Computing the asymptotic behavior of low-dimensional dynamical systems

Daniel S. Graça[1,2]

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

Algorithmic Questions in Dynamical Systems
(Toulouse, 28 March 2018)

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

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 digital 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 digital 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 digital 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 (digital) computers Laplace's demon?

Which model should we consider?

- Almost every (macroscopic) system which follows the classical (deterministic) laws of physics can be written in terms of differential equations.
- In particular many systems can be modeled as ordinary differential equations (ODEs).
- Moreover ODEs are better understood from a mathematical perspective than PDEs.

Which model should we consider?

- Almost every (macroscopic) system which follows the classical (deterministic) laws of physics can be written in terms of differential equations.
- In particular many systems can be modeled as ordinary differential equations (ODEs).
- Moreover ODEs are better understood from a mathematical perspective than PDEs.

In this talk we will study ODEs from a computational perspective

Which model should we consider?

- Almost every (macroscopic) system which follows the classical (deterministic) laws of physics can be written in terms of differential equations.
- In particular many systems can be modeled as ordinary differential equations (ODEs).
- Moreover ODEs are better understood from a mathematical perspective than PDEs.

In this talk we will study ODEs from a computational perspective

- What can be computed about ODEs using Turing-like models?
  - **This talk**

Which model should we consider?

- Almost every (macroscopic) system which follows the classical (deterministic) laws of physics can be written in terms of differential equations.
- In particular many systems can be modeled as ordinary differential equations (ODEs).
- Moreover ODEs are better understood from a mathematical perspective than PDEs.

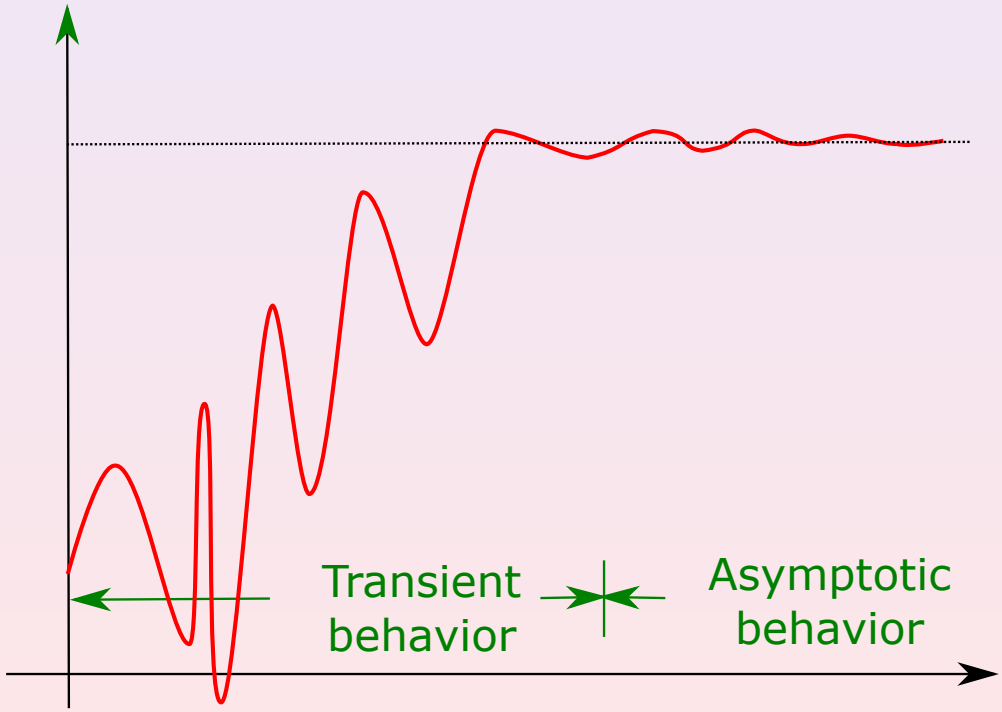
In this talk we will study ODEs from a computational perspective

- What can be computed about ODEs using Turing-like models?
  - ▶ **This talk**
- What can be computed with models of computation defined with ODEs? ▶ **Olivier Bournez's talk**

Computation of the evolution of dynamical systems

Two common problems, with many applications in practice, are to know the behavior of a dynamical system:

1. At a given time t
2. Near 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 (note that $y' = f(t, y)$ can be reduced to the above case by replacing t by a new variable $x'_{n+1} = 1$, $x(0) = 0$).

