Computability and complexity in continuous dynamical systems

Daniel S. Graça\textsuperscript{1,2}

\textsuperscript{1}FCT, Universidade do Algarve, Portugal
\textsuperscript{2}SQIG, Instituto de Telecomunicações, Portugal

14 September 2015
The big philosophical question

If we have good enough data/models what can we tell/predict about nature?
The big philosophical question

If we have good enough data/models what can we tell/predict about nature?

Questions:

- Can a computer be used to predict properties of some natural phenomena, before we can observe it?
The big philosophical question

If we have good enough data/models what can we tell/predict about nature?

Questions:
- Can a computer be used to predict properties of some natural phenomena, *before we can observe it*?
- Are there devices better suited than digital computers for the above task?
The big philosophical question

If we have good enough data/models what can we tell/predict about nature?

Questions:
- Can a computer be used to predict properties of some natural phenomena, before we can observe it?
- Are there devices better suited than digital computers for the above task?

What can we tell (compute) about the world? Is it sufficient to have high quality data and models? Are computers Laplace’s demon?
Some successes...

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

(New horizons trajectory near Pluto)
... and some failures

Yet predicting the position of a small leaf in a turbulent flow after a few minutes is a much tougher challenge!
What can we tell about natural phenomena using Turing-like computational models? *(computability)*

What can we tell about natural phenomena using Turing-like computational models *and* a reasonable amount of computational resources? *(computational complexity)*

Can we use devices based on natural phenomena to obtain super-Turing power?
A toy example: stock markets

S&P 500 (24/08/2015 - 09/09/2015)

Portugal’s GDP ~ 227.000 M€
Even the pro’s have a hard time...
Why stock markets behave so wildly?

Some explanations:

- The relevant info to determine the price of a stock is not completely known. The price is updated when some new data is known.
- Markets can be irrational.
- The models for determining the fundamental price of a stock may be incorrect.
Why stock markets behave so wildly?

Some explanations:

- The relevant info to determine the price of a stock is not completely known. The price is updated when some new data is known.
- Markets can be irrational.
- The models for determining the fundamental price of a stock may be incorrect.

But what would happen in a perfect world, where everyone is rational, the standard model describes perfectly the valuation of a stock, and we have access to all the data needed to compute the stock price in this model?
Why stock markets behave so wildly?

Some explanations:

- The relevant info to determine the price of a stock is not completely known. The price is updated when some new data is known.
- Markets can be irrational.
- The models for determining the fundamental price of a stock may be incorrect.

But what would happen in a perfect world, where everyone is rational, the standard model describes perfectly the valuation of a stock, and we have access to all the data needed to compute the stock price in this model?

The price of the stock would still be uncomputable! (Graça, 2013)
Why stock markets behave so wildly?

Some explanations:

- The relevant info to determine the price of a stock is not completely known. The price is updated when some new data is known.
- Markets can be irrational.
- The models for determining the fundamental price of a stock may be incorrect.

But what would happen in a perfect world, where everyone is rational, the standard model describes perfectly the valuation of a stock, and we have access to all the data needed to compute the stock price in this model?

The price of the stock would still be uncomputable! (Graça, 2013)

- It seems that the term “complexity theory” is a broad hat that encompasses several layers of complexity, one of them related to computation.
Two main problems

1. Given some system, can we tell something about its behavior, *before we can observe it*, using Turing machines?

2. Conversely, is it possible to create some physically realistic device which has more computational power than digital computers (Turing machines), either from a computability and/or a computational complexity perspective?
Two main problems

1. Given some system, can we tell something about its behavior, *before we can observe it*, using Turing machines?

2. Conversely, is it possible to create some physically realistic device which has more computational power than digital computers (Turing machines), either from a computability and/or a computational complexity perspective?
Some thoughts about the second problem

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?: quantum computers)
We propose to study dynamical systems which are defined by polynomial (vectorial) ordinary differential equations (ODEs):

\[ x' = p(x) \]
We propose to study dynamical systems which are defined by polynomial (vectorial) ordinary differential equations (ODEs):

\[ x' = p(x) \]

Why choose this type of dynamical systems?:

- Almost every (macroscopic) system which follows the classical laws of physics (up to my knowledge) can be written in terms of differential equations using the “usual” (elementary/closed-form) functions of Analysis: polynomials, trigonometric functions, etc.
- These differential equations can be rewritten as polynomial ODEs.
Example

