Continuous models of computation: from computability to complexity

Amaury Pouly

ÉCOLE POLYTECHNIQUE
et
UNIDERSIDADE DO ALGARVE

Laboratoire d'Informatique
de l'École Polytechnique

École Doctorale de l'École Polytechnique

THÈSE DE DOCTORAT

par **Amaury Pouly**

soutenue le **6 juillet 2015**

pour obtenir le grade de **Docteur de l'École Polytechnique**
discipline **Informatique**

Continuous models of computation: from computability to complexity

Acknowledgements[1][2]

Je remercie tout d'abord Olivier et Daniel, mes Directeurs de thèse, qui m'ont accompagné durant ces quatre années. Je suis ravi d'avoir pu travailler avec eux tout au long de cette aventure, ils ont su me conseiller et me guider lorsqu'il le fallait, tout en me laissant une grande liberté dans ma façon de travailler. Je n'oublierai pas nos nombreuses discussions, parfois houleuses[3], desquelles sont toujours ressorties de nouvelles idées, même dans les moments les plus difficiles de cette ambitieuse thèse. Je suis sûr qu'ils ne sont pas près d'oublier mes trop nombreux «Je peaufine juste quelques détails de la preuve» et «J'ajoute juste un petit résultat» mais ont heureusement su m'interrompre quand il le fallait.

Je remercie Eugene Asarin, Bruno Salvy et Martin Ziegler qui m'ont fait l'honneur d'être rapporteurs de ma thèse. Leurs remarques sur ce manuscrit ont permis, je l'espère, de le rendre plus clair et plus abordable. Je promets dorénavant de typographier correctement mes équations et mes listes[4]. Je remercie bien évidemment Enrico Formenti, Eric Goubault et Etienne Grandjean d'avoir accepté de faire partie de mon jury de thèse. Jury qui a, par ailleurs, eu l'extrême obligeance de me laisser pas moins de 55 minutes pour exposer mes travaux[5].

Je remercie mon collaborateur Walid ainsi que mes deux stagiaires, Hugo et Yassine, qui n'ont, je l'espère, pas trop souffert de mon encadrement.

Au cours de ces années au laboratoire, j'ai eu le plaisir de rencontrer de nombreuses personnes que je souhaite remercier, dans aucun ordre particulier. Merci à Daniel, François, Cécile, Julia, Élise, Nicolas, Philippe, Alain, David, Françoise, Aurore, Ben, Johan, Razvan, Olivier, Jonas, François, Chantal, Victor, Émilie, Florence avec qui j'ai passé de nombreux repas, pauses café, débats, soirées films, jeux de société et même répétitions de soutenance de thèse. Encore quelques années et j'aurai presque fini par mettre à la cryptographie[6]. Merci à Amélie et Alice pour les repas, discussions, cinémas, soirées, courses, de m'avoir converti à l'athlétisme[7] et les innombrables services que vous m'avez rendus. Merci à Steve et Mathieu pour les footing très sympas du midi. Thanks to Mahfuza for our coffee breaks, discussion, meals, cinemas and other activities, as well as your endless positivism and encouragements.

Il va sans dire que je remercie toutes les personnes m'ayant permis d'effectuer mes vacations et monitorat à Versailles, Polytechnique et Orsay, ainsi que mes collègues enseignants.

Je suis particulièrement reconnaissant envers les personnels administratifs Evelyne, Sylvie et Catherine.

Je remercie Johanne pour ses nombreux conseils et m'avoir remonté le moral quand il le fallait.

[1]If you don't read French, don't give up.

[2]Je remercie tous ceux que j'ai oublié afin de n'oublier personne. Ou pas.

[3]Je me laisserai à qualifier notre trio de l'Optimiste, le Sceptique, et le Perfectionniste, en laissant au lecteur le soin d'assigner les rôles.

[4]Pas comme dans cette thèse.

[5]Et failli provoquer un arrêt cardiaque dû au stress d'une certaine personne dans l'assemblée.

[6]Mais bon la calculabilité reste quand même le meilleur des domaines de l'informatique.

[7]Même si cela n'a pas été très dur.

Il me faut évidemment remercier mes colocataires Mikaël[8], Pierre-Marie[9], Hélène (et ses chats), Sylvain, Timothé, Alexandre, Pierre, Philippe[10] et Laure[11], grâce à qui j'ai passé quatre très bonnes années. Merci aussi à Bruno, Max, Aisling, Irène, Jill-Jênn, Juvénal, Mathilde, Tahina pour toutes les discussions et moments passés ensemble.

Je remercie Lucie, Marthe[12] et Laure, pour leur soutien inconditionnel, les discussions, les conseils et les bons moments passés ensemble.

Je remercie enfin ma famille, pour tous les moments passés ensemble, pour avoir cru en moi et m'avoir aidé lorsqu'il le fallait. Alors merci à Pascale[13], Jean-Luc, Arthur, Aloïs, Cécile[14], Joseph, Guillaume, Cédric, Danielle, Denise, Albert, Yvonne, Élizabeth, Gabriel, Clémentine, Julia, Antoine, Patrice.

Et bien évidemment, il faut mentionner le contributeur majeur[15] à cette thèse, à savoir le café du labo. J'admets avoir basculé du côté Nespresso du café[16], qui est certes obscur[17] mais surtout meilleur.

[8]Le Rageux.

[9]Ma thèse utilise le tiers-exclus et les réels existent.

[10]Qui reste stoïque.

[11]'Rage.

[12]La Fruitesse qui fructidévore les pommes.

[13]Un bisou spécial à ma maman qui est la meilleure du monde !

[14]Et Mélina.

[15]Comme nous l'a enseigné Erdős.

[16]What else ?

[17]Les capsules propriétaire c'est mal.

[42]Pour information, cette thèse en est à la version thesis_v1.4.1[42].

Contents

Chapter 1

Introduction

Fundamental Theorem of Analysis: any theorem in Analysis can be fitted onto an arbitrarily small piece of paper if you are sufficiently obscure.

Quote from Cambridge

One motivation of this thesis is the possibility of giving a machine-independent characterization of complexity classes. This has been studied in the field of Implicit Complexity, where complexity classes are typically characterized as function algebras, that is the set of functions closed under operations like composition, arithmetic, integration, limits and so on. More precisely, we want to characterize real complexity classes. There are several incomparable notions of computability over real numbers, like the BSS model [BCSS98] and $\mathbb{R}$-recursive functions [Moo96, MC04]. In this thesis, we will study Computable Analysis [Bra05a, Ko91, KW95] which is an approximation theory and has close links to numerical analysis and formal methods. The approach we follow is that of dynamical systems, in particular with continuous time and/or space. In this approach, one characterizes computability and complexity classes by putting restrictions on the allowed systems.

In many ways, this question is related to the so-called (effective) Church Thesis, which states that all discrete and reasonable models of computation have the same computational power as the Turing machine. This thesis has been shown to hold for many models, and even formalized in the context of abstract algorithms [BS03, BG03, DG08]. These works inspired a stronger thesis, known as the effective Church Thesis [BDF12], which generalizes the conjecture at the complexity level, and the physical Church Thesis [BCPT14], which conjectures that physically realistic models of computation cannot achieve super-Turing power. However, there has been some hope that one might be able to present realistic models that might compute *faster* than Turing machines in computational complexity terms. Perhaps the most-well known candidate for such model are quantum computers, first introduced by Feynman in [Fey82]: see for example the famous result on integer factorization in polynomial time of [Sho97]. The other natural candidates were continuous-time models of computation such as the General Purpose Analog Computer (GPAC). We will prove that the GPAC does not prove more computational power than Turing based paradigms.

A fundamental difficulty of continuous-time model is to define a proper notion of complexity. Indeed, it is known that Turing machines can be simulated by various classes of ordinary differential equations or analog models. This can often be done even in real time: the state $y(t)$ at time $t \in \mathbb{N}$ of the solution of the ordinary differential equation encodes the state after the execution of t steps of the Turing machine. However, a troubling problem is that many models exhibit the so-called *Zeno's phenomenon*, also known as space-time contraction. This phenomenon results from the possibility of simulating an infinite number of discrete transitions in a finite amount of time. This can be used to solve the Halting problem in finite time and to show that some models are super-Turing powerful. Many continuous time systems might undergo arbitrary time contractions to simulate the computation of a Turing machine in an arbitrary short time (see e.g. [AM98a, EN02, Ruo93, Ruo94, Moo96, Bou97, Bou99, AD90, CP02, Dav01, Cop98, Cop02]).

Since the introduction of the P and NP complexity classes, much work has been done to build a well-developed complexity theory based on Turing Machines. In particular, classical computational complexity theory is based on limiting resources used by Turing machines, usually time and space. Another approach is implicit computational complexity. The term "implicit" can sometimes be understood in various ways, but a common point of many of the characterizations is that they provide (Turing or equivalent) machine-independent definitions of classical complexity.

Implicit characterization theory has gained enormous interest in the last decade [DL12]. This has led to many alternative characterizations of complexity classes using recursive function, function algebras, rewriting systems, neural networks [SS95], lambda calculus and so on. However, most — if not all — of these models or characterizations are essentially discrete: in particular they are based on underlying models working with a discrete time on objects that are often defined in a discrete space.

In 1941, Claude Shannon introduced a model for the Differential Analyzer [Bus31], on which he worked as an operator, called the General Purpose Analog Computer (GPAC) [Sha41]. The Differential Analyzer was a mechanical device used mostly in the thirties and the forties to solve ordinary differential equations (ODEs). Electronic versions of this device greatly improved its capabilities but the emergent digital computers eventually replaced all analog computers. Originally it was presented as a model based on circuits (see Figure 1.0.1), where several units performing basic operations (e.g. sum, integration) are interconnected. Shannon claimed [Sha41] that functions generated by the GPAC were differentially algebraic. Differentially algebraic functions satisfy a differential equation of the form $p(t, y(t), y'(t), \ldots, y^{(k)}(t)) = 0$ for $t \in I$ where p is a polynomial and I an interval.

Years later, Pour-El found an error in the proof [PE74] and redefined the GPAC in terms of quasi-linear differential equations. However this approach was again found to be flawed and finally fixed by Graça and Costa [GC03] by restricting the set of allowed connections in the circuit. In this later approach, functions generated by the GPAC are exactly those that satisfy differential equations of the form:

$$y'(t) = p(y(t)) \qquad t \in I$$

where p is a (vector of) polynomials and y is a vector. This class of functions turns out to be more general than it may seem at first. Indeed, not only is it closed under the usual arithmetic operations but also one can replace p by any such generated function. For example, $y' = \sin(y)$ is in this class because sin is in this class (see Figure 1.0.2).

Analog computers have since been replaced by their digital counterpart. Nevertheless, one can wonder if those are really better suited for computation, or if the difference only lies in practical and technological grounds. More precisely, it is natural to try and compare the GPAC and Turing machines. A few years ago, it was shown [BCGH07, GCB08] that Turing-based paradigms and the GPAC have the same computational power. So switching to analog computers would not enable us to solve the Halting problem, or any uncomputable exotic problem, but it would not result in any loss of computation power. However, this result did not shed any light on what happens at a computational complexity level. In other words, analog computers do not make a difference about *what* can be computed, but maybe they could compute *faster* than a digital computer.

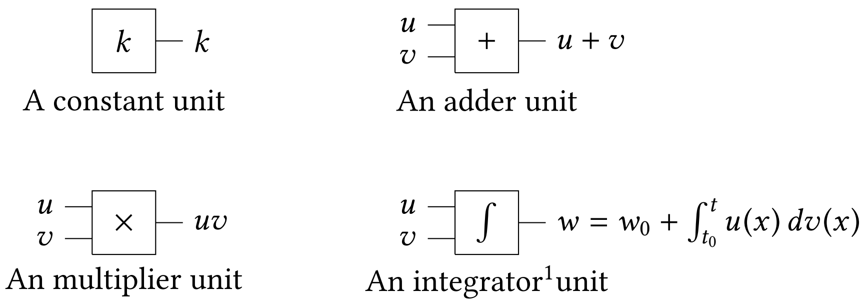
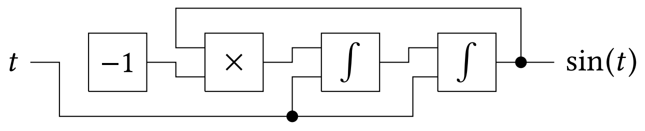


Figure 1.0.1: Basic units of the GPAC



$$\begin{cases} y'(t) = z(t) \\ z'(t) = -y(t) \\ y(0) = 0 \\ z(0) = 1 \end{cases} \Rightarrow \begin{cases} y(t) = \sin(t) \\ z(t) = \cos(t) \end{cases}$$

Figure 1.0.2: GPAC computing sine

In this thesis, we give several fundamental contributions to these questions:

- **We show that time of a computation, for the GPAC, can be measured as the length of the curve** (i.e. length of the solution curve of the ordinary differential equation associated to the GPAC). This notion of complexity turns out to be equivalent to considering both time *and* space, for a suitable notion of space. Unlike discrete models of computation, there is no relationship between time and space in this model, which means that neither alone is a good measure of complexity. We prove that the obtained notion is very robust by giving several equivalent definitions.

- **We show that the GPAC (or equivalently the class of polynomial initial value problems) has the same computational power as the Turing machine**, both at the computability and *complexity level*. More precisely, a GPAC can simulate a Turing machine and preserve its complexity, and vice versa. Previous attempts at building an analog equivalent of Turing machines either failed because they considered classes of dynamical systems that where too general, or because they used a notion of time that was not robust: see [BC08] for discussions about various attempts. Our work suggests that the class of polynomial differential equations is the right compromise to obtain an interesting and physically realistic model.

- **We also provide a purely analog, machine-independent characterization of P and Computable Analysis**. Indeed, our characterization relies only on a simple and natural class of ordinary differential equations, with polynomial right-hand side. This shows first that (classical) complexity theory can be presented in terms of ordinary differential equations problems. This fact has never been established before. This also shows as a side effect that solving ordinary differential equations leads to complete problems, even with a fixed dimension.

Contents of the thesis

[1]The integral $\int u dv$ is understood in the sense of Riemann-Stieltjes integration

- **Chapter 1 (Introduction)**
  In the remaining part of this chapter, we give more details about the General Purpose Analog Computer (GPAC) and recall some notions and results about other areas involved in this thesis, such as dynamical systems, numerical analysis, computable analysis and differential equations.

- **Chapter 2 (The PIVP class)**
  This chapter focuses on the particular class of differential equations that model the GPAC: polynomial initial value problems (PIVP), that is to say ordinary differential equations of the form $y'(t) = p(y(t))$ where p is a (vector of) polynomials and $y(t)$ a vector. We develop a theory of *generable functions*, which are functions satisfying PIVP as well as some requirements on their rate of growth. We show that this is closed by many operations, including composition and ODE solving. In particular, solutions of differential equations of the form $y' = f(y)$, where f is elementary (in the sense of Analysis), are also included in this class.

- **Chapter 3 (Solving PIVP)**
  In this chapter, we investigate the complexity of solving ordinary differential equations of the form $y' = p(y)$ over *unbounded time domains*, where p is a (vector of) polynomials. We show that one can compute y at time t with absolute error ε in time polynomial in $|t|$, $-\ln\varepsilon$, and the length of the curve y between t_0 and t: $\int_{t_0}^{t} \|p(y)\|\,dt$. As far as we know, this is first result to take all parameters into account into the complexity and provide a polynomial time complexity in most of them.

- **Chapter 4 (Computation with PIVP)**
  This chapter introduces a notion of computability with PIVP that generalizes the notion of generable functions. We introduce a notion of complexity on these systems and show that the obtained class enjoys many closure properties as well as several equivalent definitions. This provides a natural, robust and machine-independent notion of computability for real functions.

- **Chapter 5 (PIVP versus Turing computability)**
  In this chapter, we show that the class of PIVP computable functions, with the right parameters, is equivalent to the polynomial time computable functions from Computable Analysis. This way, we obtain a purely continuous definition of computability and complexity over real numbers. We also give a characterization of the complexity class P in terms of PIVP only. As far as we know, this is the first time P is characterized using ODEs.

- **Chapter 6 (Piecewise Affine Systems)**
  This chapter focuses on the problem of reachability for piecewise affine systems over $[0,1]^d$. The general reachability problem is known to be undecidable starting from dimension 2. We investigate the complexity of several decidable variants of reachability problems and of control problems. We show in particular that the region to region bounded time versions leads to NP-complete or co-NP-complete problems, starting from dimension 2.

1.1 Dynamical Systems

In their most general formulation, a dynamical system can be described as an evolution rule that defines a trajectory, in the sense of a mapping from a time space to a state (or phase) space. In this thesis, we will focus on *deterministic* systems where the evolution rule is deterministic.

More general system with nondeterministic, or stochastic evolution rules also are well-studied objects [Arn95].

Definition 1.1.1 (Dynamical system)**:** A *dynamical system* is a tuple (T, X, Φ) where T is an additive monoid, X is a set and $\Phi : U \subseteq T \times X \to X$ is a mapping satisfying:

- $\Phi(0, x) = x$
- $\Phi(t_2, \Phi(t_1, x)) = \Phi(t_1 + t_2, x)$ if $t_1, t_2, t_1 + t_2 \in U$

♦

The monoid T is generally described as the *time*, the set X as the *state* or *phase* space and Φ as the *flow*. The state space is typically taken to be included in $\mathbb{R}^d$ or more generally a manifold. Each state defines an *orbit* which is the set of all reachable states using the flow.

Definition 1.1.2 (Orbit)**:** The *orbit* or *trajectory* $\mathcal{O}_x$ is the state space defined by

$$\mathcal{O}_x = \{\Phi(t, x), t \in U\} \subseteq X$$

♦

Dynamical systems have received a lot of attention since they provide a natural way to describe many systems and phenomena. They can be categorized depending on the properties of the time and state spaces. A first distinction is usually made between dynamical system with *discrete* and *continuous* time. In addition to these two classes, hybrid systems are the object of many recent studies. By combining continuous dynamics with discrete control, hybrid systems provide a natural framework to study physical systems under computer control for example.

There are many natural questions on dynamical systems, some of which are at the core of several important fields in computer sciences. One of the most important domain is that of verification theory, which asks whether a given system satisfies a given property. A particularly important subdomain is that of reachability problems. More precisely, given a dynamical system and an initial region, decide if the system eventually reaches a final region. This problem has many application in concrete systems to check safety properties. Other well-studied properties include attractors, periodic orbits, mortality (do all orbits go through 0), convergence, stability [BS02]. Most of these problems are undecidable in the general case, and usually stay hard in restricted cases [Moo90, Bra95]. When computability is known, further refinement can be made to study the computational complexity of these problems. However, those results are quite challenging to obtain and there are still many gaps to fill to understand the full picture.

Definition 1.1.3 (Discrete dynamical system)**:** A *discrete dynamical system* is a tuple (X, f) where X is a set and $f : X \to X$ is a mapping. The associated dynamical system is $(\mathbb{N}, X, \Phi)$ where $\Phi(x, t) = f^{[t]}(x)$. The orbit of a point x is $\mathcal{O}_x = \{f^{[t]}(x), t \in \mathbb{N}\}$. ♦

Discrete dynamical systems include a number of well-studied systems from the point of view of computability theory, such as sequences, cellular automata, shifts, Turing machines, and many others [BP97, BBEP10]. They form a particularly important class of dynamical systems which is better understood than the general case. In particular, they have a natural connection with discrete models of computation [KCG94a]. These results have led to the so-called Church Thesis, stating that all reasonable discrete models of computation are equivalent. Hence many problems on dynamical systems have been characterized by computational properties.

With a more general point of view, discrete systems form a well-known class of which many more aspects have been studied and whose merits, for example in experimental sciences, have no more need to be proved [HSD04].

1.2 Numerical Analysis and Differential Equations

Generally speaking, numerical analysis is the study of numerical algorithms for problems coming from mathematical analysis. A typical and very old such problem is to compute an approximation of $\sqrt{2}$, or to compute roots of functions. In general, the goal is to design and analyze techniques to compute approximate but accurate solutions to problems. Many of these problems come from real-life applications, such as simulating a car, a plane, or even the weather. But it can also apply to the more abstract setting of optimization problems, such as body dynamics or economics. In this thesis, we restrict ourselves to the case of numerical analysis of differential equations, but this is only a subset of what exists in the literature [Atk89, Lig91].

The study of numerical approximations to the solutions of differential equations is a major field in numerical analysis. It is major by its very broad application, since differential equations can model so many physical systems. It is also major by the techniques developed, dating back to Euler which have led to a very rich literature on the subject [But87]. A particularly difficult setting is that of partial differential equations [LT03], which we do not consider in this thesis. A very general setting is that of differential algebraic equations (DAE) but we restrict ourselves to the case of ordinary differential equations (ODE). There are two reasons to do so: first it is usually possible to transform, at least locally, a DAE into an ODE, and secondly, ODEs form a too general class of functions so we focus on a particular subclass of ODEs.

Definition 1.2.1 (Differential Algebraic Equation (DAE))**:** A *differential algebraic equation* is a functional equation in y of the form

$$F(y(t), \ldots, y^{(n-1)}(t), t) = 0 \qquad \forall t \in I$$

where I is an interval, $d, n \in \mathbb{N}$, $F : (\mathbb{R}^d)^n \times \mathbb{R} \to \mathbb{R}^n$ and $y : I \to \mathbb{R}^d$. ♦

Definition 1.2.2 (Ordinary Differential Equation (ODE))**:** An *ordinary differential equation* is a functional equation in y of the form

$$y'(t) = f(y(t)) \qquad \forall t \in I$$

where I is an interval, $d \in \mathbb{N}$, $f : \mathbb{R}^d \to \mathbb{R}^d$ and $y : I \to \mathbb{R}^d$. ♦

Definition 1.2.3 (Initial Value Problem (IVP))**:** An *initial value problem* is a differential equation together with an *initial condition* of the form $y(t_0) = y_0$ where $t_0 \in I$ and $y_0 \in \mathbb{R}^d$. ♦

The Cauchy-Lipschitz is a particularly important theorem in the study of ordinary differential equations, giving existence and uniqueness of solutions with very weak requirements.

Theorem 1.2.4 (Cauchy-Lipschitz)**:** *Consider the initial value problem:*

$$y(t_0) = y_0 \qquad y'(t) = f(t, y(t))$$

Suppose that on a neighborhood of (t_0, y), f is Lipschitz continuous in y and continuous in t. Then there is a neighborhood of t_0 where there exists a unique solution y to this initial value problem. ♦

Numerical analysis of differential equations usually focuses on solving initial value problems (IVP) [But87]. More precisely, given a differential equation, an initial condition and a time t, one wants to compute $y(t)$ where y is the solution to the IVP (if it exists). Over time, many different methods have been designed to tackle this problem, and there is a general theory to study the properties of very large classes of methods over compact intervals, called *general linear methods* (GLM). Numerical methods usually belong to GLM, which contains two very important classes of methods called *linear multistep method* and *Runge-Kutta methods.* The former is the result of successive generalizations of the *Euler method*, while the latter generalizes the *classical Runge-Kutta method* (also known as RK4).

Numerical methods all share a common set of ideas which can be found in the original Euler method. Suppose we want to find an approximate solution of $y(t)$ that satisfies $y'(t) = f(y)$ and $y(t_0) = y_0$. We choose a *time step* h and discretize the problem by letting $t_i = t_0 + ih$ and $y_i = y(t_i)$. Euler's approximation consists in defining the sequence $\tilde{y}_0 = y_0$ and $\tilde{y}_{i+1} = \tilde{y}_i + hf(\tilde{y}_i)$. The idea is that for a sufficiently small time step, $y(t_{i+1}) = y(t_i + h) \approx y(t_i) + hy'(t_i)$ and we know that $y'(t_i) = f(t_i)$ since y satisfies the differential equation. The Euler method is said to be a *first order*, *one step*, *linear*, *explicit* method, we detail below what it means.

- First order: $y(t_i) + hy'(t_i)$ is a first order approximation of $y(t_i + h)$, in the sense of Taylor approximation.

- One step: $\tilde{y}_{i+1}$ is computed using $\tilde{y}_i$ only.

- Linear: $\tilde{y}_{i+1}$ is a linear combination of $\tilde{y}_i$ and $f(\tilde{y}_i)$.

- Explicit: $\tilde{y}_{i+1}$ is given by an explicit formula.

All these points can be, and have been, refined, which has led to many families of numerical methods. We give a brief overview of the possible extensions in each direction.

- Higher order: one can approximate $y(t_i + h)$ using a higher order Taylor approximation of the form $\sum_{j=0}^{k-1} \frac{y^{(j)}(t_i)}{j!} h^j$, where the *order* k can either be fixed or variable. The higher the order, the more accurate the approximation, but the more costly it is to compute.

- Multistep: one can compute $\tilde{y}_{i+m}$ from $\tilde{y}_i, \ldots, \tilde{y}_{i+m-1}$. It means the next value is computed using the m last terms which form the *history.* Under the right conditions, a longer history gives a better approximation but it introduces many more points to compute and may have convergence issues.

- Implicit: one can design a method where $\tilde{y}_{i+1}$ is obtained by solving an equation such as $G(\tilde{y}_{i+1}, \tilde{y}_i, f(\tilde{y}_i), h) = 0$ where G can be any function. In this case, the method is *implicit* because an equation must be solved in order to find $\tilde{y}_{i+1}$. Implicit methods are usually more expensive but can bring better stability properties at low orders. However, it is unclear that implicit methods are asymptotically equivalent to explicit methods.

Numerical analysis focuses on both the *accuracy* and *efficiency* of the methods. The former is most often stated in terms of *order.* More precisely, a method is said to be of order k if *local error*, that is the error made at each step, is a $O\left(h^k\right)$. This is an asymptotic property of the algorithm when $h \to 0$. The efficiency can be measured theoretically by an analysis of the complexity of the algorithm, or practically by measuring the speed of the algorithm on some benchmarks. In this thesis, we are only interested in the theoretical complexity of solving differential equations.

1.3 Computable Analysis

Recursive Analysis was introduced by Turing [Tur36], Grzegorczyk [Grz55] and Lacombe [Lac55]. It focuses on providing a rich notion of computability in order to give a computational content to mathematical analysis. In this framework, a real number x is computable if there exists a computable sequence of rational numbers quickly converging to x. A real function f is computable if there exists a functional (i.e. operator) that maps quickly converging sequences to x to quickly converging sequences to $f(x)$. One way to define such functionals is Type-2 computability [Wei00] and can be extended to functions over abstract domains via *representations*. Later work refined those approaches with complexity. Computable Analysis as described by Ker-I Ko [Ko91] builds on the usual complexity theory of Turing machines to provide a complexity theory for real functions. Recent work [BH05b, BH04, BH05a, BH06] gives a more algebraic characterization in terms of recursive functions and limit operators, building on the work of Campagnolo, Moore and Costa [CMC00, CMC02]. Recent work of Kawamura and Cook [KC10] refined the work of Weihrauch and others with a framework for the complexity of operators. Operator complexity gives a stronger, uniform notion of complexity which is very important in Analysis. Many "weak" complexity results can be lifted to this stronger framework [KO14].

We would like to mention that other approaches to real computability from Blum, Shub and Smale [BSS89, BCSS98] and Moore [Moo96] yield very different and in some cases more powerful models of computations. See [Kaw05] for a comparison of Type-2 computability and Moore's recursion for example, and [Bra05b] for some connections between BSS and Computable Analysis.

In this section, we follow Ker-I Ko presentation of Computable Analysis [Ko91]. The most basic object of this framework is that of *real number*. Contrary to recursive analysis where there are multiple equivalent ways of representing a real number, computable analysis requires greater care about the efficiency of the representation. The usual representation consists in a sequence of quickly converging rational numbers: a real number is seen as the limit of sequence of rational numbers. The presentation is computable when the sequence is computable, and polynomial time computable when the sequence is polynomial time computable. Following Weirauch and Kreitz, a real number is seen as a *Type 1* object is the sense that it is a function from integers to rational numbers.

Definition 1.3.1 (Computable real)**:** A real number x is *computable* if there is a sequence of rational numbers $(r_n)_n$ such that $|x - r_n| \leqslant 2^{-n}$ for all n, and a Turing machine $\mathcal{M}$ such that $\mathcal{M}(n)$ computes r_n. Furthermore, x is *polynomial time* computable if $\mathcal{M}$ runs in time polynomial[2] in n. The set of computable real numbers is usually denoted by $\mathbb{R}_c$ and the set of polynomial time computable real numbers by $\mathbb{R}_P$. ♦

The next basic block of the theory of computable analysis is of course the definition of a computable function. We have seen that a real number is already a function (or Type 1 object), it is thus natural to see a real function as an operator, mapping one function to another. This is called a *Type 2* function in recursive analysis. Informally, a function f is computable if there is a machine $\mathcal{M}$, such that for any real number x and any precision n, computes an approximation $r_{x,n}$ close to $f(x)$ by $\pm 2^{-n}$. However, x is a real number so we cannot "give" it as an argument to a machine: it is an infinite object. The proper concept for formalize this, is that of *oracle*. We say that f is computable if for any oracle computing x, the machine equipped with this oracle computes $f(x)$ – in the sense of the previous definition of a computable real number. More precisely, we say that the oracle computes x if when called on a tape containing an integer

[2]From the point of view of classical complexity, it is akin to writing n in unary.

n, it writes a rational number r_n such that $|r_n - x| \leqslant 2^{-n}$. In the case where the computation takes polynomial time, *independently of the oracle*, we say that the function is computable in polynomial time.

Definition 1.3.2 (Computable function)**:** A function $f : [a, b] \subseteq \mathbb{R} \to \mathbb{R}$ is *computable* if there is a machine $\mathcal{M}$ such that for any $x \in I$ and any oracle $\mathcal{O}$ computing[3] x, $\mathcal{M}^{\mathcal{O}}$ computes[4][5] $f(x)$. Furthermore, f is *polynomial time* computable if $\mathcal{M}$ runs in polynomial time for all such oracles for all points in I. ♦

A crucial observation is that any computable function must be continuous [Ko91]. Furthermore, the modulus of continuity of a computable function on a closed interval must be computable[6].

Definition 1.3.3 (Modulus of continuity)**:** Let $f : [a, b] \to \mathbb{R}$ be a continuous function. A *modulus of continuity* is a function $m : \mathbb{N} \to \mathbb{N}$ such that for any $n > 0$, and $x, y \in [a, b]$, $|x - y| \leqslant 2^{-m(n)} \Rightarrow |f(x) - f(y)| \leqslant 2^{-n}$. ♦

Another direct observation is that the restriction of f to (dyadic) rational numbers is recursive. More precisely, if we consider $\psi : \mathbb{Q} \times \mathbb{N} \to \mathbb{Q}$ such that $|\psi(d, n) - f(d)| \leqslant 2^{-n}$, then ψ is recursive. One of the most important properties of computable functions is that these two properties characterize computable functions. These properties also extend to the case of polynomial time computable functions with the natural requirements.

Theorem 1.3.4 (Alternative definition of computable functions)**:** *A real function $f : [a, b] \to \mathbb{R}$ is computable (resp. polynomial time computable) if and only if there exists a computable (resp. polynomial time computable[7]) function $\psi : (\mathbb{Q} \cap [a, b]) \times \mathbb{N} \to \mathbb{Q}$ and a computable (resp. polynomial) function $m : \mathbb{N} \to \mathbb{N}$ such that:*

- *m is a modulus of continuity for f*
- *for any $n \in \mathbb{N}$ and $d \in [a, b] \cap \mathbb{Q}$, $|\psi(d, n) - f(d)| \leqslant 2^{-n}$*

♦

The complexity of real functions is usually studied over compact intervals such as $[a, b]$ or $[0, 1]$, but it can also be applied to the case of the whole line $\mathbb{R}$ by taking into account the maximal "size" of the input.

1.4 General Purpose Analog Computer

In 1941, Claude Shannon introduced a mathematical model for the Differential Analyzer [Bus31], on which he worked as an operator, called the General Purpose Analog Computer (GPAC) [Sha41]. The Differential Analyzer was a mechanical device used mostly in the thirties and the forties to solve ordinary differential equations (ODEs). A mechanical version of the Differential Analyzer was built for the first time at the MIT by Bush in 1931, based on the ideas of Thomson to connect integrators in order to solve differential equations, dating back to 1876.

[3]meaning that when called on a tape containing n, it writes a rational number r_n such that $|r_n - x| \leqslant 2^{-n}$

[4]$\mathcal{M}^{\mathcal{O}}$ means that $\mathcal{M}$ can call the oracle $\mathcal{O}$ by writing on a special tape and entering a special state

[5]meaning that for any integer n, $\mathcal{M}^{\mathcal{O}}(n)$ computes s_n such that $|s_n - f(x)| \leqslant 2^{-n}$

[6]in the sense of classical computability for integers, or –equivalently– recursive

[7]The second argument of g must be in unary.

This machine was a programmable table made up of gears, shafts, drums and tracing tables (see Figure 1.4.1). It could solve differential equations up to order 6, but was huge and slow. Later versions of the differential analyzer were built using electronic circuits and operational amplifiers. These versions offered greater speed and precision but the emergent digital computers eventually replaced all analog computers.

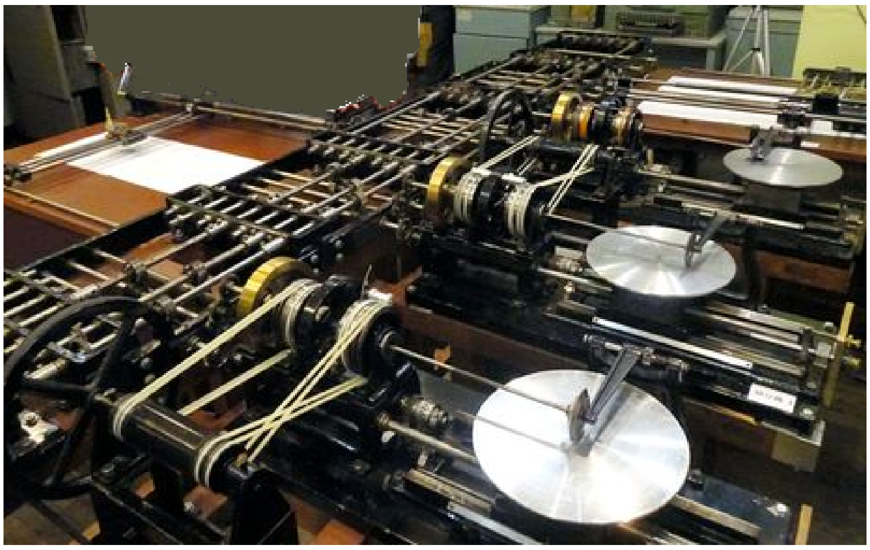


Figure 1.4.1: Photo of a restored GPAC at the Tokyo University of Science[8]

Originally the GPAC was presented as a mathematical model of the Differential Analyzer based on circuits (see Figure 1.0.1), where several units performing basic operations (e.g. sums, integration) are interconnected. Shannon stated [Sha41] that functions generated by the GPAC were differentially algebraic. Differentially algebraic functions satisfy a differential equation of the form $p(t, y(t), y'(t), \dots, y^{(k)}(t)) = 0$ for $t \in I$ where p is a polynomial.

Later on, Pour-El found an error in the proof [PE74] and redefined the GPAC in terms of quasi-linear differential equations. However this approach was again flawed and finally fixed by Graça and Costa [GC03] by restricting the set of allowed connections in the circuit[9]. In this later approach, functions generated by the GPAC are exactly those which satisfy polynomial differential equations.

Definition 1.4.2 (GPAC generable function)**:** $f : I \to \mathbb{R}$ is called *GPAC generable* if it is a component (i.e. $f = y_1$) of a solution of

$$y(0) = y_0 \qquad y'(t) = p(y(t)) \quad t \in I$$

where p is a vector of polynomials. ♦

This class of functions turns out to be more general than it may seem at first. Indeed, not only is it closed under the usual arithmetic operations but also one can replace p by any such generated function. For example, $y' = \sin(y)$ is in this class because sin is in this class (see Figure 1.0.2).

We have seen that *generable* functions capture Shannon's ideas about computability. However, this notion does not compare really well to existing classes because the *object* of the computation is really the graph of the function itself, or the *orbit* in dynamic system terms. This is

[8] Adaptation from a online public image, from `http://ajw.asahi.com/article/sci_tech/technology/AJ201412020060` with original photo taken by Hisatoshi Kabata. Copyright CC BY-SA 4.0

[9] To avoid badly defined circuits

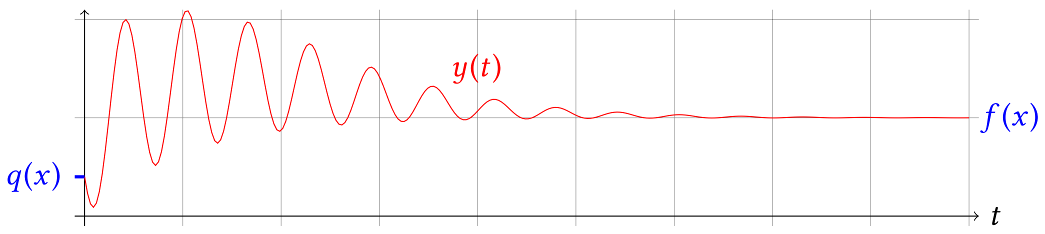


Figure 1.4.4: Graphical representation of a computation on input x

in contrast with classical computability, and especially Computable Analysis, where the orbit is not as interesting as the "limit" or stable state. For example, the equivalent of the graph for a Turing machine is the sequence of configurations from a given initial configuration. Using a generalization of the notion of generable functions, it was shown recently that Turing-based paradigms and the GPAC[10] have the same computation power [BCGH07, GCB08]. Figure 1.4.4 illustrates this notion of GPAC computability.

Definition 1.4.3 (GPAC computable function)**:** $f : \mathbb{R} \to \mathbb{R}$ is called *computable* if there exists polynomials p and q such that for any $x \in \mathbb{R}$, there exists (a unique) $y : I \to \mathbb{R}^d$ satisfying for all $t \in \mathbb{R}_+$:

- $y(0) = q(x)$ and $y'(t) = p(y(t))$ ► y satisfies a PIVP
- $|y_1(t) - f(x)| \leqslant e^{-t}$ where y_1 is the first component of y ► y_1 converges to $f(x)$

♦

1.5 Related work

In this section, we want to mention other related work, some close and some further away from the topic described in the thesis but very relevant to the context. Most of these are still under active research.

First and foremost, we would like to mention that there are many results on the decidability or complexity of problems related to dynamical systems, and differential equations in particular. It is known that the solution to an ODE of the form $y' = f(y)$ is computable if it is unique and f is computable [CG09]. It is also known that determining if the maximal interval of existence for PIVP is bounded is undecidable, even when the order and degree is fixed [GBC07, GBC09]. It has been shown previously that polynomial ODEs are sufficient to simulate Turing machines and Computable Analysis [BGZ11, GCB05, GCB08, BGP, BCGH07]. Other results on the basin of attractions and asymptotic properties are discussed in [BGZ11, BGPZ13, GZ09, GZ11]. In this work, we will intrinsically make use of robust computations – computations in the presence of perturbations – which were previously studied in [BGH10, BGH13, AB01]. Recent work also showed that Lipschitz continuous ODE are PSPACE-hard[11] [Kaw10], and interesting lower bounds have been obtained such as CH hardness for C^∞ ODEs [KORZ14].

Another aspect of this work is continuous time computation, which has a rich litterature and we point to [BC08] for the most recent survey on the subject. In particular we would like

[10]See Chapter 4 (Computation with PIVP) for more details on the meaning of computable by a GPAC

[11]A function $f : [0, 1] \to \mathbb{R}$ is PSPACE-hard if it can be used to solve as an oracle by a Turing machine to solve any problem in PSPACE.

to mention hybrid systems, signal machines [Dur12] and other physically inspired models [EN02, Fei88, WN05, BCPT14]. We would like to point out that contrary to most models, the GPAC is not only a physically inspired model, it is *the model of an existing machine* and, as such, hopefully captures the computational power of actual devices. Other such models include quantum-based paradigms [Deu85, Hir01], although most of them are only continuous in space but discrete in time.

Part of this thesis is also related to the very active field of (Computer-Aided) Verification. It is impossible to mention all the work in this area and many different techniques have been developed, ranging from bounded model checking to formal verification using logic. Many positive and negative results have been obtained, as well as very efficient algorithms, in the deterministic and probabilistic cases [Var85, CY95, KNSS02, APZ03, HLMP04, Bro99, AMP95, ASY07, PV94, Pla12]. Typical problems in this area include reachability analysis such as the one developed in this thesis. We would like to mention that PIVP are a particularly natural example of Cyber Physical Systems, which have gained much interest in the last decade.

Finally, a closely related field is that of formal methods, and series solutions of differential equations in particular. Of particular interest are D-finite series [Lip89] and CDF-series [BS95] which are very close to the generable functions in Chapter 2 (The PIVP class). The problem of quickly computing coefficients of such series has strong links to the numerical analysis of differential equations which have analytic solutions [WWS$^+$06, BCO$^+$07].

1.6 Notations

Notations for sets

Concept	Notation	Comment
Real interval	$[a,b]$	$\{x\in\mathbb{R} \mid a\leqslant x\leqslant b\}$
	$[a,b[$	$\{x\in\mathbb{R} \mid a\leqslant x< b\}$
	$]a,b]$	$\{x\in\mathbb{R} \mid a< x\leqslant b\}$
	$]a,b[$	$\{x\in\mathbb{R} \mid a< x< b\}$
Line segment	$[x,y]$	$\{(1-\alpha)x+\alpha y,\alpha\in[0,1]\}$
	$[x,y[$	$\{(1-\alpha)x+\alpha y,\alpha\in[0,1[\}$
	$]x,y]$	$\{(1-\alpha)x+\alpha y,\alpha\in]0,1]\}$
	$]x,y[$	$\{(1-\alpha)x+\alpha y,\alpha\in]0,1[\}$
Integer interval	$[\![a,b]\!]$	$\{a,a+1,\ldots,b\}$
Natural numbers	$\mathbb{N}$	$\{0,1,2,\ldots\}$
	$\mathbb{N}^*$	$\mathbb{N}\setminus\{0\}$
Integers	$\mathbb{Z}$	$\{\ldots,-2,-1,0,1,2,\ldots\}$
Rational numbers	$\mathbb{Q}$	
Dyadic rationnals	$\mathbb{D}$	$\{m2^{-n},m\in\mathbb{Z},n\in\mathbb{N}\}$
Real numbers	$\mathbb{R}$	
Non-negative numbers	$\mathbb{R}_+$	$\mathbb{R}_+=[0,+\infty[$
Non-zero numbers	$\mathbb{R}^*$	$\mathbb{R}^*=\mathbb{R}\setminus\{0\}$
Positive numbers	$\mathbb{R}_+^*$	$\mathbb{R}_+^*=]0,+\infty[$
Complex numbers	$\mathbb{C}$	
Set shifting	$x+Y$	$\{x+y,y\in Y\}$

Notations for sets

Concept	Notation	Comment
Set addition	$X + Y$	$\{x + y, x \in X, y \in Y\}$
A field	$\mathbb{K}$	If unspecified, refers to any field
Matrices	$M_{n,m}(\mathbb{K})$	Set of $n \times m$ matrices over field $\mathbb{K}$
	$M_n(\mathbb{K})$	Shorthand for $M_{n,n}(\mathbb{K})$
	$M_{n,m}$	Field is deduced from the context
Polynomials	$\mathbb{K}[X_1, \ldots, X_n]$	Ring of polynomials with variables $X_1, \ldots, X_n$ and coefficients in $\mathbb{K}$
	$\mathbb{K}[\mathbb{A}^n]$	Polynomial functions with n variables, coefficients in $\mathbb{K}$ and domain of definition $\mathbb{A}^n$
Fractions	$\mathbb{K}(X)$	Field of rational fractions with coefficients in $\mathbb{K}$
Power set	$\mathcal{P}(X)$	The set of all subsets of X
Domain of definition	$\operatorname{dom} f$	If $f : I \to J$ then $\operatorname{dom} f = I$
Cardinal	$\#X$	Number of elements
Polynomial vector	$\mathbb{K}^n[\mathbb{A}^d]$	Polynomial in d variables with coefficients in $\mathbb{K}^n$
	$\mathbb{K}[\mathbb{A}^d]^n$	Isomorphic $\mathbb{K}^n[\mathbb{A}^d]$
Polynomial matrix	$M_{n,m}(\mathbb{K})[\mathbb{A}^n]$	Polynomial in n variables with matrix coefficients
	$M_{n,m}(\mathbb{K}[\mathbb{A}^n])$	Isomorphic $M_{n,m}(\mathbb{K})[\mathbb{A}^n]$
Smooth functions	C^k	Partial derivatives of order k exist and are continuous
	C^∞	Partial derivatives exist at all orders
Real analytic functions	C^ω	Taylor series converge in the neighborhood of every point

Complexity classes

Concept	Notation	Comment
Polynomial Time	P	Class of decidable languages
	FP	Class of computable functions
Computable numbers	$\mathbb{R}_c$	See Definition 1.3.1 (Computable real)
Polynomial Space	PSPACE	Class of decidable languages
Polynomial time computable numbers	$\mathbb{R}_P$	See Definition 1.3.1 (Computable real)
Generable rationals	$\mathbb{R}_G$	See Definition 2.7.14 (Generable real numbers)
GPAC computability	$\mathrm{AC}(\Upsilon, \Omega)$	See Definition 4.2.1 (Analog computability)
	AP	Polynomial computability
Weak computability	$\mathrm{AW}(\Upsilon, \Omega)$	See Definition 4.2.7 (Analog weak computability)
	AWP	Polynomial weak computability
Robust computability	$\mathrm{AR}(\Upsilon, \Omega,)$	See Definition 4.3.1 (Analog robust computability)
	ARP	Polynomial robust computability
Strong computability	$\mathrm{AS}(\Upsilon, \Omega, \Theta)$	See Definition 4.3.9 (Analog strong computability)
	ASP	Polynomial strong computability
Online computability	$\mathrm{AO}(\Upsilon, \Omega, \Lambda)$	See Definition 4.4.17 (Online computability)

Complexity classes

Concept	Notation	Comment
	AOP	Polynomial online computability
Extreme computability	$\mathrm{AX}(\Upsilon, \Omega, \Lambda, \Theta)$	See Definition 4.4.2 (Extreme computability)
	AXP	Polynomial extreme computability

Metric spaces and topology

Concept	Notation	Comment
p-norm	$\|x\|_p$	$\left(\sum_{i=1}^{n} \|x_i\|^p\right)^{\frac{1}{p}}$
Infinity norm	$\|x\|$	$\max(\|x_1\|, \dots, \|x_n\|)$
Open ball	$B_r(u)$	Center is u, radius is r
Closed ball	$\overline{B}_r(u)$	Center is u, radius is r
Closure	$\overline{X}$	
Interior	$\mathring{X}$	
Border	∂X	

Notations for polynomials

Concept	Notation	Comment
Univariate polynomial	$\sum_{i=0}^{d} a_i X^i$	
Multi-index	α	$(\alpha_1, \dots, \alpha_k) \in \mathbb{N}^k$
	$\|\alpha\|$	$\alpha_1 + \cdots + \alpha_k$
	$\alpha!$	$\alpha_1!\alpha_2! \cdots \alpha_k!$
Multivariate polynomial	$\sum_{\|\alpha\| \leqslant d} a_\alpha X^\alpha$	where $X^\alpha = X_1^{\alpha_1} \cdots X_k^{\alpha_k}$
Degree	$\deg(P)$	Maximum degree of a monomial, X^α is of degree $\|\alpha\|$, conventionally $\deg(0) = -\infty$
	$\deg(P)$	$\max(\deg(P_i))$ if $P = (P_1, \dots, P_n)$
	$\deg(P)$	$\max(\deg(P_{ij}))$ if $P = (P_{ij})_{i \in [\![1,n]\!], j \in [\![1,m]\!]}$
Sum of coefficients	ΣP	$\Sigma P = \sum_\alpha \|a_\alpha\|$
	ΣP	$\max(\Sigma P_1, \dots, \Sigma P_n)$ if $P = (P_1, \dots, P_n)$
	ΣP	$\max(\Sigma P_{ij})$ if $P = (P_{ij})_{i \in [\![1,n]\!], j \in [\![1,m]\!]}$
A polynomial	poly	An unspecified polynomial

Miscellaneous functions

Concept	Notation	Comment
Sign function	$\operatorname{sgn}(x)$	Conventionally $\operatorname{sgn}(0) = 0$
Integer part function	$\lfloor x \rfloor$	$\max\{n \in \mathbb{Z} \mid n \leqslant x\}$

Miscellaneous functions

Concept	Notation	Comment
Ceiling function	$\lceil x \rceil$	$\min\{n \in \mathbb{Z} \mid x \leqslant n\}$
Rounding function	$\lfloor x \rceil$	$\operatorname{argmin}_{n\in\mathbb{Z}} \lvert n - x \rvert$, undefined for $x = n + \frac{1}{2}$
Integer part function	$\operatorname{int}(x)$	$\max(0, \lfloor x \rfloor)$
	$\operatorname{int}_n(x)$	$\min(n, \operatorname{int}(x))$
Fractional part function	$\operatorname{frac}(x)$	$x - \operatorname{int} x$
	$\operatorname{frac}_n(x)$	$x - \operatorname{int}_n(x)$
Composition operator	$f \circ g$	$(f \circ g)(x) = f(g(x))$
Identity function	id	$\operatorname{id}(x) = x$
Indicator function	$\mathbb{1}_X$	$\mathbb{1}_X(x) = 1$ if $x \in X$ and $\mathbb{1}_X(x) = 0$ otherwise
n^{th} iterate	$f^{[n]}$	$f^{[0]} = \operatorname{id}$ and $f^{[n+1]} = f^{[n]} \circ f$

Calculus

Concept	Notation	Comment
Derivative	f'	
n^{th} derivative	$f^{(n)}$	$f^{(0)} = f$ and $f^{(n+1)} = f^{(n)\prime}$
Partial derivative	$\partial_i f, \frac{\partial f}{\partial x_i}$	with respect to the i^{th} variable
Scalar product	$x \cdot y$	$\sum_{i=1}^{n} x_i y_i$ in $\mathbb{R}^n$
Gradient	$\nabla f(x)$	$(\partial_1 f(x), \dots, \partial_n f(x))$
Jacobian matrix	$J_f(x)$	$(\partial_j f_i(x))_{i\in[\![1,n]\!], j\in[\![1,m]\!]}$
Taylor approximation	$T_a^n f(t)$	$\displaystyle\sum_{k=0}^{n-1} \frac{f^{(k)}(a)}{k!}(t-a)^k$
Big O notation	$f(x) = \mathcal{O}(g(x))$	$\exists M, x_0 \in \mathbb{R}, \lvert f(x) \rvert \leqslant M \lvert g(x) \rvert$ for all $x \geqslant x_0$
Soft O notation	$f(x) = \tilde{\mathcal{O}}(g(x))$	Means $f(x) = \mathcal{O}\left(g(x) \log^k g(x)\right)$ for some k
Subvector	$x_{i..j}$	$(x_i, x_{i+1}, \dots, x_j)$
Matrix transpose	M^T	
Past supremum	$\sup_\delta f(t)$	$\sup_{u\in[t,t-\delta]\cap\mathbb{R}_+} f(t)$
Partial function	$f :\subseteq X \to Y$	$\operatorname{dom} f \subseteq X$
Restriction	$f\restriction_I$	$f\restriction_I(x) = f(x)$ for all $x \in \operatorname{dom} f \cap I$

Words

Concept	Notation	Comment
Alphabet	Σ, Γ	A finite set
Words	Σ^*	$\bigcup_{n\geqslant 0} \Sigma^n$
Empty word	λ	
Letter	w_i	i^{th} letter, starting from one
Subword	$w_{i..j}$	$w_i w_{i+1} \cdots w_j$
Length	$\lvert w \rvert$	

Words

Concept	Notation	Comment
Repetition	w^k	$\underbrace{ww\cdots w}_{k \text{ times}}$

Chapter 2

The PIVP class

Graphs of higher degree polynomials have this habit of doing unwanted wiggly things.

Quote from Cambridge

In this chapter we study the class of polynomial initial value problems (PIVP) in detail. We introduce the notion of *generable* functions which are solutions of a PIVP, and generalize this notion to several input variables. We will see that this class enjoys a number of stability properties which makes it suitable for use as a basis for more advanced work. We recall that a PIVP is a system of differential equations of the form:

$$\begin{cases} y'(t) = p(y(t)) \\ y(t_0) = y_0 \end{cases} \qquad t \in \mathbb{R} \tag{2.0.1}$$

where p is a vector of polynomials and $y(t)$ is vector. In other words, $y_i'(t) = p_i(y(t))$ where p_i is a multivariate polynomial.

This chapter is organized as follows:

- Section 2.1 (Generable functions) will introduce the notion of *generable* function, in the unidimensional and multidimensional case.
- Section 2.2 (Stability properties) will give a few stability properties of this class, mostly stability by arithmetic operations, composition and ODE solving.
- Section 2.3 (Analyticity of generable functions) will show that generable functions are always analytic
- Section 2.4 (Dependency in the parameters) will give a few results on the dependency of solutions to PIVP with respect to perturbations.
- Section 2.5 (Taylor series of the solutions) will recall results about the Taylor series of solutions to PIVP.
- Section 2.6 (Generable zoo) will give a list of useful generable functions, both for later use in subsequent chapters, and as a way to see what can be achieved with generable functions.
- Section 2.7 (Generable fields) will give a few properties of *generable fields* which are fields with an extra property related to generable functions.

2.1 Generable functions

In this section, we will define a notion of function generated by a PIVP. This class of functions is closed by a number of natural operations such as arithmetic operators, composition. In particular, we will see that those functions are always analytic The major property of this class is the stability by ODE solving: if f is *generable* and y satisfies $y' = f(y)$ then y is generable. This means that we can design differential systems where the right-hand side contains much more general functions than polynomials, and this system can be rewritten to use polynomials only.

In this section, $\mathbb{K}$ will always refer to a real field, for example $\mathbb{K} = \mathbb{Q}$. The basic definitions work for any such field but the main results will require some assumptions on $\mathbb{K}$. These assumptions will be formalized in Definition 2.1.8 (Generable field) and detailed in Section 2.7 (Generable fields).

2.1.I Unidimensional case

We start with the definition of generable functions from $\mathbb{R}$ to $\mathbb{R}^n$. Those are defined as the solution of some polynomial IVP (PIVP) with an additional boundedness constraint.

Definition 2.1.1 (Generable function)**:** Let $\mathtt{sp} : \mathbb{R}_+ \to \mathbb{R}_+$ and $f : \mathbb{R} \to \mathbb{R}^e$. We say that $f \in \mathrm{GVAL}_{\mathbb{K}}(\mathtt{sp})$ if and only if there exists $n \in \mathbb{N}$, $y_0 \in \mathbb{K}^n$ and $p \in \mathbb{K}^n[\mathbb{R}^n]$ such that there is a (unique) $y : \mathbb{R} \to \mathbb{R}^n$ satisfying for all time $t \in \mathbb{R}$:

- $y'(t) = p(y(t))$ and $y(0) = y_0$ ► y satisfies a differential equation
- $f(t) = y_{1..e}(t) = (y_1(t), \dots, y_e(t))$ ► f is a component of y
- $\|y(t)\| \leqslant \mathtt{sp}(|t|)$ ► y is bounded by $\mathtt{sp}$

The set of all generable functions is denoted by $\mathrm{GVAL}_{\mathbb{K}} = \bigcup_{\mathtt{sp}:\mathbb{R}\to\mathbb{R}_+} \mathrm{GVAL}_{\mathbb{K}}(\mathtt{sp})$. When this is not ambiguous, we do not specify the field $\mathbb{K}$ and write $\mathrm{GVAL}(\mathtt{sp})$ or simply GVAL. ♦

Remark 2.1.2 (Uniqueness)**:** The uniqueness of y in Definition 2.1.1 is a consequence of the Cauchy-Lipschitz theorem. Indeed a polynomial is a locally Lipschitz function. ♦

Remark 2.1.3 (Regularity)**:** As a consequence of the Cauchy-Lipschitz theorem, the solution y in Definition 2.1.1 (Generable function) is at least C^∞. It can be seen that it is in fact real analytic, as it is the case for analytic differential equations in general. ♦

Remark 2.1.4 (Multidimensional output)**:** It should be noted that although Definition 2.1.1 defines generable functions with output in $\mathbb{R}^e$, it is completely equivalent to say that f is generable if and only if each of its component is (*i.e.* f_i is generable for every i); and restrict the previous definition to functions from $\mathbb{R}$ to $\mathbb{R}$ only.

Also note that if y is the solution from Definition 2.1.1, then obviously y is generable. ♦

Although this might not be obvious at first glance, this class generalizes polynomials, and contains many elementary functions such as the exponential function, as well as the trigonometric functions. Intuitively, all functions in this class can be computed efficiently by classical machines, where $\mathtt{sp}$ measures some "hardness" in computing the function. We took care to choose the constants such as the initial time and value, and the coefficients of the polynomial in $\mathbb{K}$. The idea is to prevent any uncomputability from arising by the choice of uncomputable real numbers in the constants.

Example 2.1.5 (Polynomials are generable)**:** Let p in $\mathbb{Q}(\pi)[\mathbb{R}]$. For example $p(x) = x^7 - 14x^3 + \pi^2$. We will show that $p \in \operatorname{GVAL}_{\mathbb{K}}(\mathtt{sp})$ where $\mathtt{sp}(x) = \max(|x|, |p(x)|)$. We need to rewrite p with a polynomial differential equation: we immediately get that $p(0) = \pi^2$ and $p'(x) = 6x^6 - 42x^2$. However, we cannot express $p'(x)$ as a polynomial of $p(x)$ only: we need access to x. This can be done by introducing a new variable $v(x)$ such that $v(x) = x$. Indeed, $v'(x) = 1$ and $v(0) = 0$. Finally we get:

$$\begin{cases} p(0) = \pi^2 \\ p'(x) = 6v(x)^6 - 42v(x)^2 \end{cases} \qquad \begin{cases} v(0) = 0 \\ v'(x) = 1 \end{cases}$$

Formally, we define $y(x) = (p(x), x)$ and show that $y(0) = (\pi^2, 0) \in \mathbb{K}^2$ and $y'(x) = p(y(x))$ where $p_1(a, b) = 6b^6 - 42b^2$ and $p_2(a, b) = 1$. Also note that the coefficients are clearly in $\mathbb{Q}(\pi)$). We also need to check that $\mathtt{sp}$ is a bound on $\|y(x)\|$:

$$\|y(x)\| = \max(|x|, |p(x)|) = \mathtt{sp}(x)$$

This shows that $p \in \operatorname{GVAL}_{\mathbb{K}}(\mathtt{sp})$ and can be generalized to show that any polynomial in one variable is generable. ♦

Example 2.1.6 (Some generable elementary functions)**:** We will check that $\exp \in \operatorname{GVAL}_{\mathbb{Q}}(\exp)$ and $\sin, \cos, \tanh \in \operatorname{GVAL}_{\mathbb{Q}}(x \mapsto 1)$. We will also check that $\arctan \in \operatorname{GVAL}_{\mathbb{Q}}(x \mapsto \max(x, \frac{\pi}{2}))$.

- A characterization of the exponential function is the following: $\exp(0) = 1$ and $\exp' = \exp$. Since $\|\exp\| = \exp$, it is immediate that $\exp \in \operatorname{GVAL}_{\mathbb{Q}}(\exp)$. The exponential function might be the simplest generable function.

- The sine and cosine functions are related by their derivatives since $\sin' = \cos$ and $\cos' = -\sin$. Also $\sin(0) = 0$ and $\cos(0) = 1$, and $\|\sin(x), \cos(x)\| \leqslant 1$, we get that $\sin, \cos \in \operatorname{GVAL}_{\mathbb{Q}}(x \mapsto 1)$ with the same system.

- The hyperbolic tangent function will be very useful in this chapter. Is it known to satisfy the very simple polynomial differential equation $\tanh' = 1 - \tanh^2$. Since $\tanh(0) = 0$ and $|\tanh(x)| \leqslant 1$, this shows that $\tanh \in \operatorname{GVAL}_{\mathbb{Q}}(x \mapsto 1)$.

- Another very useful function will be the arctangent function. A possible definition of the arctangent is the unique function satisfing $\arctan(0) = 0$ and $\arctan'(x) = \frac{1}{1+x^2}$. Unfortunately this is neither a polynomial in $\arctan(x)$ nor in x. A common trick is to introduce a new variable $z(x) = \frac{1}{1+x^2}$ so that $\arctan'(x) = z(x)$, in the hope that z satisfies a PIVP. This is the case since $z(0) = 1$ and $z'(x) = \frac{-2x}{(1+x^2)^2} = -2xz(x)^2$ which is a polynomial in z and x. We introduce a new variable for x as we did in the previous examples. Finally, define $y(x) = (\arctan(x), \frac{1}{1+x^2}, x)$ and check that $y(0) = (0, 1, 0)$ and $y'(x) = (y_2(x), -2y_3(x)y_2(x)^2, 1)$. The $\frac{\pi}{2}$ bound on arctan is a textbook property, and the bound on the other variables is immediate.

♦

Not only the class of generable functions contains many classical and useful functions, but it is also closed under many operations. We will see that the sum, difference, product and composition of generable functions is still generable. Before moving on to the properties of this class, we need to mention the easily overlooked issue about constants, best illustrated as an example.

Example 2.1.7 (The issue of constants)**:** Let $\mathbb{K}$ be a field, containing at least the rational numbers. Assume that generable functions are closed under composition, that is for any two $f, g \in \text{GVAL}_{\mathbb{K}}$ we have $f \circ g \in \text{GVAL}_{\mathbb{K}}$. Let $\alpha \in \mathbb{K}$ and $g = x \mapsto \alpha$. Then for any $(f : \mathbb{R} \to \mathbb{R}) \in \text{GVAL}_{\mathbb{K}}$, $f \circ g \in \text{GVAL}_{\mathbb{K}}$. Using Definition 2.1.1 (Generable function), we get that $f(g(0)) \in \mathbb{K}$ which means $f(\alpha) \in \mathbb{K}$ for any $\alpha \in \mathbb{K}$. In other words, $\mathbb{K}$ must satisfy the following property:

$$f(\mathbb{K}) \subseteq \mathbb{K} \qquad \forall f \in \text{GVAL}_{\mathbb{K}}$$

This property does not hold for general fields. ♦

The example above outlines the need for a stronger hypothesis on $\mathbb{K}$ if we want to be able to compose functions. Motivated by this example, we introduce the following notion of *generable field*.

Definition 2.1.8 (Generable field)**:** A field $\mathbb{K}$ is *generable* if and only if $\mathbb{Q} \subseteq \mathbb{K}$ and for any $\alpha \in \mathbb{K}$ and $(f : \mathbb{R} \to \mathbb{R}) \in \text{GVAL}_{\mathbb{K}}$, we have $f(\alpha) \in \mathbb{K}$. ♦

⚠ From now on, we will assume that $\mathbb{K}$ is a generable field. See Section 2.7 (Generable fields) for more details on this assumption.

Example 2.1.9 (Usual constants are generable)**:** In this manuscript, we will use again and again that some well-known constants belong to any generable field. We detail the proof for π and e:

- It is well-known that $\frac{\pi}{4} = \arctan(1)$. We saw in Example 2.1.6 (Some generable elementary functions) that $\arctan \in \text{GVAL}_{\mathbb{Q}}$ and since $1 \in \mathbb{K}$ we get that $\frac{\pi}{4} \in \mathbb{K}$ because $\mathbb{K}$ is a generable field. We conclude that $\pi \in \mathbb{K}$ because $\mathbb{K}$ is a field and $4 \in \mathbb{K}$.

- By definition, $e = \exp(1)$ and $\exp \in \text{GVAL}_{\mathbb{Q}}$, so $e \in \mathbb{K}$ because $\mathbb{K}$ is a generable field and $1 \in \mathbb{K}$.

♦

Lemma 2.1.10 (Arithmetic on generable functions)**:** *Let $f \in \text{GVAL}(\mathrm{sp})$ and $g \in \text{GVAL}(\overline{\mathrm{sp}})$.*

- $f + g, f - g \in \text{GVAL}(\mathrm{sp} + \overline{\mathrm{sp}})$
- $fg \in \text{GVAL}(\max(\mathrm{sp}, \overline{\mathrm{sp}}, \mathrm{sp}\,\overline{\mathrm{sp}}))$
- $\frac{1}{f} \in \text{GVAL}(\max(\mathrm{sp}, \mathrm{sp}'))$ *where* $\mathrm{sp}'(t) = \frac{1}{|f(t)|}$, *if f never cancels*
- $f \circ g \in \text{GVAL}(\max(\overline{\mathrm{sp}}, \mathrm{sp} \circ \overline{\mathrm{sp}}))$

Note that the first three items only require that $\mathbb{K}$ is a field, whereas the last item also requires $\mathbb{K}$ to be a generable field. ♦

Proof. Assume that $f : \mathbb{R} \to \mathbb{R}^m$ and $g : \mathbb{R} \to \mathbb{R}^e$. We will make a detailed proof of the product and composition cases, since the sum and difference are much simpler. The intuition follows from basic differential calculus and the chain rule: $(fg)' = f'g + fg'$ and $(f \circ g)' = g'(f' \circ g)$. Note that $e = 1$ for the composition to make sense and $e = m$ for the product to make sense (componentwise). The only difficulty in this proof is technical: the differential equation may include more variables than just the ones computing f and g. This requires a bit of notation to

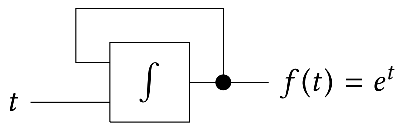


Figure 2.1.11: Simple GPAC

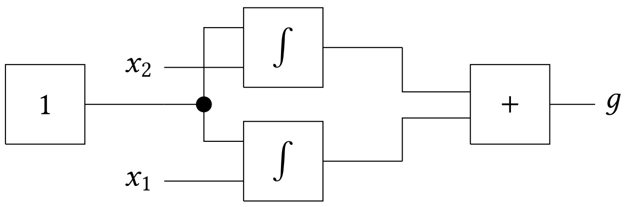


Figure 2.1.12: GPAC with two inputs

stay formal. Apply Definition 2.1.1 (Generable function) to f and g to get $p, \overline{p}, y_0, \overline{y}_0$. Consider the following systems:

$$\begin{cases} y(0) = y_0 \\ y'(t) = p(y(t)) \\ \overline{y}(0) = \overline{y}_0 \\ \overline{y}'(t) = \overline{p}(\overline{y}(t)) \end{cases} \qquad \begin{cases} z_i(0) = y_{0,i}\overline{y}_{0,i} \\ z_i'(t) = p_i(y(t))z_i(t) + y_i(t)\overline{p}_i(z(t)) \\ u_i(0) = f_i(\overline{y}_{0,1}) \\ u_i'(t) = \overline{p}_1(\overline{y}(t))p(u(t)) \end{cases} \qquad i \in [\![1, m]\!]$$

Those systems are clearly polynomial. Remember that by definition, for any $i \in [\![1, m]\!]$ and $j \in [\![1, e]\!]$, $f_i(t) = y_i(t)$ and $g_j(t) = z_j(t)$, and as a particular case $f_i(0) = y_{0,i}$ and $g_j(0) = \overline{y}_{0,j}$. Consequently, $z_i(t) = f_i(t)g_i(t)$ and $u_i(t) = f_i(g_1(t))$.

Also by definition, $\|y(t)\| \leqslant \mathtt{sp}(t)$ and $\|\overline{y}(t)\| \leqslant \overline{\mathtt{sp}}(t)$. It follows that $|z_i(t)| \leqslant |y_i(t)||\overline{y}_i(t)| \leqslant \mathtt{sp}(t)\overline{\mathtt{sp}}(t)$, and similarly $|u_i(t)| \leqslant |f_i(g_1(t))| \leqslant \mathtt{sp}(g_1(t)) \leqslant \mathtt{sp}(\overline{\mathtt{sp}}(t))$.

The case of $\frac{1}{g}$ is very similar: define $g = \frac{1}{f}$ then $g' = -f'g^2$. The only difference is that we don't have an a priori bound on g except $\frac{1}{|f|}$, and we must assume that f is never zero for g to be defined over $\mathbb{R}$.

Finally, a very important note about constants and coefficients which appear in those systems. It is clear that $y_{0,i}\overline{y}_{0,i} \in \mathbb{K}$ because $\mathbb{K}$ is a field. Similarly, for $\frac{1}{f}$ we have $\frac{1}{f(0)} = \frac{1}{y_{0,1}} \in \mathbb{K}$. However, there is no reason in general for $f_i(\overline{y}_{0,1})$ to belong to $\mathbb{K}$, and this is where we need the assumption that $\mathbb{K}$ is generable to conclude. ■

2.1.II Multidimensional case

We introduced generable functions as a special kind of function from $\mathbb{R}$ to $\mathbb{R}^n$. We saw that this class nicely generalizes polynomials, however it comes with two defects which prevents other interesting functions from being generable:

- The domain of definition is $\mathbb{R}$: this is very strong, since other "easy" targets such as tan, log or even $x \mapsto \frac{1}{x}$ cannot be defined, despite satisfying polynomial differential equations.
- The domain of definition is one dimensional: it would be useful to define generable functions in several variables, like multivariate polynomials.

The first issue can be dealt with by adding restrictions on the domain where the differentiable equation holds, and shifting the initial condition (0 might not belong to the domain). Overcoming the second problem is less obvious.

The examples below give two intuitions before introducing the formal definition. The first example draws inspiration from multivariate calculus and differential form theory. The second example focuses on GPAC composition. As we will see, both examples highlight the same properties of multidimensional generable functions.

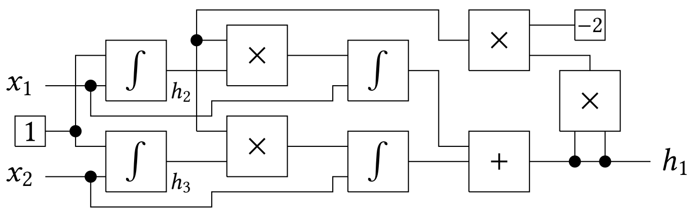


Figure 2.1.13: A more involved multidimensional GPAC

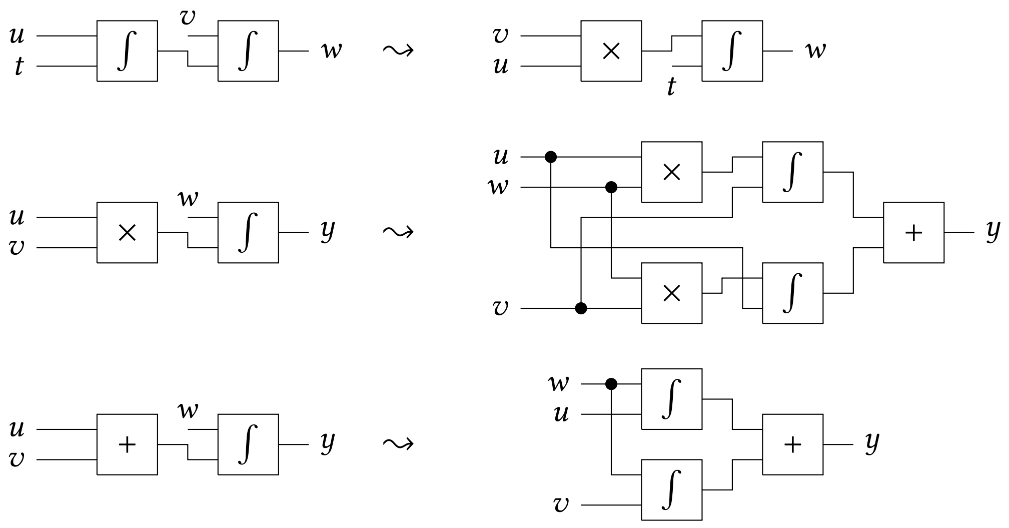


Figure 2.1.14: GPAC rewriting

Example 2.1.15 (Multidimensional GPAC)**:** The history and motivation for the GPAC is described in Section 1.4 (General Purpose Analog Computer). The GPAC is the starting point for the definition of generable functions. It crucially relies on the integrator unit to build interesting circuits. In modern terms, the integration is often done implicitly with respect to time, as shown in Figure 2.1.11 (Simple GPAC) where the corresponding equation is $f(t) = \int f$, or $f' = f$. Notice that the circuit has a single "floating input" which is t and is only used in the "derivative port" of the integrator. What would be the meaning of a circuit with several such inputs, as shown in Figure 2.1.12 (GPAC with two inputs) ? Formally writing the system and differentiating gives:

$$\begin{aligned} g &= \int 1dx_1 + \int 1dx_2 = x_1 + x_2 \\ dg &= dx_1 + dx_2 \end{aligned}$$

Figure 2.1.13 (A more involved multidimensional GPAC) gives a more interesting example to better grasp the features of these GPAC. Using the same "trick" as before we get:

$$\begin{aligned} h_2 &= \int 1dx_1 & dh_2 &= dx_1 \\ h_3 &= \int 1dx_2 & dh_3 &= dx_2 \\ h_1 &= \int -2h_1^2h_2dx_1 + \int -2h_1^2h_3dx_2 & dh_1 &= -2h_1^2h_2dx_1 - 2h_1^2h_3dx_2 \end{aligned}$$

It is now apparent that the computed function h satisfies a special property because $dh_1(x) = p_1(h_1,h_2,h_3)dx_1 + p_2(h_1,h_2,h_3)dx_2$ where p_1 and p_2 are polynomials. In other words, $dh_1 = p(h) \cdot dx$ where $h = (h_1,h_2,h_3)$, $x = (x_1,x_2)$ and $p = (p_1,p_2)$ is a polynomial vector. We obtain similar equations for h_2 and h_3. Finally, $dh = q(h)dx$ where $q(h)$ is the polynomial matrix given by:

$$q(h) = \begin{pmatrix} -2h_1^2h_2 & -2h_1^2h_3 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

This can be equivalently stated as $J_h = q(h)$. This is a generalization of PIVP to polynomial partial differential equations.

To complete this example, note that it can be solved exactly and $h_1(x_1, x_2) = \frac{1}{x_1^2+x_2^2}$ which is defined over $\mathbb{R}^2 \setminus \{(0,0)\}$. ♦

Example 2.1.16 (GPAC composition)**:** Another way to look at Figure 2.1.13 (A more involved multidimensional GPAC) and Figure 2.1.12 (GPAC with two inputs) is to imagine that $x_1 = X_1(t)$ and $x_2 = X_2(t)$ are functions of the time (produced by other GPACs), and rewrite the system in the time domain with $h = H(t)$:

$$\begin{aligned} H_2'(t) &= X_1'(t) \\ H_3'(t) &= X_2'(t) \\ H_1'(t) &= -2H_1(t)^2H_2(t)X_1'(t) - 2H(t)^2H_3(t)X_2'(t) \end{aligned}$$

We obtain a system similar to the unidimensional PIVP: for a given choice of X we have $H'(t) = q(H(t))X'(t)$ where $q(h)$ is the polynomial matrix given by:

$$q(h) = \begin{pmatrix} -2h_1^2h_2 & -2h_1^2h_3 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Note that this is the same polynomial matrix as in the previous example. The relationship between the time domain H and the original h is simply given by $H(t) = h(x(t))$. This approach has a natural interpretation on the GPAC circuit in terms of circuit rewriting. Assume that x_1 and x_2 are the outputs of two GPAC (with input t), then we can build the circuit computing H using the rule of Figure 2.1.14 (GPAC rewriting). In a normal GPAC, the time t is the only valid input of the derivative port of the integrator, so we need to rewrite integrators which violate this rule. This procedure always stops in finite time. ♦

Definition 2.1.17 (Generable function)**:** Let $d, e \in \mathbb{N}$, I an open and connected subset of $\mathbb{R}^d$, $\mathtt{sp} : \mathbb{R} \to \mathbb{R}_+$ and $f : I \to \mathbb{R}^e$. We say that $f \in \text{GVAL}_{\mathbb{K}}(\mathtt{sp})$ if and only if there exists $n \geqslant e$, $p \in M_{n,d}(\mathbb{K})[\mathbb{R}^n]$, $x_0 \in \mathbb{K}^d$, $y_0 \in \mathbb{K}^n$ and $y : I \to \mathbb{R}^n$ satisfying for all $x \in I$:

- $y(x_0) = y_0$ and $J_y(x) = p(y(x))$ (i.e. $\partial_j y_i(x) = p_{ij}(y(x))$)► y satisfies a differential equation
- $f(x) = y_{1..e}(x)$ ► f is a component of y
- $\|y(x)\| \leqslant \mathtt{sp}(\|x\|)$ ► y is bounded by $\mathtt{sp}$

♦

Remark 2.1.18 (Uniqueness)**:** The uniqueness of y in Definition 2.1.17 (Generable function) can be seen as follows: consider $x \in I$ and γ a smooth curve[1] from x_0 to x with values in I and consider $z(t) = y(\gamma(t))$ for $t \in [0,1]$. It can be seen that $z'(t) = J_y(\gamma(t))\gamma'(t) = p(y(\gamma(t))\gamma'(t) =$

[1]see Remark 2.1.20 (Domain of definition)

$p(z(t))\gamma'(t)$, $z(0) = y(x_0) = y_0$ and $z(1) = y(x)$. The initial value problem $z(0) = y_0$ and $z'(t) = p(z(t))\gamma'(t)$ satisfies the hypothesis of the Cauchy-Lipschitz theorem and as such admits a unique solution. Since this IVP is independent of y, it shows that $y(x)$ must be unique. Note that the existence of y (and thus the domain of definition) is an hypothesis of the definition. ♦

Remark 2.1.19 (Regularity)**:** In the euclidean space $\mathbb{R}^n$, C^k smoothness is equivalent to the smoothness of the order k partial derivatives. Consequently, the equation $J_y = p(y)$ on the open set I immediately proves that y is C^∞. Proposition 2.3.3 (Generable implies analytic) shows that y is in fact real analytic. ♦

Remark 2.1.20 (Domain of definition)**:** Definition 2.1.17 (Generable function) requires the domain of definition of f to be connected, otherwise it would not make sense. Indeed, we can only define the value of f at point u if there exists a path from x_0 to u in the domain of f. It could seem, at first sight, that the domain being "only" connected may be too weak to work with. This is not the case, because in the euclidean space $\mathbb{R}^d$, *open* connected subsets are always smoothly arc connected, that is any two points can be connected using a smooth C^1 (and even C^∞) arc. Proposition 2.7.4 (Generable path connectedness) extends this idea to generable arcs, with a very useful corollary. ♦

Remark 2.1.21 (Multidimensional output)**:** Remark 2.1.4 (Multidimensional output) also applies to this definition: $f :\subseteq \mathbb{R}^d \to \mathbb{R}^n$ is generable if and only if each of its component is generable (*i.e.* f_i is generable for all i). ♦

Remark 2.1.22 (Definition consistency)**:** It should be clear that Definition 2.1.17 (Generable function) and Definition 2.1.1 (Generable function) are consistent. More precisely, in the case of unidimensional function ($d = 1$) with domain of definition $I = \mathbb{R}$, both definitions are exactly the same since $J_y = y'$ and $M_{n,1}(\mathbb{R}) = \mathbb{R}^n$. ♦

Definition 2.1.23 (Polynomially bounded generable function)**:** The class of generable functions with polynomially bounded value is called GPVAL:

$$f \in \text{GPVAL} \Leftrightarrow \exists \mathtt{sp} \text{ a polynomial such that } f \in \text{GVAL}(\mathtt{sp})$$

♦

Example 2.1.24 (Generable function)**:** Let $f(x_1, x_2) = x_1 x_2^4$ which is defined over $\mathbb{R}^2$. Let $\mathtt{sp}(\alpha) = (1+\alpha)(1+\alpha^4)$. We will show that $f \in \text{GVAL}_{\mathbb{Q}}(\mathtt{sp})$. Define $y(x_1, x_2) = \left(x_1 x_2^4, x_1, x_2\right)$, and check that:

$$J_y(x_1, x_2) = \begin{pmatrix} x_2^4 & 4x_1x_2^3 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} y_3^4 & 3y_2y_3^3 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = q(y)$$

where $q \in M_{3,2}(\mathbb{Q})[\mathbb{R}^3]$ is a polynomial matrix. Furthermore, $y(0,0) = (0,0,0) \in \mathbb{Q}^3$ with $(0,0) \in \mathbb{Q}^2$. Furthermore, $y_1(x_1, x_2) = f(x_1, x_2)$. Finally, $\|y(x_1, x_2)\| = \max(|x_1|, |x_2|, |x_1||x_2^4|) \leqslant (1 + |x_1|)(1 + |x_2|^4) \leqslant \mathtt{sp}(\|x_1, x_2\|)$. ♦

The following example focuses on the second issue mentioned at the beginning of the section, namely the domain of definition.

