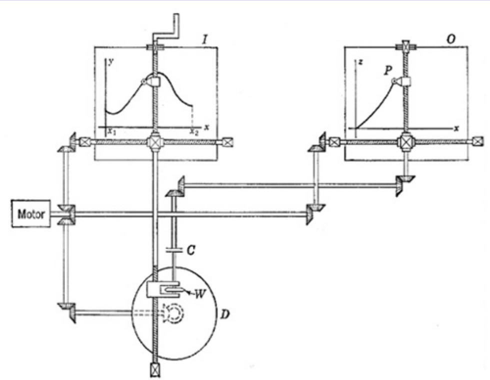
Computing with Ordinary Differential Equations

Olivier Bournez   Daniel Graça[1]   Amaury Pouly[2]

Ecole Polytechnique
Laboratoire d'Informatique de l'X
Palaiseau, France

GDR-IM
March 2017

[1]Université d'Algarve, Portugal
[2]Postdoc, MPI, Allemagne



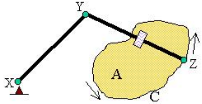
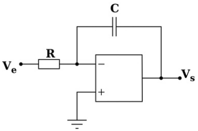
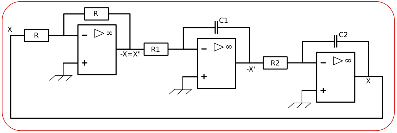
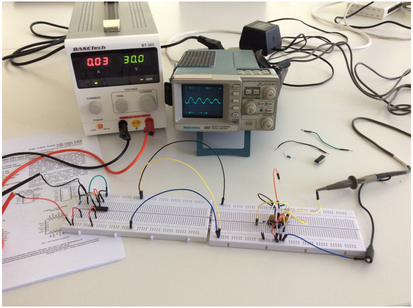


Figure 1. A simple planimeter.







Today's game

We start from

[3] With y_0, and coefficients among $0, 1, -1$.

Today's game

We start from

- 0, 1, -1

[3] With y_0, and coefficients among $0, 1, -1$.

Today's game

We start from

- 0, 1, -1
- and we consider projections of solutions of ordinary differential equations of type

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

where p is a (vector of) polynomials[3]

[3] With y_0, and coefficients among $0, 1, -1$.

Today's game

We start from

- 0, 1, -1
- and we consider projections of solutions of ordinary differential equations of type

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

where p is a (vector of) polynomials[3]

Terminology:

- Such a function $f(t) = y_1(t)$ will be said to be generated.
- $f(1)$ will then be called a (pODE) computable real.

[3] With y_0, and coefficients among $0, 1, -1$.

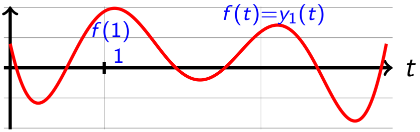
Today's game

We start from

- 0, 1, -1
- and we consider projections of solutions of ordinary differential equations of type

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

where p is a (vector of) polynomials[3]



Terminology:

- Such a function $f(t) = y_1(t)$ will be said to be generated.
- $f(1)$ will then be called a (pODE) computable real.

[3]With y_0, and coefficients among $0, 1, -1$.

Menu

Descriptive Mathematics

Descriptive Computer/Computability Science

Descriptive Computer/Complexity Science

Descriptive Algorithmic Science

In Case of Turing Nostalgy

Conclusions

Polynomial ODE descriptive mathematics

- exp is the solution of $y' = y$, $y(0) = 1$.

Polynomial ODE descriptive mathematics

- exp is the solution of $y' = y$, $y(0) = 1$.
- e is $\exp(1)$.

Polynomial ODE descriptive mathematics

- exp is the solution of $y' = y$, $y(0) = 1$.
- e is $\exp(1)$.
- tanh is the solution of $y' = 1 - y^2$, $y(0) = 1$.

Polynomial ODE descriptive mathematics

- exp is the solution of $y' = y$, $y(0) = 1$.
- e is $\exp(1)$.
- tanh is the solution of $y' = 1 - y^2$, $y(0) = 1$.
- sin is the first projection of $y' = (y_1', y_2') = (y_2, -y_1)$, $y(0) = (0, 1)$.

Polynomial ODE descriptive mathematics

- exp is the solution of $y' = y$, $y(0) = 1$.
- e is $\exp(1)$.
- tanh is the solution of $y' = 1 - y^2$, $y(0) = 1$.
- sin is the first projection of $y' = (y_1', y_2') = (y_2, -y_1)$, $y(0) = (0, 1)$.
- cos its second projection.

Polynomial ODE descriptive mathematics

- exp is the solution of $y' = y$, $y(0) = 1$.
- e is exp(1).
- tanh is the solution of $y' = 1 - y^2$, $y(0) = 1$.
- sin is the first projection of $y' = (y_1', y_2') = (y_2, -y_1)$, $y(0) = (0, 1)$.
- cos its second projection.
- sinh is the first projection of $y' = (y_2, y_1)$, $y(0) = (0, 1)$.

Polynomial ODE descriptive mathematics

- exp is the solution of $y' = y$, $y(0) = 1$.
- e is $\exp(1)$.
- tanh is the solution of $y' = 1 - y^2$, $y(0) = 1$.
- sin is the first projection of $y' = (y_1', y_2') = (y_2, -y_1)$, $y(0) = (0, 1)$.
- cos its second projection.
- sinh is the first projection of $y' = (y_2, y_1)$, $y(0) = (0, 1)$.
- cosh its second projection.

Polynomial ODE descriptive mathematics

- exp is the solution of $y' = y$, $y(0) = 1$.
- e is exp(1).
- tanh is the solution of $y' = 1 - y^2$, $y(0) = 1$.
- sin is the first projection of $y' = (y_1', y_2') = (y_2, -y_1)$, $y(0) = (0, 1)$.
- cos its second projection.
- sinh is the first projection of $y' = (y_2, y_1)$, $y(0) = (0, 1)$.
- cosh its second projection.
- $\frac{1}{1+t}$ is the solution of $y' = -y^2$, $y(0) = 1$

Polynomial ODE descriptive mathematics

- exp is the solution of $y' = y$, $y(0) = 1$.
- e is $\exp(1)$.
- tanh is the solution of $y' = 1 - y^2$, $y(0) = 1$.
- sin is the first projection of $y' = (y_1', y_2') = (y_2, -y_1)$, $y(0) = (0, 1)$.
- cos its second projection.
- sinh is the first projection of $y' = (y_2, y_1)$, $y(0) = (0, 1)$.
- cosh its second projection.
- $\frac{1}{1+t}$ is the solution of $y' = -y^2$, $y(0) = 1$
- $\frac{1}{1+t^2}$ is the first projection of $y' = (-y_2 y_1^2 - y_2 y_3 y_1^2, 1 + y_3, 0)$, $y(0, 0, 0) = (1, 0, 1)$.

