Noise vs Computational unpredictability in dynamics

Cristóbal Rojas
Joint with M. Braverman and A. Grigo.

Universidad Andrés Bello
Santiago, Chlie.

July 10, 2013

Predicting Natural Phenomena: can we compute the future?

Predicting Natural Phenomena: can we compute the future?

- Where will the pendulum be tomorrow at noon?

Predicting Natural Phenomena: can we compute the future?

- Where will the pendulum be tomorrow at noon?
- Is it gonna rain next week ?

Predicting Natural Phenomena: can we compute the future?

- Where will the pendulum be tomorrow at noon?
- Is it gonna rain next week ?
- What is the probability for rain next week?

Predicting Natural Phenomena: can we compute the future?

- Where will the pendulum be tomorrow at noon?
- Is it gonna rain next week ?
- What is the probability for rain next week?

More generally

Given an evolving system, can we compute its long term prospects?

Given a dynamical system (X, T)

Two (at least) fundamental barriers to our ability to predict the future:

Given a dynamical system (X, T)

Two (at least) fundamental barriers to our ability to predict the future:

- Chaotic behavior : $+$ approximation $\Rightarrow$ unpredictability of individual trajectories – is a prevalent situation

Given a dynamical system (X, T)

Two (at least) fundamental barriers to our ability to predict the future:

- Chaotic behavior : + approximation $\Rightarrow$ unpredictability of individual trajectories – is a prevalent situation

  Solution: focus on more global, asymptotic objects: attractors/repellers, invariant measures.

Given a dynamical system (X, T)

Two (at least) fundamental barriers to our ability to predict the future:

- Chaotic behavior : + approximation $\Rightarrow$ unpredictability of individual trajectories – is a prevalent situation

  Solution: focus on more global, asymptotic objects: attractors/repellers, invariant measures.

- Turing Completeness : rich systems can simulate universal computation $\Rightarrow$ uncomputable features

Given a dynamical system (X, T)

Two (at least) fundamental barriers to our ability to predict the future:

- Chaotic behavior : + approximation $\Rightarrow$ unpredictability of individual trajectories – is a prevalent situation

  Solution: focus on more global, asymptotic objects: attractors/repellers, invariant measures.

- Turing Completeness : rich systems can simulate universal computation $\Rightarrow$ uncomputable features

  but... is this a prevalent situation? does it occur with positive probability? does it persist after small perturbations?

Dynamical systems and Natural Phenomena: mathematical models

A dynamical system is a space of states X together with a map $T : X \to X$.

Idea: starting at state x_0, the state of the system after n units of time is:

$$T^n(x_0) = T \circ T \circ T \cdots \circ T(x_0) \qquad (n \text{ times}).$$

Dynamical systems and Natural Phenomena: mathematical models

A dynamical system is a space of states X together with a map $T : X \to X$.

Idea: starting at state x_0, the state of the system after n units of time is:

$$T^n(x_0) = T \circ T \circ T \cdots \circ T(x_0) \qquad (n \text{ times}).$$

A typical scenario can be roughly described as follows:

Dynamical systems and Natural Phenomena: mathematical models

A dynamical system is a space of states X together with a map $T : X \to X$.

Idea: starting at state x_0, the state of the system after n units of time is:

$$T^n(x_0) = T \circ T \circ T \cdots \circ T(x_0) \qquad (n \text{ times}).$$

A typical scenario can be roughly described as follows:

- phase space X can be divided into *regions* B_i.

Dynamical systems and Natural Phenomena: mathematical models

A dynamical system is a space of states X together with a map $T : X \to X$.

Idea: starting at state x_0, the state of the system after n units of time is:

$$T^n(x_0) = T \circ T \circ T \cdots \circ T(x_0) \qquad (n \text{ times}).$$

A typical scenario can be roughly described as follows:

- phase space X can be divided into *regions* B_i.
- trajectories starting in B_i approach a same "attractor"

Dynamical systems and Natural Phenomena: mathematical models

A dynamical system is a space of states X together with a map $T : X \to X$.

Idea: starting at state x_0, the state of the system after n units of time is:

$$T^n(x_0) = T \circ T \circ T \cdots \circ T(x_0) \qquad (n \text{ times}).$$

A typical scenario can be roughly described as follows:

- phase space X can be divided into *regions* B_i.
- trajectories starting in B_i approach a same "attractor"
- any probability distribution supported in the region evolves towards an invariant one, supported on the attractor.

Dynamical systems and Natural Phenomena: mathematical models

A dynamical system is a space of states X together with a map $T : X \to X$.

Idea: starting at state x_0, the state of the system after n units of time is:

$$T^n(x_0) = T \circ T \circ T \cdots \circ T(x_0) \qquad (n \text{ times}).$$

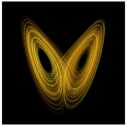
A typical scenario can be roughly described as follows:

- phase space X can be divided into *regions* B_i.
- trajectories starting in B_i approach a same "attractor"
- any probability distribution supported in the region evolves towards an invariant one, supported on the attractor.
- The "frontiers" between regions (basins) are invariant "repellers" supporting other invariant measures.

