Master of Science in Advanced Mathematics and Mathematical Engineering

MASTER'S THESIS

Title: Existence of Invariant Tori in Skew-Product Systems: A Computer Assisted Approach

Author: Eric Sandín Vidal

Advisor: Dr. Alex Haro Provinciale

Tutor: Dr. José Tomás Lázaro Ochoa

Department: Departament de Matemàtiques i Informàtica UB

Academic year: 2019-2020

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Master's Degree Thesis

Facultat de Matemàtiques i Estadística

EXISTENCE OF INVARIANT TORI IN SKEW-PRODUCT SYSTEMS: A COMPUTER-ASSISTED APPROACH

Author: Eric Sandín Vidal

Supervisor: Dr. Alex Haro Provinciale
Tutor: Dr. Jose Tomás Lázaro Ochoa

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

MAMME

January 8, 2020

Contents

Abstract

The field of dynamical systems is very broad, and one shall find all sorts of objects and structures contained within its secrets. This is the case of quasi-periodically forced maps, maps in which a quasi-periodic rotation has been applied over the torus of a skew-product dynamical system. Such systems can be studied from several points of view, such as the study of the dynamics of its subbundles, the reducibility into simpler dynamics, or, as it is our interest, the validation of invariant tori given approximately invariant tori. The main theoretical result of this work is a validation theorem that ensures the existence of an invariant torus should certain conditions be fulfilled. But given this matter has already been addressed by several authors, we will follow the next step and validate the torus computationally using computer assisted proofs. For that we will require the aid of validated numerics and the use of the necessary computational tools built for said purpose, such as the MPFI package. Those tools can be so as multi-precision numerics and intervalar arithmetics. However, the validation will also require some theoretical tools for dealing with mathematical objects. Fourier transforms and Fourier series will become the pillar over which the validation algorithm is sustained. For that, a much needed chapter of Fourier analysis results will be provided to make of this project a self-contained work.

Acknowledgements

It is such an honor to dive and learn from this fascinating topic and, of course, this work wouldn't have been possible without the inestimable help of a lot of people. First and foremost, I would like to thank my professors, Dr. José Tomás Lázaro for giving me the chance of writing a work in a more unconventional way, supporting me and revising all the work I sent to him, and Dr. Álex Haro for supervising my work and having the never ending patience to go through the theory and the coding process with me even at late hours in the night and making of this, even though at times painful, a great and fruitful experience. I would also like to thank my friend, pal, and colleague Sergio Gor for spending way too many hours helping me with software issues for the computer validation. And at last but not least, my family, for enduring the tough times that this work has brought and supporting me with all their love.

Introduction

The study of dynamical systems has proved to be of great use when it comes to solving a certain kind of problems. They allow us to predict natural and artificial phenomenon, study the most hidden properties of intricate systems or just provide helpful tools for the development of other areas of science or engineering. The case we present in this work is no different.

In the pages ahead, we will focus our attention towards a special kind of dynamical systems, the so called skew-product dynamical systems. Skew-products are systems based on a torus and its fiber, allowing us to define our map of interest over such structure, which we will refer as fiber bundle. Moreover, our interest will lie on quasi-periodically forced skew-product systems, that is, systems in which the dynamics on the torus is quasi-periodic. Amongst all the things we could study from such systems, we will deal with the problem of the existence of invariant torus under our map, providing a good result that ensures the existence of an invariant torus under our map given an approximately invariant one that fulfills certain conditions. This result is what we call the validation theorem, since it validates an invariant torus.
Besides proving the validation theorem, we will also implement it on a computer in the form of a computer assisted proof.

Computer assisted proofs have been on the rise in the last years, given that they are a very powerful tool for validating objects or simply proving theorems by computing great amounts of information. In our case, we implement a computer assisted proof that will calculate the error bounds and the constants that appear in our validation theorem and check the conditions that such result requires in order to yield a satisfactory response. To do so, we will also need other tools of a more theoretical nature, so we can quickly compute terms that could require more resources than we would like to. Such tools are Fourier series.

Fourier analysis is a very interesting field which basically deals with periodic functions. The famous Fourier Transform is a very powerful tool that allows the user to find the frequency values given only amplitude values of a signal. These Fourier coefficients make up Fourier series, which are trigonometric polynomials that are capable of rebuilding almost any function. These transforms will be of great use to us since they will allow the computer to perform quick calculations by simply moving our grid-evaluated objects onto Fourier space, where exponential operations can be easily done. With the necessary results to bound the error committed when moving from one space to another, we will be able to rigorously calculate the error bounds and the constants needed to apply our theorem and therefore validate a torus.

In addition to this theory and implementation, we will provide in the Appendix an out-of-scope brief introduction to whiskers, which are invariant manifolds attached to a torus. There we will settle the bases for a future proof of existence and an algorithm for a validation of such manifolds using similar tools to the ones used for the validation theorem.

Chapter 1

Quasi-Periodic Skew-Product Systems

In this very first chapter, we are going to introduce the basic notions of what quasi-periodic systems are, as well as some other useful properties and definitions that will be used further ahead in the project. Beyond that, we will give a motivation on the expansion of the real domain into a complex domain in order to be able to deal with real-analytic functions, such as real-analytic torus, as the pinnacle of regularity properties. But before diving directly in, we will need some general notions about the structures that will determine our working spaces, such as bundles, fiber bundles and other concepts in order to fully understand the particular case that a skew-product system is.

1.1 Introductory Definitions

1.1.1 Bundles

We present here very general definitions that can be found almost in any geometry or topology book. In our case, we will take [6] as reference.

Definition 1.1. *A bundle is a triple (E, π, B) where E is a set called the total space, B is a set called the base space of the bundle and $\pi : E \to B$ is the projection map. In addition, for each $b \in B$, $\pi^{-1}(b)$ is the fiber of the bundle over b and a bundle (E^*, π^*, B^*) is a subbundle of (E, π, B) if $B^* \subset B$, $E^* \subset E$ and $\pi^* = \pi|_{E^*}$.*

This definition of a bundle is very general but also very useful to construct the needed definition of our future working space. Even though it is now defined for general sets, we will take soon such sets as topological spaces. More restrictive conditions on regularity or set structure will be further ahead given.

Definition 1.2. *Let (E_1, π_1, B) and (E_2, π_2, B) be bundles and $f : B_1 \to B_2$ a map. Then a bundle map $F : E_1 \to E_2$ covering f is a continuous map such that $\pi_2 \circ F = f \circ \pi_1$, that is*

$$\begin{array}{ccc} E_1 & \xrightarrow{F} & E_2 \\ {\scriptstyle \pi_1}\downarrow & & \downarrow{\scriptstyle \pi_2} \\ B_1 & \xrightarrow{f} & B_2 \end{array}$$

Definition 1.3. *Let (E_1, π_1, B_1) and (E_2, π_2, B_2) be bundles and $F : E_1 \to E_2$ be a bundle map covering $f : B_1 \to B_2$. If $B_1 = B_2$ and $f = id$, then F is a bundle map over $B = B_1 = B_2$ such that $\pi_2 \circ F = \pi_1$. That is, the following diagram should commute*

$$\begin{tikzcd} E_1 \arrow[r, "F"] \arrow[d, "\pi_1"'] & E_2 \arrow[dl, "\pi_2"] \\ B \end{tikzcd}$$

Equivalently, for any point $x \in B$, F maps the fiber $E_{1_x} = \pi_1^{-1}(\{x\})$ of E_1 over x to the fiber $E_{2_x} = \pi_2^{-1}(\{x\})$ of E_2 over x.

Definition 1.4. *Let (E, π, B) be a bundle, then a section of that bundle is a continuous map $\sigma : B \to E$ such that $\pi(\sigma(x)) = x$ for all $x \in B$. That is, $\pi \circ \sigma = id$ which means that the following diagram commutes*

$$\begin{tikzcd} E \arrow[r, "\pi"] & B \\ B \arrow[u, "\sigma"] \arrow[ur, "id"'] \end{tikzcd}$$

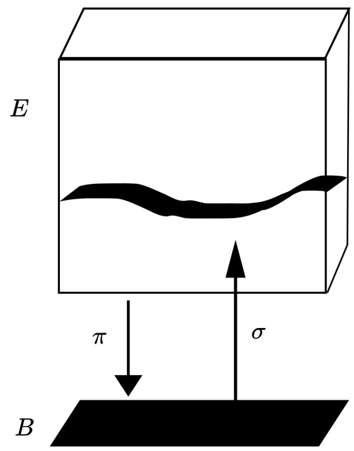


Figure 1.1: *A section σ of a bundle $\pi : E \to B$. A section σ allows the base space B to be identified with a subspace $\sigma(B)$ of E* [12].

Definition 1.5. *A fiber bundle is a structure (E, π, B, P), where E, B and P are topological spaces and $\pi : E \to B$ is a continuous surjection. The space B is connected and is called the base space of the bundle, E the total space, and P the fiber. The map π is called the projection map (or bundle projection). Such structure must satisfy the following condition.*

We require that for every $x \in E$, there is an open neighborhood $U \subset B$ of $\pi(x)$ (which will be called a trivializing neighborhood) such that there is a homeomorphism $\varphi : \pi^{-1}(U) \to U \times P$ (where $U \times P$ is the product space) in such a way that π agrees with the projection onto the *first factor. That is, the following diagram should commute*

$$\begin{tikzcd} \pi^{-1}(U) \arrow[r, "\varphi"] \arrow[d, "\pi"'] & U \times P \arrow[dl, "proj_1"] \\ U & \end{tikzcd}$$

where $proj_1 : U \times P \to U$ is the natural projection and $\varphi : \pi^{-1}(U) \to U \times P$ is a homeomorphism. The set of all $\{(U_i, \varphi_i)\}$ is called a local trivialization of the bundle.

Thus for any $y \in B$, the preimage $\pi^{-1}(\{y\})$ is homeomorphic to P (since $proj_1^{-1}(\{y\})$ clearly is) and is called the fiber over y. Every (fiber bundle) projection $\pi : E \to B$ is an open map (maps open subsets with open subsets), since projections of products are open maps. Therefore B carries the quotient topology determined by the map π.

For a better understanding of the fiber bundle concept, one shall see E locally like the product $B \times P$, except that the fibers $\pi(x)^{-1}$ for $x \in B$ may be a bit "twisted" [6].
Notice that a bundle is a generalization of a fiber bundle but with the sets lacking of a topology, which makes the condition of a local product structure drop.

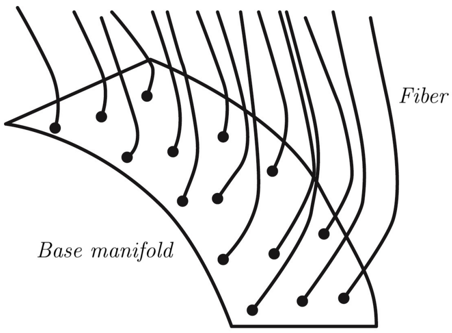


Figure 1.2: *A fiber bundle* [9].

Remark 1.6. Let $E = B \times P$ and let $\pi : E \to B$ be the projection onto the first factor. Then we will say that E is a fiber bundle (of P) over B. Here E is not just locally a product but globally one. Any such fiber bundle is called a trivial bundle [6].

We will see in the following section that the space with which we will work is a trivial bundle over a torus, thus the importance of properly building up the definition of fiber bundle and more specifically, of trivial bundle.

Definition 1.7. *A real vector bundle consists of a fiber bundle (E, π, B, P) with $P = \mathbb{R}^k$, where the compatibility condition is satisfied, that is, $\forall p \in B$, there is an open neighborhood $U \subseteq B$, and a homeomorphism $\varphi : U \times \mathbb{R}^k \to \pi^{-1}(U)$, such that $\forall x \in U$;*

1. $(\pi \circ \varphi)(x, v) = x \quad \forall v \in \mathbb{R}^k$.
2. *The map $v \mapsto \varphi(x, v)$ is a linear isomorphism between the vector spaces $\mathbb{R}^k$ and $\pi^{-1}(\{x\})$.*

Remark 1.8. The open neighborhood U together with the homeomorphism φ is called a local trivialization of the vector bundle. The local trivialization shows that, locally, the map π looks like the projection of $U \times \mathbb{R}^k$ on U [10].

Definition 1.9. *Let (E, π, B) be a bundle, given a bundle map $F : E \to E$ covering $f : B \to B$, an F-invariant section is a section that satisfies that $F \circ \sigma = \sigma \circ f$, which means the following diagram commutes*

$$\begin{tikzcd} E \arrow[r, "F"] & E \\ B \arrow[u, "\sigma"] \arrow[r, "f"'] & B \arrow[u, "\sigma"'] \end{tikzcd}$$

This last definition is very important since the main objects we will be working with are invariant sections of the map F, which will be presented in the following section.

1.1.2 Skew-Product Dynamical Systems

In this work we will deal with a particular type of fiber bundles, the aforementioned trivial bundles. Specifically, we will work with $\mathbb{R}^n \times \mathbb{T}^d$ as a trivial bundle over $\mathbb{T}^d$ with $\pi : \mathbb{R}^n \times \mathbb{T}^d \to \mathbb{T}^d$ as the corresponding bundle projection. We consider in $\mathbb{R}^n \times \mathbb{T}^d$ the product topology, so that the bundle projection is continuous. With our space defined, we can proceed to determine the norms we are going to use on them as well as the basic map over which the whole work will revolve around.

Definition 1.10. *Let $\mathbb{R}^n \times \mathbb{T}^d$ be a trivial fiber bundle with projection $\pi : \mathbb{R}^n \times \mathbb{T}^d \to \mathbb{T}^d$. A Finsler norm in the bundle is a continuous map*

$$\begin{aligned} |\cdot| : \mathbb{R}^n \times \mathbb{T}^d &\longrightarrow \mathbb{R}_+ \\ (x, \theta) &\longrightarrow |(x, \theta)| = |x|_\theta \end{aligned}$$

such that, for each $\theta \in \mathbb{T}^d$, $|\cdot|_\theta : \mathbb{R}^n \to \mathbb{R}_+$ is a norm.

In simpler terms, a Finsler norm in $\mathbb{R}^n \times \mathbb{T}^d$ is a norm $|\cdot|_\theta$ on each fiber $\mathbb{R}^n \times \{\theta\}$ that depends continuously on θ. Examples of Finsler norms are the constant Finsler norm $|\cdot|$, independent of θ, or given a norm $|\cdot|$ on $\mathbb{R}^n$, and a continuous matrix map $P : \mathbb{T}^d \to GL(\mathbb{R}^n)$, the Finsler norm $|x|_\theta = |P(\theta)x|$. We will usually omit the explicit dependence on θ of $|\cdot|_\theta$ when it is clear from the context.