Polynomial ODE descriptive mathematics

- exp is the solution of $y' = y$, $y(0) = 1$.
- e is exp(1).
- tanh is the solution of $y' = 1 - y^2$, $y(0) = 1$.
- sin is the first projection of $y' = (y_1', y_2') = (y_2, -y_1)$, $y(0) = (0, 1)$.
- cos its second projection.
- sinh is the first projection of $y' = (y_2, y_1)$, $y(0) = (0, 1)$.
- cosh its second projection.
- $\frac{1}{1+t}$ is the solution of $y' = -y^2$, $y(0) = 1$
- $\frac{1}{1+t^2}$ is the first projection of $y' = (-y_2 y_1^2 - y_2 y_3 y_1^2, 1 + y_3, 0)$, $y(0,0,0) = (1,0,1)$.
- arctan is the first projection of $y' = (y_2, -y_3 y_2^2 - y_3 y_4 y_2^2, 1 + y_4, 0)$, $y(0) = (0,1,0,1)$

Polynomial ODE descriptive mathematics

- exp is the solution of $y' = y$, $y(0) = 1$.
- e is exp(1).
- tanh is the solution of $y' = 1 - y^2$, $y(0) = 1$.
- sin is the first projection of $y' = (y_1', y_2') = (y_2, -y_1)$, $y(0) = (0, 1)$.
- cos its second projection.
- sinh is the first projection of $y' = (y_2, y_1)$, $y(0) = (0, 1)$.
- cosh its second projection.
- $\frac{1}{1+t}$ is the solution of $y' = -y^2$, $y(0) = 1$
- $\frac{1}{1+t^2}$ is the first projection of $y' = (-y_2 y_1^2 - y_2 y_3 y_1^2, 1 + y_3, 0)$, $y(0, 0, 0) = (1, 0, 1)$.
- arctan is the first projection of $y' = (y_2, -y_3 y_2^2 - y_3 y_4 y_2^2, 1 + y_4, 0)$, $y(0) = (0, 1, 0, 1)$
- 4 arctan is the first projection of $y' = (y_2 + y_5 y_2 + y_6 y_2 + y_7 y_2, -y_3 y_2^2 - y_3 y_4 y_2^2, 1 + y_4, 0, 0, 0, 0)$, $y(0) = (0, 1, 0, 1, 1, 1, 1)$.

Polynomial ODE descriptive mathematics

- exp is the solution of $y' = y$, $y(0) = 1$.
- e is $\exp(1)$.
- tanh is the solution of $y' = 1 - y^2$, $y(0) = 1$.
- sin is the first projection of $y' = (y_1', y_2') = (y_2, -y_1)$, $y(0) = (0, 1)$.
- cos its second projection.
- sinh is the first projection of $y' = (y_2, y_1)$, $y(0) = (0, 1)$.
- cosh its second projection.
- $\frac{1}{1+t}$ is the solution of $y' = -y^2$, $y(0) = 1$
- $\frac{1}{1+t^2}$ is the first projection of $y' = (-y_2 y_1^2 - y_2 y_3 y_1^2, 1 + y_3, 0)$, $y(0, 0, 0) = (1, 0, 1)$.
- arctan is the first projection of $y' = (y_2, -y_3 y_2^2 - y_3 y_4 y_2^2, 1 + y_4, 0)$, $y(0) = (0, 1, 0, 1)$
- $4\arctan$ is the first projection of $y' = (y_2 + y_5 y_2 + y_6 y_2 + y_7 y_2, -y_3 y_2^2 - y_3 y_4 y_2^2, 1 + y_4, 0, 0, 0, 0)$, $y(0) = (0, 1, 0, 1, 1, 1, 1)$.
- π is $4\arctan(1)$.

Polynomial ODE descriptive mathematics

- 2 is $+_1(1)$, with $+_1$ solution of $y' = 1$, $y(0) = 1$.
- 3 is $+_2(1)$, with $+_2$ first projection of solution of $y' = (y_2 + y_3, 0, 0)$, $y(0) = (1, 1, 1)$.

  ...

- k is $+_{k-1}(1)$, with $+_{k-1}$ first projection of solution of $y' = (y_2 + \cdots + y_k, 0, \ldots, 0)$, $y(0) = (1, 1, \ldots, 1)$.
- $-k$ is $-_{k-1}(-1)$, with $-_{k-1}$ first projection of solution of $y' = (-y_2 - \cdots - y_k, 0, \ldots, 0)$, $y(0) = (1, 1, \ldots, 1)$.

Polynomial ODE descriptive mathematics

- 2 is $+_1(1)$, with $+_1$ solution of $y' = 1,\ y(0) = 1$.
- 3 is $+_2(1)$, with $+_2$ first projection of solution of $y' = (y_2 + y_3, 0, 0),\ y(0) = (1, 1, 1)$.

  ...
- k is $+_{k-1}(1)$, with $+_{k-1}$ first projection of solution of $y' = (y_2 + \cdots + y_k, 0, \ldots, 0),\ y(0) = (1, 1, \ldots, 1)$.
- $-k$ is $-_{k-1}(-1)$, with $-_{k-1}$ first projection of solution of $y' = (-y_2 - \cdots - y_k, 0, \ldots, 0),\ y(0) = (1, 1, \ldots, 1)$.

- $0 + z$ is the solution of $+'(0, t) = 1,\ +(0, 0) = 0$.
- $y + z$ is the solution of $+'(t, z) = 1,\ +(0, z) = z$.

- $0 * z$ is the solution of $*'(0, t) = 0,\ *(0, 0) = 0$.
- $y * z$ is the solution of $*'(t, z) = z,\ +(0, z) = 0$.

Polynomial ODE descriptive mathematics

- $\frac{1}{x+1}$ is the solution of $y' = -y^2$, $y(0) = 1$
- $\frac{1}{2}$ is $\frac{1}{1+1}$
- $\ln(x+1)$ is the solution of $y' = (y_1, -y_2^2)$, $y(0) = (0, 1)$.
- $\ln(2)$ is $\ln(1+1)$.

Polynomial ODE descriptive mathematics

- $\frac{1}{x+1}$ is the solution of $y' = -y^2$, $y(0) = 1$
- $\frac{1}{2}$ is $\frac{1}{1+1}$
- $\ln(x+1)$ is the solution of $y' = (y_1, -y_2^2)$, $y(0) = (0, 1)$.
- $\ln(2)$ is $\ln(1+1)$.

