UNIVERSITAT DE BARCELONA

Facultat de Matemàtiques
i Informàtica

GRAU DE MATEMÀTIQUES

Treball final de grau

The Parameterization Method for Quasiperiodic Invariant Circles

Autora: Martina Cormand Bayés

Director: Dr. Alex Haro Provinciale
Realitzat a: Facultat de Matemàtiques i Informàtica

Barcelona, 15 de gener de 2025

Contents

Abstract

This work examines two algorithms for computing an accurate approximation of a parameterization K of a quasiperiodic invariant circle of an area preserving map F. Both approaches rely on a Newton scheme to iteratively solve the conjugacy equation that characterizes the invariant circle.

The first algorithm is the parameterization method, reviewed in [À. Haro, M. Canadell, J. L. Figueras, A. Luque and J. M. Mondelo. (2016). *The parameterization method for invariant manifolds: From Rigorous Results to Effective Computations*. Springer]. This approach exploits the area preserving properties of F to simplify computations and solve the invariance equation $F \circ K = K \circ R_\omega$, where R_ω is the rotation of angle ω.

The second algorithm, the balanced parameterization method, is a novel approach that combines concepts from the parameterization method and the method reviewed in [D. Blessing and J. D. Mireles James. (2024). Weighted Birkhoff averages and the parameterization method. *SIAM Journal on Applied Dynamical Systems*, 23(3), 1766-1804]. Specifically, we introduce a dummy parameter β into the invariance equation, reformulating it as $F \circ K = (1+\beta)\, K \circ R_\omega$. This idea, following the strategy of the parameterization method, is combined with the exploitation of the area preserving properties of F, significantly improving computational efficiency. In particular, the method reviewed in [D. Blessing and J. D. Mireles James. (2024). Weighted Birkhoff averages and the parameterization method. *SIAM Journal on Applied Dynamical Systems*, 23(3), 1766-1804] has a computational cost of $O(N^3)$, where N denotes the number of Fourier modes used in the discretization of the problem, whereas our balanced parameterization method reduces the cost to $O(N \log\, N)$.

Finally, we illustrate the utility of both approaches using the Hénon map and the standard map as examples. By employing a numerical continuation scheme, with the rotation number serving as the continuation parameter, we compute a substantial number of quasiperiodic invariant circles in these systems.

2020 Mathematics Subject Classification. 37-04, 37D10, 37J40, 37M21

Resum

Aquest treball presenta dos algorismes per calcular una aproximació precisa d'una parametrització K d'un cercle invariant quasiperiòdic d'una aplicació F que preserva l'àrea. Ambdós enfocaments es basen en el mètode de Newton per resoldre iterativament l'equació de conjugació que caracteritza el cercle invariant.

El primer algorisme és el mètode de parametrització, revisat a [À. Haro, M. Canadell, J. L. Figueras, A. Luque i J. M. Mondelo. (2016). *The parameterization method for invariant manifolds: From Rigorous Results to Effective Computations*. Springer]. Aquest enfocament aprofita les propietats de preservació de l'àrea d'F per simplificar els càlculs i resoldre l'equació d'invariància $F \circ K = K \circ R_\omega$, on R_ω és la rotació d'angle ω.

El segon algorisme, el mètode de parametrització balancejat, és un enfocament innovador que combina conceptes del mètode de parametrització i del mètode revisat a [D. Blessing i J. D. Mireles James. (2024). Weighted Birkhoff averages and the parameterization method. *SIAM Journal on Applied Dynamical Systems*, 23(3), 1766-1804]. Concretament, introduïm un paràmetre β a l'equació d'invariància, reformulant-la com a $F \circ K = (1+\beta)\, K \circ R_\omega$. Aquesta idea, seguint l'estratègia del mètode de parametrització, es combina amb l'ús de les propietats de preservació de l'àrea d'F, millorant significativament l'eficiència computacional. En particular, el mètode revisat a [D. Blessing i J. D. Mireles James. (2024). Weighted Birkhoff averages and the parameterization method. *SIAM Journal on Applied Dynamical Systems*, 23(3), 1766-1804] té un cost computacional de $O(N^3)$, on N denota el nombre de modes de Fourier utilitzats en la discretització del problema, mentre que el nostre mètode de parametrització balancejat redueix el cost a $O(N \log\, N)$.

Finalment, il·lustrem la utilitat d'ambdós enfocaments utilitzant el mapa de Hénon i el mapa estàndard com a exemples. Mitjançant un esquema de continuació numèrica, amb el nombre de rotació com a paràmetre de continuació, calculem un nombre significatiu de cercles invariants quasiperiòdics en aquests sistemes.

Agraïments

Vull expressar el meu més sincer agraïment al Dr. Alex Haro, per guiar-me i acompanyar-me durant aquests mesos de feina, i per ensenyar amb tant d'entusiasme i paciència. Gràcies per fer-me despertar una passió per les matemàtiques que pensava que ja no tenia dins.

Als amics i amigues que he conegut en aquesta etapa, gràcies per aquests meravellosos anys passats entre el claustre i la biblioteca, plens d'aprenentatges i complicitats. A la meva família, gràcies pel vostre suport incondicional i pels incomptables sopars que heu passat escoltant les meves penes i alegries matemàtiques.

Moltes gràcies a tots.

Chapter 1

Introduction

1.1 Existence and Persistence of Quasiperiodic Orbits

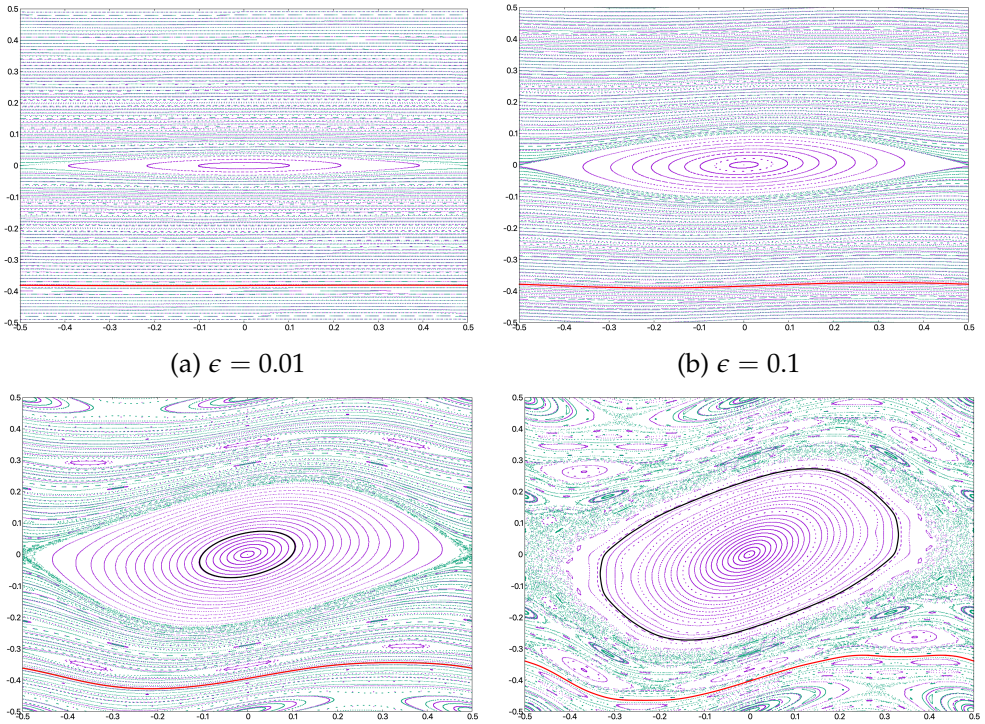
KAM theory, named after A.N. Kolmogorov, V.I. Arnold, and J.K. Moser, focuses on how small perturbations affect dynamical systems that have invariant tori carrying quasiperiodic motion. It shows that many of these invariant tori survive under small perturbations, as long as certain conditions on smoothness and non-degeneracy are satisfied. These invariant tori act as barriers that prevent chaotic motion, ensuring the system remains stable over time. KAM theory has applications in a wide range of fields where dynamical systems play a key role, such as Celestial Mechanics, Astrodynamics, Molecular Dynamics, Plasma Physics or Beam Physics.

KAM methods primarily address Hamiltonian dynamical systems that can be expressed as a perturbation of an integrable system. In this context, an integrable system refers to a Hamiltonian system that possesses a continuous family of invariant tori, on which the motion is quasiperiodic. As an example, consider the Chirikov standard map, given by

$$\begin{aligned} F_\epsilon : \quad & \mathbb{T}\times\mathbb{R} \longrightarrow \mathbb{T}\times\mathbb{R} \\ & (x,y) \longmapsto (x+y-\frac{\epsilon}{2\pi}\sin(2\pi x),\, y-\frac{\epsilon}{2\pi}\sin(2\pi x)), \end{aligned}$$

where ϵ is the perturbation parameter and $\mathbb{T} = \mathbb{R}/\mathbb{Z}$. The standard map is a discrete Hamiltonian system, so it is a symplectic map (and therefore area preserving). For $\epsilon = 0$, the system is integrable and its dynamics are simple: the orbit of any point $(x_0, y_0) \in \mathbb{T}\times\mathbb{R}$ lies in the torus $\{y = y_0\}$ and is a rotation of angle y_0, given by $F_0^n(x_0, y_0) = (x_0 + ny_0,\, y_0)$. We refer to these as rotational orbits. Note that if $y_0 = p/q \in \mathbb{Q}$ the corresponding orbit is periodic. On the contrary, if $y_0 \in \mathbb{R}\backslash\mathbb{Q}$ the orbit is dense in the invariant curve $\mathbb{T}\times\{y_0\}$. In any case, the orbit of a given point (x_0, y_0) has rotation number (or frequency) y_0 for every $x_0 \in \mathbb{T}$.

A key result of KAM theory is that many of these rotational invariant curves persist under a perturbation, provided the map preserves the area, the perturbation is sufficiently small, and the rotation numbers of the curves satisfy a diophantine condition (ensuring they are sufficiently "irrational"). A famous number satisfying such properties is the golden mean $(\sqrt{5}-1)/2$. These persisting curves are slightly deformed from their original shapes in the integrable system, but retain their quasiperiodic nature. A common approach in KAM theory is to analyze how, given a specific rotation number, the invariant curve associated with this rotation number persists as the perturbation parameter increases.



(a) $\epsilon = 0.01$ (b) $\epsilon = 0.1$

(c) $\epsilon = 0.5$ (d) $\epsilon = 0.9$

Figure 1.1: Dynamics of the standard map for various values of ϵ. As ϵ increases, the rotational invariant curves begin to break down, leading to the formation of chaotic regions in the phase space along with librational orbits. In red, we plot the rotational orbit with rotation number $\omega = (\sqrt{5}-1)/2 - 1$, and in black, the librational orbit with rotation number $\omega = 0.8883181985894327$.

Notice that, aside from rotational orbits, the perturbations with $\epsilon > 0$ also give rise to librational orbits, which loop on invariant circles that are topologically contractible to a point. These tori can form, for instance, around stable fixed points such as the origin, in the standard map. These orbits can also be assigned rotation

numbers and can be studied similarly to rotational orbits. The terms "rotational" and "librational" come from the fact that the standard map is a discretization of the pendulum equation. In the pendulum, librational orbits correspond to the pendulum swinging back and forth, while rotational orbits correspond to the pendulum spinning around its axis.

This work studies the existence of quasiperiodic objects (specifically librational orbits) using the parameterization method, which involves refining the approximation of the torus by adding a small correction function. This correction is determined by solving, approximately, the linearized invariance equation around the current torus estimate. The process results in a Newton-like iterative scheme for addressing the invariance equation. It is worth noting that the geometry of the problem significantly influences the resolution of these equations, simplifying the computations substantially.

Next, in this chapter, we formalize the concept of the rotation number and introduce the parameterization method. Chapter 2 discusses the geometric structures involved, and Chapter 3 delves into the algorithm of the parameterization method, following the ideas presented in [7] and reviewed in [1]. In Chapter 4 we present another approach to the parameterization method, adding to the argument followed in [2] by providing a more detailed analysis of the importance of the dummy parameter. This new method will be referred to as the balanced parameterization method. Finally, in Chapter 5 we provide numerical examples of the method's application by adapting and using the codes provided in [1] and [4].

1.2 Rotation Number

Let us now formalize the problem introduced in the previous section, starting by presenting some basic concepts that will lead us to the definition of the rotation number of a curve. The concept of the rotation number, though not explicitly named as such at the time, was introduced by Poincaré in his foundational studies of differential equations and dynamical systems. Poincaré examined the behavior of trajectories on periodic orbits and invariant circles, introducing the rotation number to measure the average rate of rotation of a point under repeated applications of a dynamical system.

Definition 1.1. Let $G : \mathbb{T} \to \mathbb{T}$ be a homeomorphism of the circle, where $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ is the one-dimensional torus. Let $R_\omega : \mathbb{T} \to \mathbb{T}$ be the *rotation of angle* ω, $R_\omega(\theta) = \theta + \omega \mod 1$. We say that G is *topologically conjugate* to the rotation R_ω if there exists a homeomorphism $h : \mathbb{T} \to \mathbb{T}$ such that

$$G \circ h = h \circ R_\omega. \tag{1.1}$$

If ω is irrational, we say that G is conjugate to an irrational rotation.

Definition 1.2. Let $\pi : \mathbb{R} \to \mathbb{T}$ denote the *canonical covering map* defined by

$$\pi(x) = x \mod 1,$$

which maps a real number x into $[0, 1)$ by discarding the integer part. We interpret $\theta \in [0, 1)$ as the angle describing a point on the one-dimensional torus. Note that $\pi(x + m) = \pi(x)$ for all $x \in \mathbb{R}$ and $m \in \mathbb{Z}$.

Definition 1.3. A continuous map $\hat{G} : \mathbb{R} \to \mathbb{R}$ is a *lift* of G if

$$(\pi \circ \hat{G})(x) = (G \circ \pi)(x), \qquad \forall x \in \mathbb{R}.$$

It can be shown that every continuous map of the circle has a lift.

Definition 1.4. Let $\hat{G}$ be a lift of G. A *lifted rotation number* of G is defined by

$$\hat{\omega}(\hat{G}) = \lim_{n \to \infty} \frac{\hat{G}^n(x) - x}{n}.$$

Poincaré proved that this limit exists and is independent of the base point x, making it well-defined for the lift $\hat{G}$. For any two lifts $\hat{G}_1$ and $\hat{G}_2$ of G, we have that $\hat{\omega}(\hat{G}_1) = \hat{\omega}(\hat{G}_2) + m$ for some $m \in \mathbb{Z}$.

Definition 1.5. Let $\hat{G}$ be a lift of G. The *rotation number* of G is defined by

$$\omega(G) = \pi(\hat{\omega}(\hat{G})).$$

Since the lifted rotation number is well-defined up to an integer, the rotation number of G is well-defined.

Remark 1.6. Throughout this work, we abuse notation and often refer to the rotation number and the lifted rotation number interchangeably.

The rotation number carries important dynamical meaning. If ω is a rational number, expressed as $\omega = p/q$ for some $p \in \mathbb{Z}$ and $q \in \mathbb{N}$, then the map G is a periodic orbit with period q. We focus on the case of ω being irrational, where the dynamics become more intricate. According to the Denjoy theorem, if ω is irrational and G is at least $\mathcal{C}^2$, then it is topologically conjugate to the rigid rotation map R_ω. In this case, every orbit of G is dense on the circle, reflecting a quasiperiodic motion without periodic points.

More detailed information on the rotation number can be found in [5], and for a detailed discussion on numerical methods to compute it refer to [6].

1.3 The Parameterization Method

Let us now introduce the formal context of the parameterization method. Take $\mathscr{A}$ as a subset of $\mathbb{R}^2$ with the standard symplectic structure. Let $F : \mathscr{A} \to \mathscr{A}$ be a map and $\mathscr{K}$ be a quasiperiodic invariant circle for F with a certain rotation number $\omega \in \mathbb{R}\backslash\mathbb{Q}$. Our goal is to find a suitable and regular parameterization $K : \mathbb{T} \to \mathscr{A}$ of the curve $\mathscr{K}$ such that the invariance equation is satisfied:

$$F \circ K = K \circ R_\omega. \tag{1.2}$$

Notice the similarity between this equation and the conjugacy equation (1.1). Since ω is irrational, the rotation R_ω is ergodic, and in this case $\mathscr{K} = K(\mathbb{T})$ is a quasiperiodic invariant torus. Our objective is to construct a fast algorithm to find a good approximation of the parameterization K of $\mathscr{K}$. So, in summary, the primary aim of this work is to develop and present algorithms for computing quasiperiodic invariant circles with a given rotation number.

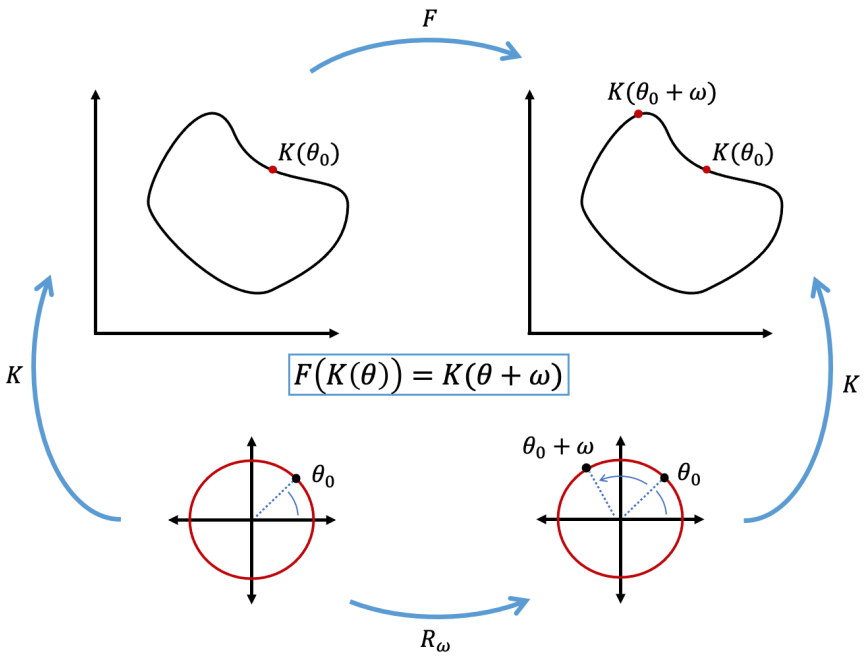


Figure 1.2: Topological conjugacy to rotation. The image of K remains invariant under F, meaning that $F \circ K$ represents a reparameterization of the curve K. Specifically, this reparameterization corresponds to a rotation by an angle ω. The diagram shows that if the invariance equation is satisfied, the dynamics induced by F on K are conjugate to the dynamics on the circle defined by R_ω.

To guarantee the existence of numerous invariant tori, the map F must preserve some structure. We focus on the preservation of a symplectic structure, which in two dimensions corresponds to the concept of a map being area preserving. This will be discussed in detail in Chapter 2. This approach to find invariant tori is known as the parameterization method. See [7] for the original presentation of the method, and Chapter 4 of [1] for a review.

