CERN SL/93-06 (AP)

A Procedure to Compute the Fixed Points and Visualize the Orbits of a 2D Map

M.N. Vrahatis, G. Servizi, G. Turchetti*, T. Boutis**

Abstract

The accurate computation of periodic orbits of nonlinear mappings and the precise knowledge of their properties are very important for studying the behaviour of many dynamical systems of physical interest. In this paper we describe a new software package, called **giotto**, which visualizes orbits of nonlinear mappings of dynamical systems and computes all the periodic orbits (stable and unstable) of any period and accuracy which occur in the study of beam dynamics in particle accelerators.

*Dipartimento di Fisica, Universita di Bologna and INFN Sezione di Bologna (Italy)
** Department of Mathematics, University of Patras (Greece)

Geneva, Switzerland
1 February, 1993

SL/Div Reps

1. Introduction

The nonlinear betatronic motion in a large hadron accelerator is conveniently described by the one–turn map on a given reference section. The geometry of orbits in phase space and their stability properties are consequently analyzed by iterating the map (tracking) or by constructing its normal form[2–15].
An effective way of tracking the orbits is provided by interactive graphics procedures on a workstation, since the regions where the most interesting or intriguing features appear can be localized and explored to the desired accuracy. In this respect the knowledge of fixed points is relevant since they determine the local features of the orbits; indeed the Taylor expansion of the map at a fixed point allows a local normal form analysis[2–15].
Implementing a viewer of 4D maps combined with a fixed points finder and normal forms algorithms is a long range program. Here we describe an efficient algorithm to compute the fixed points and the interactive procedure **giotto** to visualize the orbits of a 2D mapping. The inclusion of the interactive computation of normal forms is in progress and the extension to 4D mapping under investigation.

Fixed points

Given a two dimensional map[16,22,23]

$$F : \begin{cases} x' = f_1(x,y) \\ y' = f_2(x,y) \end{cases} \tag{1.1}$$

we say that $\mathbf{x} = (x,y)$ is a fixed point of F if $F(\mathbf{x}) = \mathbf{x}$ and a fixed point of order q (or a periodic orbit of period q) if

$$\mathbf{x} = F^q(\mathbf{x}) \equiv F(F(\ldots(F(\mathbf{x}))\ldots)) \qquad (q \text{ times}) \tag{1.2}$$

In general analytic expressions are available only if the map is a polynomial of low degree and the period is low. On the other hand it is difficult to find in the literature efficient methods for computing high period orbits if the map is not decomposable into involutions[17,22,23]. Also, traditional iterative schemes such as Newton's method and related classes of algorithms[28,ch. 7] often fail since they converge to a fixed point almost independently of the initial guess, while there exist several fixed points, close to each other, which are all desirable for the applications[1]. Moreover these methods are affected by the mapping evaluations taking large values in neighborhoods of hyperbolic fixed points. Finally, in general, these methods often fail due to the nonexistence of derivatives or poorly behaved partial derivatives[28].
In this report we describe an efficient algorithm based on the topological degree theory to provide a criterion for the existence of a fixed point within a given region. More specifically the method constructs a polyhedron in such a way that the value of the topological degree of an iterate of the mapping, relative to this polyhedron, be ± 1, which implies the existence of a fixed point within this polyhedron. Then it repeatedly subdivides this polyhedron in such a way that the new refined polyhedron also retains the property of the existence of a fixed point within its interior, without any computation of the topological degree. These subdivisions take place iteratively, until a fixed point is computed to a predetermined

accuracy. When a fixed point is computed, the method iterates the mapping to obtain all the fixed points of the same period to the same accuracy.
This algorithm becomes especially significant for the computation of high period fixed points (elliptic and hyperbolic) where other more traditional approaches (like Newton's method, etc.) cannot easily distinguish among closely neighboring fixed points.
Here we shall illustrate this method on Hénon's quadratic mapping[23,22,16]

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \mathsf{R}(\alpha) \begin{pmatrix} x \\ y + f(x) \end{pmatrix}, \tag{1.3}$$

where

$$f(x) = -x^2, \qquad \mathsf{R}(\alpha) = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix},$$

to compute fixed points for given values of α and various periods.
The criterion for the existence of a fixed point using the characteristic polyhedron and details for its construction can be found in [31,32,35]; here we point out that the initial guess of such a polyhedron is easier and fast if an interactive viewer is available:

giotto is an interactive viewer of the orbits of an area preserving map. It has been designed in such a way that the user has only to provide a FORTRAN or C function to compute the map and includes, in its library, the algorithm for the search of fixed points.
The viewer allows magnifications of any given region in phase space so that the fixed points of interest can be localized providing a good initial guess of the characteristic polyhedron.
The use of giotto is simple: the beginning of a session is guided by a series of questions to provide the basic parameters and the graphic session is menu oriented for all the desired operations such as changing colour, magnifying a region, erasing orbits, searching fixed points, writing a PS file to dump the screen for later printing.

The plan of this report is the following: in section 2 we define the topological degree and illustrate the construction of the characteristic polyhedron, in section 3 we describe the bisection method to refine the result, in section 4 we analyze the application of the method to the 2D Hénon map, in section 5 we describe the viewer giotto and in section 6 we provide an example of its use.

2. The topological degree.

In this section we shall implement topological degree theory to give a criterion for the existence of a fixed point within a given region. This criterion is based on the construction of a "characteristic polyhedron" within a scaled translation of the unit cube. So, in the sequel, by reviewing the concept of a characteristic polyhedron we briefly present a procedure for the construction of this polyhedron. The theoretical development of the concepts employed here can be found in[31,32,35].
Bisection methods for finding roots of systems of equations depend on a "criterion" whose result can guarantee that a root will lie within a given region; then this region can be

subdivided in such a way that the result of the criterion can be again applied within the new refined region.
In one dimension, this criterion consists in examining the signs of the function evaluations at the endpoints of the given region. Specifically, if we desire to locate a root of the equation $f(x) = 0$ in the interval (a, b), where $f : (a, b) \subset \mathbf{R} \rightarrow \mathbf{R}$ is continuous, we can examine whether the following relation is fulfilled:

$$\text{sgn } f(\alpha)\text{sgn } f(\beta) < 0 \tag{2.1}$$

where sgn is the well known sign function with values

$$\text{sgn } \psi = \begin{cases} -1 & \psi < 0 \\ 0 & \psi = 0 \\ 1 & \psi > 0 \end{cases} \tag{2.2}$$

and if so, then we certainly know that there is at least one solution within (α, β) (see figure 1). The above criterion is known as Bolgano's criterion.

