A Methodology for the Numerical Computation of Normal Forms, Centre Manifolds and First Integrals of Hamiltonian Systems

Àngel Jorba

CONTENTS

Keywords: normal forms, centre manifolds, first integrals, algebraic manipulators, invariant tori.

This research received financial support from Spanish grant DGICYT PB94–0215, EC grant ERBCHRXCT940460, and Catalan grant CIRIT 1996SGR–00105. It was also supported in part by the Institute for Mathematics and its Applications (IMA), with funds provided by the National Science Foundation.

This paper deals with the effective computation of normal forms, centre manifolds and first integrals in Hamiltonian mechanics. These calculations are very useful since they allow us, among other things, to give explicit estimates on the diffusion time and to compute invariant tori. The approach presented here is based on the algebraic manipulation of formal series with numerical coefficients for them. This, together with a very efficient software implementation, allows big savings in memory and execution time in comparison with the use of commercial algebraic manipulators. The algorithms are discussed together with their C/C++ implementations, and they are applied to some concrete examples from celestial mechanics.

1. INTRODUCTION

The importance of invariant objects in understanding the phase space of a dynamical system has been well-known since Poincaré. Invariant objects are interesting not only in themselves but also because they organize the nearby flow. Despite their importance, there are not many numerical methods to compute such objects. The aim of this paper is to explain some techniques that can help perform some of these computations, in the particular case in which the system is Hamiltonian. As we will see, many topics can be extended to general (analytic) systems and also to discrete dynamical systems. Among the several possible approaches, we have chosen methods based on the computation of (truncated) normal forms and (approximate) first integrals. Truncated normal forms are very useful since they can provide, under suitable hypotheses, integrable approximations to the dynamics. Integrability allows one to give explicitly all the invariant objects (for example, tori) in phase space. If the normal form approximates the true dynamics, the invariant objects of the initial system are also approximated accordingly; for examples see [Simó et al. 1995; Jorba and Villanueva 1998; 1999]. Approximate first integrals are quantities that are almost preserved by the flow of the system. This means that their surface levels are almost invariant by the flow. This property can be used to obtain information about some aspects of the dynamics. For instance, if one is able to estimate the corresponding remainders, it is not difficult to bound the diffusion velocity around elliptic fixed points. For a numerical example, see [Celletti and Giorgilli 1991].

One of the main problems faced when considering such computations is how to store the object in the computer. The easiest case is the computation of a single trajectory, which can be stored as a sequence of points in phase space. When the invariant object has higher dimension, it can be very difficult or impossible to store it by simply storing a net of points. The approach taken here is to use some kind of series expansion (such as power or Fourier expansions, or a combination of both) to represent the object. The advantage is that in many cases only “a few” terms of these series are needed to get a good accuracy and that they can be handled very easily. As disadvantages we note that sometimes the series have convergence problems, making it impossible to represent the object in this way. Due to the particularities of the problems considered here we will only focus on the use of power expansions. You can find examples with trigonometric expansions in [Jorba and Masdemont 1998; Gómez et al. 1997]. For a general discussion, see [Simó 1990].

Sometimes, when only a qualitative description of the dynamics is needed, it is enough to use a low-order computation (this is the typical situation encountered, for instance, in the analysis of a bifurcation). This is not the case considered here. The methodology presented in this paper is geared toward high-order computations, with a high degree of accuracy, ready for use in many practical applications. This necessity usually comes from the applications of the dynamical systems theory to real problems, like the design and analysis of trajectories for some spacecrafts; see [Gómez et al. 1985; 1987; Díez et al. 1991; Gómez et al. 1991a, 1991c; 1991b; 1993a; 1993b; 1993c; Simó 1998]. Even in more academic problems, one many need to perform very accurate computations. See, for example, [Simó 1994], where the computation (by means of formal expansions) of exponentially small quantities is considered.

Hence, the first point addressed is how to build an efficient algebraic manipulator (in an efficient language such as C or C++) in order to manipulate these expansions fast, using as little memory as possible. Then, as an application, we use these routines to study some aspects of the restricted three body problem (RTBP). More concretely, we show how to use these techniques to describe the dynamics near the five equilibrium points of the RTBP. We also discuss related topics such as error analysis (including the use of interval arithmetic), efficiency (from the points of view of memory and speed) and some possible extensions to these routines, such as more variables, time dependence, etc.

In the work described here we have made extensive use of the particularities of Hamiltonian systems, so that many of the algorithms cannot be used outside of this environment. However, our methodology for building algebraic manipulators is very general and can be applied in a lot of different contexts.

In order to simplify the exposition, we restrict ourselves to analytic and autonomous Hamiltonian systems with three degrees of freedom, having a fixed point at the origin. In Section 7 we will discuss possible extensions to more general contexts. The discussion assumes some knowledge of Hamiltonian mechanics, but for self-containedness we have included in Appendix A a summary of the main concepts and properties of Hamiltonian systems.

1.1. Examples

In this section we summarize a few problems where classical numerical methods, such as numerical integration of single trajectories, are not enough to give a good answer. They will be used as examples throughout the paper, and they were a primary motivation for our work. However, there are many other applications of our tools (both practical and theoretical) beyond the ones presented here. We have selected a few simple ones in order to have concrete problems to work with and be able to give concrete results. We hope that the interested reader will be able to apply these ideas to similar problems in other fields.

1.1.1. Dynamics near an elliptic equilibrium point. Assume that we are interested in the dynamics near an elliptic equilibrium point (which, for simplicity, we will locate at the origin) of a Hamiltonian system with three degrees of freedom. Since the phase space has dimension six, it is very difficult to get a picture of the dynamics using numerical integration of single trajectories.

Assume we are able to rewrite the initial Hamiltonian H as

$$H = H_0 + H_1, \tag{1–1}$$

where H_0 is an integrable Hamiltonian (so, in this case, the phase space is completely foliated by invariant tori) and H_1 is a nonintegrable one. Then, if H_1 is small enough near the point, the trajectories corresponding to H_0 are close to the trajectories of H, at least for moderate time spans. Hence, from the integrable character of H_0 it is not difficult to obtain approximations for the invariant tori of H. The effect that H_1 has on the solutions of H_0 is discussed in Appendix A. Essentially, near the origin, most of the tori of H_0 are not destroyed by H_1 but only slightly deformed. However, we cannot exclude the possibility that an orbit originating in any vicinity of the origin may escape, thus making the origin unstable. This phenomenon has been named Arnol'd diffusion, because a mechanism for it was proposed by Arnol'd [1964].

Assume that we are also interested in estimates of the diffusion time near the origin. The computational effort needed to do this by single numerical integration is too big for it to be a feasible option: the large number of trajectories one has to consider plus the huge interval of integration for each one (which also introduces the problem of accumulation of rounding errors) makes this calculation impossible for present computers. An alternative procedure is the following: Suppose we are able to rewrite the initial Hamiltonian H as in (1–1). Since H_0 is integrable, the diffusion present in H must come from H_1. Hence, one can easily derive bounds for the diffusion time in terms of the size of H_1. Of course, in order to produce realistic diffusion times one needs to have H_1 as small as possible. A standard way of producing the splitting (1–1) is by means of a normal form calculation: H_0 is the normal form and H_1 the corresponding remainder. See [Giorgilli et al. 1989; Simó 1989; Jorba and Simó 1994].

There are alternative ways of estimating the diffusion time near elliptic equilibrium points. For instance, one can construct approximate first integrals near the point and estimate the "drift" of these integrals. Of course, although one can use as many first integrals as degrees of freedom, it is enough to use a single positive-definite integral (near the point, its level surfaces split the phase space into two connected components so they act as a barrier to the diffusion).

Although from the theoretical point of view both approaches are equivalent — the first integrals we compute are in fact the action variables of the normal form — from the computational point of view they behave differently. We will see this in detail later.

1.1.2. Dynamics in a centre manifold. Consider a Hamiltonian system with three degrees of freedom with an equilibrium point at the origin, and assume that the linear flow around this point is of the type centre $\times$ centre $\times$ saddle.

We are interested in finding a description of the dynamics in a neighbourhood (as big as possible) of the origin. One possibility is called reduction to the centre manifold; it consists in performing changes of variables in order to uncouple (up to some finite order) the hyperbolic behaviour from the centre manifold behavior. (One can regard this as a partial normal form.) The restriction of the Hamiltonian to this approximate centre manifold will be a Hamiltonian system with two degrees of freedom. So, selecting an energy level $H = h$ and taking a suitable Poincaré section we can produce a collection of two-dimensional plots that can give a good description of the dynamics. As far as we know, this was first done in [Gómez et al. 1991c]; see also [Jorba and Masdemont 1999].

1.2. Methodology

Here we will present the methodology we use to deal with those computations, based on the use of algebraic manipulators. There are several possible schemes, depending on the kind of calculation we are interested in. For instance, if the procedure only needs to substitute trigonometric series in the nonlinear terms of the equations — as in the Lindstedt–Poincaré method: see [Gómez et al. 1991c; Gómez et al. 1997; Jorba and Masdemont 1999] — one of the best choices is to look for a recurrent expression of those nonlinear terms; the substitution is done simply by inserting the series into the recurrence. In this paper, we will apply schemes that work with the power expansion of the Hamiltonian. (When the system is not Hamiltonian, one must work with the differential equations — or with the equations of the map if the system is discrete — but, of course, this increases the computational effort.) So a general scheme for the problems considered here is the following:

1. Power expansion of the Hamiltonian around the origin.
2. Complexification of the Hamiltonian. This is not a necessary step but, as we will see, it allows one to simplify further computations.
3. Changes of variables (usually by means of Poisson brackets), up to some finite order.
4. Realification of the final Hamiltonian. Again, this is not a necessary step. It is done only to reduce the size of the resulting series.
5. Computation of the change of variables that goes from the initial Hamiltonian to the final one.

So, one needs computer routines for all these steps. A natural way of handling the power expansions is as a sequence of homogeneous polynomials:

$$H = \sum_{k\geq 2} H_k,$$

where H_k is an homogeneous polynomial of degree k. So one of the most important problems will be to deal with homogeneous polynomials of several variables. As we will see, the bottleneck (with respect to speed) of the methods exposed here is the handling of homogeneous polynomials.

1.3. Earlier Software

There are several computer packages that, in principle, are able to deal with the computations mentioned here. Among the commercial software the best-known packages are perhaps Maple and Mathematica. Although they have the advantage of being very general packages, dealing with much more problems than the ones discussed here, they are not very efficient in terms of either time and memory, and in any case one needs to write the high-level operations such as Poisson bracket. (In this direction, [Rand and Armbruster 1987] discusses the solution of typical problems in dynamical systems by using Macsyma, and [Raines and Uzer 1992] contains computations of normal forms using Mathematica.) So if one is interested in low-order computations (sometimes this is enough in academic problems) they can be considered as a valid option. But if one wants to reach high orders, general-purpose packages are not competitive at all. In this case one has to go to software tailored to the particularities of the problem. This is the line we have followed in this work. In fact, it is possible to write even faster routines (see Section 5) than we did, but then the code is, in our opinion, more obscure. In some cases, especially taking into account the development and debugging time, the gain in speed is not enough to justify the loss of clarity. Anyway, we hope the reader interested in this point will not have problems in modifying the software.

There are some other packages similar to this one in the literature. In fact, there is a long tradition of building algebraic manipulators in celestial mechanics. We cite [Giorgilli 1979; Broucke and Garthwaite 1969; Broucke 1989; Ricklefs et al. 1983; Meyer and Schmidt 1986; Brumberg et al. 1989; Laskar 1990]; see also references therein. These works are directed toward concrete problems of celestial mechanics and are very efficient when dealing with them. We also refer to [Henrard 1989] for a survey of those earlier works.

1.4. Programming Considerations

Our programming language is ANSI C, except when we have to deal with complex numbers, in which case we use C++ due to its ability of overloading arithmetic operators, that is, assigning different behaviors to the operator depending of the type of the operands. This technique also allows one to use more sophisticated types as coefficients, such as multiple precision, intervals, Fourier series, etc.

It is not necessary to know C or C++ to read this paper, but in order to understand the details of the implementation of the algorithms it is necessary to look at the source code. Information about C can be found in [Kernighan and Ritchie 1988], and about C++ in [Stroustrup 1992].

If one does not wish to use C++, it is straightforward, though tedious, to rewrite these operations in ANSI C. Another interesting possibility is to use the SCC precompiler [Schelter 1991]. This is a preprocessor that allows one to define new operations, and overload with them the standard arithmetic operators as well as the corresponding input and output functions (in a similar way as C++ does). SCC translates this code into C, to be processed by a standard C compiler.

It is not difficult to use these algorithms in other languages. In fact, the first version of these routines was written in Fortran 77. The main advantage of C for these problems, in our opinion, is the dynamic allocation of memory and the use of structures. In Fortran 77 one has to set some parameters before compiling in order to declare big enough arrays, to have room for the expansions as well as working space.

This paper is structured as follows. Sections 2 and 3 give the details on the algebraic manipulators. Section 4 is devoted to some applications of this software to concrete problems. Section 5 discuss the efficiency of these methods as well as some improvements to them. Section 6 contains some remarks about the propagation of the rounding errors. Finally, Section 7 points out some extensions to these methods to be used in more complex problems (for instance, when the Hamiltonian depends on time in a periodic or quasiperiodic way). We have added two Appendices in order to make the paper self-contained: Appendix A contains a short description of the properties of Hamiltonian systems used here and Appendix B contains a basic description of the restricted three body problem, as well as some properties that are used in the applications. We have included these appendices because we are not aware of similar summaries in the literature.

2. BASIC TOOLS

In this section we describe the basic algorithms and routines used to handle homogeneous polynomials. This is the most important part of the package.

To simplify the notation, we make the convention that the set $\mathbb{N}$ of natural numbers contains 0.

2.1. Storing and Retrieving Monomials

Assume that we want to store an homogeneous polynomial P_n of degree n, with 6 variables $x_0, \dots, x_5$:

$$P_n = \sum_{\substack{k\in\mathbb{N}^6 \\ |k|=n}} p_k x^k,$$

where we use the notation $x^k \equiv x_0^{k_0} \dots x_5^{k_5}$ and $|k| = k_0 + \dots + k_5$. For the moment we assume that all the coefficients p_k are different from zero. Define $\psi_6(n) = \#\{k \in \mathbb{N}^6 \text{ such that } |k| = n\}$ (that is, $\psi_6(n)$ denotes the number of monomials of P_n).

To store the polynomial we use an array of $\psi_6(n)$ components, each appropriate for the storage of one coefficient of the polynomial. We use the position (index) of a coefficient inside the array to know the monomial it corresponds to. To this end we construct a function, called `llex6`, which, given a position in the array (that is, an integer between 0 and $\psi_6(n)-1$), returns the multiindex that corresponds to this coefficient. Of course we need the inverse function, called `exll6`, to know where to store a given monomial.

Before going into the details of these functions, we stress that, from the point of view of efficiency, they are of crucial importance: if they are efficient, the package will be efficient. This will be discussed in Section 5.2.

In order to have a fast implementation, we use an integer array (we assume here that every integer is four bytes long) to store some information to be used by function `llex6`. This array has $\psi_6(n)$ components and each one contains in encoded form the multiindex of the corresponding coefficient. We use this array in the obvious way: each time we need to know the exponent of the monomial whose coefficient is stored in position j of the homogeneous polynomial, we get it from the component j of this array.

The way of encoding the multiindex k is the following: since we know the degree we are working with, one of the exponents (say k_0) is redundant, so we only need to store $k_1, \dots, k_5$. This has to be stored inside a 32-bit number, so we can use 6 bits for each index, leaving 2 unused. This introduces the restriction $k_j < 64$. Since we want to handle homogeneous polynomials, the maximum degree allowed is 63, more than enough for the applications considered here.

2.2. The Routines

Here we discuss in some detail the basic routines of the manipulator, since these routines are of first importance from the viewpoint of efficiency. Their source code is stored in the file `mp6.c`. (See the section on Electronic Availability at the end of the paper for information on how to obtain the software.) The number 6 in the file name and the routine names refers to the fact that we are working with six variables in our running example of a Hamiltonian system with three degrees of freedom.

For portability reasons, in the heading of several files we redefine the standard type `int` as `integer`. This is because the first version of these routines was developed on an old 286 machine, where `int`s were 2 bytes long, and it was run on a HP workstation, where `int`s were 4 bytes long. We can allow for different kinds of integers, simply by redefinig the type `integer`. Of course, this is not relevant for computers at present (1997).

Headings of the file `mp6.c`. Here we have placed the declarations of three variables that must be accessible by all the routines in this file. They are named `nor`, `clmo` and `psi`, and are initialized by routine `imp6`. Their meaning is explained in the next sections.

The routine `imp6`. This routine has to be called before using any other routine in the package, because it allocates and initializes some internal arrays to store the encoded multiindices. The only parameter to this routine is an integer, `nr`, which contains the maximum degree we want to use. This value is stored in the variable `nor`.

Before continuing, we define a function

$$\psi_i(n) = \#\{k \in \mathbb{N}^i \text{ such that } |k| = n\},$$

which can be easily evaluated by means of the recurrence

$$\psi_i(n) = \sum_{j=0}^{n} \psi_{i-1}(j) = \binom{n+i-1}{i-1}. \tag{2–1}$$

The routine `imp6` starts by checking that it is being called with a suitable degree and that the `integer` type of the machine (or compiler) is long enough.

The first step is to allocate space to store the values of the function $\psi_i(j)$. At this moment we only need to know ψ_6, but we also compute $\psi_2, \dots, \psi_5$, since they will be needed later. To this end we allocate a rectangular matrix `psi` with the first index ranging from 2 to 6 and the second one from 0 to `nor`. Then, the values $\psi_i(j)$ are computed (using the recurrence given in (2–1)) and stored in position (i, j) of the matrix `psi`.

The next step is to allocate space for the table `clmo`. The first dimension of this table ranges from 0 to `nor`, and it refers to the degree of the homogeneous polynomials. If the first index is i, the second index ranges from 0 to $\psi_6(i) - 1 \equiv$ `psi[6][i]` $- 1$. The position (i, j) of this array is the encoded version of the multiindex of the monomial number j of a polynomial of degree i. Once this table has been allocated, we fill it with information about the multiindices, as we now explain.

We define an order inside the set of multiindices of a given degree: Let k be a multiindex of degree n and define $\overline{k}$ as the integer expressed as $k_5k_4k_3k_2k_1k_0$ in base $n+1$. The order is given by

$$k^{(1)} < k^{(2)} \Longleftrightarrow \overline{k^{(1)}} < \overline{k^{(2)}}.$$

This is usually called reverse lexicographic order. Now, for a given degree `i`, we compute all the multiindices according to this order and we store them in the table `clmo`: the first multiindex for degree `i` is `(i,0,0,0,0,0)`, and all the others are generated by routine `prxk6` (see below). We store the components of each multiindex in the corresponding place of `clmo`, using 6 bits for each component: this means that the coded version of the multiindex is

$$k_1 + k_2 \times 2^6 + k_3 \times 2^{12} + k_4 \times 2^{18} + k_5 \times 2^{24}. \tag{2–2}$$

(We don't need to code k_0 because, since we know the degree, it is redundant.) This is the value we will store in `clmo[i][j]`, where `j` stands for the position of the multiindex (and the monomial) in this order.

Finally, the routine returns the amount of memory (in kbytes) used by these tables. It is up to the calling routine whether to use this value.

The routine `amp6`. This routine frees the memory allocated by `imp6`. Of course, once it has been called the manipulator cannot be used until a new call to `imp6` has been made.

The routine `llex6`. Given a location in the array of coefficients, `lloc`, and a degree, `no`, `llex6` computes the corresponding multiindex. The way it works is very straightforward because the multiindex is contained (encoded) in `clmo[no][lloc]`, and to decode it we only need to invert (2–2) using the modulus function. An improvement to this routine consists in directly extracting the corresponding bits from `clmo[no][lloc]`.

The routine `exll6`. Given a multiindex `k` of degree `no` (this is redundant information but it is very useful to prevent calls to these routines with wrong arguments), `exll6` returns the corresponding place. So, this is the inverse of `llex6`. The implementation of this routine can be done in many ways; here is how we have done it. Denote by $k = (k_0, \dots, k_5)$ the multiindex and let n be $k_0 + \cdots + k_5$. Define $k^{(5)}$ as $(k_0, \dots, k_4)$ and let $n_5 = n - k_5$ be the degree of $k^{(5)}$. Then, if we are able to compute

1. the number of multiindices $(l_0, \dots, l_5)$ of 6 variables with degree n such that $0 \le l_5 < k_5$,
2. the place it corresponds to $k^{(5)}$ among the multiindices of 5 variables of degree n_5,

the sum of these two quantitites is the place we are looking for. The first of these numbers is $\psi_5(n_5 + 1) + \cdots + \psi_5(n)$, and can be easily obtained from the table `psi`. The second one is the same problem we want to solve, but with one dimension less, so we can apply again the same procedure until we reach dimension 2 (this corresponds to polynomials in two variables), where the solution becomes obvious. An improvement to this routine is to use auxiliar tables to reduce these integer computations.

The routine `ntph6`. This routine returns the number of monomials of a given degree (this information is contained in the array `psi`).

The routine `prxk6`. It is used to produce all the multiindices of a given order, according to the order we are using. For more details, we refer the reader to the source code.

2.3. Taking Advantage of Symmetries

It is quite common in physical examples to have some kind of symmetry in the Hamiltonian. For instance, in the examples used in this paper we have a symmetry with respect to the variable z (see (4–1) and Appendix B). This implies that not all the possible monomials of the power expansion of the Hamiltonian are really present. In the examples used here, if i is the exponent of z and j the exponent of p_z, the only monomials that appear in the expansion are the ones in which $i+j$ is even. Hence, taking this into account it is possible to reduce the amount of memory used and the computing time by a factor of approximately two.

