Rigorous numerical computation of polynomial differential equations over unbounded domains

Olivier Bournez[1], Daniel S. Graça[2,3], Amaury Pouly[1,2]

[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

Abstract. In this abstract we present a rigorous numerical algorithm which solves initial-value problems (IVPs) defined with polynomial differential equations (i.e. IVPs of the type $y' = p(t, y)$, $y(t_0) = y_0$, where p is a vector of polynomials) for any value of t. The inputs of the algorithm are the data defining the initial-value problem, the time T at which we want to compute the solution of the IVP, and the maximum allowable error $\varepsilon > 0$. Using these inputs, the algorithm will output a value $\tilde{y}_T$ such that $\|\tilde{y}_T - y(T)\| \leq \varepsilon$ in time polynomial in T, $-\log \varepsilon$, and in several quantities related to the polynomial IVP.

1 Introduction

With the appearance of fast and cheap digital computing devices in the last decades, digital computers have become increasingly important as a simulation tool in many fields, ranging from weather forecast to finance. The idea underlying such simulations is simple: pick some system which we want to study and simulate it on a computer using some numerical method. Quite often we can obtain in this manner information about the system which we could not collect otherwise. Think, for example, about the case of weather forecast.

However, this poses a fundamental question: how reliable are these simulations? The truth is that, although historically such simulations have already given fundamental insights (like suggesting that dynamical systems can have *strange attractors* [6]), due to phenomena like sensitive dependence on initial conditions, in general not much is known about the overall error committed in such simulations.

It therefore seems to make sense to develop numerical methods with the property that we can rigorously tell which is the error done when we apply such methods. This is in contrast to what happens usually in numerical analysis where, at best, only estimates of the error are presented. On the other side, to obtain rigorous bounds on the error, we need to use more complicated methods, which are more amenable for analysis, and which are usually slower or might even be unfeasible for practical implementation. In general, it is not trivial to devise numerical methods which are practical to use and for which the error can be rigorously determined.

To achieve a balance between these contradicting requirements, it makes sense to consider restricted classes of problems, which are nonetheless general enough to be of practical importance.

In this paper, we consider initial-value problems (IVPs) defined with polynomial ordinary differential equations (ODEs)

$$\begin{cases} y'(t) = p(y) \\ y(t_0) = y_0 \end{cases} \tag{1}$$

where p is a vector of polynomials. We consider, without loss of generality that the system is autonomous since the independent variable t can always be written as an extra variable y_{n+1} satisfying $y'_{n+1} = 1$. We note that almost any IVP written with the "usual" functions of Analysis (trigonometric functions, exponentials, their inverses and compositions, etc.) can always be rewritten as a polynomial ODE (see e.g. [9], [3]).

Therefore IVPs with the format (1) are sufficiently broad to include a wide range of IVPs of practical interest. Moreover, since the right-hand side consists of relatively simple functions (polynomials), we are able to rigorously analyze the error committed when we solve numerically (1) by using properties of polynomials.

2 Solving IVPs over unbounded domains

It is standard practice to analyze numerical methods which solve IVPs only over a compact time interval $[0, T]$. This is both true in the Numerical Analysis literature (see e.g. [1]) as it is in the Theoretical Computer Science literature (see e.g. [5]).

However, in practice, people seldom set a valid time interval $[0, T]$ before implementing a numerical procedure, be it for the simple reason that they sometimes do not even know which might be the relevant value for T before doing some numerical simulations.

Therefore, it seems desirable to devise numerical methods which make no prior assumptions on the values which T might take. Of course, the time needed to execute the algorithm (the computational complexity) depends on T: in general, the higher T, the more time the algorithm will take to execute, but it seems to be a non-trivial task to determine which is the dependence of the execution time of the algorithm with respect to T.

There is a "conventional wisdom" that the unbounded time domain case can be reduced to the bounded time one, for which many results exist (see e.g. [5], [4]). However, this is not true, since in the bounded case many parameters which are important for the (unbounded) complexity are hidden in the constant of the "big-O" notation. A very simple example illustrates this problem. Assume that

$y : I \to \mathbb{R}^d$ is the solution of

$$\begin{cases} y_1(0)= 1 \\ y_2(0)= 1 \\ \quad\cdots \\ y_n(0)= 1 \end{cases} \qquad \begin{cases} y_1'(t)= y_1(t) \\ y_2(t)= y_1(t)y_2(t) \\ \quad\cdots \\ y_d'(t)= y_1(t)\cdots y_n(t) \end{cases}$$

It follows from [7] that for any fixed, compact I, y is polynomial time computable. On the other hand, this system can be solved explicitly and yields:

$$y_1(t) = e^t \qquad y_{n+1}(t) = e^{y_n(t)-1} \qquad y_d(t) = e^{e^{\cdot^{\cdot^{\cdot^{e^{e^t}-1}}}}-1}$$

One immediately sees that, since y is a tower of exponentials, y cannot be polynomial time computable over $\mathbb{R}$.

Note that this discrepancy arises because, in the bounded time case, the size of compact I is not taken as a parameter of the problem (because it is fixed). Also note that the dimension d of the system is hardly ever taken into account, although it has a huge influence on the resulting complexity. More precisely, if I is bounded then the complexity of computing $y(t)$ can be seen to be polynomial in t, but more than exponential in the length of the interval I and on d: this part is usually hidden in the "big-O" part of the constants.