Once we have the space and the norm, it is time to introduce the map that will define our dynamical system. These are called *Skew-product Dynamical Systems.*

Definition 1.11. (Skew-product Dynamical System) *Let $f : \mathbb{T}^d \to \mathbb{T}^d$ be a homeomorphism. A skew-product dynamical system in $\mathbb{R}^n$ over f is a bundle map*

$$\begin{aligned} \hat{F} = (F, f) : \mathbb{R}^n \times \mathbb{T}^d &\longrightarrow \mathbb{R}^n \times \mathbb{T}^d \\ (x, \theta) &\longrightarrow (F(x, \theta), f(\theta)) \end{aligned}$$

where for each fixed θ, $F(\cdot, \theta)$ *is a diffeomorphism of* $\mathbb{R}^n$.

From now on, we will refer to a continuous torus as a continuous section on the bundle $\mathbb{R}^n \times \mathbb{T}^d$, that is, a continuous map of the form $(K, id) : \mathbb{T}^d \to \mathbb{R}^n \times \mathbb{T}^d$, where $K : \mathbb{T}^d \to \mathbb{R}^n$ is continuous [4]. Analogously, we will refer to an analytic torus when the map $K : \mathbb{T}^d \to \mathbb{R}^n$ is real-analytic. The main goal of this work will be finding invariant tori, that is, invariant sections $\sigma = (K, id)$ such that $F \circ \sigma = \sigma \circ f$, which translates to $F(K(\theta), \theta) = K(f(\theta))$. From now on, we will omit the identity map of the section (K, id) when we refer to a torus, so we can directly say that a torus is a map $K : \mathbb{T}^d \to \mathbb{R}^n$.

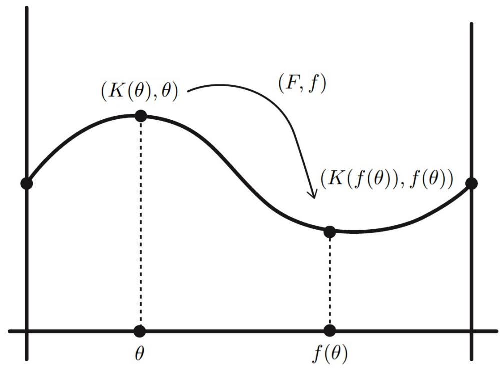


Figure 1.3: *A continuous torus.*

Moreover, we will work with a particular case of skew-products systems, quasi-periodic systems. Such systems are skew-product systems over rotations and will be properly defined in the following section. For now, in our particular case, we will denote our rotation $f(\theta) = \mathrm{R}_\omega(\theta) = \theta + \omega$, with $\omega \in \mathbb{R}^d$, turning the previous invariance equation into $F(K(\theta), \theta) = K(\theta + \omega)$.

1.2 Quasi-Periodic Maps and Invariant Tori

Clearly, before going deep into the study of our system, it is necessary that we define the proper spaces and their respective norms we will be dealing with when it comes to real-analytic functions.

1.2.1 Spaces of Analytic Functions

As said, our goal is to easily manipulate real-analytic functions, or, in our case, tori. For that, we will have to put in our toolbox some basic concepts on spaces of analytic functions, such as their domain or a tailored norm that allows us to properly measure their images.

Let $\mathbb{T}^d = (\mathbb{R}/\mathbb{T})^d$ be the real torus, and $\mathbb{T}^d_{\mathbb{C}} = \mathbb{T}^d + \mathbf{i}\mathbb{R}^d$ be the complex torus. We denote a *complex strip* (in $\mathbb{T}^d_{\mathbb{C}}$) of width $\rho > 0$ by

$$\mathbb{T}^d_\rho = \{\theta \in \mathbb{T}^d_{\mathbb{C}} \, : \, \mathrm{Im}\,|\theta_i| < \rho \, , \, i = 1, \dots, d\}.$$

Thus, we denote by $C^0(\bar{\mathbb{T}}^d_\rho, \mathbb{C}^m)$ the Banach space of continuous functions $f : \bar{\mathbb{T}}^d_\rho \to \mathbb{C}^m$ such that $f(\mathbb{T}^d) \subset \mathbb{R}^n$, endowed with the norm

$$\|f\|_\rho = \sup_{\theta \in \mathbb{T}^d_\rho} |f(\theta)|,$$

where $|\cdot|$ is the supremum norm in $\mathbb{C}^m$. We denote by $C^a(\bar{\mathbb{T}}^d_\rho, \mathbb{C}^m)$ the Banach space of continuous functions $f : \bar{\mathbb{T}}^d_\rho \to \mathbb{C}^m$, holomorphic on $\mathbb{T}^d_\rho$ and such that $f(\mathbb{T}^d) \subset \mathbb{R}^n$, that is, f is real-analytic (just analytic from now on), endowed with the supremum norm.

Consider the phase space an annulus $\mathcal{A}$ in $\mathbb{R}^n \times \mathbb{T}^d$, that is, an open set $\mathcal{A} \subset \mathbb{R}^n \times \mathbb{T}^d = \{z = (x, \theta) : x \in \mathbb{R}^n, \theta \in \mathbb{T}^d\}$ homotopic to $\mathcal{V} \times \mathbb{T}^d$, where $\mathcal{V} \subset \mathbb{R}^n$ is open. Let $\mathcal{B} \subset \mathbb{C}^n \times \mathbb{T}^d_\mathbb{C}$ be a complex neighborhood of the annulus $\mathcal{A}$.

We denote by $C^a(\bar{\mathbb{T}}^d_\mathcal{B}, \mathbb{C}^m)$ the Banach space of continuous functions $f : \bar{\mathcal{B}} \to \mathbb{C}^m$, holomorphic on $\mathcal{B}$ and such that $f(\mathcal{A}) \subset \mathbb{R}^m$ (so f is analytic), endowed with the norm [1]

$$\|f\|_\mathcal{B} = \sup_{z \in \mathcal{B}} |f(z)|.$$

1.2.2 Quasi-Periodic Maps

With the motivation of working with analytic functions, we can now extend the space of our skew-product system to the complex field, at the same time that we stretch our torus into a complex band of width ρ. Thus, we can finally provide the space in which this work will be focused, and that is $\mathbb{C}^n \times \mathbb{T}^d_\rho$.

Once we know this, we can turn our attention towards systems in which the dynamics on the torus is quasi-periodic, defined by the aforementioned irrational rotation $\mathrm{R}_\omega(\theta) = \theta + \omega$, with $\omega \in \mathbb{R}^d$. This means that we will be working with quasi-periodic skew-products $\hat{F} : \mathbb{C}^n \times \mathbb{T}^d_\rho \to \mathbb{C}^n \times \mathbb{T}^d_\rho$ of the form:

$$\begin{pmatrix} x \\ \theta \end{pmatrix} \xrightarrow{\hat{F}} \begin{pmatrix} F(x, \theta) \\ \theta + \omega \end{pmatrix}. \tag{1.1}$$

Also, we will study invariant tori by looking for parameterizations in which the motion is given by the rotation stated previously. That is, we seek those maps $K : \mathbb{T}^d_\rho \to \mathbb{C}^n$ in such a way that

$$F(K(\theta), \theta) = K(\theta + \omega). \tag{1.2}$$

If we consider the graph of K

$$\mathcal{K} = \{(K(\theta), \theta) \mid \theta \in \mathbb{T}^d_\rho\},$$

we observe that (1.2) is equivalent to saying that $\mathcal{K}$ is invariant under the skew-product (1.1). It will be convenient to think of (1.2) as an equation for the zeroes of the operator $\mathcal{F}$ defined by:

$$\mathcal{F}[K](\theta) = F(K(\theta-\omega), \theta-\omega) - K(\theta). \tag{1.3}$$

We note that if F is C^{r+l}, then, $\mathcal{F}$ is an l times differentiable operator from C^r to C^r. Hence the application of Newton method in function spaces is justified if F is differentiable enough. We also note that it is clear (and it can be justified under regularity assumptions on F) that the differential of the operator $\mathcal{F}$ in a torus K evaluated on a section $\xi : \bar{\mathbb{T}}^d_\rho \to \mathbb{C}^n$ of the bundle $\mathbb{C}^n \times \mathbb{T}^d_\rho$ is given by

$$\mathrm{D}\mathcal{F}[K]\xi(\theta) = \mathrm{D}_x F(K(\theta-\omega), \theta-\omega)\xi(\theta-\omega) - \xi(\theta). \tag{1.4}$$

The first term of $\mathrm{D}\mathcal{F}$ is called the transfer operator and we will denote it by $\mathcal{M}_\omega$. Recall that given a torus by $K : \mathbb{T}^d_\rho \to \mathbb{C}^n$, the matrix $M(\theta) = \mathrm{D}_x F(K(\theta), \theta)$ defines a linear skew product (or cocycle) by

$$\begin{pmatrix} v \\ \theta \end{pmatrix} \xrightarrow{\hat{M}} \begin{pmatrix} M(\theta)v \\ \theta+\omega \end{pmatrix}, \tag{1.5}$$

where (in an abuse of notation) $M(\theta) : E_\theta \to E_{\theta+\omega}$ takes a v in the fiber at position θ and takes it to the fiber in position $\theta+\omega$.

Now we can explicit the norm of the operator $\mathcal{M}_\omega$ when acting on analytic sections

$$\begin{aligned} \|\mathcal{M}_\omega\|_\rho &= \sup_{\|v\|_\rho=1} \|\mathcal{M}_\omega v\|_\rho = \sup_{\|v\|_\rho=1} \sup_{\theta\in\bar{\mathbb{T}}^d_\rho} |\mathcal{M}(\theta)v(\theta)|_{\theta+\omega} \le \sup_{\|v\|_\rho=1} \sup_{\theta\in\bar{\mathbb{T}}^d_\rho} |M(\theta)|_\theta \cdot |v(\theta)|_\theta \\ &\le \sup_{\theta\in\bar{\mathbb{T}}^d_\rho} |M(\theta)|_\theta = \sup_{\theta\in\mathbb{T}^d_\rho} \sup_{v\in\mathbb{C}^n,\, |v|_\theta=1} |M(\theta)v|_{\theta+\omega} = \|M\|_\rho, \end{aligned}$$

where $|\cdot|_\theta$ is a given Finsler norm. The operator $\mathcal{M}_\omega$ is clearly related to the derivative of the operator $\mathcal{F}$ since $\mathrm{D}\mathcal{F}[K] = \mathcal{M}_\omega - Id$. When using a Newton method to find a zero for $\mathcal{F}$, it is quite important to know whether 0 is in the spectrum of $\mathrm{D}\mathcal{F}$ or equivalently, whether 1 is in the spectrum of $\mathcal{M}_\omega$ [5].

1.2.3 Reducibility and Hyperbolicity

Going back to linear dynamics, it is worth explaining the concept of reducibility, a concept that may come handy when dealing with cocycles to manipulate them in a simpler way.

If the torus is invariant, the cocycle represents the linearization of the dynamics around the torus. If we think of v as an infinitesimal perturbation of the initial condition, $M(\theta)v$ describes how the disturbance propagates [5].
We will seek matrix maps $P : \mathbb{T}^d \to M_{n\times n}$, and $\Lambda : \mathbb{T}^d \to M_{n\times n}$ such that

$$P(\theta+\omega)^{-1}M(\theta)P(\theta) - \Lambda(\theta) = 0, \tag{1.6}$$

where P is an adapted frame for the torus and Λ represents the linearized dynamics. The idea of this transformation on M is to express the linearized dynamics in a simple way, that is as a triangular, constant or block-diagonal matrix. An important case, and the one we are treating here, is the case where P parametrizes two complementary invariant subbundles E_1, E_2 with rank n_1, n_2 respectively and Λ is a block-diagonal matrix.
In this work we will assume that our objects are fiberwise hyperbolic, this means that the linear dynamics can be decomposed into stable and unstable subbundles. In such context we can say that the matrix P parametrizes a stable subbundle (now E_s, of rank n_s) in its first n_s columns and an unstable subbundle (now E_u, of rank n_u) in its last n_u columns. Therefore our matrix Λ will look like the following

$$\Lambda(\theta) = \begin{pmatrix} \Lambda^s(\theta) & 0 \\ 0 & \Lambda^u(\theta) \end{pmatrix}$$

where $\Lambda^s(\theta) \in M_{n_s \times n_s}$ represents the linearized stable dynamics, which are assumed to be uniformly contracting, and $\Lambda^u(\theta) \in M_{n_u \times n_u}$ represents the linearized unstable dynamics, which are assumed to be uniformly expanding. This means that $|\Lambda^s(\theta)| < 1$ and $|(\Lambda^u(\theta))^{-1}| < 1$ for a given Finsler norm $|\cdot|_\theta$.

Remark 1.12. When there exists a matrix Λ such that equation (1.6) is satisfied, we say that the system is hyperbolically reducible. Notice that not all bundles admit global frames, which is why we consider here the case where our bundles are trivial or easily trivializable. In that case it is safe to assume that there exist global frames.

Remark 1.13. If our $\Lambda^s(\theta)$ is constant, we say that the system is reducible. We will see an example of this in the chapters ahead when we discuss rank-1 whiskers, where we will take $n_s = 1$.

Chapter 2

The Validation Theorem

In this chapter we will look at the main result of the work, and that is the validation theorem. Such theorem ensures the existence of an analytic invariant torus under an analytic quasi-periodic skew-product system given an approximately invariant analytic torus. Moreover, the theorem also states that such torus will be hyperbolic and gives a bound for the distance between the approximately invariant fibers of the initial torus and the actually invariant fibers of the newly found invariant torus.
As usual, we will require some introductory results before proving the theorem.

2.1 Preparatory Results

As we will have to deal with operators in the space of analytic functions, it will be useful to have some properties on the manipulation of such operators, and a very powerful tool is Neumann series. In addition, we will also need to understand the concept of resolvent.