Some examples

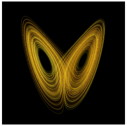
Some examples

- Lorenz equations

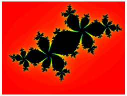


Some examples

- Lorenz equations

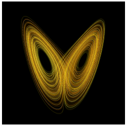


- Polynomials on the complex plane (Julia sets: repellers)

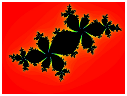


Some examples

- Lorenz equations



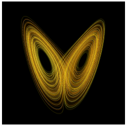
- Polynomials on the complex plane (Julia sets: repellers)



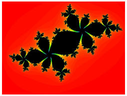
- symbolic systems: cellular automata, subshifts

Some examples

- Lorenz equations



- Polynomials on the complex plane (Julia sets: repellers)



- symbolic systems: cellular automata, subshifts
- piece-wise linear transformations
  - Neural networks – agent systems (high dimensional)
  - Billiards, ray-tracing (low-dimensional)

Computation and Dynamical Systems: interactions

Dynamical systems as computing machines

How much computational power does a dynamical system have?

Some examples:

Computation and Dynamical Systems: interactions

Dynamical systems as computing machines

How much computational power does a dynamical system have?

Some examples:

- Piece-wise linear maps in two dimensions $\equiv$ full Turing-power (Moore, Koiran et al.)

Computation and Dynamical Systems: interactions

Dynamical systems as computing machines

How much computational power does a dynamical system have?

Some examples:

- Piece-wise linear maps in two dimensions $\equiv$ full Turing-power (Moore, Koiran et al.)
- Piece-wise linear maps in one dimension $\equiv$ push-down automata (Moore, Koiran)

Computation and Dynamical Systems: interactions

Dynamical systems as computing machines

How much computational power does a dynamical system have?

Some examples:

- Piece-wise linear maps in two dimensions $\equiv$ full Turing-power (Moore, Koiran et al.)
- Piece-wise linear maps in one dimension $\equiv$ push-down automata (Moore, Koiran)
- Unimodal 1D-maps are not universal (Kurka).

Computation and Dynamical Systems: interactions

Dynamical systems as computing machines

How much computational power does a dynamical system have?

Some examples:

- Piece-wise linear maps in two dimensions $\equiv$ full Turing-power (Moore, Koiran et al.)
- Piece-wise linear maps in one dimension $\equiv$ push-down automata (Moore, Koiran)
- Unimodal 1D-maps are not universal (Kurka).
- Tilings of the plane $\equiv$ full-turing power (Berger, Robinson)

Computation and Dynamical Systems: interactions

Computability in dynamical systems

What dynamical features can be computed ?

Positive results:

Computation and Dynamical Systems: interactions

Computability in dynamical systems

What dynamical features can be computed ?

Positive results:

- Most Julia sets are computable (Rettinger, Weihrauch, Braverman, Yampolsky)

Computation and Dynamical Systems: interactions

Computability in dynamical systems

What dynamical features can be computed ?

Positive results:

- Most Julia sets are computable (Rettinger, Weihrauch, Braverman, Yampolsky)
- Smale's Horseshoe is computable (Graca, Zhong, Buescu)

Computation and Dynamical Systems: interactions

Computability in dynamical systems

What dynamical features can be computed ?

Positive results:

- Most Julia sets are computable (Rettinger, Weihrauch, Braverman, Yampolsky)
- Smale's Horseshoe is computable (Graca, Zhong, Buescu)
- Local stable and unstable manifolds in hyperbolic systems are computable (Graca, Zhong, Buescu)

Computation and Dynamical Systems: interactions

Computability in dynamical systems

What dynamical features can be computed ?

Positive results:

- Most Julia sets are computable (Rettinger, Weihrauch, Braverman, Yampolsky)
- Smale's Horseshoe is computable (Graca, Zhong, Buescu)
- Local stable and unstable manifolds in hyperbolic systems are computable (Graca, Zhong, Buescu)
- Invariant measures are computable for:
  - Piece-wise expanding maps and hyperbolic systems (Galatolo, Hoyrup, R.)
  - Harmonic measure on Julia sets (Binder, Braverman, Yampolsky, R.)

Computation and Dynamical Systems: interactions

Computability in dynamical systems

What dynamical features can be computed ?

Positive results:

- Most Julia sets are computable (Rettinger, Weihrauch, Braverman, Yampolsky)
- Smale's Horseshoe is computable (Graca, Zhong, Buescu)
- Local stable and unstable manifolds in hyperbolic systems are computable (Graca, Zhong, Buescu)
- Invariant measures are computable for:
  - Piece-wise expanding maps and hyperbolic systems (Galatolo, Hoyrup, R.)
  - Harmonic measure on Julia sets (Binder, Braverman, Yampolsky, R.)
- For ergodic systems there exists computable *generic* points (Avigad, Gerhardy, Towsner, Gacs, Galatolo, Hoyrup, R.).

Computation and Dynamical Systems: interactions

Computability in dynamical systems

What dynamical features can be computed ?