Example 2.1.25 (Inverse and logarithm functions)**:** We illustrate that the choice of the domain of definition makes important differences in the nature of the function.

- Let $\varepsilon > 0$ and define $f_\varepsilon : x \in]\varepsilon, \infty[\mapsto \frac{1}{x}$. It can be seen that $f_\varepsilon'(x) = -f_\varepsilon(x)^2$ and $f_\varepsilon(1) = 1$. Furthermore, $|f_\varepsilon(x)| \leqslant \frac{1}{\varepsilon}$ thus $f_\varepsilon \in \text{GVAL}(\alpha \mapsto \frac{1}{\varepsilon})$. So in particular, $f_\varepsilon \in \text{GPVAL}$ for any

$\varepsilon > 0$. Something interesting arises when $\varepsilon \to 0$: define $f_0(x) = x \in]0,\infty[\mapsto \frac{1}{x}$. Then f_0 is still generable and $|f_0(x)| \leqslant \frac{1}{|x|}$. Thus $f_0 \in \mathrm{GVAL}(\alpha \mapsto \frac{1}{\alpha})$ but $f_0 \notin \mathrm{GPVAL}$. Note that strictly speaking, $f_0 \in \mathrm{GVAL}(\mathtt{sp})$ where $\mathtt{sp}(\alpha) = \frac{1}{\alpha}$ and $\mathtt{sp}(0) = 0$ because the bound function needs to be defined over $\mathbb{R}_+$.

- A similar phenomenon occurs with the logarithm: define $g_\varepsilon : x \in]\varepsilon,\infty[\mapsto \ln(x)$. Then $g_\varepsilon'(x) = f_\varepsilon(x)$ and $g_\varepsilon(1) = 0$. Furthermore, $|g_\varepsilon(x)| \leqslant \max(|x|, \ln\varepsilon)$. Thus $g_\varepsilon \in \mathrm{GVAL}(\alpha \mapsto \max(\alpha, |\ln\varepsilon|, \frac{1}{\varepsilon}))$, and in particular $g_\varepsilon \in \mathrm{GPVAL}$ for any $\varepsilon > 0$. Similarly, $g_0 : x \in]0,\infty[\mapsto \ln(x)$ is generable but does not belong to GPVAL.

♦

2.2 Stability properties

In this section, the major results will the be stability of multidimensional generable functions under arithmetical operators, composition and ODE solving. Note that some of the results use properties on $\mathbb{K}$ which can be found in Section 2.7.I (Extended stability).

Lemma 2.2.1 (Arithmetic on generable functions)**:** *Let $d,e,n,m \in \mathbb{N}$, $\mathtt{sp},\overline{\mathtt{sp}} : \mathbb{R} \to \mathbb{R}_+$, $f :\subseteq \mathbb{R}^d \to \mathbb{R}^n \in \mathrm{GVAL}(\mathtt{sp})$ and $g :\subseteq \mathbb{R}^e \to \mathbb{R}^m \in \mathrm{GVAL}(\overline{\mathtt{sp}})$. Then:*

- *$f+g, f-g \in \mathrm{GVAL}(\mathtt{sp}+\overline{\mathtt{sp}})$ over $\operatorname{dom} f \cap \operatorname{dom} g$ if $d = e$ and $n = m$*
- *$fg \in \mathrm{GVAL}(\max(\mathtt{sp}, \overline{\mathtt{sp}}, \mathtt{sp}\,\overline{\mathtt{sp}}))$ if $d = e$ and $n = m$*
- *$f \circ g \in \mathrm{GVAL}(\max(\overline{\mathtt{sp}}, \mathtt{sp}\circ\overline{\mathtt{sp}}))$ if $m = d$ and $g(\operatorname{dom} g) \subseteq \operatorname{dom} f$*

♦

Proof. We focus on the case of the composition, the other cases are very similar.

Apply Definition 2.1.17 (Generable function) to f and g to respectively get $l,\bar{l} \in \mathbb{N}$, $p \in M_{l,d}(\mathbb{K})[\mathbb{R}^l]$, $\bar{p} \in M_{\bar{l},e}(\mathbb{K})[\mathbb{R}^{\bar{l}}]$, $x_0 \in \operatorname{dom} f \cap \mathbb{K}^d$, $\bar{x}_0 \in \operatorname{dom} g \cap \mathbb{K}^e$, $y_0 \in \mathbb{K}^l$, $\bar{y}_0 \in \mathbb{K}^{\bar{l}}$, $y : \operatorname{dom} f \to \mathbb{R}^l$ and $\bar{y} : \operatorname{dom} g \to \mathbb{R}^{\bar{l}}$. Define $h = y \circ g$, then $J_h = J_y(g)J_g = p(h)\bar{p}_{1..m}(\bar{y})$ and $h(\bar{x}_0) = y(\bar{y}_0) \in \mathbb{K}^l$ by Corollary 2.7.5 (Generable field stability). In other words $(\bar{y}, h)$ satisfy:

$$\begin{cases} \bar{y}(\bar{x}_0) = y_0 \in \mathbb{K}^{\bar{l}} \\ h(\bar{x}_0) = y(\bar{y}_0) \in \mathbb{K}^l \end{cases} \qquad \begin{cases} \bar{y}' = \bar{p}(\bar{y}) \\ h' = p(h)\bar{p}_{1..m}(\bar{y}) \end{cases}$$

This shows that $f\circ g = z_{1..m} \in \mathrm{GVAL}$. Furthermore, $\|(\bar{y}(x), h(x))\| \leqslant \max(\|\bar{y}(x)\|, \|y(g(x))\|) \leqslant \max(\overline{\mathtt{sp}}(\|x\|), \mathtt{sp}(\|g(x)\|)) \leqslant \max(\overline{\mathtt{sp}}(\|x\|), \mathtt{sp}(\overline{\mathtt{sp}}(\|x\|)))$. ■

Our main result is that the solution to an ODE whose right hand-side is generable, and possibly depends on an external and C^1 control, may be rewritten as a GPAC. A corollary of this result is that the solution to a generable ODE is generable.

Proposition 2.2.2 (Generable ODE rewriting)**:** *Let $d,n \in \mathbb{N}$, $I \subseteq \mathbb{R}^n$, $X \subseteq \mathbb{R}^d$, $\mathtt{sp} : \mathbb{R}_+ \to \mathbb{R}_+$ and $(f : I \times X \to \mathbb{R}^n) \in \mathrm{GVAL}_{\mathbb{K}}(\mathtt{sp})$. Define $\overline{\mathtt{sp}} = \max(\mathrm{id}, \mathtt{sp})$. Then there exists $m \in \mathbb{N}$, $(g : I \times X \to \mathbb{R}^m) \in \mathrm{GVAL}_{\mathbb{K}}(\overline{\mathtt{sp}})$ and $p \in \mathbb{K}^m[\mathbb{R}^m \times \mathbb{R}^d]$ such that for any interval J, $t_0 \in \mathbb{K} \cap J$, $y_0 \in \mathbb{K}^n \cap J$, $y \in C^1(J, I)$ and $x \in C^1(J, X)$, if y satisfies:*

$$\begin{cases} y(t_0) = y_0 \\ y'(t) = f(y(t), x(t)) \end{cases} \qquad \forall t \in J$$

then there exists $z \in C^1(J, \mathbb{R}^m)$ such that:

$$\begin{cases} z(t_0) = g(y_0, x(t_0)) \\ z'(t) = p(z(t), x'(t)) \end{cases} \qquad \begin{cases} y(t) = z_{1..d}(t) \\ \|z(t)\| \leqslant \overline{\operatorname{sp}}(\|y(t), x(t)\|) \end{cases} \qquad \forall t \in J$$

♦

Proof. Apply Definition 2.1.17 (Generable function) to f get $m \in \mathbb{N}$, $p \in M_{m,n+d}(\mathbb{K})[\mathbb{R}^m]$, $f_0 \in \operatorname{dom} f \cap \mathbb{K}^d$, $w_0 \in \mathbb{K}^m$ and $w : \operatorname{dom} f \to \mathbb{R}^m$ such that $w(f_0) = w_0$, $J_{w(v)} = p(w(v))$, $\|w(v)\| \leqslant \operatorname{sp}(\|v\|)$ and $w_{1..n}(v) = f(v)$ for all $v \in \operatorname{dom} f$. Define $u(t) = w(y(t), x(t))$, then:

$$\begin{aligned} u'(t) &= J_w(y(t), x(t))(y'(t), x'(t)) \\ &= p(w(y(t), x(t)))(f(y(t), x(t)), x'(t)) \\ &= p(u(t))(u_{1..n}(t), x'(t)) \\ &= q(u(t), x'(t)) \end{aligned}$$

where $q \in \mathbb{K}^m[\mathbb{R}^{m+d}]$ and $u(t_0) = w(y(t_0)) = w(y_0, x(t_0))$. Note that w itself is a generable function and more precisely $w \in \mathrm{GPVAL}_{\mathbb{K}}\, \operatorname{sp}$ by definition. Finally, note that $y'(t) = u_{1..d}(t)$ so that we get for all $t \in J$:

$$\begin{cases} y(t_0) = y_0 \\ y'(t) = u_{1..d}(t) \end{cases} \qquad \begin{cases} u(t_0) = w(y_0, x(t_0)) \\ u'(t) = q(u(t), x'(t)) \end{cases}$$

Define $z(t) = (y(t), u(t))$, then $z(t_0) = (y_0, w(y_0, x(t_0))) = g(y_0, x(t_0))$ where $y_0 \in \mathbb{K}^n$ and $w \in \mathrm{GVAL}_{\mathbb{K}}(\operatorname{sp})$ so $g \in \mathrm{GVAL}_{\mathbb{K}}(\overline{\operatorname{sp}})$. And clearly $z'(t) = r(z(t), x'(t))$ where $r \in \mathbb{K}^{n+m}[\mathbb{R}^{n+m}]$. Finally, $\|z(t)\| = \|y(t), w(y(t), x(t))\| \leqslant \max(\|y(t)\|, \operatorname{sp}(\|y(t), x(t)\|)) \leqslant \overline{\operatorname{sp}}(\|y(t), x(t)\|)$. ■

A simplified version of this lemma shows that generable functions are closed under ODE solving.

Corollary 2.2.3 (Generable functions are closed under ODE)**:** *Let $d \in \mathbb{N}$, $J \subseteq \mathbb{R}$ an interval, $\operatorname{sp}, \overline{\operatorname{sp}} : \mathbb{R}_+ \to \mathbb{R}_+$, $f :\subseteq \mathbb{R}^d \to \mathbb{R}^d$ in* GVAL(sp)*, $t_0 \in \mathbb{K} \cap J$ and $y_0 \in \mathbb{K}^d \cap \operatorname{dom} f$. Assume there exists $y : J \to \operatorname{dom} f$ satisfying for all $t \in J$:*

$$\begin{cases} y(t_0) = y_0 \\ y'(t) = f(y(t)) \end{cases} \qquad \|y(t)\| \leqslant \overline{\operatorname{sp}}(t)$$

Then $y \in \mathrm{GVAL}(\max(\overline{\operatorname{sp}}, \operatorname{sp} \circ \overline{\operatorname{sp}}))$ and is unique. ♦

Our last result is simple but very useful. Generable functions are continuous and continuously differentiable, so locally Lipschitz continuous. We can give a precise expression for the modulus of continuity in the case where the domain of definition is simple enough.

Proposition 2.2.4 (Modulus of continuity)**:** *Let $\operatorname{sp} : \mathbb{R}_+ \to \mathbb{R}_+$, $f \in \mathrm{GVAL}(\operatorname{sp})$. There exists $q \in \mathbb{K}[\mathbb{R}]$ such that for any $x_1, x_2 \in \operatorname{dom} f$, if $[x_1, x_2] \subseteq \operatorname{dom} f$ then $\|f(x_1) - f(x_2)\| \leqslant \|x_1 - x_2\|\, q(\operatorname{sp}(\max(\|x_1\|, \|x_2\|)))$. In particular, if $f \in \mathrm{GPVAL}_{\mathbb{K}}$ then there exists $q \in \mathbb{K}[\mathbb{R}]$ such that if $[x_1, x_2] \subseteq \operatorname{dom} f$ then $\|f(x_1) - f(x_2)\| \leqslant \|x_1 - x_2\|\, q(\max(\|x_1\|, \|x_2\|))$.* ♦

Proof. Apply Definition 2.1.17 (Generable function) to get d, e, n, p, x_0, y_0 and y. Let $k = \deg(p)$. Recall that for a matrix, the subordinate norm is given by $|||M||| = \max_i \sum_j |M_{ij}|$. Then:

$$\|f(x_1) - f(x_2)\| = \left\| \int_{x_1}^{x_2} J_{y_{1..e}}(x)dx \right\| = \left\| \int_0^1 J_{y_{1..e}}((1-\alpha)x_1 + \alpha x_2)(x_2 - x_1)d\alpha \right\|$$

$$
\begin{aligned}
&\leqslant \int_0^1 |||J_{y_{1..e}}((1-\alpha)x_1+\alpha x_2)||| \cdot \|x_2 - x_1\| \, d\alpha \\
&\leqslant \|x_2 - x_1\| \int_0^1 \max_{i\in[\![1,e]\!]} \sum_{j=1}^d |p_{ij}(y((1-\alpha)x_1+\alpha x_2))| d\alpha \\
&\leqslant \|x_2 - x_1\| \int_0^1 \max_{i\in[\![1,e]\!]} \sum_{j=1}^d \Sigma p \max(1, \|y((1-\alpha)x_1+\alpha x_2)\|)^k) d\alpha \\
&\leqslant \|x_2 - x_1\| \int_0^1 \max_{i\in[\![1,e]\!]} d\Sigma p \max(1, \mathtt{sp}(\|(1-\alpha)x_1+\alpha x_2\|))^k d\alpha \\
&\leqslant \|x_2 - x_1\| \int_0^1 d\Sigma p \max(1, \mathtt{sp}(\max(\|x_1\|, \|x_2\|)))^k d\alpha \\
&\leqslant \|x_2 - x_1\| \, d\Sigma p \max(1, \mathtt{sp}(\max(\|x_1\|, \|x_2\|)))^k
\end{aligned}
$$

■

2.3 Analyticity of generable functions

It is a well-known result that the solution to a PIVP $y' = p(y)$ (and more generally, an analytic differential equation $y' = f(y)$ where f is analytic) is real analytic on its domain of definition. In the previous section we defined a generalized notion of generable function satisfying $J_y = p(y)$ which analyticity is less immediate. In this section we go through the proof in detail, which of course subsumes the result for PIVP.

We recall a well-known characterization of analytic functions. It is indeed much easier to show that a function is infinitely differentiable and of controlled growth, rather than showing the convergence of the Taylor series.

Proposition 2.3.1 (Characterization of analytic functions)**:** *Let $f \in C^\infty(U)$ for some open subset U of $\mathbb{R}^m$. Then f is analytic on U if and only if, for each $u \in U$, there are an open ball V, with $u \in V \subseteq U$, and constants $C > 0$ and $R > 0$ such that the derivatives of f satisfy*

$$|\partial_\alpha f(x)| \leqslant C\frac{\alpha!}{R^{|\alpha|}} \qquad x \in V, \alpha \in \mathbb{N}^m$$

♦

Proof. See proposition 2.2.10 of [KP02]. ■

In order to use this result, we show that the derivatives of generable functions at a point x do not grow faster than the described bound. We use a generalization of Faà di Bruno formula for the derivatives of a composition.

Theorem 2.3.2 (Generalised Faà di Bruno's formula)**:** *Let $f : X \subseteq \mathbb{R}^d \to Y \subseteq \mathbb{R}^n$ and $g : Y \to \mathbb{R}$ where X, Y are open sets and f, g are sufficiently smooth functions*[2]*. Let $\alpha \in \mathbb{N}^d$ and $x \in X$, then*

$$\partial_\alpha(g \circ f)(x) = \alpha! \sum_{(s,\beta,\lambda)\in\mathcal{D}_\alpha} \partial_\lambda g(f(x)) \prod_{k=1}^s \frac{1}{\lambda_k!}\left(\frac{1}{\beta_k!}\partial_{\beta_k} f(x)\right)^{\lambda_k}$$

[2]more precisely, for the formula to hold for α, all the derivatives which appear in the right-hand side must exist and be continuous

where ∂_λ means $\partial_{\sum_{u=1}^{s}\lambda_u}$ and where $\mathcal{D}_\alpha$ is the list of decompositions *of α. A multi-index $\alpha \in \mathbb{N}^d$ is decomposed into $s \in \mathbb{N}$ parts $\beta_1, \dots, \beta_s \in \mathbb{N}^d$ with multiplicies $\lambda_1, \dots, \lambda_s \in \mathbb{N}^n$ respectively if $|\lambda_i| > 0$ for all i, all the β_i are distincts from each other and from 0, and $\alpha = |\lambda_1|\beta_1 + \dots + |\lambda_s|\beta_s$. Note that β and λ are multi-indices of multi-indices: $\beta \in \left(\mathbb{N}^d\right)^s$ and $\lambda \in \left(\mathbb{N}^d\right)^s$.* ♦

Proof. See [Ma09] or [EM03]. ■

Proposition 2.3.3 (Generable implies analytic)**:** *If $f \in$ GVAL then f is real-analytic on* dom f. ♦

Proof. Let $\mathtt{sp} : \mathbb{R} \to \mathbb{R}_+$, $p \in M_{n,d}[\mathbb{R}^n]$ and $y : \mathbb{R}^n \to \mathbb{R}^n$ from Definition 2.1.17 (Generable function). It is sufficient to prove that y is analytic on $D = \operatorname{dom} f$ to get the result. Let $i \in [\![1, n]\!]$, and $j \in [\![1, d]\!]$, since $J_y = p(y)$ then $\partial_j y_i(x) = p_{ij}(y(x))$ and p_{ij} is a polynomial vector so clearly C^∞. By Remark 2.1.19 (Regularity), y is also C^∞ so we can apply Theorem 2.3.2 (Generalised Faà di Bruno's formula) for any $x \in D$, $\alpha \in \mathbb{N}^d$ and get

$$\partial_\alpha(\partial_j y_i)(x) = \partial_\alpha(p_{ij} \circ y)(x) = \alpha! \sum_{(s,\beta,\lambda)\in\mathcal{D}_\alpha} \partial_\lambda p_{ij}(y(x)) \prod_{k=1}^{s} \frac{1}{\lambda_k!}\left(\frac{1}{\beta_k!}\partial_{\beta_k} y(x)\right)^{\lambda_k}$$

Define $B_\alpha(x) = \frac{1}{\alpha!}\|\partial_\alpha y(x)\|$, and denote by $\alpha + j$ the multi-index λ such that $\lambda_j = \alpha_j + 1$ and $\lambda_k = \alpha_k$ for $k \neq j$. Define $C(y(x)) = \max_{i,j,\lambda}(|\partial_\lambda p_{ij}(y(x))|)$ and note that it is well-defined because $\partial_\lambda p_{ij}$ is zero whenever $|\lambda| > \deg(p_{ij})$. Define $\mathcal{D}'_\alpha = \{(s, \beta, \lambda) \in \mathcal{D}_\alpha \mid |\lambda| \leqslant \deg(p)\}$. The equations becomes:

$$\begin{aligned}
|\partial_\alpha(\partial_j y_i)(x)| &\leqslant \alpha! \sum_{(s,\beta,\lambda)\in\mathcal{D}_\alpha} |\partial_\lambda p_{ij}(y(x))| \prod_{k=1}^{s} \frac{1}{\lambda_k!}\left|\frac{1}{\beta_k!}\partial_{\beta_k} y(x)\right|^{\lambda_k} \\
&\leqslant \alpha! C(y(x)) \sum_{(s,\beta,\lambda)\in\mathcal{D}'_\alpha} \prod_{k=1}^{s} \frac{1}{\lambda_k!} B_{\beta_k}(x)^{|\lambda_k|}
\end{aligned}$$

Note that the right-hand side of the expression doesn't depend on i. We are going to show by induction that $B_\alpha(x) \leqslant \left(\frac{C(y(x))}{R}\right)^{|\alpha|}$ for some choice of R. The initialization for $|\alpha| = 1$ is trivial because $\alpha! = 1$ and $B_\alpha(x) = \|\partial_\alpha y(x)\| \leqslant C(y(x))$ so we only need $R \leqslant 1$. The induction step is as follows:

$$\begin{aligned}
B_{\alpha+j}(x) &\leqslant C(y(x)) \sum_{(s,\beta,\lambda)\in\mathcal{D}'_\alpha} \prod_{k=1}^{s} \frac{1}{\lambda_k!} B_{\beta_k}(x)^{|\lambda_k|} \\
&\leqslant C(y(x)) \sum_{(s,\beta,\lambda)\in\mathcal{D}'_\alpha} \prod_{k=1}^{s} \frac{1}{\lambda_k!} \left(\frac{C(y(x))}{R}\right)^{|\beta_k||\lambda_k|} \\
&\leqslant C(y(x)) \sum_{(s,\beta,\lambda)\in\mathcal{D}'_\alpha} \frac{1}{\lambda!} \left(\frac{C(y(x))}{R}\right)^{\sum_{u=1}^{s}|\beta_k||\lambda_k|} \\
&\leqslant C(y(x)) \left(\frac{C(y(x))}{R}\right)^{|\alpha|} \sum_{(s,\beta,\lambda)\in\mathcal{D}'_\alpha} \frac{1}{\lambda!} \\
&\leqslant C(y(x)) \left(\frac{C(y(x))}{R}\right)^{|\alpha|} \#\mathcal{D}'_\alpha
\end{aligned}$$

Evaluating the exact cardinal of $\mathcal{D}'_\alpha$ is complicated but we only need a good enough bound to get on with it. First notice that for any $(s,\beta,\lambda)\in\mathcal{D}'_\alpha$, we have $|\lambda|\leqslant\deg(p)$ by definition, and since each $|\lambda_i|>0$, necessarily $s\leqslant\deg(p)$. This means that there is a finite number, denote it by A, of (s,λ) in $\mathcal{D}'_\alpha$. For a given λ, we must have $\alpha=\sum_{i=1}^{s}|\lambda_i|\beta_i$ which implies that $|\beta_{ij}|\leqslant|\alpha|$ and so there at most $(1+|\alpha|)^{ns}$ choices for β, and since $s\leqslant\deg(p)$, $\#\mathcal{D}'_\alpha\leqslant A(1+|\alpha|)^b$ where b and A are constants. Choose $R\leqslant 1$ such that $R^{|\alpha|}\geqslant A(1+|\alpha|)^b$ for all α to get the claimed bound on $B_\alpha(x)$.

To conclude with Proposition 2.3.1 (Characterization of analytic functions), consider $x\in D$. Let V be an open ball of D containing x. Let $M=\sup_{u\in V}C(y(x))$, it is finite because C is bounded by a polynomial, $\|y(x)\|\leqslant\mathtt{sp}(x)$ and V is an open ball (thus included in a compact set). Finally we get:

$$\|\partial_\alpha y(x)\|\leqslant\alpha!\left(\frac{M}{R}\right)^{|\alpha|}$$

■

2.4 Dependency in the parameters

One of the most useful properties of ODEs is the continuous dependency (of the solution) in the parameters. More formally, the operator $(t_0,y_0,p,t)\mapsto y(t)$ is continuous under reasonable hypothesis. This is the case for PIVPs and we can quantify this dependency with explicit bounds. We start with an elementary result about polynomials, in essence an effective version of the Lipschitz constant for polynomials on a compact set.

Lemma 2.4.1 (Effective Lipschitz bound for polynomials)**:** *Let $P\in\mathbb{R}[\mathbb{R}^d]$ and $k=\deg(P)$ its degree. For all $a,b\in\mathbb{R}^d$ such that $\|a\|,\|b\|\leqslant M$,*

$$|P(b)-P(a)|\leqslant kM^{k-1}\Sigma P\,\|b-a\|$$

where ΣP is the sum of the absolute value of the coefficients of P. ♦

Proof. We start with the case of a monomial: $P(X)=X^\alpha$ where $\alpha\in\mathbb{N}^d$. One checks by induction that:

$$b^\alpha-a^\alpha=\sum_{i=1}^{d}\left(\prod_{j<i}b_j^{\alpha_j}\right)(b_i^{\alpha_i}-a_i^{\alpha_i})\left(\prod_{j>i}a_j^{\alpha_j}\right)$$

Since it is well known that for any integer n:

$$b^n-a^n=(b-a)\sum_{i=0}^{n-1}a^ib^{n-1-i}$$

We conclude that:

$$\begin{aligned}|b^\alpha-a^\alpha|&\leqslant\sum_{i=1}^{d}\left(\prod_{j<i}|b_j|^{\alpha_j}\right)|b_i^{\alpha_i}-a_i^{\alpha_i}|\left(\prod_{j>i}|a_j|^{\alpha_j}\right)\\&\leqslant\sum_{i=1}^{d}M^{|\alpha|-\alpha_i}|b-a|\sum_{j=0}^{\alpha_i-1}M^{\alpha_i-1}\\&\leqslant\|b-a\|\sum_{i=1}^{d}M^{|\alpha|-1}\alpha_i\\&\leqslant|\alpha|\,\|a-b\|\,M^{|\alpha|-1}\end{aligned}$$

In the more general case of $P(X)=\sum_{|\alpha|\leqslant k}a_\alpha X^\alpha$ where k is the degree of P, we get:

$$\begin{aligned}|P(b)-P(a)| &\leqslant \sum_{|\alpha|\leqslant k}|a_\alpha||b^\alpha-a^\alpha|\\ &\leqslant \sum_{|\alpha|\leqslant k}|a_\alpha||\alpha|\,\|a-b\|\,M^{|\alpha|-1}\\ &\leqslant kM^{k-1}\Sigma P\,\|b-a\|\end{aligned}$$

■

We study the dependency of the solution with respect to the initial value and the polynomial. More precisely, we show that for small perturbations of the initial value and the derivative, the solution does not change much. We even generalize this result to PIVP with an external control which is itself subject to perturbations.

Theorem 2.4.2 (Parameter dependency of PIVP)**:** *Let $I=[a,b]$, $p\in\mathbb{R}^n[\mathbb{R}^{n+d}]$, $k=\deg(p)$, $e\in C^0(I,\mathbb{R}^d)$, $x,\delta\in C^0(I,\mathbb{R}^n)$ and $y_0,z_0\in\mathbb{R}^d$. Assume that $y,z:I\to\mathbb{R}^d$ satisfy:*

$$\begin{cases}y(a)=y_0\\ y'(t)=p(y(t),x(t))\end{cases}\qquad\begin{cases}z(a)=z_0\\ z'(t)=e(t)+p(z(t),x(t)+\delta(t))\end{cases}\qquad t\in I$$

Assume that there exists $\varepsilon>0$ such that for all $t\in I$,

$$\mu(t):=\left(\|z_0-y_0\|+\int_a^t\|e(u)\|+k\Sigma pM^{k-1}(u)\,\|\delta(u)\|\,du\right)\exp\left(k\Sigma p\int_a^t M^{k-1}(u)du\right)<\varepsilon$$

where $M(t)=\varepsilon+\|y(t)\|+\|x(t)\|+\|\delta(t)\|$. Then for all $t\in I$,

$$\|z(t)-y(t)\|\leqslant\mu(t)$$

♦

Proof. Let $\psi(t)=\|z(t)-y(t)\|$. For any $t\in I$, we have

$$\psi(t)\leqslant\psi(0)+\int_a^t\|p(z(u),x(u)+\delta(t))-p(y(u),x(u))\|\,du+\int_a^t\|e(u)\|\,du$$

Apply Lemma 2.4.1 (Effective Lipschitz bound for polynomials) to get, for $N(u)=\|y(u)\|+\psi(u)+\|x(u)\|+\|\delta(u)\|$, that:

$$\|p(z(u),x(u)+\delta(t))-p(y(u),x(u))\|\leqslant k\Sigma pN^{k-1}(u)(\psi(u)+\delta(u))$$

Putting everything together, we have:

$$\psi(t)\leqslant\psi(0)+\int_a^t\|e(u)\|+k\Sigma pN^{k-1}(u)\,\|\delta(u)\|\,du+k\Sigma p\int_a^t N^{k-1}(u)\psi(u)du$$

Apply the Generalized Gronwall's Inequality, using that the integral of non-negative values is non-decreasing, to get:

$$\psi(t)\leqslant\left(\|z_0-y_0\|+\int_a^t\|e(u)\|+k\Sigma pN^{k-1}(u)\,\|\delta(u)\|\,du\right)\exp\left(k\Sigma p\int_a^t N^{k-1}(u)du\right)$$

Define $t_1=\max\left\{t\in I\,|\,\forall u\in[a,t],\psi(u)\leqslant\varepsilon\right\}$ which is well-defined as the maximum of a closed and non-empty set (a belongs to it). Then for all $t\in[a,t_1]$, $N(t)\leqslant M(t)$ so $\psi(t)\leqslant\mu(t)$. We will show by contradiction that $t_1=b$, which proves the result. Assume by contradiction that $t_1<b$. Then by continuity of ψ and because $\psi(a)=\mu(a)<\varepsilon$, there exists $t_0\leqslant t_1$ such that $\psi(t_0)=\varepsilon$. But then $t_0\in[a,t_1]$ so $\psi(t_0)\leqslant\mu(t)<\varepsilon$ by hypothesis, which is impossible. ■

Lemma 2.4.3 (ODE time-scaling)**:** *Let $d \in \mathbb{N}$, $x_0 \in \mathbb{R}^d$, $p \in \mathbb{R}^d[\mathbb{R}^d]$, and $\phi \in C^0(\mathbb{R}_+, \mathbb{R}_+)$. Assume that $y, z : \mathbb{R}_+ \to \mathbb{R}^d$ satisfy for all $t \in \mathbb{R}_+$:*

$$\begin{cases} y(0) = x_0 \\ y'(t) = p(y(t)) \end{cases} \qquad \begin{cases} z(0) = x_0 \\ z'(t) = \phi(t)p(z(t)) \end{cases}$$

Then $z(t) = y\left(\int_0^t \phi(u)du\right)$ for all $t \in \mathbb{R}_+$. ♦

Lemma 2.4.4 (Perturbed time-scaling)**:** *Let $d \in \mathbb{N}$, $x_0 \in \mathbb{R}^d$, $p \in \mathbb{R}^d[\mathbb{R}^d]$, $e \in C^0(\mathbb{R}_+, \mathbb{R}^d)$ and $\phi \in C^0(\mathbb{R}_+, \mathbb{R}_+)$. Let $\psi(t) = \int_0^t \phi(u)du$. Assume that ψ is an increasing function and that $y, z : \mathbb{R}_+ \to \mathbb{R}^d$ satisfy for all $t \in \mathbb{R}_+$:*

$$\begin{cases} y(0) = x_0 \\ y'(t) = p(y(t)) + (\psi^{-1})'(t)e(\psi^{-1}(t)) \end{cases} \qquad \begin{cases} z(0) = x_0 \\ z'(t) = \phi(t)p(z(t)) + e(t) \end{cases}$$

Then $z(t) = y(\psi(t))$ for all $t \in \mathbb{R}_+$. In particular, $\int_0^{\psi(t)} \left\|(\psi^{-1})'(u)e(\psi^{-1}(u))\right\| du = \int_0^t \|e(u)\| du$ and $\sup_{u\in[0,\psi(t)]} \left\|(\psi^{-1})'(u)e(\psi^{-1}(u))\right\| = \sup_{u\in[0,t]} \frac{\|e(u)\|}{\phi(u)}$. ♦

Proof. Use that $\phi = \psi'$, $\psi' \cdot (\psi^{-1})' \circ \psi = 1$ and that $\psi' \geqslant 0$. ■

2.5 Taylor series of the solutions

It is well-known[3] that solutions of a PIVP are analytic so in particular the Taylor series at any point converges. This yields the natural question of the rate of convergence of the series, and the complexity of computing the truncated series.

In the case of a function satisfying a polynomial differential equation like (2.0.1), we can obtain a sharper bound than the one given by the classical Taylor-Lagrange theorem. These bounds are based on Cauchy majorants of series and we refer the reader to [WWS+06] for the details.

Theorem 2.5.1 (Taylor approximation for PIVP)**:** *If y satisfies (2.0.1), $k = \deg(p) \geqslant 2$, $\alpha = \max(1, \|y_0\|)$, $M = (k-1)\Sigma p\alpha^{k-1}$, $t_0 = 0$ $|t| < \frac{1}{M}$ then*

$$\|y(t) - T_0^n y(t)\| \leqslant \frac{\alpha|Mt|^n}{1 - |Mt|}$$

♦

Proof. The equations which we refer to in this proof are to be found in the original article [WWS+06], where $h = p$ and $a = y_0$ in (12). Then in (14), $\|c\| \leqslant \alpha$ and in (23), $m = k$ and $M \leqslant (k-1)\Sigma p \|c\|^{k-1}$. The result then follows from (44). ■

The following corollary gives a bound on the maximum variation of the solution of (2.0.1) on a small interval.

Corollary 2.5.2 (Maximum variation for PIVP)**:** *If y satisfies (2.0.1), $k = \deg(p) \geqslant 2$, $\alpha = \max(1, \|y_0\|)$, $M = (k-1)\Sigma p\alpha^{k-1}$, $t_0 = 0$ and $|t| < \frac{1}{M}$ then*

$$\|y(t) - y_0\| \leqslant \frac{\alpha|Mt|}{1 - |Mt|}$$

♦

[3]See Remark 2.1.3 (Regularity)

The next problem we face is to compute the truncated Taylor series of the solution over a small time interval. In this thesis, we will assume that we have access to a subroutine ComputeTaylor as follows.

Algorithm 2.5.3 Taylor Series algorithm for PIVP

```
Require: $p \in \mathbb{Q}^d[\mathbb{R}^d]$ the polynomial of the PIVP
Require: $y_0 \in \mathbb{Q}^d$ the initial condition
Require: $\omega \in \mathbb{N}$ the order of the approximation
Require: $\varepsilon \in \mathbb{Q}$ the precision requested
Require: $t \in \mathbb{Q}$ the time step
1: function ComputeTaylor($p, y_0, \omega, \varepsilon, t$)
2:     return x                ▷ such that $\left\|x - T_0^{\omega} y(t)\right\| \leqslant \varepsilon$ where $y(0) = y_0$ and $y' = p(y)$
3: end function
```

The complexity of computing this Taylor series has already been analyzed in the litterature. Let $\mathrm{TL}(d, p, y_0, \omega, \varepsilon, t)$ be the complexity of Algorithm 2.5.3 (Taylor series of the solutions). More precisely, we will refer to the *bit-complexity* as TL_{bit} and the *arithmetic complexity* as TL_{arith}.

In [BGP12] we described a very naive way of implementing Algorithm 2.5.3 (Taylor series of the solutions) by ways of formal differentiation, showing that the bit-complexity is bounded by

$$\mathrm{TL}_{bit} = O\left(\operatorname{poly}((\deg(p)\omega)^d, \log\max(1,t)\Sigma p\max(1,\|y_0\|), -\log\varepsilon)\right) \tag{2.5.4}$$

More explicit formulas can be found in [MM93]. Other much more advanced algorithms exist in the literature like [BCO+07] which shows that:

$$\mathrm{TL}_{arith} = \tilde{O}\left(\omega\deg(p)^d + (d\omega)^a\right) \tag{2.5.5}$$

(where is a is the matrix multiplication exponent) for the arithmetic complexity. Finally notice that these algorithms do not need to compute the actual coefficients of the Taylor series but only the evaluation of the truncated Taylor series with a certain precision.

2.6 Generable zoo

In this section, we introduce a number of generable functions which will be useful in the next chapters. This zoo will also illustrate the wide range of generable functions. The table below gives a list of the functions and their purpose.

We use the term "dead zone" to refer to interval(s) where the generable function does not compute the expected function (but still has controlled behavior). We use the term "high" to mean that the function is close to x (an input) within $e^{-\mu}$ where μ is another input. Conversely, the use the term "low" to mean that it is close to 0 within $e^{-\mu}$. And "X" means something in between. Finally "integral" means that function is of the form ϕx and the integral of ϕ (on some interval) is between 1 and a constant.

Generable Zoo

Name	Notation	Comment
Sign	$\mathrm{sg}(x,\mu,\lambda)$	Compute the sign of x with error $e^{-\mu}$ and dead zone in $[-\lambda^{-1},\lambda^{-1}]$. See 2.6.3
Floor	$\mathrm{ip}_1(x,\mu,\lambda)$	Compute $\mathrm{int}_1(x)$ with error $e^{-\mu}$ and dead zone in $[-\lambda^{-1},\lambda^{-1}]$. See 2.6.5

Generable Zoo

Name	Notation	Comment
Abs	$\mathrm{abs}_\delta(x)$	Compute $\|x\|$ with error δ. See 2.6.14
Max	$\mathrm{mx}_\delta(x)$	Compute $\max(x)$ with error δ. See 2.6.16
Norm	$\mathrm{norm}_{\infty,\delta}(x)$	Compute $\|\|x\|\|$ with error δ. See 2.6.18
Round	$\mathrm{ip}_\infty(x)$	Compute $\lfloor x \rceil$ with error $\frac{7}{8}\|x - \lfloor x \rceil\|$. See 2.6.9
	$\mathrm{rnd}(x,\mu,\lambda)$	Compute $\lfloor x \rceil$ with error $e^{-\mu}$ and dead zones in $[n - \frac{1}{2} + \lambda^{-1}, n + \frac{1}{2} - \lambda^{-1}]$ for all $n \in \mathbb{Z}$. See 2.6.12
Low-X-High	$\mathrm{lxh}_{[a,b]}(t,\mu,x)$	Compute 0 when $t \in]-\infty, a]$ and x when $t \in [b,\infty[$ with error $e^{-\mu}$ and a dead zone in $[a,b]$. See 2.6.22
High-X-Low	$\mathrm{hxl}_{[a,b]}(t,\mu,x)$	Compute x when $t \in]-\infty, a]$ and 0 when $t \in [b,\infty[$ with error $e^{-\mu}$ and a dead zone in $[a,b]$. See 2.6.22
Low-Integral-Low	$\mathrm{lil}_{[a,b]}(t,\mu,x)$	Compute 0 when $t \notin [a,b]$ with error $e^{-\mu}$ and $\mathrm{lil}_{[a,b]}(t,\mu,x) = \phi x$ with $\int_I \phi \in [1,K]$. See 2.6.24
Periodic L-I-L	$\mathrm{plil}_{[a,b],\tau}(t,\mu,x)$	Same as lil in a τ-periodic fashion. See 2.6.26
Select	$\mathrm{select}_{[a,b]}(t,\mu,x,y)$	Select x (resp. y) when $t \leqslant a$ (resp. $t \geqslant b$) with error $e^{-\mu}$ and a barycenter inbetween. See 2.6.28.

2.6.I Sign and rounding

We begin with a small result on the hyperbolic tangent function, which will be used to build several generable functions of interest.

Lemma 2.6.1 (Bounds on tanh): $1 - \mathrm{sgn}(t)\tanh(t) \leqslant e^{-|t|}$ *for all* $t \in \mathbb{R}$. ♦

Proof. The case of $t = 0$ is trivial. Assume that $t \geqslant 0$ and observe that $1-\tanh(t) = 1-\frac{1-e^{-2t}}{1+e^{-2t}} = \frac{2e^{-2t}}{1+e^{-2t}} = e^{-t}\frac{2e^{-t}}{1+e^{-2t}}$. Define $f(t) = \frac{2e^{-t}}{1+e^{-2t}}$ and check that $f'(t) = \frac{2e^{-t}(e^{-2t}-1)}{(1+e^{-2t})^2} \leqslant 0$ for $t \geqslant 0$. Thus f is a non-increasing function and $f(0) = 1$ which concludes.

If $t < 0$ then note that $1 - \mathrm{sgn}(t)\tanh(t) = 1 - \mathrm{sgn}(-t)\tanh(-t)$ so we can apply the result to $-t \geqslant 0$ to conclude. ■

The simplest generable function of interest uses the hyperbolic tangent to approximate the sign function. On top of the sign function, we can build a an approximate of the floor function. See Figure 2.6.6 (Graph of sg and ip_1) for a graphical representation.

Definition 2.6.2 (Sign function): For any $x, \mu, \lambda \in \mathbb{R}$ define

$$\mathrm{sg}(x,\mu,\lambda) = \tanh(x\mu\lambda)$$

♦

Lemma 2.6.3 (Sign): sg ∈ GPVAL *and for any* $x \in \mathbb{R}$ *and* $\lambda, \mu \geqslant 0$,

$$|\,\mathrm{sgn}(x) - \mathrm{sg}(x,\mu,\lambda)| \leqslant e^{-|x|\lambda\mu} \leqslant 1$$

In particular, sg *is non-decreasing in* x *and if* $|x| \geqslant \lambda^{-1}$ *then*

$$|\,\mathrm{sgn}(x) - \mathrm{sg}(x,\mu,\lambda)| \leqslant e^{-\mu}$$

♦

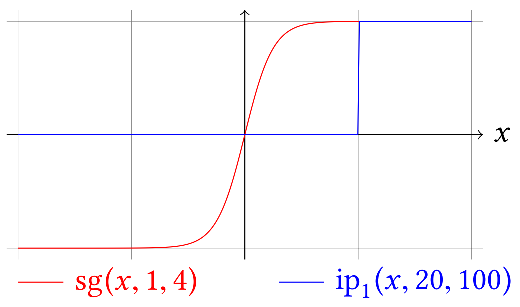


Figure 2.6.6: Graph of sg and ip_1.

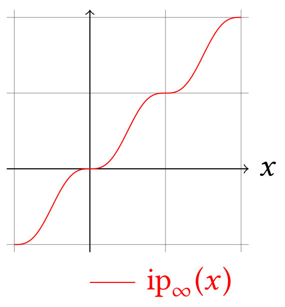


Figure 2.6.7: Graph of ip_∞

Proof. Note that $\mathrm{sg} = \tanh \circ f$ where $f(x,\mu,\lambda) = x\mu\lambda$. We saw in Example 2.1.6 (Some generable elementary functions) that $\tanh \in \mathrm{GVAL}(t \mapsto 1)$. By Lemma 2.2.1 (Arithmetic on generable functions), $f \in \mathrm{GVAL}(\mathrm{id})$. Thus $\mathrm{sg} \in \mathrm{GVAL}(\alpha \mapsto \max(1,\alpha))$.

Use Lemma 2.6.1 (Bounds on tanh) and the fact that tanh is an odd function to get the first bound. The second bound derives easily from the first. Finally, sg is a non-decreasing function because tanh is an increasing function. ■

Definition 2.6.4 (Floor function)**:** For any $x,\mu,\lambda \in \mathbb{R}$ define

$$\mathrm{ip}_1(x,\mu,\lambda) = \frac{1+\mathrm{sg}(x-1,\mu,\lambda)}{2}$$

♦

Lemma 2.6.5 (Floor)**:** $\mathrm{ip}_1 \in \mathrm{GPVAL}$ *and for any* $x \in \mathbb{R}$ *and* $\mu,\lambda \geqslant 0$,

$$|\operatorname{int}_1(x) - \mathrm{ip}_1(x,\mu,\lambda)| \leqslant \frac{e^{-|x-1|\lambda\mu}}{2} \leqslant \frac{1}{2}$$

In particular ip_1 *is non-decreasing in* x *and if* $|1-x| \geqslant \lambda^{-1}$ *then*

$$|\operatorname{int}_1(x) - \mathrm{ip}_1(x,\mu,\lambda)| < e^{-\mu}$$

♦

Another simple generable function uses the sine function to approximate the integer rounding function. Of course the approximation is pretty rough, but we can build an arbitrarily good one by iterating the function many times. See Figure 2.6.7 (Graph of ip_∞) for a graphical representation.

Definition 2.6.8 (Imprecise round function)**:** For any $x \in \mathbb{R}$ define

$$\mathrm{ip}_\infty(x) = x - \frac{1}{2\pi}\sin(2\pi x)$$

♦

Lemma 2.6.9 (Imprecise round)**:** *For any* $n \in \mathbb{Z}$ *and* $|\varepsilon| \leqslant \frac{1}{4}$

$$|\mathrm{ip}_\infty(n+\varepsilon) - n| \leqslant \frac{7}{8}\varepsilon \qquad \text{and} \qquad \left|\mathrm{ip}_\infty\left(n+\frac{1}{2}-\varepsilon\right) - n\right| \leqslant \frac{1}{2} - \frac{3}{2}\varepsilon$$

Furthermore $\mathrm{ip}_\infty \in \mathrm{GPVAL}$ *and is an increasing function.* ♦

Proof. The first statement is a direct consequence of the Taylor-Lagrange inequality:

$$|\operatorname{ip}_\infty(n+\varepsilon)-n| \leqslant \frac{1}{2!}\varepsilon^2 2\pi\sin(2\pi\varepsilon) \leqslant \frac{7}{8}\varepsilon$$

The second statement comes from the well-known inequality $\sin(2y) \geqslant y$ for $y \in [0, \frac{\pi}{4}]$:

$$\left|\operatorname{ip}_\infty\left(n+\frac{1}{2}-\varepsilon\right)-n\right| = \frac{1}{2}-\varepsilon-\frac{1}{2\pi}\sin(2\pi\varepsilon) \leqslant \frac{1}{2}-\varepsilon-\frac{\varepsilon}{2}$$

Finally, we saw that sin, id $\in$ GPVAL in Example 2.1.6 (Some generable elementary functions) and since $\pi \in \mathbb{K}$, conclude with Lemma 2.2.1 (Arithmetic on generable functions). ∎

Lemma 2.6.10 (Imprecise round)**:** *For any $\delta < \frac{1}{2}$ and $\eta > 0$, there exists $k_{\delta,\eta} \in \mathbb{N}$ and $A_{\delta,\eta} \in \mathbb{R}$ such that $f = \operatorname{ip}_\infty^{[k_{\delta,\eta}]}$ satisfies:*

$$\forall n \in \mathbb{Z}, |\varepsilon| \leqslant \delta, |f(n+\varepsilon)-n| \leqslant \eta$$

♦

We saw previously how to build a imprecise rounding function. It had the advantage of being a simple although it is quite imprecise. In the cases where this is not enough, the following function will provide a much better approximation at the cost of being significantly more complicated to define and use.

Definition 2.6.11 (Round function)**:** For any $x \in \mathbb{R}$, $\lambda \geqslant 2$ and $\mu \geqslant 0$, define

$$\operatorname{rnd}(x,\mu,\lambda) = x - \frac{1}{\pi}\arctan(\operatorname{cltan}(\pi x,\mu,\lambda))$$

$$\operatorname{cltan}(\theta,\mu,\lambda) = \frac{\sin(\theta)}{\sqrt{\operatorname{nz}(\cos^2\theta,\mu,4\lambda^2)}}\operatorname{sg}(\cos\theta,\mu,2\lambda)$$

$$\operatorname{nz}(x,\mu,\lambda) = x + \frac{2}{\lambda}\operatorname{ip}_1\left(1-x+\frac{3}{4\lambda},\mu+1,4\lambda\right)$$

♦

Lemma 2.6.12 (Round)**:** *For any $n \in \mathbb{Z}$, $\lambda \geqslant 2$, $\mu \geqslant 0$, $|\operatorname{rnd}(x,\mu,\lambda)-n| \leqslant \frac{1}{2}$ for all $x \in \left[n-\frac{1}{2}, n+\frac{1}{2}\right]$ and $|\operatorname{rnd}(x,\mu,\lambda)-n| \leqslant e^{-\mu}$ for all $x \in \left[n-\frac{1}{2}+\frac{1}{\lambda}, n+\frac{1}{2}-\frac{1}{\lambda}\right]$. Furthermore* rnd $\in$ GPVAL. ♦

Proof. Let's start with the intuition first: consider $f(x) = x - \frac{1}{\pi}\arctan(\tan(\pi x))$. It is an exact rounding function: if $x = n+\delta$ with $n \in \mathbb{N}$ and $\delta \in]\frac{-1}{2},\frac{1}{2}[$ then $\tan(\pi x) = \tan(\pi\delta)$ and since $\delta\pi \in]\frac{-\pi}{2},\frac{\pi}{2}[$, $f(x) = x-\delta = n$. The problem is that it is undefined on all points of the form $n+\frac{1}{2}$ because of the tangent function.

The idea is to replace $\tan(\pi x)$ by some "clamped" tangent cltan which will be like $\tan(\pi x)$ around integer points and stay bounded when close to $x = n+\frac{1}{2}$ instead of exploding. To do so, we use the fact that $\tan\theta = \frac{\sin\theta}{\cos\theta}$ but this formula is problematic because we cannot prevent the cosine from being zero, without loosing the sign of the expression (the cosine could never change sign). Thus the idea is to remove the sign from the cosine, and restore it, so that $\tan\theta = \operatorname{sgn}(\cos\theta)\frac{\sin\theta}{|\cos\theta|}$. And now we can replace $|\cos(\theta)|$ by $\sqrt{\operatorname{nz}(\cos^2\theta)}$, where $\operatorname{nz}(x)$ is mostly x except near 0 where is lower-bounded by some small constant (so it is never zero). The sign of cosine can be computed using our approximate sign function sg.

Formally, we begin with nz and show that:

- $\mathrm{nz} \in \mathrm{GPVAL}$
- nz is an increasing function of x
- For $x \geqslant \frac{1}{\lambda}$, $|\mathrm{nz}(x,\mu,\lambda) - x| \leqslant e^{-\mu}$
- For $x \geqslant 0$, $\mathrm{nz}(x,\mu,\lambda) \geqslant \frac{1}{2\lambda}$

The first point is a consequence of $\mathrm{ip}_1 \in \mathrm{GPVAL}$ from Lemma 2.6.5 (Floor). The second point comes from Lemma 2.6.5 (Floor): if $x \geqslant \frac{1}{\lambda}$, then $1 - x + \frac{3}{4\lambda} \leqslant 1 - \frac{1}{4\lambda}$, thus $|\mathrm{nz}(x,\mu,\lambda) - x| \leqslant \frac{2}{\lambda}e^{-\mu-1} \leqslant e^{-\mu}$ since $\lambda \geqslant 2$. To show the last point, first apply Lemma 2.6.5 (Floor): if $x \leqslant \frac{1}{2\lambda}$, then $1-x+\frac{3}{4\lambda} \geqslant 1+\frac{1}{4\lambda}$, thus $|\mathrm{nz}(x,\mu,\lambda)-x-\frac{2}{\lambda}| \leqslant \frac{2}{\lambda}e^{-\mu-1}$ Thus $\mathrm{nz}(x,\mu,\lambda) \geqslant \frac{2}{\lambda}(1-e^{-\mu-1})+x \geqslant \frac{1}{\lambda}$ since $1-e^{-\mu-1} \leqslant \frac{1}{2}$ and $x \geqslant 0$. And for $x \geqslant \frac{1}{2\lambda}$, by Lemma 2.6.5 (Floor) we get that $\mathrm{nz}(x,\mu,\lambda) \geqslant x \geqslant \frac{1}{2\lambda}$ which shows the last point.
Then we show that:

- $\mathrm{cltan} \in \mathrm{GPVAL}$, is π-periodic and is an odd function.
- For $\theta \in \left[-\frac{\pi}{2} + \frac{1}{\lambda}, \frac{\pi}{2} - \frac{1}{\lambda}\right]$, $|\mathrm{cltan}(\theta,\mu,\lambda) - \tan(\theta)| \leqslant e^{-\mu}$

The first point comes from the fact that $\sin, \cos, \mathrm{sg}, \mathrm{nz} \in \mathrm{GPVAL}$ and we further need the square-root and division to apply to lower bounded values. It is the case thanks to the results above about nz since $\sqrt{\mathrm{nz}(\cos^2\theta,\mu,2\lambda)} \geqslant \sqrt{\frac{1}{4\lambda}}$. This shows that $\mathrm{cltan}(\theta,\mu,\lambda) \leqslant 4\lambda$. The periodicity comes from the properties of sine and cosine, and the fact that sg is an odd function. It is an odd function for similar reasons. To show the second point, since it is periodic and odd, we can assume that $\theta \in \left[0, \frac{\pi}{2} - \frac{1}{\lambda}\right]$. For such a θ, we have that $\frac{\pi}{2} - \theta \geqslant \frac{1}{\lambda}$, thus $\cos(\theta) \geqslant \sin(\frac{\pi}{2} - \theta) \geqslant \frac{1}{2\lambda}$ (use that $\sin(u) \geqslant \frac{u}{2}$ for $0 \leqslant u \leqslant \frac{\pi}{2}$). By Lemma 2.6.3 (Sign) we get that $|\mathrm{sg}(\cos\theta,\mu,2\lambda) - 1| \leqslant e^{-\mu}$. Also $\cos^2\theta \geqslant \frac{1}{4\lambda^2}$ thus by the above results we get that $|\mathrm{nz}(\cos^2\theta,\mu,4\lambda^2) - \cos^2\theta| \leqslant e^{-\mu}$ and thus[4] $|\sqrt{\mathrm{nz}(\cos^2\theta,\mu,4\lambda^2)} - |\cos\theta|| \leqslant 4\lambda e^{-\mu}$.

Let $n \in \mathbb{N}$ and $x = n + \delta \in [n - \frac{1}{2}, n + \frac{1}{2}]$. Since cltan is π-periodic, $\mathrm{rnd}(x,\mu,\lambda) = n+\delta-\frac{1}{\pi}\arctan(\mathrm{cltan}(\pi\delta,\mu,\lambda))$. Furthermore $\pi\delta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ so $\cos(\pi\delta) \geqslant 0$ and $\mathrm{sgn}(\sin(\pi\delta)) = \mathrm{sgn}(\delta)$. Consequently, $\mathrm{sg}(\cos(\pi\delta),\mu,2\lambda) \in [0,1]$ by definition of sg and $\sqrt{\mathrm{nz}(\cos^2(\pi\delta),\mu,4\lambda^2)} > \sqrt{\cos^2(\pi\delta)}$ because $\mathrm{ip}_1 > 0$. We get that $|\mathrm{cltan}(\pi\delta,\mu,\lambda)| \leqslant \frac{|\sin(\pi\delta)|}{\cos(\pi\delta)}$ and $\mathrm{sgn}(\mathrm{cltan}(\pi\delta,\mu,\lambda)) = \mathrm{sgn}(\delta)$. Finally, $\frac{1}{\pi}\arctan(\mathrm{cltan}(\pi\delta,\mu,\lambda)) = \alpha$ with $|\alpha| \leqslant |\frac{1}{\pi}\arctan(\tan(\pi\delta))| \leqslant |\delta|$ and $\mathrm{sgn}(\alpha) = \mathrm{sgn}(\delta)$ which shows that $|\mathrm{rnd}(x,\mu,\lambda) - n| \leqslant \delta \leqslant \frac{1}{2}$.

Finally we can show the result about rnd: since cltan and tan are in GPVAL, then $\mathrm{rnd} \in \mathrm{GPVAL}$. Now consider $x \in \left[n - \frac{1}{2} + \frac{1}{\lambda}, n + \frac{1}{2} - \frac{1}{\lambda}\right]$, and let $\theta = \pi x - \pi n$. Then $\theta \in \left[-\frac{\pi}{2} + \frac{\pi}{\lambda}, \frac{\pi}{2} - \frac{\pi}{\lambda}\right] \subseteq \left[-\frac{\pi}{2} + \frac{1}{\lambda}, \frac{\pi}{2} - \frac{1}{\lambda}\right]$, and since cltan is periodic, then $\mathrm{rnd}(x,\mu,\lambda) = n + \frac{\theta}{\pi} - \frac{1}{\pi}\arctan(\mathrm{cltan}(\theta,\mu,\lambda))$. Finally, using the results about cltan yields: $|\mathrm{rnd}(x,\mu,\lambda) - n| = \frac{1}{\pi}|\theta - \arctan(\mathrm{cltan}(\theta,\mu,\lambda)| = \frac{1}{\pi}|\arctan(\tan(\theta)) - \arctan(\mathrm{cltan}(\theta,\mu,\lambda)| \leqslant \frac{1}{\pi}|\tan(\theta) - \mathrm{cltan}(\theta,\mu,\lambda)| \leqslant \frac{e^{-\mu}}{\pi} \leqslant e^{-\mu}$ since arctan is a 1-Lipschitz function. ■

2.6.II Absolute value, maximum and norm

A very common operation in our constructions is to compute the absolute value of a number. Of course this operation is not generable because it is not even differentiable. However, a good enough approximation can be built. In particular, this approximation has several keys features: it is positive and it is an over-approximation. On top of this, we can build an approximation of the max function and the infinite norm function.

[4] use that $|\sqrt{a} - \sqrt{b}| \leqslant \frac{|a-b|}{2\sqrt{a}}$ and for $a = \mathrm{nz}(\cos^2\theta,\mu,2\lambda)$, $\sqrt{a} \geqslant \sqrt{\frac{1}{\lambda}}$ as shown before

Definition 2.6.13 (Absolute value function)**:** For any $x \in \mathbb{R}$ and $\delta \in]0,1]$ define:

$$\operatorname{abs}_\delta(x) = \delta + \tanh(x\delta^{-1})x$$

♦

Lemma 2.6.14 (Absolute value)**:** *For any* $x \in \mathbb{R}$ *and* $\delta \in]0,1]$ *we have:*

$$|x| \leqslant \operatorname{abs}_\delta(x) \leqslant |x| + \delta$$

Furthermore $\operatorname{abs}_\delta \in \mathrm{GPVAL}$ *and is an even function.* ♦

Proof. Since tanh is an odd function, we immediately get that abs is even and abs $\geqslant 0$, and we can do the reasoning for $x \geqslant 0$ only. By Lemma 2.6.1 (Bounds on tanh) we get that $\delta + x \geqslant \operatorname{abs}_\delta(x) \geqslant x + \delta - xe^{-x\delta^{-1}}$. A simple study of the $x \mapsto xe^{-\delta^{-1}x}$ function shows that it is maximum for $x = \delta$ which gives δe^{-1} and concludes the proof. Finally $\operatorname{abs}_\delta \in \mathrm{GPVAL}$ from Lemma 2.2.1 (Arithmetic on generable functions) and Example 2.1.6 (Some generable elementary functions). ■

Definition 2.6.15 (Max function)**:** For any $x, y \in \mathbb{R}$ and $\delta \in]0,1]$ define:

$$\operatorname{mx}_\delta(x,y) = \frac{y + x + \operatorname{abs}_{2\delta}(y-x)}{2}$$

For any $x \in \mathbb{R}^n$ and $\delta \in]0,1]$ define:

$$\operatorname{mx}_\delta(x) = \operatorname{mx}_{\delta/n}(x_1, \operatorname{mx}_{\delta/n}(\ldots, \operatorname{mx}_{\delta/n}(x_{n-1}, x_n)\ldots))$$

♦

Lemma 2.6.16 (Max function)**:** *For any* $x, y \in \mathbb{R}$ *and* $\delta \in]0,1]$ *we have:*

$$\max(x,y) \leqslant \operatorname{mx}_\delta(x,y) \leqslant \max(x,y) + \delta$$

For any $x \in \mathbb{R}^n$ *and* $\delta \in]0,1]$ *we have:*

$$\max(x_1, \ldots, x_n) \leqslant \operatorname{mx}_\delta(x) \leqslant \max(x_1, \ldots, x_n) + \delta$$

Furthermore $\operatorname{mx}_\delta \in \mathrm{GPVAL}$. ♦

Proof. By Lemma 2.6.14 (Absolute value), $|y-x| \leqslant \operatorname{abs}_\delta(y-x) \leqslant |y-x| + \delta$ and the result follows because $\max(x,y) = \frac{y+x+|y-x|}{2}$. Finally $\operatorname{mx}_\delta \in \mathrm{GPVAL}$ from Lemma 2.2.1 (Arithmetic on generable functions). The generalisation to more variables is immediate. ■

Definition 2.6.17 (Norm function)**:** For any $x \in \mathbb{R}^n$ and $\delta \in]0,1]$ define:

$$\operatorname{norm}_{\infty,\delta}(x) = \operatorname{mx}_{\delta/2}(\operatorname{abs}_{\delta/2}(x_1), \ldots, \operatorname{abs}_{\delta/2}(x_n))$$

♦

Lemma 2.6.18 (Norm function)**:** *For any* $x \in \mathbb{R}^n$ *and* $\delta \in]0,1]$ *we have:*

$$\|x\| \leqslant \operatorname{norm}_{\infty,\delta}(x) \leqslant \|x\| + \delta$$

Furthermore $\operatorname{norm}_{\infty,\delta} \in \mathrm{GPVAL}$. ♦

Proof. Apply Lemma 2.6.14 (Absolute value) and Lemma 2.6.16 (Max function). ■

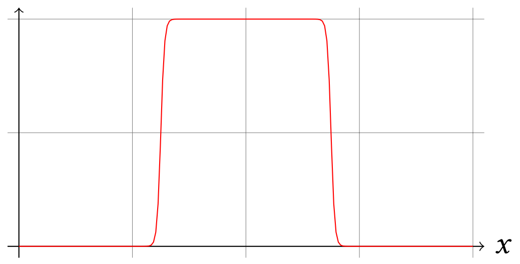


Figure 2.6.19: Graph of $\text{lil}_{[1,3]}$

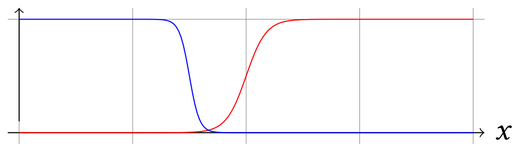


Figure 2.6.20: Graph of $\text{lxh}_{[1,3]}$ and $\text{hxl}_{[1,2]}$

2.6.III Switching functions

Later on, when using generable functions as a model of computation, we will need something which acts like a select function, which can pick between two values depending on how a third value compares to a threshold. Problem is that this operation is not continuous, and thus not generable. As a good first step, we build so-called "low-X-high" and "high-X-low" functions which act as a switch between 0 (low) and a value (high). Around the threshold will be an small uncertainty zone (X) where the exact value cannot be predicted. See Figure 2.6.20 (Graph of $\text{lxh}_{[1,3]}$ and $\text{hxl}_{[1,2]}$) for a graphical representation.

Definition 2.6.21 ("low-X-high" and "high-X-low")**:** Let $I = [a, b]$ with $b > a$, $t \in \mathbb{R}$, $\mu \in \mathbb{R}$, $x \in \mathbb{R}$, $\nu = \mu + \ln(1 + x^2)$, $\delta = \frac{b-a}{2}$ and define:

$$\text{lxh}_I(t, \mu, x) = \text{ip}_1\left(t - \frac{a+b}{2} + 1, \nu, \frac{1}{\delta}\right)x \qquad \text{hxl}_I(t, \mu, x) = \text{ip}_1\left(\frac{a+b}{2} - t + 1, \nu, \frac{1}{\delta}\right)x$$

♦

Lemma 2.6.22 ("low-X-high" and "high-X-low")**:** *Let $I = [a, b]$, $\mu \in \mathbb{R}_+$, then $\forall t, x \in \mathbb{R}$:*

- *$\exists \phi_1, \phi_2$ such that $\text{lxh}_I(t, \mu, x) = \phi_1(t, \mu, x)x$ and $\text{hxl}_I(t, \mu, x) = \phi_2(t, \mu, x)x$*
- *if $t \leqslant a$, $|\text{lxh}_I(t, \mu, x)| \leqslant e^{-\mu}$ and $|x - \text{hxl}_I(t, \mu, x)| \leqslant e^{-\mu}$*
- *if $t \geqslant b$, $|x - \text{lxh}_I(t, \mu, x)| \leqslant e^{-\mu}$ and $|\text{hxl}_I(t, \mu, x)| \leqslant e^{-\mu}$*
- *in all cases, $|\text{lxh}_I(t, \mu, x)| \leqslant |x|$ and $|\text{hxl}_I(t, \mu, x)| \leqslant |x|$*

Furthermore, $\text{lxh}_I, \text{hxl}_I \in \text{GPVAL}$. ♦

Proof. By symetry, we only prove it for lxh. This is a direct consequence of Lemma 2.6.5 (Floor) and the fact that $|x| \leqslant e^{\ln(1+x^2)}$. Indeed if $t \leqslant a$ then $t - \frac{a+b}{2} + 1 \leqslant 1 - \delta$ thus $|\text{lxh}_I(t, \nu, x)| \leqslant |x|e^{-\nu} \leqslant e^{-\mu}$. Similarly if $t \geqslant b$ then $t - \frac{a+b}{2} + 1 \geqslant 1 + \delta$ and we get a similar result. Apply Lemma 2.2.1 (Arithmetic on generable functions) multiple times to see that they are belong to GPVAL. ■

Although the "low-X-high" and "high-X-low" can very useful, in some occasion we are not really interested in the exact value of the function in the "high" part but rather the integral of non-"low". In this case, the uncertainty zone of the previous functions in unacceptable because we cannot control its effect. The "low-integral-low" function works around this problem by ensuring that the integral of the non-"low" part is not too far from the expected value, in a certain sense. See Figure 2.6.19 (Graph of $\text{lil}_{[1,3]}$) for a graphical representation. We also build a periodic variant of this function.

Definition 2.6.23 ("low-integral-low"): Let $I = [a, b]$, $t \in \mathbb{R}$, $\mu \in \mathbb{R}$, $x \in \mathbb{R}$ and define:

$$\operatorname{lil}_I(t, \mu, x) = \operatorname{lxh}_{[a,a+\delta]}(t, \nu, 1)\operatorname{hxl}_{[b-\delta,b]}(t, \nu, 1)Kx$$

where

$$\delta = \frac{b-a}{4} \qquad \nu = \mu + 2 + \ln K(1 + x^2) \qquad K = \frac{1}{\delta}$$

♦

Lemma 2.6.24 ("low-integral-low"): *Let $I = [a, b]$ with $b > a$, $\mu \in \mathbb{R}_+$ and $x \in \mathbb{R}$. Then there exists a constant K such that*

- $|\operatorname{lil}_I(t, \mu, x)| < e^{-\mu}$ *for all $t \notin I$*
- *$\exists\phi$ such that $\operatorname{lil}_I(t, \mu, x) = \phi(t, \mu, x)x$ for all $t \in I$, and for any $\alpha : I \to \mathbb{R}_+, \beta : I \to \mathbb{R}$ we have:*

$$1 \leqslant \int_a^b \phi(t, \alpha(t), \beta(t))dt \leqslant K$$

Furthermore, $\operatorname{lil}_I \in \mathrm{GPVAL}$. ♦

Proof. Apply Lemma 2.2.1 (Arithmetic on generable functions) multiple times to get that $\operatorname{lil}_I \in \mathrm{GPVAL}$. Note that it works because $K(1 + x^2) \geqslant K > 0$ so $\ln\restriction_{]K,\infty[} \in \mathrm{GPVAL}$, see Example 2.1.25 (Inverse and logarithm functions) for more details.

- If $t < a$ or $t > b$ apply Lemma 2.6.22 ("low-X-high" and "high-X-low") twice to get $|\operatorname{lil}_I(t, \mu, x)| \leqslant e^{-\nu}K|x| \leqslant e^{-\mu}$ by the choice of ν since $|x| \leqslant e^{\ln(1+x^2)}$.
- Clearly $\operatorname{lil}_I(t, \mu, x) = \phi(t, \mu, x)x$ with $\phi(t, \mu, x) = \operatorname{lxh}_{[a,a+\delta]}(t, \nu, 1)\operatorname{hxl}_{[b-\delta,b]}(t, \nu, 1)K$. Let $\alpha : I \to \mathbb{R}_+, \beta : I \to \mathbb{R}$. Since $\phi > 0$, $\int_a^b \phi(t, \alpha(t), \beta(t))dt \geqslant \int_{a+2\delta}^{b-2\delta} \phi(t, \alpha(t), \beta(t))dt$. Apply Lemma 2.6.22 ("low-X-high" and "high-X-low") to get that $\int_{a+\delta}^{b-\delta} \phi(t, \alpha(t), \beta(t))dt > 2\delta(1 - e^{-\nu})^2K \geqslant 2(1 - e^{-2})^2 \geqslant 1$ since $\nu \geqslant 2$. Conversely, applying Lemma 2.6.22 ("low-X-high" and "high-X-low") gives that $\phi(t, \mu, x) \leqslant K$ so $\int_a^b \phi(t, \alpha(t), \beta(t))dt \leqslant (b - a)K \leqslant 4$.

■

Definition 2.6.25 ("periodic low-integral-low"): Let $t \in \mathbb{R}$, $\tau \in \mathbb{R}_+$, $\mu, x \in \mathbb{R}$, $I = [a, b] \subseteq [0, \tau]$ with $0 < b - a < \tau$ and define:

$$\operatorname{plil}_{I,\tau}(t, \mu, x) = \operatorname{lxh}_J(f(t), \nu, K)x$$

where

$$\delta = b - a \qquad \omega = \frac{2\pi}{\tau} \qquad K = \frac{1}{4} + \frac{2}{\delta} \qquad t_1 = \frac{a+b}{2} - \frac{\tau}{4}$$

$$\nu = \mu + 2 + \ln(1 + x^2) \qquad f(t) = \sin(\omega(t - t_1)) \quad J = \left[f(a), f\left(a + \frac{\delta}{4}\right)\right]$$

♦

Lemma 2.6.26 ("periodic low-integral-low"): *Let $\mu, \tau \in \mathbb{R}_+$, $I = [a, b] \subsetneq [0, \tau]$ and $x \in \mathbb{R}$. Then there exists a constant K and ϕ such that* $\operatorname{plil}_{I,\tau}(t, \mu, x) = \phi(t, \mu, x)x$ *and:*

- $\operatorname{plil}_{I,\tau}(\cdot, \mu, x)$ *is τ-periodic*

- $\forall t \notin I, |\operatorname{plil}_{I,\tau}(t,\mu,x)| < e^{-\mu}$
- *for any* $\alpha : I \to \mathbb{R}_+, \beta : I \to \mathbb{R}$:

$$1 \leqslant \int_a^b \phi(t,\alpha(t),\beta(t))dt \leqslant K$$

Furthermore, $\operatorname{plil}_{I,\tau} \in \text{GPVAL}$. ♦

Proof. The τ-periodicity is trivial. Using trigonometric identities, observe that

$$f(t) - f(a) = -2\sin\left(\omega\frac{t-b}{2}\right)\sin\left(\omega\frac{t-a}{2}\right)$$

Now it is easy to see that if $t \in [0,a]$ then $\omega\frac{t-b}{2}, \omega\frac{t-a}{2} \in [-\pi,0]$ thus $f(t) \leqslant f(a)$. By the choice of J and Lemma 2.6.22 ("low-X-high" and "high-X-low"), we get that $\operatorname{lxh}_J(f(t),\mu+2,K) \leqslant e^{-\nu}$. Similarly if $t \in [b,\tau]$ then $\omega\frac{t-b}{2}, \omega\frac{t-a}{2} \in [0,\pi]$ and we get the same result. We conclude the first part of the result using that $|xe^{-\nu}| \leqslant e^{-\mu}$.

Let $\alpha : I \to \mathbb{R}_+, \beta : I \to \mathbb{R}$. Let $a' = a + \frac{\delta}{4}$ and $b' = b - \frac{\delta}{4}$. Since $\operatorname{lxh} > 0$, we have $\int_a^b \operatorname{plil}_{I,\tau}(t,\alpha(t),\beta(t))dt \geqslant \int_{a'}^{b'} \operatorname{plil}_{I,\tau}(t,\alpha(t),\beta(t))dt$. Again observe that

$$f(t) - f(a') = -2\sin\left(\omega\frac{t-b'}{2}\right)\sin\left(\omega\frac{t-a'}{2}\right)$$

Consequently, if $t \in [a',b']$ then $f(t) \geqslant f(a')$. By the choice of J and Lemma 2.6.22 ("low-X-high" and "high-X-low"), we get that $\operatorname{lxh}_J(f(t),\nu,K) \geqslant K - e^{-\nu} \geqslant K - \frac{1}{4}$ since $\nu \geqslant 2$. Finally $\int_a^b \operatorname{plil}_{I,\tau}(t,\alpha(t),\beta(t))dt \geqslant (b'-a')(K-\frac{1}{4}) \geqslant 1$ and $\int_a^b \operatorname{plil}_{I,\tau}(t,\alpha(t),\beta(t))dt \leqslant (b-a)K$ by Lemma 2.6.22 ("low-X-high" and "high-X-low").

Apply Lemma 2.2.1 (Arithmetic on generable functions) multiple times to get that $\operatorname{plil}_{I,\tau} \in$ GPVAL. ■

Definition 2.6.27 (Select): Let $I = [a,b]$, $t,x,y,\mu \in \mathbb{R}$, $\mu \in \mathbb{R}_+$ and define:

$$\operatorname{select}_I(t,\mu,x,y) = x + \operatorname{lxh}_I(t,\mu,y-x)$$

♦

Lemma 2.6.28 (Select): *Let* $I = [a,b]$, $t,x,y \in \mathbb{R}$ *and* $\mu\mathbb{R}_+$*:*

- *if* $t \leqslant a$ *then* $|\operatorname{select}_I(t,\mu,x,y) - x| \leqslant e^{-\mu}$
- *if* $t \geqslant b$ *then* $|\operatorname{select}_I(t,\mu,x,y) - y| \leqslant e^{-\mu}$
- *if* $t \in [a,b]$ *then* $\operatorname{select}_I(t,\mu,x,y) = x + \alpha(y-x)$ *where* $\alpha \in [0,1]$

Furthermore $\operatorname{select}_I \in$ GPVAL. ♦

Proof. Apply Lemma 2.6.22 ("low-X-high" and "high-X-low") to get that $\operatorname{select}_I \in$ GPVAL. If $t \leqslant a$ then $|\operatorname{lxh}_I(t,\mu,y-x)| \leqslant e^{-\mu}$ so $|\operatorname{select}_I(t,\mu,x,y) - x| \leqslant e^{-\mu}$. If $t \geqslant b$ then $|\operatorname{lxh}_I(t,\mu,y-x) - (y-x)| \leqslant e^{-\mu}$ so $|\operatorname{select}_I(t,\mu,x,y) - y| \leqslant e^{-\mu}$. If $t \in [a,b]$ then look at Definition 2.6.21 ("low-X-high" and "high-X-low") to get that $\operatorname{select}_I(t,\mu,x,y) = x+\alpha(y-x)$ where $\alpha = \operatorname{ip}_1(\ldots)$. Look at Definition 2.6.4 (Floor function) to conclude that $\alpha \in [0,1]$. ■

2.7 Generable fields

In Section 2.1 (Generable functions), we introduced the notion of *generable field*, which are fields with an additional stability property. We used this notion to ensure that the class of functions we built is closed under composition. It is well-known that if we allow any choice of constants in our computation, we will gain extra computational power because of uncomputable real numbers. For this reason, it is wise to make sure that we can exhibit at least one generable field consisting of computable real numbers only, and possibly only polynomial time computable numbers.

Intuitively, we are looking for a (the) smallest generable field, call it $\mathbb{R}_G$, in order to minimize the computation power of the real numbers it contains. The rest of this section is dedicated to the study of this field. We first recall Definition 2.1.8 (Generable field).

Definition 2.7.1 (Generable field)**:** A field $\mathbb{K}$ is *generable* if and only if $\mathbb{Q} \subseteq \mathbb{K}$ and for any $\alpha \in \mathbb{K}$, and $(f : \mathbb{R} \to \mathbb{R}) \in \mathrm{GVAL}_{\mathbb{K}}$, $f(\alpha) \in \mathbb{K}$. ♦

2.7.I Extended stability

By definition of a generable field, $\mathbb{K}$ is preserved by unidimensional generable functions. An interesting question is whether $\mathbb{K}$ is also preserved by multidimensional functions. This is not immediate because because of several key differences in the definition of multidimensional generable functions. We first recall a folklore topology lemma.

Lemma 2.7.2 (Offset of a compact set)**:** *Let $X \subseteq U \subseteq \mathbb{R}^n$ where U is open and X is compact. Then there exists $\varepsilon > 0$ such that $X_\varepsilon \subseteq U$ where the ε-offset of X is defined by $X_\varepsilon = \bigcup_{x \in X} B_\varepsilon(x)$.* ♦

Proof. This is a very classical result: let $F = \mathbb{R}^n \setminus U$, then F is closed so the distance function[5] d_F to F is continuous. Since X is compact, $d_F(X)$ is a compact subset of $\mathbb{R}_+$, and $d_F(X)$ is nowhere 0 because $X \subseteq U \subseteq F$ where U is open. Consequently $d_F(X)$ admits a positive minimum ε. Let $x \in X_\varepsilon$, then $\exists y \in X$ such that $\|x - y\| < \varepsilon$, and by the triangle inequality, $\varepsilon \leqslant d_F(y) \leqslant \|x - y\| + d_F(x)$ so $d_F(x) > 0$ which means $x \notin F$, in other words $x \in U$. ■

Lemma 2.7.3 (Polygonal path connectedness)**:** *An open, connected subset U of $\mathbb{R}^n$ is always* polygonal-path-connected*: for any $a, b \in U$, there exists a polygonal path[6] from a to b in U. Furthermore, we can take all intermediate vertices in $\mathbb{Q}^n$.* ♦

Proof. Let $a, b \in U$, since U is a connected and open subset of $\mathbb{R}^n$, it is path-connected [7] Let $\gamma : [0, 1] \to U$ be a path from a to b. Let $X = \gamma([0, 1])$, it is compact and connected because γ is continuous. By Lemma 2.7.2 (Offset of a compact set), there is $\varepsilon > 0$ such that $X_\varepsilon \subseteq U$. For any $x \in X$, define $U_x = B_\varepsilon(x) \subseteq X_\varepsilon$. Then $(U_x)_{x \in X}$ is an open cover of the compact set X, so it admits a finite subcover $(U_{x_i})_{i \in [\![1,k]\!]}$ where $x_i \in X$. Without loss of generality with can assume that:

- $a \in U_{x_1}$ since the U_{x_i} must cover $a \in X$.
- $U_{x_i} \cap U_{x_{i+1}} \neq \varnothing$ for every $i \in [\![1, k-1]\!]$ since X is connected.
- $b \in U_{x_k}$ for the same reason as a.

[5]We always use the infinite norm $\|\cdot\|$ in this chapter but it works for any distance

[6]A polygonal path is a connected sequence of line segments

[7]This is a textbook property.

For any $i \in [\![1, k-1]\!]$, pick $y_i \in U_{x_i} \cap U_{x_{i+1}} \cap \mathbb{Q}$ which is not empty because $\mathbb{Q}$ is dense in $\mathbb{R}$. Consider the polygonal path ϕ joining $a, x_1, y_1, x_2, \ldots, y_{k-1}, x_k, b$. Then the image of ϕ is included in $X_\varepsilon \subseteq U$ because:

- $a \in U_{x_1}$ so the line segment $[a, x_1] \subseteq U_{x_1} \subseteq X_\varepsilon$.
- $y_i \in U_{x_i} \cap U_{x_{i+1}}$ so $[x_i, y_i] \subseteq U_{x_i} \subseteq X_\varepsilon$ and $[y_i, x_{i+1}] \subseteq U_{x_{i+1}} \subseteq X_\varepsilon$.
- $b \in U_{x_k}$ so $[x_k, b] \subseteq U_{x_k} \subseteq X_\varepsilon$.

■

Proposition 2.7.4 (Generable path connectedness)**:** *An open, connected subset U of $\mathbb{R}^n$ is always* generable-path-connected*: for any $a, b \in U \cap \mathbb{K}^n$, there exists $(\phi : \mathbb{R} \to U) \in \mathrm{GVAL}_\mathbb{K}$ such that $\phi(0) = a$ and $\phi(1) = b$.* ♦

Proof. Let $a, b \in U \cap \mathbb{K}^n$ and apply Lemma 2.7.3 (Polygonal path connectedness) to get a polygonal path $\gamma : [0,1] \to U$ from a to b. We are going to build a highly smoothed approximation of γ. This is usually done using bump functions but bump functions are not analytic, which complicates the matter. Furthermore, we need to build a path which domain of definition is $\mathbb{R}$, although this will be a minor annoyance only. We ignore the case where $a = b$ which is trivial and focus on the case where $a \neq b$.

Let $X = \gamma([0,1])$ which is a compact connected set. Apply Lemma 2.7.2 (Offset of a compact set) to get $\varepsilon > 0$ such that $X_\varepsilon \subseteq U$. Without loss of generality, we can assume that $\varepsilon \in \mathbb{Q}$ so that it is generable.