For the first case we need to study the computability of the solutions of an ODE

$$y' = f(y)$$

over a **non-compact** domain (note that $y' = f(t, y)$ can be reduced to the above case by replacing t by a new variable $x'_{n+1} = 1$, $x(0) = 0$).

- 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 (note that $y' = f(t, y)$ can be reduced to the above case by replacing t by a new variable $x'_{n+1} = 1$, $x(0) = 0$).

- 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?

- It is possible to show that there is a maximal interval of existence in the sense that either the solution if defined for all times or it "blows up" in finite time.

- It is possible to show that there is a maximal interval of existence in the sense that either the solution if defined for all times or it "blows up" in finite time.
- The construction of this interval is as follows: a solution which is defined in a compact set is extended over and over again to a "bigger" compact set yielding, in the limit, a unique solution in the maximal interval of existence.

- It is possible to show that there is a maximal interval of existence in the sense that either the solution if defined for all times or it "blows up" in finite time.
- The construction of this interval is as follows: a solution which is defined in a compact set is extended over and over again to a "bigger" compact set yielding, in the limit, a unique solution in the maximal interval of existence.
- Alas this procedure is not computable since this limit procedure is not computable.

Maximal interval of existence

Theorem (G., Zhong, Buescu '09)

The maximal interval of existence is not computable. We cannot even decide whether it is bounded. These results hold even if f is analytic.

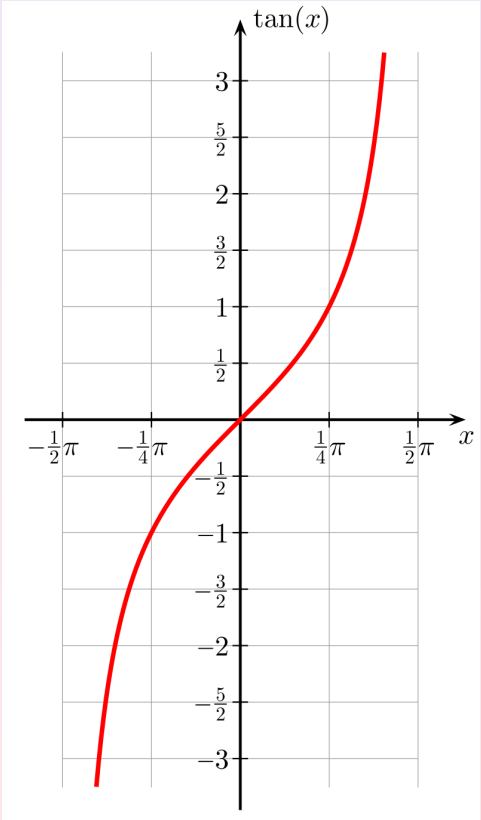
The solution of

$$\begin{cases} y' = \frac{1}{\cos^2 x} \\ y(0) = 0 \end{cases}$$

is

$$y(x) = tan(x)$$

The maximal interval of existence of the solution is $\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$



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?

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?

