Computability and computational complexity of the evolution of nonlinear dynamical systems

Olivier Bournez[1], Daniel S. Graça[2,3], Amaury Pouly[1,2] and Ning Zhong[4]

[1] Ecole Polytechnique, LIX, 91128 Palaiseau Cedex, France
[2] CEDMES/FCT, Universidade do Algarve, C. Gambelas, 8005-139 Faro, Portugal
[3] SQIG/Instituto de Telecomunicações, Lisbon, Portugal
[4] DMS, University of Cincinnati, Cincinnati, OH 45221-0025, U.S.A.

Abstract. Nonlinear dynamical systems abound as models of natural phenomena. They are often characterized by highly unpredictable behaviour which is hard to analyze as it occurs, for example, in chaotic systems. A basic problem is to understand what kind of information we can realistically expect to extract from those systems, especially information concerning their long-term evolution. Here we review a few recent results which look at this problem from a computational perspective.

1 Introduction

Scientists have always been fascinated by problems involving dynamical systems. These appear in the context of many important applications, ranging from celestial mechanics to electronics. Efforts to understand those systems have led to many important insights. A relatively complete theory was developed for the case of linear systems. It was also shown that the general theory of linear systems can often be applied even for non-linear systems: near (hyperbolic) equilibrium points, a nonlinear system has the same qualitative structure as its linearization through the Hartman-Grobman theorem [1].

Despite those early successes, over the last few decades scientists have come to understand that even simple nonlinear dynamical systems can exhibit complicated behaviour. This realization can perhaps be traced back to the work of the French mathematician Henri Poincaré in the late XIXth century [2]. Poincaré studied a problem from celestial mechanics known as the three-body problem. The three-body problem is the problem of determining the motions of three bodies (e.g. two stars in a binary system plus a planet), which interact in accordance with the laws of classical mechanics (Newton's laws of motion and of universal gravitation), given their initial positions, masses and velocities. Unlike the two-body problem, which was completely solved and shown to have a very predictable behaviour (in the case of a star and a planet it yields Kepler's laws of planetary motion), Poincaré showed that orbits for the three-body problem could be very complex — in modern terms, he showed that these orbits had elements of *chaotic* behaviour. Subsequent studies of nonlinear dynamical systems were done by mathematicians such as Hadamard [3], Birkhoff [4], Kolmogorov

[5] M.L. Cartwright and J.E. Littlewood [6], and S. Smale [7]. Although most of these studies involved physically inspired systems, the resulting papers were often hard to read for the non-specialist and remained within the pure mathematicians' community well into the middle of the XXth century. This situation changed with the arrival of the digital computer. Numerical simulations done by cheap, fast and widely available computers allowed the non-specialist scientist to get a grasp on the complexity of their favourite models. Edward Lorenz was one of these early scientists who stumbled upon chaos. He was a meteorologist interested in long-term weather prediction. During the course of his weather simulations, and to save time, he would sometimes start a simulation in the middle of its course, using data obtained in previous simulations. His computer used 6 decimal digits internally, but would only print 3 digits as a result. Since he only had access to those 3 digits, he restarted the computation using this truncated data. He soon realized, to his surprise, that the small error introduced by the truncation could have a huge effect in the evolution of the system, and that this effect could happen quite rapidly [8]. Nowadays this phenomenon is known as "sensitive dependence on initial conditions" or, more poetically, as the "butterfly effect" – a hurricane's formation could be contingent on whether or not a distant butterfly had flapped its wings several weeks before. However, this "butterfly effect" also raises critical questions about the reliability of computer-generated simulations of nonlinear systems. For example, the experiments suggested that the trajectories of the system which Lorenz was studying would converge over time to a butterfly shaped object, later known as the *Lorenz attractor*. But to rigorously prove that the figure drawn by a computer was not an artefact of round-off error accumulation was a much harder problem. It was the 14th problem on the list of problems proposed by the Fields medallist S. Smale [9] for the 21^{st} century. It was finally solved in 1998, following the work of Tucker [10].