Negative results:

Computation and Dynamical Systems: interactions

Computability in dynamical systems

What dynamical features can be computed ?

Negative results:

- Reachability problems are undecidable (Asarin, Bournez, Koiran, Blondel)

Computation and Dynamical Systems: interactions

Computability in dynamical systems

What dynamical features can be computed ?

Negative results:

- Reachability problems are undecidable (Asarin, Bournez, Koiran, Blondel)
- Entropy is uncomputable for piece-wise linear maps in dimension 4 (Koiran) and for cellular automata (Kari)

Computation and Dynamical Systems: interactions

Computability in dynamical systems

What dynamical features can be computed ?

Negative results:

- Reachability problems are undecidable (Asarin, Bournez, Koiran, Blondel)
- Entropy is uncomputable for piece-wise linear maps in dimension 4 (Koiran) and for cellular automata (Kari)
- Global stable and unstable manifolds are not computable in general (Graca, Ning, Buescu)

Computation and Dynamical Systems: interactions

Computability in dynamical systems

What dynamical features can be computed ?

Negative results:

- Reachability problems are undecidable (Asarin, Bournez, Koiran, Blondel)
- Entropy is uncomputable for piece-wise linear maps in dimension 4 (Koiran) and for cellular automata (Kari)
- Global stable and unstable manifolds are not computable in general (Graca, Ning, Buescu)
- There exists uncomputable Julia sets (Braverman, Yampolsky)

Computation and Dynamical Systems: interactions

Computability in dynamical systems

What dynamical features can be computed ?

Negative results:

- Reachability problems are undecidable (Asarin, Bournez, Koiran, Blondel)
- Entropy is uncomputable for piece-wise linear maps in dimension 4 (Koiran) and for cellular automata (Kari)
- Global stable and unstable manifolds are not computable in general (Graca, Ning, Buescu)
- There exists uncomputable Julia sets (Braverman, Yampolsky)
- There exists computable systems *without* computable invariant measures (Galatolo, Hoyrup, R.).

Computation and Dynamical Systems: interactions

Complexity in dynamical systems

What is the complexity of computing a given dynamical feature?

Some examples:

Computation and Dynamical Systems: interactions

Complexity in dynamical systems

What is the complexity of computing a given dynamical feature?

Some examples:

- Hyperbolic Julia sets are poly-time computable (Weihrauch, Rettinger, Braverman)

Computation and Dynamical Systems: interactions

Complexity in dynamical systems

What is the complexity of computing a given dynamical feature?

Some examples:

- Hyperbolic Julia sets are poly-time computable (Weihrauch, Rettinger, Braverman)
- Cremer Julia sets are arbitrarily complex (Braverman, Yampolsky)

Computation and Dynamical Systems: interactions

Complexity in dynamical systems

What is the complexity of computing a given dynamical feature?

Some examples:

- Hyperbolic Julia sets are poly-time computable (Weihrauch, Rettinger, Braverman)
- Cremer Julia sets are arbitrarily complex (Braverman, Yampolsky)
- more examples in the next talk...
- ...

What about physically plausible systems showing uncomputable/intractable phenomena?

Notable fact: it appears that all the negative results are *fragile* in one way or another.

What about physically plausible systems showing uncomputable/intractable phenomena?

Notable fact: it appears that all the negative results are *fragile* in one way or another.

Are there physically robust systems exhibiting Turing-universal power?

What about physically plausible systems showing uncomputable/intractable phenomena?

Notable fact: it appears that all the negative results are *fragile* in one way or another.

Are there physically robust systems exhibiting Turing-universal power?

YES! my laptop ...

What about physically plausible systems showing uncomputable/intractable phenomena?

Notable fact: it appears that all the negative results are *fragile* in one way or another.

Are there physically robust systems exhibiting Turing-universal power?

YES! my laptop ... but it would need unlimited storage (unlimited physical space).

What about physically plausible systems showing uncomputable/intractable phenomena?

Notable fact: it appears that all the negative results are *fragile* in one way or another.

Are there physically robust systems exhibiting Turing-universal power?

YES! my laptop ... but it would need unlimited storage (unlimited physical space).

What about low-dimensional, compact systems?

What about physically plausible systems showing uncomputable/intractable phenomena?

Notable fact: it appears that all the negative results are *fragile* in one way or another.

Are there physically robust systems exhibiting Turing-universal power?

YES! my laptop ... but it would need unlimited storage (unlimited physical space).

What about low-dimensional, compact systems?

Conjecture

Uncomputable/intractable phenomena cannot occur robustly in "reasonably constrained" systems.

Our contribution

Uncomputablity is not robust

Given a system T, we consider a small random perturbation T_ε of it.

Idea: x goes to $T(x)$ and then disperses randomly with distribution $p_{\varepsilon,T(x)}$. Where $p_{\varepsilon,x} \to \delta_x$ as $\varepsilon \to 0$.

Our contribution

Uncomputablity is not robust

Given a system T, we consider a small random perturbation T_ε of it.

