)}$ は多項式時間実関数.

3.3 定理 1.1 の証明

補題 3.1 から得られる $(g_u)_u$ と $(h_u)_u$ から滑らかな g とその常微分方程式の解でその常微分方程式の解で **PSPACE** 困難な h を構成する. 各 $h_u(1)$ には $L(u)$ の情報が含まれるため, すべての h_u を一つの関数 h に埋め込みたい, そこで [0,1] を無限の区間に分割し, h の各文字列 u に対応する区間 $[l_u^-, c_u]$ に h_u を縮小して埋め込む. ただし次の文字列 u' の計算に影響を与えないために, h_u を定義域方向について反転したものを区間 $[c_u, l_u^+]$ に埋め込むことで影響を相殺する. つまり $h(l_u^-) = 0,\ h(c_u) = 2^{-\rho'(|u|)}L(u),\ h(l_u^+) = 0$ をみたすように $h_u(t)$ 埋め込む. ただし ρ' とは ρ に縮小率をかけたものとする. 同様に g は h が常微分方程式の解となるよう, 各文字列 u に対応する区間に g_u を縮小して埋め込む.

リプシッツ連続条件と異なる点は, g_u を構成する時点で $(\infty, 1)$ 階連続微分可能にするために, $|\mathcal{D}^{(i,0)}g_u|, |\mathcal{D}^{(i,0)}g_u|$ の大きさを制限する点である [補題 3.1 の (v)]. $t = 1$ 付近, つまり $|u| \to \infty$ のときに $\mathcal{D}^{(i,0)}g_u, \mathcal{D}^{(i,1)}g_u$ が発散しないためにこのような条件をつけている.

4 任意回連続微分可能関数と常微分方程式

この節では定理 1.2, つまり (∞, k) 階連続微分可能な関数の常微分方程式の解は **DIVP(log)** 困難でありうることを示す. ただし紙面の都合上, 詳細な証明は省き, 証明の概略を説明するに止める.

証明の流れは節 3 とほぼ同じく, まず補題 3.1 を示したあと, 補題 4.1 から定理 1.2 を示す. 補題 4.1 は節 3 の補題 3.1 に相当するもので, **DIVP(log)** 困難な関数族 $(G_u)_u, (H_u)_u$ を模倣する実関数族 $(g_u)_u, (h_u)_u$ の存在を述べている. 定理 1.2 の証明では $(g_u)_u, (h_u)_u$ から求める g, h を構成する.

4.1 対数段差分方程式を模倣する関数族

補題 4.1. 任意の自然数 $k \geq 2$, 任意の言語 $L \in \mathbf{DIVP(log)}$ に対して, 係数のみに i を含む多項式 μ_i が存在して, 任意の多項式 γ に対して, 関数 $\rho\colon \mathbf{N} \to \mathbf{N}$, 関数族 g_u, h_u で, $\rho, (g_u)_u$ は多項式時間計算可能であり, 各二進文字列 u に対して以下を満たすものが存在する.

(i) $g_u\colon [0,1] \times [-1,1] \to \mathbf{R}, \quad h_u\colon [0,1] \to [-1,1]$;
(ii) h_u は g_u の常微分方程式 (1.1) の解;
(iii) g_u は (∞, k) 階連続微分可能;

(iv) 任意の $i \in \mathbf{N}$, $y \in [-1,1]$ に対して

$$\mathcal{D}^{(i,0)} g_u(0,y) = \mathcal{D}^{(i,0)} g_u(1,y) = 0$$

(v) 任意の $i \in \mathbf{N}$, $j \in \{0,\ldots,k\}$ に対して

$$\left|\mathcal{D}^{(i,j)} g_u(t,y)\right| \leq 2^{\mu_i(|u|)-\gamma(|u|)}$$

(vi) $h_u(1) = 2^{-\rho(|u|)} L(u)$.

補題 3.1 とくらべて補題 4.1 は, **PSPACE** が **DIVP(log)** に置き換わり, $(\infty,1)$ 回連続微分可能が (∞,k) 回連続微分可能に一般化されている.

補題 3.1 との本質的な違いは条件 (v) によって, h に対するフィードバックの大きさ, つまり $g(t,y)$ に対する y の値の影響がかなり制限されてしまう点にある. それにより, 模倣できる差分方程式が多項式段ではなく対数段に制限され, **PSPACE** 困難が **DIVP(log)** 困難へと置き換わる.

4.2 定理 1.2 の証明

定理 1.1 と定理 1.2 の関係は補題 3.1 と補題 4.1 の関係と等しい. つまり **PSPACE** が **DIVP(log)** に置き換わり, $(\infty,1)$ 回連続微分可能が (∞,k) 回連続微分可能に一般化されている. よって定理 1.1 の証明を, **PSPACE** を **DIVP(log)** で置き換え, $(\infty,1)$ 回連続微分可能を (∞,k) 回連続微分可能に一般化させることで, 定理 1.2 の証明が構成できる.

謝辞

本研究を遂行し発表するにあたり日本学術振興会「組織的な若手研究者等海外派遣プログラム」及び文部科学省科学研究費補助金若手研究 (B) 23700009 による援助を受けた. 記して謝意を表する.

参考文献

[1] E.A. Coddington and N. Levinson. *Theory of ordinary differential equations*. McGraw-Hill, 1955.

[2] A. Grzegorczyk. Computable functionals. *Fund. Math*, 42(19553):168–202, 1955.

[3] A. Kawamura. Complexity of initial value problems, 2010.

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

[5] K.I. Ko. On the computational complexity of ordinary differential equations. *Information and Control*, 58(1-3):157–194, 1983.

[6] K.I. Ko. *Complexity theory of real functions*. Birkhauser Boston Inc., 1991.

[7] K.I. Ko and H. Friedman. Computing power series in polynomial time. *Advances in Applied Mathematics*, 9(1):40–50, 1988.

[8] W. Miller. Recursive function theory and numerical analysis. *Journal of Computer and System Sciences*, 4(5):465–472, 1970.

[9] 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.

[10] Klaus Weihrauch. *Computable Analysis: An Introduction*. Texts in Theoretical Computer Science. Springer, 2000.

[11] 高木貞治. **解析概論**. 岩波書店, 1968.