Assume for a moment that γ is trivial, that is γ is a line segment from a to b. Let $\alpha \in \mathbb{N} \subseteq \mathbb{K}$ such that $\frac{1}{\tanh(\alpha)} \leqslant 1 + \frac{2\varepsilon}{\|b-a\|}$. It exists because $\frac{1}{\tanh(x)} \xrightarrow[x\to\infty]{} 1$. Define $\phi(t) = a + \frac{1+\mu(t)}{2}(b-a)$ where $\mu(t) = \frac{\tanh((2t-1)\alpha)}{\tanh(\alpha)}$. One can check that μ is an increasing function and that $\mu(0) = -1$ and $\mu(1) = 1$. Furthermore, if $t > 1$, $|\mu(t) - 1| < \frac{2\varepsilon}{\|b-a\|}$, and conversely, if $t < 0$, $|\mu(t) + 1| < \frac{2\varepsilon}{\|b-a\|}$. Consequently, $\phi(0) = a$, $\phi(1) = b$ and $\phi([0,1])$ is the line segment between a and b, so $\phi([0,1]) \subseteq X$. Furthermore, if $t < 0$, $\|a - \phi(t)\| \leqslant \left|\frac{1+\mu(t)}{2}\right| \|b-a\| < \varepsilon$, and if $t > 1$, $\|b - \phi(t)\| \leqslant \left|\frac{1-\mu(t)}{2}\right| \|b-a\| < \varepsilon$. We conclude from this analysis that $\phi(\mathbb{R}) \subseteq X_\varepsilon \subseteq U$. It remains to show that $\phi \in \mathrm{GVAL}_\mathbb{K}$. Using Lemma 2.1.10 (Arithmetic on generable functions), it suffices to show that $\tanh \in \mathrm{GVAL}_\mathbb{K}$ and $\frac{1}{\tanh(\alpha)} \in \mathbb{K}$. Since $\mathbb{K}$ is a field, we need to show that $\tanh(\alpha) \in \mathbb{K}$ which is a consequence of $\mathbb{K}$ being a generable field and tanh being a generable function. We already saw in Example 2.1.6 (Some generable elementary functions) that $\tanh \in \mathrm{GVAL}_\mathbb{Q} \subseteq \mathrm{GVAL}_\mathbb{K}$.

In the general case where γ is a polygonal path, there are $0 = t_1 < t_2 < \ldots < t_k = 1$ such that $\gamma\restriction_{[t_i, t_{i+1}]}$ is the line segment between $x_i = \gamma(t_i)$ and $x_{i+1} = \gamma(t_{i+1})$, furthermore we can always take $x_i \in \mathbb{Q}^n$. Note that we can choose any parametrization for the path so in particular we can take $t_i = \frac{i}{k}$ and ensure that $t_i \in \mathbb{Q}$ for $i \in [\![0, k]\!]$. Since by hypothesis $x_0, x_n \in \mathbb{K}^n$, we get that $x_i \in \mathbb{K}^n$ and $t_i \in \mathbb{K}$ for all $i \in [\![0, k]\!]$.

Let us denote by $\phi_\varepsilon^{a,b}$ the path built in the previous case. We are simply going to add several instances of this path, with the necessary shifting and scaling. Since the errors will sum up, we will increase the approximation precision of each segment. Define $\phi(t) = a + \sum_{i=1}^{k-1} \left(\phi_{\varepsilon/k}^{x_i, x_{i+1}}\left(\frac{t-t_i}{t_{i+1}-t_i}\right) - x_i\right)$ and consider the following cases:

- if $t < 0$, then $\left\|\phi_{\varepsilon/k}^{x_i, x_{i+1}}\left(\frac{t-t_i}{t_{i+1}-t_i}\right) - x_i\right\| < \frac{\varepsilon}{k}$ for all $i \in [\![1, k-1]\!]$, so $\|a - \phi(t)\| < \frac{k-1}{k}\varepsilon$ and $\phi(t) \in X_\varepsilon$

- if $t \in [t_j, t_j + 1]$ for some j, then $\left\|\phi_{\varepsilon/k}^{x_i,x_{i+1}}\left(\frac{t-t_i}{t_{i+1}-t_i}\right) - x_i\right\| < \frac{\varepsilon}{k}$ for all $i > j$, and conversely $\left\|\phi_{\varepsilon/k}^{x_i,x_{i+1}}\left(\frac{t-t_i}{t_{i+1}-t_i}\right) - x_{i+1}\right\| < \frac{\varepsilon}{k}$ for all $i < j$. Finally $u = \phi_{\varepsilon/k}^{x_j,x_{j+1}}\left(\frac{t-t_j}{t_{j+1}-t_j}\right)$ belongs to the line segment from x_j to $x_j + 1$. Since $a = x_1$, we get that $\|u - \phi(t)\| \leqslant \frac{k-1}{k}\varepsilon$ and thus $\phi(t) \in X_\varepsilon$.
- if $t > 1$ then $\|b - \phi(t)\| < \varepsilon$ for the same reason as $t < 0$, and thus $\phi(t) \in X_\varepsilon$.

We conclude that $\phi(\mathbb{R}) \subseteq X_\varepsilon \subseteq U$ and one easily checks that $\phi(0) = a$ and $\phi(1) = b$. Furthermore $\phi \in \mathrm{GVAL}_{\mathbb{K}}$ by Lemma 2.1.10 (Arithmetic on generable functions) and because the x_i and t_i belong to $\mathbb{K}$ (see the details in the case of the trivial path). ■

The immediate corollary of this result is that $\mathbb{K}$ is also preserved by multidimensional generable functions. Indeed, by composing a multidimensional function with a unidimensional one, we get back to the unidimensional case and conclude that any generable point in the input domain must have a generable image.

Corollary 2.7.5 (Generable field stability)**:** *Let* $(f :\subseteq \mathbb{R}^d \to \mathbb{R}^e) \in \mathrm{GVAL}$*, then* $f(\mathbb{K}^d \cap \operatorname{dom} f) \subseteq \mathbb{K}^e$. ♦

Proof. Apply Definition 2.1.17 (Generable function) to get $n \in \mathbb{N}$, $p \in M_{n,d}(\mathbb{K})[\mathbb{R}^n]$, $x_0 \in \operatorname{dom} f \cap \mathbb{K}^d$, $y_0 \in \mathbb{K}^n$ and $y : \operatorname{dom} f \to \mathbb{R}^n$. Let $u \in \operatorname{dom} f \cap \mathbb{K}^d$. Since $\operatorname{dom} f$ is open and connected, by Proposition 2.7.4 (Generable path connectedness), there exists $(\gamma : \mathbb{R} \to \operatorname{dom} f) \in \mathrm{GVAL}$ such that $\gamma(0) = x_0$ and $\gamma(1) = u$. Apply Definition 2.1.17 (Generable function) to γ to get $\bar{n} \in \mathbb{N}$, $\bar{p} \in M_{\bar{n},1}(\mathbb{K})[\mathbb{R}^{\bar{n}}]$, $\bar{x}_0 \in \mathbb{K}$, $\bar{y}_0 \in \mathbb{K}^{\bar{n}}$ and $\bar{y} : \mathbb{R} \to \mathbb{R}^{\bar{n}}$. Define $z(t) = y(\gamma(t)) = y(\bar{y}_{1..d}(t))$, then $z'(t) = J_y(\gamma(t))\gamma'(t) = p(y(\gamma(t)))\gamma'(t) = p(z(t))\bar{p}_{1..d}(\bar{y}(t))$ and $z(0) = y(\gamma(0)) = y(x_0) = y_0$. In other words $(\bar{y}, z)$ satisfy:

$$\begin{cases} \bar{y}(0) = x_0 \in \mathbb{K}^d \\ z(0) = y_0 \in \mathbb{K}^n \end{cases} \qquad \begin{cases} \bar{y}' = \bar{p}(\bar{y}) \\ z' = p(z)\bar{p}_{1..e}(\bar{y}) \end{cases}$$

Consequently $(z : \mathbb{R} \to \mathbb{R}^e) \in \mathrm{GVAL}$ so, by definition of a generable field, $z(\mathbb{K}) \subseteq \mathbb{K}^e$. Conclude by noticing that $z(1) = y(\gamma(1)) = y(u)$. ■

2.7.II How to build a smallest field

To make precise statements about what it means to be the smallest generable field, we will make use of order and lattice theory and Kleene or Knaster-Tarski fixed-point theorem [Tar55]. First we need to define the *complete partial order* (CPO) which contains our sets. Recall that the smallest field we are looking for is a subset of $\mathbb{R}$ but it must also contains at least $\mathbb{Q}$.

Definition 2.7.6: Define $\mathcal{L} = \{X \in \mathcal{P}(\mathbb{R}) \mid \mathbb{Q} \subseteq X\}$ the set of subsets of $\mathbb{R}$ containing $\mathbb{Q}$ and $\bot = \mathbb{Q}$. ♦

Lemma 2.7.7: *The partially ordered set* $(\mathcal{L}, \subseteq)$ *is a complete partial order, and even a lattice. Its least element is* $\bot$*, the supremum of several elements of* $\mathcal{L}$ *is the union and the infimum is the intersection:* $\sup X = \bigcup_{x \in X} x$ *and* $\inf X = \bigcap_{x \in X} x$. ♦

Proof. The only thing to prove is that $\mathcal{L}$ is closed under union and intersection. This is the case because the union of two subsets of $\mathbb{R}$ containing $\mathbb{Q}$ also contains $\mathbb{Q}$, and similarly for the intersection. All the other properties are trivial. ■

We will now consider the following operator on $\mathcal{L}$ which gives all constants we can build from a set X using generable functions with constants in X.

$$G : \begin{cases} \mathcal{L} & \rightarrow & \mathcal{L} \\ X & \mapsto & \displaystyle\bigcup_{f \in \mathrm{GVAL}_X} f(X) \end{cases}$$

Remark 2.7.8 (G is well-defined)**:** One can check that G is well-defined. Indeed $X \subseteq G(X)$ for any $X \subseteq \mathbb{R}$, thus if $X \in \mathcal{L}$ then $G(X) \supseteq X \supseteq \mathbb{Q}$, so $G(X) \in \mathcal{L}$. This stems for the fact that for any $x \in X$, the constant function $u \mapsto x$ belongs to GVAL_X. ♦

Another interesting property of G is that its definition can be simplified. More precisely, by rescaling the functions, we can always assume that the image of G is produced by the evaluation of generable functions at a particular point, say 1, instead of the entire field.

Lemma 2.7.9 (Alternative definition of G)**:** *If X is a field then,*

$$G(X) = \left\{ f(1), f \in \mathrm{GVAL}_X \right\}$$

♦

Proof. Let $x \in G(X)$, then there exists $f \in \mathrm{GVAL}_X$ and $t \in X$ such that $x = f(t)$. Consequently there exists $d \in \mathbb{N}$, $y_0 \in X^d$, $p \in X^d[\mathbb{R}^d]$ and $y : \mathbb{R} \rightarrow \mathbb{R}^d$ satisfying Definition 2.1.1 (Generable function):

- $y' = p(y)$ and $y(0) = y_0$
- $y_1 = f$

Consider $g(u) = f(ut)$ and note that $g(1) = f(t) = x$. We will see that $g \in \mathrm{GVAL}_X$. Indeed, consider $z(u) = y(tu)$ then for all $u \in \mathbb{R}$:

- $z(0) = y(0) = y_0 \in X^d$;
- $z'(u) = ty'(tu) = tp(z(u)) = q(z(u))$ where $q = tp$ is a polynomial with coefficients in X since $t \in X$ and X is a field
- $z_1(u) = y_1(tu) = g(u)$

■

A consequence of this alternative definition is a simple proof that G preserves the property of being a field.

Lemma 2.7.10 (G maps fields to fields)**:** *If X is a field containing $\mathbb{Q}$, then $G(X)$ is a field.* ♦

Proof. Let $x, y \in G(X)$, by Lemma 2.7.9 (Alternative definition of G) there exists $f, g \in \mathrm{GVAL}_X$ such that $x = f(1)$ and $y = g(1)$. Apply Lemma 2.1.10 (Arithmetic on generable functions) to get that $f \pm g$ and fg belong to GVAL_X And thus $x \pm y$ and xy belong to $G(X)$.

Finally the case of $\frac{1}{x}$ (when $x \neq 0$) is slightly more subtle: we cannot simply compute $\frac{1}{f}$ because f may cancel. Instead we are going to compute $\frac{1}{g}$ where $g(1) = f(1)$ but g nevers cancels.

First, note that we can always assume that $x > 0$ because $G(X)$ is closed under the negation, and $-\frac{1}{x} = \frac{1}{-x}$. Since $f(1) = x > 0$ and f is continuous, it means there exists $\varepsilon > 0$ such that $f(t) > 0$ for all $t \in [1-\varepsilon, 1+\varepsilon]$ and we can take $\varepsilon \in \mathbb{Q}$. Define $g(t) = f(t) + \left(1 + f(t)^2\right)\left(\frac{t-1}{\varepsilon}\right)^2$. It is not hard to see that $g(1) = f(1)$ and that $g(t) > 0$ for all $t \in \mathbb{R}$. Furthermore, $g \in \mathrm{GVAL}_X$

because of Lemma 2.1.10 (Arithmetic on generable functions). Note that we use the part of the lemma which does not assume that X is a generable field !

Using Lemma 2.1.10 (Arithmetic on generable functions), we conclude that $\frac{1}{g} \in \mathrm{GVAL}_X$ and thus $\frac{1}{x} \in G(X)$. ■

Finally, the core of what makes G very special is its finiteness property. Essentially, it means that if $x \in G(X)$ then x really only requires a finite number of elements in X to be computed.

Lemma 2.7.11 (Finiteness of G): *For any $X \subseteq \mathbb{R}$ and $x \in G(X)$, there exists a finite $Y \subseteq X$ such that $x \in G(Y)$.* ♦

Proof. Let $x \in G(X)$, then there exists $f \in \mathrm{GVAL}_X$ and $t \in X$ such that $x = f(t)$. Then there exists $y_0 \in X^d$ and a polynomial p with coefficients in X such that f satisfies Definition 2.1.1 (Generable function). Define Y as the subset of X containing t, the components of y_0 and all the coefficients of p. Then Y is finite and $f \in \mathrm{GVAL}_Y$. Furthermore $t \in Y$ so $x \in G(Y)$. ■

This consequence of this finiteness property is that G is continuous, because G only requires local (finite) information to be computed.

Lemma 2.7.12 (Continuity of G): *G is a Scott-continuous function between the CPO $(\mathcal{L}, \subseteq)$ and itself.* ♦

Proof. We have to show that G preserves all directed suprema. First, we see that G preserves all directed subsets[8] because G is monotone (also known as order preserving). Indeed, if $A \subseteq B$ then $G(A) \subseteq G(B)$ since $\mathrm{GVAL}_A \subseteq \mathrm{GVAL}_B$. Second, if $\mathcal{A}$ is a directed subset with supremum[9] P, then $\mathcal{B} = G(\mathcal{A})$ is a directed subset[10] with supremum Q and we need to show that $Q = G(P)$, that is $\sup G(\mathcal{A}) = G(\sup \mathcal{A})$.

The fact that $Q \subseteq G(P)$ is a well-known consequence of the fact that G is monotone. The other direction captures the essence of continuity. Let $q \in G(P)$, by Lemma 2.7.11 (Finiteness of G), there exists a finite subset $P_{fin} \subseteq P$ such that $q \in G(P_{fin})$. Since $P = \sup \mathcal{A} = \bigcup_{A \in \mathcal{A}} A$, there exists a finite subset $\mathcal{A}_{fin} \subseteq \mathcal{A}$ such that $P_{fin} \subseteq \bigcup_{A \in \mathcal{A}_{fin}} A$, that is $P_{fin} \subseteq \sup \mathcal{A}_{fin}$. Since $\mathcal{A}$ is a directed subset and $\mathcal{A}_{fin}$ is a finite, its supremum belongs to $\mathcal{A}$: $\sup \mathcal{A}_{fin} \in \mathcal{A}$. Consequently $G(\mathcal{A}_{fin}) \in \mathcal{B}$ and the moniticity of G yields that $q \in G(\mathcal{A}_{fin})$. And since Q is the supremum of $\mathcal{B}$, we have that $G(\mathcal{A}_{fin}) \subseteq Q$ which gives the result. ■

A consequence of this result is the existence of a least fixed point for G, as well as an explicit formula.

Corollary 2.7.13 (G has a least fixed point): *G has a least fixed point in the $(\mathcal{L}, \subseteq)$ CPO, which is supremum of the* ascending Kleene chain*:*

$$\perp \subseteq G(\perp) \subseteq G(G(\perp)) \subseteq \ldots \subseteq G^{[n]}(\perp) \subseteq \ldots$$

♦

Proof. Use Lemma 2.7.12 (Continuity of G) and Kleene or Knaster-Tarski fixed-point theorem [Tar55]. ■

[8]we recall that $A \subseteq \mathcal{P}(\mathbb{R})$ is a directed subset if every pair of elements (or equivalently every finite subset of A) has an upper bound in A

[9]in the $(\mathcal{L}, \subseteq)$ CPO, the supremum of a subset is always in $\mathcal{L}$

[10]because G is monotone

2.7.III Generable real numbers

We can now formally define our smallest generable field. Intuitively, $\mathbb{R}_G$ contains all the "generable" real numbers, thus the name. Its existence is given by Corollary 2.7.13 (G has a least fixed point).

Definition 2.7.14 (Generable real numbers)**:** The set of *generable real numbers* is the least fixed point of G, which we denote by $\mathbb{R}_G$. ♦

Before moving on to the main result of this section, we show that G preserves polynomial time computability.

Lemma 2.7.15 (G preserves polytime computability)**:** *G maps subsets of polynomial time computable real numbers into themselves,* i.e. *for any $X \subseteq \mathbb{R}_P$, $G(X) \subseteq \mathbb{R}_P$.* ♦

Proof. Let $X \subseteq \mathbb{R}_P$ and $x \in G(X)$, $f \in \text{GVAL}_X$ and $t \in X$ such that $x = f(t)$. We can apply Theorem 3.5.2 (PIVP complexity) to conclude that x is polynomial time computable, thus $x \in \mathbb{R}_P$. Note that although this theorem is in the *next* chapter, it is independent of the work in this chapter. ■

The main result of this section is, of course, that $\mathbb{R}_G$ is a generable field. But more surprisingly, we show that all the elements of $\mathbb{R}_G$ are polynomial time computable (in the sense of Computable Analysis).

Theorem 2.7.16 ($\mathbb{R}_G$ is generable subfield of $\mathbb{R}_P$)**:** *Generable real numbers form a generable subfield of polynomial time computable real numbers in the sense of Computable Analysis, i.e. $\mathbb{R}_G$ is a generable field and $\mathbb{R}_G \subseteq \mathbb{R}_P$.* ♦

Proof. From the the structure of the $(\mathcal{L}, \subseteq)$ CPO, we can derive an explicit expression for $\mathbb{R}_G$ using Corollary 2.7.13 (G has a least fixed point). Indeed, in this particular *CPO* the supremum is simply the union, so:

$$\mathbb{R}_G = \bigcup_{n \geqslant 0} G^{[n]}(\bot)$$

Since $\bot = \mathbb{Q} \subseteq \mathbb{R}_P$, iterating Lemma 2.7.15 (G preserves polytime computability) yields that $G^{[n]}(\bot) \subseteq \mathbb{R}_P$ for all $n \in \mathbb{N}$ and thus $\mathbb{R}_G \subseteq \mathbb{R}_P$.

The fact that $\mathbb{R}_G$ is a field comes from the above formula for $\mathbb{R}_G$ and the repeated application of Lemma 2.7.10 (G maps fields to fields). Finally, the fact it is a generable field is a trivial consequence of $\mathbb{R}_G$ being a fixed point of G: by definition $G(\mathbb{R}_G) = \mathbb{R}_G$, so the image of any generable number by a generable function is generable. ■

Chapter 3

Solving PIVP

The whole point of mathematics is to solve differential equations!

Quote from Cambridge

In this chapter we investigate the computational complexity of solving ordinary differential equations (ODEs) $y' = p(y)$ over *unbounded time domains*, where p is a (vector of) polynomials. Contrarily to the bounded case, this problem has not been well-studied, apparently due to the "conventional wisdom" that it can always be reduced to the bounded case by using rescaling techniques. However, as we show in this chapter, rescaling techniques do not seem to provide meaningful insights on the complexity of this problem, since the use of such techniques introduces a dependence on parameters that are hard to compute.

We present algorithms that numerically solve these ODEs over unbounded time domains. These algorithms have guaranteed precision, i.e. given some arbitrarily large time t and error bound ε as input, they will output a value $\tilde{y}$ that satisfies $\|y(t) - \tilde{y}\| \leq \varepsilon$. We analyze the complexity of these algorithms and show that they compute $\tilde{y}$ in time polynomial in several quantities including the time t, the precision of the output ε and the length of the curve y from 0 to t. We consider both algebraic complexity and bit complexity. As far as we know, this is the first time is it proved to be polynomial time computable over unbounded domains for this large class of ODEs. The restriction to a polynomial right-hand side is necessary to obtain a large enough, yet tractacle class, see Section 3.1.I (Related work for general ODEs) for more details.

This chapter is organised as follows:

- Section 3.1 (Introduction) provides an extensive introduction to this problem and a detailed explanation of why this problem was not solved before;
- Section 3.2 (The generic Taylor method) describes a generic, adaptive, variable order Taylor algorithm and proves necessary conditions for its convergence;
- Section 3.3 (The adaptive Taylor algorithm) instantiates the previous algorithm and proves that the algorithm is correct and complete;
- Section 3.4 (Enhancement on the adaptive algorithm) explains how to enhance the previous algorithm to remove the extra "hint" parameter;
- Section 3.5 (Extension to Computable Analysis) extends this result to the case where the inputs (time, coefficients) are not rational numbers, using the framework of Computable Analysis.

3.1 Introduction

The purpose of this chapter is to characterize the computational complexity needed to solve a polynomial initial-value problem (IVP) defined by

$$\begin{cases} y'(t) = p(y(t)) \\ y(t_0) = y_0 \end{cases} \tag{3.1.1}$$

over an unbounded domain. Since the system is autonomous, we can assume, without loss of generality, that $t_0 = 0$. More precisely, we want to compute $y(t)$ with precision 2^{-n}, where $t \in \mathbb{R}$, $n \in \mathbb{N}$, and a description of p are given as inputs, and y is the solution of (3.1.1).

3.1.I Related work for general ODEs

There are many results about the computational complexity of solving ODEs of the form:

$$\begin{cases} y'(t) = f(t, y(t)) \\ y(t_0) = y_0 \end{cases} \tag{3.1.2}$$

However, with very few exceptions, those results assume that the ODE is solved for $t \in I = [a, b]$, i.e. they assume that the ODE is solved over a compact time domain. This is a very convenient hypothesis, since a C^1 function f is always Lipschitz[1] over a compact set and in this case the proof of the existence-uniqueness theorem for ODEs (the Picard-Lindelöf theorem) provides a method (Picard's iterations) to compute the solution over the compact I.

The introduction of a Lipschitz condition seems to be fundamental when studying the computational complexity of (3.1.2). It is well-known (see e.g. [Ko91, Theorem 7.3]) that if f is not Lipschitz, then the solution of (3.1.2) can have arbitrarily high complexity, even if f is assumed to be polynomial-time computable and (3.1.2) has a unique solution. The Lipschitz condition plays an instrumental role in the complexity because it is used to derive the number of steps of the algorithm, for example in Picard-Lindelöf theorem it is used to bound the number of iterations.

It was recently shown [Kaw10], following an open problem from Ko [Ko91, Section 7.2], that if f is Lipschitz and polynomial-time computable, then the solution of (3.1.2) can still be PSPACE-complete (but it was already known not to be of higher complexity [Ko91]).

We can conclude from the analysis of these algorithms that the time complexity of solving (3.1.2), that is of computing $y(T) \pm 2^{-m}$, is bounded by $\Theta(f, y_0, T, K, n)$, where K is the Lipschitz constant. Note that K depends not only on f, but also on I: if f is defined on two domains I_1, I_2, (3.1.2) might be solved in less computational time in I_1 than in I_2, due to a smaller Lipschitz constant on the first domain. Think for example of $f(x) = x^2$, $I_1 = [0, 1]$ and $I_2 = [0, 10]$: the Lipschitz constants are 1 and 20 respectively.

In the compact case where I is compact and f is C^1-computable, we can always take K to be any constant greater or equal to

$$\max_{t \in [0,T]} \|f'(y(t))\|$$

due to the intermediate value theorem. Therefore, since K is a constant for a fixed compact domain and T is bounded by a constant, the computational complexity of solving (3.1.2) will be $\Theta(f, y_0, n)$ and we fall into the usual case.

The case where I is unbounded (typically $I = \mathbb{R}_+$ or $I = \mathbb{R}$) has already been addressed by some authors. For example, if the function f in (3.1.2) is Lipschitz over $\mathbb{R}$, then the solution

[1] We recall that a function is Lipschitz over I if $\|f(x) - f(y)\| \leq K\|x - y\|$ for some constant K and $x, y \in I$

of (3.1.2) is computable over $\mathbb{R}$ [Abe70], [Abe80], [Ko91]. Of course, requiring a Lipschitz condition for f over the real line is a very restrictive condition. Alternatively if an effective bound for f is known or a very restricted condition on its growth is met, the solution is also computable [Ruo96]. However these cases exclude most of the interesting IVPs, and even some very simple ones like $y' = -y^2$, $y(1) = 1$.

One of the most general result in the field is [CG09] which proves that if f is continuous and computable, and the solution of (3.1.2) is assumed to be unique[2], then it is computable over its whole domain, whether it is bounded or not. Moreover, it can even be shown that there is a computable (and continuous) function f and a computable y_0, such that (3.1.2) admits infinitely many solutions, but none of them is computable [Abe71], [PER79]. It is also known that the problem of deciding whether the maximal interval of definition of an IVP is bounded or not is undecidable and that, in general, one cannot compute this interval even when we know it is bounded [GZB09].

Finally, this problem has been widely studied in Numerical Analysis but the point of view is usually different and more focused on practically fast algorithms rather than asymptotically efficient algorithms. Some work [ISC08, Cor02, Wer79, Smi06] suggests that any polynomial time algorithm must have variable order[3], and that adaptive algorithms are theoretically superior to non-adaptive ones[4]. While adaptive algorithms have been widely studied, high-order algorithms are not mainstream become there are expensive in pratice. Variable order methods have been used in [CC82], [JZ05], [Smi06], [BRAB11], [ABBR12] and some polynomial time algorithms have been obtained over compact domains or over arbitrary domains but with stronger hypothesis.

In [BGP12] we have shown when f is a polynomial (i.e. y is solution of (3.1.1)) it can be solved in time polynomial in t, in the precision of the result, and in $\max_{u\in[0,t]} \|y(u)\|^k$ where k is the degree of p. However this result is in some sense a worst-case scenario: if $y(t)$ "spikes" for a very brief amount of time, then the resulting complexity will be high. In the present chapter we improve the bound by showing that an ODE (3.1.5) can be solved in time polynomial in t, in the precision of the result, and in $\int_0^t \|y(u)\|^k du$. Therefore even if $y(t)$ "spikes", as long as it does it for a very brief period, the running time of our algorithm will still be reasonable.

3.1.II On the difficulties of unbounded domains

There are significant problems when solving ODEs over unbounded domains. From discussions we had with other researchers, it seems that there exists a widely held (false) belief that, by rescaling techniques, the (time) complexity of solving an ODE (3.1.2) over an unbounded set $I = \mathbb{R}$ should be equal to the complexity of solving (3.1.2) over $I = [a, b]$. The argument is as follows.

Let us assume that the computational time needed to solve (3.1.2) over the time interval $I = [0, 1]$ is $\Theta(n)$, where 2^{-n} is the precision to which the solution is computed (we assume for now that f is fixed). Then we can find $y(t)$ by using the "rescaled" IVP

$$\begin{cases} x'(u) = tf(tu, x(u)) \\ x(0) = y_0 \end{cases} \qquad u \in [0,1] \tag{3.1.3}$$

The solution $x(\cdot)$ of this problem satisfies $x(u) = y(tu)$ yielding, in particular, $x(1) = y(t)$. Thus the complexity of finding $y(t)$ with precision 2^{-n} should be the complexity of finding $x(1)$ with the same precision and thus should be $\Theta(n)$. However this reasoning is incorrect. The reason is

[2]It is known that (3.1.2) can have non unique solutions for some continuous f

[3]See Section 1.2 (Numerical Analysis and Differential Equations) for a description of typical numerical algorithms

[4]This is in contrast with classical results which state the contrary but under usually unrealistic hypothesis

that Θ also depends on the Lipschitz constant valid for the current time interval, and different rescalings from $[0, t_1]$ and $[0, t_2]$ into the interval $[0, 1]$ will yield different systems (3.1.3) with different Lipschitz constants which are very difficult to predict in advance. Moreover, even if we could somehow perform the hard task of computing these Lipschitz constants, we still need to know how the computational complexity of (3.1.2) depends on the value of the Lipschitz constant on the interval $[0, 1]$.

As a concrete example that the previous argument involving rescaling is incorrect, consider the quadratic IVP defined by

$$\begin{cases} y_1(0) = 1 \\ y_2(0) = e \end{cases} \qquad \begin{cases} y_1' = y_1 \\ y_2' = y_1 y_2 \end{cases} \tag{3.1.4}$$

One can check that the solution of this IVP is given by $y_1(t) = e^t, y_2(t) = e^{e^t}$. In particular, for $t \in I = [0, 1]$ (or, for that effect, on any bounded interval $I = [a, b]$), one can can compute the solution $y(t)$ of (3.1.4) with precision 2^{-n} in time $p(n)$, where p is a polynomial [MM93]. However time polynomial in n is not enough to compute $y(t)$ for $t \in \mathbb{R}$. This is because one component of the solution of (3.1.4) is $y_2(t) = e^{e^t}$ and simply writing down the integer part of $y_2(t)$ requires time at least e^t. Consequently, for large enough t, any polynomial time algorithm will not have the time to write the solution and thus cannot be correct. For this reason, a dependence on t must also be introduced. It seems that the complexity, in this example, should instead be something like $e^t p(n)$ (probably this expression is not exact, but assume without loss of generality that this is the correct time bound), which is exponential in t. In some sense e^t is the time needed to write down the integer part of the solution, while $p(n)$ is the time needed to write the fractional part up to precision 2^{-n}.

The subtlety in the previous example is that in any *fixed* compact $I = [a, b]$, we can always say that the complexity is $O(p(n))$, since the exponential e^t (which is roughly a local Lipschitz constant) is bounded by some constant $A_{a,b} = e^b$, for $b > a \geq 0$, *which depends on the interval* $I = [a, b]$. So, for fixed intervals, the constant $A_{a,b}$ is also fixed and the computational complexity needed to solve (3.1.4) is $O(A_{a,b}p(n)) = O(p(n))$, even if $b > 1$. However, when $I = \mathbb{R}$, we can no longer hide the dependence in t of the Lipschitz constant and of the computational time needed to solve (3.1.4) into a constant — this dependency must be explicitly analyzed. Therefore, in general, the time needed to solve (3.1.2) will depend in a non-obvious manner in t.

We hope that the reader is now convinced that simply knowing the computational complexity needed to solve an IVP (3.1.2) over a compact domain is not enough to characterize the computational complexity needed to solve the IVP over an unbounded domain, unless it explicitely takes into account parameters that are traditionally assumed to be constants.

3.1.III Contributions

In this chapter we analyze the computational complexity of solving the polynomial initial value problem (3.1.1) over an unbounded time interval I that, for simplicity reasons, we assume to be the real line: $I = \mathbb{R}$. Recall that a polynomial initial value problem is a differential system of the form:

$$y'(t) = p(y(t)) \tag{3.1.5}$$

where each component of p is a polynomial and $y(t)$ is a vector. The reasons to focus on this class of systems are two-fold. First polynomials are relatively simple functions, with many properties that we can exploit to produce algorithms that solve (3.1.5) efficiently over unbounded domains. Secondly, this class is very general because many systems of the form

(3.1.2) can be rewritten as (3.1.5). See for example [CPSW05, GBC09] or Chapter 2 (The PIVP class).

In this chapter we will show that the PIVP (3.1.5) can be solved over an unbounded time interval in time polynomial in:

- the precision of the output
- the size of the initial condition
- the coefficients and the degree of the polynomial p
- the length of the solution curve from 0 to t

However the complexity is exponential in the dimension of the system. This is to be expected because evaluating the truncated Taylor series is polynomial in the number of derivatives but exponential in the dimension. Unless some breakthrough is achieved in this area, it seems unlikely to find a polynomial time algorithm in the dimension.

Note that we are measuring computational complexity against the length of the solution curve (instead of the Lipschitz constant for example). This paramater has a natural geometrical interpretation and suggests that the best we can do to solve (3.1.5) is to "follow" the curve, and thus the complexity of the algorithm is related to the distance we travel, that is the length of the curve.

Finally, our algorithm does not need to know in advance a bound on the length of the curve: it can automatically discover it. In this case, the complexity of the algorithm is not known in advance but we know that the running time is polynomial in this (unknown) quantity. Note that this algorithm does not contradict the uncomputability of the maximum interval of definition of the solution. Indeed, if the solution explodes in finite time, the length of the curve will be infinite after a finite time and thus this algorithm will not terminate for inputs outside of the interval of definition.

3.2 The generic Taylor method

We consider a generic adaptive Taylor meta-algorithm to numerically solve (2.0.1). This is a meta-algorithm in the sense that we will leave open, for now, the question of how we choose some of the parameters. The goal of this algorithm is, given as input $t \in \mathbb{Q}$ and $0 < \varepsilon < 1$ and the initial condition of (2.0.1), to compute $x \in \mathbb{Q}^d$ such that $\|x - y(t)\| < \varepsilon$.

We assume that the meta-algorithm uses the following values:

- $n \in \mathbb{N}$ is the number of steps of the algorithm
- $t_0 < t_1 < \ldots < t_n = t$ are the intermediate times
- $\delta t_i = t_{i+1} - t_i \in \mathbb{Q}$ are the time steps
- for $i \in [\![0, n-1]\!]$, $\omega_i \in \mathbb{N}$ is the order at time t_i and $\mu_i > 0$ is the rounding error at time t_i (see (3.2.1))
- $\tilde{y}_i \in \mathbb{Q}^d$ is the approximation of y at time t_i

This meta-algorithm works by solving the ODE (2.0.1) with initial condition $y(t_i) = \tilde{y}_i$ over a small time interval $[t_i, t_{i+1}]$, yielding as a result the approximation $\tilde{y}_{i+1}$ of $y(t_{i+1})$. This approximation over this small time interval is obtained using the algorithm of Section 2.5 (Taylor series of the solutions), through a Taylor approximation of order ω_i (for now we do not fix the value ω_i to analyze its influence on the error and time complexity. After this analysis is done, we can choose appropriate values for ω_i — done in Section 3.3). This procedure is repeated recursively over $[t_0, t_1], [t_1, t_2], \dots, [t_i, t_{i+1}], \dots$ until we reach the desired time $t_n = t$. Therefore the meta-algorithm is only assumed to satisfy the following inequality at each step:

$$\left\|\tilde{y}_{i+1} - T_{t_i}^{\omega_i}\Phi(t_i, \tilde{y}_i, \cdot)(t_{i+1})\right\| \leqslant \mu_i \quad \text{for some } \mu_i \tag{3.2.1}$$

We will now see what is the influence of the choice of the different parameters and under which circumstances we have $\|\tilde{y}_n - y(t)\| < \varepsilon$.

We have analyzed in Section 2.5 (Taylor series of the solutions) the error made when we approximated the solution of (2.0.1), over a small time step $[t_i, t_{i+1}]$, using a Taylor approximation. Since the time interval $[0, t]$ is split into time intervals $[t_0, t_1], [t_1, t_2], \dots$ (see Section 2.4 (Dependency in the parameters)) we want to understand how the error propagates through the time interval $[0, t]$ or, more generally, through $[0, t_i]$ when we discretize the time over this interval. Let $\|y(t_i) - \tilde{y}_i\| \leqslant \varepsilon_i$ be a bound on the error made by the algorithm at step i. We want to obtain a bound on $\varepsilon_n \leqslant \varepsilon$ given all the other parameters. A direct consequence of (3.2.1) and the triangle inequality is that

$$\begin{aligned} \varepsilon_{i+1} \leqslant\ & \|y(t_{i+1}) - \Phi(t_i, \tilde{y}_i, t_{i+1})\| \\ & + \left\|\Phi(t_i, \tilde{y}_i, t_{i+1}) - T_{t_i}^{\omega_i}\Phi(t_i, \tilde{y}_i, \cdot)(t_{i+1})\right\| \\ & + \mu_i \end{aligned} \tag{3.2.2}$$

The first term is usually called the global error: it arises because after one step, the solution we are computing lies on a different solution curve than the true solution. The second term is the local (truncation) error: at each step we only compute a truncated Taylor series instead of the full series. The third error is the rounding error: even if we truncate the Taylor series and evaluate it, we only have a finite number of bits to store it.

In order to bound the first two quantities, we will rely on the results of the previous sections. Since those results only hold for reasonable (not too big) time steps, we need to assume bounds on the time steps. To this end, we introduce the following quantities:

$$\begin{aligned} \beta_i &= k\Sigma p \max(1, \|\tilde{y}_i\|)^{k-1}\delta t_i \\ \gamma_i &= \int_{t_i}^{t_{i+1}} k\Sigma p(\varepsilon + \|y(u)\|)^{k-1} du \end{aligned} \tag{3.2.3}$$

where $\delta t_i = t_{i+1} - t_i$. It should be clear that the choice of the values for β_i and γ_i comes from Theorem 2.5.1 (Taylor approximation for PIVP) and Theorem 2.4.2 (Parameter dependency of PIVP), respectively. We make the following assumption (which is always true if we choose small enough time steps):

$$\beta_i < 1 \tag{3.2.4}$$

Back to (3.2.2), we apply Theorem 2.5.1 (Taylor approximation for PIVP) and Theorem 2.4.2 (Parameter dependency of PIVP) to get

$$\varepsilon_{i+1} \leqslant \varepsilon_i e^{\gamma_i} + \frac{\max(1, \|\tilde{y}_i\|)\beta_i^{\omega_i}}{1 - \beta_i} + \mu_i \tag{3.2.5}$$

We recall the following well-known result, which can be proved by induction:

Lemma 3.2.6 (Arithmetico-geometric sequence): *Let $(a_k)_k, (b_k)_k \in \mathbb{R}^{\mathbb{N}}$ and assume that $u \in \mathbb{R}^{\mathbb{N}}$ satisfies:*

$$u_{n+1} = a_n u_n + b_n, \quad n \geqslant 0$$

Then

$$u_n = u_0 \prod_{i=0}^{n-1} a_i + \sum_{i=0}^{n-1} b_i \prod_{j=i+1}^{n-1} a_j$$

♦

We can now apply Lemma 3.2.6 (Arithmetico-geometric sequence) to (3.2.5), since all quantities are positive, therefore obtaining a bound on ε_n. We further bound it using the fact that $\prod_{j=i+1}^{n-1} a_j \leqslant \prod_{j=0}^{n-1} a_j$ when $a_i \geqslant 1$. This gives a bound on the error done by the generic Taylor algorithm:

$$\begin{aligned} \varepsilon_n &\leqslant \varepsilon_0 e^A + AB \\ A &= \sum_{i=0}^{n-1} \gamma_i = \int_{t_0}^{t_n} k\Sigma p(\varepsilon + \|y(u)\|)^{k-1} du \\ B &= \sum_{i=0}^{n-1} \frac{\max(1, \|\tilde{y}_i\|)\beta_i^{\omega_i}}{1-\beta_i} + \sum_{i=0}^{n-1} \mu_i \end{aligned} \tag{3.2.7}$$

In other words, assuming that $\beta_i < 1$ yields a generic error bound on the algorithm. This leaves us with a large space to play with and optimize the parameters $(\beta_i, \gamma_i, \mu_i)$ to get a correct and efficient algorithm.

3.3 The adaptive Taylor algorithm

Having done the error analysis of the meta-Taylor algorithm in Section 3.2, we can now set parameters to obtain an algorithm that solves initial-value problems (2.0.1) efficiently over unbounded domains. In order to analyse the algorithm, it is useful to introduce the following quantity:

$$\mathrm{Int}(t_0, t) = \int_{t_0}^{t} k\Sigma p \max(1, \varepsilon + \|y(u)\|)^{k-1} du \tag{3.3.1}$$

This algorithm is simply an instance of the meta-algorithm described in Section 3.2 (The generic Taylor method) in which we explain how to choose the parameters (size δt_i of time intervals $[t_i, t_{i+1}]$ and the order ω_i of the Taylor approximation used in this time interval). This algorithm will be parametrized by the choice of a "hint" which we call I, and the number of steps n.

Equation (3.2.7) suggests that it is advantageous to choose β_i smaller than 1 so that $\beta_i^{\omega_i}$ is small and not too small so that the number of steps doesn't blow up due to (3.2.3). Since a reasonable choice for this parameter is nontrivial, we introduce another parameter λ to be fixed later and we assume that $\beta_i = \lambda$, except for the last time step which might require a smaller value to reach exactly on the final time. We assume that:

$$n > 0 \qquad I > 0 \qquad 0 \leqslant \lambda \leqslant \frac{1}{2} \tag{3.3.2}$$

Let us now establish the other parameters of the algorithm. We define the following values:

$$\delta t_i = \min\left(t - t_i, \frac{\lambda}{k\Sigma p \max(1, \|\tilde{y}_i\|)^{k-1}}\right) \tag{3.3.3}$$

$$\omega_i = \log_2 \frac{6n \max(1, \|\tilde{y}_i\|)}{\eta} \qquad \mu_i = \frac{\eta}{3n} \qquad \varepsilon_0 \leqslant \frac{\varepsilon}{3} e^{-I} \qquad \eta = \frac{\varepsilon}{I} \tag{3.3.4}$$

We will now see that under natural constraints on I, the algorithm will be correct.

Lemma 3.3.5 (Algorithm is conditionally correct)**:** *If $I \geqslant \operatorname{Int}(t_0, t_n)$ then the choices of the parameters, implied by* (3.3.3) *and* (3.3.4) *guarantee that $\varepsilon_n \leqslant \varepsilon$.* ♦

Proof. We only need to put all pieces together using (3.2.7). Indeed, it is clear that

$$A \leqslant \operatorname{Int}(t_0, t_n) \leqslant I. \tag{3.3.6}$$

Furthermore the way we chose λ implies $\beta_i \leqslant \lambda$, and together with (3.3.4) and (3.3.1), it implies that:

$$\frac{\beta_i^{\omega_i}}{1-\beta_i} \leqslant \frac{\lambda^{\omega_i}}{1-\lambda} \leqslant \frac{2^{-\omega_i}}{1/2} \leqslant \frac{2\eta}{6n\max(1, \|\tilde{y}_i\|)}$$

Thus:

$$\begin{aligned}\varepsilon_n &\leqslant \varepsilon_0 e^A + AB \\ &\leqslant \frac{\varepsilon}{3}e^{-I}e^I + I\left(\sum_{i=0}^{n-1}\frac{\eta}{3n} + \sum_{i=0}^{n-1}\frac{\eta}{3n}\right) \\ &\leqslant \frac{\varepsilon}{3} + I\frac{2\eta}{3} \leqslant \varepsilon\end{aligned}$$

■

This result shows that if we know $\operatorname{Int}(t_0, t)$ then we can compute the solution numerically by plugging $I = \operatorname{Int}(t_0, t)$ in the algorithm. However, this result does not tell us anything about how to find it. Furthermore, a desirable property of the algorithm would be to detect if the argument I is not large enough, instead of simply returning garbage. Another point of interest is the choice of n: if we pick n too small in the algorithm, it will be correct but $t_n < t$, in other words we won't reach the target time.

Both issues boil down to the lack of relationship between I and n: this is the point of the following lemma. We will see that it also explains how to choose λ.

Lemma 3.3.7 (Relationship between n and I)**:** *Assume that $k \geqslant 2$ and*

$$\frac{1}{\lambda} \geqslant 1 + \frac{k}{1-2k\varepsilon} \qquad I \geqslant \operatorname{Int}(t_0, t_n) \qquad \varepsilon \leqslant \frac{1}{4k}$$

Then for all $i \in [\![0, n-1]\!]$,

$$\beta_i(1-e^{-1}) \leqslant \operatorname{Int}(t_i, t_{i+1}) \leqslant \beta_i e$$

♦

Proof. Note that the hypothesis on ε is mostly to make sure λ is well defined. Pick $u \in [t_i, t_{i+1}]$ and apply Theorem 2.4.2 (Parameter dependency of PIVP) with $\mu(u) \leqslant \varepsilon_i e^{\gamma_i} \leqslant \varepsilon$ by definition of ε, to get that:

$$\left\|y(u) - \Phi_p(t_i, \tilde{y}_i, u)\right\| \leqslant \varepsilon$$

Furthermore, for any such u, apply Corollary 2.5.2 (Maximum variation for PIVP) to get:

$$\left\|\tilde{y}_{i+1} - \Phi_p(t_i, \tilde{y}_i, u)\right\| \leqslant \frac{\alpha|M(u-t_i)|}{1-|M(u-t_i)|}$$

where $M = k\Sigma p\alpha^{k-1}$ and $\alpha = \max(1, \|\tilde{y}_i\|)$. Putting everything together we get for $\xi = \frac{u-t_i}{\delta t_i}$ using (3.2.3):

$$\|y(u) - \tilde{y}_i\| \leqslant \varepsilon + \frac{\alpha M(u-t_i)}{1-M(u-t_i)} \leqslant \varepsilon + \frac{\alpha\beta_i\xi}{1-\beta_i\xi} \leqslant \varepsilon + \frac{\alpha\lambda\xi}{1-\lambda\xi}$$

Consequently:

$$\|\tilde{y}_i\| - \frac{\alpha\lambda\xi}{1-\lambda\xi} \leqslant \varepsilon + \|y(u)\| \leqslant 2\varepsilon + \|\tilde{y}_i\| + \frac{\alpha\lambda\xi}{1-\lambda\xi}$$

And since $\alpha = \max(1, \|\tilde{y}_i\|)$:

$$\|\tilde{y}_i\| - \frac{\alpha\lambda\xi}{1-\lambda\xi} \leqslant \varepsilon + \|y(u)\| \leqslant 2\varepsilon + \alpha\left(1 + \frac{\lambda\xi}{1-\lambda\xi}\right)$$

And a case analysis brings:

$$\max\left(1, \alpha - \frac{\alpha\lambda\xi}{1-\lambda\xi}\right) \leqslant \max(1, \varepsilon + \|y(u)\|) \leqslant \max\left(1, 2\varepsilon + \alpha + \frac{\alpha\lambda\xi}{1-\lambda\xi}\right)$$

Which can be overapproximated by:

$$\alpha - \frac{\alpha\lambda\xi}{1-\lambda\xi} \leqslant \max(1, \varepsilon + \|y(u)\|) \leqslant 2\varepsilon + \alpha + \frac{\alpha\lambda\xi}{1-\lambda\xi}$$

And finally:

$$\left(\alpha - \frac{\alpha\lambda\xi}{1-\lambda}\right)^{k-1} \leqslant \max(1, \varepsilon + \|y(u)\|)^{k-1} \leqslant \left(2\varepsilon + \alpha + \frac{\alpha\lambda\xi}{1-\lambda}\right)^{k-1}$$

A simple calculation shows that $\int_0^1 (a+bu)^{k-1}du = \frac{(a+b)^k - a^k}{bk}$. Integrating the previous bounds over $[t_i, t_{i+1}]$, we get, for $x = \frac{\lambda}{1-\lambda}$:

$$\int_{t_i}^{t_{i+1}} \left(\alpha - \frac{\alpha\lambda\xi}{1-\lambda}\right)^{k-1} du = \alpha^{k-1}\delta t_i \frac{1-(1-x)^k}{kx}$$

Realising that $x = \frac{1}{\frac{1}{\lambda}-1}$, the hypothesis on λ yields:

$$x \leqslant \frac{1-2k\varepsilon}{k} \leqslant \frac{1}{k} \tag{3.3.8}$$

A simple analysis of the function $x \mapsto \frac{1-(1-x)^k}{kx}$ shows that it is decreasing on $]0, \frac{1}{k}]$ and so satifies, for x in this interval,

$$\frac{1-(1-x)^k}{kx} \geqslant 1 - \left(1 - \frac{1}{k}\right)^k \geqslant 1 - e^{-k\ln(1-\frac{1}{k})} \geqslant 1 - e^{-1}$$

So finally we get:

$$\int_{t_i}^{t_{i+1}} \left(\alpha - \frac{\alpha\lambda\xi}{1-\lambda}\right)^{k-1} du \geqslant \alpha^{k-1}\delta t_i(1-e^{-1}) \tag{3.3.9}$$

On the other side, we get:

$$\int_{t_i}^{t_{i+1}} \left(2\varepsilon + \alpha + \frac{\alpha\lambda\xi}{1-\lambda}\right)^{k-1} du = \alpha^{k-1}\delta t_i \frac{(2\frac{\varepsilon}{\alpha} + 1 + x)^k - (2\frac{\varepsilon}{\alpha}+1)^k}{kx}$$

And since $\alpha \geqslant 1$ and $b^k - a^k \leqslant k(b-a)b^{k-1}$ when $b \geqslant a$, we get:

$$\int_{t_i}^{t_{i+1}} \left(2\varepsilon + \alpha + \frac{\alpha\lambda\xi}{1-\lambda}\right)^{k-1} du \leqslant \alpha^{k-1}\delta t_i(2\varepsilon + 1 + x)^{k-1}$$

We can now use (3.3.8) to get:

$$\int_{t_i}^{t_{i+1}} \left(2\varepsilon + \alpha + \frac{\alpha\lambda\xi}{1-\lambda}\right)^{k-1} du \leqslant \alpha^{k-1}\delta t_i \left(1+\frac{1}{k}\right)^{k-1} \leqslant \alpha^{k-1}\delta t_i e \qquad (3.3.10)$$

We can now put together (3.3.9) and (3.3.10) using that $M = k\Sigma p\alpha^{k-1}$:

$$\delta t_i M(1-e^{-1}) \leqslant \int_{t_i}^{t_{i+1}} k\Sigma p \max(1, \varepsilon + \|y(u)\|)^{k-1} du \leqslant \delta t_i M e$$

which shows the result since $\beta_i = M\delta t_i$. ∎

Algorithm 3.3.11 PIVP Solving algorithm

```
Require: $t_0$ the initial time
Require: $y_0 \in \mathbb{Q}^d$ the initial condition
Require: $p$ polynomial of the PIVP
Require: $t \in \mathbb{Q}$ the time step
Require: $\varepsilon \in \mathbb{Q}$ the precision requested
Require: $I \in \mathbb{Q}$ the integral hint
 1: function SolvePIVPVariable($t_0, y_0, p, t, \varepsilon, I$)
 2:     $k := \max(2, \deg(p))$
 3:     $\varepsilon := \min\left(\varepsilon, \frac{1}{4k}\right)$
 4:     $u := t_0$
 5:     Compute $\tilde{y}_0$ such that $\|\tilde{y}_0 - y_0\| \leqslant \frac{\varepsilon}{3}e^{-I}$
 6:     $\tilde{y} := \tilde{y}_0$
 7:     $i := 0$
 8:     $\lambda := 1 - \frac{k}{1-2k\varepsilon+k}$
 9:     $N := 1 + \frac{I}{\lambda(1-e^{-1})}$
10:     $\eta := \frac{\varepsilon}{I}$
11:     $\mu := \frac{\eta}{3N}$
12:     $\beta := 0$
13:     while $u < t$ do
14:         if $i \geqslant N$ then
15:             return $\perp$                                  ▹ Too many steps !
16:         end if
17:         $\delta := \min\left(t-u, \frac{\lambda}{k\Sigma p \max(1, \|\tilde{y}\|)^{k-1}}\right)$
18:         $\beta := k\Sigma p \max(1, \|\tilde{y}\|)^{k-1}\delta$
19:         $\omega := -\log_2 \frac{\eta}{6N\max(1, \|\tilde{y}\|)}$
20:         $\tilde{y} :=$ ComputeTaylor($p, \tilde{y}, \omega, \mu, \delta$)
21:         $u := u + \delta$
22:         $i := i + 1$
23:         if $I < ((i-1)\lambda + \beta)e$ then
24:             return $\perp$                                  ▹ unsafe result !
25:         end if
26:     end while
27:     return $\tilde{y}$
28: end function
```

Lemma 3.3.12 (Algorithm is correct)**:** *Let $t \in \mathbb{R}$, $I > 0$ and $\varepsilon > 0$, and assume that y satisfies* (2.0.1) *over* $[t_0, t]$. *Let* $x = \text{SolvePIVPVariable}(t_0, y_0, p, t, \varepsilon, I)$, *then*

- *Either $x = \perp$ or $\|x - y(t)\| \leqslant \varepsilon$*
- *Furthermore, if $I \geqslant \dfrac{e}{1-e^{-1}} \operatorname{Int}(t_0, t)$ then $x \neq \perp$*

♦

Proof. It is clear that the algorithm performs exactly as described in the previous section, in the sense that it either returns $\perp$ or $\tilde{y}_n$ for some $n \leqslant N$ that satisfies $t_n = t$ and $N = 1 + \frac{I}{\lambda(1-e^{-1})}$. Now consider the two possible cases.

If $I \geqslant \operatorname{Int}(t_0, t)$ then by Lemma 3.3.5 (Algorithm is conditionally correct), we get that

$$\|y(t) - \tilde{y}_n\| \leqslant \varepsilon$$

so the algorithm is correct if it returns a value instead of $\perp$. Furthermore, by Lemma 3.3.7 (Relationship between n and I):

$$(1-e^{-1}) \sum_{i=0}^{n-1} \beta_i \leqslant \operatorname{Int}(t_0, t_n) \leqslant e \sum_{i=0}^{n-1} \beta_i \tag{3.3.13}$$

And since $\beta_i \leqslant \lambda$ and $\beta_i = \lambda$ for $i < n-1$ then

$$((n-1)\lambda + \beta_{n-1})(1-e^{-1}) \leqslant \operatorname{Int}(t_0, t_n) \leqslant n\lambda e \tag{3.3.14}$$

In the case of $I \geqslant \frac{e}{1-e^{-1}} \operatorname{Int}(t_0, t)$, we further have:

$$I \geqslant \frac{e}{1-e^{-1}} \operatorname{Int}(t_0, t) \geqslant ((n-1)\lambda + \beta_{n-1})e$$

Consequently, the final test of the algoritm will fail since β in the algorithm is exactly β_{n-1}. So the algorithm will return $\tilde{y}_n$ if $I \geqslant \frac{e}{1-e^{-1}} \operatorname{Int}(t_0, t)$.

Now comes the case of $I < \operatorname{Int}(t_0, t)$. This case is more subtle because contrary to the previous case, we cannot directly apply Lemma 3.3.7 (Relationship between n and I) since we miss the hypothesis on I. We can ignore the case where $t_N < t$ because the algorithm returns $\perp$ in this case. Since the function $u \mapsto \operatorname{Int}(t_0, u)$ is continuous on $[t_0, t]$ and is 0 on t_0, we can apply the intermediate value theorem to get:

$$\exists u \in [t_0, t[\text{ such that } I = \operatorname{Int}(t_0, u)$$

Since we eliminated the case where $t_N < t$, we know that $t_n = t$ for some $n \leqslant N$ in the algorithm, so necessarily:

$$\exists i_0 \in [\![0, n-1]\!] \text{ such that } t_{i_0} \leqslant u < t_{i_0+1}$$

As a consequence of $u \mapsto \operatorname{Int}(t_0, u)$ being an increasing function,

$$I \geqslant \operatorname{Int}(t_0, t_{i_0})$$

Now imagine for a moment that we run the algorithm again with final time u instead of t. A close look at the code shows that all variables (which we call t'_i, β'_i, and so on) will be the same for $i \leqslant i_0$ but then $t'_{i_0+1} = u$. So we can apply Lemma 3.3.7 (Relationship between n and I) to this new run of the algorithm on $[t_0, t'_{i_0+1}]$ to get that

$$(1-e^{-1}) \sum_{i=0}^{i_0} \beta'_i \leqslant \operatorname{Int}(t'_0, t'_{i_0+1}) \leqslant e \sum_{i=0}^{i_0} \beta'_i$$

And since $i_0 < n$ then $\beta'_i = \lambda$ for $i < i_0$, and $\beta'_{i_0} < \beta_{i_0}$ because $u = t'_{i_0+1} < t$ so the equation becomes:

$$(1 - e^{-1})i_0\lambda \leqslant \operatorname{Int}(t_0, t'_{i_0}) \leqslant e(i_0\lambda + \beta'_{i_0}) < e(i_0\lambda + \beta_{i_0})$$

Which simplifies to

$$I < e(i_0\lambda + \beta_{i_0})$$

Which leads to

$$I < e((n-1)\lambda + \beta_{n-1})$$

because either $i_0 = n - 1$ and this trivial, or $i_0 < n - 1$ and then use $\beta_{i_0} < \lambda$.

Notice that this result is actually independent of the run the algorithm, we just used a "virtual" run of the algorithm to obtain it. Consequently, in the original algorithm, the final test will suceed and the algorithm will return $\perp$. ■

As we see from this lemma I just needs to be big enough. Otherwise the algorithm will either return a correct value or an error $\perp$. One can reformulate Lemma 3.3.12 (Algorithm is correct) as:

- Whatever the inputs are, we have a bound on the number of steps executed by the algorithm
- If I is greater than a specified value, we know that the algorithm will return a result (and not an error, i.e $\perp$)
- If the algorithm returns a result x, then this value is correct, that is $\|x - y(t)\| \leqslant \varepsilon$.

Notice that the number of steps n of Lemma 3.3.12 (Algorithm is correct) is only the number of time steps $[t_0, t_1], [t_1, t_2], \ldots, [t_{n-1}, t_n]$ used by our method. But inside each subinterval $[t_i, t_{i-1}]$ we still have to compute the solution of (2.0.1) over this subinterval using the Taylor approximation described in Section 2.5 (Taylor series of the solutions). We recall that whether we use bit complexity or algebraic complexity, the complexity of finding this Taylor approximation is polynomial in the order ω of the method, on the initial condition y_0, the precision n (actually the precision is $\mu = 2^{-n}$) and on the description of the polynomial p. Using this result and the previous theorem, we conclude to the Lemma 3.3.17 (Complexity of SolvePIVPVariable) about the computational complexity of solving (2.0.1) over unbounded domains.

In order to bound the complexity, we need to introduce another quantity which is related to Int but actually closer to what we are really interested in: the length of the curve y. We recall that the length of the curve defined by the graph of a function f between $x = a$ and $x = b$ is:

$$\text{length} = \int_a^b \sqrt{1 + (f'(x))^2}dx$$

In the case of the solution of (2.0.1), we note that the derivative of the solution y is given by $p(y)$. Since the degree of p is k, the length of the solution has a value that has an order of magnitude similar to the following quantity.

Definition 3.3.15 (Pseudo-length of a PIVP):

$$\operatorname{Len}(t_0, t) = \int_{t_0}^{t} \Sigma p \max(1, \|y(u)\|)^k du$$

♦

Lemma 3.3.16 (Relationship between Int and Len): *For any $t \geqslant t_0$ in the domain of definition of y and $\varepsilon \leqslant \frac{1}{4k}$,*

$$\operatorname{Int}(t_0, t) \leqslant 2k \operatorname{Len}(t_0, t)$$

♦

Proof.

$$\begin{aligned}
\operatorname{Int}(t_0, t) &= \int_{t_0}^{t} k\Sigma p \max(1, \varepsilon + \|y(u)\|)^{k-1} du \\
&\leqslant k \int_{t_0}^{t} \Sigma p(\varepsilon + \max(1, \|y(u)\|))^{k} du \\
&\leqslant k(1+\varepsilon)^k \int_{t_0}^{t} \Sigma p \max(1, \|y(u)\|)^{k} du \\
&\leqslant k e^{k \log(1+\frac{1}{4k})} \operatorname{Len}(t_0, t) \\
&\leqslant k e^{\frac{1}{4}} \operatorname{Len}(t_0, t)
\end{aligned}$$

■

Lemma 3.3.17 (Complexity of SolvePIVPVariable): *The arithmetic complexity of* SolvePIVPVariable *on input $(t_0, y_0, p, t, \varepsilon, I)$ is bounded by*

$$\operatorname{poly}\left(k^d, I, \log \operatorname{Len}(t_0, t), \log \|y_0\|, -\log \varepsilon\right)$$

The bit-complexity of SolvePIVPVariable$(t_0, y_0, p, t, \varepsilon, I)$ *is bounded by*

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

♦

Proof. It is clear that what makes up most of the complexity of the algorithm are the calls to ComputeTaylor. More precisely, let C be the complexity of the algorithm, then:

$$C_x = O\left(\sum_{i=0}^{n-1} \operatorname{TL}_x(d, p, \tilde{y}_i, \omega_i, \mu_i, \delta t_i)\right)$$

where $x \in \{arith, bit\}$ and $\operatorname{TL}_{arith}$ and TL_{bit} are the arithmetic and bit complexity of computing Taylor series. In Section 2.5 (Taylor series of the solutions) and (2.5.4),(2.5.5) precisely, we explained that one can show that

$$\begin{aligned}
\operatorname{TL}_{arith} &= \tilde{O}\left(\omega \operatorname{deg}(p)^d + (d\omega)^a)\right) \\
\operatorname{TL}_{bit} &= O\left(\operatorname{poly}((\operatorname{deg}(p)\omega)^d, \log \max(1, t)\Sigma p \max(1, \|y_0\|), -\log \mu)\right)
\end{aligned}$$

Recalling that $k = \operatorname{deg}(p)$ and that all time steps are lower than 1, we get

$$\begin{aligned}
\operatorname{TL}_{arith} &= \tilde{O}\left(k^d (d\omega)^a\right) \\
\operatorname{TL}_{bit} &= O\left(\operatorname{poly}((k\omega)^d, \log \Sigma p \max(1, \|y_0\|), -\log \mu)\right)
\end{aligned}$$

Consequently, using (3.3.4),

$$C_{arith} = \tilde{O}\left(\sum_{i=0}^{n-1} k^d d^a \left(\log_2 \frac{6NI \max(1, \|\tilde{y}_i\|)}{\varepsilon}\right)^a\right)$$

$$C_{bit} = O\left(\sum_{i=0}^{n-1} \text{poly}\left(\begin{array}{c} k^d, \left(\log_2 \frac{6NI\max(1,\|\tilde{y}_i\|)}{\varepsilon}\right)^d, \\ \log(\Sigma p \max(1, \|\tilde{y}_i\|)), -\log\frac{\varepsilon}{3NI} \end{array}\right)\right)$$

But we know that

$$\begin{aligned}
\|\tilde{y}_i\| &\leqslant \varepsilon + \|y(t_i)\| \\
&\leqslant \varepsilon + \left\|y_0 + \int_{t_0}^{t_i} p(y(u))du\right\| \\
&\leqslant \varepsilon + \|y_0\| + \int_{t_0}^{t_i} \|p(y(u))\|\, du \\
&\leqslant \varepsilon + \|y_0\| + \int_{t_0}^{t_i} \Sigma p \max(1, \|y(u)\|)^k du \\
&\leqslant \varepsilon + \|y_0\| + \text{Len}(t_0, t_i) \\
\max(1, \|\tilde{y}_i\|) &\leqslant 1 + \|y_0\| + \text{Len}(t_0, t)
\end{aligned}$$

Using that $\varepsilon \leqslant \frac{1}{4k}$, we also have:

$$\begin{aligned}
N &= 1 + \frac{I}{\lambda(1 - e^{-1})} = 1 + \frac{I}{1 - e^{-1}}\left(1 + \frac{k}{1 - 2k\varepsilon}\right) \\
&\leqslant 1 + 2I(1 + 2k)
\end{aligned}$$

Which gives using that $n \leqslant N$ and that $a \leqslant 3$,

$$\begin{aligned}
C_{arith} &= \tilde{O}\left(\sum_{i=0}^{n-1} k^d d^a \left(\log_2 \frac{\text{poly}(I, k, \|y_0\|, \text{Len}(t_0, t))}{\varepsilon}\right)^a\right) \\
&= \text{poly}\left(k^d, I, k, \log\left(\text{poly}\left(I, k, \|y_0\|, \text{Len}(t_0, t), \frac{1}{\varepsilon}\right)\right)\right) \\
&= \text{poly}\left(k^d, I, \log \text{Len}(t_0, t), \log\|y_0\|, -\log\varepsilon\right)
\end{aligned}$$

And similarly:

$$\begin{aligned}
C_{bit} &= O\left(\sum_{i=0}^{n-1} \text{poly}\left(k^d, \left(\log \frac{\text{poly}(I, k, \|y_0\|, \text{Len}(t_0, t))}{\varepsilon}\right)^d, \log\Sigma p\right)\right) \\
&\leqslant \text{poly}\left(k, \log I, \log\|y_0\|, \log\text{Len}(t_0, t), \log\Sigma p, -\log\varepsilon\right)^d
\end{aligned}$$

■

3.4 Enhancement on the adaptive algorithm

The algorithm of the previous section depends on an "hint" I given as input by the user. Certainly this is not a desirable feature, since the algorithm is only guaranteed to terminate (with a correct answer on that case) if

$$I \geqslant \int_{t_0}^{t} k\Sigma p(1 + \varepsilon + \|y(u)\|)^{k-1} du$$

but the user has usually no way of estimating the right-hand side of this inequality (the problem is that it requires some knowledge about the solution y which we are trying to compute).

However we know that if the hint I is large enough, then the algorithm will succeed in returning a result. Furthermore if it succeeds, the result is correct. A very natural way of solving this problem is to repeatedly try for larger values of the hint until the algorithm succeeds. We are guaranteed that this will eventually happen when the hint I reaches the theoretical bound (although it can stop much earlier in many cases). By choosing a very simple update strategy of the hint (double its value on each failure), it is possible to see that this process does not cost significantly more than if we already had the hint.

Algorithm 3.4.1 PIVP Solving algorithm

```
Require: $t_0$ the initial time
Require: $y_0 \in \mathbb{Q}^d$ the initial condition
Require: $p$ polynomial of the PIVP
Require: $t \in \mathbb{Q}$ the time step
Require: $\varepsilon \in \mathbb{Q}$ the precision requested
1: function SolvePIVPEx($t_0, y_0, p, t, \varepsilon$)
2:     $I := 1/2$
3:     repeat
4:         $I := 2I$
5:         $x :=$ SolvePIVPVariable($t_0, y_0, p, t, \varepsilon, I$)
6:     until $x \neq \bot$
7:     return $x$
8: end function
```

Theorem 3.4.2 (Complexity and correctness of SolvePIVPEx): *Let $t \in \mathbb{R}$, $\varepsilon > 0$, and assume that y satisfies* (2.0.1) *over* $[t_0, t]$. *Let*

$$x = \text{SolvePIVPEx}(t_0, y_0, p, t, \varepsilon)$$

Then

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

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

- *the bit complexity of the algorithm is bounded by*

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

♦

Proof. By Lemma 3.3.12 (Algorithm is correct), we know that the algorithm succeeds whenever $I \geqslant \frac{e}{1-e^{-1}}\text{Int}(t_0, t)$. Thus when the I in the algorithm is greater than this bound, the loop must stop. Recall that the value of I at the i^{th} iteration is 2^i (i starts at 0). Let q be the number of iterations of the algorithm: it stops with $I = 2^{q-1}$. Then:

$$2^{q-2} \leqslant \frac{e}{1-e^{-1}}\text{Int}(t_0, t)$$

Indeed, if it wasn't the case, the algorithm would have stop one iteration earlier. Using Lemma 3.3.16 (Relationship between Int and Len) we get:

$$2^{q-1} = O\left(k\,\mathrm{Len}(t_0,t)\right) \qquad q = O\left(\log(k\,\mathrm{Len}(t_0,t))\right)$$

Now apply Lemma 3.3.17 (Complexity of SolvePIVPVariable) to get that the final complexity C is bounded by:

$$\begin{aligned} C_{arith} &= O\left(\sum_{i=0}^{q-1} \mathrm{poly}(k^d, 2^i, \log \mathrm{Len}(t_0,t), \log\|y_0\|, -\log\varepsilon)\right) \\ &= O\left(q\,\mathrm{poly}(k^d, 2^{q-1}, \log \mathrm{Len}(t_0,t), \log\|y_0\|, -\log\varepsilon)\right) \\ &= \mathrm{poly}(k^d, \mathrm{Len}(t_0,t), \log\|y_0\|, -\log\varepsilon) \end{aligned}$$

Similarly:

$$\begin{aligned} C_{bit} &= O\left(\sum_{i=0}^{q-1} \mathrm{poly}(k, 2^i, \log \mathrm{Len}(t_0,t), \log\Sigma p, \log\|y_0\|, -\log\varepsilon)^d\right) \\ &= O\left(q\,\mathrm{poly}(k, 2^{q-1}, \log \mathrm{Len}(t_0,t), \log\|y_0\|, \log\Sigma p, -\log\varepsilon)^d\right) \\ &= \mathrm{poly}(k, \mathrm{Len}(t_0,t), \log\|y_0\|, \log\Sigma p, -\log\varepsilon)^d \end{aligned}$$

■

3.5 Extension to Computable Analysis

In this section, we extend the previous result to deal with non-rational inputs. To this end, we formulate the result in the framework of Computable Analysis.

Definition 3.5.1 (PIVP solving mapping): Let PIVP the partial map defined as follows. For any $d \in \mathbb{N}$, $y_0 \in \mathbb{R}^d$, $p \in \mathbb{R}^d[\mathbb{R}^d]$, $t_0, t \in \mathbb{R}$, $\mathrm{PIVP}(t_0, y_0, p, t) = y(t)$ where $y : [t_0, t] \to \mathbb{R}^d$, if it exists, satisfies $y(t_0) = y_0$ and $y'(u) = p(y(u))$ for all $u \in [t_0, t]$. ♦

Theorem 3.5.2 (PIVP complexity): PIVP *has complexity bounded by*[5]

$$\mathrm{poly}(\deg(p), \mathrm{Len}(t_0,t), \log\|y_0\|, \log\Sigma p, -\log\varepsilon)^d \tag{3.5.3}$$

More precisely, there exists a Turing machine $\mathcal{M}$ such that for any oracle $\mathcal{O}$ for (t_0, y_0, p, t), and $\mu \in \mathbb{N}$, $\left\|\mathcal{M}^{\mathcal{O}}(\mu) - \mathrm{PIVP}(t_0, y_0, p, t)\right\| \leqslant 2^{-\mu}$ if $y(t)$ exists, and the number of steps of the machine is bounded by (3.5.3) *for all such oracles.* ♦

Remark 3.5.4 (Oracle): In Computable Analysis, oracles are usually used to describe a single real number, or a tuple. We extend this notion in the immediate way and say that $\mathcal{O}$ describes a sequence $(r_k)_k \in \mathbb{R}^{\mathbb{N}}$ if $\|\mathcal{O}(k,\mu) - r_k\| \leqslant 2^{-\mu}$ for any $k, \mu \in \mathbb{N}$. Then we encode a polynomial as a sequence containing the degree and the coefficients. ♦

Remark 3.5.5 (Operator complexity): Notice that Theorem 3.4.2 (Complexity and correctness of SolvePIVPEx) provides a full characterization of the complexity of SolvePIVPEx. As a consequence, it could be possible to study and discuss the complexity of the operator $(p, t_0, y_0) \mapsto \phi_p(t_0, y_0, t)$ using the framework of computable analysis (see e.g. [Wei00, KC10]). Indeed, the space of polynomials and computable real numbers admit nice and well-understood representations (see e.g.[KMRZ12]), as does the space of analytic functions. ♦

[5]See Definition 3.3.15 (Pseudo-length of a PIVP) for the expression Len

Proof. The idea of the proof is to compute a rational approximation of all the inputs and use Theorem 3.4.2 (Complexity and correctness of SolvePIVPEx). ■

3.6 Conclusion

In this chapter we have presented a method that allows us to solve a polynomial or elementary ordinary differential equation over an unbounded domain using an arbitrary precision. Moreover our method is guaranteed to produce a result that has a certain precision, where the precision is also provided as an input to our method.

We analyzed the method and established rigorous bounds on the time it needs to output a result. In this manner we were able to bound the computational complexity of solving polynomial or elementary ODEs over unbounded domains.

Chapter 4

Computation with PIVP

These days, even the most pure and abstract mathematics is in danger to be applied.

In this chapter we investigate the possibility of giving a purely continuous (time and space) definition of computability, and in particular of complexity classes. Up until now, it has been open how to define a notion of complexity on dynamical systems such as PIVP that has good properties. One can grasp the challenge by noting that PIVP are subject to the Zeno phenomenon. Another way to formulate this is to say that contrary to discrete models of computation, time itself does not yield to a robust notion of complexity. We will see that neither does space, but that a combination of time and space is sufficient to obtain a robust and well founded notion.

This chapter is organized as follows:

- Section 4.1 (Introduction) will introduce our model of computation informally and explain why it is very natural.
- Section 4.2 (Computing functions) will introduce our most basic notions of computability using either space-time or length as a measure of complexity.
- Section 4.3 (Computing with perturbations) will introduce more advanced notions of computability in the presence of perturbations.
- Section 4.4 (Computing with variable input) will introduce notions of online computability where the input can change over time.
- Section 4.5 (Summary) summarizes our notions of complexity and relationships between the classes.
- Section 4.6 (Closure properties and computable zoo) will show some closure properties of this class, which suggest that this class is a reasonable model of computation. It also gives a list of useful or remarkable computable functions.

4.1 Introduction

In Section 1.4 (General Purpose Analog Computer), we saw the GPAC is an idealization of computers initially used to study differential equations and simulate physical systems, for

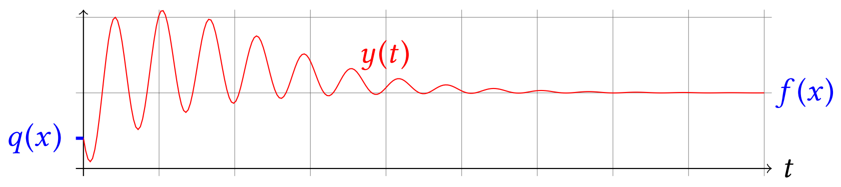


Figure 4.1.2: Graphical representation of a computation on input x

example the ballistic trajectory of a projectile. As such, it is natural to consider the trajectory of the system as the computed object. In Chapter 2 (The PIVP class) we expanded this idea to build the very interesting class of generable functions. However, the GPAC admits a more natural notion of computation. The idea is that the trajectory of the system is merely a trace of the computing process and what matters is the result. In this case the result is the final, stable state of the system. From the point of view of differential equations, this means that we require some variables of the system to converge to some value, which we call the result of the computation. The mapping between the initial state and the stable state is the function computed by the system.

This process is illustrated in Figure 4.1.2 and formalized in Definition 4.1.1 (GPAC approximability) where f is the computed function, x is the input value and $q(x)$ is the initial setup of the system for input x. On this example, y is the trace of the computation and $f(x) = \lim_{t\to\infty} y(t)$.

Definition 4.1.1 (GPAC approximability)**:** $f : \mathbb{R} \to \mathbb{R}$ is called computable if and only if there exists $d \in \mathbb{N}$, polynomials $p \in \mathbb{K}^d[\mathbb{R}^d]$ and $q \in \mathbb{K}^d[\mathbb{R}]$ such that for any $x \in \mathbb{R}$, there exists (a unique) $y : \mathbb{R}_+ \to \mathbb{R}^d$ satisfying for all $t \in \mathbb{R}_+$:

- $y(0) = q(x)$ and $y'(t) = p(y(t))$ ► y satisfies a PIVP
- $|y_1(t) - f(x)| \leqslant e^{-t}$ ► y_1 converges to $f(x)$

♦

It has been known for some time that the notion of GPAC approximability is more general than GPAC generability. For example, Euler's Gamma and Zeta functions are not generable (in the sense of Definition 2.1.1) but are approximable (in the sense of Definition 4.1.1) [Gra04]. More generally, the notion of generable function is too strong because all generable functions are analytic (see Proposition 2.3.3).

In this chapter, we will develop a theory of complexity for this model. A fundamental difficulty one faces when we try to talk about time complexity for continuous time models of computation is that time is a problematic notion for many natural classes of systems. Indeed, many models exhibit the so-called *Zeno's phenomenon*, also called time contraction. This is easily seen in the case of the GPAC because if f is computable then $f \circ \exp$ is also computable, and $f \circ \exp \circ \exp$ and so on. This means that the time t needed for y_1 to reach $f(x)$ with a certain precision is pointless: we can always make it shorter and shorter and even constant.

The time-contraction phenomenon has been well-studied and is generally considered as a sign of unrealistic computation. Indeed, such models usually allow to simulate Turing machines and time contraction can be used to solve problems in constant time, and even the Halting problem is some cases. For more details see [Ruo93], [Ruo94], [Moo96], [Bou97], [Bou99], [AD90], [CP02], [Dav01], [Cop98], [Cop02], [AM98a], [EN02].

In this chapter we give a fundamental contribution to this question: time of a computation, for the GPAC can be measured as the length of the curve (i.e. length of the solution curve of the ordinary differential equation associated to the GPAC), or equivalently, as a couple measuring both time and "space". Doing so, we get to well defined complexity classes, that we will prove, in next chapter, to turn out to be the same as traditional complexity classes. Various attempts at defining a clear complexity theory for continuous time models of computation have previously failed because they considered classes of dynamical systems that where too general, or because they used a notion of time that was not robust. See [GM02] and [SF98] for examples of natural analog complexity classes, and more in the survey [BC08].

Note that the idea of a process converging to some value with controlled error is at the heart of Computable Analysis, where the complexity measure is the one of Turing machines. In Chapter 5 (PIVP versus Turing computability), we will see that there are strong links between PIVP computation and Computable Analysis.

Remark 4.1.3 (Generable field)**:** In this entire chapter, $\mathbb{K}$ will refer to any generable field, for example $\mathbb{R}_G$. See Section 2.7 (Generable fields) for more details. ♦

4.2 Computing functions

In this section, we introduce our notion of computability and provide examples. We will introduce complexity in two different but equivalent ways: using space-time constraints and using length-based constraints.

4.2.I Space-time based complexity

As we saw in Chapter 2 (The PIVP class), generable functions have very nice properties, but those are very limited. For instance, all generable functions are analytic. In order to define a broader class of functions with the GPAC, we relax the requirements of the definition so that the computed function is somehow the "limit" of a generable function. The basic idea is simple enough: the initial condition of the system depends on the input x (with a polynomial relationship), and a set of variables of the system must converge to $f(x)$ where f is the computed function. Furthermore, another variable of the system must give a bound on the error between $f(x)$ and the value computed by the system: this variable must converge to 0. Note that this is, in fact, very similar to the notion of computability for Turing Machines where we have a sequence of configurations and only care about the last configuration, which is signaled by entering a special state.

In order to enhance this definition with an interesting complexity notion, we need two ingredients. First we need to introduce some notion of rate of convergence. But this is not enough by itself because the system can always be rescaled to improve the convergence rate. To prevent this behavior, we also introduce a bound on the maximum value of the components of the system, a similar notion to that of space. Taken separately, none of these notions is admissible, but together they provide a good framework to discuss the complexity of the GPAC. The intuition is that one can "trade" time for space, so we need to take both into account.

Definition 4.2.1 (Analog computability)**:** Let $n, m \in \mathbb{N}$, $f :\subseteq \mathbb{R}^n \to \mathbb{R}^m$ and $\Upsilon, \Omega : \mathbb{R}_+^2 \to \mathbb{R}_+$. We say that f is (Υ, Ω)-computable if and only if there exists $d \in \mathbb{N}$, and $p \in \mathbb{K}^d[\mathbb{R}^d], q \in \mathbb{K}^d[\mathbb{R}^n]$ such that for any $x \in \operatorname{dom} f$, there exists (a unique) $y : \mathbb{R}_+ \to \mathbb{R}^d$ satisfying

- $y(0) = q(x)$ and $y'(t) = p(y(t))$ for all $t \geqslant 0$ ► y satisfies a PIVP
- for all $\mu \in \mathbb{R}_+$, if $t \geqslant \Omega(\|x\|, \mu)$ then $\|y_{1..m}(t) - f(x)\| \leqslant e^{-\mu}$ ► $y_{1..m}$ converges to $f(x)$
- $\|y(t)\| \leqslant \Upsilon(\|x\|, t)$, for all $t \geqslant 0$ ► $y(t)$ is bounded

We denote by $\mathrm{AC}(\Upsilon, \Omega)$ the set of (Υ, Ω)-computable functions. ♦

Definition 4.2.2 (Analog poly-time)**:** We denote by AP the set of (poly, poly)-computable functions. ♦

Remark 4.2.3 (Polynomial versus generable)**:** In this definition, we required for simplicity that p and q be polynomials. It turns out, unsurprisingly, that the class is the same if we only assume that $p, q \in \text{GPVAL}$. This remark also applies to Definition 4.2.7 (Analog weak computability). See Proposition 4.5.2 (Polynomial versus generable) for more details. ♦

In the remaining of this section, we provide several examples of polynomial time computable functions.

