AUTOMATIC DIFFERENTIATION TOOLS IN COMPUTATIONAL DYNAMICAL SYSTEMS

A. HARO

Abstract. In this paper we describe a unified framework for the computation of power series expansions of invariant manifolds and normal forms of vector fields, and estimate the computational cost when applied to simple models.

By *simple* we mean that the model can be written using a finite sequence of compositions of arithmetic operations and elementary functions. In this case, the tools of Automatic Differentiation are the key to produce efficient algorithms.

By *efficient* we mean that the cost of computing the coefficients up to order k of the expansion of a d-dimensional invariant manifold attached to a fix point of a n-dimensional vector field ($d = n$ for normal forms) is proportional to the cost of computing the truncated product of two d-variate power series up to order k.

We present actual implementations of some of the algorithms, with special emphasis to the computation of the 4D center manifold of a Lagrangian point of the Restricted Three Body Problem.

Mathematics Subject Classification: 34C20,34C30,34C30,34C30,65Pxx,68W30

1. Introduction

Math is much about solving equations and Dynamical Systems is not an exception. In order to characterize the invariance of a manifold with respect to a given dynamical system, one uses functional equations whose unknows give the parameterization of the manifold, and the known terms come from the model. In normal forms computations, the unknowns are both the conjugacy and the normal form. In this context, the use of series expansions of the unknowns is an standard (and old) practice, specially in Celestial Mechanics.

In this paper we present a unified framework of different methods to numerically compute the coefficients of power series expansions of invariant manifolds and normal forms around a fixed point of a vector field. A similar methodology works for maps, and for invariant manifolds attached to periodic orbits and invariant tori. The framework

A.H. acknowledges the support of the Spanish Grant MEC-FEDER MTM2006-11265 and the Catalan grant CIRIT 2005 SGR-01028.

is inspired in the parameterization method of Cabré, Fontich and de la Llave [7] and the semi-numerical algorithms proposed by Simó [52], and in fact are built on the studies of Poincaré. Specialized algorithms also appear in [4]. A reference book on the theory of normal forms and its applications is [45].

We moreover focus on the applicability of the framework to *simple* models, that are written as a finite (and relatively small) number of compositions involving elementary functions such as the four arithmetic operations, the exponential, the power functions (including square roots), the logarithm, the trigonometric functions, etc. or even functions that are implicitely defined by equations involving elementary functions, etc. A key observation is that many (if not most) of the models that appear in applications are simple. A moment of reflection will convince the reader about that.

In this context, the use of Automatic Differentiation (AD for short) seems to be appropriate for the symbolic manipulation of the Taylor expansions appearing in the functional equations. AD is a set of techniques based on the mechanical application of the chain rule to obtain derivatives of a function given as a computer program, and notice that the coefficients of Taylor series are just normalized derivatives. See e.g. the reference book [26] and the web page of the AD community [1]. We explain the use of AD tools to implement the algorithms for simple models, and evaluate their cost. We obtain that the cost to compute the expansions up to a given order is proportional to the cost of making the (truncated) product of two power series up to the given order. We will refer to these algorithms as *efficient*. AD tools have been used to design efficient high-order univariate Taylor methods for the integration of ode [54, 37] [2]. This kind of tools have been also used for manipulation of Fourier and Fourier-Taylor series in computation of invariant tori and their associated invariant manifolds [28, 30] and for computation of normal forms of KAM tori [31]. Our purposes for this paper are however quite modest and we will only study AD tools for computation of Taylor expansions of parameterizations of invariant manifolds and normal forms of fixed points in vector fields. We postpone an analysis of these techniques for Hamiltonian systems to the ongoin paper [32], in which we also introduce several algorithms for generating canonical transformations, besides the standard ones based on generating functions (with mixed variables) and Lie series [31].

[1] `http://www.autodiff.org`

[2] For more information, see the web page of the course "Advanced Course on Long Time Integrations", `http://www.imub.ub.es/lti07/`

About the efficiency of the algorithms, we point out that we prioritize here the ease of their implementation over their optimal (asymptotic) complexity. For instance, we use the classical convolution formula to compute the truncated product of two power series, with a cost that is proportional to the square of the number of coefficients involved.

We would like to mention that Carles Simó and his collaborators introduced dynamical systems tools to perform orbit analysis of space missions for the ESA from 1983 to 1993, developing sophisticated algorithms for computation of invariant manifolds and quasi-periodic orbits in the Restricted Three Body Problem (RTBP) and perturbations of it. This pioneering work is partially covered in the series [21, 23, 18, 19]. The techniques include computation of high order approximations of invariant manifolds using (partial) normal forms [53, 36]. Influenced by this work, the space agencies ESA and NASA use dynamical systems techniques for many libration point missions. The NASA Genesis mission is an example. The methodology has also been used in studying certain regimes in molecular dynamics [56, 15]. The list of applications and researchers involved is too long and by no means we intend to be exhaustive.

These kind of problems are a perfect ground to test our algorithms, since there is an extensive literature and they are very-well known. We emphasize that using normal forms referred above for the computation of an invariant manifold seems to be quite expensive, since the dimension of the object in smaller than the dimension of the phase space. For example, the center manifold of one collinear fixed point in the RTBP is 4D, inside a 6D phase space, and the center manifold reduction uses 6D canonical transformations. So, it is more efficient a direct computation of the center manifold. In this paper we will compute such a manifold using AD tools, reducing drastically both the execution time and the memory space to store the coefficients of the expansions.

We would like to comment that, in studying an specific problem, it is sometimes more convinient using an specific software package rather than a commercial and general-purpose symbolic manipulator. Carles Simó has been developing specific symbolic manipulators for a long time to study a large variety of problems in Dynamical Systems, and has spread this work philosophy in the BCN group of Dynamical Systems, including myself [27]. In [35], Àngel Jorba reviews the Lie series algorithm for the numerical computation of normal forms, center manifolds and first integrals of Hamiltonian systems, and apply his C/C^{++} software package to some examples from Celestial Mechanics (see [50] for a parallel implementation). There are also symbolic manipulators

for dynamical systems that, spite of their generality, are extremely efficient [3]. Remarkable examples are TRIP [4], developed by J. Laskar and M. Gastineau, and COSY [5], developed by M. Berz and K. Makino. We have developed our own symbolic manipulator of multivariate power series, including AD tools (a preliminary version was used in [27]). Some implementation details appear in Section 5, including some benchmarks for the exceedingly meticulous reader. [6]

We finally emphasize that the computation of semi-local approximations of invariant manifolds and normal forms is just a first step for the understanding of the global dynamics. The approximations are accurate in a possibly large neighborhood of the steady state, a fundamental domain estimated using goodness tests of numerical solutions of the functional equations. Developing algorithms to globalize (and even visualize) the manifold, that is, to grow the manifold from the fundamental domain far away of the equilibrium point, is an issue that have attracted the attention of many authors. Even if we will not pursue this study here, we just mention that for 1D manifolds this is folklore, and for 2D manifolds there are several methods that have been proposed in the literature. See the interesting paper [41] for a review of several methods, some of them based on the growth of the manifold from its linear approximations. Even if those methods can work relatively well for the case of the stable and unstable manifolds, linear approximations are not accurate enough for computing center manifolds, or slow manifolds inside the stable manifold. It would be very interesting to combine those globalization methods with semi-local approximations. We also emphasize that high accuracy is also very useful if one is interested in computing accurate individual orbits on the invariant manifold, rather that in computing the whole manifold, e.g. if one is designing a space mission. Problems like the numerical computation of splitting of separatrices also need high accuracy and high order approximations (see e.g. [47, 49]).

The scheme of the paper is the following. In Section 2 we review the functional equations that characterize invariant manifolds and normal forms. The AD tools for multivariate power series are explained in Section 3. Section 4 is devoted to the algorithms of solution of the

[3] See e.g. the web pages of the "Advanced School on Specific Algebraic Manipulators", `http://www.imub.ub.es/sam07/`, and the "Advanced Course on Taylor Methods and Computer Assisted Proofs", `http://www.imub.ub.es/cap08/`.

[4] `http://www.imcce.fr/Equipes/ASD/trip/trip.php`

[5] `http://www.bt.pa.msu.edu/index_cosy.htm`

[6] The source codes can be obtained upon request by contacting the author. We however mention that, for the moment, the codes are not fully documented.

functional equations, and we evaluate their cost. Some implementation details of our software package,including some benchmarks, are given in Section 5. Actual implementations of the algorithms, to compute invariant manifolds in the RTBP, are described in Section 6. The final Section 7 contains several conclusions and proposals of future work.

2. The functional equations

Let us start reviewing the functional equations characterizing the invariance of a manifold with respect to a discrete dynamical system (a map), and then we will consider the continuous case (a flow), that is in fact the main subject of this paper. The present formulation is known under the name of *parameterization method*, from the papers [7, 8, 9]. For the sake of simplicity of the exposition, we will assume that all the objects (manifolds, maps, vector fields, etc.) are sufficiently differentiable, C^∞ or even analytic.

Given a map $F : \mathcal{N} \to \mathcal{N}$ in an n-dimensional manifold $\mathcal{N}$, a d-dimensional manifold $\mathcal{W} = \Phi(W)$ embedded in $\mathcal{N}$ through an immersion $\Phi : W \to \mathcal{N}$ is invariant under F if there exist a map $f : W \to W$ such that

$$F \circ \Phi = \Phi \circ f \ . \tag{1}$$

Notice that a point $z = \Phi(s)$ of $\mathcal{W}$, parameterized by $s \in W$, is mapped to a point

$$F(\Phi(s)) = \Phi(f(s)) \tag{2}$$

of $\mathcal{W}$, parameterized by $f(s) \in W$. We can think of Φ as a semiconjugacy, and f as a subsystem of F.

Similarly, given a vector field F in an n-dimensional manifold $\mathcal{N}$, a d-dimensional manifold $\mathcal{W} = \Phi(W)$ embedded in $\mathcal{N}$ through the immersion $\Phi : W \to \mathcal{N}$ is invariant under F if there exist a vector field f on W such that

$$F \circ \Phi = T\Phi \ f \ , \tag{3}$$

where T is the tangent functor (the differential). Notice that the vector field $T\Phi \ f$ is tangent to $\mathcal{W}$ in $\mathcal{N}$. Using local coordinates z for $\mathcal{N}$ and s for W, the vector field in $\mathcal{N}$ is written $\dot{z} = F(z)$, the vector field in W is $\dot{s} = f(s)$, and the invariance equation is

$$F(\Phi(s)) = \mathrm{D}\Phi(s) \ f(s) \ . \tag{4}$$

Observe that in both invariance equations (1) and (3) (or the local versions (2) and (4)) there are two unknowns Φ and f, meaning respectively the parameterization of the invariant manifold and the

dynamics on it. So then, the number of equations is n while the number of unknowns is $n + d$, and both equations are underdetermined. The underdeterminacy comes from the fact that if (Φ, f) is a solution of (1) (resp. (3)), then $(\Phi \circ h, h^{-1} \circ f \circ h)$ (resp. $(\Phi \circ h, (Th)^{-1} f \circ h)$) is also a solution.

There are two main philosophies in order to solve the equations (1) or (3):

- The *graph transform method*: looking for an uncomplicated parameterization of the manifold, Φ, e.g. a graph with respect to certain coordinates, and afterwards find f;
- The *parameterization method*: adapt the parameterization Φ to the shape of the manifold (even including turns, which is not possible with the graph representation), in such a way that the equations of the reduced dynamics, f, are uncomplicated, e.g. in normal form (linear or polynomial if $\mathcal{W}$ is an stable manifold of a fixed point, or a rotation if $\mathcal{W}$ is a torus, or a product of a linear map times a rotation if $\mathcal{W}$ is the stable manifold of a torus, etc.).

We can also use a mixed strategy, combining graph representation for some variables, and normal form representation for the rest of the variables. So, one can define an *style* of the parameterization. We emphasize that the parameterization method includes the computation of normal forms (total or partial) as a particular case. In this view, if $\mathcal{W} = \mathcal{N}$ and $\Phi : \mathcal{N} \to \mathcal{N}$ is a diffeomorphism, then the f in (1) and (3) is just the F written in new coordinates.

