Computational complexity of solving polynomial differential equations over unbounded domains

Amaury Pouly
Joint work with Daniel Graça

10 May 2018

Ordinary Differential Equations (ODEs)

System of ODEs:

$$\begin{cases} y_1(0) = y_{0,1} \\ \vdots \\ y_n(0) = y_{0,n} \end{cases} \qquad \begin{cases} y_1'(t) = f_1(y_1(t), \ldots, y_n(t), t) \\ \vdots \\ y_n'(t) = f_n(y_1(t), \ldots, y_n(t), t) \end{cases}$$

More compactly:

$$y(0) = y_0 \qquad y'(t) = f(y(t), t)$$

Ordinary Differential Equations (ODEs)

System of ODEs:

$$\begin{cases} y_1(0) = y_{0,1} \\ \vdots \\ y_n(0) = y_{0,n} \end{cases} \qquad \begin{cases} y_1'(t) = f_1(y_1(t), \dots, y_n(t), t) \\ \vdots \\ y_n'(t) = f_n(y_1(t), \dots, y_n(t), t) \end{cases}$$

More compactly:

$$y(0) = y_0 \qquad y'(t) = f(y(t), t)$$

Get rid of the time:

$$\begin{cases} y(0) = y_0 \\ z(0) = 0 \end{cases} \qquad \begin{cases} y'(t) = f(y(t), z(t)) \\ z'(t) = 1 \end{cases}$$

Ordinary Differential Equations (ODEs)

System of ODEs:

$$\begin{cases} y_1(0) = y_{0,1} \\ \vdots \\ y_n(0) = y_{0,n} \end{cases} \qquad \begin{cases} y_1'(t) = f_1(y_1(t), \dots, y_n(t), t) \\ \vdots \\ y_n'(t) = f_n(y_1(t), \dots, y_n(t), t) \end{cases}$$

More compactly:

$$y(0) = y_0 \qquad y'(t) = f(y(t), t)$$

Get rid of the time:

$$\begin{cases} y(0) = y_0 \\ z(0) = 0 \end{cases} \qquad \begin{cases} y'(t) = f(y(t), z(t)) \\ z'(t) = 1 \end{cases}$$

In this talk: autonomous first order explicit system of ODEs

$$y(0) = y_0 \qquad y' = f(y) \qquad y : (a, b) \to \mathbb{R}^n$$

A word on computability for real functions

Classical computability (Turing machine): compute on words, integers, rationals, ...

A word on computability for real functions

Classical computability (Turing machine): compute on words, integers, rationals, ...

Real computability: at least *two different notions*

- BSS (Blum-Shub-Smale) machine: register machine that can store arbitrary real numbers and that can compute rational functions over reals at unit cost. **Comparisons between reals are allowed.**

A word on computability for real functions

Classical computability (Turing machine): compute on words, integers, rationals, ...

Real computability: at least *two different notions*

- BSS (Blum-Shub-Smale) machine: register machine that can store arbitrary real numbers and that can compute rational functions over reals at unit cost. **Comparisons between reals are allowed.**
- Computable Analysis: reals are represented as converging Cauchy sequences, computations are carried out by rational approximations using Turing machines. **Comparisons between reals is not decidable in general.** Computable implies continuous.

A word on computability for real functions

Classical computability (Turing machine): compute on words, integers, rationals, ...

Real computability: at least *two different notions*

- BSS (Blum-Shub-Smale) machine: register machine that can store arbitrary real numbers and that can compute rational functions over reals at unit cost. **Comparisons between reals are allowed.**
- Computable Analysis: reals are represented as converging Cauchy sequences, computations are carried out by rational approximations using Turing machines. **Comparisons between reals is not decidable in general.** Computable implies continuous.

In this talk (unless specified)

We use Computable Analysis.

Computability of solutions: the theory

Let $I = (a, b)$ and $f \in C^0(\mathbb{R}^d)$. Assume $y \in C^1(I, \mathbb{R}^d)$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)). \tag{1}$$

Given $t \in I$ and $n \in \mathbb{N}$, can we compute $q \in \mathbb{Q}^d$ s.t. $\|q - y(t)\| \leqslant 2^{-n}$?

Computability of solutions: the theory

Let $I = (a, b)$ and $f \in C^0(\mathbb{R}^d)$. Assume $y \in C^1(I, \mathbb{R}^d)$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)). \tag{1}$$

Is y computable?

Computability of solutions: the theory

Let $I = (a, b)$ and $f \in C^0(\mathbb{R}^d)$. Assume $y \in C^1(I, \mathbb{R}^d)$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)). \tag{1}$$

Is y computable?

Theorem (Pour-El and Richards)

There exists a computable (hence continuous) f such that **none of the solutions** to (1) is computable.

Computability of solutions: the theory

Let $I = (a, b)$ and $f \in C^0(\mathbb{R}^d)$. Assume $y \in C^1(I, \mathbb{R}^d)$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)). \tag{1}$$

Is y computable?

Theorem (Pour-El and Richards)

There exists a computable (hence continuous) f such that **none of the solutions** to (1) is computable.

Theorem (Ruohonen)

If f is computable and (1) has a unique solution, then it is computable.

Computability of solutions: the theory

Let $I = (a, b)$ and $f \in C^0(\mathbb{R}^d)$. Assume $y \in C^1(I, \mathbb{R}^d)$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)). \tag{1}$$

Is y computable?

Theorem (Pour-El and Richards)

There exists a computable (hence continuous) f such that **none of the solutions** to (1) is computable.

Theorem (Ruohonen)

If f is computable and (1) has a unique solution, then it is computable.

Theorem (Buescu, Campagnolo and Graça)

Computing the maximum interval of life (or deciding if it is bounded) is undecidable, even if f is a polynomial.