Example 4.2.4 (Polynomials are computable)**:** The most trivial example of computable functions are polynomials. Indeed, simply using the versatility in the choice of the initial condition allows us to compute polynomials in constant time. Let $q \in \mathbb{K}[\mathbb{R}^d]$ be a multivariate polynomial: we will show that $q \in \text{AP}$. Let $x \in \mathbb{R}^d$ and consider the following system for $t \in \mathbb{R}_+$:

$$y(0) = q(x) \qquad y'(t) = 0$$

We claim that this system satisfies Definition 4.2.2 (Analog poly-time):

- The system is of the form $y(0) = \text{poly}(x)$ and $y'(t) = \text{poly}(y(t))$ where the polynomials have coefficients in $\mathbb{K}$.
- For any $t \geqslant 0$, $\|y(t) - q(x)\| = 0$ so we can take $\Omega(\alpha, \mu) = 0$.
- For any $t \in \mathbb{R}_+$, $\|y(t)\| = \|q(x)\| \leqslant \text{poly}(\|x\|)$ so we can take Υ to be a any polynomial such that $\Upsilon(\|x\|, \mu) \geqslant \|p(x)\|$.

This shows that $q \in \text{AC}(\Upsilon, \Omega)$. ♦

Example 4.2.5 (Square root is computable)**:** We will show that $\sqrt{\cdot} : [1, \infty[\to \mathbb{R}_+$ belongs to AP. The idea of the construction is the following: we use one variable (y_2) to store the input $x \in \mathbb{R}_+$ and one variable (y_1) to converge to $\sqrt{x}$. We do so by increasing $y_2(t)$ from 0 as long as $y_2(t)^2 \leqslant \sqrt{x}$, the latter being equivalent to $y_2(t) \leqslant x = y_1(t)$ which can be expressed using polynomials. Formally, let $x \in [1, \infty[$ and consider the following system for $t \in \mathbb{R}_+$:

$$\begin{cases} y_1(0) = 0 \\ y_2(0) = x \end{cases} \qquad \begin{cases} y_1'(t) = y_2(t) - y_1(t)^2 \\ y_2'(t) = 0 \end{cases}$$

It can be seen that this system admits a unique solution for $t \in \mathbb{R}_+$ which is:

$$y_1(t) = \sqrt{x}\tanh(\sqrt{x}t) \qquad y_2(t) = x$$

We claim that this system satisfies Definition 4.2.2 (Analog poly-time):

- The system is of the form $y(0) = \text{poly}(x)$ and $y'(t) = \text{poly}(y(t))$ where the polynomials have coefficients in $\mathbb{K}$.
- For any $t \geqslant 0$, apply Lemma 2.6.1 (Bounds on tanh) to get that $\sqrt{x}(1 - e^{-\sqrt{x}t}) \leqslant y_1(t) \leqslant \sqrt{x}$ thus $|y_1(t) - \sqrt{x}| \leqslant \sqrt{x}e^{-\sqrt{x}t} \leqslant \sqrt{x}e^{-t}$ because $x \geqslant 1$. Define $\Omega(\alpha, \mu) = \mu + \alpha$ and check that for all $\mu \in \mathbb{R}_+$, if $t \geqslant \Omega(|x|, \mu)$ then $|y_1(t) - \sqrt{x}| \leqslant e^{-\mu - x + \ln\sqrt{x}} \leqslant e^{-\mu}$.
- Define $\Upsilon(\alpha, t) = \alpha$ and check that for any $t \in \mathbb{R}_+$, $\|y(t)\| = \max(y_1(t), x) = x = \Upsilon(\|x\|, t)$.

This shows that $\sqrt{\cdot}\restriction_{[1,\infty[} \in \text{AC}(\Upsilon, \Omega)$. ♦

Example 4.2.6 (The error function is computable)**:** Let us recall the definition of the error function for $x \in \mathbb{R}$:

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

We will show that erf $\in$ AP. The idea of the proof is to note that e^{-t^2} is a generable function, so it can be generable using a PIVP. Consequently, erf is also a generable function so we can be build a system such that $y(t) = \operatorname{erf}(t)$, but in order to compute $\operatorname{erf}(x)$ we need to stop this system at time $t = x$. We do so by rescaling the generable system, so that in essence, $y(t) = \operatorname{erf}(x(1-e^{-t}))$. This idea, is in fact, not specific to erf and we will applied later to show that most generable functions are computable. Formally, let $x \in \mathbb{R}$ and consider the following system for $t \in \mathbb{R}_+$:

$$\begin{cases} y_1(0)= 0 \\ y_2(0)= 1 \\ y_3(0)= x \\ y_4(0)= x \end{cases} \qquad \begin{cases} y_1'(t)= \frac{2}{\sqrt{\pi}} y_3(t) y_2(t) \\ y_2'(t)= -2y_3(t)(y_4(t)-y_3(t))y_2(t) \\ y_3'(t)= -y_3(t) \\ y_4'(t)= 0 \end{cases}$$

It can be seen that this system admits a unique solution for $t \in \mathbb{R}_+$ which is:

$$y_1(t) = \operatorname{erf}(x(1-e^{-t}) \qquad y_2(t) = e^{-x^2(1-e^{-t})^2} \qquad y_3(t) = xe^{-t} \qquad y_4(t) = x$$

We claim that this system satisfies Definition 4.2.2 (Analog poly-time):

- The system is of the form $y(0) = \operatorname{poly}(x)$ and $y'(t) = \operatorname{poly}(y(t))$ where the polynomials have coefficients in $\mathbb{K}$, since $\sqrt{\pi} \in \mathbb{K}$ because $\pi \in \mathbb{K}$, $\mathbb{K}$ is a generable field and $\sqrt{\cdot}\restriction_{[1,\infty[} \in$ GPVAL.
- For any $t \in \mathbb{R}_+$, $|\operatorname{erf}(x) - y_1(t)| = |\operatorname{erf}(x) - \operatorname{erf}(x(1-e^{-t}))| = \frac{2}{\sqrt{\pi}} \left|\int_{x(1-e^{-t})}^x e^{-u^2} du\right| \leqslant \frac{2}{\sqrt{\pi}} \left|\int_{x(1-e^{-t})}^x du\right| \leqslant \frac{2|x|e^{-t}}{\sqrt{\pi}}$. Define $\Theta(\alpha,\mu) = \mu + \alpha + \ln\frac{2}{\sqrt{\pi}}$, and let $\mu \in \mathbb{R}_+$, if $t \geqslant \Theta(\|x\|,\mu)$ then $|\operatorname{erf}(x) - y_1(t)| \leqslant |x|e^{-\mu-|x|} \leqslant e^{-\mu}$.
- For any $t \in \mathbb{R}_+$, $|y_1(t)| \leqslant 1$, $|y_2(t)| \leqslant 1$, $|y_3(t)| \leqslant |x|$ and $|y_4(t)| \leqslant |x|$. Define $\Upsilon(\alpha,t) = 1+\alpha$ and then $\|y(t)\| \leqslant \Upsilon(\|x\|,t)$.

This shows that erf $\in \mathrm{AC}(\Upsilon,\Omega)$. ♦

More examples of computable functions can be found in Section 4.6 (Closure properties and computable zoo), including some non-differentiable functions like the absolute value.

More often that not, especially when proving that a function is computable, it will be useful to prove a weaker property where the precision of the output is given as a parameter of the system. More precisely, given an input x and a precision μ, the system should compute $f(x)$ with error at most $e^{-\mu}$. This is in contrast with the previous notion where given an input x, the precision of the output keeps increasing over time.

Definition 4.2.7 (Analog weak computability)**:** Let $n, m \in \mathbb{N}$, $f :\subseteq \mathbb{R}^n \to \mathbb{R}^m$, $\Omega : \mathbb{R}_+^2 \to \mathbb{R}_+$ and $\Upsilon : \mathbb{R}_+^3 \to \mathbb{R}_+$. We say that f is (Υ,Ω)-weakly-computable if and only if there exists $d \in \mathbb{N}$, $p \in \mathbb{K}^d[\mathbb{R}^d], q \in \mathbb{K}^d[\mathbb{R}^{n+1}]$ such that for any $x \in \operatorname{dom} f$ and $\mu \in \mathbb{R}_+$, there exists (a unique) $y : \mathbb{R}_+ \to \mathbb{R}^d$ satisfying for all $t \in \mathbb{R}_+$:

- $y(0) = q(x,\mu)$ and $y'(t) = p(y(t))$ ► y satisfies a PIVP

- if $t \geqslant \Omega(\|x\|, \mu)$ then $\|y_{1..m}(t) - f(x)\| \leqslant e^{-\mu}$ ► $y_{1..m}$ converges to $f(x)$
- $\|y(t)\| \leqslant \Upsilon(\|x\|, \mu, t)$ ► $y(t)$ is bounded

We denote by $\mathrm{AW}(\Upsilon, \Omega)$ the set of (Υ, Ω)-weakly-computable functions. ♦

Definition 4.2.8 (Analog weak poly-time)**:** Denote by AWP the set of (poly, poly)-weakly-computable functions. ♦

Remark 4.2.9 (Limit computability)**:** A careful look at the previous definition shows that analog weak computability is a form of limit computability. Formally, let $f : I \times \mathbb{R}_+^* \to \mathbb{R}^n$, $g : I \to \mathbb{R}^n$ and $\mho : \mathbb{R}_+^2 \to \mathbb{R}_+$ a polynomial. Assume that $f \in \mathrm{AP}$ and that for any $x \in I$ and $\tau \in \mathbb{R}_+^*$, if $\tau \geqslant \mho(\|x\|, \mu)$ then $\|f(x, \tau) - f(x)\| \leqslant e^{-\mu}$. Then $g \in \mathrm{AWP}$ because the analog system for f satisfies all the items of the definition. ♦

It is clear, by definition that any computable function is weakly-computable by the exact same system. This is summarized by the following proposition in the case of poly-time computability.

Proposition 4.2.10 (Computable ⊆ weak)**:** AP ⊆ AWP ♦

4.2.II Length-based complexity

In this section, we introduce an alternative notion of complexity based on the length of curve instead of the time/space pair. We saw in Chapter 3 (Solving PIVP) that the complexity of solving a PIVP is related to a quantity that looks like the length of the curve. Intuitively, the length captures both space and time which explains why it is a robust notion of complexity.

Definition 4.2.11 (Length of a curve)**:** Let $d \in \mathbb{N}$, I be an interval and $y \in C^1(I, \mathbb{R}^n)$, the *length* of y over I is defined by:

$$\mathrm{len}_y(a, b) = \int_I \|y'(t)\| \, dt$$

♦

Definition 4.2.12 (Analog length computability)**:** Let $n, m \in \mathbb{N}$, $f :\subseteq \mathbb{R}^n \to \mathbb{R}^m$ and $\Omega : \mathbb{R}_+^2 \to \mathbb{R}_+$. We say that f is Ω-length-computable if and only if there exists $d \in \mathbb{N}$, and $p \in \mathbb{K}^d[\mathbb{R}^d], q \in \mathbb{K}^d[\mathbb{R}^n]$ such that for any $x \in \operatorname{dom} f$, there exists (a unique) $y : \mathbb{R}_+ \to \mathbb{R}^d$ satisfying for all $t \in \mathbb{R}_+$:

- $y(0) = q(x)$ and $y'(t) = p(y(t))$ for all $t \geqslant 0$ ► y satisfies a PIVP
- for any $\mu \in \mathbb{R}_+$, if $\mathrm{len}_y(0, t) \geqslant \Omega(\|x\|, \mu)$ then $\|y_{1..m}(t) - f(x)\| \leqslant e^{-\mu}$ ► $y_{1..m}$ converges
- $\|y'(t)\| \geqslant 1$ ► technical condition

We denote by $\mathrm{AL}(\Omega)$ the set of Ω-length-computable functions. ♦

Remark 4.2.13 (Technical condition)**:** In order for the equivalence between AP and ALP to work, we need a technical condition to ensure that the length of the curve grows at least linearly with time. We could request a weaker condition, such as $\|p(y(t))\| \geqslant \frac{1}{\mathrm{poly}(t)}$ but it is unclear if the equivalence holds without such condition. Indeed, if the speed of the system becomes extremely small, it might take an exponential time to reach a polynomial length. See Example 4.2.14 (Technical condition) for such an example. ♦

Example 4.2.14 (Technical condition)**:** Define $y(t) = e^{-\ln(1+t)}$, then y satisfies a PIVP because $y(0) = 1$ and $y'(t) = z(t)y(t)$ where $z(t) = \frac{1}{1+t}$ and z satisfies $z(0) = 1$ and $z'(t) = -z(t)^2$. A simple computation shows that the length condition is violated because $\operatorname{len}_{y,z}(0,t) = \int_0^t \frac{1}{(1+u)^2}du = 1 - \frac{1}{1+t}$. Furthermore, y is clearly computing 0 but extremely slowly because one has to take $t \geqslant e^\mu - 1$ so that $\|y(t) - 0\| \leqslant e^{-\mu}$. Unfortunately, in this case, the condition on the length of the curve makes no sense because the length is bounded ! ♦

Theorem 4.2.15 (Computable = length-computable)**:** AP = ALP ♦

Proof. In one direction the proof is simple because if the system uses polynomial time and space then there is a relationship between time and length and we only need to add one variable to the system to make sure that the technical condition holds. The other direction is more involved because we need to rescale the system using the length of the curve to make sure it does not grow faster than a polynomial, this is ensured by the technical condition.

Let $f \in \mathrm{AC}(\Upsilon, \Omega)$ where Υ and Ω are polynomials, which we assume to be increasing functions. Apply Definition 4.2.1 (Analog computability) to get d, p, q, let $k = \deg(p)$ and define:

$$\Omega^*(\alpha,\mu) = \Omega(\alpha,\mu)\left(1 + \Sigma p \max\left(1, \Upsilon(\|x\|, \Omega(\alpha,\mu))\right)^k\right)$$

Let $x \in \operatorname{dom} f$ and consider the following system:

$$\begin{cases} y(0) = q(x) \\ z(0) = 0 \end{cases} \qquad \begin{cases} y'(t) = p(y(t)) \\ z'(t) = 1 \end{cases}$$

Note that $z(t) = t$, this variable is there only to ensure that the length of z grows at least linear. Let $t, \mu \in \mathbb{R}_+$ and assume that $\operatorname{len}_z(0,t) \geqslant \Omega^*(\|x\|, \mu)$. We will show that $t \geqslant \Omega(\|x\|, \mu)$ by contradiction. Assume the contrary and let $u \in [0, t]$. By definition $\|y(u), z(u)\| \leqslant 1 + \|y(u)\| \leqslant 1 + \Upsilon(\|x\|, t) < 1 + \Upsilon(\|x\|, \Omega(\|x\|, \mu))$ and thus $\|y'(u), z'(u)\| = \|1, p(y(u))\| < 1 + \Sigma p\big(1 + \Upsilon(\|x\|, \Omega(\|x\|, \mu)\big)^k$. Consequently, $\operatorname{len}_{y,z}(0,t) < t \sup_{u\in[0,t]} \|y'(u), z'(u)\| \leqslant \Omega^*(\|x\|, \mu)$ which is absurd. Since $t \geqslant \Omega(\|x\|, \mu)$, by definition we get that $\|y_{1..m}(t) - f(x)\| \leqslant e^{-\mu}$. Finally, $\|y'(t), z'(t)\| \geqslant \|z'(u)\| \geqslant 1$ for all $t \in \mathbb{R}_+$. This shows that that $f \in \mathrm{AL}(\Omega^*)$ where Ω^* is a polynomial.

Let $f \in \mathrm{AL}(\Omega)$ where Ω is a polynomial, which we assume to be an increasing function. Apply Definition 4.2.12 (Analog length computability) to get d, p, q, let $k = \deg(p)$. Apply Lemma 2.6.18 (Norm function) to get that $g(x) = \operatorname{norm}_{\infty,1}(p(x))$ belongs to GPVAL. Apply Definition 2.1.17 (Generable function) to get m, r, x_0 and z_0. Let $x \in \operatorname{dom} f$. For the analysis, it will useful to consider the following systems:

$$\begin{cases} y(0) = q(x) \\ z(x_0) = z_0 \end{cases} \qquad \begin{cases} y'(t) = p(y(t)) \\ J_z(x) = r(z(x)) \end{cases}$$

Note that by definition $z_1(x) = g(x)$. Define $\psi(t) = g(y(t))$ and $\hat{\psi}(u) = \int_0^u \psi(t)dt$. Now define the following system:

$$\begin{cases} \hat{y}(0) = q(x) \\ \hat{z}(0) = z(q(x)) \\ \hat{w}(0) = \frac{1}{g(q(x))} \end{cases} \qquad \begin{cases} \hat{y}'(u) = \hat{w}(u)p(\hat{y}(u)) \\ \hat{z}'(u) = \hat{w}(u)r(\hat{z}(u))p(\hat{y}(u)) \\ \hat{w}'(u) = -\hat{w}(u)^3 r_1(\hat{z}(u))p(\hat{y}(u)) \end{cases}$$

where by r_1 we mean the first line of r. We will check that $\hat{y}(u) = y(\hat{\psi}^{-1}(u))$, $\hat{z}(u) = z(\hat{y}(u))$ and $\hat{w}(u) = (\hat{\psi}^{-1})'(u)$. We will use the fact that for any $h \in C^1$, $(g^{-1})' = \frac{1}{g' \circ g^{-1}}$. Also note that $\hat{\psi}' = \psi$.

- $\hat{y}(0) = y(\hat{\psi}^{-1}(0)) = y(0) = q(x)$
- $\hat{y}'(u) = (\hat{\psi}^{-1})'(u)y'(\hat{\psi}^{-1}(u)) = \hat{w}(u)p(y(\hat{\psi}^{-1}(u))) = \hat{w}(u)p(\hat{y}(u))$
- $\hat{z}(0) = z(\hat{y}(0)) = z(q(x))$
- $\hat{z}'(u) = J_z(\hat{y}(u))\hat{y}'(u) = \hat{w}(u)r(z(\hat{y}(u)))p(\hat{y}(u)) = \hat{w}(u)r(\hat{z}(u))p(\hat{y}(u))$
- $\hat{w}(0) = \frac{1}{\hat{\psi}'(\hat{\psi}^{-1}(0))} = \frac{1}{\psi(0)} = \frac{1}{g(q(x))}$
- $\hat{w}'(u) = \frac{-(\hat{\psi}^{-1})'(u)\hat{\psi}''(\hat{\psi}^{-1}(u))}{(\hat{\psi}'(\hat{\psi}^{-1}(u)))^2} = -\hat{w}(u)^3\psi'(\hat{\psi}^{-1}(u)) = \nabla g(y(\hat{\psi}^{-1}(u))) \cdot y'(\hat{\psi}^{-1})$ and since $\nabla g(x) = r_1(z(x))^T$ (transpose of the first line of the jaocibian matrix of z because $g = z_1$) then $\hat{w}'(u) = -\hat{w}(u)^3 r_1(z(y(\hat{\psi}^{-1}(u))))^T \cdot p(y(\hat{\psi}^{-1}(u))) = -\hat{w}(u)^3 r_1(\hat{z}(u))p(\hat{y}(u))$

We now claim that this system computes f quickly and has polynomial space. First note that by Lemma 2.6.18 (Norm function), $\|y'(t)\| \leqslant g(y(t)) \leqslant \|y'(t)\| + 1$ thus $\operatorname{len}_y(0,t) \leqslant \hat{\psi}(t) \leqslant \operatorname{len}_y(0,t) + t$. Thus $\operatorname{len}_{\hat{y}}(0,u) = \int_0^u \|\hat{y}'(\xi)\| d\xi = \int_0^{\hat{\psi}^{-1}(u)} \left\|\hat{w}(\hat{\psi}(t))p(\hat{y}(\hat{\psi}(t)))\right\| \hat{\psi}'(t)dt = \int_0^{\hat{\psi}^{-1}(u)} \left\|(\hat{\psi}^{-1})'(\hat{\psi}(t))\hat{\psi}'(t)p(y(t))\right\| dt = \int_0^{\hat{\psi}^{-1}(u)} \|p(y(t))\| dt = \operatorname{len}_y(0, \hat{\psi}^{-1}(u)) \leqslant \hat{\psi}(\hat{\psi}^{-1}(u)) \leqslant u$. It follows that $\|\hat{y}(u)\| \leqslant \|\hat{y}(0)\| + u \leqslant \|q(x)\| + u \leqslant \operatorname{poly}(\|x\|, u)$. Similarly, $\|\hat{z}(u)\| = \|z(\hat{y}(u))\| \leqslant \operatorname{poly}(\|x\|, u)$ because $z \in$ GPVAL and thus is polynomially bounded. Finally, $\|\hat{w}\| = \frac{1}{\psi(\hat{\psi}^{-1}(u)} = \frac{1}{g(\hat{y}(u))} \leqslant \frac{1}{\left\|y'(\hat{\psi}^{-1}(u))\right\|} \leqslant 1$ because by hypothesis, $\|y'(t)\| \geqslant 1$ for all $t \in \mathbb{R}_+$. This shows that indeed $\|(\hat{y}, \hat{z}, \hat{w})(u)\|$ is polynomially bounded in $\|x\|$ and u. Now let $\mu \in \mathbb{R}_+$ and $t \geqslant 1 + \Omega(\|x\|, \mu)$ then $\operatorname{len}_{\hat{y}}(0,t) = \operatorname{len}_y(0, \hat{\psi}^{-1}(t)) \geqslant \hat{\psi}(\hat{\psi}^{-1}(t)) - \hat{\psi}^{-1}(t) \geqslant t - \hat{\psi}^{-1}(t) \geqslant 1 + \Omega(\|x\|, \mu) - \frac{1}{\psi(\hat{\psi}^{-1}(t))} \geqslant \Omega(\|x\|, \mu)$ because, as we already saw, $\left\|\psi(\hat{\psi}^{-1}(t))\right\| \geqslant 1$. Thus by definition, $\|\hat{y}_{1..m}(t) - f(x)\| \leqslant e^{-\mu}$ because $\hat{y}(t) = y(\hat{\psi}^{-1}(t))$. This shows that $f \in$ AP.

■

4.3 Computing with perturbations

We introduced the notion of (weak) computability and saw that this notion is very natural and quite elegantly formulated. However, the examples of the previous section reveal that it is very tedious to prove that a function is computable. Furthermore, the differential systems computing such functions are very much like high-precision clocks: they work flawlessly but the slightest change will break them. In this section, we introduce two new notions of computation that are robust to pertubations. We will see that, quite surprisingly, all computable systems can be made robust in this sense.

4.3.I Robust computability

The notion of robust computability builds up on the notion of weak computability but this time the system should tolerate errors up to $e^{-\Theta(\|x\|,\mu)}$. The function Θ is a parameter of the class, like Ω and Υ, so the system can indicate how much error it can handle. On the other hand, we relax a little bit the constraint on the system and allow the right-hand side to be generable instead of polynomial.

Definition 4.3.1 (Analog robust computability)**:** Let $n, m \in \mathbb{N}$, $f :\subseteq \mathbb{R}^n \to \mathbb{R}^m$, $\Theta, \Omega : \mathbb{R}_+^2 \to \mathbb{R}_+$ and $\Upsilon : \mathbb{R}_+^3 \to \mathbb{R}_+$. We say that f is $(\Upsilon, \Omega, \Theta)$-robustly-computable if and only if there exists $d \in \mathbb{N}$, and $(h : \mathbb{R}^d \to \mathbb{R}^d)$, $(g : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}^d) \in$ GPVAL such that for any $x \in \operatorname{dom} f$, $\mu \in \mathbb{R}_+$,

$e_0 \in \mathbb{R}^d$ and $e \in C^0(\mathbb{R}_+, \mathbb{R}^d)$ satisfying $\|e_0\| + \int_0^\infty \|e(t)\| \, dt \leqslant e^{-\Theta(\|x\|,\mu)}$, there exists (a unique) $y : \mathbb{R}_+ \to \mathbb{R}^d$ satisfying for all $t \in \mathbb{R}_+$:

- $y(0) = g(x, \mu) + e_0$ and $y'(t) = h(y(t)) + e(t)$ ► y satisfies a generable IVP
- if $t \geqslant \Omega(\|x\|, \mu)$ then $\|y_{1..m}(t) - f(x)\| \leqslant e^{-\mu}$ ► $y_{1..m}$ converges to $f(x)$
- $\|y(t)\| \leqslant \Upsilon(\|x\|, \mu, t)$ ► $y(t)$ is bounded

We denote by $\mathrm{AR}(\Upsilon, \Omega, \Theta)$ the set of $(\Upsilon, \Omega, \Theta)$-robustly-computable functions. ♦

Definition 4.3.2 (Analog robust poly-time)**:** Denote by ARP the set of $(\mathrm{poly}, \mathrm{poly}, \mathrm{poly})$-robustly-computable functions. ♦

Remark 4.3.3 (Domain of definition of g and h)**:** There is a subtle but important detail in this definition: we more or less replaced the polynomials p and q by generable functions g and h. It could have been temping to take this opportunity to restrict the domain of definition of g to $\mathrm{dom}\, f \times \mathbb{R}_+$ and that of h to a subset of $\mathbb{R}^d$ where the dynamics takes place. We kept the entire euclidian space for good reasons. First it makes the definition simpler. Second, it makes the notion stronger and more useful. This last point is important because we are going to use robust computability (and the next notion of strong computability) in cases where we have less or no control over the errors and thus over the trajectory of the system. On the downside, this requires to check that g and h are indeed defined over the entire space ! ♦

The examples below show how to build robustly-computable functions. In the first example, we only need to define Θ so that it works, whereas in the second case, careful design of the system is needed for it to be robust.

Example 4.3.4 (Polynomials are robustly-computable)**:** In order to make polynomials robustly-computable, we will play with the choice of Θ and see that it is enough to make the system robust. Let $q \in \mathbb{K}[\mathbb{R}^d]$ be a multivariate polynomial: we will show that $q \in \mathrm{ARP}$. Let $x \in \mathbb{R}^d$, $\mu \in \mathbb{R}_+$, $e_0 \in \mathbb{R}$ and $e \in C^0(\mathbb{R}_+, \mathbb{R})$. Assume that $|e_0| + \int_0^\infty |e(t)| dt \leqslant e^{-\mu}$ and consider the following system for $t \in \mathbb{R}_+$:

$$y(0) = q(x) + e_0 \qquad y'(t) = e(t)$$

We claim that this system satisfies Definition 4.3.2 (Analog robust poly-time):

- The system is of the form $y(0) = \mathrm{poly}(x) + e_0$ and $y'(t) = \mathrm{poly}(y(t)) + e(t)$ where the polynomials have coefficients in $\mathbb{K}$.
- For any $t \geqslant 0$, $\|y(t) - q(x)\| \leqslant |e_0| + \int_0^t |e(u)| du \leqslant |e_0| + \int_0^\infty |e(u)| du \leqslant e^{-\mu}$ so we can take $\Omega(\alpha, \mu) = 0$.
- For any $t \in \mathbb{R}_+$, $\|y(t)\| \leqslant \|q(x)\| + |e_0| + \int_0^t |e(u)| du \leqslant \|q(x)\| + 1 \leqslant \mathrm{poly}(\|x\|)$ so we can take Υ to be a any polynomial such that $\Upsilon(\|x\|, \mu) \geqslant \|p(x)\| + 1$.

This shows that $q \in \mathrm{AR}(\Upsilon, \Omega, \Theta)$ where $\Theta(\alpha, \mu) = \mu$. ♦

In the previous example, we saw that we could modify the system of some computable functions to make them robustly-computable. It appears that this is not a coincidence but a general fact. To understand how the proof works, one must first understand the problem. Let us consider a computable function $f :\subseteq \mathbb{R}^d \to \mathbb{R}$ in $\mathrm{AW}(\Upsilon, \Theta)$ and the associated system for $x \in \mathrm{dom}\, f$ and $\mu \in \mathbb{R}_+$:

$$y(0) = q(x, \mu) \qquad y'(t) = p(y(t))$$

This system converges to $f(x)$ very quickly: $\|y_1(t)-f(x)\| \leqslant e^{-\mu}$ when $t \geqslant \Omega(\|x\|,\mu)$ and $y(t)$ is bounded: $\|y(t)\| \leqslant \Upsilon(\|x\|,\mu,t)$. Let us introduce some errors in the system by taking $e_0 \in \mathbb{R}^d$ and $e \in C^0(\mathbb{R}_+,\mathbb{R}^d)$ such that $\|e_0\| + \int_0^\infty \|e(u)\|\,du \leqslant e^{-\Theta(\|x\|,\mu)}$ for some unspecified Θ and consider the perturbed system:

$$z(0) = q(x,\mu) + e_0 \qquad z'(t) = p(z(t)) + e(t)$$

The relationship between this system and the previous one is given by Theorem 2.4.2 (Parameter dependency of PIVP) and can be informally written as:

$$\begin{aligned}
\|z(t)-y(t)\| &\leqslant \left(\|e_0\| + \int_0^t \|e(u)\|\,du\right) e^{\int_0^t k\Sigma p\|y(u)\|^{k-1}du} && (4.3.5)\\
&\leqslant \left(\|e_0\| + \int_0^\infty \|e(u)\|\,du\right) e^{\int_0^t k\Sigma p\Upsilon(\|x\|,\mu,u)^{k-1}du} && \text{using the bound of } y(t)\\
&\leqslant e^{k\Sigma pt\Upsilon(\|x\|,\mu,t)^{k-1}-\Theta(\|x\|,\mu)} && \text{assuming that } \Upsilon \text{ is increasing}
\end{aligned}$$

One observes that this bound grows to infinity whatever we can choose for Θ because of the dependency in t. On the other hand, we do not need to simulate y for arbitrary large t: as soon as $t \geqslant \Theta(\|x\|,\mu)$ we can stop the system and get a good enough result. Unfortunately, one does not simply stop a differential system, however we can slow it down – like we did in Example 4.2.6 (The error function is computable). To this end, introduce $\psi(t) = (1+\Theta(\|x\|,\mu))\tanh(t)$ and $w(t) = z(\psi(t))$. If we show that w satisfies a differential system, then we are almost done. Indeed $\psi(t) \leqslant 1+\Theta(\|x\|,\mu)$ for all $t \in \mathbb{R}_+$ and if $t \geqslant 1+\Theta(\|x\|,\mu)$ then $\psi(t) \geqslant \Theta(\|x\|,\mu)$, so the system "kind of stop" between $\Theta(\|x\|,\mu)$ and $\Theta(\|x\|,\mu)+1$. Futhermore, if $t \geqslant 1+\Theta(\|x\|,\mu)$ then:

$$\begin{aligned}
\|w_1(t)-f(x)\| &\leqslant \|z(\psi(t))-y(\psi(t))\| + \|y_1(\psi(t))-f(x)\| && \text{use the triangle inequality}\\
&\leqslant e^{k\Sigma p\psi(t)\Upsilon(\|x\|,\mu,\psi(t))^{k-1}-\Theta(\|x\|,\mu)} + e^{-\mu} && \text{using (4.3.5)}\\
&\leqslant e^{k\Sigma p(1+\Theta(\|x\|,\mu))\Upsilon(\|x\|,\mu,1+\Theta(\|x\|,\mu))^{k-1}-\Theta(\|x\|,\mu)} + e^{-\mu} && \text{using the bound on } \psi\\
&\leqslant 2e^{-\mu} && \text{for a suitable choice of } \Theta
\end{aligned}$$

We are left with showing that $w(t) = z(\psi(t))$ can be be generated by a generable IVP with perturbations. In the case of no pertubations, this is very easy because $w'(t) = \psi'(t)z'(t) = x(1-\tanh(t))p(z(t))$ which is generable. The following lemma extends this idea in the case of perturbations.

Lemma 4.3.6 (PIVP Slow-Stop): *Let $d \in \mathbb{N}$, $y_0 \in \mathbb{R}^d$, $T,\theta \in \mathbb{R}_+$, $(e_{0,y},e_{0,A}) \in \mathbb{R}^{d+1}$, $(e_y,e_A) \in C^0(\mathbb{R}_+,\mathbb{R}^{d+1})$ and $p \in \mathbb{K}^d[\mathbb{R}^d]$. Assume that $\|e_0\| + \int_0^\infty \|e(t)\|\,dt \leqslant e^{-\theta}$ and consider the following system:*

$$\begin{cases} y(0) = y_0 + e_{0,y} \\ A(0) = T + 2 + e_{0,A} \end{cases} \qquad \begin{cases} y'(t) = \frac{1+\tanh(A(t))}{2}p(y(t)) + e_y(t) \\ A'(t) = -1 + e_A(t) \end{cases}$$

Then there exists an increasing function $\psi \in C^0(\mathbb{R}_+,\mathbb{R}_+)$ and $z : \psi(\mathbb{R}_+) \to \mathbb{R}^d$ such that:

$$\psi(0) = 0 \qquad z(0) = y_0 + e_{0,y} \qquad z'(t) = p(z(t)) + (\psi^{-1})'(t)e_y(\psi^{-1}(t))$$

and $y(t) = z(\psi(t))$. Furthermore $\psi(T+1) \geqslant T$ and $\psi(t) \leqslant T+4$ for all $t \in \mathbb{R}_+$. Furthermore, $|A(t)| \leqslant T+3$ for all $t \in \mathbb{R}_+$. ♦

Proof. Let $f(t) = \frac{1+\tanh(A(t))}{2}$ and note that $0 < f(t) < 1$ for all $t \in \mathbb{R}_+$. Check that we can integrate A explicitly: $A(t) = T + 2 - t + e_{0,A} + \int_0^t e_A(u)du$. Define $\psi(t) = \int_0^t f(u)du$ then ψ

is an increasing function because $f > 0$, so it is a diffeomorphism from $\mathbb{R}_+$ onto $\psi(\mathbb{R}_+)$. Note that $\psi(t) \leqslant t$ for all $t \in \mathbb{R}_+$. Let $t \geqslant T+3$, then $A(t) \leqslant T+2-t+|e_{0,A}| + \int_0^t |e_A(u)|du \leqslant T+2+e^{-\theta}-t \leqslant T+3-t \leqslant 0$ because $\theta \geqslant 0$. Apply Lemma 2.6.1 (Bounds on tanh) to get that $\tanh(A(t)) \leqslant -1+e^{T+3-t}$ and thus $f(t) \leqslant \frac{1}{2}e^{T+3-t}$ for $t \geqslant T+3$. Integrating this inequality shows that $\psi(t) \leqslant \psi(T+3) + \frac{1}{2}\int_{T+3}^t e^{T+3-u}du \leqslant T+3+\frac{1}{2}(1-e^{T+3-t}) \leqslant T+4$. This shows that $\psi(t) \leqslant T+4$ for all $t \in \mathbb{R}_+$.

Let $t \leqslant T+1$, then by the same reasoning, $A(t) \geqslant T+2-t-e^{-\theta} \geqslant T+1-t \geqslant 0$ thus $\tanh(A(t)) \geqslant 1-e^{t-T-1}$ and $f(t) \geqslant 1-\frac{1}{2}e^{t-T-1}$. Thus $\psi(T+1) \geqslant \int_0^{T+1} 1+\frac{1}{2}e^{u-T-1}du = T+1+\frac{1}{2}(1-e^{-1-T}) \geqslant T$.

Finally, apply Lemma 2.4.4 (Perturbed time-scaling) to get that $y(t) = z(\psi(t))$ where z satisfies for $t \in \psi(\mathbb{R}_+)$:

$$z(0) = y(0) \qquad z'(t) = p(z(t)) + (\psi^{-1})'(t)e_y(\psi^{-1}(t))$$

■

Theorem 4.3.7 (Weak ⊆ robust)**:** AWP ⊆ ARP. ♦

Proof. Let Υ^*, Ω^* be polynomials such that $f \in \mathrm{AW}(\Upsilon^*, \Omega^*)$. Without loss of generality, we assume they are increasing functions of both arguments. Apply Definition 4.2.7 (Analog weak computability) to get $d \in \mathbb{N}$, $p \in \mathbb{K}^d[\mathbb{R}^d]$, $q \in \mathbb{K}^d[\mathbb{R}^{n+1}]$ and let $k = \deg(p)$. Define:

$$\begin{aligned}
T(\alpha,\mu) &= \Omega^*(\alpha, \mu + \ln 2)\\
\Theta(\alpha,\mu) &= k\Sigma p(T(\alpha+1,\mu)+4)(\Upsilon^*(\alpha,\mu,T(\alpha+1,\mu)+4)+1)^{k-1} + \mu + \ln 2\\
\Omega(\alpha,\mu) &= T(\alpha+1,\mu)+1
\end{aligned}$$

Let $x \in \operatorname{dom} f$, $(e_{0,y}, e_{0,A}) \in \mathbb{R}^{d+1}$, $(e_y, e_A) \in C^0(\mathbb{R}_+, \mathbb{R}^{d+1})$ and $\mu \in \mathbb{R}_+$ such that $\|e_0\| + \int_0^\infty \|e(t)\|\,dt \leqslant e^{-\Theta(\|x\|,\mu)}$. Apply Lemma 4.3.6 (PIVP Slow-Stop) and consider the following systems (where ψ is given by the lemma):

$$\begin{cases} y(0) = q(x,\mu) + e_{0,y}\\ A(0) = T(\mathrm{norm}_{\infty,1}(x),\mu) + 2 + e_{0,A}\end{cases} \qquad \begin{cases} y'(t) = \frac{1+\tanh(A(t))}{2}p(y(t)) + e_y(t)\\ A'(t) = -1 + e_A(t)\end{cases}$$

$$\begin{cases} z(0) = q(x,\mu) + e_{0,y}\\ z'(t) = p(z(t)) + (\psi^{-1})'(t)e_y(\psi^{-1}(t))\end{cases} \qquad \begin{cases} w(0) = q(x,\mu)\\ w'(t) = p(w(t))\end{cases}$$

By definition of p and q, if $t \geqslant \Omega^*(\|x\|,\mu)$ then $\|w_{1..m}(t) - f(x)\| \leqslant e^{-\mu}$. Furthermore, $\|w(t)\| \leqslant \Upsilon^*(\|x\|,\mu,t)$ for all $t \in \mathbb{R}_+$. Define $T^* = T(\mathrm{norm}_{\infty,1}(x),\mu)$. Apply Lemma 2.6.18 (Norm function) to get that $\|x\| \leqslant \mathrm{norm}_{\infty,1}(x) \leqslant \|x\|+1$ and thus $T(\|x\|,\mu) \leqslant T^* \leqslant T(\|x\|+1,\mu)$. By construction, $\psi(t) \leqslant T^*+4$ for all $t \in \mathbb{R}_+$. Let $t \in \mathbb{R}_+$, apply Theorem 2.4.2 (Parameter dependency of PIVP) by checking that:

$$\begin{aligned}
&\left(\left\|e_{0,y}\right\| + \int_0^{\psi(t)} \left\|(\psi^{-1})'(u)e_y(\psi^{-1}(u))du\right\|\right) e^{k\Sigma p\int_0^{\psi(t)}(\|w(u)\|+1)^{k-1}du}\\
&\leqslant \left(\left\|e_{0,y}\right\| + \int_0^t \left\|e_y(u)\right\|du\right) e^{k\Sigma p\int_0^{\psi(t)}(\Upsilon^*(\|x\|,\mu,u)+1)^{k-1}du} && \text{by a change of variable}\\
&\leqslant e^{k\Sigma p\psi(t)(\Upsilon^*(\|x\|,\mu,\psi(t))+1)^{k-1}-\Theta(\|x\|,\mu)} && \text{by hypothesis on the error}\\
&\leqslant e^{k\Sigma p(T(\|x\|+1,\mu)+4)(\Upsilon^*(\|x\|,\mu,T(\|x\|+1,\mu)+4)+1)^{k-1}-\Theta(\|x\|,\mu)} && \text{because } \psi \text{ is bounded}\\
&\leqslant e^{-\mu-\ln 2} \leqslant 1 && \text{by definition of } \Theta
\end{aligned}$$

Thus $\|z(\psi(t)) - w(\psi(t))\| \leqslant e^{-\mu-\ln 2}$ for all $t \in \mathbb{R}_+$. Furthermore, if $t \geqslant \Omega(\|x\|, \mu)$ then $\psi(t) \geqslant \psi(T(\|x\|+1,\mu)+1) \geqslant \psi(T^*+1) \geqslant T^*$. By construction $\psi(T^*) \geqslant T^*$ so $\psi(t) \geqslant T^* \geqslant T(\|x\|,\mu) = \Omega^*(\|x\|, \mu+\ln 2)$ thus $\|z(\psi(t)) - f(x)\| \leqslant e^{-\mu-\ln 2}$. Consequently, $\|y(t)-f(x)\| \leqslant \|z(\psi(t)) - w(\psi(t))\| + \|w(\psi(t)) - f(x)\| \leqslant 2e^{-\mu-\ln 2} \leqslant e^{-\mu}$.

Let $t \in \mathbb{R}_+$, then $\|y(t)\| = \|z(\psi(t))\| \leqslant \|w(\psi(t))\| + e^{-\mu} \leqslant \Upsilon^*(\|x\|, \mu, \psi(t)) + 1 \leqslant \Upsilon^*(\|x\|, \mu, T(\|x\|+1,\mu)+4)+1 \leqslant \Upsilon^*(\|x\|, \mu, \Omega^*(\|x\|+1, \mu+\ln 2)+4)+1$ which is polynomially bounded in $\|x\|$ and μ. Furthermore $|A(t)| \leqslant T^*+4 \leqslant \Omega^*(\|x\|+1, \mu+\ln 2)+4$ which are both polynomially bounded in $\|x\|$, μ.

Finally, $(y,A)(0) = g(x,\mu) + e_0$ and $(y,A)'(t) = h(y(t), A(t)) + e(t)$ where g and h belong to $\text{GPVAL}_{\mathbb{K}}$ because $\tanh, \text{norm}_{\infty,1} \in \text{GPVAL}$. ■

Remark 4.3.8 (Polynomial versus generable)**:** The proof of Theorem 4.3.7 (Weak ⊆ robust) also works if q is generable (i.e. $q \in \text{GPVAL}$) instead of polynomial in Definition 4.2.1 (Analog computability) or Definition 4.2.7 (Analog weak computability). ♦

4.3.II Strong computability

In the previous sections, we introduced the notion of weak computability and saw that contrary to what one would expect, this notion is robust to slight pertubations. The notion of robust computability is a first good step toward a compositional approach for the GPAC but it is not enough. In particular, if we want to compose such systems, we will need to tackle two related problems. First, what happens if the perturbations are not within the bound given by Θ ? Second, what happens if the input is not within the domain of definition of f ? The notion of robust computability does not given any information on the behavior of the system in those cases. In the worst case, the system could explode in finite time which is not desirable.

To make an analogy with programs, until this point we were assuming that our programs only received well-formed inputs. We are now trying understand if it possible to make our programs robust against ill-formed inputs or errors during the computation. Of course, we do not expect the program to give a correct result under such circumstances, but we expect it to behave nicely (i.e. not crash). In our setting, the counterpart of a "nice behavior" is for the system to stay polynomially bounded in $\|x\|$, μ and t – in other words the Υ bound should always hold.

Definition 4.3.9 (Analog strong computability)**:** Let $n, m \in \mathbb{N}$, $f :\subseteq \mathbb{R}^n \to \mathbb{R}^m$, $\Theta, \Omega : \mathbb{R}_+^2 \to \mathbb{R}_+$ and $\Upsilon : \mathbb{R}_+^4 \to \mathbb{R}_+$. We say that f is $(\Upsilon, \Omega, \Theta)$-strongly-computable if and only if there exists $d \in \mathbb{N}$, and $(h : \mathbb{R}^d \to \mathbb{R}^d), (g : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}^d) \in \text{GPVAL}$ such that for any $x \in \mathbb{R}^n$, $\mu \in \mathbb{R}_+$, $e_0 \in \mathbb{R}^d$ and $e \in C^0(\mathbb{R}_+, \mathbb{R}^d)$, there is exists (a unique) $y : \mathbb{R}_+ \to \mathbb{R}^d$ satisfying for all $t \in \mathbb{R}_+$ and $\hat{e}(t) = \|e_0\| + \int_0^t \|e(u)\| du$:

- $y(0) = g(x,\mu) + e_0$ and $y'(t) = h(y(t)) + e(t)$ ► y satisfies a generable IVP
- if $x \in \text{dom} f$, $t \geqslant \Omega(\|x\|, \mu)$ and $\hat{e}(t) \leqslant e^{-\Theta(\|x\|,\mu)}$ then $\|y_{1..m}(t) - f(x)\| \leqslant e^{-\mu}$
- $\|y(t)\| \leqslant \Upsilon(\|x\|, \mu, \hat{e}(t), t)$ ► $y(t)$ is bounded

We denote by $\text{AS}(\Upsilon, \Omega, \Theta)$ the set of $(\Upsilon, \Omega, \Theta)$-strongly-computable functions. ♦

Definition 4.3.10 (Analog strong poly-time)**:** Denote by ASP the set of (poly, poly, poly)-strongly-computable functions. ♦

We will now see that any robustly-computable function is in fact strongly-computable. In other words, any robust system can be made to withstand inputs outside of the domain of definition and arbitrary large perturbations, without exploding in finite time, or having uncontrolled growth.

The idea of the construction is intuitive but technical. Since the system can be subject to large perturbations, we cannot rely on the robust system to well-behave so the system is going to monitor itself. More precisely, for a given robust system, we know that any valid computation satisfies $\|y(t)\| \leqslant \Upsilon(\|x\|, \mu, t)$. Converly, if at some point $\|y(t)\|$ becomes greater than $\Upsilon(\|x\|, \mu, t)$ then something wrong must have happened and the system should stop immediately to prevent further ill-behavior. Schematically, the system should look like this:

$$y(0) = g(x) + e_0 \qquad y'(t) = \operatorname{monitor}(y(t))p(y(t)) + e(t)$$

where $\operatorname{monitor}(y(t))$ is 1 if $\|y(t)\| \leqslant \Upsilon(\|x\|, \mu, t)$ and 0 otherwise. Of course, this system is ideal because we can only build continuous approximations of such a monitor, and this approximation is going to introduce even more errors in the system. Fortunately, we can control the errors introduced by the monitor and make them small so that the robust system can handle them. Another issue is that the monitor can never be exactly 0 when $\|y(t)\| \geqslant \Upsilon(\|x\|, \mu, t)$, but only "very small'. The idea is that small enough is sufficient for the system to stay bounded as it can be observed on the following toy example:

$$\begin{cases} y(0) = 1 \\ z(0) = 1 \end{cases} \qquad \begin{cases} y'(t) = \frac{1+\tanh(10-y(t))}{2} y(t)^{42} \\ z'(t) = z(t)^{42} \end{cases}$$

It is clear that $z(t)$ explodes in finite time. Define $\psi(x) = \frac{1+\tanh(10-x)}{2} x^{42}$ so that $y'(t) = \psi(y(t))$. We claim that $y(t)$ is bounded by a polynomial in t. Indeed, observe that for $x \in \mathbb{R}_+$:

$$\begin{aligned} \psi(x) &\leqslant \frac{1}{2} e^{10-x} x^{42} && \text{using Lemma 2.6.1 (Bounds on tanh)} \\ &\leqslant e^{10-42} 42^{42} && \text{by a function analysis} \end{aligned}$$

Consequently:

$$y(t) = y(0) + \int_0^t \psi(y(u))du \leqslant 1 + e^{10-42} 42^{42} t$$

Note that in fact $y(t)$ can be shown to be bounded by a (very large) constant but we will not need such a result in our proof. Furthemore, we claim that there is a relationship between y and z because:

$$y(t) = z\left(\int_0^t \phi(u)du\right) \qquad \text{where } \phi(t) = \frac{1 + \tanh(10 - y(t))}{2}$$

This is where the exact value of ϕ also matters. Indeed, if we are only interested in the behaviour of z for $z(t) \leqslant 10$ then studying z or y is roughly the same because for all $t \in \mathbb{R}_+$ such that $z(t) \leqslant 10$, we have $\frac{1}{2} \leqslant \phi(t) \leqslant 1$ and thus for all such t:

$$y(t) = z(\xi(t)) \qquad \text{where } \xi(t) \in [t/2, t]$$

Of course this is a toy example but the principle works for any z satisfying a PIVP.

On a more technical side, we will need to "apply" Definition 4.3.1 (Analog robust computability) over finite intervals and we need the following lemma to do so.

Lemma 4.3.11 (Finite time robustness)**:** *Let $f \in \operatorname{AR}(\Upsilon, \Omega, \Theta)$, $I = [0, T]$, $x \in \operatorname{dom} f$, $\mu \in \mathbb{R}_+$, $e_0 \in \mathbb{R}^d$ and $e \in C^0(I, \mathbb{R}^d)$ such that $\|e_0\| + \int_I \|e(t)\| dt < e^{-\Theta(\|x\|, \mu)}$. Assume that $y : I \to \mathbb{R}^d$ satisfies for all $t \in I$:*

$$y(0) = g(x, \mu) + e_0 \qquad y'(t) = h(y(t)) + e(t)$$

where g, h come from Definition 4.3.1 (Analog robust computability) applied to f. Then for all $t \in I$:

- $\|y(t)\| \leqslant \Upsilon(\|x\|, \mu, t)$
- *if* $t \geqslant \Omega(\|x\|, \mu)$ *then* $\|y_{1..m} - f(x)\| \leqslant e^{-\mu}$

♦

Proof. The trick is simply to extend e so that it is defined over $\mathbb{R}_+$ and such that:

$$\|e_0\| + \int_0^\infty \|e(u)\| \, du \leqslant e^{-\Theta(\|x\|,\mu)}$$

This is always possible because the truncated integral is stricer smaller than the bound. Formally, define for $t \in \mathbb{R}_+$:

$$\bar{e}(t) = \begin{cases} e(t) & \text{if } t \leqslant T \\ e(T)e^{\frac{e(T)}{\varepsilon}(T-t)} & \text{otherwise} \end{cases} \qquad \text{where } \varepsilon = e^{-\Theta(\|x\|,\mu)} - \|e_0\| - \int_I \|e(t)\| > 0$$

One easily checks that $\bar{e} \in C^0(\mathbb{R}_+, \mathbb{R}^d)$ and that:

$$\begin{aligned} \|e_0\| + \int_0^\infty \|\bar{e}(t)\| \, dt &= \|e_0\| + \int_0^T \|e(t)\| \, dt + \int_T^\infty e(T)e^{\frac{e(T)}{\varepsilon}(T-t)} dt \\ &= e^{-\Theta(\|x\|,\mu)} - \varepsilon + \left[-\varepsilon e(T) e^{\frac{e(T)}{\varepsilon}(T-t)}\right]_T^\infty \\ &= e^{-\Theta(\|x\|,\mu)} \end{aligned}$$

Assume that $z : \mathbb{R}_+ \to \mathbb{R}^d$ satisfies for $t \in \mathbb{R}_+$:

$$z(0) = g(x, \mu) \qquad z'(t) = g(z(t)) + \bar{e}(t)$$

Then z satisfies Definition 4.3.1 (Analog robust computability) so $\|z\|(t) \leqslant \Upsilon(\|x\|, \mu)$ and if $t \geqslant \Omega(\|x\|, \mu)$ then $\|z_{1..m}(t) - f(x)\| \leqslant e^{-\mu}$. Conclude by noting that $z(t) = y(t)$ for all $t \in [0, T]$ since $e(t) = \bar{e}(t)$. ■

Theorem 4.3.12 (Robust ⊆ strong)**:** ARP ⊆ ASP. ♦

Proof. Let Ω, Θ, Υ be polynomials and $(f :\subseteq \mathbb{R}^n \to \mathbb{R}^m) \in \text{AR}(\Upsilon, \Omega, \Theta)$. Without loss of generality, we assume that Ω, Θ, Υ are increasing functions of their arguments. Apply Definition 4.3.1 (Analog robust computability) to get d, h and g. Let $x \in \mathbb{R}^n$, $\mu \in \mathbb{R}_+$, $(e_{0,y}, e_{0,\ell}) \in \mathbb{R}^{d+1}$ and $(e_y, e_\ell) \in C^0(\mathbb{R}_+, \mathbb{R}^{d+1})$. Define $\hat{e}(t) = \|e_0\| + \int_0^t \|e(u)\| \, du$, and consider the following system for $t \in \mathbb{R}_+$:

$$\begin{cases} y(0) = g(x, \mu) + e_{0,y} \\ y'(t) = \psi(t)h(y(t)) + e_y(t) \\ \ell(0) = \text{mx}_1(\text{norm}_{\infty,1}(x), \mu) + 1 + e_{0,\ell} \\ \ell'(t) = 1 + e_\ell(t) \end{cases}$$

$$\psi(t) = \frac{1 + \tanh(\Delta(t))}{2} \qquad \Delta(t) = \Upsilon(\ell(t), \ell(t), \ell(t)) + 1 - \text{norm}_{\infty,1}(y(t))$$

We will first show that the system remains polynomially bounded. Apply Lemma 2.6.16 (Max function) and Lemma 2.6.18 (Norm function) to get that:

$$\begin{aligned} \|\ell(0)\| &\leqslant \max(\|x\| + 1, \mu) + 1 + \|e_{0,\ell}\| \\ &\leqslant \text{poly}(\|x\|, \mu) + \|e_{0,\ell}\| \end{aligned}$$

Consequently:

$$\begin{aligned}\|\ell(t)\| &\leqslant \|\ell(0)\| + \int_0^t 1 + \|e_\ell(u)\|\,du\\ &\leqslant \operatorname{poly}(\|x\|,\mu) + t + \|e_{0,\ell}\| + \int_0^t \|e_\ell(u)\|\,du\\ &\leqslant \operatorname{poly}(\|x\|,\mu) + t + \hat{e}(t)\\ &\leqslant \operatorname{poly}(\|x\|,\mu,t,\hat{e}(t)) \qquad (4.3.13)\end{aligned}$$

Since $g, h \in \mathrm{GPVAL}$, there exists sp and $\overline{\mathrm{sp}}$ polynomials such that $\|g(x)\| \leqslant \mathrm{sp}(\|x\|)$ and $\|h(x)\| \leqslant \overline{\mathrm{sp}}(\|x\|)$ for all $x \in \mathbb{R}^d$ and without loss of generability, we assume that sp and $\overline{\mathrm{sp}}$ are increasing functions. Let $t \in \mathbb{R}_+$, there are two possibilities:

- If $\Delta(t) \geqslant 0$ then $\operatorname{norm}_{\infty,1}(y(t)) \leqslant 1 + \Upsilon(\ell(t),\ell(t),\ell(t))$ so apply Lemma 2.6.18 (Norm function) and use (4.3.13) to conclude that $\|y(t)\| \leqslant \operatorname{poly}(\|x\|,\mu,t,\hat{e}(t))$ and thus:

$$\begin{aligned}\|\psi(t)h(y(t))\| &\leqslant \overline{\mathrm{sp}}(\|y(t)\|) && \text{use that } \tanh < 1\\ &\leqslant \operatorname{poly}(\|x\|,\mu,t,\hat{e}(t)) && (4.3.14)\end{aligned}$$

- If $\Delta(t) < 0$ then apply Lemma 2.6.1 (Bounds on tanh) to get that $\psi(t) \leqslant \frac{1}{2}e^{\Delta(t)} \leqslant e^{\Delta(t)}$. Apply Lemma 2.6.18 (Norm function) to get that $\Delta(t) \leqslant \Upsilon(\ell(t),\ell(t),\ell(t)) + 1 - \|y(t)\|$ and thus $\|y(t)\| \leqslant \Upsilon(\ell(t),\ell(t),\ell(t)) + 1 - \Delta(t)$ and thus:

$$\begin{aligned}\|\psi(t)h(y(t))\| &\leqslant e^{\Delta(t)}\overline{\mathrm{sp}}(\|y(t)\|) && \text{use the bound on } \psi\\ &\leqslant e^{\Delta(t)}\overline{\mathrm{sp}}(\Upsilon(\ell(t),\ell(t),\ell(t)) + 1 - \Delta(t)) && \text{use the bound on } \|y(t)\|\\ &\leqslant \operatorname{poly}(\ell(t))e^{\Delta(t)}\operatorname{poly}(-\Delta(t)) && \text{use that } \Upsilon \text{ is polynomial}\\ &\leqslant \operatorname{poly}(\ell(t)) \quad \text{use that } e^{-x}\operatorname{poly}(x) = O\left(1\right) \text{ for } x \geqslant 0 \text{ and fixed poly}\\ &\leqslant \operatorname{poly}(\|x\|,\mu,t,\hat{e}(t)) && (4.3.15)\end{aligned}$$

Putting (4.3.14) and (4.3.15) together, we get that:

$$\begin{aligned}\|y(t)\| &\leqslant \|g(x,\mu)\| + \left\|e_{0,y}\right\| + \int_0^t \|\psi(u)h(y(u))\| + \left\|e_y(u)\right\|\,du\\ &\leqslant \mathrm{sp}(\|x,\mu\|) + \int_0^t \operatorname{poly}(\|x\|,\mu,u,\hat{e}(u))du + \hat{e}(t)\\ &\leqslant \operatorname{poly}(\|x\|,\mu,t,\hat{e}(t))\end{aligned}$$

We will now analyze the behavior of the system when the error is bounded. Define $\Theta^*(\alpha,\mu) = \Theta(\alpha,\mu) + 1$. Define $\hat{\psi}(t) = \int_0^t \psi(u)du$ and note that it is a diffeomorphism since $\psi > 0$. Apply Lemma 2.4.4 (Perturbed time-scaling) to get that $y(t) = z(\hat{\psi}(t))$ for all $t \in \mathbb{R}_+$, where z satisfies for $\xi \in \hat{\psi}(\mathbb{R}_+)$:

$$z(0) = g(x,\mu) + e_{0,y} \qquad z'(\xi) = h(z(\xi)) + \tilde{e}(\xi) \qquad \text{where} \int_0^{\hat{\psi}(t)} \|\tilde{e}(\xi)\|\,d\xi = \int_0^t \left\|e_y(u)\right\|\,du$$

Assume that $x \in \operatorname{dom} f$ and let $T \in \mathbb{R}_+$ such that $\hat{e}(T) \leqslant e^{-\Theta^*(\|x\|,\mu)}$. Then $\hat{e}(T) < e^{-\Theta(\|x\|,\mu)}$ and for all $t \in [0,T]$:

$$\left\|e_{0,y}\right\| + \int_0^{\hat{\psi}(t)} \|\tilde{e}\|\,(u)du = \left\|e_{0,y}\right\| + \int_0^t \left\|e_y(u)\right\|\,du$$

$$\leqslant \hat{e}(t) \leqslant e^{-\Theta(\|x\|,\mu)}$$

Apply Lemma 4.3.11 (Finite time robustness) to get for all $u \in [0, \hat{\psi}(T)]$:

$$\|z(u)\| \leqslant \Upsilon(\|x\|, \mu, u) \tag{4.3.16}$$

$$\text{if } u \geqslant \Omega(\|x\|, \mu) \text{ then } \|z_{1..m}(u) - f(x)\| \leqslant e^{-\mu} \tag{4.3.17}$$

Apply Lemma 2.6.16 (Max function) and Lemma 2.6.18 (Norm function) to get for all $t \in [0, T]$:

$$\begin{aligned}\ell(t) &\geqslant \operatorname{mx}_1(\operatorname{norm}_{\infty,1}(\|x\|, \mu)) + 1 - \|e_{0,\ell}\| + t - \int_0^t \|e_\ell(u)\| \, du\\ &\geqslant \max(\|x\|, \mu) + 1 + t - \hat{e}(t)\\ &\geqslant \max(\|x\|, \mu, t) && \text{using that } \hat{e}(t) \leqslant 1\end{aligned}$$

Consequently, using Lemma 2.6.18 (Norm function), for all $t \in [0, T]$:

$$\begin{aligned}\Delta(t) &\geqslant \Upsilon(\ell(t), \ell(t), \ell(t)) - \|y(t)\|\\ &\geqslant \Upsilon(\|x\|, \mu, t) - \|y(t)\| && \text{using that } \ell(t) \geqslant \max(\|x\|, \mu, t)\\ &= \Upsilon(\|x\|, \mu, t) - \left\|z(\hat{\psi}(t))\right\| && \text{using that } y(t) = z(\hat{\psi}(t))\\ &\geqslant 0 && \text{because } \hat{\psi}(t) \in [0, \hat{\psi}(T)]\end{aligned}$$

Consequently for all $t \in [0, T]$:

$$\hat{\psi}(t) = \int_0^t \psi(u) du = \int_0^t \frac{1 + \tanh(\Delta(u))}{2} du \geqslant \frac{t}{2}$$

Define $\Omega^*(\alpha, \mu) = 2\Omega(\alpha, \mu)$. Assume that $T \geqslant \Omega^*(\|x\|, \mu)$ then $\hat{\psi}(T) \geqslant \Omega(\|x\|, \mu)$ and thus $\|y_{1..m}(T) - f(x)\| = \left\|z(\hat{\psi}(T)) - f(x)\right\| \leqslant e^{-\mu}$.

Finally, $(y, \ell)(0) = g^*(x, \mu) + e_0$ where $g^* \in$ GPVAL. Similarly $(y, \ell)'(t) = h^*((y, \ell)(t)) + e(t)$ where $h^* \in$ GPVAL. Note again that both h^* and g^* are defined over the entire space. This concludes the proof that $f \in \mathrm{AS}(\Omega^*, \mathrm{poly}, \Theta^*)$. ■

4.4 Computing with variable input

The previous notions of computability were essentially "offline", meaning that the system was given the whole input from the beginning. In the context of real computations, this is somewhat unrealistic because we usually only have access to an approximation of the input, or an arbitrarily precise approximation via an analog system for example. But certainly, the main issue of these notions of computability is that they do not compose very well. Indeed, the input is given as a whole from the beginning but the system never produces the output as the whole: it only converges to it ! This makes it impossible, or at least not obvious, to compose two functions by feeding one system with the output of another.

For this reason, we would like to have a better notion of computability where the input is not known entirely from the beginning but rather some converging approximation. In other words, we want to design "online" systems that can compute approximation of the output given an approxiamtion of the input. We will introduce two such notion: one without perturbations and one with pertubations.

4.4.I Extreme computability

In this section we introduce our strongest notion of analog computability. In this setting, the system is provided with an approximate input and, assuming the input is stable and precise enough, is required to produce a correct output with increasing precision over time. Also, we require this behavior to hold for any period of time, so the system input can change and the system will account for this change (with some delay). This gives the system an "online" behavior because it must adjust to unpredictable changes. To make things more realistic and more interesting, we even require that the system be robust to small perturbations during the computation. We will even require the system to withstand arbitrarily large perturbations without exploding in finite time, and to work with any initial condition. To support such extreme requirements, like in the case of robust computability, we allow the right-hand side to be any (polynomially bounded) generable function. Informally, the system will look like:

$$y'(t) = g(y(t), x(t), \mu(t)) + e(t)$$

where $x(t)$ is the input, $\mu(t)$ is the requested precision and $e(t)$ is the perturbation. The system must be able to start for any initial condition. Informally we require that if over a sufficiently large time interval the input is stable ($x(t) \approx x$), the precision is roughly stable ($\mu(t) \in [\mu, \mu{+}1]$) and the error is not too big then $y(t)$ converges to $f(x) \pm e^{-\mu}$.

In this definition, as well as in the next ones, we will need to measure some bounds based on the value of some variable during a small (past but still non-negative) time period. We introduce the following notation for convenience.

Definition 4.4.1 (Past sup)**:** For any $f : \mathbb{R}_+ \to \mathbb{R}$ and $\delta \geqslant 0$, define:

$$\operatorname{sup}_\delta f(t) = \sup_{u \in [t-\delta, t] \cap \mathbb{R}_+} f(u)$$

♦

Definition 4.4.2 (Extreme computability)**:** Let $n, m \in \mathbb{N}$, $f :\subseteq \mathbb{R}^n \to \mathbb{R}^m$, $\Upsilon : \mathbb{R}_+^3 \to \mathbb{R}_+$ and $\Omega, \Lambda, \Theta : \mathbb{R}_+^2 \to \mathbb{R}_+$. We say that f is $(\Upsilon, \Omega, \Lambda, \Theta)$-extremely-computable if and only if there exists $\delta \geqslant 0$, $d \in \mathbb{N}$ and $(g : \mathbb{R}^d \times \mathbb{R}^{n+1} \to \mathbb{R}^d) \in \mathrm{GPVAL}_{\mathbb{K}}$ such that for any $x \in C^0(\mathbb{R}_+, \mathbb{R}^n)$, $\mu \in C^0(\mathbb{R}_+, \mathbb{R}_+)$, $y_0 \in \mathbb{R}^d$, $e \in C^0(\mathbb{R}_+, \mathbb{R}^d)$ there exists (a unique) $y : \mathbb{R}_+ \to \mathbb{R}^d$ satisfying for all $t \in \mathbb{R}_+$:

- $y(0) = y_0$ and $y'(t) = g(t, y(t), x(t), \mu(t)) + e(t)$
- $\|y(t)\| \leqslant \Upsilon\left(\operatorname{sup}_\delta \|x\|(t), \operatorname{sup}_\delta \mu(t), \|y_0\| \mathbb{1}_{[1,\delta]}(t) + \int_{\max(0,t-\delta)}^{t} \|e(u)\|\, du\right)$
- For any $I = [a, b]$, if there exists $\bar{x} \in \operatorname{dom} f$ and $\check{\mu}, \hat{\mu} \geqslant 0$ such that for all $t \in I$, $\mu(t) \in [\check{\mu}, \hat{\mu}]$, $\|x(t) - \bar{x}\| \leqslant e^{-\Lambda(\|\bar{x}\|, \hat{\mu})}$ and $\int_a^b \|e(u)\|\, du \leqslant e^{-\Theta(\|\bar{x}\|, \hat{\mu})}$ then $\|y_{1..m}(u) - f(\bar{x})\| \leqslant e^{-\check{\mu}}$ whenever $a + \Omega(\|\bar{x}\|, \hat{\mu}) \leqslant u \leqslant b$.

We denote by $\mathrm{AX}(\Upsilon, \Omega, \Lambda, \Theta)$ the set of $(\Upsilon, \Omega, \Lambda, \Theta)$-extremely-computable functions. ♦

Definition 4.4.3 (Analog extreme poly-time)**:** Denote by AXP the set of (poly, poly, poly, poly)-extremely-computable functions. ♦

Remark 4.4.4 (Non-autonomous system)**:** Contrary to all other notions of computability, we grant extreme systems access to the time, making them non-autonomous. Intuitively, this seems necessary because the system cannot assume anything about the initial condition or the perturbations, and yet we require that it "follows" the input in a timely manner. We do not think it is possible to do this with an perturbed autonomous system. ♦

Remark 4.4.5 (Regularity of the input)**:** It is notable that the second condition of the definition is not $\|y(t)\| \leqslant \Upsilon(\|x(t)\|, \mu(t), t)$ as one could have expected. The reason for this is that we only require x to be a continuous function but it can have arbitrary quick variations.

For example, consider a GPAC y computing the identity function. We can build a continuous function x such that $x(t) = A$ for $t \leqslant 1$ and $x(t) = 0$ for $t \geqslant 1 + \varepsilon$. We can always find a sufficiently large A and sufficiently small ε such that y cannot "follow" x quickly enough to ensure that $\|y(t)\| \leqslant \Upsilon(\|x(t)\|, t)$ for any fixed Υ. Indeed, if it were the case, with small enough ε, we would have $y(t+\varepsilon) \approx y(t) \approx A$ and $x(t+\varepsilon) = 0$, so $A \leqslant \Upsilon(0, 1+\varepsilon)$ for any A and ε which is absurd.

The intuition behind this issue is that the maximum growth rate of y at a time t is limited by $\|y(t)\|$ and the dimension of the system. If x has a faster changing rate than this bound, y cannot even follow the variations of x. This is not an issue when the GPAC is computing a value because $x(t)$ is assumed to be nearly constant (variations are controlled by Λ) but for the general size bound, we cannot assume anything on x. A simple remedy is give the system a little bit of time (δ) to take the variations of x into account.

For simplicity, we take δ to be a constant because it can easily be seen that a constant time is sufficient to adjust to any change (this is mostly a consequence of Lemma 4.4.12 (Reach)). Over the time interval $[t, t-\delta]$, it seems natural and simple to consider the supremum of the norm ($\sup_{[t,t-\delta]} \|x\|$), but other measures could have been used, like the average norm ($\frac{1}{\delta}\int_{t-\delta}^{t} \|x\|$), without changing the obtained class. Another interpretation is to see δ as the minimum "sample time" of the system. ♦

4.4.II Reaching a value

The notion of extreme computability might seem so strong at first that one can wonder if anything is really computable in this sense. In this section, we will introduce a very useful pattern which we call "reaching a value". This can be seen as a proof that all constant function or generable functions are extremely-computable as well as the basic block to build more complicated such functions. As as introductory example, consider the system:

$$y'(t) = \alpha - y(t)$$

This sysem can be shown to converge to α whatever the initial value is. In this section we extend this system in several non-trivial ways. In particular, we want to ensure a certain rate of convergence in all situations and we want to make this system robust to perturbations. In other words, we want to analyze:

$$y'(t) = \alpha(t) - y(t) + e(t)$$

where $e(t)$ is a perturbation and $\alpha(t) \approx \alpha$.

Definition 4.4.6 (Reach ODE)**:** Let $T > 0$, $I = [0, T]$, $g, E : I \to \mathbb{R}$, $\phi : I \to \mathbb{R}_+^*$. Define (4.4.7) as the following differential equation for $t \in I$,

$$\begin{cases} y'(t) = \phi(t)X_3(g(t) - y(t)) + E(t) \\ y(0) = y_0 \end{cases} \quad \text{where } X_3(u) = u + u^3 \tag{4.4.7}$$

♦

Lemma 4.4.8 (Reach ODE: integral error)**:** *Let $T > 0$, $I = [0, T]$, $g, E \in C^0(I, \mathbb{R})$, $\phi \in C^0(I, \mathbb{R}_+^*)$. Assume that $\exists \eta > 0$ and $\bar{g} \in \mathbb{R}$ such that $\forall t \in I, |g(t) - \bar{g}| \leqslant \eta$. Then the solution y to* (4.4.7) *exists over I and satisfies:*

$$|y(T)-\bar{g}| \leqslant \eta + \int_0^T |E(t)|dt + \frac{1}{\sqrt{\exp(2\int_0^T \phi(u)du) - 1}}$$

Furthermore, for any $t \in I$:

$$|y(t)-\bar{g}| \leqslant \max(\eta, |y(0)-\bar{g}|) + \int_0^t |E(u)|du$$

♦

Proof. Write $f(t,x) = E(t)+\phi(t)X_3(g(t)-x)$, then $y'(t) = f(t,y(t))$. Define $I(t) = \int_0^t |E(u)|du$ and consider:

$$f_+(t,x) = |E(t)| + \phi(t)X_3\left(\bar{g}+\eta-(x-I(t))\right)$$
$$f_-(t,x) = -|E(t)| + \phi(t)X_3\left(\bar{g}-\eta-(x+I(t))\right)$$

Since X_3 and I are increasing functions, it is easily seen that $f_-(t,x) \leqslant f(t,x) \leqslant f_+(t,x)$. By a classical result of differential inequalities, we get that $y_-(t) \leqslant y(t) \leqslant y_+(t)$ where $y_-(0) = y_+(0) = y(0)$ and $y'_\pm(t) = f_\pm(t, y_\pm(t))$. Now realise that:

$$y'_+(t) - I'(t) = \phi(t)X_3(\bar{g}+\eta-(y_+(t)-I(t)))$$
$$y'_-(t) + I'(t) = \phi(t)X_3(\bar{g}-\eta-(y_-(t)+I(t)))$$

which are two instances of the following differential equation:

$$x(0) = x_0 \qquad x'(t) = \phi(t)X_3(x_\infty - x(t))$$

Since ϕ and X_3 are continuous, this equation has a unique solution by the Cauchy-Liptchiz theorem and one can check that the following is a solution:

$$x(t) = x_\infty + \underbrace{\frac{x_0 - x_\infty}{\sqrt{(e^{2\int_0^t \phi(u)du} - 1)(1+(x_0-x_\infty)^2)+1}}}_{:=\alpha(x_0,x_\infty,t)}$$

Furthermore, one can check that for any $a, b \in \mathbb{R}$ and any $t > 0$:

- $|\alpha(a,b,t)| \leqslant \frac{1}{\sqrt{e^{2\int_0^T \phi(u)du} - 1}}$
- $\min(0, a-b) \leqslant \alpha(a,b,t) \leqslant \max(0, a-b)$

It follows that:

$$\bar{g}-\eta-I(t)+\alpha(y(0),\bar{g}-\eta,t) \leqslant y(t) \leqslant \bar{g}+\eta+I(t)+\alpha(y(0),\bar{g}+\eta,t)$$
$$-\eta-I(t)+\alpha(y(0),\bar{g}-\eta,t)) \leqslant y(t)-\bar{g} \leqslant \eta+I(t)+\alpha(y(0),\bar{g}+\eta,t)$$

Using the first inequality on α we get that:

$$-\eta - I(t) - \frac{1}{\sqrt{e^{2\int_0^T \phi(u)du} - 1}} \leqslant y(t)-\bar{g} \leqslant \eta + I(t) + \frac{1}{\sqrt{e^{2\int_0^T \phi(u)du} - 1}}$$

Which proves the first result. And using the second inequality we get that:

$$-\eta - I(t) + \min(0, y(0)-(\bar{g}-\eta))| \leqslant y(t)-\bar{g} \leqslant \eta + I(t) + \max(0, y(0)-(\bar{g}+\eta))$$

This proves the second result by case analysis. ∎

Sometimes though, the previous lemma lacks some precision. In particular when ϕ is never close to 0, where the intuition tells us that we should be able to replace the $\int_0^t |E(u)|du$ with some bound that does not depend on t. The next lemma focuses on this case exclusively.

Lemma 4.4.9 (Reach ODE: worst error)**:** *Let $T > 0$, $I = [0,T]$, $g, E : I \to \mathbb{R}$, $\phi : I \to \mathbb{R}_+^*$. Assume that $\exists \eta, \phi_{min}, E_{max} > 0$ and $\bar{g} \in \mathbb{R}$ such that*

- *$\forall t \in I, |g(t) - \bar{g}| \leqslant \eta$.*
- *$\forall t \in I, |E(t)| \leqslant E_{max}$*
- *$\forall t \in I, \phi(t) \geqslant \phi_{min}$*

Then the solution y to (4.4.7) *exists over I and satisfies for all $t \in I$:*

$$|y(t) - \bar{g}| \leqslant \eta + \frac{E_{max}}{\phi_{min}} + \frac{1}{\sqrt{\exp(2\int_0^t \phi(u)du) - 1}}$$

♦

Proof. Define $\psi(t) = \int_0^t \phi(u)du$ for $t \in I$. Since $\phi(t) \geqslant \phi_{min} > 0$ then ψ is an increasing function and admits an inverse ψ^{-1}. Define for all $\xi \in [0, \psi(T)]$, $z_\infty(\xi) = g(\psi^{-1}(\xi))$ and $z(\xi) = y(\psi^{-1}(\xi))$. One sees that z satisfies

$$z'(\xi) = \underbrace{X_3(z_\infty(\xi) - z(\xi)) + \frac{E(\psi^{-1}(\xi))}{\phi(\psi^{-1}(\xi))}}_{:=f(\xi,z(\xi))}$$

for $\xi \in [0, \psi(T)]$ and $z(0) = y(0)$. Furthermore, for all such ξ, $|z_\infty(\xi) - \bar{g}| \leqslant \eta$ and $|\frac{E(\psi^{-1}(\xi))}{\phi(\psi^{-1}(\xi))}| \leqslant \frac{E_{max}}{\phi_{min}}$. Define $\alpha = \frac{E_{max}}{\phi_{min}}$, $f_+(x) = X_3(\bar{g} + \eta - x) + \alpha$ and $f_-(x) = X_3(\bar{g} - \eta - x) - \alpha$. One can check that $f_-(x) \leqslant f(\xi, x) \leqslant f_+(x)$ for any ξ and x. Consider the solutions z_- and z_+ to $z_-' = f_-(z_-)$ and $z_+' = f_+(z_+)$ where $z_-(0) = z_+(0) = z(0) = y(0)$. By a classical result of differential inequalities, we get that $z_-(\xi) \leqslant z(\xi) \leqslant z_+(\xi)$. By shifting the solutions, both are instances of a system of the form:

$$x(0) = x_0 \qquad x'(t) = -X_3(x(t)) + \varepsilon$$

Since $x \mapsto -X_3(x) + \varepsilon$ is an increasing function, there exists a unique x_∞ such that $\varepsilon = X_3(x_\infty)$. Define $f(x) = -X_3(x) + \varepsilon$ and $f^*(x) = X_3(x_\infty - x)$. One checks that $f^*(x) - f(x) = 3x_\infty(x^2 - x_\infty^2)$, thus $f^*(x) \leqslant f(x)$ if $x \leqslant x_\infty$ and $f(x) \leqslant f^*(x)$ if $x_\infty \leqslant x$. Notice that $f(x_\infty) = 0$, so by a classical result of differential equations, $x(t) - x_\infty$ must keep constant sign for the entire life of the solution (i.e. $x(t)$ cannot "cross" x_∞). Consider the solutions x_- and x_+ to $x_- = f^*(x_-)$ and $x_+ = f^*(x_+)$ where $x_-(0) = \min(x_\infty, x_0)$ and $x_+(0) = \max(x_\infty, x_0)$. Then the previous remark and a standard result guarantees that $x_-(t) \leqslant x(t) \leqslant x_+(t)$. By theorem the equations $x_\pm' = f^*(x_\pm)$ have a unique solution and one can check that the following are solutions:

$$x_\pm(t) = x_\infty + \frac{x_\pm(0) - x_\infty}{\sqrt{(e^{2t} - 1)(1 + (x_\pm(0) - x_\infty)^2) - 1)}}$$

We immediately deduce that $|x_\pm(t) - x_\infty| \leqslant \frac{1}{\sqrt{e^{2t}-1}}$ and so $|x(t) - x_\infty| \leqslant \frac{1}{\sqrt{e^{2t}-1}}$. Let δ_∞ be such that $X_3(\delta_\infty) = \alpha$. Unrolling the definitions, we get that $|z_\pm(\xi) - \bar{g} \mp \delta_\infty \mp \eta| \leqslant \frac{1}{\sqrt{e^{2t}-1}}$. So $|z(\xi) - \bar{g}| \leqslant \eta + \delta_\infty + \frac{1}{\sqrt{e^{2\xi}-1}}$. And finally, since $y(t) = z(\psi(t))$, we get that $|y(t) - \bar{g}| \leqslant \eta + \delta_\infty + \frac{1}{\sqrt{e^{2\int_0^t \phi(u)du} - 1}}$. To conclude, it suffices to note that if $X_3(\delta_\infty) = \alpha$ then $\delta_\infty \leqslant \alpha$ since $X_3(x) \geqslant x$ for all x. ■

Definition 4.4.10 (Reach function)**:** For any $\phi \geqslant 0$ and $y, g \in \mathbb{R}$, define

$$\operatorname{reach}(\phi, y, g) = 2\phi X_3(g-y) \qquad \text{where } X_3(x) = x + x^3$$

♦

Remark 4.4.11: It is useful to note that for any $\phi, \psi \in \mathbb{R}_+$ and $y, g \in \mathbb{R}$,

$$\phi \operatorname{reach}(\psi, y, g) = \operatorname{reach}(\phi\psi, y, q)$$

♦

Lemma 4.4.12 (Reach)**:** *For any $I = [a, b]$, any $\phi \in C^0(I, \mathbb{R}_+)$, any $g, E \in C^0(I, \mathbb{R})$, any $y_0, g_\infty \in \mathbb{R}$ and $\eta > 0$ such that for all $t \in I$, $|g(t) - g_\infty| \leqslant \eta$. Assume that $y : I \to \mathbb{R}$ satisfies*

$$\begin{cases} y(0) = y_0 \\ y'(t) = \operatorname{reach}(\phi(t), y(t), g(t)) + E(t) \end{cases}$$

Then for any $t \in I$,

$$|y(t) - g_\infty| \leqslant \eta + \int_a^t |E(u)|du + \exp\left(-\int_a^t \phi(u)du\right) \qquad \textit{whenever } \int_a^t \phi(u)du \geqslant 1$$

And for any $t \in I$,

$$|y(t) - g_\infty| \leqslant \max(\eta, |y(0) - g_\infty|) + \int_0^t |E(u)|du$$

Furthermore, reach $\in$ GPVAL. ♦

Proof. Apply Lemma 4.4.8 (Reach ODE: integral error) and notice that if $\int_a^t \phi(u)du \geqslant 1$, then:

$$\begin{aligned}\sqrt{\exp\left(\int_a^t 4\phi(u)du\right) - 1} &\geqslant \sqrt{(\exp\left(2\int_a^t \phi(u)du) + 1\right)(\exp\left(2\int_a^t \phi(u)du\right) - 1)} \\ &\geqslant \exp\left(\int_a^t \phi(u)du\right)\sqrt{e^2 - 1} \geqslant \exp\left(\int_a^t \phi(u)du\right)\end{aligned}$$

■

4.4.III Sampling

Another very common pattern in signal processing is known as "sample and hold", where we have a variable signal and we would like to apply some process to it. Unfortunately, the processor often assumes (almost) constant input and does not work in real time (analog-to-digital converters are typical example). In this case, we cannot feed the signal directly to the processor so we need some black box that samples the signal to capture its value, and hold this value long enough for the processor to compute its output. This process is usually used in a τ-periodic fashion: the box samples for time δ and holds for time $\tau - \delta$.

