Polynomial approximations of Poincaré maps for Hamiltonian systems, II.

Petit J-M.
C.N.R.S., Observatoire de Nice, France
petit@obs-nice.fr

Froeschlé C.
C.N.R.S., Observatoire de Nice, France
claude@obs-nice.fr

Abstract

In paper I polynomial interpolating formulae of order 3 and 5 have been proposed and tested for transforming a non–linear differential Hamiltonian system into a map without having to integrate whole orbits as in the well known Poincaré return map technique. The precision of the computations increases drastically with the order of the polynomial fit which requires an extended amount of local information, i.e. information about neighbouring points. The first part of the paper deals with another type of interpolation where the information, within the same accuracy, refers only to the nearest neighbours but takes into account gradient information. The results are in very good agreement with those obtained using an order 3 symmetrical interpolation formula well inside the phase space. Moreover the new method is more effective at the border of the phase space when compared with asymmetrical interpolation. The second part of the paper deals with higher dimensional mappings, i.e. mappings for hamiltonian systems with 3 degrees of freedom.

1 Introduction

Poincaré maps are widely used for studying the qualitative behaviour of differential equations (see Henon 1981). In order to study stability problems, many authors have sought explicit algebraic mappings which approximate, at least qualitatively, the Poincaré maps obtained from the original Newton equations. Froeschlé and Petit (1990, paper I) have reviewed some of these mappings. They rely on some approximations made in order to separate either — in the case of deterministic mappings — an integrable part and some instantaneous perturbations, or — for stochastic mappings — a source of endogeneous/exogeneous stochasticity (see Froeschlé and Rickman, 1988). These mappings are all *ad hoc* and are reliable only in a limited region of phase space where the approximations are valid. In paper I we built a mapping valid everywhere in phase space, following an idea already used by Varosi et al. (1987) but in the framework of non–Hamiltonian systems (i.e., systems where attractors do exist). The method consists of coarse–graining the phase–space surface of section and then interpolating the value of the image of a point. Linear interpolation requires a rather fine graining of the phase space, hence it is necessary to compute a lot of points on the grid. However, Taylor expansions of order 3 and 5 can provide very good results as long as symmetrical interpolation formulae are applied, for which it is necessary to use an extended grid, i.e. a grid extended over the region of phase space one wants to explore. Since there are cases where one cannot cross a given limit, asymmetrical interpolation formulae have been tested, but their accuracy was found to be inferior. Therefore we have developed another type of interpolation, where the information, including that relative to the gradients, is stored only for the nearest–neighbour vertices. Thus, not only images of vertices are computed, but also tangential mappings at each vertex of the grid. The second part of the paper deals with higher dimensional mappings i.e. Poincaré maps for Hamiltonian systems with three degrees of freedom.

2 The synthetic mappings

The basic idea of the synthetic mapping is to interpolate the image of a point, when those of a set of points located at the vertices of a grid are given.

In paper I we derived the interpolating formulae for accuracy of order 3 and 5. These formulae require the use of respectively 4 and 6 points along each dimension of the mapping. In other words, we need a certain amount of non–local information. In the following, we use more local information, i.e. we look only at the nearest neighbours but we take more information at these points in order to have the same accuracy.

2.1 Taylor interpolation

Assume that we know the values of the functions $x^{(n+1)} = f(x^{(n)}, y^{(n)})$ and $y^{(n+1)} = g(x^{(n)}, y^{(n)})$ at discrete points x_i and y_j and also their derivatives with respect to x, y and the cross derivatives with respect to x and y. The points are equally spaced in x and y. We want to interpolate the values of f and g at any point. We first derive the interpolation formula in one dimension. We write the Taylor expansion for f about a point in the middle of a given interval,

$x' = (x_i + x_{i+1})/2$:

$$
\begin{aligned}
f(x) &= f(x') + (x-x')f'(x') + \frac{(x-x')^2}{2} f''(x') \\
&\quad + \frac{(x-x')^3}{6} f'''(x') + \mathrm{O}\left((x-x')^4\right), \\
f'(x) &= f'(x') + (x-x')f''(x') \\
&\quad + \frac{(x-x')^2}{2} f'''(x') + \mathrm{O}\left((x-x')^3\right).
\end{aligned}
\tag{1}
$$

We use the middle point to ensure the symmetry of the formula. We need to know the value of f and its derivatives at x'. For this we write (1) for x_i:

$$
\begin{aligned}
f(x_i) &= f(x') - \frac{h}{2} f'(x') + \frac{h^2}{8} f''(x') \\
&\quad - \frac{h^3}{48} f'''(x') + \mathrm{O}(h^4), \\
f'(x_i) &= f'(x') - \frac{h}{2} f''(x') + \frac{h^2}{8} f'''(x') + \mathrm{O}(h^3),
\end{aligned}
\tag{2}
$$

where $h = x_{i+1} - x_i$. The corresponding formulae for $f(x_{i+1})$ and $f'(x_{i+1})$ are obtained from (2) respectively by replacing the minus signs with plus signs. Neglecting terms of order 4 or higher, one can invert (2) and write $f(x')$, $f'(x')$, $f''(x')$ and $f'''(x')$ as functions of $f(x_i)$, $f(x_{i+1})$, $f'(x_i)$ and $f'(x_{i+1})$.

$$
\begin{aligned}
f(x') &= \frac{1}{8}[4f(x_i) + 4f(x_{i+1}) + hf'(x_i) - hf'(x_{i+1})], \\
hf'(x') &= \frac{1}{4}[-6f(x_i) + 6f(x_{i+1}) - hf'(x_i) - hf'(x_{i+1})], \\
h^2 f''(x') &= -hf'(x_i) + hf'(x_{i+1}) \\
h^3 f'''(x') &= 6[2f(x_i) - 2f(x_{i+1}) + hf'(x_i) + hf'(x_{i+1})].
\end{aligned}
\tag{3}
$$

By substituting into (1) we obtain:

$$
\begin{aligned}
f(x) &= \frac{f(x_i)}{2}[1 - 3h + 4h^3] \\
&\quad + \frac{f(x_{i+1})}{2}[1 + 3h - 4h^3]
\end{aligned}
\tag{4}
$$

$$+\frac{f'(x_i)}{8}[1-2h-4h^2+8h^3]$$
$$+\frac{f'(x_{i+1})}{8}[-1-2h+4h^2+8h^3].$$

To interpolate a two dimensional function, we iterate the one dimensional interpolation. We want to know the image of point (x, y). Assume that we know the images and gradients at points (x, y_i) and (x, y_{i+1}). Then we evaluate the image using eq. (4). This means that we must first interpolate f and $\frac{\partial f}{\partial y}$ at these two points. f at (x, y_i) (resp. (x, y_{i+1})), is obtained from f and $\frac{\partial f}{\partial x}$ at (x_j, y_i) and (x_{j+1}, y_i) (resp. (x_j, y_{i+1}) and (x_{j+1}, y_{i+1})). For the value of $\frac{\partial f}{\partial y}$ at (x, y_i), we interpolate using $\frac{\partial f}{\partial y}$ and $\frac{\partial^2 f}{\partial x \partial y}$ at the vertices.

