UNIVERSIDADE TÉCNICA DE LISBOA

INSTITUTO SUPERIOR TÉCNICO

Computability with Polynomial Differential Equations

Daniel da Silva Graça
(Mestre)

Dissertação para obtenção do Grau de Doutor em Matemática

DOCUMENTO PROVISÓRIO

Janeiro de 2007

COMPUTABILIDADE COM EQUAÇÕES DIFERENCIAIS POLINOMIAIS

Nome: Daniel da Silva Graça
Curso de doutoramento em: Matemática
Orientador: Doutor Manuel Lameiras de Figueiredo Campagnolo
Co-orientador: Doutor Jorge Sebastião de Lemos Carvalhão Buescu
Provas concluídas em:

Resumo: Nesta dissertação iremos analisar um modelo de computação analógica, baseado em equações diferenciais polinomiais.

Começa-se por estudar algumas propriedades das equações diferenciais polinomiais, em particular a sua equivalência a outro modelo baseado em circuitos analógicos (GPAC), introduzido por C. Shannon em 1941, e que é uma idealização de um dispositivo físico, o Analisador Diferencial.

Seguidamente, estuda-se o poder computacional do modelo. Mais concretamente, mostra-se que ele pode simular máquinas de Turing, de uma forma robusta a erros, pelo que este modelo é capaz de efectuar computações de Tipo-1. Esta simulação é feita em tempo contínuo. Mais, mostramos que utilizando um enquadramento apropriado, o modelo é equivalente à Análise Computável, isto é, à computação de Tipo-2.

Finalmente, estudam-se algumas limitações computacionais referentes aos problemas de valor inicial (PVIs) definidos por equações diferenciais ordinárias. Em particular: (i) mostra-se que mesmo que o PVI seja definido por uma função analítica e que a mesma, assim como as condições iniciais, sejam computáveis, o respectivo intervalo maximal de existência da solução não é necessariamente computável; (ii) estabelecem-se limites para o grau de não-computabilidade, mostrando-se que o intervalo maximal é, em condições muito gerais, recursivamente enumerável; (iii) mostra-se que o problema de decidir se o intervalo maximal é ou não limitado é indecídivel, mesmo que se considerem apenas PVIs polinomiais.

Palavras-chave: Computabilidade, computação analógica, análise computável, equações diferenciais ordinárias, problemas de valor inicial, intervalo maximal.

Computability with Polynomial Differential Equations

Abstract: The purpose of the present dissertation is to analyze a model of analog computation defined with polynomial differential equations.

This work starts by studying some properties of polynomial differential equations and, in particular, their equivalence to the functions computed by Shannon's General Purpose Analog Computer (GPAC), a model that was introduced in 1941 as an idealization of a physical device, the Differential Analyzer.

We then study the model's computational power. More concretely, we show that it can perform robust simulations of Turing machines, thus achieving Type-1 computability. Our simulation is done in continuous-time. Moreover, we show that in an appropriate framework, this model is equivalent to computable analysis i.e., to Type-2 computability.

We pursue our digression on models based on differential equations by showing some computational limitations concerning initial-value problems (IVPs) defined by ordinary differential equations. Specifically: (i) we prove that even if the IVP is defined with an analytic function that, together with the initial conditions, is computable, then the maximal interval of existence for the respective solution is not necessarily computable; (ii) we establish limits for the "degree of noncomputability", by showing that the maximal interval is, under very mild conditions, recursively enumerable; (iii) we show that the problem of deciding whether the maximal interval is bounded or not is undecidable, even if we only consider polynomial IVPs.

Keywords: Computability, analog computation, computable analysis, ordinary differential equations, initial-value problems, maximal interval.

ACKNOWLEDGMENTS

Many people helped me while I was working in my PhD thesis. I would like to thank all of them. However, there were some people to which I interacted more closely and that I would like to mention here.

Firstly, I would like to express my gratitude to my advisors, Jorge Buescu and Manuel Campagnolo, for their dedication and guidance during the elaboration of this thesis. The constant encouragement they provided was essential for the determination of the course of this work. I am indebted to them for the time and attention they devoted to this project.

Several institutions gave me the conditions I needed to pursue this research. Among these, I am especially grateful to the department of Mathematics at FCT/University of Algarve for providing me support and a leave for two and half years. Also, I thank the people at the Section of Computer Science, the Center for Logic and Computation at IST/UTL, and the Security and Quantum Information Group at IT, for their constant assistance and interest in my work. In particular, the organizers of the Logic and Computation Seminar, Amílcar Sernadas and Carlos Caleiro, provided me several opportunities to present my research.

A special acknowledgement goes to the Protheo group at LORIA/INRIA, France which hosted me in several visits during the elaboration of this thesis, adding up to almost one year. Among them, I must mention Olivier Bournez and Emmanuel Hainry for all the enriching discussions we had. The work done there resulted on Chapter 5 of the current thesis. Other persons at LORIA also gave me support in other ways: Anderson Oliveira, Raúl Brito, Ricardo Marques (the Portuguese speaking people), Antoine Reilles (for my FreeBSD problems), Johanne Cohen, Chantal Llorens, and several people from the following teams: Calligrame, Carte, Cassis, and Protheo.

I would also like to thank Ning Zhong for showing interest on a problem that we had the chance to discuss during the conference CiE'2005. This would originate a fruitful collaboration, which is reflected in Chapter 6. I am also grateful for the excellent hospitality that she and Bingyu Zhang provided during my stays in Cincinnati.

Many people helped me with useful discussions and suggestions, while doing this work, in earlier or current aspects. In particular, I wish to thank J. Félix Costa, V. Brattka, and C. Moore. I would also like to thank P. Koiran, R. Lupacchini, G. Sandri, and G. Tamburrini for giving me opportunities to present my work on seminars/workshops.

Last but certainly not least, I wish to give very special thanks to my family, for their presence and unconditional support.

This research was partially supported by the following entities:

- Fundação para a Ciência e a Tecnologia and EU FEDER POCTI/POCI via Center for Logic and Computation at IST/UTL, the Security and Quantum Information Group at IT, grant SFRH/BD/17436/2004, and the project ConTComp POCTI/MAT/45978/2002,
- Fundação Calouste Gulbenkian through the *Programa Gulbenkian de Estímulo à Investigação*,
- EGIDE and GRICES under the Program *Pessoa* through the project *Calculabilité et complexité des modèles de calculs à temps continu*.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Dynamical systems are a powerful tool to model natural phenomena. Their use is transversal to almost all "exact sciences" and applications can be found ranging from fields like physics or chemistry up to biology or economics. However this versatility and wealth of applications comes with a price: dynamical systems can present an incredibly complex behavior, and their understanding is far from complete. Indeed, only recently we started to have a glimpse on the richness that even systems defined with simple rules can present. In section 1.2 we recall some classical examples of such systems.

The purpose of the present work is to give new results for a particular class of dynamical systems that, while being restricted, is still sufficiently broad to accommodate a full range of meaningful systems. Here we use a computational approach and we try to understand which are the computational strengths and weaknesses of such systems.

In general, we can consider two large families of dynamical systems: the discrete-time ones and the continuous-time ones. The evolution of the first systems correspond to the iteration of a function, and the evolution of the second ones corresponds to the solution of some differential equation.

In this thesis we study, from a computational perspective, the class of continuous-time dynamical systems defined over $\mathbb{R}^n$ that are solution of some initial-value problem

$$\begin{cases} x' = p(t,x) \\ x(t_0) = x_0 \end{cases} \tag{1.1}$$

where p is a vector of polynomials. Notice that, while the previous class of dynamical systems encompasses practically all analytic continuous-time systems that appear in applications (e.g. all examples of Section 1.2 are of this type), this class is usually not studied on its own in standard books about dynamical systems, e.g. [CL55], [Lef65], [Hal80], [HSD04].

As matter of fact, the emphasis is on linear systems, which we know how to solve explicitly, and in planar systems (i.e., defined in $\mathbb{R}^2$) whose behavior is, in a qualitative way, completely understood [HW95]. But besides that, and with the exception of a marginal number of cases, little information is known about the general properties of systems like (1.1). For instance, Hilbert's 16th problem: "determine an upper bound for the number of limit cycles in polynomial

vector fields of order n" is still an open problem despite belonging to the famous list of 22 problems formulated by D. Hilbert in 1900.

We notice that the study of dynamical systems from a computational point of view is not new. Indeed, recently there has been a renewed interest in *analog computation*, and computation over the reals. The distinction between these two areas is not clear, and varies from author to author.

To some extent, one can consider three levels when working with analog computation or computation over the reals: (i) a physics/engineering level, where the emphasis is done on building analog computing machines, (ii) an abstraction level, where one is interested in mathematical models of the previous analog machines, and (iii) a theoretical level, where one is interested in computation with reals, but where the models used do not necessarily have any connection with analog machines. The existing literature about the subject suggests that analog computation is mainly concerned with the levels (i)–(ii), and computation over the reals with levels (ii)–(iii), although this is not a strict classification.

For instance, the design of Differential Analyzers [Bus31], a kind of analog machine, falls within the first category, while the study of its abstraction, the General Purpose Analog Computer [Sha41], falls within the second. Finally, the third category includes models such as the BSS model [BSS89] or computable analysis [PER89], [Ko91], [Wei00]. Note that this separation is not strict, and the same model can be considered in different levels, depending upon the author. For instance, for some authors, computable analysis [BH98] and the BSS model [Sma92] can also be considered in level (ii).

In the present thesis we will be mainly concerned with the level (ii) described above, but we will also delve into level (iii), in Chapters 5 and 6.

Another motivation to study dynamical systems from a computational perspective comes from control theory. Many systems arising in control theory can be modeled as continuous dynamical systems or hybrid systems (where some components of the system are continuous, and others are discrete), and some questions arise naturally within this context, see e.g. [Bro89], [Bra95], [Son98]. For instance, given a point from some set (inputs), is it possible that its trajectory will reach another set (of invalid states)?

If we can answer, from a computational perspective, some of the previous questions we can then provide automated verification tools (where a digital computer is used) for those systems, ensuring that they will behave correctly [AD90], [AMP95], [AD00], [HKPV98], [PV94], [NOSS93], [BT00], [Cv04].

Again, these problems relate to the study of dynamical systems from a computational perspective, which is the core theme of the present work.

1.2 On the complexity of some simple dynamical systems

In this section we briefly recall some examples of dynamical systems of the type (1.1), defined with simple evolution rules, that yet have a complex behavior. With this we not only want to alert the reader to the difficulty of tackling them, but also to emphasize their importance in applications. This is why we choose models that derived from practical problems. We note that we will only present qualitative analysis of the respective equations (1.1), since in general, explicit solutions cannot be constructed.

Biology: Lotka-Volterra equations

In the early 1920s, Vito Volterra came up with a now famous example of differential equations in $\mathbb{R}^2$ to model a predator-prey system. This system was developed to give an answer to the following problem. Umberto d'Ancona, who was an official in the Italian bureau of fisheries in

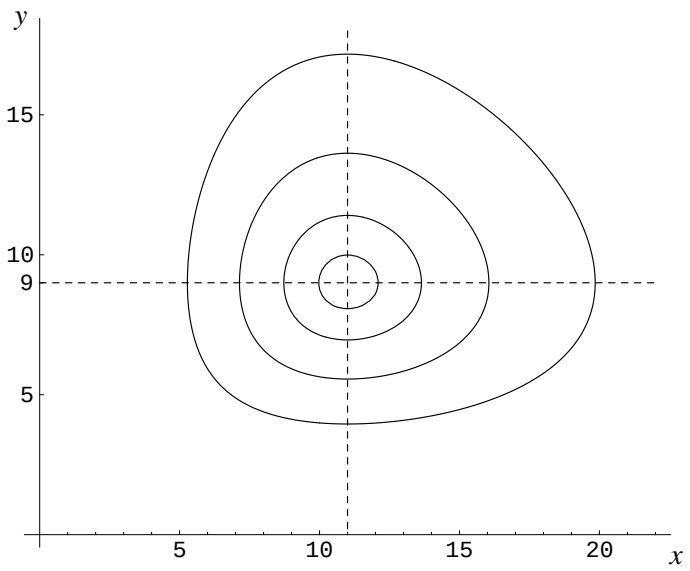


Figure 1.1: Phase plane trajectories for the Lotka-Volterra equation with parameters $\alpha = 0.9$, $\beta = 1.1$, and $\delta = \gamma = 0.1$.

Trieste during World War I, was puzzled by the statistics he kept. Specifically, he remarked that during World War I, the proportion of catch that consisted of predator fish, like sharks or skates, increased markedly over what it had been before, and what it would became later. This was somehow disconcerting, since the war period imposed severe restrictions to fishing activities, thus releasing the pressure over "food fish". He then presented this problem to Volterra, who came out with the following set of equations

$$\begin{aligned} x' &= \alpha x - \gamma xy \\ y' &= -\beta y + \delta xy \end{aligned} \tag{1.2}$$

Here $x(t)$ represents the number of prey fish and $y(t)$ the number of predators. The coefficient α represents the fertility rate of the prey fishes and γ is a coefficient that gives their mortality rate, due to encounters with predators (this value is proportional to $x(t)y(t)$). Reciprocally δ is a coefficient proportional to the number of meals that the predators have in average. It is also supposed that without any food, the predators tend to die with exponential decay $y' = -\beta y$.

In this idealized model, it can be shown [HW95] that trajectories are level curves of the function

$$F(x,y) = |x|^{\beta}|y|^{\alpha}e^{-\delta x-\gamma y}$$

i.e. they satisfy the condition $F(x,y) = k$, for some constant $k \in \mathbb{R}$. The trajectories are as depicted in Fig. 1.1. Thus, the system (1.2) has an infinite number of trajectories and all of them are periodic, except for the point $(\beta/\delta, \alpha/\gamma)$ which is itself an equilibrium. As a curiosity, we remark that d'Ancona's problem can be explained in the following manner within this model. Fishing can be interpreted as reducing the value of α and as increasing β (both prey fish and predators are catched in fishing activities). This will shift the previous equilibrium point $(\beta/\delta, \alpha/\gamma)$ to a new point with increased value of x and decreased value of y, thus explaining the paradox.

This behavior has been observed in other situations, thus confirming the utility of the information taken from (1.2). As an example [HW95], in 1868 some acacia trees were imported from Australia to California, carrying with them some parasite insects that would nearly wipe out the

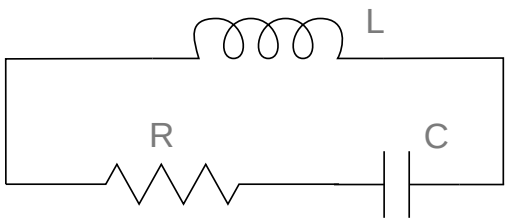


Figure 1.2: A RLC circuit.

citrus industry in California. To solve the problem, some predators of the parasite were brought from Australia. Although with a significant initial success, soon after the relation predator-prey reached an equilibrium, not completely solving the farmers' problem. Shortly before World War II, DDT was discovered and used as a panacea for the problem. But, for the farmers' surprise, the parasite insects became more numerous!

Electronics: the Van der Pol equations

In this section we study Van der Pol's equation. The original application described by Van der Pol was to an electric circuit constituted by a triode valve (nowadays it would be replaced by a transistor), the resistive properties of which change with current. We refer the reader to [GH83] for further details and references about this equation. Instead, we pick an example given in [HSD04], motivated by the RLC circuit of Fig. 1.2. It has one resistor (R), one capacitor (C), and one inductor (L). Its electric state depends of six quantities: the intensity of the current flowing to each component R, L, and C, as well as the voltage drop between the extremities of R, L, and C. Assuming that the inductor and capacitor behave in an ideal way, and considering Kirchhoff's laws, it is shown in [HSD04] that we only need two variables to fully describe the system, namely the current x flowing over L and the voltage drop y over the extremities of the capacitor. Their evolution along time is described by the following system

$$\begin{aligned} x' &= y - f(x) \\ y' &= -x \end{aligned} \tag{1.3}$$

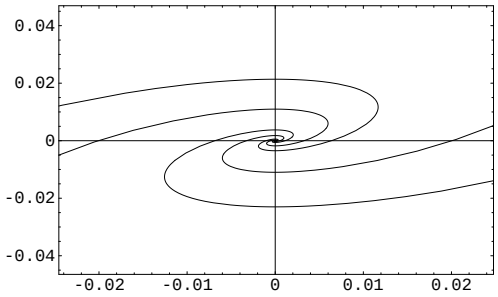
where $f : \mathbb{R} \to \mathbb{R}$ is the dependence law of the voltage drop over R with the current passing through it (ideally $f(i_R) = Ki_R$, which is Ohm's law. However, if R is replaced by a semiconductor device e.g. a diode, this dependence is no longer linear).

In particular we pick $f(x) \equiv f_\mu(x) = x^3 - \mu x$. It can be shown that this system has the following behavior for $\mu \in [-1,1]$ (cf. Fig. 1.3):

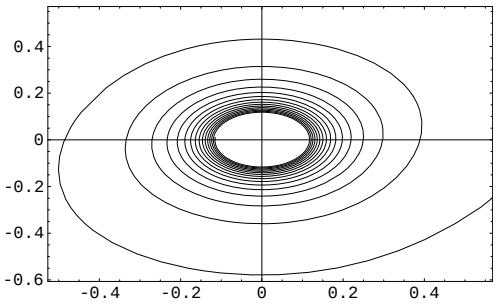
1. For $\mu \in [-1,0]$, all solutions tend to zero as $t \to \infty$ (i.e. the origin is a *sink*). In other words, all currents and voltages tend to zero;

2. For $\mu \in (0,1]$, the circuit radically changes its behavior: it starts oscillating. Now there is a unique periodic solution γ_μ, to which every nontrivial solution tends as $t \to \infty$. We also notice the existence of an unstable equilibrium at the origin: any point near the origin will be drawn away from it (in this case it is called a *source*, that corresponds to the dual notion of sink).

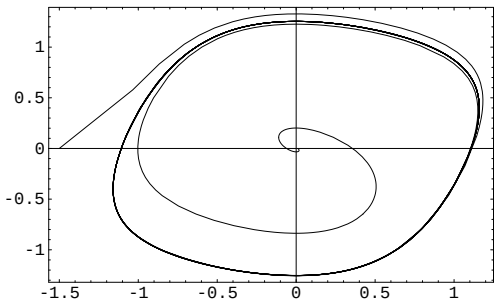
Therefore, the case $\mu = 0$ corresponds to a revolutionary value, where the behavior changes dramatically. It is an example of a *Hopf bifurcation* [HW95]. We notice that for this value of μ,



Before bifurcation: $\mu = -1$. The origin is a sink and all trajectories converge to it.



At bifurcation: $\mu = 0$. The origin is a weak sink (its linearization gives a center) and all trajectories converge very slowly to it.



After bifurcation: $\mu = 1$. The origin is now a source, and all other trajectories converge to a limit cycle.

Figure 1.3: Hopf bifurcation for system (1.3).

the system is not stable is the sense that small perturbations on this value change significantly the global dynamical behavior of the system.

Historically, the study of stable dynamical systems always had a prominent role in the development of the theory of dynamical systems. Since we cannot avoid uncertainty when performing measurements on a physical system, and we cannot fully isolate it, it seems that a good mathematical model for physical phenomena should account for some kind of robustness.

This is the idea underlying *structurally stable systems*, originated from the work of Andronov and Pontryagin [AP37]. A system is structurally stable if small perturbations of it leave the whole orbit structure unchanged, up to a continuous global change of coordinates. Andronov, Pontryagin, and Peixoto showed the following results for the plane $\mathbb{R}^2$:

(i) Structurally stable systems form an open and dense subset of the set of dynamical systems with C^1 vector fields;

(ii) Attractors for structurally stable systems consist only of fixed points and periodic orbits.

For some time it was conjectured (cf. the retrospective in [GH83]) that (i) and (ii) would hold for higher dimensions. Both conjectures would turn out to be false: not only structurally stable systems are not dense [Sma66], but they also allow other kinds of attractors.

For that reason structurally stable systems are no longer seen as the adequate class of "robust system". Moreover, this initiated a search for stable attractors other than fixed points and periodic orbits. The next application provides one of the best known examples of such attractors that, while being very robust to changes in the parameters, is not structurally stable [Via00].

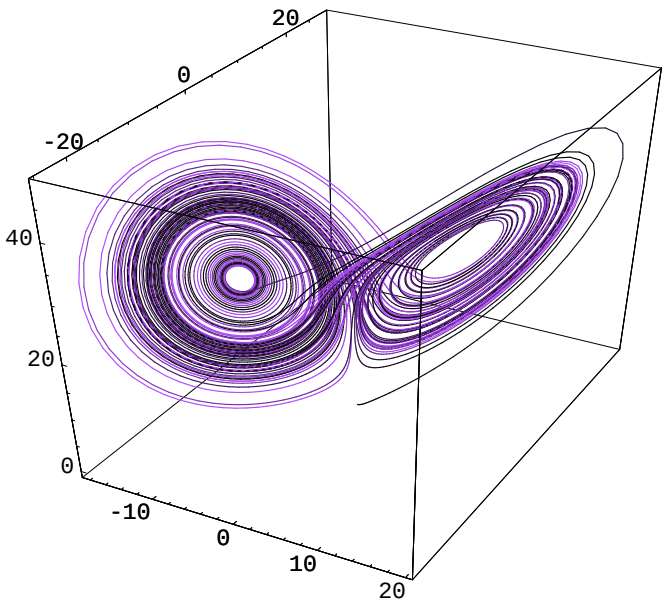


Figure 1.4: Numerical solution of Lorenz' equations for parameters $\sigma = 10, \beta = 8/3$, and $\rho = 28$, with initial solution $(0,1,0)$ set for $t_0 = 0$.

Meteorology: The Lorenz equations

In the early 1960s Lorenz, a meteorologist working at M.I.T. and interested in the foundations of long-range weather forecasting, was studying a very simplified model for atmospheric convection given by the system

$$\begin{aligned} x' &= \sigma(y-x) \\ y' &= \rho x - y - xz \\ z' &= xy - \beta z \end{aligned} \tag{1.4}$$

where σ (the Prandtl number), ρ (the Rayleigh number), and β (an aspect ratio) are positive real parameters. We refer the reader to [Sal62], [Lor63] for more physical details on this system. The values used by Lorenz and most other investigators are $\sigma = 10, \beta = 8/3$, and $\rho = 28$, though similar behavior occurs for other values [GH83]. Lorenz discovered that trajectories rapidly approached and then followed a butterfly-shaped trajectory like the one depicted in Fig. 1.4. This would be one of the very first examples of *strange attractors* with underlying physical motivation. While there is no precise definition for this mathematical object (there are a number of different definitions in the literature, see e.g. [Mil85] and [Rue89] and references therein), it has at least two properties: (i) it is robust and nearby trajectories converge to it; (ii) this is not a "simple" attractor (i.e. a point or a periodic orbit). We note that while numerical experiments furnished strong evidence for the existence of a strange attractor in Lorenz' system, despite a huge amount of theoretical work on the system (even a book [Spa82] was written on Lorenz' system), it was only very recently [Tuc98], [Tuc99] that its existence and robustness were proved, showing the inherent difficulties in analyzing this kind of systems. We remark that this is a computer-aided proof, with all the inherent objections it might raise [Via00], namely the possible existence of errors in the computation that are beyond human detection.

1.3 A computational perspective

As we mentioned earlier, there is already a significant amount of research devoted to the study of dynamical systems from a computational point of view. This section describes some of the results that can be found on the literature. This survey is far from complete, since it mainly focuses on aspects that will be relevant later in this thesis.

One of the perspectives existent in the literature is the following: given some dynamical system, can it simulate an universal Turing machine? Motivated by this issue, several authors have proved that relatively simple discrete-time systems can simulate Turing machines. The general approach is to associate each configuration of a Turing machine to a point of $\mathbb{R}^n$, and to show that there is a dynamical system with state space in $\mathbb{R}^n$ which embeds its evolution.

For example, it is known that Turing machines can be simulated on compact spaces, even of low dimension [Moo90], [KCG94], [SS95]. While compactness is a desirable property of dynamical systems, it is probably too strong a requirement since it is believed that no analytic map on a compact, finite dimensional space can simulate a Turing machine through a reasonable encoding [Moo98].

The requirement of compactness has another drawback: it prevents systems capable of simulating an arbitrary Turing machine of exhibiting robustness to noise. Indeed, Casey [Cas96], [Cas98] has shown that in the presence of bounded analog noise, recurrent neural networks can only recognize regular languages. This result was later generalized in [MO98] to other discrete-time computational systems. Robustness is a critical issue in continuous models since noncomputable behavior might arise when the use of exact real quantities is allowed. For instance, it is known that recurrent analog neural networks can present noncomputable behavior if real parameters are represented with infinite precision [SS95].

Another interesting result on the simulation of Turing machines by dynamical systems can be found in [KM99]. There it is shown that elementary maps can simulate Turing machines in an unbounded space. However, the effect of perturbations on the simulation is not studied (this will be one of the issues addressed to in this thesis).

The previously mentioned results show that finite dimensional maps are capable of simulating the *transition function* of an arbitrary Turing machine. In that respect, those are results about the computational power of hybrid systems, which are continuous with respect to the state space but evolve discretely in time. Another perspective has been to simulate the evolution of Turing machines with continuous flows in $\mathbb{R}^n$ [Bra95], [CMC02], [MC04]. However all these flows are non-analytic and, at most, infinitely differentiable. In the present work, we will show that these flows can be analytic.

Another natural approach to dynamical systems is to relate them with computable analysis [PER89], [Ko91], [Wei00]. In particular, within this approach, an interesting question is the following: given a dynamical system defined with computable data, can it have uncomputable properties? In [PER79], Pour-El and Richards showed that the IVP

$$\begin{cases} x' = f(t,x) \\ x(t_0) = x_0 \end{cases}$$

defined with computable data may have noncomputable solutions. In [PER81], [PEZ97] it is shown that there is a three-dimensional wave equation, defined with computable data, such that the unique solution is nowhere computable. However, in these examples, noncomputability is not genuine in the sense that the problems under study are ill-posed: either the solution is not unique or it is unstable [WZ02]. In other words, ill-posedness was at the origin of noncomputability in those examples. Nonetheless, we will show in this dissertation that even well-posed initial-value problems can have noncomputable properties, thus suggesting that the noncomputability results are not inherited from bad properties of the problems. For reference we also

mention the existence of other results about computability of ODEs that can be found in [Abe70], [Abe71], [BB85], [Hau85], [DL89], [Ko91], [Ruo96].

A third perspective of interest for theoretical computer science is to relate dynamical systems with models of computations over the reals, not necessarily based on Turing machines. In particular, we will be interested in this thesis on Shannon's General Purpose Computer (GPAC) [Sha41], which is discussed in Sections 3.3 and 3.4. We can relate them to a particular class of dynamical systems, namely the ones defined as solutions of ODEs $y' = p(t, y)$.

We should also mention that connections exist between the GPAC and other models of computation over the reals. In [GC03], it is shown that the GPAC and $\mathbb{R}$-recursive functions [Moo96] defined with a particular form of integration coincide. Moreover, the $\mathbb{R}$-recursive functions are shown to be equivalent to computable analysis if a different form of integration is used, as well a new zero-finding schema [BH04]. This work was restricted to lower subclasses [BH05], by substituting the zero-finding schema to a limit schema, and largely follows work from [Moo96], [Cam02], [CMC02], [MC04]. In this thesis we will show that the GPAC, under a certain framework, and computable analysis are equivalent models of computation.

The investigations of the relations between dynamics and computations attracted the attention of several research communities. One of them is highly motivated by the question of computer aided verification, and in particular by the question of computer aided verification of hybrid systems [AD90], [AMP95], [AD00], [HKPV98], [PV94], [NOSS93], [BT00], [Cv04], [AP04]. Hybrid systems combine both features from discrete and continuous systems. For instance, a hybrid system may have a state space which has continuous and discrete components.

The idea underlying computer aided verification of hybrid systems is to get some "as automatic as possible" computer system, that would take as input the description of a hybrid system, and the description of some property, call it "safety", and that would tell whether the system satisfies it or not. In this sense, one is led to make a computational reasoning about a system that has a continuous component.

Although some positive results exist along this line of research (e.g. it can be shown that for some subclasses of timed automata [AD90], or for hybrid systems with linear vector fields at each discrete location [LPY99], the reachability property is decidable), most of the results are negative. For instance, very simple classes of linear hybrid automata [HKPV98] or piecewise constant derivative systems [AMP95] have been shown to be able to simulate arbitrary Turing machines. As a consequence, verification procedures are semi-decision procedures and not decision procedures.

However, there is much ongoing research on this subject. Indeed, there are important issues that still need to be clarified. One of them is related to the "robustness to perturbations". The proofs of undecidability for properties of hybrid systems, or more generally of simulation of Turing machines, often involve the coding of the configuration of a Turing machine into some real numbers, and this requires infinite precision.

A pertinent question is to know what happens when we have to take into account some kind of "robustness to perturbations", as in real world applications. Some authors argue that undecidability results can no longer be obtained in this case [AB01]. There were several attempts to formalize and prove (or to disprove) this conjecture: it has been proved that small perturbations of the trajectory still yields undecidability [HR99]. Infinitesimal perturbations of the dynamics for a certain model of hybrid systems has shown to give rise to decidability [Frä99]. This has been extended to several models by [AB01]. However, this question is far from being settled.

In the present work, and following partially the previous line of work, we will show that one can perform robust simulations of Turing machines with continuous dynamical systems, in a particular framework.

1.4 Our contributions

In this thesis we will be concerned with the study of computational properties for systems of the type (1.1). In particular, our main contributions are the following:

1. Maps that are solutions of (1.1) can simulate Turing machines, in a robust manner, within discrete-time, i.e. by iterating the map.

2. Each Turing machine can be simulated by a system (1.1), where the input to the Turing machine sets the initial condition. We show that the simulation is still possible even if the initial condition is perturbed. In other words, systems like (1.1) are Turing universal and therefore achieve Type-1 computability.

3. We will show that, under an appropriate notion of computability, systems like (1.1) are equivalent to computable analysis, i.e. Type-2 computability.

4. We study some of the limitations when computing information about systems like (1.1). More generally, we will be concerned with initial-value problems (IVPs) defined with analytic systems $y' = f(t,y)$, where f stands for an analytic function over $\mathbb{R}^n$. We show that: (i) if these IVPs are defined with computable data, their respective maximal interval of existence for the solution can be noncomputable, but it is always recursively enumerable; (ii) the problem of deciding whether the previous maximal interval is bounded or not is undecidable, even if we restrict ourselves to the case (1.1) of polynomial IVPs.

1.5 Overview of the dissertation

The contents of the present dissertation can be summarized as follows. In Chapter 2 we review some of the basic theory we will use throughout this work. In particular, we will set some basic notation (Section 2.2) and recall some notions and results from computability theory (Section 2.3), recursive analysis theory (Section 2.4), analysis (Section 2.5), and dynamical systems theory (Section 2.6).

In Chapter 3 we recall some previous results about polynomial initial-value problems (1.1). Section 3.2 reviews some material about polynomial differential equations, namely the notion of differentially algebraic functions, that we then restrict to the case (1.1). Next we prove that the class of solutions of (1.1) is closed under several operations. Moreover, we also show (Theorem 3.2.5) that many systems of ordinary differential equations that are not written in terms of polynomials, but rather with functions involving the composition of trigonometric functions, exponentials, etc., are still equivalent to a system (1.1), thus providing an argument for studying systems like (1.1). In Section 3.3 we review a model of analog computation, the General Purpose Analog Computer, and we then study its properties in Section 3.4. In particular, we recall that within the approach of [GC03], this model is equivalent to the class defined by (1.1).

Chapter 4 focuses on the simulation of Turing machines with solutions of (1.1). Section 4.2 sets the coding of configurations to be used and, as well as Section 4.3, presents some useful tools. Then Section 4.4 shows how we can iterate solutions of (1.1) to simulate Turing machines. This simulation is robust in the sense that even if one adds some perturbation to the system in each iteration, it will still be able to accurately perform the simulation. In Section 4.5 we review some results on iterating maps with non-analytic ODEs. We then extend this construction to analytic ODEs in Section 4.6, but only for maps that are robust to perturbations around integers. This will be the case of the maps already introduced to simulate Turing machines. In that way, we show that each Turing machine can be simulated by a system of the form (1.1).

In Chapter 5 we show that computable analysis and systems like (1.1), equipped with a suitable notion of computability, are equivalent models of computation, on compact intervals. In Section 5.2, we present the motivations for this line of research, a new notion of computability for (1.1), and the main result about equivalence. This result is then proved in the following sections. Section 5.3 proves the "if" direction of the result, while Section 5.5 shows the converse relation. The latter is preceded by Section 5.4, which includes auxiliary results.

Finally Chapter 6 studies the problem of determining the maximal interval of existence for the solution of

$$\begin{cases} x' = f(t,x) \\ x(t_0) = x_0 \end{cases} \tag{1.5}$$

from a computational point of view. Section 6.2 shows that, under very mild assumptions, if f, t_0, and x_0 are computable then the maximal interval is recursively enumerable, thus providing an upper bound for the resources needed to compute it. Notwithstanding, we show in Section 6.3 that in the previous conditions, the maximal interval needs not to be recursive. We show in Section 6.4 that this result continues to hold even if f is restricted to the well-behaved class of analytic functions. In Section 6.5 we address the following problem: while for analytic functions it is not possible to compute the maximal interval from the initial data in (1.5), perhaps it could be possible to extract partial information, namely to know whether the maximal interval is bounded or not. We prove that this problem is undecidable. In Section 6.6, this result in sharpened for polynomial ODEs. More specifically, we show that the problem of deciding whether the maximal interval is bounded or not is undecidable for polynomial IVPs of degree 56.