The interplay between chaos (and other forms of nonlinear complex behaviour) and computers poses interesting questions to the theoretical computer science community; in particular, questions such as the following: Are chaotic and other highly complex systems necessarily computationally intractable? How do they compare to intractability in Turing Machines? Is it possible to relate the degree of computational intractability with some degree of "chaos complexity" (or other) of the system? On the other hand, since physical implementations of computers are subjected to natural laws, can chaotic/nonlinear computers be computationally more powerful than Turing machines (from a computability and/or computational complexity perspective)? Many of these interesting questions have to do with the long-term behaviour of dynamical systems. In this paper we review, from a computational perspective, a few selected results that we have obtained recently in this area.

2 Dynamical systems: basics

A dynamical system is a way of describing how points in a *state space* evolve (deterministically) over time. Formally it is a triple $(S, \mathbb{T}, \phi)$, where S is an open set of the Euclidean space (the state space), $\mathbb{T}$ is a semigroup which denotes the time (usually $\mathbb{T} = \mathbb{R}$ — continuous-time systems — or $\mathbb{T} = \mathbb{N}$ — discrete-time systems) and $\phi : \mathbb{T} \times S \to S$ is a map (the evolution rule). The map ϕ must also have the following properties (we write $\phi(t, x) = \phi_t(x)$, where $\phi_t : S \to S$): (i) $\phi_0 : S \to S$ is the identity; (ii) $\phi_t \circ \phi_s = \phi_{t+s}$ for each $t, s \in \mathbb{T}$. It can be shown [11] that the evolution of a discrete-time system can be obtained by iterating a map ($= \phi_1$). Iterating the map t times will yield the state of the system at time t. For C^1 continuous-time systems, the evolution rule can be rewritten as a differential equation $x' = f(x)$, where $f(x) = \partial_t \phi_t(x)|_{t=0}$.

Dynamical systems theory deals with the long-term qualitative behaviour of dynamical systems, since quantitative information is usually a too-ambitious goal. Some typical questions studied in dynamical systems include: "Will the system settle down to some steady state in the long term? If yes, which properties characterize this steady state? Can one tell if there are several co-existing steady states? Which types of steady states does a particular class of dynamical systems admit?"

We remark that these questions correspond to problems in theoretical computer science and related applications (e.g. control theory — see e.g. [12]) which are interested in analyzing the long-term behaviour of some system. The steady states mentioned above are usually known as *attractors* or, in a broader sense, as *invariant sets*. The set of points which converge to a given attractor defines its *basin of attraction*. Up to dimension two, dynamical systems are relatively well-behaved. The Poincaré-Bendixson Theorem (see [13], Section 8*.5 for more details) rules out chaos in two-dimensional systems — it states that, except for singularities (like homoclinic solutions [14]), attractors in the plane can only be points or closed (periodic) orbits. However, as soon as one enters three-dimensional space, complex chaotic behaviour can appear, as demonstrated by the Lorenz system (which is a three-dimensional system of ordinary differential equations defined by quadratic functions). Thus, low-dimensional systems defined by simple rules can have attractors and invariant sets with a much more complex structure than fixed points or periodic orbits.

3 Computability

Before analyzing the computational complexity of problems related to dynamical systems, it makes sense to first investigate whether these problems are computable. Since problems in dynamical systems theory deal with the long-term behaviour of systems one might stumble upon the Halting problem. Indeed, since Turing machines, considered as discrete-time dynamical systems, can be embedded into many classes of discrete-time or continuous-time dynamical systems, these latter systems often inherent the rich structures of Turing machines and, in particular, undecidability (see e.g. [15], [16], [17], [18]). The embedding of a Turing machine (which is a purely discrete model) into a continuous system is often done by implicitly using discrete elements, like piecewise linear functions, or more generally piecewise defined functions or dynamics.