- However the current game is **not** so **interesting**:
  - $\frac{1}{x}$ and $\ln(x)$ are not in that class.
    - $\frac{1}{x}$ is the solution of $y' = -y^2$, $y(\mathbf{1}) = 1$,
    - $\ln(x+1)$ is the solution of $y' = (y_1, -y_2^2)$, $y(\mathbf{1}) = (0, 1)$.
  - $\frac{1}{x+2}$ is not in that class:
    - $\frac{1}{x+2}$ is the solution of $y' = -y^2$, $y(0) = \mathbf{1/2}$.

Polynomial ODE descriptive mathematics

- $\frac{1}{x+1}$ is the solution of $y' = -y^2$, $y(0) = 1$
- $\frac{1}{2}$ is $\frac{1}{1+1}$
- $\ln(x+1)$ is the solution of $y' = (y_1, -y_2^2)$, $y(0) = (0, 1)$.
- $\ln(2)$ is $\ln(1+1)$.

- However the current game is **not** so **interesting**:
  - $\frac{1}{x}$ and $\ln(x)$ are not in that class.
    - $\frac{1}{x}$ is the solution of $y' = -y^2$, $y(\mathbf{1}) = 1$,
    - $\ln(x+1)$ is the solution of $y' = (y_1, -y_2^2)$, $y(\mathbf{1}) = (0, 1)$.
  - $\frac{1}{x+2}$ is not in that class:
    - $\frac{1}{x+2}$ is the solution of $y' = -y^2$, $y(0) = \mathbf{1/2}$.

- **Let's have more fun and authorize**
  - $y(x_0) = y_0$ **instead of** $y(0) = y_0$**, with** y_0 **pODE computable constant.**
  - n**-variables functions.**

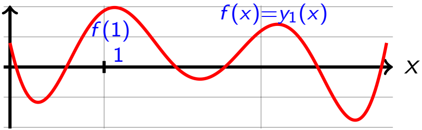
A better game: *n*-variables functions, not so restricted initial condition

We start from

- 0, 1, −1
- and we consider (projections of) solutions of ordinary differential equations of type

$$\begin{cases} y(\mathbf{x_0}) & = & y_0 \\ \mathbf{Jacobian_y(x)} & = & p(y(x)) \end{cases}$$

where p is a (vector of) polynomials, y_0 **is in the class**.



Terminology:

- Such a function $f(x) = y_{1 \dots m}(y)$ will be said to be generated.
- $f(1)$ will then be called a (pODE) computable real.

Programming Exercice

How to transform initial-value problem

$$\left\{ \begin{array}{lcl} y_1' & = & \sin^2 y_2 \\ y_2' & = & y_1 \cos y_2 - e^{e^{y_1}+t} \end{array} \right. \qquad \left\{ \begin{array}{lcl} y_1(0) & = & 0 \\ y_2(0) & = & 0 \end{array} \right.$$

Programming Exercice

How to transform initial-value problem

$$\begin{cases} y_1' &= \sin^2 y_2 \\ y_2' &= y_1 \cos y_2 - e^{e^{y_1}+t} \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \end{cases}$$

into a polynomial initial value problem

$$\begin{cases} \\ \\ \end{cases} \qquad \begin{cases} \\ \\ \end{cases}$$

Programming Exercice

How to transform initial-value problem

$$\begin{cases} y_1' &= \sin^2 y_2 \\ y_2' &= y_1 \cos y_2 - e^{e^{y_1}+t} \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \end{cases}$$

into a polynomial initial value problem

$$\begin{cases} y_1' &= y_3^2 \\ & \\ & \\ & \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ & \\ & \\ & \end{cases}$$

considering $y_3 = \sin y_2$,

Programming Exercice

How to transform initial-value problem

$$\begin{cases} y_1' &= \sin^2 y_2 \\ y_2' &= y_1 \cos y_2 - e^{e^{y_1}+t} \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \end{cases}$$

into a polynomial initial value problem

$$\begin{cases} y_1' &= y_3^2 \\ y_2' &= y_1 y_4 - y_5 \\ & \\ & \\ & \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \\ & \\ & \\ & \end{cases}$$

considering $y_3 = \sin y_2$, $y_4 = \cos y_2$,$y_5 = e^{e^{y_1}+t}$

Programming Exercice

How to transform initial-value problem

$$\begin{cases} y_1' &= \sin^2 y_2 \\ y_2' &= y_1 \cos y_2 - e^{e^{y_1}+t} \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \end{cases}$$

into a polynomial initial value problem

$$\begin{cases} y_1' &= y_3^2 \\ y_2' &= y_1 y_4 - y_5 \\ y_3' &= y_4(y_1 y_4 - y_5) \\ \\ \\ \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \\ y_3(0) &= 0 \\ \\ \\ \end{cases}$$

considering $y_3 = \sin y_2,\ y_4 = \cos y_2\ , y_5 = e^{e^{y_1}+t}$

Programming Exercice

How to transform initial-value problem

$$\begin{cases} y_1' = \sin^2 y_2 \\ y_2' = y_1 \cos y_2 - e^{e^{y_1}+t} \end{cases} \qquad \begin{cases} y_1(0) = 0 \\ y_2(0) = 0 \end{cases}$$

into a polynomial initial value problem

$$\begin{cases} y_1' = y_3^2 \\ y_2' = y_1 y_4 - y_5 \\ y_3' = y_4(y_1 y_4 - y_5) \\ y_4' = -y_3(y_1 y_4 - y_5) \end{cases} \qquad \begin{cases} y_1(0) = 0 \\ y_2(0) = 0 \\ y_3(0) = 0 \\ y_4(0) = 1 \end{cases}$$

considering $y_3 = \sin y_2,\ y_4 = \cos y_2\ , y_5 = e^{e^{y_1}+t}$

Programming Exercice

How to transform initial-value problem

$$\begin{cases} y_1' &= \sin^2 y_2 \\ y_2' &= y_1 \cos y_2 - e^{e^{y_1}+t} \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \end{cases}$$

into a polynomial initial value problem

$$\begin{cases} y_1' &= y_3^2 \\ y_2' &= y_1 y_4 - y_5 \\ y_3' &= y_4(y_1 y_4 - y_5) \\ y_4' &= -y_3(y_1 y_4 - y_5) \\ y_5' &= y_5(y_6 y_3^2 + 1) \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \\ y_3(0) &= 0 \\ y_4(0) &= 1 \\ y_5(0) &= e \end{cases}$$

considering $y_3 = \sin y_2$, $y_4 = \cos y_2$,$y_5 = e^{e^{y_1}+t}$,$y_6 = e^{y_1}$

