Efficient computational approaches to obtain periodic orbits in Hamiltonian systems: application to the motion of a lunar orbiter

Ángeles Dena[1] · **Alberto Abad**[2] · **Roberto Barrio**[3]

Abstract

In this paper, we study the problem of computing periodic orbits of Hamiltonian systems providing large families of such orbits. Periodic orbits constitute one of the most important invariants of a system, and this paper provides a comprehensive analysis of two efficient computational approaches for Hamiltonian systems. First, a new version of the grid search method, applied to problems with three degrees of freedom, has been considered to find, systematically, symmetric periodic orbits. To obtain non-symmetric periodic orbits, we use a modification of an optimization method based on an evolutionary strategy. Both methods require a great computational effort to find a big number of periodic orbits, and we apply parallelization tools to reduce the CPU time. Finally, we present a strategy to provide initial conditions of the periodic orbits with arbitrary precision. We apply all these algorithms to the problem of the motion of the lunar orbiter referred to the rotating reference frame of the Moon. The periodic orbits of this problem are very useful from the space engineering point of view because they provide low-cost orbits.

Keywords Lunar satellite orbits, Periodic orbits (p.o.), Symmetric periodic orbits (s.p.o), Paralletization algorithms, Grid search algorithm, Evolutionary strategy.

✉ Ángeles Dena
adena@unizar.es

Alberto Abad
abad@unizar.es

Roberto Barrio
rbarrio@unizar.es

[1] Centro Universitario de la Defensa Zaragoza, Zaragoza, Spain

[2] Grupo de Mecánica Espacial, University of Zaragoza, Zaragoza, Spain

[3] Departmento Matemática Aplicada, University of Zaragoza, Zaragoza, Spain

1 Introduction

Periodic orbits (p.o.) are one of the most important objects in the study of dynamical systems. Poincaré (1892) already points out that p.o. form the *skeleton* of any dynamical system. The problem is that finding p.o. is not an easy task. Shooting, multiple shooting and collocation methods have been applied to find p.o. as the solution of a two-point boundary value problem (Farantos 1998; Tadi 2005). Other important methods are based on the continuation theory, that is, varying a parameter of the dynamical system to find a family of p.o. from a particular one already known (Doedel 1981; Lara and Pelaez 2002; Dhooge et al. 2003; Wulff and Schebesch 2006). Grid search method (Markellos et al. 1974; Kazantzis and Goudas 1975; Russell 2006; Tsirogiannis et al. 2009; Barrio and Blesa 2009) becomes a powerful method when the dynamical system has two degrees of freedom if we restrain the problem to find symmetric periodic orbits (s.p.o.) as it provides directly a large number of p.o. Also, when we are interested in very unstable periodic orbits, the variational methods (Contopoulos and Harsoula 2010) provide a powerful approach. And finally, another family of methods is based on evolution strategies (Mauger et al. 2010).

The main objective of this paper is not to present a completely new method to compute p.o., but to show how generalizations of previously existing ones permit to provide with extremely powerful techniques to locate p.o. Moreover, we show how the combination of several different techniques allows us to find a large number of p.o. satisfying different requests, as symmetry conditions or not, or very high precision. Therefore, as a result, all these techniques provide with useful methods to give a detailed study of Hamiltonian systems.

The motion of a lunar orbiter (a satellite in orbit around the Moon) constitutes a very involved dynamical system. In fact, the Keplerian description of the motion is only a poor approximation, to which we must add, among others, the effects of the Moon potential (Moon is not a sphere) and the gravitational attraction of the Earth. Considering the Earth, the motion of the lunar orbiter becomes a three-body problem that is one of the most useful examples of the complexity of certain dynamical systems.

Periodic orbits for the lunar orbiter are not only a good way to explore this system, but they are very useful themselves in order to design space missions to the Moon (Weilian 2004; Folte and Quinn 2006). Space engineers need to know orbits that maintain their requirements for a long period of time, and the p.o., obtained in the rotating reference frame of the Moon, are very good examples of these kinds of orbits. The problem of searching p.o. for lunar orbiters has been extensively considered (Elipe and Lara 2003; Weilian 2004; Folte and Quinn 2006; Abad et al. 2009), but most of the efforts have been applied in finding frozen orbits. Frozen orbits are orbits in which eccentricity and argument of perigee remain stationary on average. Frozen orbits correspond to equilibria in an averaged form of the zonal problem of the satellite and are almost periodic solutions of the full (non-averaged) problem.

In this paper, we analyze the problem from two different points of view. We explore the possibility to obtain directly a massive number of p.o., not only frozen orbits. Firstly, we will apply a grid search method to obtain s.p.o., and secondly we will use an evolution strategy method to obtain non-symmetric p.o. In both cases, the computational effort to compute the orbits and the characteristics of the developed algorithms make possible to parallelize the code, which open the way to more detailed studies. The efficiency of the algorithms is also studied.

The grid search method has been extensively used to compute s.p.o. in two degrees of freedom (2DOF) Hamiltonian systems (Markellos et al. 1974; Tsirogiannis et al. 2009; Barrio and Blesa 2009). Grid search method takes advantage of a mirror configuration (Roy and

Ovenden 1955) showed by these systems due to certain symmetry conditions (Barrio and Blesa 2009). Computing s.p.o. in three degrees of freedom (3DOF) Hamiltonian systems introduces new methodological and computational difficulties to the process. For instance, in 2DOF the final search is just a one-dimensional root-finding process, but in 3DOF we will have at least a two-dimensional root-finding process, which of course is much more involved and needs a much more careful approach. In fact, there are only very few papers (Russell 2006; Tsirogiannis et al. 2009) applying the 3DOF version of the grid search method. In particular, Russell (2006) studies the motion of an orbiter around Europa, satellite of Jupiter. There are two important differences with our work, the first one is about the model, we consider the effect of the non-sphericity of the Moon, the second one is about the algorithm and the tools that we use to solve it.

To integrate the ordinary differential equations (ODEs) that appear in the problem, we use a new free software (Abad et al. 2012) named TIDES[1] (Taylor series Integrator for Differential EquationS), which applies the Taylor series method to integrate ODEs. The use of Taylor series method permits to reach a very large precision in integrations of ODEs. Moreover, compared with other integrators, TIDES shows similar results for low precisions but better behavior for high-precision calculations, and we may integrate ODEs with multiple precision arithmetic with a minimum effort. But the precision is not the main characteristic when we apply TIDES to compute p.o.; we have two more possibilities: the computation of partial derivatives of the variables with respect to the initial conditions without the formulation of the variational equations and an easy way to compute *events*, or points where the variable reaches certain values (zeros, local extrema, etc.) along the integration. We remark that any other good variable stepsize numerical method for ODEs, like the `dopri853` Runge–Kutta method (Hairer et al. 1993) or any similar RK or multi-step method, is suitable for our purposes.

The lunar orbiter problem presents the necessary symmetries to apply all the previous methods, but sometimes, for instance, if we include tesseral harmonics in the potential of the Moon, and for any other problems, those symmetries do not appear. To find p.o. in these cases, we have to use a second method, introduced in Abad and Elipe (2014) to compute p.o. around the gravitational influence of a solid circular ring. This method is an adaptation of an evolution strategy (Beyer and Schwefel 2002) that belongs to the modern optimization methods: genetic algorithms, evolution strategies, evolutionary programming, simulated annealing, Gaussian adaptation, swarm intelligence, etc. Different problems require different techniques adapted to it; for instance, the simulated annealing algorithm has been recently used with success in Mauger et al. (2010) to find periodic orbits of multi-electron atomic systems, or the particle swarm optimization method (Parsopoulos and Vrahatis 2004) used to compute periodic orbits for three-dimensional galactic potentials (Skokos et al. 2005). In this paper, we continue in this line by proposing a new method in order to increase the tools to handle the tricky task of finding periodic orbits.

Both methods, the grid search and the evolution strategy methods, are easily parallelizable. Therefore, due to the large computation effort we need for a detailed analysis of the problems, we focus on the use of parallel versions of the algorithms. We remark that, obviously, these methods are obtained as generalizations of previous ones but both of them, in spite of their simpleness, provide with quite effective and powerful methods.

Another situation when an extra algorithm is needed is when we are interested in initial conditions of a system with a very high precision, as occurs, by instance in case of orbits inside a chaotic region or for very long-time simulations. Using the data obtained from the previous

[1] http://sourceforge.net/projects/tidesodes/

algorithms, we show in this paper how to obtain (Abad et al. 2011) arbitrary precision initial conditions based on a Newton method, an arbitrary precision ODE solver (TIDES) and the singular value decomposition (SVD).

The present paper is organized as follows: Section 2 presents the formulation of the differential equations of the lunar problem. Section 3 explains the algorithm used to apply the 3DOF grid search method to locate families of periodic orbits. In Sect. 4, we present the modification of the evolutionary strategy to compute generic p.o. Section 5 explains how to refine the initial conditions of the periodic orbits to obtain arbitrary precision data. Section 6 shows the way in which we parallelize the algorithms and their application to the lunar problem. Some figures are presented to illustrate the results and the efficiency of the parallelization. Finally, Sect. 7 presents the conclusions of this work.