A more challenging task is to analyze problems which do not allow these discrete elements. Classical physics is built upon polynomials, trigonometric function, logarithms, etc. (these functions, their inverses, and all of their possible compositions are known in Analysis as *elementary functions* — not to be confused with elementary functions in computability theory, see e.g. [19] — or as closed-form functions) and elementary functions are analytic. Analyticity is a very strong property; for an analytic function, its global property is determined by its local behaviour. Thus, it is much harder to encode the Halting problem (or the behaviour of a given Turing machine) into an analytic dynamical system and it may not be evident whether these systems can present non-computable behaviour. For these reasons, we have focused much of our work on analytic systems, avoiding noncomputability results which could be obtained due only to the inherently discrete dynamics of the system.

Although the long term behaviour of a system can be very complicated and often chaotic, many problems associated with the system are still (algorithmically) decidable. For example, it is possible to decide whether or not some (rational) initial point will converge to the fixed point of

$$x' = Ax$$

at the origin, when A is an $n \times n$ hyperbolic (i.e. the real parts of the eigenvalues of A are nonzero) matrix [20].

3.1 Computability of trajectories over unbounded domains

A basic problem concerning the computability of solutions for ordinary differential equations (ODEs)is the following: given some ODE and some initial condition, can we compute the respective solution for *arbitrary* $t \geq 0$? Or equivalently, are trajectories of a continuous-time dynamical system computable? This problem is not as trivial as it first might seem. One might suggest using any standard numerical method to solve it. The problem with numerical methods is that virtually all numerical methods use a strong hypothesis: the existence of a Lipschitz constant valid for the vector field over the *entire* domain where the solution is defined. Of course, if the vector field is C^1 and the time is restricted to a closed interval $[t_0, t_1]$, then this desired global Lipschitz constant is readily obtainable (see e.g. [1] p. 71). This global Lipschitz constant is also critical for previous results concerning computability of ODEs; in fact, it is to be found in virtually any such result previously obtained (see e.g. [21]). Although there are several computability results established for ODEs without a global Lipschitz constant — as long as the solution is assumed to be unique (see [21], Section 7.1), those results however only hold for ODEs defined on a *closed interval* $[t_0, t_1]$. Another related result can be found in [22], where the author proves computability of solutions of ODEs in unbounded domains without requiring the use of Lipschitz constants. However, Ruohonen requires a very restrictive bound on the growth of f.

Thus the previous results on computability of ODEs do not address the general case where computability must be established over the time interval

$[0, +\infty)$ for functions which might grow (in absolute terms) quicker than a linear function (in which case no global Lipschitz constant exists).

Classically, existence and uniqueness results for C^1 ODEs over the *maximal interval* where the solution is defined (see e.g. [1], Section 2.4 for a precise definition of the maximal interval) is shown recursively: first establish existence and uniqueness over an interval $[a, b]$; then the interval where existence and uniqueness is guaranteed is recursively extended, and shown to converge to a maximal interval. Note that this convergence can be non-effective, as we have shown in [23]: a computable ODE with computable initial conditions can yield a non-computable maximal interval even if the ODE is analytic. The result is further refined in [24], where it is shown that the set of all initial data generating solutions with lifespans longer than k, $k \in \mathbb{N}$, is in general not computable. Although the maximal interval of existence may be non-computable, it is (lower) semi-computable and therefore, if we want to compute the solution y of the ODE at time t, it suffices to extend the interval $[a, b]$ until it includes t and then we can use a standard algorithm to compute $y(t)$ (see e.g. [23]). In this process, how to extend the interval $[a, b]$ is a key step and, concerning the extension, there are two approaches. The first approach (see e.g. [1], Section 2.4, for the similar case of C^1 functions) is to use local Lipschitz constants to extend the solution recursively. This can be done for C^1 functions. In particular, we have shown in [23] that the solution of $y' = f(y), y(t_0) = y_0$ can be computed over its maximal interval of definition from (f, f', t_0, x_0) for C^1 functions.