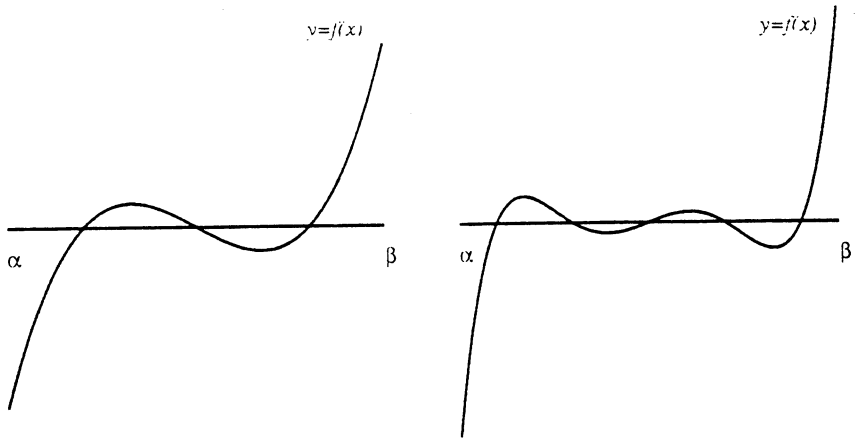


Fig.1:Bolgano's criterion for the existence of at least one root of $f(x) = 0$ within (α, β)

Instead of Bolgano's criterion we can also use the following criterion

$$\deg[f, (a, b), 0] = \frac{1}{2}\{\text{sgn } f(b) - \text{sgn } f(a)\}. \tag{2.3}$$

Now if the value of $\deg[f, (a, b), 0]$ is not zero we know with certainty that there exists at least one solution in (a, b).
The $\deg[f, (a, b), 0]$ is called *topological degree* of f at zero, relative to (a, b)[28]. Note that if the value of $\deg[f, (a, b), 0]$ is not zero, then the Bolgano's criterion is fulfilled. Also the value of $\deg[f, (a, b), 0]$ gives additional information concerning the behaviour of the roots of $f(x) = 0$ in (a, b), relative to the slopes of $f(x)$. For example, if $\deg[f, (a, b), 0] = 1$, which means that $f(b) > 0$ and $f(a) < 0$, then the number of roots at points where $f(x)$

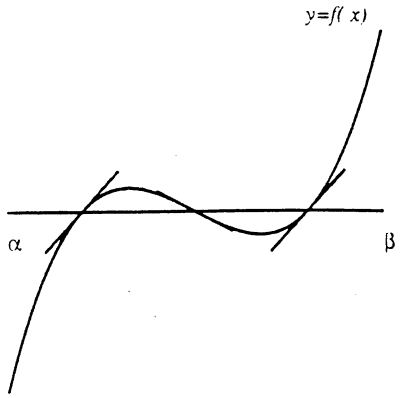


Fig.2:The slopes of $f(x)$ at roots

has a positive slope exceeds by one the number of roots at points where $f(x)$ has a negative slope[25] (see figure 2).
Now, using the value of the topological degree, we are able to calculate a root of $f(x) = 0$ by bisecting the interval (a, b). So we subdivide (a, b) into two intervals $(a, c][c, b)$ where $c = (a + b)/2$ is the midpoint of (a, b) and we keep the subinterval for which the value of topological degree is not zero, relative to itself. In this way we keep the existence of at least one root within a smaller interval. Now we can continue this procedure to approximate a root until the endpoints of the final collected subinterval differ from each other less than a prefixed amount.
This method (called *bisection method*) can be written as follows[19,20,21,31,32,34,35]:

$$x_{n+1} = x_n + \text{sgn } f(a)\text{sgn } f(x_n)(b-a)/2^{n+1}, \qquad x_0 = a, \qquad n = 0, 1, \ldots \tag{2.4}$$

A straightforward extension of Bolgano's criterion to two dimensions is to consider a box such that the function $f_1(x, y)$ may possess opposite signs at opposite sides and the function $f_2(x, y)$ also has opposite signs at the remaining sides of the square (see figure 3)
Then if this criterion is satisfied there is at least one solution of the system

$$\begin{aligned} f_1(x, y) &= 0 \\ f_2(x, y) &= 0 \end{aligned} \tag{2.5}$$

within the given box. This square, which is known as Miranda's box[27,33] can be generalized in the case of higher dimensions to provide the criterion of existence. The meaning of Miranda's box is evident if we consider a sufficiently small neighborhood $\mathcal{A}$ of a point $\mathbf{x}_*$ where (2.5) is satisfied with a non–vanishing jacobian, so that no other solution of (2.5) exists in $\mathcal{A}$. In this case $\xi_1 = f_1$ and $\xi_2 = f_2$ define a system of curvilinear coordinates, with origin at $\mathbf{x}_*$, subdividing $\mathcal{A}$ into four quadrants whose boundaries are the curves $f_1 = 0$, $f_2 = 0$. Joining any point in the first $(f_1 > 0, f_2 > 0)$, second $(f_1 > 0, f_2 < 0)$, third $(f_1 < 0, f_2 < 0)$ and fourth $(f_1 < 0, f_2 > 0)$ quadrant one obtains a Miranda's box which obviously contains the origin. However, although Miranda's box gives a criterion

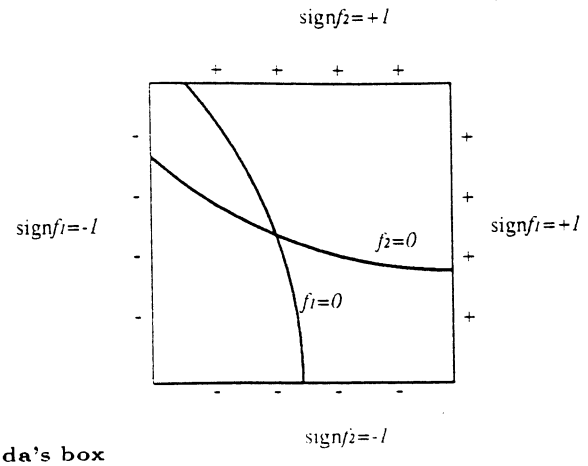


Fig.3:Miranda's box

with certainty for the existence of a root, its construction is very often impossible and cannot be applied for the computation. In overcoming these difficulties we give a "suitable" polyhedron whose construction guarantees the existence of at least one root in its interior. Before presenting the concept of this polyhedron, let us give a generalization of the topological degree to two dimensions, on which various generalized methods of bisection are based. According to these methods one establishes the existence of at least one root of the system (2.5) where $F = (f_1, f_2) : \mathcal{D} \subset \mathbf{R}^2 \rightarrow \mathbf{R}^2$ is continuous on the closure $\overline{\mathcal{D}}$ of $\mathcal{D}$ such that $F(\mathbf{x}) \neq 0$ for $\mathbf{x}$ on the boundary $\partial\mathcal{D}$ of $\mathcal{D}$, by computing the value of the topological degree of F at $\Theta = (0,0)$ relative to $\mathcal{D}$. This is denoted by $\deg[F, \mathcal{D}, \Theta]$ and can be defined by the following sum

$$\deg[F, \mathcal{D}, \Theta] = \sum_{\mathbf{x} \in F^{-1}(\Theta)} \text{sgn } \det J_F(\mathbf{x}) \tag{2.6}$$

