Continuous models of computation: computability, complexity, universality

Amaury Pouly

Joint work with Olivier Bournez and Daniel Graça

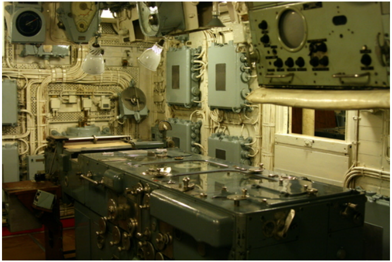
21 september 2018

What is a computer ?

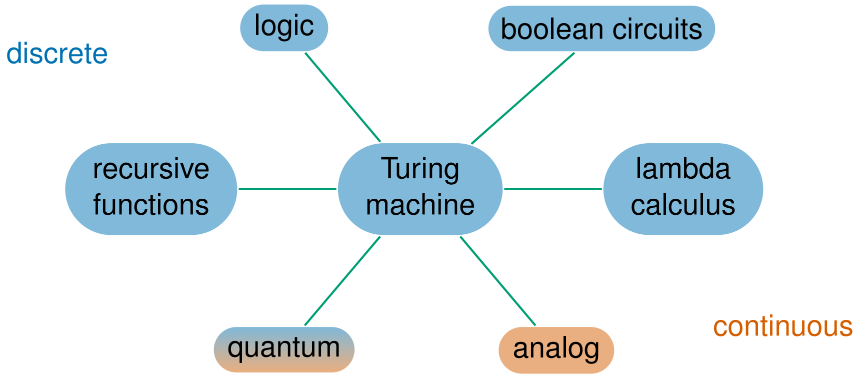
What is a computer ?

What is a computer ?

VS



Computability

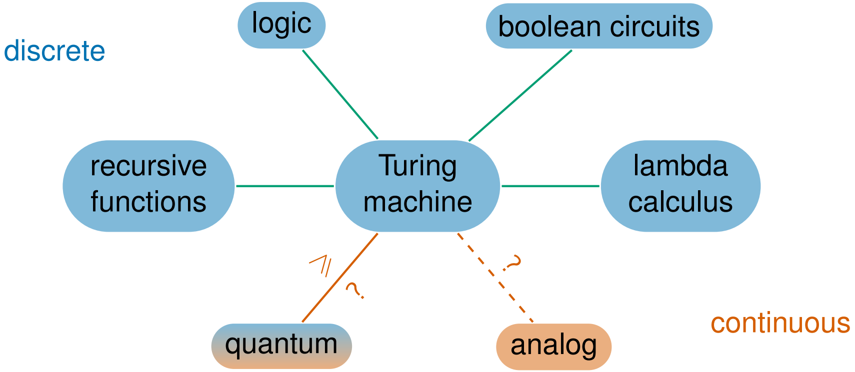


Church Thesis

All **reasonable** models of computation are equivalent.

Church Thesis

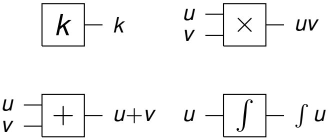
Complexity



Effective Church Thesis

All **reasonable** models of computation are equivalent for complexity.

Polynomial Differential Equations



General Purpose
Analog Computer

Differential Analyzer

Newton mechanics

Reaction networks :

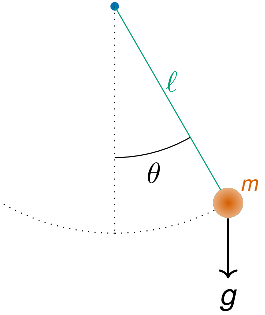
- chemical
- enzymatic

polynomial differential equations :

$$\begin{cases} y(0) = y_0 \\ y'(t) = p(y(t)) \end{cases}$$

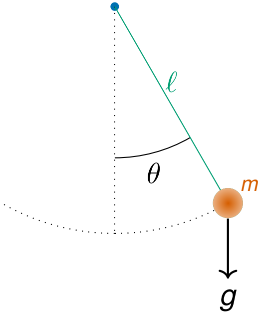
- Rich class
- Stable ($+, \times, \circ, /$, ED)
- No closed-form solution

Example of dynamical system



$$\ddot{\theta} + \frac{g}{\ell}\sin(\theta) = 0$$

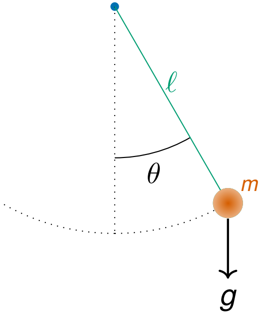
Example of dynamical system



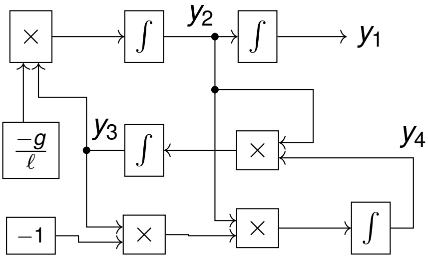
$$\ddot{\theta} + \frac{g}{\ell}\sin(\theta) = 0$$

$$\begin{cases} y_1' = y_2 \\ y_2' = -\frac{g}{l} y_3 \\ y_3' = y_2 y_4 \\ y_4' = -y_2 y_3 \end{cases} \Leftrightarrow \begin{cases} y_1 = \theta \\ y_2 = \dot{\theta} \\ y_3 = \sin(\theta) \\ y_4 = \cos(\theta) \end{cases}$$

Example of dynamical system

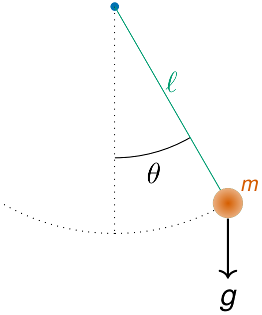


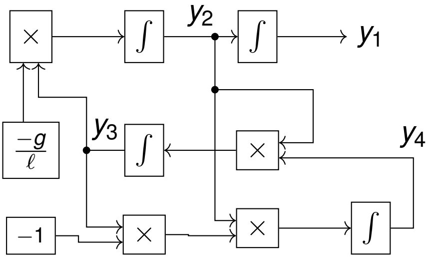
$$\ddot{\theta} + \frac{g}{\ell}\sin(\theta) = 0$$



$$\begin{cases} y_1' = y_2 \\ y_2' = -\frac{g}{l} y_3 \\ y_3' = y_2 y_4 \\ y_4' = -y_2 y_3 \end{cases} \Leftrightarrow \begin{cases} y_1 = \theta \\ y_2 = \dot{\theta} \\ y_3 = \sin(\theta) \\ y_4 = \cos(\theta) \end{cases}$$

Example of dynamical system





$$\ddot{\theta} + \frac{g}{\ell}\sin(\theta) = 0$$

$$\begin{cases} y_1' = y_2 \\ y_2' = -\frac{g}{l} y_3 \\ y_3' = y_2 y_4 \\ y_4' = -y_2 y_3 \end{cases} \Leftrightarrow \begin{cases} y_1 = \theta \\ y_2 = \dot{\theta} \\ y_3 = \sin(\theta) \\ y_4 = \cos(\theta) \end{cases}$$

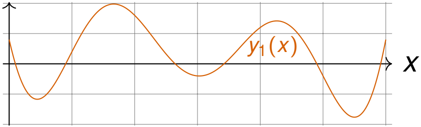
Historical remark : the word “analog”

The pendulum and the circuit have the same equation. One can study one using the other by analogy.

