Introduction to MPRI course 2.33
An introduction

Olivier Bournez
LIX, Ecole Polytechnique, France

MPRI

2018

Menu

Back to Foundations of Computer Science

Descriptive Mathematics

Descriptive Computer/Computability Science

Descriptive Computer/Complexity Science

Some applications

The subject of this course

What is a computer ?

Laptop

What is a computer ?

Laptop

Supercomputer

What is a computer ?

Laptop

Supercomputer

Servers

What is a computer ?

Laptop

Supercomputer

Servers

The highest-selling
single computer model
of all time

source: Guinness World Records

What is a computer ?

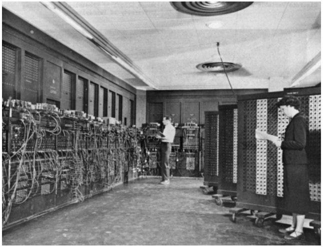
Laptop

Supercomputer

Servers

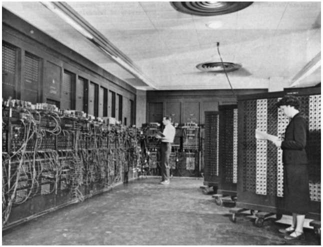
Commodore 64

What is a computer ?



ENIAC

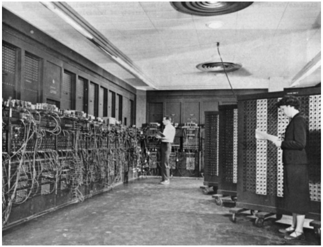
What is a computer ?



ENIAC

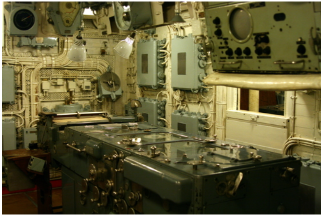
Kelvin's Tide Predicter

What is a computer ?



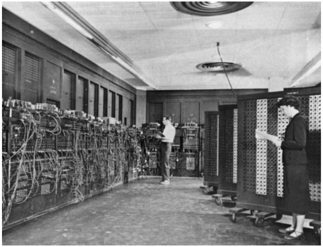
ENIAC

Kelvin's Tide Predicter



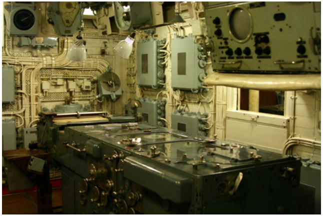
Admiralty Fire Control Table

What is a computer ?



ENIAC

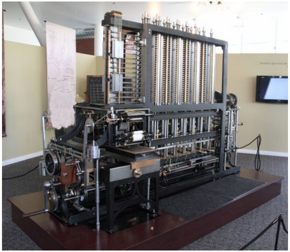
Kelvin's Tide Predicter



Admiralty Fire Control Table

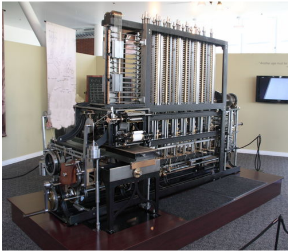
Differential Analyzer

What is a computer ?

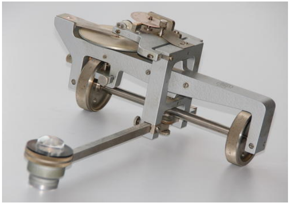


Difference Engine

What is a computer ?

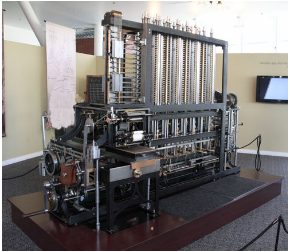


Difference Engine

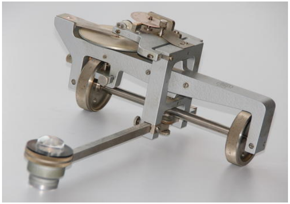


Linear Planimeter

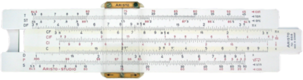
What is a computer ?



Difference Engine

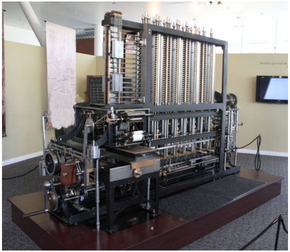


Linear Planimeter

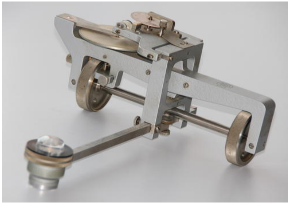


Slide Rule

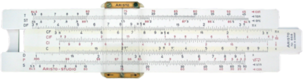
What is a computer ?