Definition 2.1. *Let X be a Banach space and let $T : X \to X$ be a bounded linear operator. Let Id be the identity operator on X. In this context, the resolvent set (or just resolvent) of the operator T over the space X is defined as*

$$\mathrm{Res}(T, X) = \{z \in \mathbb{C} \mid T - zId \textit{ is bijective } \},$$

moreover, the spectrum is the complement of the resolvent set:

$$\mathrm{Spec}(T, X) = \mathbb{C} \backslash \mathrm{Res}(T, X).$$

Theorem 2.2. (Banach Open Mapping Theorem) *If X and Y are Banach spaces and $T : X \to Y$ is a surjective continuous linear operator, then T is an open map. If moreover, $T : X \to Y$ is bijective, then $T^{-1} : Y \to X$.*

Remark 2.3. Banach's Open Mapping Theorem implies that the operator $(T - zId)^{-1}$ is also bounded if $z \in \mathrm{Res}(T, X)$.

With this, we can say that $z \in \mathrm{Res}(T,X) \iff \forall \eta \in X, \exists!\, \xi \in X$ such that $T\xi - z\xi = \eta$. In this context we will say that the operator T is hyperbolic if the unit circle is in the resolvent of T, that is $\mathrm{Spec}(T,X) \cap S_1 = \emptyset$, where $S_1 = \{z \in \mathbb{C} \mid |z| = 1\}$.
Let's proceed now with some Neumann series results.

Definition 2.4. *A Neumann series is a series of the form*

$$\sum_{k=0}^{\infty} T^k ,$$

where T is an operator and $T^k = T^{k-1} \circ T$ is the k times repeated application, with $T^0 = Id$, being Id the identity operator.

Proposition 2.5. *Let T be a bounded linear operator over X. If the Neumann series converges in the operator norm, then $Id - T$ is invertible and*

$$(Id - T)^{-1} = \sum_{k=0}^{\infty} T^k .$$

Proof. Working with partial sums we obtain

$$\begin{aligned}(Id - T) \lim_{n\to\infty} \sum_{k=0}^{n} T^k &= \lim_{n\to\infty} (Id - T) \sum_{k=0}^{n} T^k = \lim_{n\to\infty} \left(\sum_{k=0}^{n} T^k - \sum_{k=0}^{n} T^{k+1} \right) \\ &= \lim_{n\to\infty} (Id - T^{n+1}) = Id.\end{aligned}$$

where the result is given because of the series' convergence [13]. □

Lemma 2.6. *Let P, T be bounded linear operators over a space X such that $Id - PT = E$, where $\|E\| < \tau < 1$ for a small τ and any given norm $\|\cdot\|$. Then P is invertible.*

Proof. Manipulating the matrices as their associated operators we have

$$Id - PT = E \iff Id - E = PT.$$

Since $\|E\| < 1$, its Neumann series converges, and by Proposition 2.5 (which from now on will be called the Neumann series argument), we have that $(Id - E)$ is invertible, which means that PT is also invertible, resulting in $P^{-1} = T(Id - E)^{-1}$.

□

In addition, we can see that $\|P^{-1} - T\| \leq \|T\| \frac{\tau}{1-\tau}$:

$$\begin{aligned}\|P^{-1} - T\| &= \|T(Id - E)^{-1} - T\| \leq \|T\| \cdot \|(Id - E)^{-1} - Id\| \\ &= \|T\| \cdot \|(Id - E)^{-1}[Id - (Id - E)]\| \leq \|T\| \cdot \|(Id - E)^{-1}\| \cdot \|E\| \\ &\leq \|T\| \cdot \sum_{k=0}^{\infty} \|E\|^k \cdot \|E\| \leq \|T\| \frac{\tau}{1-\tau}.\end{aligned}$$

It will also be useful to give a couple of fixed point theorems to properly understand the path we will be taking in order to prove the validation theorem.

Definition 2.7. *Let (X, d) be a complete metric space. Then a map $T : X \to X$ is called a contraction mapping (or a map that satisfies the contraction principle) on X if there exists $L \in [0, 1)$ such that*

$$d(T(x), T(y)) \leq L\, d(x, y) \quad \forall x, y \in X\ .$$

Theorem 2.8. (Banach Fixed Point Theorem) *Let (X, d) be a complete metric space and $f : X \to X$ a contractive map with contraction factor $L \in [0, 1)$, then exists a unique $x_* \in X$ such that $f(x_*) = x_*$.*

Proof. Start by taking a $x_0 \in X$, and then defining the sequence $(x_n)_n$ as $x_n = f^n(x_0)$. Since our metric space is complete, it suffices to prove that our sequence is a Cauchy one. $\forall n$ and $\forall p \geq 0$

$$\begin{aligned}d(x_{n+p}, x_n) &\leq d(x_{n+p}, x_{n+p-1}) + \ldots + d(x_{n+1}, x_n) \\ &\leq (L^{n+p-1} + L^{n+p-2} + \ldots + L^n) d(x_1, x_0) \\ &\leq L^n(1 + L + \ldots + L^{p-1}) d(x_1, x_0) \leq \frac{L^n}{1-L} d(x_1, x_0).\end{aligned}$$

Where, in the third step, we have applied that

$$d(x_{m+1}, x_m) \leq L\, d(x_m, x_{m-1}) \leq \ldots \leq L^m d(x_1, x_0)$$

using the contractive property and a geometric sum in the last step. From the inequality we obtain $\lim_{n\to\infty} \sup_{p\geq 0} d(x_{n+p}, x_n) = 0$, since $\sup_{p\geq 0} d(x_{n+p}, x_n) \leq \frac{L^n}{1-L} d(x_1, x_0)$, hence it is a Cauchy sequence and therefore $(x_n)_n$ converges to a certain x_*. Thus $x_{n+1} = f(x_n) \xrightarrow[n\to\infty]{} x_* = f(x_*)$ and x_* is a fixed point of f.

The uniqueness is easily proved by assuming there are two different fixed points, x_*, y_*, and therefore

$$0 < d(x_*, y_*) = d(f(x_*), f(y_*)) \leq L\, d(x_*, y_*) \to d(x_*, y_*) \leq L\, d(x_*, y_*)$$

which is a contradiction since $L \in [0, 1)$.

□

Notice that, in addition, $d(x_*, x_0) \leq \frac{d(x_1,x_0)}{1-L}$.

Theorem 2.9. (Radial Fixed Point Theorem) *Let (X, d) be a complete metric space and let $x_0 \in X$. Let now $T : B_R(x_0) \subset X \to X$ be a map in the open set $B_R(x_0)$ such that $\forall\, r \in (0, R)$, $T_{\big|\bar{B}_r(x_0)}$ is Lipschitz, where $\bar{B}_r(x_0) = \{x \in X : d(x, x_0) \leq r\}$ and*

$$\begin{aligned} L : (0, R) &\longrightarrow \mathbb{R}_+ \\ r &\longmapsto L(r) = \sup_{\substack{x_1,x_2 \in \bar{B}_r(x_0) \\ x_1 \neq x_2}} \frac{d(T(x_2), T(x_1))}{d(x_2, x_1)}. \end{aligned}$$

Notice that L is an increasing function.
Assume that $d(T(x_0), x_0) \leq \varepsilon$, where $\varepsilon > 0$ is the error bound of the fixed point condition, and take $r \in (\varepsilon, R)$. Then if $\frac{\varepsilon}{r} + L(r) - 1 \leq 0$, there exists a unique $x_* \in \bar{B}_r(x_0)$ such that $T(x_*) = x_*$.

Proof. Since X is a Banach space, and therefore a complete space, Theorem 2.8 allows us to reduce the proof to the following two steps:

1. $T(\bar{B}_r(x_0)) \subseteq \bar{B}_r(x_0)$, so the image of the ball won't escape the ball itself.

2. $T_{\big|\bar{B}_r(x_0)}$ is contractive.

For the first step we pick $x \in \bar{B}_r(x_0)$ and we see

$$\begin{aligned} d(T(x), x_0) &\leq d(T(x), T(x_0)) + d(T(x_0), x_0) \leq L(r)d(x, x_0) + \varepsilon \\ &\leq L(r)r + \varepsilon = r\left(L(r) + \frac{\varepsilon}{r}\right) \leq r \end{aligned}$$

which means that $T(x)$ is in $\bar{B}_r(x_0)$.

Since our function T is already Lipschitz, we only need to see if the Lipschitz constant $L(r)$ dwells in the $(0, 1)$ interval. By hypothesis, $\frac{\varepsilon}{r} + L(r) - 1 \leq 0$ which leads to $L(r) \leq 1 - \frac{\varepsilon}{r} < 1$.

□

Remark 2.10. Notice that the estimation that $x_* \in \bar{B}_r(x_0)$ cannot be further refined, given that using the formula obtained before and the inequality hypothesis of the theorem, $d(x_*, x_0) \leq \frac{d(x_1,x_0)}{1-L(r)} \leq \frac{\varepsilon}{\frac{\varepsilon}{r}} = r$. Keep in mind that the best estimation is taken for the smallest r that satisfies the conditions.

2.2 The Validation Theorem

Once we have all the needed definitions and results, it is time to state and prove the most important result of the work, the theorem that proves the existence of a hyperbolic invariant torus given an approximately invariant torus under quasi-periodic dynamics.

Theorem 2.11. *Let $\mathcal{U} \subset \mathbb{C}^n \times \mathbb{T}^d_{\mathbb{C}}$ be an open set and $F : \mathcal{U} \subset \mathbb{C}^n \times \mathbb{T}^d_{\mathbb{C}} \to \mathbb{C}^n$ be an analytic map (of class C^a) with respect to the x variables, defining a skew-product over the irrational rotation $\omega \in \mathbb{R}^d$. Assume that given a $\rho > 0$ we have an analytic torus $K_0 : \bar{\mathbb{T}}^d_\rho \to \mathbb{C}^n$ (that is, continuous in $\bar{\mathbb{T}}^d_\rho$ and analytic in $\mathbb{T}^d_\rho$) satisfying $\mathcal{K}_0 = graph(K_0) = \{(K_0(\theta), \theta) \mid \theta \in \mathbb{T}^d_\rho\} \subset \mathcal{U}$ and also that there exist:*

1) Two analytic matrix-valued maps $P_1, P_2 : \bar{\mathbb{T}}^d_\rho \to M_n(\mathbb{C})$, where P_1 represents a vector bundle map (over the identity) giving the change of variables to an adapted frame, and P_2 is its approximate inverse (see condition 5.3));

2) An analytic block-diagonal matrix-valued map

$$\Lambda_0(\theta) = \begin{pmatrix} \Lambda_0^s(\theta) & 0 \\ 0 & \Lambda_0^u(\theta) \end{pmatrix}$$

where $\Lambda_0^s : \bar{\mathbb{T}}^d_\rho \to M_{n_s}(\mathbb{C})$ and $\Lambda_0^u : \bar{\mathbb{T}}^d_\rho \to M_{n_u}(\mathbb{C})$, with $n = n_s + n_u$;

3) An (adapted) Finsler metric $|\cdot|_\theta$, of the form $|\hat{v}|_\theta = |\hat{v}^s|_\theta + |\hat{v}^u|_\theta$ for $\hat{v} = (\hat{v}^s, 0) + (0, \hat{v}^u) \in \mathbb{C}^n = \mathbb{C}^{n_s} \times \mathbb{C}^{n_u}$, and the induced norm on analytic sections and vector bundle maps is denoted by $\|\cdot\|_\rho$;

4) Positive constants $\varepsilon, \sigma, \tau, \lambda, R, r, b$ with $\lambda + \sigma + \tau < 1$;

such that

5.1) $E(\theta) = P_2(\theta)(F(K_0(\theta - \omega), \theta - \omega) - K_0(\theta))$ satisfies $\|E\|_\rho \leq \varepsilon$ (as a section);

5.2) $E_{red}(\theta) = P_2(\theta+\omega)\mathrm{D}_x F(K_0(\theta), \theta) P_1(\theta) - \Lambda_0(\theta)$ satisfies $\|E_{red}\|_\rho \leq \sigma$ (as a vector bundle map over the rotation ω);

5.3) $E_{inv}(\theta) = P_2(\theta)P_1(\theta) - Id$ satisfies $\|E_{inv}\|_\rho \leq \tau$ (as a vector bundle map over the identity);

5.4) $\Lambda_0^s(\theta)$ and $\Lambda_0^u(\theta)$ satisfy $\|\Lambda_0^s\|_\rho \leq \lambda$ and $\|(\Lambda_0^u)^{-1}\|_\rho \leq \lambda$ (as vector bundle maps over the rotation ω in $\mathbb{C}^{n_s} \times \mathbb{T}^d$ and $\mathbb{C}^{n_u} \times \mathbb{T}^d$, respectively);

5.5) For all points (x, θ) in the strip

$$\bar{D}_\rho(K_0, R) = \{(x, \theta) \in \mathbb{C}^n \times \bar{\mathbb{T}}^d_\rho \mid x = K_0(\theta) + P_1(\theta)\xi,\ \xi \in \mathbb{C}^n,\ |\xi|_\theta \leq R\} \subset \mathcal{U}\,,$$

the bilinear maps over the rotation ω

$$B(x,\theta) = P_2(\theta+\omega)\mathrm{D}_x^2 F(x,\theta)[P_1(\theta)\cdot, P_1(\theta)\cdot]$$

satisfy $\|B(x,\theta)\| \leq b$ *as a norm of a bilinear form.*

The norm of such a bilinear form can be defined as

$$\|B(x,\theta)\| = \sup_{(x,\theta)\in\bar{D}(K_0,\bar{r})} \sup_{|\xi_1|,|\xi_2|\neq 1} |P_2(\theta+\omega)\mathrm{D}_x^2 F(x,\theta)[P_1(\theta)\xi_1, P_1(\theta)\xi_2]|_{\theta+\omega} .$$

We now define the constants

$$\hat{\varepsilon} = \frac{\varepsilon}{1-(\lambda+\sigma+\tau)} , \; \beta = \frac{b}{1-(\lambda+\sigma+\tau)} , \; h = \beta\hat{\varepsilon} .$$

Assume

6) $h < \frac{1}{2}$;

7) $r_0 = \frac{1-\sqrt{1-2h}}{h} \cdot \hat{\varepsilon} \leq r \leq \min\{r_1, R\}$, *where* $r_1 = \frac{1+\sqrt{1-2h}}{h} \cdot \hat{\varepsilon}$.

