Lipschitz Continuous O.D.E.s are Polynomial-Space Complete

Akitoshi KAWAMURA

University of Toronto

24th IEEE Conference on Computational Complexity
July 16, 2009
Problem

Given \( g : [0, 1] \times \mathbb{R} \rightarrow \mathbb{R} \), consider the IVP
\[
h(0) = 0, \quad h'(t) = g(t, h(t)).
\]
Problem

Given $g: [0, 1] \times \mathbb{R} \to \mathbb{R}$, consider the IVP

$$h(0) = 0, \quad h'(t) = g(t, h(t)).$$
Problem

Given \( g : [0, 1] \times \mathbb{R} \to \mathbb{R} \), consider the IVP
\[
h(0) = 0, \quad h'(t) = g(t, h(t)).
\]
Given \( g : [0, 1] \times \mathbb{R} \rightarrow \mathbb{R} \), consider the IVP
\[
h(0) = 0, \quad h'(t) = g(t, h(t)).
\]

A sufficient condition for unique solution \( h \) is that
\[
|g(t, y_0) - g(t, y_1)| \leq L|y_0 - y_1|,
\]
for some \( L \geq 0 \) (Lipschitz continuity).
Given $g: [0, 1] \times \mathbb{R} \rightarrow \mathbb{R}$, consider the IVP

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

A sufficient condition for unique solution $h$ is that

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

for some $L \geq 0$ (Lipschitz continuity).

**Question**

$g$ is easy $\implies$ $h$ is easy? (in the sense of Computable Analysis)
1. **Complexity of real functions**
   How we define \( \text{PTIME} \) real functions

2. **Warm-up: Integration**
   Assuming \( g \) is \( \text{PTIME} \) and ignores the second argument, how complex can \( h \) be?

3. **Lipschitz continuous IVP**
   Assuming \( g \) is \( \text{PTIME} \) and Lipschitz continuous, how complex can \( h \) be?

4. **Final remarks**

Outline

\[
h(0) = 0, \quad h'(t) = g(t, h(t))
\]
1. Complexity of real functions
   How we define PTIME real functions

2. Warm-up: Integration
   Assuming $g$ is PTIME and ignores the second argument, how complex can $h$ be?

3. Lipschitz continuous IVP
   Assuming $g$ is PTIME and Lipschitz continuous, how complex can $h$ be?

4. Final remarks
Read/write real numbers?

A real number cannot be encoded into a string.

So we encode it into a function from strings to strings, and give it to the machine as oracles.
We say that $\varphi : \Sigma^* \to \Sigma^*$ is a name of $t \in \mathbb{R}$ if for each $n \in \mathbb{N}$, $\varphi(0^n)$ is (the binary expansion of) $t$ rounded up or down at the $n$th bit below the point.

(Each $\varphi(0^n)$ approximates $t$ with precision $2^{-n}$)
Computing real functions

**Definition (Grzegorczyk 1955)**

Machine $M$ computes $f : [0, 1] \to \mathbb{R}$ if, for any name $\varphi$ of any $t \in [0, 1]$, $M^\varphi$ computes a name of $f(t)$.

In other words, $I_M$ Turing-reduces $f(t)$ to $t$.

Given access to approximations of $t$ to any precision, the machine yields $f(t)$ to any precision.

Note: Saying that $M$ is PTIME means that it halts in time $\text{poly}(m)$.

In particular, $n \cdot \text{poly}(m)$. 

![Diagram](image.png)
Computing real functions

Definition (Grzegorczyk 1955)

Machine $M$ computes $f : [0, 1] \rightarrow \mathbb{R}$ if, for any name $\varphi$ of any $t \in [0, 1]$, $M^\varphi$ computes a name of $f(t)$.