where $\det J_F(\mathbf{x})$ indicates the determinant of the jacobian matrix. Now, if a nonzero value of $\deg[F, \mathcal{D}, \Theta]$ is obtained, then by Kronecker's existence theorem[28,p.161] it follows that there is at least one root of the system within $\mathcal{D}$. On the other hand, if $\deg[F, \mathcal{D}, \Theta] = 0$, no conclusion can be drawn because more information about F is needed[28].
However, although the nonzero value of $\deg[F, \mathcal{D}, \Theta]$ plays an important role in the existence of a root, its exact value is useless, since it does not give any additional information about the existence of the root of system (2.5). Moreover the computation of $\deg[F, \mathcal{D}, \Theta]$ is a time–consuming procedure and cannot be accurately achieved unless the modulus of continuity of F on $\mathcal{D}$ is known. For more details about degree theory we refer the reader to[28].
In what follows, we give the concept of a characteristic polyhedron. To define it, we have to give some other tools. Consider a number i with n binary digits

$$0 \leq i = b_0^i + 2b_1^i + \cdots + 2^{n-1} b_{n-1}^i \leq 2^n - 1 \tag{2.7}$$

where $b_j^i = 0, 1$ and the matrices $\mathcal{M}_n$ defined by

$$[\mathcal{M}_n]_{i+1,j} = 2b_j^i - 1 \tag{2.8}$$

For $n = 1$ and $n = 2$ we have respectively

$$\mathcal{M}_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \mathcal{M}_2 = \begin{bmatrix} -1 & -1 \\ -1 & 1 \\ 1 & -1 \\ 1 & 1 \end{bmatrix} \tag{2.9}$$

If $n = 1$ consider the segment $[x_1, x_2]$ and evaluate the sign of $f(x)$ at the end points: if the matrix

$$S(f; x_1, x_2) = \begin{bmatrix} \text{sgn} f(x_1) \\ \text{sgn} f(x_2) \end{bmatrix} \tag{2.10}$$

agrees with $\mathcal{M}_1$, up to a permutation of the rows, then we say that $[x_1, x_2]$ is a characteristic polyhedron.
For $n = 2$ consider four points $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$, $\mathbf{X}_4$, vertices of a quadrilateral box (polyhedron) as shown in figure 4. Letting $F = (f_1, f_2)$ be a map of the plane we consider the matrix of signs of the functions f_1, f_2 at the vertices of the polyhedron defined by

$$S(F; X_1X_2X_3X_4) = \begin{bmatrix} \text{sgn}\, F(X_1) \\ \text{sgn}\, F(X_2) \\ \text{sgn}\, F(X_3) \\ \text{sgn}\, F(X_4) \end{bmatrix} = \begin{bmatrix} \text{sgn}\, f_1(X_1) & \text{sgn}\, f_2(X_1) \\ \text{sgn}\, f_1(X_2) & \text{sgn}\, f_2(X_2) \\ \text{sgn}\, f_1(X_3) & \text{sgn}\, f_2(X_3) \\ \text{sgn}\, f_1(X_4) & \text{sgn}\, f_2(X_4) \end{bmatrix} \tag{2.11}$$

The polyhedron $X_1X_2X_3X_4$ is called a characteristic polyhedron relative to F if the matrix of signs $S(F; X_1X_2X_3X_4)$ is identical with the 2-complete matrix $\mathcal{M}_2$. For example the polyhedron in figure 4 is characteristic, while the polyhedron in figure 5 isn't.

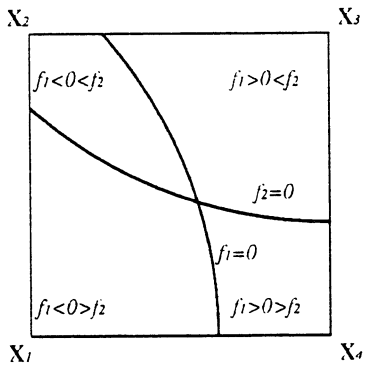


Fig.4:A characteristic polyhedron

The polyhedron $ABCD$ in figure 5 is not characteristic, since the entry $(-1, 1)$ of $\mathcal{M}_2$ is missing. Note that Miranda's box in figure 3 is a characteristic polyhedron.
Now if the boundary of $AECD$ is "sufficiently refined"[26,35] we know with certainty that there exists at least one solution within $AECD$, since the absolute value of the topological degree relative to a characteristic polyhedron is always equal to ± 1[35].

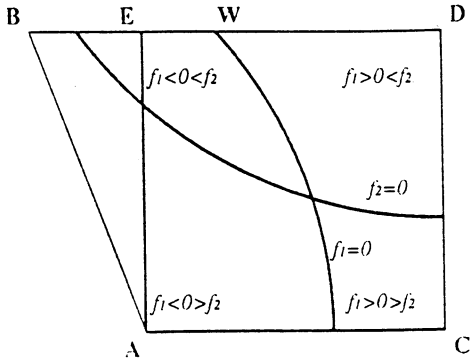


Fig.5: $ABCD$ **is not characteristic;** $AECD$ **is.**

The construction of a characteristic polyhedron starts with an arbitrary rectangle $ABCD$ which can be defined by the vertex $A = (x^0, y^0)$ and two stepsizes $h_1 = C - A$ and $h_2 = B - A$. Thus, in this case, the vertices of the rectangle $ABCD$ are:

$$A = (x^0, y^0), B = (x^0, y^0 + h_2), C = (x^0 + h_1, y^0), D = (x^0 + h_1, y^0 + h_2). \tag{2.12}$$

Now to construct a characteristic polyhedron we compare the matrix of signs $S(F; ABCD)$ with the matrix $\mathcal{M}_2$. If they are identical, then $ABCD$ is a characteristic polyhedron; otherwise we have to find out points $\mathbf{x}^*$ in $\mathbf{R}^2$ such that their vectors of signs produce the rows of $\mathcal{M}_2$ that are missing in $S(F; ABCD)$. In the case of figure 5 we observe that the row $(-1, 1)$ is missing in $S(F; ABCD)$ so we need a point $\mathbf{x}^* = (x^*, y^*)$ such that $(\operatorname{sgn} f_1(\mathbf{x}^*), \operatorname{sgn} f_2(\mathbf{x}^*)) = (-1, 1)$ which, in the case of the example of figure 5, is the point E. To find a point E we can easily compute the point W which is the root of $f_1 = 0$, lying on the edge BD. So, by holding the second component fixed and equal to $y^0 + h_2$, we can solve the one-dimensional equation

$$f_1(r, y^0 + h_2) = 0 \tag{2.13}$$

for r in the interval (B, D). Suppose that r^* is such a solution; then the point W is given by

$$W = (r^*, y^0 + h_2). \tag{2.14}$$

Now, if we change the first component of W, say

$$W = (r^* \pm \delta, y^0 + h_2) \tag{2.15}$$

by an arbitrary amount δ, we are in a position to compute a point E for the construction of a characteristic polyhedron. We can use any one of the well known one–dimensional methods to solve equation (2.15); here we use the traditional one–dimensional bisection method since frequently the edges of $ABCD$ are very long and also since a few significant digits for the computation of the root of the equation (2.15) are required. Moreover this

method has an additional advantage, since the minimum number of iterations required for obtaining an approximate root, with accuracy ϵ in some interval with length h, is known and given by

$$\mu = \lceil \log(h\epsilon^{-1}) \rceil \tag{2.16}$$

(here the notation $\lceil \cdot \rceil$ refers to the smallest integer not less than the real number quoted).