We end with a list of open questions which are suggested by the results of this dissertation. The work done in this thesis appears partially in the following references: [Gra04], [GCB05], [BCGH06], [BCGH06], [GZB06], [GZB07], and [GCB06]. It also reviews material obtained by author during its MSc thesis: [Gra02], [GC03].

CHAPTER 2

Preliminaries

2.1 Introduction

This chapter introduces basic notation and recalls results that will be used throughout this work. In Section 2.2, we review mathematical notions involving functions and strings. In Section 2.3, we recall some fundamental results of the theory of computation. This section is intended to be as much as possible self-contained, and states all the major definitions and results used in this thesis. Its contents includes some fundamental definitions like Turing machines, recursive functions and sets, recursively enumerable sets, and covers results like the undecidability of the Halting Problem. This section is followed by Section 2.4 that covers material about computable analysis (this is an extension of the classical theory of computability to the Euclidean space $\mathbb{R}^n$). Section 2.5 reviews the notions of analytic and elementary functions (not to be confused with the homonymous class defined in theoretical computer science [Odi99]), and some of their properties.

Finally the chapter ends with Section 2.6 that reviews the existence-uniqueness theory for initial-value problems defined with ordinary differential equations.

2.2 Basic mathematical notation

In this section, for convenience of the reader, we summarize some of the basic terminology that we shall use throughout this thesis.

An *alphabet* Σ is a non-empty finite set. A *symbol* is an element of Σ and a *string* is a finite sequence of symbols from Σ. The *empty word*, denoted as ϵ, is the unique string consisting of zero symbols. The set of all strings over Σ will be denoted as Σ^*. If one has a total order over Σ, one can associate a (total) *lexicographic order* $<_*$ over Σ^* as follows. Let $w = a_1...a_m$ and $v = b_1...b_n$ be strings over Σ^*, where $a_1,...,a_m,b_1,...,b_n \in \Sigma$. Then $w <_* v$ if two situations occur: (i) $m < n$ or (ii) $m = n$ and there is a j, with $1 \leq j \leq m$, such that for all $i < j$, $a_i = b_i$, but $a_j < b_j$. As an example, if one has $\Sigma = \{a,b\}$, with the order $<$ defined by $a < b$, one gets

$$\epsilon <_* a <_* b <_* aa <_* ab <_* ba <_* \ldots.$$

By $\mathbb{N}$ we denote the set of non-negative integers $\{0,1,2,...\}$, as usual in computability theory. By $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ we denote the sets of integer, rational, real, and complex numbers, respectively. The notation $\mathbb{R}^+$ is used for the set $(0,+\infty)$.

We will frequently have to deal with functions that are not defined for all the values of the arguments. Therefore, given a function $f : A \to B$ we will say that this function is *partial* if it is not defined for all elements of A and *total* otherwise. Also, we call a function $f : A^n \to B^m$ with n arguments an *n-ary function*. For the special case $n = 1$, the function is called *unary*, and for the case $n = 2$, it is called *binary*. Given a function $f : A^n \to B^m$, one can easily define m functions $f_i : A^n \to B$, with $1 \leq i \leq m$ such that $f(x) = (f_1(x), \ldots, f_m(x))$. These functions are called the *components* of f. Sometimes, we will need to *iterate* a function $f : A \to A$. We define the kth iterate of f, where $k \in \mathbb{N}$, as being the function $f^{[k]} : A \to A$ defined recursively in the following manner: $f^{[0]} = id$ and $f^{[k+1]} = f \circ f^{[k]}$, where id denotes the identity function.

When considering real functions, we shall say that a function $f : E \to \mathbb{R}^m$ is of class C^k on the open set $E \subseteq \mathbb{R}^l$ if the derivatives of f up to order k are all defined and continuous over E. The class of all C^k functions over E is denoted by $C^k(E)$. We also define the class of C^∞ functions over E, $C^\infty(E)$, as

$$C^\infty(E) = \bigcap_{k=0}^{\infty} C^k(E).$$

Obviously, to differentiate, we need to assume that to the space $\mathbb{R}^n$ is associated some norm $\|\cdot\|$ that by its turn generates a topology over $\mathbb{R}^n$. The choice of the norm is irrelevant since for $\mathbb{R}^n$, where $n \in \mathbb{N}$, all norms are equivalent [Lan05] and hence generate the same topology. However, for practical purposes, we will assume in this work that we are using the norm defined by

$$\|(x_1, \ldots, x_n)\|_\infty = \max_{1 \leq i \leq n} |x_i|$$

for all $(x_1, \ldots, x_n) \in \mathbb{R}^n$. The corresponding topology is generated by the class of open balls $B(a, r)$, where $a \in \mathbb{R}^n$, $r > 0$, and

$$B(a, r) = \{x \in \mathbb{R}^n : \|x - a\| < r\}.$$

The closure of a set $A \subseteq \mathbb{R}^n$ will be denoted by $\overline{A}$.

2.3 Classical computability

This section reviews material from classical computability theory. One of the most important ideas underlying this theory is the notion of *algorithm*. While algorithms have appeared for a long time in mathematics (e.g. recall Euclides' algorithm to find the greatest common divisor of two integers), it was only on the 1930s that this notion was formalized.

This remarkable breakthrough was achieved by logicians such as Kleene, Church, and Turing, and was aimed to solve some open problems of logic and mathematics. As an example let us mention Hilbert's 10th problem, proposed by D. Hilbert in 1900, in his now famous lecture at the International Congress of Mathematics in Paris: is there any algorithmic procedure that tells us whether a polynomial with integer coefficients has an integer root? While a positive answer could easily be checked, a negative one would present a major problem, at least without a proper definition of algorithm.

In the 1930s, Kleene, Church, and Turing defined the recursive functions, λ-calculus, and Turing machines respectively, to formalize the intuitive notion of algorithm. It was soon discovered that all these approaches were equivalent, as well as new definitions that would later appear. This led to the following conjecture:

Church-Turing Thesis: A function can be computed by an algorithm if and only if it can be computed by a Turing machine.

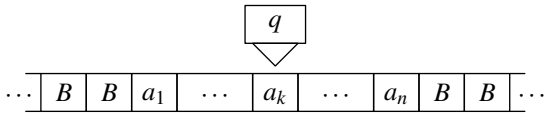


Figure 2.1: A Turing machine

Notice that the previous statement cannot be proved, since we don't have a precise definition for algorithm. Indeed, this thesis is intended to fix the meaning of algorithm and is accepted by the scientific community as correct.

Before giving the formal definition of a Turing machine (TM for short), let us briefly describe how it works. A TM consists of three elements (cf. Fig. 2.1): a tape divided into an infinite number of cells, a control unit that may be in any of a finite set of states, and a tape head that scans the symbol at one of the tape cells. Each cell of the tape contains one of a finite number of symbols. Then, depending on the current symbol being scanned by the tape head and on the current state of the control unit, the TM performs several steps, according to the definition below.

Definition 2.3.1. *A Turing machine is a 7-tuple $(Q,\Gamma,\Sigma,\delta,q_0,B,F)$, where Q and Γ are finite sets and*

1. *Q is the set of states,*
2. *Γ is the tape alphabet, where $B \in \Gamma$ and $\Sigma \subseteq \Gamma$,*
3. *Σ is the input alphabet,*
4. *$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L,N,R\}$ is the transition function,*
5. *$q_0 \in Q$ is the start state,*
6. *$B \in \Gamma$ is the blank symbol, where $B \notin \Sigma$,*
7. *$F \subseteq Q$ is the set of final states.*

A TM $M = (Q,\Gamma,\Sigma,\delta,q_0,B,F)$ computes as follows. Initially M has its tape completely filled with blank symbols. Then it receives its input $w = w_1 w_2 \dots w_n \in \Sigma^*$, writes it on the tape in n consecutive cells (maintaining the order of the symbols), and places the tape head on the leftmost cell that holds the input. The initial state of M is the start state q_0. Once M starts, the computation proceeds according to the rules described by the transition function. In particular, if M is currently on state q and reading symbol s, and if $\delta(q,s) = (q_{next}, s_{next}, move)$, then M will update its state to q_{next}, change the symbol being read by the tape head to s_{next}, and move the tape head according to the value of *move* (where L, N, R correspond to a left move, no move, and right move, respectively).

The computation of M continues until it reaches a final state at which point it halts. Then, if at this point the tape content is

$$\dots B v_1 v_2 \dots v_k B \dots$$

where $v_1 v_2 \dots v_k \in \Sigma^*$ and the tape head is over v_1, then the output will be the string $v_1 v_2 \dots v_k$. If M does not halt, then the output will be undefined. In this sense we have defined a (partial) computable function $f : \Sigma^* \to \Sigma^*$. Later, when describing Turing machines, we will use some

pseudo-algorithm instead of the full formalism of Definition 2.3.1. The Church-Turing thesis guarantees the equivalence of both approaches.

Notice that in each point of the computation, only a finite number of symbols is non-blank. Therefore, if some Turing machine M at a given instant is in the situation pictured in Fig. 2.1, where all symbols to the left of a_1 and to the right of a_n are blanks, then all the relevant data for the computation at that instant can be recorded into a triple

$$(a_1 \dots a_k, a_{k+1} \dots a_n, q) \in \Gamma^* \times \Gamma^* \times Q.$$

This triple is called the *configuration* of the Turing machine.

Since Σ is a finite set, we can define a one-to-one correspondence between $\mathbb{N}$ and Σ^*, by using the lexicographic order. For instance, if $\Sigma = \{a, b\}$, then we can set the following correspondence: $\epsilon \to 0$, $a \to 1$, $b \to 2$, $aa \to 3$, $\dots$. Therefore, without loss of generality, we can consider computability over integers instead of over strings. Moreover, if we add a new symbol $\Diamond$ to the input alphabet Σ, to delimitate strings of Σ, one can define computable functions with several arguments over Σ^* and hence over $\mathbb{N}$.

Definition 2.3.2. *A (partial) function $f : \mathbb{N}^n \to \mathbb{N}^k$ is computable (or recursive) if it can be computed by a Turing machine.*

We notice that there are computable bijective functions from $\mathbb{N}^n$ to $\mathbb{N}$, for all $n \geq 1$, such that their inverses are also computable. As an example [Odi89], consider the following pairing function $\langle \cdot, \cdot \rangle : \mathbb{N}^2 \to \mathbb{N}$ defined by

$$\langle x, y \rangle = \frac{(x+y)^2 + 3x + y}{2}. \tag{2.1}$$

This function is bijective (this can be seen by using a Cantor enumeration of $\mathbb{N}^2$) and computable. Its inverses $\pi_1, \pi_2 : \mathbb{N} \to \mathbb{N}$ defined by

$$\pi_1(\langle x, y \rangle) = x \quad \text{and} \quad \pi_2(\langle x, y \rangle) = y \tag{2.2}$$

can also be shown to be bijective and computable. Provided with this function, we can easily define computable and bijective pairing functions $\langle \cdot, \dots, \cdot \rangle : \mathbb{N}^n \to \mathbb{N}$, with computable inverses $\pi_1, \dots, \pi_n : \mathbb{N} \to \mathbb{N}$, for all $n \geq 1$. These functions appear quite often in computability, and this work will not be an exception. Next, we define computability notions for sets.

Definition 2.3.3. *A set $A \subseteq \mathbb{N}^n$ is* recursive *if there is a total computable function $\chi_A : \mathbb{N}^n \to \{0, 1\}$ such that*

$$\chi_A(x) = \begin{cases} 0 & \text{if } x \notin A \\ 1 & \text{if } x \in A. \end{cases}$$

The function χ_A is called the characteristic function *of A.*

Definition 2.3.4. *The problem "$x \in A$?" where $A \subseteq \mathbb{N}^n$, is called* decidable *if the set A is recursive.*

We should point out that mathematical problems involving natural numbers can usually be rephrased as a problem of the kind "$x \in A$?". For instance, the problem: "Given a natural number $x \in \mathbb{N}$, is x a prime number?" can be rephrased into "$x \in PRIMES$?", where

$$PRIMES = \{x \in \mathbb{N} : x \text{ is a prime number}\}.$$

One of the great achievements of computability theory was to show the existence of *undecidable* problems. Among these, of particular importance is the Halting Problem. To describe it, we first need the notion of *universal TM*. According to Def. 2.3.1, any TM can be described as a finite 7-tuple, and hence as a string. Therefore, we can put each TM M into correspondence to a number $f(M) \in \mathbb{N}$ (remark that to different TMs correspond different numbers). Reciprocally, we can put each element of $\mathbb{N}$ in correspondence with a TM (to $x \in \mathbb{N}$, associate the TM $f^{-1}(x)$. If $f^{-1}(x)$ is not defined, then associate x to some predefined TM M_0).

A universal TM is a TM M with two inputs, that on inputs $x, y \in \mathbb{N}$ outputs the result of the TM x, when it computes with input y. The proof of the existence of such universal TMs can be found in any standard book of computability theory, e.g. [Sip97], [HMU01]. In practice, one can see a standard computer (with infinite memory) as a universal TM. The first argument corresponds to the "program" in use (each time we want to perform a new task, we don't want to switch computer, i.e. we would like to continue to use the same TM), and the second argument is the input to the program. Hence a universal TM is capable of simulating any other TM from the description of that machine.

Definition 2.3.5. *Let M be an universal TM. The problem "$(x,y) \in HALT$?" where*

$$HALT = \{(x,y) \in \mathbb{N} : M \text{ halts in a finite number of steps for input } (x,y)\},$$

is called the Halting problem.

Proposition 2.3.6. *The Halting problem is undecidable.*

The proof of the previous proposition can be found in [Sip97]. For instance, an application of this proposition would be the following: given a C/C++ program x and some input y, there is no algorithmic way to tell if the program x will "crash" (i.e. enter in an infinite computation) on input y. This corresponds to the idea that programming environments can catch syntactic errors of programs, but can never catch all the semantic errors (at least for general purpose programming languages). Another example of an undecidable problems is Hilbert's 10th problem [Mat70], [Mat72], [Dav73].

Of course, in computability theory we do not want to end the classification of computational problems with the general class of undecidable problems. Instead, we would also like to establish subclasses inside this class. This is the motivation for what follows.

Definition 2.3.7. *A set $A \subseteq \mathbb{N}^n$ is called recursively enumerable (r.e. for short) if $A = \emptyset$ or if there is a total computable function $f : \mathbb{N} \to \mathbb{N}^n$ such that*

$$A = \{f(x) \in \mathbb{N}^n : x \in \mathbb{N}\}.$$

Notice that every recursive set is r.e., but the converse relation is not true as the following results show [Odi89].

Proposition 2.3.8. *A set $A \subseteq \mathbb{N}^n$ is recursive iff A and $\mathbb{N}^n \backslash A$ are r.e.*

Proposition 2.3.9. *Let M be an universal TM. Then the set*

$$\mathcal{K} = \{x \in \mathbb{N} : M \text{ halts on input } (x,x)\}$$

is r.e. but not recursive.

The last result implies, in particular (cf. Prop. 2.3.8), that $\mathbb{N} \backslash \mathcal{K}$ is a set that is not even r.e. The following result (cf. [Odi89]) will be used in the next section and in Chapter 6.

Proposition 2.3.10. *If $A \subseteq \mathbb{N}^n$ is an non-empty r.e. set that is not recursive, then there is a one to one recursive function $f : \mathbb{N} \to \mathbb{N}^n$ such that*

$$A = \{f(x) \in \mathbb{N}^n : x \in \mathbb{N}\}.$$

Because not all sets are recursive or r.e., we would like to know "how much power" we must add to a TM so that it may compute a given set. This definition is formalized by the concept of *oracle Turing machine*. Here we will use oracle TMs, but for a different purpose, as we will see in the next section. A (function) oracle TM is an ordinary TM equipped with an extra tape, the query tape. The machine also has two extra states, called the query state and the answer state. An oracle TM M has two inputs: a string that it is written in the tape as usual, and an oracle that is simply a total function $\phi : \mathbb{N}^n \to \mathbb{N}^k$. This includes the case when we have n oracles $\phi_1, \ldots, \phi_n : \mathbb{N} \to \mathbb{N}$, by considering $\phi = (\phi_1, \ldots, \phi_n)$. The oracle TM computes as an usual TM, except when it enters the query state. At that moment it reads the string t on the query tape, replaces it by $\phi(t)$ and puts the tape head of the query tape scanning the leftmost symbol of the string $\phi(t)$. Then it puts the machine into the answer state and the machine continues as usual. We also suppose that each query takes only 1 step to perform and that the machine cannot enter the answer state, unless after some query.

2.4 Computable Analysis

The previous section dealt with computability over integers, which is also known as Type-1 computability. Here we will study computability over reals. This corresponds to Type-2 computability, because a real number can be considered as a function $f : \mathbb{N} \to \mathbb{N}$ as we will see (computability over real functions would correspond to Type-3 computability, and so on).

As we mentioned in the previous chapter, there is no unified approach to computation over the reals, opposite to what happens in the discrete case. However, if we restrict ourselves to real computation that can only be performed by Turing machines over partial and finite information about real numbers, we get a rather coherent theory that is known as computable analysis (or recursive analysis). This theory was introduced by Turing [Tur36], Lacombe [Lac55], and Grzegorczyk [Grz57], and has received contributions from many other authors.

The idea underlying computable analysis is to extend the classical computability theory so that it might deal with real quantities. See [Wei00] for an up-to-date monograph on computable analysis from the computability point of view, [Ko91] for a presentation from a complexity point of view, or [PER89] for a good introduction to the subject.

Definition 2.4.1. *A sequence $\{r_n\}$ of rational numbers is called a ρ-name of a real number x if there exist three functions a, b, c from $\mathbb{N}$ to $\mathbb{N}$, such that for all $n \in \mathbb{N}$, $r_n = (-1)^{a(n)} \frac{b(n)}{c(n)+1}$ and*

$$|r_n - x| \leq \frac{1}{2^n}. \tag{2.3}$$

In the conditions of the previous definition, we say that the ρ-name $\{r_n\}$ is given as an oracle to an oracle Turing machine, if the oracle to be used is (a, b, c). The notion of the ρ-name can be extended to $\mathbb{R}^l$: a sequence $\{(r_{1n}, r_{2n}, \ldots, r_{ln})\}_{n \in \mathbb{N}}$ of rational vectors is called a ρ-name of $x = (x_1, x_2, \ldots, x_l) \in \mathbb{R}^l$ if $\{r_{jn}\}_{n \in \mathbb{N}}$ is a ρ-name of x_j, $1 \leq j \leq l$.

Definition 2.4.2. *1. A real number x is called computable if a, b, and c in (2.3) are computable (recursive) functions.*

2. *A sequence $\{x_k\}_{k\in\mathbb{N}}$ of real numbers is computable if there are three computable functions a,b,c from $\mathbb{N}^2$ to $\mathbb{N}$ such that, for all $k,n\in\mathbb{N}$,*

$$\left|(-1)^{a(k,n)}\frac{b(k,n)}{c(k,n)+1}-x_k\right|\leq\frac{1}{2^n}.$$

Similarly, one can define computable points and sequences over $\mathbb{R}^l$, $l>1$, by assuming that each component is computable.

The next result can be found in Section 0.2 of [PER89] and will be exploited in Chapter 6.

Proposition 2.4.3. *Let $a:\mathbb{N}\to\mathbb{N}$ be a one to one recursive function generating a recursively enumerable nonrecursive set. Then the series*

$$\sum_{i=0}^{\infty}2^{-a(i)}$$

converges to a noncomputable real.

Notice that functions $a:\mathbb{N}\to\mathbb{N}$ in the conditions of the previous proposition do exist. This follows as a corollary of Proposition 2.3.9 and Proposition 2.3.10.

Now we introduce computability notions for sets in $\mathbb{R}^l$ (cf. [Wei00]). Similarly to the definition of r.e. sets over $\mathbb{N}$, one would like that r.e. sets over $\mathbb{R}$ might be "enumerated". The problem is that $\mathbb{R}$ has a non-countable number of elements. But this problem can be avoided by considering only topologically relevant sets.

Definition 2.4.4. *An open set $E\subseteq\mathbb{R}^l$ is called recursively enumerable (r.e. for short) open if there are computable sequences $\{a_n\}$ and $\{r_n\}$, $a_n\in E$ and $r_n\in\mathbb{Q}$ such that*

$$E=\cup_{n=0}^{\infty}B(a_n,r_n).$$

We now prove a lemma about r.e. sets that we will often use implicitly in Chapter 6.

Lemma 2.4.5. *Suppose that $E\subseteq\mathbb{R}^l$ is a r.e. set satisfying the conditions of the definition above. Then we may assume that $\overline{B(a_n,r_n)}\subseteq E$ for all n.*

Proof. Define the following computable (double) sequence of rationals

$$r_{n,k}=\max\{0,r_n-\frac{1}{k}\}.$$

Of course, $\overline{B(a_n,r_{n,k})}\subseteq B(a_n,r_n)\subseteq E$, and $\cup_{k=0}^{\infty}B(a_n,r_{n,k})=B(a_n,r_n)$. This implies that

$$E=\cup_{i=0}^{\infty}B(a_{\pi_1(i)},r_{\pi_1(i),\pi_2(i)}),$$

where π_1 and π_2 are given by (2.2). □

Next, we define closed r.e. subsets of $\mathbb{R}^l$. Note that we cannot simply take the union of closed balls to define a closed set since, in general, an infinite union of closed sets may not be a closed set. Instead, and noting that for any closed sets $A,B\subseteq\mathbb{R}^l$ [Wei00, Exercise 5.1.1]

$$\begin{aligned}\{B(a,r):B(a,r)\cap A\neq\varnothing\text{ and }a\in\mathbb{Q}^l,b\in\mathbb{Q}\}=\\=\{B(a,r):B(a,r)\cap B\neq\varnothing\text{ and }a\in\mathbb{Q}^l,b\in\mathbb{Q}\}\end{aligned}$$

implies $A=B$, we use the following definition.

Definition 2.4.6. *A closed subset $A \subseteq \mathbb{R}^l$ is called r.e. closed if there exist computable sequences $\{b_n\}$ and $\{s_n\}$, $b_n \in \mathbb{Q}^l$ and $s_n \in \mathbb{Q}$, such that $\{B(b_n, s_n)\}_{n\in\mathbb{N}}$ lists all rational open balls intersecting A.*

Just for reference, we would like to mention the following result [Wei00, Corollary 5.1.11] that matches quite closely Definition 2.3.7.

Proposition 2.4.7. *A closed set $A \subseteq \mathbb{R}^l$ is r.e. iff it is empty or has a dense computable sequence $\{x_i\}_{i\in\mathbb{N}}$.*

Finally, having defined r.e. open and r.e. closed subsets of $\mathbb{R}^l$, we can now define recursive sets, in a spirit that follows Prop. 2.3.8.

Definition 2.4.8. *An open (closed) set $A \subseteq \mathbb{R}^l$ is called recursive (or computable) if A is r.e. open (closed) and its complement $\mathbb{R}^l \backslash A$ is r.e. closed (open).*

As a natural consequence, we have the following result [Wei00, Example 5.1.17].

Proposition 2.4.9. *A real interval $(a,b) \subseteq \mathbb{R}$ is recursive iff a and b are computable.*

Finally, we are ready to introduce the notion of computable function. Informally, a function $f : \mathbb{R} \to \mathbb{R}$ is computable if there is a computer program that does the following. Let $x \in \mathbb{R}$ be an arbitrary element in the domain of f. Given an output precision 2^{-n}, the program has to compute a rational approximation of $f(x)$ with precision 2^{-n}. More precisely:

Definition 2.4.10. *Let $E \subseteq \mathbb{R}^l$ be an open or closed r.e. set.*

1. *A function $f : E \to \mathbb{R}^m$ is computable if there is an oracle Turing machine such that for any input $n \in \mathbb{N}$ (accuracy) and any ρ-name of $x \in E$ given as an oracle, the machine will output a rational vector r satisfying $\|r - f(x)\|_\infty \leq 2^{-n}$.*
2. *A sequence of functions $\{f_i\}_{i\in\mathbb{N}}$, where $f_i : E \to \mathbb{R}^m$ is computable if there is an oracle Turing machine such that for any input $n \in \mathbb{N}$ (accuracy), any $i \in \mathbb{N}$, and any ρ-name of $x \in E$ given as an oracle, the machine will output a rational vector r satisfying $\|r - f_i(x)\|_\infty \leq 2^{-n}$.*

The following result is an adaptation of Corollary 2.14 from [Ko91] and will be used in Chapter 5.

Proposition 2.4.11. *A real function $f : [a,b] \to \mathbb{R}$ is computable iff there exist three computable functions $m : \mathbb{N} \to \mathbb{N}$, sgn, $abs : \mathbb{N}^4 \to \mathbb{N}$ such that:*

1. *m is a modulus of continuity for f, i.e. for all $n \in \mathbb{N}$ and all $x, y \in [a,b]$, one has*

$$|x-y| \leq 2^{-m(n)} \implies |f(x) - f(y)| \leq 2^{-n}$$

2. *For all $(i,j,k) \in \mathbb{N}^3$ such that $(-1)^i \frac{j}{2^k} \in [a,b]$, and all $n \in \mathbb{N}$,*

$$\left|(-1)^{sgn(i,j,k,n)} \frac{abs(i,j,k,n)}{2^n} - f\left((-1)^i \frac{j}{2^k}\right)\right| \leq 2^{-n}.$$

In particular, this proposition implies the following corollary.

Corollary 2.4.12. *A computable function $f : [a,b] \to \mathbb{R}$ is continuous.*

2.5 Analytic and elementary functions

In this section we review the mathematical notions of analytic and elementary functions, as well as some of their properties.

Definition 2.5.1. *A function $f : \mathbb{C} \to \mathbb{C}$ is called analytic in $z_0 \in \mathbb{C}$ if there is some $r > 0$ such that $f(z_0)$ can be represented as*

$$f(z_0) = \sum_{n=0}^{\infty} a_n (z - z_0)^n \tag{2.4}$$

in the ball $B(z_0, r)$. We also say that f is analytic in the open set Ω if it is analytic for all points $z_0 \in \Omega$.

One should also remark that we can define analytic functions over $\mathbb{C}^k$. Using multi-index notation, a function $f : \mathbb{C}^k \to \mathbb{C}$ is analytic in $z_0 \in \mathbb{C}^k$ if it can be represented as

$$f(z) = \sum_{\alpha} a_\alpha (z - z_0)^\alpha,$$

where $\alpha = (\alpha_1, \ldots, \alpha_k) \in \mathbb{N}^k$. By Hartogs' Theorem [Gun90], [Kra01], $f : \mathbb{C}^k \to \mathbb{C}$ is analytic if and only if it is separately analytic in each component. However, we will mainly focus on properties of one variable analytic functions. The following result will be useful for this case.

Proposition 2.5.2. *For every power series $\sum_{n=0}^{\infty} a_n (z - z_0)^n$, there is some $0 \leq r \leq \infty$ (called the radius of convergence), given by*

$$r = \overline{\lim_{n \to \infty}} \frac{1}{\sqrt[n]{|a_n|}},$$

such that:

1. *The power series converges absolutely in the ball $B(z_0, r) = \{x \in \mathbb{C} : |x - z_0| < r\}$*
2. *The series is divergent in the set $\{x \in \mathbb{C} : |x - z_0| > r\}$.*

Although we have discussed complex analytic functions in general, we will be mainly interested in real analytic functions, that is, the restriction to the real axis of complex analytic functions (2.4) in which the coefficients a_n are real. The next results for real analytic functions follow from the corresponding results for non-isolated zeros of analytic functions [Ahl79], [MH98].

Proposition 2.5.3. *Let $f : I \to R$, where $I \subseteq \mathbb{R}$ is an open interval, be an analytic function. If there is a sequence $\{y_n\} \subseteq I$ converging to $y \in I$ such that $f(y_n) = 0$ for all $n \in \mathbb{N}$, then $f(x) = 0$ for all $x \in I$.*

Corollary 2.5.4. *Let $I \subseteq \mathbb{R}$ be an open interval.*

1. *Let $f : \mathbb{R} \to \mathbb{R}$ be an analytic function such that $f(x) = 0$ for all $x \in I$. Then $f(x) = 0$ for all $x \in \mathbb{R}$.*
2. *Let $f_1, f_2 : I \to \mathbb{R}$ be analytic functions. If there is an open interval $J \subseteq I$ such that $f_1 = f_2$ on J, then $f_1 = f_2$ on I.*

Next, we define the class of elementary functions (sometimes also known as *closed-form* functions). This class should not be confused with the homonymous class defined in theoretical computer science [Odi99]. The latter corresponds to a well-defined class of functions $\mathcal{E}$, defined over the naturals by using appropriate computational complexity bounds. Let us now give a preliminary definition.

Definition 2.5.5. *Let $f : \mathbb{C} \to \mathbb{C}$ be a function and $z_0 \in \mathbb{C}$. The point z_0 is an* isolated singularity *of f if there exists some $\varepsilon > 0$ such that f is analytic in $B(z_0, \varepsilon)\backslash\{z_0\}$, but not in z_0. The singularity is:*

1. removable *if* $\lim_{z\to z_0}(z-z_0)f(z) = 0$*;*
2. *a* pole of order *m if there is some* $a \in \mathbb{R}\backslash\{0\}$ *such that* $\lim_{z\to z_0}(z-z_0)^m f(z) = a$*;*
3. *an* essential singularity *if it is not a removable singularity nor a pole.*

Definition 2.5.6. *The function f is called a meromorphic function in the domain Ω if it is analytic there, except for isolated singularities that are poles.*

By a *domain* or *region* in $\mathbb{C}^n$ (respectively $\mathbb{R}^n$) we mean a nonempty connected open subset of $\mathbb{C}^n$ (respectively $\mathbb{R}^n$). We say that a function is *elementary* [Rit48], [Ros72] if it is meromorphic on some region in $\mathbb{R}$ or $\mathbb{C}$ and is contained in an elementary extension field of the field of rational function $\mathbb{C}(z)$, where "elementary" means that we allow the use of the complex exponential and logarithm. Equivalently, elementary functions on $\mathbb{R}$ are the ones obtained from the rational functions, sin, cos, exp, ..., through finitely many compositions and inversions.

2.6 Ordinary differential equations

Let $E \subseteq \mathbb{R}^{m+1}$ be an open set, $f : E \to \mathbb{R}^m$, $x = x(t) \in \mathbb{R}^m$ be a function of $t \in \mathbb{R}$, and x' denote the derivative of x with respect to t. Then an ordinary differential equation (ODE for short) is an equation of the type $x' = f(t,x)$. However, in general, we will be interested in initial-value problems (IVP for short) defined with ODEs, i.e. we will be interested in problems of the type

$$\begin{cases} x' = f(t,x) \\ x(t_0) = x_0 \end{cases} \tag{2.5}$$

where $t_0 \in \mathbb{R}$ and $x_0 \in E$.

We begin with some preliminary definitions and results.

Definition 2.6.1. *Let $E \subseteq \mathbb{R}^l$ be an open set. A function $f : E \to \mathbb{R}^m$ is called locally Lipschitz on E if for every compact set $\Lambda \subseteq E$ there is a constant $K_\Lambda \geq 0$ such that*

$$\|f(x) - f(y)\| \leq K_\Lambda \|x - y\|, \quad \textit{for all } x, y \in \Lambda.$$

Here we will mainly deal with the case where $E \subseteq \mathbb{R}^{m+1}$. Hence, when considering a function $f : E \to \mathbb{R}^m$ with argument (t,x), we refer to $t \in \mathbb{R}$ as the first argument and $x \in \mathbb{R}^m$ as the second argument of f.

Definition 2.6.2. *Let $E \subseteq \mathbb{R}^{m+1}$ be an open set. A function $f : E \to \mathbb{R}^m$ is called locally Lipschitz in the second argument, on E, if for every compact set $\Lambda \subseteq E$ there is a constant $K_\Lambda \geq 0$ such that*

$$\|f(t,x) - f(t,y)\| \leq K_\Lambda \|x - y\|, \quad \textit{for all } (t,x), (t,y) \in \Lambda.$$

The following classical lemma [Hal80] asserts that C^1 functions are locally Lipschitz, and hence locally Lipschitz in the second argument.

Lemma 2.6.3. *If $f : E \to \mathbb{R}^m$ is of class C^1 over $E \subseteq \mathbb{R}^l$, then f is locally Lipschitz on E.*

The following result is of particular importance for Chapter 6 and follows as an immediate consequence of the fundamental existence-uniqueness theory for the initial-value problem (2.5) [CL55], [Lef65], [Hal80] (the expressions $t \to \alpha^+$ and $t \to \beta^-$ mean that t converges to α from above and to β from below, respectively).

Proposition 2.6.4. *Let E be an open subset of $\mathbb{R}^{m+1}$ and assume that $f : E \to \mathbb{R}^m$ is continuous on E and locally Lipschitz in the second argument. Then for each $(t_0, x_0) \in E$, the problem (2.5) has a unique solution $x(t)$ defined on a maximal interval (α,β), on which it is C^1. The maximal interval is open and has the property that, if $\beta < +\infty$ (resp. $\alpha > -\infty$), either $(t, x(t))$ approaches the boundary of E or $x(t)$ is unbounded as $t \to \beta^-$ (resp. $t \to \alpha^+$).*

Note that, as a particular case, when $E = \mathbb{R}^{m+1}$ and $\beta < \infty$, $x(t)$ is unbounded as $t \to \beta^-$. For instance, the IVP

$$\begin{cases} x' = -x^2 \\ x(1) = 1 \end{cases}$$

has as solution the function $g : (0,\infty) \to \mathbb{R}$, defined by $g(x) = 1/x$, and has maximal interval $(0,\infty)$.

The following proposition can be found in Section 32.4 of [Arn78].