In order to exploit the symmetry we have developped special versions of the routines of Section 2.2. The source code is stored in the files `mp6s.c` and `mp6p.c`.

File `mp6s.c` contains counterparts to the routines in `mp6.c` (with an "`s`" at the end of the name, to allow their use in the same program if necessary); they assume that the only monomials present are those for which $k_4 + k_5$ is even. Since they work in a very similar way as the routines in `mp6.c`, we only mention the main differences:

`imp6s`. The function $\psi_6(n)$ is no longer valid for computing the number of monomials, because of the symmetry. The number of monomials for a given degree n is now given by

$$\sum_{j=0}^{\lfloor n/2 \rfloor} (2j+1)\, \psi_4(n-2j),$$

where $\lfloor n/2 \rfloor$ denotes the integer part of $n/2$.

`exll6s`. To have a simple formula for the position corresponding to a given index, we have changed the order used for the monomials: we first use reverse lexicografic order for the exponents (k_4, k_5) and, in second place, reverse lexicographic order for the exponents (k_0, k_1, k_2, k_3). This is usually called product reverse lexicographic order. It allows one to easily derive a closed formula for the position (see the source code).

`prxk6s`. This routine is changed in order to produce the exponents in the product reverse lexicographic order defined above.

File `mp6p.c` contains the same routines as `mp6s.c`, but with a different symmetry: here it is assumed that all the monomials that are present satisfy that k_4+k_5 is odd (this kind of symmetry appears in some computations; see Section 4.4). The implementation is almost identical to that of `mp6s.c`, so we do not add further remarks.

In fact, since the examples considered in this paper have these symmetries, we do not make use of the routines in `mp6.c`. We have included them for the sake of completeness, and because they are the most natural ones to start describing how routines of this kind work.

Finally, note that if the symmetries are "too complex" to derive closed formulas for the routines `exll`, one can always perform a binary search on the array `clmo`. In this case, it is very convenient to use an order such that the integer values stored in `clmo` are sorted as integers. Although this is not as efficient as a closed formula, it can be easily applied in all the cases.

2.4. Different Number of Variables

Since the examples in this paper involve Hamiltonian systems with three degrees of freedom, the basic routines explained here handle polynomials with six variables. If one is interested in a different number of variables, it is not difficult to build the corresponding basic routines. For instance, in Section 4.1.3 we need to handle the normal form of a Hamiltonian system with three degrees of freedom, which involves only 3 variables. It is very easy to write the corresponding routines, using the same algorithms as for six variables. We have put those routines in file `mp3.c`, which is essentially a minor modification of file `mp6.c`. In a similar way we have derived the routines of `mp4s.c` and `mp4p.c`, needed during the reduction to the centre manifold (see Section 4.3).

3. HANDLING HOMOGENEOUS POLYNOMIALS

The routines of this section are contained in the files `basop6s.cc` and `basop6sp.cc`. There are several versions of some of them, in order to deal with polynomials with different symmetries. As before, we recommend that the reader look at the source code.

Let `p1` and `p2` be homogeneous polynomials of degrees `g1` and `g2` — that is, arrays containing the coefficients of the polynomials, as explained above.

3.1. Sums

Assume first we want to add two both polynomials of the same degree (the only case that arises), storing the result in an array called `p3`. If we call `nm` the number of monomials of each polynomial — a number that can be determined by the routine `ntph6`, for example — the sum is easily computed:

```
for (i=0; i<nm; i++)
    p3[i]=p1[i]+p2[i];
```

Here we assume that we have defined the operation `+` for the type of the coefficients of the polynomial: if they are `double` variables one does not need to do anything special since they are already defined in any C compiler. If they are of `complex` type — by which we mean a structure with two members of type `double` — we assume that we are working in C++ or that we are using a C extension able to overload the arithmetic operators with the `complex` operations. If the coefficients are more sophisticated types, we assume that we have the corresponding arithmetic, as well as a way to overload arithmetic operators.

Similarly, it is very easy to implement the product of a `complex` number by a polynomial, so we make no comments on that.

3.2. Products

Now consider the product of homogeneous polynomials `p1` and `p2`, not necessarily of the same degree. The algorithm is very simple and uses the routines explained in Section 2. To account for the contribution of the product of monomial i of `p1` with monomial j of `p2` we just have to compute the corresponding multiindices $k^{(i)}$ and $k^{(j)}$, ask for the position where the coefficient of the monomial $k^{(i)}+k^{(j)}$ must be stored, and add the product of the coefficients to the value found in this position. Doing this for all the possible values of i and j we obtain the desired product of polynomials. See the source code for more details.

3.3. Poisson Bracket

The Poisson bracket of two homogeneous polynomials can be implemented using the same ideas as the product. The algorithm we have used is based on the identity

$$\left\{ \sum_{k,l} p_{k,l} x^k y^l, \sum_{k',l'} q_{k',l'} x^{k'} y^{l'} \right\}$$
$$= \sum_{k,l,k',l'} p_{k,l} q_{k',l'} \left(\sum_{j=1}^{3} (k_j l'_j - k'_j l_j) \frac{x^{k+k'} y^{l+l'}}{x_j y_j} \right),$$

where, of course, k, l, k' and l' belong to $\mathbb{N}^3$. Thus, for any term of this sum, we proceed as in the product of homogeneous polynomials: we look first for the exponents of the monomials involved, then we compute the exponents of the resulting monomials and, finally we add the resulting coefficients to the position corresponding to those monomials. For more details, see the source code.

3.4. Input and Output

We have coded several routines in order to read and write power expansions and homogeneous polynomials, in both ASCII and binary format. We do not provide a complete set of routines to handle all the possible situations, but simply give the ones needed in the examples. As mentioned before, our intention is to show that they can be written very easily and we hope that the interested reader will have no problem in coding a similar routine.

There are a lot of different routines, each one for a different purpose. Although it is not difficult to write a common front-end for all of them we have not done so. The main reason is that the aim of this paper is not to give an easy-to-use library of functions but to show how to build such a library. Hence, we have avoided any construction that hides the inner working of the routines.

3.4.1. ASCII files. There are several routines to read and write homogeneous polynomials and series. The format is very easy: for each coefficient, we compute the corresponding exponents and write the exponents followed by the value of the coefficient. We use a single line for each coefficient.

There are several sets of routines for the different kind of series (`mp6s`, `mp6p`, real or complex coefficients, etc.). Some of the routines use a threshold to decide if a monomial has to be written or not (if the absolute value of the coefficient is smaller than the threshold, the monomial is not written).

The advantage of ASCII files is that they can be printed and read by an ordinary text editor. The main disadvantages are that they are big and that input and output are slow. Hence, ASCII files are only used to write the final results and to store intermediate values during the development and debugging.

3.4.2. Binary files. This format is used to store intermediate calculations or series that are only used as input for other programs (such as change of variable routines).

The routines that write homogeneous polynomials simply write (sequentially) all the coefficients in the file, without storing the exponents of the corresponding monomials. The reading routine will read all the coefficients in a row, without any checking (except, of course, the end of file), and they will be stored sequentially in the corresponding array. Each coefficient is then identified by its position inside the file. This is done to minimize the size of the file and to maximize processing speed.

The routines that write series simply write sequentially the homogeneous polynomials, adding a little bit of information to the file according to the kind of series stored. This extra information is put at the beginning and consists of four integer values, with the following meaning:

1. The first integer contains the number of variables of the expansion.
2. The second integer contains the kind of symmetry of the expansion. This value can be:
   - 0: No symmetry. All the monomials are present in the file.
   - 1: Only present are monomials such that the sum of the exponents of the last two variables is even.
   - 2: Only present are monomials such that the sum of the exponents of the last two variables is odd.
3. The third integer is the initial degree of the expansion, usually 1 or 2.
4. The fourth integer is the final degree of the expansion.

The reading routine checks this information and gives error messages when necessary. Observe that there is nothing indicating the kind of coefficients of the series being stored. It is up to the user to take this into account.

Of course, writing in this way assumes that the reading routine will use the same algebraic manipulator as the writing routine, since the exponent of a monomial is known from the position of the monomial inside the series. You have to take this into account if you modify these routines.

4. EXAMPLES

In this section we apply these routines to some practical computations on a concrete model. For this purpose we have selected the well-known restricted three body problem (RTBP), near one of the five equilibrium points $L_{1,\dots,5}$ of the system. For a description of this problem, including the notation, see Appendix B.

4.1. Example I: Normal Form

The Hamiltonian H of the RTBP, in suitable dimensionless units and with the origin at L_4 or L_5, takes the form

$$H = \tfrac{1}{2}(p_x^2 + p_y^2 + p_z^2) + yp_x - xp_y + \left(\tfrac{1}{2} - \mu\right)x \mp \frac{\sqrt{3}}{2}y - \frac{1-\mu}{r_{PS}} - \frac{\mu}{r_{PJ}}, \qquad (4\text{–}1)$$

where

$$r_{PS}^2 = (x - x_S)^2 + (y - y_S)^2 + z^2,$$
$$r_{PJ}^2 = (x - x_J)^2 + (y - y_J)^2 + z^2,$$

$x_S = \frac{1}{2}$, $y_S = \mp\frac{\sqrt{3}}{2}$, $x_J = -\frac{1}{2}$ and $y_J = \mp\frac{\sqrt{3}}{2}$. The "$-$" sign is for L_4 while "$+$" is for L_5. The mass ratio μ is taken below the Routh critical value, so the origin is linearly stable.

4.1.1. Complexification and power expansion. The first step is to produce a power expansion of (4–1) up to a finite order N,

$$H = \sum_{n=2}^{N} H_n,$$

where H_n denotes a homogeneous polynomial of degree n in six variables. To describe how to produce such an expansion, we focus first on the term $1/r_{PS}$ of (4–1). Naming ψ the angle between $(x_S, y_S, 0)$ and (x, y, z), and letting $\rho^2 = x^2 + y^2 + z^2$, one has

$$\frac{1}{r_{PS}} = \frac{1}{\sqrt{1 - 2\rho\cos\psi + \rho^2}} = \sum_{n=0}^{\infty} \rho^n P_n(\cos\psi),$$

where P_n is the Legendre polynomial of degree n. Define A_n as $\rho^n P_n(\cos\psi)$; note that A_n is an homogeneous polynomial of degree n. Then, from the well-known recurrence of the Legendre polynomials, one obtains

$$A_{n+1} = \frac{2n+1}{n+1}(xx_S + yy_S)A_n - \frac{n}{n+1}(x^2+y^2+z^2)A_{n-1}, \qquad (4\text{–}2)$$

starting with $A_0 = 1$ and $A_1 = xx_S + yy_S$. This recurrence can easily be implemented using a routine that multiplies homogeneous polynomials. Since it is numerically stable and not too computation-intensive, this recurrence is very suitable for practical computations. Of course, the expansion of $1/r_{PJ}$ can be done in the same way, and the remaining terms of (4–1) can be added directly to the sum of these two expansions.

Before continuing, we make a very important remark. Since the first step is to put H_2 in normal form (see Section B.1), and this is done by a linear change of variables, we can insert this change of variables directly into the recurrence (4–2), in order to produce the expansion with this first change already done. This is much better than to compose the change with the final expansion. The real normal form of H_2 is

$$H_2 = \tfrac{1}{2}\omega_1(x^2 + p_x^2) + \tfrac{1}{2}\omega_2(y^2 + p_y^2) + \tfrac{1}{2}(z^2 + p_z^2),$$

where we have kept the same notation for the variables and we have used the fact that the frequency in the vertical direction is always 1 (for all μ). In order to facilitate the computation of the generating function, it is very convenient to diagonalize H_2 (see Section A.4 for more details). This can be done by a complexifying change of variables, of the form

$$x = \frac{q_1 + \sqrt{-1}\,p_1}{\sqrt{2}}, \quad p_x = \frac{\sqrt{-1}\,q_1 + p_1}{\sqrt{2}} \qquad (4\text{–}3)$$

(similarly for the other variables). So, we compose this change with the first one to obtain a complex and symplectic linear change of variables that brings the initial H_2 into the normal form

$$H_2 = \sqrt{-1}\,\omega_1 q_1 p_1 + \sqrt{-1}\,\omega_2 q_2 p_2 + \sqrt{-1}\,q_3 p_3. \quad (4\text{–}4)$$

This is, in fact, the change inserted into the recurrence (4–2) to produce the expansion in these variables.

The routines that perform this expansion are contained in the file `exp-l5.cc`. We give a short description of them.

`ccvl5` computes the linear change of variables that puts the initial H_2 into the final normal form (4–4). This change is derived in Section B.1.

`exp_l5` is the main routine for the expansion of the Hamiltonian. It calls `exrec` and `reste`.

`exrec` performs one of the recurrences (4–2). It is called twice by `exp_l5` (first to expand $1/r_{PS}$ and then $1/r_{PJ}$).

`reste` computes the terms in (4–1) that are neither $1/r_{PS}$ nor $1/r_{PJ}$.

4.1.2. The normal form. The next step is the computation of the normal form. We use Lie series, since they are very suitable to perform explicit computations. More details on this method are contained in Sections A.3 and A.4; here we will only focus on the implementation. The main properties of the Poisson bracket used here are that it is bilinear and that, if P_r and Q_s are homogeneous polynomials of degrees r and s respectively, then $\{P_r, Q_s\}$ is an homogeneous polynomial of degree $r+s-2$.

The computation is done in several steps, one for each degree. We explain the first of these steps. We want to compute a generating function G_3 (an homogeneous polynomial of degree 3) such that the transformed Hamiltonian

$$H' = H + \{H, G_3\} + \tfrac{1}{2!}\{\{H, G_3\}, G_3\} + \tfrac{1}{3!}\{\{\{H, G_3\}, G_3\}, G_3\} + \cdots \quad (4\text{–}5)$$

has no terms of degree 3. Using the notation $H = H_2 + H_3 + H_4 + \cdots$ one obtains that the terms of degree 3 of the transformed Hamiltonian H' are

$$H_3' = H_3 + \{H_2, G_3\}.$$

Hence, we demand that $H_3' = 0$. This equation is easily solved, because H_2 is of the form (4–4): Denote by k^q the three indices of k that correspond to the variable q and by k^p the ones corresponding to p. The expressions of H_3 and G_3 can be written as

$$H_3 = \sum_{|k|=3} h_3^k q^{k^q} p^{k^p}, \quad G_3 = \sum_{|k|=3} g_3^k q^{k^q} p^{k^p}.$$

Hence, assuming that the frequencies $\omega = (\omega_1, \omega_2, 1)$ of H_2 are rationally independent, it is not difficult to obtain the coefficients g_3^k of G_3:

$$g_3^k = \frac{-h_3^k}{\sqrt{-1}\,\langle k^p - k^q, \omega\rangle}.$$

Since in this case $|k|$ is odd, the denominator is never zero. When $|k|$ is even one must consider the case $k^p = k^q$ (note that, since the components of ω are rationally independent, this is the only possibility of getting a zero divisor). This implies that this monomial cannot be eliminated and then we select the corresponding g_3^k equal to zero. Of course, if one wants to perform the normal form up to degree N, it is enough to demand that $\langle k, \omega\rangle \neq 0$ when $0 < |k| < N$. If this condition is not satisfied we can still perform a resonant normal form, that is, we can eliminate all the monomials except the ones for which $\langle k, \omega\rangle = 0$ (usually called resonant monomials). Even when the frequencies are rationally independent, some of the denominators $\langle k, \omega\rangle$ can be very small, reducing drastically the domain where these transformations are valid. In this case it is also possible to leave those monomials in the normal form, in order to keep a reasonable size for the domain of convergence (note that then the normal form will not be integrable; see [Simó 1989] for a discussion of this technique).

Once the generating function has been computed, we can use (4–5) to compute the transformed Hamiltonian. We look at the implementation we have used for this formula. Assume we are working with an expansion of H up to degree N:

$$H = H_2 + H_3 + \cdots + H_{N-1} + H_N,$$

and, for instance, we want to transform it using as a generating function an homogeneous polynomial G_3 of degree 3. To save memory, the result will be stored in the same space used for H. To give the idea, we write explicitly the firsts steps of the method:

Step 1.1. $H_N \leftarrow H_N + \{H_{N-1}, G_3\}$

Step 2.1. $H_{N-1} \leftarrow H_{N-1} + \{H_{N-2}, G_3\}$

Step 2.2. $H_N \leftarrow H_N + \frac{1}{2!}\{\{H_{N-2}, G_3\}, G_3\}$
Step 3.1. $H_{N-2} \leftarrow H_{N-2} + \{H_{N-3}, G_3\}$
Step 3.2. $H_{N-1} \leftarrow H_{N-1} + \frac{1}{2!}\{\{H_{N-3}, G_3\}, G_3\}$
Step 3.3. $H_N \leftarrow H_N + \frac{1}{3!}\{\{\{H_{N-3}, G_3\}, G_3\}, G_3\}$

$\vdots$

The Poisson bracket done in step 2.1 can be reused to compute step 2.2, the one in 3.1 can be used in 3.2 and this last one in 3.3, and so on. In this way, we are minimizing the number of arithmetic operations (each Poisson bracket is done only once), we can work on the initial Hamiltonian (the parts of it that are overwritten are not needed in further steps) and the need of working space is not very big: we need working space for two homogeneous polynomial of degree N in the worst case (one is used to store the Poisson bracket done in $i.j{-}1$ to be used in $i.j$, the other one is to compute the next Poisson bracket). This has been implemented in routine `traham` (see below).

The routines for these algorithms are contained in file `nf6s.cc`. We give a short description of them:

`nf6s` is the main routine for the computation of the normal form. It assumes that the initial Hamiltonian H_2 is in diagonal form. It gets the frequencies ω from the corresponding positions in H_2 and, for each degree, it computes the generating function of the change of variables (see `cage` below) and transforms the Hamiltonian (see `traham` below). The generating function is written to a binary file, degree by degree. Since this is not considered a series but a sequence of different generating functions, no heading is added to the file (this heading was explained in Section 3.4.2).

`cage` computes the generating function corresponding to a given degree. One of the parameters is a pointer to a function that, given the exponents of the monomial, returns 1 if the monomial has to be removed from the normal form, and 0 otherwise. This is done in this way in order to make it easy to change the “killing criterion”.

`traham` transforms the Hamiltonian according to the algorithm mentioned above, using the generating function computed in `cage`. After the transformation, the routine places zeroes in the places that corresponds to killed monomials. This line of code can be commented out if the user does not this behavior; in this case, those values will not be exactly zero because of rounding errors (see Section 6 for a more detailed discussion).

Moreover, in the file `kill-nf.c` there is a function that decides if a given monomial has to be killed or not (see remarks on routine `cage` above).

4.1.3. Back to real coordinates. The final step is to realify the transformed Hamiltonian. The case of non-integrable normal forms can be done using the considerations in Section 4.4; see also Section 4.3.

We start by using the inverse of the complexifying change (4–3),

$$q_j = \frac{x_j - \sqrt{-1}\, y_j}{\sqrt{2}}, \quad p_j = \frac{-\sqrt{-1}\, x_j + y_j}{\sqrt{2}}, \tag{4–6}$$

where $j = 1, 2, 3$ and q_1, q_2, q_3, p_1, p_2, p_3 are new names for x, y, z, p_x, p_y, p_z, respectively. In order to put the Hamiltonian in the easiest possible form, we compose this change with

$$x_j = \sqrt{2I_j}\cos\varphi_j, \quad y_j = -\sqrt{2I_j}\sin\varphi_j,$$

for $j = 1, 2, 3$. This is equivalent to

$$\begin{aligned} q_j &= I^{1/2}\exp(\sqrt{-1}\,\varphi_j),\\ p_j &= -\sqrt{-1}\, I^{1/2}\exp(-\sqrt{-1}\,\varphi_j). \end{aligned} \tag{4–7}$$

Hence, since the monomials that appear in the normal form have the same exponent both for positions and momenta ($k^q = k^p$ in the notation above), the change (4–7) makes them to depend only on the actions I_j:

$$h_k q^{k^q} p^{k^p} = h_k(\sqrt{-1}\,)^{|k^q|} I^{k^q}.$$

The routines in the file `rnf6s.cc`, which we now describe briefly, apply the change (4–7) to the normal form. (Since we have to deal with polynomials in three variables, we need the routines in `mp3.c`.)

`rnf6s` applies the change (4–7) to the normal form. Assumes that the manipulator contained in `mp3.c` has been initialized by the calling routine.

`check_rlf` checks if a given multiindex corresponds to the normal form. It is used by `rnf6s` to know which terms to realify; all others are assumed to be zero.

4.1.4. Main program and results. The file `main_nf.cc` contains a main program that uses these routines. It is very short and computes the normal form, up to a given order, around the equilibrium point L_5 of the RTBP. The output of the program is contained in several files: the normal form is stored in the ASCII file `nf.res`, the generating function is stored in the binary file `nf.gen` and the linear change of variables used to diagonalize the linearized vector field around L_5 is put in the ASCII file `nf.cvl`. The degree and the mass parameter used in the actual run are stored in the ASCII file `nf.ctl`.

In Table 1 we include the first terms of the normal form for the Earth–Moon case. The last column in that table corresponds to the imaginary part of the coefficients and it should be zero. It is not zero due to the rounding errors in this process. This column is not taken into account for subsequent computations with the normal form, but we have included it to give an heuristic estimate about how round-off errors behave in this case. See also remarks in Section 6.2.

