Computational complexity of solving polynomial differential equations over unbounded domains

Amaury Pouly
Joint work with Daniel Graça

10 May 2018
System of ODEs:

\[
\begin{align*}
  y_1(0) &= y_{0,1} \\
  & \vdots \\
  y_n(0) &= y_{0,n}
\end{align*}
\]

More compactly:

\[
\begin{align*}
  y(0) &= y_0 \\
  y'(t) &= f(y(t), t)
\end{align*}
\]
System of ODEs:

\[
\begin{cases}
y_1(0) = y_{0,1} \\
\vdots \\
y_n(0) = y_{0,n}
\end{cases}
\quad \begin{cases}
y'_1(t) = f_1(y_1(t), \ldots, y_n(t), t) \\
\vdots \\
y'_n(t) = f_n(y_1(t), \ldots, y_n(t), t)
\end{cases}
\]

More compactly:

\[
y(0) = y_0 \\
y'(t) = f(y(t), t)
\]

Get rid of the time:

\[
\begin{cases}
y(0) = y_0 \\
z(0) = 0
\end{cases} \quad \begin{cases}
y'(t) = f(y(t), z(t)) \\
z'(t) = 1
\end{cases}
\]
Ordinary Differential Equations (ODEs)

System of ODEs:

\[
\begin{array}{l}
y_1(0) = y_{0,1} \\
\vdots \\
y_n(0) = y_{0,n}
\end{array}
\quad \begin{array}{l}
y_1'(t) = f_1(y_1(t), \ldots, y_n(t), t) \\
\vdots \\
y_n'(t) = f_n(y_1(t), \ldots, y_n(t), t)
\end{array}
\]

More compactly:

\[
y(0) = y_0 \quad y'(t) = f(y(t), t)
\]

Get rid of the time:

\[
\begin{array}{l}
y(0) = y_0 \\
z(0) = 0
\end{array} \quad \begin{array}{l}
y'(t) = f(y(t), z(t)) \\
z'(t) = 1
\end{array}
\]

In this talk: autonomous first order explicit system of ODEs

\[
y(0) = y_0 \quad y' = f(y) \quad y : (a, b) \to \mathbb{R}^n
\]
A word on computability for real functions

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

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.
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 \textit{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. \textbf{Comparisons between reals are allowed.}

- Computable Analysis: reals are represented as converging Cauchy sequences, computations are carried out by rational approximations using Turing machines. \textbf{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 \( \forall t \in I: \)

\[
y(0) = 0, \quad y'(t) = f(y(t)). \quad (1)
\]

Given \( t \in I \) and \( n \in \mathbb{N} \), can we compute \( q \in \mathbb{Q}^d \) s.t. \( \| q - y(t) \| \leq 2^{-n} \)?
Let $I = (a, b)$ and $f \in C^0(\mathbb{R}^d)$. Assume $y \in C^1(I, \mathbb{R}^d)$ satisfies $\forall t \in I$:

$$
\begin{align*}
    y(0) &= 0, \\
    y'(t) &= f(y(t)).
\end{align*}
$$

(1)

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

$$
\begin{align*}
y(0) &= 0, \\
y'(t) &= f(y(t)).
\end{align*}
$$

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

$$y(0) = 0, \quad y'(t) = f(y(t)).$$  \quad (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 $\forall t \in I$:

$$y(0) = 0, \quad y'(t) = f(y(t)).$$  \hspace{1cm} (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 $\forall t \in I$:

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

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

\[ y(0) = 0, \quad y'(t) = f(y(t)). \]
Assume $f$ \textbf{Lipschitz} and computable, and $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

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

**Theorem (Folklore, simplified)**

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

$$\|q - y(t)\| \leq O(h^4)$$

Usually followed by benchmarks.

Problems with this approach:

- Accuracy of the result?