Difference Engine



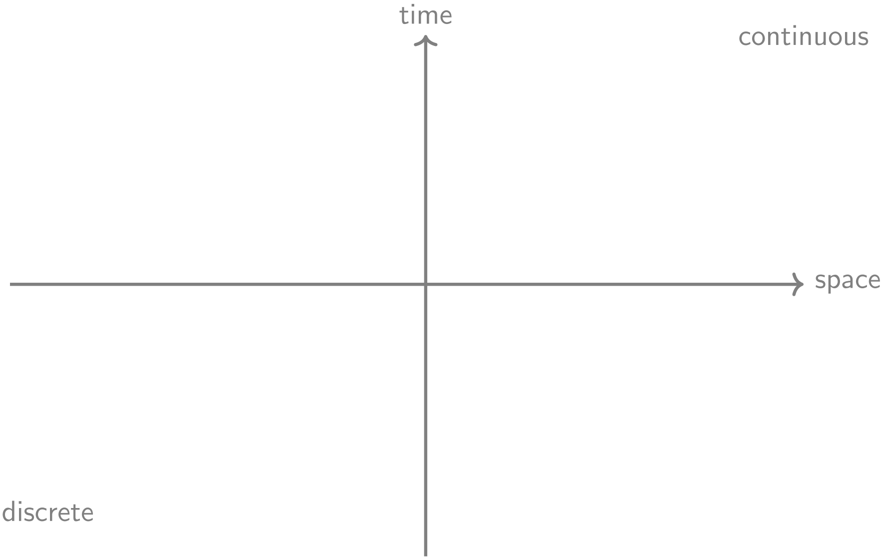
Linear Planimeter



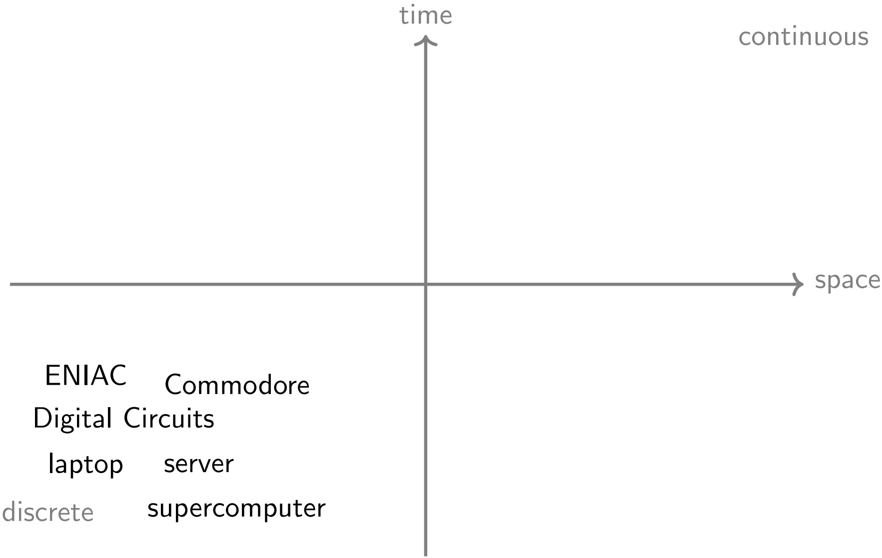
Slide Rule

Antikythera mechanism

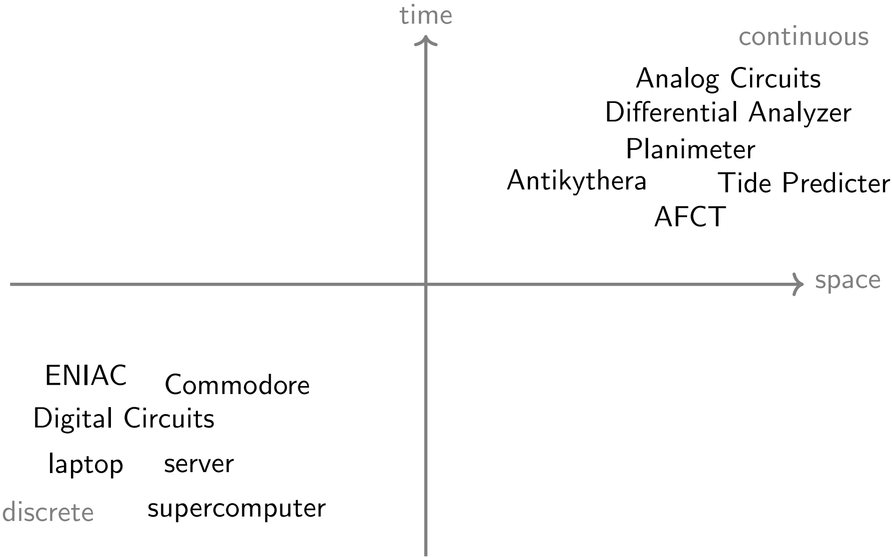
A first classification



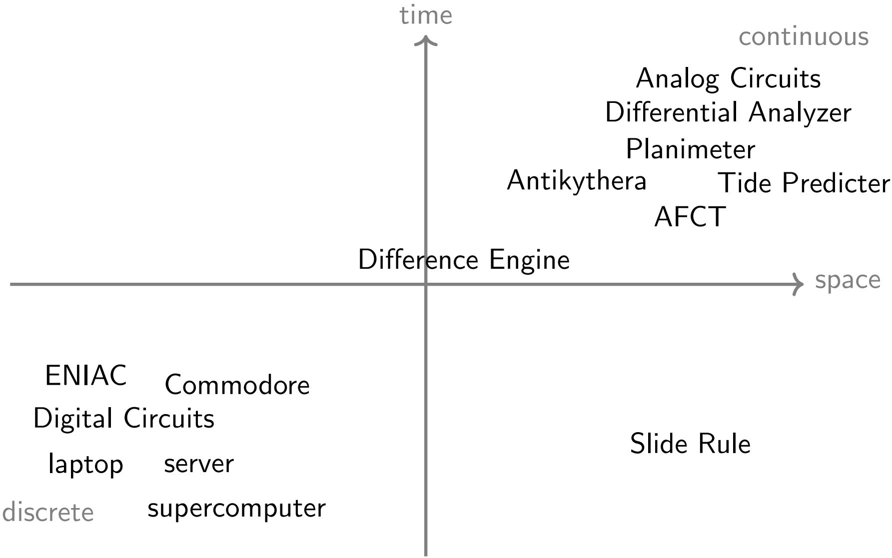
A first classification



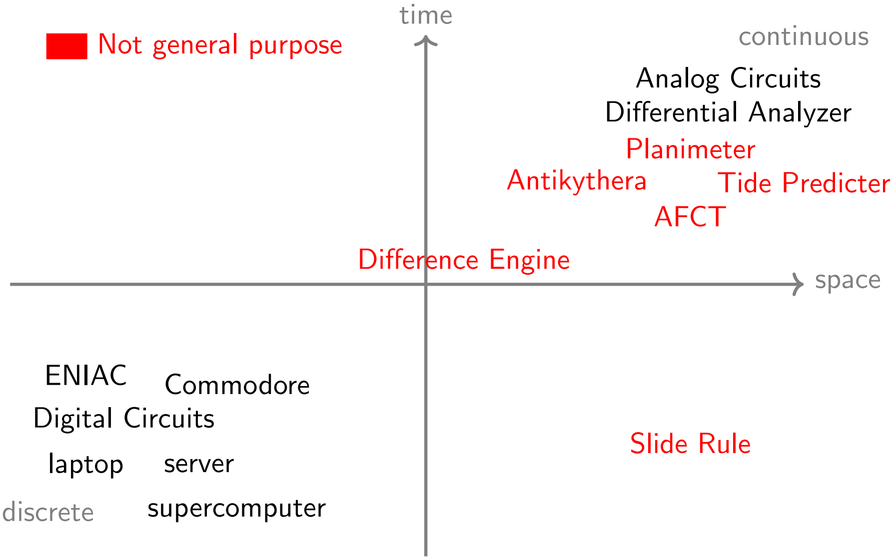
A first classification



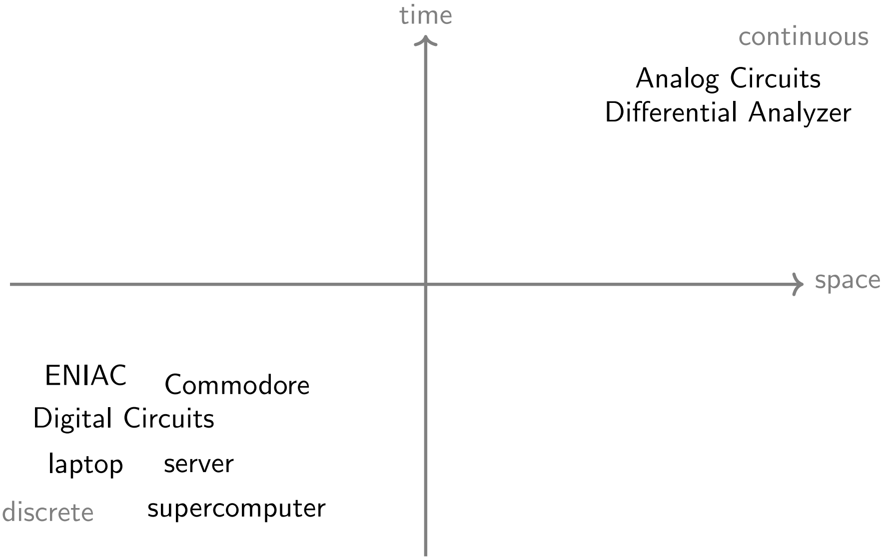
A first classification



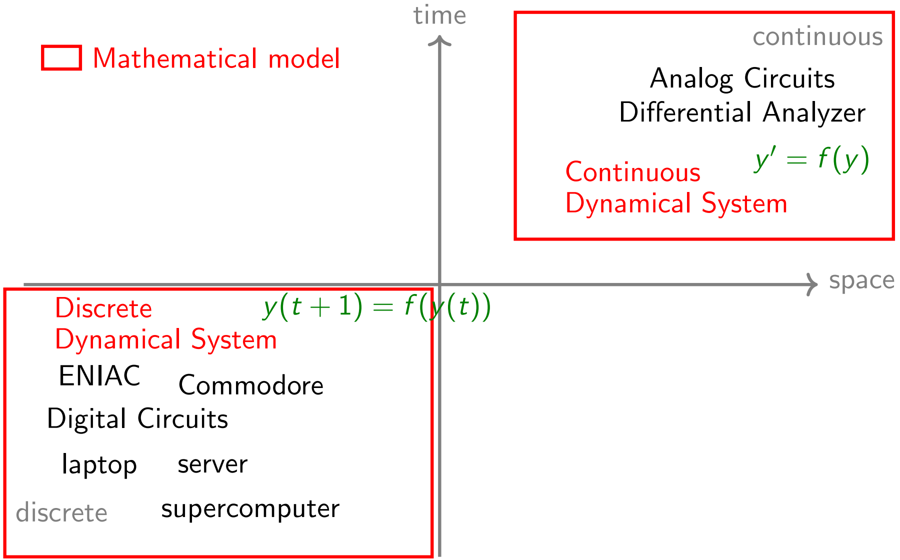
A first classification



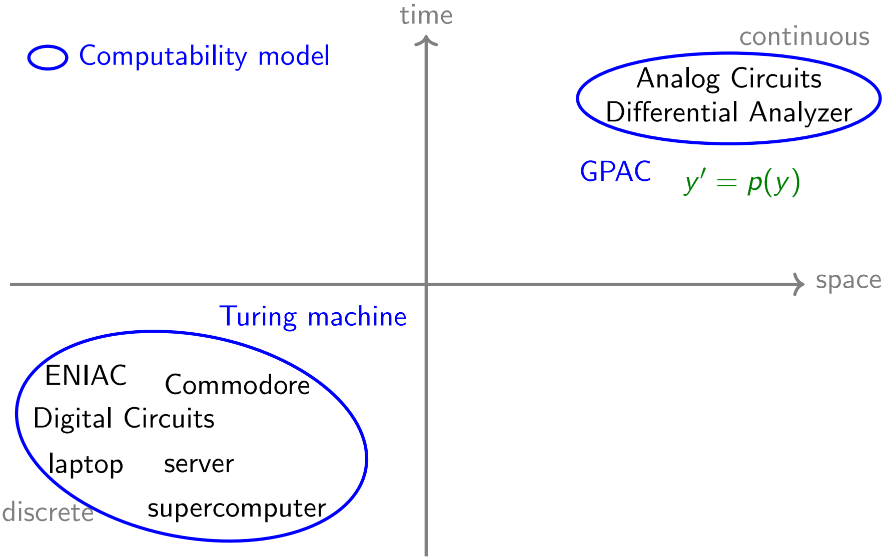
A first classification



A first classification



A first classification



More models!

Physical Computer	**Model**
Laptop, ...	Turing machines
	λ-calculus
	Recursive functions
	Circuits
	Discrete dynamical systems
Differential Analyzer, ...	GPAC
	Continuous dynamical systems

More models!

Physical Computer	Model
Laptop, ...	Turing machines
	λ-calculus
	Recursive functions
	Circuits
	Discrete dynamical systems
Differential Analyzer, ...	GPAC
	Continuous dynamical systems

Church-Turing Thesis

All reasonable models of computation are equivalent.

More models!

Physical Computer	**Model**
Laptop, ...	Turing machines
	λ-calculus
	Recursive functions
	Circuits
	Discrete dynamical systems
Differential Analyzer, ...	GPAC
	Continuous dynamical systems

Church-Turing Thesis

All reasonable models of computation are equivalent.