2 Dynamical model of a lunar orbiter

We consider the motion of an orbiter about the Moon under the Hill hypothesis (Hill 1878; Lara and Palacián 2009), that is the Moon is in circular orbit about the Earth, with radius r_e, in the equatorial plane of the Earth, and the orbit is synchronized with the rotation of the Moon.

To work with this hypothesis, instead of using the inertial reference frame, it is better to consider a rotating reference frame $Oxyz$, centered on the Moon and such that the plane Oxy coincides with the Moon's equator and the Ox axis continuously points toward the Earth. Let $\boldsymbol{\omega} = (0, 0, \omega)$ be the angular velocity vector of the rotation of the Moon; thus, the Hamiltonian of the problem is

$$\mathcal{H}(\boldsymbol{x}, \boldsymbol{p}) = \frac{1}{2}\,\boldsymbol{p}^2 - \boldsymbol{\omega} \cdot (\boldsymbol{x} \times \boldsymbol{p}) + V(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{x} = (x, y, z)$, $\boldsymbol{p} = (p_x, p_y, p_z)$ represents the position and the absolute velocity of the orbiter at the rotating reference frame, and V the potential function.

The potential $V = V_\mathrm{k} + V_\mathrm{e} + V_\mathrm{m}$ is the addition of the Keplerian potential V_k, that produce Keplerian orbits, and the perturbations over the Keplerian motion due to several effects like the gravitational attraction of the Earth over the lunar orbiter, V_e, and the non-spherical form of the Moon, V_m. The next three paragraphs show the three terms expressed at the rotating frame of the Moon.

The Keplerian potential is

$$V_\mathrm{k} = -\frac{\mu_\mathrm{m}}{r} = -\frac{\mu_\mathrm{m}}{\sqrt{x^2 + y^2 + z^2}}, \tag{2}$$

where $\mu_\mathrm{m} = \mathcal{G}\, m_\mathrm{m}$ is the product of the gravitational constant and the mass of the Moon.

The gravitational potential created by the Earth is

$$V_\mathrm{e} = -\mu_\mathrm{e}\left(\frac{1}{\|\,\boldsymbol{x}_\mathrm{e} - \boldsymbol{x}\,\|} - \frac{\boldsymbol{x}_\mathrm{e} \cdot \boldsymbol{x}}{\|\,\boldsymbol{x}_\mathrm{e}\,\|^3}\right) = -\mu_\mathrm{e}\left(\frac{1}{\sqrt{(r_\mathrm{e} - x)^2 + y^2 + z^2}} - \frac{x}{r_\mathrm{e}^2}\right), \tag{3}$$

where $\mu_\mathrm{e} = \mathcal{G} m_\mathrm{e}$ is the product of the gravitational constant by the mass of the Earth, and $\boldsymbol{x}_\mathrm{e} = (r_\mathrm{e}, 0, 0)$ the position of the Earth with respect to the Moon expressed at the rotating frame.

In this problem, we take only into account the main problem of the satellite, i.e., the Moon potential produced by the J_2 zonal harmonic, that consider the Moon an homogeneous ellipsoid instead an sphere. Thus, the potential is

$$V_{\mathrm{m}} = \frac{\mu_{\mathrm{m}}\, r_{\mathrm{m}}^2\, J_2}{2\, r^3} \left(\frac{3\, z^2}{r^2} - 1 \right) = \frac{\mu_{\mathrm{m}}\, r_{\mathrm{m}}^2\, J_2}{2\, (x^2 + y^2 + z^2)^{3/2}} \left(\frac{3\, z^2}{(x^2 + y^2 + z^2)} - 1 \right), \tag{4}$$

with r_{m} the equatorial radius of the Moon.

The evolution of the lunar orbiter is given by the ordinary differential equations obtained by applying the Hamilton's equations to the Hamiltonian (1)

$$\begin{array}{ll} \dot{x} = p_x + \omega y, & \dot{p}_x = \omega p_y - V_x, \\ \dot{y} = p_y - \omega x, & \dot{p}_y = -\omega p_x - V_y, \\ \dot{z} = p_z, & \dot{p}_z = -V_z, \end{array} \tag{5}$$

where V_x, V_y, V_z denote the partial derivatives of the potential with respect to the position vector, and $\dot{\boldsymbol{x}} = (\dot{x}, \dot{y}, \dot{z}) = (p_x + \omega\, y,\ p_y - \omega\, y,\ p_z)$ is the relative velocity. Then, the equations (5) become

$$\begin{aligned} \ddot{x} &= \omega^2\, x + 2\omega\, \dot{y} - V_x, \\ \ddot{y} &= \omega^2\, y - 2\omega\, \dot{x} - V_y, \\ \ddot{z} &= -V_z. \end{aligned} \tag{6}$$

The motion admits a constant named Jacobi constant defined by the expression

$$J = \omega^2 \left(x^2 + y^2 \right) - 2V - \boldsymbol{x} \cdot \boldsymbol{x}. \tag{7}$$

that substitutes, in the rotating case, to the classical energy integral.

3 Locating families of periodic orbits: the grid search method

One of the main important applications of the location of a great number of p.o. is to describe the different regions in the parametric phase space of a system. This is one of the most powerful ways to give a global parametric description of a dynamical system.

When the differential system depends on a parameter $\mathcal{P}$, we want to know whether a periodic solution may be continued, that is, if the p.o. is isolated or not when the parameter is changed. A theorem due to Meyer et al. (2009) permits to characterize such a situation: *an elementary periodic solution in a Hamiltonian system with a non-degenerate integral can be continued*.

Recall that a periodic solution is called elementary if the monodromy matrix has just two eigenvalues equal to one (in case of an autonomous Hamiltonian system). Therefore, for autonomous Hamiltonian systems this is the case and the periodic orbits appear in *families*. Note that if we have a non-elementary periodic solution the periodic orbit may appear or disappear due to a saddle-node bifurcation. A family of periodic orbits is represented by a smooth one-parameter continuous curve (the *characteristic curve*) in the space of initial conditions or parameters. This is the basic principle of one of the most well-known methods to locate families of periodic orbits, the continuation method. This method, once you have the initial conditions of one p.o. of the family, tries to continue on the curve of the family by changing the parameter values. This method has proved its power in numerous studies (Krauskopf et al. 2007), and there are some well-known free software packages such as AUTO (Doedel 1981) and MATCONT (Dhooge et al. 2003). The main problem is that you need the initial data, i.e., some initial conditions of periodic orbits, and if you have no initial conditions in all the parametric regions, the continuation method may ignore p.o. in important regions. Therefore, in this paper we focus on a different method to locate families of periodic orbits. This method is based on a "brute-force" approach, that combined with simplifying assumptions based on symmetry conditions, permits to locate a large number of symmetric periodic orbits in all parametric ranges of a system. Later, this approach may be combined with a continuation method if desired, as there are a large number of p.o. Note that this method is based on a very simple idea, just to take into account symmetry conditions to locate in a grid of the parametric variables phase-space initial conditions of the p.o. This method was initially proposed for 2DOF Hamiltonian systems (Markellos et al. 1974; Tsirogiannis et al. 2009; Barrio and Blesa 2009) and extended in just few cases to 3DOF. The extension to 3DOF is in some sense "direct," as the main concept is the same. What is much more involved is the multi-dimensional root-finding process and needs a much more careful approach. Therefore, although the method proposed is not, obviously, a scientific breakthrough, it gives a extremely powerful technique to locate a very large number of symmetric p.o. on the parametric variable phase space of a system. As shown at the examples presented below, very few methods in the literature may provide in such a simple way a detailed "skeleton" of periodic orbits of a system.

Let us suppose a 2DOF Hamiltonian system $\mathcal{H}(x, y, \dot{x}, \dot{y})$ that presents some symmetry conditions, for instance, a symmetry with respect to the x-axis. So, if we have a solution $\{x(t), y(t), \dot{x}(t), \dot{y}(t)\}$, then the expression $\{x(-t), -y(-t), -\dot{x}(-t), \dot{y}(-t)\}$ is also a solution. In that case, if an orbit starts at the x-axis perpendicular to it $(x_0, 0, 0, \dot{y}_0)$ and crosses the x-axis again perpendicular, then the orbit is closed and symmetric (s.p.o.). To find these orbits, we check for the first perpendicular cross to the x-axis:

$$y\,(x_0, 0, 0, \dot{y}_0;\ T/2) = \dot{x}\,(x_0, 0, 0, \dot{y}_0;\ T/2) = 0. \tag{8}$$

When these conditions are accomplished, $(x_0, 0, 0, \dot{y}_0)$ represent the initial conditions of a s.p.o. of period T.