1	0	0	9.5450087346985146 e−01	0.0000000000000000 e+00
0	1	0	−2.9820811951603865 e−01	0.0000000000000000 e+00
0	0	1	1.0000000000000000 e+00	0.0000000000000000 e+00
2	0	0	1.1568661352624510 e−01	1.9950987004677088 e−15
1	1	0	−1.7127952377596927 e+00	1.6464553654140052 e−14
0	2	0	3.3855424993051031 e−01	−1.6132819906367057 e−14
1	0	1	8.9130919974620498 e−02	7.6519569570206439 e−16
0	1	1	2.2531870698905809 e−01	−1.8153505446248392 e−15
0	0	2	−2.2354591332438556 e−03	−9.4980345474466460 e−17
3	0	0	−2.9478784724938123 e−01	−8.8195876408494016 e−14
2	1	0	8.1656946590496773 e+00	−2.4411186045905709 e−11
1	2	0	−5.4586887250177915 e+02	−1.5117692624804247 e−10
0	3	0	−5.1021278394561250 e+01	−4.4683867166008548 e−11
2	0	1	−4.3799694571855952 e−01	1.4016167918231831 e−13
1	1	1	1.4116984215354037 e+01	−9.8915697135260128 e−12
0	2	1	2.0187058976961225 e+00	−1.7373839789087187 e−12
1	0	2	−5.5905039470536266 e−02	−1.9157633989231475 e−14
0	1	2	−1.7898209821803412 e−01	1.0442695912981926 e−14
0	0	3	−5.1325740689130392 e−05	−1.8243944685427148 e−15
4	0	0	1.2775512804655591 e+00	−6.7431830234291555 e−10
3	1	0	−3.5068853734061122 e+01	−5.4267568786972957 e−10
2	2	0	−5.4875008796056733 e+04	4.6093383107028067 e−08
1	3	0	3.2223469268329442 e+04	1.5779814576740623 e−07
0	4	0	3.5185007412806153 e+03	−7.6035412461354353 e−09
3	0	1	2.1759346547114546 e+00	−4.0412062928307053 e−10
2	1	1	2.0101335538551211 e+01	−1.8846523533034277 e−09
1	2	1	1.3647631576893851 e+04	1.2205347940204729 e−08
0	3	1	1.4507386615262367 e+03	−1.4038343557712580 e−09
2	0	2	2.1938211094638973 e+00	−6.7723532456943524 e−11
1	1	2	−4.9540209943972513 e+01	6.1719014476874278 e−10
0	2	2	−1.0178742459873320 e+01	3.9061093991945888 e−11
1	0	3	3.5475354854384022 e−02	8.1934049924464103 e−13
0	1	3	7.1211245121958200 e−02	1.7453335694916767 e−11
0	0	4	5.2188851777046352 e−04	3.2941657400195349 e−13

TABLE 1. Coefficients of the normal form for the Earth–Moon case ($\mu = 1.2150581623433623 \times 10^{-2}$). The first three columns contain the exponents of the actions; the fourth and fifth columns are the real and imaginary part of the coefficients. Imaginary parts must be zero, but they are not due to the rounding errors. See more comments in the text.

4.2. Example II: First Integrals

Assume we are interested in computing (approximate) first integrals of a given Hamiltonian system H, in a neigbourhood of an equilibrium point. Of course, if H is not integrable, the first integrals will not be convergent but, close enough to the equilibrium point, they will be quantities that are almost preserved by the flow. This can be used for different purposes, for instance to bound the diffusion time around an elliptic equilibrium point. See Section A.7 for more details.

We summarize the procedure to compute those integrals. Let $H = \sum_{j\geq 2} H_j$ the power expansion of H around the equilibrium point (which, for simplicity, we assume is the origin), where each H_j is an homogeneous polynomial of degree j. Denote by $F = \sum_{j\geq 2} F_j$ the expansion for the first integral we are looking for. Then, since F must satisfy $\{H, F\} = 0$, one has the recursive equation

$$\{H_2, F_n\} = -\sum_{j=3}^{n} \{H_j, F_{n-j+2}\}, \qquad (4\text{–}8)$$

which yields F_n in terms of F_{n-1}, …, F_2 and H. To simplify the discussion, assume H_2 is in complex diagonal form, that is, $H_2 = \sum_j \sqrt{-1}\,\omega_j q_j p_j$. Since $\{q_j p_j,\, q^l p^l\} = 0$, we conclude that

1. the coefficients of the monomials $q^l p^l$ of F_n cannot be determined;
2. if the coefficient of the monomial $q^l p^l$ in the right-hand side of (4–8) is not zero, this equation cannot be solved.

There are conditions under which the right-hand side of (4–8) does not contain monomials of the form $q^l p^l$. For instance, when the frequencies are nonresonant ($\langle k, \omega\rangle = 0$ if and only if $k = 0$) and the initial Hamiltonian is reversible (an even function of the momenta).

The example we are going to use is again the RTBP near $L_{4,5}$ for the Sun–Jupiter case, for which the frequencies are nonresonant. (As in the normal form case, we only need the nonresonance condition up to a finite order. Hence, this is a condition that can be checked in practical examples.) Since in this case the Hamiltonian is not reversible, we need another kind of argument to justify the solvability of (4–8). We will use without proof the fact that this equation can be solved for the RTBP case, and that it is enough to take as zero the terms of F_n that we cannot determine $(q^l p^l)$. See [Celletti and Giorgilli 1991] for a discussion of these properties.

Another point worth mentioning is that F_2 is not determined by the method, but it should be selected by the user. In [Celletti and Giorgilli 1991], since the authors want to have three first integrals $F^{(1)}$, $F^{(2)}$, $F^{(3)}$, they use $F_2^{(j)} = \sqrt{-1}\, q_j p_j$, for $j = 1, 2, 3$. We note that, if one only wants to bound the diffusion around the point, it is enough to compute a single definite-positive first integral. This can be achieved using, for instance, $F_2 = \sum_j \sqrt{-1}\, q_j p_j$. Of course, one can put different "weights" in front of each $q_j p_j$ to try to optimize the size of the region of effective stability (we recall that this region is, in general, not spherical).

4.2.1. Implementation. Most of the routines needed for this case have already been developed for the normal form computation. In fact, we only need to implement the recursion (4–8) and the realification of the (approximate) first integral.

An overview of the program is the following. First we expand the Hamiltonian around the equilibrium point using the same rutines as in the normal form case (the ones in the file `exp-l5.cc`). In this way we obtain a complexified expansion such that the second degree terms are in diagonal form. Then, we solve recurrently equation (4–8), where the initial value F_2 is provided by the user (this is done by the routines in the file `fi.cc`). Once the first integral has been computed up to the desired order, it is realified (the routines for this are in the files `irex.cc` and `re6s.cc`, and the realification process will be explained in Section 4.5) and written to the ASCII file `fi.res`. This is the only file produced by this program. The main program that controls this process is in `main-fi.cc`.

4.3. Example III: Centre Manifolds

We now consider the dynamics near one of the collinear points $L_{1,2,3}$ of the RTBP. We recall that the linearization of the vector field at these points is of the type centre × centre × saddle. In order to give an accurate description of the dynamics in a neighbourhood of $L_{1,2,3}$ one can perform the so-called reduction to the centre manifold. This process is explained with more detail in Section A.6 and the idea is the following: assume that the diagonal form of H_2 is

$$H_2 = \lambda q_1 p_1 + \sqrt{-1}\, \omega_2 q_2 p_2 + \sqrt{-1}\, \omega_3 q_3 p_3,$$

for $\lambda, \omega_2, \omega_3 \in \mathbb{R}$. Hence, the hyperbolic direction is given (to first order) by the variables (q_1, p_1). We perform canonical transformations on the Hamiltonian (in the same way it has been done in Section 4.1.2) but now, instead of cancelling all the nonresonant monomials, we only cancel monomials such that the exponent of q_1 is different from the exponent of p_1 (for a different scheme that cancels fewer monomials, see [Simó 1996]). Then, after a finite number of transformations, the Hamiltonian takes the form

$$H = H^{(0)}(q_1 p_1, q_2, p_2, q_3, p_3) + R(q_1, p_1, q_2, p_2, q_3, p_3),$$

where $H^{(0)}$ is the part of the Hamiltonian that we have arranged and R denotes the remainder. Since $H^{(0)}$ depends on the product $q_1 p_1$ we can perform the change $I_1 = q_1 p_1$ to produce

$$H = H^{(0)}(I_1, q_2, p_2, q_3, p_3) + R(I_1, \varphi_1, q_2, p_2, q_3, p_3),$$

where φ is the conjugate variable of I_1. If we drop the remainder R (it is very small near the origin) then I_1 is a first integral of the system and putting $I_1 = 0$ we are skipping the hyperbolic part of the Hamiltonian $H^{(0)}$. The resulting two degrees of freedom Hamiltonian represents the flow inside the (approximation to the) centre manifold. So, near the origin, the phase space of the original Hamiltonian must be the phase space of $H^{(0)}(0, q_2, p_2, q_3, p_3)$ times an hyperbolic direction. To visualize the phase space of $H^{(0)}$ one can fix the value of the Hamiltonian and then use a Poincaré section. Varying the value of the Hamiltonian we will obtain a collection of two-dimensional plots representing the dynamics in the phase space. This has already been done in [Gómez et al. 1991c; Jorba and Masdemont 1998; 1999].

4.3.1. Implementation. The implementation is similar to the one of the normal form, with the only difference that now we want to kill fewer monomials. Hence, for the computation of the complex normal form we have used exactly the same routines as before (the ones contained in the file `nf6s.cc`), only changing the function used to decide which monomials are killed (this function is stored in the file `kill-nf.c` for the normal form case and now is the one in the file `kill-cm.c`).

The main difference appears when we need to realify the transformed Hamiltonian. In the normal form case, realification is done by taking advantage ot the particularities of a complete normal form. Here it is a little bit more difficult. We summarize the process. First, to save memory, the (still complex) partial normal form is written in a binary file and then it is read monomial by monomial. For each monomial corresponding to the centre manifold (namely those for which the exponent of q_1 equals that of p_1) we compute the result of applying the realifying change (4–6) to this monomial; other monomials are discarded. The process is the same one used in Section 4.5 (see there for more details), but for four variables monomials. The realified monomials are added to the realified series (different complex monomials can contribute to the same realified monomial) until all the complex monomials are transformed. The routines that perform the realifying process are stored in the files `irex.cc` and `rcm6s.cc`. Finally, the centre manifold is written to an ASCII file. The main program for this computation is stored in the file `main-cm.cc`.

The output files are: `cm.res` contains (in ASCII format) the Hamiltonian reduced to its centre manifold, `cm.gen` is a binary file with the generating function used, `cm.cvl` is an ASCII file with the linear change used to put H_2 in diagonal form and `cm.ctl` contains the parameters used in the actual run.

4.4. Changes of Variables

An important part of the computations is to produce the changes of variables going from the final coordinates (normal form or centre manifold) to the initial ones. This can be used for several purposes, ranging from estimates on the diffusion time to the practical computation of invariant tori (of any dimension). We refer to [Jorba and Villanueva 1998] for examples of this.

The global change is split in two different sub-changes. The first one is the linear change that puts H_2 in diagonal form (we will refer to these coordinates as "diagonal" coordinates) plus the translation of the origin from the libration point to the centre of masses of the RTBP. The second sub-change consists of the nonlinear change that goes from the normal form (or centre manifold) coordinates to the diagonal ones. Here we will focus on this last change since the first one is explicitly given in Appendix B.

Here is the process for obtaining the nonlinear change. We start by considering the first change of variables done on the Hamiltonian by means of a generating function G_3. The corresponding change for this transformation can be obtained by applying the transformation (4–5) to a single coordinate q_i or p_i, where $1 \le i \le 3$:

$$q_i^{(3)} = q_i + \{q_i, G_3\} + \tfrac{1}{2!}\{\{q_i, G_3\}, G_3\} + \cdots, \tag{4–9}$$

$$p_i^{(3)} = p_i + \{p_i, G_3\} + \tfrac{1}{2!}\{\{p_i, G_3\}, G_3\} + \cdots \tag{4–10}$$

where $q_i^{(3)}$ and $p_i^{(3)}$ denote the series obtained in this transformation. This is done using the algorithm explained in Section 4.1.2. Expressions (4–9) and (4–10) are changes of coordinates: they relate the coordinates of the transformed Hamiltonian under G_3, namely q_i and p_i, with the initial (diagonal) coordinates $q_i^{(3)}$ and $p_i^{(3)}$. This idea can be used to produce the changes to higher orders. For instance,

$$q_i^{(4)} = q_i^{(3)} + \{q_i^{(3)}, G_4\} + \tfrac{1}{2!}\{\{q_i^{(3)}, G_4\}, G_4\} + \cdots,$$

$$p_i^{(4)} = p_i^{(3)} + \{p_i^{(3)}, G_4\} + \tfrac{1}{2!}\{\{p_i^{(3)}, G_4\}, G_4\} + \cdots,$$

is the transformation that goes from the normal form coordinates of degree 4 to the initial diagonal coordinates. Of course, this transformation is done on the expressions (4–9) and (4–10) as if they were Hamiltonians, by means of the algorithm explained in Section 4.1.2. In this way, we obtain the explicit transformation that puts the Hamiltonian in normal form up to the desired order. When doing these transformations, it is only necessary to transform up to the same degree as in the normal form.

Note that the series obtained are still in complex coordinates. They are realified using the methods that will be explained in Section 4.5.

The change corresponding to the centre manifold has some differences with the change for the normal form case. Since the centre manifold is of dimension four (the first two variables have been set to zero), the final change is given by six real expansions, each one depending on four variables (the first four expansions are of the type `mp4s` and the last two are of the type `mp4p`).

4.4.1. The inverse change. As before, we are going to focus on the nonlinear part of the change, since the linear part is easily inverted. We only provide routines for the normal form case (the inverse change for the centre manifold can be produced similarly).

This computation is based on the following fact: the change induced by the generating function G is the inverse of the change induced by the generating function $-G$. This is because the change is the time one flow of the Hamiltonian G, and to reverse the time in this flow one has to change the sign of the vector field, i.e., of the Hamiltonian G. Hence, one can use the same scheme as before but using as generating functions $-G_n$, $-G_{n-1}$, $\ldots$, $-G_4$, $-G_3$, in this order. We refer to the previous section for more comments.

As before, the obtained series are still in complex coordinates. Section 4.5 deals with the algorithms used to realify them.

4.5. Realification of Power Expansions

A common operation at the end of these computations is the realification of the complex power expansions obtained, because we are usually interested in the dynamics corresponding to real coordinates. Hence, realified expansions are much smaller (the memory needed to store them is halved) and this implies that all the computations involving them are also faster. We stress that it is not compulsory to perform such realification, because all the computations with these expansions can be done with the complexified version. The realification is only used for efficiency reasons.

We now explain the algorithm used. To simplify the discussion, assume we have to realify a 6 variables expansion, in which all the variables have been previously complexified. (It is possible to have a complex expression in which not all the variables have been complexified; see the expansion of the Hamiltonian in Section 4.3, for instance.) To start, we focus on the realification of a single monomial,

$$c_k q_1^{k_1} p_1^{k_2} q_2^{k_3} p_2^{k_4} q_3^{k_5} p_3^{k_6}. \tag{4–11}$$

In order to apply the realifying change (4–6), we make some remarks:

1. If we know the realification of the product $q_1^{k_1} p_1^{k_2}$, for any k_1 and k_2, we know the realification of all the products $q_2^{k_3} p_2^{k_4}$, $q_3^{k_5} p_3^{k_6}$ (the only difference is in the subindices of the variables).
2. If we know the realifications of the three pairs $q_j^{k_{2j-1}} p_j^{k_{2j}}$ $(j = 1, 2, 3)$, the product of these realified expansions (note that each one of them is an homogeneous polynomial with two variables) is not difficult to compute, since we are multiplying polynomials that depend on different variables.

Hence, we will apply the following scheme: first we will compute the realifications of all the powers $q_1^{k_1} p_1^{k_2}$, where the exponent (k_1, k_2) is such that $0 < k_1 + k_2 \le n$, and n denotes the degree up to which we plan to realify. The result of each realification will be stored in a table (see below). Then, for each monomial like (4–11), we will obtain from the table the realifications of the three pairs $q_1^{k_1} p_1^{k_2}$, $q_2^{k_3} p_2^{k_4}$ and $q_3^{k_5} p_3^{k_6}$ (they will be three homogeneous polynomials of degrees $k_1 + k_2$, $k_3 + k_4$ and $k_5 + k_6$, respectively). Finally, we will form the product (4–11), taking advantage of the fact that the three homogeneous polynomials depend only on two variables, and that these variables are different. We explain this in more detail.

4.5.1. The realifying table. Now we consider the problem of computing and storing expressions like $q^i p^j$, $i \in \mathbb{N}$, $j \in \mathbb{N}$, where

$$q = \frac{x - \sqrt{-1}\, y}{\sqrt{2}}, \quad p = \frac{-\sqrt{-1}\, x + y}{\sqrt{2}}. \tag{4–12}$$

We start with the storing procedure. Fix i and j, and define $m = i + j$. Then, the substitution of (4–12) into $q^i p^j$ produces an homogeneous polynomial of degree m, in the variables x and y. A natural way of naming the different coefficients of this polynomial is to use a single integer to denote the monomial we refer to: monomial number 0 will be $x^m y^0$, monomial number 1 will be $x^{m-1} y^1$, and so on. Generically, the monomial number k will be $x^{m-k} y^k$, $0 \le k \le m$. We need three indices (i, j, k)

to identify one of these coefficients (i, j refer to the monomial q^ip^j, and k refers to the position of the coefficient inside the realification of q^ip^j). Hence, we can look at all these realifications as polynomials with three variables: the coefficient number k of the realification of q^ip^j is the coefficient of the monomial (i,j,k) of a (real but not homogeneous) polynomial of degree $2m$. This implies that, to store all these realifications, it is enough to allocate space for a three variables power expansion up to degree $2n$, where n denotes the maximum degree we plan to realify. Not all the monomials of this expansion are going to be used, but

1. the amount of memory used by the whole table is not very big (see examples below),
2. in this way the access to the elements of the table is very easy (we can use the manipulator `mp3` explained before) and very fast.

It would be possible to only allocate the elements we really need, but this would decrease the speed of the program and, as has been said, the amount of memory saved is not enough (in our opinion) to justify the increase in complexity of the program.

To simplify and speed up the computation of the realifying table we also initialize a couple of auxiliar tables, one with the negative powers of $\sqrt{2}$ and another one with the binomial coefficients. With these auxiliar tables, it is not difficult to compute the different powers q^ip^j and to store them in the corresponding place of the table.

The routines that initialize the realifying process have been stored inside the file `irex.cc`. They are:

`ini_real` allocates space for the table that will contain the realifications of the different monomials q^ip^j. It also computes and stores that table. This rutine calls routine `imp3` (file `mp3.c`) to initialize the tables needed to handle power expansions with three variables.

`end_real` frees the space allocated by `ini_real`, including a call to `amp3` to free the space allocated by `imp3`.

`coef` computes the coefficient of $x^{k-j}y^j$ in q^k or p^k.

4.5.2. The main algorithm. Now it is not very difficult to realify a power series. In order to minimize the amount of RAM used, the series to be realified is first written in a (binary) file. Then, this file is read sequentially and each monomial is realified and added to the (proper place of the) resulting series.

So, the only point that needs to be discussed is the realification of a single monomial. The process is as follows. We use the same notation as in (4–11). Each couple $q_i^{k_j}p_i^{k_{j+1}}$ becomes, once realified, an homogeneous polynomial of degree k_j+k_{j+1} in two variables, x_i and y_i. The coefficients of this polynomial are stored in the suitable places of the realifying table (see Section 4.5.1). Therefore, in order to multiply these three realified polynomials, we will use three (nested) loops to "run" over the coefficients of them (these coefficients are directly obtained from the realifying table). In this way we will obtain the coefficients of the realification of (4–11) as the product of these three coefficients with the coefficent c_k. The exponent that corresponds to this final product is easily obtained and this allows one to add the coefficient to the suitable place of the resulting series.

4.5.3. The final output. Before continuing with the description of the algorithm we explain, up to now, what we have obtained. As before, to simplify the discussion we will focus on a position-momentum pair, which we denote as q_1, p_1. We denote the initial change of variables that we want to realify as

$$\begin{aligned} q_1' &= q_1 + O_2(q,p),\\ p_1' &= p_1 + O_2(q,p), \end{aligned}$$

where the primed variables are the initial ones and the unprimed variables the final ones. Of course, by $O_2(q,p)$ we denote the higher-order terms of the change, which we do not write explicitly. After the realification process we have just described, we obtain something like

$$\begin{aligned} \frac{x_1'-\sqrt{-1}\,y_1'}{\sqrt{2}} &= \frac{x_1-\sqrt{-1}\,y_1}{\sqrt{2}} + O_2(x,y),\\ \frac{-\sqrt{-1}\,x_1'+y_1'}{\sqrt{2}} &= \frac{-\sqrt{-1}\,x_1+y_1}{\sqrt{2}} + O_2(x,y). \end{aligned}$$

The next (and final) step is to isolate $x_1' = x_1'(x,y)$ and $y_1' = y_1'(x,y)$. For instance, x_1' can be isolated from the first equation by taking real parts and multiplying by $\sqrt{2}$, and y_1' can be obtained by the first equation by taking the imaginary parts times $-\sqrt{2}$. A similar process can be applied to the second equation to obtain the same expressions.

Maybe the most important conclusion we can get from this fact is that it is enough to compute only one of the expressions for the change of variables: for instance, to obtain the changes of variables for the normal form of Section 4.1 (a Hamiltonian with three degrees of freedom) we only need to compute the changes for the three positions. The changes for the three corresponding momenta are obtained from them when realifying (note that we are using that we have complexified with respect to all the variables). Of course, we have taken advantage of this property in the software.