Computability of solutions: the theory

Let $I = (a, b)$ and $f \in C^0(\mathbb{R}^d)$. Assume $y \in C^1(I, \mathbb{R}^d)$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)). \tag{1}$$

Is y computable?

Theorem (Pour-El and Richards)

There exists a computable (hence continuous) f such that **none of the solutions** to (1) is computable.

Theorem (Ruohonen)

If f is computable and (1) has a unique solution, then it is computable.

Theorem (Buescu, Campagnolo and Graça)

Computing the maximum interval of life (or deciding if it is bounded) is undecidable, even if f is a polynomial.

Theorem (Collins and Graça)

The map $f \mapsto y(\cdot)$ for those f where y is unique, is computable.

Complexity of solutions: typical textbook result

Assume f **Lipschitz** and computable, and $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)).$$

Complexity of solutions: typical textbook result

Assume f **Lipschitz** and computable, and $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)).$$

Theorem (Folklore, simplified)

The classical Runge–Kutta method is a fourth-order method:

Complexity of solutions: typical textbook result

Assume f **Lipschitz** and computable, and $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)).$$

Theorem (Folklore, simplified)

The classical Runge–Kutta method is a fourth-order method: given a time $t \in I$ and a *time step* h, the algorithm returns $q \in \mathbb{Q}^d$ s.t. $\|q - y(t)\| \leqslant \mathcal{O}\left(h^4\right)$ and has running time $\mathcal{O}\left(\frac{1}{h^4}\right)$.

Complexity of solutions: typical textbook result

Assume f **Lipschitz** and computable, and $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)).$$

Theorem (Folklore, simplified)

The classical Runge–Kutta method is a fourth-order method: given a time $t \in I$ and a *time step* h, the algorithm returns $q \in \mathbb{Q}^d$ s.t. $\|q - y(t)\| \leqslant \mathcal{O}\left(h^4\right)$ and has running time $\mathcal{O}\left(\frac{1}{h^4}\right)$.

Usually followed by benchmarks.

Complexity of solutions: typical textbook result

Assume f **Lipschitz** and computable, and $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)).$$

Theorem (Folklore, simplified)

The classical Runge–Kutta method is a fourth-order method: given a time $t \in I$ and a *time step* h, the algorithm returns $q \in \mathbb{Q}^d$ s.t. $\|q - y(t)\| \leqslant \mathcal{O}\left(h^4\right)$ and has running time $\mathcal{O}\left(\frac{1}{h^4}\right)$.

Usually followed by benchmarks.

Problems with this approach:

- Accuracy of the result? $\mathcal{O}\left(h^4\right) \leqslant Ah^4$ but A is unknown
- Same problem with complexity
- f is Lipschitz: typically only holds over compact domains

Complexity of solutions: typical textbook result

Assume f computable and K-Lipschitz, and $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad\qquad y'(t) = f(y(t)).$$

Complexity of solutions: typical textbook result

Assume f computable and K-Lipschitz, and $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)).$$

Theorem (Folklore, simplified)

Euler's method global truncation error is:

$$\frac{hM}{2K}\left(e^{Kt} - 1\right) \qquad \text{where } M = \sup_{u \in I} \left\| y''(u) \right\|.$$

Complexity of solutions: typical textbook result

Assume f computable and K-Lipschitz, and $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)).$$

Theorem (Folklore, simplified)

Euler's method global truncation error is:

$$\frac{hM}{2K}\left(e^{Kt} - 1\right) = \mathcal{O}\left(h\right) \qquad \text{where } M = \sup_{u \in I} \left\| y''(u) \right\|.$$

In particular it has order 1 over compact time (I) domains.

Complexity of solutions: typical textbook result

Assume f computable and K-Lipschitz, and $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)).$$

Theorem (Folklore, simplified)

Euler's method global truncation error is:

$$\frac{hM}{2K}\left(e^{Kt} - 1\right) = \mathcal{O}\left(h\right) \qquad \text{where } M = \sup_{u \in I} \left\| y''(u) \right\|.$$

In particular it has order 1 over compact time (I) domains.

This bound is "useless" unless:

- you know K: f must be Lipschitz on "$\{y(u) : u \in I\}$" or globally

Complexity of solutions: typical textbook result

Assume f computable and K-Lipschitz, and $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)).$$

Theorem (Folklore, simplified)

Euler's method global truncation error is:

$$\frac{hM}{2K}\left(e^{Kt} - 1\right) = \mathcal{O}\left(h\right) \qquad \text{where } M = \sup_{u \in I} \left\| y''(u) \right\|.$$

In particular it has order 1 over compact time (I) domains.

This bound is "useless" unless:

- you know K: f must be Lipschitz on "$\{y(u) : u \in I\}$" or globally
- you know M: but it depends on y !!

Chicken-and-egg problem: the constant in the accuracy bound depends on computing the solution.

Complexity of solutions: the rescaling "myth"

Assume f computable and K-Lipschitz, and $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)) \qquad \text{with unbounded } I = [0, +\infty).$$

Complexity of solutions: the rescaling "myth"

Assume f computable and K-Lipschitz, and $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)) \qquad \text{with unbounded } I = [0, +\infty).$$

To compute $y(T)$ we could:

1. Define $z(u) = y(Tu)$, then
$$y(T) = z(1)$$
2. Observe that
$$z'(u) = Tf(z) =: f_T(z)$$
3. Solve $z(0) = y_0, z' = f_T(z)$
[0, 1] is a compact!

Complexity of solutions: the rescaling "myth"

Assume f computable and K-Lipschitz, and $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)) \qquad \text{with unbounded } I = [0, +\infty).$$

To compute $y(T)$ we could:

1. Define $z(u) = y(Tu)$, then
$$y(T) = z(1)$$
2. Observe that
$$z'(u) = Tf(z) =: f_T(z)$$
3. Solve $z(0) = y_0, z' = f_T(z)$
[0, 1] is a compact!