The proof of the convergence of these algorithms is a matter of KAM theory, and it is thoroughly discussed in Chapter 4 of [1]. To summarize its main result, let $F : \mathscr{A} \to \mathscr{A}$ be an exact symplectic map (see the basic definitions in Section 2.2), with ω satisfying certain irrationality conditions. Let $K : \mathbb{T} \to \mathscr{A}$ be an immersion of the model torus $\mathbb{T}$, and define the error function $E : \mathbb{T} \to \mathbb{R}^2$ as

$$E = F \circ K - K \circ R_\omega.$$

In this setup, under specific non-degeneracy conditions, if E is sufficiently small in a given norm, there exists a true invariant torus whose parameterization satisfies equation (1.2).

Connecting this result to the methods that will be presented in Chapters 3 and 4, if our algorithms can iteratively refine a given curve K to make the norm of E small enough, then this theorem guarantees the existence of this invariant curve with a certain rotation number ω. And relating this back to the standard map, if the perturbation parameter ϵ is increased, this theorem can be applied to ensure the continuation of the invariant tori with the same rotation number for the perturbed map.

Chapter 2

Geometric Structures

This chapter explores geometric structures critical to the development of the parameterization method. Specifically, it examines symplectic, volume-preserving, and orientation-preserving systems. The relationship between symplecticity and volume preservation is shown, demonstrating that symplectomorphisms naturally preserve both volume and orientation. Particular attention is given to two-dimensional area preserving maps, which are the focus of this work.

2.1 Volume Preserving Maps and Symplecticity

Throughout this chapter, let $\mathcal{U}$ be an open subset of $\mathbb{R}^m$ and $\mathscr{A}$ be an open subset of $\mathbb{R}^{2n}$.

Definition 2.1. A map $F : \mathcal{U} \to \mathcal{U}$ is said to be *volume preserving* if for all open and bounded subsets $\mathcal{V} \subset \mathcal{U}$ we have that $Vol(\mathcal{V}) = Vol(F^{-1}(\mathcal{V}))$, where Vol is the volume function with respect to the Lebesgue measure.

Definition 2.2. A map $F : \mathcal{U} \to \mathcal{U}$ is *orientation preserving* if $\det DF(x) > 0 \ \forall x \in \mathcal{U}$.

Lemma 2.3. *Let $F : \mathcal{U} \to \mathcal{U}$ be a $\mathcal{C}^1$ diffeomorphism. Then F is volume preserving if and only if $|\det DF(x)| = 1 \ \forall x \in \mathcal{U}$. Moreover, if F is orientation preserving, then $\det DF(x) = 1 \ \forall x \in \mathcal{U}$.*

Proof. Since F is a diffeomorphism, it is injective and we can write the volume preserving condition as $Vol(\mathcal{V}) = Vol(F(\mathcal{V})) \ \forall \mathcal{V} \subset \mathcal{U}$, where $\mathcal{V}$ is an open and bounded subset. Notice that

$$Vol(\mathcal{V}) = \int_{\mathcal{V}} 1 \, dx \quad \text{and}$$
$$Vol(F(\mathcal{V})) = \int_{F(\mathcal{V})} 1 \, dx = \int_{\mathcal{V}} |\det DF(x)| \, dx.$$

If $|\det DF(x)| = 1 \ \forall x \in \mathcal{U}$, then F is clearly volume preserving.

To see the other implication, let us first see that if a map $f : \mathcal{U} \to \mathbb{R}$ is continuous and for all $\mathcal{V} \subset \mathcal{U}$ open and bounded subset we have that $\int_{\mathcal{V}} f(x)\, dx = 0$, then $\forall x \in \mathcal{U}\ f(x) = 0$. Let us assume that there exists $x_0 \in \mathcal{U}$ such that $f(x_0) > 0$ (similarly for $f(x_0) < 0$). Since f is continuous, there exists an open ball $\mathcal{V}_0 = B_\epsilon(x_0) \subset \mathcal{U}$ such that $f(x) > 0\ \forall x \in \mathcal{V}_0$. Therefore, $\int_{\mathcal{V}_0} f(x)\, dx > 0$, which is a contradiction. Now, taking $f(x) = |\det DF(x)| - 1$, we can deduce that $|\det DF(x)| = 1$.

Moreover, if F is also orientation preserving, then obviously $\det DF(x) = 1$ $\forall x \in \mathcal{U}$. □

Definition 2.4. A *symplectic structure* in $\mathscr{A}$ is given by a 2-form $\boldsymbol{\omega}$ satisfying the conditions:

(i) $\boldsymbol{\omega}$ is non-degenerate, i.e, $\forall z \in \mathscr{A}$ we have that $\forall u \in \mathbb{R}^{2n}$, with $u \neq 0$, there exists $v \in \mathbb{R}^{2n}$ such that $\boldsymbol{\omega}_z(u, v) \neq 0$.

(ii) $\boldsymbol{\omega}$ is closed, i.e., $d\boldsymbol{\omega} = 0$.

Moreover, we say that the symplectic structure is *exact* if

(iii) $\boldsymbol{\omega}$ is exact, i.e., $\boldsymbol{\omega} = d\boldsymbol{\alpha}$ for a certain 1-form $\boldsymbol{\alpha}$.

Since we are working with an open set of $\mathbb{R}^{2n}$, for any point $z \in \mathscr{A}$ we can identify the 2-form $\boldsymbol{\omega}_z$ with a 2n-dimensional matrix $\Omega(z) = (\Omega_{ij}(z))$, where $\Omega_{ij}(z)$ denotes the (i, j)-th component of $\Omega(z)$. Using components, the closedness condition (ii) can be written as

$$\text{(ii)} \quad \frac{\partial \Omega_{ij}}{\partial z_k} + \frac{\partial \Omega_{jk}}{\partial z_i} + \frac{\partial \Omega_{ki}}{\partial z_j} = 0.$$

Also, we can identify the 1-form $\boldsymbol{\alpha}_z$ with a 2n-dimensional vector $a(z) = (a_1(z) \dots a_{2n}(z))^\top$. In this case, condition (iii) can be written as

$$\text{(iii)} \quad \Omega(z) = Da(z)^\top - Da(z), \quad \text{or} \quad \Omega_{ij}(z) = \frac{\partial a_j}{\partial z_i} - \frac{\partial a_i}{\partial z_j} \quad \forall i, j = 1, \dots, 2n.$$

It can be shown that (iii) implies (ii) and that, if $\mathscr{A}$ is simply connected, then by Poincaré's lemma (ii) implies (iii).

Example 2.5. The most common example of a symplectic structure, and the one we will work with, is the standard symplectic structure in $\mathbb{R}^{2n}$, given by

$$\boldsymbol{\omega_0} = \sum_{i=1}^{n} dz_{n+i} \wedge dz_i,$$

$$\boldsymbol{\alpha_0} = \sum_{i=1}^{n} z_{n+i}\, dz_i.$$

The matrix representations of $\boldsymbol{\alpha_0}$ and $\boldsymbol{\omega_0}$ are, respectively,

$$a_0(z) = \begin{pmatrix} O_n & I_n \\ O_n & O_n \end{pmatrix} z \qquad \text{and} \qquad \Omega_0(z) = \begin{pmatrix} O_n & -I_n \\ I_n & O_n \end{pmatrix}.$$

2.2 Symplectomorphisms and Exact Symplecticity

This section introduces symplectomorphisms, which are transformations that maintain the symplectic structure. In this section, we define these transformations, introduce exact symplecticity, and explore their link to volume and orientation preservation, with a focus on planar systems.

Definition 2.6. A diffeomorphism $F : \mathscr{A} \to \mathscr{A}$ is a *symplectomorphism* if $F^*\boldsymbol{\omega} = \boldsymbol{\omega}$. In coordinates, this condition translates to

$$DF(z)^\top \Omega(z) DF(z) = \Omega(z), \quad \forall z \in \mathscr{A}. \tag{2.1}$$

Definition 2.7. A symplectomorphism $F : \mathscr{A} \to \mathscr{A}$ is said to be *exact* if there exists a smooth function $S : \mathscr{A} \to \mathbb{R}$ such that $F^*\boldsymbol{\alpha} - \boldsymbol{\alpha} = dS$. In coordinates, this is equivalent to

$$DS(z) = a(F(z))^\top DF(z) - a(z)^\top, \quad \forall z \in \mathscr{A}. \tag{2.2}$$

The function S is known as the *primitive function* of F.

Lemma 2.8. *Let $F : \mathscr{A} \to \mathscr{A}$ be a symplectomorphism with the standard symplectic structure, and suppose $\mathscr{A} \subset \mathbb{R}^{2n}$ is simply connected. Then F is exact symplectic.*

Proof. For the standard symplectic structure, take the $2n$-vector a as in example 2.5. We want to show that there exists a primitive function of F, that is, a function $S : \mathscr{A} \to \mathbb{R}$ satisfying equation (2.2). Writing $F = (f_1, \ldots, f_n)$, we have that the exact symplecticity condition is equivalent to the system of equations

$$\begin{cases} \dfrac{\partial S}{\partial z_i} = \displaystyle\sum_{k=1}^{n} f_{n+k} \frac{\partial f_k}{\partial z_i} - z_{n+i} \quad \text{for} \quad i = 1, \ldots, n & = P_i \\ \dfrac{\partial S}{\partial z_i} = \displaystyle\sum_{k=1}^{n} f_{n+k} \frac{\partial f_k}{\partial z_i} \quad \text{for} \quad i = n+1, \ldots, 2n & = P_i. \end{cases} \tag{2.3}$$

Now, we know that (2.3) is a Pfaffian differential equation system, and that a necessary condition to solve it is $\frac{\partial P_i}{\partial z_j} = \frac{\partial P_j}{\partial z_i}$ for all $i, j = 1, \ldots, 2n$. But by Poincaré's lemma, since $\mathscr{A}$ is simply connected, we know that this integrability condition is actually sufficient. We can write $\frac{\partial P_i}{\partial z_j}$ as

$$\begin{cases} \dfrac{\partial P_i}{\partial z_j} = \displaystyle\sum_{k=1}^{n} \frac{\partial f_{n+k}}{\partial z_j}\frac{\partial f_k}{\partial z_i} + \sum_{k=1}^{n} f_{n+k}\frac{\partial^2 f_k}{\partial z_i \partial z_j} - 1 \quad \text{for} \quad i = 1, \dots, n \quad \text{and} \quad j = n+i \\ \dfrac{\partial P_j}{\partial z_i} = \displaystyle\sum_{k=1}^{n} \frac{\partial f_{n+k}}{\partial z_j}\frac{\partial f_k}{\partial z_i} + \sum_{k=1}^{n} f_{n+k}\frac{\partial^2 f_k}{\partial z_i \partial z_j} \quad \text{otherwise.} \end{cases}$$

Using the Kronecker delta, we can write this as

$$\frac{\partial P_i}{\partial z_j} = \sum_{k=1}^{n} \frac{\partial f_{n+k}}{\partial z_j}\frac{\partial f_k}{\partial z_i} + \sum_{k=1}^{n} f_{n+k}\frac{\partial^2 f_k}{\partial z_i \partial z_j} - \delta_{j(n+i)},$$

where

$$\delta_{j(n+i)} = \begin{cases} 1 & \text{for} \quad 1 \leq i \leq n \quad \text{and} \quad j = n+i \\ 0 & \text{otherwise.} \end{cases}$$

Imposing the integrability condition, we obtain

$$0 = \frac{\partial P_i}{\partial z_j} - \frac{\partial P_j}{\partial z_i} = \sum_{k=1}^{n} \frac{\partial f_{n+k}}{\partial z_j}\frac{\partial f_k}{\partial z_i} - \sum_{k=1}^{n} \frac{\partial f_{n+k}}{\partial z_i}\frac{\partial f_k}{\partial z_j} - \delta_{j(n+i)} + \delta_{i(n+j)}.$$

This is equivalent to

$$\sum_{k=1}^{n} \left(\frac{\partial f_{n+k}}{\partial z_j}\frac{\partial f_k}{\partial z_i} - \frac{\partial f_{n+k}}{\partial z_i}\frac{\partial f_k}{\partial z_j} \right) = \delta_{j(n+i)} - \delta_{i(n+j)}.$$

With a few computations we can see that the left-hand side of this equation is $((DF)^\top \Omega_0 DF)_{ij}$, and the right-hand side is $(\Omega_0)_{ij}$. And this is exactly the symplecticity condition (2.1), so we conclude that F is exact symplectic. □

Lemma 2.9. *Let $F : \mathscr{A} \to \mathscr{A}$ be a symplectomorphism with the standard symplectic structure. Then F is volume and orientation preserving.*

Proof. This proof closely follows the argument presented in [8]. Taking determinants in equation (2.1), we obtain that $\det(DF(z)^\top DF(z)) = 1 \ \forall z \in \mathscr{A}$. Therefore, $\det DF(z) = \pm 1$, and we only have to show that $\det DF(z) > 0$. Consider the matrix

$$(DF)^\top DF + I. \tag{2.4}$$

Let us first show that the determinant of matrix (2.4) is greater than 1. Since $(DF)^\top DF$ is symmetric positive definite, its eigenvalues are real and positive. Therefore, there exists a matrix P satisfying $P^{-1} = P^\top$ such that $P^\top((DF)^\top DF)P = \Sigma$ is diagonal with positive entries. This implies that $\det((DF)^\top DF) = \det \Sigma > 0$.

Then, we can write $P^\top(DF^\top DF + I)P = \Sigma + I$, and therefore $\det(DF^\top DF + I) = \det(\Sigma + I) > 1$.

From the symplectivity condition (2.1), we have that $(DF)^{-T} = \Omega_0 DF \Omega_0^{-1}$. Using this, we can write

$$(DF)^\top DF + I = (DF)^\top (DF + (DF)^{-T}) = (DF)^\top (DF + \Omega_0 DF \Omega_0^{-1}).$$

Let A, B, C, D be the four $n \times n$ blocks of DF

$$DF = \begin{pmatrix} A & B \\ C & D \end{pmatrix}.$$

Then,

$$\begin{aligned} DF + \Omega_0 DF \Omega_0^{-1} &= \begin{pmatrix} A & B \\ C & D \end{pmatrix} + \begin{pmatrix} O_n & -I_n \\ I_n & O_n \end{pmatrix} \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} O_n & I_n \\ -I_n & O_n \end{pmatrix} \\ &= \begin{pmatrix} A + D & B - C \\ C - B & A + D \end{pmatrix}. \end{aligned}$$

Now, taking $E{:=}A + D$ and $F{:=}B - C$, we have that

$$\begin{aligned} DF + \Omega_0 DF \Omega_0^{-1} &= \begin{pmatrix} E & F \\ -F & E \end{pmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{pmatrix} I_n & I_n \\ iI_n & -iI_n \end{pmatrix} \begin{pmatrix} E + iF & O_n \\ O_n & E - iF \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} I_n & -iI_n \\ I_n & iI_n \end{pmatrix}. \end{aligned}$$

Using this factorisation and the fact that $\det((DF)^\top DF + I) > 1$,

$$\begin{aligned} 0 < 1 < \det((DF)^\top DF + I) &= \det((DF)^\top (DF + \Omega_0 DF \Omega_0^{-1})) \\ &= \det(DF) \det(E + iF) \det(E - iF) \\ &= \det(DF) \det(E + iF) \det(\overline{E + iF}) \\ &= \det(DF) \det(E + iF) \overline{\det(E + iF)} \\ &= \det(DF) |\det(E + iF)|^2. \end{aligned}$$

Therefore, $\det DF > 0$ and thus $\det DF = 1$, i.e. F is volume and orientation preserving. □

The following lemma establishes that, in the plane, symplecticity is equivalent to preserving both area and orientation. Therefore, in our development of the parameterization method for planar systems, it will suffice to assume that the map F is area and orientation preserving.

Lemma 2.10. *Let $F : \mathscr{A} \subset \mathbb{R}^2 \longrightarrow \mathscr{A} \subset \mathbb{R}^2$ be a diffeomorphism. Then with the standard symplectic structure, F is area and orientation preserving if and only if F is a symplectomorphism.*

Proof. We can easily see that, $\forall z \in \mathbb{R}^2$, we have

$$DF(z)^\top \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} DF(z) = \begin{pmatrix} 0 & -\det DF(x) \\ \det DF(z) & 0 \end{pmatrix}.$$

Therefore, the symplecticity condition $DF(x)^\top \Omega_0 DF(x) = \Omega_0$ is met if and only if $\det DF(x) = 1$. That is, if and only if F is area and orientation preserving. □

Chapter 3

The Parameterization Method

This chapter shows a first approach to the parameterization method for quasi-periodic orbits in area preserving maps in the plane, following the ideas presented in [1] but adapting them to the specific case of planar systems. In Section 3.4, following Newton's method, we linearize the invariance equation around the approximate curve. A key step in solving this linearized equation involves finding a suitable expression for the tangent map of F around the curve. To achieve this, we construct an adapted frame in Section 3.2, leveraging the geometry of the problem to simplify the computations substantially. In Section 3.3, we examine how an error $E = F \circ K - K \circ R_\omega$ affects the properties studied in Section 3.2. This approach simplifies the problem to solving a basic set of equations, known as cohomological equations, which are introduced and detailed in Section 3.1. Finally, in Section 3.4, we outline the algorithm for finding an approximate parameterization of the curve.

3.1 Cohomological Equations

In this section, we define the cohomological operator and examine its properties, particularly the structure and uniqueness of zero-average solutions. These insights are essential for the development of the algorithm in Section 3.4.

Definition 3.1. Given $\omega \in \mathbb{R}$, we define the *cohomological operator* $\mathscr{L}$ on 1-periodic functions $u : \mathbb{R} \to \mathbb{R}$ as

$$\mathscr{L} u = u - u \circ R_\omega,$$

where $R_\omega : \mathbb{R} \to \mathbb{R}$ is the rotation by ω.

Definition 3.2. The *average* of a continuous 1-periodic function $u : \mathbb{R} \to \mathbb{R}$ is defined as $\langle u \rangle = \int_0^1 u(\theta)\, d\theta$.

Definition 3.3. We define the *one-bite cohomological equation*, where u is the unknown, as

$$\mathscr{L}u = v - \langle v \rangle. \tag{3.1}$$

Remark 3.4. Let $v : \mathbb{R} \to \mathbb{R}$ be a continuous 1-periodic function and assume that R_ω is an ergodic rotation (ω is irrational). If there exists a continuous zero-average solution of (3.1), then it is unique. Let us denote this solution by $\mathscr{R}v$. If v and u have Fourier expansions $v(\theta) = \sum_{k\in\mathbb{Z}} \hat{v}_k e^{2\pi ik\theta}$ and $u(\theta) = \sum_{k\in\mathbb{Z}} \hat{u}_k e^{2\pi ik\theta}$, respectively, then

$$\sum_{k\in\mathbb{Z}} \hat{u}_k e^{2\pi ik\theta} - \sum_{k\in\mathbb{Z}} \hat{u}_k e^{2\pi ik(\theta+\omega)} = \sum_{k\in\mathbb{Z}} \hat{v}_k e^{2\pi ik\theta} - \langle v \rangle.$$

Therefore, the solution of (3.1) is given by

$$\mathscr{R}v(\theta) = \sum_{k\in\mathbb{Z}\setminus\{0\}} \hat{u}_k e^{2\pi ik\theta}, \quad \hat{u}_k = \frac{\hat{v}_k}{1 - e^{2\pi ik\omega}}.$$

Definition 3.5. Given $\gamma > 0$ and $\tau \geq 1$, $\omega \in \mathbb{R}$ is said to be a (γ, τ)-*Diophantine number* if and only if

$$|k \cdot \omega - m| \geq \frac{\gamma}{|k|^\tau}, \qquad \forall k \in \mathbb{Z} \setminus \{0\},\ m \in \mathbb{Z}.$$

Lemma 3.6. *Let $\omega \in \mathbb{R}$ be a (γ, τ)-Diophantine number for some $\gamma > 0$ and $\tau \geq 1$. Let $v : \mathbb{T}_\rho \to \mathbb{C}$ be an analytic and bounded function in $\mathbb{T}_\rho = \{\theta + i\varphi : |\varphi| < \rho\}$, where $\rho > 0$. Then there exists a unique analytic solution $u : \mathbb{T}_\rho \to \mathbb{C}$ of the cohomological equation* (3.1).