2.2 A special 2D case

When using the previous algorithm, we reproduce the Taylor expansion up to order 3 in every direction. So in n dimensions, the n-dimensional Taylor expansion is reproduced up to global order 3 and we also have terms of the expansion up to global order $3n$. In the case of a two-dimensional mapping, we need to know 16 values to interpolate one point. But if we want to have the interpolation only up to global order 3, we need only to determine 10 constants. We need to use values only at the four nearest neighbours and give them the same importance. So we have 12 conditions: f, $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ at the four points. Here again we expand around the middle point ($x' = (x_i + x_{i+1})/2, y' = (y_j + y_{j+1})/2$) with all the terms up to global order 3 and two terms of global order 4, and neglecting the other terms of order 4 and higher:

$$\begin{aligned}
f(x,y) &= f(x',y') + (x-x')f_x(x',y') + (y-y')f_y(x',y') \\
&\quad + \frac{(x-x')^2}{2} f_{xx}(x',y') + \frac{(y-y')^2}{2} f_{yy}(x',y') \\
&\quad + (x-x')(y-y') f_{xy}(x',y') + \frac{(x-x')^3}{6} f_{xxx}(x',y') \\
&\quad + \frac{(x-x')^2(y-y')}{2} f_{xxy}(x',y') + \frac{(x-x')(y-y')^2}{2} f_{xyy}(x',y') \\
&\quad + \frac{(y-y')^3}{6} f_{yyy}(x',y') + \frac{(x-x')^3(y-y')}{6} f_{xxxy}(x',y') \\
&\quad + \frac{(x-x')(y-y')^3}{6} f_{xyyy}(x',y'), \qquad (5) \\
f_x(x,y) &= f_x(x',y') + (x-x')f_{xx}(x',y') + (y-y')f_{xy}(x',y')
\end{aligned}$$

$$
\begin{aligned}
&+\frac{(x-x')^2}{2}f_{xxx}(x',y')+\frac{(y-y')^2}{2}f_{xyy}(x',y')\\
&+(x-x')(y-y')f_{xxy}(x',y')+\frac{(x-x')^2(y-y')}{2}f_{xxxy}(x',y')\\
&+\frac{(y-y')^3}{6}f_{xyyy}(x',y'),\\
f_y(x,y) \;=\;& f_y(x',y')+(x-x')f_{xy}(x',y')+(y-y')f_{yy}(x',y')\\
&+\frac{(x-x')^2}{2}f_{xxy}(x',y')+\frac{(y-y')^2}{2}f_{yyy}(x',y')\\
&+(x-x')(y-y')f_{xyy}(x',y')+\frac{(x-x')^3}{6}f_{xxxy}(x',y')\\
&+\frac{(x-x')(y-y')^2}{2}f_{xyyy}(x',y').
\end{aligned}
$$

Writing that these equations must be satisfied with the 12 values we know at the four corners of the cell, we find:

$$
\begin{aligned}
f(x,y) \;=\;& \frac{f(x_i,y_i)}{4}[1+3h+3k+8hk-4h^3(1+2k)-4k^3(1+2h)]\\
&+\frac{f(x_{i-1},y_i)}{4}[1-3h+3k-8hk+4h^3(1+2k)-4k^3(1-2h)]\\
&+\frac{f(x_i,y_{i-1})}{4}[1+3h-3k-8hk-4h^3(1-2k)+4k^3(1+2h)]\\
&+\frac{f(x_{i-1},y_{i-1})}{4}[1-3h-3k+8hk+4h^3(1-2k)+4k^3(1-2h)]\\
&+\frac{f_x(x_i,y_i)}{16}[(1+2k)(-1-2h+4h^2+8h^3)]\\
&+\frac{f_x(x_{i-1},y_i)}{16}[(1+2k)(1-2h-4h^2+8h^3)] \qquad (6)\\
&+\frac{f_x(x_i,y_{i-1})}{16}[(1-2k)(-1-2h+4h^2+8h^3)]\\
&+\frac{f_x(x_{i-1},y_{i-1})}{16}[(1-2k)(1-2h-4h^2+8h^3)]\\
&+\frac{f_y(x_i,y_i)}{16}[(1+2h)(-1-2k+4k^2+8k^3)]\\
&+\frac{f_y(x_{i-1},y_i)}{16}[(1-2h)(-1-2k+4k^2+8k^3)]
\end{aligned}
$$

$$+\frac{f_y(x_i, y_{i-1})}{16}[(1+2h)(1-2k-4k^2+8k^3)]$$
$$+\frac{f_y(x_{i-1}, y_{i-1})}{16}[(1-2h)(1-2k-4k^2+8k^3)]$$

The great advantage of this formula over the one derived from (4) is that it does not require knowing the second order cross derivative of f. Suppose the mapping is given by numerical integration of differential equations. It is conceptually easy to obtain the first order derivatives of the mapping by a similar integration (see below). But the determination of the second order cross derivative requires a numerical differentiation which is not always easy to do. The problem is that it is not possible to generalize to higher dimensions. In dimension 3, we have 32 conditions (8 nearest neighbours, each with 4 values), and the terms of global order less or equal to 3 give only 20 unknown coefficients. Therefore, we have to choose arbitrarily 12 terms of global order larger than 3. We made a few choices which all lead to singular matrices. However there must exist a choice that gives a non singular matrix, but it might prove very difficult to find. And when increasing the number of dimensions, we increase even more the number of terms we have to choose arbitrarily.

3 Derivative evaluation

In the previous algorithms, we need to know the derivatives of the mapping. We have to compute the first order derivatives with the same order of precision as the orbit. Consider the differential equation:

$$\frac{dX}{dt}(t) = F(X, t). \tag{7}$$

Since we want to know a derivative, we integrate this equation for two nearby points: X and $X + \epsilon$. The evolution of the second point is governed by equation:

$$\frac{dX}{dt}(t) + \frac{d\epsilon}{dt}(t) = F(X + \epsilon, t) = F(X, t) + L_{(X,t)}.\epsilon + \mathrm{O}(\epsilon^2), \tag{8}$$

where L is the Jacobian matrix. Since ϵ is an infinitely small vector, its evolution is governed by the well known variation equations:

$$\frac{d\epsilon}{dt}(t) = L_{(X,t)}.\epsilon \tag{9}$$

This is a linear equation, so we can normalize ϵ as we wish.

Let us introduce the following notation. Γ denotes the intersection of the surface of section S and the constant energy surface E. Then we call $T\Gamma_P$ the space tangent to Γ at point P. We consider a point $X_0 \in \Gamma$ and a basis $\{e_i^{(0)}\}$ of $T\Gamma_{X_0}$ (in our case, the basis consists of two linearly independant vectors). We need to know the image $X_1 \in \Gamma$ of X_0 and the image $\{e_i^{(1)}\} \in T\Gamma_{X_1}$ of $\{e_i^{(0)}\}$. For a Hamiltonian system with two degrees of freedom, we have four dimensional phase space (x_1, x_2, x_3, x_4). We define the surface of section by $x_2 = 0$ and $x_4 > 0$ (in our particular case, $x_2 = y$ and $x_4 = dy/dt$) and the surface of constant energy by $E = E_0$. The coordinates $(\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4)$ of the vector ϵ in $T\Gamma_{X_0}$ are constrained to

$$\begin{aligned} \epsilon_2 = &\ 0, \\ \nabla E.\epsilon = &\ 0. \end{aligned} \tag{10}$$

A problem arises when integrating simultaneously equations (7) and (9), because when the orbit crosses Γ, that is in X_1, the image of ϵ does not necessarily belong to $T\Gamma_{X_1}$. We come back to the classical variation method and consider the neighbouring point $X_1 + \epsilon$ where ϵ is again infinitely small. Using an Euler integration scheme with time step h we integrate until time t when the point reaches Γ:

$$X(t+h) + \epsilon(t+h) = X(t) + \epsilon(t) + h\frac{d(X+\epsilon)}{dt} \in \Gamma. \tag{11}$$

We have to solve equation (11) for h in order to have $x_2(t+h) + \epsilon_2(t+h) = 0$, where x_2 and ϵ_2 are the second components of X and ϵ. We simplify the equation by neglecting the term $h\, d\epsilon/dt$ which is a small quantity of order 2. Since $X_1 = X(t)$ is in Γ, we have $x_2(t) = 0$. So the time step h is given by:

$$h = -\frac{\epsilon_2(t)}{\frac{dx_2(t)}{dt}}, \tag{12}$$

and the final vector $\epsilon(t+h)$ is:

$$\epsilon(t+h) = \left(\epsilon_i(t) - \frac{\epsilon_2(t)\frac{dx_i(t)}{dt}}{\frac{dx_2(t)}{dt}}\right), \quad i = 1 \ldots 4. \tag{13}$$

Since we are interested in the derivatives, the final vectors will be divided by the modulus of the initial vectors and therefore, we can use unit length vectors to perform the computations, since each operation is linear with respect to ϵ.