Programming Exercice

How to transform initial-value problem

$$\begin{cases} y_1' &= \sin^2 y_2 \\ y_2' &= y_1 \cos y_2 - e^{e^{y_1}+t} \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \end{cases}$$

into a polynomial initial value problem

$$\begin{cases} y_1' &= y_3^2 \\ y_2' &= y_1 y_4 - y_5 \\ y_3' &= y_4(y_1 y_4 - y_5) \\ y_4' &= -y_3(y_1 y_4 - y_5) \\ y_5' &= y_5(y_6 y_3^2 + 1) \\ y_6' &= y_6 y_3^2 \end{cases} \qquad \begin{cases} y_1(0) &= 0 \\ y_2(0) &= 0 \\ y_3(0) &= 0 \\ y_4(0) &= 1 \\ y_5(0) &= e \\ y_6(0) &= 1 \end{cases}$$

considering $y_3 = \sin y_2$, $y_4 = \cos y_2$,$y_5 = e^{e^{y_1}+t}$,$y_6 = e^{y_1}$

Facts and Properties

- The class of generated functions include all previously mentioned functions, and **most of the** (analytic) **common functions**.
- It is stable by many operations:
  - if f and g can be generated, then $f+g$, $f-g$, fg, $\frac{1}{f}$, $f \circ g$ can be generated.
- It is stable by ODE solving:
  - if f can be generated, and y satisfies $y' = f(y)$ then y can be generated.
- A generated function must be analytic[4].
- The set of pODE computable constants is a field.

[4] Equals to its Taylor expansion in all point.

Facts and Properties

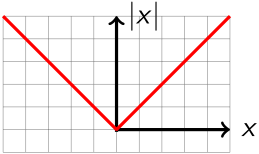
- The class of generated functions include all previously mentioned functions, and **most of the** (analytic) **common functions**.
- It is stable by many operations:
  - if f and g can be generated, then $f+g$, $f-g$, fg, $\frac{1}{f}$, $f \circ g$ can be generated.
- It is stable by ODE solving:
  - if f can be generated, and y satisfies $y' = f(y)$ then y can be generated.
- A generated function must be analytic[4].
  - **Famous** analytic **non-generable functions:** [Shannon 41]
    - Euler's Gamma function $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$ [Hölder 1887]
    - Riemann's Zeta function $\zeta(x) = \sum_{k=0}^{\infty} \frac{1}{k^x}$ [Hilbert].
- The set of pODE computable constants is a field.

[4] Equals to its Taylor expansion in all point.

Menu

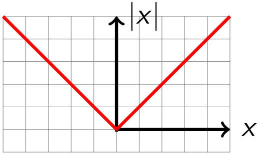
Polynomial ODE descriptive mathematics

- A generated function must be analytic.
- A basic non-generable function:

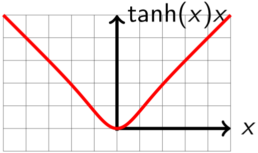


Polynomial ODE descriptive mathematics

- A generated function must be analytic.
- A basic non-generable function:



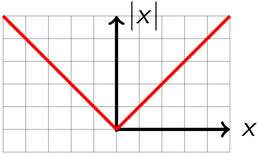
- However $|x|$ is "close" to a generable function:



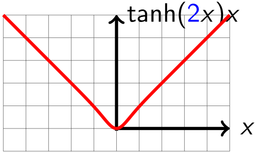
first projection of $y' = ((1 - y_2^2)y_3 + y_2, 1 - y_2^2, 1)$, $y(0) = (0, 0, 0)$.

Polynomial ODE descriptive mathematics

- A generated function must be analytic.
- A basic non-generable function:



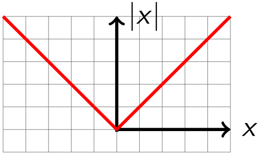
- However $|x|$ is "   close" to a generable function:



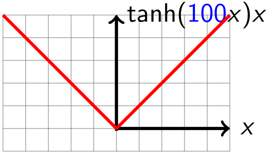
first projection of $y' = (y_4(1-y_2^2)y_3 + y_2, y_4(1-y_2^2), 1, 0)$,
$y(0) = (0, 0, 0, 2)$.

Polynomial ODE descriptive mathematics

- A generated function must be analytic.
- A basic non-generable function:



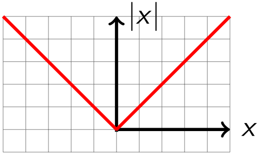
- However $|x|$ is “ close” to a generable function:



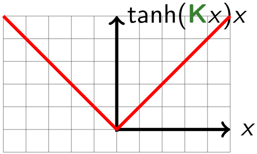
first projection of $y' = (y_4(1 - y_2^2)y_3 + y_2, y_4(1 - y_2^2), 1, 0)$, $y(0) = (0, 0, 0, 100)$.

Polynomial ODE descriptive mathematics

- A generated function must be analytic.
- A basic non-generable function:



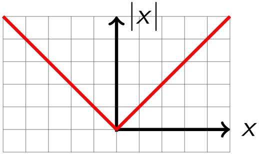
- However $|x|$ is "**uniformly** close" to a generable function:



first projection of $y' = (y_4(1 - y_2^2)y_3 + y_2, y_4(1 - y_2^2), 1, 0)$,
$y(0) = (0, 0, 0, \mathbf{K})$.

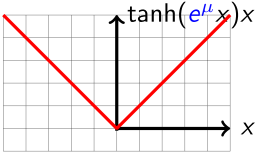
Polynomial ODE descriptive mathematics

- A generated function must be analytic.
- A basic non-generable function:



- However $|x|$ is "$e^{-\mu}$ **uniformly** close" to a generable function:
  - Formally: for all $\mu > 0$, x,

$$|x| - e^{-\mu} \leqslant y(x) \leqslant |x| + e^{-\mu}$$



first projection of $y' = (y_4(1-y_2^2)y_3 + y_2, y_4(1-y_2^2), 0, 0)$, $y(0) = (0, 0, 0, e^{\mu})$.

Alternative statement

- $|x|$ is "**uniformly** close" to a generable function:
  - Can we avoid such a "strange"/"unatural" dependance in the initial condition?
  - Yes, if we don't ask for **real time** computation!