Both methodologies give rise to rigorous proofs on existence of invariant manifolds. The celebrated theorems of the stable manifold or the center manifold have been proved many times in the literature using the graph local representation, a methodology that goes back to Hadamard (see e.g. [39]). More recently [7], the parameterization method have been introduced to prove the existence and uniqueness of a variety of invariant manifolds, including the classical stable and strong stable manifolds, but also the so called slow manifolds. Parabolic manifolds have been also studied using this method in [3, 2]. The parameterization method has been applied to produce rigorous results of existence of invariant tori and whiskers in quasiperiodically forced systems, giving rise to algorithms that have been implemented in several examples [29, 28, 30]. Different *styles* of normal forms are considered in [45].

Let us consider now the case of invariant manifolds (and normal forms) of fixed points. In this case, one can expand both Φ and f

as Taylor series, and recursively compute their coefficients from the functional equations (2) or (4). These power series provide semi-local approximations of the objects.

Looking both sides of (2) and (4), we first notice that in their left hand side, F is known and Φ is unknown. One possibility to perform the composition $F \circ \Phi$ is compute first the Taylor expansion of F around the fixed point, and then substitute formally the power series of Φ in such Taylor expansion. There are even close form expressions of the coefficients of the composition, say the Faà di Bruno's formula. But this is a hard computational task, that have been paid a lot of attention in the computer science literature (see e.g. [6, 40]). Fortunately, in many cases the given system F is *simple*, so one can produce efficient ways of performing $F \circ \Phi$. We will use here the AD tools to compute these compositions efficiently. This is the object of Section 3.

In the right hand side both Φ and f are unknown. Here both equations (2) and (4) are very different. The discrete case appears to be more difficult computationally than the continuous case, since the composition in (2) turns a matrix product in (4). After all, the infinitesimal version of the composition is the product. Notice that, especially in the discrete case, the use of the parameterization method is more efficient since one looks for the simplest f, so the simplest way of performing the composition $\Phi \circ f(s)$. In the continuous case, there is not a great computational gain, since one has to do $\mathrm{D}\Phi(s)\ f(s)$, which involves products of power series. The methods to compute recursively the unknowns Φ and f for equation (4) are explained in Section 4. See e.g. [24] for some implementations of the methodology to solve (2) using the parameterization method, and e.g. [27] for the graph method for 2D and 3D invariant manifolds.

In this paper we consider equations (2),(4) at a formal level. So, the unknown functions (Φ, f) are represented as power series around the origin. Real-analytic functions are certainly represented in this way, but power series are also asymptotic Taylor expansions of C^∞ functions. For functions that are finite differentiable one should consider only truncated power series. Sometimes, one has to appeal to the complexification trick, so one has to use complex power series. All these standard considerations are left to the reader.

3. Multivariate power series

In this section, we are interested in how to handle the left hand side of (2),(4), in case that the equations describing the models described by F are *simple*.

3.1. **Algebraic manipulations of power series.** Given a commutative ring $\mathbb{K}$, a (formal) power series in the variables $x = (x_1, \ldots, x_d)$ with coefficients in $\mathbb{K}$ is an element of the ring $\mathbb{K}[[x]] = \mathbb{K}[[x_1, \ldots, x_d]]$, whose elements are of the form

$$f(x) = \sum_{k=0}^{\infty} f_k(x) \ , \tag{5}$$

where each $f_k(x)$ is a homogeneous polynomial of order k, that is an element of $\mathbb{K}_k[x]$. Notice that the modifier "formal", that we will omit from now on, has to do with the fact we only consider the algebraic structure of $\mathbb{K}[[x]]$, and we are ignoring questions of convergence of the expansions.

In this paper we will only consider real and complex power series, that is the fields $\mathbb{K} = \mathbb{R}$ and $\mathbb{K} = \mathbb{C}$. If, for instance, $\mathbb{K}$ is the ring of Fourier series, then the elements of $\mathbb{K}[[x]]$ are Fourier-Taylor series.

Notice also that $\mathbb{K}[[x]] = \bigoplus_{k=0}^{\infty} \mathbb{K}_k[x]$ is a graded algebra. This point of view is specially important for us since the solution of the functional equations (2),(4) is made "order by order". So, in the computer implementation of our symbolic manipulator of power series, we take into account that a series is an infinite sum of homogeneous polynomials representing each order, see (5). Of course, we can only hope to store the homogeneous polynomials up to a finite degree, that is a truncated power series. We will write

$$f_{\leq \bar{k}}(x) = \sum_{k=0}^{\bar{k}} f_k(x),$$

the truncated series up to degree $\bar{k}$. We will also use the obvious notations $f_{<\bar{k}}(x)$, $f_{>\bar{k}}(x)$, etc.

Remark 3.1. There are other ways of grouping terms that are useful in some cases, if the variables enter in the problem object of study in different ways. For example, if the system depends on parameters and one uses expansions w.r.t the state variables and the parameters. In fact, one can use power series w.r.t. to the parameters, with coefficients being power series w.r.t to the state variables.

Notice that we can represent (and store) a homogeneous polynomial of order k

$$f_k(x) = \sum_{m_1+\cdots+m_d=k} f_{m_1,\ldots,m_d} x_1^{m_1} \ldots x_d^{m_d} = \sum_{|m|=k} f_m x^m$$

(we use the standard multi-index notation) as a vector with

$$\mathrm{h}_d(k) := \binom{d+k-1}{d-1} \text{ coefficients,}$$

which are ordered, e.g., using the lexicographic order w.r.t the exponents $(m_2, \ldots, m_d)$. The total number of coefficients of a polynomial of degree k is

$$\mathrm{n}_d(k) := \binom{d+k}{d} \sim \frac{1}{d!}k^d \ ,$$

where $a_k \sim b_k$ means that $\lim_{k\to\infty} \frac{a_k}{b_k} = 1$. This is important to estimate the computer memory space needed to store the coefficients.

Let us discuss briefly the four basic arithmetic operations in $\mathbb{K}[[x]]$. Let $a(x) = \sum_{k=0}^{\infty} a_k(x), b(x) = \sum_{k=0}^{\infty} b_k(x)$ be two power series. The scalar multiplication of the series $a(x)$ by a scalar $\alpha \in \mathbb{K}$ is just

$$\alpha \cdot a(x) = \sum_{k=0}^{\infty} \alpha \cdot a_k(x) \ ,$$

and the addition/subtraction of both series $a(x)$ and $b(x)$ is

$$a(x) \pm b(x) = \sum_{k=0}^{\infty} (a_k(x) \pm b_k(x)) \ .$$

Obviously, these operations are implemented at each order just like the scalar multiplication and addition/subtraction of vectors.

The product of the two series is defined as

$$a(x) \cdot b(x) = \sum_{k=0}^{\infty} \left(\sum_{l=0}^{k} a_l(x) b_{k-l}(x) \right) \ .$$

Hence, using this convolution formula one has to multiply at each order homogeneous polynomials of complementary order.

The fourth arithmetic operation is the division of two power series

$$a(x)/b(x) = \sum_{k=0}^{\infty} d_k(x) \ ,$$

where $d_0 = a_0/b_0$ (assuming that b_0 is invertible, e.g. in our case of study, $b_0 \neq 0$), and the rest of terms are computed recursively by

$$d_k(x) = \frac{1}{b_0} \left(a_k(x) - \sum_{l=0}^{k-1} d_l(x) b_{k-l}(x) \right) \ .$$

Notice that the cost for computing the division up to order k is essentially the same as the product of two series up to order k.

Remark 3.2. The (truncated) multiplication of two power series is the most critical operation because, as in the definition of the division, this and many other functions are built up on the product. Moreover, the cost of the scalar multiplication and the addition/subtraction is negligible compared with the cost of the product. So, having a good implementation for the product is crucial for having a good symbolic manipulator.

Remark 3.3. We use the naive way of multiplying power series, based on the definition, but we emphasize that there are (asymptotically) faster algorithms, based e.g. in Karatsuba or Toom-Cook methods [38, 55, 11, 5], FFT [51, 6, 40, 57, 58], including multipoint-evaluation and interpolation techniques [42, 46]. In fact, finding fast algorithms for multiplying polynomials is object of current research. Unfortunately, who writes these lines is not an expert in these matters, and we will use the classical convolution formula. In our defense we will say that the degree of the polynomials at which these fast algorithms start to be much better than the naive method seems to be quite large, say, higher than 100. Another alternative it would be paralellized computation.

Remark 3.4. One can also use Newton method for finding the (truncated) inverse of a power series, say $d(x) = 1/b(x)$. So, at each step m of Newton method, starting from $d^0(x) = 1/b_0$, one computes

$$d^m(x) = d^{m-1}(x)(2 - d^{m-1}(x)b(x)) \ .$$

It is easy to see that the error series $e^m(x) = d^m(x)b(x) - 1$ is zero up to order $2^m - 1$ (in fact, $e^m(x) = -(e^{m-1}(x))^2$), so at each step one doubles the number of correct terms. This strategy can be used for many other power series computations.

We emphasize, however, that, using the convolution formula to perform the products of the Newton recursion, the computational cost to compute the inverse of the power series up to order $k = 2^m - 1$ is essentially the same as using the naive recursion. The way of making the Newton method more competitive is using fast methods to perform the products (see e.g. [43]).

3.2. **A definition of algorithmic complexity.** In the following, we are interested in a definition of complexity of an algorithm manipulating power series which is independent from the specific algorithm and implementation used to compute truncated products, and which is useful to obtain easy estimates of the running times in actual implementations of the algorithm.

Definition 3.5. *Let* $\mathrm{p}_d(k)$ *be the cost to compute the truncated product up to order* k *of two* d*-variate power series. Given an algorithm*

to compute expansions of d-variate power series up to order k (for a given problem, e.g. compute invariant manifolds), we will say that the algorithm is efficient *if its cost is proportional to the cost of the product,* $c\mathrm{p}_d(k)$*, where the constant c does not depend on k and it is referred to as the* complexity *of the algorithm.*

This will be our working definition of complexity along the paper, in which the cost of the truncated product of power series is a parameter, depending on the specific algorithm and the implementation used.

3.3. **Radial derivative.** An elementary operator that we have found very useful is the radial derivative. The radial derivative of a function $f(x)$ is defined by

$$Rf(x) = \mathrm{grad} f(x) \cdot x = \sum_{i=1}^{d} \frac{\partial f}{\partial x_i} \cdot x_i \ .$$

It is a derivation in the algebra of differentiable functions, that is to say for all functions f,g and constants α,

- $R(\alpha f) = \alpha \cdot Rf$, $R(f+g) = Rf + Rg$ (R is linear);
- $R(fg) = Rf \cdot g + f \cdot Rg$ (Leibniz rule).

What is the most important for us is the behavior of R acting on homogeneous functions, and in particular, on homogeneous polynomials. If f_k is a homogeneous function of order k, then $Rf_k(x) = kf(x)$, which is the famous Euler's identity.

So, it is quite natural to define the radial derivative of a power series $f(x) = \sum_{k=0}^{\infty} f_k(x)$ as

$$Rf(x) = \sum_{k=0}^{\infty} k f_k(x) \ .$$

The radial derivative is also a derivation in the algebra $\mathbb{K}[[x]]$.

3.4. **Elementary functions of power series.** Composition of power series is the bottleneck in many algorithms. For instance, if $\varphi = \varphi(z)$ is function in the one dimensional variable z and $f(x) = \sum_{k=0}^{\infty} f_k(x)$ is a power series, one can certainly compute the composition $\varphi(f(x))$ by expanding φ around $f(0)$, computing the powers f^m and adding up. The same ideas work if the function φ is multivariate.

But if φ *simple*, that is composed by arithmetic operations and elementary functions, it is better to derive specific formulae. This is the philosophy of Automatic Differentiation [26]. The following is a generalization to several variables of some tricks appearing e.g. in the

Knuth's *magnus opus* "The Art of computing programming" [40]. We will explain them with the aid of the radial derivative.

An elementary observation is that, for an univariate function $\varphi = \varphi(z)$ and a multivariate function $f = f(x)$,

$$R(\varphi \circ f)(x) = \varphi'(f(x))\ Rf(x)\ . \tag{6}$$

We can think of (6) also at a formal level. So if φ satisfies e.g. some elementary differential equation, we can use (6) to obtain a recurrence for the homogeneous terms g_k of the power series $g = \varphi \circ f$, starting from the seed $g_0 = \varphi(f_0)$. In the following lines, we will specify this idea in some examples.