4.5.4. A few remarks. In some cases, it is necessary to realify not all the variables, but only some of them. A typical example appears when we have been dealing with an expansion of the kind centre $\times$ saddle. The saddle variables does not need to be complexified, since they already appear in "diagonal form" (see Section 4.1.1). Hence, once the computation is finished, they are still in real form. Of course, the realifying change have to be only applied to the pairs q_i, p_i that have been complexified. The main difference appears in the change that corresponds to variables that have not been complexified. Denote by q_1, p_1 one of these pairs. After the realification (of the complexified variables), the change for q_1, p_1 looks like

$$x_1' = x_1 + O_2(x,y), \quad y_1' = y_1 + O_2(x,y).$$

We have changed q_1, p_1 by x_1, y_1 to denote that the realification has been done. The realifying changes have been applied to variables q_j, p_j, $j \neq 1$ (they only affect to $O_2(x,y)$). Hence, we have directly the change of variables (in particular, all the imaginary parts of the coefficients of this change must vanish), without need of taking real or imaginary parts. The bad news are that now we need to compute both changes (for x_1' and y_1'), since we cannot derive easily one from another.

4.6. The Linear Part of the Change

We have seen how to produce the nonlinear change for variables used to achieve the normal form but, to reach the initial coordinates we still need to apply the linear change used at the beginning to put H_2 in normal form. This change has been computed in order to diagonalize the second degree terms of the Hamiltonian, and it has been stored in a file. In principle, this transformation goes from the "diagonal" coordinates of H_2 to the usual coordiantes of the RTBP centred at the equilibrium point. If one is interested in the inverse change, it is not difficult to see that the inverse of any symplectic matrix M can be obtained as $M^{-1} = -JM^TJ$, that is very suitable for numerical purposes.

4.7. Tests of the Software

We have done some checks on the software, to be sure that there are no bugs present. The tests we have done are very similar for the three examples so we will mainly focus on the tests for the normal form computation.

To this end, we have written the program `ninf`, that produces a numerical integration of the normal form obtained. In fact, since the normal form is integrable, this program computes the gradient of the normal form for the given actions to obtain the frequencies and then it simply tabulates the solution. Then, this table is sent through the changes of variables into the synodical coordinates of the RTBP. Finally, program `rtbp` tests this table in the following way: for each point of the table, it integrates (numerically) the point to obtain a prediction for the following point of the table. Then, the program writes the differences between the two points (the one obtained from the changes of variables and the one obtained using numerical integration). Ideally, if the normal form, the changes of variables and the numerical integration were all exact (zero error), these differences must be zero. Of course, they are not zero due to the several sources of error.

We illustrate this. We have taken the initial conditions $I_1 = I_2 = I_3 = \lambda_0$, with initial phases $\varphi_1 = \varphi_2 = \varphi_3 = 0$, for $t = 0$ (call u_0 this initial condition). We have tabulated the corresponding solution at $t = 0.1$ (call u_1 this value), and we have sent both points to synodical coordinates, to obtain two points v_0 and v_1. Then, we have computed (numerically) the trajectory of the RTBP that starts at v_0, till $t = 0.1$ (call this point v_0^1), with a local error of the order of the roundoff of the arithmetic. The difference $\left\|v_1 - v_0^1\right\|_2$ is given in Table 2.

The parameter λ_0 is, essentially, the distance from the initial condition to the origin. If the software is working properly, the error $\left\|v_1 - v_0^1\right\|_2$ is due to the

λ_0	$\left\|v_1 - v_0^1\right\|_2$
0.00001	2.4828078245222093e−16
0.00002	5.1198523403369423e−15
0.00004	1.3192410121093586e−12
0.00008	3.4023375555581652e−10
0.00016	8.8211434435268124e−08
0.00032	2.3101212284736493e−05

TABLE 2. Differences between a normal form prediction and a numerical integration near L_5 in the RTBP. The local error of the numerical integration is of the order of 10^{-16} and the normal form (and the corresponding changes of variables) have been computed up to degree 16.

truncation of the power series (to degree 16, in the case corresponding to Table 2). Hence, the error should behave like $c\lambda_0^n$, where n is the last order in the normal form that we have taken into account (see below). Then, one has that the order of the error can be approximated by

$$n \approx \frac{\log(e_1/e_2)}{\log\bigl(\lambda_0^{(1)}/\lambda_0^{(2)}\bigr)}.$$

Applying this to the results in Table 2 we obtain Table 3. The first value in this table is not very accurate because the estimation of the error is not realistic for $\lambda_0^{(1)} = 0.00001$ (it is smaller than 10^{-16} and this is not detected since we are working with double precision arithmetic). The other values are more accurate and produce an exponent for λ_0 very close to 8. Note that if the order of the normal form in the (q,p) variables is 16, it is 8 in the Poincaré variables; see (4–7). Moreover, the numerical integrations are done on the differential equations (that involve the derivatives of the Hamiltonian). Thus the error for this case is not of the order of the neglected terms of the Hamiltonian but of the neglected terms of the corresponding differential equations. Hence, as λ_0 "moves" in the space of the Poincaré coordinates,

$\lambda_0^{(1)}$	$\lambda_0^{(2)}$	n
0.00001	0.00002	4.366
0.00002	0.00004	8.009
0.00004	0.00008	8.011
0.00008	0.00016	8.018
0.00016	0.00032	8.033

TABLE 3. Estimation of the order of the error.

we expect an estimated exponent of the same order as the biggest degree present in the normal form expressed in Poincaré variables.

The same procedure can be applied for the centre manifold computation and for the first integrals, to estimate the order of the error. The concrete calculations for these cases are left to the reader.

4.8. Invariant Tori

Here we note that, using the tools we have developed, it is very easy to compute invariant tori close to any of the libration points of the RTBP. For instance, let's focus on the neighbourhood of the L_5 point of the Earth–Moon RTBP.

Figure 1 is a two-dimensional torus obtained by taking, in the normal form, the actions $I_1 = I_2 = 0.0001$, and $I_3 = 0$. This corresponds to an elliptic (planar) Lyapunov torus obtained from two of the (three) linear oscillations at L_5 [Jorba and Villanueva 1997a]. Figure 2 corresponds to a two-dimensional elliptic torus obtained taking $I_1 = I_3 = 0.0001$ and $I_2 = 0$. This torus can also be seen as coming from the linear oscillations around the periodic Lyapunov family associated to the vertical oscillation at L_5 [Jorba and Villanueva 1998]. In both cases, we have plotted a dot every 0.1 units of time.

It is not difficult to compute Poincaré sections of these trajectories, to see that they are invariant curves. We left this for the interested reader, as well as the computation of more invariant tori. Finally, note that it is also possible to ask for a torus with prefixed frequencies: one has to solve a system of three nonlinear equations to find the corresponding actions. Of course, this is only possible for suitable frequencies.

5. EFFICIENCY CONSIDERATIONS

When one considers the optimality of a given calculation, there are two main things to be taken into account: the algorithm used and its implementation. Here we are not going to discuss the efficiency of the algorithm selected (although there are other possibilities, for example to use quadratic schemes instead of linear ones; see [Llave et al. 1986], for instance), and we are going to focus on their implementation.

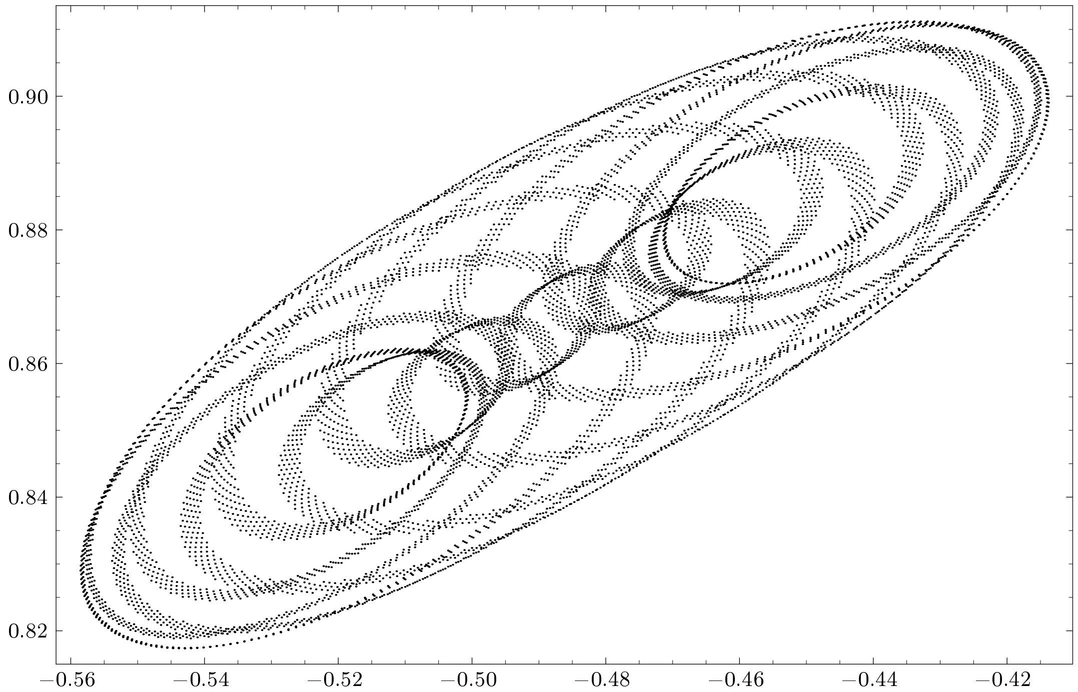

FIGURE 1. Projection on the (x, y) plane (synodical coordinates) of an elliptic two-dimensional invariant torus near L_5. The intrinsic frequencies are $\omega_1 = 0.954347344380$ and $\omega_2 = -0.298324062073$. The normal frequency is $\omega_n = 1.00003161731$. Ten thousand points are shown.

We now make a few remarks on the optimality of these routines. The implementation we have selected here (to use integer functions — sometimes called "hash functions" — to know the position corresponding to a given exponent and viceversa) allows for very easy implementations, but adds an overhead to the program (the time taken by these functions and the memory used by the integer tables). In some cases, it is possible to use specific orders for the polynomials such that the main operations can be performed directly, without the help of such functions: for instance, when dealing with polynomials of one variable, we can store the coefficient of the monomial x^j into the position number j

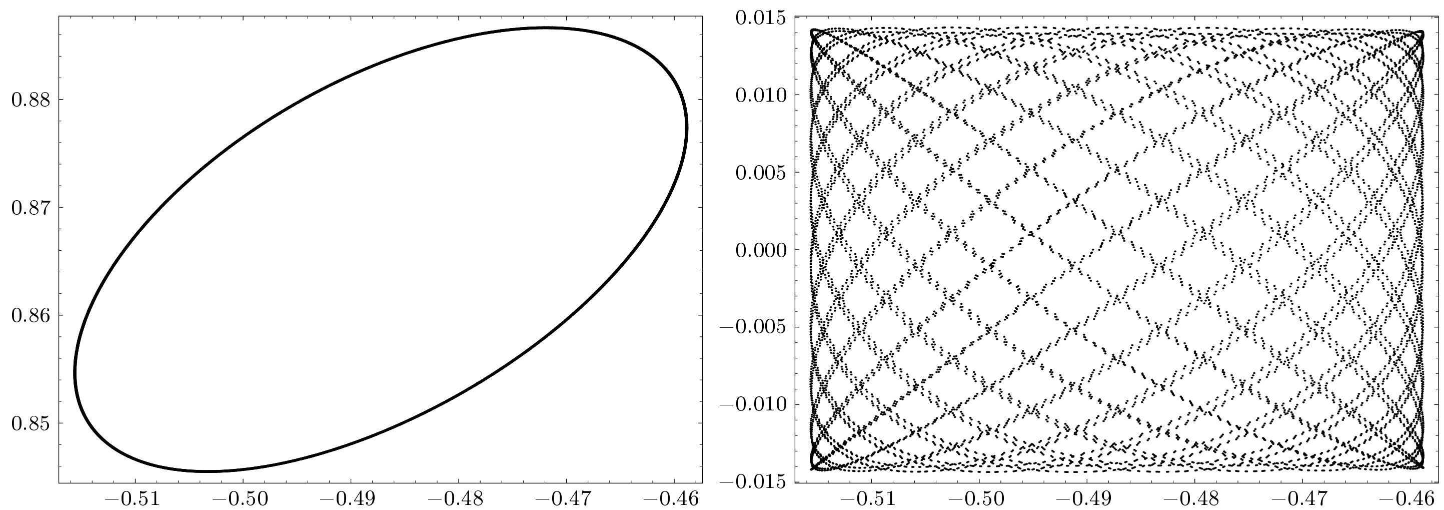

FIGURE 2. Projections on the (x, y) plane (left) and on the (x, z) plane (right) of a two-dimensional invariant torus near L_5. The intrinsic frequencies are $\omega_1 = 0.954532905738$ and $\omega_2 = 1.00000846050$. The normal frequency is $\omega_n = -0.298356646196$. Ten thousand points are shown.

of the corresponding array, so all the operations can be performed trivially (for instance, the product of two polynomials is `p[i]*q[j]` $\to$ `r[i+j]`). This is still possible in two variables but it becomes more tricky in several variables. Moreover, if the coefficients of the polynomials are of sophisticated types (such as trigonometric polynomials), the time taken by the hash functions is unnoticeable compared to the time taken by the operation involving the coefficient. So, in our opinion, the gain obtained by using tricks of this kind is not big enough to compensate for the increased complexity of the code.

In what follows, the measures of the time needed for the programs to execute have been taken from runs done on a Pentium Pro 200 MHz PC running Linux, with the GNU compiler gcc/g++ version 2.7.2.1. The amount of needed memory has been estimated directly from the size of the expansions.

5.1. Storage

We start by considering the efficiency from the point of view of the amount of memory used by the programs. Since the memory is allocated and freed dynamically, we will focus on the "worst moment" of the program, that is, when the maximum amount of memory is needed.

Table 4 displays the number of monomials for some of the expansions used here. From this table, and knowing the number of series we use in each program, it is not difficult to have an idea of the order of the amount of memory needed.

n	`mp4s`		`mp6s`		`mp6p`	
0	1	1	1	1	0	0
1	2	3	4	5	2	2
2	6	9	13	18	8	10
3	10	19	32	50	24	34
4	19	38	70	120	56	90
5	28	66	136	256	116	206
6	44	110	246	502	216	422
7	60	170	416	918	376	798
8	85	255	671	1589	616	1414
9	110	365	1036	2625	966	2380
10	146	511	1547	4172	1456	3836
11	182	693	2240	6412	2128	5964
12	231	924	3164	9576	3024	8988
13	280	1204	4368	13944	4200	13188
14	344	1548	5916	19860	5712	18900
15	408	1956	7872	27732	7632	26532
16	489	2445	10317	38049	10032	36564
17	570	3015	13332	51381	13002	49566
18	670	3685	17017	68398	16632	66198
19	770	4455	21472	89870	21032	87230
20	891	5346	26818	116688	26312	113542
21	1012	6358	33176	149864	32604	146146
22	1156	7514	40690	190554	40040	186186
23	1300	8814	49504	240058	48776	234962
24	1469	10283	59787	299845	58968	293930
25	1638	11921	71708	371553	70798	364728
26	1834	13755	85463	457016	84448	449176
27	2030	15785	101248	558264	100128	549304
28	2255	18040	119288	677552	118048	667352
29	2480	20520	139808	817360	138448	805800
30	2736	23256	163064	980424	161568	967368
31	2992	26248	189312	1169736	187680	1155048
32	3281	29529	218841	1388577	217056	1372104

TABLE 4. Number of monomials for expansions of the kind `mp4s`, `mp6s` and `mp6p`. Here n denotes the degree. In each of the remaining three sections, the first column shows the number $\Phi(n)$ of monomials in a polynomial of degree n, and the second shows the number $\sum_{j=0}^{n} \Phi(j)$ of monomials in a expansion up to degree n.

5.1.1. Normal forms. In a normal form computation as the one performed here, we use the following expansions (we denote by n the maximum degree wanted):

1. A power expansion up to degree n of the type `mp6s` (for the Hamiltonian).
2. An auxiliar power expansion (to be used only during the computation of the power expansion of the Hamiltonian) of the same degree as the Hamiltonian.
3. Three polynomials of degree n, of the type `mp6s`, to be used as a working space during the normal form computations.

The expansion in item 2 and the three polynomials in item 3 are needed in different places of the program, so we only need to take the maximum of them.

In fact we need a little bit of memory (like the inner tables of the manipulators or the three-variables expansion for the normal form), but the series mentioned above are the most important ones.

As for the amount of hard disk memory used, we need:

1. A binary file to store the generating function. This is about the size of a power expansion of degree n, of the type `mp6s`.
2. A few extra ASCII files (to store the normal form, the control parameters, etc.) that, as they are very small, we skip them.

Of course, one can modify the program in order to write more information (you can ask for intermediate series) of less (you can skip the writing of the generating function if you are not interested in the change of variables). In such a case, you should re-estimate the amount of memory needed.

Table 5 summarizes our estimates for the amount of memory needed. We have assumed that each coefficient is a double precision complex number, that is, each one needs 16 bytes to be stored.

degree	time `nf`	time `cm`	RAM	HD
8	0.40	0.46	0.058	0.025
12	8.01	9.51	0.306	0.153
16	82.25	95.77	1.218	0.609
24	3002.14	3505.71	9.595	4.798
32	48422.61	55769.46	44.435	22.217

TABLE 5. Time (in seconds) and memory (in Mbytes) needed for the normal form (`nf`) and centre manifold (`cm`) computation. The latter computation requires as temporary disk storage about the same space as the results. Thus the actual amount of free disk space needed to run the program is about twice the HD column.

5.1.2. Centre manifolds. The only difference between a normal form and a centre manifold computation (concerning the amount of memory used) appears when realifying the Hamiltonian restricted to the centre manifold. From the program, it is seen that this only affects to the amount of hard disk needed. In Table 5 we have summarized those values. As in the normal form case, we have skipped the size of the ASCII file with the final Hamiltonian, since it is not very big. We note that this file is written after erasing the temporal file, so if it was room for this file, there is enough room for the results. However, if one wants precise estimations of the final amount of used disk, one must take into account the size of that ASCII file. The concrete runs displayed there have been done for the L_1 case of the Earth–Sun system.

5.1.3. First integrals. The calculation of a first integral is a little bit simpler than a normal form one. In fact, the program needs RAM space for the Hamiltonian and the first integral, and disk space for the results as well as a temporary (binary) file used to realify the first integral. In the actual version of the program, the output file is an ASCII file, to be able to look directly at the results using an standard text editor (like `vi` or `emacs`). In Table 6 we have included the time and memory used for several runs of the program. Note that we have been using a lower degree for the calculations. This is because the huge amount of disk space needed to store the output in ASCII format. If one is interested in running to higher orders it should be better to change the program in order to store the first integral in a binary file (this is, in fact, very easy using the routines provided here). Then, the amount of disk space is similar to the one used by the centre manifold program (see Table 5).

degree	time	RAM	HD tmp.	HD final
8	0.38	0.05	0.02	0.11
12	5.36	0.30	0.15	0.67
16	49.98	1.16	0.58	2.66
20	337.09	3.56	1.78	8.17
24	1800.57	9.15	4.58	20.99

TABLE 6. Time (in seconds) and memory (in Mbytes) needed for the calculation of a first integral. The column "HD tmp." only refers to the temporary (binary) files, while the column "HD final" only refers to the final (ASCII) file.

5.1.4. Changes of variables. We now discuss the calculations needed to obtain the expansions for the changes of variables corresponding to the normal form case. We will only focus on the direct changes, since the inverse ones need approximately the same amount of memory and time.

As before, n will denote the degree of the expansion of the transformation. During the computation of the direct change, we use one expansion up to degree n and three homogeneous polynomials of degree n. In fact, we need polynomials of the type `mp6s` for the transformation corresponding to the four first variables, and of the type `mp6p` for the last two. Since the polynomials of the type `mp6s` contain more monomials than the corresponding ones of the type mp6p, we have done the memory estimations for the type mp6s. They are summarized in Table 7.

A special case is the computation of the changes of variables corresponding to the reduction to the centre manifold. In this case, we obtain six series, each one depending on four variables (see Section 4.4), so the final amount of disk space is smaller than in the normal form case. To estimate the maximum amount of disk space needed during the execution, we note that this occurs during the realification of the last couple of variables. At this moment, we have four real series (of the type mp4s) written in the disk, and we write a temporal file with a complex series (of the type mp6p) corresponding to the last couple of variables. From these observations, it is not difficult to derive the figures shown in Table 7.

degree	time cvnf	time cvcm	RAM	HD cvnf	HD cvcm
8	1.11	1.32	0.06	0.07	0.03
12	24.24	29.36	0.29	0.43	0.17
16	272.46	330.22	1.05	1.72	0.63
20	1942.21	2340.47	3.01	5.30	1.90
24	10395.05	12644.89	7.31	13.64	4.80

TABLE 7. Time (in seconds) and memory (in Mbytes) needed to compute the changes of variables for the normal form (cvnf) and the centre manifold (cvcm).