We are now able to write explicitly a synthetic mapping from Γ into Γ using the gradient method. We consider the Hill's problem in a rotating frame (cf Duncan et al., 1989, Froeschlé and Petit, 1990). The coordinates are $x_1 = x$, $x_2 = y$, $x_3 = dx/dt = \dot{x}$ and $x_4 = dy/dt = \dot{y}$. The surface of section is defined by $y = 0$, $\dot{y} > 0$ and $E = const.$ (Jacobi constant). We do not plot $x, \dot{x}$ directly since for a given value of x, there is a maximum value of $|\dot{x}|$ allowed and we do not get a rectangular grid. So we plot $z = x, \lambda = \dot{x}/|\dot{x}|_{max}$, where λ belongs to $[-1, 1]$. The limiting value of $|\dot{x}|$ is:

$$|\dot{x}|_{max} = \sqrt{x^2 + \frac{2(1-\mu)}{\mu - x} + \frac{2\mu}{\mu - x - 1} - E}. \tag{14}$$

In the general interpolation scheme (derived from equation 4), we need the second order cross derivative. This has to be computed by mean of numerical differenciation. We know the images of a point X_0 and two vectors e_1 and e_2. For the cross derivative, we integrate equation (9) for vector e_1 but with the Jacobian matrix evaluated along the orbit of point $X_0 + \epsilon\, e_2$. This second orbit (very close to the first one) is not actually computed but we approximate it by taking $X(t) + \epsilon\, e_2(t)$.

4 Memory management

So far we have always supposed that we know the value of the mapping and eventually its derivatives on a discrete grid. But this can be very memory consuming. In the case of a four dimensional mapping, with 50 points in each dimension, we need to store information for 6 250 000 points. Each point accounts for at least 4 values (when we do not use the derivatives, 16 values for each component if we use derivatives, which means 64 values for each point) needing 8 bytes each (we are using double precision arythmetic). This amount to $2.\,10^8$ bytes of storage.

But a typical orbit will visit only a small part of the total space, especially if it is an invariant orbit. Therefore in order to reduce storage requirements, we only store images of those grid points which are needed to compute the orbit. The storage scheme is based on the hashing technique: to compute the image of a point, we need to know the images of the neighbouring points of the grid. These images are computed and then labelled for the purpose of identification. The labels are just integer indices (at a point of the grid we can make a one to one correspondance with integers). The images are stored in a list which is ordered according to the label index. To compute the next image, we again need to compute the images of the new neighbouring points. Some of them are already computed and we have to locate them in the list, others have to be computed and inserted in the list. These two problems of location and insertion can be time consuming hence we use the double trick of a linked-list (for fast insertion) and a *hash function* (for fast location) (see Appendix A). In fact this hash function allows one to split a large list into a collection of shorter lists. Actually, applying the hash function to a label gives the entry point of the list where that particular label is stored. In practice, the hash function is simply the remainder modulo M:

$$h(k) = 1 + k \bmod M, \tag{15}$$

where k is the labelling index and M an arbitrary integer.

Since the amount of computer memory is limited, we may need, for those orbits that visit a large region, to delete some previous points before storing new ones. These cases are, of course, more time consuming. For technical details, see Appendix A.

5 Results

5.1 The two dimensional case

An extensive study of the point set Taylor interpolation scheme for the two dimensional case can be found in paper I. We present first a sample of results using point set and tangential Taylor interpolation of order 3. As a model problem we adopt the standard mapping:

$$\begin{aligned} x^{(n+1)} &= x^{(n)} + a\sin(x^{(n)} + y^{(n)}), \qquad (\mathrm{mod}\ 2\pi) \\ y^{(n+1)} &= x^{(n)} + y^{(n)}. \qquad (\mathrm{mod}\ 2\pi) \end{aligned}$$

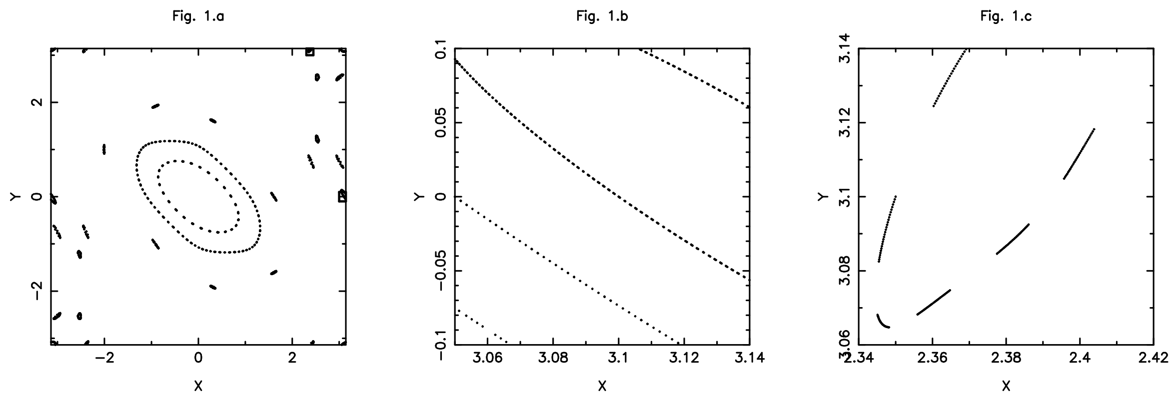
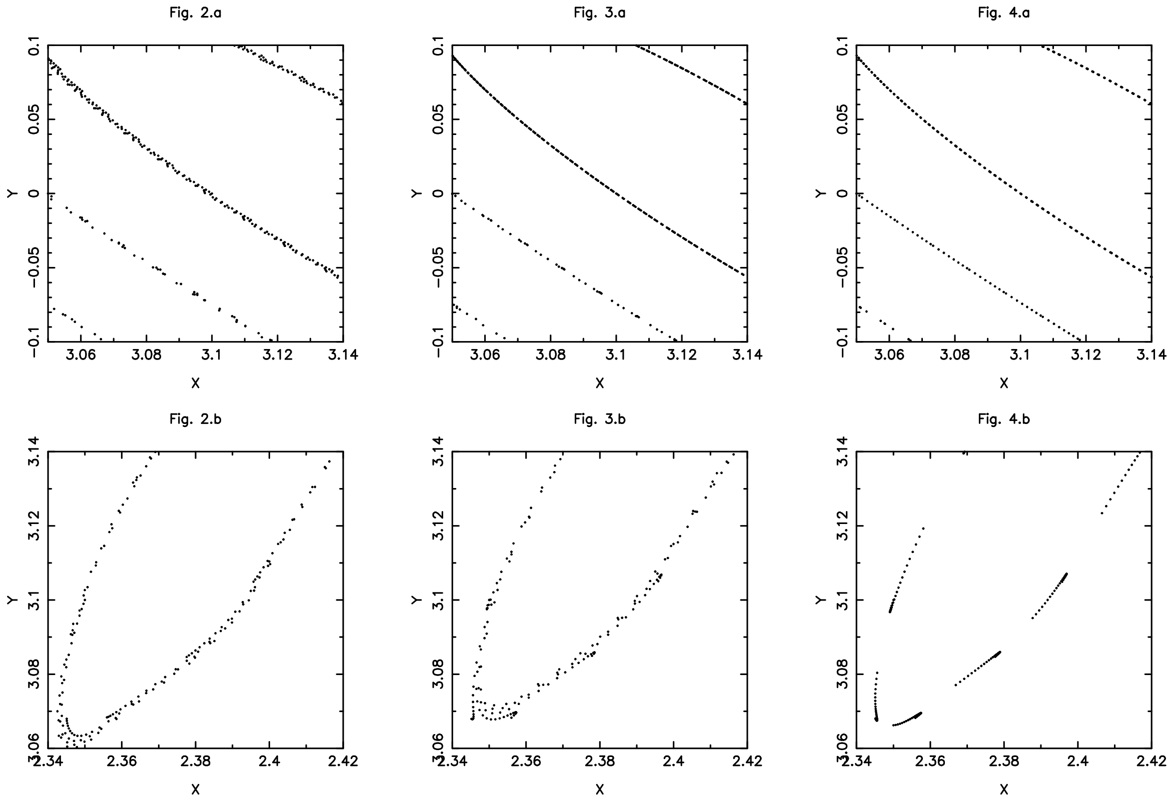


Fig. 1: (a) Plots of the standard mapping for $a = -1.3$. (b) and (c) are enlargements of the small boxes shown in (a), respectively at the right border and at the top border.