In other words,
- $M$ Turing-reduces $f(t)$ to $t$.
- Given access to approximations of $t$ to any precision, the machine yields $f(t)$ to any precision.
Computing real functions

Definition (Grzegorzcyk 1955)

Machine $M$ computes $f : [0, 1] \rightarrow \mathbb{R}$ if, for any name $\varphi$ of any $t \in [0, 1]$, $M^\varphi$ computes a name of $f(t)$.

In other words,

- $M$ Turing-reduces $f(t)$ to $t$.
- Given access to approximations of $t$ to any precision, the machine yields $f(t)$ to any precision.

Note: Saying that $M$ is PTIME means that it halts in time $\text{poly}(m)$. In particular, $n \leq \text{poly}(m)$. 

\[ \text{Oracle} \]

\[ 0^n \rightarrow 2^{-n}\text{-approximation of } t \]

\[ \text{Machine} \]

\[ 0^m \rightarrow 2^{-m}\text{-approximation of } f(t) \]
Example 1

Addition $+: [0, 1] \times [0, 1] \to \mathbb{R}$ is PTIME.

To do: Given (names of) $t$ and $t'$ as oracles and $0^m$ as input, output a $2^{-m}$-approximation of $t + t'$.

- Ask the oracles for $2^{-m-2}$-approximations of $t$ and $t'$.
- Add the answers (as rational numbers).
- Output the closest rational number to the sum that has $m$ bits below the point.
Example 2

\[ \exp: [0, 1] \rightarrow \mathbb{R} \text{ is PTIME.} \]

To get a \(2^{-m}\)-approximation of the value

\[
\exp t = \frac{1}{0!} + \frac{t}{1!} + \frac{t^2}{2!} + \frac{t^3}{3!} + \cdots ,
\]

it suffices to compute a \(2^{-m}/2\)-approximation of the sum of the first \(m\) terms (because the remaining terms add up to at most \(2^{-m}/2\)).
Example 3

\[ [\cdot] : [0, 5] \rightarrow \mathbb{R} \text{ is not computable.} \]
Example 3

\[ \lfloor \cdot \rfloor : [0, 5] \to \mathbb{R} \text{ is not computable.} \]
Example 3

$[\cdot]: [0, 5] \rightarrow \mathbb{R}$ is not computable.
$[\cdot] : [0, 5] \rightarrow \mathbb{R}$ is not computable.
Example 3

\[ [\cdot] : [0, 5] \to \mathbb{R} \text{ is not computable.} \]
Example 3

\[ \lfloor \cdot \rfloor : [0, 5] \rightarrow \mathbb{R} \text{ is not computable}. \]

Machine (computes \( \lfloor \cdot \rfloor \))

Oracle (name of \( t = 2 \))

\( t \) is approximately \( t \)

\( |t| \) is approximately \( t \)
$[\cdot] : [0, 5] \to \mathbb{R}$ is not computable.
Example 3

$[\cdot] : [0, 5] \rightarrow \mathbb{R}$ is not computable.
Example 3

\[ \lfloor \cdot \rfloor : [0, 5] \to \mathbb{R} \text{ is not computable.} \]
Computable functions are continuous

Theorem

- A computable real function is continuous.
- A \textsc{PTime} real function has a polynomial modulus of continuity.

Modulus of continuity \( p \):

\[
|t - t'| < 2^{-p(n)} \implies |f(t) - f(t')| < 2^{-n}.
\]
Outline

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

1. Complexity of real functions
   How we define $\text{PTIME}$ real functions

2. Warm-up: Integration
   Assuming $g$ is $\text{PTIME}$ and ignores the second argument, how complex can $h$ be?

3. Lipschitz continuous IVP
   Assuming $g$ is $\text{PTIME}$ and Lipschitz continuous, how complex can $h$ be?

4. Final remarks
Complexity of integration

Theorem (essentially [Friedman 1984])

There are \( g : [0, 1] \rightarrow \mathbb{R} \) and \( h : [0, 1] \rightarrow \mathbb{R} \) such that