The initial-value problem

\[
\begin{align*}
    x_1' &= \sin^2 x_2 \\
    x_2' &= x_1 \cos x_2 - e^{x_1 + t}
\end{align*}
\]

\[
\begin{align*}
    x_1(0) &= 0 \\
    x_2(0) &= 0
\end{align*}
\]
Example

The initial-value problem

\[
\begin{align*}
    x_1' &= \sin^2 x_2 \\
    x_2' &= x_1 \cos x_2 - e^{x_1 + t}
\end{align*}
\]

\[
\begin{align*}
    x_1(0) &= 0 \\
    x_2(0) &= 0
\end{align*}
\]

can be reduced to the following polynomial initial-value problem

\[
\begin{align*}
    y_1' &= y_3^2 \\
    y_2' &= y_1 y_4 - y_5 \\
    y_3' &= y_4 (y_1 y_4 - y_5) \\
    y_4' &= -y_3 (y_1 y_4 - y_5) \\
    y_5' &= y_5 (y_3^2 + 1)
\end{align*}
\]

\[
\begin{align*}
    y_1(0) &= 0 \\
    y_2(0) &= 0 \\
    y_3(0) &= 0 \\
    y_4(0) &= 1 \\
    y_5(0) &= 1
\end{align*}
\]

where \( y_1(t) = x_1(t) \) and \( y_2(t) = x_2(t) \)
Polynomial ODEs have a realistic model – Shannon’s General Purpose Analog Computer (GPAC), which can be implemented with mechanical devices or using (analog) electronics.

Fig 1
A (big) step closer to the Physical Church-Turing thesis

Theorem (Bournez, Campagnolo, Graça, Hainry)

The GPAC and Turing machines (computable analysis) are equivalent from a computability perspective
A (big) step closer to the Physical Church-Turing thesis

Theorem (Bournez, Campagnolo, Graça, Hainry)

The GPAC and Turing machines (computable analysis) are equivalent from a computability perspective

Theorem (Bournez, Graça, Pouly)

The GPAC and Turing machines (computable analysis) are (polynomially) equivalent from a computational complexity perspective
A (big) step closer to the Physical Church-Turing thesis

Theorem (Bournez, Campagnolo, Graça, Hainry)

The GPAC and Turing machines (computable analysis) are equivalent from a computability perspective

Theorem (Bournez, Graça, Pouly)

The GPAC and Turing machines (computable analysis) are (polynomially) equivalent from a computational complexity perspective

► But more on this on Olivier Bournez’s tutorial!
Given some physical system, can we tell something about its behavior, \textit{before we can observe it}, using Turing machines?
Let’s return to the first problem

1. Given some physical system, can we tell something about its behavior, *before we can observe it*, using Turing machines?

- Physical systems are often modeled as dynamical systems.
Let’s return to the first problem

1. Given some physical system, can we tell something about its behavior, *before we can observe it*, using Turing machines?

- Physical systems are often modeled as dynamical systems.
- Continuous-time dynamical systems defined on Euclidean spaces can be described by ordinary differential equations.
Two common problems, with many applications in practice, are to know the behavior of the system:

1. At a given time $t$
2. At infinity (asymptotic behavior)
For the first case we need to study the computability of the solutions of an ODE

\[ y' = f(y) \]

over a **non-compact** domain.
For the first case we need to study the computability of the solutions of an ODE

\[ y' = f(y) \]

over a non-compact domain.

- But isn’t this a trivial task?
For the first case we need to study the computability of the solutions of an ODE

\[ y' = f(y) \]

over a non-compact domain.

- But isn’t this a trivial task?
- No. The standard theory (via Picard iterations) is only guaranteed to work in a compact, where a Lipschitz constant for \( f \) exists.
- With this Lipschitz constant one is able to compute rigorous bounds on the error made by the approximation (the bound depends on the Lipschitz constant).
- But what to do in an open, potentially infinite domain, where no single Lipschitz constant is valid there?
Theorem (Graça, Zhong, Buescu)

The maximal interval of existence is not computable. You cannot even decide whether it is bounded or not.

