The Complexity of Computing in Continuous Time: Space Complexity Is Precision

Manon Blanc
Institut Polytechnique de Paris, Ecole Polytechnique, LIX, 91128 Palaiseau Cedex, France
Université Paris-Saclay, LISN, 91190 Gif-sur-Yvette, France

Olivier Bournez
Institut Polytechnique de Paris, Ecole Polytechnique, LIX, 91128 Palaiseau Cedex, France

Abstract

Models of computations over the integers are equivalent from a computability and complexity theory point of view by the (effective) Church-Turing thesis. It is not possible to unify discrete-time models over the reals. The situation is unclear but simpler for continuous-time models, as there is a unifying mathematical model, provided by ordinary differential equations (ODEs). Each model corresponds to a particular class of ODEs. For example, the General Purpose Analog Computer model of Claude Shannon, introduced as a mathematical model of analogue machines (Differential Analyzers), is known to correspond to polynomial ODEs. However, the question of a robust complexity theory for such models and its relations to classical (discrete) computation theory is an old problem. There was some recent significant progress: it has been proved that (classical) time complexity corresponds to the length of the involved curves, i.e. to the length of the solutions of the corresponding polynomial ODEs. The question of whether there is a simple and robust way to measure space complexity remains. We argue that space complexity corresponds to precision and conversely.

Concretely, we propose and prove an algebraic characterisation of **FPSPACE**, using *continuous* ODEs. Recent papers proposed algebraic characterisations of polynomial-time and polynomial-space complexity classes over the reals, but with a discrete-time: those algebras rely on discrete ODE schemes. Here, we use classical (continuous) ODEs, with the classic definition of derivation and hence with the more natural context of continuous-time associated with ODEs. We characterise both the case of polynomial space functions over the integers and the reals. This is done by proving two inclusions. The first is obtained using some original polynomial space method for solving ODEs. For the other, we prove that Turing machines, with a proper representation of real numbers, can be simulated by continuous ODEs and not just discrete ODEs. A major consequence is that the associated space complexity is provably related to the numerical stability of involved schemas and the associated required precision. We obtain that a problem can be solved in polynomial space if and only if it can be simulated by some numerically stable ODE, using a polynomial precision.

2012 ACM Subject Classification Theory of computation → Models of computation; Theory of computation → Computability; Theory of computation → Complexity classes; Mathematics of computing → Ordinary differential equations

Keywords and phrases Models of computation, Ordinary differential equations, Real computations, Analog computations, Complexity theory, Implicit complexity, Recursion scheme

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.129

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Funding This work has been partially supported by ANR Project ∂IFFERENCE and Labex Digicosme.

1 Introduction

Recently, there has been a renewed interest in models of computations over the reals and their associated complexity classes. The fact that these models appear in complexity issues of deep learning models (a.k.a. neural networks) partially explains it. For example, various problems, such as the training of fully connected neural networks, have been proved to be a $\exists\mathbb{R}$-complete problem [2, 3]. Complexity classes like **FIXP** were introduced to discuss the complexity of continuous functions' fixed points in various contexts, such as game theory [32]. These classes and statements are related to discrete-time models of computation over the reals.

For discrete-time models of computations over the reals, the most famous approaches are computable analysis, based on the Turing machine model in [55] and [59] and algebraic models such as the Blum Shub Smale (BSS) model of computation [9, 8]. The class $\exists\mathbb{R}$ corresponds to the (constant-free, equivalently uniform) non-deterministic time of the BSS model of computation. Numerous decision problems were proved recently to be in this class. Both models were tailored for very different applications and it is well-known we cannot unify existing models with the equivalent of a Church-Turing thesis. For example, computable functions in a computable analysis model need to be continuous, while the BSS model intends to consider functions and problems over the polynomials that are not. It is also explained by the fact that some models have not been introduced with the idea of corresponding to actual physical machines but also to discuss abstract complexity (lower and upper bounds) for associated problems. Notice that some characterisations of complexity classes corresponding to **PSPACE** have been obtained [10, 11] in the BSS model.

Among models of computation over the reals, we can also distinguish continuous-time models. This includes models of old, first-ever-built computers, such as the Differential Analysers [57]. A famous mathematical model of such machines is the General Purpose Analog Computer model of Claude Shannon [53]. It covers many historical machines and today's analogue devices [56, 58] too. It also includes various recent approaches and models from deep learning such as Neural ODEs [26, 44] with many variants. In the context of continuous-time, the situation is clearer than with discrete-time models, as there is a unifying way to describe these models, provided by Ordinary Differential Equations (ODEs). Each model corresponds to a particular class of ODEs. For example, the GPAC corresponds to polynomial ODEs [37], and Neural ODEs are made by selecting the best solution among a parameterised class of ODEs: see, e.g. [44].

Even if particular classes of ODEs can describe such models, defining a robust and well-defined computation theory for continuous-time computations is not an easy problem: see [21] for the most recent survey. In short, the problem with time complexity is that considering the time variable as a measure of time is not robust: a curve can always be re-parameterised using a change of variable. The problem with space complexity is similar: reparameterisation corresponds to a change of time variable, but also of space-variable, introducing space and time contractions: See e.g. [21, 17]. Furthermore, many problems for simple dynamical systems are known to be undecidable, hence forbid **PSPACE**-completeness: see [38] and [39].

There was a recent breakthrough in [19, 17], where the authors relate time with the length of the solution curve of an ODE. Polynomial ODEs and their projections are known to cover a very wide class of functions, including all common functions or functions that can be built from them [35]. As the length of a curve is preserved under reparameterization, considering the length solves the issue of a possible change of variable. The authors prove that for polynomial ODEs, this is polynomially related to the time required to solve an ODE, hence providing a robust notion of time for ODEs. These statements and underlying constructions, which allow the simulation of Turing machines, led to solving several open problems: the existence of a universal ODE [20], the proof of the Turing-completeness of chemical reactions [33], or statements about the hardness of several dynamical systems problems [39].

The question of whether we can give a simple equivalent defining *space complexity* remains. We argue here that space complexity is polynomially related and conversely to the numerical stability of ODEs and their associated precision. We prove that a problem can be solved in polynomial space iff it can be simulated by some numerically stable ODE, using a polynomial precision. We prove this holds both for classical complexity over the discrete (functions over the integers) and also for space complexity for real functions in the model of computable analysis.

▶ Remark 1. In the literature, there are two possible definitions for **FPSPACE**, according to whether functions with non-polynomial size values are allowed or not. In this article, when we talk about **FPSPACE**, we always assume the outputs remain of polynomial size. Otherwise, the class is not closed by composition: the issue is about the usual convention of not counting the input and output as part of the total space used. Given f computable in polynomial space and g in logarithmic space, $f \circ g$ (and $g \circ f$) is computable in polynomial space. But, if exponential size output is allowed, this is not true: the issue is that if we assumed only f and g to be computable in polynomial space, the first might give an output of exponential size.

These questions of providing characterisations of classical complexity using ODEs can also be seen from the so-called "implicit complexity" point of view. Having "simple" characterisations of computability and complexity classes is useful for various fundamental and applied science fields. We are interested here in "algebraic" characterisations of those classes: we want to define them as the smallest set $[f_1, \dots, f_k; o_1, \dots, o_l]$ where the f_i are functions, closed under the operators o_j. For example, the set of computable functions over the integers is well-known to be: $[0, 1, \pi_k^i;$ *composition, minimisation, primitive recursion*]. Implicit complexity aims at giving similar algebras for classes of complexity theory: a reference survey is [27, 28]. The main benefit is to avoid the use of the framework of Turing machines, which is rather heavy and not necessarily well-known outside fundamental computer science. Several characterisations for **PTIME** over the integers were proposed. The first is due to Cobham in [29], but relies on explicit *ad hoc* bounds. Other approaches have then been proposed, see surveys [27, 28]. Recently, Bournez and Durand in [13] suggested an algebra using the so-called "linear-length" discrete ODEs. Instead of having explicit bounds, the linearity of the involved discrete ODE guarantees polynomial time complexity.

Using a similar approach, Blanc and Bournez in [4] and in [5] extended the constructions to a characterisation of **PTIME** for function over the reals. The latter extended the result to **PSPACE**, defining *robust* ODEs. However, those models rely on discrete ODEs (a.k.a. finite differences), which are discrete-time and less natural than continuous ODEs. We review all those results in Section 3.2.

This paper can be related to [19, 17]: the authors of these articles provide a characterisation of **PTIME** with *continuous* ODEs, establishing that time complexity corresponds to the length of the involved curve, i.e. the motto **time complexity = length**. Here, we get a motto of the form **space complexity = precision**.

Some of our constructions have similarities with the statements in [7]. In the later paper, various robustness concepts are introduced and it is proven that they lead to tractability. See the references in [7] for similar robustness statements. Robustness can also be associated with a dual motivation: the authors of [40] introduced a concept of robust undecidability, while here, we want a concept of robustness leading to tractability.

This is not the first time **FPSPACE** is characterised using continuous ODEs. However, the existing characterisation [34, 14] is obtained with complicated conditions on ODEs, while we have a simpler statement linking complexity to precision in a direct manner. Notice that the latter approach dealt with polynomial ODEs, while we do not restrict to polynomial ODEs. We obtain our statements by revisiting the approach of the latter papers but working over a compact domain and dealing with error correction more finely.

Intuitively, this can also be read as being in **PSPACE** for an ODE is consistent with having an attractor easily discretisable when there is one. We can also define the notion of robustness, as the insensitivity to "small" perturbations.

While discussing all these issues, we propose an algebraic characterisation of **PSPACE**, using *continuous* ODEs with the following algebra ($\mathbb{RCD}$ is for Robust Continuous Differential) (Schema *robust ODE* is formally defined in Definition 9):

$$\mathbb{RCD} = [0, 1, \pi_i^k, +, -, \times, \tanh, \cos, \pi, \frac{x}{2}, \frac{x}{3}; \textit{composition}, \textit{robust ODE}].$$

For a function $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^{d'}$ sending every integer $\mathbf{n} \in \mathbb{N}^d$ to the vicinity of some integer of $\mathbb{N}^d$, say at distance less than $1/4$, we write $\mathrm{DP}(f)$ for its discrete part: this is the function from $\mathbb{N}^d \to \mathbb{N}^{d'}$ mapping $\mathbf{n} \in \mathbb{N}^d$ to the integer rounding of $\mathbf{f}(n)$. For a class $\mathcal{C}$ of such functions, we write $\mathrm{DP}(\mathcal{C})$ for the class of the discrete parts of the functions of $\mathcal{C}$.

▶ **Theorem 2.** $\mathrm{DP}(\mathbb{RCD}) = \mathbf{FPSPACE} \cap \mathbb{N}^{\mathbb{N}}$.

We also provide a characterisation of functions over the reals computable in polynomial space, inspired by [5]. This is obtained by adding a limit schema *ELim* to $\mathbb{RCD}$. If we consider $\overline{\mathbb{RCD}} = [0, 1, \pi_i^k, +, -, \times, \tanh, \cos, \pi, \frac{x}{2}, \frac{x}{3}; \textit{composition}, \textit{robust ODE}, \textit{ELim}]$ then:

▶ **Theorem 3** (Generic functions over the reals). $\overline{\mathbb{RCD}} \cap \mathbb{R}^{\mathbb{R}} = \mathbf{FPSPACE} \cap \mathbb{R}^{\mathbb{R}}$
More generally: $\overline{\mathbb{RCD}} \cap \mathbb{R}^{\mathbb{N}^d \times \mathbb{R}^{d'}} = \mathbf{FPSPACE} \cap \mathbb{R}^{\mathbb{N}^d \times \mathbb{R}^d}$.