- \( g \) is \( \text{PTIME} \);
- \( h(0) = 0, \quad h'(t) = g(t) \);
- \( h \) is \( \#\text{PTIME} \)-hard.
Complexity of integration

Theorem (essentially [Friedman 1984])

There are $g: [0, 1] \to \mathbb{R}$ and $h: [0, 1] \to \mathbb{R}$ such that

- $g$ is PTIME;
- $h(0) = 0$, $h'(t) = g(t)$; — special case of $h'(t) = g(t, h(t))$ where $g$ ignores its second argument
- $h$ is $\#\text{PTIME}$-hard.
Complexity of integration

Theorem (essentially [Friedman 1984])

There are \( g : [0, 1] \rightarrow \mathbb{R} \) and \( h : [0, 1] \rightarrow \mathbb{R} \) such that

- \( g \) is \( \text{PTIME} \);
- \( h(0) = 0, \ h'(t) = g(t) \); — special case of \( h'(t) = g(t, h(t)) \)

where \( g \) ignores its second argument
- \( h \) is \( \text{#PTIME} \)-hard, in the sense that

\[
\begin{array}{c}
0^n \downarrow \\
\text{machine for } h \\
2^{-n}\text{-approximation of } t
\end{array} \quad \begin{array}{c}
2^{-m}\text{-approximation of } h(t) \\
0^m \uparrow \\
2^{-m}\text{-approximation of } t
\end{array} \]
Complexity of integration

Theorem (essentially [Friedman 1984])

There are $g: [0, 1] \rightarrow \mathbb{R}$ and $h: [0, 1] \rightarrow \mathbb{R}$ such that

- $g$ is PTIME;
- $h(0) = 0$, $h'(t) = g(t)$; — special case of $h'(t) = g(t, h(t))$
- where $g$ ignores its second argument
- $h$ is #PTIME-hard, in the sense that

\[ u \rightarrow \text{PTIME} \rightarrow 2^{-n}\text{-approximation of } t \rightarrow \text{machine for } h \rightarrow 2^{-m}\text{-approximation of } h(t) \rightarrow \text{PTIME} \rightarrow \text{PTIME} \rightarrow \#\text{SAT}(u). \]
Reducing $\#\text{SAT}$ to integration

What we want:

$\bullet$ $g$ is easy (PTIME);
Reducing $\#\text{SAT}$ to integration

What we want:

- $g$ is easy (PTIME);
- $h$ is hard ($\#\text{SAT}$ reduces to it).

$\#\text{SAT}(u) = \text{number of}$

$(u, v)$

$\in \mathcal{R}$

(for some

$\mathcal{R} \subseteq \mathcal{P}$).

For each $u \in \Sigma$,

assign interval $[l - u, l + u]$.

Put there a pair of

reduced copies of

$g_u$.

$g(t), h(t), \ldots$
Reducing \( \#\text{SAT} \) to integration

For each \( u \in \Sigma \), assign interval \([l - u, l + u]\). Put there a pair of reduced copies of \( g_u \).

What we want:

\( \#\text{SAT}(u) \) is encoded in \( h_u(1) \), or in \( h(c_u) \).
Reducing \#SAT to integration

$$\#SAT(u) = \text{number of } v \text{ with } (u, v) \in R$$
(for some $R \in \text{PTIME}$).
Reducing $\#\text{SAT}$ to integration

$\#\text{SAT}(u) = \text{number of } v \text{ with } (u, v) \in R$
(for some $R \in \text{PTIME}$).

For each $u \in \Sigma^*$,
assign interval $[l_u^-, l_u^+]$. 

\[ \text{Put there a pair of reduced copies of } g. \]

What we want:
$\#\text{SAT}$ is easy (PTIME);
$\#\text{SAT}$ is hard ($\#\text{SAT}$ reduces to it).
Reducing $\#\text{SAT}$ to integration

$\#\text{SAT}(u) = \text{number of } v \text{ with } (u, v) \in R$
(for some $R \in \text{PTIME}$).