This has been selected since the computation of orbits is very fast and allows us to calculate many orbits and perform enough iterations for a meaningful comparison. Moreover, for the tangential Taylor interpolation, the computation of the derivatives of the mapping does not present any problem. Fig. 1 shows orbits of the standard mapping for $a = -1.3$. These mappings exhibit all the well–known typical features of problems with two degrees of freedom, such as invariant curves, “islands”, and stochastic zones where the points wander in a chaotic way. Figs. 1b and 1c are magnifications of the small boxes indicated in Fig. 1a. At this magnification level, details like second–order islands become evident and the approximation levels of the synthetic maps are easily visualized. Figs. 2a and 2b correspond to the same orbits and the same magnifications as Figs. 1b–c, but using the Taylor interpolation mapping of order $M = 3$ with decentered formulae on the edges of the mapping; here the grid was regular



Figs. 2a–b: Same as in Figs. 1b–c but using a Taylor approximation of order 3 with a regular grid and decentered formulae. **Figs. 3a–b:** The same as in Figs. 1b–c but using a Taylor approximation of order 3 with a non–regular grid and decentered formulae. **Figs. 4a–b:** The same as in Figs. 1b–c but using a Taylor approximation of order 3 with a regular grid and the gradient method.

and characterized by 40 cells. The results are very similar to those of the original map, except for the box close to the border. Figs. 3a–b correspond to the same formulae, but with cells having half the size of the previous ones close to the edges. Apparently this does not greatly improve the quality of the mapping in the border boxes. On the other hand, a definite improvement is obtained using the gradient formulae, as shown in figs. 4a–b. One can see that fig. 4b is much closer to fig. 1c than fig. 2b and even more than fig. 3b. And fig. 4a is closer to fig. 1b than fig. 2a.

We also tested the method on a restricted three body problem for which Duncan et al. (1989) have developed a special mapping. Fig. 5a exhibits orbits of the Poincaré map taking as surface of section the plane defined by the eccentricity and the mean longitude as polar coordinates, when the particle is in conjunction with the planet (i.e, in the rotating frame, when $y = 0$ and $y' > 0$). In fig. 5b are plotted the corresponding orbits for synthetic mapping T3 ($N = 100$). Fig. 5c shows the corresponding orbits for the tangential Taylor interpolation of order 3. In this case the tangential derivatives have been estimated using the method described in $ 3. The agreement is as good as with the corresponding point set interpolation. We must emphasize that all three figures correspond to the same initial conditions. One of these is at the very border of a stable region, which explains why there is one more invariant curve in fig. 5c. On the other hand, in the chaotic regions, the points do not all fit on the grid, and we have to stop the integration very early in the synthetic cases, although not for the real mapping. This explains the different number of points in the chaotic regions.

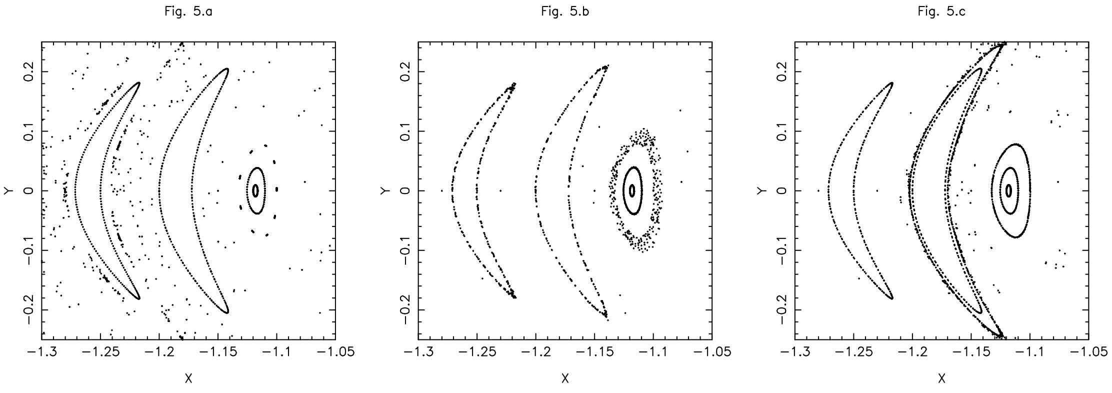


Fig. 5a: Trajectories of the Poincaré map of the restricted three–body problem, in the plane giving (as polar coordinates) eccentricity and mean longitude at conjunction with the planet, for the Neptune–to–Sun mass ratio $m/M_0 = 5.178\ 10^{-5}$ and a Jacobi constant of 3.0080694. **Fig. 5b:** Same as Fig. 5a but for the point set Taylor 3 interpolation with $N = 100$. **Fig. 5c:** Same as Fig. 5b but for the tangential Taylor 3 interpolation.

5.2 Poincaré maps for Hamiltonian systems with three degrees of freedom

5.2.1 Graphical display of symplectic four dimensional mappings

When the number of degrees of freedom is equal to 3, the surface of section has 4 dimensions and interpretations of the results are more difficult. Again, using a symplectic four-dimensional mapping appears suitable as a model problem in order to test both the different synthetic mappings and the hash function method. Furthermore a four-dimensional surface of section is populated by points and not by a continuous trajectory, even if they lie on an invariant surface. Therefore, since we compute a finite number of points, if we perform other sections, the probability of a point falling on an arbitrarily chosen surface vanishes. Therefore, we must take not a true section but rather a *slice* having a small yet finite thickness in order to catch some points. Figs. 6a and 7a are examples taken from Froeschlé (1972) for the four-dimensional mapping:

$$\begin{aligned}
x^{(n+1)} &= x^{(n)} + a_1 \sin(x^{(n)} + y^{(n)}) + b \sin(x^{(n)} + y^{(n)} + z^{(n)} + w^{(n)}), \qquad \pmod{2\pi} \\
y^{(n+1)} &= x^{(n)} + y^{(n)}, \qquad \pmod{2\pi} \\
z^{(n+1)} &= z^{(n)} + a_2 \sin(z^{(n)} + w^{(n)}) + b \sin(x^{(n)} + y^{(n)} + z^{(n)} + w^{(n)}), \qquad \pmod{2\pi} \\
w^{(n+1)} &= z^{(n)} + w^{(n)}. \qquad \pmod{2\pi}
\end{aligned}$$

A trajectory is a set of points in the (x, y, z, w) space. It is first projected on the three-dimensional (x, y, z) space; then a series of nine slices is taken, defined by $|z - z_0| < 0.01$ with nine regularly spaced values for z_0. Parameter values are $a_1 = -1.3$, $a_2 = -1$ for both figures; $b = 0.075$ for fig. 6 and $b = 0.275$ for fig 7. In the first case, the points fall on well defined curves in each slice; this suggests that they fall on a two-dimensional surface in (x, y, z) space. In the second case, the situation is rather different: the points appear to fill a three-dimensional region in (x, y, z) space. Both figures have been obtained by the tangential Taylor 3 ($\tau 3$) synthetic mapping. Figures for the point-set Taylor 3 (T3) mapping are identical. For figures with real mapping, see Figs. 11.d and 11.f from Froeschlé (1972). But again the figures are identical at least in the integrable case ($b = 0.075$).