Implicit corollary

Some models are too general/unreasonable.

More models!

Physical Computer	**Model**
Laptop, ...	Turing machines
	λ-calculus
	Recursive functions
	Circuits
	Discrete dynamical systems
Differential Analyzer, ...	**GPAC**
	Continuous dynamical systems

Church-Turing Thesis

All reasonable models of computation are equivalent.

Implicit corollary

Some models are too general/unreasonable.

Shannon's General Purpose Analog Computer

- The **GPAC** is a mathematical abstraction from Claude Shannon (1941) of the **Differential Analyzers**.

- [Graça Costa 03]: This corresponds to **polynomial Ordinary Differential Equations** (pODEs), i.e. continuous time dynamical systems of the form

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

  where
  - $y : I \to \mathbb{R}^d, t \in I$
  - and p is a (vector of) polynomials.

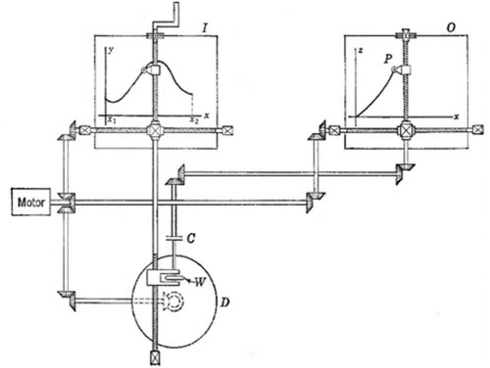
A machine from 20th Century: Differential analyzers

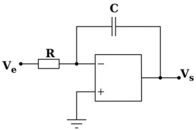
Vannevar Bush's 1938 mechanical Differential Analyser

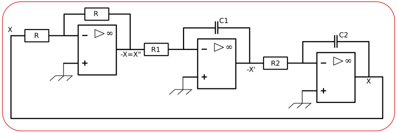
- Underlying principles: Lord Kelvin 1876.
- First ever built: V. Bush 1931 at MIT.
- Applications: from gunfire control up to aircraft design
- Intensively used during U.S. war effort.
- Electronic versions from late 40s, used until 70s

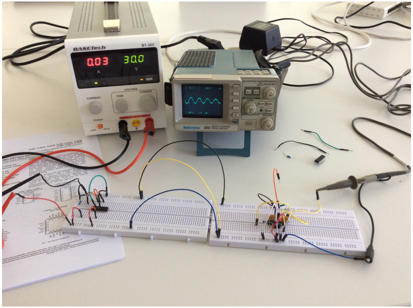
A machine from 21th Century: Analog Paradigm Model-1

- http://analogparadigm.com
- Fully modular
- Basic version.
  - 4 integrators, 8 constants, 8 adders, 8 multipliers.
  - 14 kgs.


I
O
y
x
x_1
x_2
z
P
x
Motor
C
W
D


C
R
V_e
V_s
−
+

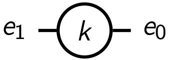

X
R
R
-X=X"
R1
C1
-X'
R2
C2
X


BASETech
BT-305
0.03
30.0
Tektronix 222

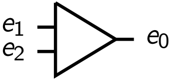
The General Purpose Analog Computer

Shannon's 41 presentation:

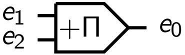
- Basic units:



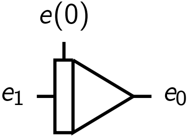
constant: $e_0 = ke_1$



summer: $e_0 = -(e_1 + e_2)$



product: $e_0 = e_1 e_2$



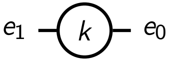
integrator:
$e_0 = -\int_0^t (e_1(u)du + e(0))$

- A function is GPAC-generated if it corresponds to the output of some unit of a GPAC.

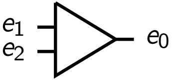
The General Purpose Analog Computer

Shannon's 41 presentation:

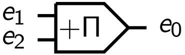
- Basic units:



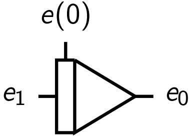
constant: $e_0 = ke_1$



summer: $e_0 = -(e_1 + e_2)$



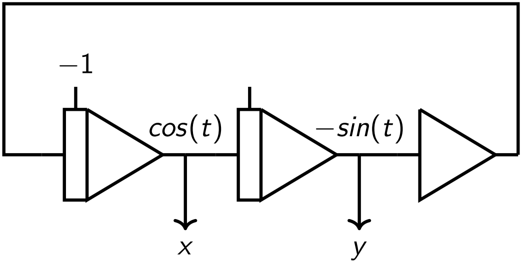
product: $e_0 = e_1 e_2$



integrator:
$e_0 = -\int_0^t (e_1(u)du + e(0))$

- (Feedback connections are allowed).
- A function is GPAC-generated if it corresponds to the output of some unit of a GPAC.

Cosinus and sinus: $x = \cos(t)$, $y = \sin(t)$



$$\begin{cases} x'(t) = -y(t) \\ y'(t) = x(t) \\ x(0) = 1 \\ y(0) = 0 \end{cases} \Rightarrow \begin{cases} x(t) = \cos(t) \\ y(t) = \sin(t) \end{cases}$$

Menu

Let's play the following game

We start from

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

Let's play the following game

We start from

- 0, 1, -1

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

Let's play the following game

We start from

- 0, 1, -1
- and we consider projections of solutions of ordinary differential equations over $\mathbb{R}^d$ of type

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

where p is a (vector of) polynomials[1]

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

Let's play the following game

We start from

- 0, 1, -1
- and we consider projections of solutions of ordinary differential equations over $\mathbb{R}^d$ of type

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

where p is a (vector of) polynomials[1]

Terminology:

- Such a function $f(t) = y_{1\ldots,m}(t)$ will be said to be generated.

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

Let's play the following game

We start from

- 0, 1, -1
- and we consider projections of solutions of ordinary differential equations over $\mathbb{R}^d$ of type

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

where p is a (vector of) polynomials[1]

Terminology:

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

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

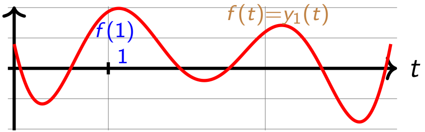
Let's play the following game

We start from

- 0, 1, -1
- and we consider projections of solutions of ordinary differential equations over $\mathbb{R}^d$ of type

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

where p is a (vector of) polynomials[1]



Terminology:

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

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

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 $y' = (-y_2 y_1^2, 1 + y_3, 0)$, $y(0, 0, 0) = (1, 0, 1)$.