Proof. Let us write the Fourier expansion of v as $v(\theta) = \sum_{k\in\mathbb{Z}} \hat{v}_k e^{2\pi ik\theta}$. We can write

$$\hat{v}_k = \int_0^1 v(\theta + i\varphi) e^{-2\pi ik(\theta+i\varphi)}\, d\theta = e^{2\pi k\varphi} \int_0^1 v(\theta + i\varphi) e^{-2\pi ik\theta}\, d\theta.$$

Since v is bounded in $\mathbb{T}_\rho$, $\exists M > 0$ such that $|v(\theta + i\varphi)| \leq M$ for all $\theta + i\varphi \in \mathbb{T}_\rho$. So, $\forall \varphi$ such that $|\varphi| < \rho$ and $\forall k \in \mathbb{Z}$, we have that

$$|\hat{v}_k| \leq M e^{2\pi k\varphi}.$$

Now, if $\varphi < 0$, for $k < 0$ we have that $|\hat{v}_k| \leq M e^{-2\pi|k|\varphi}$. And if $\varphi > 0$, for $k > 0$ we have that $|\hat{v}_k| \leq M e^{2\pi|k|\varphi}$. Therefore,

$$|\hat{v}_k| \leq M e^{-2\pi|k||\varphi|} \leq M e^{-2\pi|k|\rho} \quad \forall k \in \mathbb{Z}.$$

Now, we can control the divisors of the Fourier coefficients $\hat{u}_k$ of $\mathscr{R}v$

$$\left|1 - e^{2\pi ik\omega}\right| = 2\left|\sin(\pi k\omega)\right| \geq 4 \min_{m\in\mathbb{Z}} |k\omega - m| \geq \frac{4\gamma}{|k|^\tau}.$$

Using this and the fact that $|\hat{v}_k| \leq Me^{-2\pi|k|\rho}$, the Fourier coefficients of u are bounded by

$$|\hat{u}_k| = \left|\frac{\hat{v}_k}{1 - e^{2\pi ik\omega}}\right| \leq \frac{Me^{-2\pi|k|\rho}}{4\gamma|k|^{-\tau}} = \frac{M|k|^\tau}{4\gamma} e^{-2\pi|k|\rho}.$$

Now we want to see that u is analytic in $\mathbb{T}_{\rho-\delta}$ $\forall 0 < \delta < \rho$.

$$\begin{aligned}
\|u\|_{\rho-\delta} = \left\|\sum_{k\in\mathbb{Z}} \hat{u}_k e^{2\pi k\theta}\right\|_{\rho-\delta} &\leq \sum_{k\in\mathbb{Z}} |\hat{u}_k| e^{2\pi|k|(\rho-\delta)} \\
&\leq \sum_{k\in\mathbb{Z}} \frac{\|v\|_\rho |k|^\tau}{4\gamma} e^{-2\pi|k|\rho} e^{2\pi|k|(\rho-\delta)} \\
&\leq \frac{\|v\|_\rho}{2\gamma} \sum_{k>0} k^\tau e^{-\pi k\delta} e^{-\pi k\delta} \\
&\leq \frac{\|v\|_\rho}{2\gamma} \sum_{k>0} \left(\frac{\tau}{\pi\delta}\right)^\tau e^{-\tau} e^{-\pi k\delta} \\
&\leq \frac{\|v\|_\rho}{2\gamma} \left(\frac{\tau}{e\pi}\right)^\tau \frac{1}{\delta^\tau} \sum_{k>0} e^{-\pi k\delta} \\
&\leq \frac{1}{2\pi} \left(\frac{\tau}{e\pi}\right)^\tau \frac{1}{\gamma\delta^{\tau+1}} \|v\|_\rho .
\end{aligned}$$

where in the fourth inequality we used the fact that $x_0 = \frac{\tau}{\pi\delta}$ is the maximum of $f(x) = x^\tau e^{-\pi\delta x}$ for $x > 0$, and in the last inequality we used that

$$\sum_{k>0} e^{-\pi k\delta} \leq \int_0^\infty e^{-\pi x\delta}\, dx = \frac{1}{\pi\delta}.$$

□

Remark 3.7. Since the kernel of $\mathscr{L}$ consists of constant functions and $\mathscr{L}$ is a linear operator, it follows that $\mathscr{L}(\mathscr{R}v + C) = \mathscr{L}(\mathscr{R}v)$ for any constant $C \in \mathbb{R}$. In other words, adding a constant to our zero-average solution produces a non-zero average solution of equation (3.1).

Remark 3.8. Let $\mathbb{C}^\mathbb{Z} = \{(\hat{u}_k)_{k\in\mathbb{Z}} | \hat{u}_k \in \mathbb{C}\}$ be the space of formal Fourier series, where each element is a doubly infinite sequence of complex numbers. Note that the operator $\mathscr{L}$ maps $\mathbb{C}^\mathbb{Z}$ to $\{(\hat{v}_k = \hat{u}_k(1 - e^{2\pi ik\omega}))_{k\in\mathbb{Z}}\}$. An observation that will be useful for the development of the algorithm is that $\mathbb{C}^\mathbb{Z}$ can be written as a

direct sum of the subspace of formal Fourier series with $\hat{u}_0 = 0$ and the subspace of sequences where all terms except $\hat{u}_0$ are zero. That is,

$$\mathbb{C}^{\mathbb{Z}} = \{(\hat{u}_k)_{k\in\mathbb{Z}} | \hat{u}_0 = 0\} \oplus \{(\hat{u}_k)_{k\in\mathbb{Z}} | \hat{u}_k = 0 \text{ for } k \neq 0\}.$$

Now suppose that a function f has Fourier expansion $f(\theta) = \sum_{k\in\mathbb{Z}} \hat{f}_k e^{2\pi i k\theta}$. Then we can write the decomposition of f uniquely as

$$f(\theta) = \hat{f}_0 + \tilde{f}(\theta), \quad \text{where} \quad \tilde{f}(\theta) = \sum_{k\in\mathbb{Z}\setminus\{0\}} \hat{f}_k e^{2\pi i k\theta}.$$

Also, since the average of f is given by $\langle f \rangle = \hat{f}_0$, we know that $\tilde{f}$ is zero average.

3.2 Symplectic Frame

Following the initial construction of the problem, stated in Chapter 1, take $\mathscr{A}$ as a simply connected subset of $\mathbb{R}^2$ with the standard symplectic structure. Suppose that $F : \mathscr{A} \to \mathscr{A}$ is a symplectomorphism and that $\mathscr{K}$ is a quasiperiodic invariant curve for F with a certain rotation number ω. That is, it is parameterized by a 1-periodic regular function $K : \mathbb{R} \to \mathscr{A}$ such that the invariance equation is satisfied:

$$F \circ K = K \circ R_\omega. \tag{3.2}$$

Note that the regularity condition of K is $K'(\theta) \neq (0,0)\ \forall\theta \in \mathbb{R}$. Writing K as $K(\theta) = (x(\theta) \quad y(\theta))^\top$, this is equivalent to $x'(\theta)^2 + y'(\theta)^2 \neq 0\ \forall\theta \in \mathbb{R}$.

Remark 3.9. Unlike in Chapter 1, where K is defined on the torus, here K is defined on the real line. We are abusing notation by identifying the 1-periodic curve $K : \mathbb{R} \to \mathscr{A}$ with its counterpart $[K] : \mathbb{T} \to \mathscr{A}$, such that $[K] \circ \pi = K$, where $\pi : \mathbb{R} \to \mathbb{T}$ is the canonical covering map.

Remark 3.10. If we have a solution of $F \circ K = K \circ R_\omega$, then $\forall\theta_0 \in \mathbb{R}$ we obtain another solution $K \circ R_{\theta_0}$. To avoid this, let us fix $y_0 \in \mathbb{R}$ and impose the condition $\pi_y(\tilde{K}(0)) - y_0 = 0$, where π_y is the projection onto the y-axis. That is, we are fixing the initial value of the y-coordinate of the curve. To ensure that the line $y = y_0$ is transversal to the curve K, the choice of y_0 must be such that $y'(0) \neq 0$. The transversality condition guarantees that the curve intersects $y = y_0$ with a well-defined slope, avoiding degeneracies.

Let us start this construction by trying to find a suitable expression for the tangent map of F around the curve. Our aim is to find an adapted frame $P(\theta)$ satisfying

$$P(\theta)^\top \Omega_0 P(\theta) = \Omega_0. \tag{3.3}$$

In this case, we say that P is a *symplectic frame*. As we will see later, this will allow us to simplify certain computations substantially. Notice that (3.3) is equivalent to $\det P = 1$. Let us take

$$P(\theta) = \begin{pmatrix} L(\theta) & N(\theta) \end{pmatrix}, \tag{3.4}$$

$$L(\theta) = K'(\theta), \tag{3.5}$$

$$N(\theta) = N_0(\theta)B(\theta), \tag{3.6}$$

$$N_0(\theta) = \Omega_0 L(\theta), \tag{3.7}$$

where $B : \mathbb{R} \to \mathbb{R}$ is a 1-periodic function to be determined. Writing K as $K(\theta) = \begin{pmatrix} x(\theta) & y(\theta) \end{pmatrix}^\top$, the symplecticity condition of P translates to $x'(\theta)^2 B(\theta) + y'(\theta)^2 B(\theta) = 1$. This allows us to determine P and B as

$$P(\theta) = \begin{pmatrix} x'(\theta) & -y'(\theta)B(\theta) \\ y'(\theta) & x'(\theta)B(\theta) \end{pmatrix} \quad \text{and} \quad B(\theta) = \frac{1}{x'(\theta)^2 + y'(\theta)^2}. \tag{3.8}$$

Note that, because of the regularity of K, $x'(\theta)^2 + y'(\theta)^2 \neq 0 \ \forall \theta \in \mathbb{R}$ and therefore B is well-defined. Our next goal is to see that $DF(K(\theta))$ in the symplectic frame P is a block triangular matrix. We will write it as

$$\Lambda(\theta) = P(\theta + \omega)^{-1} DF(K(\theta)) P(\theta). \tag{3.9}$$

From the invariance equation (3.2) we have that $DF(K(\theta))L(\theta) = L(\theta + \omega)$. Therefore, we can write the first column of Λ as

$$\Lambda(\theta) = \begin{pmatrix} 1 & \cdot \\ 0 & \cdot \end{pmatrix}.$$

Since $\det DF(K(\theta)) = 1$ and $\det P = 1$, we have that $\det \Lambda(\theta) = 1$. Therefore, the second column of $\Lambda(\theta)$ is of the form

$$\Lambda(\theta) = \begin{pmatrix} 1 & T(\theta) \\ 0 & 1 \end{pmatrix}.$$

where $T(\theta)$ is a function to be determined. We will refer to $T(\theta)$ as the torsion. With a few computations, from the definition of $\Lambda(\theta)$, (3.9), we can see that $T(\theta)$ is given by

$$T(\theta) = N(\theta + \omega)^\top \Omega_0 DF(K(\theta)) N(\theta), \tag{3.10}$$

where we used that the inverse of P is

$$P(\theta)^{-1} = \begin{pmatrix} x'(\theta)B(\theta) & y'(\theta)B(\theta) \\ -y'(\theta) & x'(\theta) \end{pmatrix}.$$

In conclusion, if the invariance equation is satisfied, we say that the torus is reducible, meaning that $DK(K(\theta))$ is reducible to a triangular matrix $\Lambda(\theta)$ using the adapted frame $P(\theta)$.

3.3 Approximate Reducibility

Given a parameterization $K : \mathbb{R} \to \mathscr{A}$, consider its error function

$$E(\theta) = F(K(\theta)) - K(\theta + \omega). \tag{3.11}$$

The aim of this section is to describe how this error affects the geometrical properties discussed in Section 3.2. We will show that, if the invariance equation is satisfied with a certain error $E(\theta)$, then $DF(K(\theta))$ is approximately reducible to $\Lambda(\theta)$ up to an error that can be controlled by $E(\theta)$. The error in the reducibility of $DF(K(\theta))$ is given by

$$E_{red} = P(\theta + \omega)^{-1} DF(K(\theta))P(\theta) - \Lambda(\theta).$$

Denoting by $E^{i,j}_{red}(\theta)$ the (i,j)-th entry of E_{red}, we can see that

$$\begin{aligned} E^{1,1}_{red}(\theta) &= N(\theta + \omega)^\top \Omega_0 DF(K(\theta))L(\theta) - 1 \\ &= N(\theta + \omega)^\top \Omega_0 (DE(\theta) + L(\theta + \omega)) - 1 \\ &= N(\theta + \omega)^\top \Omega_0 DE(\theta) + 1 - 1 \\ &= N(\theta + \omega)^\top \Omega_0 DE(\theta), \end{aligned}$$

where we used that, from (3.11), $DE(\theta) = DF(K(\theta))L(\theta) - L(\theta + \omega)$, and from the symplecticity of P, $N^\top \Omega_0 L = 1$. Moreover,

$$\begin{aligned} E^{1,2}_{red}(\theta) &= N(\theta + \omega)^\top \Omega_0 DF(K(\theta))N(\theta) - T(\theta) = 0, \\ E^{2,1}_{red}(\theta) &= -L(\theta + \omega)^\top \Omega_0 DF(K(\theta))L(\theta) \\ &= -L(\theta + \omega)^\top \Omega_0 (DE(\theta) + L(\theta + \omega)) \\ &= -L(\theta + \omega)^\top \Omega_0 DE(\theta), \\ E^{2,2}_{red}(\theta) &= -L(\theta + \omega)^\top \Omega_0 DF(K(\theta))N(\theta) - 1 \\ &= -(L(\theta)^\top DF(K(\theta))^\top - DE(\theta)^\top)\Omega_0 DF(K(\theta))N(\theta) - 1 \\ &= DE(\theta)^\top \Omega_0 DF(K(\theta))N(\theta) - L(\theta)^\top DF(K(\theta))^\top \Omega_0 DF(K(\theta))N(\theta) - 1 \\ &= DE(\theta)^\top \Omega_0 DF(K(\theta))N(\theta) - L(\theta)^\top \Omega_0 N(\theta) - 1 \\ &= DE(\theta)^\top \Omega_0 DF(K(\theta))N(\theta), \end{aligned}$$

where we used that, since F is symplectic, $DF(K(\theta))^\top \Omega_0 DF(K(\theta)) = \Omega_0$. Therefore, we conclude that E_{red} is controlled by $DE(\theta)$.

Remark 3.11. Since E_{red} is controlled by $DE(\theta)$ and not $E(\theta)$, one could worry that that $DE(\theta)$ is not necessarily small even if $E(\theta)$ is. But in fact, if $E(\theta)$ is analytic, the norm of $DE(\theta)$ can be bounded by the norm of $E(\theta)$, in a slightly smaller domain. These bounds, known as the Cauchy bounds, together with the solution of the cohomological equations, are key to the development of KAM theory. See [3] for more details.

Remark 3.12. Although we are working with an approximate solution of the invariance equation, the frame $P(\theta)$ remains symplectic in the case of planar systems. That is, the error in the symplecticity of $P(\theta)$, defined as

$$E_{sym}(\theta) = P(\theta)^\top \Omega_0 P(\theta) - \Omega_0,$$

is zero. In higher dimensions, however, $E_{sym}(\theta)$ is not necessarily zero. Nevertheless, similar to the reducibility error, it can be controlled by $E(\theta)$, and thus the frame $P(\theta)$ is said to be approximately symplectic. For more details on approximate symplecticity in higher dimensions, refer to [1].

3.4 The Algorithm

In this section we present the algorithm to find an approximate parameterization K of the curve $\mathscr{K}$. We make use of the cohomological equations studied in Section 3.1, the adapted frame $P(\theta)$ constructed in Section 3.2, and the fact that $DF(K(\theta))$ is approximately reducible to $\Lambda(\theta)$.

In order to refine $K(\theta)$, in spirit of the Newton method, we add a correction $\tilde{K}(\theta) = K(\theta) + \Delta K(\theta)$ given by the approximate solution of the equation

$$0 = F(\tilde{K}(\theta)) - \tilde{K}(\theta + \omega). \tag{3.12}$$

Using the definition of $\tilde{K}$ and neglecting quadratic terms, from (3.12) we obtain

$$DF(K(\theta))\Delta K(\theta) - \Delta K(\theta + \omega) = -E(\theta). \tag{3.13}$$

Taking $\Delta K(\theta) = P(\theta)\xi(\theta)$, we can write (3.13) as

$$DF(K(\theta))P(\theta)\xi(\theta) - P(\theta + \omega)\xi(\theta + \omega) = -E(\theta).$$

Now, noticing that $DF(K(\theta))P(\theta) = P(\theta + \omega)(\Lambda(\theta) + E_{red}(\theta))$, and multiplying both sides by $P(\theta + \omega)^{-1}$, we get

$$\Lambda(\theta)\xi(\theta) + E_{red}(\theta)\xi(\theta) - \xi(\theta + \omega) = -P(\theta + \omega)^{-1}E(\theta). \tag{3.14}$$

And neglecting the quadratically small terms, we have

$$\Lambda(\theta)\xi(\theta) - \xi(\theta + \omega) = -P(\theta + \omega)^{-1}E(\theta). \tag{3.15}$$

Let us define

$$\eta(\theta) = -P(\theta + \omega)^{-1}E(\theta). \tag{3.16}$$

The key to solving system (3.15) is to recognise that by carefully manipulating averages, we can approach the solution similarly to how we handle cohomological

equations. Let us start by subtracting the term $\langle \eta^N(\theta) \rangle$ from the right-hand side of the second coordinate of (3.15), obtaining

$$\begin{pmatrix} 1 & T(\theta) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \xi^L(\theta) \\ \xi^N(\theta) \end{pmatrix} - \begin{pmatrix} \xi^L(\theta+\omega) \\ \xi^N(\theta+\omega) \end{pmatrix} = \begin{pmatrix} \eta^L(\theta) \\ \eta^N(\theta) - \langle \eta^N(\theta) \rangle \end{pmatrix}. \tag{3.17}$$

That is,

$$\begin{cases} \xi^L(\theta) - \xi^L(\theta+\omega) = \eta^L(\theta) - T(\theta)\xi^N(\theta) \\ \xi^N(\theta) - \xi^N(\theta+\omega) = \eta^N(\theta) - \langle \eta^N(\theta) \rangle \end{cases}. \tag{3.18}$$

We will later justify that we can cancel the term $\langle \eta^N(\theta) \rangle$, since it is quadratically small. Note that, with this manipulation, the right-hand side of the second equation of (3.18) now has zero average. As a result, it becomes a solvable cohomological equation, and we can write its solution as

$$\xi^N(\theta) = \mathscr{R}(\eta^N(\theta)) + \hat{\xi}_0^N,$$

where we used the same notation as in Section 3.1, and $\hat{\xi}_0^N$ is a constant to be determined. In fact, we have to choose $\hat{\xi}_0^N$ such that the right-hand side of the first equation has zero average. That is, $0 = \langle \eta^L - T\xi^N \rangle$. This allows us to determine $\hat{\xi}_0^N$ as

$$\hat{\xi}_0^N = \langle T \rangle^{-1} \langle \eta^L - T\mathscr{R}(\eta^N) \rangle.$$