Generable functions

$$\begin{cases} y(0) = y_0 \\ y'(x) = p(y(x)) \end{cases} \qquad x \in \mathbb{R}$$

$$f(x) = y_1(x)$$



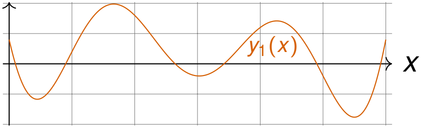
Shannon's notion

Computing with differential equations

Generable functions

$$\begin{cases} y(0) = y_0 \\ y'(x) = p(y(x)) \end{cases} \qquad x \in \mathbb{R}$$

$f(x) = y_1(x)$



Shannon's notion

sin, cos, exp, log, ...

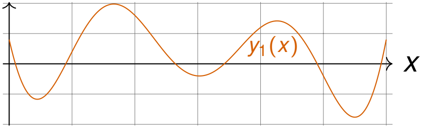
Strictly weaker than Turing machines [Shannon, 1941]

Computing with differential equations

Generable functions

$$\begin{cases} y(0) = y_0 \\ y'(x) = p(y(x)) \end{cases} \qquad x \in \mathbb{R}$$

$$f(x) = y_1(x)$$



Shannon's notion

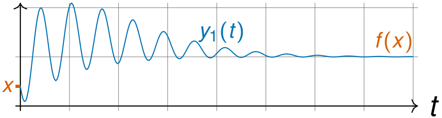
sin, cos, exp, log, ...

Strictly weaker than Turing machines [Shannon, 1941]

Computable

$$\begin{cases} y(0) = q(x) & x \in \mathbb{R} \\ y'(t) = p(y(t)) & t \in \mathbb{R}_+ \end{cases}$$

$$f(x) = \lim_{t \to \infty} y_1(t)$$



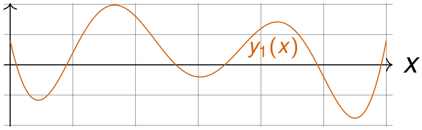
Modern notion

Computing with differential equations

Generable functions

$$\begin{cases} y(0) = y_0 \\ y'(x) = p(y(x)) \end{cases} \qquad x \in \mathbb{R}$$

$$f(x) = y_1(x)$$



Shannon's notion

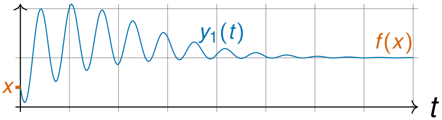
sin, cos, exp, log, ...

Strictly weaker than Turing machines [Shannon, 1941]

Computable

$$\begin{cases} y(0) = q(x) & x \in \mathbb{R} \\ y'(t) = p(y(t)) & t \in \mathbb{R}_+ \end{cases}$$

$$f(x) = \lim_{t \to \infty} y_1(t)$$



Modern notion

sin, cos, exp, log, Γ, ζ, ...

Turing powerful [Bournez et al., 2007]

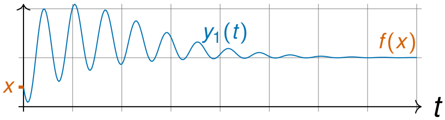
Equivalence with computable analysis

Definition (Bournez et al, 2007)

f **computable by GPAC** if $\exists p$ polynomial such that $\forall x \in [a, b]$

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

satisfies $|f(x) - y_1(t)| \leqslant y_2(t)$ et $y_2(t) \xrightarrow[t\to\infty]{} 0$.



$y_1(t) \xrightarrow[t\to\infty]{} f(x)$

$y_2(t) =$ error bound

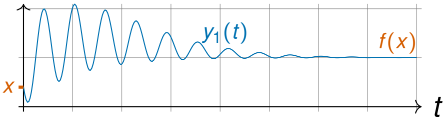
Equivalence with computable analysis

Definition (Bournez et al, 2007)

f **computable by GPAC** if $\exists p$ polynomial such that $\forall x \in [a,b]$

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

satisfies $|f(x) - y_1(t)| \leqslant y_2(t)$ et $y_2(t) \xrightarrow[t\to\infty]{} 0$.



$y_1(t) \xrightarrow[t\to\infty]{} f(x)$

$y_2(t) =$ error bound

Theorem (Bournez et al, 2007)

$f : [a,b] \to \mathbb{R}$ *computable*[1] $\Leftrightarrow$ *f computable by GPAC*

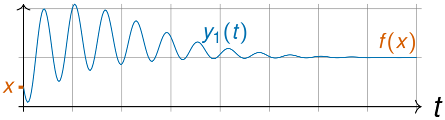
Equivalence with computable analysis

Definition (Bournez et al, 2007)

f **computable by GPAC** if $\exists p$ polynomial such that $\forall x \in [a,b]$

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

satisfies $|f(x) - y_1(t)| \leqslant y_2(t)$ et $y_2(t) \xrightarrow[t\to\infty]{} 0$.