The solution of

\[
\begin{align*}
    y' &= \frac{1}{\cos^2 x} \\
y(0) &= 0
\end{align*}
\]

is

\[y(x) = \tan(x)\]

The maximal interval of existence of the solution is

\((-\frac{\pi}{2}, \frac{\pi}{2})\)
Theorem (Cauchy-Peano)

If $f$ is continuous, then the initial-value problem $x' = f(t, x)$, $x(t_0) = x_0$
has a solution on a neighborhood of $x_0$

The notion of maximal interval of existence still makes sense in this case

- Can we still compute the solution (assuming it is unique) over its whole maximal interval of existence?
Computability of ODEs in the maximal interval

**Theorem (Cauchy-Peano)**

If $f$ is continuous, then the initial-value problem $x' = f(t, x), x(t_0) = x_0$ has a solution on a neighborhood of $x_0$.

The notion of maximal interval of existence still makes sense in this case.

- Can we still compute the solution (assuming it is unique) over its whole maximal interval of existence?

**Theorem (Collins, Graça)**

If $f$ is continuous, and $y$ is the unique solution of $x' = f(t, x), x(t_0) = x_0$, then the operator which maps $(f, t_0, x_0)$ to $y$ is computable (you can compute $y(t)$ on the whole maximal interval).
Idea of the proof

Exhaustive: generate all possible coverings of the state space (and of the tangent space)

- We can check (recursively) if a covering covers the solution
- We show that coverings of arbitrary small diameters exist

- So just keep testing coverings until you find an appropriate one (terribly inefficient, but enough for our purposes)
What about computational complexity?

Can’t we just use previous results from other authors?

No!

This is because previous results are valid on a bounded (time) domain.

Sometimes it is claimed that, by using rescaling techniques, if the solution of $y' = f(y)$ is computable in time $O(F(n))$ in $[0, 1]$, then it will also be computable in time $O(F(n))$ in its maximal interval.

▶

But this is incorrect!
What about computational complexity?

Can’t we just use previous results from other authors? No!

This is because previous results are valid on a bounded (time) domain.

Sometimes it is claimed that, by using rescaling techniques, if the solution of $y' = f(y)$ is computable in time $O(F(n))$ in $[0, 1]$, then it will also be computable in time $O(F(n))$ in its maximal interval.

But this is incorrect!
What about computational complexity?

Can’t we just use previous results from other authors? **No!**

- This is because previous results are valid on a *bounded (time) domain*. 
Can’t we just use previous results from other authors? No!

- This is because previous results are valid on a bounded (time) domain.
- Sometimes it is claimed that, by using rescaling techniques, if the solution of $y' = f(y)$ is computable in time $O(F(n))$ in $[0, 1]$, then it will also be computable in time $O(F(n))$ in its maximal interval.
What about computational complexity?

Can’t we just use previous results from other authors? No!

- This is because previous results are valid on a bounded (time) domain.
- Sometimes it is claimed that, by using rescaling techniques, if the solution of $y' = f(y)$ is computable in time $O(F(n))$ in $[0, 1]$, then it will also be computable in time $O(F(n))$ in its maximal interval.

► But this is incorrect!
Example

The solution of

\[
\begin{align*}
    y'_1(t) &= y_1(t) \\
    y'_2(t) &= y_1(t)y_2(t) \\
    &\vdots \\
    y'_n(t) &= y_1(t) \cdots y_n(t)
\end{align*}
\]

is polynomial time computable on any (time) bounded set.
Example

The solution of

\[
\begin{align*}
  y_1'(t) &= y_1(t) \\
  y_2'(t) &= y_1(t)y_2(t) \\
  &\vdots \\
  y_n'(t) &= y_1(t)\cdots y_n(t)
\end{align*}
\]

\[
\begin{align*}
  y_1(0) &= 1 \\
  y_2(0) &= 1 \\
  &\vdots \\
  y_n(0) &= 1
\end{align*}
\]

is polynomial time computable on any (time) bounded set. However its solution is

\[
\begin{align*}
  y_1(t) &= e^t \\
  y_2(t) &= e^{e^t - 1} \\
  y_n(t) &= e^{e^{\cdots e^t - 1}} - 1
\end{align*}
\]

which is obviously not polynomial time computable on its maximal interval of definition (\(\mathbb{R}\)).
In short:

It seems natural that, as $t$ increases, the more computational resources are needed to compute $y(t)$ with some precision $2^{-n}$.