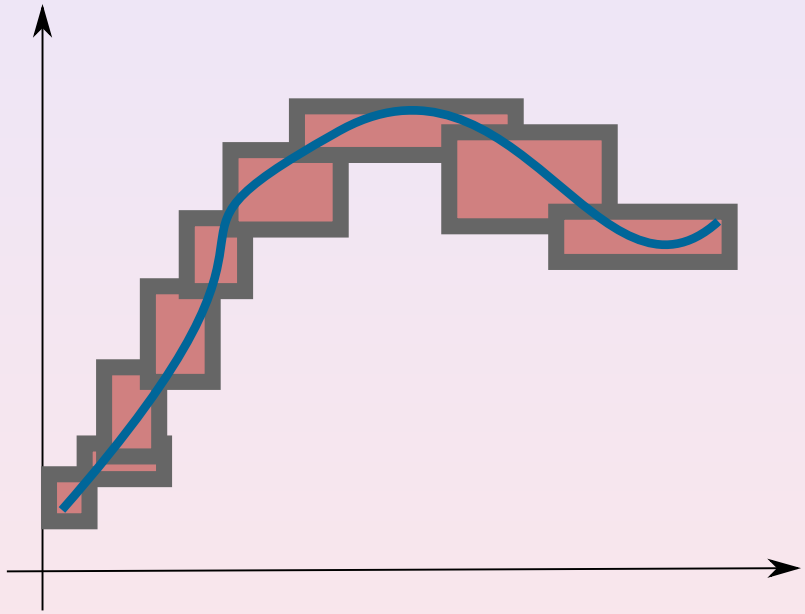
Theorem (Collins, G. '09)

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 (partial) 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 (Hence the "*ten thousand monkeys approach*" in the title of this paper. This is terribly inefficient, but enough for our purposes).

What about computational complexity?

Results for the compact case. The leftmost column has assumptions on f in $y' = f(y)$, $y(0) = 0$. The two rightmost columns indicate the complexity of the solution.

Assumptions	Upper bound	Lower bound
ODE with unique solution	Computable	Arbitrary high complexity
Lipschitz ODE	PSPACE	PSPACE
f is of class C^1	PSPACE	PSPACE
f is of class $C^k, k > 1$	PSPACE	CH
f is analytic	polynomial-time	polynomial-time

- These results were obtained by people such as [Miller '70], [Ko '83], [Müller, '87] [Kawamura, '10],[Kawamura et al., '14]

What about computational complexity?

Can't we just use the previous results?

What about computational complexity?

Can't we just use the previous results? No!

What about computational complexity?

Can't we just use the previous results? No!

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

What about computational complexity?

Can't we just use the previous results? No!

- This is because previous results are valid on a *(time) bounded 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 the previous results? No!

- This is because previous results are valid on a *(time) bounded 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

$$\begin{cases} y_1'(t) = y_1(t) \\ y_2'(t) = y_1(t)y_2(t) \\ \quad \cdots \\ y_n'(t) = y_1(t) \cdots y_n(t) \end{cases} \qquad \begin{cases} y_1(0) = 1 \\ y_2(0) = 1 \\ \quad \cdots \\ y_n(0) = 1 \end{cases}$$

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

Example

The solution of

$$\begin{cases} y_1'(t) = y_1(t) \\ y_2'(t) = y_1(t)y_2(t) \\ \quad \cdots \\ y_n'(t) = y_1(t)\cdots y_n(t) \end{cases} \qquad \begin{cases} y_1(0) = 1 \\ y_2(0) = 1 \\ \quad \cdots \\ y_n(0) = 1 \end{cases}$$

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

$$y_1(t) = e^t \qquad y_2(t) = e^{e^t - 1} \qquad y_n(t) = e^{e^{\cdot^{\cdot^{\cdot^{e^{e^t}-1}}}}-1}$$

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

In short:

In short:

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

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 not computable).

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 not computable).

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

Proposition

If (α, β) 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 $(t, y(t))$

Theorem (G., Pouly '16)

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* $x = \mathrm{SolvePIVP}(t_0, y_0, p, t, \varepsilon)$ *such that:*

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

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

where k *is the maximum degree of the components of* p, d *is the number of components of* p, Σp *is the sum of the absolute values of the coefficients of* p, *and* $\mathrm{Len}(t_0, t)$ *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 ω_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).