$y_1(t) \xrightarrow[t\to\infty]{} f(x)$

$y_2(t) =$ error bound

Theorem (Bournez et al, 2007)

$f : [a,b] \to \mathbb{R}$ *computable*[1] $\Leftrightarrow$ *f computable by GPAC*

1. In Computable Analysis, a standard model over reals built from Turing machines.

Complexity of analog systems

- Turing machines : $T(x) =$ number of steps to compute on x

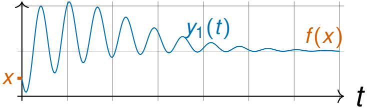
Complexity of analog systems

- Turing machines : $T(x)$ = number of steps to compute on x
- GPAC :

Tentative definition

$T(x) = ??$

$y(0) = (x, 0, \ldots, 0) \qquad y' = p(y)$



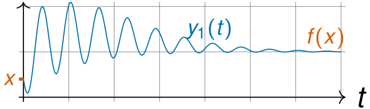
Complexity of analog systems

- Turing machines : $T(x) =$ number of steps to compute on x
- GPAC :

Tentative definition

$T(x, \mu) =$

$y(0) = (x, 0, \ldots, 0) \qquad y' = p(y)$



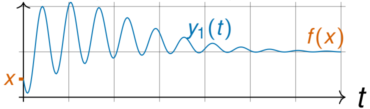
Complexity of analog systems

- Turing machines : $T(x) =$ number of steps to compute on x
- GPAC :

Tentative definition

$T(x, \mu) =$ first time t so that $|y_1(t) - f(x)| \leqslant e^{-\mu}$

$y(0) = (x, 0, \dots, 0) \qquad y' = p(y)$



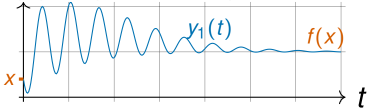
Complexity of analog systems

- Turing machines : $T(x) =$ number of steps to compute on x
- GPAC :

Tentative definition

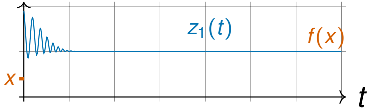
$T(x,\mu) =$ first time t so that $|y_1(t) - f(x)| \leqslant e^{-\mu}$

$y(0) = (x, 0, \ldots, 0) \qquad y' = p(y)$



$\rightsquigarrow$

$z(t) = y(e^t)$



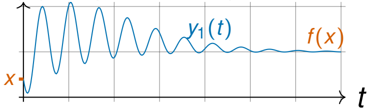
Complexity of analog systems

- Turing machines : $T(x)$ = number of steps to compute on x
- GPAC :

Tentative definition

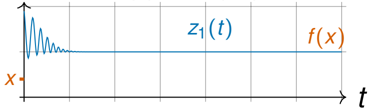
$T(x,\mu)$ = first time t so that $|y_1(t) - f(x)| \leqslant e^{-\mu}$

$y(0) = (x, 0, \ldots, 0) \qquad y' = p(y)$

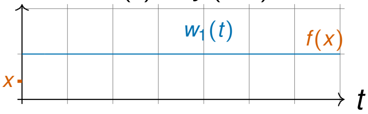


$\rightsquigarrow$

$z(t) = y(e^t)$



$w(t) = y(e^{e^t})$



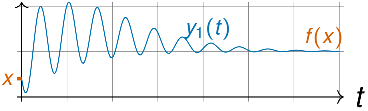
Complexity of analog systems

- Turing machines : $T(x)$ = number of steps to compute on x
- GPAC : time contraction problem $\rightarrow$ **open problem**

Tentative definition

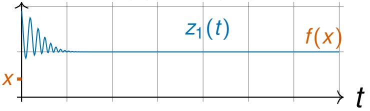
$T(x, \mu)$ = first time t so that $|y_1(t) - f(x)| \leqslant e^{-\mu}$

$y(0) = (x, 0, \ldots, 0) \qquad y' = p(y)$

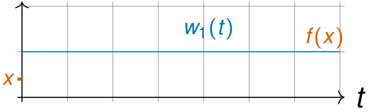


$\rightsquigarrow$

$z(t) = y(e^t)$



$w(t) = y(e^{e^t})$

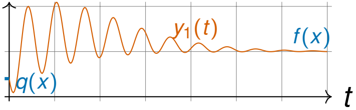


Something is wrong...

All functions have constant time complexity.

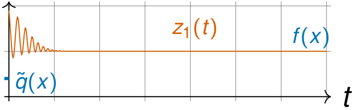
Time-space correlation of the GPAC

$y(0) = q(x) \qquad y' = p(y)$

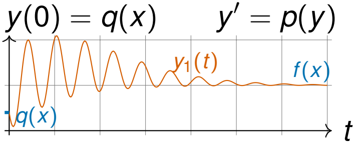


$\rightsquigarrow$

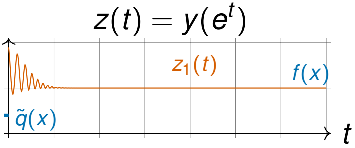
$z(t) = y(e^t)$



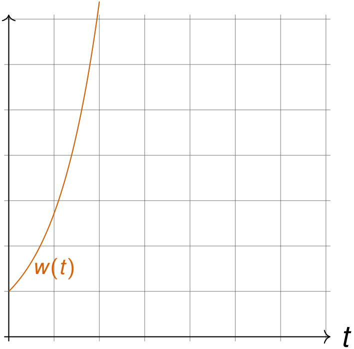
Time-space correlation of the GPAC



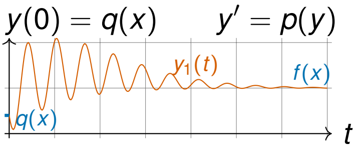
$\rightsquigarrow$



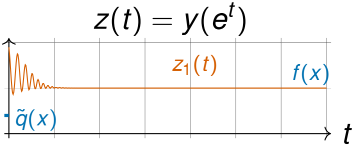
extra component : $w(t) = e^t$



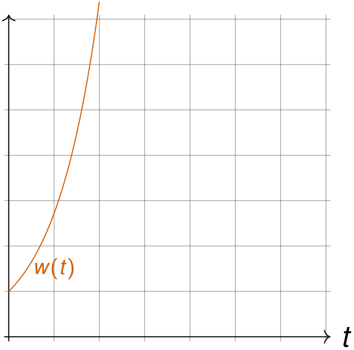
Time-space correlation of the GPAC



$\rightsquigarrow$



extra component : $w(t) = e^t$



Observation

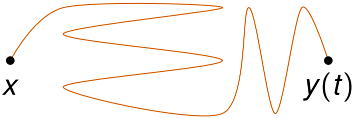
Time scaling costs "space".

$\rightsquigarrow$

Time complexity for the GPAC must involve time and space !

