Continuous robust simulation of Turing machines on low-dimensional dynamical systems

Daniel Graça[1,2]

[1]FCT, Universidade do Algarve, Portugal
[2]SQIG, Instituto de Telecomunicações, Portugal

Toulouse
31 March 2023
Journées annuelles SDA2

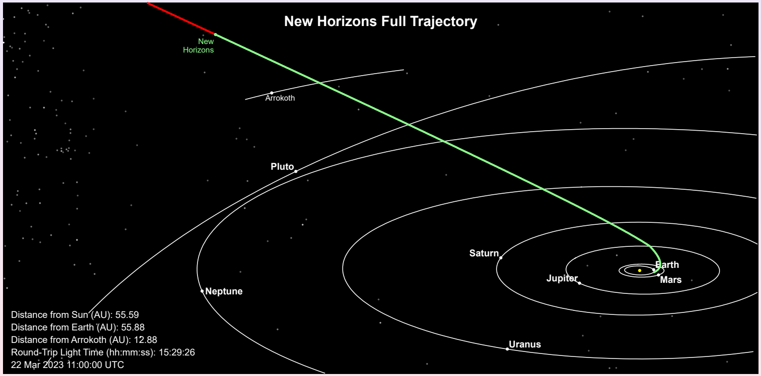
Motivation

An intellect which at a certain moment would know all forces that set nature in motion, and all positions of all items of which nature is composed, if this intellect were also vast enough to submit these data to analysis, it would embrace in a single formula the movements of the greatest bodies of the universe and those of the tiniest atom; for such an intellect nothing would be uncertain and the future just like the past would be present before its eyes.

— Pierre Simon Laplace, A Philosophical Essay on Probabilities

Some successes...

We can easily compute the position of a planet or probe years in advance, with an high degree of accuracy


New Horizons Full Trajectory
New Horizons
Arrokoth
Pluto
Neptune
Saturn
Jupiter
Earth
Mars
Uranus
Distance from Sun (AU): 55.59
Distance from Earth (AU): 55.88
Distance from Arrokoth (AU): 12.88
Round-Trip Light Time (hh:mm:ss): 15:29:26
22 Mar 2023 11:00:00 UTC

New horizons trajectory near Pluto

Image credits: NASA/Johns Hopkins University Applied Physics Laboratory/Southwest Research Institute

… and some failures

Yet predicting the position of a small leaf in a turbulent flow after a few minutes is a much tougher challenge!

To make meaningful predictions we need

data + models + **computation**

Our main questions:

1. Given some system, can we tell something about its behavior using Turing machines?
2. Are there some classes of systems which have more computational power than Turing machines?

Should we allow any type of system or should we restrict ourselves to "physically realistic" systems?

There are some well-known non-computability results:

- There are continuous-time dynamical systems which have super-Turing power (e.g PCD systems, Asarin, Maler, Bournez)
- There is an ordinary differential equation (ODE) defined with computable data which has no computable solution (Aberth, Pourl-El, Richards)
- There is a computable and C^1 initial wave with zero initial velocity such that the unique solution of the problem - the amplitude of the wave - is not computable at time $t = 1$ (Pour-El, Richards, Zhong, 1981, 1997)
- There are electronic circuits which can be shown, theoretically and provably, to have non-computable outputs and properties from computable inputs (Boche and Pohl, 2020, 2021)

But how physically realistic are those results?

What "physically realistic" means?

- Super-Turing power is not robust to "noise" (small perturbations)
- Solutions for ODEs become computable if the solution is unique
- For example, the non-computability of the wave equation is completely gone if the L^∞-norm is replaced by a more physically realistic energy norm (Weihrauch and Zhong, 1999, 2000)
- Etc.

To consider "realistic" phenomena we may use:

- Robustness to "noise"
- Usage of "sufficiently smooth" dynamics
- Usage of a suitable norm
- ...