Computation of the asymptotic behavior of ODEs

- In dynamical systems theory there is a great interest in telling what happens to a system "when time goes to infinity".

Computation of the asymptotic behavior of ODEs

- In dynamical systems theory there is a great interest in telling what happens to a system "when time goes to infinity".
  - In general this is a very hard problem!

Computation of the asymptotic behavior of ODEs

- In dynamical systems theory there is a great interest in telling what happens to a system "when time goes to infinity".
  - In general this is a very hard problem!
- 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 – the set of points on which the trajectory will converge towards a given attractor?

Are attractors (limit sets) computable?

Roughly, attractors are invariant sets to which nearby trajectories converge. Some types of attractors:

- Fixed points
- Periodic orbits (cycles)
- 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 $NW(f)$) 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...

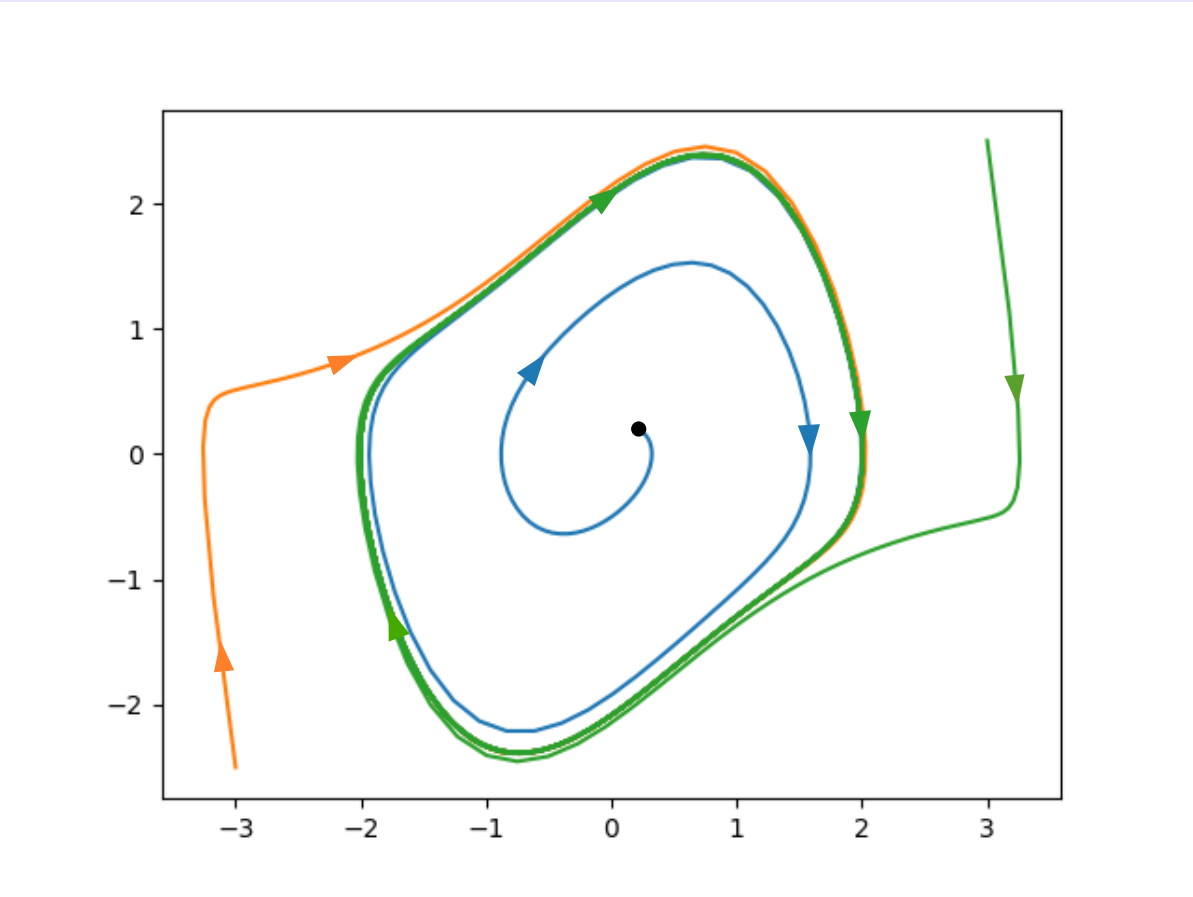
- Except for some very particular classes of systems, it is unknown in general how $NW(f)$ looks like
- We do know what happens on the (compact) two-dimensional case
- The are some theories (still on their early infancy) which try to address the three-dimensional case
- But for dimensions ≥ 4: ????