Bad analysis: $y(T) = z(1)$
Accuracy: $O(h)$ (compact)

Complexity of solutions: the rescaling "myth"

Assume f computable and K-Lipschitz, and $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)) \qquad \text{with unbounded } I = [0, +\infty).$$

To compute $y(T)$ we could:

1. Define $z(u) = y(Tu)$, then
$$y(T) = z(1)$$
2. Observe that
$$z'(u) = Tf(z) =: f_T(z)$$
3. Solve $z(0) = y_0, z' = f_T(z)$
[0, 1] is a compact!

Bad analysis: $y(T) = z(1)$
Accuracy: $O(h)$ (compact)

Better analysis:
Accuracy: $A_{K_T, M_z} h$ where

$$K_T = \text{Lipschitz constant of } f_T$$
$$M_z = \max_{u \in [0,1]} \left\| z''(u) \right\| = \max_{t \in [0,T]} \left\| y''(t) \right\|$$

Complexity of solutions: the rescaling "myth"

Assume f computable and K-Lipschitz, and $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)) \qquad \text{with unbounded } I = [0, +\infty).$$

To compute $y(T)$ we could:

1. Define $z(u) = y(Tu)$, then
$$y(T) = z(1)$$
2. Observe that
$$z'(u) = Tf(z) =: f_T(z)$$
3. Solve $z(0) = y_0, z' = f_T(z)$
[0, 1] is a compact!

Bad analysis: $y(T) = z(1)$
Accuracy: $O(h)$ (compact)

Better analysis:
Accuracy: $A_{K_T,M_z} h$ where

$$K_T = \text{Lipschitz constant of } f_T$$
$$M_z = \max_{u \in [0,1]} \|z''(u)\| = \max_{t \in [0,T]} \|y''(t)\|$$

Note: now f really needs to be globally Lipschitz.

Complexity of solutions: the rescaling "myth"

Assume f computable and K-Lipschitz, and $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)) \qquad \text{with unbounded } I = [0, +\infty).$$

To compute $y(T)$ we could:

1. Define $z(u) = y(Tu)$, then
$$y(T) = z(1)$$
2. Observe that
$$z'(u) = Tf(z) =: f_T(z)$$
3. Solve $z(0) = y_0, z' = f_T(z)$
[0, 1] is a compact!

Bad analysis: $y(T) = z(1)$
Accuracy: $O(h)$ (compact)

Better analysis:
Accuracy: $A_{K_T,M_z} h$ where

$$K_T = \text{Lipschitz constant of } f_T$$
$$M_z = \max_{u \in [0,1]} \|z''(u)\| = \max_{t \in [0,T]} \|y''(t)\|$$

Note: now f really needs to be globally Lipschitz.

Conclusion

This tells us **nothing** about the complexity of the problem.

Side note on practical methods

Assume $y : [0,1] \to \mathbb{R}^d$ satisfies $\forall t \in [0,1]$:

$$y(0) = 0, \qquad y'(t) = f(y(t)).$$

There exists methods of the form:
given h and t, compute $q \in \mathbb{Q}^d$ and $\varepsilon > 0$ such that $\|y(t) - q\| \leqslant \varepsilon$ with the guarantee that $\varepsilon \to 0$ as $h \to 0$.

These methods have **no upper bound** on complexity.

They usually rely on interval arithmetic.

Nonuniform complexity-theoretic approach

Assume $y : [0, 1] \to \mathbb{R}^d$ satisfies $\forall t \in [0, 1]$:

$$y(0) = 0, \qquad y'(t) = f(y(t)).$$

Assumption on f	**Lower bound on y**	**Upper bound on y**
computable	arbitrary	computable if unique

Nonuniform complexity-theoretic approach

Assume $y : [0,1] \to \mathbb{R}^d$ satisfies $\forall t \in [0,1]$:

$$y(0) = 0, \qquad y'(t) = f(y(t)).$$

Assumption on f	**Lower bound on y**	**Upper bound on y**
computable	arbitrary	computable if unique
PTIME	arbitrary	computable

Nonuniform complexity-theoretic approach

Assume $y : [0,1] \to \mathbb{R}^d$ satisfies $\forall t \in [0,1]$:

$$y(0) = 0, \qquad y'(t) = f(y(t)).$$

Assumption on f	**Lower bound on y**	**Upper bound on y**
computable	arbitrary	computable if unique
PTIME	arbitrary	computable
PTIME + Lipschitz	PSPACE-hard	PSPACE

Nonuniform complexity-theoretic approach

Assume $y : [0,1] \to \mathbb{R}^d$ satisfies $\forall t \in [0,1]$:

$$y(0) = 0, \qquad y'(t) = f(y(t)).$$

Assumption on f	**Lower bound on y**	**Upper bound on y**
computable	arbitrary	computable if unique
PTIME	arbitrary	computable
PTIME + Lipschitz	PSPACE-hard	PSPACE
PTIME + C^1	PSPACE-hard	PSPACE

Nonuniform complexity-theoretic approach

Assume $y : [0,1] \to \mathbb{R}^d$ satisfies $\forall t \in [0,1]$:

$$y(0) = 0, \qquad y'(t) = f(y(t)).$$

Assumption on f	**Lower bound on y**	**Upper bound on y**
computable	arbitrary	computable if unique
PTIME	arbitrary	computable
PTIME + Lipschitz	PSPACE-hard	PSPACE
PTIME + C^1	PSPACE-hard	PSPACE
PTIME + $C^k, k \geqslant 2$	CH-hard	PSPACE