The first step in the grid search method is to define a two-dimensional regular mesh that represents the energy level h (or any other parameter $\mathcal{P}$) versus the initial coordinate x. Each point, (x_i, h_i), $i = 0, \ldots, N$, of the $N \times N$ mesh, represents the initial conditions $(x_i, 0, 0, \dot{y}_i)$, where $\dot{y}_i$ is obtained from the relation $\mathcal{H}(x_i, 0, 0, \dot{y}_i) = h_i$. To find p.o. of multiplicity m (number of loops in the orbit), we integrate the problem until the orbit cross m times the Ox axis. If we just look for the first cross, we will only obtain the p.o. of multiplicity one, going further we also search for higher-order multiplicities. The second condition of (8) indicates that the s.p.o. are represented by the points of the curves $\dot{x}(x, 0, 0, \dot{y}(x, h)) = 0$ at the window (x, h). To locate such curves, we integrate consecutively from the points of the vertical line of the grid, i.e., the line with h_k fixed (we do the same for horizontal lines). Then, we obtain $\dot{x}_i(x_i, 0, 0, \dot{y}_i(x_i, h_k))$, $i = 0, \ldots N$. To find the roots of $\dot{x}$, we take the intervals $[x_i, x_{i+1}]$ where $\dot{x}_i \dot{x}_{i+1} < 0$. Thus, a root-finding process is necessary, but now it has to be used in combination with a numerical integrator because we have to integrate the differential system for each iteration of the process. The Brent's method (Brent 1971) is a good choice for this step of the grid search method. Once the convergence is reached, we have a set of initial conditions that satisfy the symmetric periodic conditions.

The 3DOF problem presents more kind of different symmetries. In particular, our problem shows two of those symmetries: the planar Oxz symmetry and the axial Ox one. Each symmetry must be handled in a different way. In this paper, we describe only how to handle the planar Oxz symmetry. The axial symmetry, together with a detailed report of the results, is yet in progress.

The Oxz symmetry implies that if $\{x(t), y(t), z(t), \dot{x}(t), \dot{y}(t), \dot{z}(t)\}$ is a solution, then $\{x(-t), -y(-t), z(-t), -\dot{x}(-t), \dot{y}(-t), -\dot{z}(-t)\}$ is also a solution. Thus, we start now the

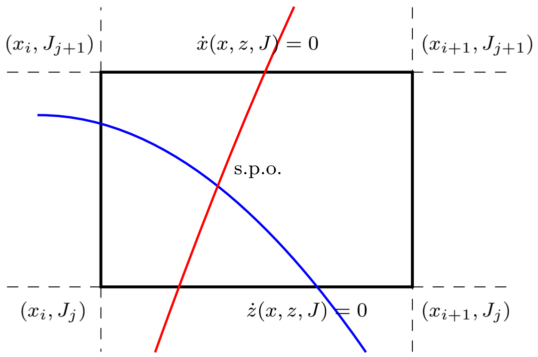

Fig. 1 The intersection of the *curves* inside the *square* of the grid represents a s.p.o

orbit at the Oxz plane and perpendicular to it: $(x_0, 0, z_0, 0, \dot{y}_0, 0)$. The periodicity condition is accomplished when the orbit crosses again the plane perpendicularly. Therefore, the conditions to find an s.p.o. are

$$\begin{aligned} y\,(x_0, 0, z_0, 0, \dot{y}_0, 0;\ T/2) &= \dot{x}\,(x_0, 0, z_0, 0, \dot{y}_0, 0;\ T/2) \\ &= \dot{z}\,(x_0, 0, z_0, 0, \dot{y}_0, 0;\ T/2) = 0. \end{aligned} \tag{9}$$

If we have a point (x_0, z_0, h_0), then the coordinate $\dot{y}_0$ of the initial point is obtained from the equation $\mathcal{H}(x_0, 0, z_0, 0, \dot{y}_0, 0) = h_0$. In our problem, we substitute the energy condition by a condition based on the Jacobi constant, J (7), instead of the energy h. The two-dimensional window (x, h) is sufficient to describe the s.p.o in the 2DOF problem; however, the 3DOF problem requires a three-dimensional grid (x, z, J). To look over the three-dimensional grid, we reduce the problem to many two-dimensional grids by fixing one of the elements of the grid. Fixing z we compute the window (x, J) or fixing the Jacobi constant we compute the window (x, z), and then we repeat the computations with different values of the fixed parameter to obtain a three-dimensional study $N \times N \times N$ as a set of N two-dimensional $N \times N$ grids.

To find orbits of multiplicity m, we integrate the ODE until the orbit cross m times the Oxz plane (this procedure ensures the first condition of periodicity $y = 0$). If a point fulfills simultaneously both conditions $\dot{x}(x, 0, z, 0, \dot{y}(x, z, J), 0) = \dot{z}(x, 0, z, 0, \dot{y}(x, z, J), 0) = 0$, it represents the initial condition of a the s.p.o. The 2DOF case locates the s.p.o. obtaining the cut points of a curve with each line of the grid; however, it is not possible in the 3DOF case because now the solution is not represented by a curve but for the intersection of two curves; then, the algorithm in this case must be completely different. Instead, to explore vertical and horizontal lines of grid points we explore consecutive squares of vertices $(x_i, J_j), (x_{i+1}, J_j), (x_i, J_{j+1}), (x_{i+1}, J_{j+1})$, $i = 0, \dots N-1$, $j = 0, \dots N-1$ (see Fig. 1). In fact, the exploration is done by horizontal lines of squares (j fixed). Taking into account the changes of sign of the functions $\dot{x}, \dot{z}$ at the vertices of the square, we may determine how many times the curves cut the sides of the square. The necessary (not sufficient) condition to have a s.p.o. inside the square is to have at least two cuts of each curve along the perimeter of the square. This condition only assures that both curves cross the square but the intersection of the curves, and, as a consequence, the existence of a s.p.o. inside the square is not guaranteed. When a possible s.p.o. are inside a square, we take the center of the square as the initial condition of an orbit close to a s.p.o. An alternative way to compute s.p.o is the characteristic bisection method (Vrahatis 1988a, b) based on the concept of characteristic polyhedron (CP). This method has been successfully used to locate periodic orbits in two-dimensional mappings (Polymilis et al. 2003). Inside a CP, under suitable assumptions, there exists, at least, one solution of a n-dimensional function because the topological degree of the function relative to the CP takes the value ± 1. In our problem, if the square is a CP, the existence of a root is guaranteed, if not the method gives a procedure to compute a CP inside the square. Obviously, this CP can not be found if no s.p.o exists into the square. Once we have the CP, the bisection method may be used to approximate the s.p.o.

Note that this method can be extended to systems with more than 3DOF, but a detailed analysis of symmetries has to be done. Also, the computational effort grows significatively.

Once we have a list of approximate p.o., we must apply a corrector method to obtain the precise initial condition of the s.p.o. This method, described in Sect. 5, is based on a modified Newton method. The method does not converge in the cases in which the two previous curves have no intersection. To find the list of approximate s.p.o., we need a numerical integrator to integrate the differential equations (6), but to correct the s.p.o. the corrector method needs also the integration of the variational equations, and the process is quite different. For this reason, we separate both processes, and we present here results about the parallelization of the first part of the method.

To compute s.p.o. in the lunar problem, we take the ODEs (6), where we consider the Keplerian part, the Earth perturbation and the non-sphericity of the Moon, then the potential is $V = V_{\mathrm{k}} + V_{\mathrm{e}} + V_{\mathrm{m}}$. We choose the Moon radius as the length unit, and the minute as the time unit. Then, the value of the parameters needful to evaluate the potential is: $r_{\mathrm{m}} = 1$, $J_2 = 0.0002033$, $\mu_{\mathrm{m}} = 0.0033614734061376$, $\omega = 0.000159702433409084$, $r_{\mathrm{e}} = 221.161037914965$, $\mu_{\mathrm{e}} = 0.273285127671081$.

To illustrate the results, we take the planar Oxz symmetry, and we compute the approximate s.p.o. in two cases: a window (x, J) taking the fixed value $z = 0$, and a window (x, z) taking the fixed value $J = 0.00255$. In both cases, we made a 1000×1000 grid window. Figure 2 presents these windows. On the top of the figure, we have the families of s.p.o. for a wide range of values of J in the window (x, J) with $z = 0$. We see on this figure a clear evolution of the families of s.p.o except for a small range of values of $J \in [0.002, 0.003]$. To see more clearly this region, we made a zoom that is presented in the middle of picture. Finally, the bottom picture presents a window perpendicular to the previous one representing the window (x, z) with a fixed value $J = 0.00255$.

In order to study the need of using a fine grid in the method, we present again the plot 2.2, but now with two different sizes of the grid 1000×1000 on the left and 100×100 on the right (see Fig. 3). From the plots, we observe that a detailed analysis of the problem really needs such a fine grid (1000×1000), and thus, it requires a large computational effort.

In Fig. 4, we illustrate the results by showing some Keplerian periodic orbits in the rotating frame of the Moon whose initial conditions, given in the Table 1, correspond with intersections of the horizontal lines $x = 2$, $x = -2$ with the curves of s.p.o. in the *top and middle* of Fig. 2 after improving the orbits with the method explained in Sect. 5. They are planar orbits because we take the window $z = 0$ that represents the planar case of the satellite lunar problem. The orbital plane is the equator of the Moon, which is placed at the origin. We show orbits with different multiplicities. We include, in the last column of Table 1, the stability index $k = \mathrm{Tr}(M) - 2$, of each orbit, where M is the monodromy matrix (Henon 1965, 1969; Skokos 2001). The periodic orbit is linearly stable if $|k| < 2$, unstable if $|k| > 2$ and critical if $|k| = 2$. Therefore, as shown in the few examples of Table 1, the method locates without any problem all kinds of s.p.o. In fact, highly unstable s.p.o., for instance with $k \approx 10^5$, are located with this method.