This result was later generalized in [25], using the second classical approach to extend the interval $[a, b]$ (see e.g. [26], p. 12, Theorem 3.1). In this approach we split the state space into several compacts and provide an algorithm to generate partial solutions inside these compacts (via Peano's Existence Theorem). This approach is more general then the previous one in the sense that it does not require Lipschitz constants (it is valid even for ODEs with non-unique solutions). But, computationally, it requires to know how to "glue" the partial solutions to get the whole solution over the maximal interval. In [25] we solve this problem by using an enumeration approach. The idea is to generate all possible "tubes" which cover the solution, and then check if this cover is valid within the desired accuracy. The proof is constructive, although terribly inefficient in practice. Nonetheless this technique shows that if the solution to an initial-value problem defined by an ODE is unique, then the solution must be computable over its maximal interval of existence, under the (minimal) classical conditions ensuring existence of a solution to an ODE initial-value problem (continuity).

Theorem 1 ([25]). *Consider the initial value problem $y' = f(y), y(t_0) = y_0$, where f is continuous on $\mathbb{R}^n$. Suppose that there is a unique solution y, defined on the maximal interval (α, β). Then $y(t)$ is computable from f, t_0, x_0, t, where $t \in (\alpha, \beta)$.*

An interesting problem which we will tackle in Section 4.1 is to determine the computational complexity of computing the solution of a given ODE over its maximal interval of existence.

3.2 Computability of attractors and invariant sets

We have seen in Section 2 that a trajectory may converge over time to some kind of attractor (steady state). It is an important problem both in theory and in practice to characterize these attractors. For example, one is often interested in verification problems. The purpose of verification theory is to prove (or disprove) that a system behaves as intended. A system may have safe states (invariant sets), where the system should be, and unsafe states, where undesirable behaviour may occur (e.g. turbulence over a wing, a nuclear reactor overheating, etc.). Thus many verification problems often amount to understand which kind of attractors/invariant sets a system has, and which are their respective basins of attraction.

For verification problems involving complex systems, computers are, of course, essential tools. Thus, it becomes useful to know which invariant sets are computable and which are not; for computable invariant sets, which can be computed efficiently. Many specialized results exist in the literature of control theory. Here we will focus on more general problems in dynamical systems. One of our initial projects was to investigate computability in the planar dynamical systems. As we have mentioned, due to the Poincaré-Bendixson theorem, invariant sets in the plane can only be fixed points or periodic orbits (with the exception of singularities). Thus it is natural to study this class of systems first. In general, fixed points can be computed since they are the zeros of the function f defining the differential equation $y' = f(y)$ and isolated zeros of a computable function are also computable — see e.g. [27]. On the other hand, many problems related to the simple planar dynamics can be undecidable, as the following results [28] show.

Theorem 2 ([28]). *Given as input an analytic function f, the problem of deciding the number of equilibrium points of $y' = f(y)$ is undecidable, even on compact sets. However, the set formed by all equilibrium points is upper semi-computable.*

Theorem 3 ([28]). *Given as input an analytic function f, the problem of deciding the number of periodic orbits of $y' = f(y)$ is undecidable, even on compact sets. However, the set formed by all hyperbolic periodic orbits is upper semi-computable.*

In short, a hyperbolic periodic orbit is an orbit to which nearby trajectories converge exponentially fast (see e.g. [1] for more details). The hyperbolic property entails stronger stability properties to small perturbations of the system. This is important when computing an invariant set, since despite round-off errors, one should still be able to effectively approximate the invariant set. In this sense, the notion of stability in dynamical systems theory seems to be intertwined with computability.

The above results show that one cannot hope for general procedures to compute invariant sets for relatively large families of dynamical systems such as planar systems. Instead, algorithms should be devised for each particular case.