Therefore it seems natural to measure the time needed to compute $y(t)$ against $n$ and $t$.

However, in some cases the time $t$ can be bounded (e.g. the case of an ODE having $\tan$ as solution) and we do not know how to tell when such cases occur (because this problem is undecidable).

▶ Therefore we have to use parametrized complexity (complexity is measured against one or more extra parameters).
In short:

- It seems natural that, as $t$ increases, the more computational resources are needed to compute $y(t)$ with some precision $2^{-n}$. 

Therefore we have to use parametrized complexity (complexity is measured against one or more extra parameters).
In short:

- It seems natural that, as $t$ increases, the more computational resources are needed to compute $y(t)$ with some precision $2^{-n}$.
- Therefore it seems natural to measure the time needed to compute $y(t)$ against $n$ and $t$. 
In short:

- It seems natural that, as $t$ increases, the more computational resources are needed to compute $y(t)$ with some precision $2^{-n}$.
- Therefore it seems natural to measure the time needed to compute $y(t)$ against $n$ and $t$.
- However, in some cases the time $t$ can be bounded (e.g., the case of an ODE having $\tan$ as solution) and we do not know how to tell when such cases occur (because this problem is undecidable).
In short:

- It seems natural that, as \( t \) increases, the more computational resources are needed to compute \( y(t) \) with some precision \( 2^{-n} \).
- Therefore it seems natural to measure the time needed to compute \( y(t) \) against \( n \) and \( t \).
- However, in some cases the time \( t \) can be bounded (e.g. the case of an ODE having \( \tan \) as solution) and we do not know how to tell when such cases occur (because this problem is undecidable).

Therefore we have to use parametrized complexity (complexity is measured against one or more extra parameters).
Proposition

If \((\alpha, \beta)\) is the maximal interval of existence of the solution \(y\) of an ODE \(y' = f(t, y)\) and \(\beta < +\infty\) then \(y(t)\) gets unbounded as \(t \to \beta\)

Solution: measure complexity against the length of the solution curve \(y\) between \((0, y(0))\) and \((t, y(t))\)
There is a numerical method SolvePIVP such that, for any $t \in \mathbb{R}$, $\varepsilon > 0$, if $y$ satisfies $y' = p(y)$, $y(t_0) = y_0$, it halts and returns a value $x = \text{SolvePIVP}(t_0, y_0, p, t, \varepsilon)$ such that:

- $\|x - y(t)\| \leq \varepsilon$
- the (bit) complexity of the algorithm is bounded by

$$\text{poly}(k, \text{Len}(t_0, t), \log \|y_0\|, \log \Sigma p, - \log \varepsilon)^d$$

where $k$ is the maximum degree of the components of $p$, $d$ is the number of components of $p$, $\Sigma p$ is the sum of the absolute values of the coefficients of $p$, and $\text{Len}(t_0, t)$ is a bound on the length of the curve $y(\cdot)$ from the point $(t_0, y(t_0))$ to the point $(t, y(t))$. 
Idea behind the proof

- Use a variable order method
- Use the hypothesis that the function defining the ODE is constituted by polynomials to get majorants
- Use an argument based on Cauchy majorants to establish a lower bound on the local radius of convergence
- Choose the step length $|t_{i+1} - t_i|$ to be a constant fraction of the estimated radius of convergence
- Choose an order of the method $\omega_i$ in the interval $[t_{i+1} - t_i]$ based on the computed majorants
- Start with a bound $l = 1$ for the length of the curve and double it in each run of the method if it does not succeed (this can be algorithmically detected).
In dynamical systems theory there is a great interest in telling what happens to a system “when time goes to infinity”.
In dynamical systems theory there is a great interest in telling what happens to a system “when time goes to infinity”.

Related problems can be found in applications (e.g. verification, control theory):
- Given an initial point $x_0$, will the trajectory starting from $x_0$ eventually reach some “unsafe region” (Reachability)?
- How many attractors (“steady states”) a system has? Can we characterize these attractors? Can we compute their basins of attractions—set of points on which the trajectory will converge towards a given attractor?
These are hard questions!