We organise the article as follows. In Section 2, we recall the concept of dynamical systems and discuss some associated complexity issues. We introduce the concept of robust ODE and prove that a robust ODE can be solved in polynomial space (Theorem 11). This is obtained, using an original method for solving ODE, optimising space, inspired by Savitch's theorem. This provides one direction of all the above theorems. The other direction is the object of the following sections, starting from Section 3. We first recall some previous results on discrete ODEs in Section 3. Using extensions of constructions from [5], we then prove that we can simulate a Turing machine using robust continuous ODEs in Section 4. This is obtained by simulating some discrete ODEs using continuous ODEs, dealing with error corrections, and using the fact that the functions are robust to a controlled error. The main result of Section 3 is Theorem 42. It states we can simulate Turing machines robustly with continuous ODEs when space remains polynomial. This theorem leads to the proof of Theorem 2 in Section 4. In Section 5, we prove Theorem 3. In Section 6, we conclude and discuss future works.

Some basic concepts

When we say that a function over the real is computable this is always in the sense of computable analysis: see e.g. [59, 46, 23]. A reference book for issues related to complexity theory in computable analysis is [46].

We assume some basic familiarities with dynamical systems. See [41] for a monography on the theory of dynamical systems from a mathematical point of view. Formally, a *discrete-time dynamical system* is given by a set D, called the *domain* and some (possibly partial) function $\mathbf{u}$ from D to D. A trajectory of the system is a sequence $\mathbf{f}(t)$ evolving according to $\mathbf{u}$: that is $\mathbf{f}(t+1) = \mathbf{u}(\mathbf{f}(t))$ for all t. A *continuous-time dynamical system* is given by a set $D \subseteq \mathbb{R}^d$ and some ODE of the form

$$\mathbf{f}' = \mathbf{u}(\mathbf{f}(t)) \tag{1}$$

on D. A trajectory starting from $\mathbf{f}_0$ is a solution of the associated Initial Value Problem (IVP), given by (1) and initial condition $\mathbf{f}(0) = \mathbf{f}_0$. A dynamical system can equivalently be described by its flow: $\Phi(\mathbf{f}_0, t)$ gives the position of the dynamics at time t, for an initial position $\mathbf{f}(0) = \mathbf{f}_0$. It satisfies the flow property

$$\Phi(\mathbf{f}_0, 0) = \mathbf{f}_0 \quad \Phi(\mathbf{f}_0, t + t') = \Phi(\Phi(\mathbf{f}_0, t), t'). \tag{2}$$

The dynamics or the flow can be parametrised by some $\mathbf{x}$: $\mathbf{u}$ is also some function of $\mathbf{x}$ and the flow function is $\Phi_{\mathbf{x}}(\mathbf{f}_0, t)$.

In the long run, dynamical systems may exhibit attractors. We refer to [48] for discussions of many possible ways of defining this concept, and to [51] for a characterisation of the hardness of computing attractors from a computable analysis point of view. Somehow, our coming results state that the uncomputability discussed in [51] is intrinsically due to the non-numerical stability of the considered dynamical systems there.

2 Dynamical systems and associated complexity issues

2.1 Some complexity results on graphs

We need to discuss the hardness of solving IVP, or equivalently of computing $\Phi(y, t)$. For pedagogical reasons, we first discuss the case of a simple setting, namely the case of a (deterministic) directed graph. Indeed, observe that a discrete-time dynamical system (D, u) can also be seen as a particular (deterministic) directed graph $G = (V, \to)$, where, in the general case, V is not necessarily finite: G corresponds to $V = D$ and $\to$ to the graph of the function u, i.e. $\mathbf{x}_t \to \mathbf{x}_{t+1}$ iff $\mathbf{x}_{t+1} = \mathbf{u}(\mathbf{x}_t)$. The obtained graph is deterministic because any vertex has an outdegree 1. Starting from some point $\mathbf{x}_0$, there is at most one possible path, and consequently, for a given time T, we can talk about its position at time t, i.e. $\Phi(\mathbf{x}_0, T)$ is Tth element of this path: (as usual in complexity theory, the length of some integer x is the length of its binary representation, denoted by $\ell(x)$).

▶ **Proposition 4** (The case of finite graphs). *Let $s(n) \geq \log(n)$ be space-constructible. Assume the vertices of $G = (V, \to)$ can be encoded in binary using words of length $s(n)$. Assume the relation $\to$ is decidable using a space polynomial in $s(n)$. Then,*

- *given the encoding of $\mathbf{u} \in V$ and of $\mathbf{v} \in V$, we can decide whether there is some path from $\mathbf{u}$ to $\mathbf{v}$, in a space polynomial in $s(n)$.*
- *given the encoding of $\mathbf{u} \in V$, and integer T in binary, we can compute $\Phi(\mathbf{u}, T)$, in a space polynomial in $s(n)$ and the length of T.*

The second item is even a characterisation of the complexity of the problem. Indeed, the converse is true: If, given the encoding of $\mathbf{u} \in V$, and integer T in binary, we can compute $\Phi(\mathbf{u}, T)$, in a space polynomial in $s(n)$ and the length of T, then as $\to$ is given by $\Phi(., 1)$, then $\to$ is decidable using a space polynomial in $s(n)$.

Proof. It is well-known that for finite graphs, given a directed graph $G = (V, \to)$ and some vertices $\mathbf{u}, \mathbf{v} \in V$, determine whether there is some path between $\mathbf{u}$ and $\mathbf{v}$ in G, denoted by $\mathbf{u} \xrightarrow{*} \mathbf{v}$ is in NLOGSPACE: the rough idea is to guess non-deterministically the intermediate nodes. The formal proof is detailed in [54]. The same algorithm, working over representations of vertices, when vertices are encoded using words of length $s(n)$ will work in NSPACE($s(n)$) (with the addition of the binary encoding of T if for the second item, if it bigger than $s(n)$). We then observe that NSPACE($s(n)$) = SPACE($s(n)$) from Savitch's theorem, recalled below. ◀

▶ **Theorem 5** (Savitch's theorem, [54, Theorem 8.5]). *For any space-constructible*[1] *function* $s : \mathbb{N} \to \mathbb{N}$ *with* $s(n) \geq \log n$*, we have* NSPACE($s(n)$) $\subseteq$ SPACE($s^2(n)$).

Recall that the key argument of the proof of Theorem 5 is to express the question as a recursive procedure (expressing reachability in less than 2^t steps, called $CANYIELD(\mathbf{x}, \mathbf{y}, t)$ in [54]) guaranteeing the required space complexity: we write that relation $CANYIELD(\mathbf{x}, \mathbf{y}, t)$ is relation $\mathbf{x} \to \mathbf{y}$ when $t = 1$, and is relation $\exists \mathbf{z}$ such that $CANYIELD(\mathbf{x}, \mathbf{z}, t/2)$ and $CANYIELD(\mathbf{z}, \mathbf{y}, t/2)$ otherwise. If one prefers, this can also be understood as "guessing" some intermediate node $\mathbf{z}$.

▶ Remark 6 (Attractor point of view). We presented the above statement in terms of computing the flow $\Phi(\mathbf{x}, T)$. This could alternatively be interpreted in terms of attractors. Indeed, when the above hypothesis holds, then the dynamics is captured by a graph. In the long run, in particular, if T is greater than the number of vertices, any trajectory loops (i.e. reaches an attractor). The above statement could then also be read as the fact that such an attractor is then polynomial space computable.

2.2 Solving efficiently ODEs: what is known

This idea leads to an original method for solving ODEs. At least, this is original for the numerical analysis literature, as far as we know. A recent survey about computability and complexity issues for solving ODEs is [39]. In short: First, it is important to distinguish the case where we want to solve the ODE on a bounded (hence a compact) domain, from the case of the full domain $\mathbb{R}$: in the latter case, we might ask questions about the evolution of the system on the long run, which is harder. Over a compact domain, it is known that there exists some polynomial-time computable function $u : [-1, 1] \times [0, 1] \to \mathbb{R}$ such that $f' = u(f, t)$ has no computable solution, even over $[0, \delta]$, for any $\delta > 0$: see [45, 50, 1]. The involved ODE has no unique solution. It is known over compact or non-compact domains that if unicity holds, then its solution is computable [30, 31, 52]. However, the complexity can be arbitrarily high [46, 47]. If we want to get to tractability, then some regularity hypotheses must be assumed. A classical hypothesis is to assume the ODE to be Lipschitz.

Over a compact domain, it has been observed in several references (see e.g. [46]) that a careful analysis of Euler's method proves that, if $u : B(0, 1) \times [0, 1] \to \mathbb{R}^n$, with $B(0, 1) \subseteq \mathbb{R}^n$, is a polynomial time computable (right-)Lipschitz function then any solution $f : [0, 1] \to B(0, 1)$ of $f' = u(f, t)$ must be **FPSPACE**: see the discussions around Theorem 3.2 in [39] with several references. Kawamura has proved in [42] that there exists a polynomial-time computable function $u : [-1, 1] \times [0, 1] \to \mathbb{R}$, which satisfies a Lipschitz condition, such that the unique solution $f : [0, 1] \to \mathbb{R}$ takes values in $[-1, 1]$ and computing it leads to a **PSPACE**-complete problem. Hence, the question of solving ODEs over a compact domain in polynomial time corresponds to the question **PTIME** = **PSPACE** [42], even for $\mathcal{C}^\infty$-functions [43].

[1] As proved in [54], this hypothesis can be avoided, at the price of a slightly more complicated proof.

However, all these results are over compact domains, and dealing with non-compact domains, i.e. in the long run, is harder. **PSPACE** membership is not true, as this is possible to simulate any Turing machine by some finite-dimensional polynomial ODE [38] over a non-compact domain. This led to many undecidability results for analytic, and even very simple ODEs: see e.g. [38]. A possible way to analyse efficiency is to analyse the complexity of the solution assuming a bound on the function's growth (i.e. using parameterised complexity). It was proved in [16] that one can solve a polynomial ODE in polynomial time assuming a bound on $\mathbf{Y}(T) = \max_{0 \leq t \leq T} \|\mathbf{f}(t)\|$. The result for polynomial ODEs was later improved in [49], where it is proved that the time T and parameter $\mathbf{Y}$ can be replaced by a single parameter, namely the length of the curve for polynomial ODEs. Furthermore, this parameter does not need to be given as input to the algorithm. This is a key argument for one direction of the motto "time complexity = length" we mentioned above. To get polynomial-time complexity over a non-compact domain, it is also mandatory not to use most classical methods from numerical analysis.

The same happens when discussing space complexity: a non-classical method is required to guarantee polynomial space complexity in the long run. No such method has yet been proposed, and this is the purpose of the coming subsection. Actually, for space complexity, in addition to all the problems mentioned, in all the above space or time analyses, the problem is that the complexity is (possibly implicitly) dependent on the Lipschitz constant or the length of the solution. In a system as simple as linear dynamics, the state at time T depends in Lipchitz way from the state at time 0, and the number of additional bits required to guarantee some precision 2^{-n} growth linearly with T. But the problem is that in a space polynomial in the input size, T has no reason to remain polynomial (consider, for example, a system simulating a Turing machine, as we will consider soon). Hence, the required precision is possibly exponential in the input size.

▶ Remark 7. The above comments can be interpreted informally as the fact that "most" (this could be "generic" in the sense of [51], i.e. (effective) descriptive theory) dynamical systems are intrinsically unstable, and an error method introduced at some step can make the method unavoidably incorrect in the long run unless we have a means to "guess" what will happen.

▶ Remark 8 (Attractor point of view). We presented the above statement in terms of computing the flow $\Phi(\mathbf{f}_0, T)$. But, this could alternatively be interpreted in terms of attractors. The point is that computing the attractors of a given dynamical system is hard in general, as this involves long-run behaviours. This explains all the undecidability results obtained in [51], even for very simple dynamics. However, as we will see, this is also explained by the fact that the latter paper discusses numerically unstable systems.

2.3 Solving efficiently ODEs: a space efficient method

This leads to an alternative approach to optimize space complexity: this can be seen as either using a non-deterministic algorithm that "guesses" the correct intermediate positions of the dynamics or, from the proof of Savitch's theorem approach, as an original recursive method to solve ODEs. As far as we know, we have never seen such a method discussed in the literature for solving ODEs.