The 2-dimensional case

Theorem (Peixoto '59)

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*



Example: Van der Pol system $\begin{cases} x' = y \\ y' = \mu(1-x^2)y - x \end{cases}$, where $\mu = 0.7$

Previous results - fixed points

Theorem (G., Zhong '11)

Given as input an analytic function f, the problem of computing the number of equilibrium points of $y' = f(y)$ is undecidable, even on compact sets.

Previous results - fixed points

Theorem (G., Zhong '11)

Given as input an analytic function f, the problem of computing the number of equilibrium points of $y' = f(y)$ is undecidable, even on compact sets.

- Noncomputability arises from the well-known non-continuity problems related to finding the zeros of f.

Previous results - fixed points

Theorem (G., Zhong '11)

Given as input an analytic function f, the problem of computing the number of equilibrium points of $y' = f(y)$ is undecidable, even on compact sets.

- Noncomputability arises from the well-known non-continuity problems related to finding the zeros of f.

Theorem (G., Zhong '11)

There is no algorithm which (uniformly) computes the limit set of a dynamical system defined by an ODE $y' = f(y)$ on the two-dimensional disk $D = [0, 1]^2$.

Previous results - fixed points

Theorem (G., Zhong '11)

Given as input an analytic function f, the problem of computing the number of equilibrium points of $y' = f(y)$ is undecidable, even on compact sets.

- Noncomputability arises from the well-known non-continuity problems related to finding the zeros of f.

Theorem (G., Zhong '11)

There is no algorithm which (uniformly) computes the limit set of a dynamical system defined by an ODE $y' = f(y)$ on the two-dimensional disk $D = [0, 1]^2$.

- Idea: use bifurcation phenomena.

Periodic orbits

Theorem (G., Zhong '11)

Given as input an analytic function f, the problem of computing the number of periodic orbits of $y' = f(y)$ is undecidable (on $\mathbb{R}^2$), even on compact sets.

Periodic orbits

Theorem (G., Zhong '11)

Given as input an analytic function f, the problem of computing the number of periodic orbits of $y' = f(y)$ is undecidable (on $\mathbb{R}^2$), even on compact sets.

- Consider the function g defined by

$$g(k,i) = \begin{cases} 0 & \text{if } TM_k \text{ stops in } \leq i \text{ steps with input } k \\ 1 & \text{otherwise.} \end{cases}$$

  Then $\sum_{i=1}^{\infty} \frac{g(k,i)}{2^i}$ is a computable number.
- The computable system given in polar coordinates

$$\begin{cases} r' = \left(r - \sum_{i=1}^{\infty} \frac{g(k,i)}{2^i}\right)(r-1) \\ \theta' = 1 \end{cases}$$

  has two periodic orbits if the Turing machine TM_k halts on input k, and has only one periodic orbit otherwise.

Work on progress with N. Zhong

The limit set $NW(f)$ is computable for structurally stable systems defined on the two-dimensional disk $D = [0,1]^2$.