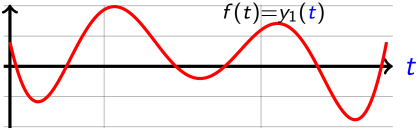
Alternative statement

- $|x|$ is "**uniformly** close" to a generable function:
  - Can we avoid such a "strange"/"unatural" dependance in the initial condition?
  - Yes, if we don't ask for **real time** computation!

Replace **real-time** concept:

- $f(t)$ must be produced **at time** t

  with precision $e^{-\mu}$

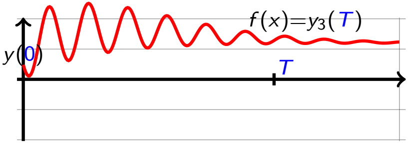


$$y(0) = F(0, \mu)$$
$$f(t) = y_1(t)$$

By a **more modern concept**:

- $f(t)$ must be produced **at time** T

  with precision $e^{-\mu}$



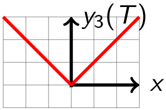
$$y(0) = (x, \mu, y_0)$$
$$f(x) = y_1(T)$$

This is a more general notion of computability

- A generated function can always be computed in that sense.
- Illustration for $|x|$
  - **Simple idea**: consider **a path** $y(t)$ going from $y(0) = (x, \mu, \dots)$ to $y(T) = (x, \mu, abs(x,\mu), \dots)$

    where $abs(x,\mu) = \tanh(e^{-\mu}x)x$ is previous function.

  - **Graphically:**



with $|x| - e^{-\mu} \leqslant y_3(T) \leqslant |x| + e^{-\mu}, \quad x = y_1(0), \mu = y_2(0)$

This is a more general notion of computability

- A generated function can always be computed in that sense.
- Illustration for $|x|$
  - **Simple idea**: consider **a path** $y(t)$ going from $y(0) = (x, \mu, \dots)$ to $y(T) = (x, \mu, abs(x, \mu), \dots)$

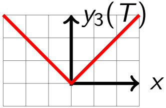
    where $abs(x, \mu) = \tanh(e^{-\mu} x) x$ is previous function.

    - For example, for $T = 1$,

    $$y(t) = (x, \mu, abs(tx, t\mu), t)$$

    solution of $\mathbf{y}'(\mathbf{t}) = (\mathbf{0}, \mathbf{0}, \mathbf{p_y}(\mathbf{y}(\mathbf{t})), \mathbf{1}), \qquad \mathbf{y}(\mathbf{0}) = (\mathbf{x}, \mu, \mathbf{1}, \mathbf{1}),$
    with
    $p_y(y(t)) = (1 - \tanh^2(e^{t\mu} tx))(\mu e^{t\mu} tx + e^{t\mu} x) + x \tanh(e^{t\mu} tx)$

  - **Graphically:**



with $|x| - e^{-\mu} \leqslant y_3(T) \leqslant |x| + e^{-\mu}, \quad x = y_1(0), \mu = y_2(0)$

This is a more general notion of computability

- A generated function can always be computed in that sense.
- Illustration for $|x|$
  - **Simple idea**: consider **a path** $y(t)$ going from $y(0) = (x, \mu, \ldots)$ to $y(T) = (x, \mu, abs(x, \mu), \ldots)$

    where $abs(x, \mu) = \tanh(e^{-\mu} x)x$ is previous function.

    - For example, for $T = 1$,

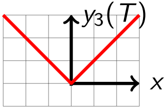
    $$y(t) = (x, \mu, abs(tx, t\mu), t)$$

    solution of $\mathbf{y}'(\mathbf{t}) = (\mathbf{0}, \mathbf{0}, \mathbf{p_y}(\mathbf{y}(\mathbf{t})), \mathbf{1}), \qquad \mathbf{y}(\mathbf{0}) = (\mathbf{x}, \mu, \mathbf{1}, \mathbf{1}),$

    with

    $p_y(y(t)) =$
    $(1 - \tanh^2(e^{y_4 y_2} y_4 y_1))(y_2 e^{y_4 y_2} y_4 y_1 + e^{y_4 y_2} y_1) + y_1 \tanh(e^{y_4 y_2} y_4 y_2)$

  - **Graphically:**

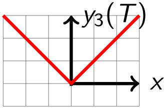


with $|x| - e^{-\mu} \leqslant y_3(T) \leqslant |x| + e^{-\mu}, \quad x = y_1(0), \mu = y_2(0)$

- If you want only polynomial ODEs:
  - Do as in previous exercice for the system for $|x|$:

$$\begin{cases} y_1' &= 0 \\ y_2' &= 0 \\ y_3' &= (1 - \tanh^2(e^{y_4 y_2} y_4 y_1))(y_2 e^{y_4 y_2} y_4 y_1 + e^{y_4 y_2} y_1) + y_1 \tanh(e^{y_4 y_2} y_4 y_2) \\ y_4' &= 1 \end{cases}$$

$$\begin{cases} y_1(0) &= x \\ y_2(0) &= \mu \\ y_3(0) &= 1 \\ y_4(0) &= 1 \end{cases}$$



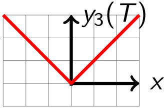
- Other paths could be used.
  E.g. if one wants better and better precision, or that this works even for $t \geq 1$.

$$y(t) = (x, \mu, abs(\min(tx, 1), t\mu), t)$$

- If you want only polynomial ODEs:
  - Do as in previous exercice for the system for $|x|$:

$$\left\{\begin{array}{lcl} y_1' & = & 0 \\ y_2' & = & 0 \\ y_3' & = & (1-\tanh^2(e^{y_4 y_2} y_4 y_1))(y_2 e^{y_4 y_2} y_4 y_1 + e^{y_4 y_2} y_1) + y_1 \tanh(e^{y_4 y_2} y_4 y_2) \\ y_4' & = & 1 \end{array}\right.$$

$$\left\{\begin{array}{lcl} y_1(0) & = & x \\ y_2(0) & = & \mu \\ y_3(0) & = & 1 \\ y_4(0) & = & 1 \end{array}\right.$$



- Other paths could be used.
  E.g. if one wants better and better precision, or that this works even for $t \geq 1$.

$$y(t) = (x, \mu, abs(\frac{1 + tx - abs(tx-1, t\mu)}{2}, t\mu), t)$$

using $\min(a,b) = (a+b-|a-b|)/2$.

Main Statement: Computability

- $|x|$ can be computed in that sense.

[5](OB, D. Graça, A. Pouly 2016's) Improvment of Journal of Complexity, 2007, OB, M. Campagnolo, D. Graça, E. Hainry

Main Statement: Computability

- $|x|$ can be computed in that sense.
- Γ, ζ can be computed in that sense.