Concretely: from the flow property, a strategy to compute $\Phi(\mathbf{f}_0, T)$ is either to use a particular numerical method if T is small, says smaller than $\Delta > 0$. Otherwise, we know that $\Phi(\mathbf{f}_0, T) = \Phi(\mathbf{z}, T/2)$, where $\mathbf{z} = \Phi(\mathbf{f}_0, T/2)$. This always holds, so if we can compute both quantities, we will solve the problem. The difficulty is that we cannot precisely compute $\mathbf{z}$ in practice, but some numerical approximation $\widetilde{\mathbf{z}}$. If the system is numerically stable, we may assume this strategy works. The case when this strategy will not work is if the trajectory starting from $\widetilde{\mathbf{z}}$, for the second half of the work from time $T/2$ to T, has a behaviour different from the one starting in $\mathbf{z}$: in other words, if there is a high instability somewhere, namely in $\mathbf{z}$.

This leads to the following concept: we write $\mathbf{a} =_\epsilon \mathbf{b}$ for $\|\mathbf{a}-\mathbf{b}\| \leq \epsilon$ for conciseness.

▶ **Definition 9** (Robust (continuous) ODE). *A function $\mathbf{f} : \mathbb{R}^{d+1} \to \mathbb{R}^{d'}$ is robustly ODE definable (from initial condition $\mathbf{g}$, and dynamic $\mathbf{u}$) if*

1. *it corresponds to the solution of the following continuous ODE:*

$$\mathbf{f}(0,\mathbf{x}) = \mathbf{g}(\mathbf{x}) \quad \text{and} \quad \frac{\partial \mathbf{f}(t,\mathbf{x})}{\partial t} = \mathbf{u}(\mathbf{f}(t,\mathbf{x}), t, \mathbf{x}), \tag{3}$$

2. *and there is some rational $\Delta > 0$, and some polynomial p such that the schema (3) is (polynomially) numerically stable on $[0,\Delta]$: for all integer n, considering $\eta(n) = p(n+\ell(\lceil \mathbf{x} \rceil))$ we can compute $\mathbf{f}(t,\mathbf{x})$ at precision 2^{-n} by working a precision $2^{-\eta(n)}$: if you consider any solution of $\tilde{\mathbf{x}} =_{2^{-\eta(n)}} \mathbf{x}$, $\tilde{\mathbf{f}}(0,\tilde{\mathbf{x}}) =_{2^{-\eta(n)}} \mathbf{g}(\mathbf{x})$ and $\frac{\partial \tilde{\mathbf{f}}(t,\tilde{\mathbf{x}})}{\partial t} =_{2^{-\eta(n)}} \mathbf{u}(\tilde{\mathbf{f}}(t,\tilde{\mathbf{x}}), t, \tilde{\mathbf{x}})$ then $\tilde{\mathbf{f}}(t,\tilde{\mathbf{x}}) =_{2^{-n}} \mathbf{f}(t,\mathbf{x})$ when $0 \leq t \leq \Delta$.*
3. *For $t \geq \Delta$, we can compute $\mathbf{f}(t,\mathbf{x})$ at precision 2^{-n} by computing some approximation $\widetilde{\mathbf{f}(t/2,\mathbf{x})}$ of $\mathbf{f}(t/2,\mathbf{x})$ at precision $2^{-\eta(n)}$, i.e. of $\Phi(\mathbf{g}(x), t/2)$, and then some approximation of $\Phi(\widetilde{\mathbf{f}(t/2,\mathbf{y})}, t/2)$, working at precision $2^{-\eta(n)}$.*

▶ Remark 10. For more clarity, and conciseness, we will assume in the proofs that $d = d' = 1$, as it can be easily extended to more general cases.

▶ **Theorem 11.** *Consider an IVP as in the previous definition. If $\mathbf{g}$ and $\mathbf{u}$ are computable in polynomial space, then the solution $\mathbf{f}$ can be computed in polynomial space.*

Proof. From definitions and above arguments, all bits of $\Phi(\mathbf{y}, t)$ can be computed non-deterministically with precision n (i.e. at 2^{-n}) using computations with precision $\eta(n)$, hence is in **NPSPACE** = **PSPACE**. From the argument of the proof of Savitch's theorem, this can also be turned into a deterministic polynomial space recursive algorithm. ◀

The above theorem is the key argument to obtain one direction of our main theorems. We now go in the reverse direction. This requires talking about discrete ODEs, and some previous constructions.

3 Discrete ODEs: some previous results and constructions

3.1 Preliminary

We will use the concept of discrete ODE defined as follows (notice that we will write $\frac{\delta \mathbf{f}}{\delta n}$ for discrete derivation, by opposition of the classical $\frac{\partial \mathbf{f}}{\partial n}$ to help to distinguish discrete vs continuous ODEs.)

▶ **Definition 12** (Discrete derivation, notation δ). *For $\mathbf{f} : \mathbb{N} \to \mathbb{R}^d \to \mathbb{R}^{d'}$, the* discrete derivation *of $\mathbf{f}$ is $\frac{\delta \mathbf{f}}{\delta n}(n,\mathbf{x}) = \mathbf{f}(n+1,\mathbf{x}) - \mathbf{f}(n,\mathbf{x})$.*

▶ Remark 13. We use the terminology "discrete ODE", as in [12, 13]. This concept has various names in other communities: this is also called *finite differences*, *difference equations*, sometimes *discretized ODE*, and the associated theory is sometimes called *discrete calculus*, *umbral calculus* in other communities . Sometimes, some of the statements seem to be rediscovered with other names, but as far as we know, the idea of computing with discrete ODEs can be associated with [12, 13], and we follow the terminology used there: we refer to the discussions and references in [13] for the references and some discussions about the various names used in literature for similar concepts.

3.2 Algebraic characterisation with discrete ODEs: state of the art

In this subsection, we review some of the results already obtained using discrete ODEs.

▶ Remark 14. Notice that we do not need any of these statements directly, even if we will sometimes reuse some of their constructions (and some of their ideas).

Characterising PTIME over the integers. The concept of derivation along the length was introduced in [12]. A characterisation of **FPTIME** for functions over the integers has then been obtained in [12]:

▶ **Theorem 15** (Functions over the integers [12]). *$\mathbb{LDL} \cap \mathbb{N}^{\mathbb{N}} = \mathbf{FPTIME} \cap \mathbb{N}^{\mathbb{N}}$, for $\mathbb{LDL} = [\mathbf{0}, \mathbf{1}, \pi_i^k, \ell(x), +, -, \times, sg(x)$; composition, linear length ODE$]$, with π_i^k the projection function, and $sg(x)$ is 0 for $x < 0$ and 1 for $x > 0$.*

Toward the real numbers: characterising real sequences. Later, we introduced in [4]

▶ **Definition 16** (Operation $ELim$). *Given $\tilde{\mathbf{f}} : \mathbb{R}^d \times \mathbb{N} \to \mathbb{R}^{d'} \in \mathbb{LDL}^{\bullet}$ such that for all $\mathbf{x} \in \mathbb{R}^d$, $n \in \mathbb{N}$, $\|\tilde{\mathbf{f}}(\mathbf{x}, 2^n) - \mathbf{f}(\mathbf{x})\| \leq 2^{-n}$ for some function $\mathbf{f}$, then $ELim(\tilde{\mathbf{f}})$ is the (uniquely defined) corresponding function $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^{d'}$.*

and then we considered the class

$$\overline{\mathbb{LDL}^{\bullet}} = [\mathbf{0}, \mathbf{1}, \pi_i^k, \ell(x), +, -, \times, \overline{\mathrm{cond}}(x), \frac{x}{2}; \textit{composition, linear length ODE, ELim}],$$

with $\overline{\mathrm{cond}}(x)$ a sigmoid valuing 0 when $x < \frac{1}{4}$ and 1 when $x > \frac{3}{4}$. We proved this provides a characterisation of functions from $\mathbb{N}$ to $\mathbb{R}$ computable in polynomial time.

▶ **Theorem 17** (Sequences of reals [4]). *$\overline{\mathbb{LDL}^{\bullet}} = \mathbf{FPTIME} \cap \mathbb{R}^{\mathbb{N}}$.*

Characterisation of PTIME and PSPACE for functions over the real with discrete ODEs. We later succeeded in obtaining a characterisation of functions over the real computable in polynomial time and even space.

▶ **Theorem 18** (**FPTIME**, Generic functions over the reals [5]).
$\overline{\mathbb{LDL}^{\circ}} \cap \mathbb{R}^{\mathbb{N}^d \times \mathbb{R}^{d'}} = \mathbf{FPTIME} \cap \mathbb{R}^{\mathbb{N}^d \times \mathbb{R}^d}$, with
$\overline{\mathbb{LDL}^{\circ}} = [\mathbf{0}, \mathbf{1}, \pi_i^k, \ell(x), +, -, \tanh, \frac{x}{2}, \frac{x}{3}; \textit{composition, linear length ODE, ELim}]$.

Consider the following schema:

▶ **Definition 19** (Robust Discrete ODE [5]). *A bounded function $\mathbf{f}$ is robustly ODE definable if:*

1. *it corresponds to the solution of the following discrete ODE:*

$$\mathbf{f}(0, \mathbf{x}) = \mathbf{g}(\mathbf{x}) \quad \textit{and} \quad \frac{\delta \mathbf{f}(t, \mathbf{x})}{\delta t} = \mathbf{u}(\mathbf{f}(t, \mathbf{x}), \mathbf{h}(t, \mathbf{x}), t, \mathbf{x}), \tag{4}$$

2. *where the schema* (4) *is (polynomially) numerically stable: there exists some polynomial* p *such that, for all integer* n*, writing* $\eta(n) = p(n + \ell(\mathbf{y}))$*, if you consider any solution of* $\tilde{\mathbf{y}} =_{2^{-\eta(n)}} \mathbf{y}$ *and* $\tilde{\mathbf{h}}(x, \tilde{\mathbf{y}}) =_{2^{-\eta(n)}} \mathbf{h}(x, \tilde{\mathbf{y}})$*, and* $\tilde{\mathbf{f}}(0, \tilde{\mathbf{y}}) =_{2^{-\eta(n)}} \mathbf{g}(\mathbf{y})$ *and* $\frac{\partial \tilde{\mathbf{f}}(x,\tilde{\mathbf{y}})}{\partial x} =_{2^{-\eta(n)}} \mathbf{u}(\tilde{\mathbf{f}}(x, \tilde{\mathbf{y}}), \tilde{\mathbf{h}}(x, \tilde{\mathbf{y}}), x, \tilde{\mathbf{y}})$ *then* $\tilde{\mathbf{f}}(x, \tilde{\mathbf{y}}) =_{2^{-n}} \mathbf{f}(x, \mathbf{y})$.

We recall the notion of *essential linearity*: The idea is to measure the degree, similarly to the classical notion of degree in polynomial expression, but considering all subterms that are within the scope of a tanh function contributes to 0 to the degree. Then, essential linearity corresponds to linearity with this concept of degree.

▶ **Definition 20** ([4]). *The degree* $\deg(x, P)$ *of a term* P *in* $x \in V$ *is defined inductively as follows:*

- $\deg(x, x) = 1$ *and for* $x' \in V \cup \mathbb{Z}$ *such that* $x' \neq x$, $\deg(x, x') = 0$*;*
- $\deg(x, P + Q) = \max\{\deg(x, P), \deg(x, Q)\}$*;*
- $\deg(x, P \times Q) = \deg(x, P) + \deg(x, Q)$*;*
- $\deg(x, \tanh(P)) = 0$*.*

A polynomial expression P *is essentially constant in* x *if* $\deg(x, P) = 0$*.*

A vectorial function (resp. a matrix or a vector) is said to be a polynomial expression if all its coordinates (resp. coefficients) are, and *essentially constant* if all its coefficients are.