In order to do so, we have to assume that $\langle T \rangle$ is invertible. This requirement corresponds to the non-degeneracy condition introduced in Section 1.1, and it is often referred to as the twist condition. Only when this condition is satisfied can we compute $\hat{\xi}_0^N$ and turn the first equation of (3.18) into a solvable cohomological equation, with its solution given by

$$\xi^L(\theta) = \mathscr{R}(\eta^L(\theta) - T(\theta)\xi^N(\theta)) + \hat{\xi}_0^L,$$

where we have the flexibility to choose any value for $\hat{\xi}_0^L \in \mathbb{R}$. But, following Remark 3.10, we fix $\hat{\xi}_0^L$ by imposing the condition

$$\pi_y(\tilde{K}(0)) - y_0 = 0$$

to the new parameterization, where y_0 is chosen such that $y'(0) \neq 0$. Let us define $e_0 = \pi_y(K(0)) - y_0$ and write $\xi^L(\theta) = \tilde{\xi}^L(0) + \hat{\xi}_0^L$ (see Remark 3.8). Then,

$$\begin{aligned} \pi_y(K(0) + \Delta K(0)) - y_0 &= \pi_y(\Delta K(0)) + \pi_y(K(0)) - y_0 \\ &= \pi_y(\Delta K(0)) + e_0 \\ &= \pi_y(L(0)\xi^L(0) + N(0)\xi^N(0)) + e_0 \\ &= \pi_y(L(0))\hat{\xi}_0^L + \pi_y(L(0)\tilde{\xi}^L(0)) + \pi_y(N(0)\xi^N(0)) + e_0. \end{aligned}$$

Using this and the fact that $y'(0) \neq 0$, we can determine $\hat{\xi}_0^L$ as

$$\hat{\xi}_0^L = -\frac{\pi_y(L(0)\tilde{\xi}^L(0)) + \pi_y(N(0)\xi^N(0)) + e_0}{\pi_y(L(0))},$$

and therefore the complete solution of (3.17) is:

$$\begin{aligned}
&\hat{\xi}_0^N = \langle T \rangle^{-1} \langle \eta^L - T\mathscr{R}(\eta^N) \rangle, \\
&\xi^N(\theta) = \mathscr{R}(\eta^N(\theta)) + \hat{\xi}_0^N, \\
&\hat{\xi}_0^L = -\frac{\pi_y(L(0)\tilde{\xi}^L(0)) + \pi_y(N(0)\xi^N(0)) + e_0}{\pi_y(L(0))}, \\
&\xi^L(\theta) = \mathscr{R}(\eta^L(\theta) - T(\theta)\xi^N(\theta)) + \hat{\xi}_0^L.
\end{aligned} \tag{3.19}$$

Remark 3.13. Although we have presented an algorithm to find an approximate parameterization of $\mathscr{K}$, we have not demonstrated the convergence of the method. This is a complex KAM problem, discussed in [1], where it is shown that, if the norm of E is sufficiently small, then the analycity of v in the cohomological equations, the diophantine condition of ω, the twist condition, and the regularity of K are sufficient conditions for the convergence of the parameterization method.

Notice that in (3.17) we added the term $\langle \eta^N(\theta) \rangle$. The following lemmas show that both the error of the linearized equation (3.14) and $\langle \eta^N(\theta) \rangle$ are quadratically small. In fact, in Lemma 3.15 we will see that this is a consequence of the exact symplecticity of F.

Lemma 3.14. *Let us assume that $\langle T(\theta) \rangle \neq 0$. Consider the solution $\xi(\theta)$ obtained in* (3.19)*, with η and T given by* (3.16) *and* (3.10)*, respectively. Then, if we take $\Delta K(\theta) = P(\theta)\xi(\theta)$,*

$$DF(K(\theta))\Delta K(\theta) - \Delta K(\theta + \omega) + E(\theta) = P(\theta + \omega)E_{lin}(\theta),$$

where

$$E_{lin}(\theta) = E_{red}(\theta)\xi(\theta) - \begin{pmatrix} 0 \\ \langle \eta^N(\theta) \rangle \end{pmatrix}.$$

Proof. Using (3.14) and (3.19), we have that

$$\begin{aligned}
&DF(K(\theta))\Delta K(\theta) - \Delta K(\theta + \omega) + E(\theta) \\
&\quad = DF(K(\theta))P(\theta)\xi(\theta) - P(\theta + \omega)\xi(\theta + \omega) + E(\theta)
\end{aligned}$$

$$
\begin{aligned}
&= P(\theta+\omega)(P(\theta+\omega)^{-1}DF(K(\theta))P(\theta)\xi(\theta) - \xi(\theta+\omega) \\
&\quad + P(\theta+\omega)^{-1}E(\theta)) \\
&= P(\theta+\omega)((\Lambda(\theta) + E_{red}(\theta))\xi(\theta) - \xi(\theta+\omega) - \eta(\theta)) \\
&= P(\theta+\omega)(E_{red}(\theta)\xi(\theta) + \eta(\theta) - \begin{pmatrix} 0 \\ \langle \eta^N(\theta) \rangle \end{pmatrix} - \eta(\theta)) \\
&= P(\theta+\omega)E_{lin}(\theta).
\end{aligned}
$$

□

Lemma 3.15. *If $K(\theta)$ is an approximately invariant closed curve with error $E(\theta)$, then*

$$\langle \eta^N(\theta) \rangle = \langle L(\theta+\omega)^\top \Omega_0 E(\theta) \rangle = \langle DE(\theta)^\top \Delta a(\theta) \rangle,$$

where

$$\Delta a(\theta) = a(F(K(\theta))) - a(K(\theta+\omega)) = \int_0^1 Da(K(\theta+\omega) + tE(\theta))E(\theta)\, dt.$$

Proof. The following argument is adapted from [1].

$$
\begin{aligned}
&L(\theta+\omega)\Omega_0 E(\theta) \\
&\quad = L(\theta+\omega)Da(K(\theta+\omega))^\top E(\theta) - L(\theta+\omega)^\top Da(K(\theta+\omega))E(\theta) \\
&\quad = D(a(K(\theta+\omega)))^\top E(\theta) + L(\theta+\omega)^\top(-a(F(K(\theta))) + a(K(\theta+\omega))) \\
&\quad = (D(a(K(\theta+\omega))E(\theta)))^\top - (DE(\theta))^\top a(K(\theta+\omega)) \\
&\qquad - (DF(K(\theta))L(\theta) - DE(\theta))^\top a(F(K(\theta))) + L(\theta+\omega)^\top a(K(\theta+\omega)) \\
&\quad = (D(a(K(\theta+\omega))E(\theta)))^\top + (DE(\theta))^\top \Delta a(\theta) \\
&\qquad - (D(S(K(\theta))))^\top - L(\theta)^\top a(K(\theta)) + L(\theta+\omega)^\top a(K(\theta+\omega))
\end{aligned}
$$

In the last identity we used that, with the standard symplectic structure, the symplecticity of F implies that F is exact symplectic (from Lemma 2.8), so we can take S as the primitive function of F. Next, taking the average of the last expression, notice that $(D(a(K(\theta+\omega))E(\theta)))^\top$ and $(D(S(K(\theta))))^\top$ have zero average, since the average of the derivative of a periodic function is zero. Moreover, since K is periodic, we have that

$$\langle L(\theta+\omega)^\top a(K(\theta+\omega)) \rangle = \langle L(\theta)^\top a(K(\theta)) \rangle$$

and we conclude that

$$\langle L(\theta+\omega)\Omega_0 E(\theta) \rangle = \langle (DE(\theta))^\top \Delta a(\theta) \rangle.$$

□

Algorithm: one step of the Newton method.

The final algorithm to refine K results from the iteration of the following process, which will be iterated until the norm of $E(\theta)$ is small enough. In the implementation of this algorithm, seen in Chapter 5, we will use the L1 norm of $E(\theta)$ as the stopping criterion. The algorithm is as follows:

(i) Given an initial approximation $K(\theta)$, compute the error $E(\theta) = F(K(\theta)) - K(\theta + \omega)$.

(ii) Construct the frame $P(\theta)$ using

$$\begin{aligned} L(\theta) &= K'(\theta), \\ B(\theta) &= \frac{1}{x'(\theta)^2 + y'(\theta)^2}, \\ N(\theta) &= \Omega_0 L(\theta) B(\theta), \\ P(\theta) &= (L(\theta) \quad N(\theta)). \end{aligned}$$

(iii) Compute η and T using the following definitions

$$\begin{aligned} \eta^L(\theta) &= -N(\theta + \omega)^\top \Omega_0 E(\theta), \\ \eta^N(\theta) &= L(\theta + \omega)^\top \Omega_0 E(\theta), \\ T(\theta) &= N(\theta + \omega)^\top \Omega_0 DF(K(\theta)) N(\theta). \end{aligned}$$

(iv) Assuming that $\langle T \rangle \neq 0$, compute the solution of system (3.17), $\xi(\theta)$, following these steps:

$$\begin{aligned} \tilde{\xi}^N(\theta) &= \mathscr{R}(\eta^N(\theta)), \\ \hat{\xi}_0^N &= \langle T \rangle^{-1} \langle \eta^L - T\tilde{\xi}^N \rangle, \\ \xi^N(\theta) &= \hat{\xi}_0^N + \tilde{\xi}^N(\theta), \\ \tilde{\xi}^L(\theta) &= \mathscr{R}(\eta^L(\theta) - T(\theta)\xi^N(\theta)), \\ \hat{\xi}_0^L &= -\frac{\pi_y(L(0)\tilde{\xi}^L(0)) + \pi_y(N(0)\xi^N(0)) + e_0}{\pi_y(L(0))}, \\ \xi^L(\theta) &= \hat{\xi}_0^L + \tilde{\xi}^L(\theta). \end{aligned}$$

(v) Compute the correction $\Delta K(\theta) = P(\theta)\xi(\theta) = L(\theta)\xi^L(\theta) + N(\theta)\xi^N(\theta)$.

(vi) Correct the parameterization $\tilde{K}(\theta) = K(\theta) + \Delta K(\theta)$. $\tilde{K}(\theta)$ will serve as the new approximation of $\mathscr{K}$ for the next iteration.

Chapter 4

The Balanced Parameterization Method

The motivation for this chapter is drawn from [2], a paper that explores an approach to computing the rotation number using Weighted Birkhoff averages (see [6]), followed by a version of the parameterization method that discretizes the functional problem and uses an iterative Newton scheme to solve it.

Revisiting our Chapter 3, the key to solving our equations was the observation that $\langle \eta^N \rangle$ was quadratically small, which allowed for the balancing of the averages in the cohomological equations and resolved the initial degeneracy of the problem. In contrast, the approach used in [2] addresses the same degeneracy by introducing an "unfolding parameter", β (referred to as a "dummy parameter" throughout this work), to "balance" the problem's equations. Without the observation that this degeneracy is caused by a quadratically small term, [2]'s approach relies purely on numerical methods, involving large linear systems that require lots of computations.

In this chapter, we adopt the idea of using a dummy parameter but enhance it by incorporating the symplectic frame construction presented in Chapter 3. We will refer to this new approach as the **balanced parameterization method**. This shifts the method from a computationally heavy numerical approach to a more geometric one, offering a more efficient and conceptually clear alternative to the approach of [2]. Additionally, we extend the paper's findings, delving deeper into the importance of this dummy parameter and its role in making the scheme solvable. The computational cost of the approach in [2] is of the order of $O(N^3)$, where N denotes the number of Fourier modes used in the discretization. Our method, on the other hand, has a computational cost of the order $O(N \log N)$, representing a substantial improvement in efficiency.

This chapter follows a structure similar to that of Chapter 3. Section 4.1 introduces the concept of a dummy parameter and describes the construction of a suitable symplectic frame to solve the problem's equations. In Section 4.2, we examine the error in the reducibility of $DF(K(\theta))$ when the invariance equation is satisfied with a specific error $E = F \circ K - (1+\beta)\, K \circ R_\omega$. Finally, Section 4.3 presents the algorithm for approximating the parameterization of the curve $\mathscr{K}$.

4.1 Dummy Parameter and Symplectic Frame

Following the same context as in the previous chapter, take $\mathscr{A}$ as a simply connected subset of $\mathbb{R}^2$ with the standard symplectic structure and $F : \mathscr{A} \to \mathscr{A}$ as a symplectomorphism. Let $\mathscr{K}$ be a quasiperiodic invariant curve for F with a certain rotation number ω, parameterized by a 1-periodic regular function $K : \mathbb{R} \to \mathscr{A}$ such that the invariance equation is satisfied, and impose the transversality condition as in Remark 3.10. That is, we have the system

$$\begin{cases} F(K(\theta)) = K(\theta + \omega) \\ \pi_y(K(0)) - y_0 = 0 \end{cases}. \tag{4.1}$$

Since we have two equations in one unknown K, we add a small dummy parameter β, satisfying $1 + \beta > 0$ and consider the equations

$$\begin{cases} F(K(\theta)) = (1+\beta)K(\theta + \omega) \\ \pi_y(K(0)) - y_0 = 0 \end{cases}. \tag{4.2}$$

Now, it is essential to examine the connection between the original system of equations and the system with the added dummy parameter. The following lemma addresses this relationship, demonstrating that solutions to the new system also satisfy the original equations.

Lemma 4.1. *If K^*, β^* is a solution of* (4.2)*, then $\beta^* = 0$, K^* is a solution of* (4.1)*.*

Proof. Let us assume that K^*, β^* is a solution of (4.2). F is an area preserving diffeomorphism, so we have that $\forall V \subset \mathscr{A}$, $A(V) = A(F(V))$, where V is an open and bounded subset. Therefore, the area enclosed by the curve parameterized by K^* before and after the application of F is the same. Now, take V as the region enclosed by K^*, and we have that

$$A(V) = \int_V 1 = \int_{F(V)} 1 = (1+\beta^*)^2 \int_V 1 = (1+\beta^*)^2 A(V).$$

Therefore, $\beta^* = 0$ or $\beta^* = -2$. But since $1 + \beta > 0$, we have that $\beta^* = 0$ and as a result K^* is a solution of (4.1). □

Remark 4.2. The assumption that $1 + \beta > 0$ is actually very mild. In practice, during the implementation of the method, β is set to be very small, and at the solution, we know that $\beta = 0$.

Now, as we did in Chapter 3, we will take the frame $P(\theta)$ as in (3.8). We want to compute $\Lambda(\theta)$, where

$$\Lambda(\theta) = P(\theta + \omega)^{-1} DF(K(\theta)) P(\theta).$$

From the invariance equation (4.2), we have that

$$DF(K(\theta))L(\theta) = (1 + \beta)L(\theta + \omega). \tag{4.3}$$

From (4.3) and since $\det \Lambda(\theta) = 1$, we can write $\Lambda(\theta)$ as

$$\Lambda(\theta) = \begin{pmatrix} 1 + \beta & T(\theta) \\ 0 & (1 + \beta)^{-1} \end{pmatrix}.$$

Finally, we can compute $T(\theta)$ as in (3.10), and we have that

$$T(\theta) = N(\theta + \omega)^\top \Omega_0 DF(K(\theta)) N(\theta). \tag{4.4}$$

4.2 Approximate Reducibility

Given a parameterization $K : \mathbb{R} \to \mathscr{A}$, consider its error function

$$E(\theta) = F(K(\theta)) - (1 + \beta)K(\theta + \omega). \tag{4.5}$$

As in Chapter 3, we will show that if the invariance equation is satisfied with a certain error $E(\theta)$, then $DF(K(\theta))$ is approximately reducible to $\Lambda(\theta)$ up to an error that can be controlled by $E(\theta)$. The error in the reducibility of $DF(K(\theta))$ is given by

$$E_{red}(\theta) = P(\theta + \omega)^{-1} DF(K(\theta)) P(\theta) - \Lambda(\theta).$$

Denoting by $E_{red}^{i,j}(\theta)$ the (i, j)-th entry of E_{red}, we can see that

$$\begin{aligned} E_{red}^{1,1}(\theta) &= N(\theta + \omega)^\top \Omega_0 DF(K(\theta))L(\theta) - (1 + \beta) \\ &= N(\theta + \omega)^\top \Omega_0 (DE(\theta) + (1 + \beta)L(\theta + \omega)) - (1 + \beta) \\ &= N(\theta + \omega)^\top \Omega_0 DE(\theta) + N(\theta + \omega)^\top \Omega_0 L(\theta + \omega)(1 + \beta) - (1 + \beta) \\ &= N(\theta + \omega)^\top \Omega_0 DE(\theta), \end{aligned}$$

$$
\begin{aligned}
E^{1,2}_{red}(\theta) &= N(\theta+\omega)^\top \Omega_0 DF(K(\theta))N(\theta) - T(\theta) = 0,\\
E^{2,1}_{red}(\theta) &= -L(\theta+\omega)^\top \Omega_0 DF(K(\theta))L(\theta)\\
&= -L(\theta+\omega)^\top \Omega_0 (DE(\theta) + (1+\beta)L(\theta+\omega))\\
&= -L(\theta+\omega)^\top \Omega_0 DE(\theta) - L(\theta+\omega)^\top \Omega_0 L(\theta+\omega)(1+\beta)\\
&= -L(\theta+\omega)^\top \Omega_0 DE(\theta),\\
E^{2,2}_{red}(\theta) &= -L(\theta+\omega)^\top \Omega_0 DF(K(\theta))N(\theta) - (1+\beta)^{-1}\\
&= -(1+\beta)^{-1}(L(\theta)^\top DF(K(\theta))^\top - DE(\theta)^\top)\Omega_0 DF(K(\theta))N(\theta) - (1+\beta)^{-1}\\
&= (1+\beta)^{-1} DE(\theta)^\top \Omega_0 DF(K(\theta))N(\theta)\\
&\quad - (1+\beta)^{-1} L(\theta)^\top DF(K(\theta))^\top \Omega_0 DF(K(\theta))N(\theta) - (1+\beta)^{-1}\\
&= (1+\beta)^{-1} DE(\theta)^\top \Omega_0 DF(K(\theta))N(\theta)\\
&\quad - (1+\beta)^{-1} L(\theta)^\top \Omega_0 N(\theta) - (1+\beta)^{-1}\\
&= (1+\beta)^{-1} DE(\theta)^\top \Omega_0 DF(K(\theta))N(\theta).
\end{aligned}
$$

Therefore, we conclude that $E_{red}(\theta)$ is controlled by $DE(\theta)$.

Remark 4.3. As in Chapter 3, in the case of planar systems, the frame $P(\theta)$ remains symplectic when the invariance equation is satisfied with a certain error $E(\theta)$. In higher dimensions, however, this is not necessarily true, and it has to be shown that the error in the symplecticity of $P(\theta)$ is controlled by $E(\theta)$.