Example: if you apply Newton’s method to solve the (complex) equation $z^3 - 1 = 0$, you define a dynamical system with 3 attractors (the roots $1, \frac{-1-i\sqrt{3}}{2}, \frac{-1+i\sqrt{3}}{2}$ of the equation $z^3 - 1 = 0$) which have fractal basins of attraction.
What about attractors?

Roughly, attractors are invariant sets to which nearby trajectories converge (fragile attractors are usually dismissed). Types of attractors:

- Fixed points
- Periodic orbits (cycles)
- Surfaces, manifolds, etc.
- Strange attractors (Smale's horseshoe, Lorenz attractor, etc.): attractors with a fractal structure
Problem

Given a dynamical system \( y' = f(y) \), is it possible to compute the set of states (the non-wandering set \( \text{NW}(f) \)) to which the dynamics converge when time goes to infinity?
Problem

Given a dynamical system $y' = f(y)$, is it possible to compute the set of states (the non-wandering set $NW(f)$) to which the dynamics converge when time goes to infinity?

This is an interesting problem, but...

- Except for some very particular classes of systems, it is unknown in general how $NW(f)$ looks like
- We do know what happens on the (compact) two-dimensional case
- There are some theories (still on their early infancy) which try to address the three-dimensional case
- But for dimensions $\geq 4$: ????
The 2-dimensional case

**Theorem (Peixoto, 1959)**

On the two-dimensional disk $D = [0, 1]^2$, the set of structurally stable systems forms an open and dense set over the class of $C^1$ dynamical systems defined over $D$ (i.e. structurally stable systems are generic on $C^1(D)$). Moreover, any structurally stable system $y' = f(y)$ over $D$ has the following properties:

- **NW($f$) consists only of a finite number of periodic orbits and fixed points**
- **All periodic orbits and equilibria are hyperbolic**
- **There are no saddle connections**
Ongoing work (with N. Zhong)

$NW(f)$ is computable for structurally stable $C^1$ dynamical systems defined over the two-dimensional disk $D = [0, 1]^2$
Ongoing work (with N. Zhong)

NW(f) is computable for structurally stable $C^1$ dynamical systems defined over the two-dimensional disk $D = [0, 1]^2$

▶ What about the three-dimensional case?
Ongoing work (with N. Zhong)

\( NW(f) \) is computable for structurally stable \( C^1 \) dynamical systems defined over the two-dimensional disk \( D = [0, 1]^2 \)

What about the three-dimensional case?

Current research points to the direction that in the 3-dimensional case, attractors can be fixed point, periodic orbits, and/or Lorenz-like attractors.
Computability of the Lorenz attractor

Ongoing work (with N. Zhong)

Any (geometrical) Lorenz attractor is computable

Lorenz attractor
Previous results - fixed points

Theorem (Graça, Zhong)

Given as input an analytic function $f$, the problem of deciding the number of equilibrium points of $y' = f(y)$ is undecidable, even on compact sets. However, the set formed by all equilibrium points is upper semi-computable.
Theorem (Graça, Zhong)

Given as input an analytic function $f$, the problem of deciding the number of periodic orbits of $y' = f(y)$ is undecidable (on $\mathbb{R}^2$), even on compact sets. However, the set formed by all hyperbolic periodic orbits is upper semi-computable.
Idea of the proof

- Noncomputability arises from non-continuity problems related to the fixed points/periodic orbits.
- Nonetheless, the set consisting of all fixed points/periodic orbits of $f$ can be upper semi-computed by discretizing the space into small squares.
Strange attractors come in different shapes and flavors

Lorenz attractor

Peter De Jong attractor
Theorem (Graça, Zhong, Buescu, 2012)

*The Smale Horseshoe is a computable (recursive) closed set.*
Idea of the proof

We show that the complement of Smale’s horseshoe is computable by using the following fact (Zhong, 1996): An open subset $U \subseteq I$ is computable if and only if there is a computable sequence of rational open rectangles (having rational corner points) in $I$, $\{J_k\}_{k=0}^{\infty}$, such that

(a) $J_k \subset U$ for all $k \in \mathbb{N}$,

(b) the closure of $J_k$, $\overline{J}_k$, is contained in $U$ for all $k \in \mathbb{N}$, and

(c) there is a recursive function $e : \mathbb{N} \to \mathbb{N}$ such that the Hausdorff distance $d(I \setminus \bigcup_{k=0}^{e(n)} J_k, I \setminus U) \leq 2^{-n}$ for all $n \in \mathbb{N}$. 