The exponential. As a first example, consider $\varphi(z) = e^z$, and $e(x) = \exp(f(x))$. The zero order term is $e_0 = \exp(f_0)$. Since $\varphi'(z) = \varphi(z)$, then $Re(x) = e(x)Rf(x)$ and the series $e(x) = \exp(f(x))$ is computed recursively by

$$e_k(x) = \frac{1}{k}\sum_{l=0}^{k-1}(k-l)f_{k-l}(x)e_l(x)\ .$$

Notice that the cost up to order k is $\sim \mathrm{p}_d(k)$.

The power function. Consider now the power function of exponent $\alpha \neq 0$, $\varphi(z) = z^\alpha$. Here it is the computation of $p(x) = (f(x))^\alpha$, whose zero order term is $p_0 = f_0^\alpha$ (we assume $f_0 \neq 0$). From the identity $f(x)Rp(x) = \alpha p(x)Rf(x)$, we obtain the recurrence

$$p_k(x) = \frac{1}{k\ f_0}\sum_{l=0}^{k-1}\left(\alpha(k-l) - l\right) f_{k-l}(x)p_l(x)\ .$$

Notice that the case $\alpha = -1$ corresponds to $p(x) = 1/f(x)$ (compare with the division algorithm), and $\alpha = \frac{1}{2}$ corresponds to $p(x) = \sqrt{f(x)}$. Notice that the cost up to order k is $\sim \mathrm{p}_d(k)$.

A special case is $\alpha = 2$, because we can compute $p(x) = f(x)^2$ by using the recurrence

$$p_k(x) = \sum_{l=0}^{\frac{k}{2}-1} f_l(x)f_{k-l}(x) + f_{\frac{k}{2}}(x) \ \text{ if } k \text{ is even}$$

$$p_k(x) = \sum_{l=0}^{\frac{k-1}{2}} f_l(x)f_{k-l}(x) \ \text{ if } k \text{ is odd}$$

Hence, the cost of making the square up to order k is $\sim \frac{1}{2}\ \mathrm{p}_d(k)$.

Similar recurrences can be obtained e.g. for the logarithmic function, with a cost $\sim \mathrm{p}_d(k)$.

Trigonometric functions. Since the trigonometric functions sin and cos satisfy the same second order linear differential equation, we compute them at the same time, even if in our model appears only one of them. If $s(x) = \sin(f(x))$ and $c(x) = \cos(f(x))$, then $Rs(x) = c(x)\ Rf(x)$ and $Rc(x) = -s(x)Rf(x)$, and we obtain the recurrences

$$s_k(x) = \frac{1}{k}\sum_{l=0}^{k-1}(k-l)f_{k-l}(x)c_l(x),$$

$$c_k(x) = -\frac{1}{k}\sum_{l=0}^{k-1}(k-l)f_{k-l}(x)c_l(x),$$

that we start from $s_0 = \sin(f(0))$ and $c_0 = \cos(f(0))$. Notice that, in this case, the cost is $\sim 2\mathrm{p}_d(k)$.

Similar recurrences can be obtained for the composition with the hyperbolic functions sinh and cosh, with a cost $\sim 2\mathrm{p}_d(k)$. Again, a similar construction can be used to compute recurrences for the composition with Jacobi elliptic functions (fixed an elliptic modulus), since the three functions sn, cn, dn satisfy a system of 3 quadratic differential equations. So, in this case one obtains a cost $\sim 3\mathrm{p}_d(k)$.

In summary, as a general (and heuristic) rule, one can derive efficient formulas for the composition $\varphi \circ f$ if φ satisfies a *simple* differential equation. In the cases mentioned above the elementary functions φ where univariate and satisfied algebraic ordinary differential equations. There is some kind of recursivity in the definition of elementary function. For instance, as long as a function satisfies an algebraic differential equation whose coefficients are previously defined elementary functions, then such a function can also be considered as elementary.

3.5. **The complexity of a simple model.** In the examples described above, the cost of the computation of the composition of a power series with an elementary function up to order k is proportional to the cost of the product $\mathrm{p}_d(k)$. Hence, rather than giving a formal definition of what a simple model is, we will appeal to the common sense and say that:

- a model (map, vector field, Hamiltonian, etc.) is *simple* if we can decompose the equations of the model in a finite sequence of single expressions, being a single expression any of the arithmetic operations or the elementary functions such as exponential, logarithm, sinus, etc.;
- the *length* of the model is the number of single expressions in which the model is decomposed;

- the *complexity* of the model is the sum of the complexities of the single expressions, where: the complexity of the scalar multiplication, the addition and subtraction is zero; the complexity of the product, division, exponential, logarithm, power function, etc. is 1; the complexity of the square is 0.5; the complexity of the trigonometric functions sin and cos, the hyperbolic functions sinh and cosh is 2; the complexity of any of the three elliptic functions sn, cn, dn is 3; etc.

If in the model appears other functions, one has to incorporate them into the definitions and in the implementation of the software package, and compute their corresponding complexities.

Notice also that, when doing compositions of the model with power series, the memory space needed to store all the intermediate steps to compose the single expressions of the model has to do with the length of the model, and the velocity to make effectively the computation has to do with the complexity of the model.

We also emphasize that both the length and the complexity are not absolute definitions and they depend on the ability (or the laziness) of who writes the codes, etc. For instance, if in the equations of the model appear the tangent function tan, but not sin and cos, rather than computing first both sin and cos and then divide both, all of this with a complexity 3, one can compute tan directly with a complexity 1.5. The researcher has to decide if gaining 1.5 is crucial or not for the computation.

3.6. **The cost of a (truncated) product.** In previous sections we have seen that the elementary operations of (truncated) series are built on the (truncated) product. So, this is the key operation and it is useful to know what is its computational cost. We use the classical convolution formula.

We measure the cost in terms of the number of operations with the coefficients, meaning essentially by operation "make one product and one addition, and index the result", which is the step that has to be repeatedly done when doing the product of polynomials. Notice that the cost of a single operation depends on the type of coefficient (integer, real, interval, complex, Fourier series, etc.) and the precision (single, double, multiprecision, etc.), but it also depends on the indexing algorithm to put the results on the data structure encoding the series. So, good implementations should consider not only the number of arithmetic operations and their cost, but also the time consumed indexing the results. In cases in which arithmetic operations are much

more expensive computationally than indexing, such as when the coefficients are in multiprecision or are (truncated) Fourier series, indexing routines are not so relevant. In our package, with the naive method using trees, specific indexing routines are not used. See Section 5 for implementation details and benchmarks.

Since we use the naive method, given two d-variate homogeneous polynomials of orders l and m, the cost of making the product is

$$\mathrm{ph}_d(l,m) := \binom{d+l-1}{d-1}\binom{d+m-1}{d-1}$$

number of operations. Hence, the number of operations in making the truncated product of two d-variables series up to degree k is

$$\mathrm{p}_d(k) = \sum_{m=0}^{k}\sum_{l=0}^{m}\mathrm{ph}_d(l,m-l) = \binom{2d+k}{2d} \sim \frac{1}{(2d)!}k^{2d}\ . \tag{7}$$

Remark 3.6. As a result of this combinatorial formulae, one easily obtains that

$$\mathrm{p}_d(k) \sim \frac{d!^2}{(2d)!}n^2.$$

where $n = \mathrm{n_d(k)}$ is the number of coefficients. Karatsuba-type methods satisfy

$$\mathrm{p}_d(k) \sim C_K\ n^{1+\epsilon}\ ,$$

for some ϵ in $]0,1[$. Asymptotically faster algorithms (with quasi-linear complexity, e.g. based on FFT) have costs like

$$\mathrm{p}_d(k) \sim C_F\ n\log^{\alpha}(n)\ \log^{\beta}(\log n),$$

where $\alpha \geq 1, \beta$ are non-negative exponents. We note that the constants C_K and C_F are usually "big".

4. Computation of invariant manifolds and normal forms

In this section we review different methods to solve formally (4), that is, to find power series expansions of invariant manifolds attached to fixed points of vector fields. We also determine the complexity of the derived algorithms, specifically for simple vector fields.

4.1. **The invariance equation.** To start with, let

$$\dot{z} = F(z)\ , \tag{8}$$

be an n-dimensional vector field, where $z = (z_1, \ldots, z_n)$. Given a fixed point z^0, with a d-dimensional invariant subspace for the linearization $\dot{z} = \mathrm{D}F(z^0)z$ around z^0, we want to associate an invariant manifold for the whole system tangent to such linear subspace.

Without loss of generality, we assume that

- The fixed point is the origin: $F(0) = 0$;
- The linearization has a block diagonal form:

$$\mathrm{D}F(0) = \begin{pmatrix} A_1 & B \\ 0 & A_2 \end{pmatrix} ,$$

with respect to the splitting $z = (x, y)$, $x = (x_1, \ldots, x_d)$, $y = (y_1, \ldots, y_{n-d})$, where $d \leq n$.

These assumptions are not restrictions, because we can obtain this formulation around a fixed point by using affine maps, which are negligible in computations. We then write the vector field (8) as

$$\begin{aligned} \dot{x} = F^x(x, y) &= A_1 x + By + F^x_{\geq 2}(x, y) \\ \dot{y} = F^y(x, y) &= \qquad A_2 y + F^y_{\geq 2}(x, y) \ . \end{aligned} \tag{9}$$

In the following, we will also use notations such as v^x or v^y to denote the projections of $v \in \mathbb{R}^n$ in x or y components, respectively.

For the sake of simplicity, we also assume that $A_1 = \mathrm{diag}(\lambda_1, \ldots, \lambda_d)$ and $A_2 = \mathrm{diag}(\mu_1, \ldots, \mu_{n-d})$. If there are complex eigenvalues, we appeal to the complexification trick. Even if most of what follows works also for triangular matrices A_1,A_2, as in the Jordan normal form, or even for general matrices, the computer implementation of the algorithms is not so straightforward. See [4, 45].

Let $z = \Phi(s)$ be a parameterization of the invariant manifold, where $s = (s_1, \ldots, s_d)$ are the coordinates, with $\Phi(0) = 0$. The motion on the manifold is described by the vector field $\dot{s} = f(s)$, with $f(0) = 0$. The invariance equation is

$$F(\Phi(s)) = \mathrm{D}\Phi(s) f(s) \ , \tag{10}$$

that we consider in terms of power series (for the unknowns Φ and f). So, we write

$$\begin{aligned} x = \varphi(s) &= s + \sum_{k \geq 2} \varphi_k(s) \ , \\ y = \psi(s) &= \qquad \sum_{k \geq 2} \psi_k(s) \ , \end{aligned} \tag{11}$$

for the parameterization and

$$f(s) = A_1 s + \sum_{k \geq 2} f_k(s) \ , \tag{12}$$

for the reduced vector field on the manifold.

Notice that if the vector field $\dot{z} = F(z)$ is simple, the composition in the left hand side of (10), $F(\Phi(s))$, up to degree k has a low cost: $c\ p_d(k)$, where c is the complexity of the system F.

4.2. **The homological equations.** The standard procedure to solve (formally) (10) is substituting the expansions (11) and (12) in (10), and find their homogeneous terms order by order. Notice that first order terms of both (11) and (12) are already known.

In the step $k > 1$, we want to compute $\Phi_k(s) = (\varphi_k(s), \psi_k(s))^\top$ and $f_k(s)$, assuming that we have already computed $\Phi_{<k}(s)$ and $f_{<k}(s)$. Moreover, when starting the step k we have already computed and stored from the previous steps the left hand side of (10) up to order $k-1$, that is, $[F(\Phi_{<k}(s))]_{<k}$.

The first computation we have to do is $[F(\Phi_{<k}(s))]_k$. In fact, when doing this computation we not only have $[F(\Phi_{<k}(s))]_{<k}$, but also all the intermediate steps in the evaluation of F. In case F is *simple*, we have stored all the results of the simple operations up to order $k-1$. So the only thing we have to do is obtaining the k-order terms of these intermediate operations. This is done using AD tools.

Then, the k-order terms in the invariance equation (10) lead us to the k-order *homological equation*

(13)
$$\mathrm{D}\Phi_k(s)A_1 s - A\Phi_k(s) + \mathrm{D}\Phi_1(s)f_k(s) = [F(\Phi_{<k}(s))]_k - \sum_{l=2}^{k-1} \mathrm{D}\Phi_{k-l+1}(s) f_l(s)\ ,$$

where the unknowns are $\varphi_k(s)$, $\psi_k(s)$ and $f_k(s)$. The splitting of the equation in (x, y) variables is

$$\mathrm{D}\varphi_k(s)A_1 s - A_1\varphi_k(s) + f_k(s) = R_k^x(s) + B\psi_k(s)\ , \tag{14}$$
$$\mathrm{D}\psi_k(s)A_1 s - A_2\psi_k(s) = R_k^y(s)\ , \tag{15}$$

where $R_k(s)$ denotes the right hand side of (13), that is known from the previous order terms.