- $O(h^4) \leq Ah^4$ but $A$ is unknown

- Same problem with complexity $f$ is Lipschitz: typically only holds over compact domains.
Assume $f$ \textbf{Lipschitz} and computable, and $y : I \rightarrow \mathbb{R}^d$ satisfies $\forall t \in I$:

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

\textbf{Theorem (Folklore, simplified)}

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

Problems with this approach:

Accuracy of the result? $\mathcal{O}(h^4) \leq Ah^4$ but $A$ is unknown.

Same problem with complexity $f$ is Lipschitz: typically only holds over compact domains.
Complexity of solutions: typical textbook result

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

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

**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.

$$\|q - y(t)\| \leq O\left(h^4\right)$$

and has running time $O\left(\frac{1}{h^4}\right)$.

Usually followed by benchmarks.
Assume \( f \) \textbf{Lipschitz} and computable, and \( y : I \rightarrow \mathbb{R}^d \) satisfies \( \forall t \in I: \)

\[
y(0) = 0, \quad y'(t) = f(y(t)).
\]

\begin{tcolorbox}[colback=blue!5!white, colframe=blue!75!black]
**Theorem (Folklore, simplified)**

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

Usually followed by benchmarks.

**Problems with this approach:**

- Accuracy of the result? \( O(h^4) \leq Ah^4 \) but \( A \) is \textbf{unknown}
- Same problem with complexity
- \( f \) is \textbf{Lipschitz}: typically only holds over \textbf{compact} domains
Assume $f$ computable and $K$-Lipschitz, and $y : I \rightarrow \mathbb{R}^d$ satisfies $\forall t \in I$:

$$y(0) = 0, \quad y'(t) = f(y(t)).$$
Assume \( f \) computable and \( K \)-Lipschitz, and \( y : I \rightarrow \mathbb{R}^d \) satisfies \( \forall t \in I : \)

\[
y(0) = 0, \quad y'(t) = f(y(t)).
\]

**Theorem (Folklore, simplified)**

Euler’s method global truncation error is:

\[
\frac{hM}{2K} \left( e^{Kt} - 1 \right) \quad \text{where} \quad M = \sup_{u \in I} \| y''(u) \|.
\]
Assume $f$ computable and $K$-Lipschitz, and $y : I \rightarrow \mathbb{R}^d$ satisfies $\forall t \in I$:

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

**Theorem (Folklore, simplified)**

Euler’s method global truncation error is:

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

In particular it has order 1 over compact time ($I$) domains.
Complexity of solutions: typical textbook result

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

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

Theorem (Folklore, simplified)

Euler’s method global truncation error is:

$$\frac{hM}{2K} \left( e^{Kt} - 1 \right) = O(h)$$

where $M = \sup_{u \in I} \|y''(u)\|$. 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
Assume $f$ computable and $K$-Lipschitz, and $y : I \to \mathbb{R}^d$ satisfies $\forall t \in I$:

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

**Theorem (Folklore, simplified)**

Euler’s method global truncation error is:

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

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 $\forall t \in I$:

$$y(0) = 0, \quad y'(t) = f(y(t)) \quad \text{with unbounded } I = [0, +\infty).$$
Complexity of solutions: the rescaling “myth”

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

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

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 $\forall t \in I$:

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

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 $z(0) = y_0, z' = f_T(z)$ on $[0, 1]$ is a compact!

Bad analysis: $y(T) = z(1)$

Accuracy: $O(h)$ (compact)

Note: now $f$ really needs to be globally Lipschitz.

Conclusion: This tells us nothing about the complexity of the problem.
Assume $f$ computable and $K$-Lipschitz, and $y : I \rightarrow \mathbb{R}^d$ satisfies $\forall t \in I$:

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

**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.
Complexity of solutions: the rescaling “myth”

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

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

**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 $z(0) = y_0, z' = f_T(z)$ on $[0, 1]$ 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 $y : I \rightarrow \mathbb{R}^d$ satisfies $\forall t \in I$:

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

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 \( \forall t \in [0, 1] \):

\[
y(0) = 0, \quad y'(t) = f(y(t)).
\]