3. The bisection method.

In this section we briefly describe our method for the accurate computation of a single fixed point. This method is particularly useful in cases where the period of the fixed point is very high, since it always converges within the initial specified region. Moreover, this algorithm is very efficient since the only computable information that is required is the algebraic signs of the components of the mapping. Thus it is not affected by the mapping evaluations taking large values in neighborhoods of unstable fixed points. The theoretical development of the method can be found in[31,35].

This method is based on the refinement of a characteristic polyhedron and is called *characteristic bisection*. Specifically, the method bisects a characteristic polyhedron in such a way that the new refined polyhedron is also a characteristic one. To do this the method computes the midpoint of an edge of the characteristic polyhedron and replaces with such point that vertex of the polyhedron for which the vectors of their signs are identical. For example, as we see in figure 6, one can subdivide

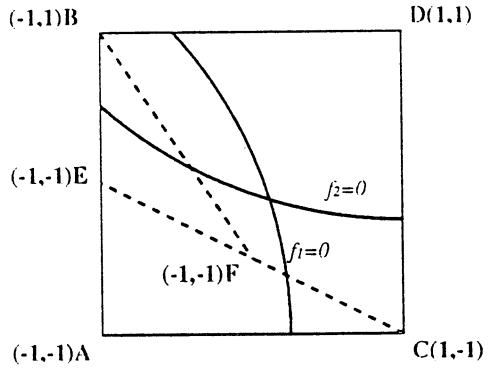


Fig.6:Characteristic bisections applied to $ABCD$. The polyhedra $ABCD$, $EBCD$ and $FBCD$ are characteristic

the polyhedron $ABCD$ in the following way: starting from the edge CD we find its midpoint E and calculate its vector of signs, which in this case is $(-1,1)$. So the vertex C is replaced by E in such a way that the new refined polyhedron $EBCD$ is also a characteristic one. Using the same procedure we are able to refine this polyhedron again by considering the midpoint F of EC and checking the vector of signs of this point. In this case its vector of signs is $(-1,1)$, hence the vertex E can be replaced by the vertex F, so that the new refined polyhedron $FBCD$ is still a characteristic one. This procedure continues until the

midpoint of the longest diagonal of a refined polyhedron approximates the root within a predetermined accuracy. Details of the above method can be found in[31,35].
The number of characteristic bisections of a characteristic polyhedron with length of its longest edge δ that is required for obtaining a new refined characteristic polyhedron with length of the longest edge smaller than ϵ is given by

$$\nu = \lceil \log_2(\delta\epsilon^{-1}) \rceil \tag{3.1}$$

(see [22] for a proof).

4. Fixed points of 2D Hénon's map.

We now give a procedure for computing all the fixed points of powers of a nonlinear mapping to any predetermined accuracy by applying the method of the previous section. The method is illustrated here on the following system of 2 equations in 2 unknowns:

$$\mathsf{T}: \begin{aligned} x_{i+1} &= x_i \cos\alpha - (y_i + x_i^2)\sin\alpha \\ y_{i+1} &= x_i \sin\alpha + (y_i + x_i^2)\cos\alpha \end{aligned} \tag{4.1}$$

with $x_n = x_0$, $y_n = y_0$ and α constant. The mapping T, expressed by the above system of equations, is known as Hénon's mapping and can be written as follows:

$$\mathsf{T} = \mathsf{R} \cdot \mathsf{S} \tag{4.2}$$

where S is a shearing parallel to the y axis:

$$x_i' = x_i, \qquad y_i' = y_i + x_i^2 \tag{4.3}$$

and R is a rotation by the angle α:

$$\begin{aligned} x_{i+1} &= x_i' \cos\alpha - y_i' \sin\alpha \\ y_{i+1} &= x_i' \sin\alpha + y_i' \cos\alpha \end{aligned} \tag{4.4}$$

A point $\mathbf{x}^*$ is called a periodic fixed point of T, with period P, $P \in N$ if $\mathsf{T}^P(\mathbf{x}^*) = \mathbf{x}^*$[29].
Such periodic orbits are identified by their rotation number $\sigma = \nu/2\pi = m_1/m_2$ where ν is the frequency of the orbit and m_1, m_2 are two positive integers[2]ch.7
Now to compute a fixed point of a particular period P, we consider the following operator

$$\mathsf{F} = \mathsf{T}^P - \mathsf{I}, \tag{4.5}$$

where I is the identity mapping, and solve the following system of equations

$$\mathsf{F}(x) = \Theta = (0,0) \tag{4.6}$$

To do this, we choose a starting point

$$\mathbf{x}^0 = (x^0, y^0) \tag{4.7}$$

and two stepsizes in each coordinate direction

$$H = (h_1, h_2) \tag{4.8}$$

in such a way that the corresponding constructed box (see figure 5) forms a domain in which the method will attempt to locate and compute a root of the system $\mathsf{F}(\mathbf{x}) = \Theta$, which is a fixed point of the mapping T^P.
Suppose now that a fixed point x_1^* has been computed within a predetermined accuracy ϵ such that

$$\|\mathsf{T}^P(\mathbf{x}_1^*) - \mathbf{x}_1^*\| \leq \epsilon. \tag{4.9}$$

Then, in order to compute all the other fixed points $\mathbf{x}_i^*$, $i = 2, \ldots, P$ with the same period and accuracy ϵ we iterate the mapping T as follows: first we obtain an approximation of the next fixed point $\mathbf{x}_2^*$ given by $\mathsf{T}(\mathbf{x}_1^*)$ and check if the following relationship is fulfilled:

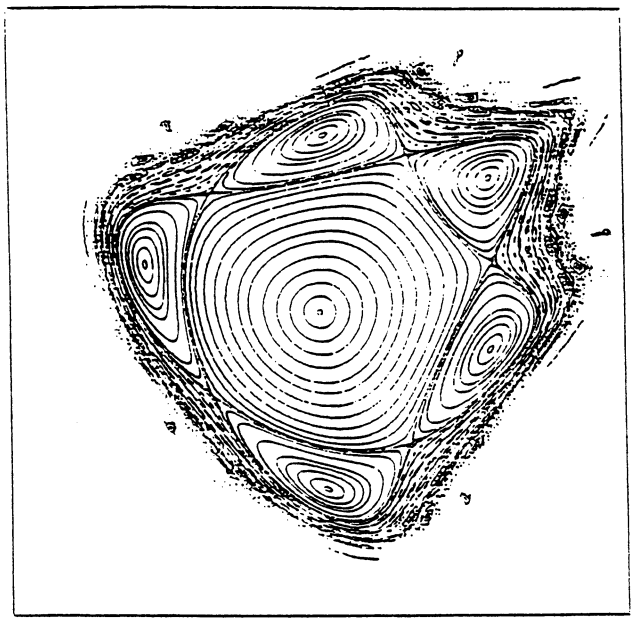
$$\|\mathsf{T}^P(\mathsf{T}(\mathbf{x}_1^*)) - \mathsf{T}(\mathbf{x}_1^*)\| \leq \epsilon. \tag{4.10}$$