5.2. Speed

Finally we discuss the optimality, according to the speed, of these routines. To start the discussion, we focus on the routine that multiplies homogeneous polynomials (see Section 3): for each couple of monomials that we are multiplying we need to know the exponents of them and the position to store the result. Since every product is an unavoidable operation (we recall that we are discussing the optimality of the implementation, not of the algorithm), all the overhead of this implementation is due to the routines that look for exponents and positions. In fact, if these routines use zero time, the product would be optimal, since all the time spend by the product would correspond to the unavoidable operations. This is also true for the other routines (Poisson brackets, power expansions, etc.). For this reason we say that the optimality of the package is basically given by the optimality of the routines of the files mp6s.c, mp6p.c, etc. In order to quantify this, we have done a run of the program nf using the profiling facilities of the compiler. The results are shown in Table 8. We must eliminate from this table the time used by routine mcount, which was introduced by the profiler itself. Then the time taken by routines exll6s and llex6s is a little bit less than

% time	time (s)	calls	time/call (ms) self	time/call (ms) total	name
40.15	51.96	269	193.16	347.79	papu6s
26.48	34.27				mcount
26.46	34.24	84136095	0.00	0.00	exll6s
6.02	7.79	55490539	0.00	0.00	llex6s
0.58	0.75	14	53.57	6737.27	traham
0.24	0.31	66	4.70	11.02	pph6s
0.04	0.05	14	3.57	3.95	cage
0.01	0.01	14	0.71	1.10	put0
0.01	0.01	1	10.00	747.48	exp_l5
0.01	0.01	1	10.00	54.09	reste
0.01	0.01	1	10.00	15.34	rnf6s
0.00	0.00	76062	0.00	0.00	kill_nf
0.00	0.00	38044	0.00	0.00	check_rlf
0.00	0.00	38032	0.00	0.00	prxk6s
0.00	0.00	1474	0.00	0.00	ntph6s
0.00	0.00	164	0.00	0.00	exll3
0.00	0.00	164	0.00	0.00	llex3
0.00	0.00	156	0.00	0.00	prxk3
0.00	0.00	26	0.00	0.00	ntph3
0.00	0.00	14	0.00	0.00	wpb6s
0.00	0.00	5	0.00	0.00	uneix
0.00	0.00	2	0.00	341.69	exrec
0.00	0.00	1	0.00	0.00	amp3
0.00	0.00	1	0.00	0.00	amp6s
0.00	0.00	1	0.00	0.00	ccvl5
0.00	0.00	1	0.00	0.00	imp3
0.00	0.00	1	0.00	0.00	imp6s
0.00	0.00	1	0.00	95140.00	main
0.00	0.00	1	0.00	94377.18	nf6s
0.00	0.00	1	0.00	0.00	wctl5
0.00	0.00	1	0.00	0.00	wcvl
0.00	0.00	1	0.00	0.00	wea3

TABLE 8. Output of the profiler for a run (up to degree 16) of the program nf. The first two columns show the time spent by the program in this function; the next column contains the number of times the function is called. The next two columns show the average time per call ("self" refers to the function itself and "total" is based on a recursive total of the time taken by the function and everything it calls.) The routine mcount does not belong to our program but to the profiler.

half the total time taken by the program. This implies that if we were able to optimize these routines in order to reduce the time they take to almost zero, the factor in the total gain in speed would be close to 2 (but no better!). Moreover, Table 8 gives precise information about the routines one must optimize to make the program run faster.

Tables 5, 6 and 7 contain the time for several runs of the software. We stress that those are approximate values: time has been taken from a single run of the program, and the amount of RAM memory needed has been estimated form the size of the several expansions used (should increase these figures a little to obtain the real amount of memory used).

6. ERROR CONTROL

A very important point is to know the numerical errors introduced in the coefficients when this huge amount of computations is performed. A first, heuristic, indication is given by the size of the imaginary parts of the real normal forms, centre manifolds or first integrals that are not zero due to the roundoff errors. It is very natural to take these values as zero because they must vanish in an exact computation.

The testing methods discussed in Section 4.7 provide a rough idea of the global amount of error we have accumulated in the computations. This should be enough if we are only interested in numerical results, since this is typically the kind of output obtained from classical numerical methods (think of the solution of an ode, pde or simply the solution of a linear system). In fact, we are in a better position compared with other numerical procedures, since we have a good checking procedure.

However, if one is interested in these methods to be used in a computer assisted proof, we need a much better mechanism to control the error. This is the reason to introduce the interval arithmetic. In what follows, we will focus on a normal form computation, although the same ideas can be extended to the other examples considered here.

6.1. Interval Arithmetic

In order to carry exact bounds on the error, assume that, instead of a floating point number, we have an interval such that it contains the number. To add two intervals, we simply add the lower bounds of the interval *using rounding toward* $-\infty$, and we add the upper bounds *using rounding toward* $+\infty$. In this way we ensure that the result of the addition is contained in the final interval. The same ideas can be used to easily derive the operations of subtraction, multiplication and division.

The next step is to code efficiently those routines. Fortunately, most of the actual processors allow to the user to alter the rounding mode, to set a rounding toward $\pm\infty$ or to the nearest (this is the default). To do this, many compilers and/or operating systems have suitable functions in their libraries. Here we have used the corresponding routines of the Linux operating system (with the compiler gcc from Gnu), running on an Intel processor. The main disadvantages of this are that the memory requirements are doubled and the execution time is much bigger. This last inconvenient is due to the architecture of the processors, since when the rounding mode of the processor is changed, the pipeline of the processor is re-started with the corresponding loss of performance.

Since the code is in C++, it is very easy to use the overloading the arithmetic operators to replace standard complex arithmetic by our interval arithmetic (you can also use [Schelter 1991] if you want to avoid using C++). Then, it is not difficult to obtain the normal form but, instead of the coefficients, we will obtain intervals containing the exact values. This is what allows one to derive computer assisted proofs. See [Celletti and Chierchia 1988] and [Llave and Rana 1990] to see concrete applications of these ideas.

6.2. An Example with Interval Arithmetic

Here we have included the computation of the normal form around L_5 for the RTBP using interval artihmetic. The idea is to give a feeling about how these computations are.

Table 9 shows the normal form, using double precision interval arithmetic, around the L_5 point of the RTBP, for the mass parameter corresponding to the Earth–Moon system. We have skipped the imaginary parts because they can be assumed to be zero (this is one advantage of interval computations). It is interesting to compare these results with the ones presented in Table 1.

	lower bound	upper bound
1 0 0	9.5450087346978552 e−01	9.5450087346991741 e−01
0 1 0	−2.9820811951634596 e−01	−2.9820811951573489 e−01
0 0 1	1.0000000000000000 e+00	1.0000000000000000 e+00
2 0 0	1.1568661303889360 e−01	1.1568661401345537 e−01
1 1 0	−1.7127952451731403 e+00	−1.7127952303486182 e+00
0 2 0	3.3855424323176919 e−01	3.3855425662676453 e−01
1 0 1	8.9130919836368838 e−02	8.9130920112820977 e−02
0 1 1	2.2531870640182916 e−01	2.2531870757604811 e−01
0 0 2	−2.2354591590257877 e−03	−2.2354591074729147 e−03
3 0 0	−2.9479121860441637 e−01	−2.9478447589701773 e−01
2 1 0	8.1656201621290165 e+00	8.1657691558011720 e+00
1 2 0	−5.4586913901624575 e+02	−5.4586860598896601 e+02
0 3 0	−5.1021371160130911 e+01	−5.1021185629532283 e+01
2 0 1	−4.3799836956028315 e−01	−4.3799552187429924 e−01
1 1 1	1.4116969490546651 e+01	1.4116998940124972 e+01
0 2 1	2.0186927381142823 e+00	2.0187190572228246 e+00
1 0 2	−5.5905224456048508 e−02	−5.5904854484518651 e−02
0 1 2	−1.7898271680742539 e−01	−1.7898147963031263 e−01
0 0 3	−5.1334316020434586 e−05	−5.1317165340935330 e−05
4 0 0	1.2677680341002997 e+00	1.2873345241823699 e+00
3 1 0	−3.5434024811722338 e+01	−3.4703682770952582 e+01
2 2 0	−5.4877274309542030 e+04	−5.4872743283411488 e+04
1 3 0	3.2220252371445298 e+04	3.2226686164319515 e+04
0 4 0	3.5177942440398037 e+03	3.5192072384618223 e+03
3 0 1	2.1707021092443028 e+00	2.1811671985342400 e+00
2 1 1	1.9986363951466046 e+01	2.0216307091992348 e+01
1 2 1	1.3647290105217136 e+04	1.3647973048501415 e+04
0 3 1	1.4506020027436316 e+03	1.4508753202967346 e+03
2 0 2	2.1927585054381780 e+00	2.1948837131021719 e+00
1 1 2	−4.9551211330863225 e+01	−4.9529208559599283 e+01
0 2 2	−1.0188391081203008 e+01	−1.0169093839605921 e+01
1 0 3	3.5386579632358917 e−02	3.5564130055718124 e−02
0 1 3	7.0933774363425073 e−02	7.1488715816371950 e−02
0 0 4	5.1925348264703075 e−04	5.2452355219756441 e−04

TABLE 9. Coefficients of the normal form obtained for the Earth–Moon case using interval arithmetic. Only the real parts are presented.

	real part	imaginary part
1 0 0	0.9545008734698507 e+00	0.0000000000000000 e+00
0 1 0	−0.2982081195160388 e+00	0.0000000000000000 e+00
0 0 1	0.1000000000000000 e+01	0.0000000000000000 e+00
2 0 0	0.1156866135262217 e+00	−0.1927100002836750 e−32
1 1 0	−0.1712795237759768 e+01	−0.8974646880952045 e−32
0 2 0	0.3385542499303071 e+00	−0.4812484744069060 e−32
1 0 1	0.8913091997461692 e−01	−0.4814824860968090 e−33
0 1 1	0.2253187069890425 e+00	−0.1155557966632342 e−32
0 0 2	−0.2235459133244455 e−02	0.0000000000000000 e+00
3 0 0	−0.2947878472529007 e+00	−0.1521362462732897 e−29
2 1 0	0.8165694658984183 e+01	0.1185016987263866 e−28
1 2 0	−0.5458688725020474 e+03	−0.1174332188770398 e−28
0 3 0	−0.5102127839458834 e+02	−0.1863024703269571 e−28
2 0 1	−0.4379969457189379 e+00	−0.1484998366081301 e−30
1 1 1	0.1411698421534677 e+02	0.3358157455523530 e−29
0 2 1	0.2018705897693666 e+01	−0.3811527650397076 e−30
1 0 2	−0.5590503947042638 e−01	0.1150165425481089 e−30
0 1 2	−0.1789820982175625 e+00	−0.5503192785483820 e−31
0 0 3	−0.5132574067108261 e−04	−0.1954016422742883 e−32
4 0 0	0.1277551279966923 e+01	0.6516229084136752 e−27
3 1 0	−0.3506885376119049 e+02	0.1930958293998747 e−25
2 2 0	−0.5487500879622420 e+05	0.1108578486500471 e−24
1 3 0	0.3222346926821930 e+05	0.1264834400557989 e−24
0 4 0	0.3518500741321633 e+04	−0.2977673781142404 e−25
3 0 1	0.2175934654360213 e+01	0.1498922726564656 e−27
2 1 1	0.2010133553242287 e+02	0.4112326735122740 e−26
1 2 1	0.1364763157688629 e+05	0.4301991684680189 e−26
0 3 1	0.1450738661531158 e+04	0.6827018990166253 e−27
2 0 2	0.2193821109377304 e+01	0.2410685221214210 e−28
1 1 2	−0.4954020994411146 e+02	0.1635601948530436 e−27
0 2 2	−0.1017874245948335 e+02	0.1985868172512715 e−27
1 0 3	0.3547535485371232 e−01	−0.4504667333242595 e−30
0 1 3	0.7121124512960337 e−01	0.9633349924466193 e−29
0 0 4	0.5218885184995916 e−03	0.1527174289047186 e−30

TABLE 10. Coefficients of the normal form obtained for the Earth–Moon case using quadruple precision. The imaginary parts (last column) should be 0.

Note the big size of the intervals, especially for the highest degrees displayed. We have not optimized the algorithm to minimize the growing of the intervals, so a different implementation might lead to narrower intervals. Of course, the size of the intervals does not prove that the coefficients in Table 1 contain big numerical errors, but it suggests that we should check this more carefully. In order to do that, we can use higher-precision arithmetic. In this case, we have taken the standard quadruple precision arithmetic that it is contained in the libraries of many compilers (this concrete computation has been done on a Sun workstation). The results are displayed in Table 10. It is interesting to compare this last table with Table 1: if we take the coefficients in Table 10 as exact, we note that the error in the ones of Table 1 is of the order of the imaginary part. This suggests an heuristic criterion to estimate the accuracy of this computation.

Now, it is clear the amplification of errors that we have in this process. There are two (standard) ways of overcoming this:

1. Interval arithmetic. Although the intervals grow very fast, they are still providing exact bounds for the coefficients, that can be useful in order to derive computer assisted proofs (they are going to be a much sharper bound than any other estimation obtained by analytical methods).
2. Multiple precision arithmetic. This is the "brute force" solution, but it is valid in several cases.

The advantages are obvious, but one should note that, when dealing with real-life problems, it is not always appropriate: for instance, the mass parameter corresponding to the Earth–Moon case is only known up to 10 or 11 digits, so there is no gain in using multiple precision.

Of course, in academic problems it is always possible to use a combination of both, to derive very accurate coefficients and/or very sharp estimates for them.

Concerning the normal form around L_5 of the RTBP, we add that the amplification of errors is bigger when the mass ratio μ is smaller.

Finally, note that the routines for interval arithmetic and the extension for quadruple precision are not included in the software.

6.3. On Computer Assisted Proofs

The methods explained here allow, among others, the computation of manifolds (such as tori, first integrals, etc.) that are nearly invariant for the dynamics of the system. We will refer to these manifolds as approximately invariant objects. A natural question is whether an approximately invariant object is an approximation of a true invariant object or not. From a generic point of view, we already know that the answer is going to be positive in some cases (such as nonresonant invariant tori) and negative in some others (such as first integrals of nonintegrable systems).

To simplify the discussion, we focus on the proofs of existence of maximal-dimensional invariant tori. The standard results show that, under generic conditions of nondegeneracy and nonresonance, invariant tori are not destroyed by small perturbations but only slightly deformed; see Appendix A.5.2. (In fact, the essential condition is nondegeneracy because it allows one to obtain nonresonant conditions by simply changing a little bit the value of the actions.) We assume that we have rewritten (numerically) the original Hamiltonian as an integrable part (the normal form) plus a small perturbation (the remainder). To know whether a nonresonant torus of the normal form persists under the effect of the remainder we need a quantitative KAM theorem (giving concrete bounds on the size of the allowed perturbation) plus rigorous estimates on the size of the perturbation (remainder), in order to know if we are in the domain of applicability of the theorem. The estimates on the size of the perturbation for each concrete application are usually obtained by means of interval arithmetic; see [Llave and Rana 1990; Celletti and Chierchia 1988].

It is important to note that the only rigorous estimate needed is the size of the remainder. That is, it is possible to compute an approximately invariant torus by using standard floating point arithmetic, and then to estimate "how invariant" (i.e., the size of the remainder) is this torus by using interval arithmetic. This is usually much more efficient than to perform all the calculations using intervals; see [Llave and Rana 1990].

7. EXTENSIONS

In this package we have only considered the case of autonomous Hamiltonians with three degrees of freedom. It is not difficult to extend the ideas and the routines presented here to more degrees of freedom. For instance, to work with a four degrees of freedom Hamiltonian system (without any symmetry) one only needs to write the basic routines of the corresponding file `mp8.c`, and to introduce minor modifications in the other routines.

If one is interested in the computation of normal forms around another objects, in [Jorba and Villanueva 1998] it is explained (from a numerical point of view) the computation of the normal form around a periodic orbit of the spatial RTBP. The routines used there are based in the methodology explained here.

The case in which the Hamiltonian depends on time can also be considered. For instance, consider the Hamiltonian of the RTBP with a perturbation that depends periodically on time. In this case, one can still use the routines here but one has to change the basic arithmetic: now, the coefficients of the monomials are going to be Fourier series. We can store Fourier series in complex form as polynomials of one variable, using an array to put the coeficients and using the place inside the array to know the corresponding exponent (in this case one should say frequency instead of exponent). Since the relation between positions and frequencies is very easy one does not need to write any special function for this. (The case is altered drastically when one has to deal with quasiperiodic time-dependent functions, because the mapping between postions and frequencies is more complex. The main problem comes from the fact that these series are usually a little bit "sparse" and it is very convenient to store only the meaningful coefficients, to save memory. This is used and discussed in [Gómez et al. 1991c; 1993a].)

Then, one needs to write the arithmetic routines (sums and products) for these Fourier series and to use them instead of the complex arithmetic for the coefficients. This can be easily done if one uses a C extension allowing for overload of arithmetic operators, such as C++ or SCC [Schelter 1991]. Finally, you have to modify the input/output routines accordingly. This is what we have done in [Jorba and Simó 1994; Simó et al. 1995] for the case of a periodically perturbed Hamiltonian system.

APPENDIX A. BASIC HAMILTONIAN MECHANICS

In this appendix we give the basic definitions and properties related to Hamiltonian systems. The information presented here is biased towards the items needed in this paper. For a more complete and rigorous presentation see [Arnol'd 1978] or [Meyer and Hall 1992], for instance, or [Abraham and Marsden 1978] for a more formal approach.

To simplify the discussion, from now on we will assume (without explicit mention) that all the functions that will appear here are analytic.

A.1. Basic Definitions

A Hamiltonian system is a (continuous) dynamical system whose flow satisfies an ordinary differential equation of the type

$$\dot{q} = \frac{\partial H}{\partial p}, \quad \dot{p} = -\frac{\partial H}{\partial q}. \tag{A–1}$$

The variable $p \in \mathbb{R}^l$ is called the momentum and $q \in \mathbb{R}^l$ is called the position. The function $H \equiv H(p,q,t)$ is called the Hamiltonian of the system (A–1), and equations (A–1) are known as the Hamilton equations. Moreover, l is known as the number of degrees of freedom of the Hamiltonian H.

Define

$$J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix},$$

where I is the identity matrix $l \times l$. Then we can write equations (A–1) as

$$\dot{z} = J\nabla H(z), \quad z = (q,p).$$

Since J satisfies $J^T = -J$, it defines a symplectic form (that is, a nondegenerate bilinear skew-symmetric form) ω^0 on $\mathbb{R}^{2l}$:

$$\omega^0(u,v) = u^T J v, \quad \text{for } u,v \in \mathbb{R}^{2l}.$$

A matrix M is said to be symplectic if it satisfies

$$M^T J M = J.$$

A function $f : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R},\ f : (p,q) \mapsto f(p,q)$, is said to be a first integral of the Hamiltonian H if its surface levels are invariant by the flow (A–1), this is, if f takes a constant value on each orbit of the system. It is immediate to check that the function H is always a first integral of the Hamiltonian H.

The Poisson bracket of two functions $f(p,q)$ and $g(p,q)$ is defined as

$$\{f,g\} = \nabla f^T J \nabla g = \frac{\partial f}{\partial q}\frac{\partial g}{\partial p} - \frac{\partial f}{\partial p}\frac{\partial g}{\partial q}.$$

It is not difficult to show that, if f is a first integral of the Hamiltonian H, then it must satisfy $\{H,f\} = 0$.

Two functions $f(p,q)$ and $g(p,q)$ are said to be in involution if their Poisson bracket is zero:

$$\{f,g\} = 0.$$

The functions $\{f_j\}_{1\le j\le n}$ are said to be independent on some open domain D if the vectors $\{\nabla f_j\}_{1\le j\le n}$, defined on the domain D, are linearly independent on each point of the domain.

In the next sections we will use the following property of the Poisson bracket: if P_r and Q_s are homogeneous polynomials of degree r and s respectively, then $\{P_r, Q_s\}$ is an homogeneous plynomial of degree $r+s-2$.

In what follows, we will assume that all the Hamiltonians that will appear here are autonomous (they do not depend on time) and with l degrees of freedom.

A.2. Basic Properties

We assume that a Hamiltonian system H has l first independent integrals, $\{f_j\}_{1\le j\le l}$, that are in involution. We define $\mathcal{M}_0$ as

$$\mathcal{M}_0 = \{(p,q) : f_j(p,q) = f_j^{(0)} \text{ for } j = 1,\dots,l\}.$$

The well-known Liouville–Arnol'd theorem [Arnol'd 1978; Arnol'd et al. 1988] says that

1. the manifold $\mathcal{M}_0$ is invariant by the flow, and
2. if $\mathcal{M}_0$ is a compact connected manifold, it is diffeomorphic to the l-dimensional torus
$$\mathbb{T}^l = \{(\varphi_1, \dots, \varphi_l) \text{mod } 2\pi\}.$$

In this latter case it is possible to introduce, by means of a change of variables $(p, q) = F(I, \varphi)$ ($I \in \mathbb{R}^l$ is the new momentum and $\varphi \in \mathbb{T}^l$ is the new position) the so-called action-angle variables (I are the actions and φ are the angles). In these variables the Hamiltonian does not depend on the angles, $H = H(I)$, so the equations of motion are of the form
$$\dot{I} = 0, \quad \dot{\varphi} = \frac{\partial H}{\partial I} \equiv \omega(I).$$
These equations can be easily integrated:
$$I(t) = I_0, \quad \varphi(t) = \omega(I_0)t + \varphi_0.$$
If the values $\omega(I_0) \equiv \omega_0$ are linearly independent over the rationals, each solution is a dense quasiperiodic trajectory on a torus of dimension l. It is very common to use the frequency vector to identify a concrete torus of the system. If the map
$$I \mapsto \frac{\partial H}{\partial I}(I) \equiv \omega(I)$$
is a diffeomorphism (between suitable domains), it is also possible to identify a torus by the value of the action variable.