There exists methods of the form:
given \( h \) and \( t \), compute \( q \in \mathbb{Q}^d \) and \( \varepsilon > 0 \) such that \( \|y(t) - q\| \leq \varepsilon \) 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 \( y : [0, 1] \rightarrow \mathbb{R}^d \) satisfies \( \forall t \in [0, 1] \):

\[
y(0) = 0, \quad y'(t) = f(y(t)).
\]

<table>
<thead>
<tr>
<th>Assumption on ( f )</th>
<th>Lower bound on ( y )</th>
<th>Upper bound on ( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>computable</td>
<td>arbitrary</td>
<td>computable if unique</td>
</tr>
</tbody>
</table>
Nonuniform complexity-theoretic approach

Assume \( y : [0, 1] \rightarrow \mathbb{R}^d \) satisfies \( \forall t \in [0, 1] \):

\[
y(0) = 0, \quad y'(t) = f(y(t)).
\]

<table>
<thead>
<tr>
<th>Assumption on ( f )</th>
<th>Lower bound on ( y )</th>
<th>Upper bound on ( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>computable PTIME</td>
<td>arbitrary</td>
<td>computable if unique</td>
</tr>
<tr>
<td></td>
<td>arbitrary</td>
<td>computable</td>
</tr>
</tbody>
</table>

But those results can be deceiving...
Nonuniform complexity-theoretic approach

Assume $y : [0, 1] \rightarrow \mathbb{R}^d$ satisfies $\forall t \in [0, 1]$:  
\[
y(0) = 0, \quad y'(t) = f(y(t)).
\]

<table>
<thead>
<tr>
<th>Assumption on $f$</th>
<th>Lower bound on $y$</th>
<th>Upper bound on $y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>computable</td>
<td>arbitrary</td>
<td>computable if unique</td>
</tr>
<tr>
<td>PTIME</td>
<td>arbitrary</td>
<td>computable</td>
</tr>
<tr>
<td>PTIME + Lipschitz</td>
<td>PSPACE-hard</td>
<td>PSPACE</td>
</tr>
</tbody>
</table>
Assume $y : [0, 1] \to \mathbb{R}^d$ satisfies $\forall t \in [0, 1]$:  

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

<table>
<thead>
<tr>
<th>Assumption on $f$</th>
<th>Lower bound on $y$</th>
<th>Upper bound on $y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>computable</td>
<td>arbitrary</td>
<td>computable if unique</td>
</tr>
<tr>
<td>PTIME</td>
<td>arbitrary</td>
<td>computable</td>
</tr>
<tr>
<td>PTIME + Lipschitz</td>
<td>PSPACE-hard</td>
<td>PSPACE</td>
</tr>
<tr>
<td>PTIME + $C^1$</td>
<td>PSPACE-hard</td>
<td>PSPACE</td>
</tr>
</tbody>
</table>
Nonuniform complexity-theoretic approach

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

\[
\begin{align*}
y(0) &= 0, & y'(t) &= f(y(t)).
\end{align*}
\]

<table>
<thead>
<tr>
<th>Assumption on ( f )</th>
<th>Lower bound on ( y )</th>
<th>Upper bound on ( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>computable PTIME</td>
<td>arbitrary</td>
<td>computable if unique computable</td>
</tr>
<tr>
<td>PTIME + Lipschitz</td>
<td>PSPACE-hard</td>
<td>PSPACE</td>
</tr>
<tr>
<td>PTIME + ( C^1 )</td>
<td>PSPACE-hard</td>
<td>PSPACE</td>
</tr>
<tr>
<td>PTIME + ( C^k ), ( k \geq 2 )</td>
<td>CH-hard</td>
<td>PSPACE</td>
</tr>
</tbody>
</table>
### Nonuniform complexity-theoretic approach

Assume \( y : [0, 1] \rightarrow \mathbb{R}^d \) satisfies \( \forall t \in [0, 1] \):

\[
y(0) = 0, \quad y'(t) = f(y(t)).
\]