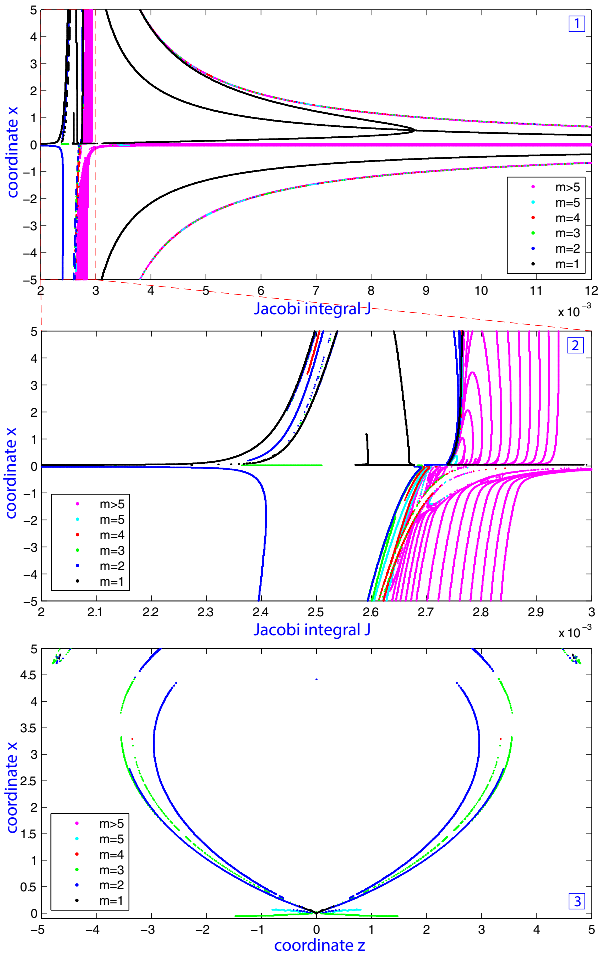

Fig. 2 Three bi-dimensional plots of the three-dimensional grid (x, z, J). On the *top* a window (x, J), $x \in [-5, 5]$, $J \in [0.002, 0.012]$, $z = 0$. In the *middle* a zoom of the *left part* of the *top* window, with $J \in [0.002, 0.003]$. On the bottom a window (x, z), $x \in [-5, 5]$, $z \in [-5, 5]$ that is perpendicular to the previous windows at $J = 0.0026$

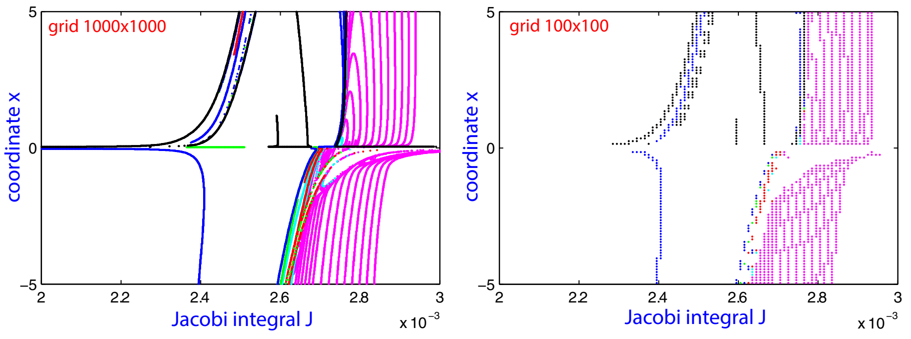

Fig. 3 A window (x, J) obtained with two different grids: 1000×1000 on the *left* and 100×100 on the *right*

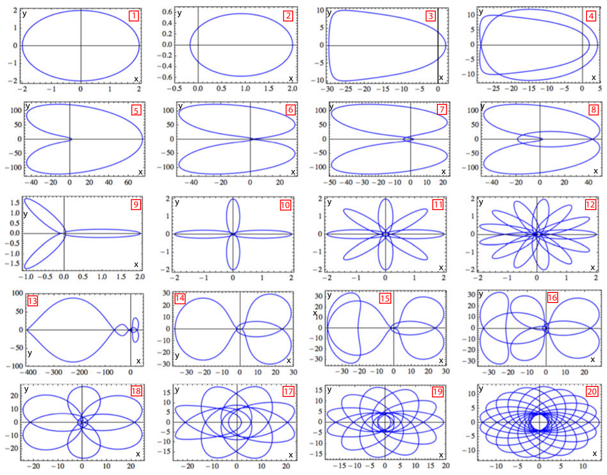

Fig. 4 Evolution of some planar s.p.o extracted from the intersection of the *horizontal lines* $x = 2$, $x = -2$ with the curves of s.p.o. in the *top* and *middle* of Fig. 2

Table 1 Initial conditions, Jacobi constant, period and stability of the orbits as shown in Fig. 4

Orbit	x_0	$\dot{y}_0$	J	T	k
1	-2	0.04132147930839	0.004125767891651	0.3041990889564e3	2.0000e0
2	2	0.01596131869958	0.005578465193585	0.1225802452123e3	1.6387e1
3	2	0.05543117487286	0.002760613740429	0.1527185667592e5	3.1587e1
4	2	0.05542269739227	0.002761553501979	0.2654505725994e5	1.0647e2
5	2	0.05823241870866	0.002442214299554	0.4167704148789e5	1.8394e5
6	2	0.05791398623192	0.002479199086943	0.4994764023415e5	3.8702e3
7	2	0.05786840604544	0.002484476469974	0.5124331522961e5	4.9442e2
8	2	0.05804033684499	0.002464548187134	0.5642304534070e5	1.9949e0
9	2	0.00580297347601	0.005799554387051	0.3306316548630e3	1.9983e0
10	-2	0.00664663331505	0.005789054809460	0.4413344680935e3	2.0000e0
11	-2	0.00735109772330	0.005779193906147	0.8863977345908e3	1.9999e0
12	2	0.00729868580715	0.005779958073703	0.1446000252549e4	3.3883e2
13	-2	0.05641710515721	0.002650342789565	0.6064465139200e5	3.2810e1
14	-2	0.05634004703846	0.002659031643588	0.3126199761735e5	3.5923e5
15	-2	0.05634378858107	0.002658610032216	0.5091698079331e5	1.1130e4
16	-2	0.05627751606207	0.002666073729768	0.5654664194648e5	1.3568e0
17	-2	0.05604076078647	0.002692665674357	0.3585532767686e5	2.3674e0
18	2	0.05527574012685	0.002777821441642	0.4973391231648e5	1.9999e0
19	2	0.05475188453984	0.002835460027550	0.4270822814253e5	2.0000e0
20	2	0.05380498282430	0.002938252711491	0.4115460996020e5	2.0000e0

Numbers of the left column correspond with the number of the orbits in the figure

4 Locating generic periodic orbits: an evolutionary strategy

Symmetric periodic orbits form a very important set of invariants of the system but sometimes we are also interested in general periodic orbits. To reach that goal, we need a different method than the grid search one.

An autonomous dynamical system, $\dot{\boldsymbol{x}} = F(\boldsymbol{x})$, has a periodic orbit if there is a vector $\boldsymbol{x}_0$ and a scalar T such that $||\boldsymbol{x}(T; \boldsymbol{x}_0) - \boldsymbol{x}_0|| = 0$. Therefore, the problem of finding periodic orbits is equivalent to find the zeros of a nonnegative $f : \mathcal{D} \in \mathbb{R}^n \to \mathbb{R}^+$, or, since the function is nonnegative function, to find the absolute minima of the function f in the domain $\mathcal{D}$. Therefore, the problem becomes an optimization problem where modern techniques, like the evolution strategy (ES), inspired on biological processes, can be applied (note that other optimization methods can be applied, like in Mauger et al. 2010) and Skokos et al. (2005). We want to remark that there are more possible options as choosing the periodicity conditions or the cost function in the literature, like the one used in variational methods where the condition is imposed along all the orbit. For instance, in Contopoulos and Harsoula (2010) a variational principle is proposed where an initial loop approximating a p.o. is evolved toward a true p.o. by minimization techniques of local errors along all the loop. This technique is extremely powerful once we have an initial approximated periodic orbit as we need the data along a loop. Our approach is different as we try to find a large number of p.o. without any previous information.

A classical $(\mu/\rho+\lambda)$-ES (Beyer and Schwefel 2002) begins with a population of μ random items $\boldsymbol{y} = (\boldsymbol{x}, \boldsymbol{\sigma})$ where $\boldsymbol{x} \in \mathcal{D} \subset \mathbb{R}^n$ is a point of the n-dimensional domain and $\boldsymbol{\sigma} \in \mathbb{R}^m$ is an m-dimensional strategy parameter vector, which controls the evolution of each point (the most usual dimension of vector $\boldsymbol{\sigma}$ is either 1 or n). By evaluating the objective function f at each given point, we may define the first generation as $\mathbb{G}_0 = (\boldsymbol{x}_i, \boldsymbol{\sigma}_i, f(\boldsymbol{x}_i)); \quad i = 1, \ldots, \mu$, therefore, a *generation* is a matrix $\mathbb{G} \in \mathcal{M}(n + m + 1, \mu)$.