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 $y' = (-y_2 y_1^2, 1 + y_3, 0)$, $y(0, 0, 0) = (1, 0, 1)$.
- arctan is the first projection of $y' = (y_2, -y_3 y_2^2, 1 + y_4, 0)$, $y(0) = (0, 1, 0, 1)$

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 $y' = (-y_2 y_1^2, 1 + y_3, 0)$, $y(0, 0, 0) = (1, 0, 1)$.
- arctan is the first projection of $y' = (y_2, -y_3 y_2^2, 1 + y_4, 0)$, $y(0) = (0, 1, 0, 1)$
- $4 \arctan$ is the first projection of $y' = (y_2 + y_5 y_2 + y_6 y_2 + y_7 y_2, -y_3 y_2^2, 1 + y_4, 0, 0, 0, 0)$, $y(0) = (0, 1, 0, 1, 1, 1, 1)$.

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 $y' = (-y_2 y_1^2, 1 + y_3, 0)$, $y(0, 0, 0) = (1, 0, 1)$.
- arctan is the first projection of $y' = (y_2, -y_3 y_2^2, 1 + y_4, 0)$, $y(0) = (0, 1, 0, 1)$
- $4\arctan$ is the first projection of $y' = (y_2 + y_5 y_2 + y_6 y_2 + y_7 y_2, -y_3 y_2^2, 1 + y_4, 0, 0, 0, 0)$, $y(0) = (0, 1, 0, 1, 1, 1, 1)$.
- 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

- $\frac{1}{1+t}$ is the solution of $y' = -y^2$, $y(0) = 1$
- $\frac{1}{2}$ is $\frac{1}{1+1}$
- $\ln(1+t)$ is the solution of $y' = (y_1, -y_2^2)$, $y(0) = (0, 1)$.
- $\ln(2)$ is $\ln(1+1)$.

Polynomial ODE descriptive mathematics

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

Polynomial ODE descriptive mathematics

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

- **Let's generalize a little bit our game**
  - $y(x_0) = y_0$ **instead of** $y(0) = y_0$**, with** y_0 **pODE computable constant.**
  - n**-variables functions.**

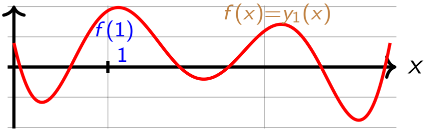
A better game: n-variables functions, not so restricted initial condition

We start from

- 0, 1, -1
- and we consider (projections of) solutions of ordinary differential equations over $\mathbb{R}^d$ 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 $f(x) = y_{1\ldots m}(y)$ 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

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[2].
- The set of pODE computable constants is a field.

[2] Equals to its Taylor expansion in every 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[2].
  - **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].

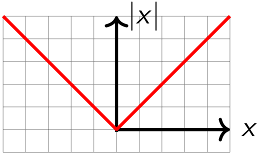
- The set of pODE computable constants is a field.

[2] Equals to its Taylor expansion in every point.

Menu

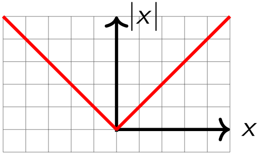
Polynomial ODE descriptive mathematics

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

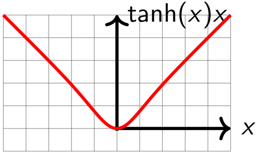


Polynomial ODE descriptive mathematics

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



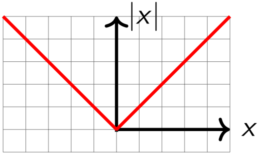
- However $|x|$ is "close" to a generable function:



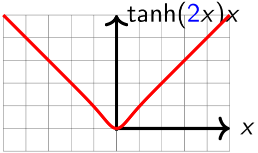
first projection of $y' = ((1 - y_2^2)y_3 + y_2, 1 - y_2^2, 1)$,
$y(0) = (0, 0, 0)$.

Polynomial ODE descriptive mathematics

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



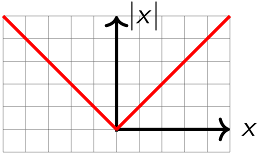
- However $|x|$ is "   close" to a generable function:



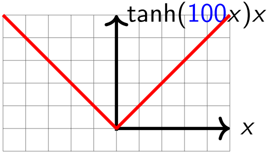
first projection of $y' = (y_4(1 - y_2^2)y_3 + y_2, y_4(1 - y_2^2), 1, 0)$, $y(0) = (0, 0, 0, 2)$.

Polynomial ODE descriptive mathematics

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



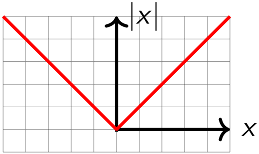
- However $|x|$ is "   close" to a generable function:



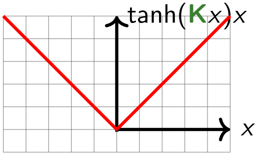
first projection of $y' = (y_4(1 - y_2^2)y_3 + y_2, y_4(1 - y_2^2), 1, 0)$,
$y(0) = (0, 0, 0, 100)$.

Polynomial ODE descriptive mathematics

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



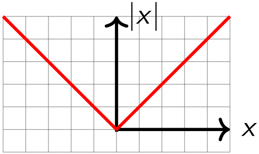
- However $|x|$ is "**uniformly** close" to a generable function:



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

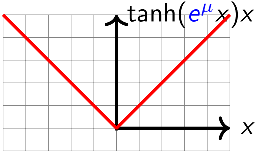
Polynomial ODE descriptive mathematics

- 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:
  - Given μ, we need to feed e^{μ} to the initial condition.
  - Can we avoid this "strange"/"unnatural" dependance in the initial condition?
  - Yes, if we don't ask for **real time** computation!