As we have seen before, invariant sets need not to be fixed points or periodic orbits, but may take complex shapes such as Lorenz attractor. Are these complex shapes computable? We have analyzed in [20] the case of Smale's horseshoe (see e.g. [14]), which was the first example of an hyperbolic invariant set which is neither an equilibrium point nor a periodic orbit. Contrarily to what one could expect a priori, Smale's horseshoe is computable.

Theorem 4 ([20]). *The Smale Horseshoe is a computable (recursive) closed set.*

3.3 Computability of basins of attraction

Similar to the case of invariant sets, when a class of dynamical systems is considered, general algorithms for computing basins of attractor do not exist, even for classes of well-behaved systems; in particular, when the class is large. For instance, Zhong [29] showed that there exists a C^∞ computable dynamical system having a unique computable hyperbolic equilibrium point but the basin of attraction of this hyperbolic equilibrium point is non-computable.

This result was generalized in a paper we just submitted:

Theorem 5. *There exists a computable analytic dynamical system having a computable hyperbolic equilibrium point such that its basin of attraction is recursively enumerable, but is not computable.*

Thus finding the basin of attraction of a given attractor is, in general, a non-computable problem, although one can semi-compute this basin from the inside.

4 Computational complexity

4.1 Computational complexity of trajectories over unbounded domains

We have seen that the trajectories of a non-linear dynamical system are computable over their maximal interval of definition. It thus becomes interesting to understand the underlying computational complexity of finding these trajectories. However, the two results [23] and [25] are not very helpful in that respect because they use an exhaustive approach — the underlying algorithm will always stop when some condition is met (and we are sure that this condition must eventually be met) — but we do not have *a priori* any clue on how much time this will take.

Recently we have been focusing on polynomial differential equations. This particular subclass has some advantages: it is well-behaved, it corresponds to a particular model of computation — Shannon's General Purpose Analog Computer [30] (thus understanding the computational complexity of polynomial differential equations is equivalent to understand the computational complexity of this computational model), and it captures more complex ODEs defined using trigonometric functions, exponentials, etc. [31].

Unfortunately, computing solutions of ODEs efficiently over unbounded domains is not easy. We can get a numerical estimate of the solution, but Lipschitz constants play a crucial role for estimating in an efficient manner the error made in the computation and are thus needed if we want to compute the solution of a polynomial ODE in the sense of computable analysis [27]. But since no global Lipschitz constant exist, in general, for polynomials in unbounded domains, we have to compute local Lipschitz constants. But these local Lipschitz constants depend on the compact set where the solution is. Thus, to be sure that the solution is in a given compact set, given only some numerical estimate of the solution, we need to know the error of this estimate, i.e. we need to know beforehand the Lipschitz constant we were trying to find in the first place.

In [32] we presented a solution to this problem: break the vicious circle using the size of the solution as one of the parameters on which the computational complexity of the solution is measured upon.

Theorem 6 ([32]). *The solution, at time T, of an initial-value problem $y' = p(y)$ with initial condition $y(t_0) = y_0 \in \mathbb{R}^d$, where p is a vector of polynomials, can be computed, in an uniform manner, with precision 2^{-n}, in time polynomial in T, n and $Y = \sup_{t_0 \leq t \leq T} \|y(t)\|$.*

Methods usually used for numerical integrations (including the basic Euler's method, Runge Kutta's methods, etc.) tend to fall in the general theory of n-order methods for some fixed $n \in \mathbb{N}$. They do compute the solution of an ODE in polynomial time in a *compact* time interval $[a, b]$, but are not guaranteed to work in polynomial time over the maximal interval of definition of the solution.

In [32] we solve this problem by using variable order methods. This is done with the help of a Taylor approximation, with the number of terms used on the approximation depending on the input. The idea of using variable order methods is not new. W. Smith mentioned it in [33], where he claimed that some classes of ODEs can be solved in polynomial time over maximal intervals, however without providing a full proof to this claim. Variable order methods have also been used in [34], [35], [36], but for the case of bounded domains.