Definition 4.4.13 (Sample and hold)**:** Let $t \in \mathbb{R}, \mu, \tau \in \mathbb{R}_+, x, g \in \mathbb{R}, I = [a, b] \subsetneq [0, \tau]$ and define:

$$\operatorname{sample}_{I,\tau}(t, \mu, x, g) = \operatorname{plil}_{I,\tau}(t, \hat{\mu}, \operatorname{reach}(\check{\mu}, x, g))$$

where

$$\check{\mu} = \frac{\mu + 1}{\min(1, |I|)} \qquad \hat{\mu} = \mu + \max(0, \ln(\tau - |I|))$$

♦

Lemma 4.4.14 (Sample and hold): *Let $\tau \in \mathbb{R}_+$, $I = [a,b] \subsetneq [0,\tau]$, $y : \mathbb{R}_+ \to \mathbb{R}$, $y_0 \in \mathbb{R}$, $x, e \in C^0(\mathbb{R}_+, \mathbb{R})$ and $\mu : \mathbb{R}_+ \to \mathbb{R}_+$ an increasing function. Suppose that for all $t \in \mathbb{R}_+$:*

$$y(0) = y_0 \qquad y'(t) = \operatorname{sample}_{I,\tau}(t, \mu(t), y(t), x(t)) + e(t)$$

Then:

$$|y(t)| \leqslant 2 + \int_{\max(0,t-\tau-|I|)}^{t} |e(u)|du + \max\left(|y(0)|\mathbb{1}_{[0,b]}(t), \sup\nolimits_{\tau+|I|}|x|(t)\right)$$

Furthermore:

- *if $t \notin I \pmod{\tau}$ then $|y'(t)| \leqslant e^{-\mu(t)} + |e(t)|$*
- *for $n \in \mathbb{N}$, if there exists $\bar{x} \in \mathbb{R}$ and $\nu, \nu' \in \mathbb{R}_+$ such that $|\bar{x} - x(t)| \leqslant e^{-\nu}$ and $\mu(t) \geqslant \nu'$ for all $t \in n\tau + I$ then $|y(n\tau + b) - \bar{x}| \leqslant \int_{n\tau+I} |e(u)|du + e^{-\nu} + e^{-\nu'}$*
- *for $n \in \mathbb{N}$, if there exists $\check{x}, \hat{x} \in \mathbb{R}$ and $\nu \in \mathbb{R}_+$ such that $x(t) \in [\check{x}, \hat{x}]$ and $\mu(t) \geqslant \nu$ for all $t \in n\tau + I$ then $y(n\tau + b) \in [\check{x} - \varepsilon, \hat{x} + \varepsilon]$ where $\varepsilon = 2e^{-\nu} + \int_{n\tau+I} |e(u)|du$*
- *for any $J = [c,d] \subseteq \mathbb{R}_+$, if there exists $\nu, \nu' \in \mathbb{R}_+$ and $\bar{x} \in \mathbb{R}$ such that $\mu(t) \geqslant \nu'$ for all $t \in J$ and $|x(t) - \bar{x}| \leqslant e^{-\nu}$ for all $t \in J \cap (n\tau + I)$ for some $n \in \mathbb{N}$, then $|y(t) - \bar{x}| \leqslant e^{-\nu} + e^{-\nu'} + \int_{t-\tau-|I|}^{t} |e(u)|du$ for all $t \in [c + \tau + |I|, d]$*
- *if there exists $\Omega : \mathbb{R}_+ \to \mathbb{R}_+$ such that for any $J = [a,b]$ and $\bar{x} \in \mathbb{R}$ such that for all $\nu \in \mathbb{R}_+$, $n \in \mathbb{N}$ and $t \in (n\tau + I) \cap [a + \Omega(\nu), b]$, $|\bar{x} - x(t)| \leqslant e^{-\nu}$; then $|y(t) - \bar{x}| \leqslant e^{-\nu}$ for all $t \in [a + \Omega^*(\nu), b]$ where $\Omega^*(\nu) = \max(\Omega(\nu + \ln 3), \mu^{-1}(\nu + \ln 3)) + \tau + |I|$*

♦

Proof. Let $n \in \mathbb{N}$. Apply Lemma 2.6.26 ("periodic low-integral-low"), Lemma 4.4.12 (Reach) and Remark 4.4.11 to get that:

- For all $t \in I_n = [n\tau + a, n\tau + b]$: $y'(t) = \phi(t)\operatorname{reach}(\check{\mu}(t), y(t), x(t)) + e(t)$ where $\int_{I_n} \phi \geqslant 1$. Since $|x(t) - 0| \leqslant \sup_{u \in I_n} |x(u)|$ and $\int_{I_n} \phi\check{\mu} = \int_{I_n} \phi\frac{1+\mu}{|I|} \geqslant 1$ then $|y(n\tau + b) - 0| \leqslant \sup_{I_n} |x(u)| + \int_{I_n} |e(u)|du + e^{-1} \leqslant 1 + \sup_{u \in I_n} |x(u)| + \int_{I_n} |e(u)|du$.
- For all $t \in [n\tau + b, (n+1)\tau + a]$: $|y'(t)| \leqslant |e(t)| + e^{-\hat{\mu}(t)} \leqslant |e(t)| + e^{-\ln(\tau-|I|)}$ thus $|y(t) - 0| \leqslant \int_{n\tau+b}^{t} |e(u)|du + (\tau - |I|)e^{-\ln(\tau-|I|)} + 1 + \sup_{u \in I_n} |x(u)| + \int_{I_n} |e(u)|du \leqslant 2 + \sup_{u \in I_n} |x(u)| + \int_{n\tau+a}^{t} |e(u)|du$.
- For all $t \in I_{n+1}$: $y'(t) = \operatorname{reach}(\phi(t)\check{\mu}(t), y(t), x(t))$ where $\int_{I_n} \phi \geqslant 1$. Since $|x(t) - 0| \leqslant \sup_{u \in I_{n+1}} |x(u)|$ then $|y(t) - 0| \leqslant \max(\sup_{u \in [(n+1)\tau+a,t]} |x(u)|, |y((n+1)\tau + a) - 0|) + \int_{(n+1)\tau+a}^{t} |e| \leqslant 2 + \sup_{u \in [n\tau+a,t]} |x(u)| + \int_{n\tau+a}^{t} |e(u)|du$.

Note that this analysis is a bit subtle: the first point *does not* give a bound on $|y(t)|$ over I_n, it only gives a bound on $|y(n\tau + b)|$. On the contrary the two other points give bounds on $|y(t)|$ over $[n\tau + b, (n+1)\tau + b]$ which covers the whole period so by correctly putting everything together, we get that for all $|y(t)| \leqslant 2 + \sup_{u \in [t, t-\tau-|I|] \cap \mathbb{R}_+} |x(u)| + \int_{t-\tau-|I|}^{t} |e(u)|du$ for all $t \geqslant b$. The case of the initial segment is similar in aspect but uses the other result from Lemma 4.4.12 (Reach):

- For all $t \in [0,a]$: $|y'(t)| \leqslant |e(t)| + e^{-\hat{\mu}(t)} \leqslant |e(t)| + e^{-\ln(\tau-|I|)}$ thus $|y(t)| \leqslant \int_0^t |e(u)|du + ae^{-\ln(\tau-|I|)} + |y_0| \leqslant \int_0^t |e(u)|du + 1 + |y_0|$

- For all $t \in [a,b]$: $y'(t) = \operatorname{reach}(\phi(t)\check{\mu}(t), y(t), x(t)) + e(t)$ where $\int_{I_n} \phi \geqslant 1$. Since $|x(t) - 0| \leqslant \sup_{u\in[a,t]} |x(u)|$ then $|y(t) - 0| \leqslant \max(\sup_{u\in[a,t]} |x(u)|, |y(a) - 0|) + \int_a^t |e(u)|du \leqslant 1 + \int_0^t |e(u)|du + \max(|y_0|, \sup_{u\in[a,t]} |x(u)|)$.

Finally, we get that for all $t \in \mathbb{R}_+$:

$$|y(t)| \leqslant 2 + \int_{t-\tau-|I|}^{t} |e(u)|du + \max\left(|y(0)|\mathbb{1}_{[0,b]}(t), \sup_{\tau+|I|}|x|(t)\right)$$

The first extra statement is a trivial consequence of Lemma 2.6.26 ("periodic low-integral-low") and the fact that $\check{\mu}(t) \geqslant \mu(t)$.

The second extra statement has mostly been proved already and uses Lemma 2.6.26 ("periodic low-integral-low") and Lemma 4.4.12 (Reach) again. Let $n \in \mathbb{N}$, assume there exists $\bar{x} \in \mathbb{R}$ and $\nu \in \mathbb{R}_+$ such as described. For all $t \in I_n = [n\tau + a, n\tau + b]$ we have $y'(t) = \phi(t)\operatorname{reach}(\check{\mu}(t), y(t), x(t)) + e(t)$ where $\int_{I_n} \phi \geqslant 1$. Since $|x(t) - \bar{x}| \leqslant e^{-\nu}$ and $\int_{I_n} \phi\check{\mu} = \int_{I_n} \phi\frac{1+\mu}{|I|} \geqslant \nu'$ then $|y(n\tau + b) - \bar{x}| \leqslant e^{-\nu} + \int_{I_n} |e(u)|du + e^{-\nu'}$.

The third statement is a consequence of the previous one: since $n\tau + I$ is a compact set and x is a continuous function, it admits a maximum over $n\tau + I$. Apply the previous statement to $\frac{\bar{x}+\sup_{n\tau+I} x}{2} \geqslant \bar{x}$ to conclude.

The last extra statement requires more work. Let $\nu \geqslant 0$ and $n \in \mathbb{N}$ such that $n\tau + a \geqslant \Omega(\nu)$. Apply Lemma 2.6.26 ("periodic low-integral-low"), Remark 4.4.11 and Lemma 4.4.12 (Reach) to get that:

- For all $t \in I_n$: $y'(t) = \phi(t)\operatorname{reach}(\check{\mu}(t), y(t), x(t))$ where $\int_{I_n} \phi \geqslant 1$. Since $t \geqslant n\tau + a \geqslant \Omega(\nu)$ and $t \in I_n$ then $|x(t) - \bar{x}| \leqslant e^{-\nu}$. And since $\int_{I_n} \phi\check{\mu} = \int_{I_n} \phi\frac{1+\mu}{|I|} \geqslant 1 + \mu(n\tau + a)$ then $|y(n\tau + b) - \bar{x}| \leqslant e^{-\nu} + e^{-\mu(n\tau+a)}$.

- For all $t \in [n\tau + b, (n + 1)\tau + a]$: $|y'(t)| \leqslant e^{-\hat{\mu}(t)} \leqslant e^{-\hat{\mu}(n\tau+a)}$ thus $|y(t) - \bar{x}| \leqslant (\tau - |I|)e^{-\hat{\mu}(n\tau+a)} + e^{-\nu} + e^{-\mu(n\tau+a)} \leqslant e^{-\nu} + 2e^{-\mu(n\tau+a)}$.

- For all $t \in I_{n+1}$: $y'(t) = \phi(t)\operatorname{reach}(\check{\mu}(t), y(t), x(t))$ where $\int_{I_n} \phi \geqslant 1$. Since $t \geqslant n\tau+a \geqslant \Omega(\nu)$ and $t \in I_n$ then $|x(t) - \bar{x}| \leqslant e^{-\nu}$. Thus $|y(t) - \bar{x}| \leqslant \max(e^{-\nu}, |y((n + 1)\tau + a) - \bar{x}|) \leqslant e^{-\nu} + 2e^{-\mu(n\tau+a)}$.

Finally, we get that $|y(t) - \bar{x}| \leqslant e^{-\nu} + 2e^{-\mu(n\tau+a)}$ for all $t \in [n\tau + b, (n + 1)\tau + b]$.

Define $\Omega^*(\nu) = \max(\Omega(\nu + \ln 3), \mu^{-1}(\nu + \ln 3)) + \tau + |I|$. Let $\nu \geqslant 0$ and $t \geqslant \Omega^*(\nu)$. Let $n \in \mathbb{N}$ such that $t \in [n\tau+b, (n+1)\tau+b]$. Then $n\tau+a = (n+1)\tau+b-\tau-|I| \geqslant t-\tau-|I| \geqslant \Omega^*(\nu)-\tau-|I| \geqslant \Omega(\nu + \ln 3)$. By the previous reasoning, we get that $|y(t) - \bar{x}| \leqslant e^{-\nu} + 2e^{-\mu(n\tau+a)}$. And since $n\tau+a \geqslant \Omega^*(\nu)-\tau-|I| \geqslant \mu^{-1}(\nu+\ln 3)$ then $\mu(n\tau+a) \geqslant \nu+\ln 3$. Thus $|y(t)-\bar{x}| \leqslant 3e^{-\nu} \leqslant e^{-\nu}$. ■

4.4.IV Equivalence

In this section, we show that if a function is strongly-computable then it is extremely-computable. The intuitive explanation is that we can build a system that reboots the computation regularly and converge by slowly increasing the precision of the computation. By rebooting, it can also account for the changes in the input and can also restart from a safe state in case the system was heavily perturbed. The robustness if necessary because this process of simulating and rebooting introduces errors in the computation. We will also show that we can rescale any extreme system so that it converges in constant time.

From a high-level point of view, the proof is not very complicated: the system will follow a cycle (see Remark 4.4.4 (Non-autonomous system)) formed by four stages. During the first stage, a variable of the system will converge to some value $\bar{\mu}$: typically some average of $\mu(t)$. We need a stable value for the next stage, the only thing that matters is that this value must be higher than the minimum of $\mu(t)$ over the interval. During the second stage, the system will converge to $g(\bar{x}, \bar{\mu})$ assuming that the input $x(t)$ is very close to some input $\bar{x}$. Of course the resulting value will have a small error, which will be gracefully handled by the strong system. During the third stage, the system will simulate the strong system $h(x(t))$ in an accelerated fashion to reach the requested precision. Again this simulation will introduce some errors but those are already dealt with. Finally, during the fourth stage, the system will copy the computed value to some auxiliary variable. This variable will keep its value nearly constant when the others are computing, thus producing a nearly constant output.

Theorem 4.4.15 (Strong $\subseteq$ extreme)**:** ASP $\subseteq$ AXP. *More precisely if $f \in$ ASP then there exists polynomials $\Upsilon, \Lambda, \Theta$ and a constant polynomial Ω such that $f \in \text{AX}(\Upsilon, \Omega, \Lambda, \Theta)$.* ♦

Proof. Let $(f :\subseteq \mathbb{R}^n \to \mathbb{R}^m) \in \text{AS}(\Upsilon, \Omega, \Theta)$ where Υ, Ω Θ are polynomials which we assume, without loss of generability, to be increasing functions of theirs inputs. Apply Definition 4.3.9 (Analog strong computability) to get d, h and g.

Let $e = 1 + d + m$, $x \in C^0(\mathbb{R}_+, \mathbb{R}^n)$, $\mu \in C^0(\mathbb{R}_+, \mathbb{R}_+)$, $(v_0, y_0, z_0) \in \mathbb{R}^e$, $(e_v, e_y, e_z) \in C^0(\mathbb{R}_+, \mathbb{R}^e)$ and consider the following system:

$$\begin{cases} v(0) = v_0 \\ y(0) = y_0 \\ z(0) = z_0 \end{cases} \qquad \begin{cases} v'(t) = \operatorname{sample}_{[0,1],4}(t, \mu^*(t), v(t), \mu(t) + \ln\Delta + 7) + e_v(t) \\ y'(t) = \operatorname{sample}_{[1,2],4}(t, \mu^*(t), y(t), g(x(t), v(t))) \\ \qquad\qquad + \operatorname{plil}_{[2,3],4}(t, \mu^*(t), A(t)h(y(t))) + e_y(t) \\ z'(t) = \operatorname{sample}_{[3,4],4}(t, \mu^*(t), z(t), y_{1..m}(t)) + e_z(t) \end{cases}$$

where

$$\Delta = 5 \qquad \Delta' = \ln\Delta + 10$$

$$\mu^*(t) = \mho^*(1 + \operatorname{norm}_{\infty,1}(x(t)), v(t) + 4)$$

$$A(t) = 1 + \Omega(1 + \operatorname{norm}_{\infty,1}(x(t)), v(t))$$

$$\Lambda^*(\alpha, \mu) = \Theta^*(\alpha, \mu) = \mho^*(\alpha, \mu + \Delta')$$

$$\mho^*(\alpha, \mu) = \mu + \ln\Delta + \Theta(\alpha, \mu) + \ln q(\alpha + \mu)$$

Let $I = [a, b]$ and assume there exists $\bar{x} \in \operatorname{dom} f$ and $\check{\mu}, \hat{\mu} \in \mathbb{R}_+$ such that for all $t \in I$, $\mu(t) \in [\check{\mu}, \hat{\mu}]$, $\|x(t) - \bar{x}\| \leqslant e^{-\Lambda^*(\|\bar{x}\|, \hat{\mu})}$ and $\int_a^b \|e(u)\|\,du \leqslant e^{-\Theta^*(\|\bar{x}\|, \hat{\mu})}$. Apply Proposition 2.2.4 (Modulus of continuity) to g to get $q \in \mathbb{K}[\mathbb{R}]$, without loss of generality we can assume that q is an increasing function and $q \geqslant 1$. We will use Lemma 2.6.18 (Norm function) to get that $\operatorname{norm}_{\infty,1}(x(t)) + 1 \geqslant \|\bar{x}\|$ because $\|x(t) - \bar{x}\| \leqslant 1$. Also note that $\mu^*, \Theta^*, \Lambda^*$ are increasing functions of their arguments. Let $n \in \mathbb{N}$ such that $[4n, 4n + 4] \subseteq I$ and $t \in [4n, 4n + 4]$. We will first analyse the variable v, note that the analysis is extremely rough to simplify the proof.

- **if** $t \in [4n, 4n + 1]$ **then** $\mu^*(t) \geqslant 0$ so apply Lemma 4.4.14 (Sample and hold) to get that $v(4n + 1) \in [\check{\mu} + \ln\Delta + 7 - \varepsilon, \hat{\mu} + \ln\Delta + 7 + \varepsilon]$ where $\varepsilon \leqslant 2e^{-0} + \int_{4n}^{4n+1} |e_v(u)|du \leqslant 3$ because $\int_a^b \|e(t)\| \leqslant 1$. Define $\bar{v} = v(4n + 1)$, then $\bar{v} \in [\check{\mu} + \ln\Delta + 4, \hat{\mu} + \underbrace{\ln\Delta + 10}_{=\Delta'}]$

- **if** $t \in [4n+1, 4n+4]$ **then** $\mu^*(t) \geqslant 0$ so apply Lemma 4.4.14 (Sample and hold) to get that $|\nu'(t)| \leqslant e^{-0} + \int_{4n+1}^{t} |e_\nu(u)| du$ and thus $|\nu(t) - \bar{\nu}| \leqslant (t - 4n - 1) + \int_{4n+1}^{t} \|e(u)\| \, du \leqslant 4$ because $\int_a^b \|e(t)\| \leqslant 1$. In other words $\nu(t) \in [\bar{\nu} - 4, \bar{\nu} + 4]$.

Furthermore for $t \in [4n+1, 4n+4]$ we have:

$$\mu^*(t) \geqslant \Theta^*(1 + \operatorname{norm}_{\infty,1}(x(t)), \nu(t) + 4) \geqslant \mho^*(\|\bar{x}\|, \bar{\nu})$$

It will also be useful to note that:

$$\begin{aligned}\Lambda^*(\|\bar{x}\|, \hat{\mu}) = \Theta^*(\|\bar{x}\|, \hat{\mu}) &\geqslant \mho^*(\|\bar{x}\|, \hat{\mu} + \Delta') \\ &\geqslant \mho^*(\|\bar{x}\|, \bar{\nu})\end{aligned}$$

We can now analyze y using this property:

- **if** $t \in [4n+1, 4n+2]$ **then** $|\nu'(t)| \leqslant e^{-\mu^*(t)} + |e_\nu(t)|$ thus $|\nu(t) - \bar{\nu}| \leqslant e^{-\mho^*(\|\bar{x}\|, \bar{\nu})} + \int_{4n+1}^{4n+2} |e_\nu(u)| du$. Furthermore $\sup_{[4n+1,4n+2]} \|x\| \leqslant \|\bar{x}\| + 1$, thus:

  $$\begin{aligned}\|g(\bar{x}, \bar{\nu}) - g(x(t), \nu(t))\| &\leqslant \max(|\nu(t) - \bar{\nu}|, \|x(t) - \bar{x}\|) q(\max(\|\bar{x}\|, |\bar{\nu}|)) \\ &\leqslant \max\left(e^{-\Theta^*(\|\bar{x}\|, \hat{\mu})} + e^{-\mho^*(\|\bar{x}\|, \bar{\nu})}, e^{-\Lambda^*(\|\bar{x}\|, \hat{\mu})}\right) q(\|\bar{x}\| + \bar{\nu}) \\ &\leqslant 2e^{-\Theta(\|\bar{x}\|, \bar{\nu}) - \ln \Delta}\end{aligned}$$

  Also note that $\left\|y'(t) - \operatorname{sample}_{[1,2],4}(t, \mu^*(t), y(t), g(x(t), \nu(t)))\right\| \leqslant e^{-\mu^*(t)}$ by Lemma 2.6.26 ("periodic low-integral-low"). So we can apply Lemma 4.4.14 (Sample and hold) to get that $\|y(4n+2) - g(\bar{x}, \bar{\nu})\| \leqslant 2e^{-\Theta(\|\bar{x}\|, \bar{\nu}) - \ln \Delta} + e^{-\mho^*(\|\bar{x}\|, \bar{\nu})} + \int_{4n+1}^{4n+2} \|e(u)\| \, du \leqslant 4e^{-\Theta(\|\bar{x}\|, \bar{\nu}) - \ln \Delta}$.

- **if** $t \in [4n+2, 4n+3]$ **then** apply Lemma 4.4.14 (Sample and hold) and Lemma 2.6.26 ("periodic low-integral-low") to get ϕ such that $\int_{4n+2}^{4n+3} \phi(u) du \geqslant 1$ and $\|y'(t) - \phi(t) A(t) h(y(t))\| \leqslant e^{-\mu^*(t)} + \left\|e_y(t)\right\|$. Define $\psi(t) = \int_{4n+2}^{t} \phi(u) A(u) du$ then $\psi(4n+3) \geqslant \Omega(\|\bar{x}\|, \bar{\nu})$ since $A(u) \geqslant \Omega(\|\bar{x}\|, \bar{\nu})$ for $u \in [4n+2, 4n+3]$. Apply Lemma 2.4.4 (Perturbed time-scaling) over $[4n+2, 4n+3]$ to get that $y(t) = w(\psi(t))$ where w satisfies $w(0) = y(4n+2)$ and $w'(\xi) = h(w(\xi)) + \tilde{e}(\xi)$ where $\tilde{e} \in C^0(\mathbb{R}_+, \mathbb{R}^d)$ satisfies $\int_0^{\psi(t)} \|\tilde{e}(\xi)\| \, d\xi = \int_{4n+2}^{t} \left\|e_y(u)\right\| du \leqslant e^{-\Theta^*(\|\bar{x}\|, \hat{\mu})} \leqslant e^{-\Theta(\|\bar{x}\|, \bar{\nu}) - \ln \Delta}$. Furthermore, $\|w(0) - g(\bar{x}, \bar{\nu})\| \leqslant 4e^{-\Theta(\|\bar{x}\|, \bar{\nu}) - \ln \Delta}$ from the result above. In other words, $w(0) = g(\bar{x}, \bar{\nu}) + \tilde{e}_0$ and $w'(t) = g(w(t)) + \tilde{e}(t)$ where $\|\tilde{e}_0\| + \int_0^{\psi(t)} \|e(u)\| \, du \leqslant 5e^{-\Theta(\|\bar{x}\|, \bar{\nu}) - \ln \Delta} \leqslant e^{-\Theta(\|\bar{x}\|, \bar{\nu})}$ because $\Delta \geqslant 5$. Apply Definition 4.3.9 (Analog strong computability) to get that $\|w_{1..m}(\psi(4n+3)) - f(\bar{x})\| \leqslant e^{-\bar{\nu}}$ since $\psi(4n+3) \geqslant \Omega(\|\bar{x}\|, \bar{\nu})$.

- **if** $t \in [4n+3, 4n+4]$ **then** $\|y'(t)\| \leqslant e^{-\mu^*(t)} + \left\|e_y(t)\right\|$ thus $\|y(t) - y(4n+3)\| \leqslant e^{-\mho^*(\|\bar{x}\|, \bar{\nu})} + \int_{4n+3}^{t} \left\|e_y(u)\right\| du \leqslant 2e^{-\bar{\nu}}$ so $\|y_{1..m}(t) - f(\bar{x})\| \leqslant 3e^{-\bar{\nu}}$.

Note that the above reasoning is also true for the last segment $[4n, b] \subseteq I$ in which case the result only applies up to time b of course. In other words, the results apply as long as $t \in [4n, 4+4] \cap I$ and $4n \geqslant a$. From this we conclude that if $t \in [a+4, b] \cap [4n+3, 4n+3]$ for some $n \in \mathbb{N}$ then $\|y_{1..m}(t) - f(\bar{x})\| \leqslant 3e^{-\bar{\nu}}$. Apply Lemma 4.4.14 (Sample and hold) to get, using that $\bar{\nu} \geqslant \check{\mu} + \ln \Delta$ and $\Delta \geqslant 5$, that for all $t \in [a+5, b]$:

$$\begin{aligned}\|z(t) - f(\bar{x})\| &\leqslant 3e^{-\bar{\nu}} + e^{-\mho^*(\|\bar{x}\|, \bar{\nu})} + \int_{t-5}^{t} \|e(u)\| \, du \leqslant 5e^{-\bar{\nu}} \\ &\leqslant e^{-\check{\mu}}\end{aligned}$$

To complete the proof, we must also analyze the norm of the system. As a shorthand, we introduce the following notation:

$$\operatorname{int}^+_\delta \alpha(t) = \int_{\max(0,t-\delta)}^{t} \alpha(u)du$$

Apply Lemma 4.4.14 (Sample and hold) to get that:

$$\begin{aligned}|v(t)| &\leqslant 2 + \int_{\max(0,t-5)}^{t} |e_v(u)|du + \max\left(|v_0|\mathbb{1}_{[0,4]}(t), \sup\nolimits_5|\mu + \ln\Delta + 7|(t)\right)\\ &\leqslant \operatorname{poly}\left(|v_0|\mathbb{1}_{[0,5]}(t) + \operatorname{int}^+_5 |e_v|(t), \sup\nolimits_5\mu(t)\right)\end{aligned}$$

The analysis of y is a bit more painful, as it uses both results about the sampling function and the strongly-robust system we are simulating. Let $n \in \mathbb{N}$, and $t \in [4n, 4n+4]$:

- **if** $t \in [4n, 4n+1]$ **then** apply Lemma 4.4.14 (Sample and hold) and Lemma 2.6.26 ("periodic low-integral-low") to get, using that $\mu(t) \geqslant 0$, that $\|y'(t)\| \leqslant 2 + \|e(t)\|$ and thus $\|y(t) - y(4n)\| \leqslant 2 + \int_{4n}^{t} \|e(u)\| du$.
- **if** $t \in [4n+1, 4n+2]$ **then** using the result on v, $\|g(x(t), v(t))\| \leqslant \sup_{[4n+1,t]} \operatorname{poly}(\|x\|, v) \leqslant \operatorname{poly}\left(|v_0|\mathbb{1}_{[0,5]}(t) + \operatorname{int}^+_6 \|e\|(t), \sup_6\mu(t), \sup_1 \|x\|(t)\right)$. Apply Lemma 4.4.14 (Sample and hold) and Lemma 2.6.26 ("periodic low-integral-low") to get, using that $\mu(t) \geqslant 0$ and the result on v, that:

$$\begin{aligned}\|y(4n+2)\| &\leqslant \sup_{[4n+1,4n+2]} \|g(x,v)\| + 2 + \int_{4n+1}^{4n+2} \|e(u)\| du\\ &\leqslant \operatorname{poly}\left(|v_0|\mathbb{1}_{[0,5]}(4n+2) + \operatorname{int}^+_6 \|e\|(4n+2), \sup\nolimits_6\mu(4n+2), \sup\nolimits_1 \|x\|(4n+2)\right)\end{aligned}$$

  and also that:

$$\begin{aligned}\|y(t)\| &\leqslant \max\left(\sup_{[4n+1,t]} \|g(x,v)\| + 2, \|y(4n+1)\|\right) + \int_{4n+1}^{t} \|e(u)\| du\\ &\leqslant \operatorname{poly}\left(|v_0|\mathbb{1}_{[0,5]}(t) + \operatorname{int}^+_6 \|e\|(t), \sup\nolimits_6\mu(t), \sup\nolimits_1 \|x\|(t), \|y(4n)\|\right)\end{aligned}$$

- **if** $t \in [4n+2, 4n+3]$ **then** apply Lemma 4.4.14 (Sample and hold), Lemma 2.6.26 ("periodic low-integral-low"), Lemma 2.4.4 (Perturbed time-scaling) and Definition 4.3.9 (Analog strong computability) to get that $\|y(t)\| \leqslant \Upsilon(0, 0, \hat{e}(\hat{A}(t)), \hat{A}(t))$ where $\hat{A}(t) = \int_{4n+2}^{t} A(u)du$ and $\hat{e}(\hat{A}(t)) = \|y(4n+2) - g(0,0)\| + \int_{4n+2}^{t} 1 + \|e(u)\| du$. Since Ω is a polynomial, and using the result on v, we get that:

$$\begin{aligned}\hat{A}(t) &\leqslant \sup_{[4n+2,t]} \operatorname{poly}(\|x\|, |v|)\\ &\leqslant \operatorname{poly}\left(|v_0|\mathbb{1}_{[0,5]}(t) + \operatorname{int}^+_6 \|e\|, \sup\nolimits_6\mu(t), \sup\nolimits_1 \|x\|(t)\right)\end{aligned}$$

  and using that $4n+2 \leqslant t \leqslant 4n+3$:

$$\begin{aligned}\|y(4n+2) - g(0,0)\| &\leqslant \|y(4n+2)\| + \|g(0,0)\|\\ &\leqslant \operatorname{poly}\left(|v_0|\mathbb{1}_{[0,5]}(t) + \operatorname{int}^+_6 \|e\|, \sup\nolimits_7\mu(t), \sup\nolimits_2 \|x\|(t)\right)\end{aligned}$$

  And since Υ is a polynomial, we conclude that:

$$\|y(t)\| \leqslant \operatorname{poly}\left(|v_0|\mathbb{1}_{[0,5]}(t) + \operatorname{int}^+_6 \|e\|(t), \sup\nolimits_7\mu(t), \sup\nolimits_2 \|x\|(t)\right)$$

- **if** $t \in [4n+3, 4n+4]$ **then** apply Lemma 4.4.14 (Sample and hold) and Lemma 2.6.26 ("periodic low-integral-low") to get, using that $\mu(t) \geqslant 0$, that $\|y'(t)\| \leqslant 2 + \|e(t)\|$ and thus $\|y(t) - y(4n+3)\| \leqslant 2 + \int_{4n+3}^{t} \|e(u)\| \, du$.

From this analysis we can conclude that for all $t \in [0,2]$:

$$\begin{aligned}\|y(t)\| &\leqslant \operatorname{poly}\left(|v_0|\mathbb{1}_{[0,5]}(t) + \operatorname{int}_6^+ \|e\|(t), \sup\nolimits_6 \mu(t), \sup\nolimits_1 \|x\|(t), \|y(0)\|\right)\\ &\leqslant \operatorname{poly}\left(|v_0| + \operatorname{int}_6^+ \|e\|(t), \sup\nolimits_6 \mu(t), \sup\nolimits_1 \|x\|(t), \|y_0\|\right)\end{aligned}$$

and for all $n \in \mathbb{N}$ and $t \in [4n+2, 4n+6]$:

$$\|y(t)\| \leqslant \operatorname{poly}\left(|v_0|\mathbb{1}_{[0,5]}(t) + \operatorname{int}_9^+ \|e\|(t), \sup\nolimits_9 \mu(t), \sup\nolimits_4 \|x\|(t)\right)$$

Putting everything together, we get for all $t \in \mathbb{R}_+$:

$$\|y(t)\| \leqslant \operatorname{poly}\left(\|y_0, v_0\| \mathbb{1}_{[0,5]}(t) + \operatorname{int}_9^+ \|e\|(t), \sup\nolimits_9 \mu(t), \sup\nolimits_4 \|x\|(t)\right)$$

Finally apply Lemma 4.4.14 (Sample and hold) to get the a similar bound on z and thus on the entire system. ■

An interesting detail about the previous theorem is that the obtained extreme system always converges in constant time. This is not very surprising since we can often rescale the time. However this is not entirely trivial either because the system is not autonomous ($y'(t) = g(t, \ldots)$). Since this can be a very useful property for proofs, we show that this kind of rescaling is always possible.

Lemma 4.4.16 (AXP time rescaling)**:** *If $f \in$ AXP then there exists polynomials $\Upsilon, \Lambda, \Theta$ and a constant polynomial Ω such that $f \in \mathrm{AX}(\Upsilon, \Omega, \Lambda, \Theta)$.* ♦

Proof. We go for the shortest proof: we will show that AXP $\subseteq$ AWP and use Theorem 4.3.7 (Weak $\subseteq$ robust) then Theorem 4.3.12 (Robust $\subseteq$ strong) followed by Theorem 4.4.15 (Strong $\subseteq$ extreme) which proves exactly our statement.

The proof that AXP $\subseteq$ AWP is next to trivial because the extreme system and some given input and precision, we can simply store the input and precision into some variables and feed them into the system. We make the system autonomous by using a variable to store the time.

Let $(f :\subseteq \mathbb{R}^n \to \mathbb{R}^m) \in \mathrm{AX}(\Upsilon, \Omega, \Lambda, \Theta)$, apply Definition 4.4.2 (Extreme computability) to get δ, d and g. Let $x \in \operatorname{dom} f$ and $\mu \in \mathbb{R}_+$, and consider the following system:

$$\begin{cases} x(0) = x \\ \mu(0) = \mu \\ \tau(0) = 0 \\ y(0) = 0 \end{cases} \qquad \begin{cases} x'(t) = 0 \\ \mu'(t) = 0 \\ \tau'(t) = 1 \\ y'(t) = g(t, y(t), x(t), \mu(t)) \end{cases}$$

Clearly he system of the form $z(0) = h(x,\mu)$ and $z'(t) = H(z(t))$ where h and H belong to GPVAL (and are defined over the entire space). Apply the definition to get that:

$$\|y(t)\| \leqslant \Upsilon(\|x\|, \mu, 0)$$

And thus the entire system in bounded by a polynomial in $\|x\|, \mu$ and t. Furthermore, if $t \geqslant \Omega(\|x\|, \mu)$ then $\|y_{1..m}(t) - f(x)\| \leqslant e^{-\mu}$. To conclude the proof, we need to rewrite the system as a PIVP using Proposition 2.2.2 (Generable ODE rewriting). ■

4.4.V Online computability

The notion of extreme computability is very strong but it is too complicated in many cases, as well as hard to understand completely. To this end, we introduce simpler notion of online computability. This notion can be seen as the online counterpart of Definition 4.2.1 (Analog computability). We do not repeat Remark 4.4.5 (Regularity of the input) which also applies to this notion.

Definition 4.4.17 (Online computability)**:** Let $n, m \in \mathbb{N}$, $f :\subseteq \mathbb{R}^n \to \mathbb{R}^m$ and $\Upsilon, \Omega, \Lambda : \mathbb{R}_+^2 \to \mathbb{R}_+$. We say that f is $(\Upsilon, \Omega, \Lambda)$-online-computable if and only if there exists $\delta \geqslant 0$, $d \in \mathbb{N}$ and $p \in \mathbb{K}^d[\mathbb{R}^d \times \mathbb{R}^n]$ and $y_0 \in \mathbb{K}^d$ such that for any $x \in C^0(\mathbb{R}_+, \mathbb{R}^n)$, there exists (a unique) $y : \mathbb{R}_+ \to \mathbb{R}^d$ satisfying for all $t \in \mathbb{R}_+$:

- $y(0) = y_0$ and $y'(t) = p(y(t), x(t))$
- $\|y(t)\| \leqslant \Upsilon\big(\sup_\delta \|x\|(t), t\big)$
- For any $I = [a, b]$, if there exists $\bar{x} \in \operatorname{dom} f$ and $\bar{\mu} \geqslant 0$ such that for all $t \in I$, $\|x(t) - \bar{x}\| \leqslant e^{-\Lambda(\|\bar{x}\|, \bar{\mu})}$ then $\|y_{1..m}(u) - f(\bar{x})\| \leqslant e^{-\bar{\mu}}$ whenever $a + \Omega(\|\bar{x}\|, \bar{\mu}) \leqslant u \leqslant b$.

We denote by $\mathrm{AO}(\Upsilon, \Omega, \Lambda)$ the set of $(\Upsilon, \Omega, \Lambda)$-online-computable functions. ♦

Definition 4.4.18 (Analog online poly-time)**:** Denote by AOP the set of (poly, poly, poly)-online-computable functions. ♦

In the previous section, we made a big jump from strong computability to extreme computability. In this section, we will show that extreme computability is stronger than online computability. Although this doesn't come as a surprise, there is a small subtlety because extreme computability allows systems of the form $y' = g(y, x)$ where g is generable, whereas online computability requires strictly a PIVP. We saw in Section 2.2 (Stability properties) that any generable ODE can be rewritten as a PIVP but in the case of external inputs (like x) it requires at least C^1 smoothness. Unfortunately, Definition 4.4.17 (Online computability) only requires a C^0 input so we cannot apply this result directly. Example 4.4.19 (C^0 input is weak) exhibit this problem on a toy ODE. To work around this problem, the idea is to use to extra variables to "smooth" the input and is detailed in Example 4.4.20 (C^0 input workaround).

Example 4.4.19 (C^0 input is weak)**:** Suppose we want to show that tanh is computable and consider the following system, for $x \in C^0(\mathbb{R}_+, \mathbb{R}_+)$:

$$y(0) = 0 \qquad y'(t) = \tanh(x(t)) - y(t)$$

It can be seen that if $x(t)$ converges to x then $y(t)$ converges to $\tanh(x)$. However this system does not satisfies Definition 4.4.17 (Online computability) because tanh is obviously not a polynomial. However we know that $\tanh' = 1 - \tanh^2$ so we can introduce a new variable $z(t) = \tanh(x(t))$. A simple computation shows that $z(0) = \tanh(x(0))$ and $z'(t) = x'(t)(1 - z(t)^2)$. Rewriting this system we get:

$$y(0) = 0 \quad z(0) = \tanh(x(0)) \qquad y'(t) = z(t) - y(t) \quad z'(t) = x'(t)(1 - z(t)^2)$$

The system becomes polynomial but we have two new problems: x needs to be C^1 and the system is polynomial in $x'(t)$ (instead of $x(t)$), furthermore the initial condition also depends on $x(0)$. ♦

Example 4.4.20 (C^0 input workaround)**:** Continuing Example 4.4.19 (C^0 input is weak), we want to work around the C^1 requirement for x and also remove the dependency in $x(0)$. The idea is to introduce a new variable x^* that will replace x. This new variable will be part of the system and only depends on x. Because x is C^0, x^* will be C^1 by the Cauchy-Liptchiz theorem and we can even choose the initial value. To keep the properties of the system, we arrange so that x^* converges to the same value as x, and adapts to the changes of x. Since x^* only has access to x and not x', it will lag a bit behind x but for convergence this is not an issue. We illustrate this approach on the previous system and define x^* as follows:

$$x^*(0) = 0 \qquad x^{*\prime}(t) = \operatorname{reach}(\phi(t), x^*(t), x(t))$$

Notice that, by definition, reach is a polynomial: this is crucial for this approach to work. Furthermore, by Lemma 4.4.12 (Reach), if $x(t)$ converges to x then $x^*(t)$ also converges to x. However there are a couple details that need to be worked with:

- we have to choose a ϕ that matches the required input convergence rate Λ of the system;
- in the previous system, the value of $\|y(t)\|$ depended only on $\|x(t)\|$ whereas in the new system it depends on $\|x^*(t)\|$ which in turns depends on $\sup_{u\in[t,t-\tau(t)]\cap\mathbb{R}_+} \|x(u)\|$. The time $\tau(t)$ corresponds to the time needed to make $\int_{\tau(t)}^{t} \phi(u)du \geqslant 1$ in Lemma 4.4.12 (Reach). For any reasonable definition of ϕ we can arrange so that $\tau(t) \leqslant \tau^*$ a constant.

♦

Theorem 4.4.21 (Extreme ⊆ online)**:** AXP ⊆ AOP ♦

Proof. Apart from the issue of the input, the system is quite intuitive: we constantly feed the extreme system with the (smoothed) input and some precision. By increasing the precision with time, we ensure that the system will converge when the input is stable. However there is a small catch: over a time interval I, if we change the precision within a range $[\check{\mu}, \hat{\mu}]$ then we must provide the extreme system with precision based on $\hat{\mu}$ in order to get precision $\check{\mu}$. Since the extreme system takes time $\Omega(\|x\|, \hat{\mu})$ to compute, we need arrange so that the requested precision doesn't change too much over periods of this duration to make things simpler. We will use to our advantage that Ω can always be assumed to be a constant.

Let $(f :\subseteq \mathbb{R}^n \to \mathbb{R}^m) \in \mathrm{AX}(\Upsilon, \Omega, \Lambda, \Theta)$ where $\Upsilon, \Omega, \Lambda$ and Θ are polynomials, which we can assume to be increasing functions of their arguments. Apply Lemma 4.4.16 (AXP time rescaling) to get $\omega > 0$ such that for all $\alpha \in \mathbb{R}^n, \mu \in \mathbb{R}_+$:

$$\Omega(\alpha, \mu) = \omega$$

Apply Definition 4.4.2 (Extreme computability) to get δ, d and g. Define:

$$\tau = \omega + 2 \qquad \delta' = \max(\delta, \tau + 1)$$

Let $x \in C^0(\mathbb{R}_+, \mathbb{R}^n)$ and consider the following systems:

$$\begin{cases} x^*(0) = 0 \\ y(0) = 0 \\ z(0) = 0 \end{cases} \qquad \begin{cases} x^{*\prime}(t) = \operatorname{reach}(\phi(t), x^*(t), x(t)) \\ y'(t) = g(t, y(t), x^*(t), \mu(t)) \\ z'(t) = \operatorname{sample}_{[\omega+1,\omega+2],\tau}(t, \mu(t), z(t), y_{1..m}(t)) \end{cases}$$

where

$$\phi(t) = \ln 2 + \mu(t) + \Lambda^*(2 + x_1(t)^2 + \cdots + x_n(t)^2, \mu(t)) \qquad \mu(t) = \frac{t}{\tau}$$

Let $t \geqslant 1$, since $\phi \geqslant 1$ then Lemma 4.4.12 (Reach) gives:

$$\|x^*(t)\| \leqslant \sup_1 \|x\|(t) + e^{-\int_{t-1}^{t}\phi(u)du} \leqslant \sup_1 \|x\|(t) + 1$$

Also for $t \in [0,1]$ we get that:

$$\|x^*(t)\| \leqslant \sup_{[0,t]} \|x\|$$

This proves that $\|x^*(t)\| \leqslant \sup_1 \|x\|(t) + 1$ for all $t \in \mathbb{R}_+$. From this we deduce that:

$$\begin{aligned}\|y(t)\| &\leqslant \Upsilon(\sup_\delta \|x^*\|(t), \sup_\delta \mu(t), 0)\\ &\leqslant \operatorname{poly}(\sup_\delta \|x\|(t), t)\end{aligned}$$

Apply Lemma 4.4.14 (Sample and hold) to get that:

$$\begin{aligned}\|z(t)\| &\leqslant 2 + \sup_{\tau+1} \|y\|(t)\\ &\leqslant \operatorname{poly}(\sup_{\delta'} \|x\|(t), t)\end{aligned}$$

Let $I = [a,b]$ and assume there exists $\bar{x} \in \operatorname{dom} f$ and $\bar{\mu}$ such that for all $t \in I$, $\|x(t) - \bar{x}\| \leqslant e^{-\Lambda(\|\bar{x}\|,\bar{\mu})}$. Note that $2 + \sum_{i=1}^n x_i(t)^2 \geqslant 1 + \|x(t)\| \geqslant \|\bar{x}\|$ for all $t \in I$. Let $n \in \mathbb{N}$ such that $n \geqslant \bar{\mu} + \ln 2$ and $[n\tau, (n+1)\tau] \subseteq I$. Note that $\mu(t) \in [n, n+1]$ for all $t \in I_n$. Apply Lemma 4.4.12 (Reach), using that $\phi \geqslant 1$, to get that for all $t \in [n\tau + 1, (n+1)\tau]$:

$$\begin{aligned}\|x^*(t) - \bar{x}\| &\leqslant e^{-\Lambda^*(\|\bar{x}\|,n)} + e^{-\int_{n\tau}^{t}\phi(u)du} \leqslant 2e^{-\Lambda^*(\|\bar{x}\|,n)}\\ &\leqslant e^{-\Lambda(\|\bar{x}\|,\bar{\mu}+\ln 2)}\end{aligned}$$

Using the definition of extreme computability, we get that for all $t \in [n\tau + 1 + \omega, (n+1)\tau] = [n\tau + \omega + 1, n\tau + \omega + 2]$:

$$\|y_{1..m} - f(\bar{x})\| \leqslant e^{-\bar{\mu}+\ln 2}$$

Define $J = [a + (1 + \bar{\mu} + \ln 2)\tau, b] \subseteq I$. Assume that $t \in J \cap [n\tau + 1, (n+1)\tau]$ for some $n \in \mathbb{N}$, then we must have $(n+1)\tau \geqslant (1 + \bar{\mu} + \ln 2)\tau$ and thus $n \geqslant \bar{\mu} + \ln 2$ so we can apply the above reasoning to get that $\|y_{1..m}(t) - f(x)\| \leqslant e^{-\bar{\mu}+\ln 2}$. Furthermore, we also have $\mu(t) \geqslant \frac{(1+\bar{\mu}+\ln 2)\tau}{\tau} \geqslant \bar{\mu} + \ln 2$ for all $t \in J$. Apply Lemma 4.4.14 (Sample and hold) to conclude that for any $t \in [a + \tau + \bar{\mu} + \ln 2 + \tau + 1, b]$, we have $\|z(t) - f(x)\| \leqslant 2e^{-\bar{\mu}+\ln 2} \leqslant e^{-\bar{\mu}}$.

To conclude the proof, we need to rewrite the system as a PIVP using Proposition 2.2.2 (Generable ODE rewriting). Note that this works because we only rewrite the variable y, and doing so we require that x^* be a C^1 function (which is the case) and the new initial variable will depend on $x^*(0) = 0$ which is constant. ■

The introduction of online computability was not only useful to give a simpler class than extreme computability, it also makes easier to see that in fact online computable functions are computable. This result closes the circle and shows that all our polytime classes are equivalent.

Theorem 4.4.22 (Online ⊆ computable)**:** AOP ⊆ AP. ♦

Proof. The proof is trivial: given x, we store it in a variable and run the online system. Since the input has no error, we can directly apply the definition to get that the online system converges.

Let $(f :\subseteq \mathbb{R}^n \to \mathbb{R}^m) \in \mathrm{AO}(\Upsilon, \Omega, \Lambda)$. Apply Definition 4.4.17 (Online computability) to get δ, d, p and y_0. Let $x \in \operatorname{dom} f$ and consider the following system:

$$\begin{cases} x(0) = x \\ y(0) = y_0 \end{cases} \qquad \begin{cases} x'(t) = 0 \\ y'(t) = p(y(t), x(t)) \end{cases}$$

We immediately get that:

$$\|y(t)\| \leqslant \Upsilon(\sup_\delta \|x\|(t), t) \leqslant \Upsilon(\|x\|, t)$$

Let $\mu \in \mathbb{R}_+$ and let $t \geqslant \Omega(\|x\|, \mu)$, then apply Definition 4.4.17 (Online computability) to $I = [0, t]$ to get that $\|y_{1..m}(t) - f(x)\| \leqslant e^{-\mu}$ since $\|x(t) - x\| = 0$. ■

4.5 Summary

In the previous sections, we gave give several notions of analog computability. Those notions are different compromises depending on what we were trying to achieve. We saw several inclusion results that show that in fact they are all equivalent.

Theorem 4.5.1 (Main equivalence)**:** ALP = AP = AWP = ARP = ASP = AXP = AOP. ♦

Proof. Apply Theorem 4.3.7, Theorem 4.3.12, Theorem 4.4.15, Theorem 4.4.21, Theorem 4.4.22 to get that:

$$\text{AP} \subseteq \text{AWP} \subseteq \text{ARP} \subseteq \text{ASP} \subseteq \text{AXP} \subseteq \text{AOP} \subseteq \text{AP}$$

Apply Theorem 4.2.15 to get that AP = ALP. ■

Proposition 4.5.2 (Polynomial versus generable)**:** *The classes* AP *and* AWP *are the same if we only assume that $p, q \in$* GPVAL *in Definition 4.2.1 (Analog computability) and Definition 4.2.7 (Analog weak computability) respectively.* ♦

Proof. If $f \in$ AP or $f \in$ AWP with $p, q \in$ GPVAL, apply Proposition 2.2.2 (Generable ODE rewriting) to rewrite the system so that only $q \in$ GPVAL and p becomes a polynomial. Use Remark 4.3.8 (Polynomial versus generable) to conclude that this function thus to ARP and use Theorem 4.5.1 (Main equivalence) to conclude. ■

The tables below summarize in an informal way the computability notions that we introduced. In the entire table, we consider a given function $f :\subseteq \mathbb{R}^n \to \mathbb{R}^m$ and some given bounds $\Upsilon, \Omega, \Theta, \Lambda$. We begin by an informal reformulation of each definition, we also add for each item a list of the relevant symbols, to help the reader match the formal definitions with the informal ones. We also put some comments on what the definitions *do not guarantee* to help identify the compromises between the definitions.

Informal convention for the symbols

Sym.	**Name**	**Convention**
Υ	Space	Υ provides a bound on the "space" used by the system, that is $\|y(t)\|$; bounding the space is crucial for the time complexity to make sense
Ω	Time	Ω gives the time needed by the system to compute the result with a certain precision; times only makes sense with respect to a space bound Υ
Λ	Modulus	Λ acts as the modulus of continuity for online systems: it gives the maximum allowed error on the input to reach a certain precision
Θ	Error	Θ gives the maximum amount of perturbation allowed in the system during a computation to reach a certain precision

Informal reformulation of the definitions

Class	Definition	Sym.
$\mathrm{AC}(\Upsilon, \Omega)$	**Computability: the most basic notion**	
	Given input $x \in \operatorname{dom} f$:	
	• y satisfies a PIVP and $y(0)$ depends on x	► p, q
	• $y(t)$ is bounded by a function of $\|x\|$ and t	► Υ
	• $y_{1..m}(t)$ converges quickly to $f(x)$	► Ω
	♦ Undefined behavior if $x \notin \operatorname{dom} f$	
	♦ Undefined behavior if PIVP is perturbed	
$\mathrm{AL}(\Omega)$	**Length computability: length-based complexity**	
	Given input $x \in \operatorname{dom} f$:	
	• y satisfies a PIVP and $y(0)$ depends on x	► p, q
	• $y_{1..m}(t)$ converges quickly to $f(x)$	► Ω
	• convergence rate depends on the length of the curve	
	♦ Undefined behavior if $x \notin \operatorname{dom} f$	
	♦ Undefined behavior if PIVP is perturbed	
$\mathrm{AW}(\Upsilon, \Omega)$	**Weak computability: computability with given precision**	
	Given input $x \in \operatorname{dom} f$ and $\mu \in \mathbb{R}_+$:	
	• y satisfies a PIVP and $y(0)$ depends on x and μ	► p, q
	• $y(t)$ is bounded by a function of $\|x\|$, μ and t	► Υ
	• $y_{1..m}(t)$ converges quickly to $f(x) \pm e^{-\mu}$	► Ω
	♦ Undefined behavior if $x \notin \operatorname{dom} f$	
	♦ Undefined behavior if PIVP is perturbed	
$\mathrm{AR}(\Upsilon, \Omega, \Theta)$	**Robust computability: weak with small pertubations**	
	Given input $x \in \operatorname{dom} f$, precision μ and errors e_0, $e(t)$:	
	• assume e_0 and $\int e(t)dt$ are very small	► Θ
	• y satisfies a perturbed PIVP and $y(0)$ depends on x and μ	► p, q
	• $y(t)$ is bounded by a function of $\|x\|$ and μ	► Υ
	• $y_{1..m}(t)$ converges quickly to $f(x) \pm e^{-\mu}$	► Ω
	♦ Undefined behavior if $x \notin \operatorname{dom} f$	
	♦ Undefined behavior if perturbation are not small enough	
$\mathrm{AS}(\Upsilon, \Omega, \Theta)$	**Strong computability: robust to any pertubations**	
	Given input $x \in \mathbb{R}^n$, precision μ and errors e_0, $e(t)$:	
	• y satisfies a perturbed PIVP and $y(0)$ depends on x and μ	► g, h
	• $y(t)$ is bounded by a function of $\|x\|$, μ, t and $e_0 + \int e(t)dt$	► Υ
	• if $x \in \operatorname{dom} f$ and errors are small, $y_{1..m}(t)$ converges to $f(x) \pm e^{-\mu}$	► Θ, Ω
$\mathrm{AO}(\Upsilon, \Omega, \Lambda)$	**Online computability: real-time computation**	
	Given input $x(t)$:	
	• y satisfies a PIVP and $y'(t)$ depends on $x(t)$	► p
	• $y(t)$ is bounded by a function of $\|x(t)\|$ and t	► Υ

Informal reformulation of the definitions

Class	Definition	Sym.
	• over any interval[1], if $x(t)$ is very close to $\bar{x} \in \operatorname{dom} f$ then $y_{1..m}(t)$ converges to $f(\bar{x}) \pm e^{-\mu}$	► Λ, Ω
	♦ Undefined behavior if system is perturbed	
$\mathrm{AX}(\Upsilon, \Omega, \Lambda, \Theta)$	**Extreme computability: online with perturbations**	
	Given input $x(t)$, precision $\mu(t)$ and error $e(t)$:	
	• y satisfies a perturbed PIVP, $y(0)$ can be arbitrary and $y'(t)$ depends on $x(t)$ and $\mu(t)$	► g
	• $y(t)$ is bounded by a function of $\|x(t)\|$, $\mu(t)$, $\|e(t)\|$	► Υ
	• over any interval[2], if $x(t)$ is very close to $\bar{x} \in \operatorname{dom} f$, $\mu(t)$ is close to $\bar{\mu}$ and errors are very small then $y_{1..m}(t)$ converges to $f(\bar{x}) \pm e^{-\mu}$	► Λ, Ω, Θ

4.6 Closure properties and computable zoo

In this section, we will show several closure properties of the AP class. We will also show that computable functions are continuous and admit a polynomial modulus of continuity. A number of useful functions are shown to belong to AP such as the minimum, maximum and rounding functions. The relationship between generable and computable functions is also studied.

4.6.I Generable functions

We introduced the notion of GPAC computability as a generalization of GPAC generability. As such, it seems only natural that any generable function must be computable. This is, however, a surprisingly non-trivial result. The reason for this is subtle and has to do with the domain of definition.

We recall that a function is generable if it satisfies a PIVP over an open connected subset. The intuition tells us that computing the value of f, a generable function, at point x is only a matter of finding a path *in the domain of definition* from the initial value x_0 to x, and simulating the differential equation along this path. We saw in Proposition 2.7.4 (Generable path connectedness) that such a path always exists, and can even be assumed to be generable. However, the proof is not constructive and we have no easy way of computing such a path given x. For what we know, it may be possible to build small systems (polynomially bounded) that have an extremely complicated domain of definition.

In full generality, we would like to prove that any function of GPVAL with maximal domain of definition has a computable (or even generable) domain. However this seems like a highly nontrivial result.

For this reason, we ignore the problem completely and restrict ourselves to the case where finding the path is trivial: star domains with a generable vantage point. There are several reasons to restrict to this class of domains. First it is a well-known notion, in particular in the field of differential forms which have close links to generable functions[3]. Another reason

[1]Means the following property is true over any time interval $J \subseteq \mathbb{R}_+$.

[2]See Foonote 1

[3]In essence generable functions are a kind of multidimensional exact differential forms of degree 1. Notably Poincaré's lemma states that on a contractible open subset of R^n, closed forms are exact and star domains are the simplest case of contractible space.

is that we will mostly need this theorem for domains of the form $\mathbb{R}^n \times \mathbb{R}^m_+$, which happen to be star domains. Finally, star domains capture the essence of necessary condition for this theorem to be true: computing paths between points is easy.

Definition 4.6.1 (Star domain)**:** A set $X \subseteq \mathbb{R}^n$ is called a *star domain* if there exists $x_0 \in X$ such that for all $x \in U$ the line segment from x_0 to x is in X, i.e $[x_0, x] \subseteq X$. Such an x_0 is called a *vantage point.* ♦

Theorem 4.6.2 (GPVAL $\subseteq$ AP over star domains)**:** *If $f \in$ GPVAL has a star domain with a generable vantage point then $f \in$ AP.* ♦

Proof. Let $(f :\subseteq \mathbb{R}^n \rightarrow \mathbb{R}^m) \in \text{GVAL}(\mathtt{sp})$ and $z_0 \in \operatorname{dom} f \cap \mathbb{K}^n$ a generable vantage point. Apply Definition 2.1.17 (Generable function) to get d, p, x_0, y_0 and y. Since y is generable and $z_0 \in \mathbb{K}^d$, apply Corollary 2.7.5 (Generable field stability) to get that $y(z_0) \in \mathbb{K}^d$. Let $x \in \operatorname{dom} f$ and consider the following system:

$$\begin{cases} x(0) = x \\ \gamma(0) = x_0 \\ z(0) = y(z_0) \end{cases} \qquad \begin{cases} x'(t) = 0 \\ \gamma'(t) = x(t) - \gamma(t) \\ z'(t) = p(z(t))(x(t) - \gamma(t)) \end{cases}$$

First note that $x(t)$ is constant and check that $\gamma(t) = x + (x_0 - x)e^{-t}$ and note that $\gamma(\mathbb{R}_+) \subseteq [x_0, x] \subseteq \operatorname{dom} f$ because it is a star domain. Thus $z(t) = y(\gamma(t))$ since $\gamma'(t) = x(t) - \gamma(t)$ and $J_y = p$. It follows that $\|f(x) - z_{1..m}(t)\| = \|f(x) - f(\gamma(t))\|$ since $z_{1..m} = f$. Apply Proposition 2.2.4 (Modulus of continuity) to f to get q, and since $\|\gamma(t)\| \leqslant \|x_0, x\|$ we have:

$$\|f(x) - z_{1..m}(t)\| \leqslant \|x - x_0\| \, e^{-t} q(\|x_0, x\|) \leqslant e^{-t} \operatorname{poly}(\|x\|)$$

Finally, $\|z(t)\| \leqslant \mathtt{sp}(\gamma(t)) \leqslant \operatorname{poly}(\|x\|)$ because $\mathtt{sp}$ is a polynomial.

As a final remark, one can observe that the issue of the domain is in fact reduced to the problem of building γ. In the case of a star domain, this is trivial. In the general case, one would need to show that there is a "generic" such γ that given a point x goes from x_0 to x and stays in the domain of f. ■

4.6.II Arithmetic operations

Theorem 4.6.3 (Closure by arithmetic operations)**:** *If $f, g \in$ AP then $f \pm g, fg \in$ AP, with the obvious restrictions on the domains of definition.* ♦

Proof. We do the proof in the case of $f + g$ in details. Let $\Omega, \Upsilon, \Omega', \Upsilon'$ polynomials such that $f \in \text{AC}(\Upsilon, \Omega)$ and $g \in \text{AC}(\Upsilon', \Omega')$. Apply Definition 4.2.1 (Analog computability) to f and g to get d, p, q and d', p', q' respectively. Let $x \in \operatorname{dom} f \cap \operatorname{dom} g$ and consider the following system:

$$\begin{cases} y(0) = q(x) \\ z(0) = q'(x) \\ w(0) = q(x) + q'(x) \end{cases} \qquad \begin{cases} y'(t) = p(y(t)) \\ z'(t) = p'(z(t)) \\ w'(t) = y'(t) + z'(t) \end{cases}$$

Let $\Omega^*(\alpha, \mu) = \max(\Omega(\alpha, \mu + \ln 2), \Omega'(\alpha, \mu + \ln 2))$ and $\Upsilon^*(\alpha, t) = \Upsilon(\alpha, t) + \Upsilon'(\alpha, t)$. Since, by construction, $w(t) = y(t) + z(t)$, if $t \geqslant \Omega^*(\alpha, \mu)$ then $\|y_{1..m}(t) - f(x)\| \leqslant e^{-\mu - \ln 2}$ and $\|z_{1..m}(t) - g(x)\| \leqslant e^{-\mu - \ln 2}$ thus $\|w_{1..m}(t) - f(x) - g(x)\| \leqslant e^{-\mu}$. Furthermore, $\|y(t)\| \leqslant \Upsilon(\|x\|, t)$ and $\|z(t)\| \leqslant \Upsilon'(\|x\|, t)$ thus $\|w(t)\| \leqslant \Upsilon^*(\|x\|, t)$.

The case of $f - g$ is exactly the same. The case of fg is slightly more involved: one need to take $w'(t) = y_1'(t)z_1(t) + y_1(t)z_1'(t) = p_1(y(t))z_1(t) + y_1(t)p_1'(z(t))$ so that $w(t) = y(t)z(t)$.

The error analysis is a bit more complicated. First note that $\|f(x)\| \leqslant 1 + \Upsilon(\|x\|, \Omega(\|x\|, 0))$ and $\|g(x)\| \leqslant 1 + \Upsilon'(\|x\|, \Omega'(\|x\|, 0))$, and denote by $\ell(\|x\|)$ and $\ell^*(\|x\|)$ those two bounds respectively. Let $t \geqslant \Omega(\|x\|, \mu + \ln 2\ell^*(\|x\|))$ then $\|y_1(t) - f(x)\| \leqslant e^{-\mu-\ln 2\|g(x)\|}$ and similarly if $t \geqslant \Omega'(\|x\|, \mu + \ln 2(1 + \ell^*(\|x\|)))$ then $\|z_1(t) - g(x)\| \leqslant e^{-\mu-\ln 2(1+\|f(x)\|)}$. Thus for t greater than the maximum of both bounds, $\|y_1(t)z_1(t) - f(x)g(x)\| \leqslant \|(y_1(t) - f(x))g(x)\| + \|y_1(t)(z_1(t) - g(x))\| \leqslant e^{-\mu}$ because $\|y_1(t)\| \leqslant 1 + \|f(x)\| \leqslant 1 + \ell(\|x\|)$. ■

4.6.III Continuity and growth

In this section we will show that all computable functions are continuous. More importantly, we will show that they admit a polynomial modulus of continuity, in a similar spirit as Proposition 2.2.4 (Modulus of continuity). This result is not very surprising because AP = AOP and online computability implicitly contains the idea of (modulus of) continuity.

Theorem 4.6.4 (Modulus of continuity)**:** *If $f \in$ AP then f admits a polynomial modulus of continuity: there exists a polynomial $\mho : \mathbb{R}_+^2 \to \mathbb{R}_+$ such that for all $x, y \in \operatorname{dom} f$ and $\mu \in \mathbb{R}_+$:*

$$\|x - y\| \leqslant e^{-\mho(\|x\|,\mu)} \quad \Rightarrow \quad \|f(x) - f(y)\| \leqslant e^{-\mu}$$

In particular f is continuous. ♦

Proof. Let $f \in$ AP, apply Theorem 4.5.1 (Main equivalence) to get that $f \in$ AP $\Upsilon\Omega\Lambda$ where Υ, Ω and Λ are polynomials. Without loss of generality, we assume Ω to be an increasing function. Apply Definition 4.4.17 (Online computability) to get δ, d, p and y_0. Let $u, v \in \operatorname{dom} f$ and $\mu \in \mathbb{R}_+$. Assume that $\|u - v\| \leqslant e^{-\Lambda(\|u\|+1,\mu+\ln 2)}$ and consider the following system:

$$y(0) = y_0 \qquad y'(t) = p(y(t), u)$$

By definition, $\|y_{1..m}(t) - f(u)\| \leqslant e^{-\mu-\ln 2}$ for all $t \geqslant \Omega(\|u\|, \mu + \ln 2)$. For the same reason, $\|y_{1..m}(t) - f(v)\| \leqslant e^{-\mu-\ln 2}$ for all $t \geqslant \Omega(\|v\|, \mu + \ln 2)$ because $\|u - v\| \leqslant e^{-\Lambda(\|u\|+1,\mu+ln2)} \leqslant e^{-\Lambda(\|v\|,\mu+\ln 2)}$. Apply both result to $t = \Omega(\|u\| + 1, \mu + \ln 2)$ to get that $\|f(u) - f(v)\| \leqslant 2e^{-\mu-\ln 2}$. ■

Although this is trivial from the definition, it is worth noting that all functions in AP are polynomially bounded.

Proposition 4.6.5: *Let $f \in$ AP, there exists a polynomial P such that $\|f(x)\| \leqslant P(\|x\|)$ for all $x \in \operatorname{dom} f$.* ♦

Proof. Assume that $f \in \mathrm{AC}(\Upsilon, \Omega)$ and apply Definition 4.2.1 (Analog computability) to get d, p, q. Let $x \in \operatorname{dom} f$ and let y be the solution of $y(0) = q(x)$ and $y' = p(y)$. Apply the definition to get that $\|f(x) - y_{1..m}(\Omega(\|x\|, 0))\| \leqslant 1$ and $\|y(\Omega(\|x\|, 0))\| \leqslant \Upsilon(\|x\|, \Omega(\|x\|, 0)) \leqslant \operatorname{poly}(\|x\|)$ since Υ and Ω are polynomials. ■

4.6.IV Composing functions

Theorem 4.6.6 (Closure by composition)**:** *If $f, g \in$ AP and $f(\operatorname{dom} f) \subseteq \operatorname{dom} g$ then $g \circ f \in$ AP.* ♦

Proof. Let $f : I \subseteq \mathbb{R}^n \to J \subseteq \mathbb{R}^m$ and $g : J \to K \subseteq \mathbb{R}^l$. We will show that $g \circ f$ is computable by using the fact that both f and g are online-computable. We could show directly that $g \circ f$ is online-computable but this would only complicated the proof for no apparent gain.

Apply Theorem 4.5.1 (Main equivalence) to get that g is $(\Upsilon,\Omega,\Lambda)$-online-computable. Apply Definition 4.4.17 (Online computability) to get e,Δ,z_0 for g. Assume that f is (Υ',Ω')-computable. Apply Definition 4.2.1 (Analog computability) to get d,p,q for f. Let $x\in I$ and consider the following system:

$$\begin{cases} y(0)= q(x)\\ y'(t)= p(y(t))\end{cases}\qquad \begin{cases} z(0)= z_0\\ z'(t)= q(z(t),y_{1..m}(t))\end{cases}$$

Define $v(t)=(x(t),y(t),z(t))$ then it immediately follows that v satisfies a PIVP of the form $v(0)=\operatorname{poly}(x)$ and $v'(t)=\operatorname{poly}(v(t))$. Furthermore, by definition:

$$\begin{aligned}\|v(t)\| &\leqslant \max(\|x\|,\|y(t)\|,\|z(t)\|)\\ &\leqslant \max\left(\|x\|,\|y(t)\|,\Upsilon\left(\sup_{u\in[t,t-\Delta]\cap\mathbb{R}_+}\|y_{1..m}(t)\|,t\right)\right)\\ &\leqslant \operatorname{poly}\left(\|x\|,\sup_{u\in[t,t-\Delta]\cap\mathbb{R}_+}\|y(t)\|,t\right)\\ &\leqslant \operatorname{poly}\left(\|x\|,\sup_{u\in[t,t-\Delta]\cap\mathbb{R}_+}\Upsilon'\left(\|x\|,u\right),t\right)\\ &\leqslant \operatorname{poly}\left(\|x\|,t\right)\end{aligned}$$

Define $\bar{x}=f(x)$, $\Upsilon^*(\alpha)=1+\Upsilon'(\alpha,0)$ and $\Omega''(\alpha,\mu)=\Omega'(\alpha,\Lambda(\Upsilon^*(\alpha),\mu))+\Omega(\Upsilon^*(\alpha),\mu)$. By definition of Υ', $\|\bar{x}\|\leqslant 1+\Upsilon'(\|x\|,0)=\Upsilon^*(\|x\|)$. Let $\mu\geqslant 0$ then by definition of Ω', if $t\geqslant\Omega'(\|x\|,\Lambda(\Upsilon^*(\|x\|),\mu))$ then $\|y_{1..m}(t)-\bar{x}\|\leqslant e^{-\Lambda(\Upsilon^*(\|x\|),\mu)}\leqslant e^{-\Lambda(\|\bar{x}\|,\mu)}$. Apply Definition 4.4.17 (Online computability) for $a=\Omega'(\|x\|,\Lambda(\Upsilon^*(\|x\|),\mu))$ to get that $\|z_{1..l}(t)-g(f(x))\|\leqslant e^{-\mu}$ for any $t\geqslant a+\Omega(\bar{x},\mu)$. And since $t\geqslant a+\Omega(\bar{x},\mu)$ whenever $t\geqslant\Omega''(\|x\|,\mu)$, we get that $g\circ f$ is computable. ■

4.6.V Absolute, minimum, maximum value

In this section, we will show that basic functions like the absolute value, the minimum and maximum value are computable. We will also show a powerful result when limiting a function to a computable range. In essence all these result follow from the fact that the absolute value belongs to AP, which is surprisingly non-trivial result (see Example 4.6.7).

Example 4.6.7 (Broken way of computing the absolute value)**:** Computing the absolute value in polynomial time/space is a surprisingly difficult operation, for unintuitive reasons. This example illustrates the problem. A natural idea is to realize that $|x|=x\operatorname{sgn}(x)$. To this end, define $f(x,t)=x\tanh(xt)$ which works because $\tanh(xt)\to\operatorname{sgn}(x)$ when $t\to\infty$. Unfortunately, $\left||x|-f(x,t)\right|\leqslant|x|e^{-|x|t}$ which **converges very slowly for small** x. Indeed, if $x=e^{-\alpha}$ then $\left||x|-f(x,t)\right|\leqslant e^{-\alpha-e^{-\alpha}t}$ so we must take $t(\mu)=e^{\alpha}\mu$ to reach a precision μ. This is unacceptable because it grows with $\frac{1}{|x|}$ instead of $|x|$. In particular, it is unbounded when $x\to 0$ which is clearly wrong. ♦

The sign function is not computable because it not continuous. However, if f is a continuous function that cancels at 0 then $\operatorname{sgn}(x)f(x)$ is continuous. We prove an effective version of this remark below. The absolute value will then follows as a special case of this result.

The proof is not difficult but the idea is not very intuitive. As the example outlines, we cannot simply compute $f(x)\tanh(g(x)t)$ and hope that it converges quickly enough when

$t \to \infty$. What if we could replace t by e^t ? It would work of course, but cannot do that. Except if we can ? The crucial point is to realize that we do not really need to compute $\tanh(g(x)e^t)$ for arbitrary large t, we only need it to "bootstrap" so that $g(x)e^t \approx \text{poly}(t)$. This can be done in a clever way by bounding the growth the function when it becomes too large.

Proposition 4.6.8 (Smooth sign)**:** *For any polynomial $p : \mathbb{R}_+ \to \mathbb{R}_+$, $H_p \in \text{AP}$ where*

$$H_p(x,z) = \operatorname{sgn}(x)z \qquad (x,z) \in U_p := \left\{(0,0)\right\} \cup \left\{(x,z) \in \mathbb{R}^* \times \mathbb{R} \;:\; \left|\tfrac{z}{x}\right| \leqslant e^{p(\|x,z\|)}\right\}$$

♦

Proof. Let $(x,z) \in U$ and consider the following system:

$$\begin{cases} s(0) = x \\ y(0) = z\tanh(x) \end{cases} \qquad \begin{cases} s'(t) = \tanh(s(t)) \\ y'(t) = \left(1 - \tanh(s(t))^2\right) y(t) \end{cases}$$

First check that $y(t) = z\tanh(s(t))$. The case of $x = 0$ is trivial because $s(t) = 0$ and $y(t) = 0 = H(x,z)$. If $x < 0$ then check that the same system for $-x$ has the opposite value for s and y so all the convergence result will the exactly the same and will be correct because $H(x,z) = -H(-x,z)$. Thus we can assume that $x > 0$.

Apply Lemma 2.6.1 (Bounds on tanh) to get that $1 - e^{-u} \leqslant \tanh(u) \leqslant 1$ for all $u \in \mathbb{R}_+$. Thus $\tanh(s(t)) \geqslant 1 - e^{-s(t)}$ and by a classical result of differential inequalities, $s(t) \geqslant w(t)$ where $w(0) = s(0) = x$ and $w'(t) = 1 - e^{-w(t)}$. Again check that $w(t) = \ln\left(1 + (e^x - 1)e^t\right)$. From this conclude that $|z - y(t)| \leqslant \frac{|z|}{1+(e^x-1)e^t} \leqslant \frac{|z|e^{-t}}{e^x-1} \leqslant \frac{|z|}{x}e^{-t} \leqslant e^{p(\|x,z\|)-t}$. Thus $|z - y(t)| \leqslant e^{-\mu}$ for all $t \geqslant \mu + p(\|x,z\|)$ which is polynomial in $\|x,z,\mu\|$. Furthemore, $|s(t)| \leqslant |x| + t$ because $s'(t)| \leqslant 1$ and $|y(t)| \leqslant |z|$ so the system is polynomially bounded. Finally, the system is of the form $(s,y)(0) = f(x)$ and $(s,y)'(t) = g((s,y)(t))$ where $f,g \in \text{GPVAL}$ so $H_p \in \text{AP}$ with generable functions. Apply Proposition 4.5.2 (Polynomial versus generable) to conclude. ■

Theorem 4.6.9 (Absolute value)**:** $(x \mapsto |x|) \in \text{AP}$. ♦

Proof. Let $p(x) = 0$ which is a polynomial, and $a(x) = H_p(x,x)$ where $H_p \in \text{AP}$ comes from Proposition 4.6.8 (Smooth sign). It is not hard to see that a is defined over $\mathbb{R}$ because $(0,0) \in U_p$ and for any $x \neq 0$, $\left|\frac{x}{x}\right| \leqslant 1 = e^{p(|x|)}$ thus $(x,x) \in U_p$. Consequently $a \in \text{AP}$ and for any $x \in \mathbb{R}$, $a(x) = \operatorname{sgn}(x)x = |x|$ which concludes. ■

Corollary 4.6.10 (Max, Min)**:** $\max, \min \in \text{AP}$. ♦

Proof. Use that $\max(a,b) = \frac{a+b}{2} + \left|\frac{a+b}{2}\right|$ and $\min(a,b) = -\max(-a,-b)$. Conclude with Theorem 4.6.9 (Absolute value), Theorem 4.6.3 (Closure by arithmetic operations) Theorem 4.6.6 (Closure by composition) ■

4.6.VI Rounding

In Section 2.6 (Generable zoo) we saw that it was possible to build a very good generable rounding function. In this section, we will see that we can do even better with computable functions. More precisely, we will build a computable function that rounds perfectly everywhere, except on a small, periodic, interval of size $e^{-\mu}$ where μ is a parameter. This is the best can do because of the continuity and modulus of continuity requirements of computable functions, as show in Theorem 4.6.4 (Modulus of continuity). We will need a few technical lemmas before getting to the rounding function itself.

Remark 4.6.11 (Constant function)**:** Let $f \in \mathrm{AOP}$, I a convex subset of $\operatorname{dom} f$ and assume that f is constant over I, with value α. Apply Definition 4.4.17 (Online computability) to get d, δ, p and y_0. Let $x \in C^0(\mathbb{R}_+, \operatorname{dom} f)$ and consider the system:

$$y(0) = y_0 \qquad y'(t) = p(y(t), x(t))$$

If there exists $J = [a, b]$ and M such that for all $x(t) \in I$ and $\|x(t)\| \leqslant M$ for all $t \in J$, then $\|y_{1..m}(t) - \alpha\| \leqslant e^{-\mu}$ for all $t \in [a + \Omega(M, \mu), b]$. This is unlike the usual case where the input must be nearly constant and it is true because whatever the system can sample from the input $x(t)$, the resulting output will be the same. Formally, it can seen from the proof of Theorem 4.4.15 (Strong $\subseteq$ extreme), or buy building a small system around the online-system that samples the input, even if it unstable. ♦

Proposition 4.6.12 (Clamped exponential)**:** *For any $a, b, c, d, x \in \mathbb{R}$ such that $a \leqslant b$ and $\ell \in \mathbb{R}_+$, define h as follows. Then $h \in \mathrm{AP}$:*

$$h(a, b, c, d, x) = \max(a, \min(b, ce^x + d))$$

♦

Proof. First note that we can assume that $d = 0$ because $h(a, b, c, d, x) = h(a-d, b-d, c, 0, x)+d$. Similarly, we can assume that $a = -b$ and $b \geqslant |c|$ because $h(a, b, c, d, x) = \max(a, \min(b, h(-|c|-\max(|a|, |b|), |c|+\max(|a|, |b|), c, d, x)))$ and min, max, $|\cdot| \in \mathrm{AP}$. So we are left with $H(\ell, c, x) = \max(-\ell, \min(\ell, ce^x))$ where $\ell \geqslant |c|$ and $x \in \mathbb{R}$. Furthermore, we can assume that $c \geqslant 0$ because $H(\ell, c, x) = \operatorname{sgn}(c)H(\ell, |c|, x)$ and it belongs to AP for all $\ell \geqslant |c|$ and $x \in \mathbb{R}$ thanks to Proposition 4.6.8 (Smooth sign). Indeed, if $c = 0$ then $H(\ell, |c|, x) = 0$ and if $c \neq 0$, $\ell \geqslant |c|$ and $x \in \mathbb{R}$, then $\left|\frac{c}{H(\ell,|c|,x)}\right| \geqslant e^{-|x|}$.

We will show that $H \in \mathrm{AWP}$, let $\ell \geqslant c \geqslant 0$, $\mu \in \mathbb{R}_+$, $x \in \mathbb{R}$ and consider the following system:

$$\begin{cases} y(0) = c \\ z(0) = 0 \end{cases} \qquad \begin{cases} y'(t) = z'(t)y(t) \\ z'(t) = (1 + \ell - y(t))(x - z(t)) \end{cases}$$

Note that formally, we should add extra variables to hold x, μ and ℓ (the inputs). Also note that to make this a PIVP, we should replace $z'(t)$ by its expression in the right-hand side, we kept $z'(t)$ to make things more readable. By construction $y(t) = ce^{z(t)}$, and since $\ell \geqslant c \geqslant 0$, by a classical differential argument, $z(t) \in [0, x]$ and $y(t) \in [0, \min(ce^x, \ell + 1)]$. This shows in particular that the system is polynomially bounded in $\|\ell, x, c\|$. There are two cases to consider.