<table>
<thead>
<tr>
<th>Assumption on ( f )</th>
<th>Lower bound on ( y )</th>
<th>Upper bound on ( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>computable PTIME</td>
<td>arbitrary</td>
<td>computable if unique computable</td>
</tr>
<tr>
<td>PTIME + Lipschitz</td>
<td>PSspace-hard</td>
<td>PSspace</td>
</tr>
<tr>
<td>PTIME + ( C^1 )</td>
<td>PSspace-hard</td>
<td>PSspace</td>
</tr>
<tr>
<td>PTIME + ( C^k ), ( k \geq 2 )</td>
<td>CH-hard</td>
<td>PSspace</td>
</tr>
<tr>
<td>PTIME + analytic</td>
<td>—</td>
<td>PTIME</td>
</tr>
</tbody>
</table>
Nonuniform complexity-theoretic approach

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

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

<table>
<thead>
<tr>
<th>Assumption on $f$</th>
<th>Lower bound on $y$</th>
<th>Upper bound on $y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>computable</td>
<td>arbitrary</td>
<td>computable if unique</td>
</tr>
<tr>
<td>PTIME</td>
<td>arbitrary</td>
<td>computable</td>
</tr>
<tr>
<td>PTIME + Lipschitz</td>
<td>PSPACE-hard</td>
<td>PSPACE</td>
</tr>
<tr>
<td>PTIME + $C^1$</td>
<td>PSPACE-hard</td>
<td>PSPACE</td>
</tr>
<tr>
<td>PTIME + $C^k$, $k \geq 2$</td>
<td>CH-hard</td>
<td>PSPACE</td>
</tr>
<tr>
<td>PTIME + analytic</td>
<td>—</td>
<td>PTIME</td>
</tr>
</tbody>
</table>

But those results can be deceiving...

$$\begin{cases} 
  y_1(0) = 1 \\
  y_2(0) = 1 \\
  \vdots \\
  y_d(0) = 1 
\end{cases} \quad \begin{cases} 
  y'_1 = y_1 \\
  y'_2 = y_1y_2 \\
  \vdots \\
  y'_n = y_{d-1}y_n 
\end{cases} \quad \rightarrow \quad y(t) = O \left( e^{e^{\cdots^{e^t}}} \right)$$

$y$ is PTIME over $[0, 1]$
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 \( f \).
Example:

\[ f \text{ PTIME analytic} \implies y \text{ PTIME} \implies y(t) \pm 2^{-n} \text{ in time } A n^k \]

But:

- “Hides” some of the complexity: \( A, k \) could be arbitrarily horrible depending on the dimension and \( f \).
- Nonconstructive: might be a different algorithm for each \( f \), or depend on uncomputable constants.
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 algorithm 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 \( \forall t \in I: \)

\[
y(0) = 0, \quad y'(t) = f(y(t)),
\]

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 $\forall t \in I:\n$

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

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

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

where $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ 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

<table>
<thead>
<tr>
<th>Assumption on $f$</th>
<th>Lower bound on $T$</th>
<th>Upper bound on $T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>computable</td>
<td>arbitrary</td>
<td>computable if unique</td>
</tr>
</tbody>
</table>
Uniform (operator) complexity approach

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

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

where $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ 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

<table>
<thead>
<tr>
<th>Assumption on $f$</th>
<th>Lower bound on $T$</th>
<th>Upper bound on $T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>computable</td>
<td>arbitrary</td>
<td>computable if unique</td>
</tr>
<tr>
<td>PTIME + analytic</td>
<td>arbitrary</td>
<td>computable</td>
</tr>
</tbody>
</table>
Uniform (operator) complexity approach

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

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

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

<table>
<thead>
<tr>
<th>Assumption on $f$</th>
<th>Lower bound on $T$</th>
<th>Upper bound on $T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>computable</td>
<td>arbitrary</td>
<td>computable if unique</td>
</tr>
<tr>
<td>PTIME + analytic</td>
<td>arbitrary</td>
<td>computable</td>
</tr>
<tr>
<td>PTIME + polynomial</td>
<td>arbitrary</td>
<td>computable</td>
</tr>
</tbody>
</table>
Uniform (operator) complexity approach