Under the hypotheses 1-7:

a) $P_1(\theta)$ *is invertible and there exists an analytic invariant torus* $K_* : \mathbb{T}^d_\rho \to \mathbb{C}^n$ *to which the Newton method converges from the initial approximation* K_0 *and*

$$\|P_1(\theta)^{-1}(K_*(\theta) - K_0(\theta))\|_\rho \leq r_0 < 2\hat{\varepsilon} .$$

But also

$$\|P_1(\theta)^{-1}(K_*(\theta) - K_0(\theta))\|_\rho \leq \min\{r_1, R\} .$$

This means that K_* *is unique within a radius* r_1 *and that, more precisely, it is contained within a radius* r_0.

b) The torus K_* *is normally hyperbolic, that is, the transfer operator* $\mathcal{M}_\omega$ *is hyperbolic.*

Let $\hat{\lambda} = \|\Lambda_0\|_\rho$. *Define the constant*

$$\mu = \frac{\lambda}{1-\lambda^2}\frac{1}{1-\tau}\left(br_0 + \sigma + \hat{\lambda}\tau\right)$$

and suppose that, moreover, it suffices;

8) $\mu < \frac{1}{2+2\sqrt{2}}$.

Then:

c) The stable and unstable bundles differ from the initial approximate invariant bundles in a distance smaller than $\frac{2\mu}{(1-2\mu)+\sqrt{-4\mu^2-4\mu+1}}$, *and can be computed using the contraction principle.*

Remark 2.12. We can actually dispose of the analytic regularity of the torus, since the Bootstrap Theorem ensures the torus to be at least as regular as the map F.

Proof. The approach we are going to take is similar to the application of a Newton-Kantorovich method (a method that given enough regularity on the map and suitable bounds, ensures the quadratic convergence of a Newton method), but using also fixed point methods. This will suffice to prove the convergence of our method.
Lastly, and in order to provide more clarity to a proof of such length, it will be convenient to separate it in subsections dedicated to each of the statements that need to be proven.

Invariant Torus

First of all, using condition 5.3) and Lemma 2.6, we see that P_1 is invertible. Let us denote P_1^{-1} its inverse. By using P_1 as a change of coordinates on the bundle $\mathbb{C}^n \times \mathbb{T}^d_\rho$, we write $\hat{K} = P_1^{-1}K$, so the approximate invariant torus in the new coordinates is $\hat{K}_0 = P_1^{-1}K_0$. The functional on $C^a(\mathbb{T}^d_\rho, \mathbb{C}^n)$ we will consider is

$$\hat{\mathcal{F}}[\hat{K}](\theta) = P_2(\theta)(F(P_1(\theta-\omega)\hat{K}(\theta-\omega), \theta-\omega) - P_1(\theta)\hat{K}(\theta)).$$

From the regularity properties of the composition operator that can be found in [2], we can say that $\hat{\mathcal{F}}$ is (at least) C^2 when acting on analytic functions (which is enough regularity for our purpose as we will see further ahead).

Clearly, in order to find a solution to our problem, which is finding an invariant K, we will have to solve $\hat{\mathcal{F}}[\hat{K}] = 0$. For that, we can use a Newton method, which is defined as follows.

$$\hat{N}[\hat{K}] = \hat{K} - (\mathrm{D}\hat{\mathcal{F}}[\hat{K}])^{-1}\hat{\mathcal{F}}[\hat{K}]\,.$$

But in order to simplify, we will apply a quasi-Newton method, which does not update $(\mathrm{D}\hat{\mathcal{F}}[\hat{K}_i])^{-1}$ for every new found tori K_i, but instead fixes it to the first one we calculate, which is $(\mathrm{D}\hat{\mathcal{F}}[\hat{K}_0])^{-1}$. The downside of such method is that the quadratic speed of convergence (once convergence is proven) will drop, but that is not a problem for us since we only want to see convergence. Hence, our new iterative method is defined as

$$\hat{N}_0[\hat{K}] = \hat{K} - (\mathrm{D}\hat{\mathcal{F}}[\hat{K}_0])^{-1}\hat{\mathcal{F}}[\hat{K}]\,.$$

So, in order to prove that the method converges, we want to apply Theorem 2.9, a fixed point theorem.

Let us define the domain of the operator $\hat{N}_0[\hat{K}]$ as

$$\bar{B}_\rho(\hat{K}_0, r) = \{\hat{K} \in C^a(\mathbb{T}^d_\rho, \mathbb{C}^n) \mid \|\hat{K} - \hat{K}_0\|_\rho \le r\}\,,$$

which implies that if a torus $\hat{K}$ is in $\bar{B}_\rho(\hat{K}_0, r)$, then the torus $K(\theta) = K_0(\theta) + P_1(\theta)(\hat{K}(\theta) - \hat{K}_0(\theta))$ is also in the tube $\bar{D}_\rho(K_0, R)$.

Recalling Theorem 2.9, in order to ensure the existence of a fixed point in our operator, we need to find

$$\|\hat{N}_0[\hat{K}_0] - \hat{K}_0\|_\rho = \|\hat{K}_0 - (\mathrm{D}\hat{\mathcal{F}}[\hat{K}_0])^{-1}\hat{\mathcal{F}}[\hat{K}_0] - \hat{K}_0\|_\rho$$

and $L(r)$ for which

$$\|\hat{N}_0[\hat{K}_2] - \hat{N}_0[\hat{K}_1]\|_\rho \leq L(r)\, \|\hat{K}_2 - \hat{K}_1\|_\rho$$

such that

$$\frac{\|\hat{N}_0[\hat{K}_0] - \hat{K}_0\|_\rho}{r} + L(r) - 1 \leq 0\,.$$

Notice that $\|\hat{K}_0 - (\mathrm{D}\hat{\mathcal{F}}[\hat{K}_0])^{-1}\hat{\mathcal{F}}[\hat{K}_0] - \hat{K}_0\|_\rho \leq \|(\mathrm{D}\hat{\mathcal{F}}[\hat{K}_0])^{-1}\|_\rho \|\hat{\mathcal{F}}[\hat{K}_0]\|_\rho$. It is clear that we have to find bounds for $\|(\mathrm{D}\hat{\mathcal{F}}[\hat{K}_0])^{-1}\|_\rho$ and $\|\hat{\mathcal{F}}[\hat{K}_0]\|_\rho$, but recall that

$$\hat{\mathcal{F}}[\hat{K}](\theta) = P_2(\theta)(F(P_1(\theta-\omega)\hat{K}(\theta-\omega), \theta-\omega) - P_1(\theta)\hat{K}(\theta)),$$

which means that $\|\hat{\mathcal{F}}[\hat{K}_0]\|_\rho$ is the invariance error, which, by condition 5.1), satisfies $\|\hat{\mathcal{F}}[\hat{K}_0]\|_\rho = \|E\|_\rho \leq \varepsilon$. Let's compute now the bound for $\|(\mathrm{D}\hat{\mathcal{F}}[\hat{K}_0])^{-1}\|_\rho$. First, we have to calculate $\mathrm{D}\hat{\mathcal{F}}[\hat{K}_0]$. The differential of $\hat{\mathcal{F}}$ is defined by

$$\begin{aligned}(\mathrm{D}\hat{\mathcal{F}}[\hat{K}_0]\hat{\xi})(\theta) &= P_2(\theta)\mathrm{D}_x F(P_1(\theta-\omega)\hat{K}_0(\theta-\omega), \theta-\omega)P_1(\theta-\omega)\hat{\xi}(\theta-\omega) - P_2(\theta)P_1(\theta)\hat{\xi}(\theta)\\ &= (\Lambda_0(\theta-\omega) + E_{red}(\theta-\omega))\hat{\xi}(\theta-\omega) - \hat{\xi}(\theta) - E_{inv}(\theta)\hat{\xi}(\theta)\,,\end{aligned}$$

where $\hat{\xi} : \bar{\mathbb{T}}^d_\rho \to \mathbb{C}^n$ is analytic. Denoting $\mathcal{L}_\omega$, $\mathcal{E}_{red,\omega}$ the transfer operators associated to $\Lambda_0(\theta)$ and $E_{red}(\theta)$ respectively (over the rotation by ω), and $\mathcal{E}_{inv}$ the transfer operator associated to $E_{inv}(\theta)$ (over the identity), that is,

- $\mathcal{L}_\omega\, \hat{\xi}(\theta) = \Lambda_0(\theta-\omega)\, \hat{\xi}(\theta-\omega)$
- $\mathcal{E}_{red,\omega}\, \hat{\xi}(\theta) = E_{red}(\theta-\omega)\, \hat{\xi}(\theta-\omega)$
- $\mathcal{E}_{inv}\, \hat{\xi}(\theta) = E_{inv}(\theta)\, \hat{\xi}(\theta)$,

we can write

$$\mathrm{D}\hat{\mathcal{F}}[\hat{K}_0] = \mathcal{L}_\omega + \mathcal{E}_{red,\omega} - Id - \mathcal{E}_{inv} = (\mathcal{L}_\omega - Id) + \mathcal{E}_{red,\omega} - \mathcal{E}_{inv}$$

Notice that, from condition 5.4) and using a Neumann series argument, by decomposing $\mathcal{L}_\omega$ into its stable and unstable blocks (given that $\hat{K}_0$ is hyperbolic) and the fact that each operator $\mathcal{L}^{s,u}_\omega$ is bounded, we can say that $Id - \mathcal{L}^{s,u}_\omega$ are invertible. This means that $(Id - \mathcal{L}^{s,u}_\omega)^{-1} = -(\mathcal{L}^{s,u}_\omega - Id)^{-1}$. Separating the stable and unstable cases we can calculate for the stable bundle

$$(\mathcal{L}^s_\omega - Id)^{-1} = -(Id - \mathcal{L}^s_\omega)^{-1} = -\sum_{k=0}^{\infty}(\mathcal{L}^s_\omega)^k\,.$$

Taking norms and using, again, 5.4)

$$\|(\mathcal{L}^s_\omega - Id)^{-1}\|_\rho = \Big\| - \sum_{k=0}^{\infty} (\mathcal{L}^s_\omega)^k \Big\|_\rho \leq \sum_{k=0}^{\infty} \|(\mathcal{L}^s_\omega)^k\|_\rho \leq \sum_{k=0}^{\infty} \|\mathcal{L}^s_\omega\|^k_\rho \leq \sum_{k=0}^{\infty} \lambda^k = \frac{1}{1-\lambda} \,, \qquad (2.1)$$

given that $\lambda < 1$. The process for the unstable bundle is analogous, one only has to notice that

$$(\mathcal{L}^u_\omega - Id) = \mathcal{L}^u_\omega \cdot (Id - (\mathcal{L}^u_\omega)^{-1}) \longrightarrow (\mathcal{L}^u_\omega - Id)^{-1} = (Id - (\mathcal{L}^u_\omega)^{-1})^{-1} \cdot (\mathcal{L}^u_\omega)^{-1}$$

and therefore

$$\|(\mathcal{L}^u_\omega - Id)^{-1}\|_\rho \leq \|(Id - (\mathcal{L}^u_\omega)^{-1})^{-1}\|_\rho \cdot \|(\mathcal{L}^u_\omega)^{-1}\|_\rho \leq \sum_{k=0}^{\infty} \|(\mathcal{L}^u_\omega)^{-1}\|^k_\rho \cdot \lambda \leq \sum_{k=0}^{\infty} \lambda^k \cdot \lambda = \frac{\lambda}{1-\lambda} \,.$$

Obtaining thus

$$\|(\mathcal{L}^s_\omega - Id)^{-1}\|_\rho \leq \frac{1}{1-\lambda} \,, \quad \|(\mathcal{L}^u_\omega - Id)^{-1}\|_\rho \leq \frac{\lambda}{1-\lambda} \,.$$

Then, $\mathcal{L}_\omega - Id$ is also invertible and using the norm defined in 3) (taking the maximum norm between both blocks), we have

$$\|(\mathcal{L}_\omega - Id)^{-1}\|_\rho \leq \frac{1}{1-\lambda} \,.$$

Notice that for a $z \in \mathbb{C}^n$ such that $|z| = 1$, $\|(\mathcal{L}_\omega - Id)^{-1}\|_\rho = \|(\mathcal{L}_\omega - zId)^{-1}\|_\rho$.
By rewriting the expression of the differential of $\hat{\mathcal{F}}$ such as

$$\mathrm{D}\hat{\mathcal{F}}[\hat{K}_0] = (\mathcal{L}_\omega - Id) + \mathcal{E}_{red,\omega} - \mathcal{E}_{inv} = (\mathcal{L}_\omega - Id) \cdot (Id + (\mathcal{L}_\omega - Id)^{-1}(\mathcal{E}_{red,\omega} - \mathcal{E}_{inv})) \,,$$

we can find an expression for its inverse,

$$(\mathrm{D}\hat{\mathcal{F}}[\hat{K}_0])^{-1} = (Id + (\mathcal{L}_\omega - Id)^{-1}(\mathcal{E}_{red,\omega} - \mathcal{E}_{inv}))^{-1}(\mathcal{L}_\omega - Id)^{-1} \,.$$

From 5.2) and 5.3), and provided that $\|(\mathcal{L}_\omega - Id)^{-1}(\mathcal{E}_{red,\omega} - \mathcal{E}_{inv})\|_\rho \leq \frac{1}{1-\lambda}(\sigma + \tau) < 1$, using Neumann series we have the estimate

$$\|(\mathrm{D}\hat{\mathcal{F}}[\hat{K}_0])^{-1}\|_\rho \leq \frac{1}{1 - \frac{\sigma+\tau}{1-\lambda}} \frac{1}{1-\lambda} \,.$$

Notice that from condition 4) we have that $\lambda + \sigma + \tau < 1$, so finally

$$\|(\mathrm{D}\hat{\mathcal{F}}[\hat{K}_0])^{-1}\|_\rho \leq \frac{1}{1-(\lambda+\sigma+\tau)} \,.$$

With those estimates, we can lastly compute

$$\|\hat{K}_0 - (\mathrm{D}\hat{\mathcal{F}}[\hat{K}_0])^{-1}\hat{\mathcal{F}}[\hat{K}_0] - \hat{K}_0\|_\rho \leq \|(\mathrm{D}\hat{\mathcal{F}}[\hat{K}_0])^{-1}\|_\rho \|\hat{\mathcal{F}}[\hat{K}_0]\|_\rho \leq \frac{\varepsilon}{1-(\lambda+\sigma+\tau)} = \hat{\varepsilon}\,.$$

The next step is to find the $L(r)$ term. For that, we proceed as usual, by checking that

$$\|\hat{N}_0[\hat{K}_2] - \hat{N}_0[\hat{K}_1]\|_\rho \leq L(r)\, \|\hat{K}_2 - \hat{K}_1\|_\rho\,.$$

We can start by expressing $\hat{\mathcal{F}}[\hat{K}_2]$ in terms of its Taylor approximation plus the residue in its integral form around $\hat{K}_1$.