Problem: can we tell to which attractor a trajectory starting in a given initial point will converge?
What about basins of attraction?

Problem: can we tell to which attractor a trajectory starting in a given initial point will converge?

- In some cases, the answer is YES (example: linear ODEs defined with hyperbolic matrices)
What about basins of attraction?

Problem: can we tell to which attractor a trajectory starting in a given initial point will converge?

- In some cases, the answer is YES (example: linear ODEs defined with hyperbolic matrices)
- In other cases the answer is NO (e.g. $C^k$ systems, where different functions can be “glued” together to allow the simulation of Turing machines)

Recall that in analytic functions, local behavior determines global behavior $\Rightarrow$ no $C^k$ gluing allowed, even if $k = +\infty$
What about basins of attraction?

Problem: can we tell to which attractor a trajectory starting in a given initial point will converge?

- In some cases, the answer is YES (example: linear ODEs defined with hyperbolic matrices)
- In other cases the answer is NO (e.g. $C^k$ systems, where different functions can be “glued” together to allow the simulation of Turing machines)

▶ But what if the system is analytic?
What about basins of attraction?

Problem: can we tell to which attractor a trajectory starting in a given initial point will converge?

- In some cases, the answer is YES (example: linear ODEs defined with hyperbolic matrices)
- In other cases the answer is NO (e.g. $C^k$ systems, where different functions can be “glued” together to allow the simulation of Turing machines)

But what if the system is analytic?

Recall that in analytic functions, local behavior determines global behavior

$\Rightarrow$ no $C^k$ gluing allowed, even if $k = +\infty$
Theorem (Graça, Zhong, 2015)

There exists a computable analytic dynamical system having a computable hyperbolic equilibrium point such that its basin of attraction is recursively enumerable, but not computable.
Theorem (Graça, Zhong, 2015)

There exists a computable analytic dynamical system having a computable hyperbolic equilibrium point such that its basin of attraction is recursively enumerable, but not computable.

Thus, even if:

- The attractor is of the simplest type (a fixed point)
- Trajectories converge in a well-behaved manner towards the fixed point (hyperbolicity: trajectories converge exponentially fast to the fixed point)
- The system is analytic (no gluing tricks allowed)
- All initial data is computable

Then the resulting basin of attraction may not be computable.
Theorem (Graça, Zhong, 2015)

There exists a computable analytic dynamical system having a computable hyperbolic equilibrium point such that its basin of attraction is recursively enumerable, but not computable.

Thus, even if:

- The attractor is of the simplest type (a fixed point)
Theorem (Graça, Zhong, 2015)

There exists a computable analytic dynamical system having a computable hyperbolic equilibrium point such that its basin of attraction is recursively enumerable, but not computable.

Thus, even if:

- The attractor is of the simplest type (a fixed point)
- Trajectories converge in a well-behaved manner towards the fixed point (hyperbolicity: trajectories converge exponentially fast to the fixed point)
Theorem (Graça, Zhong, 2015)

There exists a computable analytic dynamical system having a computable hyperbolic equilibrium point such that its basin of attraction is recursively enumerable, but not computable.

Thus, even if:

- The attractor is of the simplest type (a fixed point)
- Trajectories converge in a well-behaved manner towards the fixed point (hyperbolicity: trajectories converge exponentially fast to the fixed point)
- The system is analytic (no gluing tricks allowed)
Theorem (Graça, Zhong, 2015)

There exists a computable analytic dynamical system having a computable hyperbolic equilibrium point such that its basin of attraction is recursively enumerable, but not computable.

Thus, even if:

- The attractor is of the simplest type (a fixed point)
- Trajectories converge in a well-behaved manner towards the fixed point (hyperbolicity: trajectories converge exponentially fast to the fixed point)
- The system is analytic (no gluing tricks allowed)
- All initial data is computable
Theorem (Graça, Zhong, 2015)

There exists a computable analytic dynamical system having a computable hyperbolic equilibrium point such that its basin of attraction is recursively enumerable, but not computable.