For each $u \in \Sigma^*$,
assign interval $[l_u^-, l_u^+]$. 

\[
\begin{array}{c}
\text{y} \\
\hline
\text{l}_u^- \quad c_u \quad \text{l}_u^+ \\
\text{t}
\end{array}
\]
Reducing \( \#\text{SAT} \) to integration

\( \#\text{SAT}(u) = \) number of \( v \) with \( (u, v) \in R \) (for some \( R \in \text{PTIME} \)).

For each \( u \in \Sigma^* \), assign interval \([l_u^-, l_u^+]\).
Put there a pair of reduced copies of \( g_u \).
Reducing $\#\text{SAT}$ to integration

$\#\text{SAT}(u) = \text{number of } v \text{ with } (u, v) \in R$
(for some $R \in \text{PTIME}$).

For each $u \in \Sigma^*$, assign interval $[l_u^-, l_u^+]$.
Put there a pair of reduced copies of $g_u$. 

$g_u(t)$

$\left(u, 000\right) \notin R$

$y$

$t$

$y$

$t$

$(u, 000)$

Reduced copy

Reduced copy
Reducing \( \#\text{SAT} \) to integration

\[
\begin{align*}
\#\text{SAT}(u) &= \text{number of } v \text{ with } (u, v) \in R \\
&(\text{for some } R \in \text{PTIME}).
\end{align*}
\]

For each \( u \in \sum^* \), assign interval \([l_u^-, l_u^+]\).
Put there a pair of reduced copies of \( g_u \).
Reducing \( \# \text{SAT} \) to integration

\( \# \text{SAT}(u) = \text{number of } v \text{ with } (u, v) \in R \)

(for some \( R \in \text{PTIME} \)).

For each \( u \in \Sigma^* \),
assign interval \([l_u^-, l_u^+]\).
Put there a pair of reduced copies of \( g_u \).
Reducing \#SAT to integration

\#\text{SAT}(u) = \text{number of } \nu \text{ with } (u, \nu) \in R
\text{(for some } R \in \text{PTIME}).

For each } u \in \Sigma^*, \text{ assign interval } [l_u^-, l_u^+].
\text{Put there a pair of reduced copies of } g_u.
Reducing $\#\text{SAT}$ to integration

$\#\text{SAT}(u) = \text{number of } v \text{ with } (u, v) \in R$  
(for some $R \in \text{PTIME}$).

For each $u \in \Sigma^*$, assign interval $[l_u^-, l_u^+]$.  
Put there a pair of reduced copies of $g_u$.

$\#\text{SAT}(u)$ is encoded in $h_u(1)$. 

<table>
<thead>
<tr>
<th>(u, 000)</th>
<th>(u, 001)</th>
<th>(u, 010)</th>
<th>(u, 011)</th>
<th>...</th>
<th>(u, 111)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\notin R$</td>
<td>$\in R$</td>
<td>$\in R$</td>
<td>$\notin R$</td>
<td>...</td>
<td>$\in R$</td>
</tr>
</tbody>
</table>

$g_u(t)$

$h_u(t)$

$h'_u(t) = g_u(t)$

$\#\text{SAT}(u)$

$y = h_u(t)$

$1$

$t$

$y$

reduced copy
Reducing \( \#\text{SAT} \) to integration

\[
\#\text{SAT}(u) = \text{number of } v \text{ with } (u, v) \in R \\
\text{(for some } R \in \text{ PTIME}).
\]

For each \( u \in \Sigma^* \), assign interval \([l_u^-, l_u^+]\).
Put there a pair of reduced copies of \( g_u \).

\( \#\text{SAT}(u) \) is encoded in \( h_u(1) \), or in \( h(c_u) \).
Reducing \( \#\text{SAT} \) to integration

\[ \text{#SAT}(u) = \text{number of } v \text{ with } (u, v) \in R \]
(for some \( R \in \text{PTIME} \)).