Proposition 2.6.5. *Let $g : E \times \mathbb{R} \to \mathbb{R}^m$, where $E \subseteq \mathbb{R}^{m+1}$ is an open set, be a function defined as follows: given $(t_0, x_0) \in E$, the function $g(t_0, x_0, \cdot) : \mathbb{R} \to \mathbb{R}^m$ is the solution of the IVP (2.5). Then if f is analytic, so is g.*

CHAPTER 3

Polynomial IVPs

3.1 Introduction

This chapter reviews basic results concerning polynomial differential equations and a particular model of analog computation, the General Purpose Analog Computer. In Section 3.2, we present general properties of polynomial differential equations. In particular, we recall the notion of differentially algebraic functions, that we then restrict to the case of explicit polynomial ordinary functions. We then show that the latter class of functions is closed under many algebraic operations and we prove a new result, that can be seen as a strengthening of Rubel and Singer's elimination theorem for differentially algebraic functions [RS85], that will be useful in later constructions.

In Section 3.3, we introduce one of the first models of analog computation, Shannon's General Purpose Analog Computer (GPAC), and present some of its variants. Finally, in Section 3.4, we study the properties of the GPAC and relate it with the class of the solutions of polynomial differential equations.

The results of this chapter were partially published in the following references: [Gra02], [GC03], [Gra03], [Gra04].

3.2 Polynomial differential equations

In this section we present useful properties for polynomial differential equations. These properties are a selection of results that can be found in the literature, plus some results that are proved and that will be important later.

Perhaps the most well known class of functions satisfying polynomial differential equations are the differentially algebraic functions [Rub89]:

Definition 3.2.1. *Let $I \subseteq \mathbb{R}$ be some open interval. The unary function $y : I \to \mathbb{R}$ is called differentially algebraic (d.a. for short) on I if it satisfies a differential equation of the form*

$$p(x, y, y', ..., y^{(n)}) = 0 \quad \text{on } I,$$

where $p : \mathbb{R}^{n+2} \to \mathbb{R}$ is a real nontrivial polynomial in its $n+2$ variables, and $n \in \mathbb{N}$. If y is not d.a., then we say that y is transcendentally transcendental.

Differentially algebraic functions abound in elementary mathematical analysis. Examples of d.a. functions are the polynomials, rational functions, algebraic functions, the exponential and logarithm, the trigonometric functions, Bessel functions, among many others [Rub89]. Also, it can be shown that sums, products, differences, quotients, compositional inverses and composition of d.a. functions are again d.a. [Rub89].

Historically, the first example of a transcendentally transcendental function was Euler's Gamma function

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt.$$

This was proved by Hölder in 1887 [Höl87]. Another example is Riemann's Zeta function

$$\zeta(x) = \sum_{n=1}^{\infty} \frac{1}{n^x}$$

that, on the real line and for $x > 1$, is given by

$$\zeta(x) = \frac{1}{\Gamma(x)} \int_0^\infty \frac{u^{x-1}}{e^u - 1} du.$$

The first proof of its transcendentally transcendental nature was given by Hilbert, and was written up by Stadigh [Sta02].

Although d.a. functions are certainly quite interesting mathematical objects, especially from a differential algebraic point of view, their applications to other areas can be limited due to the fact that they rely on an implicit characterization. This is the case of the present work, where we are interested in having "normal form" ODEs with the format $y' = p(t,y)$. Notice that while (components of the) solutions of the previous ODE are obviously d.a. on an interval I, the result does not work the other way around: if $\varphi : I \to \mathbb{R}$ is d.a. then, in general, we cannot find a polynomial ODE $y' = p(t,y)$ having φ as a solution on the whole interval I (although it possible to show that φ is solution of a polynomial ODE in some nonempty open interval $I' \subseteq I$, cf. [GC03]).

So, we now consider IVPs defined with polynomial ODEs

$$\begin{cases} x' = p(t,x) \\ x(t_0) = x_0 \end{cases} \tag{3.1}$$

where $x \in \mathbb{R}^n$ and p is a vector of polynomials. In the remainder of the text, an IVP of the form (3.1) will be called a polynomial IVP, and each component of its solution will be termed a PIVP function.

Note that by the standard existence-uniqueness theory (Proposition 2.6.4), an IVP associated to a system of ODEs $x' = f(t,x)$ has a unique solution whenever f is continuous and locally Lipschitz with respect to the second argument. Since polynomials are globally analytic, these conditions are automatically satisfied for (3.1). Moreover, by Proposition 2.6.5, PIVP functions must be analytic.

Example 3.2.2. The following are examples of PIVP functions [Sha41].

1. The exponential function e^x. It is solution of the polynomial IVP

$$y' = y, \quad y(0) = 1. \tag{3.2}$$

2. The trigonometric functions cos, sin. The vector $(\cos, \sin)$ is the solution of the polynomial IVP
$$\begin{cases} y_1' = -y_2 \\ y_2' = y_1 \end{cases} \qquad \begin{cases} y_1(0) = 1 \\ y_1(0) = 0. \end{cases} \tag{3.3}$$

3. The inverse function $x \mapsto 1/x$. On the interval $(0, \infty)$, it is the solution of the polynomial IVP
$$y' = -y^2, \quad y(1) = 1.$$
A similar polynomial IVP can be obtained for the interval $(-\infty, 0)$.

Example 3.2.3. The PIVP functions are also closed under the following operations (as far as we know, these properties have only been reported in the literature for the broader case of d.a. functions):

1. Field operations $+, -, \times, /$. For instance, if $f, g : I \to \mathbb{R}$, where $I \subseteq \mathbb{R}$ is an open interval, are PIVP functions, then so is $f + g$ in I. As matter of fact, if f, g are the first components of the solutions of the polynomial IVPs
$$\begin{cases} x' = p(t,x) \\ x(t_0) = x_0 \end{cases} \quad \text{and} \quad \begin{cases} y' = q(t,y) \\ y(t_0) = y_0 \end{cases}$$
respectively then, since $f_1'(x) + f_2'(x) = p_1(t,x) + q_1(t,x)$, $f_1 + f_2$ is the last component of the solution of the polynomial IVP
$$\begin{cases} x' = p(t,x) \\ y' = q(t,y) \\ z' = p_1(t,x) + q_1(t,y) \end{cases} \qquad \begin{cases} x(t_0) = x_0 \\ y(t_0) = y_0 \\ z(t_0) = x_0 + y_0. \end{cases}$$
Similar proofs apply for the operations $-, \times, /$. It should be mentioned that the quotient f/g is only a PIVP function where g is not zero, and that the polynomial IVP remains the same only between two consecutive zeros. For instance tan is a PIVP function in $(-\pi/2, \pi/2)$.

2. Composition. If $f : I \to \mathbb{R}$, $g : J \to \mathbb{R}$, where $I, J \subseteq \mathbb{R}$ are open intervals and $f(I) \subseteq J$, are PIVP functions, then so is $g \circ f$ on I. Indeed, supposing that f, g are the first components of the solutions of the PIVPs
$$\begin{cases} x' = p(t,x) \\ x(t_0) = x_0 \end{cases} \quad \text{and} \quad \begin{cases} y' = q(t,y) \\ y(t_1) = y_0 \end{cases}$$
respectively then, since $(g \circ f)'(x) = g'(f(x)).f'(x)$, one concludes that $g \circ f$ is the last component of the solution of the PIVP
$$\begin{cases} x' = p(t,x) \\ y' = q(x_1,y) \\ z' = q_1(x_1,y)p_1(t,x) \end{cases} \qquad \begin{cases} x(t_0) = x_0 \\ y(t_0) = f(x_0) \\ z(t_0) = g \circ f(x_0) \end{cases}$$
where $p_1(t,x)$, $q_1(x_1,y)$, and $q_1(x_1,y)p_1(t,x)$ denote the derivatives $f'(t)$, $g'(f(t))$, and $(g \circ f(t))'$, respectively.

3. Differentiation. If $f : I \to \mathbb{R}$, where $I \subseteq \mathbb{R}$ is an open interval, is a PIVP function, then so is $f' : I \to \mathbb{R}$. To see this, suppose that f is the first component of the solution of the polynomial IVP
$$\begin{cases} x' = p(t,x) \\ x(t_0) = x_0. \end{cases}$$
Then
$$f'(t) = x_1''(t) = \frac{d}{dt}p_1(t,x) = \frac{\partial p_1}{\partial t} + \sum_{i=1}^{n} \frac{\partial p_1}{\partial x_i} x_i' = \frac{\partial p_1}{\partial t} + \sum_{i=1}^{n} \frac{\partial p_1}{\partial x_i} p_i(t,x)$$
which implies that f' is the last component of the solution of the polynomial IVP
$$\begin{cases} x' = p(t,x) \\ z' = \frac{\partial p_1}{\partial t} + \sum_{i=1}^{n} \frac{\partial p_1}{\partial x_i} p_i(t,x) \end{cases} \qquad \begin{cases} x(t_0) = x_0 \\ z(t_0) = f'(t_0). \end{cases}$$

4. Compositional inverses. If $f : I \to \mathbb{R}$, where $I \subseteq \mathbb{R}$ is an open interval, is a bijective PIVP function, then so is f^{-1}. This case will be shown in the end of this section. In particular, this result says that log, arcsin, arccos, and arctan are also PIVP functions.

From the preceding examples, we conclude that we have just proved the following corollary (which also seems to be stated on the literature only for the case of d.a. functions):

Corollary 3.2.4. *All closed-form functions are PIVP functions.*

When proving that some function is PIVP, we will find most convenient to make use of ODEs not only defined with polynomials, but also with other PIVP functions. For this purpose, we have to resort to the next theorem, which can be viewed as a strengthening of the elimination theorem of Rubel and Singer for differentially algebraic functions [RS85] to the case of polynomial IVPs. We should also remark that a different proof is given implicitly in [Gra04].

Theorem 3.2.5. *Consider the IVP*
$$\begin{cases} x' = f(t,x) \\ x(t_0) = x_0 \end{cases} \tag{3.4}$$
where $f : \mathbb{R}^{n+1} \to \mathbb{R}^n$ and each component of f is a composition of polynomials and PIVP functions. Then there exist $m \geq n$, a polynomial $p : \mathbb{R}^{m+1} \to \mathbb{R}^m$ and a $y_0 \in \mathbb{R}^m$ such that the solution of (3.4) is given by the first n components of $y = (y_1, ..., y_m)$, where y is the solution of the polynomial IVP
$$\begin{cases} y' = p(t,y), \\ y(t_0) = y_0. \end{cases}$$

Proof. Let $f = (f_1, \ldots, f_n)$. If every f_i, $i = 1, \ldots, n$, is a polynomial there is nothing to prove, so we suppose $k \leq n$ is the first integer such that f_k is not a polynomial. For simplicity we will handle only the case where the transcendence degree of f_k over the ring of polynomials is 1; the case of higher degree will follow from iterating the construction.

Suppose then that $f_k(t, x_1, \ldots, x_n) = g(p_k(t, x_1, ..., x_n))$, where p_k is a polynomial and g is the first component of the solution y of
$$\begin{cases} y' = q(t,y) \\ y(t_0) = y_0, \end{cases} \tag{3.5}$$

where q is a vector of polynomials in $\mathbb{R}^l$ not independent of y (thus ensuring that g is not trivially a polynomial in t). Define new variables $\overline{y}$ by performing the change of independent variable $t \mapsto p_k(t, x_1, ..., x_n)$ in (3.5), that is,

$$\overline{y}(t) = y(p_k(t, x_1, \ldots, x_n)).$$

Then

$$\begin{aligned} \frac{d\overline{y}}{dt} &= \frac{dy}{dt}(p_k(t, x_1, \ldots, x_n))\ p_k'(t, x_1, ..., x_n) \\ &= q(p_k(t, x_1, ..., x_n), \overline{y}) \left(\sum_{i=1}^{n} \frac{\partial p_k}{\partial x_i} f_i(t, x) + \frac{\partial p}{\partial t} \right), \end{aligned} \tag{3.6}$$

subject to the initial condition $\overline{y}(t_0) = y(p(t_0, x_1(t_0), ..., x_n(t_0)))$.

We now consider the new IVP constructed by appending the IVP (3.6) to (3.4) and replacing the terms $g(p(t, x_1, ..., x_n))$ by the new variable $\overline{y}_1$. By this procedure we have replaced the IVP in $\mathbb{R}^n$ (3.4) with an IVP in $\mathbb{R}^{n+l}$ substituting the first non-polynomial term of f by a polynomial and increasing the system with l new variables defined by a polynomial IVP.

If the transcendence degree of f_k over the ring of polynomials is $d > 1$ we iterate this procedure d times. In this way we have effectively eliminated all non-polynomial terms up to component k by increasing the order of the system only with polynomial equations. Repeating this procedure with the variables labeled $k+1$ up to n (whenever necessary) we end up with a polynomial IVP satisfying the stated conditions. □

Let us illustrate this theorem with some examples.

1. Consider the IVP

$$\begin{cases} x_1' = \sin^2 x_2 \\ x_2' = x_1 \cos x_2 - e^{x_1 + t} \end{cases} \qquad \begin{cases} x_1(0) = 0 \\ x_2(0) = 0 \end{cases} \tag{3.7}$$

with solution (x_1, x_2). Since sin, cos, and exp are solutions of polynomial IVPs (see (3.2) and (3.3)), Theorem 3.2.5 ensures that there is a polynomial IVP, with initial condition defined for $t_0 = 0$, whose solution is formed by x_1, x_2, and possibly other components. Since the proof of the theorem is constructive, we can derive this polynomial IVP. Indeed, one has $(\sin x_2)' = (\cos x_2)x_2'$ and $(\cos x_2)' = -(\sin x_2)x_2'$, and therefore $(\sin x_2, \cos x_2)$ is the solution of the IVP

$$\begin{cases} y_3' = y_4 x_2' \\ y_4' = -y_3 x_2' \end{cases} \Leftrightarrow \begin{cases} y_3' = y_4(x_1 \cos x_2 - e^{x_1 + t}) \\ y_4' = -y_3(x_1 \cos x_2 - e^{x_1 + t}) \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} y_3' = y_4(x_1 y_4 - e^{x_1 + t}) \\ y_4' = -y_3(x_1 y_4 - e^{x_1 + t}) \end{cases}$$

with initial condition $y_3(0) = \sin x_2(0) = 0$ and $y_4(0) = 1$. It remains to eliminate the term $e^{x_1 + t}$. But this function is the solution of the IVP

$$y_5' = y_5\,(x_1' + t') = y_5(\sin^2 x_2 + 1) = y_5(y_3^2 + 1)$$

with initial condition $y_5(0) = e^{x_1(0) + 0} = 1$. Therefore the solution of (3.7) is formed by the first two components of the solution of the polynomial IVP

$$\begin{cases} y_1' = y_3^2 \\ y_2' = y_1 y_4 - y_5 \\ y_3' = y_4(y_1 y_4 - y_5) \\ y_4' = -y_3(y_1 y_4 - y_5) \\ y_5' = y_5(y_3^2 + 1) \end{cases} \qquad \begin{cases} y_1(0) = 0 \\ y_2(0) = 0 \\ y_3(0) = 0 \\ y_4(0) = 1 \\ y_5(0) = 1. \end{cases}$$

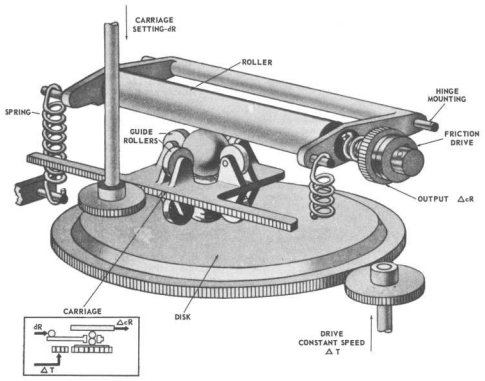


Figure 3.1: A disc type integrator device. Figure taken from [BNP71] with permission of Dover Publications, Inc.

2. Let us prove that the inverse function f^{-1} of a bijective PIVP function $f : I \to \mathbb{R}$, where $I \subseteq \mathbb{R}$ is an open interval, is also a PIVP function. We know that $(f^{-1})'(x) = 1/f'(f^{-1}(x))$. Then, between two consecutive (inverse images of) zeros a, b of f', with $a < b$, f^{-1} will be the solution of the IVP

$$y' = \frac{1}{f'(y)}, \quad y(c) = f^{-1}(c) \tag{3.8}$$

where $c = (a+b)/2$. Since f is a PIVP function, so is f'. Moreover $x \mapsto 1/x$ is also a PIVP function, and since PIVP functions are closed under composition, so is $x \mapsto 1/f'(x)$. Then (3.8) and Theorem 3.2.5 guarantee that $f^{-1} : (a,b) \to \mathbb{R}$ is a PIVP function.

3.3 The GPAC

In this section we return to the origins of analog computation with the pioneer work of Claude Shannon on the so-called *General Purpose Analog Computer* (GPAC).

In essence, a GPAC is a mathematical model of an analog device, the differential analyzer. The fundamental principles for this device were first realized by Lord Kelvin in 1876 [Bus31]. Working with an integrating device conceived by his brother James Thomson (see Fig. 3.1), he had the idea of connecting integrators to solve differential equations.

A mechanical version of the differential analyzer was first developed by V. Bush and his associates at MIT in 1931 [Bus31]. Many mechanical difficulties were overcome by ingenious solutions. However, the inherent difficulties of the mechanical devices, due to mechanical restrictions, limited the speed of these differential analyzers and originated colossal machines.

The introduction of electronic differential analyzers was able to overcome these problems to a great extent. But this was not enough to compete with emergent digital computers.

Differential analyzers were typically used in the thirties and forties to compute functions, especially to solve ODEs. Their continuous nature allowed the direct simulation of a solution from a given system of ODEs. Hence, comparatively to standard procedures that basically use a discrete version of time (i.e., the independent variable) in order to approximate the solution (with the inherent errors introduced in these processes), differential analyzers permitted faster solutions with increased precision at that time. As an example, the Rockefeller Differential Analyzer (designed by V. Bush and operational for the war effort in 1942) could solve an ODE up to order 18, being able to produce results to at least 0.1 percent accuracy [Bow96, p. 8].

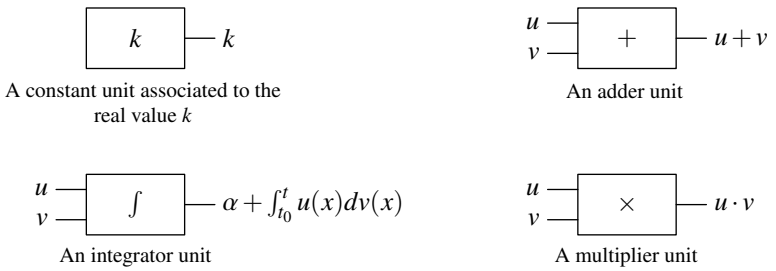


Figure 3.2: Different types of units used in a GPAC.

In short, a (mechanical) differential analyzer may be seen as a set of interconnected shafts, each of which representing one of the quantities involved in the computation. At an higher level of abstraction, there are blocks of shafts performing mathematical operations, like adding two inputs. The GPAC model was introduced by Shannon in 1941 in the paper [Sha41] as an idealization of the differential analyzer. When Shannon was a student at MIT, he worked as an operator in Bush's differential analyzer in order to make some money. He then realized that functions generated by a differential analyzer should be d.a. and published that result on [Sha41] (for further details on this result, see Section 3.4) by making use of the GPAC model. In essence, a GPAC is nothing more than a model based on circuits having several (finitely many) interconnected units. The allowed units are the ones depicted in Fig. 3.2 (actually these units are not the ones originally given by Shannon, but they are equivalent). It is required that inputs (and outputs) can never be interconnected. It is also required that each input is connected to, at most, one output.

Later, Pour-El [PE74, pp. 13-14] found a gap in Shannon's results and redefined the GPAC model in terms of quasi-linear differential equations. But, again, a gap was found by Graça and Costa [GC03, p. 647] that then redefined the GPAC in terms of circuits, but where not all kind of connections were allowed. In a subsequent work, Graça [Gra04] showed that the latter model is equivalent to a subclass of Graça and Costa's GPAC, thus providing a simpler approach. Hence, except otherwise stated, we consider that a GPAC is defined according to [Gra04] (see details below).

Before continuing, we note that for the matters of our work, it is only necessary to consider GPACs with one input. We will usually refer to this input as the time.

The model presented in this section is based in the following ideas: first, construct acyclic circuits that compute polynomials (*polynomial circuits*). We assume that a polynomial circuit may have no units at all computing, in this case, the identity. Second, use these circuits as building blocks for more complex circuits, now using integrators, that we call GPACs. A GPAC is constructed in the following manner. Take n integrators $\mathcal{I}_1, ..., \mathcal{I}_n$. Then use polynomial circuits such that the following three conditions hold:

1. Each input of a polynomial circuit is the input of the GPAC or the output of an integrator;
2. Each integrand input of an integrator is an output of a polynomial circuit;
3. Each variable of integration input of an integrator is the input of the GPAC.

Formally a polynomial circuit is defined as follows.

Definition 3.3.1. *A* polynomial circuit *is an acyclic circuit built only with adders, constants units, and multipliers.*

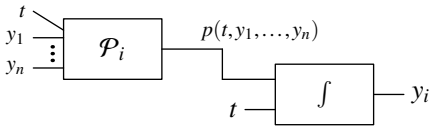


Figure 3.3: Schema of inputs and outputs for the integrator $\mathcal{I}_i$ in a GPAC. p denotes a polynomial and y_i denotes the output of $\mathcal{I}_i$.

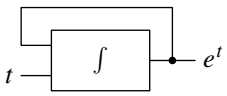


Figure 3.4: Example of a GPAC generating the exponential function e^x. The initial condition of the integrator is $y(0) = 1$.

We assume that polynomial circuits may have several inputs. The proof of the following lemma will be left to the reader.

Lemma 3.3.2. *If $x_1,...,x_n$ are the inputs of a polynomial circuit, then the output of the circuit will be $y = p(x_1,...,x_n)$, where p is a polynomial. Reciprocally, if $y = p(x_1,...,x_n)$, where p is a polynomial, then there is a polynomial circuit with inputs $x_1,...,x_n$, and output y.*

Definition 3.3.3 ([Gra04])**.** *Consider a circuit $\mathcal{U}$ with n integrators $\mathcal{I}_1,...,\mathcal{I}_n$, and one input t. Suppose that to each integrator $\mathcal{I}_i$, $i = 1,...,n$, we can associate a polynomial circuit $\mathcal{A}_i$ with the property that the integrand input of $\mathcal{I}_i$ is connected to an output of $\mathcal{A}_i$. Suppose that each input of $\mathcal{A}_i$ is connected to the output of an integrator or to the input t. Suppose also that the variable of integration input of each integrator is connected to the input t. In these conditions we say that $\mathcal{U}$ is a GPAC with input t. (cf. Fig. 3.3).*

Example 3.3.4. Let us give some examples of GPACs.

1. Consider the circuit depicted in Fig. 3.4, where the initial condition of the integrator is given by $y(0) = 1$. It is not difficult to see that this circuit is a GPAC with one integrator. The corresponding polynomial circuit has no units (notice that one could replace this polynomial circuit by a polynomial circuit having one multiplier and one constant unit associated to the value 1, that just multiplies its input by 1). Moreover, the output y must satisfy the integral equation
$$y(t) = 1 + \int_0^t y(u)du.$$
Differentiating the last equation, we have that y is solution of the polynomial IVP (3.2) i.e. the GPAC generates the function $y(t) = e^t$.

2. Consider the circuit depicted in Fig. 3.5, where the initial conditions of the integrators are given by $y_1(0) = 1$ and $y_2(0) = 0$. This circuit is a GPAC. Moreover, the outputs y_1, y_2 must satisfy
$$y_1(t) = 1 - \int_0^t y_2(u)du, \quad y_2(t) = \int_0^t y_1(u)du.$$
Differentiating the last two equations, we get that y_1 and y_2 are the solutions of the polynomial IVP (3.3), i.e. $y_1 = \cos$ and $y_2 = \sin$.

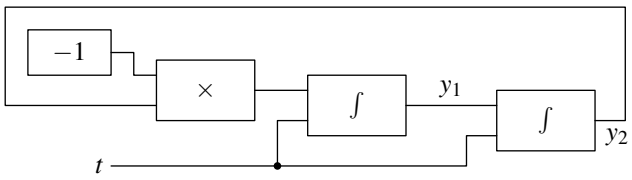


Figure 3.5: A GPAC computing sin and cos.

3.4 Properties of the GPAC

As the reader probably noted in the examples at the end of the previous section, a connection seems to exist between the class of functions generated by a GPAC and the class of PIVP functions. Actually, we will show in this section that these classes coincide. Hence, all properties described in Section 3.2 for PIVP functions are also valid for functions generated by a GPAC. Reciprocally, a GPAC may be seen as a graphical way to represent PIVP functions, describing their associated IVP.

In the remainder of this work, we will often identify PIVP functions with functions generated by a GPAC, according to the most convenient approach at that time. But before presenting this result, we would like to give a retrospective survey of the literature, concerning characterizations of functions computable by a GPAC.

In his paper [Sha41], Shannon presented the following characterization for functions generated by a GPAC.

Claim 3.4.1. *A unary function can be generated by a GPAC in Shannon's setting iff the function is differentially algebraic.*

Unfortunately, the original proof of Claim 3.4.1 has some gaps, as indicated in [PE74, pp. 13-14]. In this paper, Pour-El was also concerned with showing some relations in the spirit of claim 3.4.1. In order to achieve this result, she introduced an alternative definition for the GPAC based on differential equations. Roughly, according to Pour-El's definition, y is generated by a GPAC if there is some system of differential equations

$$A(x,y)\frac{dy}{dx} = b(x,y),$$

where $y = (y_1,...,y_n)$, such that $A(x,y)$ and $b(x,y)$ are $n \times n$ and $n \times 1$ matrices, respectively, with linear entries and, moreover, y is one of the y_i's. The original paper [PE74] can be referred for more details on this model. Pour-El proved (with some corrections made by Lipshitz and Rubel in [LR87]) the following results:

Theorem 3.4.2 (Pour-El)**.** *Let y be a differentially algebraic function on I. Then there exists a closed subinterval $I' \subseteq I$ with non-empty interior such that, on I', y can be generated by a GPAC (in Pour-El's approach).*

Theorem 3.4.3 (Pour-El, Lipshitz, Rubel)**.** *If y is generable on I by a GPAC in Pour-El's approach, then there is a closed subinterval $I' \subseteq I$ with non-empty interior such that, on I', y is differentially algebraic.*

Although the model presented by Pour-El is apparently different from Shannon's GPAC, Pour-El presented the following result:

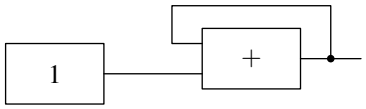


Figure 3.6: A circuit that admits no solutions as output.

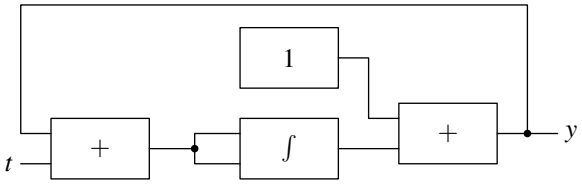


Figure 3.7: A circuit that admits two distinct solutions as outputs.

Claim 3.4.4. *If a function y is generated on I by a GPAC in the sense of Shannon, it is generated on I by a GPAC in the sense of Pour-El.*

However, the original proof of Claim 3.4.4 [PE74, proposition 1] has some gaps as pointed out in [GC03, p. 647]. In this latter paper, Graça and Costa redefined the GPAC, returning to a characterization with circuits, which is essentially the one presented in the previous section. Notice that in their characterization, not all kinds of interconnections between the units of Fig. 3.2 are allowed. Indeed, if we allow arbitrary connections, we can have circuits with no outputs (the case of the circuit depicted in Fig. 3.6), or with several outputs (the circuit of Fig 3.7). Indeed, for Fig. 3.7, it is not difficult to see that y at time t is given by

$$y(t) = 1 + \int_0^t (y(x) + x)d(y(x) + x).$$

We are supposing that $t_0 = 0$ and that the initial output of the integrator is 1. When we start the computation, we get two possible solutions:

$$y_{\pm}(t) = 1 \pm \sqrt{-2t} - t.$$

Therefore, we cannot have a physical implementation of this circuit (in particular, there is no differential analyzer that simulates this circuit. Physically, this would probably correspond to a situation where the shafts would "block", not allowing the solutions to bifurcate).

We should remark that Shannon, in its original model, restricted the GPAC to the ones with the property that "all ordinary differential equations considered have unique solutions" [Sha41, p. 338]. However, he does not give an explicit criteria to know which are the "good GPACs".

Let us now explore Graça and Costa's model to see why these situations are ruled out in this model. This comes from the following result, taken from [GC03], [Gra04], that gives the characterization of functions generated by a GPAC in terms of PIVP functions.

Proposition 3.4.5. *A function is generated by a GPAC iff it is a PIVP function.*

Proof. Suppose that we have a GPAC with n integrators $\mathcal{I}_1, \ldots, \mathcal{I}_n$, having outputs $y_1, \ldots, y_n$, respectively. To each integrator $\mathcal{I}_k$ is associated a polynomial circuit $\mathcal{A}_k$. This polynomial circuit has as inputs $y_1, \ldots, y_n$ plus the input t of the circuit. Therefore, by Lemma 3.3.2, the polynomial circuit $\mathcal{A}_k$ feeds integrator $\mathcal{I}_k$ with an input $p_k(t, y_1, \ldots, y_n)$, where p_k is a polynomial. According

to the characterization of GPACs given by Fig. 3.3, we see that the output of $\mathcal{I}_k$, y_k, must satisfy the following integral equation

$$y_k = \alpha_k + \int_{t_0}^{t} p_k(t, y_1, \ldots, y_n)\, dt,$$

where α_k is the initial condition of the integrator. Differentiating the last equation for each $k = 1, \ldots, n$, we get

$$\begin{cases} y_1' = p_1(t, y_1, \ldots, y_n) \\ \vdots \\ y_n' = p_n(t, y_1, \ldots, y_n) \end{cases} \qquad \begin{cases} y_1(t_0) = \alpha_1 \\ \vdots \\ y_n(t_0) = \alpha_n \end{cases} \tag{3.9}$$

i.e. $y_1, \ldots, y_n$ are PIVP functions. Moreover, since each output of a polynomial circuit has the format $p(t, y_1, \ldots, y_n)$, where p is a polynomial, we conclude that each output of a GPAC must be a PIVP function.

Reciprocally, suppose that f is a PIVP function, i.e. it is one of the components of the solution of (3.9), where $p_1, \ldots, p_n$ are polynomials. Then it is not difficult to build polynomial circuits $\mathcal{A}_1, \ldots, \mathcal{A}_n$ which compute $p_1, \ldots, p_n$, respectively. If we link each polynomial circuit $\mathcal{A}_k$ to an integrator $\mathcal{I}_k$ according to the layout specified in Fig. 3.3, and if we require each integrator $\mathcal{I}_k$ to have α_k as initial setting, it is not difficult to see that each output y_k of the integrator $\mathcal{I}_k$ must satisfy (3.9). This shows that $y_1, \ldots, y_n$ are generated by a GPAC. □

Some interesting consequences may be derived from the previous result.

Remark 3.4.6. Since outputs of GPACs are solution of polynomial IVPs, by Proposition 2.6.4 each output exists and is unique. Thus, we don't have the problems mentioned before and pictured in Fig. 3.6 and Fig. 3.7.

Remark 3.4.7. From the proof of the proposition it follows that, without loss of generality, each output of a GPAC may be supposed to be the output of an integrator.

Remark 3.4.8. Suppose that instead of allowing all real numbers in constant units, we only allow the use of elements from a set $A \subseteq \mathbb{R}$. Moreover, assume that A is closed under product and addition (i.e. assume that A is a subring of $\mathbb{R}$, with the operations $+$, $\times$). Then, by arguments similar to those of the proof of Proposition 3.4.5, the outputs of this subclass of GPACs are solutions of polynomial IVPs (3.9), but where all the coefficients of the polynomials $p_1, \ldots, p_n$ are elements of A. In particular, if A is the set of real computable reals, then the polynomials $p_1, \ldots, p_n$ must be computable.

CHAPTER 4

Simulation of Turing machines

4.1 Introduction

This chapter proposes new results concerning the simulation of Turing machines with analytic maps and flows. In Section 4.2, we present the encoding used in our results and, since we allow perturbed simulations of TMs, we also introduce some tools that will be helpful to keep the error under control. In section 4.3, we study some interpolation techniques that will be used in Section 4.4 where, given a TM, we present a constructive method to obtain an elementary map that simulates this TM. Moreover, the simulation is shown to be robust to perturbations.

In Section 4.5, we recall a construction that iterates a map with a system of ODEs. This result is then generalized in Section 4.6 to prove that polynomial ODEs can also perform robust simulations of TMs. As a corollary, we conclude that the GPAC is at least as powerful as Type-1 computability.

The results of this chapter were partially published in the following references: [GCB05], [GCB06].