Conjecture (Physical Church-Turing thesis)

No physically realistic device operating accordingly to the (macroscopic) physical laws will have more computational power than a Turing machine (possible exception for computational complexity?: quantum computers)

- Proving this conjecture is quite challenging due to the existence of innumerable devices that one can imagine which operate according to the known physical laws
- A different idea is to use a "divide and conquer" strategy: instead of considering every imaginable physically realistic system operating accordingly to the (macroscopic) physical laws, restrict to (large enough) classes of realistic systems and prove the conjecture for each of these classes

Our proposal

Here we consider systems which have at least one of the following properties:

- Their description uses **analytic functions** (highest degree of "smoothness"), ideally closed-form functions — the usual functions of Analysis $\Rightarrow$ ($\sin$, $\log$, , x^2, etc.) since most classical laws of physics use such functions
- Are "**robust to perturbations**"

Simulating Turing machines

To study the computational power of a class of systems and, in particular, to show that it is as powerful as a Turing machine, it is important to show that such systems can simulate Turing machines

What do we need to simulate a Turing machine?

- **Encode** a configuration as a state in the system
- To be able to simulate the transition function one needs:
  - To be able to **read** the current state and the symbol being read by the head in a given (coding of a) configuration
  - To **decide** what is the next move to make based on the previous information
  - To **update** the (coding of the) next configuration, where the state, tape contents and tape head position are updated

Situation circa 2005 – simulation of Turing machines

	analytic	robust
bounded domain	does not exist (conjecture, Moore 1998)	negative results (e.g. Maass, Orponen, 1998)
unbounded domain	positive result (Koiran, Moore, 1999)	unknown

- In 1999, Koiran and Moore proved that closed-form analytic maps can simulate Turing machines in real time

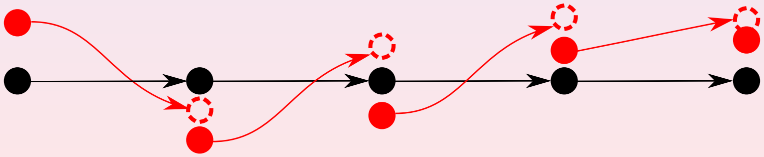
Current situation – simulation of Turing machines

	analytic	robust
bounded domain	positive result (Cardona, Miranda, Peralta-Salas, 2021)	negative results (e.g. Maass, Orponen, 1998)
unbounded domain	positive result (Koiran, Moore, 1999)	positive result (Buescu, Campagnolo, G., 2008)

Theorem (Buescu, Campagnolo, G., 2008)

Given a Turing machine M, there is an analytic closed-form map $f : \mathbb{R}^3 \to \mathbb{R}^3$ which simulates it robustly to perturbations:

- *Each configuration is encoded as an element of $\mathbb{N}^3$*
- *$f|_{\mathbb{N}}$ simulates the transition function of f*
- *We can replace any configuration $c \in \mathbb{N}^3$ by an approximation $\bar{c} \in \mathbb{N}^3$ satisfying $\|c - \bar{c}\| \leq 1/4$*
- **In addition** *of the last point, we can also replace f by some $g : \mathbb{R}^3 \to \mathbb{R}^3$ satisfying $\|f - g\| < 1/4$.*

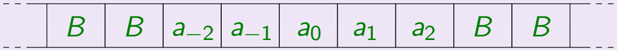


Idea of the proof

- **First step**: given some Turing machine M, find an elementary map f_M whose iteration gives the evolution of M over integers
- **Second step**: Using real-valued error-contracting functions around integers, one can make this simulation robust to perturbations in both the map f_M and in any intermediate configurations

1 – Coding of configurations

Suppose that M is a TM using 10 symbols, with tape contents



Associate to each symbol a number in $\{0, 1, ..., 9\}$ (take $B \to 0$). Then the tape contents can be coded by

$$\text{right} = a_0 + a_1 10 + ... + a_n 10^n$$
$$\text{left} = a_{-1} + a_{-2} 10 + ... + a_{-k} 10^{k-1}$$

If we associate each state q to a number in $\{1, ..., m\}$, then each configuration of M is given by a triple $(\text{left}, q, \text{right}) \in \mathbb{N}^3$

2 – Reading data from configurations

Given a configuration $c = (\text{left}, q, \text{right}) \in \mathbb{N}^3$:

- One can immediately obtain the state q as the last component of c
- To obtain the current symbol $s \in \{0, 1, ..., 9\}$ being read by the head, we note that $\text{right} = s + a_1 10 + ... + a_n 10^n$
- Hence we would like to have some function S such that $S(a_0 + a_1 10 + ... + a_n 10^n) = a_0$
  - In particular $S(0) = 0, S(1) = 1, \ldots, S(9) = 9$
  - S has period 10 over the integers: $S(n) = S(n + 10)$ for any $n \in \mathbb{N}$