Each point $\boldsymbol{x}$ may suffer a *mutation*. A typical mutation consists of adding a vector $\boldsymbol{z}$ to the point we want to mutate in the form $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{z}$, with $\boldsymbol{z} = (\sigma_1 y_1, \ldots, \sigma_n y_n)$, where the parameters σ_k are elements (repeated or not) of the parameter $\boldsymbol{\sigma}$, and y_k are random numbers obtained from a normal distribution $\mathcal{N}_k(0, 1)$, that is, $\boldsymbol{z} = (\sigma_1 y_1, \ldots, \sigma_n y_n) \in (\sigma_1 \mathcal{N}_1(0, 1), \ldots, \sigma_n \mathcal{N}_n(0, 1))$. The mutated elements obtained from this expression belong to an n-dimensional hypercube, but the properties of the normal distribution concentrate the mutated elements in an n-dimensional ellipsoid centered at $\boldsymbol{x}$ with more density of points near the center. If we use a one-dimensional vector parameter, then $\sigma_k = \sigma$, $\forall k$, and the ellipsoid is just an n-dimensional sphere of radius σ.

To build a new generation, we need to create the *offsprings*, and there are several ways of doing so. Essentially, an offspring is a mutation of a *recombination* of ρ parents (chosen at random), with ρ a predetermined fixed number between one and n. The recombination is a function of the ρ parents that gives a new point $\boldsymbol{x}^*$, for instance the average. When $\rho = 1$, there is only one parent, and the recombination is a clone of the parent. In our work, we shall take $\rho = 1$; hence, we do not mention other strategies.

Let us assume that from a population of μ points we have obtained λ offsprings; thus, we get a population of $\mu + \lambda$ points. Next, we evaluate the objective function, f, at each one of the above $\mu + \lambda$ points, and we select those μ points at which the objective function takes lower values. These last μ points, complemented with a new parameter vector $\boldsymbol{\sigma}$ (mutated from the previous one), will form the next generation $\mathbb{G}_1 = (\boldsymbol{x}_i, \boldsymbol{\sigma}_i, f(\boldsymbol{x}_i)); \quad i = 1, \ldots, \mu$.

The basic idea of the evolution strategy just described is to let the population evolve generation after generation, choosing for the next generation $\mathbb{G}_{k+1}$ the μ better adapted items among the μ items of the previous population plus the λ offsprings. The process will finish when we reach a convergence criteria or we exceed a maximum number of iterations or CPU time.

Sometimes, the objective function has many minima in the domain, and the classical evolution strategies tend to find a cluster of neighboring solutions around a peak, while the remaining solutions are not detected. To circumvent this, we use a variant of an evolution strategy named restricted evolution strategy (RES) (Im et al. 2004; Qing et al. 2005). Like in ES algorithm, the first step of a RES algorithm is to create at random the first-generation $\mathbb{G}_0$. To create the next generation, we apply a $(1/1+\lambda)$-ES method to each point of the previous generation, i.e., for each point, and inside its sphere of influence, we create λ clones and mutate them. Among the λ offsprings, we pick up the best, and in this way, the sphere of influence is moving. Sometimes, it happens that two spheres of influence have a common part; in that case, the sphere with worse behavior is abandoned and a new point is chosen at random; thus, we are always dealing with the same number of points. By doing so, we avoid the concentration of solutions, while we continue searching for new ones. To avoid the convergence of the method to previously computed solutions, a point of the population is deleted when a previous solution is inside of its sphere of influence. This technique is completely different than the deflection technique used in particle swarm optimization method (Parsopoulos and Vrahatis 2004) that changes the objective function each time that a solution is found. Note that the RES method, as other similar methods, does not guarantee to find all the most important s.p.o.

The zeros of a function $f : \mathcal{D} \in \mathbb{R}^n \to \mathbb{R}$, as a matter of fact, are the absolute minima of the nonnegative function $||f|| : \mathcal{D} \in \mathbb{R}^n \to \mathbb{R}^+$. To that goal, we may use an evolution strategy as above explained, to find zeros of a function, and consequently, to get initial conditions for p.o. As it is well known, in Hamiltonian systems, p.o. usually appear grouped in families associated with a parameter; hence, our function presents a massive number of zeros. Therefore, we use an adaptation of the RES algorithm modified to search multiple zeros of n-dimensional functions. With a RES, we obtain approximate values of the minima; however, we cannot guarantee neither the exact value of it, nor a value within a certain margin of error. As a matter of fact, we usually do not know how far is the value reached from the exact solution, which makes it difficult to fix a stop criterion of a RES. In our case, the minima are zeros; thus, it is sufficient to fix a simple criterion by stopping the algorithm when we reach a point $\boldsymbol{x}$ such that $\|f(\boldsymbol{x})\| \leq \epsilon$ for a pre-fixed value of ϵ. By doing so, the RES algorithm is improved in two aspects:

– We detect exactly the zeros with the desired precision.
– When we detect a zero, we extract it from the population and store it in a separate file containing the zeros; then, we add to the population another point chosen randomly and continue the process. With this small change, we are not restricted to obtain only μ zeros (the size of the population), and we may continue the process until we have the desired number of zeros.

To check the method, we consider the lunar problem with the Keplerian potential V=Vk .

The p.o. are characterized by the initial conditions $\boldsymbol{x}_0$, and the period T, such that $||\boldsymbol{x}(T;\boldsymbol{x}_0) - \boldsymbol{x}_0|| = 0$. Then, to find p.o. we need to find zeros of a function of dimension 7 (6 variables and period). In order to find periodic orbits with repeating ground traces (as usually demanded by the Space Agencies), we take the period of the orbit equal to the rotation period of the Moon; then, we search zeros of a six-dimensional function, and consequently the CPU time decreases considerably. Figure 5 shows some Keplerian periodic orbits in the rotating frame of the Moon whose initial conditions are given in the Table 2. The analytical expression of the state transition matrix of the Keplerian motion, given in Goodyear (1965) and Shepperd (1985), evaluated at the period by the help of an algebraic manipulator, allows to compute analytically the monodromy matrix and consequently the expression of the two stability indexes k_1, k_2 (Bray and Goudas 1967; Brouke 1969; Skokos 2001) for any Keplerian orbit, giving $k_1 = k_2 = -2$ in all cases; then, these indexes are not shown in Table 2. These are non-planar and non-symmetric periodic orbits. We remark that fixing the period is not

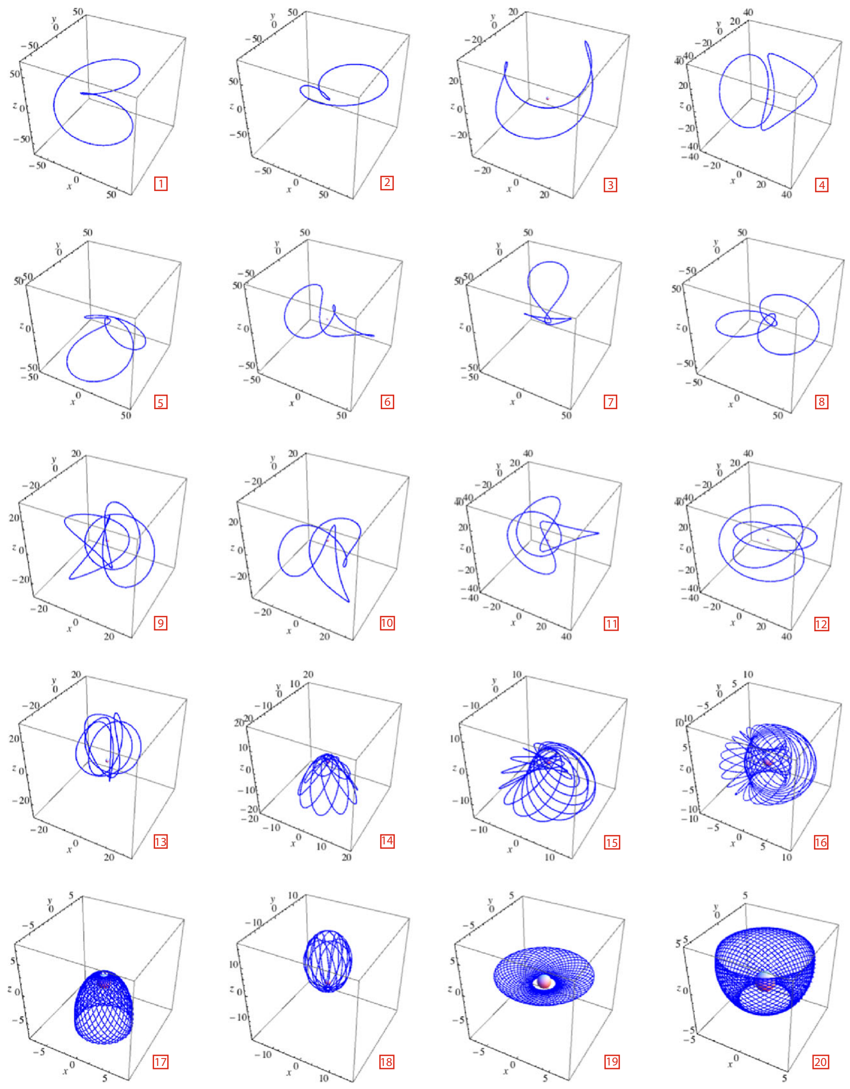

Fig. 5 Some generic Keplerian periodic orbits in the rotating frame of the Moon

necessary in the algorithm (it can be used for the variable T); however, any condition to reduce the number of variables of the cost function is welcome because it reduces drastically the CPU time.