4.2 Encoding configurations and controlling the error

In this section we present useful tools to be used in the remaining of this chapter. In Section 4.4, we will show that elementary maps can simulate TMs, i.e. that the transition function of a given TM can be extended to an elementary map. Of course, to arrive at this result, we have to associate each configuration of a Turing machine to some number. Here, we will encode each configuration as a triple $(x,y,z) \in \mathbb{N}^3$, and prove that the simulation still works if this triple is slightly perturbed. Our encoding works as follows. Without loss of generality, consider a Turing machine M using 10 symbols, the blank symbol $B = 0$, and symbols $1,2,...,9$. Let

$$\ldots BBBa_{-k}a_{-k+1}\ldots a_{-1}a_0a_1\ldots a_nBBB\ldots$$

represent the tape contents of the Turing machine M. We suppose the head to be reading symbol a_0 and $a_i \in \{0,1,...,9\}$ for all i. We also suppose that M has m states, represented by numbers 1 to m. For convenience, we consider that if the machine reaches a halting configuration it moves

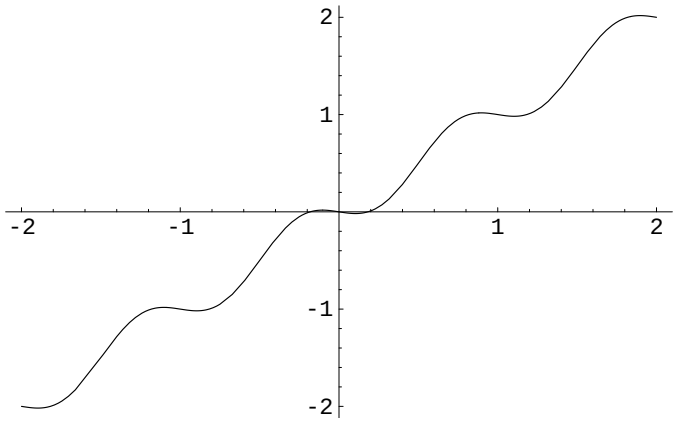


Figure 4.1: Graphical representation of the function σ.

to the same configuration. Take

$$y_1 = a_0 + a_1 10 + ... + a_n 10^n \tag{4.1}$$
$$y_2 = a_{-1} + a_{-2} 10 + ... + a_{-k} 10^{k-1}$$

and let q be the state associated to the current configuration. Then the triple $(y_1, y_2, q) \in \mathbb{N}^3$ gives the current configuration of M.

Now that we know precisely which is the encoding of configurations to be used and since we will allow simulations with perturbed values, we need some tools that allow us to maintain the error under control. For this purpose, we present three functions, $\sigma : \mathbb{R} \to \mathbb{R}$, $l_2 : \mathbb{R}^2 \to \mathbb{R}$, and $l_3 : \mathbb{R}^2 \to \mathbb{R}$, that we shall describe in the remaining of the section.

The error-contracting function σ is defined by (cf. Fig. 4.1)

$$\sigma(x) = x - 0.2 \sin(2\pi x). \tag{4.2}$$

The function σ is a uniform contraction in a neighborhood of integers:

Proposition 4.2.1. *Let $n \in \mathbb{Z}$ and let $\varepsilon \in [0, 1/2)$. Then there is some contracting factor $\lambda_\varepsilon \in (0,1)$ such that, $\forall \delta \in [-\varepsilon, \varepsilon]$, $|\sigma(n+\delta) - n| < \lambda_\varepsilon \delta$.*

Proof. It is sufficient to consider the case where $n = 0$. Because σ is odd, we only study σ in the interval $[0, \varepsilon]$. Let $g(x) = \sigma(x)/x$. This function is strictly increasing in $(0, 1/2]$. Then, noting that $g(1/2) = 1$ and $\lim_{x \to 0} g(x) = 1 - 0.4\pi \approx -0.256637$, we conclude that there exists some $\lambda_\varepsilon \in (0,1)$ such that $|\sigma(x)| < \lambda_\varepsilon |x|$ for all $x \in [-\varepsilon, \varepsilon]$. □

Remark 4.2.2. For the rest of this chapter we suppose that $\varepsilon \in [0, 1/2)$ is fixed and that λ_ε is the respective contracting factor given by Lemma 4.2.1. For instance, we can take $\lambda_{1/4} = 0.4\pi - 1 \approx 0.256637$.

The function σ will be used in our simulation to keep the error controlled when bounded quantities are involved (e.g., the actual state, the symbol being read, etc). We will also need another error-contracting function that controls the error for unbounded quantities, e.g. when using expressions that depend on the variables coding the tape contents. This will be achieved with the help of the function $l_3 : \mathbb{R}^2 \to \mathbb{R}$, which has the property that whenever $\overline{a}$ is an approximation

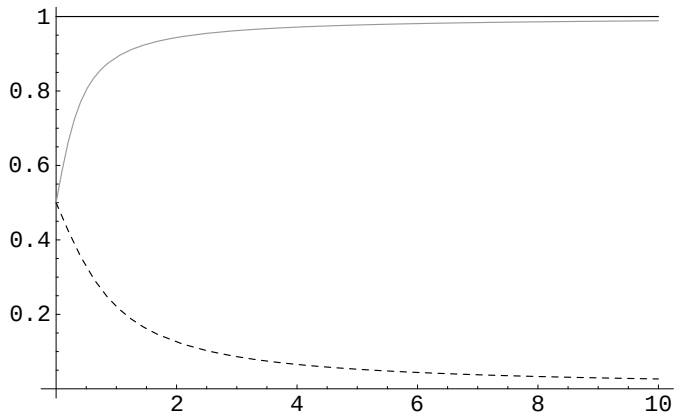


Figure 4.2: Graphical representation of the function l_2. The dashed line represents $l_2(0.2,y)$ while the gray line represents $l_2(1.2,y)$.

of $a \in \{0,1,2\}$, then $|l_3(\overline{a},y) - a| < 1/y$, for $y > 0$. In other words, l_3 is an error-contracting map, where the error is contracted by an amount specified by the second argument of l_3. We start by defining a preliminary function l_2 satisfying similar conditions, but only when $a \in \{0,1\}$ (cf. Fig. 4.2).

Lemma 4.2.3. $\left|\frac{\pi}{2} - \arctan x\right| < \frac{1}{x}$ *for* $x \in (0,\infty)$.

Proof. Let $f(x) = \frac{1}{x} + \arctan x - \frac{\pi}{2}$. It is easy to see that f is decreasing in $(0,\infty)$ and that $\lim_{x\to\infty} f(x) = 0$. Therefore $f(x) > 0$ for $x \in (0,\infty)$ and the result holds. □

Lemma 4.2.4. $\left|\frac{\pi}{2} + \arctan x\right| < \frac{1}{|x|}$ *for* $x \in (-\infty,0)$.

Proof. Take $f(x) = \frac{1}{x} + \arctan x + \frac{\pi}{2}$ and proceed as in Lemma 4.2.3. □

Proposition 4.2.5. *Let* $l_2 : \mathbb{R}^2 \to \mathbb{R}$ *be given by* $l_2(x,y) = \frac{1}{\pi}\arctan(4y(x-1/2)) + \frac{1}{2}$. *Suppose also that* $a \in \{0,1\}$. *Then, for any* $\overline{a}, y \in \mathbb{R}$ *satisfying* $|a - \overline{a}| \leq 1/4$ *and* $y > 0$,

$$|a - l_2(\overline{a},y)| < \frac{1}{y}.$$

Proof. 1. Consider $a = 0$. Then $\overline{a} - 1/2 \leq -1/4$ implies $|4y(\overline{a} - 1/2)| \geq y$. Therefore, by Lemma 4.2.4,

$$\left|\frac{\pi}{2} + \arctan(4y(\overline{a} - 1/2))\right| < \frac{1}{|4y(\overline{a} - 1/2)|} \leq \frac{1}{y}.$$

Moreover, multiplying the last inequality by $1/\pi$ and noting that $\frac{1}{\pi y} < \frac{1}{y}$, it follows that $|a - l_2(\overline{a},y)| < 1/y$.

2. Consider $a = 1$. Remark that $\overline{a} - 1/2 \geq 1/4$ and proceed as above, using Lemma 4.2.3 instead of Lemma 4.2.4.

□

Remark 4.2.6. It follows easily from the previous proof that Proposition 4.2.5 can be extended in the following way, where $y > 0$:

1. If $x \leq 1/4$, then $0 < l_2(x,y) < 1/y$;
2. If $x \geq 3/4$, then $1 - 1/y < l_2(x,y) < 1$.

We denote below, for any $x \in \mathbb{R}$, $\lceil x \rceil = \min\{k \in \mathbb{Z} : k \geq x\}$.

Proposition 4.2.7. *Let $a \in \{0,1,2\}$ and let $l_3 : \mathbb{R}^2 \to \mathbb{R}$ be given by*

$$l_3(x,y) = l_2((\sigma^{[d+1]}(x)-1)^2, 3y).(2l_2(\sigma^{[d]}(x)/2, 3y) - 1) + 1,$$

where $d = 0$ if $\varepsilon \leq 1/4$ and $d = \lceil -\log(4\varepsilon)/\log\lambda_\varepsilon \rceil$ otherwise. Then for any $\bar{a}, y \in \mathbb{R}$ satisfying $|a - \overline{a}| \leq \varepsilon$ and $y \geq 2$, we have $|a - l_3(\overline{a}, y)| < 1/y$.

Proof. Let us start by noticing that for all $x, y \in \mathbb{R}$ for which $l_2(x,y)$ is defined, we have that $0 < l_2(x,y) < 1$. Consider the case where $a = 0$ and $\overline{a} \in [-1/4; 1/4]$. By other words, take $\varepsilon \leq 1/4$. Then $\left|(\sigma(\overline{a})-1)^2 - 1\right| < 1/4$, and by the previous lemma,

$$1 - 1/y < l_2((\sigma(\overline{a})-1)^2, y) < 1.$$

Similarly, we conclude

$$-1 < 2l_2(\overline{a}/2, y) - 1 < -1 + 2/y.$$

Since $y \geq 2$, this implies

$$-1 < l_2((\sigma(\overline{a})-1)^2, y)(2l_2(\overline{a}/2, y) - 1) < (1 - 1/y)(-1 + 2/y)$$

or

$$0 < l_2((\sigma(\overline{a})-1)^2, y)(2l_2(\overline{a}/2, y) - 1) + 1 < 3/y.$$

Hence, for $a = 0$, $|a - l_3(\overline{a}, y)| < 1/y$. Proceeding similarly for $a = 1, 2$ and $\varepsilon \leq 1/4$, the same result follows.

It remains to consider the more general case $|a - \overline{a}| \leq \varepsilon$. Taking $d = \lceil -\log(4\varepsilon)/\log\lambda_\varepsilon \rceil$ and applying d times the function σ to $\overline{a}$, it follows that $\left|a - \sigma^{[d]}(\overline{a})\right| \leq 1/4$ and we fall back in the previous case (use $\sigma^{[d]}(\overline{a})$ instead of $\overline{a}$). □

4.3 Determining the next action - Interpolation techniques

In this section we present some tools, based on interpolation, that will allow us in the next section to define an elementary map that simulates the transition function of a given TM. We have two objectives. The first one is, given the integers coding the configuration of the TM, to extract the symbol being read by the tape head. The second objective is, given the current state and symbol being read by the tape head, to determine the next action to be performed (i.e. to determine the next state, the symbol to be written on the tape, and the next move).

Detecting the symbol being read by the tape head. For this purpose, we introduce an analytic extension $\omega : \mathbb{R} \to \mathbb{R}$ of the function $f : \mathbb{N} \to \mathbb{N}$ defined by $f(n) = n \bmod 10$. Indeed, from the coding (4.1) we see that $f(y_1) = a_0$, as wanted. To achieve this purpose, we can use a periodic function, of period 10, such that $\omega(i) = i$, for $i = 0, 1, ..., 9$. Then, using trigonometric interpolation (cf. [Atk89, pp. 176-182]), one may take

$$\omega(x) = a_0 + a_5\cos(\pi x) + \left(\sum_{j=1}^{4} a_j \cos\left(\frac{j\pi x}{5}\right) + b_j \sin\left(\frac{j\pi x}{5}\right)\right), \tag{4.3}$$

where $a_0,...,a_5,b_1,...,b_4$ are computable coefficients that can be explicitly obtained by solving a system of linear equations. In particular

$$a_0 = 9/2, \quad a_1 = a_2 = a_3 = a_4 = -1, \quad a_5 = -1/2$$
$$b_1 = -\sqrt{5+2\sqrt{5}}, \quad b_2 = -\sqrt{1+\frac{2}{\sqrt{5}}}, \quad b_3 = -\sqrt{5-2\sqrt{5}}, \quad b_4 = -\sqrt{1-\frac{2}{\sqrt{5}}}.$$

Due to the existence of errors in the argument of ω, it is useful to know how much can an integer input of ω be perturbed so that the propagated error does not exceed ε. This can be done as follows. Note that ω is uniformly continuous in $\mathbb{R}$ (ω has period 10 and is continuous in the interval $[0,10]$). Hence, for every $\varepsilon \in (0,1/2)$, there will be some $\zeta_\varepsilon > 0$ satisfying

$$\forall n \in \mathbb{N},\ x \in [n-\zeta_\varepsilon, n+\zeta_\varepsilon] \ \Rightarrow\ |\omega(x) - n \bmod 10| \leq \varepsilon. \tag{4.4}$$

Determining the next action to be performed. Since we want to simulate the transition function of a TM we need to know the following: given the symbol being read by the tape head, $y \in \{0,1,\ldots,9\}$, and the current state $q \in \{1,\ldots,m\}$, determine the next action — the next symbol to be written in the tape, the move to be performed, and the new state. This will be done with the help of Lagrange interpolation. Moreover, since we want to derive a simulation of Turing machines robust to (small) perturbations, we will not use exact values, but allow an error less than ε, with $0 < \varepsilon < 1/2$, on y and q. Special care is thus needed to ensure that the error does not amplify along the simulation. Here, we study the propagation of errors throughout the iteration of polynomial maps defined via Lagrange interpolation.

Let M be a Turing machine (using 10 symbols, as mentioned in Section 4.2) and consider the following functions, where $x \in \mathbb{R}$:

$$Q_i(x) = \prod_{\substack{j=1 \\ j\neq i}}^{m} \frac{(x-j)}{(i-j)}, \quad S_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{9} \frac{(x-j)}{(i-j)}.$$

Note that

$$Q_i(x) = \begin{cases} 0, \text{ if } x = 1,...,i-1,i+1,...,m \\ 1, \text{ if } x = i \end{cases} \quad \text{and} \quad S_i(x) = \begin{cases} 0, \text{ if } x = 0,...,i-1,i+1,...,9 \\ 1, \text{ if } x = i. \end{cases}$$

Suppose that on state i and symbol j, the state of the next configuration is $q_{i,j}$. Suppose also that $|q_{i,j}| \leq N$ for some suitable constant N (e.g., in this case, we may take $N = m$). Then the state that follows state q and symbol y can be given by

$$q_{next} = \sum_{i=0}^{9} \sum_{j=1}^{m} S_i(y) Q_j(q) q_{i,j}. \tag{4.5}$$

A similar procedure may be used to determine the next symbol to be written and the next move. The main problem is that we will not use q neither y, but some approximations $\overline{q}$ and $\overline{y}$, respectively, thereby obtaining $\overline{q}_{next}$. Hence, we want to increase the precision of $\overline{q}$ and $\overline{y}$ so that $|q_{next} - \overline{q}_{next}| < \varepsilon$ (ε will be an upper bound for the error allowed during a computation). To achieve this precision, we only need to compute each term in the sum given in (4.5) with an error less than $\varepsilon/(10m)$. This is verified when

$$|S_i(\overline{y}) Q_j(\overline{q}) - S_i(y) Q_j(q)| < \frac{\varepsilon}{10mN} \tag{4.6}$$

since this condition implies

$$|S_i(\overline{y})Q_j(\overline{q})q_{i,j} - S_i(y)Q_j(q)q_{i,j}| < \frac{\varepsilon}{10m}.$$

Lemma 4.3.1. *Let $n \in \mathbb{N}$ and $(x_1,\ldots,x_n),(y_1,\ldots,y_n) \in \mathbb{R}^n$. If $|x_i| \leq K, |y_i| \leq K$ for $i = 1,\ldots,n$ and some $K > 0$, then*

$$|x_1 \ldots x_n - y_1 \ldots y_n| \leq (|x_1 - y_1| + \ldots + |x_n - y_n|)\, K^{n-1}.$$

Proof. The Lemma is trivial for $n = 1$. Suppose it holds for $n \in \mathbb{N}$. Then

$$\begin{aligned}
&|x_1 \ldots x_{n+1} - y_1 \ldots y_{n+1}| \\
&\quad \leq |x_1 \ldots x_{n+1} - x_1 \ldots x_n y_{n+1}| + |x_1 \ldots x_n y_{n+1} - y_1 \ldots y_{n+1}| \\
&\quad = |x_1 \ldots x_n|\,|x_{n+1} - y_{n+1}| + |x_1 \ldots x_n - y_1 \ldots y_n|\,|y_{n+1}| \\
&\quad \leq K^n\,|x_{n+1} - y_{n+1}| + (|x_1 - y_1| + \ldots + |x_n - y_n|)K^{n-1}K \\
&\quad = (|x_1 - y_1| + \ldots + |x_n - y_n| + |x_{n+1} - y_{n+1}|)K^n
\end{aligned}$$

which proves the statement by induction. □

Note that (4.6) is satisfied if

$$\left| \prod_{\substack{r=0 \\ r\neq i}}^{9} (\overline{y} - r) \prod_{\substack{s=1 \\ s\neq j}}^{m} (\overline{q} - s) - \prod_{\substack{r=0 \\ r\neq i}}^{9} (y - r) \prod_{\substack{s=1 \\ s\neq j}}^{m} (q - s) \right| < \frac{\varepsilon}{10mN}$$

because $\left| \prod_{\substack{r=0 \\ r\neq i}}^{9} (i - r) \prod_{\substack{s=1 \\ s\neq j}}^{m} (j - s) \right| \geq 1$. Take $K = \max\{9+\varepsilon, m-1+\varepsilon\}$. Applying Lemma 4.3.1, (4.6) will hold for

$$9|y - \overline{y}| + (m-1)|q - \overline{q}| < \frac{\varepsilon}{10mNK^{m+7}}.$$

Then, in order to have $|q_{next} - \overline{q}_{next}| \leq \varepsilon$, it is sufficient to take $\overline{y}$ and $\overline{q}$ with an error less than

$$|y - \overline{y}|, |q - \overline{q}| < \frac{\varepsilon}{10mNK^{m+7}(m+8)}. \tag{4.7}$$

To achieve this, and supposing that $\overline{y}$ and $\overline{q}$ are initially given with an error less than $\varepsilon < 1/2$, we only have to apply j times the error-correcting function σ so that $\sigma^{[j]}(\overline{y})$ and $\sigma^{[j]}(\overline{q})$ have greater precision than the bound in (4.7). This condition holds for every j satisfying

$$j \geq \left\lceil \frac{\log(10mNK^{m+7}(m+8))}{-\log \lambda_\varepsilon} \right\rceil,$$

where λ_ε is given by Proposition 4.2.1.

4.4 Robust simulations of Turing machines with PIVP maps

In this section we show, in a constructive manner, how to simulate a Turing machine with an elementary map robust to (small) perturbations. We first prove the following theorem.

Theorem 4.4.1. *Let $\psi : \mathbb{N}^3 \to \mathbb{N}^3$ be the transition function of some Turing machine M. Then, given some $0 \leq \varepsilon < 1/2$, ψ admits an elementary extension $h_M : \mathbb{R}^3 \to \mathbb{R}^3$ with the property that*

$$\|(y_1,y_2,q) - (\overline{y}_1,\overline{y}_2,\overline{q})\|_\infty \leq \varepsilon \;\Rightarrow\; \|\psi(y_1,y_2,q) - h_M(\overline{y}_1,\overline{y}_2,\overline{q})\|_\infty \leq \varepsilon, \tag{4.8}$$

where $(y_1,y_2,q) \in \mathbb{N}^3$ encodes some configuration of M.

Proof. As mentioned in Section 4.2, we can assume without loss of generality that M uses 10 symbols and that we encode the configurations of M as described in (4.1). Here we will show that all quantities relevant for the computation can be obtained with sufficient precision so that the computation can continue without being flooded with uncontrollable errors. In general, except otherwise stated, we will compute each quantity with precision bounded by ε.

1. **Determine the symbol being read.** Let a_0 be the symbol being actually read by the Turing machine M. Then $\omega(y_1) = a_0$, where ω is given by (4.3). We must show that the effect of the error present in $\overline{y}_1$ can be controlled. Since $|y_1 - \overline{y}_1| \leq \varepsilon$,

$$|a_0 - \omega \circ \sigma^{[l]}(\overline{y}_1)| \leq \varepsilon, \quad \text{with } l = \left\lceil \left| \frac{\log(\zeta_\varepsilon/\varepsilon)}{\log \lambda_\varepsilon} \right| \right\rceil, \tag{4.9}$$

   where ζ_ε is given by (4.4). Then pick $\overline{y} = \omega \circ \sigma^{[l]}(\overline{y}_1)$ as an approximation of the symbol currently being read. Similarly, $\omega \circ \sigma^{[l]}(\overline{y}_2)$ gives an approximation of a_{-1}, with error bounded by ε.

2. **Determine the next state.** The map that returns the next state is defined by Lagrange interpolation. This can be done as follows. Let y be the symbol being currently read and q the current state. Recall that m denotes the number of states and $k = 10$ is the number of symbols. On the absence of error on y and q,

$$q_{next} = \sum_{i=0}^{9} \sum_{j=1}^{m} \left(\prod_{\substack{r=0 \\ r \neq i}}^{9} \frac{(y-r)}{(i-r)} \right) \left(\prod_{\substack{s=1 \\ s \neq j}}^{m} \frac{(q-s)}{(j-s)} \right) q_{i,j},$$

   where $q_{i,j}$ is the state that follows symbol i and state j. However, we are dealing with the approximations $\overline{q}$ and $\overline{y}$. Therefore, we define instead (cf. Section 4.3)

$$\overline{q}_{next} = \sum_{i=0}^{9} \sum_{j=1}^{m} \left(\prod_{\substack{r=0 \\ r \neq i}}^{9} \frac{(\sigma^{[n]}(\overline{y})-r)}{(i-r)} \right) \left(\prod_{\substack{s=1 \\ s \neq j}}^{m} \frac{(\sigma^{[n]}(\overline{q})-s)}{(j-s)} \right) q_{i,j},$$

   with

$$n = \left\lceil \frac{\log(10m^2K^{m+7}(m+8))}{-\log \lambda_\varepsilon} \right\rceil, \qquad K = \max\{19/2, m-1/2\},$$

   which yields $|\overline{q}_{next} - q_{next}| \leq \varepsilon$.

3. **Determine the symbol to be written on the tape.** Using a construction similar to the previous case, the symbol to be written, s_{next}, can be approximated with precision ε, i.e. $|s_{next} - \overline{s}_{next}| \leq \varepsilon$.

4. **Determine the direction of the move for the head.** Let h denote the direction of the move of the head, where $h = 0$ denotes a move to the left, $h = 1$ denotes a "no move", and $h = 2$ denotes a move to the right. Then the "next move" h_{next} can be approximated by means of Lagrange interpolation as in steps 2 and 3, therefore obtaining $|h_{next} - \overline{h}_{next}| \leq \varepsilon$.

5. **Update the tape contents.** In the absence of error, the "next value" of y_1, y_1^{next}, is given by Lagrange interpolation as a function of y_1, y_2, s_{next} and h_{next} (recall that $a_0 = \omega(y_1)$):

$$\begin{aligned} y_1^{next} = {} & (10(y_1 + s_{next} - \omega(y_1)) + \omega(y_2)) \frac{(1-h_{next})(2-h_{next})}{2} \\ & + (y_1 + s_{next} - \omega(y_1)) h_{next}(2-h_{next}) + \frac{y_1 - \omega(y_1)}{10} \frac{h_{next}(1-h_{next})}{-2}. \end{aligned}$$

To make the simulation robust, we define instead functions $\overline{P}_1, \overline{P}_2, \overline{P}_3$ which are intended to approximate the tape contents after the head moves left, does not move, or moves right, respectively. Let H_1 be a "sufficiently good" approximation of h_{next}, yet to be determined. Then, y_1^{next} can be approximated by

$$\overline{y}_1^{next} = \overline{P}_1 \frac{(1-H_1)(2-H_1)}{2} + \overline{P}_2\, H_1(2-H_1) + \overline{P}_3 \frac{H_1(1-H_1)}{-2} \tag{4.10}$$

with

$$\begin{aligned} \overline{P}_1 &= 10(\sigma^{[d+4]}(\overline{y}_1) + \sigma^{[d+4]}(\overline{s}_{next}) - \sigma^{[d+4]}(\overline{y})) + \sigma^{[d+2]} \circ \omega \circ \sigma^{[l]}(\overline{y}_2) \\ \overline{P}_2 &= \sigma^{[d+2]}(\overline{y}_1) + \sigma^{[d+2]}(\overline{s}_{next}) - \sigma^{[d+2]}(\overline{y}) \\ \overline{P}_3 &= \tfrac{1}{10}\left(\sigma^{[d+1]}(\overline{y}_1) - \sigma^{[d+1]}(\overline{y})\right), \end{aligned}$$

where d is given by Proposition 4.2.7 and l is given by (4.9), as we show below.

First, notice that when exact values are used and $H_1 = h_{next}$, one has $\overline{y}_1^{next} = y_1^{next}$. However, $\overline{P}_1$ in (4.10) depends on $\overline{y}_1$, which is not a bounded value. If we would simply take $H_1 = \overline{h}_{next}$, the error of the term $(1-H_1)(2-H_1)/2$ will be arbitrarily amplified when multiplied by $\overline{P}_1$. Hence, H_1 must be a sharp estimate of h_{next}, proportional to $\overline{y}_1$.

Our goal is to define an approximation H_1 of h_{next} with error at most δ, i.e. $H_1 = h_{next} + \delta$, such that $\overline{y}_1^{next}$ has error bounded by ε. Let us take $P_1 = 10(y_1 + s_{next} - \omega(y_1)) + \omega(y_2)$, $P_2 = y_1 + s_{next} - \omega(y_1)$, and $P_3 = (y_1 - \omega(y_1))/10$. To simplify the construction, we first suppose that $d = 0$, i.e. $\varepsilon \leq 1/4$. Since $h_{next} \in \{0,1,2\}$,

$$\begin{aligned} & |y_1^{next} - \overline{y}_1^{next}| \\ & \quad \leq |P_1 - \overline{P}_1| + |P_2 - \overline{P}_2| + |P_3 - \overline{P}_3| + |4\overline{P}_1\delta| + |7\overline{P}_2\delta| + |3\overline{P}_3\delta| \\ & \quad < 0.112 + |4\overline{P}_1\delta| + |7\overline{P}_2\delta| + |3\overline{P}_3\delta|. \end{aligned}$$

We used the fact that $|P_1 - \overline{P}_1| < 0.049$, $|P_2 - \overline{P}_2| < 0.05$, $|P_3 - \overline{P}_3| < 0.013$ to derive the last inequality. Hence, $|y_1^{next} - \overline{y}_1^{next}| < 1/4$ can be achieved if $|\overline{P}_1\delta|, |\overline{P}_2\delta|, |\overline{P}_3\delta| < 0.01$. Taking $|\overline{P}_1\delta| < 0.01$, one has $|\delta| < 0.01/|\overline{P}_1|$ or equivalently, considering (4.1) and the

definition of $\overline{P}_1$ (note that $|\overline{P}_1| < 10^{n+2}$, where n is the number of symbols written on the right half of the tape encoded by y_1),

$$|\delta| < 10^{-n-4}.$$

But $\bar{y}_1 > 10^n - 1/2$ implies $10000(\bar{y}_1 + 1/2) > 10^{n+4}$. Therefore we just have to take

$$|\delta| < \frac{1}{10000\,(\bar{y}_1 + 1/2)}.$$

Using a similar procedure for the inequalities $|\overline{P}_2\delta| < 0.01$ and $|\overline{P}_3\delta| < 0.01$, one reaches the same bound.

So far we have seen that to guarantee that the error $|y_1^{next} - \bar{y}_1^{next}|$ is less than ε, H_1 has to approximate h_{next} within the above bound, which depends on $\bar{y}_1$. This can be achieved with

$$H_1 = l_3(\overline{h}_{next}, 10000\,(\bar{y}_1 + 1/2) + 2),$$

as shown in Proposition 4.2.7. Notice that the extra 2 in the second argument of l_3 is only needed to ensure that $10000\,(\bar{y}_1 + 1/2) + 2 \geq 2$, as required by Proposition 4.2.7.

We can generalize this result to $\varepsilon < 1/2$ by applying d times the function σ to all the terms in the expressions of $\overline{P}_1$, $\overline{P}_2$ and $\overline{P}_3$, where d is given by Proposition 4.2.7. Therefore, $\bar{y}_1^{next}$ can be defined by (4.10).

Proceeding in the same manner for $\bar{y}_2^{next}$, one may take

$$\bar{y}_2^{next} = \overline{Q}_1 \frac{(1-H_2)(2-H_2)}{2} + \overline{Q}_2 H_2(2-H_2) + \overline{Q}_3 \frac{H_2(1-H_2)}{-2} \tag{4.11}$$

where

$$H_2 = l_3(\overline{h}_{next}, 10000\,(\bar{y}_2 + 1/4) + 2), \quad \overline{Q}_3 = 10\sigma^{[d+4]}(\bar{y}_2) + \sigma^{[d+2]}(\bar{s}_{next}),$$
$$\overline{Q}_1 = \frac{\sigma^{[d+1]}(\bar{y}_2) - \sigma^{[d+1]} \circ \omega \circ \sigma^{[l]}(\bar{y}_2)}{10}, \quad \overline{Q}_2 = \sigma^{[d+2]}(\bar{y}_2).$$

We then conclude that $|\bar{y}_1^{next} - y_1^{next}| < \varepsilon$ and $|\bar{y}_2^{next} - y_2^{next}| < \varepsilon$.

To finish the proof of Theorem 4.4.1, we put together the maps described above and define $h_M : \mathbb{R}^3 \to \mathbb{R}^3$ as $h_M(\bar{y}_1, \bar{y}_2, \bar{q}) = (\bar{y}_1^{next}, \bar{y}_2^{next}, \bar{q}_{next})$. □

We shall now prove the main theorem of this section.

Theorem 4.4.2. *Let $\psi : \mathbb{N}^3 \to \mathbb{N}^3$ be the transition function of a Turing machine M, under the encoding described above, and let $0 < \delta < \varepsilon < 1/2$. Then ψ admits a globally analytic elementary extension $f_M : \mathbb{R}^3 \to \mathbb{R}^3$, robust to perturbations in the following sense: for all f such that $\|f - f_M\|_\infty \leq \delta$, for all $j \in \mathbb{N}$, and for all $\overline{x}_0 \in \mathbb{R}^3$ satisfying $\|\overline{x}_0 - x_0\|_\infty \leq \varepsilon$, where $x_0 \in \mathbb{N}^3$ represents an initial configuration,*

$$\left\| f^{[j]}(\overline{x}_0) - \psi^{[j]}(x_0) \right\|_\infty \leq \varepsilon.$$

Proof. Using Theorem 4.4.1, one can find a map h_M such that (4.8) holds. Let $i \in \mathbb{N}$ satisfy $\sigma^{[i]}(\varepsilon) \leq \varepsilon - \delta$. Define a map $f_M = \sigma^{[i]} \circ h_M$. Then, if $x_0 \in \mathbb{N}^3$ is an initial configuration,

$$\|\overline{x}_0 - x_0\|_\infty \leq \varepsilon \;\Rightarrow\; \|f_M(\overline{x}_0) - \psi(x_0)\|_\infty \leq \varepsilon - \delta.$$

Thus, by the triangle inequality, if $\|\overline{x}_0 - x_0\|_\infty \leq \varepsilon$, then

$$\begin{aligned}\|f(\overline{x}_0) - \psi(x_0)\|_\infty &\leq \|f(\overline{x}_0) - f_M(\overline{x}_0)\|_\infty + \|f_M(\overline{x}_0) - \psi(x_0)\|_\infty \\ &\leq \delta + (\varepsilon - \delta) = \varepsilon.\end{aligned}$$

This proves the result for $j = 1$. For $j > 1$, proceed by induction. □

A few remarks are in order. First, and as noticed before, we implicitly assumed that if z is a halting configuration, then $\psi(z) = z$. Secondly, we notice that the upper bound on ε, $1/2$, results from the encoding we have chosen, which is over the integers. In fact, the bound is maximal with respect to that encoding. We also remark that the proof of the previous theorem is constructive and that f can be obtained by composing the following functions: polynomials, sin, cos, and arctan, i.e. f is elementary and also a PIVP function.

4.5 Iterating maps with ODEs

In this section we show how to iterate a map from integers to integers with smooth ODEs. By a smooth ODE we understand an ODE

$$y' = f(t, y) \tag{4.12}$$

where f is of class C^k, for $1 \leq k \leq \infty$ (but not necessarily analytic). Basically, we will describe the construction presented by Branicky in [Bra95], but following the approach of Campagnolo, Costa, and Moore [CMC00], [CM01], [Cam02]. Then using the map f_M given by Theorem 4.4.2, we will be able to simulate TMs with smooth ODEs. This result will be extended in the next section to the case of polynomial ODEs robust to perturbations.

Suppose that $f : \mathbb{Z}^k \to \mathbb{Z}^k$ is a map. For simplicity, let us assume $k = 1$. For better readability, we also break down the procedure in three subtasks.

Construction 4.5.1. *Consider a point $b \in \mathbb{R}$ (the* target*), some $\gamma > 0$ (the* targeting error*), and time instants t_0 (*departure time*) and t_1 (*arrival time*), with $t_1 > t_0$. Then obtain an IVP (the* targeting equation*) defined with an ODE (4.12), where $f : \mathbb{R}^2 \to \mathbb{R}$, such that the solution y satisfies*

$$|y(t_1) - b| < \gamma \tag{4.13}$$

independently of the initial condition $y(t_0) \in \mathbb{R}$.