▶ **Definition 21** ([4]). *A polynomial expression* $\mathbf{g}(\mathbf{f}(x, \mathbf{y}), \mathbf{h}(x, \mathbf{y}), x, \mathbf{y})$ *is essentially linear in* $\mathbf{f}(x, \mathbf{y})$ *if it is of the form:* $\mathbf{A}[\mathbf{f}(x, \mathbf{y}), \mathbf{h}(x, \mathbf{y}), x, \mathbf{y}] \cdot \mathbf{f}(x, \mathbf{y}) + \mathbf{B}[\mathbf{f}(x, \mathbf{y}), \mathbf{h}(x, \mathbf{y}), x, \mathbf{y}]$ *where* $\mathbf{A}$ *and* $\mathbf{B}$ *are polynomial expressions essentially constant in* $\mathbf{f}(x, \mathbf{y})$*.*

▶ Remark 22. A robust discrete ODE is said to be *linear* if $\mathbf{u}$ is essentially linear in $\mathbf{f}$ and $\mathbf{h}$.

Consider
$\overline{\mathbb{RLD}^{\circ}} = [\mathbf{0}, \mathbf{1}, \pi_i^k, \ell(x), +, -, \tanh, \frac{x}{2}, \frac{x}{3}; composition, robust\ linear\ ODE, ELim]$.

▶ **Theorem 23** (**FPSPACE**, Generic functions over the reals [5]).
$\overline{\mathbb{RLD}^{\circ}} \cap \mathbb{R}^{\mathbb{R}} = \mathbf{FPSPACE} \cap \mathbb{R}^{\mathbb{R}}$
More generally: $\overline{\mathbb{RLD}^{\circ}} \cap \mathbb{R}^{\mathbb{N}^d \times \mathbb{R}^{d'}} = \mathbf{FPSPACE} \cap \mathbb{R}^{\mathbb{N}^d \times \mathbb{R}^{d}}$.

Notice that previous classes mix functions with integer and real arguments. Furthermore, they all involve some various types of discrete ODEs. We need to avoid all these issues, as we consider only continuous ODEs.

3.3 Simulating a discrete ODE using a continuous ODE

We first prove that it is possible to simulate a discrete ODE with a continuous ODE. The underlying idea can be attributed to [22], and has been improved in many ways by several authors. We present here the basic ideas, reformulated in our context. A more precise analysis will come (Proposition 39).

▶ **Definition 24** ("Ideal iteration trick", [22]). *Consider the following initial value problem for a discrete ODE, given by functions* $\mathbf{g}$ *and* $\mathbf{u}$*:*

$$\begin{cases} \mathbf{f}(0, \mathbf{x}) = \mathbf{g}(\mathbf{x}) \\ \dfrac{\delta \mathbf{f}}{\delta t}(t, \mathbf{x}) = \mathbf{u}(\mathbf{f}(t, \mathbf{x}), t, \mathbf{x}) \end{cases} \tag{5}$$

Then, let $\mathbf{G}(\mathbf{v},t,\mathbf{x}) = \mathbf{u}(\mathbf{v},t,\mathbf{x}) + \mathbf{v}$*, and consider the (continuous) IVP:*

$$\begin{cases} \mathbf{y}_1(0,\mathbf{x}) = \mathbf{y}_2(0,\mathbf{x}) = \mathbf{g}(\mathbf{x}) \\ \mathbf{y}_1' = c(\mathbf{G}(r(\mathbf{y}_2), r(t), \mathbf{x}) - \mathbf{y}_1)^3 \theta(\sin(2\pi t)) \\ \mathbf{y}_2' = c(r(\mathbf{y}_1) - \mathbf{y}_2)^3 \theta(-\sin(2\pi t)) \end{cases} \tag{6}$$

where c *a constant,* $\theta(x) = 0$ *if* $x \leq 0$ *and* $\theta(x) > 0$ *if* $x > 0$*. We abusively write* $r(\mathbf{y})$ *for the application of function* $r : \mathbb{R} \to \mathbb{R}$ *componentwise on vector* $\mathbf{y}$*. Here,* r *is a rounding function: we mean, by construction,* G *preserves the integers, and* r *is a function that maps a real value close to some integer to this integer: assume, say, that for* $z \in [n - \frac{1}{4}, n + \frac{1}{2}]$*,* $r(z) = n$*, for any integer* $n \in \mathbb{Z}$*.*

▶ Remark 25. We do not need to specify what $r(z)$ values for a z not in such an interval: the following reasoning remains correct, whatever it is.

Then, the solution of continuous ODE (6) simulates in a continuous way the discrete ODE (5): Indeed, $\mathbf{y}_1$ corresponds to the actual computation of the iterates of $\mathbf{G}$ (and hence computes the successive values of $\mathbf{f}$) and $\mathbf{y}_2$ acts as a “memory” equation. Let us detail how it works.

▶ Remark 26. We describe here an “ideal” computation, as $\theta(x)$ is exacly 0 when $x \leq 0$, and $r(z)$ is exactly some integer on suitable domains. Later in the paper, we will deal with a not-so-ideal θ and r.

Initially, $\mathbf{f}(0,\mathbf{x}) = \mathbf{y}_1(0,\mathbf{x}) = \mathbf{y}_2(0,\mathbf{x}) = \mathbf{g}(\mathbf{x})$. For $t \in [0, 1/2]$, we have $\theta(-\sin(2\pi t)) = 0$, and hence $\mathbf{y}_2' = 0$, so $\mathbf{y}_2$ is fixed and kept at value $\mathbf{g}(\mathbf{x})$ for $t \in [0, \frac{1}{2}]$. Consequently, for $t \in [0, 1/2]$, $r(\mathbf{y}_2)$ is also fixed and kept at value $\mathbf{g}(\mathbf{x})$, and $r(t)$ is also fixed and kept at value 0. Consequently, on this interval, if we write $C(t) = c\theta(\sin(2\pi t))$, then the dynamics of $\mathbf{y}_1$ is given by

$$\mathbf{y}_1' = C(t)(\mathbf{G}(\mathbf{g}(\mathbf{x}), 0, \mathbf{x}) - \mathbf{y}_1)^3 \tag{7}$$

▶ **Lemma 27** (Analysis of ODE (7))**.** *The solution* $\mathbf{y}_1(t,\mathbf{x})$ *of ODE (7) is converging to* $G(\mathbf{g}(\mathbf{x}), 0, \mathbf{x})$ *for any initial condition. Furthermore, for any initial condition* $\mathbf{y}_1(0,\mathbf{x}) \neq G(\mathbf{g}(\mathbf{x}), 0, \mathbf{x})$*, we have* $\left\|\mathbf{y}_1(\frac{1}{2},\mathbf{x}) - \mathbf{G}(\mathbf{g}(x), 0, \mathbf{x})\right\| \leq \frac{\sqrt{2}}{2\sqrt{\int_0^{\frac{1}{2}} C(z)dz}}$*. In particular, for any* $m \in \mathbb{N}$*, we can select constant* c *such that for any initial condition* $\mathbf{y}_1(0,\mathbf{x})$*,* $\left\|\mathbf{y}_1(\frac{1}{2},\mathbf{x}) - \mathbf{G}(\mathbf{g}(x), 0, \mathbf{x})\right\| \leq 2^{-m}$*.*

Consequently, $\mathbf{y}_1(t,\mathbf{x})$ will approach $\mathbf{G}(\mathbf{g}(\mathbf{x}), 0, \mathbf{x}) = \mathbf{f}(1,\mathbf{x})$ on this interval. Thus, $\mathbf{y}_1(\frac{1}{2},\mathbf{x}) =_\epsilon \mathbf{f}(1,\mathbf{x})$ and $\mathbf{y}_2(\frac{1}{2},\mathbf{x}) = \mathbf{g}(\mathbf{x})$, for some $\epsilon > 0$, that we can consider less than $\frac{1}{4} = 2^{-2}$, by selecting a big enough constant c (just taking $m = 2$ above). At $t = \frac{1}{2}$, $\mathbf{y}_1$ will hence have simulated one step of discrete ODE (5).

Now, for $t \in [\frac{1}{2}, 1]$ the roles of $\mathbf{y}_1$ and $\mathbf{y}_2$ are exchanged : $\mathbf{y}_1'(t,\mathbf{x}) = 0$, so $\mathbf{y}_1$ is kept fixed, $\mathbf{y}_2$ approaches $r(\mathbf{y}_1) = \mathbf{f}(1,\mathbf{x})$, thus $\mathbf{y}_1(1,\mathbf{x}) =_\epsilon \mathbf{y}_2(1,x) =_\epsilon \mathbf{f}(1,\mathbf{x})$.

By induction, from the same reasoning, we obtain that, for all $n \in \mathbb{N}$, $\mathbf{y}_1(n,\mathbf{x}) =_\epsilon \mathbf{y}_2(n,\mathbf{x}) =_\epsilon \mathbf{f}(n,\mathbf{x})$, and actually, we also have $\mathbf{y}_1(t + \frac{1}{2},\mathbf{x}) =_\epsilon \mathbf{y}_2(t,\mathbf{x}) =_\epsilon \mathbf{f}(n,\mathbf{x})$ for all $t \in [n, n + \frac{1}{2}]$, for any integer n.

To implement such an ODE, we have to fix a function $\theta(x)$ with the above property. Taking $\mathrm{ReLU}(x) = \max(0,x)$ would satisfy it, but it is not a derivable function, and hence would not lead to a (classical) ODE. We could then take $\theta(x) = 0$ for $x \leq 0$, and $\exp(-1/x)$ for $x > 0$. The point is that such a function is not real analytic. The base functions we consider in our class $\mathbb{RCD}$ are all real analytic, and real analytic functions are preserved by composition, so we cannot get such a function by compositions from our base functions. Furthermore, it is known that a real analytic function that is constant on some interval (we assumed it is 0 for $x \leq 0$) is constant. Hence, the above-considered function $\theta(x)$ cannot be real analytic. So, implementing this trick cannot be done directly using our base functions, using only compositions.

In Proposition 39, we will do a similar construction, but dealing with errors and not exact functions $\theta(z)$ and $r(x)$. Furthermore, here the purpose of function r was to correct errors around integers, i.e. around $\mathbb{Z}$: this will be possibly around other $\mathbb{Z}\delta = \{n\delta | n \in \mathbb{Z}\}$ for some $\delta > 0$. We will then naturally assume that for $z \in [n\delta - \frac{1}{4}\delta, n\delta + \frac{1}{2}\delta]$, $r(z) = n$, for any integer $n \in \mathbb{Z}$, in order to have a similar reasoning as the one above where $\delta = 1$.

3.4 Encoding of Turing machines configurations

Our proofs rely on some constructions from [5]. Concretely, we need to simulate the execution of a Turing machine (TM) by some dynamical system over the reals. This requires to encode the configurations of a Turing machine into some real numbers. We recall some of the definitions and constructions from [5].

Consider a Turing machine defined by $\mathcal{M} = (\Sigma, Q, I, F, \delta)$, with Σ the working alphabet, Q the set of states, $I, F \subseteq Q$ respectively the sets of initial and final states, $\delta : Q \times \Sigma \to Q \times \Sigma \times \{\leftarrow, \rightarrow\}$ the transition function. For some practical reasons, similar to the ones in [5], we assume that the working alphabet is made of the symbols 1 and 3, and that the blank symbol is symbol 0.

We explicit the encoding we will use. We assume $Q = \{0, 1, \dots, |Q| - 1\}$. Let

$$\dots l_{-k} l_{-k+1} \dots l_{-1} l_0 r_0 r_1 \dots r_n \dots.$$

denote the content of the tape of the Turing machine M. In this representation, the head is in front of symbol r_0, and $l_i, r_i \in \{0, 1, 3\}$ for all i. Furthermore, we assume that there are no non-blank symbols between two blank symbols, i.e. that blank symbols, i.e. symbol 0, can only be eventually on the right, or eventually on the left. Such a configuration C can be denoted by $C = (q, l, r)$, where $l, r \in \Sigma^\omega$ are words over alphabet $\Sigma = \{0, 1, 3\}$ and $q \in Q$ denotes the internal state of M.