If $\langle k, \omega_0 \rangle = 0$ for some $k \in \mathbb{Z}^l$, then the orbits on this torus are not dense: if there are l_i independent frequencies, the torus $I = I_0$ contains a $(l - l_i)$-parametric family of l_i-dimensional tori, being each one densely filled by any trajectory starting on it. These tori of dimension l_i are known as lower-dimensional tori, while the tori of dimension l are called maximal-dimensional ones.

A.3. Canonical Transformations

We now consider the effect that the changes of variables have on Hamiltonian systems. Let $H(q, p)$ be a Hamiltonian function, and consider a change of variables $(q, p) = \Psi(x, y)$. The Hamilton equations obtained from the Hamiltonian $H \circ \Psi$ can be different from the equations obtained applying the transformation Ψ to the Hamilton equations related to H. When these differential equations coincide, it is said that the transformation Ψ preserves the Hamiltonian form.

A change of variables is called canonical when it preserves the Hamiltonian form (for any Hamiltonian function). It is not hard to show that a transformation is canonical if and only if the differential of the change (on any point) is a symplectic matrix.

Canonical transformations are very useful both from the theoretical and the practical points of view, since they allow to work on a single function (the Hamiltonian) instead of a system of $2l$ differential equations.

To produce canonical changes of variables is not an easy problem, since it is very difficult to impose that the differential be a symplectic matrix. Fortunately, there exists several techniques to produce such transformations. The one that we will use here is based on the following properties of the Hamiltonian flows:

1. Let $\Phi_t(x, y)$ be the time t flow of a Hamiltonian system. Then, $(q, p) = \Phi_t(x, y)$ is a canonical transformation.
2. Let $G(q, p)$ a Hamiltonian system with l degrees of freedom, and let $(q_0(t), p_0(t))$ be a solution of G. Then,
$$\frac{d}{dt} f(q_0(t), p_0(t)) = \{f, G\}(q_0(t), p_0(t)), \quad \text{(A–2)}$$
for any smooth function f.

It is not difficult to see that to transform a Hamiltonian H by means of the time 1 flow of a Hamiltonian G, we can apply the formula
$$\hat{H} \equiv H + \{H, G\} + \tfrac{1}{2!}\{\{H, G\}, G\} + \tfrac{1}{3!}\{\{\{H, G\}, G\}, G\} + \cdots, \quad \text{(A–3)}$$
where $\hat{H}$ denotes the transformed Hamiltonian. This formula is deduced applying the Taylor formula for the transformation and using (A–2) for the derivatives involved. The Hamiltonian G is usually called the generating function of the change of variables.

Formula (A–3) is very suitable for effective computations, since it can be easily implemented on a computer. All the operations involved are very simple if we are working with some kind of expansion, such as power expansions or Fourier expansions. One can argue that the problem for this kind of transformation (for a practical point of view) is that it is defined by an infinite series. This is not a problem since we usually work with a finite truncation of these series. This will produce a high-order approximation to the results wanted that, in many cases, are good enough for pratical purposes. On the other hand, it is possible to derive rigorous estimates on the size of this remainder so one can obtain bounds on the error of the results obtained with the truncated series (see [Simó 1989], [Jorba and Simó 1994] or [Jorba and Villanueva 1998] for numerical examples of this).

A.4. Normal Forms

We are going to restrict ourselves to the normal form around a fixed point of a Hamiltonian system. For normal forms around more complex objects (like periodic orbits or invariant tori), see [Bruno 1989; Jorba and Villanueva 1997a; 1997b].

Let H be a real analytic Hamiltonian of l degrees of freedom having an elliptic equilibrium point that, without loss of generality, we can assume that it is located at the origin. The case in which the equilibrium is of the type "some centres" times "some saddles" will be discussed later.

We start by expanding H in power series around the origin,

$$H(q,p) = H_2(q,p) + H_3(q,p) + H_4(q,p) + \cdots, \quad \text{(A–4)}$$

where $H_j(q,p)$ is an homogeneous polynomial of degree j in the variables (q,p). Our purpose is to perform (canonical) transformations in order to simplify as much as possible this expansion. Ideally, one would like to remove completely all the H_j with $j \geq 3$ but we will see that this is, generically, impossible. What we will show here is that, under some hypotheses, it is possible to remove the necessary terms to produce integrable approximations to the dynamics.

In order to simplify the subsequent steps, it is very convenient to simplify $H_2(q,p)$. Let A be the linearization of the Hamiltonian flow of H around the origin (i.e., $A = J\nabla H_2(0,0)$). Since A is an elliptic matrix, we can reduce it to $\hat{A} = C^{-1}AC$, being C a real matrix and $\hat{A}$ of the form

$$\hat{A} = \begin{pmatrix} \hat{A}_1 & & \\ & \ddots & \\ & & \hat{A}_l \end{pmatrix},$$

where the elements outside the $\hat{A}_j$ are zero and

$$\hat{A}_j = \begin{pmatrix} 0 & \omega_j \\ -\omega_j & 0 \end{pmatrix},$$

with $\omega_j \in \mathbb{R}$ for $j = 1, \ldots, l$. It is not difficult to check that this change can be selected canonical. If we call (x,y) to the new variables (x is the position and y the momentum), we want to note that the order ("permutation") of these variables to achieve this form for $\hat{A}$ is $(x_1, y_1, x_2, y_2, \ldots, x_l, y_l)$. In these coordinates, H_2 takes the form

$$\hat{H}_2(x,y) = \sum_{j=1}^{l} \frac{\omega_j}{2}\left(x_j^2 + y_j^2\right). \quad \text{(A–5)}$$

In order to simplify the computations in the normal form process (basically, the computations of generating functions), we will perform the (linear and symplectic) transformation

$$x_j = \frac{q_j + \sqrt{-1}\,p_j}{\sqrt{2}}, \quad y_j = \frac{\sqrt{-1}\,q_j + p_j}{\sqrt{2}}, \quad \text{(A–6)}$$

where we call (again) (q,p) to the new variables. In these variables, H_2 takes the form

$$H_2(q,p) = \sum_{j=1}^{l} \sqrt{-1}\,\omega_j q_j p_j.$$

In what follows, we will denote $\omega = (\omega_1, \ldots, \omega_l)$, and we will assume that the values ω_j, $1 \leq j \leq l$, are linearly independent over the rationals.

Assume that the initial expansion (A–4) has been rewritten in these variables, and we want to apply a sequence of canonical transformations (based on the scheme (A–3)). We start by trying to remove H_3, by means of a generating function G_3 that is also a homogeneous polynomial of degree 3. From (A–3) it is immediate to see that the monomials of degree 3 of the transformed Hamiltonian $\hat{H}$ obtained using a generating function G_3 are given by

$$\hat{H}_3 = H_3 + \{H_2, G_3\}.$$

We try to select a G_3 such that $\hat{H}_3$ is zero. To this end, we introduce some notation: if $z = (z_1, \dots, z_n)$ and $k = (k_1, \dots, k_n) \in \mathbb{N}^n$, we define

$$z^k = z_1^{k_1} \cdots z_n^{k_n}, \quad \text{with } |k| = k_1 + \cdots + k_n.$$

Then, we write H_3 and G_3 as

$$\begin{aligned} H_3(q,p) &= \sum_{|k_q|+|k_p|=3} h_{k_q,k_p} q^{k_q} p^{k_p}, \\ G_3(q,p) &= \sum_{|k_q|+|k_p|=3} g_{k_q,k_p} q^{k_q} p^{k_p}. \end{aligned}$$

The next step is to solve the equation $\hat{H}_3 = 0$. Note that $L_{H_2}(\,\cdot\,) = \{H_2, \cdot\}$ is a linear operator in diagonal form, because

$$\begin{aligned} L_{H_2}(q^{k_q}p^{k_p}) &= \{H_2, q^{k_q}p^{k_p}\} \\ &= \sqrt{-1}\,\langle k_p - k_q, \omega\rangle\, q^{k_q}p^{k_p}. \end{aligned}$$

This diagonal form is due to the complex coordinates introduced in (A–6). Now it is very easy to find a G_3 such that $\{H_2, G_3\} = -H_3$:

$$G_3(q,p) = \sum_{|k_q|+|k_p|=3} \frac{-h_{k_q,k_p}}{\sqrt{-1}\,\langle k_p - k_q, \omega\rangle} q^{k_q}p^{k_p}.$$

Of course, we need that the denominators $\langle k, \omega\rangle$ do not vanish for any $k \in \mathbb{Z}^l \setminus \{0\}$. Since $|k_q| + |k_p| = 3$, this condition is automatically satisfied if the components of the frequency vector $\omega = (\omega_1, \dots, \omega_l)$ are linearly independent over the rationals.

We rename the transformed Hamiltonian as H, that now takes the form

$$H(q,p) = H_2(q,p) + H_4(q,p) + H_5(q,p) + \cdots.$$

The next step is to look for a generating transformation G_4 (a homogeneous polynomial of degree 4), to remove the monomials of degree 4 from H. This is not possible in general, since L_{H_2} has some zero eigenvalues:

$$L_{H_2}(q^k p^k) = \{H_2, q^k p^k\} = 0.$$

Note that this never happens for monomials of odd degree. The monomials of the type $q^k p^k$ are usually called resonant monomials or unavoidable resonances. Hence, when we try to solve the equation $L_{H_2}(G_4) = -H_4$ we only can solve for the monomials of H_4 of the form $q^{k_q}p^{k_p}$, with $k_q \neq k_p$:

$$G_4(q,p) = \sum_{\substack{|k_q|+|k_p|=4 \\ k_q \neq k_p}} \frac{-h_{k_q,k_p}}{\sqrt{-1}\,\langle k_p - k_q, \omega\rangle} q^{k_q}p^{k_p}.$$

With this change, H takes the form (we call again H to the transformed Hamiltonian)

$$H(q,p) = H_2(q,p) + \bar{H}_4(q,p) + H_5(q,p) + \cdots, \qquad \text{(A–7)}$$

where

$$\bar{H}_4 = \sum_{|k|=2} \bar{h}_k q^k p^k.$$

Fortunately, the monomials present in $\bar{H}_4$ do not obstruct integrability: let's skip the terms in (A–7) of order bigger than 4 (this is what we call the normal form of the initial Hamiltonian (A–4) up to degree 4). Apply the canonical transformation

$$\left.\begin{aligned} x_j &= I_j^{1/2} \exp(\sqrt{-1}\,\varphi_j), \\ y_j &= -\sqrt{-1}\, I_j^{1/2} \exp(-\sqrt{-1}\,\varphi_j) \end{aligned}\right\}, \qquad \text{(A–8)}$$

where $j = 1, \dots, l$, so that the truncated Hamiltonian takes the form

$$H = H(I) = \langle \omega, I\rangle + H_2(I),$$

with

$$H_2(I) = \sum_{|k|=2} \bar{h}_k I^k \quad \text{and} \quad I = (I_1, \dots, I_l).$$

This is now an integrable Hamiltonian, that gives an approximate description of the dynamics around the equilibrium point. The equations of motion are

$$\dot{I} = 0, \quad \dot{\varphi} = \frac{\partial H(I)}{\partial I} = \bar{\omega}(I).$$

The solutions are $I = I_0$ (which corresponds to invariant tori) and $\varphi = \bar{\omega}(I_0)t + \varphi_0$, which is a quasiperiodic flow on the torus.

Of course, the process of reduction to normal form can be done up to any finite order. Omitting the remainder and using (A–8) we obtain a Hamiltonian like

$$H = H(I) = \langle \omega, I\rangle + \sum_{n=2}^{N} H_n(I).$$

A.4.1. On the convergence. Generically, normal form reduction is a divergent process. The divergence is mainly due to the effect of the divisors $\langle k, \omega\rangle$ that appear in the generating functions (in fact, it is possible to have divergence even in the absence of small divisors; see [Jorba and Llave $\geq$ 1999]). In order to control the size of these denominators, it is usual to ask for a Diophantine condition such as

$$\bigl|\langle k, \omega\rangle\bigr| > \frac{c}{|k|^{\gamma}}, \qquad \text{(A–9)}$$

with $k \in \mathbb{Z} \setminus \{0\}$ and $\gamma > l-1$. This allows one to derive estimates on the size of the remainder obtained when we stop the normal form to some order N. The set of ω such that condition (A–9) is not satisfied has Lebesgue measure $O(c)$.

In fact, one may look at a normal form as a power expansion of a nonanalytic C^∞ function at the origin. The power series is divergent but, if we stop the expansion to some order N, the remainder behaves like $O(R^{N+1})$, where R denotes the distance to the origin. This last property is what makes these expansions useful.

A.5. Stability

Here we will explain some of the applications of the normal forms. Let us consider the neighbourhood of an elliptic equilibrium point (that we locate at the origin) of a $l > 1$ degrees of freedom autonomous Hamiltonian system $H(q,p)$. Consider an initial condition close to the origin. We are interested in knowing if the corresponding trajectory will be close to the origin for all times (stability in the sense of Lyapounov), or if it is going to escape to a distance $O(1)$ from the equilibrium point.

A.5.1. The Dirichlet theorem. This is a particular case in which the stability problem can be easily solved. Call M the Hessian matrix of the Hamiltonian at the origin (we recall that M is symmetric and that $\nabla_{(q,p)}H(0,0) = 0$). Assume that M is a positive definite matrix. Then, the Dirichlet theorem says that origin is Lyapounov stable.

The proof is based on the fact that, close to the point, the level surfaces of the Hamiltonian are "like ellipsoids" having the origin inside (those manifolds are of codimension 1 so they split the phase space). Then, since they are invariant for the dynamics, they act as a barrier that the trajectories starting near the point cannot cross. The same argument holds if there exists a first integral, defined on a neighbourhood of the origin, that is positive definite at $(0,0)$.

Unfortunately, there are many interesting cases where the matrix M is not positive definite and, hence, we need a different kind of results to study the stability.

A.5.2. KAM and Nekhoroshev theory. In the last section we have seen that, using a finite number of steps of a normal form scheme, we can put the Hamiltonian into the form

$$H(x,y) = H_2(x,y) + \sum_{j=3}^{2N} \bar{H}_j(x,y) + \mathcal{R}_{2N}(x,y).$$

Now, using condition (A–9) it is possible to derive estimates on the size of the remainder $\mathcal{R}_{2N}$ that are of the kind $c_1 \exp(-c_2(1/R)^{2/(\gamma+1)})$ ($c_1 > 0$, $c_2 > 0$). Here R denotes the radius of the ball centred at the origin on which we take the norm of $\mathcal{R}_{2N}$, and it is assumed to be sufficiently small. This has been obtained optimizing the size of the remainder with respect to the degree up to which the normal form is obtained, for each value of R.

From this bound on the remainder, it is not diffcult to obtain lower bounds on the diffusion time (i.e., the time to move away) around the point. For instance, if we call $T(R)$ to the time to go out from a ball of radius $2R$ starting in a ball of radius R, we have

$$T(R) \geq c_3 \exp\left(c_4 \left(\frac{1}{R}\right)^{2/(\gamma+1)}\right),$$

c_3 and c_4 being positive constants (to obtain this estimate, analyticity plays an essential role). Of course, this is not a proof of stability but a "bound on the unstability". This kind of estimates are what is usually called Nekhoroshev estimates.

A second approach is to try to remove completely the remainder. This cannot be done using the normal form scheme we have explained in the previous sections, but it can be done through a Newton method. This is a quadratically convergent iterative scheme, that only converges on a Cantor set of the phase space. On this Cantor set, the trajectories take place on invariant tori and, hence, they never go away from a vicinity of the point. The Lebesgue measure of the complementary of this Cantor set can be bounded by $c_5 \exp(-c_6(1/R)^{2/(\gamma+1)})$, $c_5 > 0$, $c_6 > 0$. This kind of results belong to the so-called KAM theory. To decide about the stability we must take into account the motion outside the Cantor set of invariant tori. For instance, consider first the case $l = 2$. The phase space is four-dimensional and, fixing the energy level $H = h$ we restrict to a three-dimensional space. The invariant tori are of dimension 2 so they split the phase space and, hence, this allows one to conclude the Lyapounov stability of the elliptic point. The case $l = 3$ (or bigger) is much more difficult. The reason is this: fixing the energy level produces a five-dimensional invariant manifold and the invariant tori are three-dimensional so they do not split phase space and we cannot conclude stability. In fact, the stability of Hamiltonian systems with three or more degrees of freedom is today an open question. The more accepted conjecture says that they are, generically, unstable [Arnol'd 1964]. The unstability mechanism is usually known as Arnol'd diffusion.

It is outside the scope of this paper to give detailed explanations of these results. We refer the reader to [Arnol'd and Avez 1968] or [Arnol'd et al. 1988] for a general explanation, and to [Jorba and Villanueva 1997a] for more concrete results about invariant objects such as elliptic points, periodic orbits, and invariant tori.

A.6. Centre Manifolds

Now consider a Hamiltonian with three degrees of freedom, in a neighbourhood of an equilibrium point of the type centre $\times$ centre $\times$ saddle, that we will assume to be the origin. Of course, this is an unstable equilibrium point but we are interested in the existence of trajectories that remain close to the point for all times. If we consider the linearization of the vector field at this point, and we skip the hyperbolic part, we obtain a couple of harmonic oscillators. Hence, for the linearized vector field, we have a couple of families of periodic orbits near the point, plus the quasiperiodic solutions obtained as product of the two families of periodic orbits. These quasiperiodic solutions are sometimes called Lissajous orbits. Consider now the effect of the nonlinear terms of the vector field on these bounded solutions. Under generical conditions the well-known Lyapounov centre theorem says that, for each linear (periodic) oscillation, there exists a one-parametric family of periodic orbits of the complete Hamiltonian system that emanates from the point in a tangent way to the linear family of oscillations. The limit frequency of these periodic orbits at the fixed point is the frequency of the linear oscillations (for a proof see [Siegel and Moser 1971], for instance). A similar result holds for the Lissajous orbits. Under general hypotheses, it can be shown that these linear oscillations can be extended to the complete system as a Cantorian family of invariant tori. Moreover, the measure of the gaps between tori is exponentially small with the distance to the origin (for the proofs, see [Jorba and Villanueva 1997a]).

To give a more accurate description of the dynamics around the point, we apply a normal form technique, as has been done in previous sections. We start expanding the Hamiltonian in power series around the point, as in (A–4). Next, we write the second degree terms H_2 in real coordinates such that

$$H_2(x,y) = \lambda x_1 y_1 + \frac{\omega_2}{2}(x_2^2+y_2^2) + \frac{\omega_3}{2}(x_3^2+y_3^2),$$

for $(\lambda, \omega_2, \omega_3) \in \mathbb{R}^3$. The coordinates x_1, y_1 are already in diagonal form, so we only need to complexify the pairs (x_2, y_2) and (x_3, y_3). Using the change (A–6) for these two pairs we obtain

$$H_2(q,p) = \lambda q_1 p_1 + \sqrt{-1}\,\omega_2 q_2 p_2 + \sqrt{-1}\,\omega_3 q_3 p_3.$$

Now we can start a normal form process as the one described in Section A.4 but, instead of killing all the possible monomials, we will only kill the monomials such that the exponent of q_1 is different from the exponent of p_1 (for a different killing criterion, see [Simó 1996]). That is, the generating function used to remove monomials of degree n will be of the form

$$\sum_{k_{q_1}\neq k_{p_1}} \frac{-h_{k_q k_p}}{(k_{p_1}-k_{q_1})\lambda+\sqrt{-1}\,(k_{p_2}-k_{q_2})\omega_2+\sqrt{-1}\,(k_{p_3}-k_{q_3})\omega_3}.$$

Since $k_{q_1} \neq k_{p_1}$, the denominators of the generating function are bounded from below. This is why the normal form process diverges very slowly (like an harmonic series; see [Jorba and Llave $\geq$ 1999]).

If we stop this scheme after a finite number of steps, we obtain a Hamiltonian of the form

$$H(q,p) = H_N(q_1p_1, q_2, q_3, p_2, p_3) + \mathcal{R}(q,p).$$

Neglecting the remainder $\mathcal{R}$, which is very small near the origin, we can define $I_1 = q_1 p_1$ (this is a canonical change if we define properly the corresponding angle variable) to obtain a Hamiltonian

$$H_N(I_1, q_2, q_3, p_2, p_3).$$

The equation corresponding to the variable I_1 is $\dot{I}_1 = 0$, so this is a first integral of the system. Selecting the value $I_1 = 0$ we are restricting the Hamiltonian H_N to an invariant manifold that is tangent at the origin with the linear central part of the system. This is the so called reduction to the centre manifold.

Once I_1 has been replaced by 0, we have obtained a two degrees of freedom Hamiltonian system $H_c \equiv H_N(0, q_2, q_3, p_2, p_3)$, where the origin is an elliptic equilibrium point. It is not difficult to produce a qualitative description of the dynamics of H_c: the phase space is four-dimensional, so fix a energy level $H_c = h_c$ to reduce to a three-dimensional phase space. Now, Poincaré sections are two-dimensional and can be plotted easily. Doing several plots for several values of h_c one gets a description of the trajectories that remain close to the origin. The dynamics of the initial Hamiltonian near the origin can be obtained adding the hyperbolic part that we have skipped when reducing to the centre manifold. See [Jorba and Masdemont 1999] or [Jorba and Masdemont 1998] for examples of this.

Although this reduction is divergent in general, we can apply KAM techniques to show, under suitable hypotheses, the existence of a Cantorian centre manifold, completely filled up by invariant tori. (A Cantorian manifold is parametrized by two parameters, each of which moves in a Cantor set.) The complementary of the measure of this manifold (in the parameters space) decreases exponentially with the distance to the origin. See [Jorba and Villanueva 1997a] for more details. For a general discussion of the main features of centre manifolds, see [Sijbrand 1985] or [Vanderbauwhede 1989].