Nonuniform complexity-theoretic approach

Assume $y : [0,1] \to \mathbb{R}^d$ satisfies $\forall t \in [0,1]$:

$$y(0) = 0, \qquad y'(t) = f(y(t)).$$

Assumption on f	**Lower bound on y**	**Upper bound on y**
computable	arbitrary	computable if unique
PTIME	arbitrary	computable
PTIME + Lipschitz	PSPACE-hard	PSPACE
PTIME + C^1	PSPACE-hard	PSPACE
PTIME + $C^k, k \geqslant 2$	CH-hard	PSPACE
PTIME + analytic	—	PTIME

Nonuniform complexity-theoretic approach

Assume $y : [0,1] \to \mathbb{R}^d$ satisfies $\forall t \in [0,1]$:

$$y(0) = 0, \qquad y'(t) = f(y(t)).$$

Assumption on f	**Lower bound on y**	**Upper bound on y**
computable	arbitrary	computable if unique
PTIME	arbitrary	computable
PTIME + Lipschitz	PSPACE-hard	PSPACE
PTIME + C^1	PSPACE-hard	PSPACE
PTIME + $C^k, k \geqslant 2$	CH-hard	PSPACE
PTIME + analytic	—	PTIME

But those results can be deceiving...

$$\begin{cases} y_1(0) = 1 \\ y_2(0) = 1 \\ \vdots \\ y_d(0) = 1 \end{cases} \qquad \begin{cases} y_1' = y_1 \\ y_2' = y_1 y_2 \\ \vdots \\ y_n' = y_{d-1} y_n \end{cases} \qquad \to \qquad y(t) = \mathcal{O}\left(e^{e^{\cdot^{\cdot^{\cdot^{e^t}}}}}\right)$$

y is PTIME over $[0,1]$

Nonuniform complexity: limitation

Example:

$$f \text{ PTIME analytic } \Rightarrow y \text{ PTIME } \Rightarrow y(t) \pm 2^{-n} \text{ in time } An^k$$

But:

Nonuniform complexity: limitation

Example:

$$f \text{ PTIME analytic} \Rightarrow y \text{ PTIME} \Rightarrow y(t) \pm 2^{-n} \text{ in time } An^k$$

But:

- "Hides" some of the complexity: A,k *could* be arbitrarily horrible depending on the dimension and f.

Nonuniform complexity: limitation

Example:

$$f \text{ PTIME analytic} \Rightarrow y \text{ PTIME} \Rightarrow y(t) \pm 2^{-n} \text{ in time } An^k$$

But:

- "Hides" some of the complexity: A,k *could* be arbitrarily horrible depending on the dimension and f.
- Nonconstructive: might be a different algrithm for each f, or depend on uncomputable constants.

Nonuniform complexity: limitation

Example:

$$f \text{ PTIME analytic } \Rightarrow\ y \text{ PTIME } \Rightarrow\ y(t) \pm 2^{-n} \text{ in time } An^k$$

But:

- "Hides" some of the complexity: A,k *could* be arbitrarily horrible depending on the dimension and f.
- Nonconstructive: might be a different algrithm for each f, or depend on uncomputable constants.

Conclusion

This only **slightly** better than the previous approach.

Uniform (operator) complexity approach

Assume $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)),$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$ is .... Then $y(t) \pm 2^{-n}$ can be computed in time

$$T(t, n, K_d, K_f)$$

where

- K_d: depends on the dimension d
- K_f: depends on f and its representation

Uniform (operator) complexity approach

Assume $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)),$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$ is .... Then $y(t) \pm 2^{-n}$ can be computed in time

$$T(t, n, K_d, K_f)$$

where

- K_d: depends on the dimension d
- K_f: depends on f and its representation

Assumption on f	Lower bound on T	Upper bound on T

Uniform (operator) complexity approach

Assume $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)),$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$ is .... Then $y(t) \pm 2^{-n}$ can be computed in time

$$T(t, n, K_d, K_f)$$

where

- K_d: depends on the dimension d
- K_f: depends on f and its representation

Assumption on f	**Lower bound on T**	**Upper bound on T**
computable	arbitrary	computable if unique

Uniform (operator) complexity approach

Assume $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)),$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$ is .... Then $y(t) \pm 2^{-n}$ can be computed in time

$$T(t, n, K_d, K_f)$$

where

- K_d: depends on the dimension d
- K_f: depends on f and its representation

Assumption on f	**Lower bound on** T	**Upper bound on** T
computable	arbitrary	computable if unique
PTIME + analytic	arbitrary	computable

Uniform (operator) complexity approach

Assume $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)),$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$ is .... Then $y(t) \pm 2^{-n}$ can be computed in time

$$T(t, n, K_d, K_f)$$

where

- K_d: depends on the dimension d
- K_f: depends on f and its representation

Assumption on f	**Lower bound on T**	**Upper bound on T**
computable	arbitrary	computable if unique
PTIME + analytic	arbitrary	computable
PTIME + polynomial	arbitrary	computable

Uniform (operator) complexity approach

Assume $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)),$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$ is .... Then $y(t) \pm 2^{-n}$ can be computed in time

$$T(t, n, K_d, K_f)$$

where

- K_d: depends on the dimension d
- K_f: depends on f and its representation

Assumption on f	**Lower bound on T**	**Upper bound on T**
computable	arbitrary	computable if unique
PTIME + analytic	arbitrary	computable
PTIME + polynomial	arbitrary	computable
PTIME + linear	—	exponential?

Uniform (operator) complexity approach

Assume $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)),$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$ is .... Then $y(t) \pm 2^{-n}$ can be computed in time

$$T(t, n, K_d, K_f)$$

where

- K_d: depends on the dimension d
- K_f: depends on f and its representation

Assumption on f	**Lower bound on T**	**Upper bound on T**
computable	arbitrary	computable if unique
PTIME + analytic	arbitrary	computable
PTIME + polynomial	arbitrary	computable
PTIME + linear	—	exponential?