If so, then we choose $\mathbf{x}_2^* = \mathsf{T}(\mathbf{x}_1^*)$; otherwise we choose $\mathsf{T}(\mathbf{x}_1^*)$ as initial guess for $\mathbf{x}_2^*$ and refine the result until the desired accuracy is reached. The above procedure is applied repeatedly until the last fixed point x_P^* is obtained.
Let us now illustrate the above procedure on Hénon's mapping for a given value of α and various periods P. Thus, taking e.g. $\alpha = cos^{-1}(0.24)$ we can see in the phase plot of Hénon's mapping (see figure 7)
that there is a chain of five big islands around the origin. So, in this case, we can search for five elliptic and five hyperbolic fixed points of period $P = 5$ and rotation number $\sigma = 1/5$. We could also choose, of course, other values of α. The reason for this choice is that it produces a large region of stability around the origin and thus may be of interest to applications in beam dynamics.
Now, in order to find the elliptic fixed points of this period, we choose one island, include it into a box by taking appropriate values for $\mathbf{x}^0$ and H,

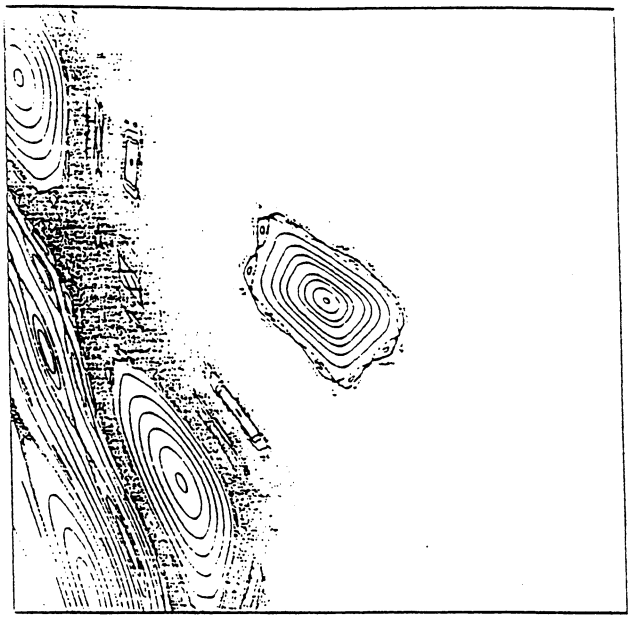
$$\mathbf{x}^0 = \begin{pmatrix} x^0 \\ y^0 \end{pmatrix} = \begin{pmatrix} 0.3 \\ -0.3 \end{pmatrix}, \qquad H = \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = \begin{pmatrix} 0.4 \\ 0.4 \end{pmatrix} \tag{4.11}$$

and apply the bisection method with accuracy $\epsilon = 10^{-16}$.
In this case, we compute the following fixed point utilizing around 6.5 msec of CPU time on the CERN VAX 9000-410 system:
$x_1^5 = 0.5672405470221847$ $\qquad$ $y_1^5 = -0.1223202134278941$

Of course, we can easily verify that this point is indeed an elliptic fixed point by applying the traditional technique of checking the eigenvalues of the linearized mapping[22]
Now, we can proceed with the computation of the rest of the elliptic fixed points of period five. To do this we can either apply the bisection method or iterate the mapping, since we have computed one of them within the predetermined accuracy of $\epsilon = 10^{-16}$. We prefer to do the latter and successively compute the following points
$x_2^5 = 0.5672405470221847$ $\qquad$ $y_2^5 = 0.4440820516139216$
$x_3^5 = 0.0173925844399303$ $\qquad$ $y_3^5 = 0.5800185952239573$

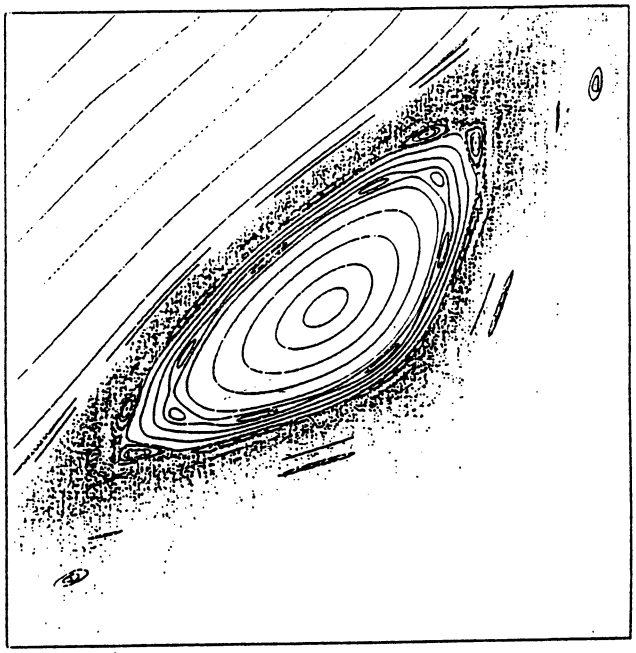


-1. 1.



0.7549 0.9387

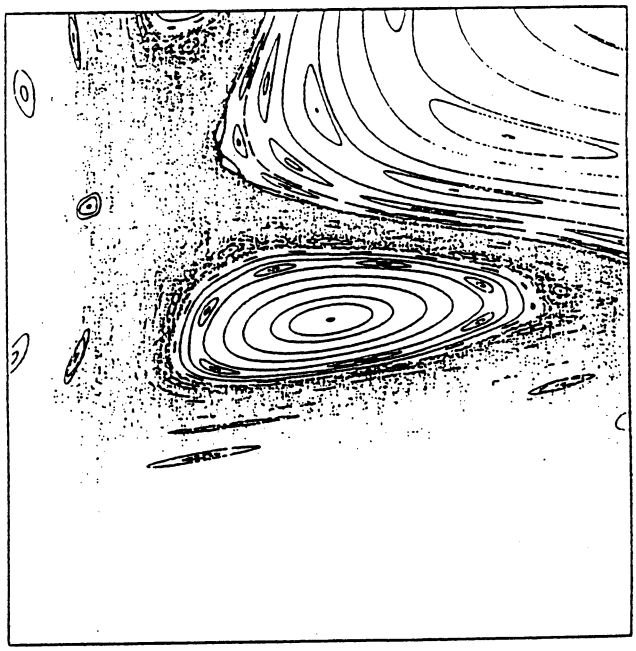
0.1386



0.1292

0.8637 0.8731

0.1325



0.1316

0.8652 0.8661

$x_4^5 = -0.5585984457571741$ $y_4^5 = 0.1560161118011652$
$x_5^5 = 0.0173925844399305$ $y_5^5 = -0.5797160932304572$
utilizing around 71 μsecs of CPU time on the same machine. Now to compute the hyperbolic fixed points of period $P = 5$ we choose the following initial values

$$\mathrm{x}^0 = \begin{pmatrix} x^0 \\ y^0 \end{pmatrix} = \begin{pmatrix} 0.1 \\ -0.7 \end{pmatrix}, \qquad H = \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = \begin{pmatrix} 0.4 \\ 0.4 \end{pmatrix} \tag{4.12}$$