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

\[
y(0) = 0, \quad y'(t) = f(y(t)),
\]

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

<table>
<thead>
<tr>
<th>Assumption on ( f )</th>
<th>Lower bound on ( T )</th>
<th>Upper bound on ( T )</th>
</tr>
</thead>
<tbody>
<tr>
<td>computable</td>
<td>arbitrary</td>
<td>computable if unique</td>
</tr>
<tr>
<td>PTIME + analytic</td>
<td>arbitrary</td>
<td>computable</td>
</tr>
<tr>
<td>PTIME + polynomial</td>
<td>arbitrary</td>
<td>computable</td>
</tr>
<tr>
<td>PTIME + linear</td>
<td>—</td>
<td>exponential?</td>
</tr>
</tbody>
</table>
Uniform (operator) complexity approach

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

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

where $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ 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

<table>
<thead>
<tr>
<th>Assumption on $f$</th>
<th>Lower bound on $T$</th>
<th>Upper bound on $T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>computable</td>
<td>arbitrary</td>
<td>computable if unique</td>
</tr>
<tr>
<td>PTIME + analytic</td>
<td>arbitrary</td>
<td>computable</td>
</tr>
<tr>
<td>PTIME + polynomial</td>
<td>arbitrary</td>
<td>computable</td>
</tr>
<tr>
<td>PTIME + linear</td>
<td>—</td>
<td>exponential?</td>
</tr>
</tbody>
</table>

Problem: we cannot predict the behaviour of $y$ based on $f$ only.
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?
Goal: Assume $y : I \rightarrow \mathbb{R}^d$ satisfies $\forall t \in I$:

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

where $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is nice. Then $y(t) \pm 2^{-n}$ can be computed in time $\text{poly}(t, n, K_d, K_f, K_y(t))$.
Goal: Assume $y : I \rightarrow \mathbb{R}^d$ satisfies $\forall t \in I$: 

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

where $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ 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)
Goal: Assume \( y : I \to \mathbb{R}^d \) satisfies \( \forall t \in I: \)
\[
y(0) = 0, \quad y'(t) = f(y(t)),
\]
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)

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

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

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

Theorem (TCS 2016)

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

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

where:

- $\Sigma p$: sum of absolute value of coefficients of $p$
Parametrized complexity result

Assume \( y : I \rightarrow \mathbb{R}^d \) satisfies \( \forall t \in I : \)

\[
y(0) = 0, \quad y'(t) = p(y(t)),
\]

where \( p : \mathbb{R}^d \rightarrow \mathbb{R}^d \) is vector of multivariate polynomials.

**Theorem (TCS 2016)**

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

\[
\text{poly}(\deg p, \log \Sigma p, n, \ell_y(t))^d
\]

where:

- \( \Sigma p \): 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

\[ y(0) = 0 \quad y'(t) = p(y(t)) \]

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

\[
y(t + h) \approx y(t) + hy'(t) \quad \leadsto \quad \tilde{y}^{i+1} = \tilde{y}^i + hp(\tilde{y}^i)
\]

Linear approximation at each step.
Euler method

\[ y(0) = 0 \quad y'(t) = p(y(t)) \]

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

\[ y(t + h) \approx y(t) + h y'(t) \quad \sim \quad \tilde{y}^{i+1} = \tilde{y}^i + hp(\tilde{y}^i) \]

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

\[ y(0) = 0 \quad y'(t) = p(y(t)) \]

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

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

Do a \( \omega \)-th order Taylor approximation at each step.
Taylor method

\[ y(0) = 0 \quad y'(t) = p(y(t)) \]

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

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

Do a \( \omega \)-th order Taylor approximation at each step.