5 Locating periodic orbits with arbitrary precision: a corrector algorithm

Another situation when an extra algorithm is needed is when we are interested in initial conditions of a system with a very high precision. This requirements may appear when we are interested to follow orbits inside a chaotic region or for very long-time simulations or for periodic orbits in a sticky-chaos region (Contopoulos and Harsoula 2010). Using the p.o. data

Table 2 Initial conditions of the orbits as shown in Fig. 5

Orbit	x_0	y_0	z_0
1	3.300540091234878e1	2.108099265187861e1	3.417438892149968e1
2	−1.816869796500198e1	−1.976809363691883e1	2.875553802353993e1
3	9.112731970696395e0	2.669153580213064e1	−1.140788896382731e1
4	1.213831775993075e0	4.750665453936941e1	2.759102029434862e1
5	−3.159366030357849e1	−1.724903548116221e1	−3.550521442901420e1
6	4.895980398203288e1	7.203791557902568e0	−8.168972302663168e0
7	1.068588086559215e1	−8.115378001711470e0	3.078070062321854e0
8	5.300852679036238e1	−6.392769858981069e0	−1.312512669725957e1
9	−6.903541987027292e0	−2.661201720905471e1	−8.341046175694927e0
10	−7.531902718343555e0	−8.937095166787440e0	−1.569402849285869e0
11	3.001531465714566e1	−3.633224107344410e0	2.078747939466323e1
12	3.430616652873464e1	1.279815766268968e1	4.673296782804202e0
13	1.200231885945300e1	4.714389198877880e0	2.148052669154326e1
14	−3.020928962441515e0	9.974725381119498e0	−1.345897284566149e1
15	−4.986812156309238e0	−1.008232288481315e1	−2.551161512618732e0
16	4.338127665712836e0	−1.168486935257796e0	−1.266167832140468e0
17	1.757945300435834e0	3.342701435656420e0	−3.290020081054985e0
18	−2.532302273386746e0	−3.153190677288939e0	1.525068788284411e1
19	−5.237836639006277e0	1.655467965802949e0	5.497079791245928e1
20	−1.785593134175071e0	4.067931768573944e0	1.400822997391712e0
Orbit	$\dot{x}_0$	$\dot{y}_0$	$\dot{z}_0$
1	3.303384050334531e − 3	−1.318366571980379e − 2	8.226295912745999e − 4
2	4.722861864923698e − 3	7.097106575800928e − 3	−5.013828401409188e − 3
3	−4.081184075788002e − 3	−4.664028509626036e − 3	−6.017132249243100e − 3
4	1.120750709250688e − 2	−2.184626789255233e − 3	6.476713716229274e − 4
5	4.585889364650893e − 4	8.597499378965763e − 3	2.274951473329514e − 3
6	3.246730218755892e − 3	−5.745978303591500e − 3	4.527058855458678e − 3
7	1.547171792563295e − 2	−2.980216851668819e − 3	1.003571217185862e − 2
8	−3.987613954779899e − 3	−1.064801951680535e − 2	−1.964148094690957e − 3
9	−4.372602207680582e − 3	3.507958438683189e − 3	−7.858552589295834e − 3
10	1.578811571896955e − 3	−7.970904891788288e − 3	−1.842580501623897e − 2
11	−6.314275655114385e − 3	−5.967625285084249e − 3	3.402474594700446e − 3
12	3.676049436653138e − 3	−1.330653824696322e − 2	3.652448840791945e − 3
13	5.372188625680729e − 3	2.330577729782557e − 3	−5.985084494490858e − 3
14	−3.665248943736108e − 3	−4.267161503278209e − 3	7.677328047076539e − 3
15	−1.233298112899599e − 2	−2.667469749349587e − 3	−1.047347922239669e − 2
16	−1.911744623256852e − 3	−1.847603603984891e − 2	2.518716493843953e − 2
17	−1.167065164527244e − 2	−4.529347064052614e − 3	2.440314622970590e − 2
18	1.242504667321969e − 3	−6.864845718686833e − 3	−4.898374627181791e − 3
19	6.314958288053613e − 3	−1.878713357141473e − 2	−4.849729414436984e − 3
20	2.043718853728191e − 2	6.661521171427789e − 3	−1.585610998818093e − 2

Numbers of the left column correspond with the number of the orbits in the figure obtained from the previous sections, we enter in a correction algorithm to obtain the initial conditions with the required precision level. This algorithm, presented in Abad et al. (2011), is based simply on applying the Newton method but with the use of an arbitrary precision ODE solver (TIDES is one of the few numerical ODE techniques able to solve an ODE system with arbitrary precision) and the SVD decomposition. For completeness, we describe briefly the algorithm used.

In order to find accurate initial conditions of a periodic orbit, that is, to find the roots of the periodicity condition equation $\boldsymbol{x}(T;\boldsymbol{x_0})-\boldsymbol{x_0}=\boldsymbol{0}$ with $\boldsymbol{x}\in\mathbb{R}^n$, we use an iterative algorithm based on the n-dimensional Newton method. We start from approximate initial conditions $\boldsymbol{x_0}$ of a periodic orbit with estimated period T_0, $\boldsymbol{x}(T_0;\boldsymbol{x_0})\approx\boldsymbol{x_0}$, of a dynamical problem given by the ordinary differential equation $\dot{\boldsymbol{x}}=\boldsymbol{f}(\boldsymbol{x})$. Let us suppose that at the step i we have a set of corrected initial conditions $(\boldsymbol{x_i},T_i)$. To improve these initial conditions and, therefore, to obtain the approximate corrections $(\Delta\boldsymbol{x_i},\Delta T_i)$, we expand the periodicity condition

$$\boldsymbol{x}(T_i+\Delta T_i;\boldsymbol{x_i}+\Delta\boldsymbol{x_i})=\boldsymbol{x_i}+\Delta\boldsymbol{x_i}, \tag{10}$$

in a multivariable Taylor series up to the first order. As a result, we have the next linear system of n equations with $n+1$ unknowns

$$\big((M-I)\ \boldsymbol{f}(\boldsymbol{y}_{T_i})\big)\begin{pmatrix}\Delta\boldsymbol{x_i}\\ \Delta T_i\end{pmatrix}=\big(\boldsymbol{x_i}-\boldsymbol{x}(T_i;\boldsymbol{x_i})\big), \tag{11}$$

where $I\in\mathbb{R}^{n\times n}$ is the identity matrix, and the monodromy matrix M is the $n\times n$ matrix $\partial\boldsymbol{x}/\partial\boldsymbol{x_0}$, solution of the variational equations evaluated at $(\boldsymbol{x_i},T_i)$, and $\boldsymbol{y}_{T_i}=\boldsymbol{x}(T_i,\boldsymbol{y}_i)$. Note that the free software TIDES allows us to compute simultaneously the solution of an ODE and its partial derivatives automatically up to any precision level. Moreover, we add the condition

$$\begin{pmatrix}\boldsymbol{f}^T(\boldsymbol{y}_i) & 0\end{pmatrix}\begin{pmatrix}\Delta\boldsymbol{x_i}\\ \Delta T_i\end{pmatrix}=0, \tag{12}$$

to have a correction $(\Delta\boldsymbol{x_i},\Delta T_i)$ orthogonal to the vector field and to optimize each iteration of the method.

In the lunar orbiter problem, we have the Jacobi integral $J(\boldsymbol{x})=h$. Thus, in order to maintain this constant along each iteration the constrain condition has the form

$$(\nabla_{\boldsymbol{x}}J)|_{(T_i;\boldsymbol{x}_i)}\,\Delta\boldsymbol{x}_i=h-h_{T_i}. \tag{13}$$

We add it to the conditions (11) and (12), and therefore, the $(i+1)$-step is defined by the $(n+2)\times(n+1)$ linear system

$$\begin{pmatrix}M-I & \boldsymbol{f}(\boldsymbol{y}_{T_i})\\ \boldsymbol{f}^T(\boldsymbol{y}_i) & 0\\ (\nabla_{\boldsymbol{x}}J)|_{(T_i;\boldsymbol{x}_i)} & 0\end{pmatrix}\begin{pmatrix}\Delta\boldsymbol{x_i}\\ \Delta T_i\end{pmatrix}=\begin{pmatrix}\boldsymbol{x_i}-\boldsymbol{x}_{T_i}\\ 0\\ h-h_{T_i}\end{pmatrix}, \tag{14}$$

where h is the desired value of the Jacobi integral and $\boldsymbol{x}_{T_i}=\boldsymbol{x}(T_i;\boldsymbol{x_i})$.

Note that the linear system (14) is not squared due to the introduction of the integrals of the system, and so, instead of solving the system we use the least-norm method. Thus, we have chosen the singular value decomposition (SVD) method (Demmel 1997; Trefethen and Bau 1997) since we want to develop a general algorithm that is specially well adapted for rank-deficient problems. Therefore, this method leads to a robust solver, which allows us to find a solution that minimizes the residual error.