Let $\phi : \mathbb{R} \to \mathbb{R}_0^+$ be some function satisfying $\int_{t_0}^{t_1} \phi(t)dt > 0$ and consider the following ODE

$$y' = c(b - y)^3 \phi(t), \tag{4.14}$$

where $c > 0$. There are two cases to consider: (i) $y(t_0) = b$, (ii) $y(t_0) \neq b$. In the first case, the solution is given by $y(t) = b$ for all $t \in \mathbb{R}$ and (4.13) is trivially satisfied. For the second case, note that (4.14) is a separable equation, which gives

$$\begin{aligned}\frac{1}{(b - y(t_1))^2} - \frac{1}{(b - y(t_0))^2} = 2c \int_{t_0}^{t_1} \phi(t)dt \quad &\Longrightarrow \\ \frac{1}{2c \int_{t_0}^{t_1} \phi(t)dt} > (b - y(t_1))^2.&\end{aligned}$$

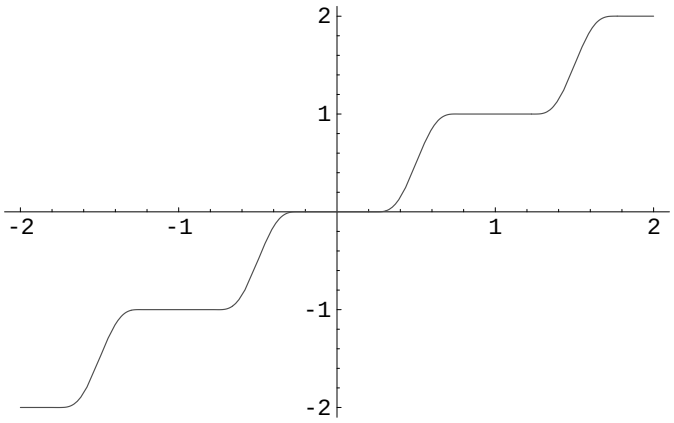


Figure 4.3: Graphical representation of the function r.

Hence, (4.13) is satisfied if c satisfies $\gamma^2 \geq (2c\int_{t_0}^{t_1}\phi(t)dt)^{-1}$ i.e. if

$$c \geq \frac{1}{2\gamma^2\int_{t_0}^{t_1}\phi(t)dt}. \tag{4.15}$$

Construction 4.5.2. *Obtain an IVP defined with an ODE (4.12), where $f:\mathbb{R}^2\to\mathbb{R}$, such that the solution r satisfies (cf. Fig. 4.3)*

$$r(x) = j \quad \textit{whenever } x \in [j-1/4, j+1/4] \textit{ for all } j \in \mathbb{Z}. \tag{4.16}$$

We want a function $r:\mathbb{R}\to\mathbb{R}$ satisfying this condition for the following reason. Suppose that on Construction 4.5.1, $\gamma < 1/4$ and that $b \in \mathbb{N}$. Then $r(y(t_1)) = b$, i.e. r corrects the error present in $y(t_1)$ when approaching an integer value b. This will be useful later in this chapter.

First let $\theta_j:\mathbb{R}\to\mathbb{R}$, $j \in \mathbb{N}-\{0,1\}$, be the function defined by

$$\theta_j(x) = 0 \text{ if } x \leq 0, \quad \theta_j(x) = x^j \text{ if } x > 0.$$

For $j=\infty$ define

$$\theta_j(x) = 0 \text{ if } x \leq 0, \quad \theta_j(x) = e^{-\frac{1}{x}} \text{ if } x > 0.$$

These functions can be seen [CMC00] as a C^{j-1} version of Heaviside's step function $\theta(x)$, where $\theta(x) = 1$ for $x \geq 0$ and $\theta(x) = 0$ for $x < 0$.

With the help of θ_j, we define a "step function" $s:\mathbb{R}\to\mathbb{R}$, that matches the identity function on the integers, as follows:

$$\begin{cases} s'(x) = \lambda_j\theta_j(-\sin 2\pi x) \\ s(0) = 0, \end{cases}$$

where

$$\lambda_j = \int_{1/2}^{1}\theta_j(-\sin 2\pi x)dx > 0.$$

For $x \in [0,1/2]$, $s(x) = 0$ since $\sin 2\pi x \geq 0$. On $(1/2,1)$, s strictly increases and satisfies $s(1) = 1$. Using the same argument for $x \in [j, j+1]$, for all integer j, we conclude that $s(x) = j$ whenever $x \in [j, j+1/2]$. Then defining $r:\mathbb{N}\to\mathbb{N}$ by $r(x) = s(x+1/4)$, it is easy to see that r satisfies the

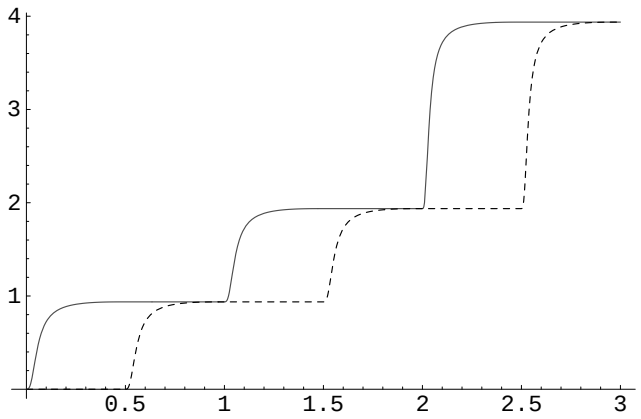


Figure 4.4: Simulation of the iteration of the map $f(n) = 2^n$ via ODEs. The solid line represents the variable z_1 and the dashed line represents z_2.

conditions set for Construction 4.5.2. One should remark that, for each $j \in \mathbb{N} \cup \{\infty\} - \{0,1\}$, we get a different function r, but they all have the same fundamental property (4.16). So, we choose to omit the reference to index j when defining r (this does not represent any problem in later results).

Construction 4.5.3. *Iterate the map $f : \mathbb{Z} \to \mathbb{Z}$ with a smooth ODE (4.12).*

Let $\tilde{f} : \mathbb{R} \to \mathbb{R}$ be an arbitrary smooth extension to the reals of f, and consider the IVP defined with the smooth ODE

$$\begin{cases} z_1' = c_{j,1}(\tilde{f}(r(z_2)) - z_1)^3 \theta_j(\sin 2\pi t) \\ z_2' = c_{j,2}(r(z_1) - z_2)^3 \theta_j(-\sin 2\pi t) \end{cases} \tag{4.17}$$

and the initial condition

$$\begin{cases} z_1(0) = x_0 \\ z_2(0) = x_0, \end{cases}$$

where $x_0 \in \mathbb{N}$. We shall use the previous two constructions to iterate f. First, we use Construction 4.5.1 with parameters satisfying: $\gamma \leq 1/4$, $t_0 = 0$, $t_1 = 1/2$, $\phi = \phi_1$ where $\phi_1(t) = \theta_j(\sin 2\pi t)$, and $c = c_{j,1}$ given by (4.15). With these parameters, let us look to (4.17). We have that for $t \in [0, 1/2]$, $z_2'(t) = 0$. Therefore the first equation of (4.17) becomes

$$z_1' = c(b - z_1)^3 \phi(t),$$

where $b = f(x_0)$. Thus one has $|z_1(1/2) - f(x_0)| < \gamma \leq 1/4$. Now, for $t \in [1/2, 1]$, $z_1'(t) = 0$, and Construction 4.5.2 ensures that $r(z_1(t)) = f(x_0)$ (z_1 "remembers" the value of $f(x_0)$ for $t \in [1/2, 1]$). If we take Construction 4.5.1, but now changing $t_0 = 1/2$, $t_1 = 1$, the function ϕ to $\phi(t) = \phi_2(t) = \theta_j(-\sin 2\pi t)$ and $c = c_{j,2}$ accordingly, the second equation of (4.17) becomes

$$z_2' = c(b - z_2)^3 \phi(t),$$

where $b = f(x_0)$. Hence, one has $|z_2(1) - f(x_0)| < \gamma \leq 1/4$. Now, for $t \in [1, 3/2]$, $z_2'(t) = 0$, and Construction 4.5.2 ensures that $\tilde{f}(r(z_2(t))) = f^{[2]}(x_0)$ (z_2 "remembers" the value of $f(x_0)$ for

$t \in [1, 1+1/2]$). Noting that both $\sin 2\pi t$ and $-\sin 2\pi t$ are periodic with period one, we see that the above procedure can be repeated for all time intervals $[j, j+1]$, where $j \in \mathbb{N}$ (cf. Fig. 4.4). Moreover, one has that for any given $x_0 \in \mathbb{N}$

$$r(z_2(t)) = f^{[j]}(x_0) \text{ whenever } t \in [j, j+1/2]$$

for all $j \in \mathbb{N}$.

In this sense (4.17) simulates the iteration of the function $f : \mathbb{Z} \to \mathbb{Z}$. A straightforward adaptation of this construction can be applied for the more general case when $f : \mathbb{Z}^k \to \mathbb{Z}^k$, for $k \geq 1$. We then obtain an ODE with $2k$ equations, with a pair of equations simulating each component $f_1, \ldots, f_k$ of f.

4.6 Robust simulations of Turing machines with polynomial ODEs

We now adapt the construction presented in the previous section to simulate a TM with polynomial ODEs, even under the influence of perturbations. The idea is to iterate the map f_M given by Theorem 4.4.2 with ODEs, as described in the previous section. The problem is that the ODE must be polynomial and hence analytic, and therefore we can no longer use the functions θ_k for $1 \leq k \leq \infty$. Instead, we have to use a new variation of this construction.

The main idea of the construction to be presented in this section is the following. Let $\psi : \mathbb{N}^3 \to \mathbb{N}^3$ be the transition function of a Turing machine M. If we want to iterate ψ with analytic ODEs, using a system similar to (4.17), we cannot allow z_1' and z_2' to be 0 in half-unit intervals — cf. Corollary 2.5.4. Instead, we allow them to be very close to zero, which will add some errors to the system (4.17). Therefore at time $t = 1$ both variables will have values close to $\psi(x_0)$. But Theorem 4.4.2 shows that there exists some analytic function f_M, robust to errors, that simulates ψ. This allows us to repeat the process an arbitrary number of times, keeping the error under control. We now state the main results of this section.

Theorem 4.6.1. *Let $\psi : \mathbb{N}^3 \to \mathbb{N}^3$ be the transition function of a Turing machine M, under the encoding (4.1) and let $0 < \varepsilon < 1/4$. There is a polynomial $p_M : \mathbb{R}^{m+4} \to \mathbb{R}^{m+3}$, with $m \in \mathbb{N}$, and a constant $y_0 \in \mathbb{R}^m$ such that the ODE $z' = p_M(t,z)$ simulates M in the following sense: for all $x_0 \in \mathbb{N}^3$ and for all $\bar{x}_0 \in \mathbb{R}^3$ satisfying $\|\bar{x}_0 - x_0\|_\infty \leq \varepsilon$, the solution $z(t)$ of the IVP defined by the previous ODE plus the initial condition $(\bar{x}_0, y_0)$, defined for $t_0 = 0$, satisfies*

$$\left\| z_1(j) - \psi^{[j]}(x_0) \right\|_\infty \leq \varepsilon,$$

for all $j \in \mathbb{N}$, where $z \equiv (z_1, z_2)$ with $z_1 \in \mathbb{R}^3$ and $z_2 \in \mathbb{R}^m$.

Indeed, we will prove the following robust version of Theorem 4.6.1.

Theorem 4.6.2. *Given the conditions of Theorem 4.6.1, there is a PIVP function $f_M : \mathbb{R}^7 \to \mathbb{R}^6$ and a constant $y_0 \in \mathbb{R}^3$ such that the ODE $z' = f_M(t,z)$ robustly simulates M in the following sense: for all g satisfying $\|g - f_M\|_\infty < 1/2$, there is $0 < \eta < 1/2$ such that for all $(\bar{x}_0, \bar{y}_0) \in \mathbb{R}^6$ satisfying $\|(\bar{x}_0, \bar{y}_0) - (x_0, y_0)\|_\infty \leq \varepsilon$, the solution $z(t)$ of*

$$z' = g(t,z), \qquad z(0) = (\bar{x}_0, \bar{y}_0)$$

satisfies, for all $j \in \mathbb{N}$ and for all $t \in [j, j+1/2]$,

$$\left\| z_1(t) - \psi^{[j]}(x_0) \right\|_\infty \leq \eta,$$

where $z \equiv (z_1, z_2)$ with $z_1 \in \mathbb{R}^3$ and $z_2 \in \mathbb{R}^3$.

Proof. Let us prove Theorem 4.6.2. For simplicity, and without loss of generality, we consider that the function to be iterated, ψ, is a one-dimensional map as in (4.17). We begin with some preliminary results about the introduction of perturbations in (4.17).

Studying the perturbed targeting equation. (cf. Construction 4.5.1) Because the iterating procedure relies on the basic ODE (4.14), we have to study the following perturbed version of (4.14)

$$z' = c(\overline{b}(t) - z)^3 \phi(t) + E(t), \tag{4.18}$$

where $\left|\overline{b}(t) - b\right| \leq \rho$ and $|E(t)| \leq \delta$. We take the departure time to be $t_0 = 0$ and the arrival time to be $t_1 = 1/2$ as in (4.17). Therefore we must require that $\int_0^{1/2} \phi(t)dt > 0$, where c satisfies (4.15) and $\gamma > 0$ is the targeting error. Let $\overline{z}$ be the solution of this new ODE, with initial condition $\overline{z}(0) = \overline{z}_0$ and let z_+, z_- be the solutions of $z' = c(b+\rho-z)^3\phi(t)+\delta$ and $z' = c(b-\rho-z)^3\phi(t)-\delta$, respectively, with initial conditions $z_+(0) = z_-(0) = \overline{z}_0$. For simplicity denote

$$\begin{aligned} f(t,z) &= c(\overline{b}(t) - z)^3 \phi(t) + E(t) \\ f_+(t,z) &= c(b+\rho-z)^3\phi(t) + \delta \\ f_-(t,z) &= c(b-\rho-z)^3\phi(t) - \delta. \end{aligned} \tag{4.19}$$

We have that for all $(t,x) \in \mathbb{R}^2$,

$$f_-(t,x) \leq f(t,x) \leq f_+(t,x). \tag{4.20}$$

Since $\overline{z}$ is the solution of the ODE $z' = f(t,z)$ and $z_\pm$ are the solutions of the ODEs $z' = f_\pm(t,z)$, all with the same initial condition $\overline{z}(0) = z_+(0) = z_-(0) = \overline{z}_0$, from (4.20) and a standard differential inequality from the basic theory of ODEs (see e.g. [HW95, Appendix T]), it follows that $z_-(t) \leq \overline{z}(t) \leq z_+(t)$ for all $t \in \mathbb{R}$. Now, if we put upper and lower bounds on z_+ and z_-, respectively, we get immediately bounds for $\overline{z}$.

Let us study what happens with z_+. For convenience, let y_+ be the solution of

$$y' = c(b+\rho-y)^3\phi(t) \tag{4.21}$$

(i.e. $y' = f_+(t,y) - \delta$), with initial condition $y_+(0) = \overline{z}_0$. Since $f_+(t,x) > f_+(t,x) - \delta$ for all $(t,x) \in \mathbb{R}^2$, we have similarly to the case of z_-, $\overline{z}$, and z_+, that

$$y_+(t) \leq z_+(t) \quad \text{for all } t \in [0,1/2]. \tag{4.22}$$

We consider two cases:

1. $\overline{z}_0 \leq b+\rho$. Since y_+ is the solution of the targeting equation (4.21), we have from Construction 4.5.1 and (4.22) that

$$b+\rho-\gamma < y_+(1/2) \implies b+\rho-\gamma < z_+(1/2). \tag{4.23}$$

Moreover, since $z_+(0) = \overline{z}_0 \leq b+\rho$, we have that if $z_+(t) \geq b+\rho$ for some $t \in (0,1/2)$, then (4.19) gives $z_+'(t) = f_+(t,z) \leq \delta$. Therefore $z_+(t)$ cannot grow at a rate bigger than δ when its value exceeds $b+\rho$. This and (4.23) give

$$b+\rho-\gamma < z_+(1/2) < b+\rho+\delta/2.$$

2. $\bar{z}_0 > b+\rho$. Since y_+ is solution of the targeting equation (4.21), we have from Construction 4.5.1 and (4.22) that

$$b+\rho < y_+(t) < z_+(t) \text{ for all } t \in [0,1/2].$$

This condition then gives $c(b+\rho-y_+)^3\phi(t)+\delta > c(b+\rho-z_+)^3\phi(t)+\delta$ which together with (4.21) implies

$$\delta > z'_+(t) - y'_+(t) \quad \text{for all } t \in [0,1/2].$$

Integrating the last equation, we have

$$\frac{\delta}{2} > z_+(1/2) - y_+(1/2) \implies \frac{\delta}{2} + y_+(1/2) > z_+(1/2).$$

The latter inequality plus the fact that $b+\rho < y_+(t) < b+\rho+\gamma$, where y_+ is solution of (4.21), yield that

$$b+\rho < z_+(1/2) < b+\rho+\gamma+\frac{\delta}{2}.$$

Now that we have studied both cases $\bar{z}_0 \leq b+\rho$ and $\bar{z}_0 > b+\rho$, we conclude that

$$b+\rho-\gamma < z_+(1/2) < b+\rho+\gamma+\frac{\delta}{2}.$$

A similar analysis can be performed for $z_-(1/2)$, ultimately yielding

$$|\bar{z}(1/2) - b| < \rho+\gamma+\frac{\delta}{2}. \tag{4.24}$$

Removing the θ_j's from (4.17). We must remove the θ_j's in two places: in the function r and in the terms $\theta_j(\pm\sin 2\pi t)$. Since in (4.17) we are using an extension $\tilde{f}:\mathbb{R}\to\mathbb{R}$ (actually, $\tilde{f} \equiv f_M : \mathbb{R}^3 \to \mathbb{R}^3$, but we consider the one-dimensional case for simplicity) of $f:\mathbb{N}\to\mathbb{N}$ ($\equiv \psi:\mathbb{N}^3\to\mathbb{N}^3$) which is robust to perturbations, we no longer need the corrections performed by r. On the other hand we cannot use this technique to treat the terms $\theta_j(\pm\sin 2\pi t)$. We need to substitute $\phi(t)=\theta_j(\sin 2\pi t)$ by an analytic (PIVP) function $\zeta:\mathbb{R}\to\mathbb{R}$ with the following ideal behavior:

(i) ζ has period 1;
(ii) $\zeta(t)=0$ for $t\in[1/2,1]$;
(iii) $\zeta(t)\geq 0$ for $t\in[0,1/2]$ and $\int_0^{1/2}\zeta(t)dt>0$.

Of course, conditions (ii) and (iii) are incompatible due to Proposition 2.5.4. Instead, we approach ζ using a function ζ_ϵ, where $\epsilon>0$. This function must satisfy the following conditions:

(ii)' $|\zeta_\epsilon(t)|\leq\epsilon$ for $t\in[1/2,1]$;
(iii)' $\zeta_\epsilon(t)\geq 0$ for $t\in[0,1/2]$ and $\int_0^{1/2}\zeta_\epsilon(t)dt > I > 0$, where I is independent of ϵ.

Our idea to define such a function ζ_ϵ is to use the function l_2 introduced in Proposition 4.2.5. Then define

$$\zeta_\epsilon(t) = l_2(\vartheta(t), 1/\epsilon), \tag{4.25}$$

where $\epsilon>0$ is the precision up to which ζ_ϵ should approximate 0 in the interval $[1/2,1]$ and $\vartheta:\mathbb{R}\to\mathbb{R}$ is an elementary periodic function of period 1 satisfying the following conditions:

(a) $|\vartheta(t)|\leq 1/4$ for $t\in[1/2,1]$;

(b) $\vartheta(t) \geq 3/4$ for $t \in (a,b) \subseteq (0,1/2)$.

Notice that Proposition 4.2.5 and (a) ensure that $|\zeta_\epsilon(t)| < \epsilon$ for $t \in [1/2,1]$, i.e. they ensure (ii)', and that Proposition 4.2.5 and (b) ensure $|\zeta_\epsilon(t)| > 1-\epsilon$ for $t \in (a,b)$ which gives $\int_0^{1/2} \zeta_\epsilon(t) \geq (1-\epsilon)(b-a) > 3(b-a)/4$ for $\epsilon < 1/4$, which yields (b) (remark that for all $(t,x) \in \mathbb{R}^2$, $l_2(t,x) > 0$ and thus $\zeta_\epsilon(t) > 0$ for all $t \in \mathbb{R}$). It is not difficult to see that one can pick $\vartheta : \mathbb{R} \to \mathbb{R}$ as

$$\vartheta(t) = \frac{1}{2}(\sin^2(2\pi t) + \sin(2\pi t)) \tag{4.26}$$

since it satisfies all the conditions imposed above (e.g. $a = 0.16$ and $b = 0.34$). Hence, $\theta_j(\sin 2\pi t)$ will be replaced by the PIVP function $\zeta_\epsilon(t) = l_2(\vartheta(t), 1/\epsilon)$, where ϑ is given by (4.26). Similarly, $\theta_j(-\sin 2\pi t)$ will be replaced by the PIVP function $\zeta_\epsilon(-t)$.

Performing Construction 4.5.3 with PIVP functions. We are now ready to perform a simulation of an integer map with a system similar to (4.17), but only using PIVP (and hence analytic) functions. Choose a targeting error $\gamma > 0$ such that

$$2\gamma + \delta/2 \leq \varepsilon < 1/4, \tag{4.27}$$

where $\delta = \|g - f_M\|_\infty < 1/2$ is the maximum amplitude of the perturbations that can affect our system of ODEs (we suppose, without loss of generality, that $\delta/2 < \varepsilon$) and take the following system of ODEs

$$\begin{aligned} z_1' &= c_1(f_M \circ \sigma^{[m]}(z_2) - z_1)^3 \zeta_{\epsilon_1}(t), \\ z_2' &= c_2(\sigma^{[n]}(z_1) - z_2)^3 \zeta_{\epsilon_2}(-t) \end{aligned} \tag{4.28}$$

with initial conditions $z_1(0) = z_2(0) = \overline{x}_0$, where σ is the error-contracting function defined in (4.2) and $c_1, c_2, m, n, \epsilon_1$, and ϵ_2 are still to be defined.

We would like that (4.28) satisfies the following property: on $[0,1/2]$,

$$\left|z_2'(t)\right| \leq \gamma. \tag{4.29}$$

This can be achieved by taking $\epsilon_2 = \gamma/K$, where K is a bound for $c_2(\sigma^{[n]}(z_1) - z_2)^3$ in the interval $[0,1]$. Since $|x|^3 \leq x^4 + 1$ for all $x \in \mathbb{R}$, we can take $\epsilon_2 = \gamma/c_2(\sigma^{[n]}(z_1) - z_2)^{-4} + \gamma/c_2$. Now notice that $z_2(0)$ has an error bounded by ε. This plus (4.29) and the fact that z_2' might be subjected to perturbations of amplitude not exceeding δ, imply that

$$|z_2(t) - x_0| \leq \varepsilon + (\delta + \gamma)/2 = \eta < 1/2 \quad \text{for } t \in [0,1/2]. \tag{4.30}$$

Therefore, for m satisfying $\sigma^{[m]}(\eta) < \gamma$, we have that $\left|\sigma^{[m]}(z_2(t)) - x_0\right| < \gamma$ for all $t \in [0,1/2]$. Hence, from the study of the perturbed targeting equation (4.18), where $\phi(t) = \zeta_{\epsilon_1}(t)$ and c_1 is obtained accordingly, we have (take $\rho = \gamma$ and consider (4.27))

$$|z_1(1/2) - \psi(x_0)| < 2\gamma + \frac{\delta}{2} \leq \varepsilon. \tag{4.31}$$

For the interval $[1/2,1]$ the roles of z_1 and z_2 are switched. Similarly to the reasoning done for z_2 on $[0,1/2]$, take $\epsilon_1 = \gamma/c_1(f_M \circ \sigma^{[m]}(z_2) - z_1)^{-4} + \gamma/c_1$ so that on $[0,1/2]$,

$$\left|z_1'(t)\right| \leq \gamma.$$

From this inequality, (4.31), and the fact that z_2' might be subjected to perturbations of amplitude not exceeding δ, we get that

$$|z_1(t) - \psi(x_0)| \leq \varepsilon + (\delta + \gamma)/2 = \eta < 1/2 \quad \text{for } t \in [1/2, 1].$$

Therefore, for $n = m$, we have $\left|\sigma^{[n]}(z_1(t)) - \psi(x_0)\right| < \gamma$ for all $t \in [1/2, 1]$. Hence, from the study of the perturbed targeting equation (4.18), where $\phi(t) = \zeta_{\epsilon_2}(t)$ and c_2 is obtained accordingly, we have

$$|z_2(1) - \psi(x_0)| < 2\gamma + \frac{\delta}{2} \leq \varepsilon.$$

Now we can repeat the procedure for intervals $[1,2]$, $[2,3]$, etc. to conclude that for all $j \in \mathbb{N}$ and for all $t \in [j, j + 1/2]$,

$$\left|z_1(t) - \psi^{[j]}(x_0)\right| \leq \eta.$$

Moreover, z_1 is defined as the solution of an ODE written in terms of PIVP functions. □

As a corollary, we prove Theorem 4.6.1.

Proof of Theorem 4.6.1. From the previous proof, it follows that

$$\left|z_1(t) - \psi^{[j]}(x_0)\right| \leq \eta < 1/2.$$

Let k be an integer such that $\sigma^{[k]}(\eta) < \varepsilon$. Then the function y_1 defined by $y_1 = \sigma^{[k]}(z_1(t))$ is also a PIVP function (see Theorem 3.2.5) satisfying

$$\left|y_1(t) - \psi^{[j]}(x_0)\right| \leq \varepsilon.$$

for all $j \in \mathbb{N}$ and for all $t \in [j, j + 1/2]$. □

CHAPTER 5

The GPAC and Computable Analysis are equivalent models

5.1 Introduction

In this chapter we propose a new definition of computability for the GPAC — one that captures more closely the ideas underlying computable analysis. In that manner we show that functions that are computable according to recursive analysis are also computable in that new model, and vice versa. This clarifies some of the issues discussed previously concerning the (non-) computability of functions like the Gamma function Γ or Euler's Zeta function ζ.

In Section 5.2 we present our new notion of GPAC-computability and state the main result of this chapter: on compact intervals, a function is GPAC-computable (in the previous sense) iff it is computable according to recursive analysis. The proof of this result then follows throughout Sections 5.3, 5.4, and 5.5.

In Section 5.3 we present the proof of the "if" direction: if $f : [a,b] \to \mathbb{R}$ is a GPAC-computable function, then it is computable. In Section 5.4 we present a preliminary theorem that will be necessary to prove the "only if" direction. Namely, we show that a GPAC can simulate an oracle Turing machine in the following sense: given a computable function $f : [a,b] \to \mathbb{R}$, we can design a GPAC with two initial conditions, $x \in [a,b]$ and an approximation $\overline{n}$ of a positive integer $n \in \mathbb{N}$, such that the GPAC outputs an approximation of $f(x)$ with error bounded by 2^{-n}.

Finally, in Section 5.5 we prove the "only if" direction of the main result of this chapter. As a corollary, we conclude that the GPAC is equivalent to Type-2 computability, thus completing the computational characterization of the GPAC given in Chapter 4.

The results of this chapter were partially published in the following references: [BCGH06], [BCGHar].

5.2 The main result

As we had occasion to see in Section 3.2, the Gamma function Γ as well Euler's Zeta function ζ are not PIVP functions, and therefore are not computable by a GPAC, according to Proposition 3.4.5. This becomes a problem if we want to relate the GPAC with computable analysis, since Γ and ζ are computable in the computable analysis framework [PER89]. Thus, it seems that the GPAC is a less powerful model that computable analysis, at least if we restrict the GPAC to

use computable constants, as in Remark 3.4.8. However, as mentioned in [CMC00, pp. 657-658], this comparison is based on two non-equivalent definitions of computability and, therefore, different arguments are needed. The GPAC computes in real time — a very restrictive form of computation, where given one input t to the GPAC, the output $f(t)$ is immediately computed. In contrast, in computable analysis, the computation takes infinite time, but we have at all time partial results that converge to the output.

Following these ideas, it was proved in [Gra04] that if we redefine the notion of GPAC-computability in a manner that matches more closely the essence underlying computable analysis, namely the "converging computation" behavior, then one can compute the Γ function as well as Riemann's ζ function.

Here we further strengthen this result and show that every *computable function* can be computed by a GPAC in the above sense. Reciprocally, we show that under some reasonable assumptions, the converse is also true. These "reasonable assumptions" have to do with the following. The GPAC admits arbitrary and hence noncomputable constants, trivially leading to super-Turing computations. To avoid this problem, we only allow the use of computable constants.

Let us precise our definitions and results.

Definition 5.2.1. *A function $f : [a,b] \to \mathbb{R}$ is GPAC-computable iff there exist some computable polynomials $p : \mathbb{R}^{n+1} \to \mathbb{R}^n$, $p_0 : \mathbb{R} \to \mathbb{R}$, and $n-1$ computable real values $\alpha_1, ..., \alpha_{n-1}$ such that:*

1. *$(y_1, ..., y_n)$ is the solution of the Cauchy problem $y' = p(t,y)$, $y(0) = (\alpha_1, ..., \alpha_{n-1}, p_0(x))$;*
2. *There are $i, j \in \{1, ..., n\}$ such that $\lim_{t\to\infty} y_j(t) = 0$ and $|f(x) - y_i(t)| \leq y_j(t)$ for all $x \in [a,b]$ and all $t \in [0, +\infty)$ (we assume that $y(t)$ is defined for all $t \geq 0$).*

The previous definition basically says the following. A function is GPAC-computable if there are two PIVP functions y_i and y_j, defined with a polynomial IVP which uses computable polynomials and computable initial conditions (except for the initial condition defined with the input x), such that $y_i(t)$ converges to $f(x)$ as $t \to \infty$, with error bounded by $y_j(t)$.

Remark 5.2.2. In the remaining of this chapter, the expression "GPAC-computable" means a function computable by a GPAC in the sense of Definition 5.2.1, while other expressions like "generated by a GPAC" mean functions computed by a GPAC in a standard way (see Sections 3.3 and 3.4), i.e. PIVP functions.

Remark 5.2.3. If we are given a GPAC with initial condition $(\alpha_1, ..., \alpha_{n-1}, p_0(x))$ set at time $t_0 = 0$, where $\alpha_1, ..., \alpha_{n-1}$ are fixed computable values, and where x may vary for each computation, as in Definition 5.2.1, we say that the *initial condition x sets the output of the GPAC*. The initial condition includes $p_0(x)$ and not x because it is sometimes convenient not to use x directly as initial condition, but some value that can be easily obtained from it, e.g. $2x$.

Let us now state the main result of this chapter.

Theorem 5.2.4. *Let $a, b \in \mathbb{R}$ be computable constants satisfying $a < b$. A function $f : [a,b] \to \mathbb{R}$ is computable iff it is GPAC-computable.*

The remaining sections of this chapter are devoted to the proof of this result.

5.3 Proof of the "if" direction

Let $f : [a,b] \to \mathbb{R}$ be a GPAC-computable function. We want to show that f is computable in the sense of computable analysis. By definition, we know that there is a polynomial ODE

$$\begin{cases} y' = p(t,y) \\ y(0) = (\alpha, p_0(x)) \end{cases}$$

whose solution has two components $y_i : \mathbb{R}^2 \to \mathbb{R}$ and $y_j : \mathbb{R}^2 \to \mathbb{R}$ such that

$$|f(x) - y_i(t,x)| \leq y_j(t,x) \text{ and } \lim_{t\to\infty} y_j(t,x) = 0. \tag{5.1}$$

We note that, according to Theorem 6.2.4, to be proved later in Chapter 6, it follows y_i and y_j are computable in $(0,\infty) \times [a,b]$. Suppose that we want to compute $f(x)$ with precision 2^{-n}. Then proceed with the following algorithm.

1. Set $t = 1$
2. Compute an approximation $\overline{y}_j$ of $y_j(t,x)$ with precision $2^{-(n+2)}$
3. If $\overline{y}_j > 2^{-(n+2)}$ then set $t := t+1$ and go to Step 2
4. Compute $y_i(t,x)$ with precision $2^{-(n+1)}$ and output the result

Steps 1, 2 and 3 are used to determine an integer value of t for which $|y_j(t,x)| \leq 2^{-(n+1)}$. Once this value is obtained, an approximation of $y_i(t,x)$ with precision $2^{-(n+1)}$ will provide an approximation of $f(x)$ with error 2^{-n}, due to (5.1), thus providing the desired output.

5.4 Simulating partial computations with GPACs

Before entering the proof of the "only if" direction of Theorem 5.2.4, we present in this section a result that shows that GPACs can simulate oracle Turing machines, in the context of computable analysis, as in Definition 2.4.10.

From Proposition 4.6.1, we know how to simulate a Turing machine. However the error of the output is bounded by some fixed quantity $\varepsilon > 0$ whereas in oracle Turing machines, as in Definition 2.4.10, we would like that the output is given with error bounded by 2^{-n}, where n is one of the inputs of the machine. The next theorem shows how this can be done with a GPAC.