Works well for \( \omega \geq 3 \) but

- How to choose \( h \) and \( \omega \)? **One more parameter to choose!**
- Error analysis is less obvious
- Complexity increases with \( \omega \)
Adaptive Taylor method

Adapt $h$ and $\omega$ at each step.

\[ y(0) = 0 \quad y'(t) = p(y(t)) \]

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

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

Do a $\omega_i$-th order Taylor approximation at each step.
Adaptive Taylor method

Adapt $h$ and $\omega$ at each step.

\[ y(0) = 0 \quad y'(t) = p(y(t)) \]

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

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

Do a $\omega_i$-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 $\omega_i$:

- $h_i$: estimate the radius of convergence
- $\omega_i$: 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 $\omega_i$:
- $h_i$: estimate the radius of convergence
- $\omega_i$: 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 $\omega_i$:

- $h_i$: estimate the radius of convergence
- $\omega_i$: 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 $h_i, \omega_i$ based on some high-level geometrical feature.
Adaptive Taylor method: parameter choice

How to choose the time steps $h_i$ and orders $\omega_i$:

- $h_i$: estimate the radius of convergence
- $\omega_i$: 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 $h_i, \omega_i$ based on some high-level geometrical feature.

Our algorithm in one sentence: choose $h_i, \omega_i$ so that

at each step, we increase the length of the solution by 1
Interesting (practical?) consequences

Compute $y(t) \pm \varepsilon$

<table>
<thead>
<tr>
<th>Method</th>
<th>Max. Order</th>
<th>Number of steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed $\omega$</td>
<td>$\omega - 1$</td>
<td>$O \left( L^{\omega-1} \varepsilon^{\frac{1}{\omega-1}} \right)$</td>
</tr>
</tbody>
</table>

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

Compute \( y(t) \pm \varepsilon \)

<table>
<thead>
<tr>
<th>Method</th>
<th>Max. Order</th>
<th>Number of steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed ( \omega )</td>
<td>( \omega - 1 )</td>
<td>( O \left( L^{\omega+1} \varepsilon^{1/\omega-1} \right) )</td>
</tr>
<tr>
<td>Euler (( \omega = 2 ))</td>
<td>1</td>
<td>( O \left( \frac{L^3}{\varepsilon} \right) )</td>
</tr>
</tbody>
</table>

where \( L \approx \int_{0}^{t} \max(1, \|y'(u)\|) du \)
Interesting (practical?) consequences

Compute \( y(t) \pm \varepsilon \)

<table>
<thead>
<tr>
<th>Method</th>
<th>Max. Order</th>
<th>Number of steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed ( \omega )</td>
<td>( \omega - 1 )</td>
<td>( \mathcal{O} \left( L^{\frac{\omega+1}{\omega-1}} \frac{1}{\varepsilon^{\frac{1}{\omega-1}}} \right) )</td>
</tr>
<tr>
<td>Euler (( \omega = 2 ))</td>
<td>1</td>
<td>( \mathcal{O} \left( \frac{L^3}{\varepsilon} \right) )</td>
</tr>
<tr>
<td>Taylor2 (( \omega = 3 ))</td>
<td>2</td>
<td>( \mathcal{O} \left( \frac{L^2}{\sqrt{\varepsilon}} \right) )</td>
</tr>
</tbody>
</table>

\[ L \approx \int_0^t \max(1, \| y'(u) \|) du \]
Interesting (practical?) consequences

Compute $y(t) \pm \varepsilon$

<table>
<thead>
<tr>
<th>Method</th>
<th>Max. Order</th>
<th>Number of steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed $\omega$</td>
<td>$\omega - 1$</td>
<td>$O\left(L^{\frac{\omega+1}{\omega-1}}\varepsilon^{-\frac{1}{\omega-1}}\right)$</td>
</tr>
<tr>
<td>Euler ($\omega = 2$)</td>
<td>1</td>
<td>$O\left(\frac{L^3}{\varepsilon}\right)$</td>
</tr>
<tr>
<td>Taylor2 ($\omega = 3$)</td>
<td>2</td>
<td>$O\left(\frac{L^2}{\sqrt{\varepsilon}}\right)$</td>
</tr>
<tr>
<td>Taylor4 ($\omega = 5$)</td>
<td>4</td>
<td>$O\left(\frac{L^{3/2}}{4\sqrt{\varepsilon}}\right)$</td>
</tr>
</tbody>
</table>