After solving (13) (or (14) and (15)), we compute $[F(\Phi_{\le k}(s))]_{\le k}$ just adding $A\Phi_k(s)$ to $[F(\Phi_{<k}(s))]_k$.

Let us consider now the solution of (13). From now on, we will also use the notation

$$\Phi(s) = (\underbrace{\varphi^1(s), \ldots, \varphi^d(s)}_{\varphi(s))^\top}, \underbrace{\psi^1(s), \ldots, \psi^{n-d}(s)}_{\psi(s)^\top})^\top.$$

First, we solve (15). We denote $\hat{\psi}_k(s) = R_k^y(s)$. Since the matrices A_1, A_2 are diagonal, then we can also diagonalize the homological equations (15), writing for $j = 1, \ldots, n-d$

$$\lambda_1 \frac{\partial \psi_k^j}{\partial s_1} s_1 + \cdots + \lambda_d \frac{\partial \psi_k^j}{\partial s_d} s_d - \mu_j \psi_k^j(s) = \hat{\psi}_k^j(s)\ .$$

If $\psi_k^j(s) = \sum_{|m|=k} \psi_m^j s^m$ and $\hat{\psi}_k^j(s) = \sum_{|m|=k} \hat{\psi}_m^j s^m$, then

$$\psi_m^j = \frac{\hat{\psi}_m^j}{\lambda \cdot m - \mu_j}$$

provided that the denominators are non zero. So, we can solve (15) (and (13)) at all orders if $\forall |m| \geq 2\, , \forall j = 1, \ldots, n-d, \lambda \cdot m - \mu_j \neq 0$. We will refer to the pairs $(m, j) \in \mathbb{N}^d \times \{1, \ldots, n-d\}$ with $|m| \geq 2$ such that $\lambda \cdot m - \mu_j = 0$ as *primary resonances*. These are obstructions to solve the homological equations, and so to the computation of the expansions of the paramaterizations. So, we will refer to the non-existence of primary resonances as the *primary non-resonance condition*.

Notice that there many ways of grouping the eigenvalues for which the primary non-resonance condition holds. Among them:

- stable manifold: $\mathrm{Re}\lambda_i < 0, \mathrm{Re}\mu_j \geq 0$;
- unstable manifold: $\mathrm{Re}\lambda_i > 0, \mathrm{Re}\mu_j \leq 0$;
- center manifold: $\mathrm{Re}\lambda_i = 0, \mathrm{Re}\mu_j \neq 0$;
- center-stable manifold: $\mathrm{Re}\lambda_i \leq 0, \mathrm{Re}\mu_j > 0$;
- center-unstable manifold: $\mathrm{Re}\lambda_i \geq 0, \mathrm{Re}\mu_j < 0$;

where Re denotes the real part of a complex number.

Let us consider (14), and denote $\hat{\varphi}_k(s) = R_k^x(s) + B\psi_k(s)$ (at this point, we already know $\psi_k(s)$). Again, we split (14) in d equations, writing for $i = 1, \ldots, d$,

$$\lambda_1 \frac{\partial \varphi_k^i}{\partial s_1} s_1 + \cdots + \lambda_d \frac{\partial \varphi_k^i}{\partial s_d} s_d - \lambda_i \varphi_k^i(s) + f_k^i(s) = \hat{\varphi}_k^i(s)\ .$$

We have some freedom to solve it, and there are several ways of finding solutions. Among them:

- *Graph transform method.* We take at each step $\varphi_k(s) = 0$ and $f_k(s) = \hat{\varphi}_k(s)$. This corresponds to obtain the invariant manifold as a graph $y = \psi(x) = \sum_{k \geq 2} \psi_k(x)$. Notice that the construction is equivalent to solve $\mathrm{D}\psi(x) f(x, \psi(x)) = g(x, \psi(x))$, and get $f(x) = f(x, \psi(x))$.
- *Parameterization method.* We reduce the dynamics on the manifold, as obtaining a normal form for f (in fact, if $n = d$ this is doing a normal form of the vector field). To do so, for $|m| = k$ and $i = 1, \ldots, d$, we choose

$$f_m^i = 0\ ,\ \varphi_m^i = \frac{\hat{\varphi}_m^i}{\lambda \cdot m - \lambda_i}, \quad \text{if } \lambda \cdot m - \lambda_i \neq 0; \tag{16}$$

$$f_m^i = \hat{\varphi}_m^i\ ,\ \varphi_m^i = 0, \quad \text{if } \lambda \cdot m - \lambda_i = 0. \tag{17}$$

We refer to the pairs $(m, i) \in \mathbb{N}^d \times \{1, \dots, d\}$ with $|m| \geq 2$ such that $\lambda \cdot m - \lambda_i = 0$ as secondary resonances.

In summary, primary resonances $\lambda \cdot m - \mu_j = 0$ are obstructions to solve the homological equations and compute the expansions, while secondary resonances $\lambda \cdot m - \lambda_i = 0$ are obstructions to linearization (or simplification) of the dynamics on the manifold.

Remark 4.1. If the matrices A_1, A_2 are not diagonal, the non-resonance conditions are the same, but the solution of the homological equations is harder [4, 45]. A possibility is using BLAS (Basic Linear Algebra Subprograms), included e.g. in LAPACK and GSL.

For instance, if we compute the stable manifold, there are not primary resonances and only a finite number of secondary resonances, so the equations on the stable manifold can be reduced to polynomial (or even to linear, if there are no secondary resonances), just like in the Poincaré-Dulac normal form. We can also use the parameterization method if the eigenvalues $\lambda_1, \dots, \lambda_d$ have negative real part, and there are no primary resonances. If the eigenvalues are those that have "less negative" real part, the invariant manifold is a slow manifold, that dominates the dynamics on the stable manifold [7]. Similar considerations work for the case of eigenvalues with positive real part.

If we compute the center manifold, that is the eigenvalues $\lambda_1, \dots, \lambda_d$ are those with zero real part, there are infinitely many secondary resonances, so the graph method is a good choice. (There are cases in which one can use a more refined parameterization method for a center manifold, to eliminate "most" of the secondary resonances. See [3, 2]). Even if generically the power series expansions of center manifolds are divergent, they provide good local approximations.

One can also used a mixed strategies. For instance, in the parameterization of a center-stable manifold one can distinguish between center parameters and stable parameters, for which one uses a combination of the graph representation and the parameterization method.

Finally, we emphasize that the freedom to solve the homological equations is also suitable for stability of numerical computations. One can save the close to resonance coefficients, in order to avoid small divisors. This is also useful in parameter depending models crossing a resonance, since one gets smooth dependence on parameters of the parameterizations and dynamics of the manifold.

4.3. **Computational cost.** We will estimate now the computational cost of solving the invariance equation (10) up to order k. The solutions

are either paramaterizations of invariant manifolds ($d < n$) or normal form transformations ($d = n$), with different styles.

The cost of solving the homological equations at each step is negligible (e.g. for diagonal matrices A_1 and A_2), so at each step the cost is concentrated in the right hand side of (13). There are two terms, one is the composition of the parameterization Φ with the model F, and the second involves both Φ and f.

As a result, the computation up to order k of the power series expansion of the parameterization of a d-dimensional invariant manifold of a simple n dimensional vector field of complexity c, and the corresponding reduced vector field, has a cost $\mathrm{w}_{n,d}(k)$ that is

$$\mathrm{w}_{n,d}(k) \sim c\ \mathrm{p}_d(k) + nd\ \mathrm{p}_d(k).$$

In fact, this is an overestimate of the cost, and specific methods (i.e. graph and parameterization) give rise to lower estimates.

- In the graph transform method, $\Phi(s) = (s, \psi(s))^\top$, so in the computation of $\mathrm{D}\Phi(s)f(s)$ one has to do $(n-d)d$ products, the cost is

$$\mathrm{w}_{n,d}(k) \sim c\ \mathrm{p}_d(k) + (n-d)d\ \mathrm{p}_d(k). \tag{18}$$

- In the parameterization method, if there are only a finite number of secondary resonances, so then we can get $f(s)$ of polynomial type, the cost is

$$\mathrm{w}_{n,d}(k) \sim c\ \mathrm{p}_d(k), \tag{19}$$

  because multiplying a power series with a polynomial (which has bounded degree, and in practice assumed to be small w.r.t. k) has a negligible computational cost. Notice that $d = n$ in the case of computation of the Poincaré-Dulac normal form.

In summary, we obtain the following estimate.

Proposition 4.2. *Let* $\mathrm{w}_{n,d}(k)$ *be the computational cost of solving* (10) *up to order* k*. Then,*

$$\mathrm{w}_{n,d}(k) \sim C\mathrm{p}_d(k),$$

where the constant C *depends only on* n*,* d*, the complexity of the simple vector field* F *(c), and the style of parameterization (and* $n = d$ *for normal forms, total or partial).*

5. Some implementation details of a symbolic manipulator

In this section we explain some details of actual implementations of a computer package to manipulate multivariate power series using AD tools. This is the core of the programs to compute invariant manifolds

and normal forms. We have implemented the algorithms using the programming language C.

5.1. **On the implementation.** In our implementation of the symbolic manipulator, we use a combination of vector and tree data structures to handle homogeneous polynomials, that involves the idea of recursivity. In a few words, a homogeneous polynomial of d variables $x = (x_1, \ldots, x_d)$ of degree k is a combination of $k+1$ homogeneous polynomials of the firts $d-1$ variables $\hat{x} = (x_1, \ldots, x_{d-1})$ of degrees $k, k-1, \ldots, 0$:

$$f_k(x) = f_k^d(\hat{x}) + f_{k-1}^d(\hat{x})x_d + \cdots + f_0^d(\hat{x})x_d^k \ .$$

The coefficients are ordered in the vector following such scheme, that is using a lexicographical order. This methodology allows us to construct a very general symbolic manipulator, that works in principle for any number of variables (save the limits imposed by the computer memory capacity). Moreover, one avoids the use of indexing routines to locate coefficients and hash tables, an approach that is used e.g. in [35, 16].

The vector representation is used e.g. when implementing the addition or the scalar multiplication of homogeneous polynomials, while the tree representation is specially useful when implementing the product of homogeneous polynomials. The routines for the product are written in recursive form, making the codes easy to write and easy to read, at least when using the naive algorithm.

5.2. **Benchmark for the truncated product.** In Table 1 we display several execution times, in seconds, for making the truncated product of polynomials (d is the number of variables and k is the degree). The velocity is measured in megaflops, i.e. millions of operations per second. The computation has been carried out in two machines, which we call Mac I and Mac II. See below for details.

There are several points the reader has to take into account when reading the results:

- The algorithm: we use the naive formula for the truncated product of polynomials, so the number of operations, $\mathrm{p}_d(k)$, counts all the multiplications of the coefficients in the definition of the product. Notice that a single operation (what we define here as one flop) involves one multiplication of two monomials, locating the monomial in the result, and finally the addition of monomials.
- The implementation: we have written our programs in C, making the implementation quite efficient both in computer memory management and execution time. The use of commercial

symbolic manipulators like Mathematica or Maple, etc. could multiply the running time by several hundreds or even thousands.

- The coefficients: these are real numbers using double-precision floating-point arithmetic, the variable type `double` in C. This amounts 8 bytes per coefficient. Changing to other types, like `long double`, `complex`, etc. slow the times.
- The computer: we use two Macs for doing the benchmarks, both running under the operating systems Mac OS X 10.4.11 (Tiger). Their technical specifications are:
  - Mac I: MacBook Pro 15"; Processor: 2.16 GHz Intel Core Duo, 2 MB L2 Cache; Memory: 2 GB 667 MHz DDR2 SDRAM;
  - Mac II: iMac; 2 GHz Intel Core Due, 4MB L2 Cache; Memory: 1 GB 667 MHz DDR2 SDRAM.
- The compiler: we use `gcc`, version 4.0.1, with different options and flags. Finding better options for the actual computer is a kind of art. In fact, we found that for both Macs the options and flags should be different to improve the performance of the programs. Use of other compilers, like `icc`, can also change running times.
- The plug: execution time can vary a lot if a laptop is not plugged in and work with the battery (depending on the system preferences with respect to the battery).

The list of contingencies does not finish here.