Theorem 5.4.1. *Let $f : [a,b] \to \mathbb{R}$ be a computable function. Then there exists a GPAC with an output y_i such that if we set the initial conditions $(x,\overline{n}) \in [a,b] \times \mathbb{R}$, where $|\overline{n} - n| \leq \varepsilon < 1/2$, with $n \in \mathbb{N}$, then there exists some $T \geq 0$ such that $|y_i(t) - f(x)| \leq 2^{-n}$ for all $t \geq T$.*

Before giving the proof of the theorem, we provide some preliminary lemmas. To compute $f(x)$ with a GPAC, we want to use the hypothesis that f is computable. Hence, it would be useful to construct a GPAC that, when we set the initial condition $x \in [a,b]$, outputs a sequence of rational numbers converging to $p_0(x)$. This sequence could then be used to compute approximations of $f(x)$, as in condition 2 of Proposition 2.4.11. The problem is, given $x \in [a,b]$ and $n \in \mathbb{N}$, to get integers i, j, and 2^k such that $(-1)^i j/2^k$ approximates x enough to compute $f(x)$ with precision 2^{-n}, and to compute the values $sgn(i,j,k,n)$ and $abs(i,j,k,n)$.

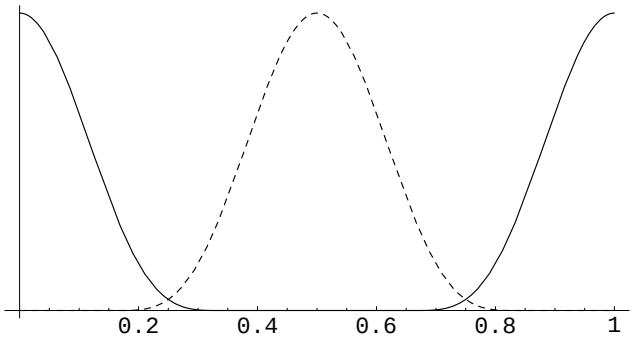


Figure 5.1: Functions ω_1 and ω_2. The solid line represents ω_1, while the dashed line represents ω_2.

We assume first in what follows that $[a,b] \subseteq \mathbb{R}^+$, so that x is always positive. It follows that i can be considered as the constant 0. Now, from Proposition 2.4.11, it is sufficient to take $k = m(n)$ (where m is a modulus of continuity), and $j \simeq x2^{m(n)}$.

In the following lemma, $g(j,n)$ is intended to represent $abs(0,j,m(n),n)$. By a barycenter of $x,y \in \mathbb{R}$ we mean a value of the form $tx+(1-t)y$, for $t \in [0,1]$. By other words, a barycenter is a point on the segment of line joining x to y.

Lemma 5.4.2. *Let $g : \mathbb{N}^2 \to \mathbb{N}$ be a recursive function, $[a,b] \subseteq \mathbb{R}^+$ be a bounded interval, $m : \mathbb{N} \to \mathbb{N}$ be a recursive function, and ε be a real number satisfying $0 < \varepsilon < 1/4$. Then there is a GPAC with the following property: for all $x \in [a,b]$ and all j, $n \in \mathbb{N}$ satisfying $j \leq x2^{m(n)} < j+1$, there exists some $T > 0$ and some index i such that, when we set initial conditions $\overline{n}$, $\overline{x2^{m(n)}}$, where $|n-\overline{n}| < \varepsilon$ and $\left|\overline{x2^{m(n)}} - x2^{m(n)}\right| < \varepsilon$, the output y_i of the GPAC satisfies $|y_i(t) - c| \leq \varepsilon$ for all $t \geq T$, where c is a barycenter of $g(j,n)$ and $g(j+1,n)$.*

Proof. By Theorem 4.6.2 there is a GPAC $\mathcal{G}$ that when set on initial condition $\overline{k}, \overline{n}$, where $\left|k-\overline{k}\right|$, $|n-\overline{n}| < 1/3$ (the reason why we use $1/3$ will be clear later) and $k,n \in \mathbb{N}$, ultimately (i.e. at any time $t \geq T$ for some $T > 0$) outputs $g(k,n)$ with an error less than or equal to $\varepsilon/2$ (k is intended to be j or $j+1$). Define $\overline{k}_1 = \overline{x2^{m(n)}}$. We would like to use $\overline{k} = \overline{k}_1$ as an initial condition to GPAC $\mathcal{G}$. However, $\overline{k}_1$ is not guaranteed to be close to an integer (i.e. within distance $1/3$). We now show how to overcome this. Let us consider two cases:

1. If $\overline{k}_1 \in [l-1/4, l+1/4]$, for some $l \in \{j, j+1\}$, then we can set $\overline{k} = \overline{k}_1$. With this initial condition, the output of GPAC $\mathcal{G}$ (let us call it y_1) will be $g(j,n)$ or $g(j+1,n)$, plus an error not exceeding $\varepsilon/2$. Therefore, the output satisfies the conditions imposed by the lemma. Notice that this reasoning extends for the case $\overline{k}_1 \in [l-1/3, l+1/3]$, because (integer) initial conditions of the GPAC can be perturbed by an amount bounded by $1/3$;

2. If $\overline{k}_1 \in [j+1/4, j+3/4]$, then we can set the initial condition $\overline{k} = \overline{k}_1 - 1/2$. With this initial condition, the output of GPAC $\mathcal{G}$ (let us call it y_2) will be $g(j,n)$ plus an error not exceeding $\varepsilon/2$. Therefore, the output satisfies the conditions imposed by the lemma. Notice that this reasoning extends for the case $\overline{k}_1 \in [l+1/6, l+5/6]$.

The real problem here is to implement both cases in a single GPAC. Since GPACs do not allow the existence of discontinuous functions that might work like a case checker, we have to resort to a different approach.

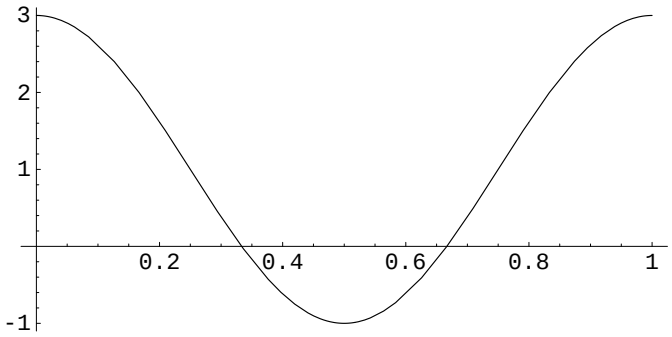


Figure 5.2: Function Υ.

From the study of the previous cases, we know the following: there is a GPAC $\mathcal{G}$ which on initial conditions $\overline{k}_1$ or $\overline{k}_1 - 1/2$, outputs y_1 or y_2, respectively. We now consider a new GPAC, obtained with two copies of $\mathcal{G}$, but where one copy has initial condition set to $\overline{k}_1$, and the other has initial condition set to $\overline{k}_1 - 1/2$ (from Remark 5.2.3, both cases are covered by the expression "each copy has initial condition set to $\overline{k}_1$"). Hence, this GPAC outputs both y_1 and y_2. We now combine these outputs to get the desired result.

Assume we had two periodic functions ω_1 and ω_2, with period 1 and graphs similar to the ones depicted in Fig. 5.1. We do not explicitly define ω_1 and ω_2, but rather state their most important properties: (i) for every $t \in \mathbb{R}$, $\omega_1(t) \geq 0$, $\omega_2(t) \geq 0$, and $\omega_1(t) + \omega_2(t) > 0$, (ii) $\omega_1(t) > 0$ implies that $t \in (a - 1/3, a + 1/3)$ for some $a \in \mathbb{N}$, and (iii) $\omega_2(t) > 0$ implies that $t \in (a + 1/6, a + 5/6)$ for some $a \in \mathbb{N}$. Remark that, for all $a \in \mathbb{N}$, (ii) implies $\omega_1(t) = 0$ for all $t \in (a + 1/3, a + 2/3)$, and (iii) implies $\omega_2(t) = 0$ for all $t \in (a - 1/6, a + 1/6)$. Then, taking into account the previous two cases described above, one sees that we could output the value

$$\overline{y} = \frac{\omega_1(\overline{k}_1) y_1 + \omega_2(\overline{k}_1) y_2}{\omega_1(\overline{k}_1) + \omega_2(\overline{k}_1)}. \tag{5.2}$$

that would be correct in any of the two cases above. Indeed, the only case where both $\omega_1(\overline{k}_1)$ and $\omega_2(\overline{k}_1)$ are nonzero is whenever $\overline{k}_1 \in [l + 2/3, l + 5/6]$, where both outputs y_1 and y_2 are valid, and the result is a barycenter of y_1 and y_2, i.e. a barycenter of $g(j,n)$ and $g(j+1,n)$ plus an error not exceeding $\varepsilon/2$.

However, ω_1 and ω_2 are not GPAC-generable since they are not analytic. Alternatively, we will use closed-form functions that approximate ω_1 and ω_2. In particular, we use the function l_2 defined in Proposition 4.2.5. We also use the periodic function $\Upsilon : \mathbb{R} \to \mathbb{R}$ defined by

$$\Upsilon(x) = 1 + 2 \sin 2\pi(x + 1/4)$$

with period 1 and whose graph is depicted in Fig. 5.2. Notice that for $x \in [1/3, 2/3]$, $\Upsilon(x) \leq 0$, and for $x \in [-1/4, 1/4]$, $\Upsilon(x) \geq 1$. Therefore, we can take

$$\overline{\omega}_1(x) = l_2(\Upsilon(x), 1/\delta) \simeq \omega_1(x), \quad \overline{\omega}_2(x) = l_2(\Upsilon(x - 1/2), 1/\delta) \simeq \omega_2(x)$$

since $|\omega_1(x) - \overline{\omega}_1(x)| \leq \delta$, for $x \in [a + 1/3, a + 2/3]$, where $a \in \mathbb{Z}$, and similarly for ω_2. Moreover, $|\overline{\omega}_1(x) - 1| \leq \delta$ for $x \in [a - 1/4, a + 1/4]$, and similarly for ω_2, which implies that

$$\overline{\omega}_1(t) + \overline{\omega}_2(t) > 1 - 2\delta > 0,$$

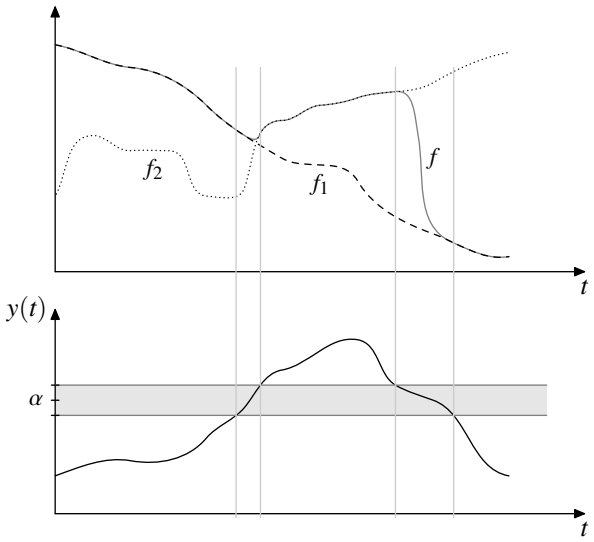


Figure 5.3: Switching functions. Functions f_1 and f_2 are represented in the first graph by the dashed and dotted line, respectively. The resulting function is represented in gray. The second graph displays the control function y, where α is the threshold.

for all $t \in \mathbb{R}$. Now, we just have to substitute (5.2) by

$$\overline{y} \simeq \frac{\overline{\omega}_1(\overline{k}_1)y_1 + \overline{\omega}_2(\overline{k}_1)y_2}{\overline{\omega}_1(\overline{k}_1) + \overline{\omega}_2(\overline{k}_1)}. \tag{5.3}$$

If we pick $1/\delta = \gamma(y_1 + y_2 + 1)$, for some $\gamma > 0$ (the value 1 is to avoid a singularity for $y_1 = y_2 = 0$), we conclude that $\overline{\omega}_1(\overline{k}_1)y_1$ approaches $\omega_1(\overline{k}_1)y_1$ with error bounded by γ, and similarly for the other term. Moreover, $\overline{\omega}_1(\overline{k}_1) + \overline{\omega}_2(\overline{k}_1) > 1 - 2\gamma$. This implies that $\overline{y}$ in (5.3) is computed with error bounded by $2\gamma/(1-2\gamma) < \varepsilon/2$ for γ sufficiently small. By other words, this yields a GPAC with an output y_i such that for some $T > 0$, $|y_i(t) - c| \leq \varepsilon$ for all $t \geq T$, where c is a barycenter of $g(j,n)$ and $g(j+1,n)$. □

In many occasions it will be useful to switch the behavior of a GPAC upon some "control function" $y : \mathbb{R} \to \mathbb{R}$ which is also the output of some GPAC. Ideally, we would like to have the situation pictured in Fig. 5.3, which illustrates a coupled system that behaves like a "switch". There one can see on the above graph two functions f_1 and f_2, generated by GPACs. The graph below represents the control function and a value $\alpha \in \mathbb{R}$ called the threshold value. Then we would like to have a GPAC with output z such that, if $y(t) < \alpha$, then $z(t) = f_1(t)$, and $z(t) = f_2(t)$ otherwise.

Of course, the previous idea cannot be implemented with a GPAC, since we allow immediate transitions between two distinct functions, which would yield a discontinuous function. To remedy that, we allow some transition zone around the threshold value (in gray in the second graph of Fig. 5.3).

The construction of switching functions has to be further relaxed to cope with the fact that only analytic functions can be used. Therefore, function z in the following lemma will just be an approximation of f_1 and f_2 (cf. Corollary 2.5.4).

Lemma 5.4.3 (Switching functions)**.** *Let $y, f_1, f_2 : \mathbb{R} \to \mathbb{R}$ be functions generated by GPACs (y is called the control function) and $\varepsilon > 0$. Then there is a function $f : \mathbb{R} \to \mathbb{R}$ generated by a GPAC with the following property: for all $t \in \mathbb{R}$,*

$$\begin{cases} |f(t) - f_1(t)| \leq \varepsilon \text{ if } y(t) \leq \alpha - 1/4 \\ |f(t) - f_2(t)| \leq \varepsilon \text{ if } y(t) \geq \alpha + 1/4. \end{cases}$$

Proof. This can be done in a quite straightforward way using the function l_2 introduced in Proposition 4.2.5. It suffices to take

$$f = f_1.l_2\left(\alpha + 1/2 - y(t), \frac{f_1 + f_2}{\varepsilon/2}\right) + f_2.l_2\left(y(t) - \alpha + 1/2, \frac{f_1 + f_2}{\varepsilon/2}\right).$$

It is easy to see that f satisfies the given conditions. □

We will mainly use this lemma to switch between dynamics simulating different Turing machines. Suppose that the GPACs $z_1' = p_1(t, z_1)$ and $z_2' = p_2(t, z_2)$ simulate two Turing machines TM_1 and TM_2 as in Theorem 4.6.1, respectively. Then if we have a control function $y_i(t)$, provided by the output of a third GPAC $y' = p(t, y)$, we can build a function f that switches between $f_1 = p_1$ and $f_2 = p_2$, yielding a new system $y'(t) = p(t, y(t))$, $z'(t) = f(t, z(t), y_i(t))$, simulating the transition function of TM_1 and TM_2, according to the value of $y_i(t)$. From Proposition 3.2.5, this corresponds to a GPAC.

A useful application will be to simulate two Turing machines TM_1 and TM_2 working in series, i.e. where the output of TM_1 is to be used as the input of TM_2.

Indeed, when simulating TM_1 with a GPAC as in Theorem 4.6.1 we can suppose that the states are coded by the integers $1, \ldots, m$, where state m corresponds to the halting state, and that there is a variable y_i of the GPAC giving the current state of TM_1 in the simulation with error bounded by $1/4$ for every $t \in [n, n + 1/2]$. Therefore, if we set $\alpha = m - 1/2$ and $y = y_i$ in Lemma 5.4.3, we have a way of switching between the dynamics of a GPACs simulating TM_1 and another simulating TM_2 upon the value of y_i, i.e. depending on whether TM_1 is still running, or already halted.

We can obtain a GPAC having the desired property in the following manner. The initial condition sets the input of TM_1 and this GPAC simulates TM_1 until it halts. When this happens, the variables coding the tape contents have the input of TM_2. Then we switch the evolution law of this GPAC so that now it simulates TM_2, thus giving the desired output (to avoid interference problems, the control function should be given by a separated GPAC that just simulates TM_1 and that stays in a halting configuration after this Turing machine halts).

From the robustness conditions of Theorem 4.6.2, by choosing ε sufficiently small in the lemma above one can ensure that errors will stay controlled at each step, so that a correct simulation of TM_1 and then TM_2 will happen.

In some cases, we will need to switch to a dynamics that sets one variable to some value coded in another variable. This can be done with the following lemma.

Lemma 5.4.4 (Resetting configurations)**.** *Let $\varepsilon \in \mathbb{R}$ satisfy $1/4 \geq \varepsilon > 0$. Let y be some GPAC generated function and $t_0 < t_1$ be some reals.*

There is a polynomial $p : \mathbb{R}^{m+n+1} \to \mathbb{R}^n$, with $n \geq m$, such that the solution of $z' = p(t, z, y)$ with some fixed initial condition at t_0 satisfies $\|(z_1(t_1), \ldots, z_m(t_1)) - k\|_\infty < \varepsilon$, whenever $\|y(t) - k\|_\infty \leq \varepsilon$ for all $t \in [t_0, t_1]$ for some vector $k \in \mathbb{N}^m$.

Proof. This lemma follows from the study carried out in Section 4.6 for the perturbed version of the Construction 4.5.1, where a "target" must be approached. The error done is given by (4.24), which can then be corrected by applying the function σ of Proposition 4.2.1 to the variable(s) approaching k a fixed number of times. The resulting variable is the output of a GPAC. □

Proof of Theorem 5.4.1. For simplicity, let us suppose that $a > 0$ so that we don't have to care about the sign of $x \in [a,b]$. This is not problematic since, if $a < 0$, we can always shift $[a,b]$ by an amount $k \in \mathbb{N}$ such that $a+k > 0$, to define a new computable function $h : [a+k, b+k] \to \mathbb{R}$ satisfying $h(x) = f(x-k)$.

Suppose also that $f(x)$ always takes the same sign for all $x \in [a,b]$. The case where the sign of $f(x)$ switches can be reduced to this one. Indeed, since $[a,b]$ is compact, there is some $l \in \mathbb{Z}$ such that $g(x) = l + f(x) > 0$, for all $x \in [a,b]$. Once we have a GPAC computing $g(x)$, we just have to subtract l to the output to obtain a GPAC computing $f(x)$. So, to fix ideas, let us suppose that $f(x)$ always takes positive values.

Let us now proceed with the proof. Since f is computable, according to Proposition 2.4.11, and previous discussions, there are recursive functions $m : \mathbb{N} \to \mathbb{N}$, abs (the function sgn is no longer needed since $f(x)$ takes positive values) such that given $x \in [a,b]$ and non-negative integers j,n satisfying

$$\left| j/2^{m(n)} - x \right| < 2^{-m(n)}, \tag{5.4}$$

one has

$$\left| \frac{abs(0,j,m(n),n)}{2^n} - f(x) \right| < \frac{2}{2^n}.$$

We will design a GPAC with an output y_i such that, for some $T > 0$, for all $t \geq T$, $y_i(t)$ is always close to

$$\frac{abs(0,j,m(n),n)}{2^n}, \tag{5.5}$$

the error between the two values being bounded by 2^{-n}. This will be sufficient to prove the theorem.

Let us show how we can compute (5.5) with a GPAC. Let TM_0 and TM_1 be Turing machines computing 2^n and $2^{m(n)}$ on input n. From Theorem 4.6.1, there are GPACs simulating these Turing machines. This yields two GPACs with outputs y_1 and y_2, so that on initial condition $\overline{n}$ close to n, one has $|y_1(t) - 2^n| \leq \varepsilon$ and $|y_2(t) - 2^{m(n)}| \leq \varepsilon$ for all $t \geq T_1$, for some T_1. Moreover, since $[a,b]$ is bounded, we can suppose that $|xy_2(t) - x2^{m(n)}| \leq \varepsilon$ (if necessary, apply σ a fixed number of times, independent of n, to y_2). The values $\overline{n}$ and xy_2 can then be used to feed the GPAC described in Lemma 5.4.2, with $g(j,n) = abs(0,j,m(n),n)$. This GPAC $\mathcal{U}_3$ has an output y_3, which, after some time T_2, yields a barycenter of $abs(0,j,m(n),n)$ and $abs(0,j+1,m(n),n)$ plus an error bounded by ε. Since m is a modulus of continuity,

$$|abs(0,j,m(n),n) - abs(0,j+1,m(n),n)| \leq 1.$$

This implies that the output of $\mathcal{U}_3$, let us call it $\overline{abs(j,n)}$, satisfies

$$|y_3(t) - abs(0,j,m(n),n)| \leq 1 + \varepsilon$$

for all $t \geq T_2$. Since $[a,b]$ is bounded, there is some $\eta \in \mathbb{N}$ such that $abs(0,j,m(n),n) \leq 2^n\eta$ for all $n \in \mathbb{N}$ and all j satisfying (5.4) for every $x \in [a,b]$. Therefore

$$\begin{aligned} \left| \frac{y_3(t)}{y_1(t)} - \frac{abs(0,j,m(n),n)}{2^n} \right| &\leq \frac{\varepsilon.abs(0,j,m(n),n) + 2^n(1+\varepsilon)}{2^n(2^n - \varepsilon)} \\ &\leq \frac{\varepsilon\eta + 1 + \varepsilon}{2^n 2^{-1}} \leq \lambda 2^{-n} \end{aligned} \tag{5.6}$$

with $\lambda = 2(\varepsilon\eta + 1 + \varepsilon)$ independent of j,n, for all $t \geq T = \max\{T_1, T_2\}$, thus giving an appropriate approximation of $abs(0,j,m(n),n)/2^n$, with error bounded by $\lambda 2^{-n}$. □

Remark 5.4.5. Notice that after the time T referred to in Theorem 5.4.1, the corresponding GPAC continues to output forever an approximation of $f(x)$ with error bounded by 2^{-n}. This is because, as assumed in Section 4.2, the halting configurations of the Turing machines simulated by the GPAC are fixed points (modulo some error bounded by ε).

5.5 Proof of the "only if" direction

Our idea is the following. From Theorem 5.4.1, we already know how to generate $f(x)$ with precision 2^{-n} with a GPAC $\mathcal{G}$ fed with approximations of n and $x2^{m(n)}$ as initial conditions, say in components y_1 and y_2, respectively. Hence, to get a GPAC with an output converging to $f(x)$, it suffices to implement the previous theorem in a cyclic way: start the computation with $n = 0$. When the computation finishes, increment n, repeat the computation, and so on.

To do so, we need to address several problems. The first one is to know when the computation with the current n is over. Indeed, Theorem 5.4.1 tells us that this happens after some time T, but does not give us any procedure to compute this instant T.

This can be solved by building another GPAC that provides a corresponding control function. Indeed, consider a clocking Turing machine TM_0 that basically simulates all involved Turing machines in the constructions of Theorem 5.4.1 on all possible arguments $x \in [a,b]$ of type $k2^n$, with $k \in \mathbb{Z}$ (that are finitely many). This guarantees that whenever TM_0 terminates on input n, we are sure that all involved Turing machines in the constructions of Theorem 5.4.1 have had enough time to do their computations in GPAC $\mathcal{G}$, and so that the output is correct. The description of TM_0, on input $n \in \mathbb{N}$, is as follows:

1. Compute the first $k_1 \in \mathbb{Z}$ such that $k_1/n \geq a$, and the last $k_2 \in \mathbb{Z}$ such that $k_2/n \leq b$
2. For $i = k_1$ to k_2 simulate the Turing machines involved in the proof of Theorem 5.4.1 on input (i,n).

Observe that Step 1 can be implemented because, by hypothesis, a and b are computable constants. The Turing machine TM_0 can be simulated by a GPAC (independent from GPAC $\mathcal{G}$). The output y_i of this GPAC that encodes the state of TM_0 can be used as a control function. Indeed, whenever it becomes greater than $m_0 - 1/2$, where m_0 is the number of states of TM_0, this means that TM_0 halted, and hence the output of $\mathcal{G}$ is correct. This solves our first problem.

Actually, we need to simulate $\mathcal{G}$ on increasing n. So, more precisely, we consider a Turing machine TM_1 that does the following:

1. Start with $n = 0$
2. Simulate TM_0 with input n
3. Increment n and go to Step 2.

Suppose, without loss of generality, that this Turing machine has m_1 states, where m_1 is the halting state (that is never reached), and $m_1 - 1$ is a special state, only reached in the transition of Step 3 to Step 1. Value $m_1 - 3/2$ can then be used a threshold. Whenever the output encoding the state of TM_1, call it y_i, is higher than $m_1 - 3/2$, we know that the computation of $\mathcal{G}$ is over for the corresponding n.

We can then use Lemmas 5.4.3 and 5.4.4 so that the control function y_i resets the value of y_1 and y_2 to approximations of $n+1$ and $x2^{m(n+1)}$, respectively, and begins a new cycle by allowing again the simulation of $\mathcal{G}$ on these new values for y_1 and y_2.

So far we have seen that while y_1 and y_2 are, respectively, approximations of n and $x2^m(n)$, one of the components of the system, say y_j, approaches $f(x)$ with error 2^{-n}. However, during the following time period, when n is incremented, y_j fluctuates before converging to $f(x)$ with error $2^{-(n+1)}$. Therefore, y_j doesn't match condition 2 of Definition 5.2.1 for all times.

To define a component of the system that converges to $f(x)$ with time, we use the components of the system that encode the values of $abs(0, j, m(n), n)$ and 2^n in each final stage of a computation period. By a computation period we mean the time gap that occurs between corrects results for n and $n+1$. We create a pair of components, say z_1, z_2, that are reset to the values of $abs(0, j, m(n), n)$ and 2^n respectively, at the end of each computation period. This can be done with Lemma 5.4.4 using the control function y_i. More precisely, when y_i is above the threshold, z_1, z_2 approach $abs(0, j, m(n), n)$ and 2^n. During the following computation period, i.e. while y_i is below the threshold, z_1, z_2 are kept approximately constant. This can be done with Lemma 5.4.3 by switching the dynamics from our current system to a GPAC that simulates a Turing machine in a fixed configuration. Again, we use y_i as the control function.

Hence, a sufficient approximation of $f(x)$ is then given by z_1/z_2, and a bound on current error on $f(x)$, as required in Definition 5.2.1, is given by $1/z_2$ both values being valid at any time.

CHAPTER 6

The maximal interval problem

6.1 Introduction

This chapter introduces new results about the degree of computational resources needed to compute the maximal interval of existence for an IVP defined with an ordinary differential equation. In Section 6.2, we give upper bounds for these computational resources, under very mild hypothesis. In particular, we show that if the IVP is solved over an r.e. set (with computable data), then the corresponding maximal interval is also a r.e. set.

Nevertheless, we show in Section 6.3 that, under the previous hypothesis, the maximal interval can be non-recursive. This section shows the result for piecewise linear functions, while Section 6.4 shows the same result, with a different approach, for the stronger case of analytic functions. In Section 6.5, we show that even the partial problem of deciding if the IVP has a bounded maximal interval or not, for the analytic case, is also undecidable. This resulted is also derived for polynomial IVPs is Section 6.6, using techniques from Chapter 4. In particular, we show that this problem is undecidable for polynomial IVP of degree 56.

The results of this chapter were partially published in the following references: [GZB06], [GZB07].

6.2 The maximal interval is r.e. open

In this section, we give a computable counterpart for the fundamental existence-uniqueness theory for the initial-value problems

$$\begin{cases} x' = f(t,x) \\ x(t_0) = x_0 \end{cases} \tag{6.1}$$

presented in Section 2.6. To achieve this purpose, we begin by presenting computable counterparts of the notions involved in that section. Recall (Definition 2.6.1) that a function $f : E \to \mathbb{R}^m$, $E \subseteq \mathbb{R}^l$, is said to be locally Lipschitz on E if it satisfies a Lipschitz condition on every compact set $V \subset E$. The following definition gives a computable analysis analog of this condition.

Definition 6.2.1. *Let $E = \cup_{n=0}^{\infty} B(a_n, r_n) \subseteq \mathbb{R}^l$, where $\overline{B(a_n, r_n)} \subseteq E$, be a r.e. open set. A function $f : E \to \mathbb{R}^m$ is called effectively locally Lipschitz on E if there exists a computable sequence $\{K_n\}$ of positive integers such that*

$$|f(x) - f(y)| \leq K_n |x - y| \text{ whenever } x, y \in \overline{B(a_n, r_n)}.$$

Strictly speaking, it would only be necessary to require in the above definition that the sequence $\{K_n\}$ is computable and formed by positive reals. However, given a (computable) local Lipschitz constant L_n for a compact set, any integer $K_n \geq L_n$ is also a local Lipschitz constant for that set, and therefore there is no loss of generality in assuming that the sequence $\{K_n\}$ in Definition 6.2.1 consists only of integers.

In the rest of the chapter, we shall deal with the case of interest for ODEs, that is, $E \subset \mathbb{R}^{m+1}$ and $f : E \to \mathbb{R}^m$. From now on $E \subseteq \mathbb{R}^{m+1}$ will always denote a r.e. open set, and $\{a_n\}$, $\{r_n\}$, and $B(a_n, r_n)$ are the corresponding sequences in Definition 2.4.4. Also for notational convenience, we will sometimes simply write f for $f(t,x)$ and refer to $t \in \mathbb{R}$ as the first argument and $x \in \mathbb{R}^m$ as the second argument of f. To get a computable counterpart of Proposition 2.6.4, we introduce the following definition.

Definition 6.2.2. *Let $E = \cup_{n=0}^{\infty} B(a_n, r_n) \subseteq \mathbb{R}^l$, where $\overline{B(a_n, r_n)} \subseteq E$, be a r.e. open set. A function $f : E \to \mathbb{R}^m$ is called effectively locally Lipschitz in the second argument if there exists a computable sequence $\{K_n\}$ of positive integers such that*

$$|f(t,x) - f(t,y)| \leq K_n |y - x| \text{ whenever } (t,x),(t,y) \in \overline{B(a_n, r_n)}.$$

Obviously an effectively locally Lipschitz function $f : E \to \mathbb{R}^m$ is also effectively locally Lipschitz on the second argument. We now extend Lemma 2.6.3 to computable functions in the following way.

Theorem 6.2.3. *Assume that $f : E \to \mathbb{R}^m$ is a computable function in $C^1(E)$ (meaning that both f and its derivative f' are computable). Then f is effectively locally Lipschitz on E.*

Proof. Let K_n be an integer greater than or equal to $\max_{x \in \overline{B(a_n, r_n)}} |f'(x)|$. Since f', a_n, r_n are computable, the real number $\max_{x \in \overline{B(a_n, r_n)}} |f'(x)|$ is also computable (notice that, because f' is computable, it has a modulus of continuity that can be used to obtain the maximum over $\overline{B(a_n, r_n)}$ within any preassigned precision). Moreover, we may assume that the sequence $\{K_n\}$ is a computable sequence of positive integers. Now, for any $x, y \in \overline{B(a_n, r_n)}$, let $u = y - x$. Then $x + su \in \overline{B(a_n, r_n)}$ for $0 \leq s \leq 1$ because $\overline{B(a_n, r_n)}$ is a convex set. Define $F : [0,1] \to \mathbb{R}^n$ by $F(s) = f(x + su)$. By the chain rule

$$F'(s) = f'(x + su) \cdot u = f'(x + su) \cdot (y - x).$$

Therefore,

$$|f(x) - f(y)| = |F(1) - F(0)| = \left| \int_0^1 F'(s) ds \right| \leq \int_0^1 |f'(x + su) \cdot (y - x)| ds \leq K_n |x - y|.$$

□

Let us now consider the IVP (6.1) and prove the main result of this section.

Theorem 6.2.4. *Let $E \subseteq \mathbb{R}^{m+1}$ be a r.e. open set and $f : E \to \mathbb{R}^m$ be a computable function that is also effectively locally Lipschitz in the second argument. Let (α, β) be the maximal interval of existence of the solution $x(t)$ of the initial-value problem (6.1), where (t_0, x_0) is a computable point in E. Then (α, β) is a r.e. open interval and x is a computable function on (α, β).*

Proof. We consider the right maximal interval (t_0, β) and prove (t_0, β) is r.e. open and x is computable on it. The same argument applies to the left maximal interval (α, t_0). For simplicity, we assume that E is an open subset of $\mathbb{R}^2$.

Since $\{a_n\}$ and $\{r_n\}$ are computable sequences and f is a computable function on E, both sequences $\{M_n\}$, $M_n = \max_{z\in\overline{B(a_n,r_n)}} |f(z)|+1$, and $\{K_n\}$ as defined in Definition 6.2.2 are computable, where $z=(t,x)$. A ρ-name of z is used as an oracle in computations involving z on oracle Turing machines.

Next we construct three (oracle) Turing machines, denoted as TM_1, TM_2 and TM_3. Let TM_1 be the Turing machine defined as follows: on any input $\{z_k\}_{k\in\mathbb{N}}$, where $\{z_k\}_{k\in\mathbb{N}}$ is a ρ-name of some $z\in E$, TM_1 computes $|z_k-a_n|$ and halts if $|z_k-a_n| < r_n - 2^{-k+1}$. In this case, we say that the machine TM_1 halts at (k,n). Obviously the machine will halt at any given ρ-name of every $z\in E$ and if the machine halts at (k,n), then $B(z,2^{-k})\subset B(a_n,r_n)$. The output of TM_1, on input $\{z_k\}_{k\in\mathbb{N}}$, is the least (first) integer $\langle k,n\rangle$ (cf. (2.1)) such that TM_1 halts at (k,n).