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

Compute $y(t) \pm \varepsilon$

<table>
<thead>
<tr>
<th>Method</th>
<th>Max. Order</th>
<th>Number of steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed $\omega$</td>
<td>$\omega - 1$</td>
<td>$O \left( L^{\frac{\omega+1}{\omega-1}} \frac{1}{\varepsilon^{\frac{1}{\omega-1}}} \right)$</td>
</tr>
<tr>
<td>Euler ($\omega = 2$)</td>
<td>1</td>
<td>$O \left( \frac{L^3}{\varepsilon} \right)$</td>
</tr>
<tr>
<td>Taylor2 ($\omega = 3$)</td>
<td>2</td>
<td>$O \left( \frac{L^2}{\sqrt{\varepsilon}} \right)$</td>
</tr>
<tr>
<td>Taylor4 ($\omega = 5$)</td>
<td>4</td>
<td>$O \left( \frac{L^{3/2}}{4\sqrt{\varepsilon}} \right)$</td>
</tr>
<tr>
<td>Smart ($\omega = 1 + \log \frac{L}{\varepsilon}$)</td>
<td>$\log \frac{L}{\varepsilon}$</td>
<td>$O \left( L^{-1} \right)$</td>
</tr>
</tbody>
</table>

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

Compute $y(t) \pm \varepsilon$

<table>
<thead>
<tr>
<th>Method</th>
<th>Max. Order</th>
<th>Number of steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed $\omega$</td>
<td>$\omega - 1$</td>
<td>$O\left(\frac{L^{\omega+1}}{\omega-1}\varepsilon^{-1}\right)$</td>
</tr>
<tr>
<td>Euler ($\omega = 2$)</td>
<td>1</td>
<td>$O\left(\frac{L^3}{\varepsilon}\right)$</td>
</tr>
<tr>
<td>Taylor2 ($\omega = 3$)</td>
<td>2</td>
<td>$O\left(\frac{L^2}{\sqrt{\varepsilon}}\right)$</td>
</tr>
<tr>
<td>Taylor4 ($\omega = 5$)</td>
<td>4</td>
<td>$O\left(\frac{L^{3/2}}{4\sqrt{\varepsilon}}\right)$</td>
</tr>
<tr>
<td>Smart ($\omega = 1 + \log \frac{L}{\varepsilon}$)</td>
<td>$\log \frac{L}{\varepsilon}$</td>
<td>$O\left(L^{-1}\right)$</td>
</tr>
<tr>
<td>Taylor$\infty$ ($\omega = \infty$)</td>
<td>$\infty$</td>
<td>$O\left(L\right)$</td>
</tr>
</tbody>
</table>

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

Compute $y(t) \pm \varepsilon$

<table>
<thead>
<tr>
<th>Method</th>
<th>Max. Order</th>
<th>Number of steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed $\omega$</td>
<td>$\omega - 1$</td>
<td>$O\left( L^{\frac{\omega+1}{\omega-1}} \frac{1}{\varepsilon^{\frac{1}{\omega-1}}} \right)$</td>
</tr>
<tr>
<td>Euler ($\omega = 2$)</td>
<td>1</td>
<td>$O\left( \frac{L^3}{\varepsilon} \right)$</td>
</tr>
<tr>
<td>Taylor2 ($\omega = 3$)</td>
<td>2</td>
<td>$O\left( \frac{L^2}{\sqrt{\varepsilon}} \right)$</td>
</tr>
<tr>
<td>Taylor4 ($\omega = 5$)</td>
<td>4</td>
<td>$O\left( \frac{L^{3/2}}{4\sqrt{\varepsilon}} \right)$</td>
</tr>
<tr>
<td>Smart ($\omega = 1 + \log \frac{L}{\varepsilon}$)</td>
<td>$\log \frac{L}{\varepsilon}$</td>
<td>$O\left( L^{\sim 1} \right)$</td>
</tr>
<tr>
<td>Taylor$\infty$ ($\omega = \infty$)</td>
<td>$\infty$</td>
<td>$O\left( L \right)$</td>
</tr>
<tr>
<td>Variable</td>
<td>$O\left( \log \frac{L}{\varepsilon} \right)$</td>
<td>$O\left( L \right)$</td>
</tr>
</tbody>
</table>