$$\hat{\mathcal{F}}[\hat{K}_2] = \hat{\mathcal{F}}[\hat{K}_1] + \int_0^1 \mathrm{D}\hat{\mathcal{F}}[\hat{K}_1 + t(\hat{K}_2 - \hat{K}_1)]\mathrm{d}t\, (\hat{K}_2 - \hat{K}_1)\,.$$

Then,

$$\begin{aligned}\hat{N}_0[\hat{K}_2] - \hat{N}_0[\hat{K}_1] &= \hat{K}_2 - (\mathrm{D}\hat{\mathcal{F}}[\hat{K}_0])^{-1}\hat{\mathcal{F}}[\hat{K}_2] - \hat{K}_1 + (\mathrm{D}\hat{\mathcal{F}}[\hat{K}_0])^{-1}\hat{\mathcal{F}}[\hat{K}_1] \\ &= \hat{K}_2 - (\mathrm{D}\hat{\mathcal{F}}[\hat{K}_0])^{-1}\left(\hat{\mathcal{F}}[\hat{K}_1] + \int_0^1 \mathrm{D}\hat{\mathcal{F}}[\hat{K}_1 + t(\hat{K}_2 - \hat{K}_1)]\mathrm{d}t\, (\hat{K}_2 - \hat{K}_1)\right) \\ &\quad - \hat{K}_1 + (\mathrm{D}\hat{\mathcal{F}}[\hat{K}_0])^{-1}\hat{\mathcal{F}}[\hat{K}_1]\,.\end{aligned}$$

Before continuing, let's find another expression for $\mathrm{D}\hat{\mathcal{F}}[\hat{K}_1 + t(\hat{K}_2 - \hat{K}_1)]$.

$$\begin{aligned}\mathrm{D}\hat{\mathcal{F}}[\hat{K}_1 + t(\hat{K}_2 - \hat{K}_1)] &= \mathrm{D}\hat{\mathcal{F}}[\hat{K}_0] + \int_0^1 \frac{\mathrm{d}}{\mathrm{d}s}\left(\mathrm{D}\hat{\mathcal{F}}\left[\hat{K}_0 + s(\hat{K}_1 + t(\hat{K}_2 - \hat{K}_1) - \hat{K}_0)\right]\right)\mathrm{d}s \\ &= \mathrm{D}\hat{\mathcal{F}}[\hat{K}_0] + \int_0^1 \mathrm{D}^2\hat{\mathcal{F}}\left[\hat{K}_0 + s\big((\hat{K}_1 - \hat{K}_0) + t(\hat{K}_2 - \hat{K}_1)\big)\right]\mathrm{d}s\, \cdot \\ &\quad \cdot \Big((\hat{K}_1 - \hat{K}_0) + t(\hat{K}_2 - \hat{K}_1)\Big)\,.\end{aligned}$$

Notice that we can transform

$$\begin{aligned}(\hat{K}_1 - \hat{K}_0) + t(\hat{K}_2 - \hat{K}_1) &= (\hat{K}_1 - \hat{K}_0) + t(\hat{K}_2 - \hat{K}_1) - t\hat{K}_0 + t\hat{K}_0 \\ &= (\hat{K}_1 - \hat{K}_0) + t(\hat{K}_2 - \hat{K}_0) - t(\hat{K}_1 - \hat{K}_0) \\ &= (1-t)(\hat{K}_1 - \hat{K}_0) + t(\hat{K}_2 - \hat{K}_0)\end{aligned}$$

and obtain

$$\begin{aligned}\hat{N}_0[\hat{K}_2] - \hat{N}_0[\hat{K}_1] = \int_0^1\int_0^1 &(\mathrm{D}\hat{\mathcal{F}}[\hat{K}_0])^{-1}\mathrm{D}^2\hat{\mathcal{F}}\left[\hat{K}_0 + s\left((\hat{K}_1 - \hat{K}_0) + t(\hat{K}_2 - \hat{K}_1)\right)\right]\mathrm{d}s \cdot \\ &\cdot \Big((1-t)(\hat{K}_1 - \hat{K}_0) + t(\hat{K}_2 - \hat{K}_0)\Big)\,\mathrm{d}t\,(\hat{K}_2 - \hat{K}_1)\,.\end{aligned}$$

We estimate now the norm of $\mathrm{D}^2\hat{\mathcal{F}}$ on functions in $\bar{B}_\rho(\hat{K}_0, r)$. Recall that, since $\|\hat{K}(\theta) - \hat{K}_0(\theta)\|_\rho \leq r \leq R$, then $(K(\theta), \theta) \in \bar{D}_\rho(K_0, R)$ for $\theta \in \bar{\mathbb{T}}^d_\rho$. With the second differential being

$$\mathrm{D}^2\hat{\mathcal{F}}[\hat{K}][\hat{\xi}_1, \hat{\xi}_2](\theta) = P_2(\theta+\omega)\mathrm{D}^2_x F(P_1(\theta)\hat{K}(\theta), \theta)[P_1(\theta)\hat{\xi}_1(\theta), P_1(\theta)\hat{\xi}_2(\theta)]$$

and applying condition 5.5) we obtain

$$\|\mathrm{D}^2\hat{\mathcal{F}}[\hat{K}]\| \leq b$$

for any $\hat{K} \in \bar{B}_\rho(\hat{K}_0, r)$. This way, we can see that we can bound

$$\left\| (\mathrm{D}\hat{\mathcal{F}}[\hat{K}_0])^{-1}\mathrm{D}^2\hat{\mathcal{F}}\left[\hat{K}_0 + s\left((\hat{K}_1 - \hat{K}_0) + t(\hat{K}_2 - \hat{K}_1)\right)\right] \right\|_\rho \leq \frac{b}{1-(\lambda+\sigma+\tau)} = \beta$$

and $\|\hat{K}_2 - \hat{K}_1\|_\rho, \|\hat{K}_1 - \hat{K}_1\|_\rho \leq r$. With this,

$$\|\hat{N}_0[\hat{K}_2] - \hat{N}_0[\hat{K}_1]\|_\rho \leq \frac{1}{2}\beta r \, \|\hat{K}_2 - \hat{K}_1\|_\rho \,,$$

so $L(r) = \frac{1}{2}\beta r$.

Once we have our estimates, we have to check the theorem's hypothesis, and that is $\frac{\hat{\varepsilon}}{r} + \frac{1}{2}\beta r - 1 \leq 0$, which is equivalent to $\hat{\varepsilon} + \frac{1}{2}\beta r^2 - r \leq 0$. By solving the inequation we find two values,

$$r_0 = \frac{1-\sqrt{1-2h}}{\beta} = \frac{1-\sqrt{1-2h}}{h}\,\hat{\varepsilon}\,, \quad r_1 = \frac{1+\sqrt{1-2h}}{\beta} = \frac{1+\sqrt{1-2h}}{h}\,\hat{\varepsilon}$$

for which r has to satisfy $r_0 \leq r \leq \min\{r_1, R\}$ (since we have to remain inside the tube). Such condition is satisfied due to hypotheses 6) and 7), which implies the satisfaction of the hypothesis of Theorem 2.9 and therefore the existence of a fixed point in our quasi-Newton method and hence the existence of an invariant torus K_*. Notice that Theorem 2.9 also tells us that our Newton operator is contractive since $L(r) < 1$.

Recall that the estimates we just found mean that the newly found torus K_* will be contained within a radius r_0 and that, furthermore, it will be unique in a radius r_1.

Hyperbolicity

In order to prove the hyperbolicity of K_*, we will prove that, for any $\hat{K}(\theta) \in \bar{B}_\rho(\hat{K}_0, r_0)$ (defined as before) the its transfer operator $\mathcal{M}_\omega$ is hyperbolic. As we saw before, we have to check that the unit circle is in the resolvent of the transfer operator, that is, $\forall z \in \mathbb{C}^n$ such that $|z| = 1$, we have to check that $z \in \mathrm{Res}(\mathcal{M}_\omega, C^a(\bar{\mathbb{T}}^d_\rho, \mathbb{C}^n))$. This means that $\forall \eta \in C^a(\bar{\mathbb{T}}^d_\rho, \mathbb{C}^n)$, $\exists!\, \xi \in C^a(\bar{\mathbb{T}}^d_\rho, \mathbb{C}^n)$ such that $\mathcal{M}_\omega\, \xi - z\, \xi = \eta$.

Hence, for any $z \in \mathbb{C}$ with $|z| = 1$, and given $\eta \in C^a(\bar{\mathbb{T}}^d_\rho, \mathbb{C}^n)$, we have to solve the equation

$$\eta(\theta) = \mathrm{D}_x F(K(\theta - \omega), \theta - \omega)\xi(\theta - \omega) - z\xi(\theta)$$

which is equivalent to

$$P_2(\theta)P_1(\theta)\hat{\eta}(\theta) = P_2(\theta)\mathrm{D}_x F(K(\theta - \omega), \theta - \omega)P_1(\theta - \omega)\hat{\xi}(\theta - \omega) - zP_2(\theta)P_1(\theta)\hat{\xi}(\theta) \quad (2.2)$$

once we perform the change of variables giving rise $\xi = P_1\hat{\xi}$ and $\eta = P_1\hat{\eta}$.

A first step is to compare the transfer operators associated to $M(\theta) = \mathrm{D}_x F(K(\theta), \theta)$ and $M_0(\theta) = \mathrm{D}_x F(K_0(\theta), \theta)$. To do so, we consider the vector bundle map (over the rotation ω) defined by

$$\begin{aligned} B(\theta) &= P_2(\theta)(\mathrm{D}_x F(K(\theta - \omega), \theta - \omega) - \mathrm{D}_x F(K_0(\theta - \omega), \theta - \omega))P_1(\theta - \omega) && (2.3) \\ &= \int_0^1 P_2(\theta)\mathrm{D}^2_x F(tK(\theta - \omega) + (1 - t)K_0(\theta - \omega), \theta - \omega) && (2.4) \\ &\quad [P_1(\theta - \omega)(\hat{K}(\theta - \omega) - \hat{K}_0(\theta - \omega)), P_1(\theta - \omega)\cdot]\, \mathrm{d}t\,. && (2.5) \end{aligned}$$

The transfer operator for such map would be $\mathcal{B}_\omega\hat{\xi}(\theta) = B(\theta - \omega)\hat{\xi}(\theta - \omega)$ and we see that $\|\mathcal{B}_\omega\|_\rho \le br_0$. Then, we can manipulate equation (2.2) and get

$$\begin{aligned} P_2(\theta)P_1(\theta)\hat{\eta}(\theta) &= P_2(\theta)\mathrm{D}_x F(K(\theta - \omega), \theta - \omega)P_1(\theta - \omega)\hat{\xi}(\theta - \omega) - zP_2(\theta)P_1(\theta)\hat{\xi}(\theta) \\ &= P_2(\theta)\mathrm{D}_x F(K(\theta - \omega), \theta - \omega)P_1(\theta - \omega)\hat{\xi}(\theta - \omega) \\ &- P_2(\theta)\mathrm{D}_x F(K_0(\theta - \omega), \theta - \omega)P_1(\theta - \omega)\hat{\xi}(\theta - \omega) \\ &+ P_2(\theta)\mathrm{D}_x F(K_0(\theta - \omega), \theta - \omega)P_1(\theta - \omega)\hat{\xi}(\theta - \omega) - zP_2(\theta)P_1(\theta)\hat{\xi}(\theta) \\ &= P_2(\theta)(\mathrm{D}_x F(K(\theta), \theta) - \mathrm{D}_x F(K_0(\theta), \theta))P_1(\theta - \omega)\hat{\xi}(\theta - \omega) \\ &+ P_2(\theta)\mathrm{D}_x F(K_0(\theta - \omega), \theta - \omega)P_1(\theta - \omega)\hat{\xi}(\theta - \omega) - zP_2(\theta)P_1(\theta)\hat{\xi}(\theta)\,, \quad (2.6) \end{aligned}$$

which turns into

$$(Id + E_{inv}(\theta))\hat{\eta}(\theta) = (\Lambda_0(\theta - \omega) + E_{red,\omega}(\theta - \omega) + B(\theta - \omega))\hat{\xi}(\theta - \omega) - z(Id + E_{inv}(\theta))\hat{\xi}(\theta)\,.$$

Using transfer operator notation, the previous equation becomes

$$(Id + \mathcal{E}_{inv})\hat{\eta} = ((\mathcal{L}_\omega - zId) + \mathcal{E}_{red,\omega} + \mathcal{B}_\omega - z\mathcal{E}_{inv})\hat{\xi}\,,$$

which can be expressed as a product as

$$(Id + \mathcal{E}_{inv})\hat{\eta} = (\mathcal{L}_\omega - zId)(Id + (\mathcal{L}_\omega - zId)^{-1}(\mathcal{B}_\omega + \mathcal{E}_{red,\omega} - z\mathcal{E}_{inv}))\hat{\xi}\,,$$

where the invertibility of $(\mathcal{L}_\omega - zId)$ has been proven before. Following similar procedures, the solution of the previous equation is:

$$\hat{\xi} = (Id + (\mathcal{L}_\omega - zId)^{-1}(\mathcal{B}_\omega + \mathcal{E}_{red,\omega} - z\mathcal{E}_{inv}))^{-1}(\mathcal{L}_\omega - zId)^{-1}(Id + \mathcal{E}_{inv})\hat{\eta}\ .$$

Even though the existence (and therefore the hyperbolicity) is already proven, we can go a bit further and provide a bound for $\|(\mathcal{M}_\omega - zId)^{-1}\|_\rho$. Since $\eta = (\mathcal{M}_\omega - zId)\xi$, then $\xi = (\mathcal{M}_\omega - zId)^{-1}\eta$, so $\|\xi\|_\rho = \|(\mathcal{M}_\omega - zId)^{-1}\|_\rho\|\eta\|_\rho$. Recall that $\eta = P_1\,\hat{\eta}$ and $\xi = P_1\,\hat{\xi}$, so with the last expression found for $\hat{\xi}$, we have

$$\xi = P_1(Id + (\mathcal{L}_\omega - zId)^{-1}(\mathcal{B}_\omega + \mathcal{E}_{red,\omega} - z\mathcal{E}_{inv}))^{-1}(\mathcal{L}_\omega - zId)^{-1}(Id + \mathcal{E}_{inv})P_1^{-1}\eta\ .$$