- The set consisting of all fixed points/periodic orbits of f can be upper semi-computed by discretizing the space into small squares and by defining a transition function over a finite set. We can then compute the invariant sets of this discretization.
- Structural stability ensures that approximating a system by finite but arbitrarily accurate approximations will still provide meaningful results.
- We can then measure the "diameter" of the approximation to get a bound on its accuracy

A small detour: Hilbert's 16th problem

Hilbert's 16th Problem

Determine an upper bound on the maximum number of periodic orbits that a nth degree polynomial ODE

$$\begin{cases} x' = p_n(x, y) \\ y' = q_n(x, y) \end{cases}$$

in the plane can have (the original formulation also asks for the relative position of the periodic orbits).

Problem

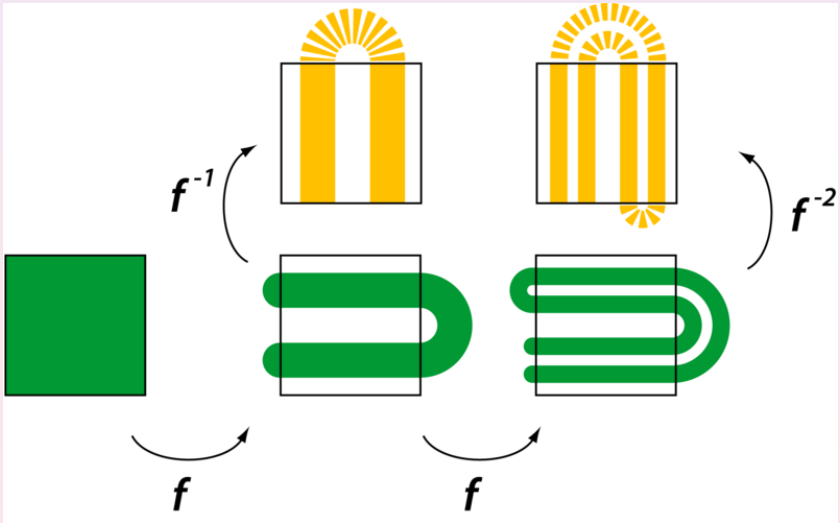
Can we uniformly compute the maximum number periodic orbits that a nth degree polynomial ODE can have?

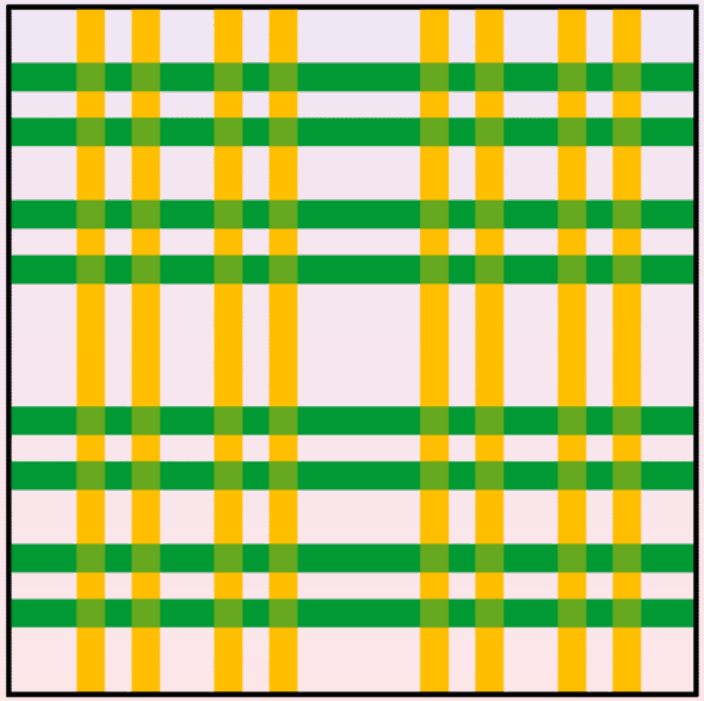
- Non-computability does not imply that no solution exists for Hilbert's 16th problem.

What about strange attractors?

Theorem (G., Zhong, Buescu '12)

The Smale Horseshoe is a computable (recursive) closed set.