Problem: we cannot predict the behaviour of y based on f only.

Are you confused?

You should be!

- practical methods: “no complexity”
- nonuniform complexity: misleading
- uniform worst-case complexity: everything looks hard

Are you confused?

You should be!

- practical methods: “no complexity”
- nonuniform complexity: misleading
- uniform worst-case complexity: everything looks hard

Question: are we looking at the problem the wrong way?

Parametrized complexity approach

Goal: Assume $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)),$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$ is nice. Then $y(t) \pm 2^{-n}$ can be computed in time

$$\text{poly}(t, n, K_d, K_f, K_y(t))$$

Parametrized complexity approach

Goal: Assume $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)),$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$ is nice. Then $y(t) \pm 2^{-n}$ can be computed in time

$$\text{poly}(t, n, K_d, K_f, K_y(t))$$

- K_d: depends on the dimension d
- K_f: depends on f and its representation
- K_y: is a reasonable parameter of y that must be unknown to the algorithm (i.e. not part of the input)

Parametrized complexity approach

Goal: Assume $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = f(y(t)),$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$ is nice. Then $y(t) \pm 2^{-n}$ can be computed in time

$$\operatorname{poly}(t, n, K_d, K_f, K_y(t))$$

- K_d: depends on the dimension d
- K_f: depends on f and its representation
- K_y: is a reasonable parameter of y that must be unknown to the algorithm (i.e. not part of the input)

Important differences with “textbook” approach:

- Result is always correct
- K_y not assumed to be known (e.g. K and M of previous slides)

Parametrized complexity result

Assume $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad y'(t) = p(y(t)),$$

where $p : \mathbb{R}^d \to \mathbb{R}^d$ is vector of multivariate polynomials.

Theorem (TCS 2016)

Assuming $t \in I$, computing $y(t) \pm 2^{-n}$ takes time:

$$\operatorname{poly}(\deg p, \log \Sigma p, n, \ell_y(t))^d$$

where:

- Σp: sum of absolute value of coefficients of p

Parametrized complexity result

Assume $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \qquad\qquad y'(t) = p(y(t)),$$

where $p : \mathbb{R}^d \to \mathbb{R}^d$ is vector of multivariate polynomials.

Theorem (TCS 2016)

Assuming $t \in I$, computing $y(t) \pm 2^{-n}$ takes time:

$$\operatorname{poly}(\deg p, \log \Sigma p, n, \ell_y(t))^d$$

where:

- Σp: sum of absolute value of coefficients of p
- $\ell_y(t)$: "length" of y over $[0, t]$

$$\ell_y(t) = \int_0^t \max(1, \|y'(u)\|) du$$

Note: the algorithm find $\ell(t)$ automatically, **it is not part of the input**

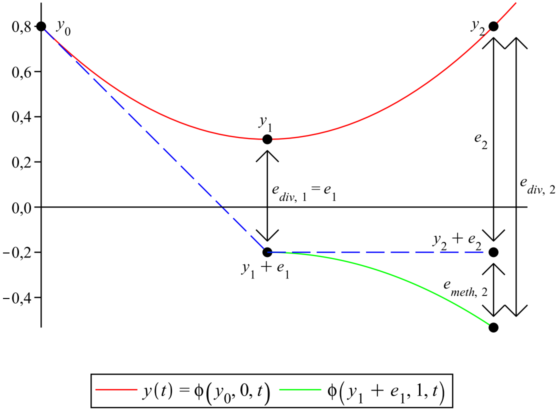
Euler method

$$y(0) = 0 \qquad y'(t) = p(y(t))$$

Time step h, discretize and compute $\tilde{y}^i \approx y(ih)$:

$$y(t+h) \approx y(t) + hy'(t) \quad \rightsquigarrow \quad \tilde{y}^{i+1} = \tilde{y}^i + hp(\tilde{y}^i)$$

Linear approximation at each step.



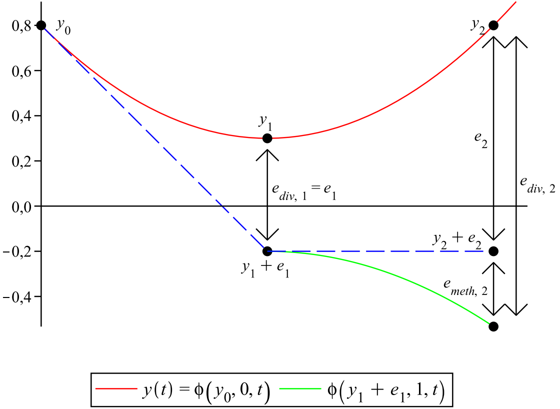
Euler method

$$y(0) = 0 \qquad y'(t) = p(y(t))$$

Time step h, discretize and compute $\tilde{y}^i \approx y(ih)$:

$$y(t+h) \approx y(t) + hy'(t) \quad \rightsquigarrow \quad \tilde{y}^{i+1} = \tilde{y}^i + hp(\tilde{y}^i)$$

Linear approximation at each step. Does not work well in practice.



Taylor method

$$y(0) = 0 \qquad y'(t) = p(y(t))$$

Time step h, discretize and compute $\tilde{y}^i \approx y(ih)$:

$$y(t+h) \approx y(t) + \sum_{i=1}^{\omega} h^i y^{(i)}(t) \qquad \text{using } y^{(i)}(t) = \text{poly}_i(y(t))$$

Do a ω-th order Taylor approximation at each step.