For each \( u \in \Sigma^* \), assign interval \([l_u^-, l_u^+]\).
Put there a pair of reduced copies of \( g_u \).

\( \text{#SAT}(u) \) is encoded in \( h_u(1) \), or in \( h(c_u) \).
Reducing $\#\text{SAT}$ to integration

$\#\text{SAT}(u) = \text{number of } v \text{ with } (u, v) \in R$
(for some $R \in \text{PTIME}$).

For each $u \in \Sigma^*$, assign interval $[l_u^-, l_u^+]$.
Put there a pair of reduced copies of $g_u$.

$\#\text{SAT}(u)$ is encoded in $h_u(1)$, or in $h(c_u)$. 
Outline

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

1. Complexity of real functions
   How we define \textsc{PTime} real functions

2. Warm-up: Integration
   Assuming $g$ is \textsc{PTime} and ignores the second argument, how complex can $h$ be?

3. Lipschitz continuous IVP
   Assuming $g$ is \textsc{PTime} and Lipschitz continuous, how complex can $h$ be?

4. Final remarks
Complexity of Lipschitz IVPs

Main Theorem

There are \( g : [0, 1] \times [-1, 1] \to \mathbb{R} \) and \( h : [0, 1] \to [-1, 1] \) such that

- \( g \) is \( \text{PTIME} \) and Lipschitz continuous;
- \( h(0) = 0, \quad h'(t) = g(t, h(t)) \);
- \( h \) is \( \text{PSPACE} \)-hard in the sense that

Cf. Upper bound: \( h \) is in \( \text{PS} \) \( \text{PACE} \) by the Euler method \([\text{Ko} \ 1983]\).
Main Theorem

There are \( g: [0, 1] \times [-1, 1] \rightarrow \mathbb{R} \) and \( h: [0, 1] \rightarrow [-1, 1] \) such that

- \( g \) is \( \text{PTIME} \) and Lipschitz continuous;
- \( h(0) = 0, \quad h'(t) = g(t, h(t)) \);
- \( h \) is \( \text{PSPACE} \)-hard in the sense that

\[ PTIME \]
\[ 0^n \]
\[ 2^{-n} \text{-approximation of } t \]

machine for \( h \)

\[ 0^m \]
\[ 2^{-m} \text{-approximation of } h(t) \]

\( u \)

\[ \text{PTIME} \]
\[ \text{PTIME} \]

\( \text{QBF}(u) \)

Cf. Upper bound: \( h \) is in \( \text{PSPACE} \) by the Euler method [Ko 1983].
Proof (1/3): An attempt to reduce \textsc{PSPACE} to IVP

As before, we need blocks $g_u$ such that $h_u(1)$ indicates if $u \in \text{QBF}$. 

\[
\begin{align*}
\text{Differential equation:} \\
\frac{dh_u}{dt} = g_u(t, h(t))
\end{align*}
\]
Proof (1/3): An attempt to reduce \textsc{PSPACE} to IVP

As before, we need blocks \( g_u \) such that \( h_u(1) \) indicates if \( u \in \text{QBF} \).

\[
y(t) = h_u(t)
g_u(t, y)
\]

Differential equation:
\[
h'_u(t) = g_u(t, h(t))
\]
Proof (1/3): An attempt to reduce \textbf{PSPACE} to IVP

As before, we need blocks \( g_u \) such that \( h_u(1) \) indicates if \( u \in \text{QBF} \).