- If $\ell \geqslant ce^x$ then $\ell - y(t) = \ell - ce^{z(t)} \geqslant c(e^x - e^{z(t)}) \geqslant c(x - z(t)) \geqslant 0$ thus by a classical differential inequalities reasoning, $z(t) \geqslant w(t)$ where w satisfies $w(0) = 0$ and $w'(t) = (x - w(t))$. This system can be solved exactly and $w(t) = x(1 - e^{-t})$. Thus $y(t) \geqslant ce^{w(t)} \geqslant ce^x e^{-xe^{-t}} \geqslant ce^x(1 - xe^{-t}) \geqslant ce^x - cxe^{x-t}$. So if $t \geqslant \mu + x + c$ then $y(t) \geqslant ce^x - e^{-\mu}$ then since $y(t) \leqslant ce^x$ it shows that $|y(t) - ce^x| \leqslant e^{-\mu}$.

- If $\ell \leqslant ce^x$ then by the above reasoning, $\ell + 1 \geqslant y(t) \geqslant \ell$ when $t \geqslant \mu + x + c$.

We will modify this sytem to feed y to an online-system computing $\min(-\ell, \max(\ell, \cdot))$. The idea is that when $y(t) \geqslant \ell$, this online-system is constant so the input does not need to be stable.

Let $G(x) = \min(\ell, x)$ then $G \in \mathrm{AOP}$, apply Definition 4.4.17 (Online computability) to get d, δ, p and y_0. Let x, c, ℓ, μ and consider the following system (where y and z are from the previous system):

$$w(0) = y_0 \qquad w'(t) = p(w(t), y(t))$$

Again, there are two cases.

- If $\ell \geqslant ce^x$ then $|y(t) - ce^x| \leqslant e^{-\Lambda(\ell,\mu)} \leqslant e^{-\Lambda(ce^x,\mu)}$ when $t \geqslant \Lambda(\ell,\mu) + x + c$, thus $|w_1(t) - G(ce^x)| \leqslant e^{-\mu}$ when $t \geqslant \Lambda(\ell,\mu) + x + c + \Omega(\ell,\mu)$ and this concludes because $G(ce^x) = ce^x$.

- If $\ell \leqslant ce^x$ then by the above reasoning, $\ell + 1 \geqslant y(t) \geqslant \ell$ when $t \geqslant \Lambda(\ell,\mu) + x + c$ and thus $|w_1(t) - \ell| \leqslant e^{-\mu}$ when $t \geqslant \Lambda(\ell,\mu) + x + c + \Omega(\ell,\mu)$ by Remark 4.6.11 (Constant function) because $G(x) = \ell$ for all $x \geqslant \ell$.

To conclude the proof that $H \in$ AWP, note that w is also polynomially bounded. ■

Definition 4.6.13 (Round)**:** Let $\text{rnd}^* \in C^0(\mathbb{R},\mathbb{R})$ be the unique function such that:

- $\text{rnd}^*(x,\mu) = n$ for all $x \in \left[n - \frac{1}{2} + e^{-\mu}, n + \frac{1}{2} - e^{-\mu}\right]$ for all $n \in \mathbb{Z}$

- $\text{rnd}^*(x,\mu)$ is affine over $\left[n + \frac{1}{2} - e^{-\mu}, n + \frac{1}{2} + e^{-\mu}\right]$ for all $n \in \mathbb{Z}$

♦

Theorem 4.6.14 (Round)**:** $\text{rnd}^* \in$ AP. ♦

Proof. The idea of the proof is to build a function computing the "fractional part" function, by this we mean a 1-periodic function that maps x to x over $[-1 + e^{-\mu}, 1 - e^{-\mu}]$ and is affine at the border to be continuous. The rounding function immediately follows by subtracting the fractional of x to x. In the details, building this function is not immediate. The intuition is that $\frac{1}{2\pi}\arccos(\cos(2\pi x))$ works well over $[0, 1/2 - e^{-\mu}]$ but needs to be fixed at the border (near $1/2$), and also its parity needs to be fixed based on the sign of $\sin(2\pi x)$.

Formally, define for $c \in [-1,1]$, $x \in \mathbb{R}$ and $\mu \in \mathbb{R}_+$:

$$g(c,\mu) = \max(0, \min((1 - \tfrac{e^\mu}{2})(\arccos(c) - \pi), \arccos(c)))$$

$$f(x,\mu) = \frac{1}{2\pi}\operatorname{sgn}(\sin(2\pi x))g(\cos(2\pi x),\mu)$$

Check that $g \in$ AP because of Proposition 4.6.12 (Clamped exponential) and the fact that $\arccos \in$ AP because $\arccos \in$ GPVAL. Then $f \in$ AP by Proposition 4.6.8 (Smooth sign). Indeed, if $\sin(2\pi x) = 0$ then $g(\cos(2\pi x),\mu) = 0$ and if $\sin(2\pi x) \neq 0$, a tedious computation shows that $\left|\frac{g(\cos(2\pi x),\mu)}{\sin(2\pi x)}\right| = \min\left((1 - \frac{e^\mu}{2})\frac{\arccos(\cos(2\pi x)) - \pi}{\sin(2\pi x)}, \frac{\arccos(\cos(2\pi x))}{\sin(2\pi x)}\right) \leqslant 2\pi e^\mu$ because $g(\cos(2\pi x),\mu)$ is piecewise affine with slope e^μ at most (see below for more details).

Note that f is 1-periodic because of the sine and cosine so we only need to analyze if over $[-\frac{1}{2}, \frac{1}{2}]$, and since f is an odd function, we only need to analyze it over $[0, \frac{1}{2}]$. Let $x \in [0, \frac{1}{2}]$ and $\mu \in \mathbb{R}_+$ then $2\pi x \in [0,\pi]$ thus $\arccos(\cos(2\pi x)) = 2\pi x$ and $f(x,\mu) = \min((1 - \frac{e^\mu}{2})(x - \frac{1}{2}), \frac{x}{2\pi})$. There are two cases:

- if $x \in [0, \frac{1}{2} - e^{-\mu}]$ then $x - \frac{1}{2} \leqslant -e^{-\mu}$ thus $(1 - \frac{e^\mu}{2})(x - \frac{1}{2}) \geqslant \frac{1}{2} - e^{-\mu} \geqslant \frac{x}{2\pi}$ so $f(x,\mu) = x$

- if $x \in [\frac{1}{2} - e^{-\mu}, \frac{1}{2}]$ then $0 \geqslant x - \frac{1}{2} \geqslant -e^{-\mu}$ thus $(1 - \frac{e^\mu}{2})(x - \frac{1}{2}) \leqslant \frac{1}{2} - e^{-\mu} \leqslant \frac{x}{2\pi}$ so $f(x,\mu) = (1 - \frac{e^\mu}{2})(x - \frac{1}{2})$ which is affine

Finally define $\text{rnd}^*(x,\mu) = x - f(x,\mu)$ to get the desired function. ■

4.6.VII Mixing functions

Suppose that we have two continuous functions f_0 and f_1 that partially cover $\mathbb{R}$ but such that $\operatorname{dom} f_0 \cup \operatorname{dom} f_1 = \mathbb{R}$. We would like to build a new continuous function defined over $\mathbb{R}$ out of them. One way of doing this is to build a function f that equals f_0 over $\operatorname{dom} f_0 \setminus \operatorname{dom} f_1$, f_1 over $\operatorname{dom} f_1 \setminus \operatorname{dom} f_0$ and a linear combination of both in between. For example consider $f_0(x) = x^2$ defined over $]-\infty, 1]$ and $f_1(x) = x$ over $[0, \infty[$. This approach may work from a mathematical point of view, but it raises severe computational issues: how do we describe the two domains ? How do we compute a linear interpolation between arbitrary sets ? What is the complexity of this operation ? This would require to discuss the complexity of real sets, which is a whole subject by itself.

A more elementary solution to this problem is what we call *mixing*. We assume that we are given an indicator function i that covers the domain of both functions. Such an example would be $i(x) = x$ in the previous example. The intuition is that i describes both the domains and the interpolation. Precisely, the resulting function should be $f_0(x)$ if $i(x) \leqslant 0$, $f_1(x)$ if $i(x) \geqslant 1$ and a *mix* of $f_0(x)$ and $f_1(x)$ inbetween. The consequence of this choice is that the domain of f_0 and f_1 must overlap on the region $\{x \mid 0 < i(x) < 1\}$. In the previous example, we need to define f_0 over $]-\infty, 1[= \{x \mid i(x) < 1\}$ and f_1 over $]0, \infty] = \{x \mid i(x) > 0\}$. Several types of mixing are possible, the simplest being linear interpolation: $(1-i(x))f_0(x)+i(x)f_1(x)$. Formally, we are building the following continuous function:

$$f(x) = \begin{cases} f_0(x) & \text{if } i(x) \leqslant 0 \\ (1-i(x))f_0(x) + i(x)f_1(x) & \text{if } 0 < i(x) < 1 \\ f_1(x) & \text{if } i(x) \geqslant 1 \end{cases}$$

The main result of this section is to show that if f_0, f_1 and i are analog-polytime then f is also analog-polytime.

Essentially, the proof looks very much like the proof of Theorem 4.6.3 (Closure by arithmetic operations) since the mixing function is a linear interpolation. The tricky detail is that x does not belong to the domain of definition of f_0 and f_1 all the time. In those cases, we still run the system "computing" f_0 and f_1 but we must ensure that invalid outputs are ignored.

Definition 4.6.15 (Mixing function)**:** Let $f_0 :\subseteq \mathbb{R}^n \to \mathbb{R}^d$, $f_1 :\subseteq \mathbb{R}^n \to \mathbb{R}^d$ and $i :\subseteq \mathbb{R}^n \to \mathbb{R}$. Assume that $\{x \mid i(x) < 1\} \subseteq \operatorname{dom} f_0$ and $\{x \mid i(x) > 0\} \subseteq \operatorname{dom} f_1$, and define for $x \in \operatorname{dom} i$:

$$\operatorname{mix}(i, f_0, f_1)(x) = \begin{cases} f_0(x) & \text{if } i(x) \leqslant 0 \\ (1-i(x))f_0(x) + i(x)f_1(x) & \text{if } 0 < i(x) < 1 \\ f_1(x) & \text{if } i(x) \geqslant 1 \end{cases}$$

♦

Theorem 4.6.16 (Closure by mixing)**:** *Let $f_0 :\subseteq \mathbb{R}^n \to \mathbb{R}^d$, $f_1 :\subseteq \mathbb{R}^n \to \mathbb{R}^d$ and $i :\subseteq \mathbb{R}^n \to \mathbb{R}$. Assume that $f_0, f_1, i \in \mathrm{AP}$, that $\{x \mid i(x) < 1\} \subseteq \operatorname{dom} f_0$ and that $\{x \mid i(x) > 0\} \subseteq \operatorname{dom} f_1$. Then* $\operatorname{mix}(i, f_0, f_1) \in \mathrm{AP}$. ♦

Proof. We first modify i so that it takes values in $[0, 1]$ only. To this end, introduce:

$$i^\infty(x) = \max(0, \min(1, i(x)))$$

Apply Theorem 4.6.6 (Closure by composition) and Corollary 4.6.10 (Max, Min) to get that $i^\infty \in \mathrm{AP}$. Apply Theorem 4.5.1 (Main equivalence) to get that i^∞ is $(\Upsilon^\infty, \Omega^\infty)$-computable and f_0, f_1

are $(\Upsilon^0,\Omega^0,\Lambda^0)$-online-computable and $(\Upsilon^1,\Omega^1,\Lambda^1)$-online-computable, respectively. Without loss of generality, we can assume that Υ and Ω functions are all increasing. Apply Definition 4.2.1 (Analog computability) to i^∞ to get $d^\infty,p^\infty,q^\infty$. Apply Definition 4.4.17 (Online computability) to f_0 and f_1 to get δ^0,d^0,y_0^0,p^0 and δ^1,d^1,y_0^1,p^1 respectively. Let $x\in\operatorname{dom} i$ and consider the following system:

$$\begin{cases}u(0)=y_{0,1}^1+q_1^\infty(x)(y_{0,1}^1-y_{0,1}^0)\\ v(0)=q^\infty(x)\\ y^0(0)=y_0^0\\ y^1(0)=y_0^1\\ x(0)=x\end{cases}\qquad\begin{cases}u'(t)=(1-v_1(t))y_{1..d}^0(t)+v_1(t)y_{1..d}^1(t)\\ v'(t)=(\psi(0)+\psi'(t))p^\infty(v(t))\\ {y^0}'(t)=p^0(y^0(t),x(t))\\ {y^1}'(t)=p^1(y^1(t),x(t))\\ x'(t)=0\end{cases}$$

$$\psi(t)=\Omega^\infty(\operatorname{norm}_{\infty,1}(x),t+\Upsilon^0(\operatorname{norm}_{\infty,1}(x),t)+\Upsilon^1(\operatorname{norm}_{\infty,1}(x),t))$$

The system looks complicated but most parts of it are straightfoward: $x(t)$ is a constant function storing the input x, y^0 is computing $f_0(x)$, y^1 is computing $f_1(x)$, v is computing $i^\infty(x)$ and u is computing $\operatorname{mix}(i,f_0,f_1)(x)$. Note that we did not write $u'(t)$ directly to make it more readable but one easily checks that $u'(t)$ can be written as a polynomial in the other variables of the system. Note that v is an accelerated version of the system for i^∞, in other words if

$$w(0)=q(x)\qquad w'(t)=p^\infty(w(t))$$

then $v(x)=w(\psi(t)+(t-1)\psi(0))$ by Lemma 2.4.3 (ODE time-scaling). We will use the fact that if $t\geqslant1$ then $\psi(t)+(t-1)\psi(0)\geqslant\psi(t)$, and that $\operatorname{norm}_{\infty,1}(x)\geqslant\|x\|$.

One easily checks that the system is bounded by a polynomial: y^0 is bounded by Υ^0, y^1 is bounded by Υ^1, v is bounded by Υ^∞, x is bounded by $\|x\|$ and finally u is a sum and product of the previous variables.

It remains to see that the system correctly computes the mix function. Let $\mu\geqslant0$ and $t\geqslant\ln6+\mu$, then

$$\psi(t)+(t-1)\psi(0)\geqslant\psi(t)\geqslant\Omega^\infty(\|x\|,\mu+\ln6+\Upsilon^1(\|x\|,t)+\Upsilon^0(\|x\|,t))$$

If $i(x)\leqslant0$ **then** $i^\infty(x)=0$ and by hypothesis $x\in\operatorname{dom} f_0$. Thus $\|v_1(t)-i^\infty(x)\|=\|v_1(t)\|\leqslant e^{-\Upsilon^1(\|x\|,t)-\Upsilon^0(\|x\|,t))-\mu-\ln6}$. If also $t\geqslant\Omega^0(\|x\|,\mu)$ then $\left\|y_{1..d}^0(t)-f_0(x)\right\|\leqslant e^{-\mu-\ln6}$. It follows, using that $xe^{-x}\leqslant1$, that:

$$\begin{aligned}\|u(t)-f_0(x)\|&\leqslant\left\|y_{1..d}^0(t)-f_0(x)\right\|+\|u(t)\|\left(\left\|y_{1..d}^0(t)\right\|+\left\|y_{1..d}^1(t)\right\|\right)\\&\leqslant e^{-\mu-\ln6}+e^{-\Upsilon^1(\|x\|,t)-\Upsilon^0(\|x\|,t))-\mu-\ln6}\left(\Upsilon^1(\|x\|,t)+\Upsilon^0(\|x\|,t))\right)\\&\leqslant e^{-\mu}\end{aligned}$$

If $i(x)\geqslant1$ **then** $i^\infty(x)=1$ and $x\in\operatorname{dom} f_1$ and the proof is exactly the same as previously.

If $i(x)\in]0,1[$ **then** $i^\infty(x)=i(x)$ and $x\in\operatorname{dom} f_0\cap\operatorname{dom} f_1$. Thus $\|v_1(t)-i(x)\|\leqslant e^{-\Upsilon^1(\|x\|,t)-\Upsilon^0(\|x\|,t))-\mu-1}$ so in particular $\|v_1(t)\|\leqslant2$. If also $t\geqslant\max(\Omega^0(\|x\|,\mu+\ln6),\Omega^1(\|x\|,\mu+\ln6))$ then $\left\|y_{1..d}^0(t)-f_0(x)\right\|\leqslant e^{-\mu-\ln6}$ and $\left\|y_{1..d}^1(t)-f_1(x)\right\|\leqslant e^{-\mu-\ln6}$. It follows, using that $xe^{-x}\leqslant1$, that:

$$\begin{aligned}\|u(t)-\operatorname{mix}(i,f_0,f_1)(x)\|&=\left\|(1-v_1(t))y_{1..d}^0(t)+v_1(t)y_{1..d}^1(t)-(1-i(x))f_0(x)-i(x)f_1(x)\right\|\\&\leqslant\|i(x)-v_1(t)\|\,\|f_0(x)-f_1(x)\|+\|1-v_1(x)\|\left\|y_{1..d}^0(x)-f_0(x)\right\|\\&\quad+\|v_1(x)\|\left\|y_{1..d}^1(t)-f_1(x)\right\|\end{aligned}$$

$$\leqslant e^{-\Upsilon^1(\|x\|,t)-\Upsilon^0(\|x\|,t))-\mu-\ln 6}\left(\Upsilon^1(\|x\|,t)+\Upsilon^0(\|x\|,t))\right)+4e^{-\mu-\ln 6}$$
$$\leqslant e^{-\mu}$$

Finally, after differentiating the equation for $u(t)$, the system is of the form $z(0) = f(x)$ and $z'(t) = g(z(t))$ where $f, g \in \text{GPVAL}$ so $\text{mix}(i, f_0, f_1) \in \text{AP}$ with generable functions. Apply Proposition 4.5.2 (Polynomial versus generable) to conclude. ■

4.6.VIII Computing limits

Intuitively, this model of computation already contains the notion of limit. More precisely, if f is computable and is such that $f(x,t) \to g(x)$ when $t \to \infty$ then g is computable. This is just a reformulation of equivalence between computability and weak-computability. The result below extends this result to the case where the limit is restricted to $t \in \mathbb{N}$. The optimality of the assumptions is discussed in Remark 4.6.18 (Optimality).

The idea of the proof is to show that g is weakly-computable and use the equivalence with computability. Given x and μ, we want to run f on $(x, \lceil\omega\rceil) \in I \times J$ where $\omega = \mho(\|x\|,\mu)$. Unfortunately we cannot compute the ceiling value in a continuous fashion. The trick is to run two systems in parallels: one on $(x, (\text{rnd}\,\omega))$ and one on $(x, \text{rnd}(\omega + \frac{1}{2}))$. This way one system will always have a correct input value but we must select which one. If rnd is a good rounding function around $[n-\frac{1}{3}, n+\frac{1}{3}]$, we build the selecting function to pick the first system in $[n, n+\frac{1}{6}]$, a barycenter of both in $[n+\frac{1}{6}, n+\frac{1}{3}]$ and the second system in $[n+\frac{1}{3}, n+\frac{2}{3}]$ and so on. The crucial point is that in the region where we mix both system, both have correct inputs so the mixing process doesn't create any error. Furthermore, we can easily build such a continuous selecting function and the mixing process has already been studied in a previous section.

Theorem 4.6.17 (Closure by limit)**:** *Let $f : I \times J \subseteq \mathbb{R}^{n+1} \to \mathbb{R}^m$, $g : I \to \mathbb{R}^m$ and $\mho : \mathbb{R}_+^2 \to \mathbb{R}_+$ a polynomial. Assume that $f \in \text{AP}$ and that $J \supseteq \mathbb{N}$. Further assume that for all $(x,\tau) \in I \times J$ and $\mu \geqslant 0$, if $\tau \geqslant \mho(\|x\|,\mu)$ then $\|f(x,\tau) - g(x)\| \leqslant e^{-\mu}$. Then $g \in \text{AP}$.* ♦

Proof. First note that $\frac{1}{2} - e^{-2} \geqslant \frac{1}{3}$ and define for $x \in I$ and $n \in \mathbb{N}$:

$$f_0(x,\tau) = f(x, \text{rnd}^*(\tau, 2)) \qquad \tau \in \left[n - \tfrac{1}{3}, n + \tfrac{1}{3}\right]$$
$$f_1(x,\tau) = f(x, \text{rnd}^*(\tau + \tfrac{1}{2}, 2)) \qquad \tau \in \left[n + \tfrac{1}{6}, n + \tfrac{5}{6}\right]$$

By Definition 4.6.13 (Round) and hypothesis on f, both are well-defined because $\mathbb{N} \subseteq J$. Also note that their domain of definition overlap on $[n+\frac{1}{6}, n+\frac{1}{3}]$ and $[n+\frac{2}{3}, n+\frac{5}{6}]$ for all $n \in \mathbb{N}$. Apply Theorem 4.6.14 (Round) and Theorem 4.6.6 (Closure by composition) to get that $f_0, f_1 \in \text{AP}$. We also need to build the indicator function: this is where the choice of above values will prove convenient. Define for any $\tau \in \mathbb{R}_+$:

$$i(x,\tau) = \tfrac{1}{2} - \cos(2\pi\tau)$$

It is now easy to check that:

$$\{(x,n) \mid i(x) < 1\} = \mathbb{R}_+ \cap \cup_{n\in\mathbb{N}} \left]n - \tfrac{1}{3}, n + \tfrac{1}{3}\right[\subseteq \text{dom}\, f_0$$
$$\{(x,n) \mid i(x) > 0\} = \mathbb{R}_+ \cap \cup_{n\in\mathbb{N}} \left]n + \tfrac{1}{6}, n + \tfrac{5}{3}\right[\subseteq \text{dom}\, f_1$$

Define for any $x \in I$ and $\mu \in \mathbb{R}_+$:

$$f^*(x,\tau) = \text{mix}(i, f_0, f_1)(x,\tau)$$

We can thus apply Theorem 4.6.16 (Closure by mixing) to get that $f^* \in \mathrm{AP}$. Note that f^* is defined over $I \times \mathbb{R}_+$. We now claim that for any $x \in I$ and $\mu \in \mathbb{R}_+$, if $t \geqslant 1 + \mho(\|x\|, \mu)$ then $\|f^*(x,\tau) - g(x)\| \leqslant 2e^{-\mu}$. There are three cases to consider:

- If $\tau \in [n - \frac{1}{6}, n + \frac{1}{6}]$ for some $n \in \mathbb{N}$ then $i(x) \leqslant 0$ so $\operatorname{mix}(i, f_0, f_1)(x,\tau) = f_0(x,\tau) = f(x,n)$ and since $n \geqslant \tau - \frac{1}{6}$ then $n \geqslant \mho(\|x\|, \mu)$ thus $\|f^*(x,\tau) - g(x)\| \leqslant e^{-\mu}$.
- If $\tau \in [n+\frac{1}{3}, n+\frac{2}{3}]$ for some $n \in \mathbb{N}$ then $i(x) \geqslant 1$ so $\operatorname{mix}(i, f_0, f_1)(x,\tau) = f_1(x,\tau) = f(x, n+1)$ and since $n \geqslant \tau - \frac{2}{3}$ then $n + 1 \geqslant \mho(\|x\|, \mu)$ thus $\|f^*(x,\tau) - g(x)\| \leqslant e^{-\mu}$.
- If $\tau \in [n + \frac{1}{6}, n + \frac{1}{3}] \cup [n + \frac{2}{3}, n + \frac{5}{6}]$ for some $n \in \mathbb{N}$ then $i(x) \in [0,1]$ so $f^*(x,\tau) = (1 - i(x,\tau))f_0(x,\tau) + i(x,\tau)f_1(x,\tau) = (1 - i(x,\tau))f(x, \lfloor\tau\rceil) + i(x,\tau)f(x, \left\lfloor\tau + \frac{1}{2}\right\rceil)$. Since $\lfloor\tau\rceil, \left\lfloor\tau + \frac{1}{2}\right\rceil \geqslant \mho(\|x\|, \mu)$ then $\|f(x, \lfloor\tau\rceil) - g(x)\| \leqslant e^{-\mu}$ and $\left\|f(x, \left\lfloor\tau + \frac{1}{2}\right\rceil) - g(x)\right\| \leqslant e^{-\mu}$ thus $\|f^*(x,\tau) - g(x)\| \leqslant 2e^{-\mu}$ because $|i(x,\tau)| \leqslant 1$.

It follows that g is the limit of f^* and thus $g \in \mathrm{AWP}$ (see Remark 4.2.9 (Limit computability)) and one concludes using that AWP = AP. ■

Remark 4.6.18 (Optimality)**:** The condition that $\mho$ be a polynomial is essentially optimal. Intuitively, if $f \in \mathrm{AP}$ and satisfies that $\|f(x,\tau) - g(x)\| \leqslant e^{-\mu}$ whenever $\tau \geqslant \mho(\|x\|, \mu)$ then $\mho$ is a modulus of continuity for g. By Theorem 4.6.4 (Modulus of continuity), if $g \in \mathrm{AP}$ then it admits a polynomial modulus of continuity so $\mho$ must be a polynomial. For a formal proof of this intuition, see examples 4.6.19 and 4.6.20 ♦

Example 4.6.19 ($\mho$ must be polynomial in x)**:** Let $f(x,\tau) = \min(e^x, \tau)$ and $g(x) = e^x$. Trivially $f(x,\cdot)$ converges to g because $f(x,\tau) = g(x)$ for $\tau \geqslant e^x$. But $g \notin \mathrm{AP}$ because it is not polynomially bounded. In this case $\mho(x,\mu) = e^x$ which is exponential and $f \in \mathrm{AP}$ by Proposition 4.6.12 (Clamped exponential). ♦

Example 4.6.20 ($\mho$ must be polynomial in μ)**:** Let $g(x) = \frac{-1}{\ln x}$ for $x \in [0, e]$ which is defined in 0 by continuity. Observe that $g \notin \mathrm{AP}$, indeed its modulus of continuity is exponential around 0 because $g(e^{-e^{\mu}}) = e^{-\mu}$ for all $\mu \geqslant 0$. However note that $g^* \in \mathrm{AP}$ where $g^*(x) = g(e^{-x}) = \frac{1}{x}$ for $x \in [1, +\infty[$. Let $f(x,\tau) = g^*(\min(-\ln x, \tau))$ and check, using that g is increasing and non-negative, that: $|f(x,\tau) - g(x)| = |g(\max(x, e^{-\tau})) - g(x)| \leqslant g(\max(x, e^{-\tau})) \leqslant \frac{1}{\tau}$. Thus $\mho(\|x\|, \mu) = e^{\mu}$ which is exponential and $f \in \mathrm{AP}$ because $(x,\tau) \mapsto \min(-\ln x, \tau) \in \mathrm{AP}$ by a proof similar to Proposition 4.6.12 (Clamped exponential). ♦

4.6.IX Iterating functions

In this section, we show that iterating a computable function is computable under reasonable assumptions. Iteration is a powerful operation, which is why reasonable complexity classes are never closed under unrestricted iteration. In the context of GPAC computability, there are at least two immediate necessary conditions: the iterates cannot grow faster than a polynomial and the iterates must keep a polynomial modulus of continuity. The optimality of these conditions is discussed in Remark 4.6.22 and Remark 4.6.23. However there is the subtler issue of the domain of definition that comes into play and is discussed in Remark 4.6.24. Our conditions to iterate a function can be summarized as follows:

- f has domain of definition I;
- there are subsets I_n of I such that points of I_n can be iterated up to n times;
- the iterates of f on x over I_n grow at most polynomially in $\|x\|$ and n;

- each point x in I_n has an open neighborhood in I of size at least $e^{-\operatorname{poly}(\|x\|)}$ and f has modulus of continuity of the form $\operatorname{poly}(\|x\|)+\mu$ over this set.

Theorem 4.6.21 (Closure by iteration)**:** *Let $I\subseteq\mathbb{R}^m$, $(f:I\to\mathbb{R}^m)\in\mathrm{AP}$, $\eta\in[0,1/2[$ and assume that there exists a family of subsets $I_n\subseteq I$, for all $n\in\mathbb{N}$ and polynomials $\mho:\mathbb{R}_+\to\mathbb{R}_+$ and $\Pi:\mathbb{R}_+^2\to\mathbb{R}_+$ such that for all $n\in\mathbb{N}$:*

- *$I_{n+1}\subseteq I_n$ and $f(I_{n+1})\subseteq I_n$*
- *for all $x\in I_n$, $\left\|f^{[n]}(x)\right\|\leqslant\Pi(\|x\|,n)$*
- *for all $x\in I_n$, $y\in\mathbb{R}^m,\mu\in\mathbb{R}_+$, if $\|x-y\|\leqslant e^{-\mho(\|x\|)-\mu}$ then $y\in I$ and $\|f(x)-f(y)\|\leqslant e^{-\mu}$*

Define $f_\eta^*(x,u)=f^{[n]}(x)$ for $x\in I_n$, $u\in[n-\eta,n+\eta]$ and $n\in\mathbb{N}$. Then $f_\eta^*\in\mathrm{AP}$. ♦

Proof. We use three variables y, z and w and build a cycle to be repeated n times. At all time, y is an online system computing $f(w)$. During the first stage of the cycle, w stays still and y converges to $f(w)$. During the second stage of the cycle, z copies y while w stays still. During the last stage, w copies z thus effectively computing one iterate.

The crucial point is in the error estimation, which we informally develop here. Denote the k^{th} iterate of x by $x^{[k]}$ and by $x^{(k)}$ the point computed after k cycles in the system. Because we are doing an approximation of f at each step step, the relationship between the two is that $x_0=x^{[0]}$ and $\left\|x^{(k+1)}-f(x_k)\right\|\leqslant e^{-\nu_{k+1}}$ where ν_{k+1} is the precision of the approximation, that we control. Define μ_k the precision we need to achieve at step k: $\left\|x^{(k)}-x^{[k]}\right\|\leqslant e^{-\mu_k}$ and $\mu_n=\mu$. The triangle inequality ensures that the following choice of parameters is safe:

$$\nu_k\geqslant\mu_k+\ln 2\qquad \mu_{k-1}\geqslant\mho\left(\left\|x^{[k-1]}\right\|\right)+\mu_k+\ln 2$$

This is ensured by taking $\mu_k\geqslant\sum_{i=k}^{n-1}\mho(\Pi(\|x\|,i))+\mu+(n-k)\ln 2$ which is indeed polynomial in k, μ and $\|x\|$. Finally a point worth mentionning is that the entire reasoning makes sense because the assumption ensures that $x^{(k)}\in I$ at each step.

Formally, apply Theorem 4.5.1 (Main equivalence) to get that $f\in\mathrm{AX}(\Upsilon,\Omega,\Lambda,\Theta)$ where $\Upsilon,\Lambda,\Theta,\Omega$ are polynomials. Without loss of generability we assume that $\Upsilon,\Lambda,\Theta,\mho$ and Π are increasing functions. Apply Lemma 4.4.16 (AXP time rescaling) to get $\omega\geqslant 1$ such that for all $\alpha\in\mathbb{R},\mu\in\mathbb{R}_+$:

$$\Omega(\alpha,\mu)=\omega\geqslant 1$$

Apply Definition 4.4.2 (Extreme computability) to get δ,d and g. Define:

$$\tau=\omega+2$$

We will show that $f_0^*\in\mathrm{AWP}=\mathrm{AP}$: let $n\in\mathbb{N}$, $x\in I_n$, $\mu\in\mathbb{R}_+$ and consider the following system:

$$\begin{cases}\ell(0)=\operatorname{norm}_{\infty,1}(x)\\ \mu(0)=\mu\\ n(0)=n\end{cases}\qquad\begin{cases}\ell'(t)=0\\ \mu'(t)=0\\ n'(t)=0\end{cases}\qquad\begin{cases}y(0)=0\\ z(0)=x\\ w(0)=x\end{cases}$$

$$\begin{cases}y'(t)=g(t,y(t),w(t),\nu(t))\\ z'(t)=\operatorname{sample}_{[\omega,\omega+1],\tau}(t,\nu(t),z(t),y_{1..n}(t))\\ w'(t)=\operatorname{hxl}_{[0,1]}(t-n\tau,\nu(t)+t,\operatorname{sample}_{[\omega+1,\omega+2],\tau}(t,\nu^*(t)+\ln(1+\omega),w(t),z(t)))\end{cases}$$

$$\ell^*=1+\Pi(\ell,n)\qquad \nu=n\mho(\ell^*)+n\ln 6+\mu+\ln 3\qquad \nu^*=\nu+\Lambda(\ell^*,\nu)$$

First notice that ℓ, μ and n are constant functions and we identify $\mu(t)$ with μ and $n(t)$ with n. Apply Lemma 2.6.18 (Norm function) to get that $\|x\| \leqslant \ell \leqslant \|x\| + 1$, so in particular ℓ^*, ν and ν^* are polynomially bounded in $\|x\|$ and n. We will need a few notations: for $i \in [\![0, n]\!]$, define $x^{[i]} = f^{[i]}(x)$ and $x^{(i)} = w(i\tau)$. Note that $x^{[0]} = x^{(0)} = x$. We will show by induction for $i \in [\![0, n]\!]$ that:

$$\left\|x^{(i)} - x^{[i]}\right\| \leqslant e^{-(n-i)\mho(\ell^*)-(n-i)\ln 6-\mu-\ln 3}$$

Note that this is trivially true for $i = 0$. Let $i \in [\![0, n-1]\!]$ and assume that the result is true for i, we will show that it holds for $i + 1$ by analyzing the behavior of the sytem during period $[i\tau, (i + 1)\tau]$.

- **For y and w, if** $t \in [i\tau, i\tau + \omega + 1]$ **then** apply Lemma 2.6.22 (“low-X-high” and “high-X-low”) to get that hxl $\in [0, 1]$ and Lemma 4.4.14 (Sample and hold) to get that $\|w'(t)\| \leqslant e^{-\nu^*-\ln(1+\omega)}$. Conclude that $\|w(i) - w(t)\| \leqslant e^{-\nu^*}$, in other words $\left\|w(t) - x^{(i)}\right\| \leqslant e^{-\Lambda(\|x^{(i)}\|,\nu)}$ since $\left\|x^{(i)}\right\| \leqslant \left\|x^{[i]}\right\|+1 \leqslant 1+\Pi(\|x\|, i) \leqslant \ell^*$. Thus, by definition of extreme computability, $\left\|f(x^{(i)}) - y_{1..n}(u)\right\| \leqslant e^{-\nu}$ if $u \in [i\tau + \omega, i\tau + \omega + 1]$ because $\Omega\left(\left\|x^{(i)}\right\|, \nu\right) = \omega$.
- **For z, if** $t \in [i\tau + \omega, i\tau + \omega + 1]$ **then** apply Lemma 4.4.14 (Sample and hold) to get that $\left\|f(x^{(i)}) - z(i\tau + \omega + 1)\right\| \leqslant 2e^{-\nu}$.
- **For z and w, if** $t \in [i\tau+\omega+1, i\tau+\omega+2]$ **then** apply Lemma 4.4.14 (Sample and hold) to get that $\|z'(t)\| \leqslant e^{-\nu}$ thus $\left\|f(x^{(i)}) - z(t)\right\| \leqslant 3e^{-\nu}$. Apply Lemma 2.6.22 (“low-X-high” and “high-X-low”) to get that $\left\|y'(t) - \text{sample}_{[\omega+1,\omega+2],\tau}(t, \nu^* + \ln(1 + \omega), w(t), z(t))\right\| \leqslant e^{-\nu-t}$. Apply Lemma 4.4.14 (Sample and hold) again to get that $\left\|f(x^{(i)}) - w(i\tau + \omega + 2)\right\| \leqslant 4e^{-\nu} + e^{-\nu^*} \leqslant 5e^{-\nu}$.

Our analysis concluded that $\left\|f(x^{(i)}) - z((i + 1)\tau)\right\| \leqslant 5e^{-\nu}$. Also, by hypothesis, $\left\|x^{(i)} - x^{[i]}\right\| \leqslant e^{-(n-i)\mho(\ell^*)-(n-i)\ln 6-\mu-\ln 3} \leqslant e^{-\mho(\|x^{[i]}\|)-\mu^*}$ where $\mu^* = (n - i - 1)\mho(\ell^*) + (n - i)\ln 6 + \mu + \ln 3$ because $\left\|x^{[i]}\right\| \leqslant \ell^*$. Consequently, $\left\|f(x^{(i)}) - x^{[i+1]}\right\| \leqslant e^{-\mu^*}$ and thus:

$$\left\|x^{(i+1)} - x^{[i+1]}\right\| \leqslant 5e^{-\nu} + e^{-\mu^*} \leqslant 6e^{-\mu^*} \leqslant e^{-(n-1-i)\mho(\ell^*)-(n-1-i)\ln 6-\mu-\ln 3}$$

From this induction we get that $\left\|x^{(n)} - x^{[n]}\right\| \leqslant e^{-\mu-\ln 3}$. We still have to analyze the behavior after time $n\tau$.

- **If** $t \in [n\tau, n\tau + 1]$ **then** apply Lemma 4.4.14 (Sample and hold) and Lemma 2.6.22 (“low-X-high” and “high-X-low”) to get that $\|w'(t)\| \leqslant e^{-\nu^*-\ln(1+\omega)}$ thus $\left\|w(t) - x^{(n)}\right\| \leqslant e^{-\nu^*-\ln(1+\omega)}$.
- **If** $t \geqslant n\tau + 1$ **then** apply Lemma 2.6.22 (“low-X-high” and “high-X-low”) to get that $\|w'(t)\| \leqslant e^{-\nu-t}$ thus $\|w(t) - w(n\tau + 1)\| \leqslant e^{-\nu}$.

Putting everything together we get for $t \geqslant n\tau + 1$ that:

$$\begin{aligned}\left\|w(t) - x^{[n]}\right\| &\leqslant e^{-\mu-\ln 3} + e^{-\nu^*-\ln(1+\omega)} + e^{-\nu}\\ &\leqslant 3e^{-\mu-\ln 3} \leqslant e^{-\mu}\end{aligned}$$

We also have to show that the system does not grow to fast. The analysis during the time interval $[0, n\tau + 1]$ has already been done (although we did not write all the details, it is an implicit consequence). For $t \geqslant n\tau + 1$, have $\|w(t)\| \leqslant \left\|x^{[n]}\right\| + 1 \leqslant \Pi(\|x\|, n) + 1$ which is polynomially bounded. The bound on y comes from Definition 4.4.2 (Extreme computability):

$$\|y(t)\| \leqslant \Upsilon\left(\sup_\delta \|w\|(t), \nu, 0\right) \leqslant \Upsilon(\Pi(\|x\|, n), \nu, 0) \leqslant \text{poly}(\|x\|, n, \mu)$$

And finally, apply Lemma 4.4.14 (Sample and hold) to get that:

$$\|z(t)\| \leqslant 2 + \sup_{\tau+1} \|y_{1..n}\|(t) \leqslant \operatorname{poly}(\|x\|, n, \mu)$$

This conclude the proof that $f_0^* \in \mathrm{AWP}$.

We will now tackle the case of $\eta > 0$. Let $\eta \in]0, \frac{1}{2}[$ and define $g_\eta(x,\mu) = \operatorname{rnd}(x, \mu, \frac{1}{2} - \eta)$ for $x \in \mathbb{Z} +]-\eta, \eta[$. Apply Lemma 2.6.12 (Round) to get that rnd $\in$ GPVAL and Theorem 4.6.2 (GPVAL $\subseteq$ AP over star domains) to get that $g_\eta \in$ AP. By definition, $\left\|g_\eta(x,\mu) - n\right\| \leqslant e^{-\mu}$ if $x \in [n-\eta, n+\eta]$ thus we can apply Theorem 4.6.17 (Closure by limit) to get that $g_\eta^*(x) = \lim_{\mu\to\infty} g_\eta(x,\mu)$ belongs to AP and $g_\eta^*(x) = n$ for any $x \in [n-\eta, n+\eta]$. Now define $f_\eta^*(x,u) = f_0^*(x, g_\eta^*(u))$ and apply Theorem 4.6.6 (Closure by composition) to conclude. As a final remark, note that g_η^* is a pretty good rounding function but we can do much better: see Theorem 4.6.14 (Round) for more details. ■

Remark 4.6.22 (Optimality of growth constraint)**:** It is easy to see that without any condition, the iterates can produce an exponential function. Pick $f(x) = 2x$ then $f \in$ AP and $f^{[n]}(x) = 2^n x$ which is clearly not polynomial in x and n. More generally, by Proposition 4.6.5, it is necessary that f^* be polynomially bounded so clearly $f^{[n]}(x)$ must be polynomially bounded in $\|x\|$ and n. ♦

Remark 4.6.23 (Optimality of modulus constraint)**:** Without any constraint, it is easy to build an iterated function with exponential modulus of continuity. Define $f(x) = \sqrt{x}$ then $f \in$ AP and $f^{[n]}(x) = x^{\frac{1}{2^n}}$. For any $\mu \in \mathbb{R}$, $f^{[n]}(e^{-2^n\mu}) - f^{[n]}(0) = (e^{-2^n\mu})^{\frac{1}{2^n}} = e^{-\mu}$. Thus f^* has exponential modulus of continuity in n. ♦

Remark 4.6.24 (Domain of definition)**:** Intuitively we could have written the theorem differently, only requesting that $f(I) \subseteq I$, however this has some problems. First if I is discrete, the iterated modulus of continuity becomes useless and the theorem is false. Indeed, define $f(x,k) = (\sqrt{x}, k+1)$ and $I = \left\{(\sqrt[2^n]{e}, n), n \in \mathbb{N}\right\}$: $f\restriction_I$ has polynomial modulus of continuity $\mho$ because I is discrete, yet $f^*\restriction_I \notin$ AP as we saw in Remark 4.6.23. But in reality, the problem is more subtle than that because if I is open but the neighbourhood of each point is too small, a polynomial system cannot take advantage of it. To illustrate this issue, define $I_n = \left]0, \sqrt[2^n]{e}\right[\times \left]n - \frac{1}{4}, n + \frac{1}{4}\right[$ and $I = \cup_{n\in\mathbb{N}} I_n$. Clearly $f(I_n) = I_{n+1}$ so I is f-stable but $f^*\restriction_I \notin$ AP for the same reason as before. ♦

Remark 4.6.25 (Classical error bound)**:** The third condition in Theorem 4.6.21 (Closure by iteration) is usually far more subtle than necessary. In practice, is it useful to note this condition is satisfied in f verifies for some constants $\varepsilon, K > 0$ that

$$\text{for all } x \in I_n, y \in \mathbb{R}^m, \text{ if } \|x-y\| \leqslant \varepsilon \text{ then } y \in I \text{ and } \|f(x) - f(y)\| \leqslant K\|x-y\|$$

♦

Remark 4.6.26 (Dependency of $\mho$ in n)**:** In the statement of thereom, $\mho$ is only allowed to depend on $\|x\|$ whereas it might be useful to also make it depend on n. In fact the theorem is still true if the last condition is modified to be $\|x-y\| \leqslant e^{-\mho(\|x\|,n)-\mu}$. The proof is straightfoward: ♦

Chapter 5

PIVP versus Turing computability

What we have is a zoo of functions. Some of them are in their trees and we can't get them down.

In this chapter we give a purely continuous (time and space) definition of classical computability, and in particular of well-known complexity classes. Namely, we give a natural characterization of the P class in terms of PIVP. We also show that the $P_{C[a,b]}$ class is the same as AP over any interval $[a, b]$. In other words, we have a characterization of Computable Analysis in terms of PIVP computability. This is the first time such a characterization is obtained. To do so, an analog simulation of Turing machines is obtained, most notably using our iteration scheme on GPAC computable functions. Using this simulation, we can obtain FP and P, and Computable Analysis using the limit operation on computable functions.

This chapter is organized as follows:

- Section 5.1 (Introduction) provides some context on the subject.
- Section 5.2 (Simulating Turing machines) explains how to encode and simulate one step of a Turing machine using GPAC computable functions, in a robust way.
- Section 5.3 (Equivalences with Turing computability) gives a characterization of FP, P and Computable Analysis in terms of GPAC computable functions and PIVPs.

5.1 Introduction

Since the introduction of the P and NP complexity classes, much work has been done to build a well-developed complexity theory based on Turing Machines. In particular, classical computational complexity theory is based on limiting resources used by Turing machines, like time and space. Another approach is implicit computational complexity. The term "implicit" in "implicit computational complexity" can sometimes be understood in various ways, but a common point of many of the characterizations is that they provide (Turing or equivalent) machine-independent alternative definitions of classical complexity.

Implicit characterization theory has gained enormous interest in the last decade [DL12]. This has led to many alternative characterizations of complexity classes using recursive function, function algebras, rewriting systems, neural networks, lambda calculus and so on.

However, most of — if not all — of these models or characterizations are essentially discrete: in particular they are based on underlying models working with a discrete time on objects that are often defined in a discrete space.

Models of computation working on a continuous space have also been considered: they include Blum Shub Smale machines [BCSS98], and in some sense Computable Analysis [Wei00], or quantum computers [Fey82] which usually feature discrete-time and continuous-space. Machine-independent characterizations of the corresponding complexity classes have also been devised: see e.g. [BCdNM05, GM95]. However, the resulting characterizations are still essentially discrete, since time is still considered to be discrete.

In this chapter, we provide a purely analog machine-independent characterization of complexity classes. Indeed, our characterization relies only on a simple and natural class of ordinary differential equations (ODEs): P is characterized using ordinary differential equations with polynomial right-hand side. *This shows first that classical complexity theory can be presented in terms of ordinary differential equations problems.* This also shows as a side effect that solving ordinary differential equations leads to P-complete problems, even with a fixed dimension. More importantly, *we also provide machine-independent characterization of real complexity classes, as defined by Computable Analysis.* This shows that the complexity theory of real functions, although based on Turing Machines, has a very elegant definition in terms of differential equations. All these results have never been established before and are unexpected. The only similar results, to our knowledge, are those of [MC06] which characterize P and NP in the real space using more powerful operators than ODE solving, like limit-taking and operators that solve partial differential equations, and the result in [Kaw10] which shows that Lipschitz ordinary differential equations are polynomial-space complete.

A fundamental difficulty one faces when one tries to talk about time complexity for continuous time models of computation is that time is a problematic notion for many natural classes of systems: indeed, it is known that Turing machines can be simulated by various classes of ordinary differential equations or analog models. This can be often done even in real time: the state $y(T)$ at time $T \in \mathbb{N}$ of the solution of the ordinary differential equation encodes the state after the execution of T steps of the Turing machine. However, a troubling problem is that many models exhibit the so-called *Zeno's phenomenon*, that is to say the possibility of realizing an infinite number of discrete transitions in a finite amount of time. This was done in [EN02] using black holes, as mentioned earlier, to solve the Halting problem. Another example is in [AM98b], where it is shown that arithmetical sets can be recognized in finite time by piecewise constant derivative systems: basically such constructions are based on a time contraction. Actually, many continuous time systems might undergo arbitrary time contractions to simulate the computation of a Turing machine in an arbitrary short time (see e.g. [Ruo93], [Ruo94], [Moo96], [Bou97], [Bou99], [AD90], [CP02], [Dav01], [Cop98], [Cop02]).

In this chapter we give a fundamental contribution to this question: time of a computation for the GPAC can be measured as the length of the curve (i.e. length of the solution curve of the ordinary differential equation associated to the GPAC), or equivalently, as a couple measuring both time and "space". Doing so, we get to well defined complexity classes, that turn out to be the same as traditional complexity classes.

This is so natural at the end, that it might seem trivial, and only a statement that continuous time models of computation do satisfy the effective Church Turing thesis. However, various attempts of defining a clear complexity theory for continuous time models of computation have previously failed. Thus stating that the effective Church Turing thesis covers continuous time models of computation such as the GPAC requires at minimum a formal proof. See [GM02] and [SF98] for examples of natural analog complexity classes, and more in the survey [BC08].

Somehow, in that spirit, our contribution is to point out that the effective Church Turing thesis indeed holds for continuous time systems, if reasonable is understood as "defined by an ordinary differential equation with polynomial right-hand side", and if time is measured by "the length of the curve".

Remark 5.1.1 (Generable field)**:** Unless stated, in this entire chapter, $\mathbb{K}$ will refer to any generable field, for example $\mathbb{R}_G$. See Section 2.7 (Generable fields) for more details. ♦

5.2 Simulating Turing machines

In this section, we will show how to encode and simulate one step of a Turing machine with a computable function in a robust way.

5.2.I Turing Machine

There are many possible definitions of Turing machines. The exact kind we pick is usually not important but since we are going to simulate one with differential equations, it is important to specify all the details of the model. We will simulate deterministic, one-tape Turing machines, with complete transition functions.

Definition 5.2.1 (Turing Machine)**:** A *Turing Machine* is a tuple $\mathcal{M} = (Q, \Sigma, b, \delta, q_0, q_\infty)$ where $Q = [\![0, m-1]\!]$ are the states of the machines, $\Sigma = [\![0, k-2]\!]$ is the alphabet and $b = 0$ is the blank symbol, $q_0 \in Q$ is the initial state, $q_\infty \in Q$ is the halting state and $\delta : Q \times \Sigma \to Q \times \Sigma \times \{L, S, R\}$ is the transition function with $L = -1$, $S = 0$ and $R = 1$. We write $\delta_1, \delta_2, \delta_3$ as the components of δ. That is $\delta(q, \sigma) = (\delta_1(q, \sigma), \delta_2(q, \sigma), \delta_3(q, \sigma))$ where δ_1 is the new state, δ_2 the new symbol and δ_3 the head move direction. We require that $\delta(q_\infty, \sigma) = (q_\infty, \sigma, S)$. ♦

Remark 5.2.2 (Choice of k)**:** The choice of $\Sigma = [\![0, k-2]\!]$ will be crucial for the simulation, to ensure that the transition function be continuous. See Lemma 5.2.15 (Encoding range). ♦

For completeness, and also to make the statements of the next theorems easier, we introduce the notion of configuration of a machine, and define one step of a machine on configurations. This allows us to define the result of a computation. Contrary to many presentations, our machines not only accept or reject a word, but compute a output word.

Definition 5.2.3 (Configuration)**:** A *configuration* of $\mathcal{M}$ is a tuple $c = (x, \sigma, y, q)$ where $x \in \Sigma^*$ is the part of the tape at left of the head, $y \in \Sigma^*$ is the part at the right, $\sigma \in \Sigma$ is the symbol under the head and $q \in Q$ the current state. More precisely x_1 is the symbol immediately at the left of the head and y_1 the symbol immediately at the right. See Figure 5.2.7 for a graphical representation. The set of configurations of $\mathcal{M}$ is denoted by $C_\mathcal{M}$. The *initial configuration* is defined by $c_0(w) = (\lambda, b, w, q_0)$ and the *final configuration* by $c_\infty(w) = (\lambda, b, w, q_\infty)$ where λ is the empty word. ♦

Definition 5.2.4 (Step)**:** The *step* function of a Turing machine $\mathcal{M}$ is the function, acting on configurations, denoted by $\mathcal{M}$ and defined by:

$$\mathcal{M}(x, \sigma, y, q) = \begin{cases} (\lambda, b, \sigma' y, q') & \text{if } d = L \text{ and } x = \lambda \\ (x_{2..|x|}, x_1, \sigma' y, q') & \text{if } d = L \text{ and } x \neq \lambda \\ (x, \sigma', y, q') & \text{if } d = S \\ (\sigma' x, b, \lambda, q') & \text{if } d = R \text{ and } y = \lambda \\ (\sigma' x, y_1, y_{2..|y|}, q') & \text{if } d = R \text{ and } y \neq \lambda \end{cases} \quad \text{where} \begin{cases} q' = \delta_1(q, \sigma) \\ \sigma' = \delta_2(q, \sigma) \\ d = \delta_3(q, \sigma) \end{cases}$$

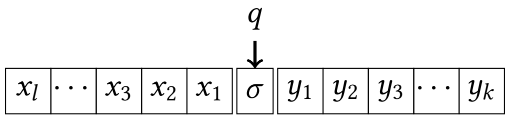


Figure 5.2.7: Example of generic configuration $c = (x, \sigma, y, q)$

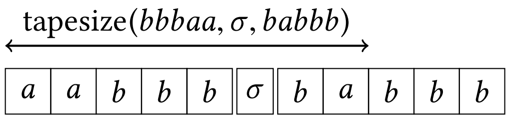


Figure 5.2.8: Example of configuration and tapesize

♦

Definition 5.2.5 (Result of a computation)**:** The *result of a computation* of $\mathcal{M}$ on a word $w \in \Sigma^*$ is defined by:

$$\mathcal{M}(w) = \begin{cases} x & \text{if } \exists n \in \mathbb{N}, \mathcal{M}^{[n]}(c_0(w)) = c_\infty(x) \\ \perp & \text{otherwise} \end{cases}$$

♦

Remark 5.2.6: The result of a computation is well-defined because we imposed that when a machine reaches a halting state, it does not move, change state or change the symbol under the head. ♦

5.2.II Polynomial interpolation

In order to implement the transition function of the Turing Machine, we will use a polynomial interpolation scheme (Lagrange interpolation). But since our simulation may have to deal with some amount of error in inputs, we have to investigate how this error propagates through the interpolating polynomial.

Definition 5.2.9 (Lagrange polynomial)**:** Let $d \in \mathbb{N}$ and $f : G \to \mathbb{R}$ where G is a finite subset of $\mathbb{R}^d$, we define

$$L_f(x) = \sum_{\bar{x} \in G} f(\bar{x}) \prod_{\substack{y \in G \\ y \neq \bar{x}}} \prod_{i=1}^{d} \frac{x_i - y_i}{\bar{x}_i - y_i}$$

♦

Lemma 5.2.10 (Lagrange interpolation)**:** *For any finite $G \subseteq \mathbb{K}^d$ and $f : G \to \mathbb{K}$, $L_f \in \mathrm{AP}$ and $L_f \restriction_G = f$.* ♦

Proof. The fact that L_f matches f on G is a classical calculation. Also L_f is a polynomial with coefficients in $\mathbb{K}$ so clearly it belongs to AP. ■

Remark 5.2.11 (Robustness of Lagrange interpolation)**:** It is customary to prove robustness of the interpolation, which means that on the neighborhood of G, L_f is nearly constant. However this result is a byproduct of the effective continuity of L_f, thanks to Theorem 4.6.4 (Modulus of continuity). ♦

We will often need to interpolate characteristic functions, that is polynomials that value 1 when $f(x) = a$ and 0 otherwise. For convenience we define a special notation for it.

Definition 5.2.12 (Characteristic interpolation)**:** Let $d \in \mathbb{N}$, $f : G \to \mathbb{R}$ where G is a finite subset of $\mathbb{R}^d$, $\alpha \in \mathbb{R}$, and define:

$$D_{f=\alpha}(x) = L_{f_\alpha}(x) \qquad D_{f\neq\alpha}(x) = L_{1-f_\alpha}(x) \qquad f_\alpha(x) = \begin{cases} 1 & \text{if } f(x) = \alpha \\ 0 & \text{otherwise} \end{cases}$$

♦

Lemma 5.2.13 (Characteristic interpolation)**:** *For any finite $G \subseteq \mathbb{K}^d$, $f : G \to \mathbb{K}$ and $\alpha \in \mathbb{K}$, $D_{f=\alpha}, D_{f\neq\alpha} \in \mathrm{AP}$.* ♦

Proof. Observe that $f_\alpha : G \to \{0,1\}$ and $\{0,1\} \subseteq \mathbb{K}$. Apply Lemma 5.2.10. ■

5.2.III Encoding

In order to simulate a machine, we will need to encode configurations with real numbers. There are several ways of doing so but not all of them are suitable for use when proving complexity results. This particular issue is discussed in Remark 5.3.2 (Encoding size). For our purpose, it is sufficient to say that we will encode a configuration as a tuple, we store the state and current letter as integers and the left and right parts of the tape as real numbers between 0 and 1. Intuitively, the tape is represented as two numbers whose digits in a particular basis are the letters of the tape.

Definition 5.2.14 (Real encoding)**:** Let $c = (x, \sigma, y, q)$ be a configuration of $\mathcal{M}$, the *real encoding* of c is $\langle c \rangle = (0.x, \sigma, 0.y, q) \in \mathbb{Q}\times\Sigma\times\mathbb{Q}\times Q$ where $0.x = x_1k^{-1}+x_2k^{-2}+\cdots+x_{|w|}k^{-|w|} \in \mathbb{Q}$. ♦

Lemma 5.2.15 (Encoding range)**:** *For any word $x \in [\![0, k-2]\!]^*$, $0.x \in \left[0, \frac{k-1}{k}\right]$.* ♦

Proof. $0 \leqslant 0.x = \sum_{i=1}^{|x|} x_ik^{-i} \leqslant \sum_{i=1}^{\infty}(k-2)k^{-i} \leqslant \frac{k-2}{k-1} \leqslant \frac{k-1}{k}$. ■

The same way we define the step function for Turing machines on configurations, we have to define a step function that works directly the encoding of configuration. This function is ideal in the sense that it is only defined over real numbers that are encoding of configurations.

Definition 5.2.16 (Ideal real step)**:** The *ideal real step* function of a Turing machine $\mathcal{M}$ is the function defined over $\langle C_\mathcal{M} \rangle$ by:

$$\langle \mathcal{M} \rangle_\infty (\tilde{x}, \sigma, \tilde{y}, q) = \begin{cases} \left(\operatorname{frac}(k\tilde{x}), \operatorname{int}(k\tilde{x}), \frac{\sigma'+\tilde{y}}{k}, q'\right) & \text{if } d = L \\ (\tilde{x}, \sigma', \tilde{y}, q') & \text{if } d = S \\ \left(\frac{\sigma'+\tilde{x}}{k}, \operatorname{int}(k\tilde{y}), \operatorname{frac}(k\tilde{y}), q'\right) & \text{if } d = R \end{cases} \quad \text{where} \quad \begin{cases} q' = \delta_1(q,\sigma) \\ \sigma' = \delta_2(q,\sigma) \\ d = \delta_3(q,\sigma) \end{cases}$$

♦

Lemma 5.2.17 ($\langle \mathcal{M} \rangle_\infty$ is correct)**:** *For any machine $\mathcal{M}$ and configuration c, $\langle \mathcal{M} \rangle_\infty (\langle c \rangle) = \langle \mathcal{M}(c) \rangle$.* ♦

Proof. Let $c = (x, \sigma, y, q)$ and $\tilde{x} = 0.x$. The proof boils down to a case analysis (the analysis is the same for x and y):

- If $x = \lambda$ then $\tilde{x} = 0$ so $\operatorname{int}(k\tilde{x}) = b$ and $\operatorname{frac}(k\tilde{x}) = 0 = 0.\lambda$ because $b = 0$.
- If $x \neq \lambda$, $\operatorname{int}(k\tilde{x}) = x_1$ and $\operatorname{frac}(k\tilde{x}) = 0.x_{2..|x|}$ because $k\tilde{x} = x_1 + 0.x_{2..|x|}$ and Lemma 5.2.15 (Encoding range).

■

The previous function was ideal but this is not enough to simulate a machine: the step function must be robust to small perturbations and must be computable. For this reason, we define a new step function with both features and that relates closely to the ideal function.

Definition 5.2.18 (Real step)**:** For any $\bar{x},\bar{\sigma},\bar{y},\bar{q}\in\mathbb{R}$ and $\mu\in\mathbb{R}_+$, define the *real step* function of a Turing machine $\mathcal{M}$ by:

$$\langle\mathcal{M}\rangle(\bar{x},\bar{\sigma},\bar{y},\bar{q},\mu)=\langle\mathcal{M}\rangle^*(\bar{x},\operatorname{rnd}^*(\bar{\sigma},\mu),\bar{y},\operatorname{rnd}^*(\bar{q},\mu),\mu)$$

where:

$$\langle\mathcal{M}\rangle^*(\bar{x},\bar{\sigma},\bar{y},\bar{q},\mu)=\langle\mathcal{M}\rangle^\star\left(\bar{x},\bar{y},L_{\delta_1}(\bar{q},\bar{\sigma}),L_{\delta_2}(\bar{q},\bar{\sigma}),L_{\delta_2}(\bar{q},\bar{\sigma}),\mu\right)$$

where:

$$\langle\mathcal{M}\rangle^\star\left(\bar{x},\bar{y},\bar{q},\bar{\sigma},\bar{d},\mu\right)=\begin{pmatrix}\operatorname{choose}\left[\operatorname{frac}^*(k\bar{x}),\bar{x},\frac{\bar{\sigma}+\bar{x}}{k}\right]\\ \operatorname{choose}\left[\operatorname{int}^*(k\bar{x}),\bar{\sigma},\operatorname{int}^*(k\bar{y})\right]\\ \operatorname{choose}\left[\frac{\bar{\sigma}+\bar{y}}{k},\bar{y},\operatorname{frac}^*(k\bar{y})\right]\\ \bar{q}\end{pmatrix}$$

where:

$$\operatorname{choose}[l,s,r]=D_{\mathrm{id}=L}(\bar{d})l+D_{\mathrm{id}=S}(\bar{d})s+D_{\mathrm{id}=R}(\bar{d})r$$

$$\operatorname{int}^*(x)=\operatorname{rnd}^*\left(x-\tfrac{1}{2}+\tfrac{1}{2k},\mu+\ln k\right)\qquad\operatorname{frac}^*(x)=x-\operatorname{int}^*(x)$$

rnd^* is defined in Definition 4.6.13 (Round)

♦

Theorem 5.2.19 (Real step is robust)**:** *For any machine $\mathcal{M}$, $c\in C_{\mathcal{M}}$, $\mu\in\mathbb{R}_+$ and $\bar{c}\in\mathbb{R}^4$, if $\|\langle c\rangle-\bar{c}\|\leqslant\frac{1}{2k^2}-e^{-\mu}$ then $\|\langle\mathcal{M}\rangle(\bar{c},\mu)-\langle\mathcal{M}(c)\rangle\|\leqslant k\|\langle c\rangle-\bar{c}\|$. Furthermore $\langle\mathcal{M}\rangle\in$ AP.* ♦

Proof. We begin by a small result about int^* and frac^*: if $\|\bar{x}-0.x\|\leqslant\frac{1}{2k^2}-e^{-\mu}$ then $\operatorname{int}^*(k\bar{x})=\operatorname{int}(k0.x)$ and $\|\operatorname{frac}^*(k\bar{x})-\operatorname{frac}(k0.x)\|\leqslant k\|\bar{x}-0.x\|$. Indeed, by Lemma 5.2.15 (Encoding range), $k0.x=n+\alpha$ where $n\in\mathbb{N}$ and $\alpha\in\left[0,\frac{k-1}{k}\right]$. Thus $\operatorname{int}^*(k\bar{x})=\operatorname{rnd}^*\left(k\bar{x}-\frac{1}{2}+\frac{1}{2k},\mu\right)=n$ because $\alpha+k\|\bar{x}-0.x\|-\frac{1}{2}+\frac{1}{2k}\in\left[-\frac{1}{2}+ke^{-\mu},\frac{1}{2}-ke^{-\mu}\right]$. Also, $\operatorname{frac}^*(k\bar{x})=k\bar{x}-\operatorname{int}^*(k\bar{x})=k\|\bar{x}-0.x\|+kx-\operatorname{int}(kx)=\operatorname{frac}(kx)+k\|\bar{x}-0.x\|$.

Write $\langle c\rangle=(x,\sigma,y,q)$ and $\bar{c}=(\bar{x},\bar{\sigma},\bar{y},\bar{q})$. Apply Definition 4.6.13 (Round) to get that $\operatorname{rnd}^*(\bar{\sigma},\mu)=\sigma$ and $\operatorname{rnd}^*(\bar{q},\mu)=q$ because $\|(\bar{\sigma},\bar{q})-(\sigma,q)\|\leqslant\frac{1}{2}-e^{-\mu}$. Consequently, $L_{\delta_i}(\bar{q},\bar{\sigma})=\delta_i(q,\sigma)$ and $\langle\mathcal{M}\rangle(\bar{c},\mu)=\langle\mathcal{M}\rangle^\star(\bar{x},\bar{y},q',\sigma',d')$ where $q'=\delta_1(q,\sigma)$, $\sigma'=\delta_2(q,\sigma)$ and $d'=\delta_3(q,\sigma)$. In particular $d'\in\{L,S,R\}$ so there are three cases to analyze.

- **If** $d'=L$ **then** $\operatorname{choose}[l,s,r]=l$, $\operatorname{int}^*(k\bar{x})=\operatorname{int}(kx)$, $\|\operatorname{frac}^*(k\bar{x})-\operatorname{frac}(kx)\|\leqslant k\|\bar{x}-x\|$ and $\left\|\frac{\sigma'+\bar{y}}{k}-\frac{\sigma'+y}{k}\right\|\leqslant\|\bar{x}-x\|$. Thus $\|\langle\mathcal{M}\rangle(\bar{c},\mu)-\langle\mathcal{M}\rangle_\infty(\langle c\rangle)\|\leqslant k\|\bar{c}-\langle c\rangle\|$. Conclude using Lemma 5.2.17 ($\langle\mathcal{M}\rangle_\infty$ is correct).
- **If** $d'=S$ **then** $\operatorname{choose}[l,s,r]=s$ so we immediately have that $\|\langle\mathcal{M}\rangle(\bar{c},\mu)-\langle\mathcal{M}\rangle_\infty(\langle c\rangle)\|\leqslant\|\bar{c}-\langle c\rangle\|$. Conclude using Lemma 5.2.17 ($\langle\mathcal{M}\rangle_\infty$ is correct).
- **If** $d'=R$ **then** $\operatorname{choose}[l,s,r]=r$ and everything else is similar to the case of $d'=L$.

Finally apply Lemma 5.2.10 (Lagrange interpolation), Theorem 4.6.14 (Round), Theorem 4.6.3 (Closure by arithmetic operations) and Theorem 4.6.6 (Closure by composition) to get that $\langle\mathcal{M}\rangle\in$ AP. ■

A very useful fact is that we can reencode words, in other words change alphabets, in polynomial time. We also get for free that we can compute the size of word assuming it does not have any blank character.

In some cases, we want to reencode words but Corollary 5.2.27 (Reencoding) does not apply. A very useful such case is a binary alphabet $\{0,1\}$ in basis 2. It cannot be decoded in general for continuity reasons, because there could be arbitrary close encoding with different decoding. Is it still possible, however, to decode under extra assumptions.

This theorem is a remarkable application of Theorem 4.6.21 (Closure by iteration) and will push nearly all the assumptions of the theorem to their extreme.

Theorem 5.2.20 (Word decoding): *Let $k_1, k_2 \in \mathbb{N}^*$ and $\kappa : [\![0, k_1 - 1]\!] \rightarrow [\![0, k_2 - 1]\!]$. There exists a function* $(\operatorname{decode}_\kappa :\subseteq \mathbb{R}\times\mathbb{N}\times\mathbb{R} \rightarrow \mathbb{R}) \in \mathrm{AP}$ *such that for any word $w \in [\![0, k_1 - 1]\!]^*$ and $\mu, \varepsilon \geqslant 0$:*

$$\text{if } \varepsilon \leqslant k_1^{-|w|}(1-e^{-\mu}) \text{ then } \operatorname{decode}_\kappa\left(\sum_{i=1}^{|w|} w_i k_1^{-i} + \varepsilon, |w|, \mu\right) = \left(\sum_{i=1}^{|w|} \kappa(w_i) k_2^{-i}, \#\{i | w_i \neq 0\}\right)$$

♦

Proof. We will iterate a function that works on tuple of the form (x, x', n, m, μ) where x is the remaining part to process, x' is the processed part, n the size of the processed part, m the number of nonzero symbols and μ will stay constant. The function will remove the "head" of x, reencode it with κ and "queue" on x', increasing n and m if the head is not 0.

In the remaining of this proof, we write $\overline{0.x}^{k_i}$ to denote $0.x$ in basis k_i instead of k. Define for any $x, y \in \mathbb{R}$ and $n \in \mathbb{N}$:

$$g(x,y,n,m,\mu) = \left(\operatorname{frac}^*(k_1 x), y + k_2^{-n-1} L_\kappa(\operatorname{int}^*(k_1 x)), n+1, m + D_{\mathrm{id}\neq 0}(\operatorname{int}^*(k_1 x)), \mu\right)$$

where:

$$\operatorname{int}^*(x) = \operatorname{rnd}^*\left(x - \tfrac{1}{2} + \tfrac{3e^{-\mu}}{4}, \mu\right) \qquad \operatorname{frac}^*(x) = x - \operatorname{int}^*(x)$$

and rnd^* is defined in Definition 4.6.13 (Round). Apply Lemma 5.2.10 (Lagrange interpolation) to get that $L_\kappa \in \mathrm{AP}$ and Lemma 5.2.13 (Characteristic interpolation) to get that $D_{\mathrm{id}\neq 0} \in \mathrm{AP}$. It follows that $g \in \mathrm{AP}$. We need by a small result about int^* and frac^*. For any $w \in [\![0, k_1]\!]^*$ and $x \in \mathbb{R}$, define the following proposition:

$$A(x,w,\mu) : -k_1^{-|w|}\frac{e^{-\mu}}{2} \leqslant x - \overline{0.w}^{k_1} \leqslant k_1^{-|w|}(1-e^{-\mu})$$

We will show that:

$$A(x,w,\mu) \Rightarrow \begin{cases} \operatorname{int}^*(k_1 x) = \operatorname{int}(k_1 \overline{0.w}^{k_1}) \\ \left|\operatorname{frac}^*(k_1 x) - \operatorname{frac}(k_1 \overline{0.w}^{k_1})\right| \leqslant k_1 \left|x - \overline{0.w}^{k_1}\right| \end{cases} \tag{5.2.21}$$

Indeed, in this case, since $|w| \geqslant 1$:

$$\begin{gathered}
-k_1^{1-|w|}\tfrac{e^{-\mu}}{2} \leqslant k_1 x - k_1\overline{0.w}^{k_1} \leqslant k_1^{1-|w|}(1-e^{-\mu}) \\
-k_1^{1-|w|}\tfrac{e^{-\mu}}{2} \leqslant k_1 x - w_1 \leqslant k_1^{1-|w|}(1-e^{-\mu}) + \overline{0.w_{2..|w|}}^{k_1} \\
-\tfrac{e^{-\mu}}{2} \leqslant k_1 x - w_1 \leqslant k_1^{1-|w|} - e^{-\mu} + \textstyle\sum_{i=1}^{|w|-1}(k_1-1)k_1^{-i} \\
-\tfrac{e^{-\mu}}{2} \leqslant k_1 x - w_1 \leqslant k_1^{1-|w|} - e^{-\mu} + 1 - k_1^{1-|w|} \\
-\tfrac{1}{2} - \tfrac{e^{-\mu}}{2} \leqslant k_1 x - \tfrac{1}{2} - w_1 \leqslant \tfrac{1}{2} - e^{-\mu} \\
-\tfrac{1}{2} + \tfrac{e^{-\mu}}{4} \leqslant k_1 x - \tfrac{1}{2} + \tfrac{3e^{-\mu}}{4} - w_1 \leqslant \tfrac{1}{2} - \tfrac{e^{-\mu}}{4}
\end{gathered}$$

And conclude by applying Theorem 4.6.14 (Round) because $\operatorname{int}(k_1\overline{0.w}^{k_1}) = w_1$. The result on frac follows trivially. It is then not hard to derive from (5.2.21) applied twice that:

$$\begin{gathered} A(x,w,\mu) \quad \wedge \quad A(x',w,\mu') \\ \Downarrow \\ \|g(x,y,n,m,\mu) - g(x',y',n',m',\nu)\| \leqslant 2k_1 \|(x,y,n,m,\mu)-(x',y',n',m',\mu')\| \end{gathered} \tag{5.2.22}$$

It also follows that proposition A is preserved by applying g:

$$A(x,w,\mu) \quad \Rightarrow \quad A(\operatorname{frac}(k_1x), w_{2..|w|}, \mu) \tag{5.2.23}$$

Furthermore, A is stronger for longer words:

$$A(x,w,\mu) \quad \Rightarrow \quad A(x, w_{1..|w|-1}, \mu) \tag{5.2.24}$$

Indeed, if we have $A(x,w,\mu)$ then:

$$\begin{gathered} -k_1^{-|w|}\tfrac{e^{-\mu}}{2} \leqslant x - \overline{0.w}^{k_1} \leqslant k_1^{-|w|}(1-e^{-\mu}) \\ -k_1^{-|w|}\tfrac{e^{-\mu}}{2} \leqslant x - \overline{0.w_{1..|w|-1}}^{k_1} \leqslant k_1^{-|w|}(1-e^{-\mu}) + w_{|w|}k_1^{-|w|} \\ -k_1^{1-|w|}\tfrac{e^{-\mu}}{2} \leqslant x - \overline{0.w_{1..|w|-1}}^{k_1} \leqslant k_1^{-|w|}(1-e^{-\mu}) + (k_1-1)k_1^{-|w|} \\ -k_1^{1-|w|}\tfrac{e^{-\mu}}{2} \leqslant x - \overline{0.w_{1..|w|-1}}^{k_1} \leqslant k_1^{-|w|}(k_1-e^{-\mu}) \\ -k_1^{1-|w|}\tfrac{e^{-\mu}}{2} \leqslant x - \overline{0.w_{1..|w|-1}}^{k_1} \leqslant k_1^{1-|w|}(1-e^{-\mu}) \end{gathered}$$

It also follows from the definition of g that:

$$A(x,w,\mu) \quad \Rightarrow \quad \|g(x,y,n,m,\mu)\| \leqslant \max(k_1, 1 + \|x,y,n,m,\mu\|) \tag{5.2.25}$$

Indeed, if $A(x,w,\mu)$ then $\operatorname{int}^*(k_1x) \in [\![0,k_1-1]\!]$ thus $L_\kappa(\operatorname{int}^*(k_1x)) \in [\![0,k_2]\!]$ and $D_{\mathrm{id}\neq 0}(\operatorname{int}^*(k_1x)) \in \{0,1\}$, the inequality follows easily. A crucial property of A is that it is open with respect to x:

$$A(x,w,\mu) \quad \wedge \quad |x-y| \leqslant e^{-|w|\ln k_1-\mu-\nu} \quad \Rightarrow \quad A(y,w,\mu-\ln\tfrac{3}{2}) \tag{5.2.26}$$

Indeed, if $A(x,w,\mu)$ and $|x-y| \leqslant e^{-|w|\ln k_1-\mu-\ln 4}$ we have:

$$\begin{gathered} -k_1^{-|w|}\tfrac{e^{-\mu}}{2} \leqslant x - \overline{0.w}^{k_1} \leqslant k_1^{-|w|}(1-e^{-\mu}) \\ -k_1^{-|w|}\tfrac{e^{-\mu}}{2} + y - x \leqslant y - \overline{0.w}^{k_1} \leqslant k_1^{-|w|}(1-e^{-\mu}) + y - x \\ -k_1^{-|w|}\tfrac{e^{-\mu}}{2} - |y-x| \leqslant y - \overline{0.w}^{k_1} \leqslant k_1^{-|w|}(1-e^{-\mu}) + |y-x| \\ -k_1^{-|w|}\tfrac{e^{-\mu}}{2} - e^{-|w|\ln k_1-\mu-\ln 4} \leqslant y - \overline{0.w}^{k_1} \leqslant k_1^{-|w|}(1-e^{-\mu}) + e^{-|w|\ln k_1-\mu-\ln 4} \\ -k_1^{-|w|}(e^{-\mu-\ln 4} + \tfrac{e^{-\mu}}{2}) \leqslant y - \overline{0.w}^{k_1} \leqslant k_1^{-|w|}(1-e^{-\mu}+e^{-\mu-\ln 4}) \\ -k_1^{-|w|}\tfrac{3e^{-\mu}}{4} \leqslant y - \overline{0.w}^{k_1} \leqslant k_1^{-|w|}(1-\tfrac{3e^{-\mu}}{4}) \\ -k_1^{-|w|}\tfrac{3e^{-\mu}}{4} \leqslant y - \overline{0.w}^{k_1} \leqslant k_1^{-|w|}(1-\tfrac{6e^{-\mu}}{4}) \\ -k_1^{-|w|}\tfrac{e^{\ln\frac{3}{2}-\mu}}{2} \leqslant y - \overline{0.w}^{k_1} \leqslant k_1^{-|w|}(1-e^{\ln\frac{3}{2}-\mu}) \end{gathered}$$

In order to formally apply Theorem 4.6.21 (Closure by iteration), define for any $n \in \mathbb{N}$:

$$I_n = \left\{(x,y,\ell,m,\mu) \in \mathbb{R}^2 \times \mathbb{R}_+^3 \mid \exists w \in [\![0,k_1-1]\!]^n, A(x,w,\mu)\right\}$$

It follows from (5.2.24) that $I_{n+1} \subseteq I_n$. It follows from (5.2.23) that $g(I_{n+1}) \subseteq I_n$. It follows from (5.2.25) that $\left\|g^{[n]}(x)\right\| \leqslant \max(k_1, \|x\| + n)$ for $x \in I_n$. Now assume that $X = (x,y,n,m,\mu) \in$

I_n, $\nu \in \mathbb{R}_+$ and[1] $\|X - X'\| \leqslant e^{-\|X\| - n\ln k_1 - \nu}$ where $X' = (x', y', n', m, \mu')$ then by definition $A(x, w, \mu)$ for some $w \in [\![0, k_1 - 1]\!]^n$. It follows from (5.2.26) that $A(y, w, \mu - \ln\frac{3}{2})$ since $\|X\| + n\ln k_1 \geqslant |w|\ln k_1 + \mu$. Thus by (5.2.22) we have $\|g(X) - g(X')\| \leqslant 2k_1\|X - X'\|$ which is enough by Remark 4.6.25 (Classical error bound). We are thus in good shape to apply Theorem 4.6.21 (Closure by iteration) and get $g_0^* \in \mathrm{AP}$. Define:

$$\mathrm{decode}_\kappa(x, n, \mu) = \pi_{2,4}(x, 0, 0, 0, \mu, n))$$

where $\pi_{2,4}$ is the second and fourth projection. Clearly $\mathrm{decode}_\kappa \in \mathrm{AP}$, it remains to see that it satisfies the theorem. We will prove this by induction on the size of $|w|$. More precisely we will prove that for $|w| \geqslant 0$:

$$\varepsilon \in [0, k_1^{-|w|}(1 - e^{-\mu})] \quad \Rightarrow \quad g^{[|w|]}(\overline{0.w}^{k_1} + \varepsilon, 0, 0, 0, \mu) = (k_1^{|w|}\varepsilon, \overline{0.\kappa(w)}^{k_2}, |w|, \#\{i|w_i \neq 0\}, \mu)$$

The case of $|w| = 0$ is trivial since it will act as the identity function:

$$\begin{aligned} g^{[|w|]}(\overline{0.w}^{k_1} + \varepsilon, 0, 0, 0, \mu) &= g^{[0]}(\varepsilon, 0, 0, 0, \mu) \\ &= (\varepsilon, 0, 0, 0, \mu) \\ &= (k_1^{|w|}\varepsilon, \overline{0.\kappa(w)}^{k_2}, |w|, \#\{i|w_i \neq 0\}, \mu) \end{aligned}$$

We can now show the induction step. Assume that $|w| \geqslant 1$ and define $w' = w_{1..|w|-1}$. Let $\varepsilon \in [0, k_1^{-|w|}(1 - e^{-\mu})]$ and define $\varepsilon' = k_1^{-|w|}w_{|w|} + \varepsilon$. It is clear that $\overline{0.w}^{k_1} + \varepsilon = \overline{0.w'}^{k_1} + \varepsilon'$. Then by definition $A(\overline{0.w'}^{k_1} + \varepsilon', |w|, \mu)$ so

$$\begin{aligned} g^{[|w|]}(\overline{0.w}^{k_1} + \varepsilon, 0, 0, 0, \mu) &= g(g^{[|w|-1]}(\overline{0.w'}^{k_1} + \varepsilon', 0, 0, 0, \mu)) \\ &= g(k_1^{|w|-1}\varepsilon', \overline{0.\kappa(w')}^{k_2}, |w'|, \#\{i|w'_i \neq 0\}, \mu) \qquad \text{By induction} \\ &= g(k_1^{-1}w_{|w|} + k_1^{|w|-1}\varepsilon, \overline{0.\kappa(w')}^{k_2}, |w'|, \#\{i|w'_i \neq 0\}, \mu) \\ &= (\mathrm{frac}^*(w_{|w|} + k_1^{|w|}\varepsilon), \qquad \text{Where } k_1^{-|w|}\varepsilon \in [0, 1 - e^{-\mu}] \\ &\qquad \overline{0.\kappa(w')}^{k_2} + k_2^{-|w'|-1}L_\kappa(\mathrm{int}^*(w_{|w|} + k_1^{|w|}\varepsilon)), \\ &\qquad |w'| + 1, \#\{i|w'_i \neq 0\} + D_{\mathrm{id}\neq 0}(\mathrm{int}^*(w_{|w|} + k_1^{|w|}\varepsilon)), \mu) \\ &= (k_1^{|w|}\varepsilon, \overline{0.\kappa(w')}^{k_2} + k_2^{-|w|}L_\kappa(w_{|w|}), \\ &\qquad |w|, \#\{i|w'_i \neq 0\} + D_{\mathrm{id}\neq 0}(w_{|w|}), \mu) \\ &= (k_1^{|w|}\varepsilon, \overline{0.\kappa(w')}^{k_2}, |w|, \#\{i|w_i \neq 0\}, \mu) \end{aligned}$$

We can now conclude to the result. Let $\varepsilon \in [0, k_1^{-|w|}(1 - e^{-\mu}]$ then $A(\overline{0.0w}^{k_1} + \varepsilon, |w|, \mu)$ so in particular $(\overline{0.w}^{k_1} + \varepsilon, 0, 0, 0, \mu) \in I_{|w|}$ so:

$$\begin{aligned} \mathrm{decode}_\kappa(\overline{0.w}^{k_1} + \varepsilon, |w|, \mu) &= \pi_{2,4}(g_0^*(\overline{0.w}^{k_1} + \varepsilon, 0, 0, 0, \mu)) \\ &= \pi_{2,4}(g^{[|w|]}(\overline{0.w}^{k_1} + \varepsilon, 0, 0, 0, \mu)) \\ &= \pi_{2,4}(\varepsilon, \overline{0.\kappa(w)}^{k_2}, |w|, \#\{i|w_i \neq 0\}, \mu) \\ &= (\overline{0.\kappa(w)}^{k_2}, \#\{i|w_i \neq 0\}) \end{aligned}$$

■

The previous result is very general and in fact optimal in the sense that no computable (and thus continuous) function can do better. However for most practical cases, a much simpler version can be derived, assuming the encoding is *sparse*, meaning that we encode in base k but only use symbols from 0 to $k - 2$ (instead of $k - 1$).