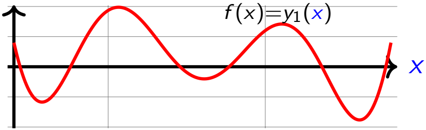
Alternative statement

- $|x|$ is "**uniformly** close" to a generable function:
  - Given μ, we need to feed e^{μ} to the initial condition.
  - Can we avoid this "strange"/"unnatural" dependance in the initial condition?
  - Yes, if we don't ask for **real time** computation!

Replace **real-time** concept:

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

  with precision $e^{-\mu}$

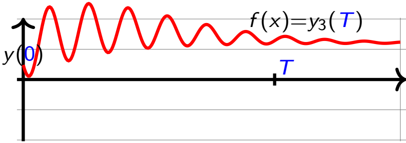


$$y(0) = F(0, \mu)$$
$$f(x) = y_1(x)$$

By a **more modern concept**:

- $f(x)$ must be produced **at some time** $T = T(x)$

  with precision $e^{-\mu}$



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

This is a more general notion of computability

- Key fact: Any generated function is computable in that sense.
- Illustration for $|x|$

This is a more general notion of computability

- Key fact: Any generated function is computable in that sense.
- Illustration for $|x|$
  - Discrete time Computer Science reasoning: Given μ,
    1. Compute $e^{-\mu}$
    2. Then compute $abs(x, \mu) = \tanh(e^{-\mu}x)x$

i.e. previous function
starting from $(0, 0, 0, e^{\mu})$

This is a more general notion of computability

- Key fact: Any generated function is computable in that sense.
- Illustration for $|x|$
  - Discrete time Computer Science reasoning: Given μ,
    1. Compute $e^{-\mu}$
    2. Then compute $abs(x,\mu) = \tanh(e^{-\mu}x)x$

i.e. previous function
starting from $(0,0,0,e^{\mu})$

  - But we are in a continuous time world:

This is a more general notion of computability

- Key fact: Any generated function is computable in that sense.
- Illustration for $|x|$
  - Discrete time Computer Science reasoning: Given μ,
    1. Compute $e^{-\mu}$
    2. Then compute $abs(x, \mu) = \tanh(e^{-\mu}x)x$

i.e. previous function
starting from $(0, 0, 0, e^{\mu})$

  - But we are in a continuous time world:

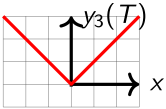
both steps can be done simultaneously !

This is a more general notion of computability

- Illustration for $|x|$ continued:
  - **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

- Illustration for $|x|$ continued:
  - **Simple idea**: consider **a path** $y(t)$ going from $y(0) = (x, \mu, \ldots)$ to $y(T) = (x, \mu, abs(x, \mu), \ldots)$

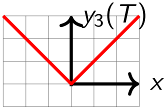
    where $abs(x, \mu) = \tanh(e^{-\mu} x) x$ is previous function.

    - For example, for $T = 1$,

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

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

- Illustration for $|x|$ continued:
  - **Simple idea**: consider **a path** $y(t)$ going from $y(0) = (x, \mu, \dots)$ to $y(T) = (x, \mu, abs(x, \mu), \dots)$

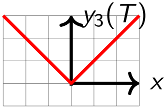
    where $abs(x, \mu) = \tanh(e^{-\mu}x)x$ is previous function.

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

  - **Graphically:**

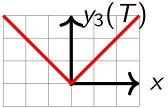


with $|x| - e^{-\mu} \leqslant y_3(T) \leqslant |x| + e^{-\mu}, \quad x = y_1(0), \mu = y_2(0)$

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

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

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



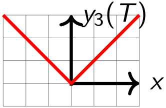
- Other paths could be used.
  E.g. if one wants better and better precision, or that this works even for $t \geq 1$.

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

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

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

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



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

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

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

Main Statement: Computability

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

[3] OB, D. Graça, A. Pouly Journal of the ACM [**?**]'s Improvement of OB, M. Campagnolo, D. Graça, E. Hainry Journal of Complexity [**?**]

Main Statement: Computability

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

[3] OB, D. Graça, A. Pouly Journal of the ACM [**?**]'s Improvement of OB, M. Campagnolo, D. Graça, E. Hainry Journal of Complexity [**?**]

Main Statement: Computability

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

- **Theorem**[3] Every computable function can be computed in that sense, and conversely.

[3] OB, D. Graça, A. Pouly Journal of the ACM [**?**]'s Improvement of OB, M. Campagnolo, D. Graça, E. Hainry Journal of Complexity [**?**]

Main Statement: Computability

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

- **Theorem**[3] Every computable function can be computed in that sense, and conversely.

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

[3] OB, D. Graça, A. Pouly Journal of the ACM [**?**]'s Improvement of OB, M. Campagnolo, D. Graça, E. Hainry Journal of Complexity [**?**]

Main Statement: Computability

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

- **Theorem**[3] Every 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.

[3] OB, D. Graça, A. Pouly Journal of the ACM [**?**]'s Improvement of OB, M. Campagnolo, D. Graça, E. Hainry Journal of Complexity [**?**]

Formal Theorem [4]

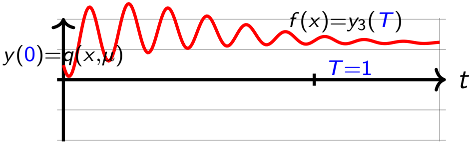
Let $a, b \in \mathbb{Q}$.

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

iff

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

- Picture:



[4] OB, D. Graça, A. Pouly Journal of the ACM [?]'s Improvement of OB, M. Campagnolo, D. Graça, E. Hainry Journal of Complexity [?]

Formal Theorem [4]

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

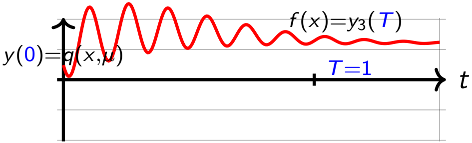
- $f \in C^0([a,b],\mathbb{R})$ is computable

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:

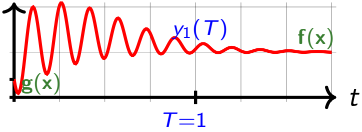


[4] OB, D. Graça, A. Pouly Journal of the ACM [?]'s Improvement of OB, M. Campagnolo, D. Graça, E. Hainry Journal of Complexity [?]