3 Contributions

The main contribution of this abstract is to show that there is a numerical method, which we denote as SolvePIVPEx (see Section 4 for more details), which can rigorously solve IVPs (1) over unbounded domains.

Theorem 1 (Complexity and correctness of SolvePIVPEx**).** *Let $t \in \mathbb{R}$, $\varepsilon > 0$, and assume that y satisfies* (1) *over* $[t_0, t]$. *Let*

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

where SolvePIVPEx *is a numerical method described in Section 4. Then*

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

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

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

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

where k is the maximum degree of the components of p, d is the number of components of p, Σp is the sum of the absolute values of the coefficients of p, and $\text{Len}(t_0, t)$ *is a bound on the length of the curve $y(\cdot)$ from the point $(t_0, y(t_0))$ to the point $(t, y(t))$.*

4 The numerical method SolvePIVPEx and sketch of the proof of Theorem 1

For reasons of space, we will not present the algorithm defining the numerical method SolvePIVPEx nor the detailed proof of Theorem 1 (see [8] for more details). However, in this section, we briefly sketch the ideas underlying SolvePIVPEx and the proof of Theorem 1.

The numerical method SolvePIVPEx is based on a generic adaptive Taylor meta-algorithm which numerically solves (1). This is a meta-algorithm in the sense that, in a first approach, we leave open the question of how we choose some of the parameters of the algorithm. The goal of this meta-algorithm is, given as input $t \in \mathbb{Q}$ and $0 < \varepsilon < 1$ and the initial condition of (1), to compute $x \in \mathbb{Q}^d$ such that $\|x - y(t)\| < \varepsilon$.

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

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

This meta-algorithm works by solving the ODE (1) with initial condition $y(t_i) = \tilde{y}_i$ over a small time interval $[t_i, t_{i+1}]$, yielding as a result the approximation $\tilde{y}_{i+1}$ of $y(t_{i+1})$. This approximation over this small time interval is obtained using a Taylor approximation of order ω_i (we also do not fix, in a first approach, the value ω_i to analyze its influence on the error and on the time complexity of the algorithm. After this analysis is done, we can choose appropriate values for ω_i) using the polynomial algorithm given in [2]. This procedure is repeated recursively over $[t_0, t_1], [t_1, t_2], \ldots, [t_i, t_{i+1}], \ldots$ until we reach the desire time $t_n = t$. This introduces three potential sources of errors: (i) a global error due to the fact that, on the interval $[t_i, t_{i+1}]$ we do not solve $y' = p(y)$ with the initial value $y(t_i)$ but instead with the initial value $\tilde{y}_i$; (ii) a truncation error over $[t_i, t_{i+1}]$ because we only compute a truncated Taylor series of the solution instead of the full Taylor series; (iii) a rounding error because we might only have a finite number of bits to store partial results.

Using the crucial fact that the right-hand side of (1) consists of polynomials, at each time step t_i, one can present an argument based on Cauchy majorants to establish a lower bound on the local radius of convergence. We can choose the step length $t_{i+1} - t_i$ to be a constant fraction of the estimated radius of convergence, and the majorants can also be used to select a suitable truncation order ω_i. One can also show, using Gronwall's Lemma, that the propagation of errors from one step to the next can be controlled, and depends on a bound on the length of the curve $y(\cdot)$ over the domain under consideration. This last parameter needs to be fed to the algorithm, but we can automatically determine a suitable value for it, since we can decide if a (rational) value is large enough to be fed as a bound to the length of the curve. By using some (arbitrary, say the value 1) initial guess and by restarting the method with a larger guess if needed, we can continue this procedure until we decide that we have obtained a high enough value which can be used as a bound for the length of the curve.

Proceeding in this manner we end up fixing the parameters of the meta-algorithm (length of time steps, order of the Taylor approximation of each step, etc.) and we end up with an algorithm SolvePIVPEx which satisfies the conditions of Theorem 1.

Acknowledgments. D. 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 UID/EEA/50008/2013.

References

1. Atkinson, K.E.: An Introduction to Numerical Analysis. John Wiley & Sons, 2nd edn. (1989)
2. Bostan, A., Chyzak, F., Ollivier, F., Salvy, B., Schost, É., Sedoglavic, A.: Fast computation of power series solutions of systems of differential equations. In: SODA'07. pp. 1012–1021 (Jan 2007)
3. Graça, D.S., Campagnolo, M.L., Buescu, J.: Computability with polynomial differential equations. Adv. Appl. Math. 40(3), 330–349 (2008)
4. Kawamura, A.: Lipschitz continuous ordinary differential equations are polynomial-space complete. Computational Complexity 19(2), 305–332 (2010)
5. Ko, K.I.: Computational Complexity of Real Functions. Birkhäuser (1991)
6. Lorenz, E.N.: Deterministic non-periodic flow. J. Atmos. Sci. 20, 130–141 (1963)
7. Müller, N., Moiske, B.: Solving initial value problems in polynomial time. In: Proc. 22 JAIIO - PANEL '93, Part 2. pp. 283–293 (1993)
8. Pouly, A.: Continuous Models of Computation: From Computability to Complexity. Ph.D. thesis, Ecole Polytechnique/Universidade do Algarve (2015)
9. Warne, P.G., Warne, D.P., Sochacki, J.S., Parker, G.E., Carothers, D.C.: Explicit a-priori error bounds and adaptive error control for approximation of nonlinear initial value differential systems. Comput. Math. Appl. 52(12), 1695–1710 (Dec 2006), http://dx.doi.org/10.1016/j.camwa.2005.12.004