Notice that, in Table 1, the number of megaflops depends on the number d of variables of the polynomials, and grows with respect to the order of the truncated product. So, it seems that time is sublinear in the number of operations (or subquadratic on the degree). But, using the convolution formula, the cost of the truncated product should be linear in the number of operations (or quadratic in the degree). We have fit the timings in Table 1 (in fact, an extended table including k's that are multiple of 5), with respect to $n = \mathrm{n}_d$ with functions of the form $t(n) = A\, n^b$, and the results for the Mac I computer are shown in log-log scale in Figure 1. The estimates of the parameters $a = \log_{10} A$ and b are:

- For $d = 3$: $a = -8.57 \pm 0.06$, $b = 1.72 \pm 0.01$;
- For $d = 4$: $a = -8.77 \pm 0.06$, $b = 1.73 \pm 0.01$;
- For $d = 5$: $a = -8.85 \pm 0.05$, $b = 1.71 \pm 0.01$;
- For $d = 6$: $a = -8.93 \pm 0.06$, $b = 1.70 \pm 0.01$.

				Mac I		Mac II	
d	k	$n_d(k)$	$p_d(k)$	time (s)	Mflops	time (s)	Mflops
3	10	286	8008	4.956e–05	161.6	4.080e–05	196.3
3	20	1771	230230	8.984e–04	256.3	7.595e–04	303.1
3	30	5456	1947792	5.916e–03	329.3	4.977e–03	391.3
3	40	12341	9366819	2.458e–02	381.1	2.072e–02	452.1
3	50	23426	32468436	7.744e–02	419.3	6.564e–02	494.7
3	70	62196	218618940	4.650e–01	470.1	4.005e–01	545.9
3	80	91881	470155077	9.630e–01	488.2	8.390e–01	560.4
3	90	129766	927048304	1.848e+00	501.6	1.620e+00	572.3
3	100	176851	1705904746	3.323e+00	513.3	2.937e+00	580.9
4	10	1001	43758	3.019e–04	145.0	2.458e–04	178.0
4	20	10626	3108105	1.462e–02	212.5	1.181e–02	263.2
4	30	46376	48903492	1.773e–01	275.9	1.457e–01	335.7
4	40	135751	377348994	1.157e+00	326.2	9.592e–01	393.4
4	50	316251	1916797311	5.237e+00	366.0	4.377e+00	438.0
4	60	635376	7392009768	1.857e+01	398.1	1.563e+01	472.9
4	70	1150626	23446881315	5.528e+01	424.1	4.687e+01	500.3
4	80	1929501	64276915527	1.443e+02	445.3	1.232e+02	521.6
4	90	3049501	157366449604	3.399e+02	463.0	2.923e+02	538.3
4	100	4598126	352025629371	7.377e+02	477.2	6.384e+02	551.4
5	10	3003	184756	1.352e–03	136.7	1.122e–03	164.6
5	20	53130	30045015	1.624e–01	185.0	1.332e–01	225.5
5	30	324632	847660528	3.565e+00	237.8	2.945e+00	287.8
5	40	1221759	10272278170	3.597e+01	285.6	2.996e+01	342.9
5	50	3478761	75394027566	2.319e+02	325.1	1.937e+02	389.3
5	60	8259888	396704524216	1.109e+03	357.6	9.292e+02	426.9
5	70	17259390	1646492110120	4.274e+03	385.3	3.599e+03	457.5
6	10	8008	646646	4.863e–03	133.0	4.140e–03	156.2
6	20	230230	225792840	1.356e+00	166.6	1.110e+00	203.5
6	30	1947792	11058116888	5.226e+01	211.6	4.300e+01	257.2
6	40	9366819	206379406870	8.109e+02	254.5	6.725e+02	306.9
6	50	32468436	2160153123141	7.400e+03	291.9	6.205e+03	348.1

TABLE 1. Benchmark of the execution time of the truncated product of two polynomials of d variables up to degree k, with the computers Mac I and Mac II. The number of coefficients per polynomial is $n_d(k)$, and the total number of operations using the naive algorithm is $p_d(k)$. See the text for more details.

So, the exponent in the power law dependence of the time cost of the truncated product w.r.t. the degree is close to 1.70 (and similar estimates for the Mac II computer), rather than the theoretical estimate 2. This is possibly due to the fact that the theoretical estimate counts

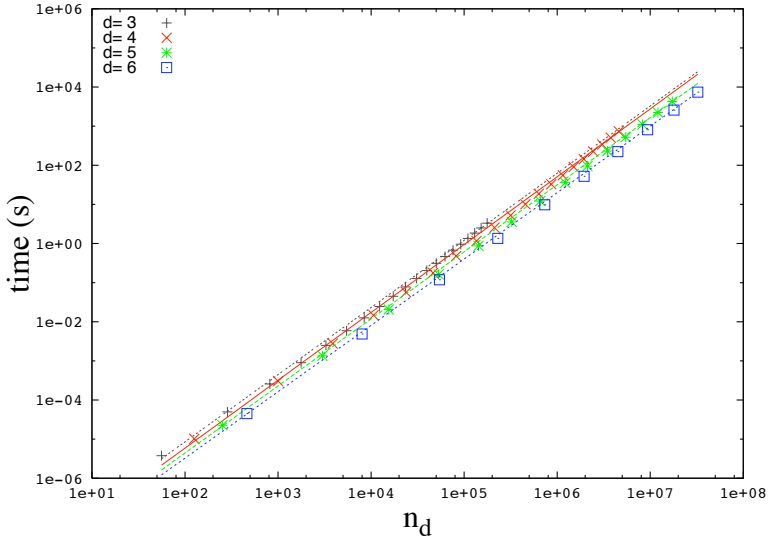


FIGURE 1. Timings of the truncated product as a function of the truncated order, in log-log scale.

the total number of arithmetics operations and addressings of the coefficients, and one assumes that the total time depends linearly on all of them. But addressing depends a lot on the type of data structure in the implementation, so its total cost is non-linear w.r.t. the number of addressed coefficients. Moreover, in modern architectures, addressing memory is performed in an orderly fashion so that CPU cache can work very efficiently, processors support multi-threading (instruction level parallelism, allowing speed up), or even processors are multi-core, etc. All these eventualities (and many others this author ignore) make the total time sublinear w.r.t. the total number of operations (for the examples in the present paper, with exponents around 0.92).

Even if we present here the timings for two Mac computers, we have also compiled and run the programs in several Linux PC's, and in a cluster system of 46 Sun Fire V20z servers (each of them with two 2.2Ghz AMD Opteron processors, 4 GB of RAM) running under Linux.

Since all the estimates of the cost of the algorithms are based on the cost of a product, one can easily make time estimates using his/her routines for the product. Of course, better algorithms and better implementations of the product of homogeneous polynomials improve the efficiency and execution times of the algorithms.

Remark 5.1. We emphasize again that, besides the computational complexity of the algorithms, there are many other factors to be considered in actual implementations of those algorithms, and depend also on the kind of problems one is dealing with. For instance, in [46] are proposed several fast algorithms for computing truncated multivariate power series, based on multipoint evaluation and interpolation techniques. When computing local expansions of invariant manifolds, it is not clear how to choose the interpolation nodes, and how the election influences the accuracy of the computations. Moreover, the computer language, the data structures, etc. are also important. In the same reference, the computation of the truncated exponential up to order 10 of a 10-variate power series, using double-precision floating-point arithmetic in Mathematica on a PC, took hours (a few years ago), while with our C programs based on the slow classical algorithms the same computation took around 0.20 seconds.

6. An example: invariant manifolds in the Restricted Three Body Problem

In this section we present a specific example of computation of invariant manifolds: center, center-stable and center-unstable manifolds of a collinear equilibrium point in the Restricted Three Body Problem. The motivation is two-fold: the example is non-trivial (since the manifolds are 4D and 5D, in a 6D phase space), and computation of these manifolds has a long tradition in the astrodynamics literature (see [17, 53, 36, 35, 21, 23, 18, 19, 35, 20] and a long etcetera). Notably, similar computations have been also done in studying some problems in chemical dynamics [56]. Moreover, we also mention that there are more recent studies including globalization of center manifolds using numerical computation of the periodic orbits and invariant tori laying in its interior [22, 44].

So, by no means we intend to redo all these exhaustive studies, which have been carried out with a rather ad hoc methodology. Our intention is just to show how our methodology is effective to do computations of invariant manifolds in this and many other non-trivial problems.

6.1. **A brief description of the RTBP.**

Although the Restricted Three Body Problem (RTBP for short) needs no presentation, we will say a few words, mainly to fix notation. It deals with the motion of a massless body under the gravitational forces induced by two punctual masses, usually called primaries, that evolve in circular Keplerian motion around the center of mass. One considers a rotating coordinate system with origin in the center of mass of the primaries, in which the

primaries are fixed in the x axis and the z axis is perpendicular to the ecliptic plane. Moreover, one scales units such that one year is 2π and the total mass of the primaries is 1. So, the primaries have masses $1-\mu$ and μ, with $\mu \in]0, \frac{1}{2}]$, and are located at the points $(\mu, 0, 0)$ and $(\mu - 1, 0, 0)$, respectively. With these conventions, the motion of the massless body is described by the three degrees of freedom Hamiltonian

$$H(x, y, z, p_x, p_y, p_z) = \frac{1}{2}(p_x^2 + p_y^2 + p_z^2) + y\ p_x - x\ p_y + V(x, y, z) \tag{20}$$

where V is the gravitational potential

$$V(x, y, z) = -\frac{1-\mu}{r_1} - \frac{\mu}{r_2},$$

being $r_1 = \sqrt{(x-\mu)^2 + y^2 + z^2}$ and $r_2 = \sqrt{(x-\mu+1)^2 + y^2 + z^2}$ the distances of the body to the primaries. The equations of motion are

$$\begin{aligned}
\dot{x} &= p_x + y\ , & \dot{p}_x &= p_y - \frac{1-\mu}{r_1^3}(x-\mu) - \frac{\mu}{r_2^3}(x-\mu+1)\ , \\
\dot{y} &= p_y - x\ , & \dot{p}_y &= -p_x - \frac{1-\mu}{r_1^3} y - \frac{\mu}{r_2^3} y\ , \\
\dot{z} &= p_z\ , & \dot{p}_z &= -\frac{1-\mu}{r_1^3} z - \frac{\mu}{r_2^3} z\ .
\end{aligned} \tag{21}$$

Notice that the equations of the RTBP are simple. In fact, the complexity of the vector field (21) is $c_{\mathrm{vf}} = 6.5$, and the complexity of the Hamiltonian (20) is $c_{\mathrm{H}} = 8$.

6.2. **Computation of center manifolds in the RTBP.** We have applied our algorithms to compute the expansions of the invariant manifolds of the equilibrium points of the RTBP. Once one has chosen the mass parameter μ, the equilibrium point L_p ($p = 1, 2, 3, 4, 5$), the dimension of the manifold d and the order of its expansion k, the tangent space at the origin (the suitable (complex) eigenvectors of the differential at the fixed point), and the representation of the manifold (in the following, a graph), the computer program produces the coefficients of the (complex) power series expansions of the (in the present example) graph representation of the manifold.

The mass parameter we have chosen to illustrate the methodology is $\mu = 0.0121505816234$, that corresponds to the Earth-Moon system. For such a case the collinear points L_1, L_2, L_3 are of center×center×saddle type, so they have attached 4D center manifolds, 5D center-(un)stable

		Mac I			Mac II		
d	k	product	graph	ratio	product	graph	ratio
4	10	4.352e–04	7.790e–03	17.90	3.841e–04	6.090e–03	15.86
4	20	2.533e–02	4.048e–01	15.98	2.054e–02	3.039e–01	14.80
4	30	3.582e–01	5.497e+00	15.34	2.650e–01	3.819e+00	14.41
4	40	2.590e+00	3.921e+01	15.14	1.813e+00	2.641e+01	14.57
4	50	1.259e+01	1.900e+02	15.09	8.500e+00	1.243e+02	14.62
4	60	4.708e+01	7.104e+02	15.08	3.104e+01	4.555e+02	14.67
4	70	1.460e+02	2.207e+03	15.12	9.481e+01	1.397e+03	14.73
5	10	1.924e–03	2.640e–02	11.84	1.671e–03	2.070e–02	12.39
5	20	2.646e–01	3.176e+00	12.00	2.203e–01	2.412e+00	10.95
5	30	6.630e+00	7.851e+01	11.84	5.140e+00	5.702e+01	11.09
5	40	7.425e+01	8.786e+02	11.83	5.431e+01	6.151e+02	11.33

TABLE 2. Benchmark of execution time for constructing expansions of the center manifold ($d = 4$) and the center stable manifold ($d = 5$), using the graph method, of the L_1 point of the RTBP of the Earth-Moon system ($\mu = 0.0121505816234$). The computations have been done with the machines Mac I and Mac II (see the text for details).

manifolds and 1D (un)stable manifolds. The triangular points are linearly stable, so will be not considered here.