and apply the bisection method according to the accuracy of $\epsilon = 10^{-16}$ to find one of them. In this case, we have computed the following point utilizing 6.5 msecs:
$x_1^5 = 0.2942106885737921$ $y_1^5 = -0.4274862418615337$
Then by iterating the mapping we compute the rest of the hyperbolic fixed points of period five, within the predetermined accuracy of $\epsilon = 10^{-16}$:
$x_2^5 = 0.5696326513533602$ $y_2^5 = 0.1622406787439296$
$x_3^5 = 0.2942106885737916$ $y_3^5 = 0.5140461711325987$
$x_4^5 = -0.3443814883177751$ $y_4^5 = 0.3882084578625210$
$x_5^5 = -0.3443814883177746$ $y_5^5 = -0.2696098483665559$
Let us now apply this procedure to compute fixed points of higher periods. Looking at the phase plot (figure 7) of Hénon's map for the same value of α as before, we are able to distinguish a chain of 16 islands around the central region of stability of the mapping. So, in this case, we look for 16 elliptic and 16 hyperbolic fixed points of period $P = 16$. To do this we choose one island and enclose it into a box by choosing appropriate values for x^0 and H^0.

$$\mathrm{x}^0 = \begin{pmatrix} x^0 \\ y^0 \end{pmatrix} = \begin{pmatrix} 0.8 \\ 0.1 \end{pmatrix}, \quad H = \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = \begin{pmatrix} 0.1 \\ 0.1 \end{pmatrix}. \tag{4.13}$$

Applying our generalized bisection method (with accuracy of $\epsilon = 10^{-16}$) we compute the following fixed point utilizing 11.1 msecs:
$x_1^{16} = 0.8504309709743801$ $y_1^{16} = 0.1490801034942473$
Now using the previous procedure we can compute the rest of the elliptic fixed points of period 16, whose rotation number is $\sigma = 3/16$. Proceeding in the same way we have been able to compute the points of several periodic orbits of higher period. Of course, islands of such periods can be visualized by enlarging the correponding vicinity of the (x, y) plane.

We end this section by presenting, in the concise form of Table 1 below, our results on some higher period, stable periodic orbits of Hénon's $2D$-mapping.

Table 1

Rotation number σ	x_1	y_1	CPU time
1/5	0.567240547	0.44408205161	6.5 msec
3/16	0.850430971	0.1490801035	11.1 msec
27/144	0.86850069	0.1341772209	46 msec
246/1296	0.865653945	0.1320418594	2.2 sec

Figure 7 gives for $\cos\alpha = .24$ the phase portrait of the map and its subsequent magnifications where the orbits around the fixed points listed in table 1 are visible. Finally we wish

to point out that our method of finding characteristic polyhedra is easily generalized to compute periodic orbits of $2N$-dimensional conservative mappings, with $N > 1$. In fact, we have already carried out such a generalization and computed some of the low order (period $P = 3, 4, 5, \ldots$) periodic orbits of the 4-dimensional quadratic mapping

$$\mathsf{T} = \begin{pmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \end{pmatrix} = \begin{pmatrix} \cos\alpha_1 & -\sin\alpha_1 & 0 & 0 \\ \sin\alpha_1 & \cos\alpha_1 & 0 & 0 \\ 0 & 0 & \cos\alpha_2 & -\sin\alpha_2 \\ 0 & 0 & \sin\alpha_2 & \cos\alpha_2 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 - x_1^2 + x_2^2 \\ x_2 \\ y_2 - 2x_1x_2 \end{pmatrix} \tag{4.14}$$

This is an interesting model, since mappings of this form are of direct relevance to the dynamics of particle beams (in a two–dimensional phase space) passing repeatedly through FODO cells of magnetic focusing elements[36]. In this context the computation of high period orbits can be quite useful in helping us establish the existence and study the structure of nearby invariant surfaces, which may be encompassing within them large regions of bounded orbits. More detailed results in this direction, however, are expected to appear in a future publication[37].

5. The viewer giotto.

giotto is an interactive 2D–graphic tool especially designed for computing and visualizing colour images of phase plots generated by mappings of the plane onto itself. It has been developed on Silicon Graphics IRIS and Hewlett Packard 9000/700 workstations but could, in principle, run on any computer supporting a 2D–graphics library and a C compiler with minor changes in the low–level interface between the program and the graphic firmware of the host machine, whose capabilities are fully exploited, as well as those arising from an efficient use of the mouse as an input device, allowing the user to choose operations and to give data to the program by simply clicking on menu buttons. giotto enables to visualize up to four different (and completely independent from each other) mappings on the same screen, to resize the window of the plane where images are drawn, to change colour choosing from a 256-colour palette, to erase unwanted trajectories, to find interactively fixed points of any iterate of the mapping, to save images in disk files for later use, to re-visualize saved images and to produce POST_SCRIPT code for both monochromatic and colour printing of the generated picture. These last two features are either built in the program or can be made possible by taking advantage of the tools usually installed in the workstation itself, such as the IRIS snapshot utility.

Writing the mapping

The user needs only to write a C function (or a FORTRAN integer function, or a procedure written in any supported language) which would be able to return to a calling program the values of the coordinates x and y after one iteration of the mapping(s). In other words, given a pair of real numbers (which we label as x_n and y_n) the function must compute a new pair (x_{n+1}, y_{n+1}) according to a recursion algorithm

$$x_{n+1} = f_1^{(i)}(x_n, y_n)$$
$$y_{n+1} = f_2^{(i)}(x_n, y_n)$$

where the label i identifies which one among four possible algorithms should be used. We illustrate the parameters which enter the function by providing a specific example where four maps are computed. These mps correspond to (1.3) for different nonlinearities: the Hénon map, the gaussian and discontinuous beam-beam maps and a Zaslavski–like map.

The function quoted below is written in FORTRAN, but it could be written in C as well

```
      integer function mappa_ ( Xc, Param, Imap, N )
      implicit real*8 (a-h, o-z)
      real*8 Xc(*), Param(10,4), Nonlin
      integer done
      common /static/ Rc, Rs, Alpha, Wind, done
      x = Xc(1)
      y = Xc(2)
      if (done .ne. 1) then
                Wind = Param(1,1) * 2.0 * 3.14159265358
                Rc = cos( Wind )
                Rs = sin( Wind )
                Alpha = Param(2,1)
                done = 1
      endif
      go to (1,2,3,4), I
1                       Nonlin = Alpha*x*x
      go to 5
2                       Nonlin = 1-exp(-Alpha*x*x)
      go to 5
3                       Sign=1.0
                       If(x.lt.0.) Sign=-1.0
                       Nonlin = Alpha*Sign
      go to 5
4                       Nonlin = (1.0 - cos(Alpha*x*x))/Alpha
C
5      Xnew = Rc * x + ( Nonlin + y ) * Rs
      Ynew = -Rs * x + ( Nonlin + y ) * Rc
      Xc(1) = Xnew
      Xc(2) = Ynew
      end
```

The function takes as its arguments the following:

1) a two–entries array Xc(*) of real numbers in double precision which on input holds the values of x_n and y_n respectively and on output the computed values of x_{n+1} and y_{n+1};
2) a two indices array of real double precision numbers Param(10,4) storing up to 10 real parameters for each of the mapping(s)
3) the integer Imap labelling the algorithm to be used among the four eventually available;