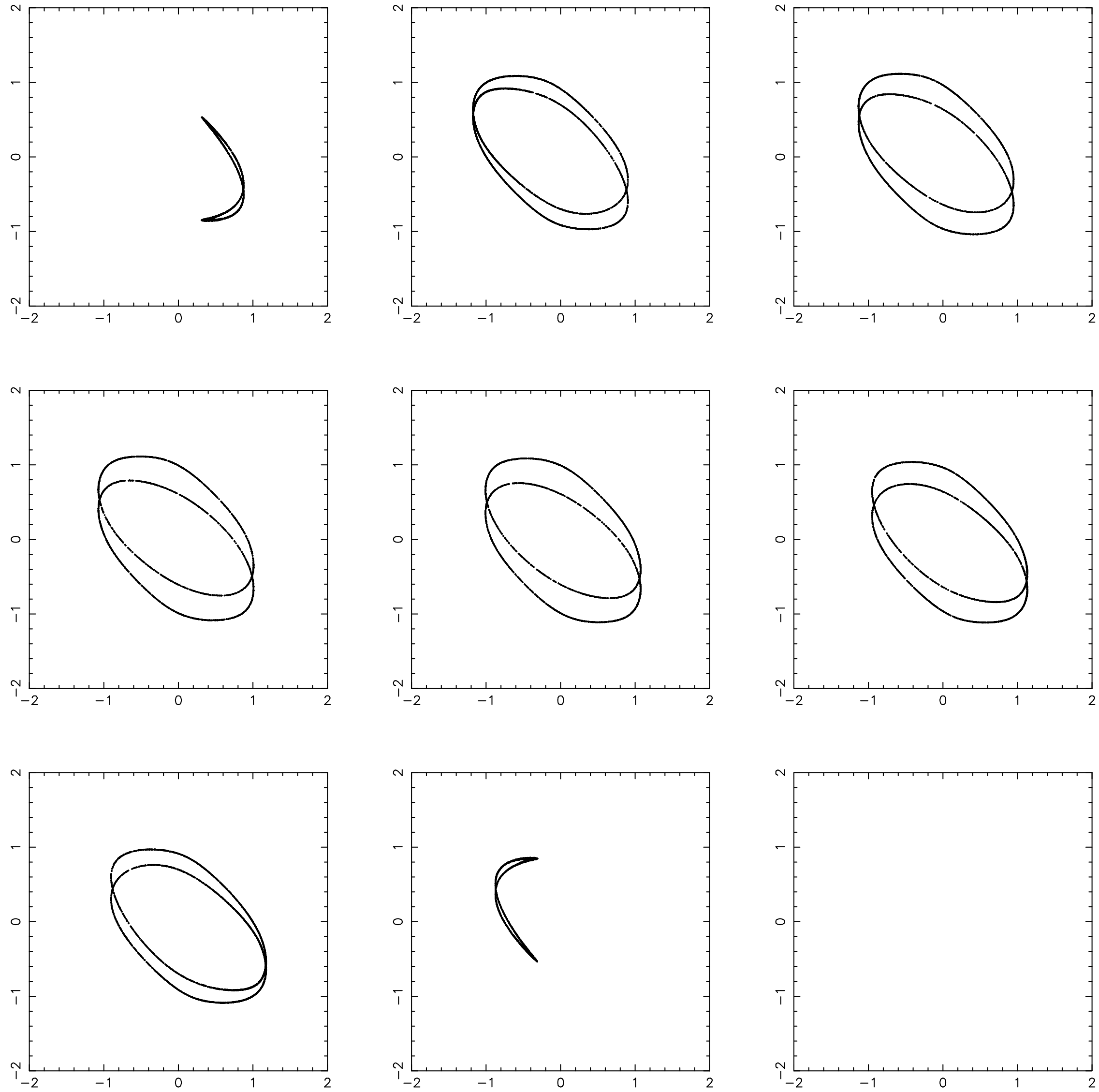


Fig. 6: Slices of the subspace (x, y, z) for the tangential Taylor 3 interpolation of the four-dimensional standard map, for $a_1 = -1.3$, $a_2 = -1.$ and $b = 0.075$, the starting point being $P = (1, 0, 0.5, 0.5)$. Then slices are defined by $|z - z_0| < 0.01$ with $z_0 = -1.5 + i/3$, where i varies from 1 to 9 from left to right and top to bottom.

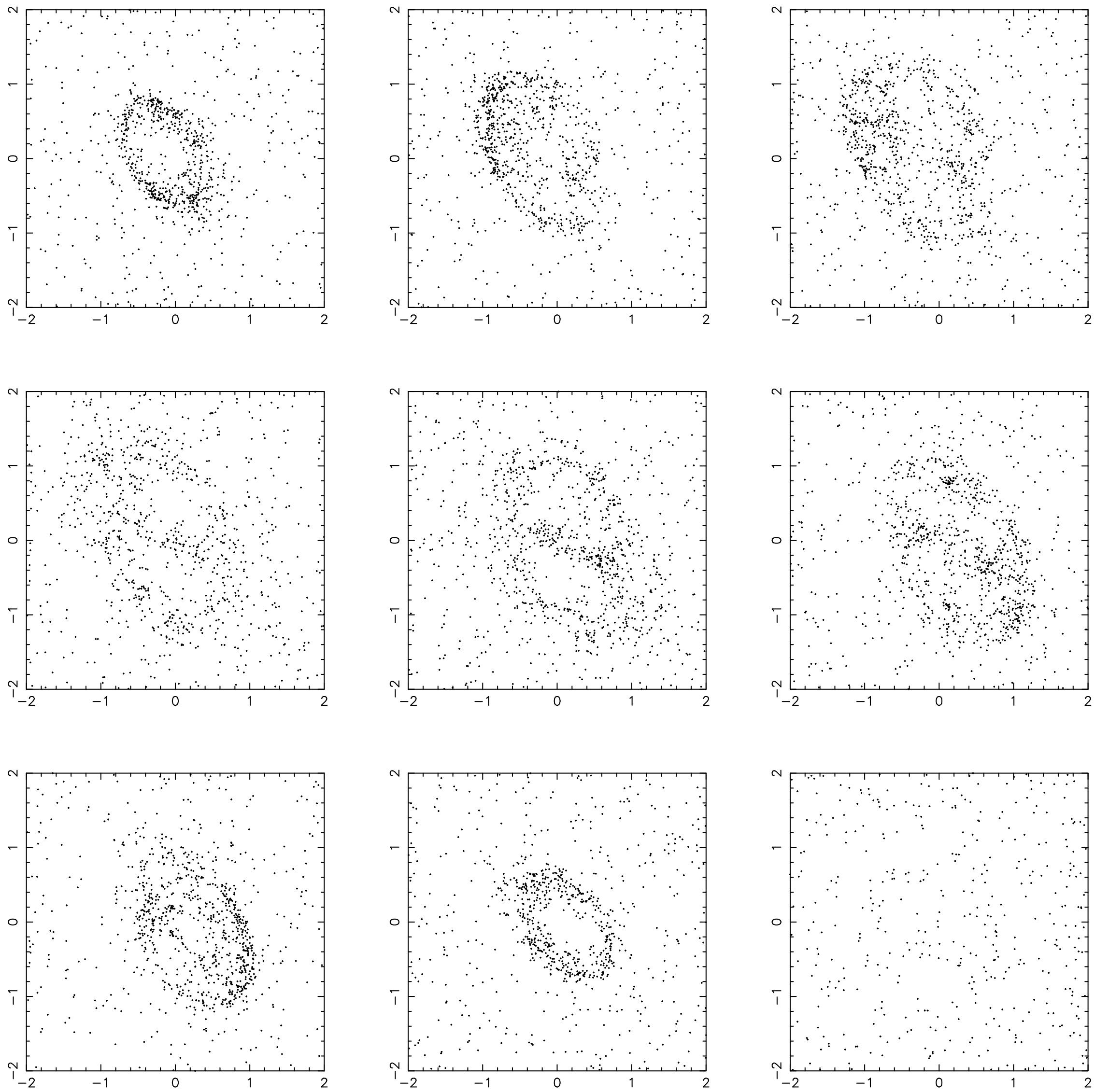


Fig. 7: Same as fig. 6, but with $b = 0.275$.

5.2.2 The Lyapounov characteristic numbers (LCNs)

A rather different technique consists in determining numerically the Lyapounov characteristic exponents of the trajectories. This allows one to distinguish between quasi-periodic and chaotic orbits and to map the regular and chaotic regions in phase space. The question raised by Froeschlé (1970) about the number of quasi-integrals for a one parameter set of orbits of the non planar restricted three-body problem has been solved more easily, using LCEs, by Gonczi and Froeschlé (1981). The combination of the two methods (slice cutting and LCEs) has been very useful for discovering and illustrating a sticking-exploding phenomenon. Of course, this technique does not provide any information about the shape of an orbit in phase space, nor about the set of points in the space of section since it gives just two numbers for each trajectory.