Taylor method

$$y(0) = 0 \qquad y'(t) = p(y(t))$$

Time step h, discretize and compute $\tilde{y}^i \approx y(ih)$:

$$y(t+h) \approx y(t) + \sum_{i=1}^{\omega} h^i y^{(i)}(t) \qquad \text{using } y^{(i)}(t) = \text{poly}_i(y(t))$$

Do a ω-th order Taylor approximation at each step.

Works well for $\omega \geqslant 3$ but

- How to choose h and ω ? One more parameter to choose!
- Error analysis is less obvious
- Complexity increases with ω

Adaptive Taylor method

Adapt h and ω at each step.

$$y(0) = 0 \qquad y'(t) = p(y(t))$$

Time step h_i, discretize and compute $\tilde{y}^i \approx y(\sum_{j \leqslant i} h_i)$:

$$y(t + h_i) \approx y(t) + \sum_{i=1}^{\omega_i} h_i^i y^{(i)}(t) \qquad \text{using } y^{(i)}(t) = \text{poly}_i(y(t))$$

Do a ω_i-th order Taylor approximation at each step.

Adaptive Taylor method

Adapt h and ω at each step.

$$y(0) = 0 \qquad y'(t) = p(y(t))$$

Time step h_i, discretize and compute $\tilde{y}^i \approx y(\sum_{j\leqslant i} h_i)$:

$$y(t + h_i) \approx y(t) + \sum_{i=1}^{\omega_i} h_i^i y^{(i)}(t) \qquad \text{using } y^{(i)}(t) = \text{poly}_i(y(t))$$

Do a ω_i-th order Taylor approximation at each step.

Adapt the amount of computation to the hardness of the problem but

- Many more parameters to choose
- Error analysis is challenging
- Complexity analysis usually not done

Adaptive Taylor method: parameter choice

How to choose the time steps h_i and orders ω_i:

- h_i: estimate the radius of convergence
- ω_i: try to guess the accuracy loss

Use voodoo magic and interval arithmetic to ensure correctness.

Adaptive Taylor method: parameter choice

How to choose the time steps h_i and orders ω_i:

- h_i: estimate the radius of convergence
- ω_i: try to guess the accuracy loss

Use voodoo magic and interval arithmetic to ensure correctness.

It works but most complexity insights are lost because we have no idea what we are doing.

Adaptive Taylor method: parameter choice

How to choose the time steps h_i and orders ω_i:

- h_i: estimate the radius of convergence
- ω_i: try to guess the accuracy loss

Use voodoo magic and interval arithmetic to ensure correctness.

It works but most complexity insights are lost because we have no idea what we are doing.

Our idea: we need to choose h_i, ω_i based on some high-level geometrical feature.

Adaptive Taylor method: parameter choice

How to choose the time steps h_i and orders ω_i:

- h_i: estimate the radius of convergence
- ω_i: try to guess the accuracy loss

Use voodoo magic and interval arithmetic to ensure correctness.

It works but most complexity insights are lost because we have no idea what we are doing.

Our idea: we need to choose h_i, ω_i based on some high-level geometrical feature.

Our algorithm in one sentence: choose h_i, ω_i so that

at each step, we increase the length of the solution by 1

Interesting (practical ?) consequences

Compute $y(t) \pm \varepsilon$

Method	Max. Order	Number of steps
Fixed ω	$\omega - 1$	$\mathcal{O}\left(L^{\frac{\omega+1}{\omega-1}}\varepsilon^{-\frac{1}{\omega-1}}\right)$

where $\quad L \approx \int_0^t \max(1, \|y'(u)\|)du$

Interesting (practical ?) consequences

Compute $y(t) \pm \varepsilon$

Method	Max. Order	Number of steps
Fixed ω	$\omega - 1$	$\mathcal{O}\left(L^{\frac{\omega+1}{\omega-1}} \varepsilon^{-\frac{1}{\omega-1}}\right)$
Euler ($\omega = 2$)	1	$\mathcal{O}\left(\frac{L^3}{\varepsilon}\right)$

where $\quad L \approx \int_0^t \max(1, \|y'(u)\|) du$

Interesting (practical ?) consequences

Compute $y(t) \pm \varepsilon$

Method	**Max. Order**	**Number of steps**
Fixed ω	$\omega - 1$	$\mathcal{O}\left(L^{\frac{\omega+1}{\omega-1}}\varepsilon^{-\frac{1}{\omega-1}}\right)$
Euler ($\omega = 2$)	1	$\mathcal{O}\left(\frac{L^3}{\varepsilon}\right)$
Taylor2 ($\omega = 3$)	2	$\mathcal{O}\left(\frac{L^2}{\sqrt{\varepsilon}}\right)$

where $$L \approx \int_0^t \max(1, \|y'(u)\|)du$$

Interesting (practical ?) consequences

Compute $y(t) \pm \varepsilon$

Method	Max. Order	Number of steps
Fixed ω	$\omega - 1$	$\mathcal{O}\left(L^{\frac{\omega+1}{\omega-1}}\varepsilon^{-\frac{1}{\omega-1}}\right)$
Euler ($\omega = 2$)	1	$\mathcal{O}\left(\frac{L^3}{\varepsilon}\right)$
Taylor2 ($\omega = 3$)	2	$\mathcal{O}\left(\frac{L^2}{\sqrt{\varepsilon}}\right)$
Taylor4 ($\omega = 5$)	4	$\mathcal{O}\left(\frac{L^{3/2}}{\sqrt[4]{\varepsilon}}\right)$