4) the integer N, the current index of iteration allowing computation of non autonomous maps for which the algorithms f_1 and f_2 do explicitly depend on the iteration number:

$$x_{n+1} = f_1^{(i)}(x_n, y_n; n)$$
$$y_{n+1} = f_2^{(i)}(x_n, y_n; n)$$

It is possible to define less than four maps. The user doesn't need to initialize neither the Imap nor the N and Xc(1), Xc(2); this is done directly by giotto once a click of the mouse takes place on a given pixel of the screen: this implies resetting N to zero, Imap to the number (from one to four) corresponding to the quarter of the screen where the mouse was clicked on and Xc(1)= x_0, Xc(2)= y_0 to the image of the screen coordinates Sx and Sy of that point through the obvious transformations

$$x_0 = x_{min} + (x_{max} - x_{min}) \frac{Sx - Sx_{min}}{Sx_{max} - Sx_{min}}$$
$$y_0 = y_{min} + (y_{max} - y_{min}) \frac{Sy - Sy_{min}}{Sy_{max} - Sy_{min}}$$

where the *min*'s and *max*'s are meant as minimum and maximum of the world and screen windows, the former being chosen by the user and the latter by giotto. Hence the only argument of the function that must be initialized by the user is the forty–entries array of real parameters for the mapping. Of course not the whole array needs to be initialized but only those entries that will be actually used during the computation of the algorithms f_1 and f_2 (may be none). The initialization of the array is made either interactively, letting giotto itself ask for parameters and answering as needed, or through another user–defined C function (possibly a FORTRAN one) which can load the array in the way one wishes, even opening and reading disk files owned by the user and created by any other application. Such a function should however exist (even if as an empty function) also in the case the user had chosen to answer at giotto's inquiring, because missing it could prevent the linker from producing the executable code for the program. The arguments of this last function are, of course, the same forty–entries array which appeared in the first one and an integer four–entries array where the user can (if he/she wishes) store how many parameters have been initialized for each of the four algorithms f_1 and f_2. This information, if present, is used by giotto to report in the POST_SCRIPT code the numerical values of the parameters used, producing in this way a printed picture completed with a short memo of the values which generated that image. Here it follows a FORTRAN sample of this second function which initializes the real parameters used in the previously quoted sample mappa_ to give values for the variables wind and alpha.

```
      integer function input_mio_( Param, N )
      real*8 Param(10,4), datum
      integer N(*)
      write(*,*) ' value of omega '
      read (*,*) datum
      do 1 m=1, 4
1                 Param(1,m) = datum
```

```
      write(*,*) ' value of alpha '
      read (*,*) datum
      do 2 m=1, 4
      N(m) = 2
2                 Param(2,m) = datum
      end
```

Finally we stress that the names of the two user-defined functions are mandatory: they are respectively equal to the two lowercase strings "mappa_" and "input_mio_". Care should be taken, when writing them in a language other than C, about the ending underscore. For instance, the FORTRAN compiler of the IRIS workstation adds, for some mysterious reason, an underscore character to the names of functions which will undergo a binding with C calling programs; hence in this case the user should call his/her FORTRAN functions simply "mappa" and "input_mio" WITHOUT the underscore! This behaviour wasn't noted in the Hewlett-Packard workstations.

6. Executing giotto.

When **giotto** starts executing it asks for some numerical data such as the number of mappings, the world's window coordinates, the number of iterations to be done, the "overflow" value beyond which the iterations would stop and (eventually) the parameters the mapping(s) depend on. These data are given by typing their values on the workstation's keyboard and all have (except for the parameters of the mapping) a default value that can be retained by simply typing the "enter" key instead of a number. At the end of this initial input section **giotto** opens a graphic window on the screen consisting of a square black (in the default colour palette) region, eventually subdivided in quarters, with a magenta-coloured column at its right side. This column, we refer to as the "menu tablet", is in turn subdivided in six boxes, each marked with a keyword corresponding to a particular function of **giotto**. From now on every operation of the program is driven by a mouse click on a proper box of the "menu tablet" and the square black region itself could be thought as a seventh box corresponding to the "default option" of the menus currently displayed. There are indeed several different "menu tablets" which do appear when **giotto** wants a decision to be taken: each of them has a different colour to help the user both to associate every colour to a particular set of functions and to notice, through the change in colour of the "menu tablet", that **giotto** is waiting for some choice.

Run time: Once the user has written his/her functions he/she can run **giotto** by simply giving the names of the two files where they are contained as two parameters of the command "**giotto**". Of course he/she must either specify the full pathname of the command or add it to his/her PATH shell variable. The pathname depend on the installation and should be asked to the system manager. Assuming that the PATH shell variable is correctly set and that the two functions have been written in C language and are contained in files "mappa.c" and "input.c", the command line

```
giotto mappa.c input.c
```

will start the execution. First of all the user is asked whether he/she wants help or not. This is intended to remind a "more–than–beginner" user some features he/she had forgotten. The best way to get help from giotto we suggest to a "true–beginner" is to run the program, while monitoring on another close screen the output of the command

```
giotto help
```

The keyword "help" is not understood as a file name; the program enters "help mode" instead, displaying on the screen a keyword–selected information on all its main features. In this way the "beginner" can try directly on the other screen the procedures he/she has just learnt and after a few minutes and even less mistakes (we hope) he/she will become familiar with giotto. And since the best way to speak about a graphic package is to show how it works and to use it in front of a workstation's screen, we stop here its "off–line" presentation.

Test run: We end presenting a test run for the computation and visualization of a pure Hénon quadratic map of winding number $\omega/2\pi = 0.21$ on the workstation Hewlett–Packard 720 installed at CERN with TCP–IP number 128.141.6.163 using all the default values of giotto. First of all it is necessary either to add the path

```
/usr/local/bin
```

to the PATH shell variable of the user or to specify the full pathname of the command

```
/usr/local/bin/giotto
```

to start the program. Assuming that the user had written the two files input.f and mappa.f and typed the command given in the previous section by typing

```
giotto input.f mappa.f
```

one compiles, links and generates the executable program which is intantly executed. If two C functions were written namely mappa.c and input.c then the same result is obtained with

```
giotto input.c mappa.c
```

It is furthermore allowed mixed use of both languages, say

```
giotto input.c mappa.f
```

The execution begins with a series of questions to which one can answer by typing the *enter* key. Any numerical input can be changed by simply typing the value one wishes; in particular, if the user answers with any number instead of *enter* at the question "choose options:", giotto will prompt the user also for the bounds of the "world window" and for the physical parameters of the mapping. In this case the user can specify a window not necessarily centered at the origin and moreover she/he doesn't need anymore to write a "true" function "input_mio_". The price to be payed, especially when several mappings

are being computed, is a somewhat ennoying input sequence because it is necessary to give four numbers for each window rather than only one. If any error occurred during the compilation of the user's functions or the linking process the executable code is not produced and an error message is issued. The user is informed of the type of error by means of text files left in the current directory on exit from the command: if only compilation errors occurred they are reported in the file schifezze; if the compilation was successful, but some inconsistency was detected at linking level the corresponding system error message is reported in the file CASOTTO; the user should find the bug of her/his logic, make the necessary corrections and try again the command. At the end of the run the executable code can be (optionally) saved in the executable file GIOTTO (please note the capital letters) and run again by simply typing the command

GIOTTO

Please remember, however, that the executable code GIOTTO is bound to the function mappa_ that was linked with the lowercase command giotto.