4.3 The Algorithm

In this section we derive formulas for computing the correction ΔK. The followed approach is similar to the one in the previous chapter, but with the addition of a dummy parameter β to balance the averages of the cohomological equations. To refine $K(\theta)$ and β we add corrections

$$\tilde{K}(\theta) = K(\theta) + \Delta K(\theta) \quad \text{and} \quad \tilde{\beta} = \beta + \Delta\beta,$$

given by the approximate solution of the equation

$$0 = F(\tilde{K}(\theta)) - (1+\tilde{\beta})\tilde{K}(\theta+\omega). \tag{4.6}$$

Using the definition of $\tilde{K}$ and $\tilde{\beta}$, and neglecting second-order error terms, from (4.5) we obtain

$$DF(K(\theta))\Delta K(\theta) - \Delta\beta\, K(\theta+\omega) - (1+\beta)\Delta K(\theta+\omega) = -E(\theta). \tag{4.7}$$

Taking $\Delta K(\theta) = P(\theta)\xi(\theta)$, neglecting quadratic terms, and denoting $C(\theta) = P(\theta+\omega)^{-1}K(\theta+\omega)$ and $\eta(\theta) = -P(\theta+\omega)^{-1}E(\theta)$, we can write (4.7) as

$$\Lambda(\theta)\xi(\theta) - (1+\beta)\xi(\theta+\omega) - \Delta\beta\, C(\theta) = \eta(\theta). \tag{4.8}$$

In matrix form, we have

$$\begin{pmatrix} 1+\beta & T(\theta) \\ 0 & (1+\beta)^{-1} \end{pmatrix} \begin{pmatrix} \xi^L(\theta) \\ \xi^N(\theta) \end{pmatrix} - (1+\beta) \begin{pmatrix} \xi^L(\theta+\omega) \\ \xi^N(\theta+\omega) \end{pmatrix} - \Delta\beta \begin{pmatrix} C^L(\theta) \\ C^N(\theta) \end{pmatrix} = \begin{pmatrix} \eta^L(\theta) \\ \eta^N(\theta) \end{pmatrix}. \tag{4.9}$$

That is,

$$\begin{cases} (1+\beta)\xi^L(\theta) - (1+\beta)\xi^L(\theta+\omega) = \eta^L(\theta) - T(\theta)\xi^N(\theta) + \Delta\beta\, C^L(\theta) \\ (1+\beta)^{-1}\xi^N(\theta) - (1+\beta)\xi^N(\theta+\omega) = \eta^N(\theta) + \Delta\beta\, C^N(\theta) \end{cases}. \tag{4.10}$$

Following Remark 3.8, let us take

$$\begin{aligned} \xi^L(\theta) &= \hat{\xi}^L_0 + \tilde{\xi}^L(\theta), \\ \xi^N(\theta) &= \hat{\xi}^N_0 + \tilde{\xi}^N(\theta), \\ C^L(\theta) &= \hat{C}^L_0 + \tilde{C}^L(\theta), \\ C^N(\theta) &= \hat{C}^N_0 + \tilde{C}^N(\theta), \\ \eta^L(\theta) &= \hat{\eta}^L_0 + \tilde{\eta}^L(\theta), \\ \eta^N(\theta) &= \hat{\eta}^N_0 + \tilde{\eta}^N(\theta), \end{aligned}$$

where $\tilde{\xi}^L(\theta)$, $\tilde{\xi}^N(\theta)$, $\tilde{C}^L(\theta)$, $\tilde{C}^N(\theta)$, $\tilde{\eta}^L(\theta)$ and $\tilde{\eta}^N(\theta)$ have zero average. We can write the second equation as

$$\begin{aligned} ((1+\beta)^{-1} - (1+\beta))\hat{\xi}^N_0 + (1+\beta)^{-1}\tilde{\xi}^N(\theta) - (1+\beta)\tilde{\xi}^N(\theta+\omega) \\ = \hat{\eta}^N_0 + \tilde{\eta}^N(\theta) + \Delta\beta(\hat{C}^N_0 + \tilde{C}^N(\theta)). \end{aligned} \tag{4.11}$$

Let us first solve the zero-average parts of the equation, and later we will adjust the constant terms. Taking the zero-average terms of (4.11), we have

$$(1+\beta)^{-1}\tilde{\xi}^N(\theta) - (1+\beta)\tilde{\xi}^N(\theta+\omega) = \tilde{\eta}^N(\theta) + \Delta\beta\,\tilde{C}^N(\theta). \tag{4.12}$$

Remark 4.4. Similarly to Remark 3.4, where we obtained the zero-average solution of a cohomological equation, take $a \in \mathbb{R}$, satisfying $a \neq 0$, and suppose we have an equation of the form

$$a^{-1} \cdot u - a \cdot u \circ R_\omega = v - \langle v \rangle, \tag{4.13}$$

where u is the unknown, $u : \mathbb{R} \to \mathbb{R}$ and $v : \mathbb{R} \to \mathbb{R}$ are continuous 1-periodic functions, and R_ω is an ergodic rotation (ω is irrational). If v and u have Fourier expansions $v(\theta) = \sum_{k\in\mathbb{Z}} \hat{v}_k e^{2\pi ik\theta}$ and $u(\theta) = \sum_{k\in\mathbb{Z}} \hat{u}_k e^{2\pi ik\theta}$, respectively, then $\mathscr{R}_a v(\theta)$ is a zero-average solution of (4.13), where

$$\forall k \neq 0, \quad \hat{u}_k = \frac{\hat{v}_k}{a^{-1} - a\, e^{2\pi ik\omega}} \qquad \text{and} \qquad \mathscr{R}_a v(\theta) = \sum_{k\in\mathbb{Z}\setminus\{0\}} \hat{u}_k e^{2\pi ik\theta}.$$

Therefore, we can solve (4.12) as

$$\begin{aligned}\tilde{\xi}^N(\theta) &= \mathscr{R}_{(1+\beta)}(\tilde{\eta}^N(\theta) + \Delta\beta\,\tilde{C}^N(\theta)) \\ &= \tilde{\xi}_1^N(\theta) + \Delta\beta\tilde{\xi}_2^N(\theta),\end{aligned}$$

where $\tilde{\xi}_1^N(\theta) = \mathscr{R}_{(1+\beta)}(\tilde{\eta}^N(\theta))$, $\tilde{\xi}_2^N(\theta) = \mathscr{R}_{(1+\beta)}(\tilde{C}^N(\theta))$ and $\Delta\beta$ is yet to be determined.

Remark 4.5. Note that, if $a \neq 1$, then the average of the left-hand side of (4.13) is different than zero, and the corresponding adjustment of averages would not be required. However, in practice, $a = 1 + \beta$ is set to be very close to 1, and possibly equal to 1, so it becomes necessary to adjust the constant terms.

Now, taking the zero-average terms of the first equation of (4.10), we have

$$(1+\beta)(\tilde{\xi}^L(\theta) - \tilde{\xi}^L(\theta+\omega)) = \tilde{\eta}^L(\theta) - \widetilde{T\xi}^N(\theta) + \Delta\beta\,\tilde{C}^L(\theta), \tag{4.14}$$

where $\widetilde{T\xi}^N(\theta) = T(\theta)\xi^N(\theta) - \langle T\xi^N\rangle$. This is a cohomological equation, so we can solve it as

$$\tilde{\xi}^L(\theta) = (1+\beta)^{-1}\mathscr{R}(\tilde{\eta}^L(\theta) - \widetilde{T\xi}^N(\theta) + \Delta\beta\,\tilde{C}^L(\theta)).$$

Let us now adjust the constant terms. From the second equation, we have that

$$((1+\beta)^{-1} - (1+\beta))\hat{\xi}_0^N - \Delta\beta\,\hat{C}_0^N = \hat{\eta}_0^N.$$

From the first equation and the fact that $\tilde{\xi}^N(\theta) = \tilde{\xi}_1^N(\theta) + \Delta\beta\tilde{\xi}_2^N(\theta)$, we have that

$$\langle T\rangle\hat{\xi}_0^N + \langle T\tilde{\xi}_1^N\rangle + \Delta\beta\langle T\tilde{\xi}_2^N\rangle - \Delta\beta\,\hat{C}_0^L = \hat{\eta}_0^L.$$

Therefore, we can determine $\Delta\beta$ and $\hat{\xi}_0^N$ by solving the linear system given by

$$\begin{pmatrix}(1+\beta)^{-1} - (1+\beta) & -\hat{C}_0^N \\ \langle T\rangle & \langle T\tilde{\xi}_2^N\rangle - \hat{C}_0^L\end{pmatrix}\begin{pmatrix}\hat{\xi}_0^N \\ \Delta\beta\end{pmatrix} = \begin{pmatrix}\hat{\eta}_0^N \\ \hat{\eta}_0^L - \langle T\tilde{\xi}_1^N\rangle\end{pmatrix}. \tag{4.15}$$

Lemma 4.6. *Let us assume that $\beta = 0$ and that the twist condition, $\langle T\rangle \neq 0$, is satisfied. Then the linear system* (4.15) *has a unique solution.*

Proof. We have to see that the determinant of the matrix in (4.15) is different than zero. Since $\langle T\rangle \neq 0$, this is equivalent to $\hat{C}_0^N \neq 0$. And we have that

$$C^N(\theta) = -L(\theta+\omega)^\top\Omega_0 K(\theta+\omega) = -y'(\theta+\omega)x(\theta+\omega) + x'(\theta+\omega)y(\theta+\omega).$$

Therefore,

$$\hat{C}_0^N = \langle C^N \rangle = \int_0^1 (-y'(\theta+\omega)x(\theta+\omega) + x'(\theta+\omega)y(\theta+\omega))\, d\theta.$$

By Green's theorem, we have that this expression is exactly two times the area enclosed by the curve parameterized by K. Therefore, $\hat{C}_0^N \neq 0$. □

Notice that we had to impose that $\beta = 0$, but if β is sufficiently small, then the matrix in (4.15) is still invertible. And in practice, during the implementation of the method, β is set to be very small, so (4.15) will have a unique solution.

Finally, following the same strategy as in the previous chapter, we can determine $\hat{\xi}_0^L$ by imposing the condition $\pi_y(\tilde{K}(0)) - y_0 = 0$ to the new parameterization, where y_0 is chosen such that $y'(0) \neq 0$. We obtain the same result

$$\hat{\xi}_0^L = -\frac{\pi_y(L(0)\tilde{\xi}^L(0)) + \pi_y(N(0)\xi^N(0)) + e_0}{\pi_y(L(0))}.$$

As a result, we have that the solution of (4.8) is given by

$$\begin{aligned}
\xi^N(\theta) &= \hat{\xi}_0^N + \mathscr{R}_{(1+\beta)}(\tilde{\eta}^N(\theta)) + \Delta\beta\, \mathscr{R}_{(1+\beta)}(\tilde{C}^N(\theta)),\\
\hat{\xi}_0^L &= -\frac{\pi_y(L(0)\tilde{\xi}^L(0)) + \pi_y(N(0)\xi^N(0)) + e_0}{\pi_y(L(0))},\\
\xi^L(\theta) &= (1+\beta)^{-1}\mathscr{R}(\eta^L(\theta) - T\xi^N(\theta) + \Delta\beta\, C^L(\theta)) + \hat{\xi}_0^L,
\end{aligned}$$

where $\hat{\xi}_0^N$ and $\Delta\beta$ are obtained by solving (4.15).

Algorithm: one step of the Newton method.

The final algorithm to refine K results from the iteration of this process:

(i) Given the initial guesses K and β (on the first iteration, set $\beta = 0$), evaluate the error $E(\theta) = F(K(\theta)) - (1+\beta)K(\theta+\omega)$.

(ii) Construct the frame $P(\theta)$ as in Chapter 3, using

$$\begin{aligned}
L(\theta) &= K'(\theta),\\
B(\theta) &= \frac{1}{x'(\theta)^2 + y'(\theta)^2},\\
N(\theta) &= \Omega_0 L(\theta) B(\theta),\\
P(\theta) &= (L(\theta) \quad N(\theta)).
\end{aligned}$$

(iii) Compute η, T and C using the following definitions

$$\begin{aligned}
\eta^L(\theta) &= -N(\theta+\omega)^\top \Omega_0 E(\theta),\\
\eta^N(\theta) &= L(\theta+\omega)^\top \Omega_0 E(\theta),\\
T(\theta) &= N(\theta+\omega)^\top \Omega_0 DF(K(\theta))N(\theta),\\
C(\theta) &= P(\theta+\omega)^{-1} K(\theta+\omega).
\end{aligned}$$

(iv) Assuming that $\langle T\rangle \neq 0$, compute $\Delta\beta$ and the solution of system (4.8), $\xi(\theta)$, following these steps:

$$\begin{aligned}
&\tilde{\xi}_1^N(\theta) = \mathscr{R}_{(1+\beta)}(\tilde{\eta}^N(\theta)),\\
&\tilde{\xi}_2^N(\theta) = \mathscr{R}_{(1+\beta)}(\tilde{C}^N(\theta)),\\
&\text{Compute } \Delta\beta_n \text{ and } \hat{\xi}_0^N \text{ by solving the linear system (4.15)},\\
&\xi^N(\theta) = \hat{\xi}_0^N + \tilde{\xi}_1^N(\theta) + \Delta\beta\tilde{\xi}_2^N(\theta),\\
&\tilde{\xi}^L(\theta) = (1+\beta)^{-1}\mathscr{R}(\eta^L(\theta) - T(\theta)\xi^N(\theta) + \Delta\beta\, C^L(\theta)),\\
&\hat{\xi}_0^L = -\frac{\pi_y(L(0)\tilde{\xi}^L(0)) + \pi_y(N(0)\xi^N(0)) + e_0}{\pi_y(L(0))},\\
&\xi^L(\theta) = \hat{\xi}_0^L + \tilde{\xi}^L(\theta).
\end{aligned}$$

(v) Compute the correction $\Delta K(\theta) = P(\theta)\xi(\theta) = L(\theta)\xi^L(\theta) + N(\theta)\xi^N(\theta)$.

(vi) Correct the parameterization $\tilde{K}(\theta) = K(\theta) + \Delta K(\theta)$ and the dummy parameter $\tilde{\beta} = \beta + \Delta\beta$. These will serve as the new approximation of $\mathscr{K}$ and the new value of β for the next iteration.

To tie back to the introduction of this chapter, let us consider the significance of the dummy parameter β in this approach of the parameterization method. We introduced a parameter that, while not altering the solutions of the new system, played a key role in enabling us to find a solution. At each step of the Newton method, β was used as a tool to balance the averages of cohomological equations, ensuring they were solvable. So what might seem like a straightforward addition to the equations is, in fact, a crucial element for avoiding the degeneracy of the problem. While the introduction of β was not explored in this depth in [2], this work discusses both the geometric and functional significance of this parameter, highlighting its essential role in the method.

Chapter 5

Numerical Examples

In this chapter, we present numerical examples to demonstrate the application of both the parameterization method and the balanced parameterization method to the standard map and the Hénon map. To achieve this, we adapt the existing codes from [1], which implement the parameterization method for rotational orbits in $2n$ dimensions, to handle librational orbits in planar systems. Additionally, using the codes from [4], we compute a suitable initial guess for the parameterization, which we use as input for the parameterization method. Moreover, we implement the balanced parameterization method and verify the consistency of results between this approach and the parameterization method from Chapter 3.

5.1 Codes

This section provides a brief description of the codes, implemented in C and C++, used to compute the results presented in this chapter.

Code 1: **Code from [4]**: Given a perturbation ϵ and an initial condition (x_0, y_0), Code 1 computes the rotation number ω of the corresponding librational orbit.

Code 2: **Adaptation of the codes from [4]**: Using the output of Code 1, Code 2 computes a suitable initial guess for this librational orbit. The result is represented as a grid of points approximating the curve.

Code 3: **Adaptation of the codes from [1]**: These codes, originally designed for rotational orbits in $2n$ dimensions, were modified to handle librational orbits in planar systems. Starting with the initial approximation of a librational orbit (computed by Code 2) for a specific rotation number ω, Code 3 refines the parameterization using the parameterization method,

described in Chapter 3. It then studies the continuation of the curve with respect to ϵ: increasing the perturbation parameter ϵ, it uses the previous solution as an initial guess to compute the curve with rotation number ω for next value of ϵ.

Code 4: **Adaptation of the codes from [1]**: Code 4 follows the same steps as Code 3, but applies the balanced parameterization method to refine the curve.

5.2 Standard Map

Let us consider again the standard map, given by

$$\begin{aligned} F_\epsilon : \quad & \mathbb{T} \times \mathbb{R} \longrightarrow \mathbb{T} \times \mathbb{R} \\ & (x, y) \longmapsto (x + y - \frac{\epsilon}{2\pi} \sin(2\pi x),\ y - \frac{\epsilon}{2\pi} \sin(2\pi x)). \end{aligned}$$

Below, we summarize the results obtained using the various codes applied to the standard map.

Code 1: With $\epsilon = 0.5$ and the initial condition $(x_0, y_0) = (0.1, 0)$, the rotation number of the corresponding librational orbit is computed as $\omega = 0.8883181985894327$.

Code 2: The output of Code 2 is an initial approximation of the librational orbit with rotation number ω, shown in Figure 5.1.

Code 3: Using the curve from Code 2 as input, we study the persistence of the librational orbit as ϵ increases in increments of $\Delta\epsilon = 0.01$. The results show that the curve persists for ϵ values ranging from 0.5 to 0.98. We also modified Code 3 to study the continuation of the curve as ϵ decreases, with an increment of $\Delta\epsilon = -0.01$. With the same initial approximation, the curve persists from $\epsilon = 0.5$ to $\epsilon = 0.48$. All approximations obtained during this process for each value of ϵ are shown in Figure 5.2. Moreover, in Figure 5.3, we show the continuation of the librational orbit for four different values of ϵ against the dynamics of the standard map.

Code 4: With the same initial guess, Code 4 produces results identical to those of Code 3, demonstrating that the curve persists from $\epsilon = 0.5$ to $\epsilon = 0.98$. In Table 5.1, for some of the invariant curves obtained with a specific value of ϵ, we show a comparison of the number of Fourier modes and the error E for both methods. The results show that they perform almost identically.

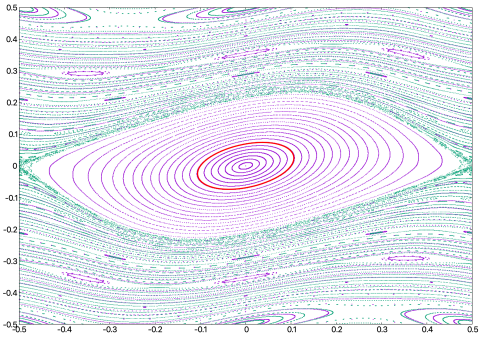


Figure 5.1: In red, with $\epsilon = 0.5$, initial approximation of the studied librational orbit.