A.7. First Integrals

Again, consider the dynamics near an equilibrium point of a Hamiltonian system. Now we are interested in producing first integrals of the motion. Of course, if the Hamiltonian is not integrable (this is, in fact, the general case) these integrals are not going to exist but, as we will see, it is still possible to produce approximate first integrals that can be useful for some applications.

To simplify the discussion, we will assume that the equilibrium point is at the origin and that it is of elliptic type. The case in which some directions are hyperbolic can be done in a very similar way.

As in the previous cases, assume that the Hamiltonian is expanded in power series as in (A–4), with H_2 in diagonal form as in (A–5). Denote by F the desired first integral, which we will expand in power series around the origin as $F = \sum_{j\geq 2} F_j$, where F_j denotes a homogeneous polynomial of degree j. From the condition $\{H, F\} = 0$ it is immediate to obtain the recurrence

$$\{H_2, F_n\} = -\sum_{j=3}^{n} \{H_j, F_{n-j+2}\}. \tag{A–10}$$

Hence, due to the diagonal form of H_2, it is very easy to solve F_n in terms of $F_2, \ldots, F_{n-1}$, assuming the standard nonresonant conditions on the frequencies of the point.[1] Then, given a F_2, we can compute the following terms F_3, F_4 and so on.

As usual, the series $F = \sum_{j\geq 2} F_j$ is divergent. However, from its asymptotic character we can derive quasi-integrals of motion by simply truncating the series to finite order. This means that, if f_n denotes a quasi-integral and $(q(t), p(t))$ is an orbit of the Hamiltonian system H then,

$$\dot{f}_n(q(t), p(t)) = \{H, f_n\}(q(t), p(t))$$

Bounding the Poisson bracket of this formula in a neighbourhood of the elliptic point one can derive estimates on the diffusion time near the point. For an application of these techniques, see [Celletti and Giorgilli 1991]. See also [Marchal 1980] for an early construction of quasi-integrals.

APPENDIX B. LINEAR NORMAL FORM FOR THE EQUILIBRIUM POINTS OF THE RTBP

We start with a brief description of the so-called restricted three body problem (RTBP). More details can be obtained in [Szebehely 1967] or other textbooks on celestial mechanics.

Consider two point masses (usually called primaries) that attract each other according to the gravitational Newton's law. Assume that they are moving in circular orbits around their common centre of masses, and consider the motion of an infinitesimal particle (here, infinitesimal means that its mass is so small that we neglect the effect it has on the motion of the primaries and we only take into account the effect of the primaries on the particle) under the

[1] As already mentioned, the operator $L_{H_2}(\cdot) = \{H_2, \cdot\}$ is not bijective. Then it is possible that, if the right hand side of (A–10) contains resonant monomials, this equation cannot be solved. There are several cases when it can be proved that such monomials never appear. See [Celletti and Giorgilli 1991] for a discussion of this.

attraction of the two primaries. The study of the motion of the infinitesimal particle is what is known as RTBP.

To simplify the equations of motion, take units of mass, length and time such that the sum of masses of the primaries, the gravitational constant and the period of the motion of the primaries is 1, 1 and 2π respectively. With these units the distance between the primaries is also equal to 1. We denote as μ the mass of the smallest primary (the mass of the biggest is then $1-\mu$), $\mu \in (0, \frac{1}{2}]$.

The system of reference is defined as follows: the origin is taken at the centre of masses of the primaries, the X-axis points to the biggest primary (with this orientation), the Z-axis points to the direction of the vector of angular motion of the primaries with respect to their common centre of mass (it is perpendicular to the plane of motion) and the Y-axis is defined such that we obtain an orthogonal, positive-oriented system of reference. Note that we have defined a rotating system of reference, that is usually called synodic. In this system, the primary of mass μ is at the point $(\mu-1, 0, 0)$ and the one of mass $1-\mu$ is at $(\mu, 0, 0)$.

Defining momenta as $P_X = \dot{X} - Y$, $P_Y = \dot{Y} + X$ and $P_Z = \dot{Z}$, the equations of motion can be written in Hamiltonian form. The corresponding Hamiltonian function is

$$H = \tfrac{1}{2}(P_X^2 + P_Y^2 + P_Z^2) + YP_X - XP_Y - \frac{1-\mu}{r_1} - \frac{\mu}{r_2}, \tag{B–1}$$

with

$$r_1^2 = (X-\mu)^2 + Y^2 + Z^2,$$
$$r_2^2 = (X-\mu+1)^2 + Y^2 + Z^2.$$

It is well-known that the system defined by (B–1) has five equilibrium points. Two of them can be found as the third vertex of the two equilateral triangles that can be formed using the two primaries as vertices (usually called $L_{4,5}$ or Lagrangian points). The other three lie on the X-axis and are usually called $L_{1,2,3}$ or Eulerian points; see Figure 3.

In the next sections we will study the linear behaviour around these equilibrium points. We will obtain the linear normal form around them as well as the corresponding (symplectic) changes of variables. These calculations, summarized in [Giorgilli et al. 1989; Gómez et al. 1991c], are given here in detail, for completeness.

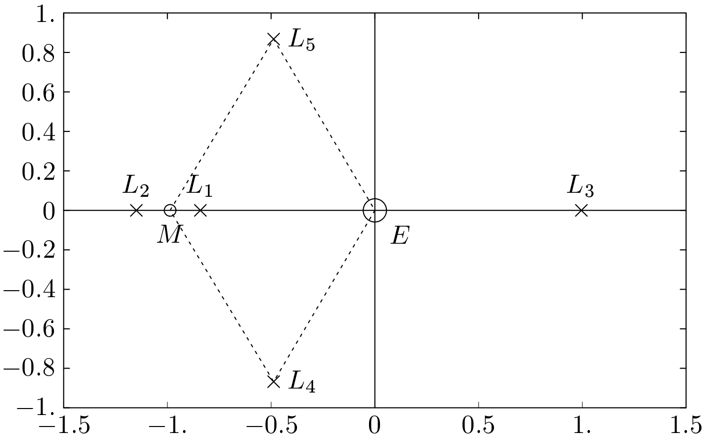

FIGURE 3. The five equilibrium points of the RTBP. The graphic corresponds to the Earth–Moon case, with $\mu \approx 0.01215$.

B.1. The Equilateral Points

The equilibrium points L_4 and L_5 are located at $(\mu - \frac{1}{2}, \mp\frac{\sqrt{3}}{2}, 0)$, where the upper ("−") sign is for L_4 while the lower ("+") one is for L_5. These points are known to be linearly stable when the mass parameter μ is less than the Routh critical value $\mu_R = \frac{1}{2}\bigl(1 - \sqrt{\frac{23}{27}}\bigr) \approx 0.03852$. In what follows we will assume that our mass parameter is less than μ_R (the interested reader should not have any problem to complete the opposite case).

The first step is to translate the origin of coordinates to the equilibrium point. This is done applying the (symplectic) change

$$\begin{aligned} X &= x + \mu - \tfrac{1}{2}, & P_X &= p_x \pm \tfrac{\sqrt{3}}{2}, \\ Y &= y \mp \tfrac{\sqrt{3}}{2}, & P_Y &= p_y + \mu - \tfrac{1}{2}, \\ Z &= z, & P_Z &= z, \end{aligned}$$

to the Hamiltonian (B–1). As before, the upper sign is for the L_4 case and the lower one for the L_5 case (this rule for the signs will be used along this section). To simplify the notation, we call again H to the Hamiltonian obtained,

$$H = \tfrac{1}{2}(p_x^2 + p_y^2 + p_z^2) + yp_x - xp_y + \bigl(\tfrac{1}{2} - \mu\bigr)x \mp \tfrac{\sqrt{3}}{2}y - \frac{1-\mu}{r_{PS}} - \frac{\mu}{r_{PJ}},$$

where

$$r_{PS}^2 = (x-x_S)^2 + (y-y_S)^2 + z^2,$$
$$r_{PJ}^2 = (x-x_J)^2 + (y-y_J)^2 + z^2,$$

$x_S = \frac{1}{2}$, $y_S = \mp\frac{\sqrt{3}}{2}$, $x_J = -\frac{1}{2}$ and $y_J = \mp\frac{\sqrt{3}}{2}$. (The subscripts stand for "Sun" and "Jupiter", and correspond to a classical example for the RTBP, where the small particle can be an asteroid.) Note that $(x_S, y_S, 0)$ are the coordinates of the big primary in the new coordinates and that $(x_J, y_J, 0)$ is the position of the small one.

The next step is to expand H around the origin. Since the origin is an equilibrium point, the first-order terms must vanish (we simply don't care about the constant value $H(0)$, since it is irrelevant to the dynamics). The first nontrivial terms are of second order and they are responsible for the linear dynamics around the point. They are

$$H_2 = \tfrac{1}{2}(p_x^2+p_y^2+p_z^2)+yp_x-xp_y+\tfrac{1}{8}x^2-\tfrac{5}{8}y^2-axy+\tfrac{1}{2}z^2,$$

where $a = \pm\frac{3\sqrt{3}}{4}(1-2\mu)$. The behaviour in the (z, p_z) directions is uncoupled from the behaviour in the (x, y, p_x, p_y) directions. Moreover, the motion on the z-axis corresponds to an harmonic oscillator with frequency 1 (for all μ), that it is already in (real) normal form. Hence, we restrict ourselves to the (x, y, p_x, p_y)-plane:

$$H_2 = \tfrac{1}{2}(p_x^2+p_y^2)+yp_x-xp_y+\tfrac{1}{8}x^2-\tfrac{5}{8}y^2-axy. \qquad \text{(B–2)}$$

Define the 4×4 matrix J as

$$J = \begin{pmatrix} 0 & I_2 \\ -I_2 & 0 \end{pmatrix},$$

where I_2 denote the 2×2 identity matrix. The equations of motion of (B–2) are given by the linear system

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{p_x} \\ \dot{p_y} \end{pmatrix} = J\nabla H_2 = J\,\mathrm{Hess}(H_2) \begin{pmatrix} x \\ y \\ p_x \\ p_y \end{pmatrix}. \qquad \text{(B–3)}$$

An easy computation shows that the matrix $M = J\,\mathrm{Hess}(H_2)$ is given by

$$M = \begin{pmatrix} 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ -\frac{1}{4} & a & 0 & 1 \\ a & \frac{5}{4} & -1 & 0 \end{pmatrix}. \qquad \text{(B–4)}$$

The characteristic polynomial is $p(\lambda) = \lambda^4 + \lambda^2 + \frac{27}{16} - a^2$. From this expression it is easy to conclude that system (B–3) is stable if $\mu \le \mu_R = \frac{1}{2}\big(1-\sqrt{\frac{23}{27}}\big)$ (this is the so-called Routh mass) and unstable if $\mu_r < \mu \le \frac{1}{2}$. Since we are studying the case $\mu < \mu_R$, we assume that the solutions of $p(\lambda) = 0$ are all purely imaginary, that is, $\lambda_j = \pm\omega_j\sqrt{-1}$ for $j = 1, 2$. The real values ω_j are the frequencies of the linear oscillations at the equilibrium points $L_{4,5}$, and it is trivial to show that they always differ when $0 < \mu < \mu_R$. Call ω_1 the one that satisfies $\omega_1^2 > \frac{1}{2}$ and ω_2 the one such that $\omega_2^2 < \frac{1}{2}$. For the moment we do not specify the sign we take for each frequency. These signs will be determined below.

Now we want to obtain a real (and symplectic) change of variables such that the Hamiltonian (B–2) is reduced to its (real) normal form. The first step will be to look for the eigenvectors of the matrix M given by (B–4). To simplify the computation, we wil take advantage of the special form of this matrix. We denote by M_λ the matrix $M - \lambda I_4$, and we define a splitting

$$M_\lambda = \begin{pmatrix} A_\lambda & I_2 \\ B & A_\lambda \end{pmatrix}$$

into 2×2 blocks

$$A_\lambda = \begin{pmatrix} -\lambda & 1 \\ -1 & -\lambda \end{pmatrix}, \quad B = \begin{pmatrix} -\frac{1}{4} & a \\ a & \frac{5}{4} \end{pmatrix},$$

where λ denotes one of the eigenvalues of the matrix M. The kernel of M_λ is now easy to find: to solve

$$\begin{pmatrix} A_\lambda & I_2 \\ B & A_\lambda \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

we can start by solving $(B - A^2)w_1 = 0$ and then $w_2 = -Aw_1$ (note that the kernel of $B - A^2$ is trivial to find since it is a 2×2 matrix). In this way, we find the eigenvector

$$(2\lambda + a, \lambda^2 - \tfrac{3}{4}, \lambda^2 + a\lambda + \tfrac{3}{4}, \lambda^3 + \tfrac{5}{4}\lambda + a)^T.$$

Since the eigenvalues of M satisfy $\lambda = \sqrt{-1}\,\omega$, $\omega \in \mathbb{R}$, we conclude that the frequencies ω are determined by the equation

$$\omega^4 - \omega^2 + \tfrac{27}{16} - a^2 = 0. \qquad \text{(B–5)}$$

We also apply $\lambda = \sqrt{-1}\,\omega$ to the expression of the eigenvector. Separating real and imaginary parts, we conclude that it can be expressed as $u + \sqrt{-1}\, v$, where

$$
\begin{aligned}
u(\omega) &= \left(a, -\omega^2 - \tfrac{3}{4}, -\omega^2 + \tfrac{3}{4}, a\right)^T, \\
v(\omega) &= \left(2\omega, 0, a\omega, -\omega^3 + \tfrac{5}{4}\omega\right)^T.
\end{aligned}
\tag{B–6}
$$

We start considering the change of variables given by the matrix $C = (u_1, u_2, v_1, v_2)$, where u_j and v_j denote the values of u and v given by (B–6) corresponding to the frequencies ω_j, $j = 1, 2$. For the moment we do not specify which sign is taken for each frequency. In order to know whether C is simplectic or not, we check the property $C^T J C = J$: a tedious but not difficult computation produces

$$
C^T J C = \begin{pmatrix} 0 & D \\ -D & 0 \end{pmatrix}, \quad D = \begin{pmatrix} d(\omega_1) & 0 \\ 0 & d(\omega_2) \end{pmatrix}.
$$

where $d(\omega) = \omega(2\omega^4 + \frac{1}{2}\omega^2 - \frac{3}{4})$. Of course, to derive this expression you need to use the properties (B–5) and $\omega_1^2\omega_2^2 = \frac{27}{16} - a^2$. Note that the zeros obtained in $C^T J C$ and D were expected, due to the way we have constructed C. The only question was to know whether d were 1 or not. Since it is not, we need to perform some scaling to the columns of C: define $s_j = \sqrt{d(\omega_j)}$, for $j = 1, 2$, and redefine C as $(u_1/s_1, u_2/s_2, v_1/s_1, v_2/s_2)$. This matrix is now symplectic, but we also want C to be real, that is, we want the values $d(\omega_j)$ to be positive. This will determine the signs we must choose for the frequencies ω_j. Since $\omega_1^2 < \frac{1}{2}$, if one wants $d(\omega_1) > 0$ is necessary to take $\omega_1 > 0$ and, conversely, as $\omega_2^2 < \frac{1}{2}$ implies that we must take $\omega_2 < 0$ in order to have $d(\omega_2) > 0$. Hence, the change we have obtained is real, symplectic and it brings the Hamiltonian (B–2) into the real normal form

$$
H_2 = \frac{\omega_1}{2}(x^2 + p_x^2) + \frac{\omega_2}{2}(y^2 + p_y^2), \tag{B–7}
$$

where we recall that $\omega_1 > 0$ and $\omega_2 < 0$.

In the paper we have used a complex normal form for H_2, because it allows one to solve the homological equation that determines the generating function (see Section A.4) in a very easy way. Now it is not difficult to derive the change that brings (B–7) into complex normal form. We compose the complexifying change

$$
x = \frac{q_1 + \sqrt{-1}\, p_1}{\sqrt{2}}, \quad p_x = \frac{\sqrt{-1}\, q_1 + p_1}{\sqrt{2}},
$$
$$
y = \frac{q_2 + \sqrt{-1}\, p_2}{\sqrt{2}}, \quad p_y = \frac{\sqrt{-1}\, q_2 + p_2}{\sqrt{2}},
$$

with the above-defined matrix C to produce the final change used in the paper. If we define $r_j = \sqrt{\omega_j\left(4\omega_j^4 + \omega_j^2 - \frac{3}{2}\right)}$ for $j = 1, 2$, the matrix of this change is

$$
\begin{pmatrix}
\dfrac{a}{r_1} + \dfrac{2\omega_1}{r_1}\sqrt{-1} & \dfrac{2\omega_1}{r_1} + \dfrac{a}{r_1}\sqrt{-1} & \vdots & \vdots \\
\dfrac{-\omega_1^2 - \frac{3}{4}}{r_1} & \dfrac{-\omega_1^2 - \frac{3}{4}}{r_1}\sqrt{-1} & \vdots & \vdots \\
\dfrac{-\omega_1^2 + \frac{3}{4}}{r_1} + \dfrac{a\omega_1}{r_1}\sqrt{-1} & \dfrac{a\omega_1}{r_1} + \dfrac{-\omega_1^2 + \frac{3}{4}}{r_1}\sqrt{-1} & \vdots & \vdots \\
\dfrac{a}{r_1} + \dfrac{-\omega_1^3 + \frac{5}{4}\omega_1}{r_1}\sqrt{-1} & \dfrac{-\omega_1^3 + \frac{5}{4}\omega_1}{r_1} + \dfrac{a}{r_1}\sqrt{-1} & \vdots & \vdots
\end{pmatrix},
$$

where the columns indicated by dots are obtained from the first two by replacing the subscript 1 by 2 everywhere. Here the order is (x, y, p_x, p_y) for the initial variables and (q_1, q_2, p_1, p_2) for the final variables. (In the implementation of the software the order is different.)

B.2. The Collinear Points

For $j = 1, 2$, define γ_j as the distance from the smallest primary (the one of mass μ) to the point L_j. (Note: the literature disagrees on the convention for the subscripts 1 and 2; typically, books on astrodynamics define L_1 and L_2 as in our Figure 3, but books on celestial mechanics, including the already-cited reference [Szebehely 1967], tend to interchange them.) Let γ_3 be the distance from the biggest primary to L_3. It is well-known — see [Szebehely 1967], for example — that γ_j is the only positive solution of the Euler quintic equation, which takes the form

$$
\gamma_j^5 \mp (3 - \mu)\gamma_j^4 + (3 - 2\mu)\gamma_j^3 - \mu\gamma_j^2 \pm 2\mu\gamma_j - \mu = 0
$$

for $j = 1, 2$ (where the upper sign is for $j = 1$ and the lower one for $j = 2$), and

$$
\gamma_j^5 + (3 - \nu)\gamma_j^4 + (3 - 2\nu)\gamma_j^3 - \nu\gamma_j^2 - 2\nu\gamma_j - \nu = 0
$$

for $j = 3$, with $\nu = 1 - \mu$. These equations can be solved numerically by Newton's method, using the starting point $(\frac{1}{3}\mu)^{1/3}$ for $j = 1, 2$ and $1 - \frac{7}{12}\mu$ for $j = 3$.

Next step would be to translate the origin to the selected point L_j, as has been done for the triangular points. In this case, however, to have good numerical properties for the coefficients of the final expansions it is better to perform some scaling; see [Richardson 1980; Gómez et al. 1991c; 1997]. Since the scalings are not symplectic transformations, consider the following process: first we write the differential equations related to (B–1) and then, on these equations, we perform the substitution

$$\begin{aligned} X &= \mp\gamma_j x + \mu + \alpha_j, \\ Y &= \mp\gamma_j y, \\ Z &= \gamma_j z, \end{aligned}$$

where the upper sign corresponds to $j = 1, 2$ and the lower to $j = 3$, and we have set $\alpha_1 = -1 + \gamma_1$, $\alpha_2 = -1 - \gamma_2$ and $\alpha_3 = \gamma_3$. The unit of distance is now the distance from the equilibrium point to the closest primary. Finally, it is not difficult to check that the differential equations obtained can be rewritten in Hamiltonian form, with Hamiltonian

$$H_{L_j} = \frac{1}{2}(p_x^2 + p_y^2 + p_z^2) + yp_x - xp_y - \sum_{n\geq 2} c_n(\mu)\rho^n P_n\left(\frac{x}{\rho}\right),$$

where $\rho^2 = x^2 + y^2 + z^2$, P_n is the Legendre polynomial of degree n and the coefficients $c_n(\mu)$ are given by

$$c_n(\mu) = \frac{1}{\gamma_j^3}\left((\pm 1)^n\mu + (-1)^n\frac{(1-\mu)\gamma_j^{n+1}}{(1\mp\gamma_j)^{n+1}}\right)$$

for $j = 1, 2$ (where as before the upper sign is for $j = 1$ and the lower for $j = 2$), and

$$c_n(\mu) = \frac{(-1)^n}{\gamma_j^3}\left(1 - \mu + \frac{\mu\gamma_j^{n+1}}{(1+\gamma_j)^{n+1}}\right)$$

for $j = 3$.

The linearization around the equilibrium point is given by the second order terms (linear terms must vanish) of the Hamiltonian, which, after some rearranging, takes the form

$$\begin{aligned} H_2 = {} & \tfrac{1}{2}(p_x^2+p_y^2)+yp_x-xp_y \\ & -c_2x^2+\tfrac{1}{2}c_2y^2+\tfrac{1}{2}p_z^2+\tfrac{1}{2}c_2z^2. \end{aligned} \qquad \text{(B–8)}$$

It is not difficult to derive intervals for the values of c_2 when $\mu \in [0, \frac{1}{2}]$ (see Figure 4). Since $c_2 > 0$ (for the three collinear points), the vertical direction is an harmonic oscillator with frequency $\omega_2 = \sqrt{c_2}$. In what follows, we will focus on the planar directions, i.e.,

$$H_2 = \frac{1}{2}\left(p_x^2 + p_y^2\right) + yp_x - xp_y - c_2x^2 + \frac{c_2}{2}y^2, \qquad \text{(B–9)}$$

where, for simplicity, we keep the name H_2 for the Hamiltonian.