[5] (OB, D. Graça, A. Pouly 2016's) Improvment of Journal of Complexity, 2007, OB, M. Campagnolo, D. Graça, E. Hainry

Main Statement: Computability

- $|x|$ can be computed in that sense.
- Γ, ζ can be computed in that sense.

- **Theorem** [OB, M. Campagnolo, D. Graça, E. Hainry[5]] Any computable function can be computed in that sense, and conversely.

[5](OB, D. Graça, A. Pouly 2016's) Improvment of Journal of Complexity, 2007, OB, M. Campagnolo, D. Graça, E. Hainry

Main Statement: Computability

- $|x|$ can be computed in that sense.
- Γ, ζ can be computed in that sense.

- **Theorem** [OB, M. Campagnolo, D. Graça, E. Hainry[5]] Any computable function can be computed in that sense, and conversely.

- **The notion of computable function can be defined using pODE only !!**

[5](OB, D. Graça, A. Pouly 2016's) Improvment of Journal of Complexity, 2007, OB, M. Campagnolo, D. Graça, E. Hainry

Main Statement: Computability

- $|x|$ can be computed in that sense.
- Γ, ζ can be computed in that sense.

- **Theorem** [OB, M. Campagnolo, D. Graça, E. Hainry[5]] Any computable function can be computed in that sense, and conversely.

- **The notion of computable function can be defined using pODE only !!**
  - No need to talk of Turing machines, or equivalent concept to define computable functions.

[5](OB, D. Graça, A. Pouly 2016's) Improvment of Journal of Complexity, 2007, OB, M. Campagnolo, D. Graça, E. Hainry

Formal Theorem[6]

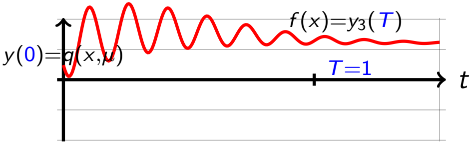
Let $a, b \in \mathbb{Q}$.

- $f \in C^0([a,b],\mathbb{R})$ is computable

iff

- ▶ y satisfies a pODE
- ▶ $y_{1..m}$ is $e^{-\mu}$-close to $f(x)$ after time $T = 1$

- Picture:



[6](OB, D. Graça, A. Pouly 2016's) Improvment of Journal of Complexity, 2007, OB, M. Campagnolo, D. Graça, E. Hainry

Formal Theorem[6]

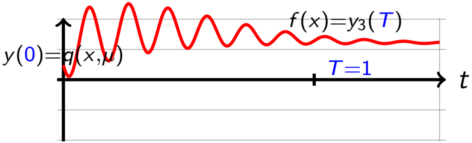
Let $a, b \in \mathbb{Q}$.

- $f \in C^0([a,b],\mathbb{R})$ is computable

  iff

  $\exists$ polynomials p, q s.t. $\forall x \in \operatorname{dom} f$,
  there exists a (unique) y satisfying for all $t \in \mathbb{R}_+$:
  - $y(0) = q(x,\mu)$ and $y'(t) = p(y(t))$ with $\|y'(t)\|_\infty \geqslant 1$
    - y satisfies a pODE
  - if $t \geqslant T = 1$ then $\|y_{1..m}(t) - f(x)\|_\infty \leqslant e^{-\mu}$
    - $y_{1..m}$ is $e^{-\mu}$-close to $f(x)$ after time $T = 1$

- Picture:



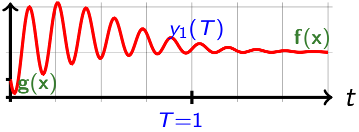
[6](OB, D. Graça, A. Pouly 2016's) Improvment of Journal of Complexity, 2007, OB, M. Campagnolo, D. Graça, E. Hainry

Menu

Time complexity for continuous systems

- Variable t is rather arbitary.

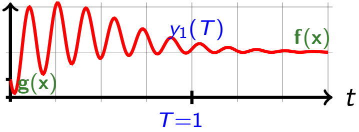
$$y(0) = g(x) \qquad y' = h(y)$$



Time complexity for continuous systems

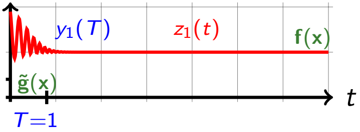
- Variable t is rather arbitary.

$y(0) = g(x) \qquad y' = h(y)$



$z(t){=}y(e^t)$ $\rightsquigarrow$

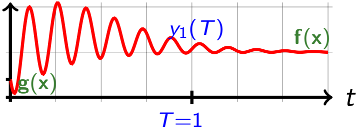
$z(0) = \tilde{g}(x) \qquad z' = \tilde{h}(z)$



Time complexity for continuous systems

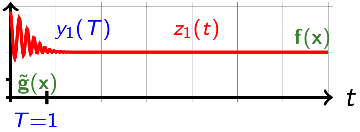
- Variable t is rather arbitary.

$y(0) = g(x) \qquad y' = h(y)$



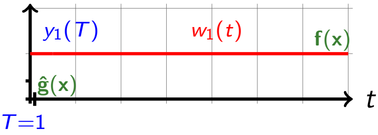
$z(t)=y(e^t)$ $\rightsquigarrow$

$z(0) = \tilde{g}(x) \qquad z' = \tilde{h}(z)$

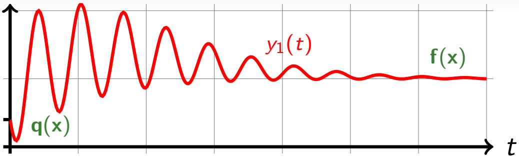


$w(t)=y\left(e^{e^t}\right)$ $\rightsquigarrow$

$w(0) = \hat{g}(x) \qquad w' = \hat{h}(w)$



A Simple & Key Idea: curvi-linear abscissa



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

Length based: T

$$\ell(t) = \text{length of } y \text{ over } [0, t]$$
$$= \int_0^t \|p(y(u))\|_\infty \, du$$

Consider parameterization

$$t = \text{length of } y \text{ over } [0, t]$$

I.e.:

Follow curve **at constant speed.**

Main Statement: Complexity

- **Theorem**[7] Any polynomial time computable function can be computed in polynomial length, and conversely.

[7] ICALP 2016 Track B Best Paper Award, OB, D. Graça, A. Pouly

Main Statement: Complexity

- **Theorem**[7] Any polynomial time computable function can be computed in polynomial length, and conversely.