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

\[ y(0) = 0 \quad y'(t) = p(y(t)) \quad t \in I \]

**Lemma:** \( y^{(k)}(t) = P_k(y(t)) = \text{poly}(y(t)) \)
Taylor method

\[ y(0) = 0 \quad y'(t) = p(y(t)) \quad t \in I \]

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

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

\[ y(t + h) \approx \sum_{j=0}^{K} \frac{h^j}{j!} y^{(j)}(t) \quad \sim \quad \tilde{y}^{i+1} = \sum_{j=0}^{K} \frac{h^j}{j!} P_k(\tilde{y}^i) \]
Taylor method

\[ y(0) = 0 \quad y'(t) = p(y(t)) \quad t \in I \]

**Lemma:** \[ y^{(k)}(t) = P_k(y(t)) = \text{poly}(y(t)) \]

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

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

- **Fixed order** \( K \): theoretically not enough
Taylor method

\[ y(0) = 0 \quad y'(t) = p(y(t)) \quad t \in I \]

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

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

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

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

\[ y(0) = 0 \quad y'(t) = p(y(t)) \quad t \in I \]

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

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

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

- **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

\[ y(0) = 0 \quad y'(t) = p(y(t)) \quad t \in I \]

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

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

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

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

$$\|y^{(k)}(t) - P_k(y(t))\| \leq \frac{\alpha(Mt)^k}{1 - Mt}$$
Choice of the parameters

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

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

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

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

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

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

\[
\begin{cases}
  x(t) = tu(t) \\
  u(t) = e^{-t} - (1 - e^{-t}) \frac{1}{v(t)} \\
  v(t) = v_0
\end{cases}
\]

\[
\begin{cases}
  x(t) \sim t^{1/v_0} \\
  u(t) \rightarrow \frac{1}{v_0} \\
  v(t) = v_0
\end{cases}
\]

Remark

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

Value are time \( t = 2 \) can be arbitrary large for arbitrary small \( v_0 \)

Theorem

There is no universal bound in \( p, y_0, t_0, t \) and \( \mu \).
But wait...

This is impossible, right ?!

Example

\[
\begin{align*}
    x(t) &= t^u(t) \\
    u(t) &= e^{-t} - (1 - e^{-t}) \frac{1}{v(t)} \\
    v(t) &= v_0
\end{align*}
\]

Remark

- All parameters are fixed except \( y_0 = (1, 1, v_0) \)
But wait...

This is impossible, right ?!

**Example**

\[
\begin{align*}
    x(t) &= t^u(t) \\
    u(t) &= e^{-t} - (1 - e^{-t}) \frac{1}{v(t)} \\
    v(t) &= v_0
\end{align*}
\]

\[
\begin{align*}
    x(t) &\sim t^{\frac{1}{v_0}} \\
    u(t) &\rightarrow \frac{1}{v_0} \\
    v(t) &= v_0
\end{align*}
\]

**Remark**

- All parameters are fixed except \( y_0 = (1, 1, v_0) \)
- Value are time \( t = 2 \) can be arbitrary large for arbitrary small \( v_0 \)
This is impossible, right ?!

**Example**

\[
\begin{align*}
    x(t) &= tu(t) \\
    u(t) &= e^{-t} - (1 - e^{-t}) \frac{1}{v(t)} \\
    v(t) &= v_0
\end{align*}
\]

\[
\begin{align*}
    x(t) &\sim t \frac{1}{v_0} \\
    u(t) &\rightarrow \frac{1}{v_0} \\
    v(t) &= v_0
\end{align*}
\]

**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 \( \mu \).