Since we can bound

$$\|(\mathcal{L}_\omega - zId)^{-1}(\mathcal{B}_\omega + \mathcal{E}_{red,\omega} - z\mathcal{E}_{inv})\|_\rho \leq \frac{br_0 + \sigma + \tau}{1-\lambda}\ ,$$

by using Neumann series, one has

$$\|\hat{\xi}\|_\rho \leq \frac{1}{1 - \frac{\tau + br_0 + \sigma}{1-\lambda}} \cdot \frac{1+\tau}{1-\lambda}\,\|\hat{\eta}\|_\rho = \frac{1+\tau}{1-(\lambda+\sigma+\tau+br_0)}\,\|\hat{\eta}\|_\rho\ .$$

Notice that $\lambda + \sigma + \tau + br_0$ is a "dirtier" hyperbolicity constant than λ, therefore, if $\lambda + \sigma + \tau + br_0 < 1$, we can ensure that the operator is a contraction. In order to prove such bound, we have to check that $br_0 < 1$ and that when added to $\lambda + \sigma + \tau$, the sum is still less than 1. For that we use hypothesis 6) and the fact that $r_0 < 2\hat{\varepsilon}$ to obtain:

$$\frac{br_0}{1-(\lambda+\sigma+\tau)} < 2\beta\hat{\varepsilon} = 2h < 1\ .$$

With that, we have

$$\|\xi\|_\rho \leq \|P_1\|_\rho\,\frac{1+\tau}{1-(\lambda+\sigma+\tau+br_0)}\,\|P_1^{-1}\|_\rho\,\|\eta\|_\rho\ ,$$

which implies

$$\|(\mathcal{M}_\omega - zId)^{-1}\|_\rho \leq \|P_1\|_\rho\,\frac{1+\tau}{1-(\lambda+\sigma+\tau+br_0)}\,\|P_1^{-1}\|_\rho\ .$$

Invariant Bundles

Once we have computed the invariant torus K_*, we are ready to compute its stable and unstable subbundles from the approximate invariant bundles.
The equation to be solved is

$$P(\theta+\omega)^{-1}\mathrm{D}_x F(K_*(\theta),\theta)P(\theta) - \Lambda(\theta) = 0\ . \tag{2.7}$$

The unknowns in the previous equation are P and $\Lambda = \text{blockdiag}(\Lambda^{\text{s}}, \Lambda^{\text{u}})$. Instead of using these unknowns, we will introduce new variables which take advantage of the fact that we have an approximate solution. Such variables will be Q, Δ^s and Δ^u, where

$$Q(\theta) = \begin{pmatrix} 0 & Q^{su}(\theta) \\ Q^{us}(\theta) & 0 \end{pmatrix}$$

with $Q^{us}(\theta) : \bar{\mathbb{T}}^d_\rho \to \mathbb{C}^{n_s \times n_u}$, $Q^{su}(\theta) : \bar{\mathbb{T}}^d_\rho \to \mathbb{C}^{n_u \times n_s}$ are analytic maps. In addition, we will have the Δ^s and Δ^u corrections, which will also be analytic maps $\Delta^s(\theta) : \bar{\mathbb{T}}^d_\rho \to \mathbb{C}^{n_s \times n_s}$, $\Delta^u(\theta) : \bar{\mathbb{T}}^d_\rho \to \mathbb{C}^{n_u \times n_u}$.

The key idea of this is that we are looking for correction variables (the entries of the matrix Q) such that when applied on the approximate fibers they will lead us to the invariant fibers. We can say that P_1 is the matrix of the base of eigenvectors on the approximate subbundles split into stable and unstable eigenvectors (to be precise, the eigenvectors that generate the stable and unstable subbundles). And the same goes for P and K_* (we can affirm that such splitting exists for K_* because we just proved it is normally hyperbolic). Dropping the θ dependence for a moment for the sake of the conceptual explanation and calling $P_1 = (v_1^s \,|\, v_1^u)$ and $P = (v^s \mid v^u)$, we can write

$$\begin{cases} v^s = v_1^s + Q^{us} v_1^u \\ v^u = v_1^u + Q^{su} v_1^s \end{cases}$$

which is equivalent to

$$(v^s \mid v^u) = (v_1^s \mid v_1^u) + (v_1^s \mid v_1^u) Q \, ,$$

or using matrices

$$P = P_1(Id + Q) \, .$$

Hence,

$$P(\theta) = P_1(\theta) + P_1(\theta)Q(\theta) \, , \quad \Lambda^s(\theta) = \Lambda_0^s(\theta) + \Delta^s(\theta) \, , \quad \Lambda^u(\theta) = \Lambda_0^u(\theta) + \Delta^u(\theta) \, . \tag{2.8}$$

We will use the contraction principle to analyze (2.7).
We take $Q^{us} = 0$, $Q^{su} = 0$, $\Delta^s = 0$ and $\Delta^u = 0$ as the first elements of the iteration. Then, by adding and subtracting the differential of K_0 (more specifically $\Lambda_0(\theta) + E_{red}(\theta)$), the error can be expressed as

$$\begin{aligned} \widetilde{E}_{red}(\theta) &= P_1(\theta + \omega)^{-1} \mathrm{D}_x F(K_*(\theta), \theta) P_1(\theta) - \Lambda_0(\theta) \\ &= (Id + E_{inv}(\theta + \omega))^{-1} (\Lambda_0(\theta) + E_{red}(\theta) + B(\theta + \omega)) - \Lambda_0(\theta) \\ &= ((Id + E_{inv}(\theta + \omega))^{-1} - Id) \Lambda_0(\theta) + (Id + E_{inv}(\theta + \omega))^{-1} (E_{red}(\theta) + B(\theta + \omega)) \, , \end{aligned} \tag{2.9}$$

where B is defined as in (2.3) but taking K_* as K.
Notice that in the second equality we have applied the same reasoning as in (2.6). Notice that

$$\begin{aligned}((Id + E_{inv}(\theta+\omega))^{-1} - Id) &= (Id + E_{inv}(\theta+\omega))^{-1}(Id - (Id + E_{inv}(\theta+\omega))) \\ &= (Id + E_{inv}(\theta+\omega))^{-1}(-E_{inv}(\theta+\omega)) \,.\end{aligned}$$

Therefore, taking norms directly from the last expression in (2.9) we have

$$\|\widetilde{E}_{red}\|_\rho \leq \frac{\tau}{1-\tau}\hat{\lambda} + \frac{1}{1-\tau}(\sigma + br_0) = \tilde{\sigma} \,, \tag{2.10}$$

where $\hat{\lambda} = \|\Lambda_0\|_\rho$ (as defined in the hypotheses). Now equation (2.7) reads

$$\begin{aligned}0 &= P_1(\theta+\omega)^{-1}\mathrm{D}_x F(K_*(\theta),\theta)P_1(\theta)(Id + Q(\theta)) - (Id + Q(\theta+\omega))(\Lambda_0(\theta) + \Delta(\theta)) \\ &= (\widetilde{E}_{red}(\theta) + \Lambda_0(\theta))(Id + Q(\theta)) - (Id + Q(\theta+\omega))(\Lambda_0(\theta) + \Delta(\theta)) \\ &= \Lambda_0(\theta)Q(\theta) - Q(\theta+\omega)\Lambda_0(\theta) - \Delta(\theta) + \widetilde{E}_{red}(\theta)(Id + Q(\theta)) - Q(\theta+\omega)\Delta(\theta) \,. \end{aligned} \tag{2.11}$$

We can write

$$\widetilde{E}_{red}(\theta) = \begin{pmatrix} \widetilde{E}^{ss}_{red}(\theta) & \widetilde{E}^{su}_{red}(\theta) \\ \widetilde{E}^{us}_{red}(\theta) & \widetilde{E}^{uu}_{red}(\theta) \end{pmatrix}$$

so we can deal with (2.11) as a product of matrices and express the result block by block. The diagonal blocks result in

$$\begin{aligned}-\Delta^s(\theta) + \widetilde{E}^{su}_{red}(\theta)Q^{us}(\theta) + \widetilde{E}^{ss}_{red}(\theta) = 0 \,, \\ -\Delta^u(\theta) + \widetilde{E}^{us}_{red}(\theta)Q^{su}(\theta) + \widetilde{E}^{uu}_{red}(\theta) = 0 \,,\end{aligned}$$

which can be expressed as

$$\begin{aligned}\Delta^s(\theta) = \widetilde{E}^{su}_{red}(\theta)Q^{us}(\theta) + \widetilde{E}^{ss}_{red}(\theta) \,, \\ \Delta^u(\theta) = \widetilde{E}^{us}_{red}(\theta)Q^{su}(\theta) + \widetilde{E}^{uu}_{red}(\theta) \,.\end{aligned}$$

Using this expressions, we can write the results of the remaining blocks as

$$\begin{aligned}\Lambda^s_0(\theta)Q^{su}(\theta) - Q^{su}(\theta+\omega)\Lambda^u_0(\theta) = -\left(\widetilde{E}^{ss}_{red}(\theta)Q^{su}(\theta) - Q^{su}(\theta+\omega)\widetilde{E}^{uu}_{red}(\theta)\right) + \\ Q^{su}(\theta+\omega)\widetilde{E}^{us}_{red}(\theta)Q^{su}(\theta) - \widetilde{E}^{su}_{red}(\theta) \,, \end{aligned} \tag{2.12}$$

$$\begin{aligned}\Lambda^u_0(\theta)Q^{us}(\theta) - Q^{us}(\theta+\omega)\Lambda^s_0(\theta) = -\left(\widetilde{E}^{uu}_{red}(\theta)Q^{us}(\theta) - Q^{us}(\theta+\omega)\widetilde{E}^{ss}_{red}(\theta)\right) + \\ Q^{us}(\theta+\omega)\widetilde{E}^{su}_{red}(\theta)Q^{us}(\theta) - \widetilde{E}^{us}_{red}(\theta) \,. \end{aligned} \tag{2.13}$$

Hence, we just have to solve (2.12) and (2.13). We will make explicit the calculations for (2.12), since the ones for (2.13) can be obtained by applying the results from (2.12) to the inverse mapping.
Multiplying on both sides of the equation by $(\Lambda^u_0)^{-1}$ and by defining the linear operator $\mathcal{L}^{su}_\omega$

acting on analytic vector bundle maps (over the identity) $Q^{su}(\theta) : \mathbb{R}^{n_u} \to \mathbb{R}^{n_s}$ as

$$\mathcal{L}^{su}_\omega Q^{su}(\theta) := \mathcal{L}^{su}_\omega [Q^{su}](\theta) = \Lambda^s_0(\theta-\omega) Q^{su}(\theta-\omega)(\Lambda^u_0(\theta-\omega))^{-1} \tag{2.14}$$

we can write equation (2.12) as

$$Q^{su} = (\mathcal{L}^{su}_\omega - Id)^{-1} \circ (-\widetilde{\mathcal{E}}^{ss}_{red,\omega} Q^{su} + Q^{su}_+(\widetilde{\mathcal{E}}^{us}_{red,\omega} Q^{su} + \widetilde{\mathcal{E}}^{uu}_{red,\omega}) - \widetilde{\mathcal{E}}^{su}_{red,\omega}) \circ (\mathcal{L}^u_\omega)^{-1} \,, \tag{2.15}$$

where $Q_+ := Q(\theta+\omega)$. By applying Neumann series reasoning on (2.14),

$$\|(\mathcal{L}^{su}_\omega - Id)^{-1}\|_\rho \le \frac{1}{1-\lambda^2} \,.$$

From now on, the right hand side of equation (2.15) will be considered as an operator $T(Q^{su}) := T[Q^{su}]$ acting on Q^{su}. So equation (2.15) reads as a fixed point equation $T(Q^{su}) = Q^{su}$, which leads us back to the application of the fixed point Theorem 2.9. We will assume that Q^{su} is contained in a ball of radius α. Again, the first step will be the estimation of $\|T(0) - 0\|_\rho$.

Remark 2.13. Keep in mind that when we specify that the map $Q^{su} = 0$, we are also saying that 0 is a matrix of the size of Q^{su}.

Observe that from estimate (2.10) we obtain estimates $\|\widetilde{\mathcal{E}}^{ss}_{red,\omega}\|_\rho \le \tilde{\sigma}$, $\|\widetilde{\mathcal{E}}^{su}_{red,\omega}\|_\rho \le \tilde{\sigma}$, $\|\widetilde{\mathcal{E}}^{us}_{red,\omega}\|_\rho \le \tilde{\sigma}$ and $\|\widetilde{\mathcal{E}}^{uu}_{red,\omega}\|_\rho \le \tilde{\sigma}$ because the norm of every block of the matrix cannot be bigger than the norm of the matrix itself given that we work with supremum norms.