Menu

- Variable t is rather arbitrary.

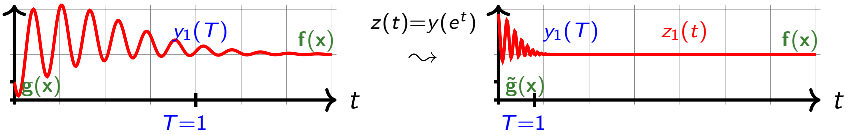
$$y(0) = g(x) \qquad y' = h(y)$$



Time complexity for continuous systems

- Variable t is rather arbitrary.

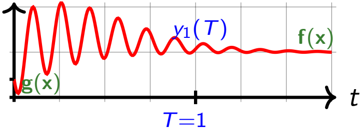
$y(0) = g(x) \qquad y' = h(y)$ $\qquad\qquad$ $z(0) = \tilde{g}(x) \qquad z' = \tilde{h}(z)$



Time complexity for continuous systems

- Variable t is rather arbitrary.

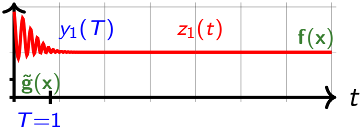
$y(0) = g(x) \qquad y' = h(y)$



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

$\rightsquigarrow$

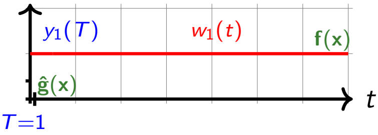
$z(0) = \tilde{g}(x) \qquad z' = \tilde{h}(z)$



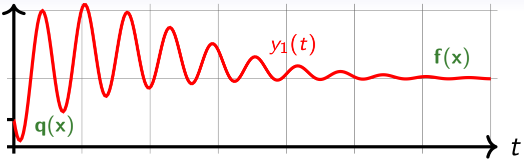
$w(0) = \hat{g}(x) \qquad w' = \hat{h}(w)$

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

$\rightsquigarrow$



A Simple & Key Idea: curvilinear 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 parametrization

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

I.e.:
Follow curve **at constant speed.**

Main Statement: Complexity

- **Theorem**[5] Any polynomial time computable function can be computed in polynomial length, and conversely.

[5] OB, D. Graça, A. Pouly ICALP Track B Best Paper Award [**?**], Journal of the ACM [**?**]

Main Statement: Complexity

- **Theorem**[5] 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 !!**

[5] OB, D. Graça, A. Pouly ICALP Track B Best Paper Award [**?**], Journal of the ACM [**?**]

Main Statement: Complexity

- **Theorem**[5] 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.

[5] OB, D. Graça, A. Pouly ICALP Track B Best Paper Award [**?**], Journal of the ACM [**?**]

Formal Theorem

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

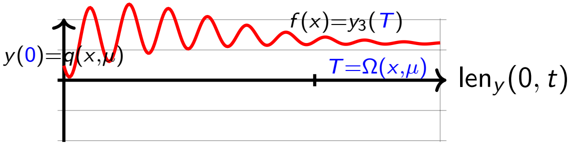
- $f \in C^0([a,b],\mathbb{R})$ is polynomial-time computable

iff

- ▶ y satisfies a pODE

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

- Picture:



Formal Theorem

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

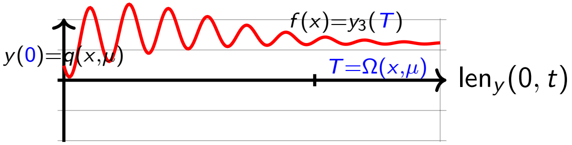
- $f \in C^0([a,b],\mathbb{R})$ is polynomial-time computable

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:



For Discrete People

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

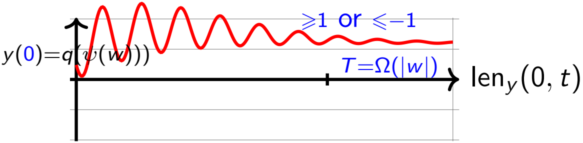
iff

▶ y satisfies a pODE

▶ decision is made after a polynomial length

▶ and corresponds to $\mathcal{L}$

- Picture:



For Discrete People

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

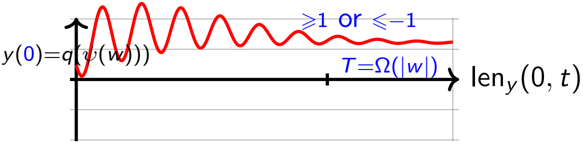
- For example $\psi(w) = \left(\sum_{i=1}^{|w|} w_i k^{-i}, |w|\right)$, and $\geqslant 1$, $\leqslant -1$.

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

Sub-menu

There exists a Universal ODE

- **Theorem**[6]
  There exists a fixed vector of polynomial p such that for
  1. any continuous $f : \mathbb{R} \to \mathbb{R}$,
  2. and continous $\epsilon : \mathbb{R} \to \mathbb{R}^+$

  there exists some $\alpha \in \mathbb{R}^e$ such that

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

  has a **unique solution** $y : \mathbb{R} \to \mathbb{R}^d$ such that

  $$|y_1(t) - f(t)| \leq \epsilon(t)$$

  for all t.

[6] OB, A. Pouly ICALP [**?**]

Sub-menu

Some applications

A Universal ODE

Biochemical Reaction Networks

More speculative applications

- **Main Theorem**[7]

  - The systems of elementary biochemical reactions on finite universes of molecules are (strong) Turing-complete in differential semantics.

- **Considered systems:** at most binary reactions with mass action law kinetics
  1. $A + B \xrightarrow{k \cdot A \cdot B} C$
  2. $A \xrightarrow{k \cdot A} B + C$
  3. $A \xrightarrow{k \cdot A} B$
  4. $\_ \xrightarrow{k} A$
  5. $A \xrightarrow{k \cdot A} \_$

[7] François Fages, Guillaume Le Guludec, OB, Amaury Pouly CMSB Best Paper Award 2017 [**?**]

Sub-menu

Some applications

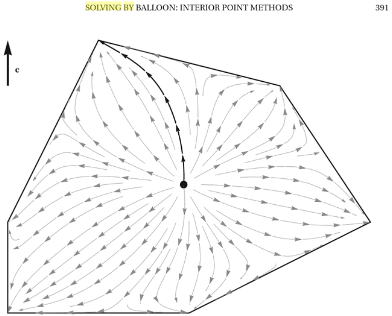
A Universal ODE

Biochemical Reaction Networks

More speculative applications

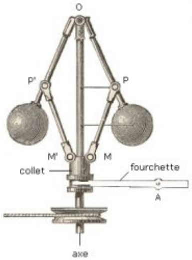
- Finding zeros of a function: $x' = -f(x)$

- Linear Programming:

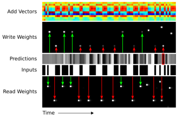


See e.g.: The Nature of Computation, C. Moore and S. Mertens, Oxford University Press.