Complexity of solving polynomial ODEs

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



Complexity of solving polynomial ODEs

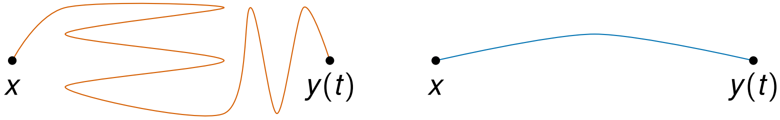
$$y(0) = x \qquad y'(t) = p(y(t))$$

Theorem

If $y(t)$ exists, one can compute p, q such that $\left|\frac{p}{q} - y(t)\right| \leqslant 2^{-n}$ in time

$$\operatorname{poly}(\textit{size of } x \textit{ and } p, n, \ell(t))$$

where $\ell(t) \approx$ length of the curve (between x and $y(t)$)

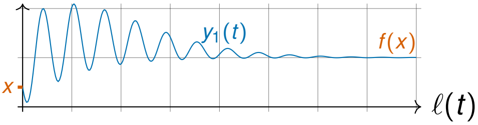


length of the curve = complexity = ressource

Characterization of real polynomial time

Definition : $f : [a, b] \to \mathbb{R}$ in ANALOG-P$_{\mathbb{R}}$ $\Leftrightarrow \exists p$ polynomial, $\forall x \in [a, b]$

$$y(0) = (x, 0, \dots, 0) \qquad y' = p(y)$$



Characterization of real polynomial time

Definition : $f : [a,b] \to \mathbb{R}$ in ANALOG-P$_{\mathbb{R}}$ $\Leftrightarrow \exists p$ polynomial, $\forall x \in [a,b]$

$$y(0) = (x, 0, \ldots, 0) \qquad y' = p(y)$$

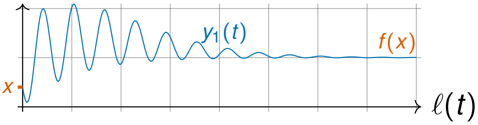
satisfies :

1. $|y_1(t) - f(x)| \leqslant 2^{-\ell(t)}$

   «greater length $\Rightarrow$ greater precision»

2. $\ell(t) \geqslant t$

   «length increases with time»



Characterization of real polynomial time

Definition : $f : [a,b] \to \mathbb{R}$ in ANALOG-P$_{\mathbb{R}}$ $\Leftrightarrow$ $\exists p$ polynomial, $\forall x \in [a,b]$

$$y(0) = (x, 0, \ldots, 0) \qquad y' = p(y)$$

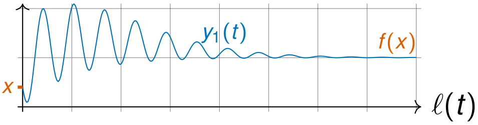
satisfies :

1. $|y_1(t) - f(x)| \leqslant 2^{-\ell(t)}$

   «greater length $\Rightarrow$ greater precision»

2. $\ell(t) \geqslant t$

   «length increases with time»



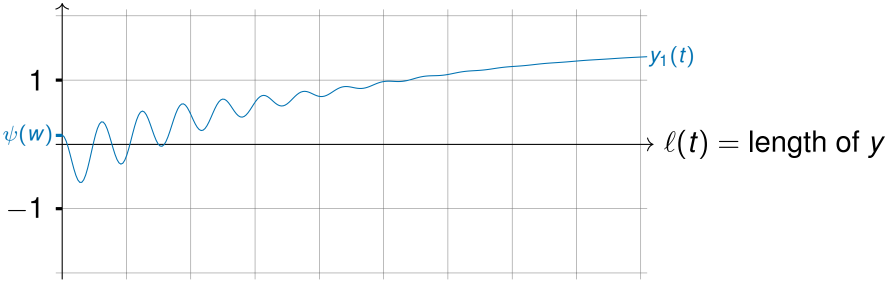
Theorem

$f : [a,b] \to \mathbb{R}$ computable in polynomial time $\Leftrightarrow$ $f \in$ ANALOG-P$_{\mathbb{R}}$.

Characterization of polynomial time

Definition : $\mathcal{L} \in$ ANALOG-PTIME $\Leftrightarrow \exists p$ polynomial, $\forall$ word w

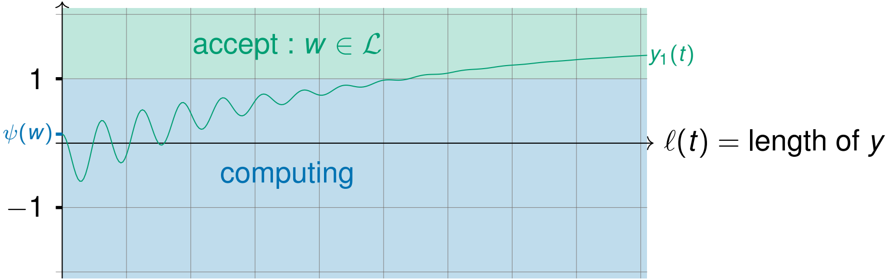
$$y(0) = (\psi(w), |w|, 0, \ldots, 0) \qquad y' = p(y) \qquad \psi(w) = \sum_{i=1}^{|w|} w_i 2^{-i}$$



Characterization of polynomial time

Definition : $\mathcal{L} \in$ ANALOG-PTIME $\Leftrightarrow \exists p$ polynomial, $\forall$ word w

$$y(0) = (\psi(w), |w|, 0, \ldots, 0) \qquad y' = p(y) \qquad \psi(w) = \sum_{i=1}^{|w|} w_i 2^{-i}$$



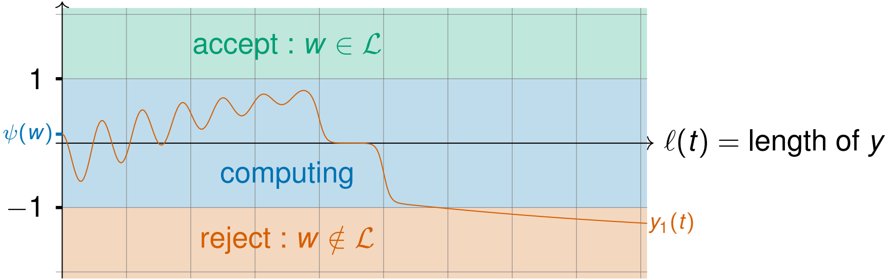
satisfies

1. if $y_1(t) \geqslant 1$ then $w \in \mathcal{L}$

Characterization of polynomial time

Definition : $\mathcal{L} \in$ ANALOG-PTIME $\Leftrightarrow \exists p$ polynomial, $\forall$ word w

$$y(0) = (\psi(w), |w|, 0, \ldots, 0) \qquad y' = p(y) \qquad \psi(w) = \sum_{i=1}^{|w|} w_i 2^{-i}$$