We see then

$$\|T(0) - 0\|_\rho = \|T(0)\|_\rho \le \|(\mathcal{L}^{su}_\omega - Id)^{-1}\|_\rho \, \|\widetilde{\mathcal{E}}^{us}_{red,\omega}\|_\rho \, \|(\mathcal{L}^u_\omega)^{-1}\|_\rho \le \frac{\lambda}{1-\lambda^2}\tilde{\sigma} = \mu \,.$$

The next step is finding $L(\alpha)$ such that

$$\|T(Q^{su}_2) - T(Q^{su}_1)\|_\rho \le L(\alpha)\|Q^{su}_2 - Q^{su}_1\|_\rho \,.$$

For that, we proceed directly

$$\begin{aligned}
&\|T(Q^{su}_2) - T(Q^{su}_1)\|_\rho \le \\
&\qquad \le \|(\mathcal{L}^{su}_\omega - Id)^{-1} \circ (-\widetilde{\mathcal{E}}^{ss}_{red,\omega} Q^{su}_2 + Q^{su}_{+2}(\widetilde{\mathcal{E}}^{us}_{red,\omega} Q^{su}_2 + \widetilde{\mathcal{E}}^{uu}_{red,\omega}) - \widetilde{\mathcal{E}}^{su}_{red,\omega}) \circ (\mathcal{L}^u_\omega)^{-1} \\
&\qquad - (\mathcal{L}^{su}_\omega - Id)^{-1} \circ (-\widetilde{\mathcal{E}}^{ss}_{red,\omega} Q^{su}_1 + Q^{su}_{+1}(\widetilde{\mathcal{E}}^{us}_{red,\omega} Q^{su}_1 + \widetilde{\mathcal{E}}^{uu}_{red,\omega}) - \widetilde{\mathcal{E}}^{su}_{red,\omega}) \circ (\mathcal{L}^u_\omega)^{-1}\|_\rho \\
&\qquad \le \frac{\lambda}{1-\lambda^2}\| - \widetilde{\mathcal{E}}^{ss}_{red,\omega}(Q^{su}_2 - Q^{su}_1) + (Q^{su}_{+2} - Q^{su}_{+1})\widetilde{\mathcal{E}}^{uu}_{red,\omega} + Q^{su}_{+2}\widetilde{\mathcal{E}}^{us}_{red,\omega} Q^{su}_2 \\
&\qquad\quad - Q^{su}_{+2}\widetilde{\mathcal{E}}^{us}_{red,\omega} Q^{su}_1 - Q^{su}_{+1}\widetilde{\mathcal{E}}^{us}_{red,\omega} Q^{su}_1 + Q^{su}_{+2}\widetilde{\mathcal{E}}^{us}_{red,\omega} Q^{su}_1\|_\rho \\
&\qquad \le \frac{\lambda}{1-\lambda^2}(2\tilde{\sigma} + 2\alpha\tilde{\sigma})\|Q^{su}_2 - Q^{su}_1\|_\rho \le 2\mu(1+\alpha)\|Q^{su}_2 - Q^{su}_1\|_\rho \,,
\end{aligned}$$

where we have used that $\|Q^{su}_{1,2}\|_\rho \le \alpha$. So $L(\alpha) = 2\mu(1+\alpha)$ and we just have to check the fixed point theorem's condition:

$$\frac{\mu}{\alpha} + 2\mu(1+\alpha) - 1 \leq 0 \iff \mu + (2\mu - 1)\alpha + 2\mu\alpha^2 \leq 0$$

Solving for α we obtain

$$\alpha_{\pm} = \frac{(1-2\mu) \pm \sqrt{-4\mu^2 - 4\mu + 1}}{4\mu}\,,$$

for these solutions to exist we need a non-negative discriminant, so it is required

$$-4\mu^2 - 4\mu + 1 \geq 0\,,$$

and for that, μ needs to satisfy

$$\mu < \frac{\sqrt{2}-1}{2} = \frac{1}{2+2\sqrt{2}}\,.$$

By hypothesis 8), $\mu < \frac{1}{2+2\sqrt{2}}$, which means that α solutions exist for the inequation of the fixed point theorem, and therefore there exists a fixed point of the operator $T(Q^{su})$ and hence exists Q^{su}, which is the correction matrix for approximately invariant subbundles, implying the existence of actually invariant subbundles.

Recall as well from Remark 2.10 that the estimate given by Theorem 2.9 cannot be further improved, meaning that the fixed point is bounded by the radius of the ball within which it is contained. More specifically, it will be bounded by α_-.

$$\|Q^{su}\|_\rho \leq \alpha_- = \frac{(1-2\mu) - \sqrt{-4\mu^2 - 4\mu + 1}}{4\mu} = \frac{2\mu}{(1-2\mu) + \sqrt{-4\mu^2 - 4\mu + 1}}\,,$$

which is the distance between the approximate invariant subbundles and the invariant ones.

□

Chapter 3

Fourier Series

Fourier series are widely known for being an excellent tool for alternative function representation. The capability of expressing function values using Fourier coefficients can be of great use when it comes to computation. And that is exactly why we need them. In order to easily manipulate points on a grid as we will have to when dealing with operations over the torus, it will be convenient to use their Fourier coefficients so simple transformations can be applied over exponentials. Since a computer cannot work with continuous arrays, we will also introduce the discrete version of Fourier series and an algorithm for a faster execution of such calculations. The statements and results from this chapter have been adapted from [11] and [3].

3.1 The Fourier Transform and the Discrete Fourier Transform

For an analytic functions $u : \mathbb{T}_\rho \to \mathbb{C}$, we write its the Fourier expansion as

$$u(\theta) = \sum_{k\in\mathbb{Z}} \hat{u}_k \, e^{2\pi i k\theta} \,, \quad \hat{u}_k = \int_0^1 u(\theta)\, e^{-2\pi i k\theta} d\theta$$

and we note the average of u as $\langle u \rangle = \hat{u}_0 = \int_0^1 u(\theta) d\theta$. Notice that $\hat{u}_k^* = \hat{u}_{-k}$, where $\hat{u}_k^*$ denotes the complex conjugate of $\hat{u}_k$.
Then we consider the Fourier norm

$$\|u\|_{F,\,\rho} = \sum_{k\in\mathbb{Z}} |\hat{u}_k| \, e^{2\pi|k|\rho} \,.$$

We observe that $\|u\|_\rho \leq \|u\|_{F,\,\rho}\,, \ \forall \rho > 0$.

Now we are ready to introduce the Discrete Fourier Transform and its properties. We provide the definition of Fourier series given any function $f : \mathbb{T} \to \mathbb{C}$:

$$f(\theta) = \sum_{k\in\mathbb{Z}} \hat{f}_k \, e^{2\pi i k\theta}$$

where the Fourier coefficients are given by the Fourier Transform (FT)

$$\hat{f}_k = \int_0^1 f(\theta)\, e^{-2\pi ik\theta} d\theta. \tag{3.1}$$

We consider a sample of points on the regular grid of size $N \in \mathbb{N}$, $\theta_j := \frac{j}{N}$, where $0 \leq j < N$. This defines a sampling $\{f_j\}$, with $f_j = f(\theta_j)$ and a total number of points N.
The integrals in (3.1) are approximated using the trapezoidal rule on the regular grid, obtaining the Discrete Fourier Transform (DFT)

$$\tilde{f}_k = \frac{1}{N} \sum_{j=0}^{N-1} f_j\, e^{-2\pi ik\theta_j} \ .$$

Remark 3.1. $\tilde{f}_k$ can be defined for all $k \in \mathbb{Z}$. Moreover, they are periodic with period N, $\tilde{f}_{k+N} = \tilde{f}_k$.

The function f is approximated by the discrete Fourier approximation

$$\tilde{f}(\theta) = \sum_{k=-\left[\frac{N}{2}\right]}^{\left[\frac{N-1}{2}\right]} \tilde{f}_k\, e^{2\pi ik\theta} \ .$$

Along this section, we will use the standard notation $[x] = \max\{j \in \mathbb{Z} : j \leq x\}$ for the integer part of x.

Remark 3.2. The DFT approximation $\tilde{f}(\theta)$ interpolates the data on the grid. That is $\forall j = 0, \ldots, N-1$, $\tilde{f}(\theta_j) = f(\theta_j)$.

Notice that the stated process turns the sampling of points on the grid onto the Fourier coefficients for the DFT. The inverse process will get the Fourier coefficients for the DFT and turn them onto the values on the grid. This process is called the Inverse Discrete Fourier Transform (IDFT) and uses the following formula

$$f_j = \sum_{k=0}^{N-1} \tilde{f}_k\, e^{\frac{2\pi i}{N} jk} \ .$$

Remark 3.3. As we have previously stated, the Fourier coefficients are symmetrical, that is, $\hat{f}_k^* = \hat{f}_{-k}$, which holds for the DFT coefficients as well, $\tilde{f}_k^* = \tilde{f}_{-k}$. This presents a problem regarding the way we have defined the DFT. See that since we are treating the real analytic case, our function f evaluated over the points of the grid will acquire real values, but depending on the parity of the size of the grid, N, the discrete approximation will not. The reason behind this phenomenon lies on the fact that if N is odd, due to the coefficients' symmetry, the resulting function will remain real, but if N is even, then $N-1$ is odd, which means that the term $-\left[\frac{N}{2}\right]$ of the sum, called the Nyquist term, will be unpaired. The lack of its symmetrical pair results on a complex function whose derivative will have the imaginary term i. This does not present a major issue since the Nyquist term will naturally be very small. Nonetheless, if it is desired to look for a way to express the function f in terms of its DFT without this little problem, one shall eliminate the Nyquist term, thus obtaining

$$p(\theta) = \sum_{k=-\left[\frac{N-1}{2}\right]}^{\left[\frac{N-1}{2}\right]} \tilde{f}_k \, e^{2\pi i k\theta} .$$

Although this solves the previous issue, it presents another one, the main reason why we are not taking p in our process. Since we have set the Nyquist term to 0, this approximation will not interpolate the data on the grid, which is a property of great use to us. Thus we will keep using $\tilde{f}$.

3.2 Error Estimates on Approximations

3.2.1 Analytic Periodic Functions

As we have seen, there are discrete ways of expressing a function in terms of a trigonometric polynomial. The DFT supposes a great advantage for computing Fourier series with a machine. But of course, the loss of exact information when interpolating between grid points produces an approximation error. Coming up next we present the error between DFT coefficients and FT coefficients and the error when approximating a function with the DFT approximation.

Lemma 3.4. *Fixed the grid size $N \in \mathbb{N}$, the coefficients of the DFT are obtained from the coefficients of the FT by*

$$\tilde{f}_k = \sum_{m\in\mathbb{Z}} \hat{f}_{k+Nm} .$$

Proof. The proof for the Lemma starts by substituting f_j by its aforementioned Fourier series expression

$$\tilde{f}_k = \frac{1}{N}\sum_{j=0}^{N-1} f_j \, e^{-2\pi i k\theta_j} = \frac{1}{N}\sum_{j=0}^{N-1}\sum_{l\in\mathbb{Z}} \hat{f}_l \, e^{2\pi i l\theta_j} \, e^{-2\pi i k\theta_j} = \sum_{l\in\mathbb{Z}} \hat{f}_l \left(\frac{1}{N}\sum_{j=0}^{N-1} e^{2\pi i (l-k)\frac{j}{N}} \right) .$$

Notice that $\frac{1}{N}\sum\limits_{j=0}^{N-1} e^{2\pi i(l-k)\frac{j}{N}} = 1$ if $l-k$ is a multiple of N since $\frac{l-k}{N}, j \in \mathbb{Z}$ and then $e^{2\pi i(l-k)\frac{j}{N}} = 1$ and $\frac{1}{N}\sum\limits_{j=0}^{N-1} e^{2\pi i(l-k)\frac{j}{N}} = \frac{1}{N}\sum\limits_{j=0}^{N-1} 1 = 1.$
Let's see now the case where $l-k$ is not a multiple of N.

$$\frac{1}{N}\sum_{j=0}^{N-1} e^{2\pi i(l-k)\frac{j}{N}} = \frac{1}{N}\sum_{j=0}^{N-1} (e^{2\pi i\frac{(l-k)}{N}})^j = \frac{1}{N}\frac{1-(e^{2\pi i\frac{(l-k)}{N}})^N}{1-e^{2\pi i\frac{(l-k)}{N}}} = \frac{1}{N}\frac{1-e^{2\pi i(l-k)}}{1-e^{2\pi i\frac{(l-k)}{N}}} .$$

Since $l-k \in \mathbb{Z}$, $e^{2\pi i(l-k)} = 1$ and $1-e^{2\pi i(l-k)} = 0$. By hypothesis, $l-k$ is not a multiple of N, which means that $1-e^{2\pi i\frac{(l-k)}{N}} \neq 0$.

Wrapping up, we have

$$\frac{1}{N}\sum_{j=0}^{N-1} e^{2\pi i(l-k)\frac{j}{N}} = \begin{cases} 1 & \text{if } l-k \text{ is a multiple of } N \\ 0 & \text{otherwise} \end{cases} .$$

This means that the first sum will only have terms if $l-k = Nm$ for $m \in \mathbb{Z}$, that is for the terms $l = k+Nm$ and hence

$$\tilde{f}_k = \sum_{l\in\mathbb{Z}} \hat{f}_l \left(\frac{1}{N}\sum_{j=0}^{N-1} e^{2\pi i(l-k)\frac{j}{N}} \right) = \sum_{m\in\mathbb{Z}} \hat{f}_{k+Nm} \,.$$

□

Proposition 3.5. *Let $f : \mathbb{T}_{\hat{\rho}} \longrightarrow \mathbb{C}$ be a real analytic and bounded function in the complex strip $\mathbb{T}_{\hat{\rho}}$ of size $\hat{\rho} > 0$. Let $\tilde{f}$ be the discrete Fourier approximation of f in the regular grid of size $N \in \mathbb{N}$ with Fourier coefficients $\tilde{f}_k$. Then for $k = -\left[\frac{N}{2}\right], \cdots, \left[\frac{N-1}{2}\right]$,*

$$|\tilde{f}_k - \hat{f}_k| \leq S_N^*(k,\hat{\rho}) \cdot \|f\|_{\hat{\rho}}$$

where

$$S_N^*(k,\hat{\rho}) = \frac{e^{-2\pi\hat{\rho}N}}{1-e^{-2\pi\hat{\rho}N}} \left(e^{-2\pi\hat{\rho}k} + e^{2\pi\hat{\rho}k} \right).$$

Proof. Let $k \in \mathbb{Z}$. From Lemma 3.4 and the fact that $|\hat{f}_k| \leq e^{-2\pi|k|\hat{\rho}} \|f\|_{\hat{\rho}}$, we obtain

$$\begin{aligned} |\tilde{f}_k - \hat{f}_k| &= \Big| \sum_{m\in\mathbb{Z}} \hat{f}_{k+Nm} - \hat{f}_k \Big| = \Big| \sum_{m\in\mathbb{Z}\setminus\{0\}} \hat{f}_{k+Nm} \Big| \leq \\ &\leq \sum_{m\in\mathbb{Z}\setminus\{0\}} |\hat{f}_{k+Nm}| \leq \sum_{m\in\mathbb{Z}\setminus\{0\}} e^{-2\pi\hat{\rho}|k+Nm|} \cdot \|f\|_{\hat{\rho}}. \end{aligned}$$

Then, we define

$$S_N^*(k,\hat{\rho}) = \sum_{m\in\mathbb{Z}\setminus\{0\}} e^{-2\pi\hat{\rho}|k+Nm|}$$

so we have $|\tilde{f}_k - \hat{f}_k| \leq S_N^*(k,\hat{\rho}) \cdot \|f\|_{\hat{\rho}}$. Notice that for $k = -\left[\frac{N}{2}\right], \cdots, \left[\frac{N-1}{2}\right]$, if $m > 0$, $k+Nm > 0$, and if $m < 0$, $k+Nm < 0$. We must find then a suitable expression for $S_N^*(k,\hat{\rho})$, so

$$\begin{aligned} S_N^*(k,\hat{\rho}) &= \sum_{m>0} e^{-2\pi\hat{\rho}(k+Nm)} + \sum_{m<0} e^{-2\pi\hat{\rho}(-k-Nm)} = e^{-2\pi\hat{\rho}k} \sum_{m>0} e^{-2\pi\hat{\rho}Nm} + e^{2\pi\hat{\rho}k} \sum_{m<0} e^{2\pi\hat{\rho}Nm} \\ &\leq e^{-2\pi\hat{\rho}k} \sum_{m>0} e^{-2\pi\hat{\rho}Nm} + e^{2\pi\hat{\rho}k} \sum_{m>0} e^{-2\pi\hat{\rho}Nm} = \frac{e^{-2\pi\hat{\rho}N}}{1-e^{-2\pi\hat{\rho}N}} \left(e^{-2\pi\hat{\rho}k} + e^{2\pi\hat{\rho}k} \right). \end{aligned}$$