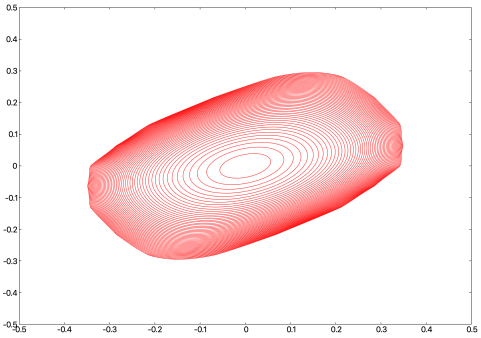
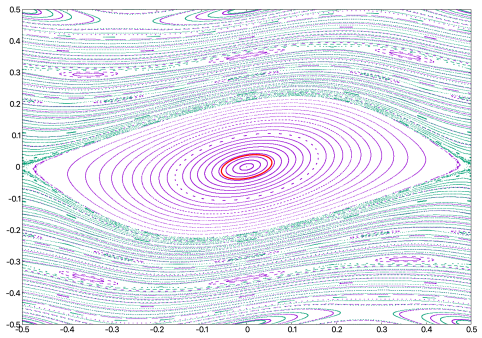
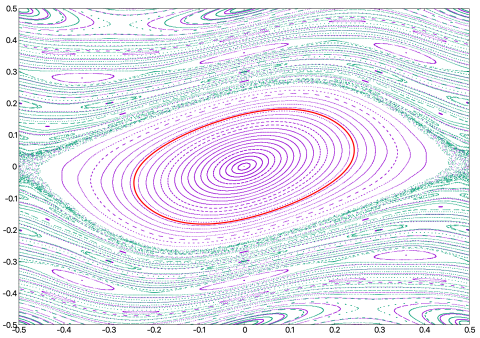


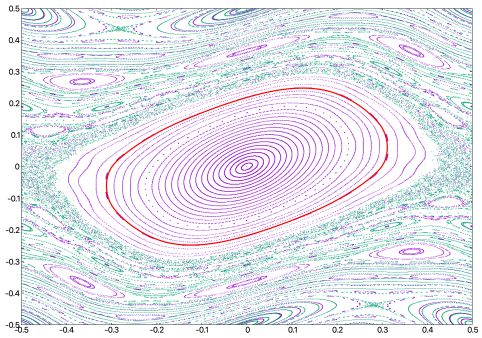
Figure 5.2: Continuation of the studied librational orbit with values of ϵ ranging from 0.48 to 0.98.



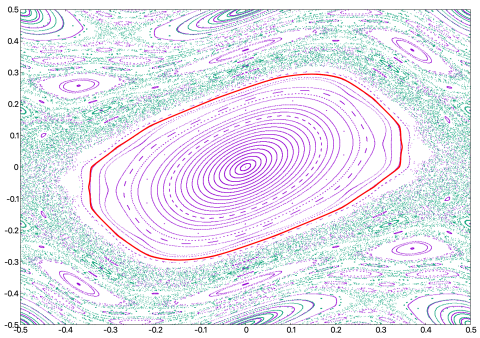
(a) $\epsilon = 0.48$



(b) $\epsilon = 0.65$



(c) $\epsilon = 0.82$



(d) $\epsilon = 0.98$

Figure 5.3: In red, the continuation of the librational orbit with rotation number $\omega = 0.8883181985894327$ through different values of ϵ.

	Parameterization Method		Balanced Parameterization Method	
ϵ	Fourier modes	E	Fourier modes	E
0.48	128	1.074574591520889e-15	128	1.265355234697337e-15
0.50	64	7.178676993783525e-16	64	7.374834361046558e-16
0.52	128	3.519059123520587e-15	128	3.734085796986845e-15
0.54	128	1.172172397917295e-15	128	1.269949830310100e-15
0.56	128	7.942573690876034e-11	128	7.562867963383475e-11
0.58	128	1.304418522377109e-11	128	1.246007718964801e-11
0.60	128	2.838575494313423e-12	128	2.731029526150514e-12
0.62	128	7.520961982863491e-13	128	7.309078172068545e-13
0.64	128	2.325636719865173e-13	128	2.287806019160406e-13
0.66	256	8.263478266170098e-14	256	8.292074905781875e-14
0.68	256	3.554314015830569e-14	256	3.627393233198803e-14
0.70	256	1.859591074201832e-14	256	1.933363920828944e-14
0.72	256	1.259489609465639e-14	256	1.261873699518996e-14
0.74	256	1.133563607878040e-14	256	1.153136616299137e-14
0.76	256	1.197105618636198e-14	256	1.119125227015857e-14
0.78	256	1.446498421743053e-14	256	1.563819235740612e-14
0.80	512	2.352429854132137e-14	512	2.364110019525797e-14
0.82	512	4.201567010158711e-14	512	4.256594969102212e-14
0.84	512	9.251395359862613e-14	512	9.278521850302439e-14
0.86	512	2.382984918699916e-13	512	2.404957727106839e-13
0.88	512	7.169945781313939e-13	512	7.180567678769527e-13
0.90	1024	2.193830719306903e-12	1024	2.202593363803408e-12
0.92	1024	5.877983039824475e-12	1024	5.906981360780706e-12
0.94	1024	1.194977135824353e-11	1024	1.204518316702534e-11
0.96	2048	2.109604421110672e-11	2048	2.105422697064340e-11
0.98	8192	3.048790308934882e-12	4096	3.349439974922291e-12

Table 5.1: Comparison of the results obtained with the parameterization method (using Code 3) and the balanced parameterization method (using Code 4). For each value of ϵ, we show the number of Fourier modes used and the error E obtained for both methods.

5.3 Hénon Map

Let us now consider the Hénon map, given by

$$\begin{array}{rccc} H_\alpha: & \mathbb{R}^2 & \longrightarrow & \mathbb{R}^2 \\ & \begin{pmatrix} x \\ y \end{pmatrix} & \longmapsto & \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x \\ y - x^2 \end{pmatrix}. \end{array}$$

The results obtained executing the codes for the Hénon map are summarized below.

Code 1: For $\alpha = 1.328430475755933$ and the initial condition $(x_0, y_0) = (0.4, 0)$, Code 1 computes the rotation number of the associated librational orbit as $\omega = 0.2061745148657143$.

Code 2: A first approximation for the librational orbit with rotation number ω, obtained with Code 2, is shown in Figure 5.4.

Code 3: Using this approximation as input, Code 3 examines the persistence of the librational orbit as α increases in increments of $\Delta\alpha = 0.01$. Results show that the curve persists from $\alpha = 1.328430475755933$ to $\alpha = 1.388430475755933$. And as α decreases, with an increment of $\Delta\alpha = -0.01$, the curve persists from $\alpha = 1.328430475755933$ to $\alpha = 1.298430475755933$. All approximations obtained during this process are shown in Figure 5.5. Moreover, in Figure 5.6, we show the continuation of the librational orbit for four different values of α against the dynamics of the Hénon map.

Code 4: Finally, this code repeats the analysis performed by Code 3, confirming that the same results are obtained for both increasing and decreasing values of α. Table 5.2 shows a comparison of the number of Fourier modes and the error E for each invariant curve obtained with a specific value of α. Again, the results show that the two methods perform almost identically.

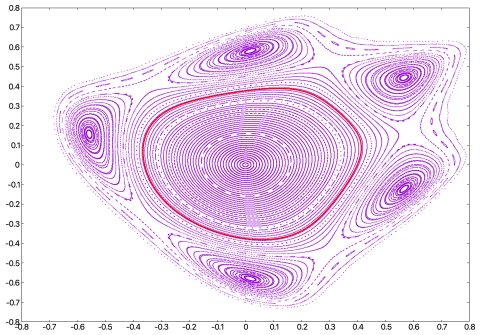


Figure 5.4: In red, with $\alpha = 1.328430475755933$, initial approximation of the studied librational orbit.

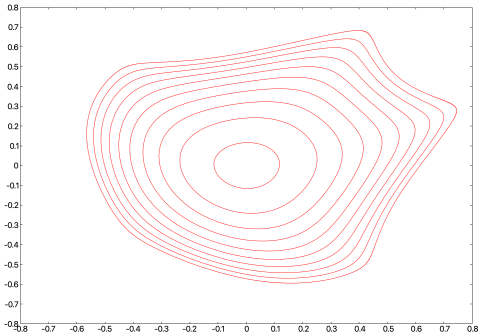
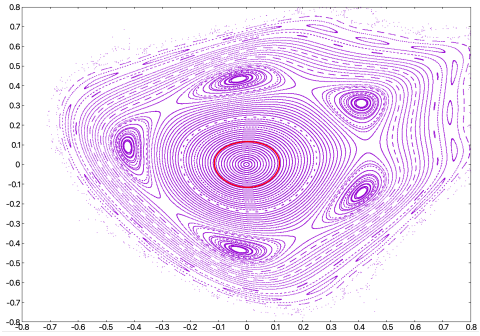
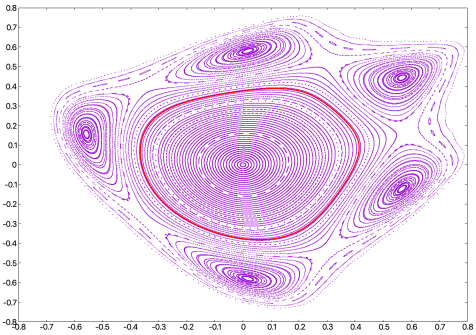


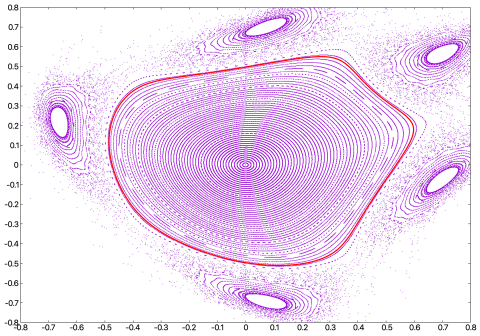
Figure 5.5: Continuation of the studied librational orbit with values of α ranging from 1.298430475755933 to 1.388430475755933.



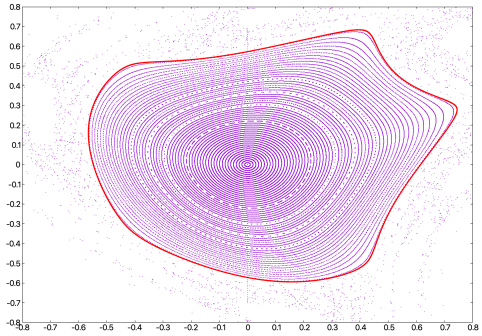
(a) $\alpha = 1.298430475755933$



(b) $\alpha = 1.328430475755933$



(c) $\alpha = 1.358430475755933$



(d) $\alpha = 1.388430475755933$

Figure 5.6: In red, the continuation of the librational orbit with rotation number $\omega = 0.2061745148657143$ through different values of α.

	Parameterization Method		**Balanced Parameterization Method**	
α	Fourier modes	E	Fourier modes	E
1.2984	256	1.468239862419898e-15	256	2.024099854373832e-15
1.3084	256	1.964855577695347e-15	256	1.590675775713342e-15
1.3184	256	2.888449837437586e-12	256	3.101615097619397e-12
1.3284	128	9.635875355667767e-13	128	9.882913377259651e-13
1.3384	256	8.025064859060146e-12	256	7.995466567395057e-12
1.3484	256	7.533514183442294e-13	256	7.617443955223730e-13
1.3584	512	2.273982341330406e-13	512	2.354624372893241e-13
1.3684	512	1.635755321454707e-12	512	1.570792254667962e-12
1.3784	2048	4.172723245864120e-13	2048	4.897947603314123e-13
1.3884	8192	2.300973959370269e-11	8192	3.635949565883142e-11

Table 5.2: For each value of α, we show the number of Fourier modes used and the error E obtained for both methods.

5.4 Conclusions

In this chapter, we explored the practical use of the parameterization method and the balanced parameterization method on two well-known systems: the standard map and the Hénon map. Our findings confirm that both methods have consistent results capturing the persistence of librational orbits over a range of perturbation parameter values. The Fourier modes required by each method were generally identical, with only one exception where the balanced parameterization method required fewer modes (see Table 5.1). Additionally, both methods achieved nearly the same level of accuracy.

These results show that the balanced parameterization method performs comparably to the traditional parameterization approach, further validating its applicability.

Appendix A

Codes Used For the Numerical Examples

A.1 Input files

The following files, obtained using codes from [4], contain the initial approximation of the librational orbit for the standard map and the Hénon map. These files are used as input for Codes 3 and 4.

A.1: `standardK.txt`

```
1.0e-12
1.0e-10
1.0e-16
8.883181985894327e-01
0.5
32
1e-2
1
+1.05081826e-01 +1.39503911e-02
+1.06223619e-01 +2.74675452e-02
+1.03391698e-01 +4.01031395e-02
+9.66701190e-02 +5.13877645e-02
+8.62651245e-02 +6.08411404e-02
+7.25131877e-02 +6.80024009e-02
+5.58880328e-02 +7.24787965e-02
+3.70013558e-02 +7.40027111e-02
+1.65907642e-02 +7.24787966e-02
-4.51078630e-03 +6.80024011e-02
-2.54239835e-02 +6.08411406e-02
-4.52823541e-02 +5.13877647e-02
-6.32885577e-02 +4.01031398e-02
-7.87560733e-02 +2.74675455e-02
```

```
-9.11314351e-02 +1.39503915e-02
-9.99999999e-02 +1.72241951e-10
-1.05081826e-01 -1.39503911e-02
-1.06223619e-01 -2.74675452e-02
-1.03391698e-01 -4.01031395e-02
-9.66701190e-02 -5.13877645e-02
-8.62651245e-02 -6.08411404e-02
-7.25131877e-02 -6.80024009e-02
-5.58880328e-02 -7.24787965e-02
-3.70013558e-02 -7.40027111e-02
-1.65907642e-02 -7.24787966e-02
+4.51078630e-03 -6.80024011e-02
+2.54239835e-02 -6.08411406e-02
+4.52823541e-02 -5.13877647e-02
+6.32885577e-02 -4.01031398e-02
+7.87560733e-02 -2.74675455e-02
+9.11314351e-02 -1.39503915e-02
+9.99999999e-02 -1.72090101e-10
```

A.2: `henonK.txt`

```
1.0e-12
1.0e-10
1.0e-16
2.061745148657143e-01
1.328430475755933e+00
32
1e-2
1
+4.13547861e-01 +6.59325304e-02
+4.10738519e-01 +1.24097782e-01
+3.92027698e-01 +1.78006443e-01
+3.60565723e-01 +2.30754748e-01
+3.21388407e-01 +2.82054298e-01
+2.78497718e-01 +3.27650061e-01
+2.32035554e-01 +3.62234828e-01
+1.78710529e-01 +3.82713680e-01
+1.14621771e-01 +3.88860017e-01
+3.85386490e-02 +3.82569808e-01
-4.56376616e-02 +3.67121356e-01
-1.29535412e-01 +3.45658745e-01
-2.04811152e-01 +3.18836471e-01
-2.67084264e-01 +2.84045306e-01
-3.15350585e-01 +2.36988433e-01
-3.48772728e-01 +1.74152060e-01
-3.64940790e-01 +9.56122704e-02
-3.61581792e-01 +7.02270956e-03
-3.39244696e-01 -8.21991378e-02
-3.00605631e-01 -1.64187488e-01
```

```
-2.47562593e-01 -2.35682012e-01
-1.80667729e-01 -2.95978056e-01
-1.01750612e-01 -3.43129337e-01
-1.67084392e-02 -3.72871225e-01
+6.55075802e-02 -3.81614763e-01
+1.37683170e-01 -3.69346309e-01
+1.97791843e-01 -3.38306283e-01
+2.48319351e-01 -2.90356559e-01
+2.93329383e-01 -2.27168012e-01
+3.35199667e-01 -1.52697926e-01
+3.72344296e-01 -7.43174841e-02
+4.00000000e-01 +9.14182098e-13
```

A.2 Code 3

Both Code 3 and Code 4 use the C++ classes "grid.h", "matrix.h" and "complex.h" from [1], which can be found here. To run the code for the Hénon map, make sure to comment line 117 and uncomment line 118 and then run the program with the following instructions

```
g++ -O3 -c param.cc
g++ -O3 -o param.exe param.o -lm
./param.exe henonK.txt
```

To run it with the standard map, execute `./param.exe standardK.txt` instead.

A.3: `param.cc`

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <math.h>
#include <cstring>
#include "complex.h"
#include "grid.h"
#include "matrix.h"

using namespace std;

#define myreal double

#define DTOR (1)    // Dimension of the invariant torus
#define DMAP (2)    // Dimension of phase space
#define MNEW (10)    // Maximum number of Newton iterations
#define MAXF (8192) // Maximum number of Fourier coefficients allowed

int position(int *nn, int *index, int ndim);
```

```
21 void indices(int pos, int *nn, int *index, int ndim);
22 void realloc_torus(matrix &paramR, matrix &paramF,
23    matrix &paramR0, matrix &paramF0,
24    int &nn, int &newnn);
25 int kam_torus(matrix &paramR, matrix &paramF, myreal omega, myreal &
       error, int nn,
26 int &tail0,
27 void (*map)(complex *, complex *, complex **));
28 void map_standard(complex *z, complex *fz, complex **Dfz);
29 void map_henon(complex *z, complex *fz, complex **Dfz);
30
31 /* General global variables */
32 myreal pi, pi2, val0, val05, val1, val2, val3, val4, val5, val10;
33 myreal global_twist; // Norm of inverse of torsion matrix T(theta)
34 myreal toltail;     // Tolerance on the size of the tails of the
       parameterization
35 myreal tolinva;     // Tolerance on the error of invariance
36 myreal tolinte;     // Tolerance on intermediate computation (e.g.
       matrix inverses)
37
38 /* Global variables of the problem */
39 myreal epsilon; // Continuation parameter
40
41 fstream file_torus, file_input;
42
43 int main(int argc, char *argv[])
44 {
45    char name[80];
46
47    int nn, index, newnn;
48    complex *z, *fz, **Dfz;
49    myreal omega, tol;
50    myreal error, deps, epsilon0;
51    matrix paramR, paramF, paramR0, paramF0;
52    int tail0;
53    int conv, iter;
54
55    /**** START We set global variables ****/
56    val0=0.0;
57    val1=1.0;
58    val2=2.0;
59    val05=val1/val2;
60    val3=3.0;
61    val4=4.0;
62    val5=5.0;
63    val10=10.0;
64    pi2=6.2831853071795864769252867665590057684;
65    /**** END   We set global variables ****/
```

```
66
67    cout << scientific;
68    cout.precision(15);
69    /**** START We read an invariant torus from the file ****/
70    file_input.open(argv[1],ios::in);
71    file_input << scientific;
72    file_input >> toltail;
73    file_input >> tolinva;
74    file_input >> tolinte;
75    start_matrix(tolinte);
76    start_grid(tolinte);
77    file_input >> omega;
78    int auxint;
79    file_input >> epsilon;
80    file_input >> nn;
81    file_input >> deps;
82    file_input >> auxint;
83    paramR=matrix(DMAP,1,DTOR,&nn); // Memory for the torus
84    if(auxint!=0){
85       for (int l=0;l<nn;l++){
86          for (int j=0;j<DMAP;j++){
87             paramR.coef[j][0].elem[l] = val0;
88             file_input >> paramR.coef[j][0].elem[l].real;
89          }
90       }
91    } else {
92       for (int l=0;l<nn;l++){
93          paramR.coef[0][0].elem[l] = val0;
94          paramR.coef[1][0].elem[l] = omega;
95       }
96    }
97    /**** END   We read an invariant torus from the file ****/
98    file_input.close();
99
100   paramF=fft_F(paramR);
101   paramR0=matrix(paramR);
102   paramF0=matrix(paramF);
103   epsilon0=epsilon;
104
105   /**** START Continuation with respect to epsilon ****/
106   int fail=0;
107   do {
108      paramR=paramR0;
109      paramF=paramF0;
110      epsilon=epsilon0;
111
112      /**** START Newton method to correct the invariant torus ****/
113      cout << "# We try to compute the torus for epsilon=" << epsilon
```