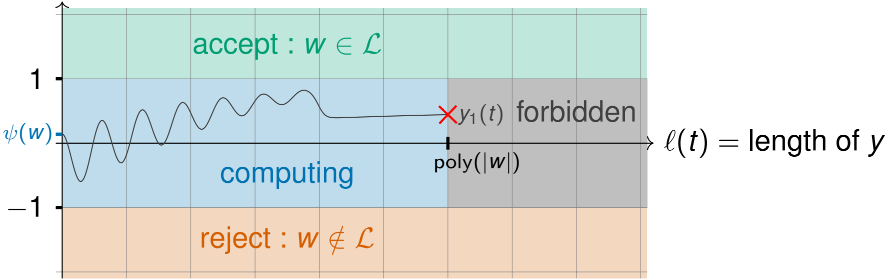
satisfies

2. if $y_1(t) \leqslant -1$ then $w \notin \mathcal{L}$

Characterization of polynomial time

Definition : $\mathcal{L} \in$ ANALOG-PTIME $\Leftrightarrow \exists p$ polynomial, $\forall$ word w

$$y(0) = (\psi(w), |w|, 0, \ldots, 0) \qquad y' = p(y) \qquad \psi(w) = \sum_{i=1}^{|w|} w_i 2^{-i}$$



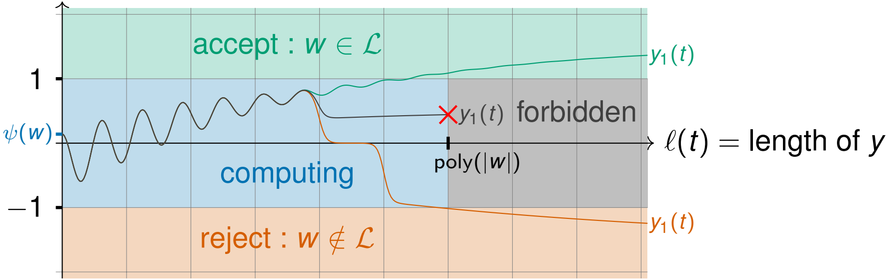
satisfies

3. if $\ell(t) \geqslant \text{poly}(|w|)$ then $|y_1(t)| \geqslant 1$

Characterization of polynomial time

Definition : $\mathcal{L} \in$ ANALOG-PTIME $\Leftrightarrow \exists p$ polynomial, $\forall$ word w

$$y(0) = (\psi(w), |w|, 0, \ldots, 0) \qquad y' = p(y) \qquad \psi(w) = \sum_{i=1}^{|w|} w_i 2^{-i}$$

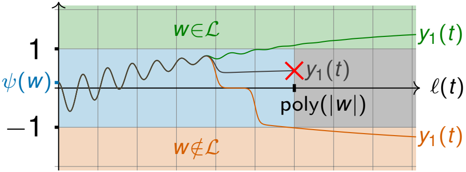


Theorem

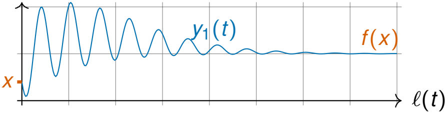
PTIME = ANALOG-PTIME

Summary

ANALOG-PTIME



ANALOG-$\mathrm{P}_{\mathbb{R}}$

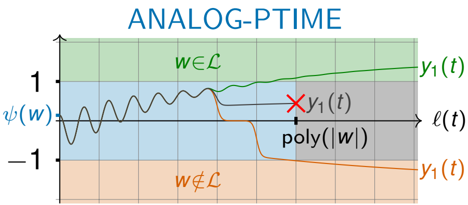


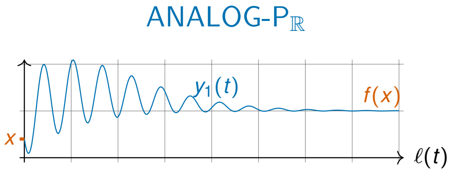
Theorem

- $\mathcal{L} \in$ PTIME *of and only if* $\mathcal{L} \in$ ANALOG-PTIME
- $f : [a, b] \to \mathbb{R}$ *computable in polynomial time* $\Leftrightarrow f \in$ ANALOG-$\mathrm{P}_{\mathbb{R}}$

- Analog complexity theory based on length
- Time of Turing machine $\Leftrightarrow$ length of the GPAC
- Purely continuous characterization of PTIME

Summary





Theorem

- $\mathcal{L} \in$ PTIME *of and only if* $\mathcal{L} \in$ ANALOG-PTIME
- $f : [a, b] \to \mathbb{R}$ *computable in polynomial time* $\Leftrightarrow f \in$ ANALOG-$P_{\mathbb{R}}$

- Analog complexity theory based on length
- Time of Turing machine $\Leftrightarrow$ length of the GPAC
- Purely continuous characterization of PTIME
- Only rational coefficients needed

In the remaining time...

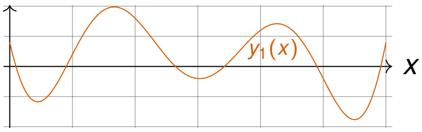
Two applications of the "technology" we have developed :

Universal differential equation

Chemical Reaction Networks

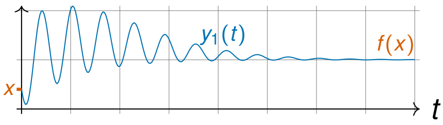
Universal differential equations

Generable functions



subclass of analytic functions

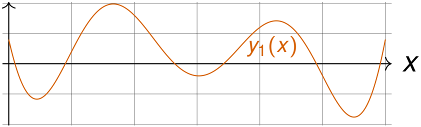
Computable functions



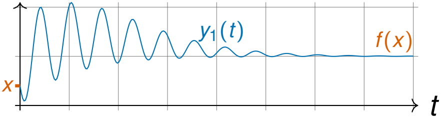
any computable function

Universal differential equations

Generable functions

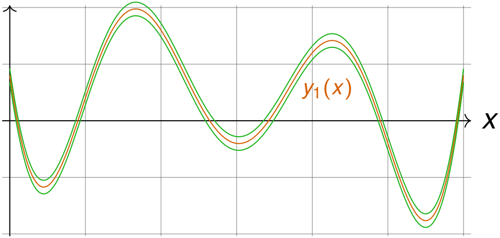


Computable functions

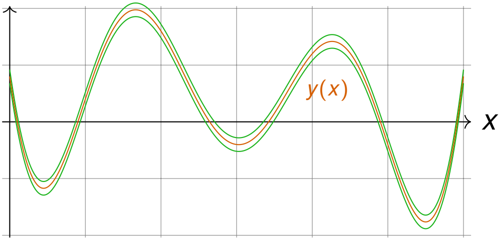


subclass of analytic functions

any computable function



Universal differential algebraic equation (DAE)



Theorem (Rubel, 1981)