Idea: x goes to $T(x)$ and then disperses randomly with distribution $p_{\varepsilon,T(x)}$. Where $p_{\varepsilon,x} \to \delta_x$ as $\varepsilon \to 0$.

Theorem A.(Braverman, Grigo, R.) Let T be a computable system over a compact subset X of $\mathbb{R}^d$. *Assume* $p_{\varepsilon,T(x)}$ *is uniform on the* ε*-ball around* $T(x)$. *Then, for almost every* $\varepsilon > 0$, *the ergodic measures of the perturbed system* T_ε *are all computable.*

Our contribution

Uncomputablity is not robust

Given a system T, we consider a small random perturbation T_ε of it.

Idea: x goes to $T(x)$ and then disperses randomly with distribution $p_{\varepsilon,T(x)}$. Where $p_{\varepsilon,x} \to \delta_x$ as $\varepsilon \to 0$.

Theorem A.(Braverman, Grigo, R.) Let T be a computable system over a compact subset X of $\mathbb{R}^d$. *Assume $p_{\varepsilon,T(x)}$ is uniform on the ε-ball around $T(x)$. Then, for almost every $\varepsilon > 0$, the ergodic measures of the perturbed system T_ε are all computable.*

Remarks:

- The noise does not need to be uniform, absolute continuity is enough.

Our contribution

Uncomputablity is not robust

Given a system T, we consider a small random perturbation T_ε of it.

Idea: x goes to $T(x)$ and then disperses randomly with distribution $p_{\varepsilon,T(x)}$. Where $p_{\varepsilon,x} \to \delta_x$ as $\varepsilon \to 0$.

Theorem A.(Braverman, Grigo, R.) Let T be a computable system over a compact subset X of $\mathbb{R}^d$. *Assume $p_{\varepsilon,T(x)}$ is uniform on the ε-ball around $T(x)$. Then, for almost every $\varepsilon > 0$, the ergodic measures of the perturbed system T_ε are all computable.*

Remarks:

- The noise does not need to be uniform, absolute continuity is enough.
- Intuitively, this says that the uncomputable phenomena is broken by the noise.

Our contribution

Intractability is not robust

Theorem B. (Braverman, Grigo, R.) *Suppose the perturbed system T_ε is uniquely ergodic and the function T is poly-time computable. Then there exists an algorithm $\mathcal{A}$ that computes μ with precision α in time $O_{T,\varepsilon}(poly(\frac{1}{\alpha}))$.*

Remarks:

Our contribution

Intractability is not robust

Theorem B. (Braverman, Grigo, R.) *Suppose the perturbed system* T_ε *is uniquely ergodic and the function* T *is poly-time computable. Then there exists an algorithm* $\mathcal{A}$ *that computes* μ *with precision* α *in time* $O_{T,\varepsilon}(poly(\frac{1}{\alpha}))$.

Remarks:

- The upper bound is exponential in the number of precision bits.

Our contribution

Intractability is not robust

Theorem B. (Braverman, Grigo, R.) *Suppose the perturbed system T_ε is uniquely ergodic and the function T is poly-time computable. Then there exists an algorithm $\mathcal{A}$ that computes μ with precision α in time $O_{T,\varepsilon}(poly(\frac{1}{\alpha}))$.*

Remarks:

- The upper bound is exponential in the number of precision bits.
- The algorithm can be implemented using $poly(log(\frac{1}{\alpha}))$ space.

Our contribution

Intractability is not robust

Theorem C. (Braverman, Grigo, R.) *If the noise "is nice" (is not a source of additional complexity), then the computation of* μ *at precision* $\alpha < O(\varepsilon)$ *requires time* $O_{T,\varepsilon}(poly(\log \frac{1}{\alpha}))$.

Remark:

Our contribution

Intractability is not robust

Theorem C. (Braverman, Grigo, R.) *If the noise "is nice" (is not a source of additional complexity), then the computation of* μ *at precision* $\alpha < O(\varepsilon)$ *requires time* $O_{T,\varepsilon}(poly(\log \frac{1}{\alpha}))$.

Remark:

- Intuition: at scales below the noise level, the "computationally simple" behavior takes over.

Some details about the results

Statistical behavior: Invariant and ergodic masures

A probability (Borel) measure over X is **invariant** if the probability of events do not change in time:

$$\mu(T^{-1}E) = \mu(E) \qquad \text{for every Borel set } E.$$

Some details about the results

Statistical behavior: Invariant and ergodic masures

A probability (Borel) measure over X is **invariant** if the probability of events do not change in time:

$$\mu(T^{-1}E) = \mu(E) \qquad \text{for every Borel set } E.$$

Invariant measures correspond to *equilibrium states* of the system. The **ergodic** measures are the ones that can not be *decomposed*:

$$\text{For every invariant set } E, \text{ either } \mu(E) = 1 \quad \textit{or} \quad \mu(E) = 1.$$