▶ **Solution:** Define S using trigonometric interpolation

3 – Determining what to do next

- Given the state q and symbol s being read by the head, the next step will only depend on finitely many values (q, s)
- Define a function $N : \mathbb{R}^2 \to \mathbb{R}$ such that each pair (q, s) corresponds to a "next" value $N_{(q,s)}$.
- ▶ **Solution:** Define N using Lagrange polynomial interpolation
- Determine the next state $q_{next}(q, s)$:

$$q_{next}(q,s) = \sum_{i=0}^{9} \sum_{j=1}^{m} \left(\prod_{\substack{r=0 \\ r \neq i}}^{9} \frac{(s-r)}{(i-r)} \right) \left(\prod_{\substack{s=1 \\ s \neq j}}^{m} \frac{(q-s)}{(j-s)} \right) q_{i,j}$$

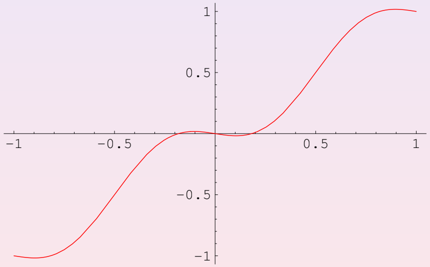
- Determine the next symbol $s_{next}(q, s)$ to be written
- Determine the next move $m_{next}(q, s)$ to be made by the tape head.

4 – Update tape contents

- Some calculations needed to be obtain the right expressions, but nothing too fancy

5 – robustness to perturbations

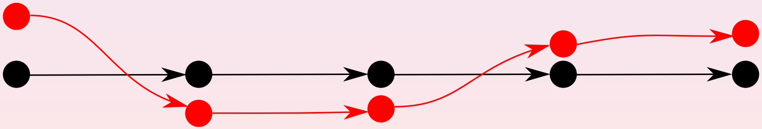
- Use error-contracting functions to keep any error under control
- Example of an *error-contracting function* which we used: $\sigma(x) = x - 0.2\sin(2\pi x)$ contracts the error in the vicinity of integers



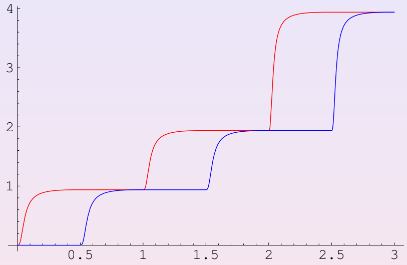
Theorem (Buescu, Campagnolo, G., 2008)

Given a Turing machine M, there is an analytic closed-form ODE $y' = f(t, y)$, with $f : \mathbb{R}^7 \to \mathbb{R}^6$ which simulates it robustly to perturbations:

- *Each configuration is encoded as an element of* $\mathbb{N}^3$
- *The input of* M *is encoded in the initial configuration*
- $y_{4,5,6}(t)$ *codes the configuration at step* $n \in \mathbb{N}$ *if* $t \in [n, n + 1/2]$
- *The initial condition and* f *can be both perturbed and yet the simulation will succeed*



Idea of the proof



Iteration of the map $f(x) = 2^x$, with $x_0 = 0$

Problem: This simulation only works for *non-analytic functions*

But by allowing non-exact values (recall f_M is robust to perturbations), we can overcome the restriction (the proof has lots of technical constructions).

Lowest dimension?

- What is the lowest dimension one can use in the previous analytic and robust simulations of Turing machines?

Theorem (G., Zhong, to appear)

- *For a map the lowest dimension is* $n = 1$
- *For ODEs, dimension* $n = 2$ *suffices*
    - *Under reasonable assumptions, it must be* $n > 1$

Idea of the proof - map/unbounded case

- It is well known that there is a bijection $I_3 : \mathbb{N}^3 \to \mathbb{N}$
- It can be built from

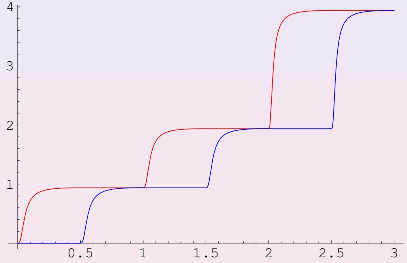
$$I_2(x, y) = \frac{(x+y)^2 + 3x + y}{2}$$