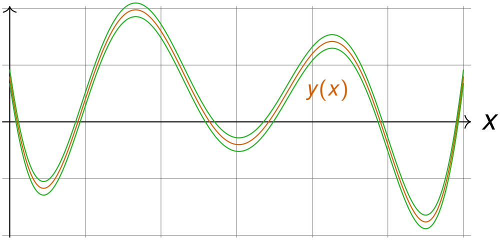
For any continuous functions f and ε, there exists $y : \mathbb{R} \to \mathbb{R}$ solution to

$$\begin{aligned} 3y'^4 y'' y''''^2 \quad & -4y'^4 y'''^2 y'''' + 6y'^3 y''^2 y''' y'''' + 24y'^2 y''^4 y'''' \\ & -12y'^3 y'' y'''^3 - 29y'^2 y''^3 y'''^2 + 12y''^7 \qquad = 0 \end{aligned}$$

such that $\forall t \in \mathbb{R}$,

$$|y(t) - f(t)| \leqslant \varepsilon(t).$$

Universal differential algebraic equation (DAE)



Theorem (Rubel, 1981)

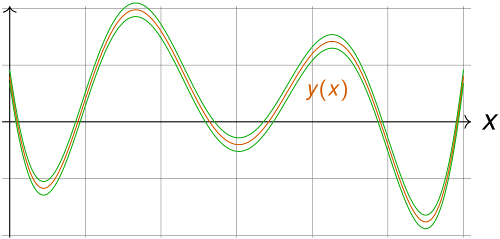
*There exists a **fixed** polynomial p and $k \in \mathbb{N}$ such that for any continuous functions f and ε, there exists a solution $y : \mathbb{R} \to \mathbb{R}$ to*

$$p(y, y', \ldots, y^{(k)}) = 0$$

such that $\forall t \in \mathbb{R}$,

$$|y(t) - f(t)| \leqslant \varepsilon(t).$$

Universal differential algebraic equation (DAE)



Theorem (Rubel, 1981)

There exists a ***fixed*** *polynomial p and $k \in \mathbb{N}$ such that for any continuous functions f and ε, there exists a solution $y : \mathbb{R} \to \mathbb{R}$ to*

$$p(y, y', \ldots, y^{(k)}) = 0$$

such that $\forall t \in \mathbb{R}$,

$$|y(t) - f(t)| \leqslant \varepsilon(t).$$

Problem : this is «weak» result.

The problem with Rubel's DAE

The solution y is not unique, **even with added initial conditions** :

$$p(y, y', \ldots, y^{(k)}) = 0, \quad y(0) = \alpha_0, y'(0) = \alpha_1, \ldots, y^{(k)}(0) = \alpha_k$$

In fact, this is fundamental for Rubel's proof to work !

The problem with Rubel's DAE

The solution y is not unique, **even with added initial conditions** :

$$p(y, y', \ldots, y^{(k)}) = 0, \quad y(0) = \alpha_0, y'(0) = \alpha_1, \ldots, y^{(k)}(0) = \alpha_k$$

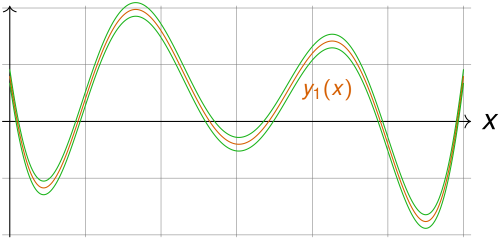
In fact, this is fundamental for Rubel's proof to work !

- Rubel's statement : this DAE is universal
- More realistic interpretation : this DAE allows almost anything

Open Problem (Rubel, 1981)

Is there a universal ODE $y' = p(y)$?
Note : explicit polynomial ODE $\Rightarrow$ unique solution

Universal initial value problem (IVP)



Theorem

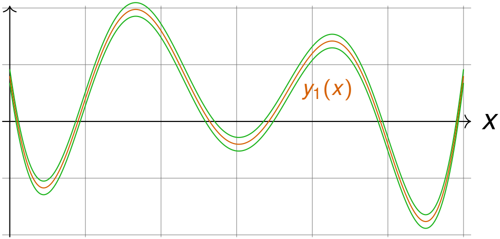
*There exists a **fixed** (vector of) polynomial p such that for any continuous functions f and ε, there exists $\alpha \in \mathbb{R}^d$ such that*

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

*has a **unique solution** y* $: \mathbb{R} \to \mathbb{R}^d$ *and* $\forall t \in \mathbb{R}$,

$$|y_1(t) - f(t)| \leqslant \varepsilon(t).$$

Universal initial value problem (IVP)



Notes :

- **system** of ODEs,
- y is analytic,
- we need $d \approx 300$.

Theorem

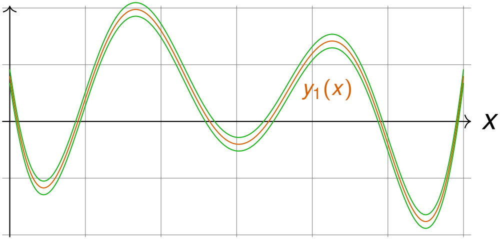
*There exists a **fixed** (vector of) polynomial p such that for any continuous functions f and ε, there exists $\alpha \in \mathbb{R}^d$ such that*

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

*has a **unique solution** $y : \mathbb{R} \to \mathbb{R}^d$ and $\forall t \in \mathbb{R}$,*

$$|y_1(t) - f(t)| \leqslant \varepsilon(t).$$

Universal initial value problem (IVP)



Notes :

- **system** of ODEs,
- y is analytic,
- we need $d \approx 300$.