It is not only polynomial ODEs which can be solved in polynomial time over their maximal interval of definition, many analytic ODEs can also be solved in polynomial time, as long as the function defining the ODE and its solution do not grow quicker than a very generous bound [37].

Theorem 7 ([37]). *Let $y(t)$ be the solution of the initial-value problem $y' = f(y)$, $y(t_0) = y_0$, in $\mathbb{C}^d$, where f is analytic in $\mathbb{C}^d$ and p-poly-bounded (in $\mathbb{C}^d$), f, t_0, x_0 are polynomial-time computable, and $y(t)$ admits an analytic extension to $\mathbb{C}^d$ and is poly-bounded over $\mathbb{C}^d$. Then the function which maps f, x_0, t_0, and t to the value $y(t)$ is polynomial-time computable.*

(a function is p-poly-bounded if $\|f(x)\|$ is bounded by $2^{p(\log \|x\|)}$). This result uses previous results from Müller et al. [38], [34], [39] which say that, locally, the solution of an analytic ODE can be computed in polynomial time. We then extract (in polynomial-time) the coefficients of the Taylor series of the solution, which allow us to compute, in polynomial time, the solution of the ODE in its maximal interval of definition using the hypothesis of poly-boundedness. The hypothesis that f is analytic on the complex space (and poly-bounded there) and that the solution of the ODE admits an analytic extension to $\mathbb{C}^d$ are needed because we use the Cauchy integral formula in the proof.

5 Acknowledgments

D.S. Graça was partially supported by *Fundação para a Ciência e a Tecnologia* and EU FEDER POCTI/POCI via SQIG - Instituto de Telecomunicações through the FCT project PEst-OE/EEI/LA0008/2011.

References

1. Perko, L.: Differential Equations and Dynamical Systems. 3rd edn. Springer (2001)
2. Poincaré, H.: Sur le problème des trois corps et les équations de la dynamique. Acta Math. **13** (1889) 1–270
3. Hadamard, J.: Les surfaces à courbures opposées et leurs lignes géodésiques. J. Math. Pures et Appl **4** (1898) 27–73
4. Birkhoff, G.D.: Dynamical Systems. vol. 9 of the American Mathematical Society Colloquium Publications. American Mathematical Society (1927)
5. Kolmogorov, A.N.: Preservation of conditionally periodic movements with small change in the hamiltonian function. Doklady Akademii Nauk SSSR **98** (1954) 527–530
6. Cartwright, M.L., Littlewood, J.E.: On non-linear differential equations of the second order, i: The equation $y'' + k(1 - y^2)y' + y = b\lambda k \cos(\lambda t + a)$, k large. J. Lond. Math. Soc. **20**(3) (1945) 180–189
7. Smale, S.: Differentiable dynamical systems. Bull. Amer. Math. Soc. **73** (1967) 747–817
8. Lorenz, E.N.: Deterministic non-periodic flow. J. Atmos. Sci. **20** (1963) 130–141
9. Smale, S.: Mathematical problems for the next century. Math. Intelligencer **20** (1998) 7–15
10. Tucker, W.: The Lorenz attractor exists. In: C. R. Acad. Sci. Paris. Volume 328 of Series I - Mathematics. (1999) 1197–1202
11. Hirsch, M.W., Smale, S.: Differential Equations, Dynamical Systems, and Linear Algebra. Academic Press (1974)
12. Sontag, E.D.: Mathematical Control Theory. 2nd edn. Springer (1998)
13. Hubbard, J.H., West, B.H.: Differential Equations: A Dynamical Systems Approach — Higher-Dimensional Systems. Springer (1995)
14. Hirsch, M.W., Smale, S., Devaney, R.: Differential Equations, Dynamical Systems, and an Introduction to Chaos. Academic Press (2004)
15. Branicky, M.S.: Universal computation and other capabilities of hybrid and continuous dynamical systems. Theoret. Comput. Sci. **138**(1) (1995) 67–100
16. Asarin, E., Maler, O.: Achilles and the tortoise climbing up the arithmetical hierarchy. J. Comput. System Sci. **57**(3) (1998) 389–398