- This map I_2 (and thus I_3) can be readily extended to the reals using error-contracting functions in order to be robust to perturbations
- **Main idea:** If $\bar{f}_M : \mathbb{R}^3 \to \mathbb{R}^3$ simulates the Turing machine, then take $f_M : \mathbb{R} \to \mathbb{R}$ as $f_M = I_3 \circ \bar{f}_M \circ I_3^{-1}$
- **Problem:** The inverse of I_2 (and of I_3) cannot be computed by a formula, only by an algorithm

- **Solution:** Use the previous simulation of TM with (3D) ODEs but retain only the component which gives the part of the tape with the result
- Some pernicious problems: given some $n \in \mathbb{N}$, the (integer) encoding of n is not n when simulating a TM, the same happening to the output.
- Solvable with several tricks, involving the use of unary functions, etc.

Idea of the proof - ODE/unbounded case

- In this case the proof is very simple: just simulate the iteration of the previous map with an ODE, as shown earlier.



Iteration of the map $f(x) = 2^x$, with $x_0 = 0$

Lowest dimension for compact case?

Theorem (Cardona, Miranda, Peralta-Salas, 2021)

There is a polynomial ODE which is Turing universal on the sphere $\mathbb{S}^n$, where $n \geq 17$.

Theorem (Cardona, Miranda, Peralta-Salas, Presas, 2021)

There is a fluid-flow (uses partial differential equations) which is Turing universal on a Riemannian 3-dimensional sphere.

- Our result:

Theorem (G., Zhong, to appear)

There is a C^∞ ODE which is Turing universal on the sphere $\mathbb{S}^2$.

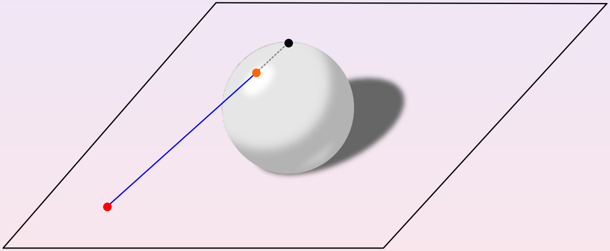
Idea of the proof – compact case

Theorem (Cardona, Miranda, Peralta-Salas, 2021)

There is a polynomial ODE which is Turing universal on the sphere $\mathbb{S}^n$, where $n \geq 17$.

- **Proof idea:**
  - The previous simulation of Turing machines with a map $\bar{f}_M : \mathbb{R}^3 \to \mathbb{R}^3$ can be done with a polynomial ODE with an higher dimension ($n = 16$) (result from (Buescu, Campagnolo, G., 2008))
  - Use one extra variable x_{17} to model time in $x' = p(t, x)$, by taking $x_{17}' = 1$, $x_{17}(0) = 0$
  - Thus there is an autonomous polynomial ODE $x' = p(t, x)$ with dimension $n = 17$ which simulates Turing machines

- Map the polynomial flow to the sphere $\mathbb{S}^{17}$ via the stereographic map
- The resulting flow is polynomial and is defined at the north pole (it is a zero of the flow)



Theorem (G., Zhong, to appear)

There is a C^∞ ODE which is Turing universal on the sphere $\mathbb{S}^2$.

- We use our previous construction that defines an analytic 2D ODE to simulate ODEs
- However, we have to limit the growth of solutions to polynomial growth (tweak the ODE)
- Apply the stereographic map as Cardona et al. to map the flow to $\mathbb{S}^2$
- The flow in $\mathbb{S}^2$ is only C^∞ due to the north pole (the trick of Cardona et al. to obtain analyticity is only valid for polynomials)

ODEs- What about the case of dimension 1?

Some assumptions:

- Each configuration c_M of the Turing machine M is encoded as a set $\chi(c_M)$
- $c_M \neq c_{M'} \Rightarrow \chi(c_M) \cap \chi(c_{M'}) = \varnothing$: **not enough** – case of intermingled sets

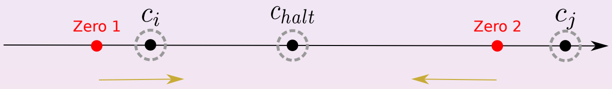
- We assume that there are two computable maps $a : \mathbb{N} \to \mathbb{Q}^k$, $r : \mathbb{N} \to \mathbb{Q}$ such that:
  - For all $i \in \mathbb{N}$, $\chi(c_i) \subseteq \overline{B(a(i), r(i))} = \{x \in \mathbb{R}^k : \|x - a(i)\| \leq r(i)\}$
  - If $i \neq j$ then $\overline{B(a(i), r(i))} \cap \overline{B(a(j), r(j))} = \varnothing$
- The ODE $y' = f(y)$ simulates the TM M if $y(n) \in \chi(\psi^{[n]}(c_0))$ for all $n \in \mathbb{N}$
- Assume that f is computable and only has isolated zeros