Theorem

*There exists a **fixed** (vector of) polynomial p such that for any continuous functions f and ε, there exists $\alpha \in \mathbb{R}^d$ such that*

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

*has a **unique solution** $y : \mathbb{R} \to \mathbb{R}^d$ and $\forall t \in \mathbb{R}$,*

$$|y_1(t) - f(t)| \leqslant \varepsilon(t).$$

Remark : α is usually transcendental, but computable from f and ε

Chemical Reaction Networks

Definition : a **reaction system** is a finite set of

- molecular species $y_1, \ldots, y_n$
- reactions of the form $\sum_i a_i y_i \xrightarrow{f} \sum_i b_i y_i \qquad (a_i, b_i \in \mathbb{N}, f = \text{rate})$

Example :

$$\begin{array}{cccccc} 2H_2 & + & O & \rightarrow & 2H_2O \\ C & + & O_2 & \rightarrow & CO_2 \end{array}$$

Chemical Reaction Networks

Definition : a **reaction system** is a finite set of

- molecular species $y_1, \ldots, y_n$
- reactions of the form $\sum_i a_i y_i \xrightarrow{f} \sum_i b_i y_i \qquad (a_i, b_i \in \mathbb{N},\ f = \text{rate})$

Example :

$$\begin{array}{cccc} 2H_2 & + & O & \rightarrow & 2H_2O \\ C & + & O_2 & \rightarrow & CO_2 \end{array}$$

Assumption : law of mass action

$$\sum_i a_i y_i \xrightarrow{k} \sum_i b_i y_i \quad \rightsquigarrow \quad f(y) = k \prod_i y_i^{a_i}$$

Chemical Reaction Networks

Definition : a **reaction system** is a finite set of

- molecular species $y_1, \ldots, y_n$
- reactions of the form $\sum_i a_i y_i \xrightarrow{f} \sum_i b_i y_i \qquad (a_i, b_i \in \mathbb{N},\ f = \text{rate})$

Example :

$$\begin{array}{ccccc} 2H_2 & + & O & \rightarrow & 2H_2O \\ C & + & O_2 & \rightarrow & CO_2 \end{array}$$

Assumption : law of mass action

$$\sum_i a_i y_i \xrightarrow{k} \sum_i b_i y_i \;\leadsto\; f(y) = k \prod_i y_i^{a_i}$$

Semantics :

- discrete
- differential
- stochastic

Chemical Reaction Networks

Definition : a **reaction system** is a finite set of

- molecular species $y_1, \dots, y_n$
- reactions of the form $\sum_i a_i y_i \xrightarrow{f} \sum_i b_i y_i \qquad (a_i, b_i \in \mathbb{N},\ f = \text{rate})$

Example :

$$\begin{array}{cccccc} 2H_2 & + & O & \rightarrow & 2H_2O \\ C & + & O_2 & \rightarrow & CO_2 \end{array}$$

Assumption : law of mass action

$$\sum_i a_i y_i \xrightarrow{k} \sum_i b_i y_i \ \rightsquigarrow \ f(y) = k \prod_i y_i^{a_i}$$

Semantics :

- discrete
- differential $\rightarrow$
- stochastic

$$y_i' = \sum_{\text{reaction } R} (b_i^R - a_i^R) f^R(y)$$

Chemical Reaction Networks

Definition : a **reaction system** is a finite set of

- molecular species $y_1, \ldots, y_n$
- reactions of the form $\sum_i a_i y_i \xrightarrow{f} \sum_i b_i y_i \qquad (a_i, b_i \in \mathbb{N},\ f = \text{rate})$

Example :

$$\begin{array}{rcccl} 2H_2 & + & O & \rightarrow & 2H_2O \\ C & + & O_2 & \rightarrow & CO_2 \end{array}$$

Assumption : law of mass action

$$\sum_i a_i y_i \xrightarrow{k} \sum_i b_i y_i \quad \rightsquigarrow \quad f(y) = k \prod_i y_i^{a_i}$$

Semantics :

- discrete
- differential $\rightarrow$
- stochastic

$$y_i' = \sum_{\text{reaction } R} (b_i^R - a_i^R) k^R \prod_j y_j^{a_j}$$

Chemical Reaction Networks (CRNs)

- CRNs with differential semantics and mass action law = polynomial ODEs
- polynomial ODEs are Turing complete

Chemical Reaction Networks (CRNs)

- CRNs with differential semantics and mass action law = polynomial ODEs
- polynomial ODEs are Turing complete

CRNs are Turing complete ?

Chemical Reaction Networks (CRNs)

CRNs are Turing complete ? Two “slight” problems :

- concentrations cannot be negative ($y_i < 0$)
- arbitrary reactions are not realistic

Chemical Reaction Networks (CRNs)

CRNs are Turing complete ? Two "slight" problems :

- concentrations cannot be negative ($y_i < 0$) ► easy to solve
- arbitrary reactions are not realistic ► what is realistic ?

Chemical Reaction Networks (CRNs)

CRNs are Turing complete ? Two “slight” problems :

- concentrations cannot be negative ($y_i < 0$) ► easy to solve
- arbitrary reactions are not realistic ► what is realistic ?

Definition : a reaction is **elementary** if it has at most two reactants
$\Rightarrow$ can be implemented with DNA, RNA or proteins

Chemical Reaction Networks (CRNs)

CRNs are Turing complete ? Two "slight" problems :

- concentrations cannot be negative ($y_i < 0$) ▶ easy to solve
- arbitrary reactions are not realistic ▶ what is realistic ?

Definition : a reaction is **elementary** if it has at most two reactants
$\Rightarrow$ can be implemented with DNA, RNA or proteins

Elementary reactions correspond to **quadratic** ODEs :

$$ay + bz \xrightarrow{k} \cdots \qquad \rightsquigarrow \qquad f(y, z) = ky^a z^b$$

Chemical Reaction Networks (CRNs)

CRNs are Turing complete? Two "slight" problems:

- concentrations cannot be negative ($y_i < 0$) ▶ easy to solve
- arbitrary reactions are not realistic ▶ what is realistic?

Definition: a reaction is **elementary** if it has at most two reactants
$\Rightarrow$ can be implemented with DNA, RNA or proteins

Elementary reactions correspond to **quadratic** ODEs:

$$ay + bz \xrightarrow{k} \cdots \qquad \rightsquigarrow \qquad f(y, z) = ky^a z^b$$

Theorem (Folklore)

Every polynomial ODE can be rewritten as a quadratic ODE.

Chemical Reaction Networks (CRNs)

Definition : a reaction is **elementary** if it has at most two reactants
$\Rightarrow$ can be implemented with DNA, RNA or proteins

Elementary reactions correspond to **quadratic** ODEs :

$$ay + bz \xrightarrow{k} \cdots \qquad \rightsquigarrow \qquad f(y,z) = ky^a z^b$$

Theorem (CMSB, joint work with François Fages, Guillaume Le Guludec)

Elementary mass-action-law reaction system on finite universes of molecules are Turing-complete under the differential semantics.

Notes :

- proof preserves polynomial length
- in fact the following elementary reactions suffice :

$$\varnothing \xrightarrow{k} x \qquad x \xrightarrow{k} x + z \qquad x + y \xrightarrow{k} x + y + z \qquad x + y \xrightarrow{k} \varnothing$$

Future work

Reaction networks :

- chemical
- enzymatic

$$y' = p(y)$$

?