Even if for this example we have chosen the graph transform method, there are still some freedom to be fixed. In this case, we can choose the lenght of the eigenvectors (which is equivalent to scale the parameterization). These choices are made to improve the numerical stability in the whole procedure, controlling the growth of the coefficients in the expansions. In this particular example, the natural choice is taking the lengths of the eigenvectors as the distance of the equilibrium point to the nearest primary. This is the standard.

In Table 2 we show some benchmarks of execution time of the computation of the center manifold ($d = 4$) and the center stable manifold ($d = 5$) of the L_1 point. We include the ration between the total time of computation and the time of the truncated product of complex power series (up to order k). Notice that the theoretical ratio between the execution time of the graph and the execution time of the truncated product is $(6.5 + 2 * 4) = 14.5$ for the center manifold, and $(6.5 + 1 * 5) = 11.5$ for the center-stable manifold. See (18). Our numerical experiment produces close estimates of such ratios.

Remark 6.1. We emphasize again the the computation has been done using complex polynomials. So, the execution times of the truncated product of polynomials shown in Table 2 roughly doubles the ones appearing in Table 1 (and this depends a lot on the processor). The execution time for the computation of the expansions of the manifold could be halved if we used real polynomials, either solving real homological equations or translating them into complex numbers. These methods were used e.g. in [27].

Remark 6.2. Special symmetries of the problem are not used, so adapted implementations of the symbolic manipulator can halve execution times. See e.g. [35] for this approach.

6.3. **Growth of the coefficients of the center manifold.** Even if the center manifold is not analytic, the coefficients of its power series expansions can grow in a mild way. If $\ell_1(k)$ equals the maximum of the ℓ_1 norms of the two vector of coefficients of the two k-order homogeneous polynomial of the two components of the expansion of the graph of the center manifold, then it seems to be that

$$\ell_1(k) \sim \ell(k) = A\lambda^k (\log k)^{ck}.$$

This growth estimate was rigorously established in the unpublished manuscript [34] for the coefficients in partial normal forms computations of the center manifold.

Figure 2 shows the fit of $\sqrt[k]{\ell_1(k)}$ to the function $\sqrt[k]{\ell(k)} = \sqrt[k]{A}\lambda(\log k)^c$ via the parameters $a = \log A$,$b = \log \lambda$ and c, where

$$a = -1.25 \pm 0.05\ ,\ b = -0.212 \pm 0.008\ ,\ c = 0.252 \pm 0.005\ .$$

Similar behaviour is observed in the growth of the coefficients when considering the four components of the reduced vector field on the center manifold. Notice that if the expansions were analytic the constant c should be 0. Notice also that if the expansions where of Gevrey class, the term $(\log k)^k$ should be $k!$.

6.4. **Dynamics on the center manifold.** The center manifold of the L_1 point is a 4D manifold, and we are able of computing a high order approximation of a parameterization of the manifold Φ and its dynamics f. Notice that the computation does not rely on geometric properties of the dynamics, and in particular on its Hamiltonian feature. The algorithms work in a general setting.

But in the present example the dynamics is Hamiltonian, so one can obtain additional information to study it. In particular, the 4D reduced vector field on the center manifold, f, is Hamiltonian, with

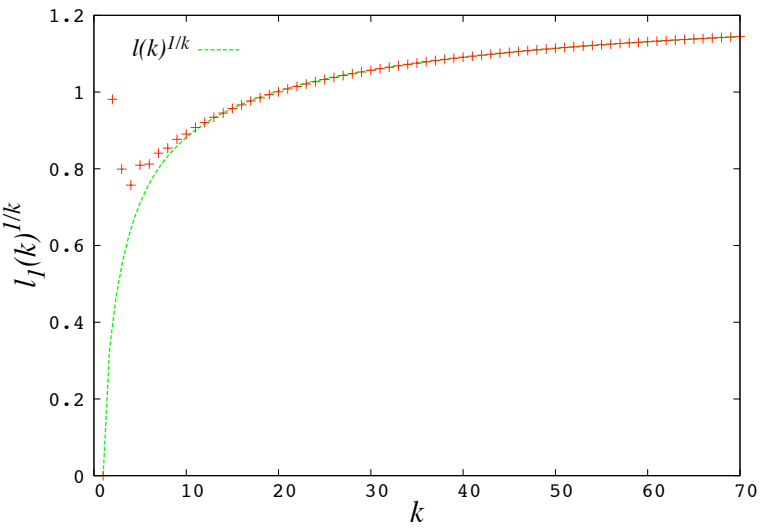


FIGURE 2. Fit of k-root of the ℓ_1 norm of the k-order terms of the expansion of the center manifold with the function $\ell(k)^{1/k} = A^{1/k}\lambda(\log k)^c$ via the parameters $a = \log A$,$b = \log \lambda$,c.

Hamiltonian $H\circ\Phi$, with respect to the restricted symplectic form $\Phi^*\omega$. In particular $H\circ\Phi$ is a conserved quantity on the center manifold.

Remark 6.3. Using Darboux theorem (see e.g. [1]), we could reparameterize the center manifold in such a way the restricted symplectic form is the standard. See [27] for an example. In [13] is described an algorithm which computes the parameterization of the center manifold giving the (standard) symplectic structure on the center manifold.

A nowadays standard practice to study the dynamics of the 4D reduced Hamiltonian vector field f on the center manifold is using the Poincaré section trick on different Hamiltonian levels to obtain a collection of 2D plots. This technique was already used e.g. in [17, 36] (see also [21, 23, 18, 19] and [35]), and has been also succesfully used to study some problems in molecular dynamics [56, 15]. In these references the reduction of the dynamics to the center manifold follows a partial normal form strategy, so working in a 6D phase space to finally consider a 4D reduced vector field. In this view, the methodology is quite expensive, since the invariant objects to be considered are only 4D (and one has to deal with 6-variate power series). But once one has obtained (an approximation of the) reduced dynamics, the steps to perform a study of the reduced dynamics follows similar lines.

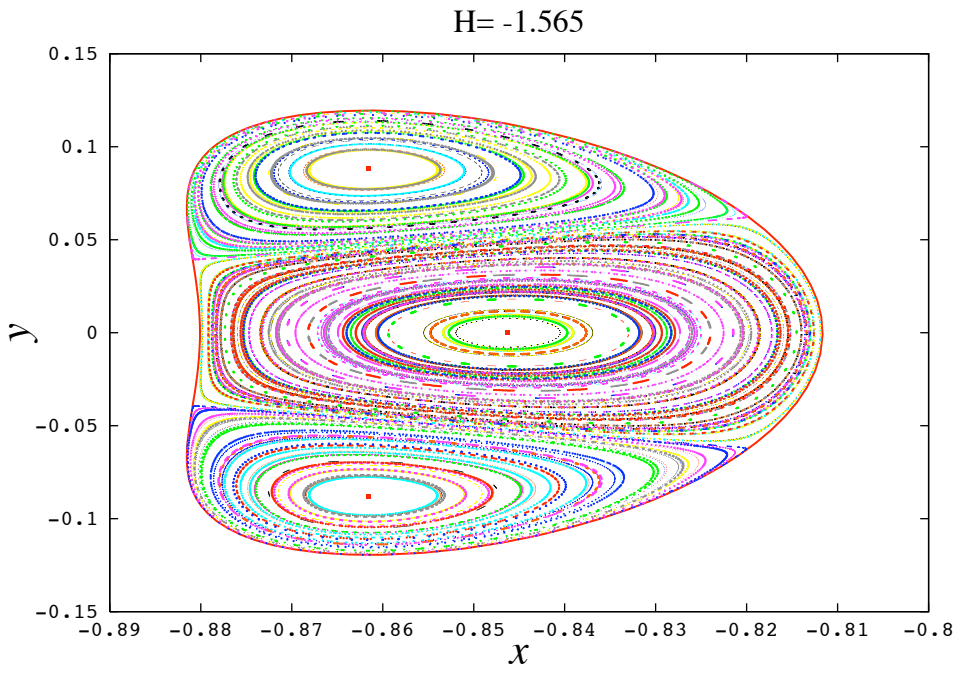


FIGURE 3. Poincaré section at the energy $H = -1.565$.

For the sake of completeness, we present in Figure 3 one phase portrait of the 2D Poincaré section with $z = 0$ of the reduced dynamics on the center manifold at the Hamiltonian level $H = -1.565$, visualizing the typical features of 2D area preserving maps. Similar pictures are shown in the references mentioned above. Even if we have computed the expansion of both the center manifold and its dynamics up to order 70, for such an energy level it is enough to consider expansions up to order 50 in order to have a "good" accuracy in the computations, say the invariance equation and the preservation of the Hamiltonian is typically obtained with an error less than 10^{-8}. See the next section. (We emphasize that the closer to $H_{L_1} \simeq -1.594171$ the energy level H is, the lower order is needed. For $H = -1.580$, order 20 is enough, and for $H = -1.560$, the whole series up to order 70 is needed in order to get good accuracy).

The integration of the orbits along the center manifold (and their Poincaré sections) is performed integrating the reduced vector field, using a Runge-Kutta method of order 7-8 with automatic step size control (to get a local error of 10^{-15}). Since at each step of the numerical integration method it has to evaluate the field at different points (in fact, 13 points, and in this case these are 4D), and evaluation of power

series is a costly task, the whole Poincaré return map is quite time consuming (depending on the order of the expansions). Fortunately, computing several iterations of the Poincaré map (in Figure 3, 256 iterations), for a collection of points in the center manifold (in Figure 3, 96 points) can be easily parallellized in a cluster of computers. For our computations, we have used a 46 node AMD opteron cluster system. For the example shown in Figure 3, with $H = -1.565$ and using power series up to order 50, each return map took around 45 seconds, but for $H = -1.580$ with series up to order 20, each map took less than one second.

In Figure 3 we observe the planar Lyapunov orbit, which is the boundary of the center manifold at such energy level, and in this case it is unstable (even on the center manifold). We also observe the section of the vertical Lyapunov periodic orbit on the x axis and both halo periodic orbits, surrounded by 1D invariant curves that correspond to 2D invariant tori (around the vertical Lyapunov periodic orbit, these are the Lissajous orbits; and arround the halo orbits, these are the quasi-halo orbits). The 3D view of these periodic orbits at such an energy level $H = -1.565$ is shown in Figure 4. We refer the readers to the references mentioned above for exhaustive studies, including applications to astrodynamics and molecular dynamics.

6.5. **Error estimates.** We have already mention that error bounds depend on the energy level and the orbits themselves. For an orbit $s(t)$ on the invariant manifold, one can consider several error estimates, such as the error in the invariance condition along the orbit,

$$e_I(t) = \|F(\Phi(s(t))) - \mathrm{D}\Phi(s(t))f(s(t))\|_\infty$$

and the error in the preservation of the Hamiltonian

$$e_H(t) = |H(\Phi(s(t))) - H(\Phi(s(0)))|.$$

Since the orbit $\Phi(s(t))$ in the phase space should be a solution of the original vector field $\dot{Z} = F(Z)$, one can also integrate both the reduced field f from $s(0)$ and the whole field F from $Z(0) = \Phi(s(0))$ and compare the produced orbits. We then produce the orbital error estimate

$$e_O(t) = \|\Phi(s(t)) - Z(t)\|_\infty.$$

We emphasize that this error has to be measured in small elapses of time, since for orbits in the center manifold the "hyperbolic" directions produce error growth. This is also an standard practice.