Theorem (G., Zhong, to appear)

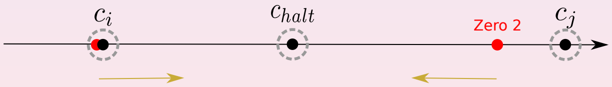
No one-dimensional ODE can simulate universal Turing machines in the above conditions.

Idea of the proof

- Isolated zeros of a computable function are computable
- **Case 1:**

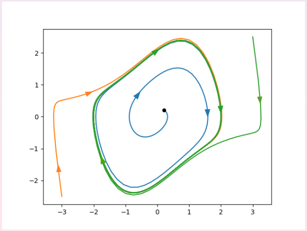


- **Case 2:**



Can non-computability be robust?

- Traditional non-computability results for continuous systems are "fragile" (wave equation, electronic circuits, solution of ODE, etc.)
- If we see digital computers as having analog components (and thus as analog computers) shouldn't the associated Halting Problem be robust to "noise"?
- Notion of **basin of attraction** for an **attractor**



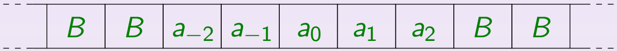
Theorem (G., Zhong, work in progress)

The is an analytic ODE $y' = f(y)$ with the following properties:

- *It has a stable attracting equilibrium point (a sink) x_0 which is unique in a neighborhood $\mathcal{U}$*
- *The basin of attraction of x_0 is not computable*
- *Any "closeby" ODE $y' = g(y)$ ($\|f - g\| \leq \varepsilon$) has a unique sink in $\mathcal{U}$ which has similar properties*

Main problem: Ensuring that the halting configuration corresponds to a unique point which is a sink.

Other useful simulations of Turing machines



- Encoding of inputs $\Psi_k(w) = \left(\sum_{i=1}^{|w|} \gamma(w_i)k^{-i}, |w|\right)$
- $\langle c \rangle = (0.x, a_0, y, 0.q)$ is (the coding of) the configuration c, where $0.x = x_1 k^{-1} + x_2 k^{-2} + \ldots + x_{|x|} k^{|x|}$ and similarly for $0.y$
- We can simulate (time-bounded) Turing machines with this coding using polynomial ODEs $y' = p(y)$

Characterizing complexity classes with ODEs

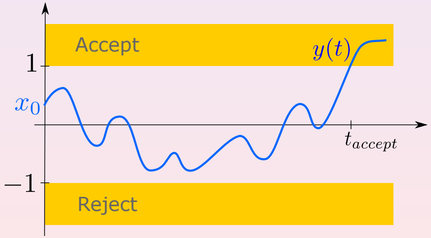
But what if we use continuous models of computation?

- Can we define meaningful measures of complexity for continuous models?
- Can we obtain complexity classes which relate to the "traditional" ones ($\mathsf{P}, \mathsf{NP}, \mathsf{PSPACE}, \mathsf{EXPTIME}, \ldots$)?

Characterizing complexity classes with ODEs

But what if we use continuous models of computation?

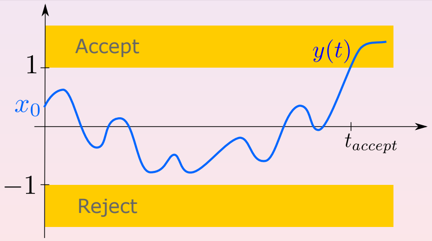
- Can we define meaningful measures of complexity for continuous models?
- Can we obtain complexity classes which relate to the "traditional" ones (P, NP, PSPACE, EXPTIME, ...)?

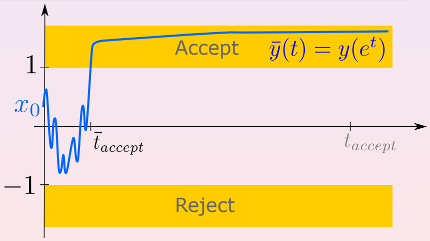


Characterizing complexity classes with ODEs

But what if we use continuous models of computation?