```
   << endl;
114      iter=0;
115      do {
116         cout << "# Iteration " << iter+1 << " : " << endl;
117         conv = kam_torus(paramR,paramF,omega,error,nn,tail0,
   map_standard);
118         //conv = kam_torus(paramR,paramF,omega,error,nn,tail0,
   map_henon);
119         if (tail0==1){
120            /* If the tail is too large, we refine the grid and
   restart the computation */
121            if (nn==MAXF) {
122               cout << "# We reached the maximum number of Fourier
   modes" << endl;
123               return 0;
124            }
125            newnn=2*nn;
126
127            realloc_torus(paramR,paramF,paramR0,paramF0,nn,newnn);
128         }
129         iter++;
130      } while (conv==0 && iter<MNEW && tail0==0);
131      /**** END   Newton method to correct the invariant torus ****/
132
133      if(conv==1) {
134         cout.precision(15);
135         /* If we converge, we update the last computed torus and
   store the results */
136         paramR0=paramR;
137         paramF0=paramF;
138         epsilon0=epsilon + deps;
139
140         /**** START We print the information related to the
   successful continuation step ****/
141         cout << epsilon << " ";
142         cout << nn << " ";
143         cout << global_twist << " ";
144         cout << normsobo(paramF.coef[0][0],val2) << " ";
145         cout << error << endl;
146         /**** END   We print the information related to the
   successful continuation step ****/
147
148         /**** START We save the computed invariant torus in a
   separated file ****/
149         sprintf(name,"output_torus%.4lf",epsilon);
150         file_torus.open(name,ios::out);
151         file_torus << scientific;
152         file_torus.precision(15);
```

```
153          file_torus << toltail << endl;
154          file_torus << tolinva << endl;
155          file_torus << tolinte << endl;
156          file_torus << omega << endl;
157          file_torus << epsilon << endl;
158          file_torus << nn << endl;
159          file_torus << deps << endl;
160          file_torus << 1 << endl;
161          z=new complex [DMAP];
162          for (int l=0;l<nn;l++){
163             file_torus << l << " ";
164
165             z[0]= paramR.coef[0][0].elem[l];
166             z[1]= paramR.coef[1][0].elem[l];
167
168             file_torus << z[0].real << " ";
169             file_torus << z[1].real << endl;
170          }
171          file_torus.close();
172          delete [] z;
173          /**** END   We save the computed invariant torus in a
     separated file ****/
174       } else {
175          if (tail0==1) cout << "# We need more Fourier modes" << endl;
176          else{
177             cout << "# Newton method does not converge" << endl;
178             deps=deps/val10;
179             if (deps<1e-5) fail=1;
180          }
181          cout << "##########################################" << endl
     ;
182       }
183    } while (fail==0);
184    /**** END Continuation with respect to epsilon ****/
185
186    return 0;
187 }
188
189 void realloc_torus(matrix &paramR, matrix &paramF,
190    matrix &paramR0, matrix &paramF0,
191    int &nn, int &newnn)
192 {
193    matrix newparamF;
194    int pos;
195
196    newparamF=matrix(DMAP,1,DTOR,&newnn);
197
198    for(int l=0;l<nn;l++){
```

```
199         pos = (l >= nn/2) ? l+nn : l;
200         for(int j=0;j<DMAP;j++)
201            newparamF.coef[j][0].elem[pos] = paramF.coef[j][0].elem[l];
202      }
203      paramF=newparamF;
204
205      for(int l=0;l<nn;l++){
206         pos = (l >= nn/2) ? l+nn : l;
207         for(int j=0;j<DMAP;j++)
208            newparamF.coef[j][0].elem[pos] = paramF0.coef[j][0].elem[l];
209      }
210      paramF0=newparamF;
211
212      paramR=fft_B(paramF);
213      paramR0=fft_B(paramF0);
214
215      nn = newnn;
216   }
217
218   int kam_torus(matrix &paramR, matrix &paramF, myreal omega, myreal &
         error, int nn,
219      int &tail0,
220      void (*map)(complex *, complex *, complex **))
221   {
222      complex *z, *fz, **Dfz;
223      myreal **twist0, **neweta0;
224      myreal **invT, tolqr, **xiN0, **xiL0;
225      myreal aux;
226      myreal tg;
227      matrix FparamR(DMAP,1,DTOR,&nn);
228      matrix ErrorR(DMAP,1,DTOR,&nn);
229      matrix ErrorF(DMAP,1,DTOR,&nn);
230      matrix DFKR (DMAP,DMAP,DTOR,&nn);
231      matrix OmegaR (DMAP,DMAP,DTOR,&nn);
232      matrix OmegaF (DMAP,DMAP,DTOR,&nn);
233      matrix KshiftF(DMAP,1,DTOR,&nn);
234      matrix KshiftR(DMAP,1,DTOR,&nn);
235      matrix DparamF(DMAP,DTOR,DTOR,&nn);
236      matrix DparamR(DMAP,DTOR,DTOR,&nn);
237      matrix LR(DMAP,DTOR,DTOR,&nn);
238      matrix LF(DMAP,DTOR,DTOR,&nn);
239      matrix LshiftR(DMAP,DTOR,DTOR,&nn);
240      matrix LshiftF(DMAP,DTOR,DTOR,&nn);
241      matrix GR(DTOR,DTOR,DTOR,&nn);
242      matrix BR(DTOR,DTOR,DTOR,&nn);
243      matrix NR(DMAP,DTOR,DTOR,&nn);
244      matrix NF(DMAP,DTOR,DTOR,&nn);
245      matrix NshiftR(DMAP,DTOR,DTOR,&nn);
```

```
246    matrix NshiftF(DMAP,DTOR,DTOR,&nn);
247    matrix etaLR(DTOR,1,DTOR,&nn);
248    matrix etaNR(DTOR,1,DTOR,&nn);
249    matrix etaLF(DTOR,1,DTOR,&nn);
250    matrix etaNF(DTOR,1,DTOR,&nn);
251    matrix RetaNR(DTOR,1,DTOR,&nn);
252    matrix RetaNF(DTOR,1,DTOR,&nn);
253    matrix newetaR(DTOR,1,DTOR,&nn);
254    matrix newetaF(DTOR,1,DTOR,&nn);
255    matrix xiNR(DTOR,1,DTOR,&nn);
256    matrix xiLR(DTOR,1,DTOR,&nn);
257    matrix xiLF(DTOR,1,DTOR,&nn);
258    matrix twistR(DTOR,DTOR,DTOR,&nn);
259    matrix newparamR(DMAP,1,DTOR,&nn);
260    matrix newparamF(DMAP,1,DTOR,&nn);
261    complex **Omega;
262
263    z=new complex [DMAP];
264    fz=new complex [DMAP];
265    Dfz = new complex*[DMAP];
266    Omega = new complex*[DMAP];
267    for(int i=0;i<DMAP;i++){
268       Dfz[i]=new complex[DMAP];
269       Omega[i]=new complex[DMAP];
270    }
271    twist0 = new myreal *[DTOR];
272    neweta0 = new myreal *[DTOR];
273    xiN0 = new myreal *[DTOR];
274    xiL0 = new myreal *[DTOR];
275    for (int i=0;i<DTOR;i++){
276       twist0[i] = new myreal [DTOR];
277       neweta0[i]= new myreal [1];
278       xiN0[i]= new myreal [1];
279       xiL0[i]= new myreal [1];
280    }
281    invT=new myreal*[DTOR];
282    for (int i=0;i<DTOR;i++){
283       invT[i]=new myreal[DTOR];
284    }
285
286    Omega[0][0] = val0;
287    Omega[0][1] = -val1;
288    Omega[1][0] = val1;
289    Omega[1][1] = val0;
290
291    /*****************************************************************
292    STEP 0 Evaluation of the tail
293    *****************************************************************/
```

```
294   tail(paramF,&tg);
295
296   tail0=0;
297   cout.precision(3);
298   cout << "#    - Size of the grid: ";
299   cout << nn << " ";
300   cout << endl;
301   cout << "#    - Tail of the parameterization: ";
302   cout << tg << " ";
303   if (tg>toltail) {tail0=1;}
304
305   cout << endl;
306   if (tail0==1) return 0;
307   clean(paramF);
308   paramR=fft_B(paramF);
309
310   /*****************************************************************
311   STEP 1 Evaluation of the invariance error
312   *****************************************************************/
313   for (int l=0;l<nn;l++){
314
315      z[0]= paramR.coef[0][0].elem[l];
316      z[1]= paramR.coef[1][0].elem[l];
317
318      (*map)(z,fz,Dfz);
319      for(int i=0;i<DMAP;i++){
320         FparamR.coef[i][0].elem[l]=fz[i];
321         for(int j=0;j<DMAP;j++){
322            DFKR.coef[i][j].elem[l]=Dfz[i][j];
323            OmegaR.coef[i][j].elem[l]=Omega[i][j];
324         }
325      }
326   }
327
328   KshiftF=shift(paramF,&omega);
329   KshiftR=fft_B(KshiftF);
330
331   ErrorR = FparamR - KshiftR;
332
333   ErrorF = fft_F(ErrorR);
334   error = norm(ErrorF);
335   cout << "#    - Error of invariance: ";
336   cout << error << endl;
337
338   if (error<tolinva){
339      cout << "#    - No correction is needed!" << endl;
340      for(int i=0;i<DMAP;i++){
341         delete [] Dfz[i];
```

```
342       }
343       delete [] z;
344       delete [] fz;
345       delete [] Dfz;
346       for (int i=0;i<DTOR;i++){
347          delete [] twist0[i];
348          delete [] neweta0[i];
349          delete [] invT[i];
350          delete [] xiN0[i];
351          delete [] xiL0[i];
352       }
353       delete [] twist0;
354       delete [] neweta0;
355       delete [] invT;
356       delete [] xiN0;
357       delete [] xiL0;
358
359       return 1;
360    }
361
362    /*****************************************************************
363    STEP 2 Construction of the symplectic frame
364    *****************************************************************/
365    DparamF = diff(paramF);
366    DparamR = fft_B(DparamF);
367
368    LR=DparamR;
369
370    GR = trans(LR)*LR;
371    BR = inv(GR);
372    NR = OmegaR*LR*BR;
373
374    LF = fft_F(LR);
375    NF = fft_F(NR);
376
377    /*****************************************************************
378    STEP 3 Computation of the correction on the symplectic frame
379    *****************************************************************/
380    LshiftF=shift(LF,&omega);
381    LshiftR=fft_B(LshiftF);
382    NshiftF=shift(NF,&omega);
383    NshiftR=fft_B(NshiftF);
384    OmegaF=fft_F(OmegaR);
385    etaLR  = -trans(NshiftR)*OmegaR*ErrorR;
386    etaNR  =  trans(LshiftR)*OmegaR*ErrorR;
387    twistR = trans(NshiftR)*OmegaR*DFKR*NR;
388    etaNF  = fft_F(etaNR);
389    RetaNF = cohomological(etaNF,&omega);
```

```
390    RetaNR = fft_B(RetaNF);
391    newetaR= etaLR-twistR*RetaNR;
392
393    aver(twistR,twist0);
394    invT[0][0]=1./twist0[0][0];
395    global_twist=invT[0][0];
396
397    aver(newetaR,neweta0);
398    cout << "#    - Norm inverse twist: ";
399    cout << global_twist << endl;
400
401    xiN0[0][0] = invT[0][0]*neweta0[0][0];
402    xiNR = RetaNR;
403
404    for(int l=0;l<nn;l++){
405       xiNR.coef[0][0].elem[l] = xiNR.coef[0][0].elem[l] + xiN0[0][0];
406    }
407
408    newetaR= etaLR-twistR*xiNR;
409    newetaF= fft_F(newetaR);
410    xiLF   = cohomological(newetaF,&omega);
411    xiLR   = fft_B(xiLF);
412
413    /*****************************************************************
414    STEP 4 New parameterization
415    *****************************************************************/
416    newparamR = paramR + LR*xiLR + NR*xiNR;
417    newparamF = fft_F(newparamR);
418
419    for(int i=0;i<DMAP;i++){
420       delete [] Dfz[i];
421       delete [] Omega[i];
422    }
423    for (int i=0;i<DTOR;i++){
424       delete [] twist0[i];
425       delete [] neweta0[i];
426       delete [] invT[i];
427       delete [] xiN0[i];
428       delete [] xiL0[i];
429    }
430    delete [] z;
431    delete [] fz;
432    delete [] Dfz;
433    delete [] Omega;
434    delete [] twist0;
435    delete [] neweta0;
436    delete [] invT;
437    delete [] xiN0;
```

```
438     delete [] xiL0;
439 
440     paramR=newparamR;
441     paramF=newparamF;
442 
443     /* This is just to show the size of the correction */
444     newparamR = LR*xiLR + NR*xiNR;
445     newparamF = fft_F(newparamR);
446     aux = norm(newparamF);
447     cout << "#    - Norm of the correction: ";
448     cout << aux << endl;
449 
450     return 0;
451 }
452 
453 void map_standard(complex *z, complex *fz, complex **Dfz)
454 {
455     fz[1] = z[1] - epsilon*sin(pi2*z[0])/pi2;
456     fz[0] = z[0] + fz[1];
457 
458     Dfz[1][0] = - epsilon*cos(pi2*z[0]);
459     Dfz[1][1] = val1;
460 
461     Dfz[0][0] = val1 + Dfz[1][0];
462     Dfz[0][1] = Dfz[1][1];
463 }
464 
465 void map_henon(complex *z, complex *fz, complex **Dfz)
466 {
467     fz[0] = z[0]*cos(epsilon)-(z[1]-z[0]*z[0])*sin(epsilon);
468     fz[1] = z[0]*sin(epsilon)+(z[1]-z[0]*z[0])*cos(epsilon);
469 
470     Dfz[1][0] = sin(epsilon) - 2*z[0]*cos(epsilon);
471     Dfz[1][1] = cos(epsilon);
472 
473     Dfz[0][0] = cos(epsilon) + 2*z[0]*sin(epsilon);
474     Dfz[0][1] = -sin(epsilon);
475 }
```

A.3 Code 4

To run Code 4, make sure to download "grid.h", "matrix.h" and "complex.h" here. For the Hénon map example, comment line 124 and uncomment line 125 and then execute the program with the following instructions