Notice that in the computation of the center manifold and its dynamics we do not use explicitly the Hamiltonian characteristic of the

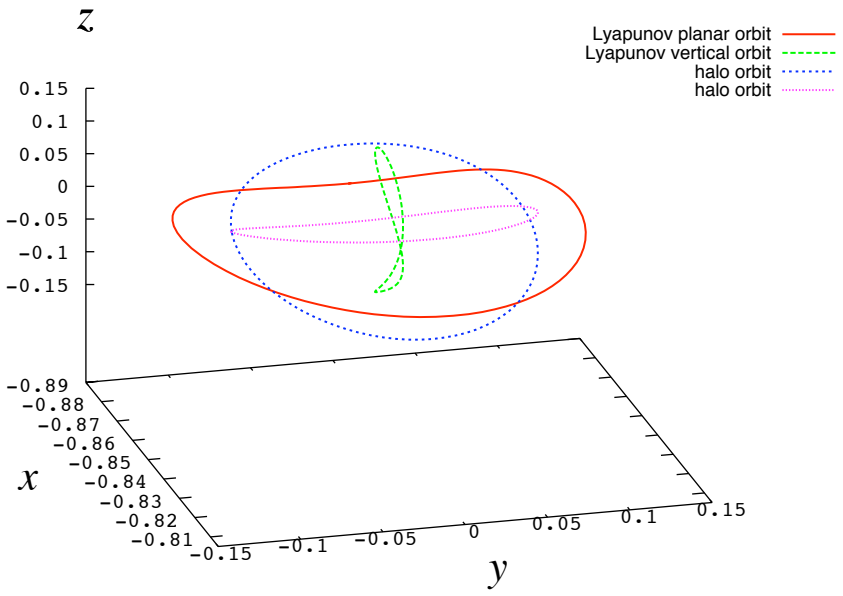


FIGURE 4. Periodic orbits on the center manifold ($H = -1.565$).

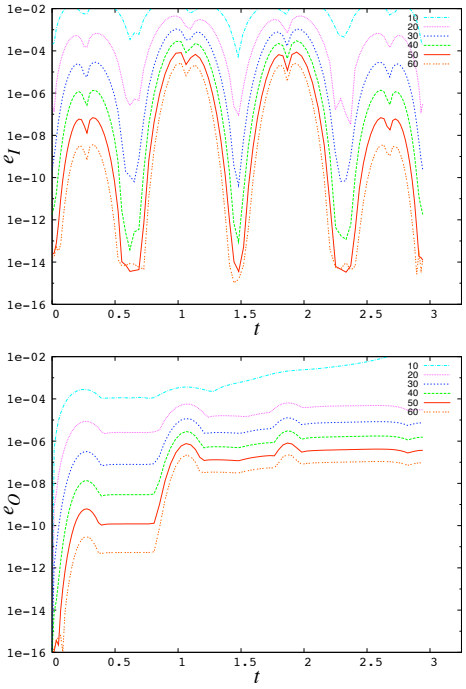
dynamics, so preservation of the Hamiltonian is not only a test for the numerical integrator, but also for the quality of the approximation of the invariant object.

In Figures 5,6,7 we show, in logarithmic scale, the error estimates e_I,e_H,e_O for, respectively, the planar Lyapunov orbit, the vertical Lyapunov orbit and one of the halo orbits (again, for $H = -1.565$). The estimates have been produced for different orders of the expansions of the center manifold (from 10 to 60), during the corresponding periods. Notice that the error bounds for the planar Lyapunov orbit are worse that those estimated for the other two periodic orbits. This is due to the fact that the planar Lyapunov orbit lays in the boundary of the center manifold and, for the energy level considered, it is unstable.

Since in this numerical exercise we have computed the periodic orbits inside the center manifold, we also show in each of the figures the computed period T, and the errors e^1_{PO} and e^2_{PO} in the return map on the manifold and on the whole phase space, respectively.

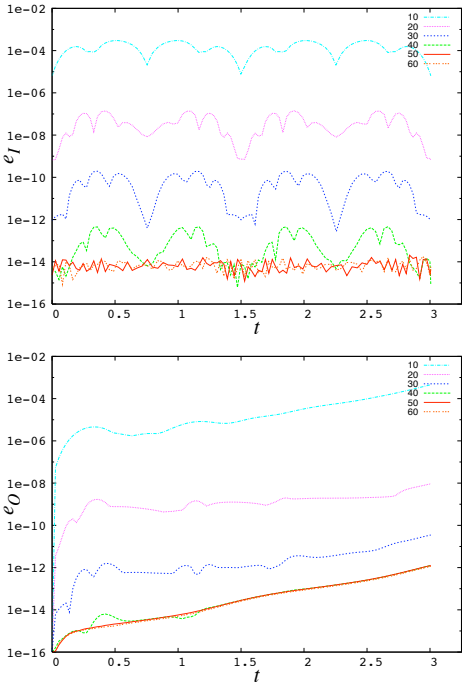
7. Conclusions, related and future work and challenges

In this paper we have presented a methodology for the computation of expansions of invariant manifolds and normal forms, associated



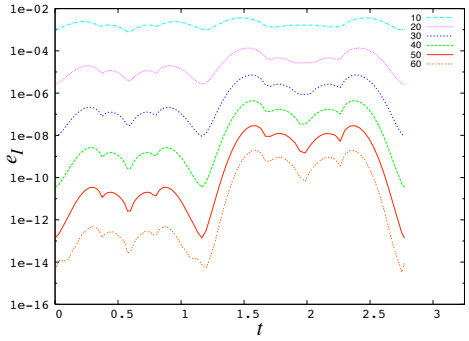
d	T	e^1_{PO}	e^2_{PO}
10	2.944806	2.48e-15	3.35e-02
20	2.943184	9.99e-16	3.03e-05
30	2.943083	1.67e-15	7.38e-06
40	2.943068	1.44e-15	1.53e-06
50	2.943065	1.67e-15	3.67e-07
60	2.943065	1.44e-15	9.53e-08

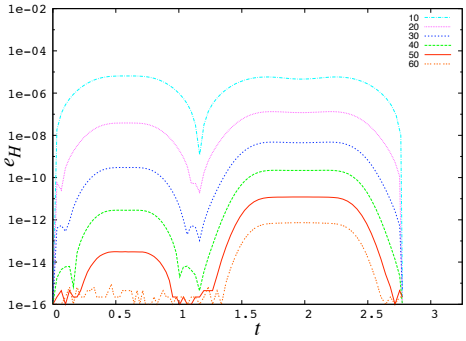
FIGURE 5. Errors for the planar Lyapunov orbit ($H = -1.565$)

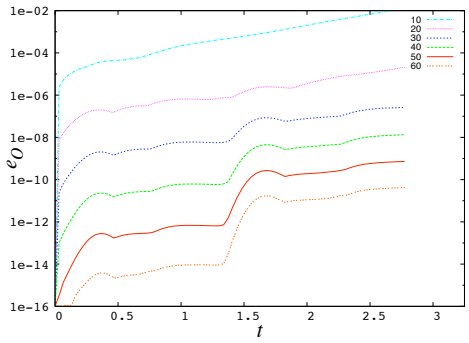


d	T	e^1_{PO}	e^2_{PO}
10	3.006224	9.99e-16	4.54e-04
20	3.006243	8.88e-16	9.33e-09
30	3.006243	7.77e-16	3.51e-11
40	3.006243	3.89e-16	1.25e-12
50	3.006243	4.37e-16	1.27e-12
60	3.006243	4.37e-16	1.14e-12

FIGURE 6. Errors for the vertical Lyapunov orbit ($H = -1.565$)







d	T	e^1_{PO}	e^2_{PO}
10	2.775478	1.22e-15	1.31e-02
20	2.774291	2.78e-16	2.11e-05
30	2.774274	2.54e-14	2.59e-07
40	2.774274	2.66e-15	1.34e-08
50	2.774274	1.33e-15	7.24e-10
60	2.774274	7.77e-16	4.23e-11

FIGURE 7. Errors for the halo orbits ($H = -1.565$)

to equilibrium points of vector fields. In all cases, the problem is reduced to solving functional equations in the space of power series. We have also evaluated the complexity of the algorithms for simple vector fields. The complexity is estimated in terms of the number of (truncated) products of power series that have to be done along the process, and in all cases this number does not depend on the order of the expansions and can be easily computed. We have also benchmarked some implementations of the algorithms, checking that the timings fit the obtained rigorous estimates.

The following is a mixture of some related issues and challenges for future work.

- The adaptation of the algorithms to discrete systems given by maps is straightforward, even if for these problems there are some inevitable compositions. We emphasize, however, that in absence of secondary resonances (or the existence of just a few of them), the parameterization method is more efficient than the graph transform method. The computation of 2D center, and 3D center-stable and center-unstable manifolds in a 4D (symplectic) map using graph transform method and AD tools was carried out in [27] (in these cases, there are infinitely many

secondary resonances). For computation of 1D and 2D invariant manifolds using the parameterization method and AD tools in 4D models in some problems of Macroeconomics, see [24].

- Hamiltonian dynamics is an important part of the area of Dynamical systems, with its own methodology. For instance, in normal form computations one has to deal with canonical transformations, or stable manifolds are Lagrangian and one can take advantage of this geometric property to improve the algorithms of power series expansions. In the ongoin paper [32] we plan a study of different methods to generate canonical transformations which, in combination with AD tools, are the base of efficient methods of computation of normal forms.
- There are also the more challenging problems of computation of invariant tori (including periodic orbits) and their associated invariant manifolds, in different types of problems (general flows or maps, quasiperiodic systems, Hamiltonian systems, etc.) For invariant tori, the most efficient methods are based on Newton, see e.g. [10], where the unknowns are represented as Fourier series. For the whiskers of the tori, if any, suitable representations are given by Fourier-Taylor series. Normal forms around invariant tori are also computed using Fourier-Taylor series [31]. Refined Newton methods for both invariant tori and whiskers in quasiperiodic systems are discussed in [29, 28], and actual implementations of the algorithms including AD tools are described in [30]. For KAM and lower dimensional tori, there is some current work to implement the algorithms, arising from the rigorous results described in [12, 14, 25]. See [25, 33]. It would be very useful to have an unified view of all these methodologies, and producing efficient software packages to manage with them, comparing different techniques (e.g. AD versus FFT) from different points of view (e.g. velocity, accuracy, stability), ... Partial work have been already done in these directions, but much more effort is demanded.
- In all the problems mentioned above, one can ask also about the possibility of paralellization of the algorithms. Paralellization has been already considered with promising results in [50] for the case of computation of normal forms in Hamiltonians systems [35], and in [48] for the computation of invariant tori in quasiperiodic systems [28, 30].

- The routines of multiplication of homogeneous polynomials are the core of the programs. We have used here the naive formula to perform the product (combined with a tree data structure to handle the homogeneous polynomials), even if we are aware that there are (asymptotic) faster algorithms. See e.g. [38, 55, 11, 5, 51, 6, 40, 57, 58, 42, 46], and many other papers, for different approaches. In fact, finding fast algorithms for multiplying polynomials (and other "hard" operations, such as composition) is object of current research. It would be interesting to study the applicability of these fast methods in computational dynamical systems. We emphasize that, besides computational complexity of the algorithms, there are other issues that also important, such as accuracy and stability, and ease of actual implementations. So, comparing those methods from these different perspectives, is an important issue. (In Computer Algebra, the coefficients are in many cases "exact", e.g. integer or rational numbers, elements in some finite field, etc. so accuracy and stability is not an issue in this area).
- The methods to compute the power series expansions of the invariant manifolds are based on the parameterization method of [7]. See also [45] for an extensive study of normal forms. Of course, this part of the algorithms works for general systems, not only for simple systems. For general systems, one has to perform composition of power series. Finding methods for fast composition of power series is again an important issue in Computational Algebra (see e.g. [6, 40]). Another interesting issue is considering composition of power series with functions that are defined through recursions.
- In order to solve the functional equations, one can also use the Newton method. So, instead of solving the equations "order by order", one could double the number of correct terms at each step. But, since the multiplications of polynomials are inevitable, there is no gain in the total algorithmic complexity if one uses the classical definition. We however think that the combination of Newton method and fast polynomial multiplication could lead to optimal complexity algorithms [43].

Acknowledgements. I am very grateful to Rafael de la Llave and Carles Simó for their valuable comments in reviewing a former version of this paper. Thanks also to Àngel Jorba for some worthy remarks. I would like also to thanks to the organizers and participants of the

course "Advanced School on Specific Algebraic Manipulators", that took place in Barcelona at September 12-15, 2007, for their stimulation to finish these notes.

Special thanks to M. Gastineau and Carles Simó (again) for several gratifying days we had making benchmarks with our respective programs, that help me a lot to improve my own codes.

Thanks also to the computer facilities provided by the University of Texas at Austin, especially the 46 node AMD opteron cluster system.

References

[1] V. I. Arnol′d. *Mathematical methods of classical mechanics*, volume 60 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 199? Translated from the 1974 Russian original by K. Vogtmann and A. Weinstein, Corrected reprint of the second (1989) edition.