- Can we define meaningful measures of complexity for continuous models?
- Can we obtain complexity classes which relate to the "traditional" ones (P, NP, PSPACE, EXPTIME, . . .)?

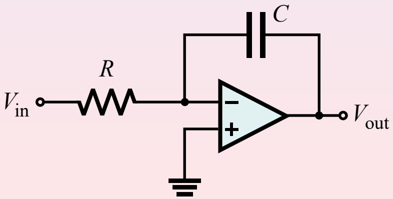




- Defining an adequate (invariant) notion of continuous “time complexity” is challenging (due to Zeno-like behavior)
- In the rest of this talk we will use as our model of computation dynamical systems which are defined by polynomial (vectorial) ODEs:

$$x' = p(x)$$

  - This model is mathematically simple to describe and has “nice” properties
  - Polynomial ODEs have a realistic model – Shannon’s General Purpose Analog Computer (GPAC), which can be implemented with mechanical devices or using (analog) electronics



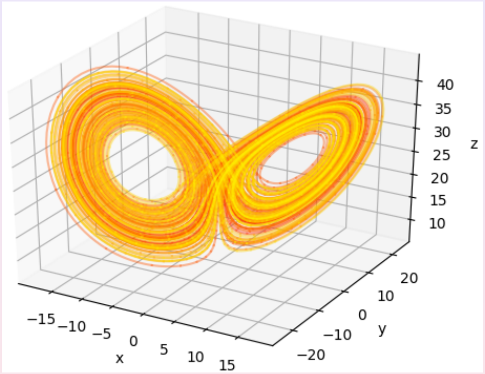
Theorem (Buescu, Campagnolo, G., 2008)

Polynomial ODEs can simulate universal Turing machines.

Theorem (Bournez, Campagnolo, G., Hainry, 2007)

Polynomial ODEs and Turing machines (computable analysis) are equivalent from a computability perspective.

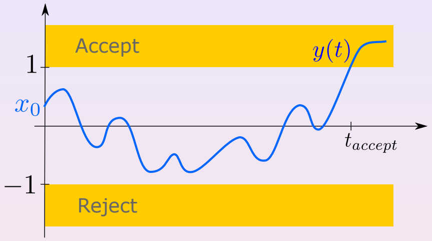
- Defining a notion of “continuous space” is also hard!



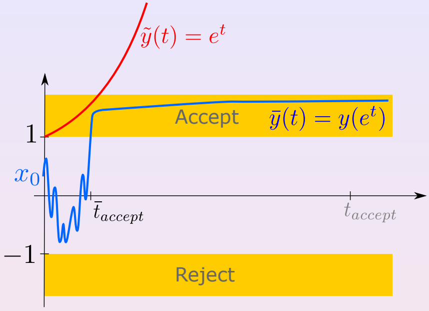
$$\begin{cases} x' = \sigma(y - x) \\ y' = x(\rho - z) - y \\ z' = xy - \beta z \end{cases}$$

Classical values for parameters: $\sigma = 10, \rho = 28, \beta = 8/3$

“Time” complexity for continuous-time systems

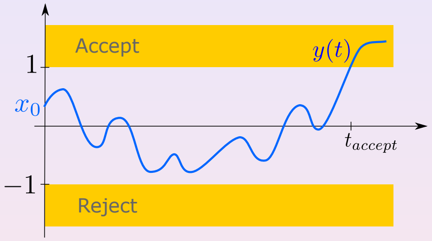


$y' = p(y)$ with $y(0) = q(x)$

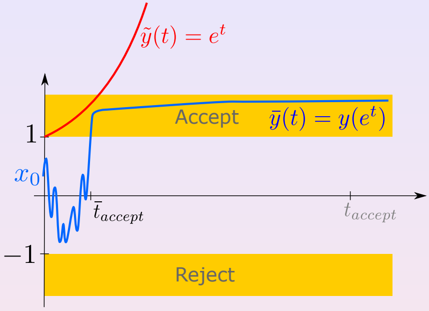


$$\begin{cases} \overline{y}' = \overline{z}p(\overline{y}) \\ \overline{z}' = \overline{z} \end{cases} \quad \begin{cases} \overline{y}(0) = \overline{q}(x) \\ \overline{z}(0) = 1 \end{cases}$$