where $$L \approx \int_0^t \max(1, \|y'(u)\|)du$$

Interesting (practical ?) consequences

Compute $y(t) \pm \varepsilon$

Method	Max. Order	Number of steps
Fixed ω	$\omega - 1$	$\mathcal{O}\left(L^{\frac{\omega+1}{\omega-1}} \varepsilon^{-\frac{1}{\omega-1}}\right)$
Euler ($\omega = 2$)	1	$\mathcal{O}\left(\frac{L^3}{\varepsilon}\right)$
Taylor2 ($\omega = 3$)	2	$\mathcal{O}\left(\frac{L^2}{\sqrt{\varepsilon}}\right)$
Taylor4 ($\omega = 5$)	4	$\mathcal{O}\left(\frac{L^{3/2}}{\sqrt[4]{\varepsilon}}\right)$
Smart $\left(\omega = 1 + \log \frac{L}{\varepsilon}\right)$	$\log \frac{L}{\varepsilon}$	$\mathcal{O}\left(L^{\sim 1}\right)$

where $$L \approx \int_0^t \max(1, \|y'(u)\|) du$$

Interesting (practical ?) consequences

Compute $y(t) \pm \varepsilon$

Method	Max. Order	Number of steps
Fixed ω	$\omega - 1$	$\mathcal{O}\left(L^{\frac{\omega+1}{\omega-1}}\varepsilon^{-\frac{1}{\omega-1}}\right)$
Euler ($\omega = 2$)	1	$\mathcal{O}\left(\frac{L^3}{\varepsilon}\right)$
Taylor2 ($\omega = 3$)	2	$\mathcal{O}\left(\frac{L^2}{\sqrt{\varepsilon}}\right)$
Taylor4 ($\omega = 5$)	4	$\mathcal{O}\left(\frac{L^{3/2}}{\sqrt[4]{\varepsilon}}\right)$
Smart $\left(\omega = 1 + \log\frac{L}{\varepsilon}\right)$	$\log\frac{L}{\varepsilon}$	$\mathcal{O}\left(L^{\sim 1}\right)$
Taylor∞ ($\omega = \infty$)	∞	$\mathcal{O}(L)$

where $$L \approx \int_0^t \max(1, \|y'(u)\|)du$$

Interesting (practical ?) consequences

Compute $y(t) \pm \varepsilon$

Method	Max. Order	Number of steps
Fixed ω	$\omega - 1$	$\mathcal{O}\left(L^{\frac{\omega+1}{\omega-1}} \varepsilon^{-\frac{1}{\omega-1}}\right)$
Euler ($\omega = 2$)	1	$\mathcal{O}\left(\frac{L^3}{\varepsilon}\right)$
Taylor2 ($\omega = 3$)	2	$\mathcal{O}\left(\frac{L^2}{\sqrt{\varepsilon}}\right)$
Taylor4 ($\omega = 5$)	4	$\mathcal{O}\left(\frac{L^{3/2}}{\sqrt[4]{\varepsilon}}\right)$
Smart $\left(\omega = 1 + \log \frac{L}{\varepsilon}\right)$	$\log \frac{L}{\varepsilon}$	$\mathcal{O}\left(L^{\sim 1}\right)$
Taylor∞ ($\omega = \infty$)	∞	$\mathcal{O}(L)$
Variable	$\mathcal{O}\left(\log \frac{L}{\varepsilon}\right)$	$\mathcal{O}(L)$

where $$L \approx \int_0^t \max(1, \|y'(u)\|)du$$

Conclusion

Solving Ordinary Differential Equations numerically:

- vastly different algorithms/results for vastly different expectations
- practical methods: no complexity
- nonuniform complexity: imprecise/misleading
- uniform worst-case complexity: everything is hard
- uniform parametrized complexity: encouraging

Questions:

- how far can we push parametrized complexity?
- can theory bring insight to practice?
- geometric complexity?

Taylor method

$$y(0) = 0 \qquad y'(t) = p(y(t)) \quad t \in I$$

Lemma: $y^{(k)}(t) = P_k(y(t)) = \text{poly}(y(t))$

Taylor method

$$y(0) = 0 \qquad y'(t) = p(y(t)) \quad t \in I$$

Lemma: $y^{(k)}(t) = P_k(y(t)) = \mathrm{poly}(y(t))$

Order K, time step h, discretize compute $\tilde{y}^i \approx y(ih)$:

$$y(t+h) \approx \sum_{j=0}^{K} \frac{h^j}{j!} y^{(j)}(t) \quad \rightsquigarrow \quad \tilde{y}^{i+1} = \sum_{j=0}^{K} \frac{h^j}{j!} P_k(\tilde{y}^i)$$

Taylor method

$$y(0) = 0 \qquad y'(t) = p(y(t)) \quad t \in I$$

Lemma: $y^{(k)}(t) = P_k(y(t)) = \text{poly}(y(t))$

Order K, time step h, discretize compute $\tilde{y}^i \approx y(ih)$:

$$y(t+h) \approx \sum_{j=0}^{K} \frac{h^j}{j!} y^{(j)}(t) \quad \rightsquigarrow \quad \tilde{y}^{i+1} = \sum_{j=0}^{K} \frac{h^j}{j!} P_k(\tilde{y}^i)$$

- **Fixed order** K: theoretically not enough

Taylor method

$$y(0) = 0 \qquad y'(t) = p(y(t)) \quad t \in I$$

Lemma: $y^{(k)}(t) = P_k(y(t)) = \mathrm{poly}(y(t))$

Order K, time step h, discretize compute $\tilde{y}^i \approx y(ih)$:

$$y(t+h) \approx \sum_{j=0}^{K} \frac{h^j}{j!} y^{(j)}(t) \quad \rightsquigarrow \quad \tilde{y}^{i+1} = \sum_{j=0}^{K} \frac{h^j}{j!} P_k(\tilde{y}^i)$$

- **Fixed order** K: theoretically not enough
- **Variable order** K: choose K depending on i, p, n and $\tilde{y}^i$

Taylor method

$$y(0) = 0 \qquad y'(t) = p(y(t)) \quad t \in I$$

Lemma: $y^{(k)}(t) = P_k(y(t)) = \text{poly}(y(t))$

Order K, time step h, discretize compute $\tilde{y}^i \approx y(ih)$:

$$y(t+h) \approx \sum_{j=0}^{K} \frac{h^j}{j!} y^{(j)}(t) \quad \rightsquigarrow \quad \tilde{y}^{i+1} = \sum_{j=0}^{K} \frac{h^j}{j!} P_k(\tilde{y}^i)$$

- **Fixed order** K: theoretically not enough
- **Variable order** K: choose K depending on i, p, n and $\tilde{y}^i$

What about h ?