Now, write $\gamma_{word} : \Sigma^\omega \to \mathbb{R}$ for the function that maps a word $w = w_0 w_1 w_2 \dots$ to the dyadic (hence real) number $\gamma_{word}(w) = \sum_{n \geq 0} w_n 4^{-(n+1)}$.

The idea is that configuration C can also be encoded by some element $\overline{C} = (q, \overline{l}, \overline{r}) \in \mathbb{N} \times \mathbb{R}^2$, by considering $\overline{r} = \gamma_{word}(r)$ and $\overline{l} = \gamma_{word}(l)$. In other words, we encode the configuration of a bi-infinite tape Turing machine M by real numbers using their radix 4 encoding, but using only digits 1,3. Notice that this lives in $Q \times [0, 1]^2$. Denoting the image of $\gamma_{word} : \Sigma^\omega \to \mathbb{R}$ by $\mathcal{I}$, this even lives in $Q \times \mathcal{I}^2$.

In other words, we consider the following encodings:

$$\gamma_{config}(C) = (q, \bar{l}, \overline{r})$$

with

$$\begin{array}{rcl} \bar{l} & = & l_0 4^{-1} + l_{-1} 4^{-2} + \dots + l_{-k} 4^{-(k+1)} + \dots \\ \overline{r} & = & r_0 4^{-1} + r_1 4^{-2} + \dots + l_n 4^{-(n+1)} + \dots \end{array}$$

3.5 Revisiting some previous constructions

We denote by $\mathbb{RCD}_*$ the algebra $[0, 1, \pi_i^k, +, -, \times, \tanh, \cos, \pi, \frac{x}{2}, \frac{x}{3}; \textit{composition}]$. This is close to the class $\mathbb{LDL}^\circ = [\mathbf{0}, \mathbf{1}, \pi_i^k, \ell(x), +, -, \tanh, \frac{x}{2}, \frac{x}{3}; \textit{composition}, \textit{linear length ODE}]$, considered in [6, 5], but without the function $\ell(x)$, and without the possiblity of defining functions using linear length ODE (and with multiplication added).

We will reuse some of the construction from [5] (some corrections and more details can be found in [6]) but avoid systematically any use of linear length ODE and the length function $\ell(x)$. Furthermore, the class considered in [5] is mixing functions from the integers to the reals, and from the reals to the reals, and we need to keep only functions over the reals.

The following was stated in [5, Lemma 19].

▶ **Lemma 28.** *We denote by $Y(x, 2^{m+2})$ the function $\frac{1+\tanh(2^{m+2}x)}{2}$. For all integer m, for all $x \in \mathbb{R}$, $|\operatorname{ReLU}(x) - xY(x, 2^{m+2})| \leq 2^{-m}$, where $\operatorname{ReLU}(x) = \max(0, x)$.*

First, we observe that considering $Y(x, z) = \frac{1+\tanh(4xz)}{2}$ would yield a function in $\mathbb{RCD}_*$ with the same property: we avoid the computation of 2^m by a substitution of a variable, and using a multiplication. We then write $\operatorname{ReLU}\text{-}\mathfrak{s}(Y, x)$ for $xY(x, z)$: we have $|\operatorname{ReLU}\text{-}\mathfrak{s}(2^m, x) - \operatorname{ReLU}(x)| \leq 2^{-m}$.

In particular, this was used to prove we can uniformly approximate the continuous sigmoid functions (when $1/(b-a)$ is in $\mathbb{LDL}^\circ$) defined as $\mathfrak{s}(a, b, x) = 0$ whenever $w \leq a$, $\frac{x-a}{b-a}$ whenever $a \leq x \leq b$, and 1 whenever $b \leq x$. The above trick provides a new version of [5, Lemma 20].

▶ **Lemma 29** (Uniform approximation of any piecewise continuous sigmoid). *Assume $a, b, \frac{1}{b-a}$ is in $\mathbb{RCD}_*$. Then there is some function $\mathcal{C}\text{-}\mathfrak{s}(z, a, b, x) \in \mathbb{RCD}_*$ such that for all integer m,*

$$|\,\mathcal{C}\text{-}\mathfrak{s}(2^m, a, b, x) - \mathfrak{s}(a, b, x)| \leq 2^{-m}.$$

Proof. Take $\mathcal{C}\text{-}\mathfrak{s}(z, a, b, x) = \frac{(x-a)Y(x-a, z2^{1+c}) - (x-b)Y(x-b, z2^{1+c})}{b-a}$. observing that $(b - a)\,\mathfrak{s}(a, b, x) = \operatorname{ReLU}(x-a) - \operatorname{ReLU}(x-b)$. From triangle inequality, it will hold, choosing c with $\frac{1}{b-a} \leq 2^c$. ◀

The authors of [5] proved the existence of some function corresponding to a continuous (controlled) approximation of the fractional part function: we write by $\{.\}$ the fractional part function.

▶ **Theorem 30** ([5, Lemma 28]). *There exists some function $\xi : \mathbb{N}^2 \to \mathbb{R}$ in $\mathbb{LDL}^\circ$ such that for all $n, m \in \mathbb{N}$ and $x \in [-2^n, 2^n]$, whenever $x \in [\lfloor x \rfloor + \frac{1}{8}, \lfloor x \rfloor + \frac{7}{8}]$, $\left|\xi(2^m, 2^n, x) - \{x - \frac{1}{8}\}\right| \leq 2^{-m}$.*

We say that some real function is a real extension of a function over the integers if they coincide for integer arguments. It is not clear that we have an extension over the reals of ξ in our algebra $\mathbb{RCD}_*$, but if we add a real extension of such a function, from the proof of [5, Corollary 22], we obtain the bestiary of functions considered in [5, Corollary 22]: we write $\mathbb{RCD}_* + \xi$ for the algebra where some real extension of function ξ is added as a base function.

▶ **Corollary 31** (A bestiary of functions). *There exist*

1. $\xi_1, \xi_2 : \mathbb{N}^2 \times \mathbb{R} \mapsto \mathbb{R} \ \in \mathbb{RCD}_* + \xi$ *such that, for all $n, m \in \mathbb{N}$, $\lfloor x \rfloor \in [-2^n + 1, 2^n]$, whenever $x \in [\lfloor x \rfloor - \frac{1}{2}, \lfloor x \rfloor + \frac{1}{4}]$, $|\xi_1(2^m, 2^n, x) - \{x\}| \leq 2^{-m}$, and whenever $x \in [\lfloor x \rfloor, \lfloor x \rfloor + \frac{3}{4}]$, $|\xi_2(2^m, 2^n, x) - \{x\}| \leq 2^{-m}$.*
2. $\sigma_1, \sigma_2 : \mathbb{N}^2 \times \mathbb{R} \mapsto \mathbb{R} \ \in \mathbb{RCD}_* + \xi$ *such that, for all $n, m \in \mathbb{N}$, $\lfloor x \rfloor \in [-2^n + 1, 2^n]$, whenever $x \in [\lfloor x \rfloor - \frac{1}{2}, \lfloor x \rfloor + \frac{1}{4}]$, $|\sigma_1(2^m, 2^n, x) - \lfloor x \rfloor| \leq 2^{-m}$, and whenever $x \in I_2 = [\lfloor x \rfloor, \lfloor x \rfloor + \frac{3}{4}]$, $|\sigma_2(2^m, 2^n, x) - \lfloor x \rfloor| \leq 2^{-m}$.*

3. $\lambda : \mathbb{N}^2 \times \mathbb{R} \mapsto [0,1] \in \mathbb{RCD}_* + \xi$ *such that for all* $m, n \in \mathbb{N}$, $\lfloor x \rfloor \in [-2^n+1, 2^n]$, *whenever* $x \in [\lfloor x \rfloor + \frac{1}{4}, \lfloor x \rfloor + \frac{1}{2}]$, $|\lambda(2^m, 2^n, x) - 0| \ \leq 2^{-m}$, *and whenever* $x \in [\lfloor x \rfloor + \frac{3}{4}, \lfloor x \rfloor + 1]$, $|\lambda(2^m, 2^n, x) - 1| \leq 2^{-m}$.
4. $\mod_2 : \mathbb{N}^2 \times \mathbb{R} \mapsto [0,1] \in \mathbb{RCD}_* + \xi$ *such that for all* $m, n \in \mathbb{N}$, $\lfloor x \rfloor \in [-2^n+1, 2^n]$, *whenever* $x \in [\lfloor x \rfloor - \frac{1}{4}, \lfloor x \rfloor + \frac{1}{4}]$, $| \mod_2(2^m, 2^n, x)$-$\lfloor x \rfloor \mod 2| \leq 2^{-m}$.
5. $\div_2 : \mathbb{N}^2 \times \mathbb{R} \mapsto [0,1] \in \mathbb{RCD}_* + \xi$ *such that for all* $m, n \in \mathbb{N}$, $\lfloor x \rfloor \in [-2^n+1, 2^n]$, *whenever* $x \in [\lfloor x \rfloor - \frac{1}{4}, \lfloor x \rfloor + \frac{1}{4}]$, $| \div_2 (2^m, 2^n, x) - \lfloor x \rfloor // 2| \leq 2^{-m}$, *with* $//$ *the integer division.*

Similarly, the equivalent of [5, Lemmas 23,24 and 25] still hold in $\mathbb{RCD}_* + \xi$:

▶ **Lemma 32.** *There exists* $\mathcal{C}$-if $\in \mathbb{RCD}_* + \xi$ *such that,* $l \in [0,1]$, *if we take* $|d' - 0| \leq 1/4$, *then* $|\mathcal{C}\text{-if}(2^m, d', l) - 0| \leq 2^{-m}$, *and if we take* $|d' - 1| \leq 1/4$, *then* $|\mathcal{C}\text{-if}(2^m, d', l) - l| \leq 2^{-m}$.

▶ **Lemma 33.** *Let* $\alpha_1, \alpha_2, \dots, \alpha_n$ *be some integers, and* $V_1, V_2, \dots, V_n$ *some constants. We write* $\mathrm{send}(\alpha_i \mapsto V_i)_{i \in \{1,\dots,n\}}$ *for the function that maps any* $x \in [\alpha_i - 1/4, \alpha_i + 1/4]$ *to* V_i, *for all* $i \in \{1, \dots, n\}$.

There is some function in $\mathbb{RCD}_* + \xi$, *that we write* $\mathcal{C}\text{-send}(2^m, \alpha_i \mapsto V_i)_{i \in \{1,\dots,n\}}$, *that maps any* $x \in [\alpha_i - 1/4, \alpha_i + 1/4]$ *to a real at distance at most* 2^{-m} *of* V_i, *for all* $i \in \{1, \dots, n\}$.

▶ **Lemma 34.** *Let* N *be some integer. Let* $\alpha_1, \alpha_2, \dots, \alpha_n$ *be some integers, and* $V_{i,j}$ *for* $1 \leq i \leq n$ *some constants, with* $0 \leq j < N$. *We write* $\mathrm{send}((\alpha_i, j) \mapsto V_{i,j})_{i \in \{1,\dots,n\}, j \in \{0,\dots,N-1\}}$ *for the function that maps any* $x \in [\alpha_i - 1/4, \alpha_i + 1/4]$ *and* $y \in [j - 1/4, j + 1/4]$ *to* $V_{i,j}$, *for all* $i \in \{1, \dots, n\}$, $j \in \{0, \dots, N-1\}$.

There is some function in $\mathbb{RCD}_* + \xi$, *that we write* $\mathcal{C}\text{-send}(2^m, (\alpha_i, j) \mapsto V_{i,j})_{i \in \{1,\dots,n\}, j \in \{0,\dots,N-1\}}$, *that maps any* $x \in [\alpha_i - 1/4, \alpha_i + 1/4]$ *and* $y \in [j - 1/4, j + 1/4]$ *to a real at distance at most* 2^{-m} *of* $V_{i,j}$, *for all* $i \in \{1, \dots, n\}$, $j \in \{0, \dots, N-1\}$.