Statistical behavior

Small random perturbations

Here X is a space on which Lebesgue measure can be defined. Consider a family $\{p_x^\varepsilon\}_{x \in X} \in M(X)$ (a probability kernel) such that

$$p_x^\varepsilon \to \delta_x \text{ as } \varepsilon \to 0.$$

Statistical behavior

Small random perturbations

Here X is a space on which Lebesgue measure can be defined. Consider a family $\{p_x^\varepsilon\}_{x\in X} \in M(X)$ (a probability kernel) such that

$$p_x^\varepsilon \to \delta_x \text{ as } \varepsilon \to 0.$$

Definition

A **random perturbation of** T, T_ε is a Markov Chain X_n, $n = 0, 1, 2, \ldots$ with transition probabilities $P(A|x) = p^\varepsilon_{T(x)}(A)$. Given $\mu \in M(X)$, the **push forward** of μ under T_ε is defined by $(T_\varepsilon \mu)(A) = \int_X P(A|x)\, d\mu$.

Statistical behavior

Small random perturbations

Here X is a space on which Lebesgue measure can be defined. Consider a family $\{p_x^\varepsilon\}_{x \in X} \in M(X)$ (a probability kernel) such that

$$p_x^\varepsilon \to \delta_x \text{ as } \varepsilon \to 0.$$

Definition

A **random perturbation of** T, T_ε is a Markov Chain X_n, $n = 0, 1, 2, \dots$ with transition probabilities $P(A|x) = p^\varepsilon_{T(x)}(A)$. Given $\mu \in M(X)$, the **push forward** of μ under T_ε is defined by $(T_\varepsilon \mu)(A) = \int_X P(A|x)\, d\mu$.

Definition

A probability measure μ on X is called an **invariant measure of the random perturbation** T_ε **of** T if $T_\varepsilon \mu = \mu$.

The space of measures

Let M_{inv} denote the space of invariant probability measures.

- M_{inv} is a compact, convex, non empty set,
- The extremal points are the **ergodic** measures,
- if M_{inv} contains just one measure, then the system is called **uniquely ergodic**.

Which invariant measures are computable ?

Computability of probability measures

Let $M(X) := \{\text{Probability measures over } X\}$.

Computability of probability measures

Let $M(X) := \{\text{Probability measures over } X\}$.

- If X is separable and complete, then so is $M(X)$. And it can be metrized (Prokhorov distance ρ)

Computability of probability measures

Let $M(X) := \{\text{Probability measures over } X\}$.

- If X is separable and complete, then so is $M(X)$. And it can be metrized (Prokhorov distance ρ)
- Let $\mathcal{D} := \{\text{Finite convex combinations of Dirac measures}\}$.

Computability of probability measures

Let $M(X) :=$ {Probability measures over X}.

- If X is separable and complete, then so is $M(X)$. And it can be metrized (Prokhorov distance ρ)
- Let $\mathcal{D} :=$ {Finite convex combinations of Dirac measures}.

Proposition

The triple $(M(X), \mathcal{D}, \rho)$ *is a computable metric space.*

Computability of probability measures

Let $M(X) :=$ {Probability measures over X}.

- If X is separable and complete, then so is $M(X)$. And it can be metrized (Prokhorov distance ρ)
- Let $\mathcal{D} :=$ {Finite convex combinations of Dirac measures}.

Proposition

The triple $(M(X), \mathcal{D}, \rho)$ is a computable metric space.

... so we have *a* notion of computable measure to work with.

Computability of probability measures

A useful simple observation:

Computability of probability measures

A useful simple observation:

- The pushforward (or transition) operator $\mathcal{P} : \mu \to T_\varepsilon \mu$ is computable.

Computability of probability measures

A useful simple observation:

- The pushforward (or transition) operator $\mathcal{P} : \mu \to T_\varepsilon \mu$ is computable.
- If X is effectively compact, so is M_{inv}.

Computability of probability measures

A useful simple observation:

- The pushforward (or transition) operator $\mathcal{P} : \mu \to T_\varepsilon \mu$ is computable.
- If X is effectively compact, so is M_{inv}.
- It follows that uniquely ergodic systems have a computable invariant measure.

Main proof ideas

Proof of Theorem A

Theorem A. If $p_{\varepsilon,T(x)}$ *is uniform on the* ε*-ball around* $T(x)$. *Then, for almost every* $\varepsilon > 0$, *the (finitely many) ergodic measures of the perturbed system* T_ε *are all computable.*

Main proof ideas

Proof of Theorem A

Theorem A. If $p_{\varepsilon,T(x)}$ *is uniform on the* ε*-ball around* $T(x)$*. Then, for almost every* $\varepsilon > 0$*, the (finitely many) ergodic measures of the perturbed system* T_ε *are all computable.*

Remarks:

- The requirement of being uniform can be relaxed to absolute continuity.

Main proof ideas

Proof of Theorem A

Theorem A. If $p_{\varepsilon,T(x)}$ *is uniform on the* ε*-ball around* $T(x)$*. Then, for almost every* $\varepsilon > 0$*, the (finitely many) ergodic measures of the perturbed system* T_ε *are all computable.*

Remarks:

- The requirement of being uniform can be relaxed to absolute continuity.
- T_ε can have at most finitely many ergodic measures.