17. Bournez, O.: Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy. Theoret. Comput. Sci. **210**(1) (1999) 21–71
18. Koiran, P., Moore, C.: Closed-form analytic maps in one and two dimensions can simulate universal Turing machines. Theoret. Comput. Sci. **210**(1) (1999) 217–223
19. Odifreddi, P.: Classical Recursion Theory. Volume 2. Elsevier (1999)
20. Graça, D., Zhong, N., Buescu, J.: Computability, noncomputability, and hyperbolic systems. Appl. Math. Comput. **219**(6) (2012) 3039–3054
21. Ko, K.I.: Computational Complexity of Real Functions. Birkhäuser (1991)
22. Ruohonen, K.: An effective Cauchy-Peano existence theorem for unique solutions. Internat. J. Found. Comput. Sci. **7**(2) (1996) 151–160
23. Graça, D., Zhong, N., Buescu, J.: Computability, noncomputability and undecidability of maximal intervals of IVPs. Trans. Amer. Math. Soc. **361**(6) (2009) 2913–2927
24. Rettinger, R., Weihrauch, K., Zhong, N.: Topological complexity of blowup problems. Journal of Universal Computer Science **15**(6) (2009) 1301–1316
25. Collins, P., Graça, D.S.: Effective computability of solutions of differential inclusions — the ten thousand monkeys approach. Journal of Universal Computer Science **15**(6) (2009) 1162–1185
26. Hartman, P.: Ordinary Differential Equations. 2nd edn. Birkhäuser (1982)
27. Weihrauch, K.: Computable Analysis: an Introduction. Springer (2000)
28. Graça, D.S., Zhong, N.: Computability in planar dynamical systems. Natural Computing **10**(4) (2011) 1295–1312
29. Zhong, N.: Computational unsolvability of domain of attractions of nonlinear systems. Proc. Amer. Math. Soc. **137** (2009) 2773–2783
30. Graça, D.S.: Some recent developments on Shannon's General Purpose Analog Computer. Math. Log. Quart. **50**(4-5) (2004) 473–485
31. Graça, D.S., Campagnolo, M.L., Buescu, J.: Computability with polynomial differential equations. Adv. Appl. Math. **40**(3) (2008) 330–349
32. Bournez, O., Graça, D.S., Pouly, A.: On the complexity of solving polynomial initial value problems. In: 37th International Symposium on Symbolic and Algebraic Computation (ISSAC 2012). (2012)
33. Smith, W.D.: Church's thesis meets the n-body problem. Appl. Math. Comput. **178**(1) (2006) 154–183
34. Müller, N., Moiske, B.: Solving initial value problems in polynomial time. In: Proc. 22 JAIIO - PANEL '93, Part 2. (1993) 283–293
35. Werschulz, A.G.: Computational complexity of one-step methods for systems of differential equations. Math. Comput. **34** (1980) 155–174
36. Corless, R.M.: A new view of the computational complexity of IVP for ODE. Numer. Algorithms **31** (2002) 115–124
37. Bournez, O., Graça, D.S., Pouly, A.: Solving analytic differential equations in polynomial time over unbounded domains. In Murlak, F., Sankowski, P., eds.: Proc. 36th International Symposium on Mathematical Foundations of Computer Science (MFCS 2011). LNCS 6907, Springer (2011) 170–181
38. Müller, N.T.: Uniform computational complexity of taylor series. In Ottmann, T., ed.: 14th International Colloquium on Automata, Languages and Programming. LNCS 267, Springer (1987) 435–444
39. Müller, N.T., Korovina, M.V.: Making big steps in trajectories. Electr. Proc. Theoret. Comput. Sci. **24** (2010) 106–119