[1]We use Remark 4.6.26 (Dependency of $\mho$ in n) to allow a dependence of $\mho$ in n.

Corollary 5.2.27 (Reencoding)**:** *Let* $k_1, k_2 \in \mathbb{N}^*$ *and* $\kappa : [\![1, k_1 - 2]\!] \to [\![0, k_2 - 1]\!]$. *There exists a function* $(\operatorname{reenc}_\kappa :\subseteq \mathbb{R} \times \mathbb{N} \to \mathbb{R} \times \mathbb{N}) \in \mathrm{AP}$ *such that for any word* $w \in [\![1, k_1 - 2]\!]^*$ *and* $n \geqslant |w|$ *we have:*

$$\operatorname{reenc}_\kappa\left(\sum_{i=1}^{|w|} w_i k_1^{-i}, n\right) = \left(\sum_{i=1}^{|w|} \kappa(w_i) k_2^{-i}, |w|\right)$$

♦

Proof. The proof is immediate: extend κ with $\kappa(0) = 0$ and define

$$\operatorname{reenc}_\kappa(x, n) = \operatorname{decode}_\kappa(x, n, 0)$$

Since $n \geqslant |w|$, we can apply Theorem 5.2.20 (Word decoding) with $\varepsilon = 0$ to get the result. Note that stricly speaking, we are not applying the theorem to w but rather to w padded with as many 0 symbols as necessary, ie $w0^{n-|w|}$. Since w does not contain the symbol 0 so its length is the same as the number of non-blank symbols it contains. ■

Remark 5.2.28 (Nonreversible reencoding)**:** Note that the previous theorem and corrolary allows from nonreversible reencoding when $\kappa(\alpha) = 0$ or $\kappa(\alpha) = k_2 - 1$ for some $\alpha \neq 0$. For example, it allows one to reencode a word over $\{0, 1, 2\}$ with $k_1 = 4$ to a word over $\{0, 1\}$ with $k_2 = 2$ with $\kappa(1) = 0$ and $\kappa(2) = 1$ but the resulting number cannot be decoded in general (for continuity reasons). In some cases, only the more general Theorem 5.2.20 (Word decoding) provides a way to recover the encoding. ♦

In the next section, we will need to recover the size of the tape at the end of the computation. One way to do this is to keep track of the tape size during the computation, but this usually requires a modified machine and some delimiters on the tape. Instead, we use the previous theorem to recover the size from the encoding, assuming it does not contain any blank character.

Corollary 5.2.29 (Size recovery)**:** *For any machine* $\mathcal{M}$, *there exists a function* $(\operatorname{tsize}_{\mathcal{M}} : \langle C_{\mathcal{M}} \rangle \times \mathbb{N} \to \mathbb{N}) \in \mathrm{AP}$ *such that for any word* $w \in (\Sigma \setminus \{b\})^*$ *and any* $n \geqslant |w|$, $\operatorname{tsize}_{\mathcal{M}}(0.w, n) = |w|$. ♦

Proof. It is an immediate consequence of Corollary 5.2.27 (Reencoding) with $k_1 = k_2 = k$ and $\kappa = \mathrm{id}$ where we throw away the reencoding. ■

Our last tool is that of "digit extraction". In Theorem 5.2.20 (Word decoding) we saw that we can decode a value, as long as we are close enough to a word. In essence, this theorem works around the continuity problem by creating gaps in the domain of the definition. One problem with this approach is that on rare occasions we really want to extract some information about the encoding. How is it possible to achieve this without breaking the continuity requirement ? The compromise is to ask for *less information.* More precisely, instead of computing "the i^{th} bit of the encoding" we rather compute "$cos(x)$ where x is the i^{th} but of the encoding". For simplicity, we will only state this result for the binary encoding.

Theorem 5.2.30 (Extraction)**:** *There exists* extract $\in \mathrm{AP}$ *such that for any* $x \in \mathbb{R}$ *and* $n \in \mathbb{N}$*:*

$$\operatorname{extract}(x, n) = \cos(2\pi 2^n x)$$

♦

Proof. The proof relies of the well-known following trigonometric identity:

$$\cos(2x) = 2\cos^2(x) - 1$$

Indeed, define for any $x \in \mathbb{R}$:

$$f(x) = 2x^2 - 1$$

Clearly $f \in \mathrm{AP}$ and it follows that for any $x \in \mathbb{R}$ and $n \in \mathbb{N}$:

$$\cos(2\pi 2^n x) = f^{[n]}(\cos(2\pi x))$$

We now have to show that we can iterate f over $[-1, 1]$ because $\cos(2\pi\mathbb{R}) = [-1, 1]$. This turns out to be very easy because $f([-1, 1]) = [-1, 1]$ so the iterates of f over $[-1, 1]$ are trivially bounded:

$$\left|f^{[n]}(x)\right| \leqslant 1 \qquad x \in [-1, 1], n \in \mathbb{N}$$

And for any $x \in [-1, 1]$, $\mu \geqslant 0$ and $y \in \mathbb{R}$, if $|x - y| \leqslant e^{-\mu-\ln 6}$ then clearly $|y| \leqslant |x| + 1 \leqslant 2$ so:

$$|f(x) - f(y)| = 2|x - y||x + y| \leqslant 2e^{-\mu-\ln 6}3 \leqslant e^{-\mu}$$

This shows that we can apply Theorem 4.6.21 (Closure by iteration) and get $f_0^* \in \mathrm{AP}$. Define for any $x \in \mathbb{R}$ and $n \in \mathbb{N}$:

$$g(x, n) = f_0^*(\cos(2\pi x), n)$$

Clearly $g \in \mathrm{AP}$ since $\cos(2\pi\mathbb{R}) = [-1, 1]$ and by construction we have:

$$g(x, n) = f^{[n]}(\cos(2\pi x)) = \cos(2\pi 2^n x)$$

■

5.3 Equivalences with Turing computability

In this section, we fix an alphabet Γ and all languages are considered over Γ, so in particular $P \subset \Gamma^*$. It is common to take $\Gamma = \{0, 1\}$ but the proofs work for any finite alphabet. We will assume that Γ comes with an injective mapping $\gamma : \Gamma \to \mathbb{N}^*$, in other words every letter has an uniquely assigned positive number. By extension, γ applies letterwise over words.

5.3.I Equivalence with FP

We will provide a characterization of FP by introducing a notion of function *emulation* with the GPAC. This characterization builds on our notion of computability introduced in the previous chapter. More precisely, we can simulate an FP function by iterating the robust step function built in Section 5.2 (Simulating Turing machines) using Theorem 4.6.21 (Closure by iteration). In the other direction, we can solve a PIVP using Theorem 3.5.2 (PIVP complexity). At the end of this section, we also characterize FP as the discrete (i.e. integer) part of AP, since FP can be seen a set of functions over integers.

Definition 5.3.1 (Discrete emulation)**:** $f : \Gamma^* \to \Gamma^*$ is called *emulable* if there exists $g \in \mathrm{AP}_{\mathbb{R}_G}$ and $k \geqslant 1 + \max(\gamma(\Gamma))$ such that for any word $w \in \Gamma^*$:

$$g(\psi(w)) = \psi(f(w)) \qquad \text{where} \quad \psi(w) = \left(\sum_{i=1}^{|w|} \gamma(w_i)k^{-i}, |w|\right)$$

We say that g *emulates* f with k. ♦

Remark 5.3.2 (Encoding size)**:** The exact details of the encoding are not extremely important, however the size of the encoding is crucial. More precisely, the proof heavily relies on the fact that $\|\psi(w)\| \approx |w|$. Note that this works both ways:

- $\|\psi(w)\|$ must be polynomially bounded in $|w|$ so that Theorem 3.5.2 (PIVP complexity) runs in polynomial time in $|w|$.
- $\|\psi(w)\|$ must be polynomially lower bounded in $|w|$ so that we can recover the output length from the size of its encoding.

♦

It is useful to note that the choice of k in the definition is not important and can always be changed. This is useful to alphabet changes. See Lemma 5.3.4 (Emulation reencoding) for more details.

Theorem 5.3.3 (FP equivalence)**:** *$f \in \text{FP}$ if and only if f is emulable (with $k = 2 + \max(\gamma(\Gamma))$).*

♦

Proof. Let $f \in \text{FP}$, then there exists a Turing machine $\mathcal{M} = (Q, \Sigma, b, \delta, q_0, F)$ where $\Sigma = [\![0, k-2]\!]$ and $\gamma(\Gamma) \subset \Sigma \setminus \{b\}$, and a polynomial $p_{\mathcal{M}}$ such that for any word $w \in \Gamma^*$, $\mathcal{M}$ halts in at most $p_{\mathcal{M}}(|w|)$ steps, that is $\mathcal{M}^{[p_{\mathcal{M}}(|w|)]}(c_0(\gamma(w))) = c_\infty(\gamma(f(w)))$. Note that we assume that $p_{\mathcal{M}}(\mathbb{N}) \subseteq \mathbb{N}$. Also note that $\psi(w) = (0.\gamma(w), |w|)$ for any word $w \in \Gamma^*$.

Let $\mu = \ln(4k^2)$ and $h(c) = \mathcal{M}(c, \mu)$ for all $c \in \mathbb{R}^4$. Define $I_\infty = \langle C_{\mathcal{M}} \rangle$ and $I_n = I_\infty + [-\varepsilon_n, \varepsilon_n]^4$ where $\varepsilon_n = \frac{1}{4k^{2+n}}$ for all $n \in \mathbb{N}$. Note that $\varepsilon_{n+1} \leqslant \frac{\varepsilon_n}{k}$ and that that $\varepsilon_0 \leqslant \frac{1}{2k^2} - e^{-\mu}$. Apply Theorem 5.2.19 (Real step is robust) to get that $h \in \text{AP}$ and $h(I_{n+1}) \subseteq I_n$. In particular $\left\|h^{[n]}(\bar{c}) - h^{[n]}(c)\right\| \leqslant k^n \|c - \bar{c}\|$ for all $c \in I_\infty$ and $\bar{c} \in I_n$, for all $n \in \mathbb{N}$. Let $\delta \in \left[0, \frac{1}{2}\right[$ and define $J = \cup_{n \in \mathbb{N}} I_n \times [n-\delta, n+\delta]$. Apply Theorem 4.6.21 (Closure by iteration) to get $(h^* : J \to I_0) \in \text{AP}$ such that for all $c \in I_\infty$ and $n \in \mathbb{N}$ and $h^*(c, n) = h^{[n]}(c)$.

Let π_3 denote the third projection, that is $\pi_3(a, b, c, d) = c$, then $\pi_3 \in \text{AP}$. Define $g(y, \ell) = \pi_3(h^*(0, b, y, q_0, p_{\mathcal{M}}(\ell)))$ for $y \in \psi(\Gamma^*)$ and $\ell \in \mathbb{N}$. Note that $g \in \text{AP}$ and is well-defined. Indeed, if $\ell \in \mathbb{N}$ then $p_{\mathcal{M}}(\ell) \in \mathbb{N}$ and if $y = \psi(w) = 0.w$ then $(0, b, y, q_0) = \langle(\lambda, b, w, q_0)\rangle = \langle c_0(w)\rangle \in I_\infty$. Furthermore, by construction, for any word $w \in \Gamma^*$ we have:

$$\begin{aligned}
g(\psi(w), |w|) &= \pi_3\left(h^*(\langle c_0(w)\rangle, p_{\mathcal{M}}(|w|))\right) \\
&= \pi_3\left(h^{[p_{\mathcal{M}}(|w|)]}(c_0(w))\right) \\
&= \pi_3\left(\left\langle C_{\mathcal{M}}^{[p_{\mathcal{M}}(|w|)]}(c_0(w))\right\rangle\right) \\
&= \pi_3\left(\langle c_\infty(\gamma(f(w)))\rangle\right) \\
&= 0.\gamma(f(w)) = \psi(f(w))
\end{aligned}$$

Furthermore, the size of the tape cannot be greater than the initial size plus the number of steps, thus $|f(w)| \leqslant |w| + p_{\mathcal{M}}(|w|)$. Apply Corollary 5.2.29 (Size recovery) to get that $\text{tsize}_{\mathcal{M}}(g(\psi(w), |w|), |w| + p_{\mathcal{M}}(|w|)) = |f(w)|$ since $f(w)$ does not contain any blank character (this is true because $\gamma(\Gamma) \subset \Sigma \setminus \{b\}$). This proves that f is emulable because $g \in \text{AP}$ and $\text{tsize}_{\mathcal{M}} \in \text{AP}$.

Conversely, assume that f is emulable and apply Definition 5.3.1 (Discrete emulation) to get $g \in \text{AC}(\Upsilon, \Omega)$ where Υ, Ω are polynomials, and $k \in \mathbb{N}$. Let $w \in \Gamma^*$: we will describe an FP algorithm to compute $f(w)$. Apply Definition 4.2.1 (Analog computability) to g to get d, p, q and consider the following system:

$$y(0) = q(\psi(w)) \qquad y'(t) = p(y(t))$$

Note that by construction, y is defined over $\mathbb{R}_+$. Also note, **and that is absolutely crucial** that the coefficients of p, q belong to $\mathbb{R}_P$ which means that they are polynomial time computable. And since $\psi(w)$ is a pair of rational numbers with polynomial size (with respect to $|w|$), then $q(\psi(w)) \in \mathbb{R}_G^d \subseteq \mathbb{R}_P^d$ by Theorem 2.7.16 ($\mathbb{R}_G$ is generable subfield of $\mathbb{R}_P$).

The algorithm works in two steps: first we compute a rough approximation of the output to guess the size of the output. Then we rerun the system with enough precision to get the full output.

Let $t_w = \Omega(|w|, 2)$ for any $w \in \Sigma^*$, note that $t_w \in \mathbb{R}_P$ and that it is polynomially bounded in $|w|$ because Ω is a polynomial. Apply Theorem 3.5.2 (PIVP complexity) to compute $\tilde{y}$ such that $\|\tilde{y} - y(t_w)\| \leqslant e^{-2}$: this takes a time polynomial in $|w|$ because t_w is polynomially bounded and because[2] $\operatorname{Len}(0, t_w) \leqslant \operatorname{poly}(t_w, \sup_{[0,t_w]} \|y\|)$ and by construction, $\|y(t)\| \leqslant \Upsilon(\|\psi(w)\|, t_w)$ for $t \in [0, t_w]$ where Υ is a polynomial. Furthermore, by definition $\|y(t_w) - g(\psi(w))\| \leqslant e^{-2}$ thus $\|\tilde{y} - \psi(f(w))\| \leqslant 2e^{-2} \leqslant \frac{1}{3}$. But since $\psi(f(w)) = (0.\gamma(f(w)), |f(w)|)$, from $\tilde{y}_2$ we can find $|f(w)|$ by rounding to the closest integer (which is unique because at distance at most $\frac{1}{3}$). In other words, we can compute $|f(w)|$ in polynomial time in $|w|$. Note that this implies that $|f(w)|$ is at most polynomial in $|w|$.

Let $t'_w = \Omega(|w|, 2 + |f(w)| \ln k)$ which is polynomial in $|w|$ because Ω is a polynomial and $|f(w)|$ is at most polynomial in $|w|$. We can use the same reasoning and apply Theorem 3.5.2 (PIVP complexity) to get $\tilde{y}$ such that $\|\tilde{y} - y(t'_w)\| \leqslant e^{-2-|f(w)|\ln k}$. Again this takes a time polynomial in $|w|$. Furthermore, $\|\tilde{y}_1 - 0.\gamma(f(w))\| \leqslant 2e^{-2-|f(w)|\ln k} \leqslant \frac{1}{3}k^{-|f(w)|}$. We claim that this allows to recover $f(w)$ unambiguously in polynomial time in $|f(w)|$. Indeed, it implies that $\left\|k^{|f(w)|}\tilde{y}_1 - k^{|f(w)|}0.\gamma(f(w))\right\| \leqslant \frac{1}{3}$. Unfolding the definition shows that $k^{|f(w)|}0.\gamma(f(w)) = \sum_{i=1}^{|f(w)|} \gamma(f(w)_i)k^{|f(w)|-i} \in \mathbb{N}$ thus by rounding $k^{|f(w)|}\tilde{y}_1$ to the nearest integer, we recover $\gamma(f(w))$, and then $f(w)$. This is all done in polynomial time in $|f(w)|$, which proves that f is polynomial time computable. ■

An interesting question arises when looking at this theorem: does the choice of k in Definition 5.3.1 (Discrete emulation) matters, especially for the equivalence with FP ? Fortunately not, as long as k is large enough, as shown in the next lemma.

Lemma 5.3.4 (Emulation reencoding)**:** *Assume that $g \in$ AP emulates f with $k \in \mathbb{N}$. Then for any $k' \geqslant k$, there exists $h \in$ AP that emulates f with k'.* ♦

Proof. The proof follows from Corollary 5.2.27 (Reencoding). More precisely, ket $k' \geqslant k$ and define $\kappa : [\![1, k']\!] \to [\![1, k]\!]$ and $\kappa^{-1} : [\![1, k]\!] \to [\![1, k']\!]$ as follows:

$$\kappa(w) = \begin{cases} w & \text{if } w \in \gamma(\Gamma) \\ 1 & \text{otherwise} \end{cases} \qquad \kappa^{-1}(w) = w$$

In the following, ψ (resp. ψ' denotes ψ with basis k (resp. k'). Similarly, $0.w$ (resp. $0'.w$) denotes the rational encoding in basis k (resp. k'). Apply Corollary 5.2.27 (Reencoding) twice to get that $\operatorname{reenc}_\kappa, \operatorname{reenc}_{\kappa^{-1}} \in$ AP. Define:

$$h = \operatorname{reenc}_{\kappa^{-1}} \circ g \circ \operatorname{reenc}_\kappa$$

Note that $\gamma(\Gamma) \subseteq [\![1, k-1]\!]^* \subseteq [\![1, k'-1]\!]^*$ since γ never maps letters to 0 and $k \geqslant 1 + \max(\gamma(\Gamma))$ by definition. Consequently for $w \in \Gamma^*$:

$$\begin{aligned} h(\psi'(w)) &= h(0'.\gamma(w), |w|) && \text{By definition of } \psi' \\ &= \operatorname{reenc}_{\kappa^{-1}}(g(\operatorname{reenc}_\kappa(0'.\gamma(w), |w|))) \\ &= \operatorname{reenc}_{\kappa^{-1}}(g(0.\kappa(\gamma(w)), |w|)) && \text{Because } \gamma(w) \in [\![1, k']\!]^* \\ &= \operatorname{reenc}_{\kappa^{-1}}(g(0.\gamma(w), |w|)) && \text{Because } \gamma(w) \in \gamma(\Gamma)^* \\ &= \operatorname{reenc}_{\kappa^{-1}}(g(\psi(w))) && \text{By definition of } \psi \end{aligned}$$

[2]See Definition 3.3.15 (Pseudo-length of a PIVP) for the expression Len

$$
\begin{aligned}
&= \operatorname{reenc}_{\kappa^{-1}}(\psi(f(w))) && \text{Because } g \text{ emulates } f\\
&= \operatorname{reenc}_{\kappa^{-1}}(0.\gamma(f(w)), |f(w)|) && \text{By definition of } \psi\\
&= (0'.\kappa^{-1}(\gamma(f(w))), |f(w)|) && \text{Because } \gamma(f(w)) \in \gamma(\Gamma)^*\\
&= (0'.\gamma(f(w)), |f(w)|) && \text{By definition of } \kappa^{-1}\\
&= \psi'(f(w)) && \text{By definition of } \psi'
\end{aligned}
$$

■

The previous result was for single input function, which is sufficient in theory because we can always encode tuples of words using a single word or give Turing machines several input/output tapes. For what follows, it will be useful to have function with multiple inputs/ouputs without going through an encoding. We extend the notion of discrete encoding in the natural way to handle this case.

Definition 5.3.5 (Discrete emulation)**:** $f : (\Gamma^*)^n \to (\Gamma^*)^m$ is called *emulable* if there exists $g \in \mathrm{AP}_{\mathbb{R}_G}$ and $k \in \mathbb{N}$ such that for any word $\vec{w} \in (\Gamma^*)^n$:

$$g(\psi(\vec{w})) = \psi(f(\vec{w})) \qquad \text{where} \quad \psi(x_1, \ldots, x_\ell) = (\psi(x_1), \ldots, \psi(x_\ell))$$

and ψ is defined as in Definition 5.3.1. ♦

Remark 5.3.6 (Consistency)**:** It is trivial that Definition 5.3.5 (Discrete emulation) matches Definition 5.3.1 (Discrete emulation) in the case of unidimensional functions, thus the two definitions are consistent with each other. ♦

Theorem 5.3.7 (Multidimensional FP equivalence)**:** *Let* $f : (\Gamma^*)^n \to (\Gamma^*)^m$*. Then* $f \in \mathrm{FP}$ *if and only if* f *is emulable.* ♦

Proof. First note that we can always assume that $m = 1$ by applying the result componentwise. Similarly, we can always assume that $n = 2$ by applying the result repeatedly. Since FP is robust to the exact encoding used for pairs, we choose a particular encoding to prove the result. Let # be a fresh symbol not found in Γ and define $\Gamma^\# = \Gamma \cup \{\#\}$. We naturally extend γ to $\gamma^\#$ which maps $\Gamma^\#$ to $\mathbb{N}^*$ injectively. Let $h : \Gamma^{\#*} \to \Gamma^*$ and define for any $w, w' \in \Gamma^*$:

$$h^\#(w, w') = h(w\#w')$$

It follows[3] that

$$f \in \mathrm{FP} \text{ if and only if } \exists h \in \mathrm{FP} \text{ such that } h^\# = f$$

Assume that $f \in \mathrm{FP}$, then there exists $h \in \mathrm{FP}$ such that $h^\# = f$. Note that h naturally induces a function (still called) $h : \Gamma^{\#*} \to \Gamma^{\#*}$ so we can apply Theorem 5.3.3 (FP equivalence) to get that h is emulable over alphabet $\Gamma^\#$. Apply Definition 5.3.1 (Discrete emulation) to get $g \in \mathrm{AP}$ and $k \in \mathbb{N}$ that emulate h. In the remaining of the proof, ψ denotes encoding of Definition 5.3.1 for this particular k, in other words:

$$\psi(w) = \left(\sum_{i=1}^{|w|} \gamma^\#(w_i)k^{-i}, |w|\right)$$

Define for any $x, x' \in \mathbb{R}$ and $n, n' \in \mathbb{N}$:

$$\varphi(x, n, x', n) = \left(x + \left(\gamma^\#(\#) + x'\right) k^{-n-1}, n + m + 1\right)$$

[3]This is folklore, but mostly because this particular encoding of pairs is polytime computable.

We claim that $\varphi \in \mathrm{AP}$ and that for any $w, w' \in \Gamma^*$, $\varphi(\psi(w), \psi(w')) = \psi(w\#w')$. The fact that $\varphi \in \mathrm{AP}$ is immediate using Theorem 4.6.3 (Closure by arithmetic operations) and the fact that $n \mapsto k^{-n-1}$ is analog-polytime-computable[4]. The second fact is follows from a calculation:

$$\begin{aligned}
\varphi(\psi(w), \psi(w')) &= \varphi\left(\sum_{i=1}^{|w|} \gamma^\#(w_i)k^{-i}, |w|, \sum_{i=1}^{|w'|} \gamma^\#(w'_i)k^{-i}, |w'|\right)\\
&= \left(\sum_{i=1}^{|w|} \gamma^\#(w_i)k^{-i} + \left(\gamma^\#(\#) + \sum_{i=1}^{|w'|} \gamma^\#(w'_i)k^{-i}\right)k^{-|w|-1}, |w| + |w'| + 1\right)\\
&= \left(\sum_{i=1}^{|w\#w'|} \gamma^\#((w\#w')_i)k^{-i}, |w\#w'|\right)\\
&= \psi(w\#w')
\end{aligned}$$

Define $G = g \circ \varphi$, we claim that G emulates f with k. First $G \in \mathrm{AP}$ thanks to Theorem 4.6.6 (Closure by composition). Second, for any $w, w' \in \Gamma^*$, we have:

$$\begin{aligned}
G(\psi(w, w')) &= g(\varphi(\psi(w), \psi(w'))) && \text{By definition of } G \text{ and } \psi\\
&= g(\psi(w\#w')) && \text{By the above equality}\\
&= \psi(h(w\#w')) && \text{Because } g \text{ emulates } h\\
&= \psi(h^\#(w, w')) && \text{By definition of } h^\#\\
&= \psi(f(w, w')) && \text{By the choice of } h
\end{aligned}$$

Conversely, assume that f is emulable. Define $F : \Gamma^{\#^*} \to \Gamma^{\#^*} \times \Gamma^{\#^*}$ as follows for any $w \in \Gamma^{\#^*}$:

$$F(w) = \begin{cases} (w', w'') & \text{if } w = w'\#w'' \text{ where } w', w'' \in \Gamma^*\\ (\lambda, \lambda) & \text{otherwise} \end{cases}$$

Clearly $F_1, F_2 \in \mathrm{FP}$ so apply Theorem 5.3.3 (FP equivalence) to get that they are emulable. Thanks to Lemma 5.3.4 (Emulation reencoding), there exists h, g_1, g_2 that emulate f, F_1, f_2 respectively with the same k. Define:

$$H = h \circ (g_1, g_2)$$

Clearly $H \in \mathrm{AP}$ because $g_1, g_2, h \in \mathrm{AP}$. Furthermore, H emulates $f \circ F$ because for any $w \in \Gamma^{\#^*}$:

$$\begin{aligned}
H(\psi(w)) &= h(g_1(\psi(w)), g_2(\psi(w)))\\
&= h(\psi(g_1(w)), \psi(g_2(w))) && \text{Because } g_i \text{ emulates } F_i\\
&= h(\psi(F(w))) && \text{By definition of } \psi\\
&= \psi(f(F(w))) && \text{Because } h \text{ emulates } f
\end{aligned}$$

Since $f \circ F : \Gamma^{\#^*} \to \Gamma^{\#^*}$ is emulable, we can apply Theorem 5.3.3 (FP equivalence) to get that $f \circ F \in \mathrm{FP}$. It is now trivial so see that $f \in \mathrm{FP}$ because for any $w, w' \in \Gamma^*$:

$$f(w, w') = (f \circ F)(w\#w')$$

and $((w, w') \mapsto w\#w') \in \mathrm{FP}$ ∎

[4]Note that it works only because $n \geqslant 0$.

5.3.II Equivalence with P

We will now use this characterization of FP to give a characterization of P. Instead of simply using the notion of computability, we will characterize it in terms of PIVP directly. This will hopefully give a simpler and more natural characterization of P than FP. We choose to measure the complexity in terms of the length of the curve because it is more natural, but it would be easy to use the space and time instead. The idea of equivalence is to reuse the equivalence with FP and the ingredients of the equivalence between AP and ALP from Theorem 4.2.15 (Computable = length-computable). Again, we will make use of Theorem 3.5.2 (PIVP complexity) but this time, we will really use the length of the curve as a complexity measure.

Definition 5.3.8 (Discrete recognizability)**:** A language $\mathcal{L} \subseteq \Gamma^*$ is called *recognizable* if there exists $d \in \mathbb{N}$, $q \in \mathbb{R}_G^d[\mathbb{R}^2]$, $p \in \mathbb{R}_G^d[\mathbb{R}^d]$ and a polynomial $\Omega : \mathbb{R}_+ \to \mathbb{R}_+$ such that for all $w \in \Gamma^*$, there is a (unique) $y : \mathbb{R}_+ \to \mathbb{R}^d$ such that for all $t \in \mathbb{R}_+$:

- $y(0) = q(\psi(w))$ and $y'(t) = p(y(t))$ ► y satisfies a differential equation
- if $|y_1(t)| \geqslant 1$ then $|y_1(u)| \geqslant 1$ for all $u \geqslant t$ ► decision is stable
- if $w \in \mathcal{L}$ (resp. $\notin \mathcal{L}$) and $\operatorname{len}_y(0,t) \geqslant \Omega(|w|)$ then $y_1(t) \geqslant 1$ (resp. $\leqslant -1$) ► decision
- $\operatorname{len}_y(0,t) \geqslant t$ ► technical condition

♦

Theorem 5.3.9 (P equivalence)**:** $\mathcal{L} \in \mathrm{P}$ *if and only if* $\mathcal{L}$ *is recognizable.* ♦

Proof. The proof is based on the equivalence between AP and ALP, and the FP equivalence. Indeed, decidability can be seen as the computability of particular functions with boolean output. The only technical point is to make sure that the decision of the system is irreversible. To do that, we run the system from the FP equivalence (which will output 0 or 1) for long enough so that the output is approximate but good enough. Only then will another variable reach −1 or 1. The fact that the decision complexity is based on the length of the curve also makes the proof slightly more complicated because the system we build essentially takes a decision after a certain time (and not length).

Let $\mathcal{L} \in \mathrm{P}$, then there exists $f \in \mathrm{FP}$ and two distinct symbols $\bar{0}, \bar{1} \in \Gamma$ such that for any $w \in \Gamma^*$, $f(w) = \bar{1}$ if $w \in \mathcal{M}$ and $f(w) = \bar{0}$ otherwise. Let dec be defined by $\operatorname{dec}(k^{-1}\gamma(\bar{0})) = -2$ and $\operatorname{dec}(k^{-1}\gamma(\bar{1})) = 2$. Recall that $L_{\text{dec}} \in \mathrm{AP}$ by Lemma 5.2.10 (Lagrange interpolation). Apply Theorem 5.3.3 (FP equivalence) to get g and k that emulate f. Note in particular that for any $w \in \Gamma^*$, $f(w) \in \{\bar{0}, \bar{1}\}$ so $\psi(f(w)) = (\gamma(\bar{0})k^{-1}, 1)$ or $(\gamma(\bar{1})k^{-1}, 1)$. Define $g^*(x) = L_{\text{dec}}(g_1(x))$ and check that $g^* \in \mathrm{AP}$. Furthermore, $g^*(\psi(w)) = 2$ if $w \in \mathcal{L}$ and $g^*(\psi(w)) = -2$ otherwise, by definition of the emulation and the interpolation. Let Ω and Υ be polynomials such that $g^* \in \mathrm{AC}(\Upsilon, \Omega)$ and assume, without loss of generality, that they are increasing functions. Apply Definition 4.2.1 (Analog computability) to get d, p, q. Let $w \in \Gamma^*$ and consider the following system:

$$\begin{cases} y(0)= q(\psi(w)) \\ v(0)= \psi(w) \\ z(0)= 0 \\ \tau(0)= 0 \end{cases} \qquad \begin{cases} y'(t)= p(y(t)) \\ v'(t)= 0 \\ z'(t)= \operatorname{lxh}_{[0,1]}(\tau(t) - \tau^*, 1, y_1(t) - z(t)) \\ \tau'(t)= 1 \end{cases}$$

$$\tau^* = \Omega(v_2(t), \ln 2)$$

In this system, y computes $g^* f$, v is a constant variable used to store the input and in particular the input size ($v_2(t) = |w|$), $\tau(t) = t$ is used to keep the time and z is the decision variable. Let $t \in [0, \tau^*]$, then by Lemma 2.6.22 ("low-X-high" and "high-X-low"), $\|z'(t)\| \leqslant e^{-1-t}$ thus $\|z(t)\| \leqslant e^{-1} < 1$. In other words, at time τ^* the system has still not decided if $w \in \mathcal{L}$ or not. Let $t \geqslant \tau^*$, then by definition of Ω and since $v_2(t) = \psi_2(w) = |w| = \|\psi(w)\|$, $\|y_1(t) - g^*(\psi(w))\| \leqslant e^{-\ln 2}$. Recall that $g^*(\psi(w)) \in \{-2, 2\}$ and let $\varepsilon \in \{-1, 1\}$ such that $g^*(\psi(w)) = \varepsilon 2$. Then $\|y_1(t) - \varepsilon 2\| \leqslant \frac{1}{2}$ which means that $y_1(t) = \varepsilon\lambda(t)$ where $\lambda(t) \geqslant \frac{3}{2}$. Apply Lemma 2.6.22 ("low-X-high" and "high-X-low") and Definition 2.6.21 ("low-X-high" and "high-X-low") to conclude that z satisfies for $t \geqslant \tau^*$:

$$z(\tau^*) \in [-e^{-1}, e^{-1}] \qquad z'(t) = \phi(t)(\varepsilon\lambda(t) - z(t))$$

where $\phi(t) \geqslant 0$ and $\phi(t) \geqslant 1 - e^{-1}$ for $t \geqslant \tau^* + 1$. Let $z_\varepsilon(t) = \varepsilon z(t)$ and check that z_ε satisfies:

$$z_\varepsilon(\tau^*) \in [-e^{-1}, e^{-1}] \qquad z_\varepsilon'(t) \geqslant \phi(t)(\tfrac{3}{2} - z_\varepsilon(t))$$

It follows that z_ε is an increasing function and from a classical argument about differential inequalities that:

$$z_\varepsilon(t) \geqslant \frac{3}{2} - \left(\frac{3}{2} - z_\varepsilon(\tau^*)\right) e^{-\int_{\tau^*}^t \phi(u)du}$$

In particular for $t^* = \tau^* + 1 + 2\ln 4$ we have:

$$z_\varepsilon(t) \geqslant \frac{3}{2} - (\tfrac{3}{2} - z_\varepsilon(\tau^*))e^{-2\ln 4(1-e^{-1})} \geqslant \frac{3}{2} - 2e^{-\ln 4} \geqslant 1$$

This proves that $|z(t)|$ is an increasing function, so in particular once it has reached 1, it stays greater than 1. Furthermore, if $w \in \mathcal{L}$ then $z(t^*) \geqslant 1$ and if $w \notin \mathcal{L}$ then $z(t^*) \leqslant 1$. Also note that $\|(y, v, z, w)'(t)\| \geqslant 1$ for all $t \geqslant 1$. Also note that z is bounded by a constant, by a very similar reasoning. This shows that if $Y = (y, v, z, \tau)$, then $\|Y(t)\| \leqslant \operatorname{poly}(\|\psi(w)\|, t)$ because $\|y(t)\| \leqslant \Upsilon(\|\psi(w)\|, t)$. Consequently, there is a polynomial Υ^* such that $\|Y'(t)\| \leqslant \Upsilon^*$ (this is immediate from the expression of the system), and without loss of generality, we can assume that Υ^* is an increasing function. And since $\|Y'(t)\| \geqslant 1$, we have that $t \leqslant \operatorname{len}_Y(0, t) \leqslant t\sup_{u\in[0,t]} \|Y'(u)\| \leqslant t\Upsilon^*(\|\psi(w)\|, t)$. Define $\Omega^*(\alpha) = t^*\Upsilon^*(\alpha, t^*)$ which is a polynomial becase t^* is polynomially bounded in $\|\psi(w)\| = |w|$. Let t such that $\operatorname{len}_Y(0, t) \geqslant \Omega^*(|w|)$, then by the above reasoning, $t\Upsilon^*(|w|, t) \geqslant \Omega^*(|w|)$ and thus $t \geqslant t^*$ so $|z(t)| \geqslant 1$, i.e. the system has decided.

The other direction of the proof is easier: assume that $\mathcal{L}$ is recognizable by a GPAC. Apply Definition 5.3.8 (Discrete recognizability) to get d, q, p and Ω. Let $w \in \Gamma^*$ and consider the following system:

$$y(0) = q(\psi(w)) \qquad y'(t) = p(y(t))$$

We will show that we can decide in time polynomial in $|w|$ whether $w \in \mathcal{L}$ or not. Indeed, q is a polynomial with coefficients in $\mathbb{R}_G \subseteq \mathbb{R}_P$ by Theorem 2.7.16 ($\mathbb{R}_G$ is generable subfield of $\mathbb{R}_P$) and $\psi(w)$ is a rational number so $q(\psi(w)) \in \mathbb{R}_P^d$. Similarly, p has coefficients in $\mathbb{R}_P$. Finally, note that[5]:

$$\begin{aligned}\operatorname{Len}(0, t) &= \int_0^t \Sigma p \max(1, \|y(u)\|)^k du \\ &\leqslant t\Sigma p \max\left(1, \sup_{u\in[0,t]} \|y(u)\|^k\right)\end{aligned}$$

[5]See Definition 3.3.15 (Pseudo-length of a PIVP) for the expression Len

$$\begin{aligned}
&\leqslant t\Sigma p \max\left(1, \sup_{u\in[0,t]}\left(\|y(0)\| + \operatorname{len}_y(0,t)\right)^k\right)\\
&\leqslant t \operatorname{poly}(\operatorname{len}_y(0,t))\\
&\leqslant \operatorname{poly}(\operatorname{len}_y(0,t))
\end{aligned}$$

where the last inequality holds because $\operatorname{len}_y(0,t) \geqslant t$ thanks to the technical condition. We can now apply Theorem 3.5.2 (PIVP complexity) to conclude that we are able to compute $y(t) \pm e^{-\mu}$ in time polynomial in t, μ and $\operatorname{len}_y(0,t)$. At this point, there is a slight subtlety: intuitively we would like to evaluate y at time $\Omega(|w|)$ but it could be that the length of the curve is exponential at this time. Fortunately, the algorithm that solves the PIVP works by making small time steps, and at each step the length cannot increase by more than a constant[6]. This means that we can stop the algorithm as soon as the length is greater than $\Omega(|w|)$. Let t^* be the time at which the algorithm stops. Then the running time of the algorithm will be polynomial in t^*, μ and $\operatorname{len}_y(0,t^*) \leqslant \Omega(|w|) + \mathcal{O}(1)$. Finally, thanks to the technical condition, $t^* \leqslant \operatorname{len}_y(0,t^*)$ so this algorithm has running time polynomial in $|w|$ and μ. Take $\mu = \ln 2$ then we get $\tilde{y}$ such that $\|y(t^*) - \tilde{y}\| \leqslant \frac{1}{2}$. By definition of Ω, $y_1(t) \geqslant 1$ or $y_1(t) \leqslant -1$ so we can decide from $\tilde{y}_1$ if $w \in \mathcal{L}$ or not. ■

5.3.III Equivalence with Computable Analysis

In this section, we show our main result, that is the equivalence between GPAC/PIVP polynomial time computability, and real polynomial time computability (in the sense of Computable Analysis). This is probably the most surprising and beautiful result of this thesis. Indeed, it gives a purely analog and machine-independent characterization of real polynomial time computability, using a realistic model, namely the GPAC. We choose to state this theorem over intervals, see Remark 5.3.11 (Domain of definition) for possible extensions, and limitations.

We now give a high-level overview of the proof. Given $x \in [a,b]$ and $\mu \in \mathbb{N}$, we will compute an approximation of $f(x) \pm 2^{-\mu}$ and take the limit when $\mu \to \infty$ using Theorem 4.6.17 (Closure by limit). To compute f, we will use Theorem 1.3.4 (Alternative definition of computable functions) which provides us will a polynomial time computable function g that computes f over rationals, and m a modulus of continuity. All we have to do is simulate g with input $\tilde{x}$ and μ, where $\tilde{x} = x \pm 2^{-m(\mu)}$ because we can only feed the machine with a finite input of course. The nontrivial part of the proof is how to obtain the encoding of $\tilde{x}$ from x and μ. Indeed, the encoding is a discrete quantity whereas x is real number, so by a simple continuity argument, one can see that no such function can exist. The trick is to proceed as we did for the limit operation: from x and μ, we can compute two encodings ψ_1 and ψ_2 such that at least one of them is valid, and we know which one it is. So we are going to simulate g on both inputs and then select the result. Again, the select operation cannot be done continuously unless we agree to "mix" both results, i.e. we will compute $\alpha g(\psi_1) + (1-\alpha)g(\psi_2)$. The trick is to ensure that $\alpha = 1$ or 0 when only one encoding is valid, $\alpha \in]0,1[$ when both are valid (by "when" we mean with respect to x). This way, the mixing will ensure continuity but in fact when both encodings are valid, the outputs are nearly the same so we are still computing f.

In the other direction, the proof is much easier using Theorem 3.5.2 (PIVP complexity): we simply simulate the system long enough to get the desired precision and use Theorem 2.7.16 ($\mathbb{R}_G$ is generable subfield of $\mathbb{R}_P$) to get that all the coefficients are polytime computable thus proving that the solution can be computed in polynomial time.

[6]For the unconvinced reader, it is still possible to write this argument formally by running the algorithm for increasing values of t, starting from a very small value and making sure that at each step the increase in the length of the curve is at most constant

Theorem 5.3.10 (Main equivalence): *For any $a, b \in \mathbb{R}_P$ and $f \in C^0([a,b],\mathbb{R})$, f is polynomial time computable if and only if $f \in \mathrm{AP}_{\mathbb{R}_P}$.* ♦

Proof. Assume that the theorem is true for functions in $C^0([0,1/2])$, then we claim the theorem follows. Indeed, if $f \in C^0([a,b],\mathbb{R})$ is polynomial time computable, then there exists[7] $m, M \in \mathbb{R}_P$ such that $m < f(x) < M$ for all $x \in [a,b]$. Define for $\alpha \in [0,1/2]$:

$$g(\alpha) = \frac{f(a + 2\alpha(b-a)) - m}{2(M-m)}$$

then clearly $g \in C^0([0,1/2])$ and is polytime computable because $a, b, m, M \in \mathbb{R}_P$. It follows that $g \in \mathrm{AP}_{\mathbb{R}_P}$ and then $f \in \mathrm{AP}_{\mathbb{R}_P}$ by the closure properties of AP. Conversely, if $f \in C^0([0,1/2])$ belongs to $\mathrm{AP}_{\mathbb{R}_P}$ then there also exists[8] $m, M \in \mathbb{R}_P$ as above and the reasoning is exactly the same. In the remaining of the proof, we assume that $f \in C^0([0,1/2])$. This restriction is useful to simplify the encoding used later in the proof.

Let $f \in C^0([0,1/2])$ be a polynomial time computable function. Apply Theorem 1.3.4 (Alternative definition of computable functions) to get g and m (we renamed ψ to g to avoid a name clash). Note that $g : \mathbb{Q} \cap [0,1/2] \times \mathbb{N} \to \mathbb{Q} \cap [0,1/2]$ has its second argument written in unary. In order to apply the FP characterization, we need to discuss the encoding of rational numbers and unary integers. Let us choose a binary alphabet $\Gamma = \{0,1\}$ with $\gamma(0) = 1$ and $\gamma(1) = 2$ and define for any $w, w' \in \Gamma^*$:

$$\psi_{\mathbb{N}}(w) = |w| \qquad \psi_{\mathbb{Q}}(w) = \sum_{i=1}^{|w|} w_i 2^{-i}$$

Note that $\psi_{\mathbb{Q}}$ is a bijection from Γ^* to $\mathbb{Q} \cap [0,1[$. Define for any $w, w' \in \Gamma^*$:

$$g_\Gamma(w,w') = \psi_{\mathbb{Q}}^{-1}(g(\psi_{\mathbb{Q}}(w), \psi_{\mathbb{N}}(w'))$$

Since ψ_Q is a polytime computable encoding, then $g_\Gamma \in \mathrm{FP}$ because it has running time polynomial in the size of $\psi_Q(w)$ and the (unary) value of $\psi_{\mathbb{N}}(w')$, which are the size of w and w' respectively, by definition of $\psi_{\mathbb{Q}}$ and $\psi_{\mathbb{N}}$. Apply Theorem 5.3.7 (Multidimensional FP equivalence) to get that g_Γ is emulable. Thus there exits $h \in \mathrm{AP}$ and $k \in \mathbb{N}$ such that for all $w, w' \in \Gamma^*$:

$$h(\psi(w,w')) = \psi(g_\Gamma(w,w'))$$

where ψ is defined as in Definition 5.3.5, for this specific value of k. Define $\kappa : [\![0, k-2]\!] \to \{0,1\}$ by $\kappa(\gamma(0)) = 0$ and $\kappa(\gamma(1)) = 1$ and $\kappa(\alpha) = 0$ otherwise, and define:

$$\psi_{\mathbb{Q}}^*(x,n) = \mathrm{reenc}_{\kappa,1}(x,n)$$

It follows from Corollary 5.2.27 (Reencoding) that $\psi_{\mathbb{Q}}^* \in \mathrm{AP}$ and:

$$\psi_{\mathbb{Q}}^*(\psi(w)) = \mathrm{reenc}_{\kappa,1}\left(\sum_{i=1}^{|w|} \gamma(w_i)k^{-i}, |w|\right) = \sum_{i=1}^{|w|} \kappa(\gamma(w_i))2^{-i} = \sum_{i=1}^{|w|} w_i 2^{-i} = \psi_{\mathbb{Q}}(w)$$

We can now define:

$$g_\Gamma^*(x,n,x',n') = \psi_{\mathbb{Q}}^*(h(x,n,x',n')$$

and get that for any $w, w' \in \Gamma^*$:

$$g_\Gamma^*(\psi(w,w')) = \psi_{\mathbb{Q}}^*(h(\psi(w,w'))) = \psi_{\mathbb{Q}}^*(\psi(g_\Gamma(w,w')) = \psi_{\mathbb{Q}}(g_\Gamma(w,w')) = g(\psi_{\mathbb{Q}}(w), \psi_{\mathbb{N}}(w'))$$

[7]To see that, observe that any polytime computable function is bounded by a polynomial.

[8]By Proposition 4.6.5, functions in AP are bounded by a polynomial.

Let us summarize what we have done so far: we built $g^*_\Gamma \in \mathrm{AP}$ that, if provided with the encoding of w, w', compute $g(\psi_\mathbb{Q}(w), \psi_\mathbb{N}(w'))$. To use this function, we need to be able to compute, from the input $x \in [0,1]$ and the requested precision $\mu \geqslant 0$, words w, w' such that $|w'| \geqslant \mu$ and $|x - \psi_\mathbb{Q}(w)| \leqslant 2^{-m(\psi_\mathbb{N}(w'))}$ so that we can run g^*_Γ and get an approximation of $f(x) \pm 2^{-\mu}$. The problem is that for continuity reason, it is impossible to compute such w, w' in general. This is where mixing comes into play: given x and μ, we will compute two pairs w, w' and u, u' such that at least one of them satisfies the above criteria. We will then apply g^*_Γ on both of them and mix the result.

Define[9] $\iota : \Gamma \to [\![0, k-1]\!]$ by $\iota = \gamma$. Apply Theorem 5.2.20 (Word decoding) and Theorem 5.2.30 (Extraction) to get $\mathrm{decode}_\iota, \mathrm{extract} \in \mathrm{AP}$. Define for any $n \in \mathbb{N}$:

$$u(n) = \left(\tfrac{1-k^{-n}}{k-1}, n\right)$$

Clearly $u \in \mathrm{AP}$ and one checks that $u(n) = \psi(0^n)$ because:

$$\psi(0^n) = \left(\sum_{i=1}^{n} \gamma(0)k^{-i}, n\right) = \left(\tfrac{1}{k}\tfrac{1-k^n}{1-k}, n\right) = u(n)$$

Now define for any $n \in \mathbb{N}$ and relevant[10] $x \in [0,1]$:

$$v(x,n) = (\mathrm{decode}_{\iota,1}(x,n,2), n)$$

It follows from Theorem 5.2.20 (Word decoding) and the fact that $1 - e^{-2} \geqslant \frac{2}{3}$ that:

$$\text{if } x = \psi_\mathbb{Q}(w) + \varepsilon \text{ for some } w \in \Gamma^n \text{ and } \varepsilon \in \left[0, 2^{-n}\tfrac{2}{3}\right] \text{ then } v(x,n) = \psi(w)$$

Now define for any $n \in \mathbb{N}$ and relevant $x \in [0,1]$:

$$\begin{aligned} f_0(x,n) &= g^*_\Gamma(v(x,n), u(n)) \\ f_1(x,n) &= g^*_\Gamma\left(v\left(x + 2^{-n-1}, n\right), u(n)\right) \\ i(x,n) &= \tfrac{1}{2} + \mathrm{extract}\left(x + 2^{-n}\tfrac{1}{6}, n\right) \end{aligned}$$

From the domain of definition of v, it follows that:

$$\bigcup_{w \in \Gamma^n} \left[\psi_\mathbb{Q}(w), \psi_\mathbb{Q}(w) + 2^{-n}\tfrac{2}{3}\right] \subseteq \mathrm{dom}\, f_0$$

$$\bigcup_{w \in \Gamma^n} \left[\psi_\mathbb{Q}(w) - 2^{-n-1}, \psi_\mathbb{Q}(w) + 2^{-n}\tfrac{1}{6}\right] \subseteq \mathrm{dom}\, f_1$$

First off, check that for any $n \in \mathbb{N}$:

$$\bigcup_{w \in \times\Gamma^n} \left[\psi_\mathbb{Q}(w), \psi_\mathbb{Q}(w) + 2^{-n}\right[= [0, 1[$$

Check that for any $n \in \mathbb{N}$, $\varepsilon \in [0, 2^{-n}[$, $n \in \mathbb{N}$ and $w \in \Gamma^n$:

$$i(\psi_\mathbb{Q}(w) + \varepsilon, n) = \tfrac{1}{2} + \cos(2\pi 2^n \varepsilon + \tfrac{\pi}{3})$$

It follows that any $n \in \mathbb{N}$, $\varepsilon \in [0, 2^{-n}[$, $w \in \Gamma^n$ and $x = \psi_\mathbb{Q}(w) + \varepsilon$ we have:

$$\varepsilon \in \left[0, 2^{-n}\tfrac{1}{6}\right[\qquad \Rightarrow \qquad i(x,n) \in [0, 1[$$

[9]This is a technicality because decode_ι will encode the output in basis k if $\iota : \Gamma \to [\![0, k-1]\!]$.

[10]We will discuss the domain of definition of v right after.

$$\begin{aligned}
\varepsilon \in \left[2^{-n}\tfrac{1}{6}, 2^{-n}\tfrac{1}{2}\right] &\quad\Rightarrow\quad i(x,n) \leqslant 0\\
\varepsilon \in \left]2^{-n}\tfrac{1}{2}, 2^{-n}\tfrac{2}{3}\right[&\quad\Rightarrow\quad i(x,n) \in]0,1[\\
\varepsilon \in \left[2^{-n}\tfrac{2}{3}, 2^{-n}\right[&\quad\Rightarrow\quad i(x,n) \geqslant 1
\end{aligned}$$

Thus:

$$\{(x,n) \mid i(x,n) < 1\} \subseteq \operatorname{dom} f_0 \qquad\qquad \{(x,n) \mid i(x,n) > 0\} \subseteq \operatorname{dom} f_1$$

Define for any $x \in [0,1/2]$ and $n \in \mathbb{N}$:

$$g^*(x,n) = \operatorname{mix}(i, f_0, f_1)(x,n)$$

We can thus apply Theorem 4.6.16 (Closure by mixing) to get that $g^* \in \mathrm{AP}$. Note that g^* is defined over $[0,1[\times\mathbb{N}$ which obviously contains $[0,1/2]\times\mathbb{N}$. We will now see that g^* approximates f and conclude that $f \in \mathrm{AWP}$. To do so, we will show the following statement by a case analysis, for all $x \in [0,1/2]$ and $n \in \mathbb{N}$:

$$\exists y,z \in \mathbb{Q}\cap[0,1/2], \alpha \in [0,1], |x-y|, |x-z| \leqslant 2^{-n} \text{ and } g^*(x,n) = \alpha g(y,n) + (1-\alpha) g(z,n)$$

To see that, first note that there exists[11] $w \in \Gamma^n$ such that[12] $x \in [\psi_{\mathbb{Q}}(w), \psi_{\mathbb{Q}}(w) + 2^{-n}]$. Furthermore, since $x \in [0,1/2]$, we can always assume that $w \in \{0\}\times\Gamma^{n-1}$. Write $\varepsilon = x - \psi_{\mathbb{Q}}(w)$, then there are four possible cases. It will be useful to keep in mind that $u(n) = \psi(0^n)$ as shown previously and that $\psi_{\mathbb{N}}(0^n) = n$. Also remember that we showed that $g^*_\Gamma(\psi(w,w')) = g(\psi_{\mathbb{Q}}(w), \psi_{\mathbb{N}}(w'))$. In almost all cases, we will define $y = \psi_{\mathbb{Q}}(w)$ and thus $|x-y| = \varepsilon \leqslant 2^{-n}$.

- **If** $\varepsilon \in \left[0, 2^{-n}\tfrac{1}{6}\right[$ **then** $i(x,n) \in [0,1[$ thus $g^*(x,n) = i(x,n) f_0(x,n) + (1 - i(x,n)) f_1(x,n)$. By construction of v, $v(x,n) = \psi(w)$ thus $f_0(x,n) = g^*_\Gamma(\psi(w), \psi(0^n)) = g^*_\Gamma(\psi(w, 0^n)) = g(\psi_{\mathbb{Q}}(w), \psi_{\mathbb{N}}(0^n)) = g(y,n)$. Since, $x + 2^{-n-1} - \psi_{\mathbb{Q}}(w) \in [0, 2^{-n}\tfrac{2}{3}]$ we similarly have $v(x + 2^{-n-1}, n) = \psi(w)$ and thus $f_1(x,n) = f_0(x,n) = g(y,n)$. It follows that $g^*(x,n) = g(y,n)$. So in this case, $z = y \in [0,1/2]$ and α can be anything.

- **If** $\varepsilon \in \left[2^{-n}\tfrac{1}{6}, 2^{-n}\tfrac{1}{2}\right]$ **then** $i(x,n) \leqslant 0$ thus $g^*(x,n) = f_0(x,n)$. By construction of v, $v(x,n) = \psi(w)$ thus $f_0(x,n) = g^*_\Gamma(\psi(w), \psi(0^n)) = g^*_\Gamma(\psi(w, 0^n)) = g(\psi_{\mathbb{Q}}(w), \psi_{\mathbb{N}}(0^n)) = g(y,n)$. It follows that $g^*(x,n) = g(y,n)$. So in this case, $z = y \in [0,1/2]$ and α can be anything.

- **If** $\varepsilon \in \left]2^{-n}\tfrac{1}{2}, 2^{-n}\tfrac{2}{3}\right[$ **then** $i(x,n) \in [0,1[$ thus $g^*(x,n) = i(x,n) f_0(x,n) + (1 - i(x,n)) f_1(x,n)$. By construction of v, $v(x,n) = \psi(w)$ thus $f_0(x,n) = g^*_\Gamma(\psi(w), \psi(0^n)) = g^*_\Gamma(\psi(w, 0^n)) = g(\psi_{\mathbb{Q}}(w), \psi_{\mathbb{N}}(0^n)) = g(y,n)$. However, $x + 2^{-n-1} - \psi_{\mathbb{Q}}(w) \in [2^{-n}, 2^{-n}(1 + \tfrac{1}{6})]$. Thus define $w' \in \Gamma^n$ such that[13] $\psi_{\mathbb{Q}}(w') = \psi_{\mathbb{Q}}(w) + 2^{-n}$ and define $z = \psi_{\mathbb{Q}}(w')$. It follows that $x - \psi_{\mathbb{Q}}(w') \in [0, 2^{-n}\tfrac{1}{6}]$ thus $v(x + 2^{-n-1}, n) = \psi(w')$ and thus $f_1(x,n) = f_0(x,n) = g(z,n)$. It follows that $g^*(x,n) = \alpha g(y,n) + (1-\alpha) g(z,n)$ where $\alpha = i(x,n) \in [0,1]$. Furthermore, $|z - x| \leqslant 2^{-n}$ by construction of w'.

- **If** $\varepsilon \in \left]2^{-n}\tfrac{2}{3}, 2^{-n}\right]$ **then** $i(x,n) \geqslant 1$ thus $g^*(x,n) = f_1(x,n)$. Define $w' \in \Gamma^n$ such that[13] $\psi_{\mathbb{Q}}(w') = \psi_{\mathbb{Q}}(w) + 2^{-n}$ and define $z = \psi_{\mathbb{Q}}(w')$. It follows that $x - \psi_{\mathbb{Q}}(w') \in [0, 2^{-n}\tfrac{1}{2}]$ thus $v(x + 2^{-n-1}, n) = \psi(w')$ and thus $f_1(x,n) = f_0(x,n) = g(z,n)$. So in this case, $y = z \in [0,1/2]$ and α can be anything.

[11]It may not be unique since we closed the interval on both sides in order to get all of $[0,1/2]$ with words in $\{0\}\times\Gamma^n$. If we opened the interval on the right, we would only get $[0,1/2[$ with such words.

[12]The use of this assumption will become later on. Essentially, it is there to ensure that the y we construct belongs to $[0,1/2]$ so that we can apply the function g to it.

[13]This is always possible, formally if w is seen as a number, written in binary, then w' is $w+1$.

We are now in position to conclude thanks to the modulus of continuity of f. Recall that by definition, m is a polynomial such that for any $x, y \in [0, 1/2]$ and $k \in \mathbb{N}$, if $|x - y| \leqslant 2^{-m(k)}$ then $|f(x) - f(y)| \leqslant 2^{-k}$. Without loss of generality, we can assume[14] that $m(\mathbb{N}) \subseteq \mathbb{N}$ and $m(n) \geqslant n$. Now define for any $x \in [0, 1/2]$ and $n \in \mathbb{N}$:

$$g^{**}(x, n) = g^*(x, m(n + 1))$$

Clearly $g \in \text{AP}$ since $g \in \text{AP}$ and m is a polynomial. Let $x \in \mathbb{N}$ and $n \in \mathbb{N}$. Then we have shown that there exists $y, z \in \mathbb{Q} \cap [0, 1/2]$ and $\alpha \in \mathbb{N}$ such that $|x - y|, |x - z| \leqslant 2^{-m(n+1)}$ and $g^*(x, m(n+1)) = \alpha g(y, m(n+1)) + (1-\alpha) g(z, m(n+1))$. By definition of g, $|g(y, m(n+1)) - f(y)| \leqslant 2^{-m(n+1)} \leqslant 2^{-n-1}$ since $m(n + 1) \geqslant 2$. Similarly, $|g(z, m(n + 1)) - f(z)| \leqslant 2^{-n-1}$. Furthermore, $|f(y) - f(x)|, |f(z) - f(x)| \leqslant 2^{-n-1}$. Thus:

$$\begin{aligned}
|g^{**}(x, n) - f(x)| &\leqslant \alpha |g^*(y, m(n + 1)) - f(x)| + (1 - \alpha)|g^*(z, m(n + 1)) - f(x)| \\
&\leqslant \alpha(2^{-n-1} + |f(y) - f(x)|) + (1 - \alpha)(2^{-n-1} + |f(z) - f(x)|) \\
&\leqslant \alpha 2^{-n} + (1 - \alpha)2^{-n} \\
&\leqslant 2^{-n}
\end{aligned}$$

Using Remark 4.2.9 (Limit computability), we have thus shown that $f \in \text{AWP}$ and since AWP = AP by Theorem 4.5.1 (Main equivalence), $f \in \text{AP}$.

In the other direction, the proof is much easier. Assume that $f \in \text{AC}(\Upsilon, \Omega)$ where Υ, Ω are polynomials which we can assume to be increasing function. Apply Definition 4.2.1 (Analog computability) to get d, p and q. Apply Theorem 4.6.4 (Modulus of continuity) to f to get $\mho$ and define:

$$m(n) = \tfrac{1}{\ln 2}\mho(\max(|a|, |b]), n \ln 2)$$

It follows from the definition that m is a modulus of continuity of f since for any $n \in \mathbb{N}$ and $x, y \in [a, b]$ such that $|x - y| \leqslant 2^{-m(n)}$ we have:

$$|x - y| \leqslant 2^{-\frac{1}{\ln 2}\mho(\max(|a|,|b|), n \ln 2)} = e^{-\mho(\max(|a|,|b|), n \ln 2)} \leqslant e^{-\mho(|x|, n \ln 2)}$$

Thus $|f(x) - f(y)| \leqslant e^{-n \ln 2} = 2^{-n}$. We will now see how to approximate f in polynomial time. Let $r \in \mathbb{Q}$ and $n \in \mathbb{N}$, we would like to compute $f(r) \pm 2^{-n}$. By definition of f, there exists a unique $y : \mathbb{R}_+ \to \mathbb{R}^d$ such that for all $t \in \mathbb{R}_+$:

$$y(0) = q(r) \qquad y'(t) = p(y(t))$$

Furthermore, $|y_1(\Omega(|r|, \mu)) - f(r)| \leqslant e^{-\mu}$ for any $\mu \in \mathbb{R}_+$ and $\|y(t)\| \leqslant \Upsilon(|r|, t)$ for all $t \in \mathbb{R}_+$. Note that the coefficients of p and q belongs to $\mathbb{R}_G$ and since $\mathbb{R}_G \subseteq \mathbb{R}_P$ by Theorem 2.7.16 ($\mathbb{R}_G$ is generable subfield of $\mathbb{R}_P$), it follows that we can apply Theorem 3.5.2 (PIVP complexity) to compute y. In the details, one can compute a rational r' such that $|y(t) - r'| \leqslant 2^{-n}$ in time:

$$\operatorname{poly}(\deg(p), \operatorname{Len}(0, t), \log \|y(0)\|, \log \Sigma p, -\log 2^{-n})^d$$

Recall that in this case, all the parameters $d, \Sigma p, \deg(p)$ only depend on f and thus fixed and that $|r|$ is bounded by a constant. Thus these are all considered constants. So in particular, we can compute r' such that $|y(\Omega(|r|, (n + 1) \ln 2) - r'| \leqslant 2^{-n-1}$ in time:

$$\operatorname{poly}(\operatorname{Len}(0, \Omega(|r|, (n + 1) \ln 2)), \log \|q(r)\|, (n + 1) \ln 2)$$

[14]Do do so, consider the same polynomial where each coefficient is the ceiling value of the absolute value of the corresponding coefficient of m, and add the monomial $x \mapsto x$.

Note that $|r| \leqslant \max(|a|,|b|)$ and since a and b are constants and q is a polynomial, $\|q(r)\|$ is bounded by a constant. Furthermore,

$$\begin{aligned}\mathrm{Len}(0,\Omega(|r|,(n+1)\ln 2)) &= \int_0^{\Omega(|r|,(n+1)\ln 2)} \max(1,\|y(t)\|)^{\deg(p)}dt\\ &\leqslant \int_0^{\Omega(|r|,(n+1)\ln 2)} \mathrm{poly}(\Upsilon(\|r\|,t))dt\\ &\leqslant \Omega(|r|,(n+1)\ln 2)\,\mathrm{poly}(\Upsilon(|r|,\Omega(|r|,(n+1)\ln 2)))dt\\ &\leqslant \mathrm{poly}(|r|,n) \leqslant \mathrm{poly}(n)\end{aligned}$$

Thus r' can be computed in time:

$$\mathrm{poly}(n)$$

Which is indeed polynomial time since n is written in unary. Finally:

$$\begin{aligned}|f(r)-r'| &\leqslant |f(r)-y(\Omega(|r|,(n+1)\ln 2))| + |y(\Omega(|r|,(n+1)\ln 2))-r'|\\ &\leqslant e-(n+1)\ln 2 + 2^{-n-1}\\ &\leqslant 2^{-n}\end{aligned}$$

This show that f is polytime computable. ∎

Remark 5.3.11 (Domain of definition)**:** The equivalence holds over any interval $[a,b]$ but it can be extended in several ways. First it is possible to state an equivalence over $\mathbb{R}$. Indeed, classical real computability defines the complexity of $f(x)$ over $\mathbb{R}$ as polynomial in n and p where n is the precision and k the size of input, defined by $x \in [-2^k,2^k]$. Secondly, the equivalence also holds for multidimensional domains of the form $I_1 \times I_2 \times \cdots \times I_n$ where $I_k = [a_k,b_k]$ or $I_k = \mathbb{R}$. However, extending this equivalence to partial functions requires some caution. Indeed, our definition does not specify the behavior of functions outside of the domain, whereas classical discrete computability and some authors in computable analysis mandates that the machine never terminates on such inputs. More work is needed in this direction to understand how to state the equivalence in this case, in particular how to translate the "never terminates" part. Of course, the equivalence holds for partial functions where the behavior outside of the domain is not defined. ♦

Chapter 6

Bounded Time Reachability for Piecewise Affine Systems

Trying to solve differential equations is a youthful aberration that you will soon grow out of.

Quote from Cambridge

Reachability for piecewise affine systems is known to be undecidable, starting from dimension 2. In this chapter we investigate the exact complexity of several decidable variants of reachability and control questions for piecewise affine systems. We show in particular that the region to region bounded time versions leads to NP-complete or co-NP-complete problems, starting from dimension 2. This chapter corresponds to the work published in [BBGP14].

This chapter is organized as follows:

- Section 6.1 (Introduction) provides some background in this problem;
- Section 6.2 (Preliminaries) gives the formal definitions of the objects and decision problems involved;
- Section 6.3 (Bounded Time Reachability is NP-hard) shows that the two reachability problems are NP and co-NP hard respectively;
- Section 6.4 (Bounded Time Reachability is in NP) shows that the same two problems belong to NP and co-NP respectively;
- Section 6.5 (Other results) mentions similar results for the associated control problems.

6.1 Introduction

A crucial problem in such systems is the *reachability question*: given a system $\mathcal{H}$ and $R_0, R \subseteq X$, determine if there is a trajectory starting from a point of R_0 that falls in R. Reachability is known to be *undecidable* for very simple functions f. Indeed, it is well-known that various types of dynamical systems, such as hybrid systems, piecewise affine systems, or saturated linear systems, can simulate Turing machines, see e.g., [KCG94a, HKPV98, Moo91, SS95].

This question is at the heart of *verification* of systems. Indeed, a safety property corresponds to the determination if there is a trajectory starting from some set R_0 of possible initial states to the set R of bad states. The industrial and economical impact of having efficient computer tools, that are able to guarantee that a given system does satisfy its specification, is significant. This motivated many researchers to study verification problems, and several results about those problems are now known. Particularly, many undecidability and complexity-theoretic results about the hardness of verification of safety properties have been obtained in the model checking community. However, as far as we know, the exact complexity of *natural restrictions* of the reachability question for systems as simple as piecewise affine maps are not known, despite their practical interest.

Indeed, existing results mainly focus on the frontier between decidability and undecidability. For example, in the case of hybride system, it is known that reachability is undecidable for piecewise constant derivative systems of dimension 3, whereas it is decidable for dimension 2 [AMP95]. In the case of discrete systems, it is known that piecewise affine maps of dimension 2 can simulate Turing machines [KCG94b], whereas the question for dimension 1 is still open and can be related to other natural problems [AS02, ASY01, BC13]. Variations of such problems over the integers have recently been investigated [BA13].

Some complexity facts follow immediately from these (un)computability results: for example, point to point bounded time reachability for piecewise affine maps is P-complete as it corresponds to configuration to configuration reachability for Turing machines.

However, there still are many natural variants of reachability questions which complexity have not yet been established. For example, in the context of verification, point to point reachability is often not sufficient. On the contrary, region to region reachability is a more general question, whose complexity does not follow from existing results.

In this chapter we choose to restrict to the case of piecewise affine maps and we consider the following natural variant of the problem.

CONTINUOUS BOUNDED TIME: we want to know if region R is reached in less than some prescribed time T, with f assumed to be continuous

Remark 6.1.1: We consider piecewise affine maps over the domain $[0,1]^d$, that is to say we do not restrict to the integers as in [BA13]. That would make the problem rather different. We also assume f to be continuous which makes the hardness result more natural. All regions are assumed to be polyhedrons described with rational coefficients. ♦

In an orthogonal way, control of systems or constructions of controllers for systems often yield dual questions. Instead of asking if some trajectory reaches region R, one wants to know if all trajectories reach R. The questions of stability, mortality, or nilpotence for piecewise affine maps and saturated linear systems have been established in [BBKT01]. Still in this context, the complexity of the problem when restricting to bounded time or fixed precision is not known.

This chapter provides an exact characterization of the *algorithmic complexity* of those two types of reachability for discrete time dynamical systems. Let PAF_d denote the set of piecewise-affine *continuous* functions over $[0,1]^d$. At the end we get the following picture.

Problem: `REACH-REGION`
Inputs: two polyhedral regions R_0 and R, and a continuous PAF_d f
Question: $\exists x_0 \in R_0, t \in \mathbb{N}, f^{[t]}(x_0) \in R$?

Theorem 6.1.2 ([KCG94b]): *Problem REACH-REGION is undecidable (and recursively enumerable-complete).* ♦

Problem: `CONTROL-REGION`
Inputs: two polyhedral regions R_0 and R, and a continuous PAF_d f
Question: $\forall x_0 \in R_0, \exists t \in \mathbb{N}, f^{[t]}(x_0) \in R$?

Theorem 6.1.3 ([BBKT01]): *Problem CONTROL-REGION is undecidable (and co-recursively enumerable complete) for $d \geqslant 2$.* ♦

Problem: `REACH-REGION-TIME`
Inputs: two polyhedral regions R_0 and R, a time $T \in \mathbb{N}$ in unary and a continuous PAF_d f
Question: $\exists x_0 \in R_0, \exists t \leqslant T, f^{[t]}(x_0) \in R$?

Theorem 6.1.4: *Problem REACH-REGION-TIME is NP-complete for $d \geqslant 2$.* ♦

Problem: `CONTROL-REGION-TIME`
Inputs: two polyhedral regions R_0 and R, a time $T \in \mathbb{N}$ in unary and a continuous PAF_d f
Question: $\forall x_0 \in R_0, \exists t \leqslant T, f^{[t]}(x_0) \in R$?

Theorem 6.1.5: *Problem CONTROL-REGION-TIME is coNP-complete for $d \geqslant 2$.* ♦

All our problems are region to region reachability questions, which requires new proof techniques. Indeed, classical tricks to simulate a Turing machine using a piecewise affine maps encode a Turing machine configuration by a point, and assume that all the points of the trajectories encode (possibly ultimately) valid Turing machines configurations.

This is not a problem in the context of point to point reachability, but this can not be extended to region to region reachability. Indeed, a (non-trivial) region consists mostly in invalid points: almost all points do not correspond to encoding of Turing machines for all the encodings considered in references above.

In order to establish hardness results, the trajectories of all (valid and invalid) points must be carefully controlled. This turns out not to be easily possible using the classical encodings.

Let us insist on the fact that we restrict our results to continuous dynamics. In this context, this is an additional source of difficulties. Indeed, such a system must necessarily have a sub-region which dynamics cannot be easily interpreted in terms of configurations.

In other words, the difficulty is in dealing with points and trajectories not corresponding to valid configurations or evolutions.

6.2 Preliminaries

6.2.I Piecewise affine functions

Let $d \in \mathbb{N}$. A convex closed polyhedron in the space $[0,1]^d$ is the solution set of some linear system of inequalities:

$$A\vec{x} \leq \vec{b} \tag{6.2.1}$$

with coefficient matrix A and offset vector $\vec{b}$. A function $f\colon [0,1]^d \to [0,1]^d$ is piecewise-affine continuous if:

- f is continuous,
- there exists a sequence $(P_i)_{1\leq i\leq p}$ of convex closed polyhedron with nonempty interior such that $f_i = f\restriction_{P_i}$ is affine, $[0,1]^d = \bigcup_{i=1}^{p} P_i$ and $\mathring{P}_i \cap \mathring{P}_j = \emptyset$ for $i \neq j$.

Let PAF_d denote the set of piecewise-affine continuous functions over $[0,1]^d$.

In the following discussion we will always assume that any polyhedron P can be defined by a finite set of linear inequalities, where all the elements of A and $\vec{b}$ in (6.2.1) are all rationals. A polyhedron over which f is affine we also be called a region.

6.2.II Decision problems

In this chapter, we will show hardness results by reduction to known hard problems. We give the statement of these latter problems in the following.

Problem: SUBSET-SUM
Inputs: a goal $B \in \mathbb{N}$ and integers $A_1, \dots, A_n \in \mathbb{N}$.
Question: $\exists I \subseteq \{1, \dots, n\}, \sum_{i \in I} A_i = B$?

Theorem 6.2.2 ([GJ79])**:** *SUBSET-SUM is NP-complete.* ♦

Problem: NOSUBSET-SUM
Inputs: a witness $B \in \mathbb{N}$ and integers $A_1, \dots, A_n \in \mathbb{N}$.
Question: $\forall I \subseteq \{1, \dots, n\}, \sum_{i \in I} A_i \neq B$?

Theorem 6.2.3: *NOSUBSET-SUM is coNP-complete.* ♦

Proof. Basically the same proof as Theorem 6.2.2 ([GJ79]) ■

6.3 Bounded Time Reachability is NP-hard

In this section, we will show that REACH-REGION-TIME is an NP-hard problem by reducing SUBSET-SUM to it.

6.3.I Solving SUBSET-SUM by iteration

We will now show how to solve the SUBSET-SUM problem by iterating a function. Consider an instance $\mathcal{I} = (B, A_1, \dots, A_n)$ of SUBSET-SUM. We will need to introduce some notions before defining our piecewise affine function. Our first notion is that of configurations, which represent partial summation of the number of $\mathcal{I}$, for a given choice of $\mathcal{I}$.

Remark 6.3.1: Without loss of generality, we will only consider instances where $A_i \leqslant B$, for all i. Indeed, if $A_i > B$, it will never be part of a subset sum and so we can simply remove this variable from the problem. This ensures that $A_i < B + 1$ in everything that follows. ♦

Definition 6.3.2 (Configuration)**:** A configuration of $\mathcal{I}$ is a tuple $(i, \sigma, \varepsilon_i, \dots, \varepsilon_n)$ where $i \in \{1, \dots, n+1\}$, $\sigma \in \{0, \dots, B+1\}$, $\varepsilon_i \in \{0, 1\}$ for all i. Let $C_{\mathcal{I}}$ be the set of all configurations of $\mathcal{I}$. ♦

The intuitive understanding of a configuration, made formal in the next definition, is the following: $(i, \sigma, \varepsilon_i, \dots, \varepsilon_n)$ represents a situation where after having summed a subset of $\{A_1, \dots, A_{i-1}\}$, we got a sum σ and ε_j is 1 if and only if we are to pick A_j in the future.

Definition 6.3.3 (Transition function)**:** The transition function $T_{\mathcal{I}} : C_{\mathcal{I}} \to C_{\mathcal{I}}$, is defined as follows:

$$T_{\mathcal{I}}(i, \sigma, \varepsilon_i, \dots, \varepsilon_n) = \begin{cases} (i, \sigma) & \text{if } i = n+1 \\ (i+1, \min(B+1, \sigma + \varepsilon_i A_i), \varepsilon_{i+1}, \dots, \varepsilon_n) & \text{otherwise} \end{cases}$$

♦

It should be clear, by definition of a subset sum that we have the following simulation result.

Lemma 6.3.4: *For any configuration $c = (i, \sigma, \varepsilon_i, \dots, \varepsilon_n)$ and $k \in \{0, \dots, n+1-i\}$,*

$$T_{\mathcal{I}}^{[k]}(c) = \left(i+k, \min\left(B+1, \sigma + \Sigma_{j=i}^{i+k-1}\varepsilon_j A_j\right), \varepsilon_{i+k}, \dots, \varepsilon_n\right)$$

♦

Proof. By induction. ■

A consequence of this simulation by iterating $T_{\mathcal{I}}$, is that we can reformulate satisfiability in terms of reachability.

Lemma 6.3.5: *$\mathcal{I}$ is a satisfiable instance (*i.e.*, admits a subset sum) if and only if there exists a configuration $c = (1, 0, \varepsilon_1, \dots, \varepsilon_n) \in C_{\mathcal{I}}$ such that $T_{\mathcal{I}}^{[n]}(c) = (n+1, B)$.* ♦

Proof. The "only if" direction is the simplest: assume there exists $I \subseteq \{1, \dots, n\}$ such that $\sum_{i \in I} A_i = B$. Define $\varepsilon_i = 1$ if $i \in I$ and 0 otherwise. We get that $\sum_{i=1}^n \varepsilon_i A_i = B$. Apply Lemma 6.3.4 to get that:

$$\begin{aligned} T_{\mathcal{I}}^{[n]}(1, 0, \varepsilon_1, \dots, \varepsilon_n) &= (n+1, \min\left(B+1, 0 + \sum_{i=1}^n \varepsilon_i A_i\right)) \\ &= (n+1, \min(B+1, B)) = (n+1, B) \end{aligned}$$

The "if" direction is very similar: assume that there exists $c = (1, 0, \varepsilon_1, \dots, \varepsilon_n)$ such that $T_{\mathcal{I}}^{[n]}(c) = (n+1, B)$. Lemma 6.3.4 gives:

$$T_{\mathcal{I}}^{[n]}(1, 0, \varepsilon_1, \dots, \varepsilon_n) = (n+1, \min\left(B+1, 0 + \sum_{i=1}^n \varepsilon_i A_i\right))$$

We can easily conclude that $\sum_{i=1}^n \varepsilon_i A_i = B$ and thus by defining $I = \{i \mid \varepsilon_i = 1\}$ we get that $\sum_{i \in I} A_i = B$. Hence, $\mathcal{I}$ is satisfiable. ■

6.3.II Solving SUBSET-SUM with a piecewise affine function

In this section, we explain how to simulate the function $T_{\mathcal{I}}$ using a piecewise affine function and some encoding of the configurations for a given $\mathcal{I} = (B, A_1, \dots, A_n)$.

Definition 6.3.6 (Encoding)**:** Define $p = \lceil \log_2(n+2) \rceil$, $\omega = \lceil \log_2(B+2) \rceil$, $q = p + \omega + 1$ and $\beta = 5$. Also define $0^\star = 1$ and $1^\star = 4$. For any configuration $c = (i, \sigma, \varepsilon_i, \dots, \varepsilon_n)$, define the encoding of c as follows:

$$\langle c \rangle = \left(i2^{-p} + \sigma 2^{-q}, 0^\star \beta^{-n-1} + \sum_{j=i}^n \varepsilon_i^\star \beta^{-i} \right)$$

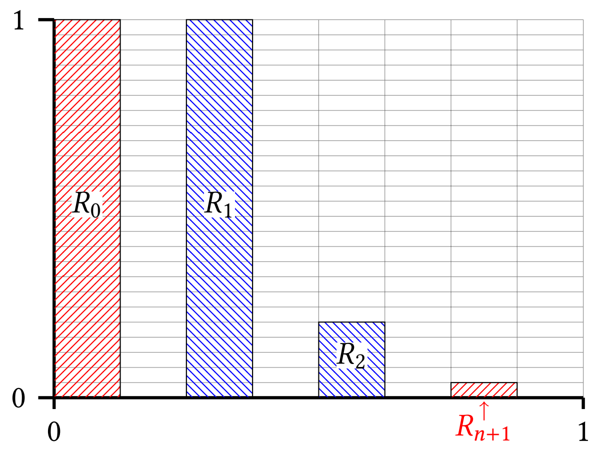
Also define the following regions for any $i \in \{1, \dots, n+1\}$ and $\alpha \in \{0, \dots, \beta - 1\}$:

$$R_0 = [0, 2^{-p-1}] \times [0, 1] \qquad R_i = [i2^{-p}, i2^{-p} + 2^{-p-1}] \times [0, \beta^{-i+1}] \quad (i \geqslant 1)$$

$$R_{i,\alpha} = \left[i2^{-p}, i2^{-p} + 2^{-p-1}\right] \times \left[\alpha\beta^{-i}, (\alpha+1)\beta^{-i}\right] \qquad R_i = \cup_{\alpha \in \mathbb{N}_\beta} R_{i,\alpha}$$

$$R_{i,1^\star}^{lin} = \left[i2^{-p}, i2^{-p} + (B + 1 - A_i)2^{-q}\right] \times \left[1^\star\beta^{-i}, 5\beta^{-i}\right] \qquad R_{i,1^\star}^{sat} = R_{i,1^\star} \setminus R_{i,1^\star}^{lin}$$

♦



$B = 2 \qquad n = 2 \qquad p = 2 \qquad \omega = 2 \qquad \beta = 5$

Figure 6.3.7: Graphical representation of the regions

The rationale behind this encoding is the following. On the first coordinate we put the current number i, "shifted" by as many bits as necessary to be between 0 and 1. Following i, we put σ, also shifted by as many bits as necessary. Notice that there is one padding bit between i and σ; this is necessary to make the regions R_i disjoint from each other. On the second component, we put the description of the variables ε_j, written in basis β to get some "space" between consecutive encodings. The choice of the value 1 and 4 for the encoding of 0 and 1, although not crucial, has been made to simplify the proof as much as possible.