“Time” complexity for continuous-time systems



$y' = p(y)$ with $y(0) = q(x)$

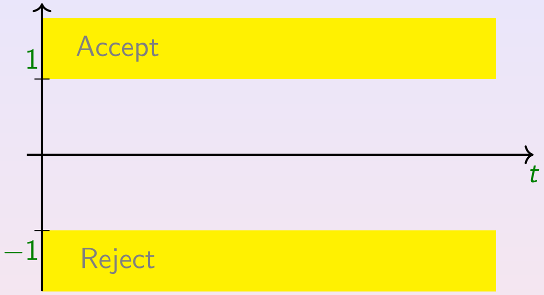


$\begin{cases} \overline{y}' = \overline{z}p(\overline{y}) \\ \overline{z}' = \overline{z} \end{cases}$ $\begin{cases} \overline{y}(0) = \overline{q}(x) \\ \overline{z}(0) = 1 \end{cases}$

Solution: define complexity using the length of the solution curve:

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

A continuous characterization of P and EXPTIME



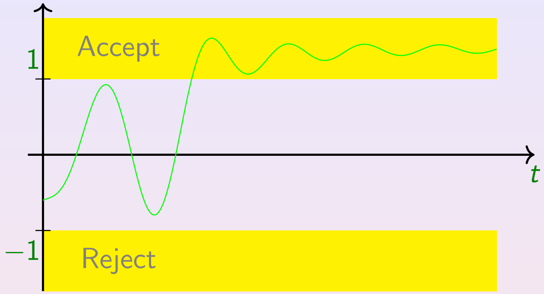
Theorem (Bournez, G., Pouly, 2017)

$L \in P$ iff L is poly-length recognizable.

Theorem (Gozzi and G., 2023)

$L \in EXPTIME$ iff L is exp-length recognizable.

A continuous characterization of P and EXPTIME



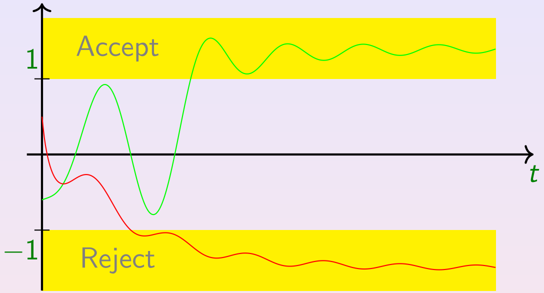
Theorem (Bournez, G., Pouly, 2017)

$L \in P$ iff L is poly-length recognizable.

Theorem (Gozzi and G., 2023)

$L \in EXPTIME$ iff L is exp-length recognizable.

A continuous characterization of P and EXPTIME



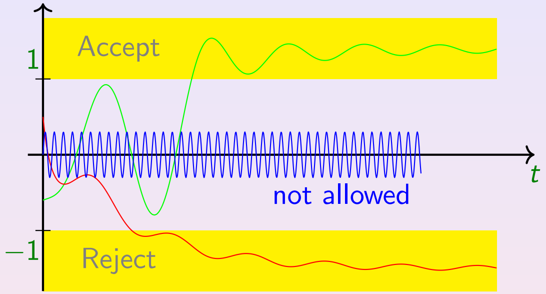
Theorem (Bournez, G., Pouly, 2017)

$L \in P$ iff L is poly-length recognizable.

Theorem (Gozzi and G., 2023)

$L \in EXPTIME$ iff L is exp-length recognizable.

A continuous characterization of P and EXPTIME



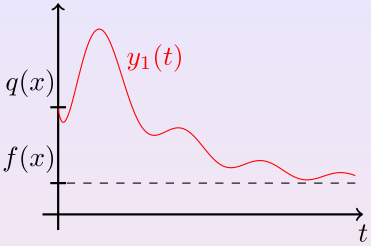
Theorem (Bournez, G., Pouly, 2017)

$L \in P$ iff L is poly-length recognizable.

Theorem (Gozzi and G., 2023)

$L \in EXPTIME$ iff L is exp-length recognizable.