Idea of the proof

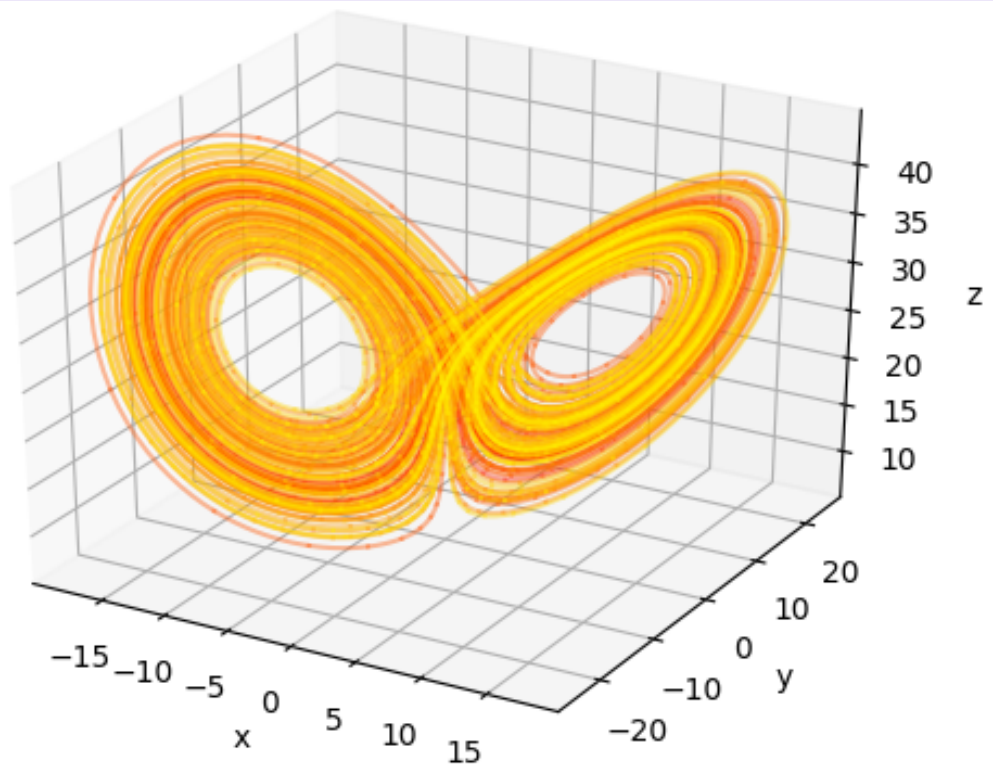
We show that the complement of Smale's horseshoe is computable by using the following fact (Zhong, 1996): An open subset $U \subseteq I$ is computable if and only if there is a computable sequence of rational open rectangles (having rational corner points) in I, $\{J_k\}_{k=0}^{\infty}$, such that

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

(b) the closure of J_k, $\bar{J}_k$, 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 $d(I \setminus \cup_{k=0}^{e(n)} J_k, I \setminus U) \leq 2^{-n}$ for all $n \in \mathbb{N}$.