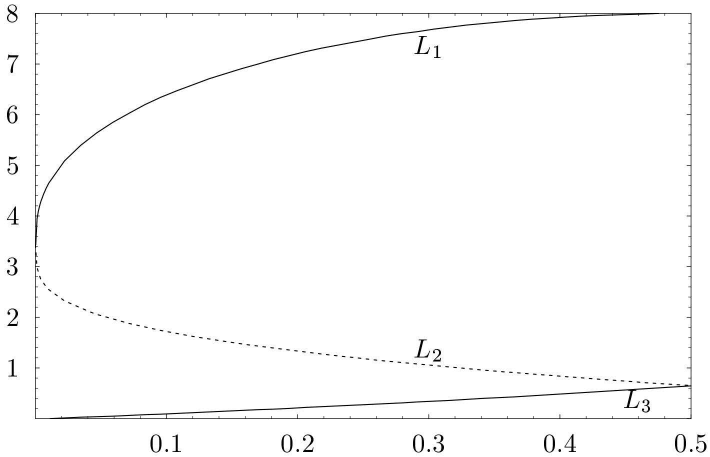

FIGURE 4. Values of $c_2(\mu)$, with $\mu \in [0, \frac{1}{2}]$.

We now proceed as in Section B.1. Define

$$M = J\,\mathrm{Hess}(H_2) = \begin{pmatrix} 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ 2c_2 & 0 & 0 & 1 \\ 0 & -c_2 & -1 & 0 \end{pmatrix}. \qquad \text{(B–10)}$$

The characteristic polynomial of M is $p(\lambda) = \lambda^4 + (2 - c_2)\lambda^2 + (1 + c_2 - 2c_2^2)$. Setting $\eta = \lambda^2$, the roots of $p(\lambda) = 0$ are

$$\begin{aligned} \eta_1 &= \frac{c_2 - 2 - \sqrt{9c_2^2 - 8c_2}}{2}, \\ \eta_2 &= \frac{c_2 - 2 + \sqrt{9c_2^2 - 8c_2}}{2}. \end{aligned}$$

Since $\mu > 0$, we obtain $c_2 > 1$, which forces $\eta_1 < 0$ and $\eta_2 > 0$. This shows that the equilibrium point is a centre $\times$ centre $\times$ saddle. Thus, define ω_1 as $\sqrt{-\eta_1}$ and λ_1 as $\sqrt{\eta_2}$. For the moment, we do not specify the sign taken for each value (this will be discussed later on).

Now, as we did in the previous section, we want to find a symplectic linear change of variables casting (B–9) into its real normal form and, hence, we will look for the eigenvectors of matrix (B–10). As usual, we will take advantage of the special form of this matrix: if we denote by M_λ the matrix $M - \lambda I_4$, then

$$M_\lambda = \begin{pmatrix} A_\lambda & I_2 \\ B & A_\lambda \end{pmatrix}$$

with

$$A_\lambda = \begin{pmatrix} -\lambda & 1 \\ -1 & -\lambda \end{pmatrix}, \quad B = \begin{pmatrix} 2c_2 & 0 \\ 0 & -c_2 \end{pmatrix}.$$

The kernel of M_λ can be found using the same tricks as in the previous section: denoting as $(w_1^T, w_2^T)^T$ the elements of the kernel, we start by solving

$$(B - A^2)w_1 = 0$$

and then $w_2 = -Aw_1$. Thus, the eigenvectors of M are

$$(2\lambda,\ \lambda^2-2c_2-1,\ \lambda^2+2c_2+1,\ \lambda^3+(1-2c_2)\lambda)^T,$$

where λ denotes the eigenvalues.

We first consider the eigenvectors related to ω_1. From $p(\lambda) = 0$, we conclude that ω_1 satisfies

$$\omega_1^4 - (2 - c_2)\omega_1^2 + (1 + c_2 - 2c_2^2) = 0.$$

We also apply $\lambda = \sqrt{-1}\,\omega_1$ to the expression of the eigenvector and, separating real and imaginary parts as $u_{\omega_1} + \sqrt{-1}\,v_{\omega_1}$, we obtain

$$\begin{aligned} u_{\omega_1} &= (0,\ -\omega_1^2-2c_2-1,\ -\omega_1^2+2c_2+1,\ 0)^T, \\ v_{\omega_1} &= (2\omega_1,\ 0,\ 0,\ -\omega_1^3+(1-2c_2)\omega_1)^T. \end{aligned}$$

Now consider the eigenvalues related to $\pm\lambda_1$,

$$u_{+\lambda_1} = (2\lambda,\ \lambda^2-2c_2-1,\ \lambda^2+2c_2+1,\ \lambda^3+(1-2c_2)\lambda)^T$$

and

$$\begin{aligned} v_{-\lambda_1} = (-2\lambda,\ \lambda^2-2c_2-1,\ \lambda^2+2c_2+1, \\ -\lambda^3-(1-2c_2)\lambda)^T. \end{aligned}$$

We consider, initially, the change of variables $C = (u_{+\lambda_1}, u_{\omega_1}, v_{-\lambda_1}, v_{\omega_1})$. To know whether this matrix is symplectic or not, we check $C^T JC = J$. A tedious computation shows that

$$C^T JC = \begin{pmatrix} 0 & D \\ -D & 0 \end{pmatrix}, \quad D = \begin{pmatrix} d_{\lambda_1} & 0 \\ 0 & d_{\omega_1} \end{pmatrix}.$$

This implies that we need to apply some scaling on the columns of C in order to have a symplectic change. The scaling is given by the factors

$$\begin{aligned} d_{\lambda_1} &= 2\lambda_1\big((4 + 3c_2)\lambda_1^2 + 4 + 5c_2 - 6c_2^2\big), \\ d_{\omega_1} &= \omega_1\big((4 + 3c_2)\omega_1^2 - 4 - 5c_2 + 6c_2^2\big). \end{aligned}$$

Thus, we define $s_1 = \sqrt{d_{\lambda_1}}$ and $s_2 = \sqrt{d_{\omega_1}}$. Since we want the change to be real, we have to demand that $d_{\lambda_1} > 0$ and $d_{\omega_1} > 0$. It is not difficult to check that this condition is satisfied for $0 < \mu \le \frac{1}{2}$ in all the points $L_{1,2,3}$, if $\lambda_1 > 0$ and $\omega_1 > 0$.

To obtain the final change, we have to take into account the vertical direction (z, p_z): to put it into real normal form we use the substitution

$$z \mapsto \frac{1}{\sqrt{\omega_2}}\, z, \quad p_z \mapsto \sqrt{\omega_2}\, p_z.$$

This implies that the final change is given by the symplectic matrix C given below.

$$C = \begin{pmatrix} \dfrac{2\lambda}{s_1} & 0 & 0 & \dfrac{-2\lambda}{s_1} & \dfrac{2\omega_1}{s_2} & 0 \\ \dfrac{\lambda^2-2c_2-1}{s_1} & \dfrac{-\omega_1^2-2c_2-1}{s_2} & 0 & \dfrac{\lambda^2-2c_2-1}{s_1} & 0 & 0 \\ 0 & 0 & \dfrac{1}{\sqrt{\omega_2}} & 0 & 0 & 0 \\ \dfrac{\lambda^2+2c_2+1}{s_1} & \dfrac{-\omega_1^2+2c_2+1}{s_2} & 0 & \dfrac{\lambda^2+2c_2+1}{s_1} & 0 & 0 \\ \dfrac{\lambda^3+(1-2c_2)\lambda}{s_1} & 0 & 0 & \dfrac{-\lambda^3-(1-2c_2)\lambda}{s_1} & \dfrac{-\omega_1^3+(1-2c_2)\omega_1}{s_2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \sqrt{\omega_2} \end{pmatrix}$$

Matrix for the final change of coordinates. The order of the variables is (x, y, z, p_x, p_y, p_z). (In the software the order is (x, p_x, y, p_y, z, p_z) instead.)

Finally, to produce the change that brings (B–8) into its complex normal form, we compose C with the same complexification as in the previous section.

ACKNOWLEDGEMENTS

The techniques expounded here have been learned or developed together with some colleagues from the Dynamical Systems Group at Barcelona (http://www.maia.ub.es/dsg). I am particularly indebted with Carles Simó for the fruitful discussions we have had about this subject. My first algebraic manipulator [Díez et al. 1991] started as an undergraduate project at the University of Barcelona.

I thank R. de la Llave for suggesting this work and for his comments to a preliminary version of this paper. I also thank J. Villanueva for his remarks and R. Broucke, A. Giorgilli and J. Henrard for some bibliographical comments. I also want to acknowledge the hospitality of TICAM (University of Texas at Austin), where this project started.

ELECTRONIC AVAILABILITY

The source code for the implementation of the algorithms discussed in this paper can be retrieved from http://www.maia.ub.es/dsg, in the section Preprints and Publications.

REFERENCES

[Abraham and Marsden 1978] R. Abraham and J. E. Marsden, *Foundations of mechanics*, 2nd ed., Benjamin/Cummings, Reading, MA, 1978.

[Arnol'd 1964] V. I. Arnol'd, "Instability of dynamical systems with several degrees of freedom", *Dokl. Akad. Nauk SSSR* **156** (1964), 9–12. In Russian; translated in *Sov. Math. Dokl.* **5**:3 (1964), 581–585.

[Arnol'd 1978] V. I. Arnol'd, *Mathematical methods of classical mechanics*, Graduate Texts in Math. **60**, Springer, New York, 1978. Second edition, 1989.

[Arnol'd and Avez 1968] V. I. Arnol'd and A. Avez, *Ergodic problems of classical mechanics*, W. A. Benjamin, Inc., New York, 1968.

[Arnol'd et al. 1988] V. I. Arnol'd, V. V. Kozlov, and A. I. Neĭshtadt, *Dynamical systems III*, Encyclopaedia of mathematical sciences **3**, Springer, Berlin, 1988. Second edition, 1993.

[Broucke 1989] R. Broucke, "A Fortran-based Poisson series processor and its applications in celestial mechanics", *Celestial Mechanics* **45** (1989), 255–265.

[Broucke and Garthwaite 1969] R. Broucke and K. Garthwaite, "A programming system for analytical series expansions on a computer", *Celestial Mechanics* **1** (1969), 271–284.

[Brumberg et al. 1989] V. A. Brumberg, S. V. Tarasevich, and N. N. Vasiliev, "Specialized celestial mechanics systems for symbolic manipulation", *Celestial Mechanics* **45** (1989), 149–162.

[Bruno 1989] A. D. Bruno, *Local methods in nonlinear differential equations*, Springer Series in Soviet Mathematics, Springer, Berlin, 1989.

[Celletti and Chierchia 1988] A. Celletti and L. Chierchia, "Construction of analytic KAM surfaces and effective stability bounds", *Comm. Math. Phys.* **118**:1 (1988), 119–161.

[Celletti and Giorgilli 1991] A. Celletti and A. Giorgilli, "On the stability of the Lagrangian points in the spatial restricted three body problem", *Celestial Mechanics* **50** (1991), 31–58.

[Díez et al. 1991] C. Díez, A. Jorba, and C. Simó, "A dynamical equivalent to the equilateral libration points of the Earth-Moon system", *Celestial Mechanics* **50** (1991), 13–29.

[Giorgilli 1979] A. Giorgilli, "A computer program for integrals of motion", *Comp. Phys. Comm.* **16** (1979), 331–343.

[Giorgilli et al. 1989] A. Giorgilli, A. Delshams, E. Fontich, L. Galgani, and C. Simó, "Effective stability for a Hamiltonian system near an elliptic equilibrium point, with an application to the restricted three-body problem", *J. Differential Equations* **77**:1 (1989), 167–198.

[Gómez et al. 1985] G. Gómez, J. Llibre, R. Martínez, and C. Simó, "Station keeping of libration point orbits", Final Report, ESOC Contract 5684/83/D/JS (SC), 1985.

[Gómez et al. 1987] G. Gómez, J. Llibre, R. Martínez, and C. Simó, "Study on orbits near the triangular libration points in the perturbed restricted three-body problem", Final Report, ESOC Contract 6139/84/D/JS (SC), 1987.

[Gómez et al. 1991a] G. Gómez, A. Jorba, J. Masdemont, and C. Simó, "A dynamical systems approach for the analysis of the SOHO mission", pp. 449–454 in *Proceedings of the Third International Symposium on Spacecraft Flight Dynamics*, ESA Publications Division, ESTEC, Noordwijk, Holland, 1991.

[Gómez et al. 1991b] G. Gómez, A. Jorba, J. Masdemont, and C. Simó, "A quasiperiodic solution as a substitute of L_4 in the Earth-Moon system", pp. 35–41 in *Proceedings of the Third International Symposium on Spacecraft Flight Dynamics*, ESA Publications Division, ESTEC, Noordwijk, Holland, 1991.

[Gómez et al. 1991c] G. Gómez, A. Jorba, J. Masdemont, and C. Simó, "Study of refinement of semi-analytical halo orbit theory", Final Report, ESOC Contract 8625/89/D/MD (SC), 1991.

[Gómez et al. 1993a] G. Gómez, A. Jorba, J. Masdemont, and C. Simó, "Study of Poincaré maps for orbits near Lagrangian points", Final Report, ESOC Contract 9711/91/D/IM (SC), 1993.

[Gómez et al. 1993b] G. Gómez, A. Jorba, J. Masdemont, and C. Simó, "Study of the transfer between halo orbits in the solar system", *Adv. Astronautical Sci.* **84** (1993), 623–637.

[Gómez et al. 1993c] G. Gómez, A. Jorba, J. Masdemont, and C. Simó, "Study of the transfer from the Earth to a halo orbit around the equilibrium point L_1", *Celestial Mechanics* **56** (1993), 541–562.

[Gómez et al. 1997] G. Gómez, J. Masdemont, and C. Simó, "Lissajous orbits around Halo orbits", pp. 117–134 in *Spaceflight Mechanics* 1997 (Huntsville, AL, 1997), edited by K. C. Howell et al., Advances in the Astronautical Sciences **95**, American Astronautical Society and Univelt, San Diego, 1997.

[Henrard 1989] J. Henrard, "A survey of Poisson series processors", *Celestial Mechanics* **45** (1989), 245–253.

[Jorba and Llave $\geq$ 1999] A. Jorba and R. de la Llave, "Regularity properties of center manifolds and applications", In preparation.

[Jorba and Masdemont 1998] A. Jorba and J. Masdemont, "Dynamics in the center manifold of the collinear points of the restricted three body problem", preprint, 1998. To appear in *Physica D*.

[Jorba and Masdemont 1999] A. Jorba and J. Masdemont, "Nonlinear dynamics in an extended neighbourhood of the translunar equilibrium point", pp. 430–434 in *Hamiltonian systems with three or more degrees of freedom* (S'Agaro, Spain, 1995), edited by C. Simó, NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci. **533**, Kluwer, New York, 1999.

[Jorba and Simó 1994] A. Jorba and C. Simó, "Effective stability for periodically perturbed Hamiltonian systems", pp. 245–252 in *Hamiltonian mechanics*, edited by J. Seimenis, Plenum Press, New York, 1994.

[Jorba and Villanueva 1997a] À. Jorba and J. Villanueva, "On the normal behaviour of partially elliptic lower-dimensional tori of Hamiltonian systems", *Nonlinearity* **10**:4 (1997), 783–822.

[Jorba and Villanueva 1997b] À. Jorba and J. Villanueva, "On the persistence of lower-dimensional invariant tori under quasi-periodic perturbations", *J. Nonlinear Sci.* **7**:5 (1997), 427–473.

[Jorba and Villanueva 1998] À. Jorba and J. Villanueva, "Numerical computation of normal forms around some periodic orbits of the restricted three-body problem", *Phys. D* **114**:3-4 (1998), 197–229.

[Jorba and Villanueva 1999] A. Jorba and J. Villanueva, "Effective stability around periodic orbits of the spatial RTBP", pp. 628–632 in *Hamiltonian systems with three or more degrees of freedom* (S'Agaro, Spain, 1995), edited by C. Simó, NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci. **533**, Kluwer, New York, 1999.

[Kernighan and Ritchie 1988] B. Kernighan and D. Ritchie, *The C programming language*, 2nd ed., Prentice-Hall, Englewood Cliffs, NJ, 1988.

[Laskar 1990] J. Laskar, "Manipulation des series", in *Modern methods in celestial mechanics* (Gif-sur-Yvette, 1989), edited by D. Benest and C. Froeschlé, Editions Frontières, Paris, 1990. Second edition, 1992.

[Llave and Rana 1990] R. de la Llave and D. Rana, "Accurate strategies for small divisor problems", *Bull. Amer. Math. Soc.* (*N.S.*) **22**:1 (1990), 85–90.

[Llave et al. 1986] R. de la Llave, J. M. Marco, and R. Moriyón, "Canonical perturbation theory of Anosov systems and regularity results for the Livšic cohomology equation", *Ann. of Math.* (2) **123**:3 (1986), 537–611.

[Marchal 1980] C. Marchal, "The quasi integrals", *Celestial Mechanics* **21** (1980), 183–191.

[Meyer and Hall 1992] K. R. Meyer and G. R. Hall, *Introduction to Hamiltonian dynamical systems and the N-body problem*, Springer, New York, 1992.

[Meyer and Schmidt 1986] K. R. Meyer and D. S. Schmidt, "The stability of the Lagrange triangular point and a theorem of Arnol'd", *J. Differential Equations* **62**:2 (1986), 222–236.

[Raines and Uzer 1992] P. E. Raines and T. A. Uzer, "Computing normal forms of nonseparable Hamiltonians by symbolic manipulation", *Comp. Phys. Comm.* **70** (1992), 569–578.

[Rand and Armbruster 1987] R. H. Rand and D. Armbruster, *Perturbation methods, bifurcation theory* *and computer algebra*, Applied Mathematical Sciences **65**, Springer, New York, 1987.

[Richardson 1980] D. L. Richardson, "A note on a Lagrangian formulation for motion about the collinear points", *Celestial Mechanics* **22** (1980), 231–236.

[Ricklefs et al. 1983] R. Ricklefs, W. Jefferys, and R. Broucke, "A general precompiler for algebraic manipulation", *Celestial Mechanics* **29** (1983), 179–190.

[Schelter 1991] W. Schelter, "SCC: An extension of ANSI C", 1991. See ftp://ftp.math.utexas.edu/pub/maxima/scc-3.tgz.

[Siegel and Moser 1971] C. L. Siegel and J. K. Moser, *Lectures on celestial mechanics*, Die Grundlehren der mathematischen Wissenschaften **187**, Springer, New York, 1971. Reprinted 1995.

[Sijbrand 1985] J. Sijbrand, "Properties of center manifolds", *Trans. Amer. Math. Soc.* **289**:2 (1985), 431–469.

[Simó 1989] C. Simó, "Estabilitat de sistemes hamiltonians", *Mem. de la Real Acad. de Ciencias y Artes de Barcelona* **48**:7 (1989).

[Simó 1990] C. Simó, "Analytical and numerical computation of invariant manifolds", pp. 285–330 in *Modern methods in celestial mechanics* (Gif-sur-Yvette, 1989), edited by D. Benest and C. Froeschlé, Editions Frontières, Paris, 1990.

[Simó 1994] C. Simó, "Averaging under fast quasiperiodic forcing", pp. 13–34 in *Hamiltonian mechanics* (Toruń, 1993), edited by J. Seimenis, NATO Adv. Sci. Inst. Ser. B Phys. **331**, Plenum, New York, 1994.

[Simó 1996] C. Simó, "Effective computations in Hamiltonian dynamics", pp. 1–23 in *Mécanique céleste: Cent ans après les Méthodes Nouvelles de H. Poincaré*, Soc. Math. France, Paris, 1996.

[Simó 1998] C. Simó, "Effective computations in celestial mechanics and astrodynamics", pp. 55–102 in *Modern methods of analytical mechanics and their applications* (Udine, Italy, 1997), edited by V. V. Rumyantsev and A. V. Karapetyan, CISM Courses and Lectures **387**, Springer, 1998.

[Simó et al. 1995] C. Simó, G. Gómez, A. Jorba, and J. Masdemont, "The bicircular model near the triangular libration points of the RTBP", pp. 343–370 in *From Newton to Chaos: modern techniques for understanding and coping with chaos in n-body dynamical systems*, edited by A. E. Roy and B. A. Steves, NATO Adv. Sci. Inst. Ser. B Phys. **336**, Plenum Press, New York, 1995.

[Stroustrup 1992] B. Stroustrup, *The C++ programming language*, 2nd ed., Addison Wesley, Reading, MA, 1992.

[Szebehely 1967] V. Szebehely, *Theory of orbits: the restricted problem of three bodies*, Academic Press, New York, 1967.

[Vanderbauwhede 1989] A. Vanderbauwhede, "Centre manifolds, normal forms and elementary bifurcations", pp. 89–169 in *Dynamics reported*, vol. 2, Wiley, Chichester, 1989.

Àngel Jorba, Departament de Matemàtica Aplicada i Anàlisi, Universitat de Barcelona, Gran Via 585, 08007 Barcelona, Spain (angel@maia.ub.es)

Received January 12, 1998; accepted in revised form July 31, 1998