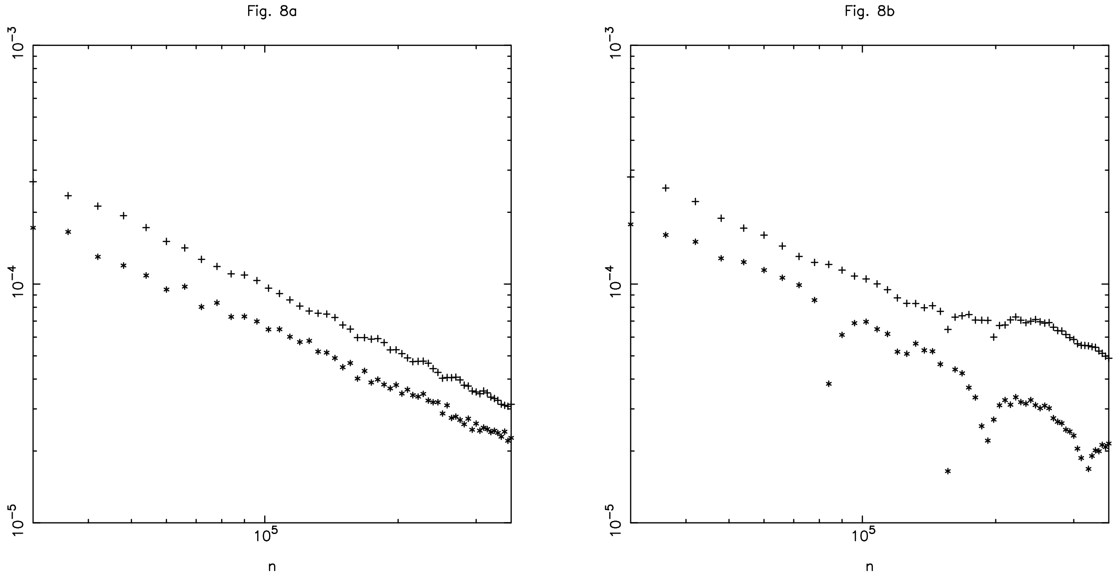
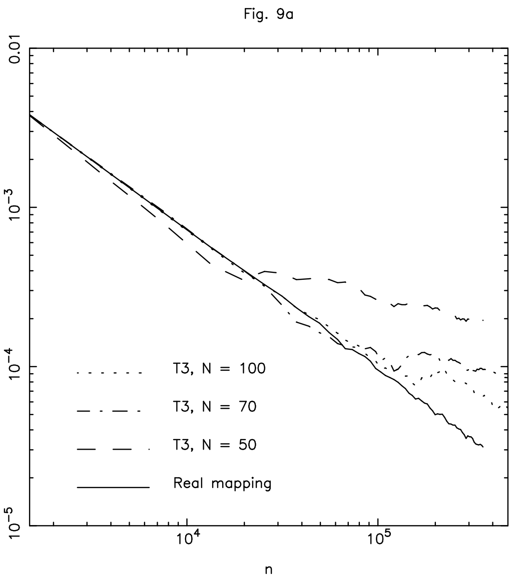


Fig. 8a: Positive Lyapounov characteristic indicators for the same orbit as in fig. 6 with $b = 0.075$, for the real mapping. **Fig. 8b:** Same as fig. 8a for the synthetic mapping $\tau 3$.

Fig. 8a shows, on a log-log scale, the behaviour of the two positive Lyapounov characteristic indicators γ_i^n (whose limits are identified with the Lyapounov characteristic numbers) as a function of time (i.e. the number n of iterations), for initial conditions taken in the ordered region ($P = (1, 0, 0.5, 0.5)$), with $b = 0.075$. Both γ_i^n appear to be decreasing functions of time. The largest shows very little departure from the expected linear behaviour.

Fig. 8b shows the same numbers but computed from the synthetic mappings $\tau 3$. The general behaviour is similar to the previous case, albeit the slope is slightly smaller and the secondary Lyapounov number presents some pathologic patterns. This discrepency is due to the fact that some grid points we use belong to the chaotic zone and therefore influence the computed numbers. This has been tested first by using a finer grid ($N = 100$ instead of $N = 50$). The discrepency between the real and the synthetic mapping almost disappears. Then we have computed the Lyapounov characteristic indicators with the former grid ($N = 50$) for an orbit closer to the center, that is well within the stable region. This allowed to use only points in the stable region. In this case again, the descrepency disappears.

The agreement between the real mapping and the T3 synthetic mapping is even more dependent of the size of the grid. Since we use grid points further away from the actual point we are looking at, we have more chance to mix stable and choatic regions. This can easily be seen on figs. 9a and 9b which represent, respectively, the dominant and secondary Lyapounov characteristic numbers for different grid sizes.



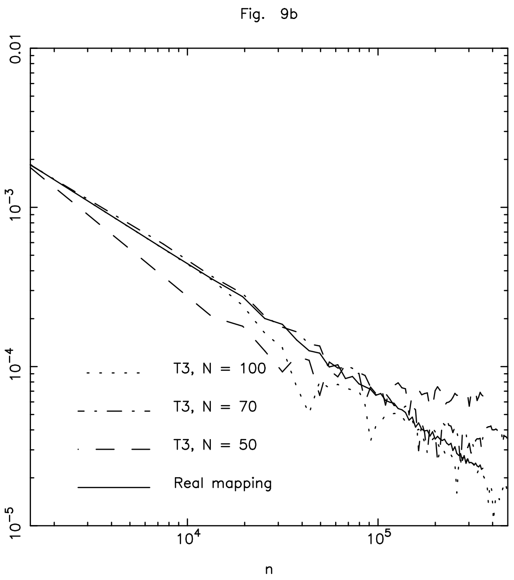


Fig. 9a: Dominant Lyapounov characteristic numbers for the same orbit as in fig. 6 with $b = 0.075$, for the real mapping, the synthetic mapping T3 with $N = 50$, $N = 70$ and $N = 100$. **Fig. 9b:** Same as fig. 9a for the secondary Lyapounov characteristic number.

In order to visualize the evolution in time and show in more concrete terms the sticking–exploding phenomenon we have plotted on figs. 10a-c the median slices ($|z| < 0.01$) of the projections along the w axes of the points P_n for $(i-1) \times 40000 \leq n < i \times 40000$, with i varying from 1 to 9 from left to right and top to bottom, and $b = 0.275$, for the real, T3 and $\tau 3$ mappings respectively. Figs. 11a-c show the corresponding variations of the two positive γ_i^n. Note that one should not pay too much attention to the fact that the times of explosion coincide, since this time is distributed according to a Poisson distribution (Froeschlé and Petit, 1992). We found slightly worse agreement for the explosion time when using the real mapping on two different computers that handle the last bit in floating point operations differently, and that put the operations in different order due to differences in the compilers.