The Lorenz attractor



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

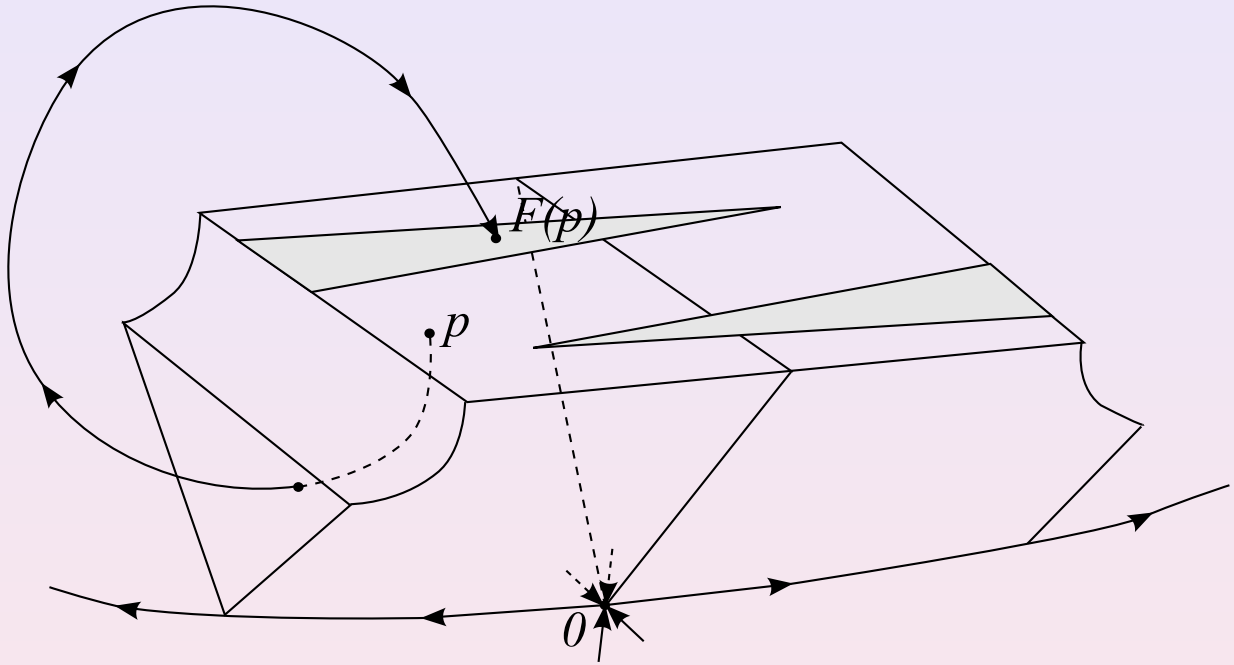
Classical values for parameters: $\sigma = 10, \rho = 28, \beta = 8/3$

The butterfly effect

- However the Lorenz system has sensitive dependence on initial conditions – i.e. it exhibits the "butterfly effect": the flutter of a butterfly's wing can ultimately cause a typhoon halfway around the world
- This term was coined by E. Lorenz since the Lorenz system represents a simplified mathematical model for atmospheric convection.

Some preliminaries

- In the late 1970s several authors suggested the use of geometrical Lorenz models to better understand the Lorenz attractor.
- Such models assume the qualitative behavior which was numerically observed on the Lorenz system.
- It was soon shown that geometrical Lorenz models have a strange attractor with properties compatible to those observed via numerical experiments.
- W. Tucker essentially showed in 2002 (using rigorous numerics and normal form theory) that the Lorenz system behaves like a geometric Lorenz model, thus supporting a strange attractor.


F(p)
p
0

Theorem (G., Rojas, Zhong '18)

Let ϕ be the a (C^2) flow of some Lorenz geometric system. Then:

1. *The global attractor $\mathcal{A}$ of a geometric Lorenz flow ϕ is computable from a (C^2) name of ϕ.*
2. *The geometric Lorenz flow admits a physical measure which is computable from a (C^2) name of ϕ.*

What about the Lorenz attractor

Question

Does the previous result prove computability of the (real) Lorenz attractor?

What about the Lorenz attractor

Question

Does the previous result prove computability of the (real) Lorenz attractor?

- Not yet!
- The problem has to do with the fact that we need a constructive version of Tucker's proof.
- In particular it is not enough to show that a foliation exists for the Lorenz attractor.
- We still have to show that a computable foliation of the Lorenz attractor exists which can be computably mapped into the standard foliation.
- Probably some combination of theory about foliations and rigorous numerics is needed to prove that result (to be done!)

What about basins of attraction?

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)

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 (G., Zhong '15)

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 (G., Zhong '15)

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:

Theorem (G., Zhong '15)

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 (G., Zhong '15)

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 (G., Zhong '15)

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 (G., Zhong '15)

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 (G., Zhong '15)

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)

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

Thank you!