Let TM_2 be the Turing machine defined as follows: on any input ρ-name $\{z_j^*\}_{j\in\mathbb{N}}$ of $z^*=(t^*,x^*)\in E$, positive integers M, K, and L satisfying $\overline{B(z^*,2^{-L})}\subset E$, $\max_{|z-z^*|\le 2^{-L}}|f(z)|+1\le M$, and $|f(z_1)-f(z_2)|\le K|x_1-x_2|$ for all $z_1=(t,x_1), z_2=(t,x_2)\in\overline{B(z^*,2^{-L})}$, TM_2 outputs the solution of the IVP

$$\begin{cases} x' = f(t,x) \\ x(t') = x^* \end{cases}$$

over the time interval $[t^*,t^*+c^*]$, where $c^*=2^{-L}/M$. TM_2 can be constructed by making use of the classical proof of Picard-Lindelöf's theorem. The following construction follows essentially the proof of Theorem 3.1 of [CL55]. Let $c^*=2^{-L}/M$ and $I=[t^*,t^*+c^*]$. Define the successive approximations x_k on I as follows: $x_0(t)=x^*$ and $x_{k+1}(t)=x^*+\int_{t^*}^t f(s,x_k(s))ds$ for $k=0,1,2,\ldots$. By induction on k it can be shown that every x_k exists on I, $x_k\in C^1$ there, and $(t,x_k(t))\in B(z^*,2^{-L})$ for all $t\in I$. Since f is a computable function and both integration and primitive recursion are computable operators, the sequence $\{x_k\}$ of the successive approximations is computable from (t^*,x^*) plus the positive integers M, K and L. Also it can be shown that x_k converges uniformly on I to a continuous limit function x, which is the solution of the given initial-value problem. Moreover, an upper bound for the error in approximating the solution x by the kth approximation x_k is easily calculable by the definition of x_k, and is given by $|x_k(t)-x(t)|\le \frac{M}{K}\frac{(Kc^*)^{k+1}}{(k+1)!}e^{Kc^*}$, $t\in I$. It remains to show how TM_2 works. On any input ρ-name $\{z_j^*\}_{j\in\mathbb{N}}$ of $z^*=(t^*,x^*)\in E$, positive integers M, K and L satisfying the required conditions plus any $n\in\mathbb{N}$ (output accuracy) and any ρ-name of t (input as an oracle), $t\in[t^*,t^*+c^*]$, TM_2 first computes a $k\in\mathbb{N}$ such that $\frac{M}{K}\frac{(Kc^*)^{k+1}}{(k+1)!}e^{Kc^*}\le 2^{-(n+1)}$. Then it computes a rational vector r such that $|r-x_k(t)|\le 2^{-(n+1)}$, and subsequently $|r-x(t)|\le 2^{-n}$.

Let TM_3 be a Turing machine defined as follows: the input to TM_3 is the same as TM_2 and the output of TM_3 is a ρ-name of $(t^*+c^*,x(t^*+c^*))\in E$.

Next we present an algorithm that computes a sequence $\{b_l\}_{l\in\mathbb{N}}$ converging to β from below. Fix a ρ-name $\{z_{0k}\}_{k\in\mathbb{N}}$ of $z_0=(t_0,x_0)$. On input l, set $j=0$. Now input $\{z_{0k}\}$ into TM_1. Let $n=\langle k_0,n_0\rangle$ be the output of TM_1 on $\{z_{0k}\}$. By the construction of TM_1, $B(z_0,2^{-k_0})\subset B(a_{n_0},r_{n_0})$. Next input $\{z_{0k}\}_{k\in\mathbb{N}}, M_{n_0}, K_{n_0}, k_0$ into TM_2 and TM_3. Then TM_2 will output the solution of (6.1) over the time interval $[t_0,t_0+c_0]$, where $c_0=2^{-k_0}/M_{n_0}$. Denote this solution as $x^0:[t_0,t_0+c_0]\to\mathbb{R}^m$. Separately, the machine TM_3 will output a ρ-name $\{z_{1k}\}_{k\in\mathbb{N}}$ for $z_1=(t_0+c_0,x^0(t_0+c_0))\in E$. Now increase j by 1, i.e. set $j=1$. Repeat the above computation on the input $\{z_{1k}\}_{k\in\mathbb{N}}$, i.e. input $\{z_{1k}\}_{k\in\mathbb{N}}$ into TM_1. Let $n=\langle k_1,n_1\rangle$ be the output of TM_1. Next input $\{z_{1k}\}$, M_{n_1}, K_{n_1} and k_1 into TM_2 and TM_3. Then TM_2 will output a sequence of rational polynomials which rapidly converges to the solution $x^1:[t_0+c_0,t_0+c_0+c_1]\to\mathbb{R}^m$ of the problem

$$\begin{cases} x' = f(t,x) \\ x(t_0+c_0) = x^0(t_0+c_0), \end{cases}$$

where $c_1 = 2^{-k_1}/M_{n_1}$. Also separately, the machine TM_3 will output a ρ-name $\{z_{2k}\}_{k\in\mathbb{N}}$ of $z_2 = (t_0 + c_0 + c_1, x^1(t_0 + c_0 + c_1)) \in E$. Now increase j by 1 again, i.e. set $j = 2$. Repeat the computation on the input $\{z_{2k}\}_{k\in\mathbb{N}}$. Halt the computation on input l the first time when $j > l$ and output $b_l = t_0 + c_0 + c_1 + \ldots + c_l$. Since $l \mapsto b_l$ is an input-output function of a Turing algorithm, the increasing sequence $\{b_l\}_{l\in\mathbb{N}}$ of rational numbers is a computable sequence. Also by the uniqueness of the solution of (6.1) and Pour-El/Richards' Patching Theorem [PER89], it follows that the map $x^{b_l} : [t_0, b_l] \to E$, $x^{b_l}(t) = x^j(t)$ if $t_0 + c_0 + \ldots + c_{j-1} \le t \le t_0 + c_0 + \ldots + c_{j-1} + c_j$, $0 \le j \le l$ with $c_{-1} = 0$, is the solution of the initial-value problem (6.1) over the time interval $[t_0, b_l]$ and this solution is computable. Let $O = \cup_{l=0}^{\infty}(t_0, b_l]$. To complete the proof, we need to show that (a) O is the right maximal interval of existence of the solution of (6.1); (b) O is r.e. open; and (c) x is computable on O. For simplicity, we take $t_0 = 0$.

To prove (a), assume that O is not the maximal interval of existence of the solution of the problem (6.1). Then (6.1) has a solution on an interval $(0,\beta)$ with $O \subsetneq (0,\beta)$. Choose $\beta^* \in (0,\beta) \setminus O$. Then $O \subsetneq (0,\beta^*] \subsetneq (0,\beta)$. Since $\{b_l\}_{l\in\mathbb{N}}$ is a monotonically increasing sequence bounded above by β^*, it converges to a limit γ less than or equal to β^*. Since $\gamma \in (0,\beta)$, $(\gamma, x(\gamma))$ must lie in E. Then there exists an n such that $(\gamma, x(\gamma)) \in B(a_n, r_n)$. Moreover, there is also an integer M such that $(\gamma, x(\gamma)) \in B((\gamma, x(\gamma)), 2^{-M+2}) \subset B(a_n, r_n)$. Since $x : [0,\beta^*] \to \mathbb{R}^m$ is continuous and $b_l \to \gamma$ as $l \to \infty$, it follows that $x(b_l) \to x(\gamma)$ as $l \to \infty$. Consequently there exists an integer N such that

$$(b_N, x(b_N)) \in B((b_N, x(b_N)), 2^{-M+1}) \subset B((\gamma, x(\gamma)), 2^{-M+2}) \subset B(a_n, r_n)$$

and

$$b_N + \min_{0\le\langle i,j\rangle\le\langle M+2,n\rangle} 2^{-i}/M_j > \gamma.$$

We observe that for any ρ-name $\{y_k\}$ of $(b_N, x(b_N))$,

$$\begin{aligned} |y_{M+2} - a_n| &\le |y_{M+2} - (b_N, x(b_N))| + |(b_N, x(b_N)) - a_n| \\ &< 2^{-(M+2)} + r_n - 2^{-M+1} \\ &= r_n - 2^{-(M+2)+1}(4 - 1/2) < r_n - 2^{-(M+2)+1}. \end{aligned}$$

By the construction of the machine TM_1 it follows that on any ρ-name of $(b_N, x(b_N))$ given as input, TM_1 will halt no later than $\langle M+2, n\rangle$. Thus $b_{N+1} = b_N + c_{N+1}$, where $c_{N+1} = 2^{-i}/M_j$ for some $\langle i, j\rangle$ less than or equal to $\langle M+2, n\rangle$, which implies that $b_{N+1} > \gamma$. There is a contradiction because $\{b_l\}_{l\in\mathbb{N}}$ is an increasing sequence converging to γ. This completes the proof of (a).

We now prove (b). Since by Proposition 2.6.4 the maximal interval of existence is open, then O is an open interval. It is also independent of the choice of ρ-names of x_0. By the proof of (a), it follows that $O = \cup_{l=0}^{\infty}(0, b_l] = \cup_{l=0}^{\infty}(0, b_l)$. Since $\{b_l\}_{l\in\mathbb{N}}$ is a computable sequence of rational numbers, O is r.e. open by definition.

Finally we prove (c), that is, x is computable on O. Recall that $O = \cup_{l=0}^{\infty}(0, b_l)$ and $\{b_l\}_{l\in\mathbb{N}}$ is a computable sequence of rational numbers. For any $t \in O$, to compute $x(t)$, we first compute an $l \ge 0$ such that $t < b_l$ and then compute $x^{b_l}(t)$. By definition, $x(t) = x^{b_l}(t)$. □

We mention that the above proof is effective in the sense that given (E, f, t_0, x_0), one can compute (α, β, x), β from below and α from above, i.e. one can compute a sequence of rationals that converges to β from below and a sequence of rationals that converges to α from above. However, the rate of the convergence might not be computable, i.e. α and β can be noncomputable, as we show in the next section.

6.3 ... But not recursive

In this section, we present some noncomputability results concerning ODEs. In particular, we show that for the initial-value problem (6.1) defined by the computable data f and (t_0, x_0), the maximal interval may be noncomputable. We will present two versions of this result; the proof methods are different and the results are interesting on their own. In the first case, where only continuity is required, we can explicitly construct f with a finite expression on bounded domains. For that reason, we prove a preliminary lemma. The second result is for the stronger case where f is analytic, but lacks the finiteness feature: the function f is defined as a (computable) power series. The latter case will be studied in the next section.

Lemma 6.3.1. *Let $a : \mathbb{N} \to \mathbb{N}$ be a computable function. Then there exists a computable and effectively locally Lipschitz function $f : \mathbb{R} \to \mathbb{R}$ such that the unique solution of the problem*

$$\begin{cases} x' = f(x) \\ x(0) = 0 \end{cases} \tag{6.2}$$

is defined on a maximal interval $(-\alpha, \alpha)$ with

$$\alpha = \sum_{i=0}^{\infty} \frac{1}{2^{a(i)}}.$$

Proof. We only need to construct the function f. The idea is as follows: f is constructed piecewisely on intervals of the form $[i, i+1]$, $i \in \mathbb{N}$ (for negative values, we take $f(x) = f(|x|)$) in such a way that, for a fixed i, the solution of the initial-value problem

$$x' = f(x),\ x(0) = i \tag{6.3}$$

satisfies $x(2^{-a(i)}) = i+1$, which implies that the solution of the problem $x' = f(x)$ and $x(0) = 0$ will satisfy $x(2^{-a(0)}) = 1$, $x(2^{-a(0)} + 2^{-a(1)}) = 2$, ..., or more generally

$$x\left(\sum_{i=0}^{n} 2^{-a(i)}\right) = n+1, \quad \text{for all } n \in \mathbb{N}.$$

Notice that f does not depend on t and therefore the solution is invariant under time translations. If we take $\alpha = \sum_{i=0}^{\infty} 2^{-a(i)}$, then $x(t) \to \infty$ as $t \to \alpha^-$. For $t < 0$, since we required $f(x) = f(|x|)$, (6.3) implies that $x(t) \to -\infty$ as $t \to -\alpha^+$. Therefore the maximal interval must be $(-\alpha, \alpha)$.

We now construct the desired function f on intervals of the form $[i, i+1]$, $i \in \mathbb{N}$. Since f must be continuous, we need to glue the values of f at the endpoints of these intervals. This is achieved by assuming that $f(i) = 1$ for all $i \in \mathbb{N}$ (in principle, the value of 1 is rather arbitrary; however some singularities may arise when we consider other values, e.g. 0).

The function f is defined on each interval $[i, i+1]$ as suggested by Fig. 6.1: f is piecewise linear and consists of three line segments, which meet at points $x_i = x(t_{i,1})$ and $y_i = x(t_{i,2})$, with $0 < t_{i,1} < t_{i,2} < 2^{-a(i)}$ and $i = x(0) < x_i < y_i < i+1$. The points x_i and y_i are to be defined. For the moment let us assume that $a(i) \geq 1$. The more general case $a(i) \geq 0$ will be dealt with later in the proof. We now define the function f on the interval $[i, i+1]$ as follows:

$$f(x) = \begin{cases} 1 + (x-i)2^{a(i)}/(x_i - i) & \text{if } x \in [i, x_i) \\ 1 + 2^{a(i)} & \text{if } x \in [x_i, y_i) \\ 1 + 2^{a(i)} - (x - y_i)2^{a(i)}/(i+1-y_i) & \text{if } x \in [y_i, i+1) \end{cases}$$

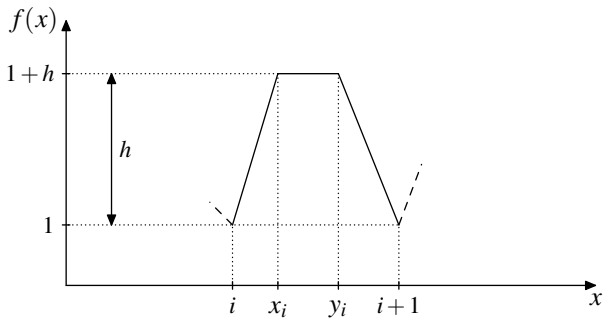


Figure 6.1: Sketch of the function f on the interval $[i, i+1]$, for $i \in \mathbb{N}$.

i.e. assume that $h = 2^{a(i)}$ in Fig. 6.1. Supposing that x_i and y_i are equidistant from i and $i+1$, respectively, and assuming that the solution $x(t)$ of the initial-value problem $x' = f(x)$, $x(0) = i$ satisfies $x(2^{-a(i)}) = i+1$, then one obtains the following values for x_i and y_i:

$$x_i = i + \frac{1-\Delta_i}{2}, \quad y_i = i + \frac{1+\Delta_i}{2},$$

where

$$0 < \Delta_i = \frac{2^{-a(i)} - 2^{-a(i)} \ln(2^{a(i)}+1)}{(1+2^{a(i)})^{-1} - 2^{-a(i)} \ln(2^{a(i)}+1)} < 1.$$

It remains to treat the general case where $a(i) \geq 0$. This can be easily done as follows. First define the recursive function $a^* : \mathbb{N} \to \mathbb{N}$ by $a^*(i) = a(i) + 1$. Using the previous result, we can construct an IVP $x' = f(x)$, $x(0) = 0$ with maximal interval $(-\alpha^*, \alpha^*)$, where

$$\alpha^* = \frac{1}{2}\alpha, \quad \alpha = \sum_{i=0}^{\infty} \frac{1}{2^{a(i)}}.$$

Thus, if in the previous problem time is slowed down by a linear factor of $1/2$, i.e. the change of independent variables $\tilde{t} = t/2$ is performed, we arrive at an initial-value problem $x' = f(x)/2$, $x(0) = 0$ whose maximal interval of existence is $(-\alpha, \alpha)$. □

Theorem 6.3.2 (continuous case)**.** *There exists a continuous computable and effectively locally Lipschitz function $f : \mathbb{R} \to \mathbb{R}$ such that the unique solution of the problem*

$$\begin{cases} x' = f(x) \\ x(0) = 0 \end{cases}$$

is defined on a noncomputable maximal interval.

Proof. From Proposition 2.4.3, if $a : \mathbb{N} \to \mathbb{N}$ is a one to one recursive function generating a recursively enumerable nonrecursive set A, then $\alpha = \sum_{i=0}^{\infty} 2^{-a(i)}$ is a noncomputable real number. Consequently, the open interval $(-\alpha, \alpha)$ is noncomputable. The theorem now follows immediately from the previous lemma. □

The function f in Theorem 6.3.2 can be constructed so that f is of class C^∞ and all its derivative are computable functions (just "smooth" the "corners"). This condition matches the assumption set down in Theorem 6.2.4. Thus, Theorem 6.2.4 gives rise to the best possible result concerning computability of a maximal interval for smooth functions.

6.4 Analytic case

We now show that the result of Section 6.3 can be strengthened to cover the case of computable analytic functions.

Lemma 6.4.1. *Let $a : \mathbb{N} \to \mathbb{N}$ be a one to one recursive function generating a recursively enumerable nonrecursive set. Then there is a computable analytic function φ with the following properties:*

1. *φ is defined on $(-\alpha,\alpha)$, where $\alpha = \sum_{i=0}^{\infty} 2^{-a(i)}$ is a noncomputable real (cf. Proposition 2.4.3);*
2. *$\varphi(x) \to \pm\infty$ as $x \to \pm\alpha^{\mp}$;*
3. *$\varphi : (-\alpha,\alpha) \to \mathbb{R}$ is odd and bijective.*

Proof. Define φ as

$$\varphi(x) = \sum_{n=0}^{\infty} a_n x^n, \quad a_n = \begin{cases} \left(\sum_{i=0}^{n} 2^{-a(i)}\right)^{-n} & \text{if } n \text{ is odd} \\ 0 & \text{if } n \text{ is even} \end{cases}$$

The radius of convergence of this function is given by

$$R = \frac{1}{\overline{\lim}_{n\to\infty} \sqrt[n]{a_n}} = \sum_{i=0}^{\infty} 2^{-a(i)} = \alpha.$$

Moreover, one has $a_{2n+1} > 1/\alpha^{2n+1}$, which in turn implies

$$\varphi(x) = \sum_{n=0}^{\infty} a_{2n+1} x^{2n+1} > \sum_{n=0}^{\infty} \left(\frac{x}{\alpha}\right)^{2n+1}.$$

Therefore $\varphi(x) \to +\infty$ as $x \to \alpha^-$. Since φ is odd by construction, it follows that $\varphi(x) \to -\infty$ as $x \to -\alpha^+$. Note also that

$$\varphi'(x) = \sum_{n=0}^{\infty} (n+1)\, a_{n+1} x^n \tag{6.4}$$

and thus $\varphi'(x) > 0$ for all $x \in (-\alpha,\alpha)$ (all coefficients $(n+1)\,a_{n+1}$ are nonnegative, and only even powers have nonzero coefficients). This implies that φ is injective and therefore bijective according to condition (2) of the statement. It also follows from (6.4) and our choice of a_n that φ' is strictly increasing on $[0,\alpha)$ and, since φ' is even, decreasing on $(-\alpha,0]$.

It remains to show that φ is computable. Assume, without loss of generality, that $x \geq 0$. Since $a : \mathbb{N} \to \mathbb{N}$ is computable by assumption, there is a TM that, for any input $k \in \mathbb{N}$ (output precision) and any $x \in (-\alpha,\alpha)$ with $x \geq 0$, computes first a rational number $\varepsilon > 0$ satisfying $0 \leq x < \alpha - \varepsilon$, then an $n(k) \in \mathbb{N}$ satisfying $\sum_{i=0}^{n(k)} 2^{-a(i)} > x + \varepsilon$ and $\left(\frac{x}{x+\varepsilon(x)}\right)^{n(k)} \frac{(x+\varepsilon(x))^2}{(x+\varepsilon(x))^2 - x^2} < 2^{-k-1}$. We observe that

$$\left|\varphi(x) - \sum_{i=0}^{n(k)} a_i x^i\right| = \sum_{i=n(k)+1}^{\infty} a_i x^i = \sum_{i=\left\lceil \frac{n(k)}{2} \right\rceil}^{\infty} \left(\frac{x}{\sum_{j=0}^{2i+1} 2^{-a(j)}}\right)^{2i+1}$$

$$\leq \sum_{i=\left\lceil \frac{n(k)}{2} \right\rceil}^{\infty} \left(\frac{x}{x+\varepsilon(x)}\right)^{2i+1} \leq \left(\frac{x}{x+\varepsilon(x)}\right)^{n(k)} \frac{(x+\varepsilon(x))^2}{(x+\varepsilon(x))^2 - x^2} \leq 2^{-k-1}.$$

Then, if our TM computes a rational r_k satisfying

$$\left| r_k - \sum_{i=0}^{n(k)} a_i x^i \right| \leq 2^{-k-1}$$

one concludes that $|r_k - \varphi(x)| \leq 2^{-k}$. Thus φ is computable. □

Theorem 6.4.2. *There exists an analytic computable function $f : \mathbb{R} \to \mathbb{R}$ such that the unique solution of the problem*

$$x' = f(x),\ x(0) = 0$$

is defined on a noncomputable maximal interval.

Proof. Define the function φ as in the previous lemma. By the lemma, $\varphi'(x) > 0$ for all $x \in (-\alpha, \alpha)$, and consequently φ^{-1} exists over $\mathbb{R}$. Denote $\psi = \varphi^{-1}$. Then

$$\psi'(x) = \frac{1}{\varphi'(\psi(x))} > 0 \quad \text{for all } x \in \mathbb{R}. \tag{6.5}$$

Similarly, $\varphi'(x) = (\psi'(\varphi(x)))^{-1}$ for all $x \in (-\alpha, \alpha)$, and therefore φ is the solution of the IVP

$$\begin{cases} y' = f(y) \\ y(0) = 0 \end{cases}$$

where $f : \mathbb{R} \to \mathbb{R}$ is defined by $f(x) = 1/\psi'(x)$ (note that f is defined for all real numbers due to (6.5)). Since ψ (and thus ψ') is analytic and computable (see e.g. [Wei00]), so is f. □

6.5 Boundedness is undecidable

In the previous two sections, it is shown that, in general, given an IVP (6.1) and $t > t_0$, we cannot devise an algorithm that tells us how close we are from one of the endpoints of the maximal interval. Nevertheless, this does not rule out the existence of an algorithm that can determine some partial information about the maximal interval. A further question of interest is if there exists an algorithm that can decide whether a given analytic IVP has a bounded maximal interval. As we now show, the answer to this question is also negative.

Theorem 6.5.1. *Given an IVP (6.1) with maximal interval (α,β), where f is analytic, f and (t_0,x_0) are computable, there is no effective (i.e. computable) procedure to determine whether $\beta < \infty$ or $\beta = \infty$.*

Proof. Suppose that there is an effective procedure that determines whether $\beta < \infty$ or $\beta = \infty$, i.e. there is a Turing machine M that with input $\langle f, t_0, x_0 \rangle$ returns 1 if $\beta < \infty$ and 0 otherwise (f, t_0, x_0 mean the canonical encodings of the machines computing f, t_0, and x_0, respectively; see [Wei00] for further details), where $\langle f, t_0, x_0 \rangle$ is the data defining the IVP $x' = f(t,x)$ and $x(t_0) = x_0$. Consider the following undecidable problem (cf. Proposition 2.3.6): "Let $\psi : \mathbb{N}^2 \to \mathbb{N}$ be the function generated by an universal Turing machine. Then, given $i \in \mathbb{N}$, decide if $\psi(i,i)$ is defined". Let M_1 be a Turing machine that computes ψ. Define the recursive function $g : \mathbb{N}^2 \to \mathbb{N}$ by

$$g(i,j) = \begin{cases} 0 & \text{if } M_1 \text{ halts with input } (i,i) \text{ in } \leq j \text{ steps} \\ 1 & \text{otherwise.} \end{cases}$$

Note that

$$\psi(i,i) \text{ is defined} \quad \text{iff} \quad \exists j_0 \in \mathbb{N} \quad (\forall j \geq j_0,\ g(i,j) = 0). \tag{6.6}$$

Next consider the sequence of functions $\{\varphi_i\}$, where $\varphi_i : \mathbb{R} \to \mathbb{R}$ is defined by

$$\varphi_i(x) = \sum_{n=1}^{\infty} a_{i,2n}\, x^{2n}, \quad \text{where } a_{i,n} = \left(\frac{1}{3}\right)^{n^2} + \left(\frac{1}{3}\right)^{n} g(i,n).$$

Following arguments similar to those of Section 6.4, one concludes that: (i) φ_i is analytic and computable, (ii) the sequence $\{\varphi_i\}$ is computable by an oracle machine N since $g : \mathbb{N}^2 \to \mathbb{N}$ is recursive, (iii) the radius of convergence of φ_i is $+\infty$ iff $\psi(i,i)$ is defined, (iv) we can design a Turing machine M_2 in the following way: on input $i \in \mathbb{N}$, M_2 computes $\varphi_i' \circ U_1^2$ and then runs M on the input $\langle \varphi_i' \circ U_1^2, 0, 0\rangle$ (note that, given some $i \in \mathbb{N}$, the "code" for φ_i' can be obtained from the "code" of φ_i that, in turn, can be obtained from the "code" of the Turing machine N), where φ_i' is the derivative of φ_i and $U_1^2 : \mathbb{R}^2 \to \mathbb{R}$ is the projection function defined by $U_1^2(t,x) = t$.

Recall that $\langle \varphi_i' \circ U_1^2, 0, 0\rangle$ is the data defining the IVP $x' = \varphi_i'(t)$ and $x(0) = 0$. Then by (6.6), (iii), and the design of M_2, we arrive at the following conclusion

$$M_2 \text{ on input } i \text{ outputs } \begin{cases} 0 & \text{if } \psi(i,i) \text{ is defined} \\ 1 & \text{otherwise} \end{cases}$$

i.e. M_2 decides an undecidable problem, and we have a contradiction. □

6.6 Boundedness for the polynomial case

Here we sharpen the result of the previous section to the case of polynomial ODEs i.e., when the components of f are polynomial functions. In particular, we would like to know for which degree of the polynomials the boundedness problem is decidable or not. We begin with a simple result, which states that the boundedness problem is decidable for linear IVPs.

Theorem 6.6.1. *Consider the IVP (6.1) with $f(t,x) = A(t,x)x + h(t)$, where A and h are $m \times m$ and $m \times 1$ matrices, respectively, and each entry $A_{jk} : \mathbb{R} \to \mathbb{R}$, $h_j : \mathbb{R} \to \mathbb{R}$ is a continuous function, for $j,k = 1,\ldots,m$. Then the maximal interval associated to this IVP is $(-\infty,\infty)$. In particular, the boundedness problem is decidable for linear problems.*

Proof. See [Hal80, p. 79]. □

Concerning undecidability, we will prove the following theorem.

Theorem 6.6.2. *There is a vector $p : \mathbb{R}^{m+1} \to \mathbb{R}^m$ constituted by polynomials, where each component has degree less than or equal to 56, such that the following problem is undecidable: "Given $(t_0, x_0) \in \mathbb{R} \times \mathbb{R}^m$, decide whether the maximal interval of the IVP*

$$\begin{cases} x' = p(t,x) \\ x(t_0) = x_0 \end{cases} \tag{6.7}$$

is bounded or not".

Corollary 6.6.3. *Let $\mathbb{R}_n[x]$ be the class of all polynomials having degree less than or equal to $n \in \mathbb{N}$. Then, for every $n \geq 56$, the following problem is undecidable: "Given $p : \mathbb{R}^{m+1} \to \mathbb{R}^m$ with components in $\mathbb{R}_n[x]$ and $(t_0, x_0) \in \mathbb{R} \times \mathbb{R}^m$, decide whether the maximal interval of the IVP (6.7) is bounded or not".*

Proof. If the previous problem is decidable, then using p as the function introduced in Theorem 6.6.2, we conclude that the problem defined there is also decidable, which is absurd (if $n > 56$, just add a "dummy" equation of degree n, e.g. $y_p' = t^n$). □

A similar proof applies for the next corollary.

Corollary 6.6.4. *The following problem is undecidable: "Given $p : \mathbb{R}^{m+1} \to \mathbb{R}^m$ with polynomial components and $(t_0, x_0) \in \mathbb{R} \times \mathbb{R}^m$, decide whether the maximal interval of the IVP (6.7) is bounded or not".*

The rest of this section will be devoted to prove Theorem 6.6.2. The following lemma is a result from [Rog96].

Lemma 6.6.5 ([Rog96])**.** *There exists a universal Turing machine with* 4 *states and* 6 *symbols.*

Lemma 6.6.6. *The Turing machine of Lemma 6.6.5 can be simulated by an IVP defined by a set of polynomial ODEs of degree at most* 56 *in the following manner (cf. Section 4.6):*

1. *The input of the TM is coded in the initial conditions of the IVP, through one or more variables of the IVP;*
2. *Let $q(k) \in \{1,2,3,4\}$ denote the state of the TM at step k. Then there is one variable y_q of the IVP that approximates $q(k)$ with error bounded by $5/16$ in each time interval $[k, k+1/2]$, where $k \in \mathbb{N}$;*
3. *During each time interval $[k+1/2, k+1]$, with $k \in \mathbb{N}$, the variable y_q only takes values between the previous state $q(k)$ and the next state $q(k+1)$, plus an error bounded by $5/16$.*

Proof. The simulation of the Turing machine is done essentially as described in Section 4.6. Note that the IVP obtained there can be reduced to a polynomial IVP with the help of Theorem 3.2.5. Since the entire procedure is constructive, it is possible to determine the degrees of the polynomials appearing in the IVP.

In this setting, the variable y_q is simply the component z_2 of (4.28), where f_M is the component of the transition function that gives the next state. The value $5/6$ is a consequence of the choices for the errors allowed in the system, which we analyze later on this proof.

In order to optimize our results, we use some simple adaptations on the results of Section 4.6. In particular, since our TM uses only 6 symbols and not 10, we change the encoding (4.1) to base 6 (i.e. in the equations defining y_1 and y_2, substitute each power 10^j by 6^j). We set the error $\varepsilon = 1/4$ in the construction presented in Theorem 4.4.1. Also, it is possible to reduce the number of terms in the function ω defined in (4.3) by considering the trigonometric interpolation only for $i = 0, \ldots, 5$. We then have

$$\omega(x) = a_0 + a_3 \cos(\pi x) + \left(\sum_{j=1}^{2} a_j \cos\left(\frac{j\pi x}{5}\right) + b_j \sin\left(\frac{j\pi x}{5}\right) \right),$$

where $a_0, a_1, a_2, a_3, b_1, b_2$ are computable coefficients, and $\zeta_\varepsilon = 0.0735$ in (4.4) (substitute the 10 there by 6).

Then the variables set in the proof of Theorem 4.4.1 have the following value: $l = 1$ in (4.9),

$$\overline{q}_{next} = \sum_{i=0}^{5} \sum_{j=1}^{4} \left(\prod_{\substack{r=0 \\ r \neq i}}^{5} \frac{(\sigma^{[14]}(\overline{y}) - r)}{(i-r)} \right) \left(\prod_{\substack{s=1 \\ s \neq j}}^{4} \frac{(\sigma^{[14]}(\overline{q}) - s)}{(j-s)} \right) q_{i,j}, \tag{6.8}$$

$\overline{s}_{next}$ has a similar equation (the value $\sigma^{[14]}$ still works) and also $\overline{h}_{next}$ (but where $\sigma^{[14]}$ is changed to $\sigma^{[13]}$); $\overline{y}_1^{next}$ is still given by (4.10), but with the following adaptations:

$$\begin{aligned}
&\overline{P}_1 = 6(\sigma^{[3]}(\overline{y}_1) + \sigma^{[3]}(\overline{s}_{next}) - \sigma^{[3]}(\overline{y})) + \sigma^{[2]} \circ \omega \circ \sigma(\overline{y}_2), \\
&\overline{P}_2 = \sigma^{[2]}(\overline{y}_1) + \sigma^{[2]}(\overline{s}_{next}) - \sigma^{[2]}(\overline{y}), \\
&\overline{P}_3 = \tfrac{1}{6}\left(\sigma(\overline{y}_1) - \sigma(\overline{y})\right), \\
&H_1 = l_3(\overline{h}_{next}, 6000(\overline{y}_1 + 1/4) + 2);
\end{aligned} \tag{6.9}$$

$\overline{y}_2^{next}$ is also given by (4.11), but with the following adaptations:

$$\begin{aligned}
&H_2 = l_3(\overline{h}_{next}, 6000(\overline{y}_2 + 1/4) + 2), \;\; \overline{Q}_3 = 6\sigma^{[2]}(\overline{y}_2) + \sigma^{[2]}(\overline{s}_{next}), \\
&\overline{Q}_1 = \frac{\sigma(\overline{y}_2) - \sigma \circ \omega \circ \sigma(\overline{y}_2)}{6}, \quad \overline{Q}_2 = \sigma^{[2]}(\overline{y}_2).
\end{aligned} \tag{6.10}$$

Remark that in the proof of Theorem 4.4.1, the arguments of $\overline{q}_{next}$, $\overline{y}_1^{next}$, and $\overline{y}_2^{next}$ are given by a triple $(\overline{q}, \overline{y}_1, \overline{y}_2)$ i.e. the first argument of $\overline{q}_{next} : \mathbb{R}^3 \to \mathbb{R}$ in equation (6.8) is $\overline{q}$, the second is $\overline{y}_1$, and so on.