Acknowledgements.

One of us (MNV) wishes to thanks the Italian Ministery of Foreign Affairs for its support, as well as the Physics Department of the University of Bologna, INFN and CERN for their hospitality and permission to use their computing facilities, both of which proved very helpful for the completion of this work.

References.

1. Allgower E.L., Jeppson M.M. "The approximation of solutions of nonlinear elliptic boundary value problems having several solutions", Springer Lecture Notes 333, (1973), 1-20.
2. Bazzani A., Todesco E., Turchetti G., Servizi G., "Normal forms for symplectic maps and nonlinear theory of betatronic motion", CERN Yellow Report (to appear).
3. Servizi G., Turchetti G. "Perturbative expansion for area preserving maps" Nuovo Cimento B95, (1986), 121.
4. Bazzani A., Turchetti G., Mazzanti P., Servizi G. "Normal forms for hamiltonian maps and nonlinear effects in particle accelerators" Il Nuovo Cimento B102, (1988), 51.
5. Bazzani A., Leemann B., Scandale W., Servizi G., Turchetti G. "Description of nonlinear beam dynamics in the CERN large hadron collider by using normal form algorithms" in European Particle Accelerators Conference (EPAC 1988) Ed. S. Tazzari, World Scientific (1989).
6. Bazzani A., Giovannozzi M., Servizi G., Turchetti G., Todesco E. "Normal forms for area preserving maps resonances and dynamical aperture" CERN SPS/89-24 (AMS) (1989).
7. Bazzani A., Turchetti G. "Normal form results for the 1988 dynamical aperture experiment" CERN SPS/89-25 (AMS) (1989).
8. Servizi G., "Padé Approximants of normal forms for area preserving maps" Proc. of the Conf. "*Nonlinear problems in future particle accelerators*, World Scientific (1990).
9. Servizi G., Turchetti G. "Normal forms singularities in area preserving maps and analytic continuation by Padé Approximants", Phys. Letters 151A, (1990), 485.
10. Bazzani A., Servizi G., Todesco E., Turchetti G. Normal forms and invariants in the description of a magnetic lattice Nuc. Phys. Instr. and Meth. in Physics Research A298, (1990), 102.
11. Turchetti G., Bazzani A., Giovannozzi M., Servizi G., Todesco E. "Normal forms in symplectic maps and stability of beams in particle accelerators" in Dynamical Symmetries and Chaotic Behaviour in Physical Systems Ed. Maino, World Scientific (1990).
12. Turchetti G. "Beam stability analysis via normal forms and non perturbative methods" in Non Linear Problems in Future Particle Accelerators Ed. W. Scandale, G. Turchetti, World Scientific (1991).
13. Bazzani A., Turchetti G. "Singularities of normal formal forms and topology of orbits in area preserving maps" J. Phys. A25, (1992), L427.
14. Bazzani A., Turchetti G. "Perturbation theory and analyticity of normalizing transformations for area preserving maps" Conference Chaos: theory and practice, Patras (1991).
15. Bazzani A., Giovannozzi M., Servizi G., Turchetti G., Todesco E. "Resonant normal forms and stability analysis for area preserving maps" submitted to Physica D.
16. Bountis T.C., Helleman R.H.G., "On the stability of periodic orbits of 2-dimensional mappings", J. Math. Phys. 22, (1981) 1867.

17. De Vogelaere R., "On the structure of symmetric periodic solutions of conservative systems with applications". In S. Lefschetz ed., Contributions to the theory of nonlinear oscillations, (1958), 53. Princeton N.J.
18. Eiger A., Sikorski K., Stenger F., "A bisection method for systems of nonlinear equations". ACM Trans Math. Softw. 10, (1984), 367-377.
19. Grapsa T.N., Vrahatis M.N., "The implicit function theorem for solving systems of nonlinear equations in $\mathbb{R}^2$", Inter. J. Comp. Math. 28 (1989), 171.
20. Grapsa T.N., Vrahatis M.N., "A dimension–reducing method for solving systems of nonlinear equations in $\mathbb{R}^n$, Inter. J. Comp. Math. 32 (1990), 205.
21. Grapsa T.N., Vrahatis M.N., Bountis T.C., "Solving systems of nonlinear equations in $\mathbb{R}^n$ using a rotating hyperplane in $\mathbb{R}^{n+1}$ Inter. J. Comp. Math. 35 (1990), 133.
22. Helleman R.H.G., "On the iterative solution of a stochastic mapping". In U. Landman ed., Statistical mechanics and statistical methods in theory and application, 343.
23. Hénon M., "Numerical study of quadratic area–preserving mappings", Quart. Appl. Math. 27 (1969), 291.
24. Greene J.M., "A method for determining a stochastic transition", J. Math. Phys. 20 (1979), 1183.
25. Greene J.M., "Locating three–dimensional roots by a bisection method", J. Comp. Phys. 98 (1922), 194.
26. Kearfott B., "An efficient degree–computation method for a generalized method of bisection", Numer. Math. 32 (1979), 109.
27. Miranda C., "Un'osservazione su un teorema di Brouwer", Bol. Un. Mat. Ital. 3 (1940), 5.
28. Ortega J.M., Rheinboldt W.C., "Iterative solution of nonlinear equations in several variables", Academic Press, New York (1970).
29. Verhalst F., "Nonlinear differential equations and dynamical systems", Springer–Verlag, Berlin Heidelberg (1990).
30. Vrahatis M.N., "An error estimation for the method of bisection in $\mathbb{R}^n$", Bull. Soc. Math. Grèce 27 (1986), 161.
31. Vrahatis M.N., "Solving systems of nonlinear equations using the nonzero value of the topological degree", ACM Trans. Math. Software 14 (1988), 312.
32. Vrahatis M.N., "CHABIS: A mathematical software package for locating and evaluating roots of systems of nonlinear equations", ACM Trans. Math. Software 14 (1988), 330.
33. Vrahatis M.N., "A short proof and a generalization of Miranda's existence theorem", Proc. Amer. Math. Soc. 107 (1989), 701.
34. Vrahatis M.N., Bountis T.C., Bundinsky, "A convergence improving iterative method for computing periodic orbits near bifurcation points", J. Comp. Phys. 88 (1990), 1.
35. Vrahatis M.N., Iordanidis K.I., "A rapid generalized method of bisection for solving systems of nonlinear equations".
36. Bountis T., Tompaidis S., "Strong and weak instabilities in a 4D mapping model of FODO cell dynamics", in Future Problems in Nonlinear Particle Accelerators, eds. G. Turchetti, W. Scandale (World Scientific, 1990).

37. Vrahatis M.N., Turchetti G., Servizi G., Bountis T., "Periodic orbits and invariant surfaces in 4D mapping models of particle beams", in preparation.