As test example, we have selected a symmetric p.o. (orbit number 1 in Table 1) which approximate initial conditions are obtained in the previous sections. In Fig. 6, we present

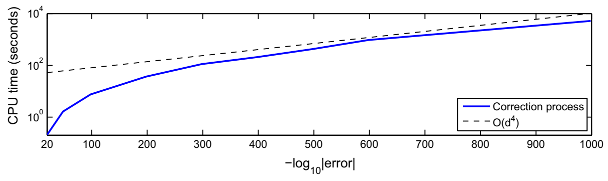

Fig. 6 Computational relative error versus CPU time

the CPU time of the correction algorithm for the p.o. just to show that with this simple algorithm and techniques we are able to compute in a reasonable time p.o. with 1000 precision digits. The method is quadratically convergent, and the complexity is polynomial in the number of digits (Abad et al. 2012): the computational complexity behaves like $\mathcal{O}(d^4)$ being $d = -\log_{10}(\texttt{TOL})$ (see Fig. 6). To help the reader to check his/her own routines, we provide the initial conditions of the p.o. ($y = z = \dot{x} = \dot{z} = 0$) with 100 precision digits (the CPU time in a personal computer PC Intel quad-core i7, CPU 860, 2.80 GHz with 100 digits is 7.48 s and with 1000 digits is 5204.28 s and using multiple precision libraries MPFR and GMP

$$\begin{aligned}
x &= -0.1999999999999966829270967239587202346539810580936781\\
&\quad 1213469041905394793496912275428875411833136945\text{e}+1\\
\dot{y} &= 0.4132147930839713912686441213395409434513876194201050\\
&\quad 670127812483433837733859377230179507888210977187\text{e}-1\\
T &= 0.3041990889564870032346101366033245391456746155483856\\
&\quad 340773159908834963129264753081145009541100959078\text{e}+3
\end{aligned}$$

6 Parallelization and application to the lunar problem

One of the main objectives of this work is to explore the use of parallelization tools to find both kinds of p.o., symmetric or not. Finding p.o. for the lunar problem requires a big computational effort, and it gives us a good example to check the advantages of parallelization techniques. In both cases, symmetric and non-symmetric p.o., we use the software OPENMP (Chandra et al. 2001) as the parallelization tool. Note that both methods commented in this paper are easily parallelizable as the work may be distributed in an homogeneous way to a large number of computer cores. Therefore, in this section we show the effectivity of parallel strategies that increase the computer performance of the proposed methods.

The 3DOF grid search algorithm studies consecutively each of the $N \times N$ squares of each bi-dimensional window. We use the information of the two right vertexes of each square to initialize the information of the contiguous right square; then, we parallelize the algorithm by computing each horizontal line of squares on each processor. A previous task to make possible an efficient parallelization of the code was a modification of the software TIDES (the core of our computing methods), in order to distribute the trajectory integrations over several cores without communications between them.

To parallelize the evolutionary strategy, we take advantage of the main difference between the classical ES and the modified RES. Instead, the classical $(\mu/\rho + \lambda)$-ES, a restricted evolutionary strategy, is the combination of μ families described by a $(1/1 + \lambda)$-ES. We parallelize the code by sending each family to a different processor. With this simple scheme, we pay a penalty over the sequential algorithm because now it is difficult to compare two

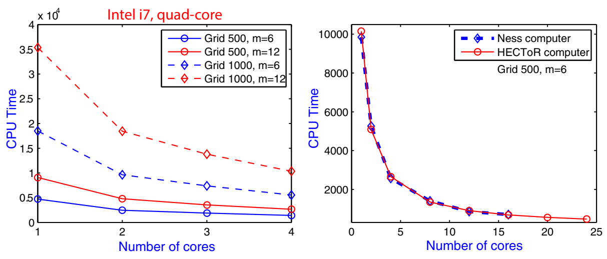

Fig. 7 CPU time versus the number of cores in obtaining a window of s.p.o. until multiplicity m with the grid search method

Table 3 CPU time (seconds) and efficiency for each number of cores in a grid 500×500 and $m = 6$, with Intel i7 (quad-core), Ness and HECToR computers

Intel i7 (quad-core)								
Cores	$p = 1$	2	3	4				
CPU	4717	2461	1889	1420				
$E(p)$	–	0.958	0.832	0.830				
Ness computer								
Cores	$p = 1$	2	4	8	12	16		
CPU	9848	5264	2568	1406	859	705		
$E(p)$	–	0.935	0.958	0.875	0.954	0.873		
HECToR computer								
Cores	$p = 1$	2	4	8	12	16	20	24
CPU	10153	5087	2658	1350	904	685	560	470
$E(p)$	–	0.998	0.954	0.940	0.935	0.926	0.905	0.899

families in order to delete one of them when they approach to the same periodic orbit. But in fact this penalty is very small compared with the benefits of parallelization.

In this work, we have used three different computers. A personal computer PC Intel quad-core i7, CPU 860, 2.80 GHz under a 2.6.32-29-generic SMP x86 64 GNU/Linux system. The Ness computer, that is a high-performance computing resource of The Edinburgh Parallel Computing Centre (EPCC), has sixteen processors of 2.6 GHz AMD Opteron (AMD64e) with 2 GB of memory, and the supercomputer HECToR (High-End Computing Terascale Resources) that is located at the University of Edinburgh in Scotland. HECToR's hardware configuration (2011, phase 2b (XE6)) contains a large number of compute nodes, each one with two 12-core AMD Opteron 2.1 GHz Magny Cours processors. Each 12-core processor shares 16Gb of memory, giving a system total of 59.4 Tb. We just use one of these compute nodes.

One standard metric to measure the performance of a parallel algorithm is just the CPU time. Other standard metric to measure the performance of a parallel algorithm is the *Efficiency*, $E(p)$, defined as

$$E(p) = \frac{T(1)}{p \cdot T(p)},$$

Table 4 Number of orbits computed with different multiplicities that span from one to twelve, for grids of 500×500 and 1000×1000, respectively

Grid 500×500							
Multiplicity	$m = 1$	2	3	4	5	6	
No. of orbits	10	1070	3368	7878	8243	10927	
Multiplicity	7	8	9	10	11	12	TOTAL
No. of orbits	13447	15295	16405	15390	15062	13901	120996
Grid 1000×1000							
Multiplicity	$m = 1$	2	3	4	5	6	
No. of orbits	22	2092	6322	18666	21820	31790	
Multiplicity	7	8	9	10	11	12	TOTAL
No. of orbits	40190	50870	55108	53723	56256	51355	388214

Table 5 CPU time (seconds) and efficiency for each number of cores of the evolution strategy algorithm for computing 1000 p.o., with Intel i7 (quad-core), Ness and HECToR computers

Intel i7 (quad-core)								
Cores	$p = 1$	2	3	4				
CPU	6543	2702	2444	1516				
$E(p)$	–	1.2	0.892	1.078				
Ness computer								
Cores	$p = 1$	4	8	16				
CPU	9693	1767	1335	797				
$E(p)$	–	1.37	0.907	0.760				
HECToR computer								
Cores	$p = 1$	2	4	8	12	16	20	24
CPU	10215	5422	2925	1225	765	651	477	429
$E(p)$	–	0.942	0.873	1.042	1.112	0.981	1.071	0.992

where $T(1)$ is the CPU time of the sequential algorithm, and $T(p)$ is the CPU time of the parallel algorithm executed on p processors.

Figure 7 shows the CPU time, in seconds, versus the number of cores when we use the grid search method to compute a particular window (x, z), $J = 0.0026$ searching s.p.o. until multiplicity m ($m = 6, 12$), with a 500×500 or 1000×1000 grid. Tables 3 and 4 show the results of different tests for this problem. Table 3 contains the CPU time and efficiency for different cases, and Table 4 shows the number of s.p.o. of a given multiplicity found for each grid. Note that the total number of s.p.o. we have been able to compute is very large, which will permit to perform detailed analysis of the problems. In the benchmarking tests, we have obtained very good results and a relative efficiency around the 90 % with 24 cores at HECToR computer. This means that this algorithm is perfectly adapted to massive parallel computing because it distributes the trajectory integration over several cores without communications between them, and thus, the expected efficiency of the parallel methods is close to 1.

Table 6 Averaged efficiency of the evolution strategy to compute 100 p.o. 50 times in an Intel i7 (quad-core)

Cores	$p = 1$	2	3	4
CPU	561.98	302.13	206.67	146.93
$E(p)$	–	0.930	0.906	0.956

Finally, in Table 5 we show the CPU time and efficiency of the parallelization of the evolution strategy algorithm to compute p.o. in the lunar problem. The results correspond to run the algorithm until we find 1000 p.o. Note that in the efficiency results we may have values greater than one, because the algorithm has a random part and the CPU time changes on each application of the algorithm. In order to obtain a more realistic result about efficiency, we run the algorithm 50 times to find, each time, 100 p.o. We summarize the results in Table 6, where the time represents the averaged CPU time of each test, and E(p) is the averaged efficiency. Note that now, as the time for one core is an averaged time, we obtain a more realistic table with efficiency values over 90 %.

7 Conclusions

In this paper, we study the problem of locating families of symmetric periodic orbits by means of the grid search method and generic periodic orbits (symmetric or not) using an evolutionary strategy. Both methods have been parallelized, and in spite of their simpleness, they permit to locate a very large number of initial conditions of periodic orbits, and therefore, they are quite useful in the global study of a Hamiltonian system.