```
g++ -O3 -c beta.cc
g++ -O3 -o beta.exe beta.o -lm
```

```
./beta.exe henonK.txt
```

To run it with the standard map, execute `./beta.exe standardK.txt` instead.

A.4: `beta.cc`

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <math.h>
#include <cstring>
#include "complex.h"
#include "grid.h"
#include "matrix.h"

using namespace std;

#define myreal double

#define DTOR (1)   // Dimension of the invariant torus
#define DMAP (2)   // Dimension of phase space
#define MNEW (10)   // Maximum number of Newton iterations
#define MAXF (8192) // Maximum number of Fourier coefficients allowed

int position(int *nn, int *index, int ndim);
void indices(int pos, int *nn, int *index, int ndim);
void realloc_torus(matrix &paramR, matrix &paramF,
   matrix &paramR0, matrix &paramF0,
   int &nn, int &newnn);
int kam_torus(matrix &paramR, matrix &paramF, myreal omega, myreal &
    error, int nn,
int &tail0, myreal &beta,
void (*map)(complex *, complex *, complex **));
void map_standard(complex *z, complex *fz, complex **Dfz);
void map_henon(complex *z, complex *fz, complex **Dfz);
void linear (myreal A[2][2], myreal B[2], myreal &x, myreal &y);
grid cohomologicalBeta(const grid& g, double *omega, double a);
matrix cohomologicalBeta(const matrix& m, double *omega, double a);

/* General global variables */
myreal pi, pi2, val0, val05, val1, val2, val3, val4, val5, val10;
myreal global_twist; // Norm of inverse of torsion matrix T(theta)
myreal toltail;     // Tolerance on the size of the tails of the
    parameterization
myreal tolinva;     // Tolerance on the error of invariance
myreal tolinte;     // Tolerance on intermediate computation (e.g.
    matrix inverses)

/* Global variables of the problem */
```

```
42 myreal epsilon; // Continuation parameter
43
44 fstream file_torus, file_input;
45
46 int main(int argc, char *argv[])
47 {
48    char name[80];
49
50    int nn, index, newnn;
51    complex *z, *fz, **Dfz;
52    myreal omega, tol;
53    myreal error, deps, epsilon0;
54    matrix paramR, paramF, paramR0, paramF0;
55    myreal beta, beta0;
56    int tail0;
57    int conv, iter;
58
59    /**** START We set global variables ****/
60    val0=0.0;
61    val1=1.0;
62    val2=2.0;
63    val05=val1/val2;
64    val3=3.0;
65    val4=4.0;
66    val5=5.0;
67    val10=10.0;
68    pi2=6.28318530717958647692528676655900576 84;
69    /**** END   We set global variables ****/
70
71    cout << scientific;
72    cout.precision(15);
73    /**** START We read an invariant torus from the file ****/
74    file_input.open(argv[1],ios::in);
75    file_input << scientific;
76    file_input >> toltail;
77    file_input >> tolinva;
78    file_input >> tolinte;
79    start_matrix(tolinte);
80    start_grid(tolinte);
81    file_input >> omega;
82    int auxint;
83    file_input >> epsilon;
84    file_input >> nn;
85    file_input >> deps;
86    file_input >> auxint;
87    paramR=matrix(DMAP,1,DTOR,&nn); // Memory for the torus
88    if(auxint!=0){
89       for (int l=0;l<nn;l++){
```

```
90          for (int j=0;j<DMAP;j++){
91             paramR.coef[j][0].elem[l] = val0;
92             file_input >> paramR.coef[j][0].elem[l].real;
93          }
94       }
95    } else {
96       for (int l=0;l<nn;l++){
97          paramR.coef[0][0].elem[l] = val0;
98          paramR.coef[1][0].elem[l] = omega;
99       }
100   }
101   /**** END   We read an invariant torus from the file ****/
102   file_input.close();
103
104   paramF=fft_F(paramR);
105   paramR0=matrix(paramR);
106   paramF0=matrix(paramF);
107   epsilon0=epsilon;
108   beta0 = 0;
109
110   /**** START Continuation with respect to epsilon ****/
111   int fail=0;
112   do {
113      paramR=paramR0;
114      paramF=paramF0;
115      epsilon = epsilon0;
116      beta = beta0;
117
118      /**** START Newton method to correct the invariant torus ****/
119      cout << "# We try to compute the torus for epsilon=" << epsilon
     << endl;
120      iter=0;
121      do {
122         cout << "# Iteration " << iter+1 << " : " << endl;
123         conv = kam_torus(paramR,paramF,omega,error,nn,tail0, beta,
     map_standard);
124         //conv = kam_torus(paramR,paramF,omega,error,nn,tail0, beta,
     map_henon);
125         if (tail0==1){
126            /* If the tail is too large, we refine the grid and
     restart the computation */
127            if (nn==MAXF) {
128               cout << "# We reached the maximum number of Fourier
     modes" << endl;
129               return 0;
130            }
131            newnn=2*nn;
132
```

```
133             realloc_torus(paramR,paramF,paramR0,paramF0,nn,newnn);
134          }
135          iter++;
136       } while (conv==0 && iter<MNEW && tail0==0);
137       /**** END   Newton method to correct the invariant torus ****/
138
139       if(conv==1) {
140          cout.precision(15);
141          /* If we converge, we update the last computed torus and
    store the results */
142          paramR0=paramR;
143          paramF0=paramF;
144          epsilon0=epsilon + deps;
145          beta0 = beta;
146
147          /**** START We print the information related to the
    successful continuation step ****/
148          cout << epsilon << " ";
149          cout << nn << " ";
150          cout << global_twist << " ";
151          cout << normsobo(paramF.coef[0][0],val2) << " ";
152          cout << error << endl;
153          /**** END   We print the information related to the
    successful continuation step ****/
154
155          /**** START We save the computed invariant torus in a
    separated file ****/
156          sprintf(name,"output_torus%.4lf",epsilon);
157          file_torus.open(name,ios::out);
158          file_torus << scientific;
159          file_torus.precision(15);
160          file_torus << toltail << endl;
161          file_torus << tolinva << endl;
162          file_torus << tolinte << endl;
163          file_torus << omega << endl;
164          file_torus << epsilon << endl;
165          file_torus << nn << endl;
166          file_torus << deps << endl;
167          file_torus << 1 << endl;
168          z=new complex [DMAP];
169          for (int l=0;l<nn;l++){
170             file_torus << l << " ";
171
172             z[0]= paramR.coef[0][0].elem[1];
173             z[1]= paramR.coef[1][0].elem[1];
174
175             file_torus << z[0].real << " ";
176             file_torus << z[1].real << endl;
```

```
177          }
178          file_torus.close();
179          delete [] z;
180          /**** END   We save the computed invariant torus in a
     separated file ****/
181       } else {
182          if (tail0==1) cout << "# We need more Fourier modes" << endl;
183          else{
184             cout << "# Newton method does not converge" << endl;
185             deps=deps/val10;
186             if (deps<1e-5) fail=1;
187          }
188          cout << "##########################################" << endl
     ;
189       }
190    } while (fail==0);
191    /**** END Continuation with respect to epsilon ****/
192
193    return 0;
194 }
195
196 void realloc_torus(matrix &paramR, matrix &paramF,
197    matrix &paramR0, matrix &paramF0,
198    int &nn, int &newnn)
199 {
200    matrix newparamF;
201    int pos;
202
203    newparamF=matrix(DMAP,1,DTOR,&newnn);
204
205    for(int l=0;l<nn;l++){
206       pos = (l >= nn/2) ? l+nn : l;
207       for(int j=0;j<DMAP;j++)
208          newparamF.coef[j][0].elem[pos] = paramF.coef[j][0].elem[l];
209    }
210    paramF=newparamF;
211
212    for(int l=0;l<nn;l++){
213       pos = (l >= nn/2) ? l+nn : l;
214       for(int j=0;j<DMAP;j++)
215          newparamF.coef[j][0].elem[pos] = paramF0.coef[j][0].elem[l];
216    }
217    paramF0=newparamF;
218
219    paramR=fft_B(paramF);
220    paramR0=fft_B(paramF0);
221
222    nn = newnn;
```

```
223 }
224
225 int kam_torus(matrix &paramR, matrix &paramF, myreal omega, myreal &
        error, int nn,
226     int &tail0, myreal &beta,
227     void (*map)(complex *, complex *, complex **))
228 {
229     complex *z, *fz, **Dfz;
230     myreal **twist0;
231     myreal **invT, tolqr, **xiN0;
232     myreal aux;
233     myreal tg;
234     matrix FparamR(DMAP,1,DTOR,&nn);
235     matrix ErrorR(DMAP,1,DTOR,&nn);
236     matrix ErrorF(DMAP,1,DTOR,&nn);
237     matrix DFKR (DMAP,DMAP,DTOR,&nn);
238     matrix OmegaR (DMAP,DMAP,DTOR,&nn);
239     matrix OmegaF (DMAP,DMAP,DTOR,&nn);
240     matrix KshiftF(DMAP,1,DTOR,&nn);
241     matrix KshiftR(DMAP,1,DTOR,&nn);
242     matrix DparamF(DMAP,DTOR,DTOR,&nn);
243     matrix DparamR(DMAP,DTOR,DTOR,&nn);
244     matrix LR(DMAP,DTOR,DTOR,&nn);
245     matrix LF(DMAP,DTOR,DTOR,&nn);
246     matrix LshiftR(DMAP,DTOR,DTOR,&nn);
247     matrix LshiftF(DMAP,DTOR,DTOR,&nn);
248     matrix GR(DTOR,DTOR,DTOR,&nn);
249     matrix BR(DTOR,DTOR,DTOR,&nn);
250     matrix NR(DMAP,DTOR,DTOR,&nn);
251     matrix NF(DMAP,DTOR,DTOR,&nn);
252     matrix NshiftR(DMAP,DTOR,DTOR,&nn);
253     matrix NshiftF(DMAP,DTOR,DTOR,&nn);
254     matrix etaLR(DTOR,1,DTOR,&nn);
255     matrix etaNR(DTOR,1,DTOR,&nn);
256     matrix etaLF(DTOR,1,DTOR,&nn);
257     matrix etaNF(DTOR,1,DTOR,&nn);
258     matrix RetaNR(DTOR,1,DTOR,&nn);
259     matrix RetaNF(DTOR,1,DTOR,&nn);
260     matrix xiNR(DTOR,1,DTOR,&nn);
261     matrix xiLR(DTOR,1,DTOR,&nn);
262     matrix xiLF(DTOR,1,DTOR,&nn);
263     matrix twistR(DTOR,DTOR,DTOR,&nn);
264     matrix newparamR(DMAP,1,DTOR,&nn);
265     matrix newparamF(DMAP,1,DTOR,&nn);
266     complex **Omega;
267     myreal A[DMAP][DMAP], B[DMAP], **etaN0, **etaL0, **CL0, **CN0, **
        TxiN10, **TxiN20, beta0, newbeta;
268     matrix CLR(DTOR,1,DTOR,&nn);
```

```
269    matrix CNR(DTOR,1,DTOR,&nn);
270    matrix CLF(DTOR,1,DTOR,&nn);
271    matrix CNF(DTOR,1,DTOR,&nn);
272    matrix RCNF(DTOR,1,DTOR,&nn);
273    matrix RCNR(DTOR,1,DTOR,&nn);
274    matrix auxR(DMAP,1,DTOR,&nn);
275    matrix auxF(DMAP,1,DTOR,&nn);
276
277    z=new complex [DMAP];
278    fz=new complex [DMAP];
279    Dfz = new complex*[DMAP];
280    Omega = new complex*[DMAP];
281    for(int i=0;i<DMAP;i++){
282       Dfz[i]=new complex[DMAP];
283       Omega[i]=new complex[DMAP];
284    }
285    twist0 = new myreal *[DTOR];
286    etaN0 = new myreal *[DTOR];
287    etaL0 = new myreal *[DTOR];
288    CL0 = new myreal *[DTOR];
289    CN0 = new myreal *[DTOR];
290    TxiN10 = new myreal *[DTOR];
291    TxiN20 = new myreal *[DTOR];
292    xiN0 = new myreal *[DTOR];
293    for (int i=0;i<DTOR;i++){
294       twist0[i] = new myreal [DTOR];
295       etaN0[i] = new myreal [DTOR];
296       etaL0[i] = new myreal [DTOR];
297       CL0[i] = new myreal [DTOR];
298       CN0[i] = new myreal [DTOR];
299       TxiN10[i] = new myreal [DTOR];
300       TxiN20[i] = new myreal [DTOR];
301       xiN0[i]= new myreal [1];
302    }
303    invT=new myreal*[DTOR];
304    for (int i=0;i<DTOR;i++){
305       invT[i]=new myreal[DTOR];
306    }
307
308    Omega[0][0] = val0;
309    Omega[0][1] = -val1;
310    Omega[1][0] = val1;
311    Omega[1][1] = val0;
312
313    /*****************************************************************
314    STEP 0 Evaluation of the tail
315    *****************************************************************/
316    tail(paramF,&tg);
```

```
317
318   tail0=0;
319   cout.precision(3);
320   cout << "#    - Size of the grid: ";
321   cout << nn << " ";
322   cout << endl;
323   cout << "#    - Tail of the parameterization: ";
324   cout << tg << " ";
325   if (tg>toltail) {tail0=1;}
326
327   cout << endl;
328   if (tail0==1) return 0;
329   clean(paramF);
330   paramR=fft_B(paramF);
331
332   /*****************************************************************
333   STEP 1 Evaluation of the invariance error
334   *****************************************************************/
335   for (int l=0;l<nn;l++){
336
337      z[0]= paramR.coef[0][0].elem[l];
338      z[1]= paramR.coef[1][0].elem[l];
339
340      (*map)(z,fz,Dfz);
341      for(int i=0;i<DMAP;i++){
342         FparamR.coef[i][0].elem[l]=fz[i];
343         for(int j=0;j<DMAP;j++){
344            DFKR.coef[i][j].elem[l]=Dfz[i][j];
345            OmegaR.coef[i][j].elem[l]=Omega[i][j];
346         }
347      }
348   }
349
350   KshiftF=shift(paramF,&omega);
351   KshiftR=fft_B(KshiftF);
352
353   ErrorR = FparamR - KshiftR * (1+beta);
354
355   ErrorF = fft_F(ErrorR);
356   error = norm(ErrorF);
357   cout << "#    - Error of invariance: ";
358   cout << error << endl;
359
360   if (error<tolinva){
361      cout << "#    - No correction is needed!" << endl;
362      for(int i=0;i<DMAP;i++){
363         delete [] Dfz[i];
364      }
```

```
365       delete [] z;
366       delete [] fz;
367       delete [] Dfz;
368       for (int i=0;i<DTOR;i++){
369          delete [] twist0[i];
370          delete [] etaN0[i];
371          delete [] etaL0[i];
372          delete [] CL0[i];
373          delete [] CN0[i];
374          delete [] TxiN10[i];
375          delete [] TxiN20[i];
376          delete [] invT[i];
377          delete [] xiN0[i];
378       }
379       delete [] twist0;
380       delete [] etaN0;
381       delete [] etaL0;
382       delete [] CL0;
383       delete [] CN0;
384       delete [] TxiN10;
385       delete [] TxiN20;
386       delete [] invT;
387       delete [] xiN0;
388
389       return 1;
390    }
391
392    /*****************************************************************
393    STEP 2 Construction of the symplectic frame
394    *****************************************************************/
395    DparamF = diff(paramF);
396    DparamR = fft_B(DparamF);
397
398    LR=DparamR;
399
400    GR = trans(LR)*LR;
401    BR = inv(GR);
402    NR = OmegaR*LR*BR;
403
404    LF = fft_F(LR);
405    NF = fft_F(NR);
406
407    /*****************************************************************
408    STEP 3 Computation of the correction on the symplectic frame
409    *****************************************************************/
410    beta0 = beta;
411    LshiftF=shift(LF,&omega);
412    LshiftR=fft_B(LshiftF);
```

```
413   NshiftF=shift(NF,&omega);
414   NshiftR=fft_B(NshiftF);
415   OmegaF=fft_F(OmegaR);
416   etaLR  = -trans(NshiftR)*OmegaR*ErrorR;
417   etaNR  =  trans(LshiftR)*OmegaR*ErrorR;
418   twistR = trans(NshiftR)*OmegaR*DFKR*NR;
419   CLR = trans(NshiftR)*OmegaR*KshiftR;
420   CNR = -trans(LshiftR)*OmegaR*KshiftR;
421
422   aver(etaNR, etaN0);
423   aver(CNR, CN0);
424   for(int l=0;l<nn;l++){
425      etaNR.coef[0][0].elem[l] = etaNR.coef[0][0].elem[l] - etaN0
       [0][0];
426      CNR.coef[0][0].elem[l] = CNR.coef[0][0].elem[l] - CN0[0][0];
427   }
428   CNF = fft_F(CNR);
429   etaNF  = fft_F(etaNR);
430
431   RetaNF = cohomologicalBeta(etaNF,&omega, 1+beta0);
432   RCNF = cohomologicalBeta(CNF,&omega, 1+beta0);
433   RetaNR = fft_B(RetaNF);
434   RCNR = fft_B(RCNF);
435
436   // Construct and solve the linear system
437   aver(twistR,twist0);
438   invT[0][0]=1./twist0[0][0];
439   global_twist=invT[0][0];
440   cout << "#    - Norm inverse twist: ";
441   cout << global_twist << endl;
442
443   aver(etaLR, etaL0);
444   aver(CLR, CL0);
445   aver(twistR * RetaNR, TxiN10);
446   aver(twistR * RCNR, TxiN20);
447
448   A[0][0] = 1./(1+beta0) - (1 + beta0);
449   A[0][1] = - CN0[0][0];
450   A[1][0] = twist0[0][0];
451   A[1][1] = - CL0[0][0] + TxiN20[0][0];
452
453   B[0] = etaN0[0][0];
454   B[1] = etaL0[0][0] - TxiN10[0][0];
455
456   linear(A, B, xiN0[0][0], beta);
457
458   // Compute xiNR
459   xiNR = RetaNR + RCNR * beta;
```

```
460    for(int l=0;l<nn;l++){
461       xiNR.coef[0][0].elem[l] = xiNR.coef[0][0].elem[l] + xiN0[0][0];
462    }
463
464    // Compute xiLR
465    auxR = etaLR - twistR * xiNR + CLR * beta;
466    auxF = fft_F(auxR);
467    xiLF = cohomological(auxF, &omega);
468    xiLR = fft_B(xiLF) * (1./(1+beta0));
469
470    /*****************************************************************
471    STEP 4 New parameterization
472    *****************************************************************/
473    newparamR = paramR + LR*xiLR + NR*xiNR;
474    newparamF = fft_F(newparamR);
475    newbeta = beta0 + beta;
476
477    for(int i=0;i<DMAP;i++){
478       delete [] Dfz[i];
479       delete [] Omega[i];
480    }
481    for (int i=0;i<DTOR;i++){
482       delete [] twist0[i];
483       delete [] etaN0[i];
484       delete [] etaL0[i];
485       delete [] CL0[i];
486       delete [] CN0[i];
487       delete [] TxiN10[i];
488       delete [] TxiN20[i];
489       delete [] invT[i];
490       delete [] xiN0[i];
491    }
492    delete [] z;
493    delete [] fz;
494    delete [] Dfz;
495    delete [] Omega;
496    delete [] twist0;
497    delete [] etaN0;
498    delete [] etaL0;
499    delete [] CL0;
500    delete [] CN0;
501    delete [] TxiN10;
502    delete [] TxiN20;
503    delete [] invT;
504    delete [] xiN0;
505
506    paramR = newparamR;
507    paramF = newparamF;
```

```
508     beta = newbeta;
509
510     /* This is just to show the size of the correction */
511     newparamR = LR*xiLR + NR*xiNR;
512     newparamF = fft_F(newparamR);
513     aux = norm(newparamF);
514     cout << "#    - Norm of the correction: ";
515     cout << aux << endl;
516     cout << "#    - Beta: ";
517     cout << beta << endl;
518
519     return 0;
520 }
521
522 void map_standard(complex *z, complex *fz, complex **Dfz)
523 {
524     fz[1] = z[1] - epsilon*sin(pi2*z[0])/pi2;
525     fz[0] = z[0] + fz[1];
526
527     Dfz[1][0] = - epsilon*cos(pi2*z[0]);
528     Dfz[1][1] = val1;
529
530     Dfz[0][0] = val1 + Dfz[1][0];
531     Dfz[0][1] = Dfz[1][1];
532 }
533
534 void map_henon(complex *z, complex *fz, complex **Dfz)
535 {
536     fz[0] = z[0]*cos(epsilon)-(z[1]-z[0]*z[0])*sin(epsilon);
537     fz[1] = z[0]*sin(epsilon)+(z[1]-z[0]*z[0])*cos(epsilon);
538
539     Dfz[1][0] = sin(epsilon) - 2*z[0]*cos(epsilon);
540     Dfz[1][1] = cos(epsilon);
541
542     Dfz[0][0] = cos(epsilon) + 2*z[0]*sin(epsilon);
543     Dfz[0][1] = -sin(epsilon);
544 }
545
546 void linear (myreal A[2][2], myreal B[2], myreal &x, myreal &y) {
547     myreal det = A[0][0] * A[1][1] - A[0][1] * A[1][0];
548
549     if (fabs(det) < tolinte) {
550         std::cerr << "Error: Determinant is zero, the system cannot be
      solved." << std::endl;
551         return;
552     }
553
554     // Compute the solution using Cramer's rule
```

```
555     x = (B[0] * A[1][1] - B[1] * A[0][1]) / det;
556     y = (A[0][0] * B[1] - A[1][0] * B[0]) / det;
557 }
558 
559 matrix cohomologicalBeta(const matrix& m, double *omega, double a)
560 {
561     matrix coho(m.nrows,m.ncols);
562     if(a < tolgrid) cout << "Error en cohomologicalBeta: a = 0" << endl
        ;
563 
564     for(int i=0;i<m.nrows;i++){
565        for(int j=0;j<m.ncols;j++){
566           coho.coef[i][j]=cohomologicalBeta(m.coef[i][j],omega, a);
567        }
568     }
569     return coho;
570 }
571 
572 grid cohomologicalBeta(const grid& g, double *omega, double a)
573 {
574     int *index;
575     int *index_serie;
576     grid coho(g.ndim,g.nn);
577 
578     double pi2, aux, aux1;
579     pi2=6.28318530717958647692528676655900576 84;
580 
581     index=new int[g.ndim];
582     index_serie=new int[g.ndim];
583 
584     coho.elem[0]=0.0;
585     for(int i=1;i<g.nelem;i++){
586        indices(i,g.nn,index,g.ndim);
587        trigo_to_series(g.nn,index,index_serie,g.ndim);
588        aux=0.0;
589        for(int j=0;j<g.ndim;j++) aux = aux + index_serie[j]*omega[j];
590 
591        aux1=abs(g.elem[i]);
592        if (aux1<tolgrid) coho.elem[i]=complex(0.0,0.0);
593        else coho.elem[i]=g.elem[i]/((1./a)-a*complex(cos(pi2*aux),sin(
        pi2*aux)));
594     }
595 
596     delete [] index;
597     delete [] index_serie;
598 
599     return coho;
600 }
```

Bibliography

[1] À. Haro, M. Canadell, J. L. Figueras, A. Luque and J. M. Mondelo. (2016). *The parameterization method for invariant manifolds: From Rigorous Results to Effective Computations*. Springer.

[2] D. Blessing and J. D. Mireles James. (2024). Weighted Birkhoff averages and the parameterization method. *SIAM Journal on Applied Dynamical Systems*, 23(3), 1766-1804. https://doi.org/10.1137/23m1579546

[3] R. de la Llave. (2001). A tutorial on KAM theory. *Proceedings of Symposia in Pure Mathematics*, 175-292. https://doi.org/10.1090/pspum/069/1858536

[4] S. X. R. Guaitoli. (2024). *Superconvergence of weighted Birkhoff averages for quasiperiodic orbits*. [Bachelor's Thesis]. University of Barcelona.

[5] A. Katok and B. Hasselblatt. (1995). *Introduction to the modern theory of dynamical Systems*. Cambridge University Press.

[6] S. Das, Y. Saiki, E. Sander, and J.A. Yorke. (2017). Quantitative Quasiperiodicity. *Nonlinearity*, 30(11), 4111. https://doi.org/10.48550/arXiv.1601.06051

[7] R. de la Llave, A. González, À. Jorba, and J. Villanueva. (2005). KAM theory without action-angle variables. *Nonlinearity*, 18(2), 855-895. https://doi.org/10.1088/0951-7715/18/2/020

[8] D. Rim. (2017). An elementary proof that symplectic matrices have determinant one. *Advances in Dynamical Systems and Applications*, 12(1), 15-20. https://doi.org/10.37622/adsa/12.1.2017.15-20