- **The notion of polynomial time computable function can be defined using pODE only !!**

[7] ICALP 2016 Track B Best Paper Award, OB, D. Graça, A. Pouly

Main Statement: Complexity

- **Theorem**[7] Any polynomial time computable function can be computed in polynomial length, and conversely.

- **The notion of polynomial time computable function can be defined using pODE only !!**

  - No need to talk of Turing machines, or equivalent concept to define polynomial time computable functions.

[7] ICALP 2016 Track B Best Paper Award, OB, D. Graça, A. Pouly

Formal Theorem [8]

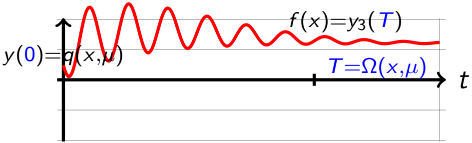
Let $a, b \in \mathbb{Q}$.

- $f \in C^0([a, b], \mathbb{R})$ is polynomial-time computable

iff

  - ▶ y satisfies a pODE
  - ▶ $y_{1..m}$ is $e^{-\mu}$-close to $f(x)$ after a polynomial length

- Picture:



[8] ICALP 2016 Track B Best Paper Award, OB, D. Graça, A. Pouly

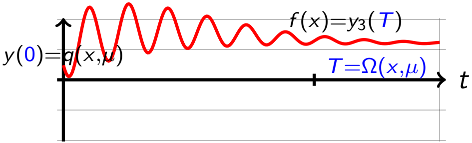
Formal Theorem [8]

Let $a, b \in \mathbb{Q}$.

- $f \in C^0([a,b],\mathbb{R})$ is polynomial-time computable

  iff

  $\exists$ polynomials p, q, Ω s.t. $\forall x \in \operatorname{dom} f$,
  there exists a (unique) y satisfying for all $t \in \mathbb{R}_+$:
  - $y(0) = q(x,\mu)$ and $y'(t) = p(y(t))$ with $\|y'(t)\|_\infty \geqslant 1$
    - y satisfies a pODE
  - if $\operatorname{len}_y(0,t) \geqslant \Omega(\|x\|_\infty, \mu)$ then $\|y_{1..m}(t) - f(x)\|_\infty \leqslant e^{-\mu}$
    - $y_{1..m}$ is $e^{-\mu}$-close to $f(x)$ after a polynomial length
- Picture:



[8]ICALP 2016 Track B Best Paper Award, OB, D. Graça, A. Pouly

For Discrete People [9]

Fix a "reasonable" way to encode words w, length of input, and decision:

- For example $\psi(w) = \left(\sum_{i=1}^{|w|} w_i k^{-i}, |w|\right)$, and $\geqslant 1$, $\leqslant -1$.

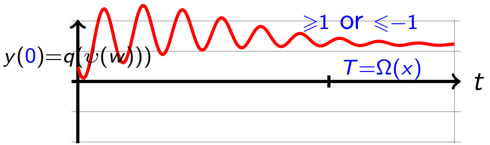
Then:

- $\mathcal{L} \subseteq \{0,1\}^*$ is polynomial-time computable

iff

- y satisfies a pODE
- decision is made after a polynomial length
- and corresponds to $\mathcal{L}$

- Picture:



For Discrete People [9]

Fix a "reasonable" way to encode words w, length of input, and decision:

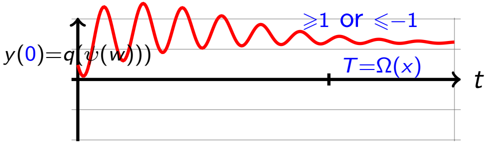
- For example $\psi(w) = \left(\sum_{i=1}^{|w|} w_i k^{-i}, |w|\right)$, and $\geqslant 1$, $\leqslant -1$.

Then:

- $\mathcal{L} \subseteq \{0,1\}^*$ is polynomial-time computable

iff

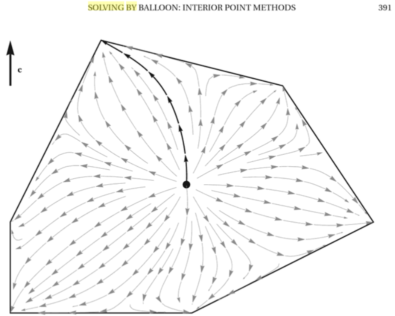
- $\exists$ polynomials p, q, Ω s.t. $\forall w$,
  there exists a (unique) y satisfying for all $t \in \mathbb{R}_+$:
  - $y(0) = q(\psi(w))$ and $y'(t) = p(y(t))$ with $\|y'(t)\|_\infty \geqslant 1$
    ▶ y satisfies a pODE
  - if $\text{len}_y(0,t) \geqslant \Omega(|w|)$ then $|y_1(t)| \geqslant 1$
    ▶ decision is made after a polynomial length
  - $w \in \mathcal{L}$ iff $y_1(t) \geqslant 1$
    ▶ and corresponds to $\mathcal{L}$
- Picture:



Menu

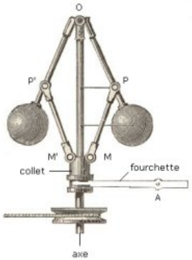
- Finding zeros of a function: $x' = -f(x)$

- Linear Programming:

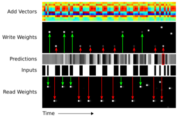


See e.g.: The Nature of Computation, C. Moore and S. Mertens, Oxford University Press.

- Computing optimal solutions:

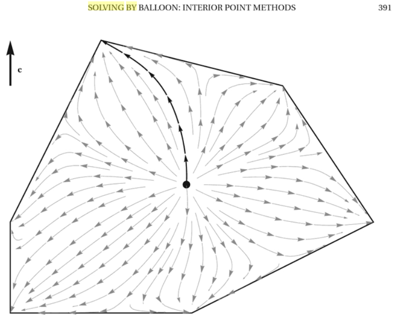


- Neural Networks, Deep learning, Differential Neural Computers, Neural Turing Machines, and variants...



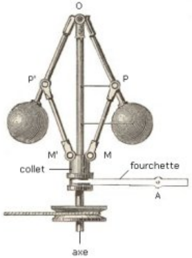
- Finding zeros of a function: $x' = -f(x)$

- Linear Programming:

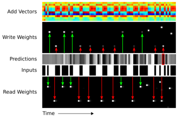


See e.g.: The Nature of Computation, C. Moore and S. Mertens, Oxford University Press.

- Computing optimal solutions:



- Neural Networks, Deep learning, Differential Neural Computers, Neural Turing Machines, and variants...



- And Turing machines.