[2] I. Baldomá and A. Haro. One dimensional invariant manifolds of Gevrey type in real-analytic maps. *Discrete Contin. Dyn. Syst. Ser. B*, 10(2-3):295–322, 2008.

[3] Inmaculada Baldomá, Ernest Fontich, Rafael de la Llave, and Pau Martín. The parameterization method for one-dimensional invariant manifolds of higher dimensional parabolic fixed points. *Discrete Contin. Dyn. Syst.*, 17(4):835–865, 2007.

[4] Wolf-Jürgen Beyn and Winfried Kleß. Numerical Taylor expansions of invariant manifolds in large dynamical systems. *Numer. Math.*, 80(1):1–38, 1998.

[5] Marco Bodrato. Towards optimal Toom-Cook multiplication for univariate and multivariate polynomials in characteristic 2 and 0. In Claude Carlet and Berk Sunar, editors, *WAIFI 2007 proceedings*, volume 4547 of *LNCS*, pages 116–133. Springer, June 2007. URL: `http://bodrato.it/papers/#WAIFI2007`.

[6] R. P. Brent and H. T. Kung. Fast algorithms for manipulating formal power series. *J. Assoc. Comput. Mach.*, 25(4):581–595, 1978.

[7] Xavier Cabré, Ernest Fontich, and Rafael de la Llave. The parameterization method for invariant manifolds. I. Manifolds associated to non-resonant subspaces. *Indiana Univ. Math. J.*, 52(2):283–328, 2003.

[8] Xavier Cabré, Ernest Fontich, and Rafael de la Llave. The parameterization method for invariant manifolds. II. Regularity with respect to parameters. *Indiana Univ. Math. J.*, 52(2):329–360, 2003.

[9] Xavier Cabré, Ernest Fontich, and Rafael de la Llave. The parameterization method for invariant manifolds. III. Overview and applications. *J. Differential Equations*, 218(2):444–515, 2005.

[10] Enric Castellà and Àngel Jorba. On the vertical families of two-dimensional tori near the triangular points of the bicircular problem. *Celestial Mech. Dynam. Astronom.*, 76(1):35–54, 2000.

[11] Stephen A. Cook. *On the minimum computation time of functions*. PhD thesis, Department of Mathematics, Harvard University, 1966. URL: `http://cr.yp.to/bib/entries.html#1966/cook`.

[12] R. de la Llave, A. González, À. Jorba, and J. Villanueva. KAM theory without action-angle variables. *Nonlinearity*, 18(2):855–895, 2005.

[13] Wei-Hua Du and Wolf-Jürgen Beyn. The numerical approximation of center manifolds in Hamiltonian systems. *J. Math. Anal. Appl.*, 288(1):28–46, 2003.

[14] E. Fontich, R. de la Llave, and Y. Sire. Construction of invariant whiskered tori by a parameterization method. part i: Maps and flows in finite dimensions. In progress, 2008.

[15] Frederic Gabern, Wang S. Koon, Jerrold E. Marsden, and Shane D. Ross. Theory and computation of non-RRKM lifetime distributions and rates in chemical systems with three or more degrees of freedom. *Phys. D*, 211(3-4):391–406, 2005.

[16] Mickaël Gastineau. Multiplication of polynomials. In *Advanced School on Specific Algebraic Manipulators*, 2007. URL: `http://www.imub.ub.es/sam07/sam07mul.pdf`.

[17] G. Gómez, À. Jorba, J. Masdemont, and C. Simó. Study refinement of semi-analytical halo orbit theory. Technical report, European Space Agency, 1991.

[18] G. Gómez, À. Jorba, C. Simó, and J. Masdemont. *Dynamics and mission design near libration points. Vol. III*, volume 4 of *World Scientific Monograph Series in Mathematics*. World Scientific Publishing Co. Inc., River Edge, NJ, 2001. Advanced methods for collinear points.

[19] G. Gómez, À. Jorba, C. Simó, and J. Masdemont. *Dynamics and mission design near libration points. Vol. IV*, volume 5 of *World Scientific Monograph Series in Mathematics*. World Scientific Publishing Co. Inc., River Edge, NJ, 2001. Advanced methods for triangular points.

[20] G. Gómez, W. S. Koon, M. W. Lo, J. E. Marsden, J. Masdemont, and S. D. Ross. Connecting orbits and invariant manifolds in the spatial restricted three-body problem. *Nonlinearity*, 17(5):1571–1606, 2004.

[21] G. Gómez, J. Llibre, R. Martínez, and C. Simó. *Dynamics and mission design near libration points. Vol. I*, volume 2 of *World Scientific Monograph Series in Mathematics*. World Scientific Publishing Co. Inc., River Edge, NJ, 2001. Fundamentals: the case of collinear libration points, With a foreword by Walter Flury.

[22] G. Gómez and J. M. Mondelo. The dynamics around the collinear equilibrium points of the RTBP. *Phys. D*, 157(4):283–321, 2001.

[23] G. Gómez, C. Simó, J. Llibre, and R. Martínez. *Dynamics and mission design near libration points. Vol. II*, volume 3 of *World Scientific Monograph Series in Mathematics*. World Scientific Publishing Co. Inc., River Edge, NJ, 2001. Fundamentals: the case of triangular libration points.

[24] Pere Gomis and Àlex Haro. A geometric description of a macroeconomic model with a center manifold. *J. Econom. Dynam. Control*, to appear.

[25] A. González, A. Haro, and R. de la Llave. Nontwist KAM theory. In progress, 2008.

[26] Andreas Griewank. *Evaluating derivatives*, volume 19 of *Frontiers in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000. Principles and techniques of algorithmic differentiation.

[27] A. Haro. Center and center-(un)stable manifolds of elliptic-hyperbolic fixed points of 4D-symplectic maps. An example: the Froeschlé map. In *Hamiltonian systems with three or more degrees of freedom (S'Agaró, 1995)*, volume 533 of

NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci., pages 403–407. Kluwer Acad. Publ., Dordrecht, 1999.

[28] À. Haro and R. de la Llave. A parameterization method for the computation of invariant tori and their whiskers in quasi-periodic maps: numerical algorithms. *Discrete Contin. Dyn. Syst. Ser. B*, 6(6):1261–1300 (electronic), 2006.

[29] A. Haro and R. de la Llave. A parameterization method for the computation of invariant tori and their whiskers in quasi-periodic maps: rigorous results. *J. Differential Equations*, 228(2):530–579, 2006.

[30] A. Haro and R. de la Llave. A parameterization method for the computation of invariant tori and their whiskers in quasi-periodic maps: explorations and mechanisms for the breakdown of hyperbolicity. *SIAM J. Appl. Dyn. Syst.*, 6(1):142–207 (electronic), 2007.

[31] Àlex Haro. An algorithm to generate canonical transformations: application to normal forms. *Phys. D*, 167(3-4):197–217, 2002.

[32] Àlex Haro. Automatic differentiation tools in computational hamiltonian systems. In progress.

[33] Gemma Huguet. *The role of hyperbolic invariant objects: from Arnold difussion to biological clocks*. PhD thesis, Departament de Matemàtica Aplicada I, Universitat Politècnica de Catalunya, 2008. URL: http://www.ma.utexas.edu/users/ghuguet/tesi/.

[34] A. Jorba and R. de la Llave. Regularity properties of center manifolds and applications. Manuscript.

[35] Àngel Jorba. A methodology for the numerical computation of normal forms, centre manifolds and first integrals of Hamiltonian systems. *Experiment. Math.*, 8(2):155–195, 1999.

[36] Àngel Jorba and Josep Masdemont. Dynamics in the center manifold of the collinear points of the restricted three body problem. *Phys. D*, 132(1-2):189–213, 1999.

[37] Àngel Jorba and Maorong Zou. A software package for the numerical integration of ODEs by means of high-order Taylor methods. *Experiment. Math.*, 14(1):99–117, 2005.

[38] A. Karatsuba and Ofman Yu. Multiplication of many-digital numbers by automatic computers. *Dokl. Akad. Nauk SSSR*, 145:293–294, 1962. Translation in Physics-Doklady, 7 (1963), 595-596.

[39] Al Kelley. The stable, center-stable, center, center-unstable, unstable manifolds. *J. Differential Equations*, 3:546–570, 1967.

[40] Donald E. Knuth. *The art of computer programming. Vol. 2: Seminumerical algorithms*. Addison-Wesley Publishing Co., Reading, Mass.-London-Don Mills, Ont, third revised edition, 1997.

[41] B. Krauskopf, H. M. Osinga, E. J. Doedel, M. E. Henderson, J. Guckenheimer, A. Vladimirsky, M. Dellnitz, and O. Junge. A survey of methods for computing (un)stable manifolds of vector fields. *Internat. J. Bifur. Chaos Appl. Sci. Engrg.*, 15(3):763–791, 2005.

[42] G. Lecerf and E. Schost. Fast multivariate power series multiplication in characteristic zero. *SADIO Electronic Journal*, 5(1), 2003.

[43] John D. Lipson. Newton's method: a great algebraic algorithm. In *SYMSAC '76: Proceedings of the third ACM symposium on Symbolic and algebraic computation*, pages 260–270, New York, NY, USA, 1976. ACM.

[44] J.M. Mondelo, E. Barrabés, G. Gómez, and M. Ollé. Numerical parametrisations of libration point trajectories and their invariant manifolds. In *AAS/AIAA Astrodynamics Specialists Conference*. AAS, 2007. URL: http://www.ma1.upc.edu/recerca/reportsre/0607/rep060703olle.pdf.

[45] James Murdock. *Normal forms and unfoldings for local dynamical systems*. Springer Monographs in Mathematics. Springer-Verlag, New York, 2003.

[46] Richard D. Neidinger. Directions for computing truncated multivariate Taylor series. *Math. Comp.*, 74(249):321–340 (electronic), 2005.

[47] Anatoly Neishtadt, Carles Simó, and Alexei Vasiliev. Geometric and statistical properties induced by separatrix crossings in volume-preserving systems. *Nonlinearity*, 16(2):521–557, 2003.

[48] Estrella Olmedo. *On the parallel computation of invarian tori*. PhD thesis, Departament de Matemàtica Aplicada i Anàlisi, Universitat de Barcelona, 2007.

[49] Rafael Ramírez-Ros. Exponentially small separatrix splittings and almost invisible homoclinic bifurcations in some billiard tables. *Phys. D*, 210(3-4):149–179, 2005.

[50] Pau Roldán. *Analytical and numerical tools for the study of normally hyperbolic invariant manifolds in Hamiltonian systems and their associated dynamics*. PhD thesis, Departament de Matemàtica Aplicada I, ETSEIB-UPC, 2007. URL: `http://www.ma1.upc.edu/ roldan/tesi/tesi.pdf` .

[51] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing (Arch. Elektron. Rechnen)*, 7:281–292, 1971.

[52] Carles Simó. On the analytical and numerical approximation of invariant manifolds. In Daniel Benest and Claude Froeschlé, editors, *Les Méthodes Modernes de la Mecánique Céleste (Course given at Goutelas, France, 1989)*, pages 285–329, Paris, 1990. Editions Frontières.

[53] Carles Simó. Effective computations in Hamiltonian dynamics. In *Mécanique céleste*, volume 1996 of *SMF Journ. Annu.*, page 23. Soc. Math. France, Paris, 1996.

[54] Carles Simó. Global dynamics and fast indicators. In *Global analysis of dynamical systems*, pages 373–389. Inst. Phys., Bristol, 2001.

[55] A. L. Toom. The complexity of a scheme of functional elements simulating the multiplication of integers. *Dokl. Akad. Nauk SSSR*, 150:496–498, 1963.

[56] T. Uzer, Charles Jaffé, Jesús Palacián, Patricia Yanguas, and Stephen Wiggins. The geometry of reaction dynamics. *Nonlinearity*, 15(4):957–992, 2002.

[57] Joris van der Hoeven. Relax, but don't be too lazy. *J. Symbolic Comput.*, 34(6):479–542, 2002.

[58] Joris van der Hoeven. The truncated Fourier transform and applications. In *ISSAC 2004*, pages 290–296. ACM, New York, 2004.

Departament de Matemàtica Aplicada i Anàlisi.
Facultat de Matemàtiques, Universitat de Barcelona.
Gran Via de les Corts Catalanes 585, 08007 Barcelona (Spain).
E-mail address, A. Haro: `alex@maia.ub.es`