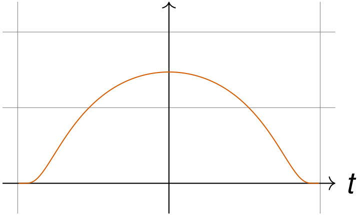
$$y' = p(y) + e(t)$$

- Finer time complexity (linear)
- Nondeterminism
- Robustness
- « Space» complexity
- Other models
- Stochastic

Rubel's proof in one slide

- Take $f(t) = e^{\frac{-1}{1-t^2}}$ for $-1 < t < 1$ and $f(t) = 0$ otherwise.

$$\text{It satisfies} \quad (1-t^2)^2 f''(t) + 2tf'(t) = 0.$$



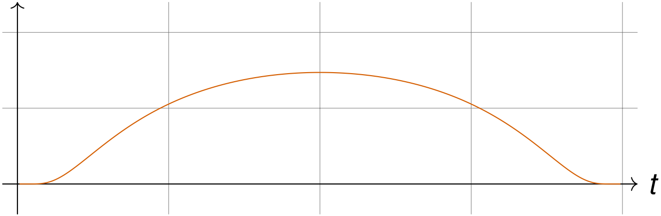
Rubel's proof in one slide

- Take $f(t) = e^{\frac{-1}{1-t^2}}$ for $-1 < t < 1$ and $f(t) = 0$ otherwise.

  It satisfies $\quad (1-t^2)^2 f''(t) + 2tf'(t) = 0.$
- For any $a, b, c \in \mathbb{R}$, $y(t) = cf(at+b)$ satisfies

$$3y'^4 y'' y''''^2 \quad -4y'^4 y''^2 y'''' + 6y'^3 y''^2 y''' y'''' + 24y'^2 y''^4 y'''' \\ -12y'^3 y'' y'''^3 - 29y'^2 y''^3 y'''^2 + 12y''^7 = 0$$

Translation and rescaling :



Rubel's proof in one slide

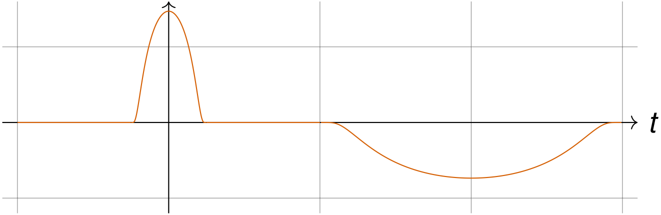
- Take $f(t) = e^{\frac{-1}{1-t^2}}$ for $-1 < t < 1$ and $f(t) = 0$ otherwise.

  $$\text{It satisfies} \quad (1-t^2)^2 f''(t) + 2tf'(t) = 0.$$

- For any $a, b, c \in \mathbb{R}$, $y(t) = cf(at+b)$ satisfies

  $$3y'^4y''y''''^2 - 4y'^4y''^2y'''' + 6y'^3y''^2y'''y'''' + 24y'^2y''^4y'''' - 12y'^3y''y'''^3 - 29y'^2y''^3y'''^2 + 12y''^7 = 0$$

- Can glue together arbitrary many such pieces

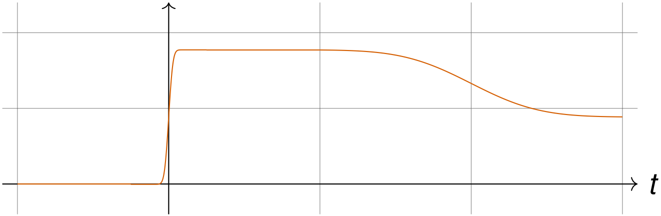


Rubel's proof in one slide

- Take $f(t) = e^{\frac{-1}{1-t^2}}$ for $-1 < t < 1$ and $f(t) = 0$ otherwise.

  $$\text{It satisfies} \quad (1-t^2)^2 f''(t) + 2tf'(t) = 0.$$
- For any $a, b, c \in \mathbb{R}$, $y(t) = cf(at+b)$ satisfies

  $$3y'^4y''y''''^2-4y'^4y''^2y''''+6y'^3y''^2y'''y''''+24y'^2y''^4y''''-12y'^3y''y'''^3-29y'^2y''^3y'''^2+12y''^7=0$$
- Can glue together arbitrary many such pieces
- Can arrange so that $\int f$ is solution : piecewise pseudo-linear



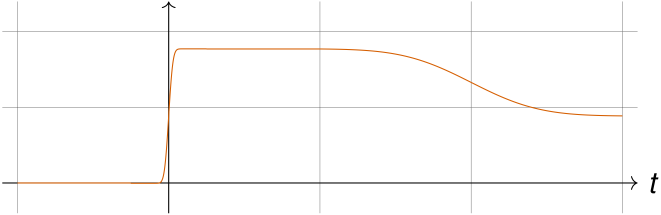
Rubel's proof in one slide

- Take $f(t) = e^{\frac{-1}{1-t^2}}$ for $-1 < t < 1$ and $f(t) = 0$ otherwise.

  It satisfies $\quad (1-t^2)^2 f''(t) + 2tf'(t) = 0.$
- For any $a, b, c \in \mathbb{R}$, $y(t) = cf(at+b)$ satisfies

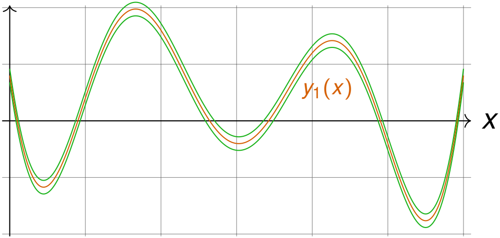
$$3y'^4y''y''''^2 - 4y'^4y''^2y'''' + 6y'^3y''^2y'''y'''' + 24y'^2y''^4y'''' - 12y'^3y''y'''^3 - 29y'^2y''^3y'''^2 + 12y''^7 = 0$$

- Can glue together arbitrary many such pieces
- Can arrange so that $\int f$ is solution : piecewise pseudo-linear



Conclusion : Rubel's equation allows any piecewise pseudo-linear functions, and those are **dense in** C^0

Universal DAE revisited



Theorem

*There exists a **fixed** polynomial p and $k \in \mathbb{N}$ such that for any continuous functions f and ε, there exists $\alpha_0, \ldots, \alpha_k \in \mathbb{R}$ such that*

$$p(y, y', \ldots, y^{(k)}) = 0, \quad y(0) = \alpha_0, y'(0) = \alpha_1, \ldots, y^{(k)}(0) = \alpha_k$$

*has a **unique analytic solution** and this solution satisfies such that*

$$|y(t) - f(t)| \leqslant \varepsilon(t).$$