To show the applicability of the methods, we have applied them to find periodic orbits in a 3-DOF system, the motion of a lunar problem. These orbits may serve as preliminary orbits for a detailed study in a Space Agency. The 3DOF grid search method is used to find symmetric periodic orbits. By using this method, we obtain several families of symmetric periodic orbits of the lunar problem. A modification of an evolutionary strategy method has been used to find non-symmetric periodic orbits in a simplified version of the lunar orbiter problem. The method has been applied to obtain 1000 general periodic orbits.

A parallel version of both algorithms has been developed and checked. The results of efficiency of the parallel algorithms prove that the methods admit a simple and good parallelization strategy, that is very useful when we try to find a massive number of periodic orbits.

Acknowledgments A. Dena thanks Prof. G. Lord for his hospitality during her stay at the Heriot-Watt University, Edinburgh. The work of A. Dena was carried out under the HPC-EUROPA2 project (project number: 228398) with the support of the European Commission - Capacities Area - Research Infrastructures. Besides, this work made use of the facilities of HECToR, the UK's national high-performance computing service, which is provided by UoE HPCx Ltd at the University of Edinburgh, Cray Inc and NAG Ltd, and funded by the Office of Science and Technology through EPSRC's High-End Computing Programme. The authors A. Abad, R. Barrio and A. Dena were partially supported by the Spanish Research project MTM2012-31883.

References

Abad, A., Barrio, R., Blesa, F., Rodríguez, M.: Algorithm 924: TIDES, a Taylor series integrator for differential equations. ACM Trans. Math. Softw. **39**(1), 1–28 (2012)

Abad, A., Barrio, R., Dena, A.: Computing periodic orbits with arbitrary precision. Phys. Rev. E **84**, 016701 (6) (2011)

Abad, A., Elipe, A.: Evolution strategies for computing periodic orbits. Math. Comput. Simul. (2014). doi:10.1016/j.matcom.2014.05.014

Abad, A., Elipe, A., Tresaco, E.: Analytical model to find frozen orbits for a lunar orbiter. J. Guid. Control Dyn. **32**(3), 888–898 (2009)

Barrio, R., Blesa, F.: Systematic search of symmetric periodic orbits in 2DOF Hamiltonian systems. Chaos Solitons Fractals **41**(2), 560–582 (2009)

Beyer, H., Schwefel, H.: Evolution strategies: a comprehensive introduction. Nat. Comput. **1**, 3–52 (2002)

Bray, T.A., Goudas, C.L.: Doubly symmetric orbits about the collinear Lagrangian points. Astron. J. **72**, 202–213 (1967)

Brent, R.: An algorithm with guaranteed convergence for finding a zero of a function. Comput. J. **14**(4), 422–425 (1971)

Brouke, R.: Stability of periodic orbits in the elliptic restricted three-body problem. AIAA J. **7**(4), 1003–1009 (1969)

Chandra, R., Maydan, D., Kohr, D., Dagum, L.: Parallel Programming in OpenMP. Morgan Kaufmann Publishers, Los Altos (2001)

Contopoulos, G., Harsoula, M.: Stickiness effects in conservative systems. Int. J. Bifurcat. Chaos **20**(7), 2005–2043 (2010)

Demmel, J.W.: Applied Numerical Linear Algebra. Society for Industrial and Applied Mathematics (SIAM), Philadelphia (1997)

Dhooge, A., Govaerts, W., Kuznetsov, Y.A.: MATCONT: a MATLAB package for numerical bifurcation analysis of ODEs. ACM Trans. Math. Softw. **29**(2), 141–164 (2003)

Doedel, E.: AUTO: a program for the automatic bifurcation analysis of autonomous systems. In: Proceedings of the Tenth Manitoba Conference on Numerical Mathematics and Computing, 1980, vol. **30**, pp. 265–284. Winnipeg, Manitoba (1981)

Elipe, A., Lara, M.: Frozen orbits about the moon. J. Guid. Control Dyn. **26**(2), 238–243 (2003)

Farantos, S.: POMULT: a program for computing periodic orbits in Hamiltonian systems based on multiple shooting algorithms. Comput. Phys. Commun. **108**, 240–258 (1998)

Folte, D., Quinn, D.: Lunar frozen orbits. In: AAS/AIAA Astrodynamics Specialist Conference. Paper AIAA 06-6749 (2006)

Goodyear, W.H.: Completely general closed-form solution for coordinates and partial derivatives of the two-body problem. Astron. J. **70**, 189–192 (1965)

Hairer, E., Norsett, S.P., Wanner, G.: Solving Ordinary Differential Equations I. Nonstiff Problems, Second revised edn. Springer, Heidelberg (1993)

Henon, M.: Exploration numerique du probleme restreint. II Masses egales, stabilite des orbites periodiques. Ann. d'Astrophys. **28**, 992–1007 (1965)

Henon, M.: Numerical exploration of the restricted problem. V. Hill's Case: periodic orbits and their stability. Astron. Astrophys. **1**, 223–238 (1969)

Hill, G.W.: Researches in the lunar theory. Am. J. Math. **1**, 5–26 (1878)

Im, C., Kim, J., Jung, H.: A novel algorithm for multimodal function optimization based on evolution strategy. IEEE Trans. Magn. **40**(2), 1224–1227 (2004)

Kazantzis, P.G., Goudas, C.L.: A grid search for three-dimensional motions and three new types of such motions. Astrophys. Space Sci. **32**, 95–113 (1975)

Krauskopf, B., Osinga, H.M., Galán-Vioque, J. (eds.): Numerical continuation methods for dynamical systems. Springer, Heidelberg (2007)

Lara, M., Palacián, J.: Hill problem analytical theory to the order four: application to the computation of frozen orbits around planetary satellites. Math. Probl. Eng. Article ID 753653 (2009)

Lara, M., Pelaez, J.: On the numerical continuation of periodic orbits. An intrinsic, 3-dimensional, differential, predictor-corrector algorithm. Astron. Astrophys. **389**(2), 692–701 (2002)

Mauger, F., Chandre, C., Uzer, T.: Simulated annealing algorithm for finding periodic orbits of multi-electron atomic systems. Commun. Nonlinear Sci. Numer. Simul. **16**(7), 2845–2852 (2010)

Markellos, V., Black, W., Moran, P.E.: A grid search for families of periodic orbits in the restricted problem of three bodies. Celest. Mech. **9**, 507–512 (1974)

Meyer, K.R., Hall, G.R., Offin, D.: Introduction to Hamiltonian Dynamical Systems and the N-body Problem. Applied Mathematical Sciences, vol. 90, 2nd edn. Springer, New York (2009)

Parsopoulos, K.E., Vrahatis, M.N.: On the computation of all global minimizers through particle swarm optimization. IEEE Trans. Evol. Comput. **8**, 211–224 (2004)

Poincaré, H.: Les Méthodes nouvelles de la Mécanique Céleste. Gauthier-Villarset fils, Paris (1892)

Polymilis, C., Servizi, G., Skokos, Ch., Turchetti, G., Vrahatis, M.N.: Topological degree theory and local analysis of area preserving maps. Chaos **13**, 94–104 (2003)

Qing, L., Gang, W., Qiuping, W.: Restricted evolution based multimodal function optimization in holographic grating design. Evol. Comput. **1**, 789–794 (2005)

Roy, A., Ovenden, M.: On the occurrence of commensurable mean motions in the solar system: the mirror theorem. Mon. Not. R. Astron. Soc. **115**(3), 296–309 (1955)

Russell, R.: Global search for planar and three-dimensional periodic orbits near Europa. J. Astron. Sci. **54**(2), 199–226 (2006)

Shepperd, S.W.: Universal Keplerian state transition matrix. Celest. Mech. **35**, 129–144 (1985)

Skokos, Ch.: On the stability of periodic orbits of high dimensional autonomous Hamiltonian system. Phys. D **159**, 155–179 (2001)

Skokos, Ch., Parsopoulos, K.E., Patsis, P.A., Vrahatis, M.N.: Particle swarm optimization: an efficient method for tracing periodic orbits in three-dimensional galactic potentials. Mon. Not. R. Astron Soc. **359**, 251–260 (2005)

Tadi, M.: On computing periodic orbits. J. Sound Vib. **283**, 495–506 (2005)

Trefethen, L.N., Bau III, D.: Numerical Linear Algebra. Society for Industrial and Applied Mathematics (SIAM), Philadelphia (1997)

Tsirogiannis, G., Perdios, E., Markellos, V.V.: Improved grid search method: an efficient tool for global computation of periodic orbits. Celest. Mech. Dyn. Astron. **103**, 49–78 (2009)

Vrahatis, M.N.: Solving systems of nonlinear equations using the nonzero value of the topological degree. ACM Trans. Math. Softw. **14**, 312–329 (1988)

Vrahatis, M.N.: Algorithm 666: CHABIS: a mathematical software package for locating and evaluating roots of systems of nonlinear equations. ACM Trans. Math. Softw. **14**, 330–336 (1988)

Weilian, Y.: Frozen orbit for lunar orbiter. Adv. Astron. Sci. **117**, 379–388 (2004)

Wulff, C., Schebesch, A.: Numerical continuation of symmetric periodic orbits. SIAM J. Appl. Dyn. Syst. **5**(3), 435–457 (2006)