Main proof ideas

Proof of Theorem A

The result can essentially be obtained from the following observations:

Main proof ideas

Proof of Theorem A

The result can essentially be obtained from the following observations:

- For all but countably many $\varepsilon > 0$, there exists open sets $A_1, ..., A_{N(\varepsilon)}$ such that for all $i = 1, ..., N(\varepsilon)$:
  (i) $supp(\mu_i) \subset A_i$ and,

Main proof ideas

Proof of Theorem A

The result can essentially be obtained from the following observations:

- For all but countably many $\varepsilon > 0$, there exists open sets $A_1, \ldots, A_{N(\varepsilon)}$ such that for all $i = 1, \ldots, N(\varepsilon)$:
  (i) $supp(\mu_i) \subset A_i$ and,
  (ii) for every $x \in A_i$, $\mu_x = \mu_i$, where μ_x is the limiting distribution of T_ε starting at x.

Main proof ideas

Proof of Theorem A

The result can essentially be obtained from the following observations:

- For all but countably many $\varepsilon > 0$, there exists open sets $A_1, \ldots, A_{N(\varepsilon)}$ such that for all $i = 1, \ldots, N(\varepsilon)$:
  - (i) $supp(\mu_i) \subset A_i$ and,
  - (ii) for every $x \in A_i$, $\mu_x = \mu_i$, where μ_x is the limiting distribution of T_ε starting at x.
- We can "explore" the space to algorithmically find regions A_i like above.

Main proof ideas

Proof of Theorem A

The result can essentially be obtained from the following observations:

- For all but countably many $\varepsilon > 0$, there exists open sets $A_1, \dots, A_{N(\varepsilon)}$ such that for all $i = 1, \dots, N(\varepsilon)$:
  (i) $supp(\mu_i) \subset A_i$ and,
  (ii) for every $x \in A_i$, $\mu_x = \mu_i$, where μ_x is the limiting distribution of T_ε starting at x.
- We can "explore" the space to algorithmically find regions A_i like above.
- Restricted to each region, T_ε is uniquely ergodic. Computability of each measure now follows from compactness.

Main proof ideas

Proof of Theorem B

Theorem B. *Suppose the perturbed system T_ε is uniquely ergodic and the function T is polynomial-time computable. Then there exists an algorithm A that computes μ with precision α in time $O_{T,\varepsilon}(poly(\frac{1}{\alpha}))$.*

Remarks:

Main proof ideas

Proof of Theorem B

Theorem B. *Suppose the perturbed system T_ε is uniquely ergodic and the function T is polynomial-time computable. Then there exists an algorithm A that computes μ with precision α in time $O_{T,\varepsilon}(poly(\frac{1}{\alpha}))$.*

Remarks:

- The upper bound is exponential in the number of precision bits.

Main proof ideas

Proof of Theorem B

Theorem B. *Suppose the perturbed system T_ε is uniquely ergodic and the function T is polynomial-time computable. Then there exists an algorithm A that computes μ with precision α in time $O_{T,\varepsilon}(poly(\frac{1}{\alpha}))$.*

Remarks:

- The upper bound is exponential in the number of precision bits.
- Upon input α, the algorithm outputs a list $\{w_\mathfrak{a}\}_{\mathfrak{a}\in\zeta}$ of $poly(1/\alpha)$ dyadic numbers representing the piece-wise constant function

$$\mathcal{A}(\alpha) = \sum_{\mathfrak{a}\in\zeta} w_\mathfrak{a} \mathbf{1}\{x \in \mathfrak{a}\}$$

where P is a regular-size partition with $poly(1/\alpha)$ pieces.

Main proof ideas

Proof of Theorem B

Idea: exploit the mixing properties $\mathcal{P}$.

- Since $\mathcal{P}$ may not have a spectral gap, we construct a related transition operator $\overline{\mathcal{P}}$ that has the same invariant measure as $\mathcal{P}$ while having a a spectral gap.

Main proof ideas

Proof of Theorem B

Idea: exploit the mixing properties $\mathcal{P}$.

- Since $\mathcal{P}$ may not have a spectral gap, we construct a related transition operator $\overline{\mathcal{P}}$ that has the same invariant measure as $\mathcal{P}$ while having a a spectral gap.
- Compute a finite matrix approximation Q of $\overline{\mathcal{P}}$ s.t.:
  i) Q has a simple real eigenvalue near 1
  ii) the corresponding eigenvector ψ is nonegative and
  iii) the density associated to ψ is L^1-close to the stationary distribution of $\mathcal{P}$.

Main proof ideas

Proof of Theorem B

Idea: exploit the mixing properties $\mathcal{P}$.

- Since $\mathcal{P}$ may not have a spectral gap, we construct a related transition operator $\overline{\mathcal{P}}$ that has the same invariant measure as $\mathcal{P}$ while having a a spectral gap.
- Compute a finite matrix approximation Q of $\overline{\mathcal{P}}$ s.t.:
  i) Q has a simple real eigenvalue near 1
  ii) the corresponding eigenvector ψ is nonegative and
  iii) the density associated to ψ is L^1-close to the stationary distribution of $\mathcal{P}$.
- Q corresponds (roughly) to a piece-wise constant approximation of $\mathcal{P}$ on a finite partition ζ.
- Computing μ here means to have the vector ψ.