Thus, even if:

- The attractor is of the simplest type (a fixed point)
- Trajectories converge in a well-behaved manner towards the fixed point (hyperbolicity: trajectories converge exponentially fast to the fixed point)
- The system is analytic (no gluing tricks allowed)
- All initial data is computable

Then the resulting basin of attraction may not be computable.
Idea behind the proof

- Simulate a Turing machine with an analytic map (use interpolation techniques, and allow a certain error in the simulation—the map can still simulate a Turing machine even if the initial point and/or the dynamics are constantly perturbed. Use special techniques to keep the error under control)

Suspend the previous map into an ODE. The classical suspension technique does not work here because it is not constructive. Instead we develop a new whole "computable" suspension technique which allows to embed a computable map into a computable ODE, under certain conditions.

The previous ODE will simulate a Turing machine and we "massage" the ODE so that the halting state corresponds to an hyperbolic fixed point (the ODE simulation of TMs is robust to perturbations).

Then deciding which initial points will converge to the previous hyperbolic fixed point is equivalent to solving the Halting Problem.
Idea behind the proof

- Simulate a Turing machine with an analytic map (use interpolation techniques, and allow a certain error in the simulation—the map can still simulate a Turing machine even if the initial point and/or the dynamics are constantly perturbed. Use special techniques to keep the error under control)

- Suspend the previous map into an ODE. The classical suspension technique does not work here because it is not constructive. Instead we develop a new whole “computable” suspension technique which allows to embed a computable map into a computable ODE, under certain conditions
Idea behind the proof

- Simulate a Turing machine with an analytic map (use interpolation techniques, and allow a certain error in the simulation—the map can still simulate a Turing machine even if the initial point and/or the dynamics are constantly perturbed. Use special techniques to keep the error under control)

- Suspend the previous map into an ODE. The classical suspension technique does not work here because it is not constructive. Instead we develop a new whole “computable” suspension technique which allows to embed a computable map into a computable ODE, under certain conditions

- The previous ODE will simulate a Turing machine and we “massage” the ODE so that the halting state corresponds to an hyperbolic fixed point (the ODE simulation of TMs is robust to perturbations)
Idea behind the proof

- Simulate a Turing machine with an analytic map (use interpolation techniques, and allow a certain error in the simulation—the map can still simulate a Turing machine even if the initial point and/or the dynamics are constantly perturbed. Use special techniques to keep the error under control)

- Suspend the previous map into an ODE. The classical suspension technique does not work here because it is not constructive. Instead we develop a new whole “computable” suspension technique which allows to embed a computable map into a computable ODE, under certain conditions

- The previous ODE will simulate a Turing machine and we “massage” the ODE so that the halting state corresponds to an hyperbolic fixed point (the ODE simulation of TMs is robust to perturbations)

- Then deciding which initial points will converge to the previous hyperbolic fixed point is equivalent to solving the Halting Problem
Takeaways of this talk

- Having high quality data and models is not enough to analyze natural phenomena.
Takeaways of this talk

- Having high quality data and models is not enough to analyze natural phenomena.
- The Physical Church-Turing thesis seems to hold for a large class of physical devices, both at a *computability* and *complexity* level. It seems that digital computers (Turing or Type-2 machines) are the best computing devices available (except quantum computers?).
Takeaways of this talk

- Having high quality data and models is not enough to analyze natural phenomena.
- The Physical Church-Turing thesis seems to hold for a large class of physical devices, both at a *computability* and *complexity* level. It seems that digital computers (Turing or Type-2 machines) are the best computing devices available (except quantum computers?).
- The solution of differential equations can be computed over unbounded domains (and this case is more general than the bounded case). However, complexity of the computation might need to be parametrized.
Takeaways of this talk

- Having high quality data and models is not enough to analyze natural phenomena.

- The Physical Church-Turing thesis seems to hold for a large class of physical devices, both at a *computability* and *complexity* level. It seems that digital computers (Turing or Type-2 machines) are the best computing devices available (except quantum computers?).

- The solution of differential equations can be computed over unbounded domains (and this case is more general than the bounded case). However, complexity of the computation might need to be parametrized.

- Although the dynamics of systems (of dimension \( \leq 3 \)) might be complicated, their attractors are usually computable, though the basins of attraction are not, even for “well-behaved” systems.
Thank you!