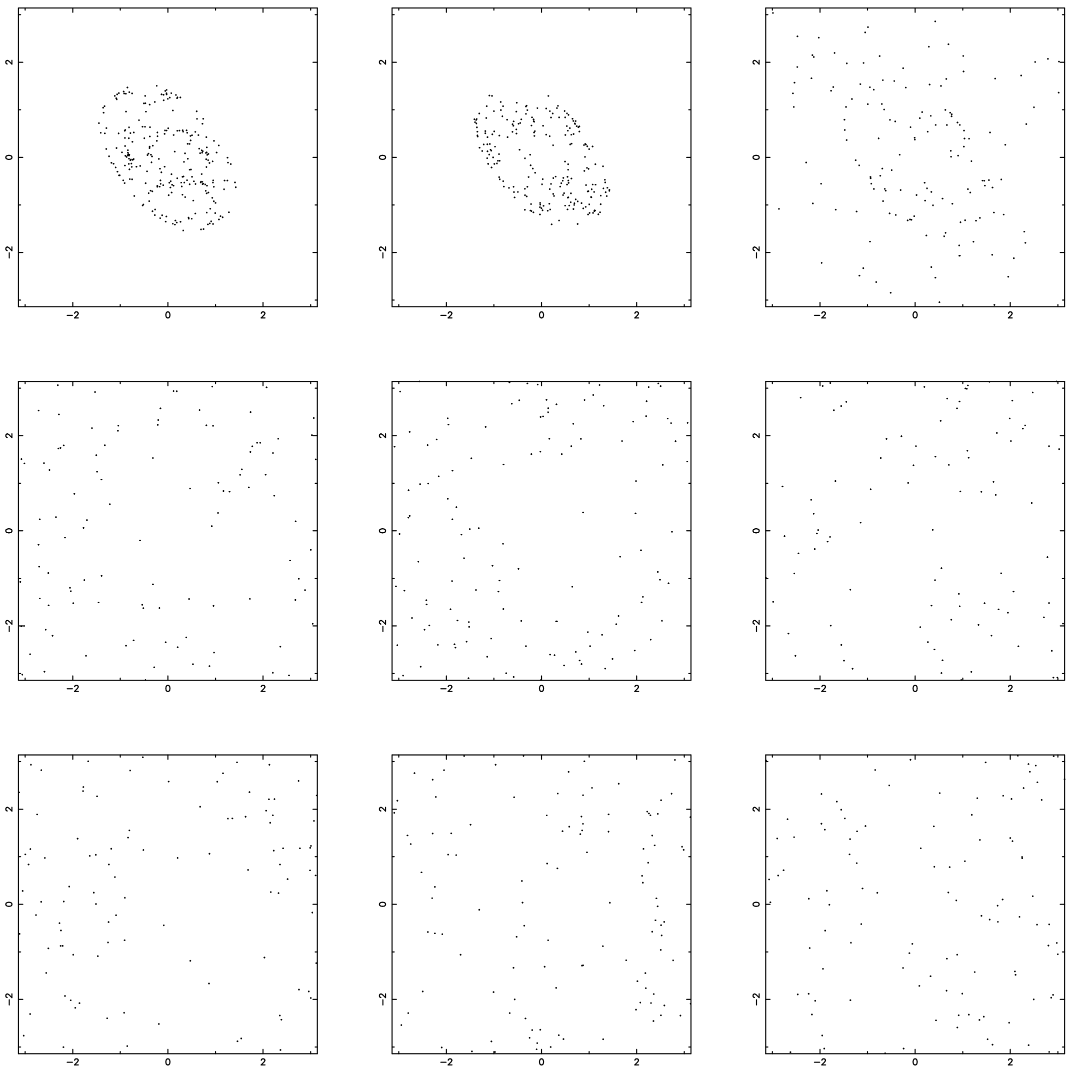


Fig. 10a: Median slice ($|z| < 0.01$) for the orbit of fig. 6 with $b = 0.275$ for the real mapping. In each square are plotted the points of 40000 successive iterations.

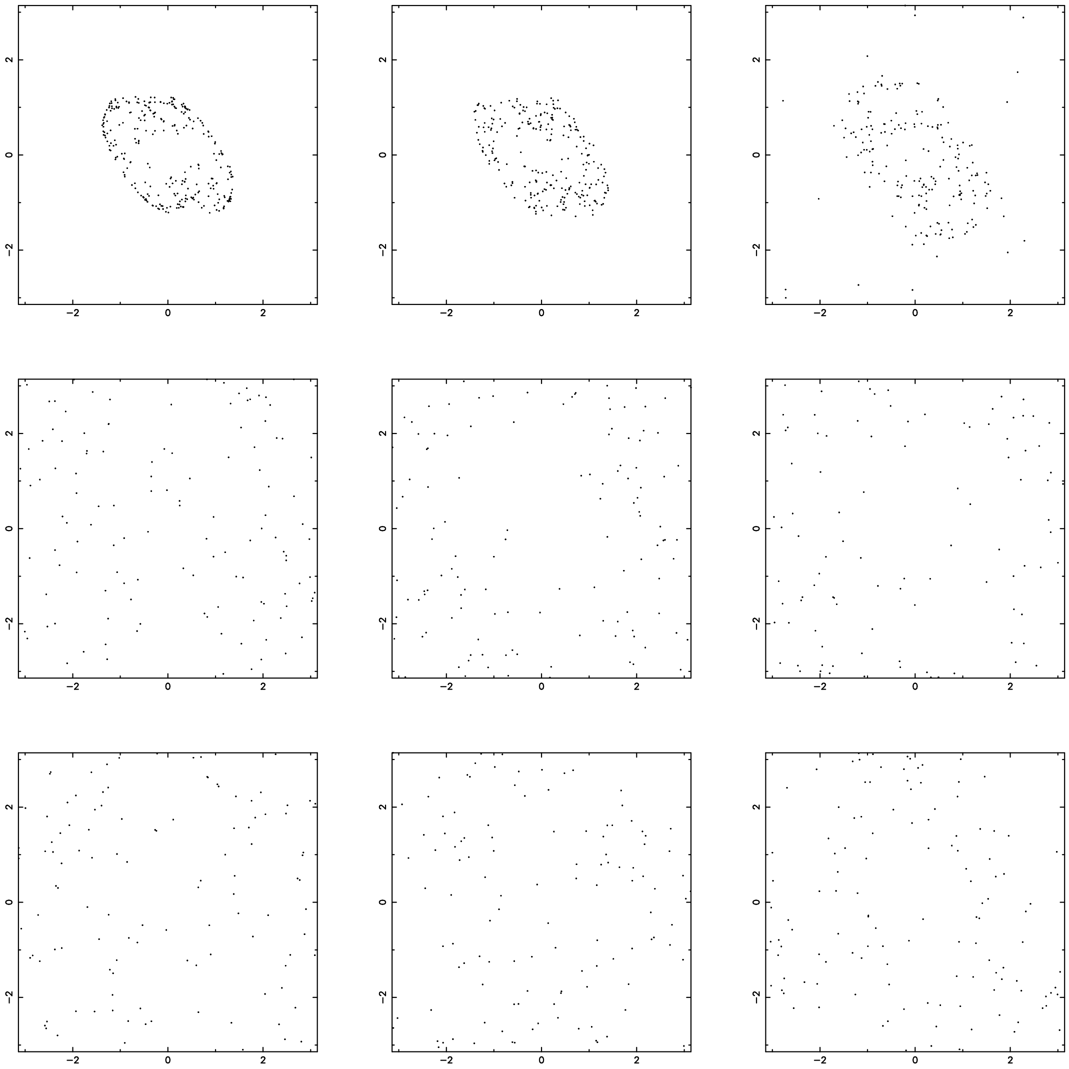


Fig. 10b: Same as fig. 10a for the synthetic mapping T3.