Main proof ideas

Proof of Theorem C

Theorem C. *Suppose the noise* $p^{\varepsilon}_{T(x)}(\cdot)$ *is "nice". Then the computation of* μ *at precision* $\delta < O(\varepsilon)$ *requires time* $O_{T,\varepsilon}(poly(\log \frac{1}{\delta}))$.

Main proof ideas

Proof of Theorem C

Theorem C. *Suppose the noise* $p^{\varepsilon}_{T(x)}(\cdot)$ *is "nice". Then the computation of* μ *at precision* $\delta < O(\varepsilon)$ *requires time* $O_{T,\varepsilon}(poly(\log \frac{1}{\delta}))$.

Remarks:

Main proof ideas

Proof of Theorem C

Theorem C. *Suppose the noise $p^{\varepsilon}_{T(x)}(\cdot)$ is "nice". Then the computation of μ at precision $\delta < O(\varepsilon)$ requires time $O_{T,\varepsilon}(poly(\log \frac{1}{\delta}))$.*

Remarks:

- Here we actually prove that μ has a poly-time computable analytic density. And therefore $\mu[0, x]$ is poly-time computable.

Main proof ideas

Proof of Theorem C

Theorem C. *Suppose the noise $p^{\varepsilon}_{T(x)}(\cdot)$ is "nice". Then the computation of μ at precision $\delta < O(\varepsilon)$ requires time $O_{T,\varepsilon}(poly(\log \frac{1}{\delta}))$.*

Remarks:

- Here we actually prove that μ has a poly-time computable analytic density. And therefore $\mu[0, x]$ is poly-time computable.
- The noise kernel $p^{\varepsilon}(y, x)$ is "nice" if there exists constants $C > 0$ and $\gamma > 0$ such that

$$|\partial_2^k p_{\varepsilon}(y, x)| \leq C\, k!\, e^{\gamma k} \qquad \text{for all } k \in \mathbb{N} \text{ and all } x, y \in X.$$

Main proof ideas

Proof of Theorem C

Theorem C. *Suppose the noise $p^{\varepsilon}_{T(x)}(\cdot)$ is "nice". Then the computation of μ at precision $\delta < O(\varepsilon)$ requires time $O_{T,\varepsilon}(poly(\log \frac{1}{\delta}))$.*

Remarks:

- Here we actually prove that μ has a poly-time computable analytic density. And therefore $\mu[0, x]$ is poly-time computable.
- The noise kernel $p^{\varepsilon}(y, x)$ is "nice" if there exists constants $C > 0$ and $\gamma > 0$ such that

$$|\partial_2^k p_{\varepsilon}(y, x)| \leq C\, k!\, e^{\gamma k} \qquad \text{for all } k \in \mathbb{N} \text{ and all } x, y \in X.$$

- Thus, if $\nu \in M(X)$, then the transition operator $\mathcal{P}$ is given by

$$P\nu(dx) = \rho(x)\, dx\ , \qquad \rho(x) = \int_X p_{\varepsilon}(T(y), x)\nu(dy)\ ,$$

Main proof ideas

Proof of Theorem C

Theorem C. *Suppose the noise $p^{\varepsilon}_{T(x)}(\cdot)$ is "nice". Then the computation of μ at precision $\delta < O(\varepsilon)$ requires time $O_{T,\varepsilon}(poly(\log \frac{1}{\delta}))$.*

Remarks:

- Here we actually prove that μ has a poly-time computable analytic density. And therefore $\mu[0,x]$ is poly-time computable.
- The noise kernel $p^{\varepsilon}(y,x)$ is "nice" if there exists constants $C > 0$ and $\gamma > 0$ such that

$$|\partial_2^k p_\varepsilon(y,x)| \leq C\, k!\, e^{\gamma k} \qquad \text{for all } k \in \mathbb{N} \text{ and all } x, y \in X.$$

- Thus, if $\nu \in M(X)$, then the transition operator $\mathcal{P}$ is given by

$$P\nu(dx) = \rho(x)\, dx\ , \qquad \rho(x) = \int_X p_\varepsilon(T(y),x)\nu(dy)\ ,$$

- In particular, $P\nu(dx)$ has a density for any probability measure ν.

Main proof ideas

Proof of Theorem C

Some observations on the previous proof:

Main proof ideas

Proof of Theorem C

Some observations on the previous proof:

- We approximated $\mathcal{P}$ by a finite matrix Q.

Main proof ideas

Proof of Theorem C

Some observations on the previous proof:

- We approximated $\mathcal{P}$ by a finite matrix Q.
- in order to increase the precision $\alpha = 2^{-n}$, we had to increase the resolution of ζ.

Main proof ideas

Proof of Theorem C

Some observations on the previous proof:

- We approximated $\mathcal{P}$ by a finite matrix Q.
- in order to increase the precision $\alpha = 2^{-n}$, we had to increase the resolution of ζ.
- The size of Q was exponential in n.