Using the equations (4.28) and the results of Section 4.6, and using the error bounds $\varepsilon = 1/4$, $\gamma = 1/8$, and $\delta = 0$, one can see that the Turing machine can be simulated by the following system of ODEs:

$$\begin{cases}
z_1' = c_1(\overline{q}_{next} \circ \sigma(z_2, z_4, z_6) - z_1)^3 \, \zeta_{\epsilon_1}(t) \\
z_2' = c_2(\sigma(z_1) - z_2)^3 \, \zeta_{\epsilon_2}(-t) \\
z_3' = c_3(\overline{y}_1^{next} \circ \sigma(z_2, z_4, z_6) - z_3)^3 \, \zeta_{\epsilon_3}(t) \\
z_4' = c_4(\sigma(z_3) - z_4)^3 \, \zeta_{\epsilon_4}(-t) \\
z_5' = c_5(\overline{y}_2^{next} \circ \sigma(z_2, z_4, z_6) - z_5)^3 \, \zeta_{\epsilon_5}(t) \\
z_6' = c_6(\sigma(z_5) - z_6)^3 \, \zeta_{\epsilon_6}(-t)
\end{cases} \tag{6.11}$$

where $\overline{q}_{next} \circ \sigma(z_2, z_4, z_6)$ means $\overline{q}_{next}(\sigma(z_2), \sigma(z_4), \sigma(z_6))$ and so on, ζ_ϵ is given by (4.25), ϵ_1, …, ϵ_6 are defined as in the proof of Theorem 4.6.2, i.e., $\epsilon_1 = \gamma/c_1(\overline{q}_{next} \circ \sigma(z_2, z_4, z_6) - z_1)^{-4} + \gamma/c_1$, …, $\epsilon_6 = \gamma/c_6(\sigma(z_5) - z_6)^{-4} + \gamma/c_6$, where $\gamma = 1/8$ is the targeting error. Actually, without loss of generality, we can add 4 to $1/\epsilon_1$, …, $1/\epsilon_6$. This is useful to determine the values of c_1, …, c_6. Indeed $\vartheta(t) \geq 3/4$ for $t \in [0.16, 0.34]$ and because $\epsilon_1 \geq 4$, …, $\epsilon_6 \geq 4$, one has $|\zeta_{\epsilon_i}(t) - 1| \leq 1/4$ in $[0.16, 0.34]$, which implies $\int_0^{1/2} \zeta_{\epsilon_i}(t)dt > 3/4(0.34 - 0.16) = 0.135$ for $i = 1, \ldots, 6$.

Hence, if we consider the requirement (4.15) (change ϕ to ζ_{ϵ_i} and $t_0 = 0$, $t_1 = 1/2$ for n even or $t_0 = 1/2$, $t_1 = 1$ otherwise), we conclude that we just have to pick $c_i \geq 8/0.135$, e.g. $c_i = 60$ for $i = 1, \ldots, 6$.

Then, from the construction outlined in the proof of Theorem 4.6.2, one can see that (6.11) simulates the Turing machine as described in the statement of that theorem. Notice that $z_2(t)$, for $t \in [k, k + 1/2]$ gives the state of M at step k with an error bounded by $\varepsilon + (\delta + \gamma)/2 = 5/16$ (cf. (4.30)). So, we can take $y_q = z_2$.

However, the system (6.11) is not polynomial. Nevertheless, according to Theorem 3.2.5, the ODE (6.11) can be converted to a polynomial ODE, in a constructive manner. So, we just have to show that (6.11) can be reduced to a polynomial ODE of degree 56 to complete the proof of the theorem.

We recall that the proof of Theorem 3.2.5 is bottom-up, where each non-polynomial function, e.g. $\sin y_1$, is replaced by a variable plus a set of polynomial ODEs. Let us see how we can reduce the PIVP functions appearing in the system (6.11). In what follows we assume that x and y are variables in an IVP, whose derivatives can be written as a polynomial (possibly involving

other variables of the IVP) of degrees k and n, respectively (for short, we will simply say that x and y have degree k and n). Then we want to know what is the degree of the polynomial IVP giving functions like $\sin x$, $l_2(x,y)$, etc.

1. The case of sin and cos. We have
$$\begin{cases} (\sin x)' = x' \cos x \\ (\cos x)' = -x' \sin x \end{cases} \implies \begin{cases} y_1' = x' y_2 \\ y_2' = -x' y_1 \end{cases}$$
where y_1 and y_2 substitute $\sin x$ and $\cos x$, respectively. So, if x has degree k, $\sin x$ and $\cos x$ can be replaced by variables having degree $k+1$.

2. The case of arctan. One has
$$\begin{cases} (\arctan x)' = \frac{x'}{1+x^2} \\ \left(\frac{1}{1+x^2}\right)' = -\frac{2\acute{x}x}{(1+x^2)^2} \end{cases} \implies \begin{cases} y_1' = x' y_2 \\ y_2' = -2x' x y_2^2 \end{cases}$$
where y_1 replaces $\arctan x$. So, $\arctan x$ can be replaced by a variable of degree $k+1$, but also introduces another variable of degree $k+3$.

3. Proceeding similarly for the case of $\sin x$, one concludes that $\sigma(x)$, where σ is given by (4.2), can be substituted by a variable of degree $k+1$.

4. The function l_2. From Proposition 4.2.5, one has $l_2(x,y) = \pi^{-1} \arctan(4xy-2y) + 1/2$. Let $y_1 = 4xy-2y$ and $y_2 = \arctan y_1$. Then y_1 has degree $\max(k,n)+1$, and y_2 has degree $\max(k,n)+2$ but introduces another variable of degree $\max(k,n)+4$. Then, if $z = l_2(x,y)$, we conclude that $z' = \pi^{-1} y_2'$ i.e. $l_2(x,y)$ can be replaced by a variable of degree $\max(k,n)+2$ but introduces variables of degree up to $\max(k,n)+4$.

5. We now study the function l_3 of Proposition 4.2.7, where $\varepsilon = 1/4$ (and hence $d = 0$). Let w_1 be a variable substituting $(\sigma(x)-1)^2$. Then
$$w_1' = 2\sigma'(x)(\sigma(x)-1)$$
which implies that w_1 has degree $k+2$. Then, if w_2 substitutes $l_2((\sigma(x)-1)^2, 3y) = l_2(w_1, 3y)$, one has that w_2 has degree $\max(k+2,n)+2$ but introduces variables of degree at most $\max(k+2,n)+4$. Now let w_3 substitute $l_2(x/2, 3y)$. The variable w_3 has degree $\max(k,n)+2$, but introduces another variable of degree $\max(k,n)+4$. Finally, $l_3(x,y)$ can be replaced by a variable
$$w_4 = w_2(2w_3 - 1) + 1$$
satisfying
$$w_4' = w_2'(2w_3 - 1) + w_2 2w_3'.$$
Thus $l_3(x,y)$ can be replaced by a variable of degree $\max(k+2,n)+3$, but introduces other auxiliary variables of degree up to $\max(k+2,n)+5$.

6. The case of $\zeta_\epsilon(t)$, given by (4.25). Actually, for reasons of convenience, we study the case of
$$\zeta_{1/y}(x) = l_2(\vartheta(x), y),$$
where ϑ is given by (4.26). It's not hard to see that $\vartheta(x)$ can be replaced by a variable of degree $k+2$ (proceed as in the case for sin and cos). Therefore, $\zeta_{1/y}(x)$ can be replaced by a variable of degree $\max(k+2,n)+2$, but introduces variables of degree up to $\max(k+2,n)+4$.

With these auxiliary results, we are ready to determine the degree of a polynomial ODE equivalent to (6.11).

(i) First, consider the case of z_2, z_4, and z_6 in (6.11). For the variable z_2, we have that the function $\sigma(z_1)$ will be replaced by a variable, as well as $\zeta_{\epsilon_2}(t)$. Therefore, the derivative of z_2 will be a polynomial of degree 4. Proceeding similarly for z_4 and z_6, we conclude that z_2, z_4, and z_6 are variables with degree 4.

(ii) Let us consider the case of z_1. Note that the expression of $\overline{q}_{next} \circ \sigma(z_2, z_4, z_6)$ in (6.11) is equivalent to substituting $\overline{y}$ in (6.8) by $\overline{y} = \omega \circ \sigma^{[2]}(z_4)$ and $\overline{q}$ by $\sigma(z_2)$. Thus $\overline{y}$ and $\overline{q}$ are variables of degree 7 and 5, respectively, in the system (6.11). Moreover, $\sigma^{[14]}(\overline{y})$ and $\sigma^{[14]}(\overline{q})$ can be replaced by variables of degree 21 and 19, respectively. Thus $\overline{q}_{next} \circ \sigma(z_2, z_4, z_6)$ is a polynomial in several variables, with degree $(6-1)+(4-1) = 8$. More, $\zeta_{\epsilon_1}(t)$ corresponds to a variable, and thus the derivative of z_1 can be written as a polynomial of variables having degree $3 \times 8 + 1 = 25$.

(iii) Let us consider the case of z_3. The variable $\overline{y}_1^{next}$ is given by (4.10). The value H_1 in (6.9) will be replaced by a variable (as we did for l_3 previously), and $\overline{P}_1$, $\overline{P}_2$, and $\overline{P}_3$ will be linear functions on the variables replacing $\sigma^{[3]}(\overline{y}_1)$, etc. Thus, from (4.10), we conclude that $\overline{y}_1^{next}$ is a polynomial of degree 3 on several variables. Thus, from (6.11), we conclude that the derivative of z_3 can be written as a polynomial of degree $3 \times 3 + 1 = 10$.

(iv) The case of z_5 is similar to z_3. In particular, it follows that z_5 is a variable of degree 10.

However, in the previous analysis, we didn't study the degree of some of the auxiliary variables used in our reasonings. For instance, in point (i), we just asserted that $\zeta_{\epsilon_2}(t)$ will be replaced by a variable, say z, but we still don't know which is the degree of this variable. In the following, we determine the degrees of those variables. To match more closely the previous analysis, we also divide them into separate items, where (i)' corresponds to the analysis of variables introduced in (i), and so on.

(i)' The variable z_1 has degree 25, and hence $\sigma(z_1)$ has degree 26. Concerning the case of $\zeta_{\epsilon_2}(t)$, we remark that

$$\frac{1}{\epsilon_2} = \frac{1}{\gamma}\left(c_2(\sigma(z_1) - z_2)^4 + c_2\right).$$

Then, since $\sigma(z_1)$ can be replaced by a variable of degree 26, we conclude that $1/\epsilon_2$ can be replaced by a variable y of degree $26 + 3 = 29$. Thus $\zeta_{\epsilon_2}(t) = \zeta_{1/y}(t)$ can be replaced by a variable of degree 31, but introduces also variables of degree 33. The same result holds for $\zeta_{\epsilon_4}(t)$ and $\zeta_{\epsilon_6}(t)$.

(ii)' Concerning ζ_{ϵ_1}, we have that $\overline{q}_{next} \circ \sigma(z_2, z_4, z_6)$ is a polynomial in several variables (that were already studied — $\sigma^{[14]}(\overline{y})$, ...), with degree 8 and that its derivative has degree $21 + 7 = 28$. This yields that

$$\frac{1}{\epsilon_1} = \frac{1}{\gamma}\left(c_1(\overline{q}_{next} \circ \sigma(z_2, z_4, z_6) - z_1)^4 + c_1\right)$$

can be replaced by a variable of degree $28 + 3 \times 8 = 52$. Thus $\zeta_{\epsilon_1}(t)$ can be replaced by a variable of degree 54, but introduces also variables of degree 56.

(iii)' In the equation defining z_3', we still have to analyze the variables H_1, $\overline{P}_1$, $\overline{P}_2$, and $\overline{P}_3$, defined in (6.9). However, there we must substitute $\overline{y}_1$, $\overline{y}_2$, and $\overline{y}$, by $\sigma(z_4)$, $\sigma(z_6)$, and $\omega \circ \sigma^{[2]}(z_4)$, respectively. The analysis done for $\overline{q}_{next}$ in point (ii) applies for $\overline{s}_{next}$ and $\overline{h}_{next}$. In particular, $\overline{s}_{next}$ and $\overline{h}_{next}$ are polynomials in several variables, with degree 8, and the derivative of these variables are polynomials with degree up to 21. Thus, $\overline{s}_{next}$ and $\overline{h}_{next}$ can be considered as a variable of degree $21 + 7 = 28$. Since $\sigma^{[3]}(\overline{s}_{next})$ has degree 31, $\overline{P}_1$, $\overline{P}_2$, and $\overline{P}_3$ can be replaced by variables of degree up to 31. Relatively to H_1, from the analysis done for l_3 in point

5, we conclude that it can considered as a variable of degree $28+2+3=33$, introducing also variables of degree up to 35.

Having concluded our analysis, we see that the highest degree for a variable that appears in the polynomial expansion of (6.11) is the one of the variables introduced in the expansion of $\zeta_{\epsilon_1}(t)$, that has degree 56 — cf. point (ii)'. □

Proof of Theorem 6.6.2. Let M be the Turing machine of Lemma 6.6.5. Suppose that in the proof of Lemma 6.6.6 the state 4 is a halting one. Using the IVP defined on that lemma one has that for every $k \in \mathbb{N}$

$$\begin{cases} y_q(t) \leq 3 + \frac{5}{16} & \text{if } M \text{ has not halted at step } k \text{ and } t \leq k \\ y_q(t) \geq 4 - \frac{5}{16} & \text{if } M \text{ has already halted at step } k \text{ and } t \geq k. \end{cases} \tag{6.12}$$

Consider the IVP

$$\begin{cases} z_1' = y_q - 7/2 \\ z_2 = \frac{1}{z_1} \end{cases} \iff \begin{cases} z_1' = y_q - 7/2 \\ z_2' = -y_q z_2^2 \end{cases} \tag{6.13}$$

where $z_1(0) = z_2(0) = -1$. Since y_q appears as a component, we assume that this IVP is coupled with the polynomial IVP defined by Lemma 6.6.6. It is easy to see that while M hasn't halted, $y_q - 7/2 \leq -3/16$. Thus z_1 keeps decreasing and the IVP is defined in $(0, +\infty)$, i.e. the maximal interval is unbounded.

On the other hand, if M eventually halts, z_1 starts increasing at a rate of at least $3/16$ and will do that forever. So, at some time it will have to assume the value 0. When this happens, a singularity appears for z_2 and the maximal interval is therefore (right-)bounded. For negative values of t just replace t by $(-t)$ in (6.11) and assume t to be positive. The behavior of the system will be similar, and we reach the same conclusions for the left bound of the maximal interval. So M halts iff the maximal interval of the polynomial IVP (6.13) is bounded, i.e. boundedness is undecidable. □

Note that the degree 56 mentioned in Theorem 6.6.2 is not necessarily optimal. Actually, it remains an open question whether there are IVPs defined with polynomials of degree between 2 and 55, for which the boundedness problem is also undecidable. The techniques used in the proof of Theorem 6.6.2 could be used for proving undecidability for IVP defined with polynomials of lower degree than 56 provided that: (i) one can show that there is an universal Turing machine smaller than the one of Lemma 6.6.5, where the size is given by the product of the number of symbols and of the number of states (24 in that case) or (ii) one uses a technique more efficient than the one provided by Theorem 3.2.5 to convert the IVP (6.11) to a polynomial IVP.

CHAPTER 7

Conclusion

7.1 Concluding remarks

The present thesis has deepened our understanding of continuous dynamical systems defined with polynomial ODEs, at least from a computable perspective. The focus has essentially been made on the connections between polynomial ODEs and traditional computability (Turing machines, computable analysis, decidability questions). In that regard, our results helped to shorten the gap between discrete computation and analog computation by showing that these two areas overlap to a significant extent.

We have thus shown that analog computation can have more solid foundations. In particular, we presented PIVP functions as a model with many advantages:

1. It is mathematically sound in the sense that it defines a class that is closed under many important mathematical operations (e.g. arithmetical operations, composition, obtaining new functions as the solution of ODEs) — cf. Section 3.2.

2. Contrarily to what happens with other models of "real computation" (e.g. BSS model, computable analysis), it uses a language (ODEs) that is common in the field of continuous dynamical systems. Thus, under certain circumstances, this model can be seen as more natural and may be better suited to study some classes of continuous systems.

3. It is realistic in the sense that it can be implemented with existing physical devices (differential analyzers) — cf. Sections 3.3, 3.4.

4. It can simulate Turing machines, even under the influence of some perturbations, thereby having the power of Type-1 computability — cf. Chapter 4.

5. Under a suitable and natural variation, it is equivalent to Type-2 computability (i.e. computable analysis) — cf. Chapter 5.

Besides these interesting features, this model also gave us some hints about the limitations that digital computers have in modeling continuous processes. For instance, our results of Chapter 6 showed that even systems defined with computable ODEs that are mathematically well-behaved (analytic, polynomial) can have properties which cannot be computed nor verified with the help of a digital computer.

7.2 Directions for further work

Below we list some questions and possible new directions of research raised by this work.

1. The model presented in this thesis relies on polynomial ODEs. Can it be extended to other functions in such a way that it still preserves its capability to be implemented by a physical device as well as its nice mathematical properties?

2. On the other way round, should we allow the use of all polynomial ODEs? It is known that there are unstable systems, in which some of their properties no longer hold if some perturbation is added to the system, and thus can only be implemented by ideal physical devices cf. [GH83]. This subject seems to be quite interconnected with the idea of "stable systems" that often arises in the dynamical systems literature. Roughly, stable systems should correspond to "physically feasible" systems. However, the quest for an appropriate mathematical characterization of such systems seem to be a major challenge for mathematicians [Via01].

3. Can a polynomial IVP simulate a Turing machine if "noise" is not only added only on the initial condition, but also during the evolution of the system? Can we also simulate oracle (Type-2) machines under these conditions?

4. Can we present an adequate notion of computational complexity for continuous dynamical systems? Can we establish connections between this notion and the respective notions for Turing machines or oracle (Type-2) machines? A first step in that direction can be found in [SBHF99], [BHSF02], for a rather restricted case.

5. Is it possible to present other noncomputability and undecidability results for computable and analytic systems as in Chapter 6?

6. Which is the lower (integer) degree n of the class of polynomials for which the maximal interval problem becomes undecidable? Our results of Section 6.6 only provide the (constructible) upper bound $n = 56$, but the optimality question of obtaining the least such n remains open.

Bibliography

[AB01] E. Asarin and A. Bouajjani. Perturbed Turing machines and hybrid systems. In *Logic in Computer Science, 2001. Proc. 16th Annual IEEE Symposium*, pages 269–278, 2001.

[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.

[AD90] R. Alur and D. L. Dill. Automata for modeling real-time systems. In *Automata, Languages and Programming, 17th International Colloquium*, LNCS 443, pages 322–335. Springer, 1990.

[AD00] R. Alur and D. L. Dill. Are timed automata updatable? In E. A. Emerson and A. P. Sistla, editors, *Computed Aided Verification, 12th International Conference*, LNCS 1855, pages 464–479. Springer, 2000.

[Ahl79] L. Ahlfors. *Complex Analysis*. McGraw-Hill, 1979.

[AMP95] E. Asarin, O. Maler, and A. Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoret. Comput. Sci.*, 138:35–65, 1995.

[AP37] A. A. Andronov and L. Pontryagin. Systèmes grossiers. *Dokl. Akad. Nauk. SSSR*, 14:247–251, 1937.

[AP04] R. Alur and G. J. Pappas, editors. *Hybrid Systems: Computation and Control: 7th International Workshop (HSCC 2004)*. LNCS 2993. Springer, 2004.

[Arn78] V. I. Arnold. *Ordinary Differential Equations*. MIT Press, 1978.

[Atk89] K. E. Atkinson. *An Introduction to Numerical Analysis*. John Wiley & Sons, 2nd edition, 1989.

[BB85] E. Bishop and D. S. Bridges. *Constructive Analysis*. Springer, 1985.

[BCGH06] O. Bournez, M. L. Campagnolo, D. S. Graça, and E. Hainry. The General Purpose Analog Computer and Computable Analysis are two equivalent paradigms of analog computation. In J.-Y. Cai, S. B. Cooper, and A. Li, editors, *Theory and Applications of Models of Computation TAMC'06*, LNCS 3959, pages 631–643. Springer-Verlag, 2006.

[BCGHar] O. Bournez, M. L. Campagnolo, D. S. Graça, and E. Hainry. Polynomial differential equations compute all real computable functions. *J. Complexity*, to appear.

[BH98] V. Brattka and P. Hertling. Feasible real random access machines. *J. Complexity*, 14(4):490–526, 1998.

[BH04] O. Bournez and E. Hainry. Real recursive functions and real extentions of recursive functions. In M. Margenstern, editor, *Machines, Computations and Universality (MCU'2004)*, volume 3354 of *LNCS*, pages 116–127, 2004.

[BH05] Olivier Bournez and Emmanuel Hainry. Elementarily computable functions over the real numbers and $\mathbb{R}$-sub-recursive functions. *Theoret. Comput. Sci.*, 348(2–3):130–147, 2005.

[BHSF02] A. Ben-Hur, H. T. Siegelmann, and S. Fishman. A theory of complexity for continuous time systems. *J. Complexity*, 18(1):51–86, 2002.

[BNP71] *Basic Machines and How They Work*. Bureau of Naval Personnel, Dover Publications, Inc., 1971.

[Bow96] M. D. Bowles. U. S. technological enthusiasm and british technological skepticism in the age of the analog brain. *IEEE Ann. Hist. Comput.*, 18(4):5–15, 1996.

[Bra95] M. S. Branicky. Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoret. Comput. Sci.*, 138(1):67–100, 1995.

[Bro89] R. W. Brockett. Smooth dynamical systems which realize arithmetical and logical operations. In H. Nijmeijer and J. M. Schumacher, editors, *Three Decades of Mathematical Systems Theory*, LNCS 135, pages 19–30. Springer, 1989.

[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.

[BT00] V. D. Blondel and J. N. Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 9(36):1249–1274, 2000.

[Bus31] V. Bush. The differential analyzer. A new machine for solving differential equations. *J. Franklin Inst.*, 212:447–488, 1931.

[Cam02] M. L. Campagnolo. *Computational Complexity of Real Valued Recursive Functions and Analog Circuits*. PhD thesis, Instituto Superior Técnico/Universidade Técnica de Lisboa, 2002.

[Cas96] M. Casey. The dynamics of discrete-time computation, with application to recurrent neural networks and finite state machine extraction. *Neural Comp.*, 8:1135–1178, 1996.

[Cas98] M. Casey. Correction to proof that recurrent neural networks can robustly recognize only regular languages. *Neural Comp.*, 10:1067–1069, 1998.

[CL55] E. A. Coddington and N. Levinson. *Theory of Ordinary Differential Equations*. McGraw-Hill, 1955.

[CM01] M. Campagnolo and C. Moore. Upper and lower bounds on continuous-time computation. In I. Antoniou, C. Calude, and M. Dinneen, editors, *2nd International Conference on Unconventional Models of Computation - UMC'2K*, pages 135–153. Springer, 2001.

[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] M. L. Campagnolo, C. Moore, and J. F. Costa. An analog characterization of the Grzegorczyk hierarchy. *J. Complexity*, 18(4):977–1000, 2002.

[Cv04] P. Collins and J. H. van Schuppen. Observability of piecewise-affine hybrid systems. In R. Alur and G. J. Pappas, editors, *Hybrid Systems: Computation and Control, 7th International Workshop*, LNCS 2993, pages 265–279. Springer, 2004.

[Dav73] M. Davis. Hilbert's tenth problem is undecidable. *Amer. Math. Monthly*, 80:233–269, 1973.

[DL89] J. Denef and L. Lipshitz. Decision problems for differential equations. *J. Symbolic Logic*, 54(3):941–950, 1989.

[Frä99] M. Fränzle. Analysis of hybrid systems: An ounce of realism can save an infinity of states. In J. Flum and M. Rodríguez-Artalejo, editors, *Computer Science Logic (CSL'99)*, LNCS 1683, pages 126–140. Springer, 1999.

[GC03] D. S. Graça and J. F. Costa. Analog computers and recursive functions over the reals. *J. 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.

[GCB06] D. S. Graça, M. L. Campagnolo, and J. Buescu. Computability with polynomial differential equations. Preprint, CLC, Department of Mathematics, Instituto Superior Técnico, 1049-001 Lisboa, Portugal, 2006. Submitted for publication.

[GH83] J. Guckenheimer and P. Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcation of Vector Fields*. Springer, 1983.

[Gra02] D. S. Graça. The General Purpose Analog Computer and Recursive Functions over the Reals. Master's thesis, IST/UTL, 2002.

[Gra03] D. S. Graça. Computability via analog circuits. In V. Brattka, M. Schröder, K. Weihrauch, and N. Zhong, editors, *Proceedings of the International Conference on Computability and Complexity in Analysis (CCA 2003)*, pages 229–240. FernUniversität in Hagen, 2003.

[Gra04] D. S. Graça. Some recent developments on Shannon's General Purpose Analog Computer. *Math. Log. Quart.*, 50(4-5):473–485, 2004.

[Grz57] A. Grzegorczyk. On the definitions of computable real continuous functions. *Fund. Math.*, 44:61–71, 1957.

[Gun90] R. Gunning. *Introduction to Holomorphic Functions of Several Variables*. Chapman & Hall/CRC, 1990.

[GZB06] D. S. Graça, N. Zhong, and J. Buescu. The ordinary differential equation defined by a computable function whose maximal interval of existence is non-computable. In G. Hanrot and P. Zimmermann, editors, *Proceedings of the 7th Conference on Real Numbers and Computers (RNC 7)*, pages 33–40. LORIA/INRIA, 2006.

[GZB07] D.S. Graça, N. Zhong, and J. Buescu. Computability, noncomputability and undecidability of maximal intervals of IVPs. *Trans. Amer. Math. Soc.*, 2007. To appear.

[Hal80] J. K. Hale. *Ordinary Differential Equations*. Robert E. Krieger Pub. Co, 2nd edition, 1980.

[Hau85] J. Hauck. Ein kriterium für die konstruktive lösbarkeit der differentialgleichung $y' = f(x,y)$. *Z. Math. Logik Grundlag. Math.*, 31(4):357–362, 1985.

[HKPV98] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *J. Comput. System Sci.*, 57(1):94–124, 1998.

[HMU01] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 2nd edition, 2001.

[Höl87] O. Hölder. Über die eigenschaft der gamma funktion keiner algebraische differentialgleichung zu genügen. *Math. Ann.*, 28:1–13, 1887.

[HR99] T. Henzinger and J.-F. Raskin. Robust undecidability of timed and hybrid systems. In F. Vaandrager and J. H. van Schuppen, editors, *Hybrid systems: computation and control; Second International Workshop, (HSCC'99)*, LNCS 1569. Springer, 1999.

[HSD04] M. W. Hirsch, S. Smale, and R. Devaney. *Differential Equations, Dynamical Systems, and an Introduction to Chaos*. Academic Press, 2004.

[HW95] J. H. Hubbard and B. H. West. *Differential Equations: A Dynamical Systems Approach — Higher-Dimensional Systems*. Springer, 1995.

[KCG94] P. Koiran, M. Cosnard, and M. Garzon. Computability with low-dimensional dynamical systems. *Theoret. Comput. Sci.*, 132:113–128, 1994.

[KM99] P. Koiran and C. Moore. Closed-form analytic maps in one and two dimensions can simulate universal Turing machines. *Theoret. Comput. Sci.*, 210(1):217–223, 1999.

[Ko91] K.-I Ko. *Computational Complexity of Real Functions*. Birkhäuser, 1991.

[Kra01] S. Krantz. *Function Theory of Several Complex Variables*. AMS Chelsea Publishing, 2nd edition, 2001.

[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.

[Lan05] S. Lang. *Real and Functional Analysis*. Springer, 3rd edition, 2005.

[Lef65] S. Lefshetz. *Differential Equations: Geometric Theory*. Interscience, 2nd edition, 1965.

[Lor63] E. N. Lorenz. Deterministic non-periodic flow. *J. Atmos. Sci.*, 20:130–141, 1963.

[LPY99] G. Lafferriere, G. J. Pappas, and S. Yovine. A new class of decidable hybrid systems. In *Hybrid Systems: Computation and Control (HSCC'99)*, LNCS 1569, pages 137–151. Springer, 1999.

[LR87] L. Lipshitz and L. A. Rubel. A differentially algebraic replacement theorem, and analog computability. *Proc. Amer. Math. Soc.*, 99(2):367–372, 1987.

[Mat70] Y. Matijasevič. Enumerable sets are diophantine. *Dokl. Akad. Nauk*, 191:279–282, 1970.

[Mat72] Y. Matijasevič. Diophantine sets. *Upsekhi Mat. Nauk*, 27:124–164, 1972.

[MC04] J. Mycka and J. F. Costa. Real recursive functions and their hierarchy. *J. Complexity*, 20(6):835–857, 2004.

[MH98] J. E. Marsden and M. J. Hoffman. *Basic Complex Analysis*. W. H. Freeman, 3rd edition, 1998.

[Mil85] J. Milnor. On the concept of attractor. *Comm. Math. Phys.*, 99(2):177–195, 1985.

[MO98] W. Maass and P. Orponen. On the effect of analog noise in discrete-time analog computations. *Neural Comput.*, 10(5):1071–1095, 1998.

[Moo90] C. Moore. Unpredictability and undecidability in dynamical systems. *Phys. Rev. Lett.*, 64(20):2354–2357, 1990.

[Moo96] C. Moore. Recursion theory on the reals and continuous-time computation. *Theoret. Comput. Sci.*, 162:23–44, 1996.

[Moo98] C. Moore. Finite-dimensional analog computers: Flows, maps, and recurrent neural networks. In C. Calude, J. Casti, and M. Dinneen, editors, *1st International Conference on Unconventional Models of Computation - UMC'98*, pages 59–71. Springer, 1998.

[NOSS93] X. Nicollin, A. Olivero, J. Sifakis, and S.Yovine. An approach to the description and analysis of hybrid systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems*, LNCS 736, pages 148–178. Springer, 1993.

[Odi89] P. Odifreddi. *Classical Recursion Theory*, volume 1. Elsevier, 1989.

[Odi99] P. Odifreddi. *Classical Recursion Theory*, volume 2. Elsevier, 1999.

[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.

[PER81] M. B. Pour-El and J. I. Richards. The wave equation with computable initial data such that its unique solution is not computable. *Adv. Math.*, 39:215–239, 1981.

[PER89] M. B. Pour-El and J. I. Richards. *Computability in Analysis and Physics*. Springer, 1989.

[PEZ97] M. B. Pour-El and N. Zhong. The wave equation with computable initial data whose unique solution is nowhere computable. *Math. Log. Quart.*, 43:499–509, 1997.

[PV94] A. Puri and P. Varaiya. Decidability of hybrid systems with rectangular differential equations. In *6th Workshop on Computer-Aided Verification*, LNCS 818, pages 95–104. Springer, 1994.

[Rit48] J. F. Ritt. *Integration in Finite Terms*. Columbia Univ. Press, 1948.

[Rog96] Y. Rogozhin. Small universal Turing machines. *Theoret. Comput. Sci.*, 168(2):215–240, 1996.

[Ros72] M. Rosenlicht. Integration in finite terms. *Amer. Math. Monthly*, 79(9):963–972, 1972.

[RS85] L. A. Rubel and F. Singer. A differentially algebraic elimination theorem with application to analog computability in the calculus of variations. *Proc. Amer. Math. Soc.*, 94(4):653–658, 1985.

[Rub89] L. A. Rubel. A survey of transcendentally transcendental functions. *Amer. Math. Monthly*, 96(9):777–788, 1989.

[Rue89] D. Ruelle. *Chaotic Evolution and Strange Attractors*. Cambridge University Press, 1989.

[Ruo96] K. Ruohonen. An effective Cauchy-Peano existence theorem for unique solutions. *Internat. J. Found. Comput. Sci.*, 7(2):151–160, 1996.

[Sal62] B. Salzmann. Finite amplitude free convection as an initial value problem. *J. Atmos. Sci.*, 19:239–341, 1962.

[SBHF99] H. T. Siegelmann, A. Ben-Hur, and S. Fishman. Computational complexity for continuous time dynamics. *Phys. Rev. Lett.*, 83(7):1463–1466, 1999.

[Sha41] C. E. Shannon. Mathematical theory of the differential analyzer. *J. Math. Phys. MIT*, 20:337–354, 1941.

[Sip97] M. Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 1997.

[Sma66] S. Smale. Structurally stable systems are not dense. *Amer. J. Math.*, 88:491–496, 1966.

[Sma92] S. Smale. *Mathematical Research Today and Tomorrow*, chapter Theory of computation, pages 59–69. Springer, 1992.

[Son98] E. D. Sontag. *Mathematical Control Theory*. Springer, 2nd edition, 1998.

[Spa82] C. Sparrow. *The Lorenz equations: Bifurcations, Chaos, and Stange Attractors*. Springer, 1982.

[SS95] H. T. Siegelmann and E. D. Sontag. On the computational power of neural networks. *J. Comput. System Sci.*, 50(1):132–150, 1995.

[Sta02] V. E. E. Stadigh. Ein satz ueber funktionen die algebraische differentialgleichungen befriedigen und ueber die eigenschaft der function $\zeta(s)$ keiner solchen gleichung zu genügen. Thesis, Helsinki, 1902.

[Tuc98] W. Tucker. *The Lorenz attractor exists*. PhD thesis, Univ. Uppsala, 1998.

[Tuc99] W. Tucker. The Lorenz attractor exists. In *C. R. Acad. Sci. Paris*, volume 328 of *Série I, Mathématique*, pages 1197–1202, 1999.

[Tur36] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Math. Soc.*, 2(42):230–265, 1936.

[Via00] M. Viana. What's new on Lorenz strange attractors? *Math. Intelligencer*, 22(3):6–19, 2000.

[Via01] M. Viana. Dynamical systems: Moving into the next century. In B. Engquist and W. Schmid, editors, *Mathematics Unlimited - 2001 and Beyond*, pages 1167–1178. Springer, 2001.

[Wei00] K. Weihrauch. *Computable Analysis: an Introduction*. Springer, 2000.

[WZ02] K. Weihrauch and N. Zhong. Is wave propagation computable or can wave computers beat the Turing machine? *Proc. London Math. Soc.*, 85(3):312–332, 2002.

Index