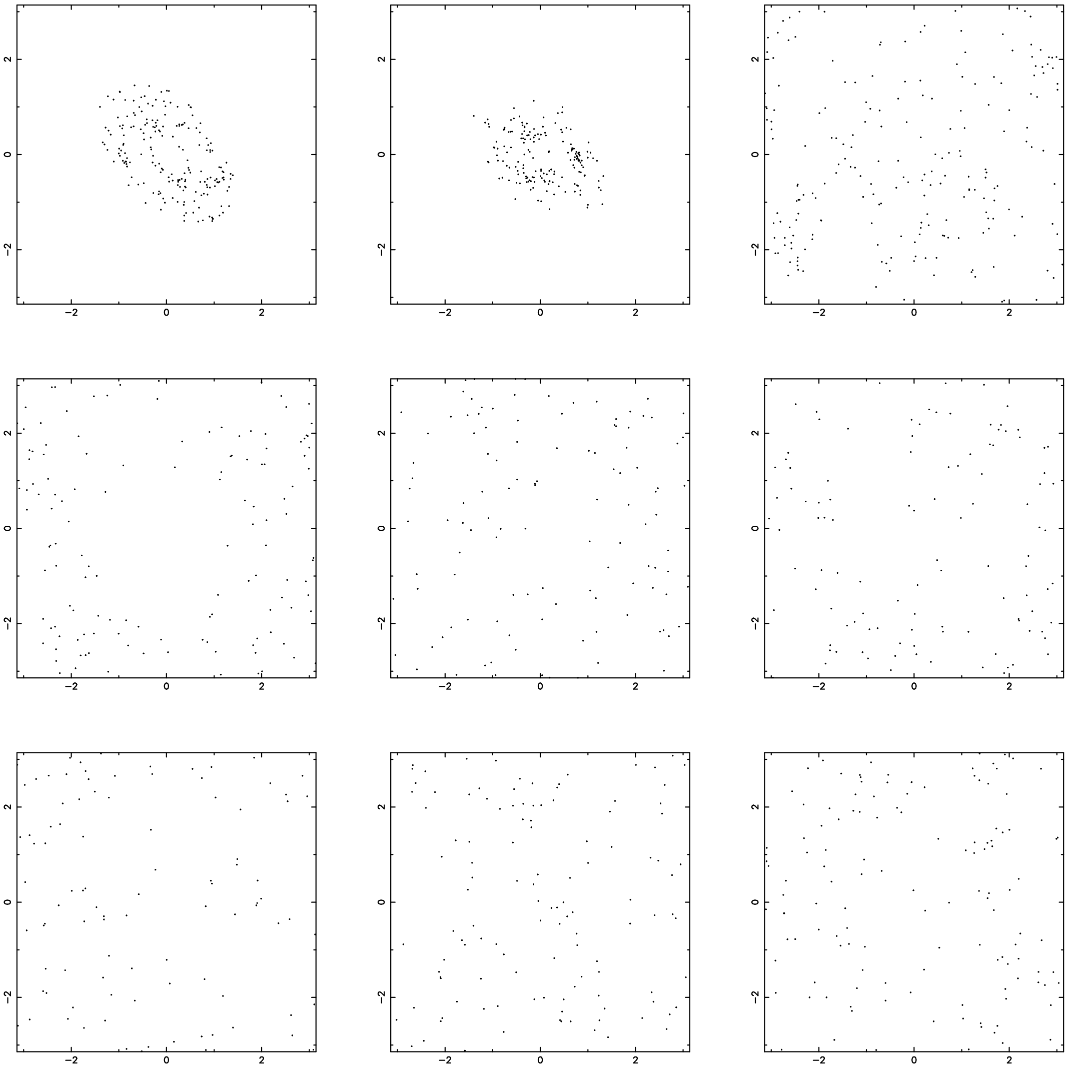


Fig. 10c: Same as fig. 10a for the synthetic mapping $\tau 3$.

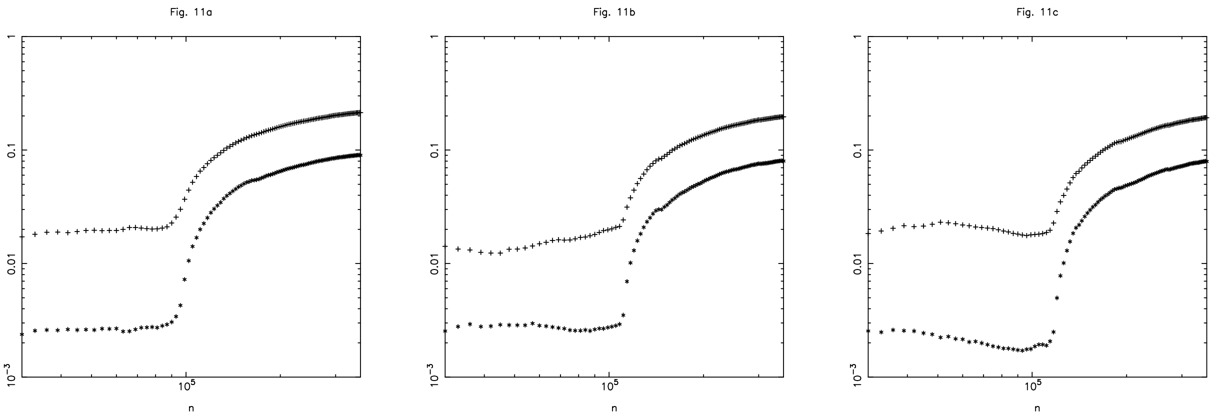


Fig. 11a: Positive Lyapounov characteristic numbers for the same orbit as in fig. 10a, for the real mapping. **Fig. 11b:** Same as fig. 11a for the synthetic mapping T3. **Fig. 11c:** Same as fig. 11a for the synthetic mapping $\tau 3$.

6 Conclusions

Synthetic maps appear to be valuable tools for celestial mechanics. Depending on the dynamical constraints of the problem like border zones, different interpolations can be used. Even for problems with more than two degrees of freedom, where the situation is less straightforward the use of a hash function has allowed us to overcome difficulties associated with the task of storing and recalling the values of the computed images at the vertices.

Acknowledgments

The authors want to thank Dr. R. Grauer for stimulating discussions at the outset of this research. They also thank Dr. G. Contopoulos for useful comments during the writing of the final version of this paper.

Appendix A

We first describe the algorithm for finding a value in an ordered linked-list. We have records composed of labelled data. We store and retrieve data according to the value of the label. The basic idea is to directly reference locations in a table by doing arithmetic transformations of labels into table addresses.

The first step in a search using hashing is to compute a *hash function* which transforms the search label into a table address. Almost no hash function is perfect, and two or more labels might hash to the same table address: the second part of a hashing search is a *collision resolution* process which deals with such labels. The collision resolution method we describe here uses linked lists, and is appropriate in a highly dynamic situation where the number of search labels cannot be predicted.

Hashing is a good example of a "time–space tradeoff". If there were no memory limitations, then we could do any search with only one memory access by simply using the label as a memory address. If there were no time limitations, then we could get by with only a minimum amount of memory by using a sequential search method. Hashing provides a way of using a reasonable amount of memory and time to strike a balance between these two extremes. Efficient use of available memory and fast access to the memory are prime concerns of any hashing method.

Hash functions

The first problem we must address is the computation of the hash function which transforms labels into table addresses. This is an arithmetic computation with properties similar to those of random number generators. What is needed is a function which transforms labels (here integers derived from the indices of points in the grid) into integers in the range $[1, M]$ where M is the number of entries in the table. An ideal hash function is one which is easy to compute and approximates a "random" function: for each input, every output should be "equally likely". The method we use here is, for any label k, to compute $h(k) = 1 + k \bmod M$. This is a straightforward method which is easy to compute in many environments and spreads the label values out well.

Separate chaining

The hash function above will convert labels into table addresses: we still need to decide how to handle the case when two labels hash to the same address. The most straightforward method is to simply build a linked list, for each table address, of the records whose labels hash to that address. Since the labels which hash to the same table position are kept in a linked list, they might as well be kept in order. This leads directly to a generalization of the elementary list searching method. Rather than maintaining a single list with a single list header node, we maintain M lists with M list header nodes. Obviously, the amount of time required for a search depends on the length of the lists (and the relative positions of the labels in them).

Details can be provided by the authors on request.

References

[1] Duncan, M., Quinn, T., & Tremaine S., 1989. Icarus **82**, p. 402.

[2] Froeschlé, C., 1972. Astron. Astrophys. **16**, p. 172.

[3] Froeschlé, C., & Petit, J.M., 1990. Astron. Astrophys. **238**, pp. 413–423.

[4] Froeschlé, C., & Petit, J.M., 1992. Celestial Mechanics **57**, pp. 123–130.

[5] Froeschlé, C., & Rickman, H., 1988. *CelestialMechanics.* **43**, p. 265.

[6] Gonczi, R., & Froeschlé, C., 1981. Celestial Mechanics **25**, p. 271.

[7] Hénon, M., 1981. in *Cours des Houches,* **XXXVI**, North Holland, Amsterdam, p. 57.

[8] Knuth, D., 1975. in *The art of computer programming: sorting and searching,* **3**, Addison Wesley, p. 506.

[9] Varosi, F., Grebogi, C., & Yorke, J.A., 1987. Physics Letters A **124**, p. 59.