The region R_0 is for initialization purposes and is defined differently for the other R_i. The regions R_i correspond to the different values of i in the configuration (the current number). Each R_i is further divided into the $R_{i,\alpha}$ which correspond to all the possible values of the next ε variable (recall that it is encoded in basis β). In the special case of $\varepsilon = 1$, we cut the region $R_{i,1^\star}$ into a linear part and a saturated part. This is needed to emulate the $\min(\sigma + A_i, B + 1)$ in Definition 6.3.3 (Transition function): the linear part corresponds to $\sigma + A_i$ and the saturated part to $B + 1$.

Figure 6.3.7 (Graphical representation of the regions) and Figure 6.3.13 (Zoom on one R_i with the subregions and formulas) give a graphical representation of the regions.

Lemma 6.3.8: *For any configuration $c = (i, \sigma, \varepsilon_i, \ldots, \varepsilon_n)$, if $i = n + 1$ then $\langle c\rangle \in R_{n+1,0^\star}$, otherwise $\langle c\rangle \in R_{i,\varepsilon_i^\star}$. Furthermore if $\varepsilon_i = 1$ and $\sigma + A_i \leqslant B + 1$, then $\langle c\rangle \in R^{lin}_{i,1^\star}$, otherwise $\langle c\rangle \in R^{sat}_{i,1^\star}$.* ♦

Proof. Recall that $\omega = \lceil \log_2(B + 2)\rceil$ so $B + 1 < 2^\omega$, and $q = p + \omega + 1$. Since $\sigma \leqslant B + 1$ by definition, $(n+1)2^{-p} \leqslant \langle c\rangle_1 \leqslant (n+1)2^{-p} + (B+1)2^{-p-1-\omega} \leqslant (n+1)2^{-p} + 2^{-p-1}$. This shows the result for the first component. In the case where $\sigma + A_i \leqslant B + 1$ then $\sigma 2^{-q} \leqslant (B + 1 - A_i)2^{-q}$ which gives the result for the second part of the result for the first component.

If $i = n + 1$, then $\langle c\rangle_2 = 0^\star\beta^{-p-1}$ which trivially belongs to $[0^\star\beta^{-n-1}, (0^\star + 1)\beta^{-n-1}]$. Otherwise, $\varepsilon_i^\star\beta^{-i} \leqslant \langle c\rangle_2 \leqslant \varepsilon_i^\star\beta^{-i} + \sum_{j=i+1}^{n+1} 1^\star\beta^{-j} \leqslant \varepsilon_i^\star\beta^{-i} + 1^\star\beta^{-i-1}\frac{1-\beta^{n-i}}{1-\beta^{-1}} \leqslant \varepsilon_i^\star\beta^{-i} + 4\beta^{-i-1}\frac{\beta}{\beta-1} \leqslant \varepsilon_i^\star\beta^{-i} + \beta^{-i} \leqslant (\varepsilon_i^\star + 1)\beta^{-i}$. This shows the result when $i < n + 1$, for the second component of the result. ■

We can now define a piecewise affine function that will mimic the behavior of $T_{\mathcal{I}}$. The region R_0 is here to ensure that we start from a "clean" value on the first coordinate.

Definition 6.3.9 (Piecewise affine simulation)**:**

$$f_{\mathcal{I}}(a,b)=\begin{cases}(2^{-p},b) & \text{if } (a,b)\in R_0\\ (a,b) & \text{if } (a,b)\in R_{n+1}\\ (a+2^{-p},b-0^\star\beta^{-i}) & \text{if } (a,b)\in R_{i,0^\star}\\ (a+2^{-p}+A_i2^{-q},b-1^\star\beta^{-i}) & \text{if } (a,b)\in R^{lin}_{i,1^\star}\\ ((i+1)2^{-p}+(B+1)2^{-q},b-1^\star\beta^{-i}) & \text{if } (a,b)\in R^{sat}_{i,1^\star}\end{cases}$$

♦

Lemma 6.3.10 (Simulation is correct)**:** *For any configuration $c\in C_{\mathcal{I}}$, $\langle T_{\mathcal{I}}(c)\rangle = f_{\mathcal{I}}(\langle c\rangle)$.* ♦

Proof. Let $c=(i,\sigma,\varepsilon_i,\dots,\varepsilon_n)$. There are two cases to consider. If $i=n+1$ then $T_{\mathcal{I}}(c)=c$, also by Lemma 6.3.8, $\langle c\rangle\in R_{n+1,0^\star}$. Thus by definition of f, $f_{\mathcal{I}}(\langle c\rangle)=\langle c\rangle=\langle T(c)\rangle$ which shows the result. If $i<n+1$, we have three more cases to consider: the case where we don't take the value ($\varepsilon_i=0$) and the two cases where we take it ($\varepsilon_i=0$) with and without saturation.

- If $\varepsilon_i=0$ then $T_{\mathcal{I}}(c)=(i+1,\sigma,\varepsilon_{i+1},\dots,\varepsilon_n)$. On the other hand, $\langle c\rangle=(a,b)=(i2^{-p}+\sigma2^{-q},0^\star\beta^{-i}+\sum_{j=i+1}^n\varepsilon_j\beta^{-j}+0^\star\beta^{-n-1})$. By Lemma 6.3.8, $\langle c\rangle\in R_{i,0^\star}$ so by definition of f:

$$\begin{aligned}f_{\mathcal{I}}(\langle c\rangle)&=(a+2^{-p},b-0^\star\beta^{-i})\\&=((i+1)2^{-p}+\sigma2^{-q},\sum_{j=i+1}^{n}\varepsilon_j\beta^{-j}+0^\star\beta^{-n-1})\\&=\langle(i+1,\sigma,\varepsilon_{i+1},\dots,\varepsilon_n)\rangle=\langle T_{\mathcal{I}}(c)\rangle\end{aligned}$$

- If $\varepsilon_i=1$ and $\sigma+A_i\leqslant B+1$ then $T_{\mathcal{I}}(c)=(i+1,\sigma+A_i,\varepsilon_{i+1},\dots,\varepsilon_n)$. On the other hand, $\langle c\rangle=(a,b)=(i2^{-p}+\sigma2^{-q},1^\star\beta^{-i}+\sum_{j=i+1}^n\varepsilon_j\beta^{-j}+0^\star\beta^{-n-1})$. By Lemma 6.3.8, $\langle c\rangle\in R^{lin}_{i,1^\star}$ so by definition of f:

$$\begin{aligned}f_{\mathcal{I}}(\langle c\rangle)&=(a+2^{-p}+A_i2^{-q},b-1^\star\beta^{-i})\\&=((i+1)2^{-p}+(\sigma+A_i)2^{-q},\sum_{j=i+1}^{n}\varepsilon_j\beta^{-j}+0^\star\beta^{-n-1})\\&=\langle(i+1,\sigma+A_i,\varepsilon_{i+1},\dots,\varepsilon_n)\rangle=\langle T_{\mathcal{I}}(c)\rangle\end{aligned}$$

- If $\varepsilon_i=1$ and $\sigma+A_i>B+1$ then $T_{\mathcal{I}}(c)=(i+1,B+1,\varepsilon_{i+1},\dots,\varepsilon_n)$. By Lemma 6.3.8, $\langle c\rangle\in R^{sat}_{i,1^\star}$ so by definition of f:

$$\begin{aligned}f_{\mathcal{I}}(\langle c\rangle)&=((i+1)2^{-p}+(B+1)2^{-q},b-1^\star\beta^{-i})\\&=\langle(i+1,B+1,\varepsilon_{i+1},\dots,\varepsilon_n)\rangle=\langle T_{\mathcal{I}}(c)\rangle\end{aligned}$$

■

Notice that we have defined f over a subset of the entire space and it is clear that this subspace is not stable in any way[1]. In order to match the definition of a piecewise affine function, we need to define f over the entire space or a stable subspace (which contains the initial region). We follow this second approach and extend the definition of f on some more regions. More precisely, we need to define f over $R_i=R_{i,0}\cup R_{i,1}\cup R_{i,2}\cup R_{i,3}\cup R_{i,3}$ and at the

[1]For example $R_{1,1}\subseteq f(R_0)$ but f is not defined over $R_{1,1}$.

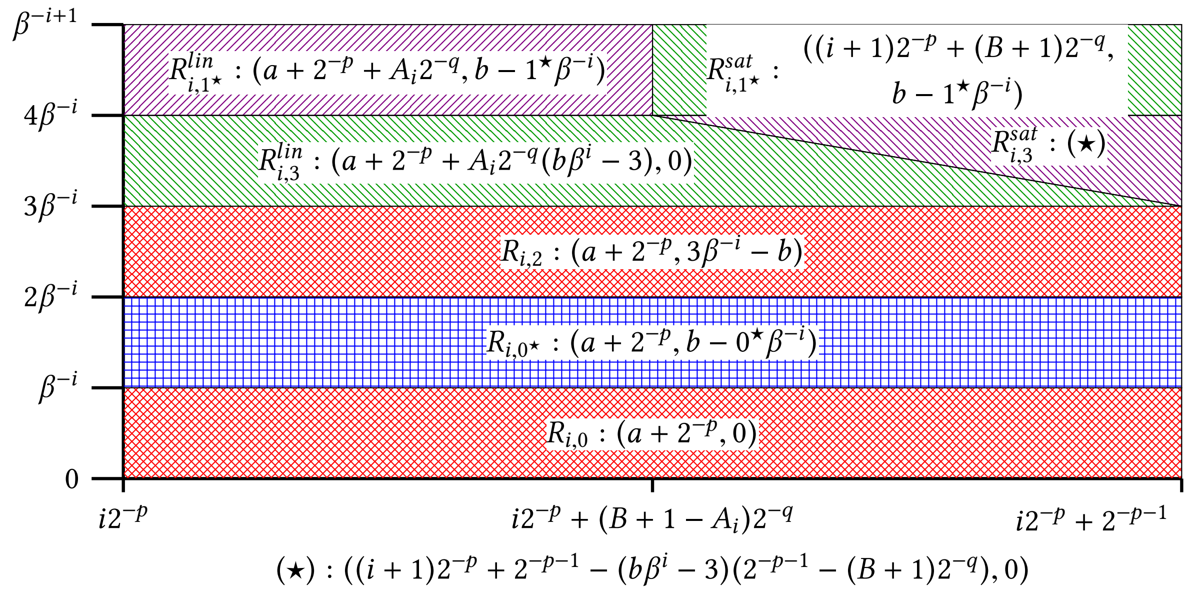

β^{-i+1}
$4\beta^{-i}$
$3\beta^{-i}$
$2\beta^{-i}$
β^{-i}
0
$R^{lin}_{i,1^\star} : (a + 2^{-p} + A_i 2^{-q}, b - 1^\star\beta^{-i})$
$R^{sat}_{i,1^\star} : ((i+1)2^{-p} + (B+1)2^{-q}, b - 1^\star\beta^{-i})$
$R^{lin}_{i,3} : (a + 2^{-p} + A_i 2^{-q}(b\beta^i - 3), 0)$
$R^{sat}_{i,3} : (\star)$
$R_{i,2} : (a + 2^{-p}, 3\beta^{-i} - b)$
$R_{i,0^\star} : (a + 2^{-p}, b - 0^\star\beta^{-i})$
$R_{i,0} : (a + 2^{-p}, 0)$
$i2^{-p}$
$i2^{-p} + (B + 1 - A_i)2^{-q}$
$i2^{-p} + 2^{-p-1}$
$(\star) : ((i+1)2^{-p} + 2^{-p-1} - (b\beta^i - 3)(2^{-p-1} - (B+1)2^{-q}), 0)$

Figure 6.3.13: Zoom on one R_i with the subregions and formulas

moment we have only defined f over $R_{i,1} = R_{i,0^\star}$ and $R_{i,4} = R_{i,1^\star}$. Also note that $R_{i,4} = R^{lin}_{i,4} \cup R^{sat}_{i,4}$ and we define f separately on those two subregions.

In order to correctly and continuously extend f, we will need to further split the region $R_{i,3}$ into linear and saturated parts $R^{slo}_{i,3}$ and $R^{shi}_{i,3}$: see Figure 6.3.13 (Zoom on one R_i with the subregions and formulas).

Definition 6.3.11 (Extended region splitting)**:** For $i \in \{1, \dots, n\}$ and $\alpha \in \{0, \dots, \beta - 1\}$, define:

$$R^{lin}_{i,3} = R_{i,3} \cap \left\{ (a,b) \,\middle|\, b\beta^i - 3 \leqslant \frac{2^{-p-1} + i2^{-p} - a}{2^{-p-1} - (B + 1 - A_i)2^{-q}} \right\} \qquad R^{sat}_{i,3} = R_{i,3} \setminus R^{lin}_{i,3}$$

♦

It should be clear by definition that $R^{sat}_{i,3} = R^{slo}_{i,3} \cup R^{shi}_{i,3}$ and that the two subregions are disjoint except on the border.

Definition 6.3.12 (Extended piecewise affine simulation)**:**

$$f_{\mathcal{I}}(a,b) = \begin{cases} (a + 2^{-p}, 0) & \text{if } (a,b) \in R_{i,0} \\ (a + 2^{-p}, 3\beta^{-i} - b) & \text{if } (a,b) \in R_{i,2} \\ (a + 2^{-p} + A_i 2^{-q}(b\beta^i - 3), 0) & \text{if } (a,b) \in R^{lin}_{i,3} \\ ((i + \frac{3}{2})2^{-p} - (b\beta^i - 3)(2^{-p-1} - (B+1)2^{-q}), 0) & \text{if } (a,b) \in R^{sat}_{i,3} \end{cases}$$

♦

This extension was carefully chosen for its properties. In particular, we will see that f is still continuous, which is a requirement of the piecewise affine functions we consider. Also, the domain of definition of f is f-stable (i.e. $f(\operatorname{dom} f) \subseteq \operatorname{dom} f$). And finally, we will see that f is somehow "reversible".

Lemma 6.3.14 (Simulation is continuous)**:** *For any $i \in \{1, \dots, n\}$, $f_{\mathcal{I}}(R_i)$ is well-defined and continuous over R_i.* ♦

Proof. As outlined on Figure 6.3.13 (Zoom on one R_i with the subregions and formulas), we need to check that the definitions of f match at the borders of each subregions of R_i. More precisely, we need to check that Definition 6.3.9 (Piecewise affine simulation) and Definition 6.3.12 (Extended piecewise affine simulation) agree on all borders.

- $(a, b) \in R_{i,0} \cap R_{i,0^\star}$: the first component is computed using the same formula so is clearly continuous, the second component is always 0 on both side of the border because $b - 0^\star\beta^{-i} = 0$ for $b = \beta^{-i}$
- $(a, b) \in R_{i,0^\star} \cap R_{i,2}$: the first component is computed using the same formula so is clearly continuous, the second component is always β^{-i} on both side of the border because $b - 0^\star\beta^{-i} = 3\beta^{-i} - b = \beta^{-i}$ for $b = 2\beta^{-i}$
- $(a, b) \in R_{i,2} \cap R^{lin}_{i,3}$: the first component is $a + 2^{-p}$ and the second component is 0 on both side of the border because $3\beta^{-i} - b = b\beta^i - 3 = 0$ for $b = 3\beta^{-i}$
- $(a, b) \in R^{lin}_{i,3} \cap R^{lin}_{i,1^\star}$: the first component is $a + 2^{-p} + A_i 2^{-q}$ and the second component is 0 on both side of the border because $b\beta^i - 3 = 1$ and $b - 1^\star\beta^{-i} = 0$ for $b = 4\beta^{-i}$
- $(a, b) \in R^{lin}_{i,3} \cap R^{sat}_{i,3}$: the second component is always 0 on both regions so is clearly continuous. From Definition 6.3.11 (Extended region splitting) one can see that $b\beta^i - 3 = \frac{Y}{X}$ holds on the border where $Y = 2^{-p-1} + i2^{-p} - a$ and $X = 2^{-p-1} - (B + 1 - A_i)2^{-q}$. Consequently, if we compute the difference between the two expression at the borders, we get:

  $$\begin{aligned}&\Big((i + 1)2^{-p} + 2^{-p-1} - (b\beta^i - 3)(2^{-p-1} - (B + 1)2^{-q})\Big) - \Big(a + 2^{-p} + A_i 2^{-q}(b\beta^i - 3)\Big)\\ &= i2^{-p} + 2^{-p-1} - a - \frac{Y}{X}(2^{-p-1} - (B + 1)2^{-q} + A_i 2^{-q})\\ &= Y + \frac{Y}{X}X = 0\end{aligned}$$

  which proves that they are equal.
- $(a, b) \in R^{sat}_{i,3} \cap R^{sat}_{i,1^\star}$: the first component is $(i+1)2^{-p}+(B+1)2^{-q}$ and the second component is 0 on both side of the border because $b\beta^i - 3 = 1$ and $b - 1^\star\beta^{-i} = 0$ for $\beta = 4\beta^{-i}$
- $(a, b) \in R^{lin}_{i,1^\star} \cap R^{sat}_{i,1^\star}$: the first component is $(i+1)2^{-p}+(B+1)2^{-q}$ on both side of the border because $a = i2^{-p} + (B+1-A_i)2^{-q}$, and the second component is computed using the same formula so is clearly continuous

■

Lemma 6.3.15 (Simulation is stable)**:** *For any $i \in \{1, \dots, n\}$, $f_{\mathcal{I}}(R_i) \subseteq R_{i+1}$. Furthermore, $f(R_0) \subseteq R_1$ and $f(R_{n+1}) \subseteq R_{n+1}$.* ♦

Proof. We need to examinate all possible cases for $(a, b) \in R_i$. Since $R_i = \bigcup_{\alpha=0}^{4} R_{i,\alpha}$ and that $R_{i,\alpha} = R^{lin}_{i,\alpha} \cup R^{sat}_{i,\alpha}$ we indeed cover all cases.

- If $(a, b) \in R_0$: then $f_{\mathcal{I}}(a, b) = (a + 2^{-p}, b)$ so $f_{\mathcal{I}}(R_0) = f_{\mathcal{I}}([0, 2^{-p-1}] \times [0, 1]) = [2^{-p}, 2^{-p} + 2^{-p-1}] \times [0, 1] = R_1$.
- If $(a, b) \in R_{n+1}$: then $f_{\mathcal{I}}(a, b) = (a, b)$ so $f_{\mathcal{I}}(R_{n+1}) = R_{n+1}$.
- If $(a, b) \in R_{i,0}$: then $f_{\mathcal{I}}(a, b) = (a + 2^{-p}, 0)$ so $f_{\mathcal{I}}(R_{i,0}) = f_{\mathcal{I}}([i2^{-p}, i2^{-p} + 2^{-p-1}] \times [0, \beta^{-i}]) = [(i + 1)2^{-p}, (i + 1)2^{-p} + 2^{-p-1}] \times \{0\} \subseteq R_{i+1}$.

- If $(a,b)\in R_{i,1}=R_{i,0^\star}$: then $f_{\mathcal{I}}(a,b)=(a+2^{-p},b-0^\star\beta^{-i})$ so $f_{\mathcal{I}}(R_{i,1})=f_{\mathcal{I}}([i2^{-p},i2^{-p}+2^{-p-1}]\times[\beta^{-i},2\beta^{-i}])=[(i+1)2^{-p},(i+1)2^{-p}+2^{-p-1}]\times[0,\beta^{-i}]=R_{i+1}$.
- If $(a,b)\in R_{i,2}$: then $f_{\mathcal{I}}(a,b)=(a+2^{-p},3\beta^{-i}-b)$ so $f_{\mathcal{I}}(R_{i,2})=f_{\mathcal{I}}([i2^{-p},i2^{-p}+2^{-p-1}]\times[2\beta^{-i},3\beta^{-i}])=[(i+1)2^{-p},(i+1)2^{-p}+2^{-p-1}]\times[0,\beta^{-i}]=R_{i+1}$.
- If $(a,b)\in R_{i,3}^{lin}$: the image of the second component is always 0 so it's easy for this one, also from Definition 6.3.11 (Extended region splitting), $b\beta^i-3\leqslant\frac{2^{-p-1}+i2^{-p}-a}{2^{-p-1}-(B+1-A_i)2^{-q}}\leqslant\frac{2^{-p-1}+i2^{-p}-a}{A_i2^{-q}}$ because $2^{-p-1}-(B+1)2^{-q}\geqslant 0$ since $(B+1)2^{-q}\leqslant 2^{\omega}2^{-q}\leqslant 2^{-p-1}$. Consequently, for the first coordinate we get that $f_{\mathcal{I}}(a,b)_1\leqslant a+2^{-p}+A_i2^{-q}\frac{2^{-p-1}+i2^{-p}-a}{A_i2^{-q}}\leqslant(i+1)2^{-p}+2^{-p-1}$. Also, since $i2^{-p}\leqslant a\leqslant i2^{-p}+2^{-p-1}$, it is clear that $f_{\mathcal{I}}(a,b)_1\geqslant(i+1)2^{-p}$. So finally, $f_{\mathcal{I}}(R_{i,3}^{lin})\subseteq[(i+1)2^{-p},(i+1)2^{-p}+2^{-p-1}]\times\{0\}\subset R_{i+1}$.
- If $(a,b)\in R_{i,3}^{sat}$: the image of the second component is always 0 so it's easy for this one, also from Definition 6.3.11 (Extended region splitting), $b\beta^i-3\geqslant\frac{2^{-p-1}+i2^{-p}-a}{2^{-p-1}-(B+1-A_i)2^{-q}}\geqslant\frac{2^{-p-1}+i2^{-p}-a}{2^{-p-1}-(B+1)2^{-q}}$ because $A_i\geqslant 0$. Consequently, for the first coordinate we get that $f_{\mathcal{I}}(a,b)_1\leqslant(i+1)2^{-p}+2^{-p-1}-(2^{-p-1}-(B+1)2^{-q})\frac{2^{-p-1}+i2^{-p}-a}{2^{-p-1}-(B+1)2^{-q}}\leqslant(i+1)2^{-p}+2^{-p-1}+i2^{-p}+2^{-p-1}-a\leqslant(i+1)2^{-p}+2^{-p-1}$ since $a\leqslant i2^{-p}+2^{-p-1}$. Also since $b\beta^i-3\leqslant 1$ we get that $f_{\mathcal{I}}(a,b)_1\geqslant(i+1)2^{-p}+2^{-p-1}-(2^{-p-1}-(B+1)2^{-q})\times 1\geqslant(i+1)2^{-p}+(B+1)2^{-q}$. So finally, $f_{\mathcal{I}}(R_{i,3}^{sat})\subseteq[(i+1)2^{-p}+(B+1)2^{-q},(i+1)2^{-p}+2^{-p-1}]\times\{0\}\subset R_{i+1}$.
- If $(a,b)\in R_{i,4}^{lin}=R_{i,1^\star}^{lin}$: then $f_{\mathcal{I}}(a,b)=(a+2^{-p}+A_i2^{-q},b-1^\star\beta^{-i})$ so $f_{\mathcal{I}}(R_{i,4}^{lin})=f_{\mathcal{I}}([i2^{-p},i2^{-p}+(B+1-A_i)2^{-q}]\times[4\beta^{-i},5\beta^{-i}])=[(i+1)2^{-p}+A_i2^{-q},(i+1)2^{-p}+(B+1)2^{-q}]\times[0,\beta^{-i}]\subseteq R_{i+1}$ because $(B+1)2^{-q}\leqslant 2^{-p-1}$.
- If $(a,b)\in R_{i,4}^{sat}$: then $f_{\mathcal{I}}(a,b)=((i+1)2^{-p}+(B+1)2^{-q},0)$ so $f_{\mathcal{I}}(R_{i,4}^{sat})=\{(i+1)2^{-p}+(B+1)2^{-q}\}\times\{0\}\subseteq R_{i+1}$.

■

We now get to the core lemma of the simulation. Up to this point, we were only interested in forward simulation: that is given a point, what are the iterates of x. In order to prove the NP-hardness result, we need a backward result: given a point, what are the possible preimages of it. To this end, we introduce new subregions R_i^{unsat} of the R_i, which we call *unsaturated.* Intuitively, R_i^{unsat} corresponds to the encodings where $\sigma\leqslant B$, that is the sum did not saturate at $B+1$. We also introduce the R_{fin} region, which will be the region to reach. We will be interested in the preimages of R_{fin}.

Definition 6.3.16 (Unsaturated regions)**:** For $i\in\{1,\dots,n+1\}$, define

$$R_i^{unsat}=[i2^{-p},i2^{-p}+B2^{-q}]\times[\beta^{-n-1},\beta^{-i+1}-\beta^{-n-1}]$$

$$R_{fin}=[(n+1)2^{-p}+B2^{-q}-2^{-q-1},(n+1)2^{-p}+B2^{-q}]\times[\beta^{-n-1},2\beta^{-n-1}]$$

♦

Lemma 6.3.17 (Simulation is reversible)**:** *Let $i\in\{2,\dots,n\}$ and $(a,b)\in R_i^{unsat}$ Then the only points $\vec{x}$ such that $f_{\mathcal{I}}(\vec{x})=(a',b')$ are:*

- $\vec{x}=(a-2^{-p},b'+0^\star\beta^{-i+1})\in R_{i-1,0^\star}\cap R_{i-1}^{unsat}$
- $\vec{x}=(a-2^{-p},\beta^i-b'+0^\star\beta^{-i+1})\in R_{i-1,2}\cap R_{i-1}^{unsat}$

- $\vec{x} = (a - 2^{-p} - A_i 2^{-q}, b' + 1^\star \beta^{-i+1}) \in R^{lin}_{i-1,1^\star} \cap R^{unsat}_{i-1}$ *(only if $a \geqslant 2^{-p} + A_i 2^{-q}$)*

♦

Proof. First notice that since $f_\mathcal{I}(R_i) \subseteq R_{i+1}$ for all $i \in \{0, \dots, n\}$, the only candidates for $\vec{x}$ must belong to R_{i-1}. Furthermore, on each affine region, there can only be one candidate except if the function is trivial.

A close look at the proof of Lemma 6.3.15 (Simulation is stable) reveals that:

- $f_\mathcal{I}(R_{i-1,0}) \subseteq [(i+1)2^{-p}, (i+1)2^{-p} + 2^{-p-1}] \times \{0\}$, which shares no point with R^{unsat}_i, so there is no possible candidate
- $f_\mathcal{I}(R_{i-1,1}) = R_i$ and there is only one possible candidate
- $f_\mathcal{I}(R_{i-1,2}) = R_i$ and there is only one possible candidate
- $f_\mathcal{I}(R_{i-1,3}) \subseteq [(i+1)2^{-p}, (i+1)2^{-p} + 2^{-p-1}] \times \{0\}$ so like $R_{i-1,0}$ there is no possible candidate
- $f_\mathcal{I}(R^{lin}_{i-1,4}) \subseteq R_i$ and there is only one possible candidate
- $f_\mathcal{I}(R^{sat}_{i-1,4}) \subseteq [(i+1)2^{-p} + (B+1)2^{-q}, (i+1)2^{-p} + 2^{-p-1}] \times [0, \beta^{-i}]$, which shares no point with R^{unsat}_i, so there is no possible candidate

It is then only a matter of checking that the claimed formulas work and they trivially do except for the case of $R^{lin}_{i-1,4}$ where we need the potential candidate to belong to the region. ■

The goal of those results in to show that if there is a point in R_{fin} that is reachable from R_0 then we can extract, from its trajectory, a configuration that also reaches R_{fin}. Furthermore, we arranged so that R_{fin} contains the encoding of only one configuration:$(n+1, B)$ (see Lemma 6.3.5).

Lemma 6.3.18 (Backward-forward identity): *For any point $\vec{x} \in R_{fin}$, if there exists a point $\vec{y} \in R_0$ and an integer k such that $f^{[k]}_\mathcal{I}(\vec{y}) = \vec{x}$ then there exists a configuration $c = (1, 0, \varepsilon_1, \dots, \varepsilon_n)$ such that $f^{[k]}_\mathcal{I}(\langle c \rangle) \in R_{fin}$.* ♦

Proof. Define $\vec{y}_0 = \vec{y}$ and $\vec{y}_{i+1} = f_\mathcal{I}(\vec{y}_i)$ for all $i \in \{0, k-1\}$. Since $\vec{y}_0 \in R_0$, we immediately get that $\vec{y}_i \in R_i$ using Lemma 6.3.15 (Simulation is stable) and in particular, $k \geqslant n+1$ because $\vec{y}_k = x \in R_{n+1,0^\star}$.

Now apply Lemma 6.3.17 (Simulation is reversible) starting from $\vec{y}_{n+1} \in R^{unsat}_{n+1}$: we conclude that for all $i \geqslant 1$, $\vec{y}_i \in (R_{i,1} \cup R_{i,2} \cup R^{lin}_{i,4}) \cap R^{unsat}_n$. Define $\varepsilon_i = 0$ if $\vec{y}_i \in R_{i,1} \cup R_{i,2}$ and 1 if $\vec{y}_i \in R^{lin}_{i,4}$. Write $\vec{y}_i = (a_i, b_i)$. Again using Lemma 6.3.15 (Simulation is stable) we get that $a_{i-1} = a_i - 2^{-p} - \varepsilon_i A_i 2^{-q}$ (just check all three cases). Also since $\vec{x} = \vec{y_{n+1}} \in R_{fin}$ then $a_{n+1} \in [(n+1)2^{-p} + B2^{-q} - 2^{-q-1}, (n+1)2^{-p} + B2^{-q}]$. Finally, $\vec{y}_0 \in R_0$ so $f_\mathcal{I}(a_0, b_0) = (2^{-p}, 0) = (a_1, b_1)$. We conclude that $a_1 = 2^{-p}$. Putting everything together we get:

$$\begin{cases} a_{n+1} = (n+1)2^{-p} + 2^{-q} \sum_{i=1}^n \varepsilon_i A_i \\ a_{n+1} \in [(n+1)2^{-p} + B2^{-q} - 2^{-q-1}, (n+1)2^{-p} + B2^{-q}] \end{cases}$$

Since the A_i, B are integers and $\varepsilon_i \in \{0, 1\}$, we get that $B = \sum_{i=1}^n A_i \varepsilon_i$. Apply Lemma 6.3.10 (Simulation is correct) on the configuration to conclude. ■

Lemma 6.3.19 (Final region is accepting): *For any configuration c, if $\langle c \rangle \in R_{fin}$ then $c = (n+1, B)$.* ♦

Proof. Write $c = (i, \sigma, \varepsilon_i, \dots, \varepsilon_n)$, then$\langle c \rangle = \left(i2^{-p} + \sigma 2^{-q}, \sum_{j=i}^n \varepsilon_i^\star \beta^{-i} + 0^\star \beta^{-n-1}\right)$. It implies that $i2^{-p} + \sigma 2^{-q} \in [(n+1)2^{-p} + B2^{-q} - 2^{-q-1}, (n+1)2^{-p} + B2^{-q}]$ and because i is an integer in range $[0, n+1]$ and σ an integer in range $[0, B+1]$, necessarily $i = n+1$ and $\sigma = B$. ■

6.3.III REACH-REGION-TIME is NP-hard

We now have all the tools to show that REACH-REGION-TIME is an NP-hard problem.

Theorem 6.3.20: *REACH-REGION-TIME is NP-hard for $d \geqslant 2$.* ♦

Proof. Let $\mathcal{I} = (B, A_1, \ldots, A_n)$ be a instance of SUBSET-SUM. We consider the instance $\mathcal{J}$ of REACH-REGION-TIME defined in the previous section with maximum number of iterations set to n (the number of A_i), the initial region set to R_0 and the final region set to R_{fin}. One easily checks that this instance has polynomial size in the size of $\mathcal{I}$. The two directions of the proofs are:

- If $\mathcal{I}$ is satisfiable then use Lemma 6.3.4 and Lemma 6.3.10 (Simulation is correct) to conclude that there is a point $x \in R_0$ in the initial region such that $f_{\mathcal{I}}^{[n]}(x) \in R_{fin}$ so $\mathcal{J}$ is satisfiable.

- If $\mathcal{J}$ is satisfiable then there exists $x \in R_0$ and $k \leqslant n$ such that $f_{\mathcal{I}}^{[k]}(x) \in R_{fin}$. Use Lemma 6.3.18 (Backward-forward identity) and Lemma 6.3.10 (Simulation is correct) to conclude that there exists a configuration $c = (1, 0, \varepsilon_1, \ldots, \varepsilon_n)$ such that $\langle T_{\mathcal{I}}^{[k]}(c)\rangle = f_{\mathcal{I}}^{[k]}(\langle c\rangle) \in R_{fin}$. Apply Lemma 6.3.19 (Final region is accepting) and use the injectivity of the encoding to conclude that $T_{\mathcal{I}}^{[k]}(c) = (n+1, B)$ and Lemma 6.3.5 to get that $\mathcal{I}$ is satisfiable.

■

6.4 Bounded Time Reachability is in NP

In the previous section we have shown that the REACH-REGION-TIME problem is NP-hard. We now give a more precise characterization of the complexity of this problem, by proving that it is NP-complete. Since we have shown its NP-hardness, the only thing that remains to be shown is that REACH-REGION-TIME belongs to NP. This is done in this section.

6.4.I Notations and definitions

For any $i = 1, \ldots, d$, let $\pi_i^d \colon [0,1]^d \to [0,1]$ denote the i^{th} projection function, that is, $\pi(x_1, \ldots, x_d) = x_i$. Let $g_d \colon [0,1]^{d+1} \to [0,1]^d$ be defined by $g_d(x_1, \ldots, x_{d+1}) = (x_1, \ldots, x_d)$. For a square matrix A of size $(d+1)\times(d+1)$ define the following pair of projection functions. The first function $h_{1,d}$ takes as input a square matrix A of size $(d+1)\times(d+1)$ and returns a square matrix of size $d \times d$ that is the upper-left block of A. The second function $h_{2,d}$ takes as input a square matrix A of size $(d+1)\times(d+1)$ and returns the vector of size d given by $[a_{1,d+1} \cdots a_{d,d+1}]^T$ (the last column of A minus the last element).

Let s denote the size function, its domain of objects will be overloaded and understood from the context. For $x \in \mathbb{Z}$, $s(x)$ is the length of the encoding of x in base 2. For $x \in \mathbb{Q}$ with $x = \frac{p}{q}$ we have $s(x) = \max(s(p), s(q))$. For an affine function f we define the size of $f(\vec{x}) = A\vec{x} + \vec{b}$ (where all entries of A and $\vec{b}$ are rationals) as: $s(f) = \max(\max_{i,j}(s(a_{i,j})), \max(s(b_i)))$. We define the size of a polyhedron r defined by $A\vec{x} \leqslant \vec{b}$ as: $s(r) = \max(s(A), s(\vec{b}))$.

We define the size of a piecewise affine function f as: $s(f) = \max_i(s(f_i), s(r_i))$ where f_i denotes the restriction of f to r_i the i^{th} region.

We define the *signature* of a point $\vec{x}$ as the sequence of indices of the regions traversed by the iterates of f on $\vec{x}$ (that is, the region trajectory).

6.4.II REACH-REGION-TIME is in NP

In order to solve a reachability problem, we will formulate it with linear algebra. However a crucial issue here is that of the size of the numbers, especially when computing powers of matrices. Indeed, if taking the n^{th} power of A yields a representation of exponential size, no matter how fast our algorithm is, it will run on exponentially large instances and thus be slow.

First off, we show how to move to homogeneous coordinates so that f becomes piecewise linear instead of piecewise affine.

Lemma 6.4.1: *Assume that $f(\vec{x}) = A\vec{x} + \vec{b}$ with $A = (a_{i,j})_{1 \leqslant i,j \leqslant d}$ and let $y = A'(\vec{x}, 1)^T$ where A' is the block matrix $\begin{pmatrix} A & \vec{b} \\ 0 & 1 \end{pmatrix}$. Then $f(x) = g_d(A'(\vec{x}, 1)^T)$.* ♦

Remark 6.4.2: Notice that this lemma extends nicely to the composition of affine functions: if $f(\vec{x}) = A\vec{x} + \vec{b}$ and $h(\vec{x}) = C\vec{x} + \vec{d}$ then $h(f(x)) = g_d(C'A'(\vec{x}, 1)^T)$. ♦

We can now state the main lemma, namely that the size of the iterates of f vary linearly in the number of iterates, assuming that f is piecewise affine.

Lemma 6.4.3: *Let $d \geqslant 2$ and $f \in PAF_d$. Assume that all the coefficients of f on all regions are rationals. Then for all $t \in \mathbb{N}$, $s(f^{[t]}) \leqslant (d+1)^2 s(f)pt + (t-1)\lceil \log_2(d+1) \rceil$ where p is the number of regions of f. This inequality holds even if all rationals are taken to have the same denominator.* ♦

Proof. Using Lemma 6.4.1, we get that $f^{[t]}(\vec{x}) = g_d(h^{[t]}([\vec{x}\ 1]^T))$, where h is a piecewise linear function in dimension $d+1$ such that $s(h) = s(f)$. We show this result by induction on t for h. The result then follows for f. In all cases we take all rationals to have the same denominator.

In the case $t = 1$, it suffices to see that taking all rationals to have the same denominator involves multiplying the numerator and denominators by at most the lowest common multiple of all numbers, which is at most $2^{s(f)(p(d+1)^2)}$. Indeed the greatest number is $2^{s(h)}$ by definition, and there are $(d+1)^2$ numbers per region (the entries of the matrix).

Assume the result is true for $t \in \mathbb{N}$. Let $\vec{y} \in \mathbb{Q}^{d+1}$. Then $h^{[t+1]}(\vec{y}) = B_{t+1} \cdots B_1 \vec{y}$, where B_i's are the matrices corresponding to some regions of h. In particular, $s(B_i) \leqslant s(h)$. From the induction hypothesis we can assume that all rationals have the same denominator and we get that $s(B_t \cdots B_1) \leqslant (d+1)^2 s(h)pt + (t-1)\lceil \log_2(d+1) \rceil$. It follows[2] that for any $1 \leqslant i,j \leqslant d+1$:

$$\begin{aligned}
s((B_{t+1} \cdots B_1)_{i,j}) &= s\left(\sum_{k=1}^{d+1} (B_{t+1})_{i,k} \, (B_t \cdots B_1)_{k,j} \right) \\
&\leqslant \lceil \log_2(d+1) \rceil + s(B_{t+1}) + s(B_t \cdots B_1) \\
&\leqslant \lceil \log_2(d+1) \rceil + s(h) + (d+1)^2 s(h)pt + (t-1)\lceil \log_2(d+1) \rceil \\
&\leqslant (d+1)^2 s(h)p(t+1) + t\lceil \log_2(d+1) \rceil
\end{aligned}$$

This shows the result for the particular region where y belongs. Since the bound does not depend on y and $h^{[t+1]}$ has finitely many regions, it is true for all regions of $h^{[t+1]}$. ∎

Finally, we need some result about the size of solutions to systems of linear inequalities. Indeed, if we are going to quantify over the existence of a solution of polynomial size, we must ensure that the size constraints do not change the satisfiability of the system.

[2]Use elementary properties of the size function: $s(xy) \leqslant s(x) + s(y)$, $s(x_1 + \cdots + x_k) \leqslant s(k) + \max_k s(x_k)$

Lemma 6.4.4 ([Koi94])**:** *Let A be a $N \times d$ integer matrix and $\vec{b}$ an integer vector. If the $A\vec{x} \leqslant \vec{b}$ system admits a solution, then there exists a rational solution x_s such that $s(x_s) \leqslant (d+1)L + (2d+1)\log_2(2d+1)$ where $L = \max(s(A), s(b))$.* ♦

Proof. See Theorem 5 of [Koi94]: $s(x_s) \leqslant s\left((2d+1)!2^{L(2d+1)}\right)$. ■

Putting everything together, we obtain a fast nondeterministic algorithm to solve REACH-REGION-TIME. The nondeterministism allows use to choose a signature for the solution. Once the signature is fixed, we can write it as a linear program of reasonable size using Lemma 6.4.3 and solve it. The remaining issue is the one of the size of solution but fortunately Lemma 6.4.4 ([Koi94]) ensures us that there is a small solution that can be found quickly.

Theorem 6.4.5: *REACH-REGION-TIME is in NP.* ♦

Proof. The idea of the proof is to nondeterministically choose a signature for a solution, that is a sequence of regions for the iterates of the solution. We then build a system of linear inequalities stating that a point $\vec{x}$ belongs to the initial region and that the iterates match the signature chosen and finally that the iterates reaches the final region. Using the results of the previous section, we can build this system in polynomial time and solve it in non-deterministic polynomial time. Here is an outline of the algorithm:

- Non-deterministically choose $t \leqslant T$
- Non-deterministically choose regions s $r_1, \dots, r_{t-1}$ of f
- Define $r_0 = R_0$ the initial region and $r_t = R$ the final region
- Build (S) the system $A\vec{x} \leqslant \vec{b}$ stating that the signature of x matches r
- Non-deterministically choose $\vec{x_s}$ a rational of polynomial size in the size of (S)
- Accept if $A\vec{x_s} \leqslant \vec{b}$

We have two things to prove. First we need to show that this algorithm indeed has non-deterministic polynomial running time. Second we need to show that it is correct. Recall that T is a unary input of the problem.

The complexity of the algorithm is clear, assuming that (S) is of polynomial size. Indeed verifying that a rational point satisfies a system of linear inequalities with rationals coefficients can be done in polynomial time.

We build (S) this way: $(S) = \cup_{i=1}^{t}(S_i)$ where (S_i) states that $f^{[i]}(\vec{x}) \in r_i$. Since we choose a signature of x we know that if x satisfies the system then from Lemma 6.4.1 $f^{[i]}(\vec{x}) = g_d\left(A'_{i-1}\cdots A'_1(\vec{x},1)^T\right)$ where A'_j is the matrix corresponding to the region r_j. Write $C_i = A'_{i-1}\cdots A'_1$ and define (S_i) by the system $g_d\left(C_i(\vec{x},1)^T\right) \in r_i$. Since r_i is a polyhedron, (S_i) is indeed a system of linear inequalities[3].

We can now see that S is of polynomial size using Lemma 6.4.3. Indeed, $s(C_i) \leqslant s(f^{[i]}) \leqslant \text{poly}(s(f), i)$, thus $s((S_i)) \leqslant s(C_i) + s(r_i) \leqslant \text{poly}(s(f), i)$ because the description of the regions is part of the size of f. And finally $s((S)) \leqslant \text{poly}(s(f), t)$.

The correctness follows from the construction of the system and Lemma 6.4.4 ([Koi94]). More precisely we show that $\vec{x} \in (S)$ if and only if $\forall i \in \{0,\dots,t\}, f^{[i]}(\vec{x}) \in r_i$. Indeed, $(S) \Leftrightarrow \forall i \in \{0,\dots,t\}, \vec{x} \in (S_i)$ and by definition $(S_i) \Leftrightarrow f^{[i]}(\vec{x}) \in r_i$ since $g_d(C_i(\vec{x},1)^T) = f^{[i]}(\vec{x})$. Then by Lemma 6.4.4 ([Koi94]), we get that $\exists\vec{x} \in (S) \Leftrightarrow \exists\vec{x} \in (S)$ and $s(\vec{x}) \leqslant \text{poly}(s((S)))$. ■

[3]More precisely if r_i is defined by $P_i(\vec{x},1)^T \leqslant 0$ then (S_i) is the system $P_iC_i(\vec{x},1)^T \leqslant 0$

6.5 Other results

In this section, we give succint proofs of the other result mentioned in the introduction about CONTROL-REGION-TIME. The proof is based on the same arguments as before.

Theorem 6.5.1: *Problem CONTROL-REGION-TIME is coNP-hard for $d \geqslant 2$.* ♦

Proof. The proof is exactly the same except for two details:

- we modify f over R_{n+1} as follows: divide R_{n+1} in three regions: R_{low} which is below R_{fin}, R_{fin}n and R_{high} which is above R_{fin}. Then build f such that $f(R_{low}) \subseteq R_{low}$, $f(R_{fin}) \subseteq R_{fin}$ and $f(R_{high}) \subseteq R_{low}$.
- we choose a new final region $R'_{fin} = R_{low}$.

Let $\mathcal{I} = (B, A_1, \dots, A_n)$ be an instance of NOSUBSET-SUM, let $\mathcal{J}$ be the corresponding instance of CONTROL-REGION-TIME we just built. We have to show that $\mathcal{I}$ has no subset sum if and only if $\mathcal{J}$ is "controlled". This is the same as showing that $\mathcal{I}$ has a subset sum if and only if $\mathcal{J}$ has points never reaching R'_{fin}.

Now assume for a moment that the instance is in SUBSET-SUM (as opposed to NOSUBSET-SUM), then by the same reasoning as the previous proof, there will be a point that reaches the old R_{fin} region (which is disjoint from R'_{fin}). And since R_{fin} is a f-stable region, this point will never reach R'_{fin}.

And conversely, if the control problem is not satisfied, necessarily there is a point whose trajectory went through the old R_{fin} (otherwise if would have reached either $R_{low} = R'_{fin}$ or R_{high} but $f(R_{high}) \subseteq R_{low}$). Now we proceed as in the proof of Theorem 6.3.20 to conclude that there is a subset that sums to B, and thus $\mathcal{I}$ is satisfiable. ■

Theorem 6.5.2: *Problem CONTROL-REGION-TIME is in coNP.* ♦

Proof. Again the proof is very similar to that of Theorem 6.4.5: we have to build a non-deterministic machine that accepts the "no" instances. The algorithm is exactly the same except that we only choose signatures that avoid the final region (as opposed to ending in the final region) and are of maximum length (that is $t = T$ as opposed to $t \leqslant T$). Indeed, if there is a such a trajectory, the problem is not satisfied. And for the same reasons as Theorem 6.4.5, it runs in non-deterministic polynomial time. ■

Chapter 7

Conclusion

Yeah, I used to think it was just recreational... then I started doin' it during the week... you know, simple stuff: differentiation, kinematics. Then I got into integration by parts... I started doin' it every night: path integrals, holomorphic functions. Now I'm on diophantine equations and sinking deeper into transfinite analysis. Don't let them tell you it's just recreational.
Fortunately, I can quit any time I want.

In this thesis, we gave several fundamental contributions regarding the relationship between the General Purpose Analog Computer (GPAC), which is a realistic model of computation, and Turing-based paradigms. In particular:

- We showed that time of a computation, for the GPAC, can be measured as the length of the curve (of the solution of the ODE). Alternatively, considering both time *and* space, for a suitable notion of space, provides an equivalent notion of complexity. Note that unlike most discrete models, there is no relationship between time and space in this model, which explains why any robust notion of complexity must consider both time and space. We proved that both notions of complexity are very robust and we gave several natural and equivalent definitions to illustrate this fact.

- We showed that the GPAC (or equivalently the class of PIVPs) has the same computational power as the Turing machine at the complexity level. More precisely, any GPAC can be efficiently solved by a Turing machine and any Turing machine can be efficiently simulated by a GPAC. This equivalence builds on the robust notion of complexity we introduced and suggests that the class of polynomial differential equations is the right compromise between power and tractability.

- We also gave a purely analog and machine-independent characterization of P and of polynomial time computable real functions. Indeed, our characterization relies solely on polynomial differential equations. This shows that even classical complexity classes can be presented in purely analog terms. As a side effect, this shows that solving polynomial differential equations is P-complete.

Despite our best efforts, we left a number of open problems and interesting questions which we intend to investigate in the future. We selected a few of the most interesting ones below.

Open Problem 1 (Complexity of solving PIVP)**:** In Theorem 3.5.2 we saw the complexity of computing $y(t) \pm \varepsilon$ where y is the solution of the polynomial initial value problem $y(t_0) = y_0$ and $y' = p(y)$ is bounded by:

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

Is it possible to replace $\operatorname{Len}(t_0, t)$ by the actual length of the curve[1] ? Is it possible to extend this result to more general ODEs like analytic computable ODEs ? Is it possible to distinguish between the time and space complexity ? ♦

Open Problem 2 (Characterization of FPSPACE)**:** In Theorem 5.3.3 we gave an equivalence between FP and a subset of AP. In can be seen that the proof will generalize to FPSPACE under two conditions:

- Prove a better bound for the space complexity in Theorem 3.5.2 (PIVP complexity), in particular it should be logarithmic in $\operatorname{Len}(t_0, t)$.
- Introduce a class like AP for space, in particular most proofs in this thesis will be the same except for the subtle issue that the Υ bound (in Definitions 4.2.7 ,4.3.1 and 4.4.2 for example) should not depend on the time t.

♦

Open Problem 3 (Characterization of NP)**:** In Theorem 5.3.9 we gave an equivalence between P and a class of PIVPs. Is it possible to find a natural characterization of NP ? One way to achieve this is to modify the PIVP to add a controller u: $y'(t) = p(y(t), u(t))$. One can show without much difficulty that some rather artificial restrictions on u are sufficient. For example, it is enough to require that $u \in$ GPVAL. ♦

More generally, one can wonder if and how it is possible to characterize the entire polynomial hierarchy. However we believe that characterizing a class like linear time will require substantial work, assuming it is even possible. Similarly, sublinear space will require entirely different techniques. Most probably the input will have to be given in an online fashion (just like Turing machines where the input is on a different, special tape) although the details are unclear.

Open Problem 4 (Length of the curve)**:** Is it possible to use the length of the curve as a measure of complexity for more general classes of ODEs ? The current proof heavily relies on the properties of polynomials to work but intuitively, other types of functions could work as long as they are smooth enough and computable. ♦

Open Problem 5 (Oracles)**:** Is it possible to introduce a natural notion of oracle in this model ? We think it is possible to do so by introducing an external controller that "reads" some variables of the system. More precisely, the system would be of the form $y'(t) = p(y(t), o(t))$ where $o(t)$ is the output of the oracle and $o(t)$ only depends on a subset $y_{k..l}$ of the variables. ♦

Open Problem 6 (Partial functions)**:** In [Ko91], partial functions are defined as those functions that can be computed by a machine over the domain of definition, with the additional property that the machine never halts if the input does not belong to this domain. This is in contrast with what with have done. Indeed, our notion of computability does not specify the behavior of the system outside of the domain. Is it possible to define a natural and equivalent

[1] Recall that Len is a over-approximation of the length, it particular, there is a gap between those two notions when the speed ($\|y'\|$) of the system becomes very small

notion ? One such idea would be to require that the system explodes in finite time if the output is outside of the domain. Another approach would be to make systems output both the result and the precision, and require that the precision never converges to 0 on bad inputs. ♦

Finally, we also mention some technical problems of interest but somehow low-level and more difficult to state.

Open Problem 7 (Domain of definition of generable functions)**:** We introduced generable functions in Definition 2.1.17 as solutions of particular polynomial partial differential equations, and we saw in Theorem 4.6.2 that some assumptions on the domain of definition were necessary for a generable function to be computable. Is it true that for any $f \in \text{GPVAL}_{\mathbb{K}}$, there exist a generable point $x_0 \in \operatorname{dom} f \cap \mathbb{K}^n$ and a parametrization $\gamma \in \text{GPVAL}_{\mathbb{K}}$ such that for any $x \in \operatorname{dom} f$, $\gamma(x,0) = x_0$, $\gamma(x,1) = x$ and $\gamma(x,[0,1]) \subseteq \operatorname{dom} f$? This condition can be relaxed to $\|x - \gamma(x,t)\| \leqslant e^{-t}$ and $\gamma(x,\mathbb{R}_+) \subseteq \operatorname{dom} f$. ♦

Open Problem 8 (Technical condition on the length of the curve)**:** In Definition 4.2.12 (Analog length computability) and Definition 5.3.8 (Discrete recognizability) we had to introduce a technical condition on the length of the curve (or equivalently on the speed of the system), namely that it grows at least linearly with time. Is it possible to remove this condition ? The rational for this condition is the following: if the speed of the system becomes very small, the system will need a possibly exponential time to reach the request length. On the other hand, if it moves slowly, it should be possible to simulate it much faster than real-time. See Example 4.2.14 (Technical condition) for such an example. Note however that this technical condition is not really a problem because it is always possible to add an extra variable (the time for example) so that this condition is always true. ♦

Bibliography

[AB01] Eugene Asarin and Ahmed Bouajjani. Perturbed Turing machines and hybrid systems. In *16th Annual IEEE Symposium on Logic in Computer Science*, page 269, 2001.

[ABBR12] A. Abad, R. Barrio, F. Blesa, and M. Rodríguez. Algorithm 924: Tides, a Taylor series integrator for differential equations. *ACM Trans. Math. Softw.*, 39(1):5:1–5:28, 2012.

[Abe70] O. Aberth. Computable analysis and differential equations. In A. Kino, J. Myhill, and R.E. Vesley, editors, *Intuitionism and Proof Theory*, Studies in Logic and the Foundations of Mathematics, pages 47–52. North-Holland, 1970.

[Abe71] O. Aberth. The failure in computable analysis of a classical existence theorem for differential equations. *Proc. Amer. Math. Soc.*, 30:151–156, 1971.

[Abe80] O. Aberth. *Computable Analysis.* McGraw-Hill, 1980.

[AD90] Rajeev Alur and David L. Dill. Automata for modeling real-time systems. In Mike Paterson, editor, *Automata, Languages and Programming, 17th International Colloquium, ICALP90, Warwick University, England, July 16-20, 1990, Proceedings*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer, 1990.

[AM98a] Eugene Asarin and Oded Maler. Achilles and the tortoise climbing up the arithmetical hierarchy. *Journal of Computer and System Sciences*, 57(3):389–398, December 1998.

[AM98b] Eugene Asarin and Oded Maler. Achilles and the tortoise climbing up the arithmetical hierarchy. *Journal of Computer and System Sciences*, 57(3):389–398, December 1998.

[AMP95] Eugene Asarin, Oded Maler, and Amir Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science*, 138:35–65, 1995.

[APZ03] Tamarah Arons, Amir Pnueli, and Lenore Zuck. Verification by probabilistic abstraction. In *POPL'2003*, 2003.

[Arn95] Ludwig Arnold. Random dynamical systems. In Russell Johnson, editor, *Dynamical Systems*, volume 1609 of *Lecture Notes in Mathematics*, pages 1–43. Springer Berlin Heidelberg, 1995.

[AS02] Eugene Asarin and Gerardo Schneider. Widening the boundary between decidable and undecidable hybrid systems. In Lubos Brim, Petr Jancar, Mojmír Kretínský, and Antonín Kucera, editors, *CONCUR 2002 - Concurrency Theory, 13th International Conference, Brno, Czech Republic, August 20-23, 2002, Proceedings*, volume 2421 of *Lecture Notes in Computer Science*, pages 193–208. Springer, 2002.

[ASY01] Eugene Asarin, Gerardo Schneider, and Sergio Yovine. On the decidability of the reachability problem for planar differential inclusions. In Maria Domenica Di Benedetto and Alberto L. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control, 4th International Workshop, HSCC 2001, Rome, Italy, March 28-30, 2001, Proceedings*, volume 2034 of *Lecture Notes in Computer Science*, pages 89–104. Springer, 2001.

[ASY07] Eugene Asarin, Gerardo Schneider, and Sergio Yovine. Algorithmic analysis of polygonal hybrid systems, part i: Reachability. *Theoretical Computer Science*, 379:231–265, 2007.

[Atk89] K. E. Atkinson. *An Introduction to Numerical Analysis.* John Wiley & Sons, 2nd edition, 1989.

[BA13] Amir M Ben-Amram. Mortality of iterated piecewise affine functions over the integers: Decidability and complexity. In *STACS*, pages 514–525, 2013.

[BBEP10] Marie-Pierre Béal, Jean Berstel, Soren Eilers, and Dominique Perrin. Symbolic dynamics. 2010. to appear in Handbook of Automata.

[BBGP14] Hugo Bazille, Olivier Bournez, Walid Gomaa, and Amaury Pouly. On the complexity of bounded time reachability for piecewise affine systems. In *Reachability Problems - 8th International Workshop, RP 2014, Oxford, UK, September 22-24, 2014. Proceedings*, pages 20–31, 2014.

[BBKT01] Vincent D. Blondel, Olivier Bournez, Pascal Koiran, and John Tsitsiklis. The stability of saturated linear dynamical systems is undecidable. *Journal of Computer and System Science*, 62(3):442–462, May 2001.

[BC08] Olivier Bournez and Manuel L. Campagnolo. *New Computational Paradigms. Changing Conceptions of What is Computable*, chapter A Survey on Continuous Time Computations, pages 383–423. Springer-Verlag, New York, 2008.

[BC13] PaulC. Bell and Shang Chen. Reachability problems for hierarchical piecewise constant derivative systems. In ParoshAziz Abdulla and Igor Potapov, editors, *Reachability Problems*, volume 8169 of *Lecture Notes in Computer Science*, pages 46–58. Springer Berlin Heidelberg, 2013.

[BCdNM05] 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.

[BCGH07] O. Bournez, M. L. Campagnolo, D. S. Graça, and E. Hainry. Polynomial differential equations compute all real computable functions on computable compact intervals. *J. Complexity*, 23(3):317–335, 2007.

[BCO$^+$07] Alin Bostan, Frédéric Chyzak, François Ollivier, Bruno Salvy, Éric Schost, and Alexandre Sedoglavic. Fast computation of power series solutions of systems of differential equations. In *SODA'07*, pages 1012–1021, January 2007.

[BCPT14] Edwin J. Beggs, José Félix Costa, Diogo Poças, and John V. Tucker. An analogue-digital church-turing thesis. *Int. J. Found. Comput. Sci.*, 25(4):373–390, 2014.

[BCSS98] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation.* Springer, 1998.

[BDF12] Olivier Bournez, Nachum Dershowitz, and Evgenia Falkovich. Towards an axiomatization of simple analog algorithms. In Manindra Agrawal, S. Barry Cooper, and Angsheng Li, editors, *Theory and Applications of Models of Computation - 9th Annual Conference, TAMC 2012, Beijing, China, May 16-21, 2012. Proceedings*, volume 7287 of *Lecture Notes in Computer Science*, pages 525–536. Spinger-Verlag, 2012.

[BG03] A. Blass and Y. Gurevich. Abstract state machines capture parallel algorithms. *ACM Transactions on Computational Logic (TOCL)*, 4(4):578–651, 2003.

[BGH10] Olivier Bournez, Daniel S. Graça, and Emmanuel Hainry. Robust computations with dynamical systems. In *Mathematical Foundations of Computer Science, MFCS'2010*, volume 6281 of *Lecture Notes in Computer Science*, pages 198–208. Springer, 2010.

[BGH13] Olivier Bournez, Daniel S. Graça, and Emmanuel Hainry. Computation with perturbed dynamical systems. *Journal of Computer and System Sciences*, 79(5):714 – 724, 2013.

[BGP] O. Bournez, D. S. Graça, and A. Pouly. Turing machines can be efficiently simulated by the general purpose analog computer. Submitted for publication.

[BGP12] Olivier Bournez, Daniel S. Graça, and Amaury Pouly. On the complexity of solving initial value problems. In *37h International Symposium on Symbolic and Algebraic Computation (ISSAC)*, volume abs/1202.4407, 2012.

[BGPZ13] Olivier Bournez, Daniel S. Graça, Amaury Pouly, and Ning Zhong. Computability and computational complexity of the evolution of nonlinear dynamical systems. In *The Nature of Computation. Logic, Algorithms, Applications - 9th Conference on Computability in Europe, CiE 2013, Milan, Italy, July 1-5, 2013. Proceedings*, pages 12–21, 2013.

[BGZ11] J. Buescu, D.S. Graça, and N. Zhong. Computability and dynamical systems. In Mauricio Matos Peixoto, Alberto Adrego Pinto, and David A. Rand, editors, *Dynamics, Games and Science I*, volume 1 of *Springer Proceedings in Mathematics*, pages 169–181. Springer Berlin Heidelberg, 2011.

[BH04] Olivier Bournez and Emmanuel Hainry. An analog characterization of elementary computable functions over the real numbers. In Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, editors, *International Colloquium on Automata, Languages and Programming (ICALP 2004)*, volume 3142 of *Lecture Notes in Computer Science*, pages 269–280, 2004.

[BH05a] Olivier Bournez and Emmanuel Hainry. Elementary computable functions over the real numbers and R-sub-recursive functions. *Theoretical Computer Science*, 348(2-3):130–147, December 2005.

[BH05b] Olivier Bournez and Emmanuel Hainry. Real recursive functions and real extentions of recursive functions. In Maurice Margenstern, editor, *Machines, Computations, and Universality, MCU 2004*, volume 3354 of *Lecture Notes in Computer Science*, pages 116–127. Springer-Verlag, 2005.

[BH06] Olivier Bournez and Emmanuel Hainry. Recursive analysis characterized as a class of real recursive functions. 74(4):409–433, 2006.

[Bou97] Olivier Bournez. Some bounds on the computational power of piecewise constant derivative systems (extended abstract). In *ICALP*, pages 143–153, 1997.

[Bou99] Olivier Bournez. Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy. *Theoret. Comput. Sci.*, 210(1):21–71, 1999.

[BP97] Marie-Pierre Béal and Dominique Perrin. Symbolic dynamics and finite automata. In *Handbook of formal languages, Vol. 2*, pages 463–505. Springer, Berlin, 1997.

[Bra95] M. S. Branicky. Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoret. Comput. Sci.*, 138(1):67–100, 1995.

[Bra05a] Vasco Brattka. Computability & complexity in analysis. Tutorial donné à *Computability in Europe (CIE'2005)*, 2005.

[Bra05b] Mark Braverman. On the complexity of real functions. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 155–164, 2005.

[BRAB11] R. Barrio, M. Rodríguez, A. Abad, and F. Blesa. Breaking the limits: the Taylor series method. *Appl. Math. Comput.*, 217(20):7940–7954, 2011.

[Bro99] M. Broucke. Geometric approach to bisimulation and verification of hybrid systems. volume 1569 of *Lecture Notes in Computer Science*, pages 61–??, 1999.

[BS95] François Bergeron and Ulrike Sattler. Constructible differentially finite algebraic series in several variables. *Theoretical Computer Science*, 144(1–2):59 – 65, 1995.

[BS02] M. Brin and G. Stuck. *Introduction to Dynamical Systems.* Cambridge University Press, 2002.

[BS03] E. Börger and R.F. Stärk. *Abstract State Machines.* Springer, 2003.

[BSS89] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bull. Amer. Math. Soc.*, 21(1):1–46, 1989.

[Bus31] V. Bush. The differential analyzer. A new machine for solving differential equations. *J. Franklin Inst.*, 212:447–488, 1931.

[But87] J. C. Butcher. *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods.* Wiley-Interscience, New York, NY, USA, 1987.

[CC82] G. Corliss and Y. F. Chang. Solving ordinary differential equations using Taylor series. *ACM Trans. Math. Softw.*, 8(2):114–144, 1982.

[CG09] P. Collins and D. S. Graça. Effective computability of solutions of differential inclusions — the ten thousand monkeys approach. *Journal of Universal Computer Science*, 15(6):1162–1185, 2009.

[CMC00] M. L. Campagnolo, C. Moore, and J. F. Costa. Iteration, inequalities, and differentiability in analog computers. *J. Complexity*, 16(4):642–660, 2000.

[CMC02] Manuel L. Campagnolo, Cristopher Moore, and José Félix Costa. An analog characterization of the Grzegorczyk hierarchy. *J. Complexity*, 18(4):977–1000, 2002.

[Cop98] B. Jack Copeland. Even Turing machines can compute uncomputable functions. In C.S. Calude, J. Casti, and M.J. Dinneen, editors, *Unconventional Models of Computations*. Springer-Verlag, 1998.

[Cop02] B. Jack Copeland. Accelerating Turing machines. *Minds and Machines*, 12:281–301, 2002.

[Cor02] Robert M. Corless. A new view of the computational complexity of IVP for ODE. *Numerical Algorithms*, 31(1-4):115–124, 2002.

[CP02] C. S. Calude and B. Pavlov. Coins, quantum measurements, and Turing's barrier. *Quantum Information Processing*, 1(1-2):107–127, April 2002.

[CPSW05] D. C. Carothers, G. E. Parker, J. S. Sochacki, and P. G. Warne. Some properties of solutions to polynomial systems of differential equations. *Electron. J. Diff. Eqns.*, 2005(40), April 2005.

[CY95] Costas Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, July 1995.

[Dav01] E. B. Davies. Building infinite machines. *The British Journal for the Philosophy of Science*, 52:671–682, 2001.

[Deu85] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. R. Soc. Lond. Ser. A*, A400:97–117, 1985.

[DG08] N. Dershowitz and Y. Gurevich. A natural axiomatization of computability and proof of Church's Thesis. *The Bulletin of Symbolic Logic*, 14(3):299–350, 2008.

[DL12] Ugo Dal Lago. A short introduction to implicit computational complexity. In *Lectures on Logic and Computation*, pages 89–109. Springer, 2012.

[Dur12] Jérôme Durand-Lose. Abstract geometrical computation 7: geometrical accumulations and computably enumerable real numbers. *Natural Computing*, 11(4):609–622, 2012.

[EM03] L.Hernàndez Encinas and J.Muñoz Masqué. A short proof of the generalized faà di bruno's formula. *Applied Mathematics Letters*, 16(6):975 – 979, 2003.

[EN02] Gábor Etesi and István Németi. Non-Turing computations via Malament-Hogarth space-times. *International Journal Theoretical Physics*, 41:341–370, 2002.

[Fei88] D. G. Feitelson. *Optical computing: a survey for computer scientists.* MIT Press, 1988.

[Fey82] R. P. Feynman. Simulating physics with computers. *Internat. J. Theoret. Phys.*, 21(6/7):467–488, 1982.

[GBC07] D. S. Graça, J. Buescu, and M. L. Campagnolo. Boundedness of the domain of definition is undecidable for polynomial ODEs. In R. Dillhage, T. Grubba, A. Sorbi, K. Weihrauch, and N. Zhong, editors, *4th International Conference on Computability and Complexity in Analysis (CCA 2007)*, volume 202 of *Electron. Notes Theor. Comput. Sci.*, pages 49–57. Elsevier, 2007.

[GBC09] D. S. Graça, J. Buescu, and M. L. Campagnolo. Computational bounds on polynomial differential equations. *Appl. Math. Comput.*, 215(4):1375–1385, 2009.

[GC03] Daniel S. Graça and José Félix Costa. Analog computers and recursive functions over the reals. *Journal of Complexity*, 19(5):644–664, 2003.

[GCB05] D. S. Graça, M. L. Campagnolo, and J. Buescu. Robust simulations of Turing machines with analytic maps and flows. In S. B. Cooper, B. Löwe, and L. Torenvliet, editors, *CiE 2005: New Computational Paradigms*, LNCS 3526, pages 169–179. Springer, 2005.

[GCB08] D. S. Graça, M. L. Campagnolo, and J. Buescu. Computability with polynomial differential equations. *Adv. Appl. Math.*, 40(3):330–349, 2008.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., 1979.

[GM95] Erich Grädel and Klaus Meer. Descriptive complexity theory over the real numbers. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, pages 315–324, Las Vegas, Nevada, 29May–1June 1995. ACM Press.

[GM02] Marco Gori and Klaus Meer. A step towards a complexity theory for analog systems. *Mathematical Logic Quarterly*, 48(Suppl. 1):45–58, 2002.

[Gra04] D. S. Graça. Some recent developments on Shannon's General Purpose Analog Computer. *Math. Log. Quart.*, 50(4-5):473–485, 2004.

[Grz55] A. Grzegorczyk. Computable functionals. *Fund. Math.*, 42:168–202, 1955.

[GZ09] D. S. Graça and N. Zhong. Computing domains of attraction for planar dynamics. In C. S. Calude, J. F. Costa, N. Dershowitz, E. Freire, and G. Rozenberg, editors, *8th International Conference on Unconventional Computation (UC 2009)*, LNCS 5715, pages 179–190. Springer, 2009.

[GZ11] Daniel Graça and Ning Zhong. Computability in planar dynamical systems. *Natural Computing*, 10(4):1295–1312, 2011.

[GZB09] D.S. Graça, N. Zhong, and J. Buescu. Computability, noncomputability and undecidability of maximal intervals of IVPs. *Trans. Amer. Math. Soc.*, 361(6):2913–2927, 2009.

[Hir01] M. Hirvensalo. *Quantum Computing.* Springer, 2001.

[HKPV98] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57(1):94–124, August 1998.

[HLMP04] Thomas Hérault, Richard Lassaigne, Frédéric Magniette, and Sylvain Peyronnet. Approximate probabilistic model checking. In Bernhard Steffen and Giorgio Levi, editors, *Verification, Model Checking, and Abstract Interpretation, 5th International Conference, VMCAI 2004, Venice, January 11-13, 2004, Proceedings*, volume 2937 of *Lecture Notes in Computer Science*, pages 73–84. Springer, 2004.

[HSD04] M. W. Hirsch, S. Smale, and R. Devaney. *Differential Equations, Dynamical Systems, and an Introduction to Chaos.* Academic Press, 2004.

[ISC08] Silvana Ilie, Gustaf Söderlind, and Robert M. Corless. Adaptivity and computational complexity in the numerical solution of odes. *J. Complexity*, 24(3):341–361, 2008.

[JZ05] À. Jorba and M. Zou. A software package for the numerical integration of odes by means of high-order Taylor methods. *Experimental Mathematics*, 14(1):99–117, 2005.

[Kaw05] Akitoshi Kawamura. Type-2 computability and moore's recursive functions. *Electronic Notes in Theoretical Computer Science*, 120(0):83 – 95, 2005. Proceedings of the 6th Workshop on Computability and Complexity in Analysis (CCA 2004) Computability and Complexity in Analysis 2004.

[Kaw10] A. Kawamura. Lipschitz continuous ordinary differential equations are polynomial-space complete. *Computational Complexity*, 19(2):305–332, 2010.

[KC10] A. Kawamura and S. A. Cook. Complexity theory for operators in analysis. In Leonard J. Schulman, editor, *STOC*, pages 495–502. ACM, 2010.

[KCG94a] P. Koiran, M. Cosnard, and M. Garzon. Computability with low-dimensional dynamical systems. *Theoret. Comput. Sci.*, 132:113–128, 1994.

[KCG94b] Pascal Koiran, Michel Cosnard, and Max Garzon. Computability with Low-Dimensional Dynamical Systems. *Theoretical Computer Science*, 132:113–128, 1994.

[KMRZ12] A. Kawamura, N. Th. Müller, C. Rösnick, and M. Ziegler. Parameterized uniform complexity in numerics: from smooth to analytic, from np-hard to polytime. *CoRR*, abs/1211.4974, 2012.

[KNSS02] Marta Kwiatkowska, Gethin Norman, Roberto Segala, and Jeremy Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science*, 282(1):101–150, May 2002.

[Ko91] Ker-I Ko. *Complexity Theory of Real Functions*. Progress in Theoretical Computer Science. Birkhaüser, Boston, 1991.

[KO14] Akitoshi Kawamura and Hiroyuki Ota. Small complexity classes for computable analysis. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Mathematical Foundations of Computer Science 2014*, volume 8635 of *Lecture Notes in Computer Science*, pages 432–444. Springer Berlin Heidelberg, 2014.

[Koi94] Pascal Koiran. Computing over the reals with addition and order. *Theor. Comput. Sci.*, 133(1):35–47, 1994.

[KORZ14] Akitoshi Kawamura, Hiroyuki Ota, Carsten Rösnick, and Martin Ziegler. Computational complexity of smooth differential equations. *Logical Methods in Computer Science*, 10(1), 2014.

[KP02] S. G. Krantz and H. R. Parks. *A Primer of Real Analytic Functions*. Birkhäuser, 2nd edition, 2002.

[KW95] K. Ko and K. Weihrauch. Computability and complexity in analysis, 1995.

[Lac55] D. Lacombe. Extension de la notion de fonction récursive aux fonctions d'une ou plusieurs variables réelles III. *C. R. Acad. Sci. Paris*, 241:151–153, 1955.

[Lig91] W. Light. *Advances in Numerical Analysis*, chapter Numerical methods for dynamical systems. Clarendon press-Oxford, 1991.

[Lip89] L Lipshitz. D-finite power series. *Journal of Algebra*, 122(2):353 – 373, 1989.

[LT03] Stig Larsson and Vidar Thomée. *Partial differential equations with numerical methods.* Texts in applied mathematics. Springer, Berlin, 2003.

[Ma09] Tsoy-Wo Ma. Higher chain formula proved by combinatorics. *Electr. J. Comb.*, 16(1), 2009.

[MC04] Jerzy Mycka and José Félix Costa. Real recursive functions and their hierarchy. *Journal of Complexity*, 20(6):835–857, 2004.

[MC06] J. Mycka and J. F. Costa. The $p \neq np$ conjecture in the context of real and complex analysis. *J. Complexity*, 22(2):287–303, 2006.

[MM93] N. Müller and B. Moiske. Solving initial value problems in polynomial time. In *Proc. 22 JAIIO - PANEL '93, Part 2*, pages 283–293, 1993.

[Moo90] Cristopher Moore. Unpredictability and undecidability in dynamical systems. *Phys. Rev. Lett.*, 64:2354–2357, May 1990.

[Moo91] Cristopher Moore. Generalized shifts: unpredictability and undecidability in dynamical systems. *Nonlinearity*, 4(3):199–230, 1991.

[Moo96] Cristopher Moore. Recursion theory on the reals and continuous-time computation. *Theoretical Computer Science*, 162(1):23 – 44, 1996.

[PE74] M. B. Pour-El. Abstract computability and its relations to the general purpose analog computer. *Trans. Amer. Math. Soc.*, 199:1–28, 1974.

[PER79] M. B. Pour-El and J. I. Richards. A computable ordinary differential equation which possesses no computable solution. *Ann. Math. Logic*, 17:61–90, 1979.

[Pla12] André Platzer. Dynamic logics of dynamical systems. *CoRR*, abs/1205.4788, 2012.

[PV94] A. Puri and P. Varaiya. Decidability of hybrid systems with rectangular differential equations. In *Proc. 6th Workshop on Computer-Aided Verification*, LNCS 818, pages 95–104. Springer, 1994.

[Ruo93] Keijo Ruohonen. Undecidability of event detection for ODEs. *Journal of Information Processing and Cybernetics*, 29:101–113, 1993.

[Ruo94] Keijo Ruohonen. Event detection for ODEs and nonrecursive hierarchies. In *Proceedings of the Colloquium in Honor of Arto Salomaa. Results and Trends in Theoretical Computer Science (Graz, Austria, June 10-11, 1994)*, volume 812 of *Lecture Notes in Computer Science*, pages 358–371. Springer-Verlag, Berlin, 1994.

[Ruo96] K. Ruohonen. An effective Cauchy-Peano existence theorem for unique solutions. *Internat. J. Found. Comput. Sci.*, 7(2):151–160, 1996.

[SF98] H. T. Siegelmann and S. Fishman. Analog computation with dynamical systems. *Phys. D*, 120:214–235, 1998.

[Sha41] C. E. Shannon. Mathematical theory of the differential analyser. *Journal of Mathematics and Physics MIT*, 20:337–354, 1941.

[Sho97] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26:1484–1509, 1997.

[Smi06] Warren D. Smith. Church's thesis meets the N-body problem. *Applied Mathematics and Computation*, 178(1):154–183, 2006.

[SS95] H. T. Siegelmann and E. D. Sontag. On the computational power of neural networks. *J. Comput. System Sci.*, 50(1):132–150, 1995.

[Tar55] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Math.*, 5(2):285–309, 1955.

[TG95] A. Turing and J.-Y. Girard. *La machine de Turing*. Seuil, 1995.

[Tur36] A. M. Turing. On computable numbers with an application to the entscheidungsproblem. *Proc. London Mathematical Society*, 42(2):230–265, 1936. Traduction [TG95].

[Var85] Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *26th Annual Symposium on Foundations of Computer Science*, pages 327–338, Portland, Oregon, 21–23October 1985. IEEE Computer Society Press.

[Wei00] K. Weihrauch. *Computable Analysis: an Introduction.* Springer, 2000.

[Wer79] A.G. Werschulz. Computational complexity of one-step methods for a scalar autonomous differential equation. *Computing*, 23(4):345–355, 1979.

[WN05] Damien Woods and Thomas J. Naughton. An optical model of computation. *Theoretical Computer Science*, 334(1-3):227–258, 2005.

[WWS$^+$06] P. G. Warne, D.A. Polignone Warne, J. S. Sochacki, G. E. Parker, and D. C. Carothers. Explicit a-priori error bounds and adaptive error control for approximation of nonlinear initial value differential systems. *Comput. Math. Appl.*, 52(12):1695–1710, December 2006.

Titre: Modèles de calcul à temps continu: de la calculabilité à la complexité

Résumé:
En 1941, Claude Shannon définit le General Purpose Analog Computer (GPAC), un modèle de calcul analogique à temps continu. Le GPAC est un modèle réaliste car il peut être construit mécaniquement ou bien à l'aide circuit électroniques. Il s'avère que les fonctions calculables par ce modèle sont exactement les solutions d'une certaine classe d'équations différentielles à second membre polynomial. Bien que les ordinateurs digitaux aient remplacé les ordinateurs analogiques, la question de savoir si ces modèles sont comparables reste en suspens.

Il y a quelques années, cette thématique est réapparue à l'occasion d'une preuvre montrant que le GPAC et les machines de Turing ont la même puissance de calcul. Toutefois ce résultat ne nous aide guère à comprendre la relation entre ces deux modèles au niveau de la complexité des calculs. En d'autres termes, les ordinateurs analogiques ne calculent pas plus de fonctions que les ordinateurs classiques, mais il se pourrait qu'ils les calculent plus vite. Cette problématique est intrinsèquement reliée à celle, fondamentale, de la définition même de la complexité d'un système à temps et espace continu. En effet, ces systèmes exhibent le paradoxe troublant de Zenon, c'est à dire celui de la contraction de l'espace et du temps.

Cette thèse apporte des réponses fondamentales à ces questions. Nous montrons que la complexité d'un calcul par le GPAC peut être mesurée par la longueur de la courbe ainsi définie. Nous montrons ensuite que le GPAC et les machines de Turing ont la même puissance de calcul au niveau de la complexité. Enfin nous donnons une caractérisation purement analogique, et indépendante de toute notion de machine, de la classe P ainsi que de l'analyse récursive.

Title: Continuous models of computation: from computability to complexity

Summary:
In 1941, Claude Shannon introduced a continuous-time analog model of computation, namely the General Purpose Analog Computer (GPAC). The GPAC is a physically feasible model in the sense that it can be implemented in practice through the use of analog electronics or mechanical devices. It can be proved that the functions computed by a GPAC are precisely the solutions of a special class of differential equations where the right-hand side is a polynomial. Analog computers have since been replaced by digital counterpart. Nevertheless, one can wonder how the GPAC could be compared to Turing machines.

A few years ago, it was shown that Turing-based paradigms and the GPAC have the same computational power. However, this result did not shed any light on what happens at a computational complexity level. In other words, analog computers do not make a difference about what can be computed; but maybe they could compute faster than a digital computer. A fundamental difficulty of continuous-time model is to define a proper notion of complexity. Indeed, a troubling problem is that many models exhibit the so-called Zeno's phenomenon, also known as space-time contraction.

In this thesis, we give several fundamental contributions to these questions. We show that the GPAC has the same computational power as the Turing machine, at the complexity level. We also provide as a side effect a purely analog, machine-independent characterization of P and Computable Analysis.