Working with one step of a Turing machine

As the proof of [5, Lemmas 30] is done using all the functions provided by these lemmas, we obtain:

▶ **Lemma 35.** *We can construct some function* $\overline{Next}$ *in* $\mathbb{RCD}_* + \xi$ *that simulates one step of* M, *i.e. computing the Next function sending a configuration* $\overline{C}$ *of Turing machine* M *to* $\overline{C}'$, *where* C' *is the next one:* $\|Next(2^m, 2^S, \overline{C}) - \overline{C}'\| \leq 2^{-m}$. *Furthermore, it is robust to errors on its input, up to space* S: *considering* $\|\tilde{C} - \overline{C}\| \leq 4^{-(S+2)}$, $\|Next(2^m, 2^S, \tilde{C}) - \overline{C}'\| \leq 2^{-m}$ *remains true.*

Converting integers an dyadics to words and conversely

The article [5] also defined some functions for converting integers and dyadics to their encoding as words, and conversely. Namely, the following encoding is considered: every digit in the binary expansion of dyadic d is encoded by a pair of symbols in the radix 4 expansion of $\overline{d} \in \mathcal{I} \cap [0,1]$: digit 0 (respectively: 1) is encoded by 11 (resp. 13) if before the “decimal” point in d, and digit 0 (respectively: 1) is encoded by 31 (resp. 33) if after. For example, for $d = 101.1$ in base 2, $\overline{d} = 0.13111333$ in base 4. Conversely, given $\overline{d}$, the article provided a way to construct d. This corresponds to [5, Lemmas 33 and 34]:

▶ **Lemma 36** (From $\mathbb{N}$ to $\mathcal{I}$)**.** *We can construct some function* $Decode : \mathbb{N}^2 \to \mathbb{R}$ *in* $\mathbb{LDL}^\circ$ *that maps* m *and* n *to some point at distance less than* 2^{-m} *from* $\gamma_{word}(\overline{n})$.

▶ **Lemma 37** (From $\mathcal{I}$ to $\mathbb{R}$, and multiplying in parallel)**.** *We can construct some function $EncodeMul : \mathbb{N}^2 \times [0,1] \times \mathbb{R} \to \mathbb{R}$ in $\mathbb{LDL}^\circ$ that maps m, 2^S, $\gamma_{word}(\overline{d})$ and (bounded) λ to some real at distance at most 2^{-m} from λd, whenever $\overline{d}$ is of length less than S.*

As for ξ, it is not clear that we have some real extensions of these functions in $\mathbb{RCD}_*$: we write $\mathbb{RCD}_* + \xi + Decode + Encode$ for the algebra where some real extension of these functions is added as a base function.

3.6 Constructing the missing functions

We need a way to construct some substitute of "missing functions" (ξ, *Decode* and *EncodeMul*). As all of them are defined using discrete ODEs, an idea is to use a continuous ODE to simulate the respective discrete ODEs: we hence revisit the construction of the ideal iteration trick of Section 3.3, dealing with errors and not exact functions $\theta(z)$ and $r(x)$.

The key is to revisit Lemma 27, and do a more detailed analysis of possible involved errors in dynamics of the form (7). This equation has been studied by various authors in several articles, including [24, 25, 36, 18, 34]. We use the following statement from [34, Lemma 4.5], [14, Lemma 5.2], obtained basically by a case analysis of error propagations in Lemma 27.

▶ **Lemma 38** (Improved error analysis of ODE (7), [34, Lemma 4.5] [14, 5.2])**.** *Consider a point $b \in \mathbb{R}$, some $\gamma > 0$ some reals $t_0 < t_1$, and a function $\phi : \mathbb{R} \to \mathbb{R}$ with the property that $\phi(t) \geq 0$ for all $t \geq t_0$ and $\int_{t_0}^{t_1} \phi(t)dt > 0$. Let $\rho, \delta \geq 0$ and let $\bar{b}, E : \mathbb{R} \to \mathbb{R}$ be functions such that that $|\bar{b}(t) - b| \leq \rho$ and $|E(t)| \leq \delta$ for all $t \geq t_0$. Then the IVP defined by*

$$z' = c(\bar{b}(t) - z)^3 \phi(t) + E(t)$$

with the initial condition $z(t_0) = \bar{z}_0$, where $\gamma > 0$ and $c \geq \frac{1}{2\gamma^2 \int_{t_0}^{t_1} \phi(t)dt}$ satisfies

1. *$|z(t_1) - b| < \rho + \gamma + \delta(t_1 - t_0)$, independently of the initial condition $\bar{z}_0 \in \mathbb{R}$*
2. *$\min(\bar{z}_0, b - \rho) - \delta(t_1 - t_0) \leq z(t) \leq \max(\bar{z}_0, b + \rho) + \delta(t_1 - t_0)$ for all $t \in [t_0, t_1]$.*

▶ **Proposition 39** (Simulating a discrete ODE by a continuous ODE)**.** *Assume $\mathbf{G}$ is almost constant around $\mathbb{N}\delta$ and r is a rounding function around $\mathbb{N}\delta$ for some $\delta > 0$: for $z \in [n\delta - \frac{1}{4}\delta, n\delta + \frac{1}{2}\delta]$, $r(z) = n$, for any integer $n \in \mathbb{Z}$.*

Suppose that, in (6), we replace function $\theta(z)$ and function $r(z)$ by some suitable approximations: we take $\theta(x) = \mathrm{ReLU}(x)$, $\theta_{\epsilon'}(x)$, $r_{\epsilon'(z)}$ such that $\theta(z) =_{\epsilon'} \theta(z)$, and $r_{\epsilon'}(x) =_{\epsilon'}$, and take constant c big enough. Then the solution of the obtained ODE will continuously simulate the discrete ODE (5), with the same bounds as in the analysis in Section 3.3, i.e. with error at most ϵ if ϵ' is taken sufficiently small. To guarantee $\epsilon = 2^{-n}$, it is sufficient to take $\epsilon' = 2^{-p(n)}$ and $\theta_{\epsilon'}(x) = \mathrm{ReLU}\text{-}\mathfrak{s}(2^{p(n)}, x)$ for some polynomial p.

Proof. The key is that the involved errors additively propagate, from Lemma 38. Namely, they are in $\mathcal{O}(\epsilon')$, but they are then corrected from the reasoning in Section 3.3: rounding function corrects errors or order ϵ whenever its argument is at a distance less than $1/4\delta$ of some $n\delta$ exactly as in the reasoning in Section 3.3 (where $\delta = 1$, even if now it introduces some error ϵ' at every step; but the latter is corrected at the next step). Observe that the involved constant c, is of order 2^n.

More formally, we claim that for all $n \in \mathbb{N}$, $\mathbf{y}_1(n, \mathbf{x}) =_\epsilon \mathbf{y}_2(n, \mathbf{x}) =_\epsilon \mathbf{f}(n, \mathbf{x})$, and $\mathbf{y}_1(t + \frac{1}{2}, \mathbf{x}) =_\epsilon \mathbf{y}_2(t, \mathbf{x}) =_\epsilon \mathbf{f}(n, \mathbf{x})$ for all $t \in [n, n + \frac{1}{2}]$.

For $n = 0$, initially $\mathbf{f}(0, \mathbf{x}) = \mathbf{y}_1(0, \mathbf{x}) = \mathbf{y}_2(0, \mathbf{x}) = \mathbf{g}(\mathbf{x})$. For $t \in [n, n + 1/2]$, we then have $\theta(-\sin(2\pi t)) =_{\epsilon'} 0$, and hence $\mathbf{y}_2' =_{\epsilon'} 0$, so $\mathbf{y}_2$ is kept close to value $\mathbf{g}(\mathbf{x})$ for $t \in [0, \frac{1}{2}]$, with an error less than $\frac{1}{2}\epsilon'$.

Consequently, for $t \in [0, 1/2]$, $r(\mathbf{y}_2)$ is kept close to a constant value $\mathbf{g}(\mathbf{x})$, when an error less than ϵ', if we choose $\epsilon' < \frac{1}{4}\delta$. Meanwhile, $r(t)$ is also at a value close to n with an error lesser than ϵ'.

Consequently, on this interval, if we write $C(t) = c\theta(\sin(2\pi t))$, then the dynamics of $\mathbf{y}_1$ is given by a dynamic of the form of Lemma 38. This lemma states that $\mathbf{y}_1(t, \mathbf{x})$ will approach $\mathbf{G}(\mathbf{g}(\mathbf{x}), 0, \mathbf{x}) = \mathbf{f}(1, \mathbf{x})$ on this interval, with an error of order $\epsilon' + \epsilon' + \frac{1}{2}\epsilon'$.

Here the hypothesis that $\mathbf{G}$ is almost constant around $\mathbb{N}\delta$ means that its value is guaranteed to be at ϵ' from $\mathbf{G}(\mathbf{g}(\mathbf{x}), 0, \mathbf{x})$ on the interval.

Thus, $\mathbf{y}_1(\frac{1}{2}, \mathbf{x}) =_{\epsilon/2} \mathbf{f}(1, \mathbf{x})$, if we choose $\frac{5}{2}\epsilon' < \epsilon/2$. At $t = n + \frac{1}{2}$, $\mathbf{y}_1$ will hence have simulated one step of discrete ODE (5), with an error less than $\epsilon/2$, and $\mathbf{y}_2$ will be close to $\mathbf{g}(\mathbf{x})$ with an error less than $\epsilon' < \epsilon/2$.

Now, for $t \in [n + \frac{1}{2}, n + 1]$ the roles of $\mathbf{y}_1$ and $\mathbf{y}_2$ are exchanged: $\mathbf{y}_1'(t, \mathbf{x}) =_{\epsilon'} 0$, so $\mathbf{y}_1$ is kept almost fixed, with a new error less than $\frac{1}{2}\epsilon'$. In the same time $\mathbf{y}_2$ approaches $r(\mathbf{y}_1) = \mathbf{f}(1, \mathbf{x})$ by Lemma 38, with some new error of order less than $\frac{5}{2}\epsilon' < \epsilon/2$.

Consequently, we get the property at rank $n + 1$. ◀

▶ Remark 40. Observe that, somehow, the idea is that the constructions always replace every function with a function that does not change much locally (i.e. changes in a controlled way). This is the key that provides a robust ODE as in Definition 9, leading to polynomial space complexity by Theorem 11.

In other words, whenever we have some discrete ODE as in (5) defining some function $\mathbf{f}(t, \mathbf{x})$, we can construct some continuous ODE, using only functions from $\mathbb{RCD}_*$, such that one of its projection provides a function $\mathbf{f}(z, t, \mathbf{x})$, with the guarantee $\mathbf{f}(2^n, t, \mathbf{x})$ is 2^{-n} close to $\mathbf{f}(n, \mathbf{x})$, whenever t is close (at a distance less than $1/4$) to some integer n.

This works, as we can obtain such a $r_{\epsilon'}(x)$ from the functions from Corollary 31: Consider $r(x, 2^m) = \sigma_2(2^m, 2^n, x + \frac{1}{4})$ that works over $\lfloor x \rfloor \in [-2^n + 1, 2^n]$, and observe that this is sufficient to apply the trick for the required functions, from the form of the considered discrete ODE in [5].

Except that we have a bootstrap problem: ξ was defined using a discrete ODE in [5], and as the functions from Corollary 31 are defined above using ξ, we cannot apply this reasoning to get function ξ. But the point is that for the special case of ξ, it is easy to construct a function in $\mathbb{RCD}$ that corresponds to some real extension of ξ, as we have functions such as $\sin(x) = \cos(\frac{\pi}{2} - x)$ and π.