Differential equation:
\[ h'_u(t) = g_u(t, h(t)) \]

Description of a \textbf{PSPACE} machine on input \( u \):
\[ H_u(T+1) = G_u(T, H_u(T)) \]
Proof (1/3): An attempt to reduce \textsc{PSPACE} to IVP

As before, we need blocks $g_u$ such that $h_u(1)$ indicates if $u \in \text{QBF}$.

\[ y = h_u(t) \]

Differential equation:
\[ h'_u(t) = g_u(t, h(t)) \]

Description of a \textsc{PSPACE} machine on input $u$:
\[ H_u(T + 1) = G_u(T, H_u(T)) \]
Proof (2/3): Why this attempt does not work

Not all $G_u$ can be translated to $g_u$.

$g_u(t, y)$

Flows cannot cross. In fact, the Lipschitz condition keeps them from widening or narrowing fast.
Proof (2/3): Why this attempt does not work

Not all $G_u$ can be translated to $g_u$.

Flows can never cross.
Not all $G_u$ can be translated to $g_u$.

Flows can never cross.

In fact, the Lipschitz condition keeps them from widening or narrowing fast.
Proof (2/3): Why this attempt does not work

Not all $G_u$ can be translated to $g_u$.

Flows can never cross.

In fact, the Lipschitz condition keeps them from widening or narrowing fast.

The ‘feedback’ (on $h_u$) of equation $h'_u(t) = g_u(t, h_u(t))$ is very weak under the Lipschitz condition.
Proof (3/3): Layered \text{PSPACE} tableaux

General \text{PSPACE} computation tableaux cannot be simulated by the differential equation.

\[
H_u(T + 1) = G_u(T, H_u(T))
\]
Proof (3/3): Layered PSPACE tableaux

General PSPACE computation tableaux cannot be simulated by the differential equation.

Split each cell into parts, and put restrictions on which part can affect which.
Proof (3/3): Layered PSPACE tableaux

\[
H_u(0,0) \quad H_u(0,1) \quad H_u(0,2)
\]

\[
H_u(1,0) \quad H_u(1,1)
\]

\[
H_u(2,0)
\]

\[
H_u(i,T + 1) = H_u(i,T) + G_u(T, H_u(i - 1, T))
\]

General PSPACE computation tableaux cannot be simulated by the differential equation.

Split each cell into parts, and put restrictions on which part can affect which.

This tableau can be simulated by the equation, and is PSPACE-complete despite the restriction.

\[
H_u(0, 2Q(|u|))
\]
Outline

$\begin{align*}
    h(0) &= 0, \\
    h'(t) &= g(t, h(t))
\end{align*}$

1. Complexity of real functions
   How we define \texttt{PTIME} real functions

2. Warm-up: Integration
   Assuming $g$ is \texttt{PTIME} and ignores the second argument, how complex can $h$ be?

3. Lipschitz continuous IVP
   Assuming $g$ is \texttt{PTIME} and Lipschitz continuous, how complex can $h$ be?

4. Final remarks
### Related results

\[ h(0) = 0, \quad h'(t) = g(t, h(t)) \]

**Assuming \( g \) is \( \text{PTIME} \), how complex can \( h \) be?**

<table>
<thead>
<tr>
<th>Assumptions</th>
<th>Positive results on ( h )</th>
<th>Negative results on ( h )</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>---</td>
<td>can be (non-unique and) all non-computable [Pour-El 1979]</td>
</tr>
<tr>
<td>( h ) is the unique solution</td>
<td>computable [implicit in Osgood 1898]</td>
<td>time (or space) cannot be bounded [Miller 1970]</td>
</tr>
<tr>
<td>( g ) is Lipschitz continuous</td>
<td>( \text{PSPACE} ) [Ko 1983]</td>
<td>can be ( \text{PSPACE})-hard</td>
</tr>
<tr>
<td>( g ) is analytic</td>
<td>( \text{PTIME} ) [Ko 1988, K.]</td>
<td>---</td>
</tr>
</tbody>
</table>
Open problems

- What happens between the Lipschitz case ($\text{PSPACE}$) and the analytic case ($\text{PTIME}$)?
- Effective versions? (Complexity of “computing $h$ from $g$”)
- Other differential equations