- **Fixed** h: wasteful

Taylor method

$$y(0) = 0 \qquad y'(t) = p(y(t)) \quad t \in I$$

Lemma: $y^{(k)}(t) = P_k(y(t)) = \text{poly}(y(t))$

Order K, time step h, discretize compute $\tilde{y}^i \approx y(ih)$:

$$y(t+h) \approx \sum_{j=0}^{K} \frac{h^j}{j!} y^{(j)}(t) \quad \rightsquigarrow \quad \tilde{y}^{i+1} = \sum_{j=0}^{K} \frac{h^j}{j!} P_k(\tilde{y}^i)$$

- **Fixed order** K: theoretically not enough
- **Variable order** K: choose K depending on i, p, n and $\tilde{y}^i$

What about h ?

- **Fixed** h: wasteful
- **Adaptive** h: choose h depending on i, p, n and $\tilde{y}^i$

Choice of the parameters

Choice of h based on an effective lower bound on radius of convergence of the Taylor series:

Lemma: If $y' = p(y)$, $\alpha = \max(1, \|y_0\|)$, $k = \deg(p)$, $M = (k-1)\Sigma p\alpha^{k-1}$ then:

$$\left\| y^{(k)}(t) - P_k(y(t)) \right\| \leqslant \frac{\alpha (Mt)^k}{1 - Mt}$$

Choice of the parameters

Choice of h based on an effective lower bound on radius of convergence of the Taylor series:

Lemma: If $y' = p(y)$, $\alpha = \max(1, \|y_0\|)$, $k = \deg(p)$, $M = (k-1)\Sigma p \alpha^{k-1}$ then:

$$\left\| y^{(k)}(t) - P_k(y(t)) \right\| \leqslant \frac{\alpha (Mt)^k}{1 - Mt}$$

Choose $Mt \approx \frac{1}{2}$:

- $t \approx \frac{1}{M}$: adaptive step size
- local error $\approx (Mt)^k \approx 2^{-k}$: order gives the number of correct bits

Choice of the parameters

Choice of h based on an effective lower bound on radius of convergence of the Taylor series:

Lemma: If $y' = p(y)$, $\alpha = \max(1, \|y_0\|)$, $k = \deg(p)$, $M = (k-1)\Sigma p \alpha^{k-1}$ then:

$$\left\| y^{(k)}(t) - P_k(y(t)) \right\| \leqslant \frac{\alpha (Mt)^k}{1 - Mt}$$

Choose $Mt \approx \frac{1}{2}$:

- $t \approx \frac{1}{M}$: adaptive step size
- local error $\approx (Mt)^k \approx 2^{-k}$: order gives the number of correct bits

I spare you the analysis of the global error !

But wait...

This is impossible, right ?!

But wait...

This is impossible, right ?!

Example

$$\begin{cases} x(t) = t^{u(t)} \\ u(t) = e^{-t} - (1 - e^{-t})\frac{1}{v(t)} \\ v(t) = v_0 \end{cases} \rightsquigarrow \begin{cases} x(t) \sim t^{\frac{1}{v_0}} \\ u(t) \to \frac{1}{v_0} \\ v(t) = v_0 \end{cases}$$

But wait...

This is impossible, right ?!

Example

$$\begin{cases} x(t) = t^{u(t)} \\ u(t) = e^{-t} - (1 - e^{-t})\frac{1}{v(t)} \\ v(t) = v_0 \end{cases} \rightsquigarrow \begin{cases} x(t) \sim t^{\frac{1}{v_0}} \\ u(t) \to \frac{1}{v_0} \\ v(t) = v_0 \end{cases}$$

Remark

- All parameters are fixed except $y_0 = (1, 1, v_0)$

But wait...

This is impossible, right ?!

Example

$$\begin{cases} x(t) = t^{u(t)} \\ u(t) = e^{-t} - (1 - e^{-t})\frac{1}{v(t)} \\ v(t) = v_0 \end{cases} \rightsquigarrow \begin{cases} x(t) \sim t^{\frac{1}{v_0}} \\ u(t) \to \frac{1}{v_0} \\ v(t) = v_0 \end{cases}$$

Remark

- All parameters are fixed except $y_0 = (1, 1, v_0)$
- Value are time $t = 2$ can be arbitrary large for arbitrary small v_0

But wait...

This is impossible, right ?!

Example

$$\left\{\begin{array}{l} x(t) = t^{u(t)} \\ u(t) = e^{-t} - (1 - e^{-t})\frac{1}{v(t)} \\ v(t) = v_0 \end{array}\right. \rightsquigarrow \left\{\begin{array}{l} x(t) \sim t^{\frac{1}{v_0}} \\ u(t) \to \frac{1}{v_0} \\ v(t) = v_0 \end{array}\right.$$

Remark

- All parameters are fixed except $y_0 = (1, 1, v_0)$
- Value are time $t = 2$ can be arbitrary large for arbitrary small v_0

Theorem

There is no universal bound in p, y_0, t_0, t and μ.