▶ **Lemma 41.** *Function ξ has some real extension in $\mathbb{RCD}_*$.*

Proof. If we succeed to obtain a function $i(2^m, 2^n, x)$ that values $\lfloor x \rfloor$ whenever $x \in [\lfloor x \rfloor, \lfloor x \rfloor + \frac{3}{4}]$, we are done, as we can then obtain $\xi(2^m, 2^n, x)$ by considering $\xi(2^m, 2^n, x) = x + \frac{7}{8} - i(2^m, 2^n, x + \frac{7}{8})$.

A possible solution is then the following: consider function $R_e(x) := \mathfrak{s}(x, 0, e/2)$, and then $t_e(x) = (1 - R_e(\sin(2\pi x)))((1 - R_e(\sin(4\pi x)))$. If we put aside some interval of width $e/2$ around $\frac{1}{2}$ and $\frac{7}{8}$ where it takes values in $[0, 1]$, it values 0 on $[\lfloor x \rfloor, \lfloor x \rfloor + \frac{7}{8}]$, and then 1 on $[\lfloor x \rfloor + \frac{7}{8}, \lfloor x \rfloor + 1]$. We can then consider $I_e(t) = 8 \int_0^t t_e(x) dx$ (i.e. the solution of ODE $l_e' = 8t_e$), and then $i(t) =_{e.t} l_e(t)$. It is then sufficient to replace $\mathfrak{s}$ by $\mathcal{C}$-$\mathfrak{s}$, in the above expressions, in order to control the error and make it smaller than 2^{-m}. ◀

Consequently, this is true that we can substitute a discrete ODE with a continuous ODE for the required functions *Decode* and *EncodeMul*: just replace ξ in the involved schemas by the above function. Notice that we can also easily get a real extension of the function that maps n to 2^n.

3.7 Working with all steps of a Turing machines

We can then go from one step of a Turing machine, to arbitrarily many steps. We are following the idea of [5], but replacing discrete ODEs with continuous ODEs.

▶ **Theorem 42.** *Consider some Turing machine M that computes some function $f : \Sigma^* \to \Sigma^*$ in some polynomial space $S(\ell(\omega))$ on input ω. One can construct some function $\tilde{f} : \mathbb{N}^2 \times \mathbb{R} \to \mathbb{R}$ in $\mathbb{RCD}$ that does the same: we have $\tilde{f}(2^m, 2^{S(\ell(\omega))}, \gamma_{word}(\omega))$ that is at most 2^{-m} far from $\gamma_{word}(f(\omega))$.*

Proof. We denote by $\mathcal{M}$ the Turing machine computing f. Similarly to the arguments in [5], we can state that there exists a function $Exec$ solution of a robust linear discrete ODE (E) that "computes" the execution of $\mathcal{M}$, with C_{init} the initial configuration :

$$(E) : \begin{cases} Exec(2^m, 0, 2^S, C_{init}) = C_{init} \\ \dfrac{\delta Exec(2^m, t, 2^S, C_{init})}{\delta t} = Next(2^m, 2^S, Exec(2^m, t, 2^S, C_{init})) - Exec(2^m, t, 2^S, C_{init}). \end{cases}$$

For any configuration $\overline{C}$ of $\mathcal{M}$, let write $F(\overline{C})) = F(2^m, 2^S, \overline{C}) = Next(2^m, 2^S, \overline{C}) + \overline{C}$, associated to the righthand side of the above discrete ODE. Denoting by $\tilde{C}$ the errorless encoding of the configuration C, from the constructions of [5] (Lemma 35), it is true that if $\left|\overline{C} - \tilde{C}\right| \leq 4^{-(S+2)}$, then $\left|F(\overline{C}) - F(\tilde{C})\right| \leq 4^{-(S+2)}$. F does not change much locally on the space of configuration. Denoting by S the space of $\mathcal{M}$, and replacing m by $m+2S+4$ as in [5], we have $\left|Next(2^m, 2^S, \overline{C}) - \overline{C}\right| \leq 4^{-(S+2)}$. So at each step of the TM, the error is fixed (and bounded). We can then apply the above arguments (Proposition 39) to simulate continuously (E), with some controlled error: all involved quantities have encoding polynomials in the size of the inputs. ◀

4 Proof of Theorem 2

Proof. $\subseteq$: In this direction, we just need to prove that $\mathbb{RCD}$ contains only functions over the reals that are computable in polynomial space. Indeed, then for a function $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^{d'}$ sending every integer $\mathbf{n} \in \mathbb{N}^d$ to the vicinity of some integer of $\mathbb{N}^d$, at a distance less than $1/4$, by approximating its value with precision $1/4$ on its input arguments, and taking the closest integer, we will get a function from the integers to the integers, that corresponds to $\mathrm{DP}(f)$, and that will be in $\mathbf{FPSPACE} \cap \mathbb{N}^{\mathbb{N}}$.

This is indeed the case, since i) all the base functions of $\overline{\mathbb{RCD}}$ are in $\mathbf{FPSPACE}$: they are even in $\mathbf{FPTIME}$, see [46] ii) $\mathbb{R}^{\mathbb{R}} \cap \mathbf{FPSPACE}$ is stable under *composition.* iii) stability under *robust ODE* follows from Theorem 11.

$\supseteq$: In the other direction, we use an argument similar to [5]: namely, as the function is polynomial space computable, this means that there is a polynomial space computable function $g : \mathbb{N}^{d''+1} \to \{1, 3\}^*$ so that on $\mathbf{m}, 2^n$, it provides the encoding $\overline{\phi(\mathbf{m}, n)}$ of some dyadic $\phi(\mathbf{m}, n)$ with $\|\phi(\mathbf{m}, n) - \mathbf{f}(\mathbf{m})\| \leq 2^{-n}$ for all $\mathbf{m}$. The problem is then to decode, compute and encode the result to produce this dyadic. More precisely, from Theorem 42, we get $\tilde{g}$ with

$$|\tilde{g}(2^e, 2^{p(max(\mathbf{m},n))}, Decode(2^e, \mathbf{m}, n)) - \gamma_{word}(g(\mathbf{m}, n))| \leq 2^{-e}$$

for some polynomial p corresponding to the time required to compute g, and $e = \max(p(max(\mathbf{m}, n)), n)$. Then we need to transform the value to the correct dyadic: we mean

$$\tilde{\mathbf{f}}(\mathbf{m}, n) = EncodeMul(2^e, 2^t, \tilde{g}(2^e, 2^t, Decode(2^e, \mathbf{m}, n)), 1),$$

where $t = p(max(\mathbf{m}, n))$, $e = \max(p(max(\mathbf{m}, n)), n)$ provides a solution with $\|\tilde{\mathbf{f}}(\mathbf{m}, 2^n) - \mathbf{f}(\mathbf{m})\| \leq 2^{-n}$. ◀

5 Proof of Theorem 3

Proof. $\subseteq$: To prove that $\overline{\mathbb{RCD}} \subseteq \mathbb{R}^{\mathbb{R}} \cap \mathbf{FPSPACE}$, we only need to add to the previous arguments that $\mathbb{R}^{\mathbb{R}} \cap \mathbf{FPSPACE}$ is also stable under $ELim$.
$\supseteq$: In this direction, we have the same issue as in [5]: the strategy of decoding, working with the Turing machine, and encoding is not guaranteed to work for all inputs. But, we can solve it by using an adaptative barycenter technique as in [5].

We recall the principle here for a function whose domain is $\mathbb{R}$, but it can be generalised to $\mathbb{R}^d$. The idea is to construct some function $\lambda : \mathbb{N}^2 \times \mathbb{R} \to [0,1]$ definable in $\mathbb{RCD}_*$ as in Corollary 31, but with a continuous ODE : Adapting the proof from [5] and using the simulation of ξ in our continuous framework, we can consider $\lambda(2^m, N, x) = \Psi(\Xi(2^{m+1}, N, x - 9/8))$ where $\Psi(x) = \mathcal{C}\text{-}\mathfrak{s}(2^{m+1}, 1/4, 1/2, x)$. In particular, by definition, $\lambda \in \mathbb{RCD}_*$. Thus, by Lemma 30, if $\lambda(2^m, N, x) =_{2^{-m}} 0$, then $\sigma_2(2^m, N, x) =_{2^{-m}} \lfloor x \rfloor$. If $\lambda(2^m, N, x) =_{2^{-m}} 1$, then $\sigma_1(2^m, N, x) =_{2^{-m}} \lfloor x \rfloor$ and if $\lambda(2^m, N, x) \in (0,1)$, then $\sigma_1(2^m, N, x) =_{2^{-m}} \lfloor x \rfloor + 1$ and furthermore $\sigma_2(2^m, N, x) =_{2^{-m}} \lfloor x \rfloor$. So, $\lambda(\cdot, 2^n, x) Formula_1(x, u, M, n) + (1 - \lambda(\cdot, 2^n, n)) Formula_2(x, u, M, n)$ and we are sure to be close (up to some bounded error) to some 2^{-m} approximation of a function f. ◀

6 Conclusion

We characterised polynomial space using an algebraically defined class of functions, by using a finite set of basic functions, closure under composition, and a schema for defining functions from robust ODEs. We proposed a concept of robust ODEs solvable in polynomial space. As far as we know, this is an original method for solving ODEs optimising space. It is based on classical constructions such as Savitch's theorem. We extended existing characterisations to a characterisation of functions over the reals and not only over the integers.

The interesting message from our statements is that we provide a clear and simple concept associated with continuous ODEs for space: space corresponds to the precision for numerically stable systems. Hence, compiled with [17], we now know the length of solutions corresponds to time and precision to memory.

Considering future work: We have an algebraically defined class of functions. It remains to know whether this could be transferred at the level of polynomial ODE. We know that the solutions of polynomial ODEs define a very robust class of functions, stable by many operations: sum, products, division, ODE solving, etc: see [35, 15]. Hence, all the base functions we consider in our algebraic class can be turned into polynomial ODEs, by adding some variables. It would be interesting to understand if we could define space complexity directly at the level of polynomial ODEs, using precision.

Recently, another characterisation of **PSPACE** was obtained for polynomial ODEs using rather ad-hoc definitions in [34, 14] and working over a non-compact space. Could our characterisation be put at this simpler class of ODEs, but working with precision? The point is that this characterisation uses unbounded domains, so precision is harder to interpret in their constructions, where the schemas are somehow done to control errors.

Of course, from our statements, adding any **FPSPACE**-computable function over the reals among the base functions would not change the class. However, we did not intend to minimise the number of base functions. For example, $\tanh(t)$ is the solution of the ODE $f' = 1 + f^2$ and $\cos(t)$ can be obtained by the two-dimensional ODE $y_1' = -y_2$, $y_2' = y_1$. Minimising the number of base functions is also left for future work. We believe that even in this setting, proving space complexity corresponds to precision is already significant, independently of this question of a minimal set of base functions.

References

1 Oliver Aberth. The concect of effective method applied to computational problems of linear algebra. *J. Comput. Syst. Sci.*, 5(1):17–25, 1971. doi:10.1016/S0022-0000(71)80004-1.

2 Mikkel Abrahamsen, Linda Kleist, and Tillmann Miltzow. Training neural networks is er-complete. *Advances in Neural Information Processing Systems*, 34:18293–18306, 2021. URL: https://proceedings.neurips.cc/paper/2021/hash/9813b270ed0288e7c0388f0fd4ec68f5-Abstract.html.

3 Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, and Simon Weber. Training fully connected neural networks is ∃r-complete. *CoRR*, abs/2204.01368, 2022. doi:10.48550/arXiv.2204.01368.

4 Manon Blanc and Olivier Bournez. A characterization of polynomial time computable functions from the integers to the reals using discrete ordinary differential equations. In Jérôme Durand-Lose and György Vaszil, editors, *Machines, Computations, and Universality - 9th International Conference, MCU 2022, Debrecen, Hungary, August 31 - September 2, 2022, Proceedings*, volume 13419 of *Lecture Notes in Computer Science*, pages 58–74. Springer, 2022. MCU'22 Best Student Paper Award. doi:10.1007/978-3-031-13502-6_4.