Main proof ideas

Proof of Theorem C

Some observations on the previous proof:

- We approximated $\mathcal{P}$ by a finite matrix Q.
- in order to increase the precision $\alpha = 2^{-n}$, we had to increase the resolution of ζ.
- The size of Q was exponential in n.

How to get rid of this exponential approximation?

Main proof ideas

Proof of Theorem C

Solution:

- We use a fixed partition ζ that depends only on the noise ($\operatorname{diam}\zeta < \frac{1}{e^{\gamma}}$).

Main proof ideas

Proof of Theorem C

Solution:

- We use a fixed partition ζ that depends only on the noise (diam$\zeta < \frac{1}{e^\gamma}$).
- Instead of the "piece-wise constant", we approximate $\mathcal{P}$ *exactly* on each $\mathfrak{a} \in \zeta$ by a Taylor series.

Main proof ideas

Proof of Theorem C

Solution:

- We use a fixed partition ζ that depends only on the noise ($\mathrm{diam}\zeta < \frac{1}{e^\gamma}$).
- Instead of the "piece-wise constant", we approximate $\mathcal{P}$ *exactly* on each $\mathfrak{a} \in \zeta$ by a Taylor series.
- The regularity of the kernel implies the regularity of $\mathcal{P}\rho$, for any initial density ρ.

Main proof ideas

Proof of Theorem C

Solution:

- We use a fixed partition ζ that depends only on the noise (diam$\zeta < \frac{1}{e^{\gamma}}$).
- Instead of the "piece-wise constant", we approximate $\mathcal{P}$ *exactly* on each $\mathfrak{a} \in \zeta$ by a Taylor series.
- The regularity of the kernel implies the regularity of $\mathcal{P}\rho$, for any initial density ρ.
- This provides an "infinite" matrix representation for $\mathcal{P}$, organized in a fixed number of blocks.

Main proof ideas

Proof of Theorem C

- We now can *truncate* the series representations and get a finite matrix P_N, corresponding to a finite approximation of $\mathcal{P}$.

Main proof ideas

Proof of Theorem C

- We now can *truncate* the series representations and get a finite matrix P_N, corresponding to a finite approximation of $\mathcal{P}$.
- P_N can be iterated.

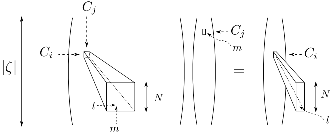


Figure: Graphical representation of the equation $P_N \, \rho_N^{(t)} = \rho_N^{(t+1)}$.

Main proof ideas

Proof of Theorem C

- We now can *truncate* the series representations and get a finite matrix P_N, corresponding to a finite approximation of $\mathcal{P}$.
- P_N can be iterated.

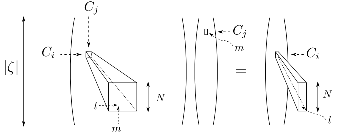


Figure: Graphical representation of the equation $P_N\,\rho_N^{(t)} = \rho_N^{(t+1)}$.

- The size of P_N depends **linearly** on the number n of precision bits !

Main proof ideas

Proof of Theorem C

- We now can *truncate* the series representations and get a finite matrix P_N, corresponding to a finite approximation of $\mathcal{P}$.
- P_N can be iterated.

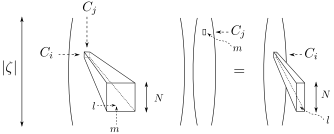


Figure: Graphical representation of the equation $P_N \, \rho_N^{(t)} = \rho_N^{(t+1)}$.

- The size of P_N depends **linearly** on the number n of precision bits !
- The invariant density π is computed by iterating $P_N \, \rho_N^{(t)}$ of any initial density ρ sufficiently many times (also linear in n) and then use the resulting vector and Taylor formula to compute $\pi(x)$.

Further work

How powerful can noisy systems be?

So... adding noise to the system may erase uncomputability (intractability).

Further work

How powerful can noisy systems be?

So... adding noise to the system may erase uncomputability (intractability).

- How much power does it retain?

Further work

How powerful can noisy systems be?

So... adding noise to the system may erase uncomputability (intractability).

- How much power does it retain?
- How much memory does it have after the addition of noise?

Further work

How powerful can noisy systems be?

So... adding noise to the system may erase uncomputability (intractability).

- How much power does it retain?
- How much memory does it have after the addition of noise?
- lower bounds? upper bounds?

Further work

How powerful can noisy systems be?

So... adding noise to the system may erase uncomputability (intractability).

- How much power does it retain?
- How much memory does it have after the addition of noise?
- lower bounds? upper bounds?
- The system has a limited amount of robustly distinguishable states...

Further work

How powerful can noisy systems be?

So… adding noise to the system may erase uncomputability (intractability).

- How much power does it retain?
- How much memory does it have after the addition of noise?
- lower bounds? upper bounds?
- The system has a limited amount of robustly distinguishable states…
- Hard to formalize.

THANKS !