Menu

For Nostalgic of Turing Machines: Some ideas of the proof

- Polynomial time ODE can be solved in a time polynomial in their length[10].

- Need to simulate a **Turing machine** using polynomial ODEs.

  - **Ingredient 1:** simulating a Turing machine using iterations of piecewise linear function

  - **Ingredient 2:** iterating a function using polynomial ODEs

  - **Ingredient 3:** everything must be dealt with analytic functions, i.e. by keeping errors under control.

[10] TCS 2016 A. Pouly, D. Graça

Turing Machines

- Let M be some one tape Turing machine, with m states and 10 symbols.
- If

$$\ldots B\ B\ B\ a_{-k}\ a_{-k+1} \ldots\ a_{-1}\ a_0\ a_1 \ldots\ a_n\ B\ B\ B \ldots$$

is the tape content of M, it can be seen as

$$\begin{aligned} y_1 &= a_0 a_1 \ldots a_n \\ y_2 &= a_{-1} a_{-2} \ldots a_{-k} \end{aligned} \tag{1}$$

- The configuration of M is then given by three values: its internal state s, y_1 and y_2.

Alternative View of a Turing Machine

$$\begin{aligned} y_1 &= a_0 10^{-1} + a_1 10^{-2} + \ldots + a_n 10^{-n-1} \\ y_2 &= a_{-1} 10^{-1} + a_{-2} 10^{-2} + \ldots + a_{-k} 10^{-k}. \end{aligned} \quad (2)$$

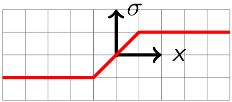
$$y(t+1) = f(y(t))$$

Turing Machine	PAM
State Space $\{q_1, q_2, \cdots, q_m\} \times \Sigma^*$	State Space $[1, m+1] \times [0, 1]$
State $(q_i, a_{-m} \ldots a_{-1}, a_0 \ldots a_n)$	State $x = s + y_2,\ y = y_1$
q_1: if 2 is read, then write 4; goto q_2	$\begin{cases} x := x+1 \\ y := y + \frac{2}{10} \end{cases}$ if $\begin{cases} 1 \leq x < 2 \\ \frac{2}{10} \leq y < \frac{3}{10} \end{cases}$
q_5: if 3 is read, then move right; goto q_1	$\begin{cases} x := \frac{x-5}{10} + \frac{3}{10} + 1 \\ y := 10 * y - 3 \end{cases}$ if $\begin{cases} 5 \leq x < 6 \\ \frac{3}{10} \leq y < \frac{4}{10} \end{cases}$
q_3: if 5 is read, then move left; goto q_7	$\begin{cases} x := 10(x-3) - j + 7 \\ y := \frac{y}{10} + \frac{j}{10} \end{cases}$ if $\begin{cases} 3 + \frac{j}{10} \leq x < 3 + \frac{j+1}{10} \\ \frac{5}{10} \leq y < \frac{6}{10} \end{cases}$ for $j \in \{0, 1, \ldots, 9\}$.

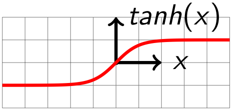
Key remark: f is piecewise affine

Morality

- If you prefer, a Turing Machine can be seen as a **piecewise affine function**
  - $x_i(t+1) = \sigma\left(\sum_{j=1}^{N} a_{i,j} x_j(t) + c_i\right)$ is even (basically) sufficient.



  - Analytic version:



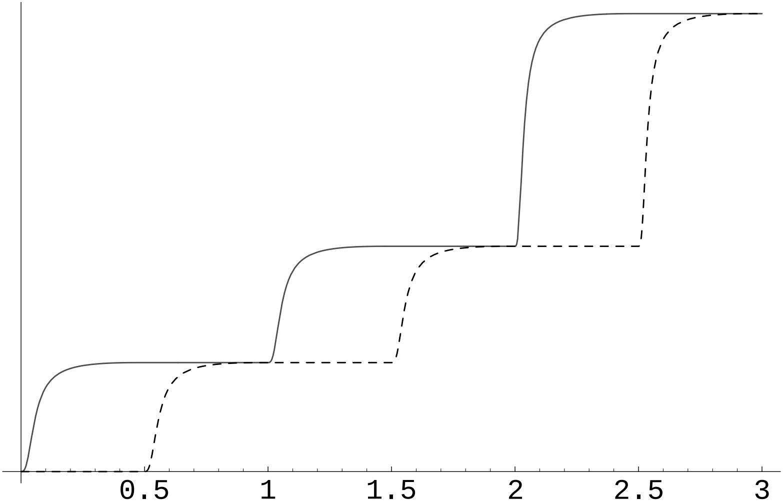
- It remains to simulate

$$y(t+1) := y(t)$$

for $t = 1, 2, \ldots$.

- Remaining analytic...

Example: $y(t+1) := 2 * y(t)$



Simulation of iterations of $h(n) = 2^n$ by ODEs.

Ingredient 2: Branicky's clock (1995): with non-analytic functions

- We want to alternate $z_2 := \omega(z_1)$, $z_1 := z_2$.

- Key observation: the solution of

$$y' = c(g - y)^3 \phi(t)$$

converges at $t = 1/2$ the goal g with some arbitray precision, independantly from initial condition at $t = 0$

for any function ϕ of positive integral if c is sufficiently big.

  - **If you prefer, this roughly does $y(1/2) := g$.**

- The following system is a solution

$$\left\{ \begin{array}{lcl} z_1' & = & c_1(z_2 - z_1)^3 \theta(-\sin(2\pi t)) \\ z_2' & = & c_2(\omega(z_1) - z_2)^3 \theta(\sin(2\pi t)) \end{array} \right. \quad \left\{ \begin{array}{lcl} z_1(0) & = & x_0 \\ z_2(0) & = & x_0 \end{array} \right.$$

considering functions:

  - θ such that $\theta(x) = 0$ if $x \leq 0$, $\theta(x) = x^2$ if $x \geq 0$.

Menu

Conclusion/Take Home Message

- Programming with ODEs is **simple** and **fun**.
- Many concepts from **mathematics** can be defined using polynomial ODEs
- Many concepts from **computer science** can be defined using polynomial ODEs
  - Computable functions.
  - Polynomial Time Computable Functions

Conclusion/Take Home Message

- Programming with ODEs is **simple** and **fun**.
- Many concepts from **mathematics** can be defined using polynomial ODEs
- Many concepts from **computer science** can be defined using polynomial ODEs
  - Computable functions.
  - Polynomial Time Computable Functions
  - *NP*, *PSPACE*, ...?