5 Manon Blanc and Olivier Bournez. A characterisation of functions computable in polynomial time and space over the reals with discrete ordinary differential equations: Simulation of turing machines with analytic discrete odes. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science, MFCS 2023, August 28 to September 1, 2023, Bordeaux, France*, volume 272 of *LIPIcs*, pages 21:1–21:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.MFCS.2023.21.

6 Manon Blanc and Olivier Bournez. Simulation of turing machines with analytic discrete odes: FPTIME and FPSPACE over the reals characterised with discrete ordinary differential equations. *CoRR*, abs/2307.11747, 2023. doi:10.48550/arXiv.2307.11747.

7 Manon Blanc and Olivier Bournez. Quantifiying the robustness of dynamical systems. relating time and space to length and precision. In Aniello Murano and Alexandra Silva, editors, *32nd EACSL Annual Conference on Computer Science Logic, CSL 2024, February 19-23, 2024, Naples, Italy*, volume 288 of *LIPIcs*, pages 17:1–17:20, Naples, Italy, February 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CSL.2024.17.

8 Lenore Blum. *Complexity and real computation*. Springer, 1998. URL: https://www.worldcat.org/oclc/37004484.

9 Lenore Blum, Mike Shub, and Steve Smale. On a theory of computation and complexity over the real numbers; NP completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, 21(1):1–46, July 1989.

10 Olivier Bournez, Felipe Cucker, Paulin Jacobé de Naurois, and Jean-Yves Marion. Computability over an arbitrary structure. sequential and parallel polynomial time. In Andrew D. Gordon, editor, *Foundations of Software Science and Computational Structures, 6th International Conference (FOSSACS'2003)*, volume 2620 of *Lecture Notes in Computer Science*, pages 185–199, Warsaw, 2003. Springer. doi:10.1007/3-540-36576-1_12.

11 Olivier Bournez, Felipe Cucker, Paulin Jacobé de Naurois, and Jean-Yves Marion. Implicit complexity over an arbitrary structure: Sequential and parallel polynomial time. *Journal of Logic and Computation*, 15(1):41–58, 2005. doi:10.1093/logcom/exh036.

12 Olivier Bournez and Arnaud Durand. Recursion schemes, discrete differential equations and characterization of polynomial time computation. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th Int Symposium on Mathematical Foundations of Computer Science, MFCS*, volume 138 of *LIPIcs*, pages 23:1–23:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.MFCS.2019.23.

13 Olivier Bournez and Arnaud Durand. A characterization of functions over the integers computable in polynomial time using discrete ordinary differential equations. *Computational Complexity*, 32(2):7, 2023. doi:10.1007/s00037-023-00240-1.

14 Olivier Bournez, Riccardo Gozzi, Daniel S Graça, and Amaury Pouly. A continuous characterization of PSPACE using polynomial ordinary differential equations. *Journal of Complexity*, 77:101755, August 2023. doi:10.1016/j.jco.2023.101755.

15 Olivier Bournez, Daniel Graça, and Amaury Pouly. On the Functions Generated by the General Purpose Analog Computer. *Information and Computation*, 257:34–57, 2017. doi:10.1016/j.ic.2017.09.015.

16 Olivier Bournez, Daniel S Graça, and Amaury Pouly. On the complexity of solving initial value problems. In *Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation*, pages 115–121, 2012. doi:10.1145/2442829.2442849.

17 Olivier Bournez, Daniel S. Graça, and Amaury Pouly. Polynomial Time corresponds to Solutions of Polynomial Ordinary Differential Equations of Polynomial Length. *Journal of the ACM*, 64(6):38:1–38:76, 2017. doi:10.1145/3127496.

18 Olivier Bournez, Daniel Silva Graça, and Amaury Pouly. Computing with polynomial ordinary differential equations. *Journal of Complexity*, 36:106–140, 2016. doi:10.1016/j.jco.2016.05.002.

19 Olivier Bournez, Daniel Silva Graça, and Amaury Pouly. Polynomial time corresponds to solutions of polynomial ordinary differential equations of polynomial length: The general purpose analog computer and computable analysis are two efficiently equivalent models of computations. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 109:1–109:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.109.

20 Olivier Bournez and Amaury Pouly. A universal ordinary differential equation. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 116:1–116:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.116.

21 Olivier Bournez and Amaury Pouly. A survey on analog models of computation. In *Handbook of Computability and Complexity in Analysis*, pages 173–226. Springer, 2021.

22 M. S. Branicky. Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoretical Computer Science*, 138(1):67–100, 6 February 1995. doi:10.1016/0304-3975(94)00147-B.

23 Vasco Brattka, Peter Hertling, and Klaus Weihrauch. A tutorial on computable analysis. In *New computational paradigms*, pages 425–491. Springer, 2008.

24 Mark Braverman. Hyperbolic Julia sets are poly-time computable. *Electronic Notes in Theoretical Computer Science*, 120:17–30, 2005. doi:10.1016/j.entcs.2004.06.031.

25 Manuel L. Campagnolo, Cristopher Moore, and José Félix Costa. Iteration, inequalities, and differentiability in analog computers. *Journal of Complexity*, 16(4):642–660, 2000. doi:10.1006/jcom.2000.0559.

26 Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pages 6571–6583, 2018. URL: https://proceedings.neurips.cc/paper/2018/hash/69386f6bb1dfed68692a24c8686939b9-Abstract.html.

27 P. Clote. Computational models and function algebras. In Edward R. Griffor, editor, *Handbook of Computability Theory*, pages 589–681. North-Holland, Amsterdam, 1998. doi:10.1016/s0049-237x(99)80033-0.

28 Peter Clote and Evangelos Kranakis. *Boolean Functions and Computation Models*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2002. doi:10.1007/978-3-662-04943-3.

29 Alan Cobham. The intrinsic computational difficulty of functions. In Y. Bar-Hillel, editor, *Proceedings of the International Conference on Logic, Methodology, and Philosophy of Science*, pages 24–30. North-Holland, Amsterdam, 1962.

30 Pieter Collins and Daniel S Graça. Effective computability of solutions of ordinary differential equations the thousand monkeys approach. *Electronic Notes in Theoretical Computer Science*, 221:103–114, 2008. doi:10.1016/j.entcs.2008.12.010.

31 Pieter Collins and Daniel S. Graça. Effective Computability of Solutions of Differential Inclusions The Ten Thousand Monkeys Approach. *Journal of Universal Computer Science*, 15(6):1162–1185, 2009. doi:10.3217/jucs-015-06-1162.

32 Kousha Etessami and Mihalis Yannakakis. On the complexity of Nash equilibria and other fixed points. *SIAM Journal on Computing*, 39(6):2531–2597, 2010. doi:10.1137/080720826.

33 François Fages, Guillaume Le Guludec, Olivier Bournez, and Amaury Pouly. Strong turing completeness of continuous chemical reaction networks and compilation of mixed analog-digital programs. In Jérôme Feret and Heinz Koeppl, editors, *Computational Methods in Systems Biology - 15th International Conference, CMSB 2017, Darmstadt, Germany, September 27-29, 2017, Proceedings*, volume 10545 of *Lecture Notes in Computer Science*, pages 108–127. Springer, 2017. CMSB'2017 Best Paper Award. doi:10.1007/978-3-319-67471-1_7.

34 Riccardo Gozzi. *Analog Characterization of Complexity Classes*. PhD thesis, Instituto Superior Técnico, Lisbon, Portugal and University of Algarve, Faro, Portugal, 2022.

35 Daniel S. Graça. *Computability with Polynomial Differential Equations*. PhD thesis, Instituto Superior Técnico, 2007.

36 Daniel S. Graça, Manuel L. Campagnolo, and Jorge Buescu. Robust simulations of Turing machines with analytic maps and flows. In B. Cooper, B. Loewe, and L. Torenvliet, editors, *Proceedings of CiE'05, New Computational Paradigms*, volume 3526 of *Lecture Notes in Computer Science*, pages 169–179. Springer-Verlag, 2005. doi:10.1007/11494645_21.

37 Daniel S. Graça and José Félix Costa. Analog computers and recursive functions over the reals. *jcomp*, 19(5):644–664, 2003. doi:10.1016/S0885-064X(03)00034-7.

38 Daniel S. Graça, N. Zhong, and J. Buescu. Computability, noncomputability and undecidability of maximal intervals of IVPs. *Transactions of the American Mathematical Society*, 2006. To appear.

39 Daniel S. Graça and Ning Zhong. *Handbook of Computability and Complexity in Analysis*, chapter Computability of Differential Equations, pages 71–99. Springer, 2021.

40 Daniel Silva Graça and Ning Zhong. Robust non-computability and stability of dynamical systems. *CoRR*, abs/2305.14448, 2023. doi:10.48550/arXiv.2305.14448.

41 Morris W. Hirsch, Stephen Smale, and Robert Devaney. *Differential Equations, Dynamical Systems, and an Introduction to Chaos*. Elsevier Academic Press, 2003.

42 Akitoshi Kawamura. Lipschitz continuous ordinary differential equations are polynomial-space complete. In *2009 24th Annual IEEE Conference on Computational Complexity*, pages 149–160. IEEE, 2009. doi:10.1109/CCC.2009.34.

43 Akitoshi Kawamura, Hiroyuki Ota, Carsten Rösnick, and Martin Ziegler. Computational complexity of smooth differential equations. *Logical Methods in Computer Science*, 10, 2014. doi:10.2168/LMCS-10(1:6)2014.

44 Patrick Kidger. On neural differential equations. *CoRR*, abs/2202.02435, 2022. doi:10.48550/arXiv.2202.02435.

45 Ker-I Ko. On the computational complexity of ordinary differential equations. *Information and Control*, 58(1-3):157–194, July/August/September 1983. doi:10.1016/S0019-9958(83)80062-X.

46 Ker-I Ko. *Complexity theory of real functions*, volume 3 of *Progress in theoretical computer science*. Birkhäuser, Boston, 1991.

47 Webb Miller. Recursive function theory and numerical analysis. *Journal of Computer and System Sciences*, 4(5):465–472, October 1970. doi:10.1016/S0022-0000(70)80043-5.

48 John Milnor. On the concept of attractor. *Communications in Mathematical Physics*, 99:177–195, 1985.

49 Amaury Pouly and Daniel Silva Graça. Computational complexity of solving polynomial differential equations over unbounded domains. *Theoretical Computer Science*, 626:67–82, 2016. doi:10.1016/j.tcs.2016.02.002.

50 Marian Boykan Pour-El and J. Ian Richards. A computable ordinary differential equation which possesses no computable solution. *Annals of Mathematical Logic*, 17:61–90, 1979.

51 Cristobal Rojas and Mathieu Sablik. On the algorithmic descriptive complexity of attractors in topological dynamics. *CoRR*, abs/2311.15234, 2023. doi:10.48550/arXiv.2311.15234.

52 Keijo Ruohonen. An effective Cauchy-Peano existence theorem for unique solutions. *International Journal of Foundations of Computer Science*, 7(2):151–160, 1996. doi:10.1142/S0129054196000129.

53 Claude E. Shannon. Mathematical theory of the differential analyser. *Journal of Mathematics and Physics MIT*, 20:337–354, 1941.

54 Michael Sipser. *Introduction to the Theory of Computation.* PWS Publishing Company, 1997.

55 Alan M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937. Reprinted in Martin Davis. The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions. Raven Press, 1965. doi:10.1112/plms/s2-42.1.230.

56 Bernd Ulmann. *Analog computing.* Walter de Gruyter, 2013.

57 Bernd Ulmann. *Analog and hybrid computer programming.* De Gruyter Oldenbourg, 2020.

58 Veritasum. Future computers will be radically different (analog computing). Youtube video, 2022. URL: https://www.youtube.com/watch?v=GVsUOuSjvcg.

59 Klaus Weihrauch. *Computable Analysis - An Introduction.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 2000. doi:10.1007/978-3-642-56999-9.