Computing functions



$$\begin{tikzcd} \mathbb{R}^k \arrow[r, "F"] \arrow[d, "\psi"] & \mathbb{R}^k \arrow[d, "\psi"] \\ \Gamma^* \arrow[r, "f"] & \Gamma^* \end{tikzcd}$$

Polynomial complexity for continuous-time systems

$f : \mathbb{R} \to \mathbb{R}$ is poly-length computable if there is a polynomial Π such that $\forall \mu > 0$, if $len_y(0,t) \geq \Pi(\|x\|, \mu)$, then $|y_1(t) - f(x)| \leq 2^{-\mu}$ (with $\|y'(t)\| \geq 1$).

Theorem (Bournez, G., Pouly, 2017)

A (discrete) function f belongs to FP iff it is emulable by some poly-length computable function.

- The previous construction uses properties that polynomials have but e.g. exponential functions don't (e.g. closure under composition)
- A careful argument allows generalization of this argument

Exponential complexity for continuous-time systems

$f : \mathbb{R} \to \mathbb{R}$ is exp-length computable if there is an exponential function Π_1 and a polynomial Υ_1 such that $\forall \mu > 0$, if

$$len_y(0, t) \geq \Pi(\|x\|, \mu) = \Pi_1(\|x\|)\Upsilon_1(\mu)$$

then $|y_1(t) - f(x)| \leq 2^{-\mu}$ (with $\|y'(t)\| \geq 1$).

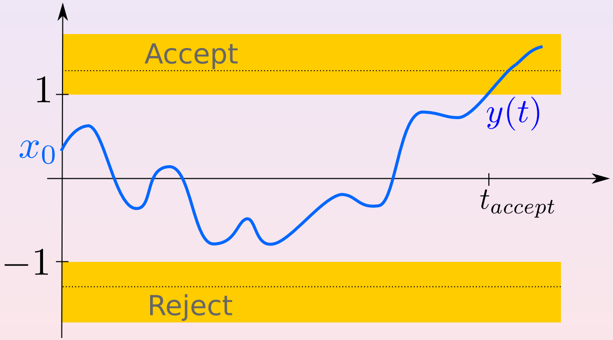
Theorem (Gozzi and G.)

A (discrete) function f belongs to FEXPTIME iff it is emulable by some exp-length computable function.

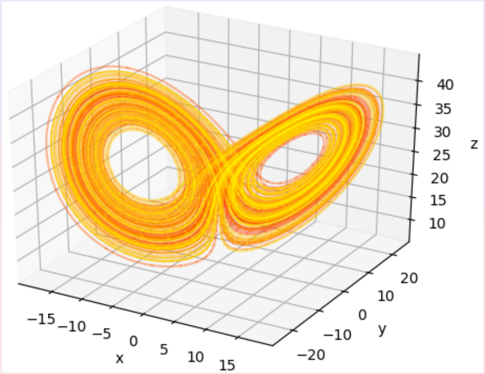
- The above procedure can be generalized to other classical classes:
  - All levels of the Grzegorczyk hierarchy
  - Primitive recursive functions
- The main problem is to obtain appropriate complexity bounds, which we also assume to be solutions of a polynomial ODE – polynomials and exponentials admit trivial extensions to the reals, while this is not the case for the Grzegorczyk hierarchy

Characterizing PSPACE

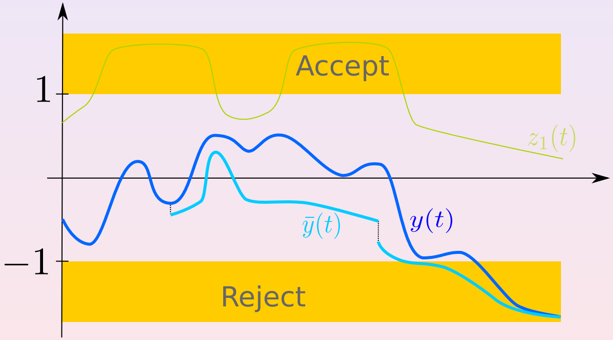
1. Halting decision is irreversible
2. Halting decision is eventually taken with correct output



3. Bounded space: $\|(y(t), z(t))\| \leq \phi \circ g(x)$ for any x satisfying $\Psi(x) = w$ and for all $t \geq 0$;



4. Robustness to perturbations
5. Robustness is common
6. Robustness preserves main properties



Characterizing PSPACE with polynomial ODEs

Theorem (Bournez, Gozzi, G., Pouly, to appear)

A language $L \subseteq \Gamma^*$ belongs to PSPACE iff it is $\hat{\Psi}$-decided in polynomial space.

Thank you!

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 731143