- Computing optimal solutions:

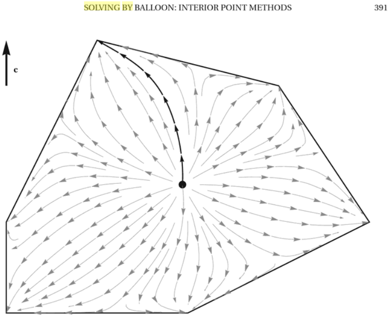


- Neural Networks, Deep learning, Differential Neural Computers, Neural Turing Machines, and variants...



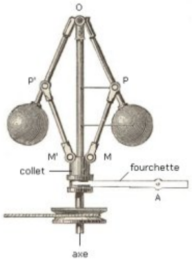
- Finding zeros of a function: $x' = -f(x)$

- Linear Programming:

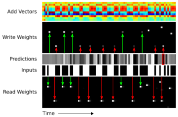


See e.g.: The Nature of Computation, C. Moore and S. Mertens, Oxford University Press.

- Computing optimal solutions:



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



- And Turing machines.

Menu

Back to Foundations of Computer Science

Descriptive Mathematics

Descriptive Computer/Computability Science

Descriptive Computer/Complexity Science

Some applications

The subject of this course

Sub-menu

Consider you preferred function $f : \mathbb{R} \to \mathbb{R}$.

Consider you preferred function $f : \mathbb{R} \to \mathbb{R}$.

Is f computable?

Consider you preferred function $f : \mathbb{R} \to \mathbb{R}$.

Is f computable?

Several notions of computability for real functions

Consider you preferred function $f : \mathbb{R} \to \mathbb{R}$.

Is f computable?

Several notions of computability for real functions

- Turing machine approach: Recursive Analysis.

Consider you preferred function $f : \mathbb{R} \to \mathbb{R}$.

Is f computable?

Several notions of computability for real functions

- Turing machine approach: Recursive Analysis.
- Continuous time analog models

Consider you preferred function $f : \mathbb{R} \to \mathbb{R}$.

Is f computable?

Several notions of computability for real functions

- Turing machine approach: Recursive Analysis.
- Continuous time analog models
- Blum Shub Smale machines

Consider you preferred function $f : \mathbb{R} \to \mathbb{R}$.

Is f computable?

Several notions of computability for real functions

- Turing machine approach: Recursive Analysis.
- Continuous time analog models
- Blum Shub Smale machines
- ...

Consider you preferred function $f : \mathbb{R} \to \mathbb{R}$.

Is f computable?

Several notions of computability for real functions

with various motivations:

Consider you preferred function $f : \mathbb{R} \to \mathbb{R}$.

Is f computable?

Several notions of computability for real functions

with various motivations:

- computability theory

Consider you preferred function $f : \mathbb{R} \to \mathbb{R}$.

Is f computable?

Several notions of computability for real functions

with various motivations:

- computability theory
- lower bounds / upper bounds

Consider you preferred function $f : \mathbb{R} \to \mathbb{R}$.

Is f computable?

Several notions of computability for real functions

with various motivations:

- computability theory
- lower bounds / upper bounds
- verification
- control theory

Consider you preferred function $f : \mathbb{R} \to \mathbb{R}$.

Is f computable?

Several notions of computability for real functions

with various motivations:

- computability theory
- lower bounds / upper bounds
- verification
- control theory
- ...

Sub-menu

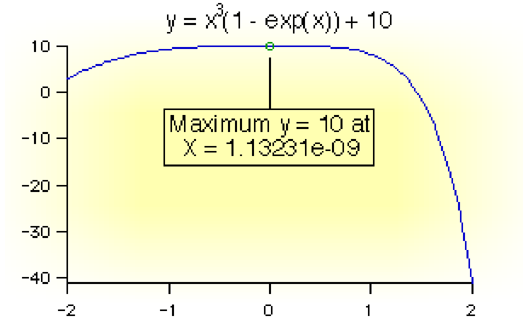
Motivation 1: Models of Computation

NACA Lewis Flight Propulsion Laboratory's Differential Analyser

Question: What is the computational power of this machine?

Sub-menu

Motivation 2: Effectivity in Analysis



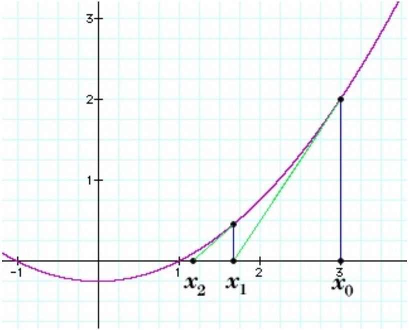
Question: Can we compute the maximum of a continuous function over a compact domain? A point on which it is maximal?

Sub-menu

The subject of this course

- THE question
- Motivation 1: Models of Computation
- Motivation 2: Effectivity in Analysis
- Motivation 3: Algebraic Complexity
- Motivation 4: Verification/Control

Motivation 3: Algebraic Complexity



Question: What is the complexity of Newton's method?

Sub-menu

The subject of this course

- THE question
- Motivation 1: Models of Computation
- Motivation 2: Effectivity in Analysis
- Motivation 3: Algebraic Complexity
- Motivation 4: Verification/Control

Motivation 4: Verification/Control

- Model $\mathcal{M}$ made of a mixture of continuous/discrete parts.
- Specification ϕ (e.g. reachability property).

Informal question: Can we avoid that?

Formal question:

$$\mathcal{M} \models \phi?$$