□

Theorem 3.6. *Let $f : \mathbb{T}_{\hat\rho} \longrightarrow \mathbb{C}$ be an analytic and bounded function in the complex strip $\mathbb{T}_{\hat\rho}$ of size $\hat\rho > 0$. Let $\tilde f$ be the discrete Fourier approximation of f in the regular grid of size N even. Then, for $0 \le \rho < \hat\rho$, we have*

$$\|\tilde f - f\|_\rho \le C_N(\rho,\hat\rho) \cdot \|f\|_{\hat\rho}$$

where

$$C_N(\rho,\hat\rho) = S_N^{*1}(\rho,\hat\rho) + S_N^{*2}(\rho,\hat\rho) + T_N(\rho,\hat\rho)$$

with

$$\begin{aligned} S_N^{*1}(\rho,\hat\rho) &= \frac{e^{-2\pi\hat\rho N}}{1-e^{-2\pi\hat\rho N}} \frac{e^{-2\pi(\hat\rho+\rho)}+1}{e^{-2\pi(\hat\rho+\rho)}-1}\left(1-e^{\pi(\hat\rho+\rho)N}\right), \\ S_N^{*2}(\rho,\hat\rho) &= \frac{e^{-2\pi\hat\rho N}}{1-e^{-2\pi\hat\rho N}} \frac{e^{2\pi(\hat\rho-\rho)}+1}{e^{2\pi(\hat\rho-\rho)}-1}\left(1-e^{-\pi(\hat\rho-\rho)N}\right), \\ T_N(\rho,\hat\rho) &= \frac{e^{2\pi(\hat\rho-\rho)}+1}{e^{2\pi(\hat\rho-\rho)}-1} e^{-\pi(\hat\rho-\rho)N}. \end{aligned}$$

Proof. From the definition of the discrete Fourier approximation $\tilde f$ of f, we have

$$\|\tilde f - f\|_\rho \le \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} |\tilde f_k - \hat f_k|\, e^{2\pi\rho|k|} + \sum_{k=-\infty}^{-\frac{N}{2}-1} |\hat f_k|\, e^{-2\pi\rho k} + \sum_{k=fracN2}^{\infty} |\hat f_k|\, e^{2\pi\rho k}.$$

From Proposition 3.5 and the growth rate properties of the Fourier coefficients of an analytic function, we get

$$\|\tilde f - f\|_\rho \le (S_N^*(\rho,\hat\rho) + T_N(\rho,\hat\rho)) \cdot \|f\|_{\hat\rho}$$

where

$$S_N^*(\rho,\hat\rho) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} S_N^*(k,\hat\rho)\, e^{2\pi\rho|k|}$$

and

$$T_N(\rho,\hat\rho) = \sum_{k=-\infty}^{-\frac{N}{2}-1} e^{2\pi(\hat\rho-\rho)k} + \sum_{k=\frac{N}{2}}^{\infty} e^{-2\pi(\hat\rho-\rho)k}.$$

Let's express $T_N(\rho,\hat\rho)$ in computable terms. Notice that

$$\begin{aligned} T_N(\rho,\hat\rho) &= e^{-2\pi(\hat\rho-\rho)k\frac{N}{2}} + 2\sum_{k=\frac{N}{2}+1}^{\infty} e^{-2\pi(\hat\rho-\rho)k} = e^{-2\pi(\hat\rho-\rho)k\frac{N}{2}} + 2\,\frac{e^{-2\pi(\hat\rho-\rho)(\frac{N}{2}+1)}}{1-e^{-2\pi(\hat\rho-\rho)}} \\ &= e^{-\pi(\hat\rho-\rho)N}\frac{1+e^{-2\pi(\hat\rho-\rho)}}{1-e^{-2\pi(\hat\rho-\rho)}} = \frac{e^{2\pi(\hat\rho-\rho)}+1}{e^{2\pi(\hat\rho-\rho)}-1} e^{-\pi(\hat\rho-\rho)N}. \end{aligned}$$

Using the results obtained in Proposition 3.5 we compute

$$
\begin{aligned}
S_N^*(\rho,\hat\rho) &= \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} S_N^*(k,\hat\rho)\, e^{2\pi\rho|k|} \le \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \frac{e^{-2\pi\hat\rho N}}{1-e^{-2\pi\hat\rho N}} \left(e^{-2\pi\hat\rho k} + e^{2\pi\hat\rho k}\right) e^{2\pi\rho|k|} \\
&= \frac{e^{-2\pi\hat\rho N}}{1-e^{-2\pi\hat\rho N}} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \left(e^{-2\pi\hat\rho k} + e^{2\pi\hat\rho k}\right) e^{2\pi\rho|k|} \,.
\end{aligned}
$$

Let's compute the last sum. It is equal to

$$
\begin{aligned}
&\left(e^{2\pi\hat\rho\frac{N}{2}} + e^{-2\pi\hat\rho\frac{N}{2}}\right) e^{2\pi\rho\frac{N}{2}} + \sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}-1} \left(e^{-2\pi\hat\rho k} + e^{2\pi\hat\rho k}\right) e^{2\pi\rho|k|} \\
&= \left(e^{\pi(\hat\rho+\rho)N} + e^{-\pi(\hat\rho-\rho)N}\right) + 2 + 2\sum_{k=1}^{\frac{N}{2}-1} \left(e^{-2\pi\hat\rho k} + e^{2\pi\hat\rho k}\right) e^{2\pi\rho k} \\
&= e^{\pi(\hat\rho+\rho)N} + e^{-\pi(\hat\rho-\rho)N} + 2 + 2e^{-2\pi(\hat\rho-\rho)} \frac{e^{-2\pi(\hat\rho-\rho)(\frac{N}{2}-1)}-1}{e^{-2\pi(\hat\rho-\rho)}-1} + 2e^{2\pi(\hat\rho+\rho)} \frac{e^{2\pi(\hat\rho+\rho)(\frac{N}{2}-1)}-1}{e^{2\pi(\hat\rho+\rho)}-1} \\
&= 2 + \frac{e^{-2\pi(\hat\rho-\rho)-\pi(\hat\rho-\rho)N} - e^{-\pi(\hat\rho-\rho)N} + 2e^{-\pi(\hat\rho-\rho)N} - 2e^{-2\pi(\hat\rho-\rho)} + e^{-\pi(\hat\rho-\rho)N}}{e^{-2\pi(\hat\rho-\rho)}} \\
&\quad + \frac{e^{2\pi(\hat\rho+\rho)+\pi(\hat\rho+\rho)N} - e^{\pi(\hat\rho+\rho)N} + 2e^{\pi(\hat\rho+\rho)N} - 2e^{2\pi(\hat\rho+\rho)}}{e^{2\pi(\hat\rho+\rho)}} \\
&= \frac{e^{-\pi(\hat\rho-\rho)N}\left(1+e^{-2\pi(\hat\rho-\rho)}\right) - \left(1+e^{-2\pi(\hat\rho-\rho)}\right)}{e^{-2\pi(\hat\rho-\rho)}-1} + \frac{e^{\pi(\hat\rho+\rho)N}\left(1+e^{2\pi(\hat\rho+\rho)}\right) - \left(1+e^{2\pi(\hat\rho+\rho)}\right)}{e^{2\pi(\hat\rho+\rho)}-1} \\
&= \left(1-e^{-\pi(\hat\rho-\rho)N}\right) \frac{1+e^{-2\pi(\hat\rho-\rho)}}{1-e^{-2\pi(\hat\rho-\rho)}} + \left(e^{\pi(\hat\rho+\rho)N}-1\right) \frac{1+e^{-2\pi(\hat\rho+\rho)}}{1-e^{-2\pi(\hat\rho+\rho)}} \\
&= \frac{e^{-2\pi(\hat\rho+\rho)}+1}{e^{-2\pi(\hat\rho+\rho)}-1} \left(1-e^{\pi(\hat\rho+\rho)N}\right) + \frac{e^{2\pi(\hat\rho-\rho)}+1}{e^{2\pi(\hat\rho-\rho)}-1} \left(1-e^{-\pi(\hat\rho-\rho)N}\right) \,.
\end{aligned}
$$

Hence we have that

$$
S_N^*(\rho,\hat\rho) = \frac{e^{-2\pi\hat\rho N}}{1-e^{-2\pi\hat\rho N}} \left(\frac{e^{-2\pi(\hat\rho+\rho)}+1}{e^{-2\pi(\hat\rho+\rho)}-1} \left(1-e^{\pi(\hat\rho+\rho)N}\right) + \frac{e^{2\pi(\hat\rho-\rho)}+1}{e^{2\pi(\hat\rho-\rho)}-1} \left(1-e^{-\pi(\hat\rho-\rho)N}\right) \right) \,.
$$

Which finally gives us

$$
\begin{aligned}
S_N^{*1}(\rho,\hat\rho) &= \frac{e^{-2\pi\hat\rho N}}{1-e^{-2\pi\hat\rho N}} \frac{e^{-2\pi(\hat\rho+\rho)}+1}{e^{-2\pi(\hat\rho+\rho)}-1} \left(1-e^{\pi(\hat\rho+\rho)N}\right), \\
S_N^{*2}(\rho,\hat\rho) &= \frac{e^{-2\pi\hat\rho N}}{1-e^{-2\pi\hat\rho N}} \frac{e^{2\pi(\hat\rho-\rho)}+1}{e^{2\pi(\hat\rho-\rho)}-1} \left(1-e^{-\pi(\hat\rho-\rho)N}\right) \,.
\end{aligned}
$$

□

Remark 3.7. We have proved the case in which N is even since in our implementation we will choose our N even. As we will soon see, the fact that N is even (and furthermore, a power of two) speeds the calculations up for a certain type of transform, the Fast Fourier Transform (FFT). The proof for the case in which N is odd can be found in [11], and a more general proof for a multi-dimensional scenario can be found in [3].

3.2.2 Matrices of Periodic Functions

In this section we will focus on the control of the propagation error when we perform matrix operations, mainly products and inverses. The procedures for other operations are analogous. The results hereby presented are no more than consequences of Theorem 3.6 from the previous section.

Corollary 3.8. *Let us consider two matrix functions $A : \mathbb{T} \to \mathbb{C}^{m_1 \times m_2}$, and $B : \mathbb{T} \to \mathbb{C}^{m_2 \times m_3}$, such that their entries are analytic and bounded functions in the complex strip $\mathbb{T}_{\hat{\rho}}$ of size $\hat{\rho} > 0$. We denote by AB the product matrix and $\widetilde{AB}$ the corresponding approximation given by DFT. Given a grid of size $N \in \mathbb{N}$, we evaluate A and B in the grid, and we interpolate the points $AB(\theta_j) = A(\theta_j)B(\theta_j)$. Then, we have*

$$\|AB - \widetilde{AB}\|_\rho \leq C_N(\rho, \hat{\rho})\, \|A\|_{\hat{\rho}}\, \|B\|_{\hat{\rho}}$$

for every $0 \leq \rho < \hat{\rho}$.

Corollary 3.9. *Let us consider a matrix function $A : \mathbb{T} \to \mathbb{C}^{m \times m}$ whose entries are analytic and bounded functions in the complex strip $\mathbb{T}_{\hat{\rho}}$ of size $\hat{\rho} > 0$. Given a grid of size $N \in \mathbb{N}$, we evaluate A in the grid and compute the inverses $X(\theta_j) = A(\theta_j)^{-1}$. Then, if $\widetilde{X}$ is the corresponding discrete Fourier approximation associated with the sample $X(\theta_j)$, the error $E(\theta) = Id_m - A(\theta)\widetilde{X}(\theta)$ satisfies*

$$\|E\|_\rho \leq C_N(\rho, \hat{\rho})\, \|A\|_{\hat{\rho}}\, \|\widetilde{X}\|_{\hat{\rho}}$$

for $0 \leq \rho < \hat{\rho}$. Moreover, if $\|E\|_\rho < 1$, there exists an analytic inverse $A^{-1} : \mathbb{T} \to \mathbb{C}^{m \times m}$ satisfying

$$\|A^{-1} - \widetilde{X}\|_\rho \leq \frac{\|\widetilde{X}\|_{\hat{\rho}}\, \|E\|_\rho}{1 - \|E\|_\rho}\,.$$

Proof. To obtain the first inequality of the Corollary, we observe that if $\widetilde{A\widetilde{X}}$ is the discrete Fourier approximation of $A\widetilde{X}$, then it turns out that

$$(A\widetilde{X})(\theta_j) = A(\theta_j)\widetilde{X}(\theta_j) = Id_m$$

for all points in the grid. This implies that $\widetilde{A\widetilde{X}} = Id_m$, and we end up with

$$\|E\|_\rho = \|Id_m - A\widetilde{X}\|_\rho = \|\widetilde{A\widetilde{X}} - A\widetilde{X}\|_\rho$$

and the inequality follows applying Corollary 3.8. The second inequality follows from the expression $E = Id_m - A\widetilde{X}$, simply writing $A^{-1} = \widetilde{X}(Id_m - E)^{-1}$ and using a Neumann series argument.

□

3.3 The Fast Fourier Transform

A Fast Fourier Transform (FFT) is an implementation algorithm for the Discrete Fourier Transform (DFT) but with a significant decrease of computational cost. Even though the number of operations of a regular DFT has a $O(N^2)$ order, the number of operations for the FFT has a $O(N \log N)$ order. There are several algorithms that are able to achieve such low computational cost, but the most common and used is the Cooley-Tukey FFT algorithm, which is the one we are going to explain in this section (as extracted from [7]).

The main idea of the Cooley-Tukey algorithm is to break down a DFT of any composite size $N = N_1 N_2$ into many smaller DFTs of sizes N_1 and N_2. This allows us to combine this algorithm with any other algorithm for the DFT, for instance algorithms that are able to handle large prime factors that cannot be decomposed by Cooley-Tukey.
The decomposition we are going to explain is the one used in the best known use of the Cooley-Tukey algorithm. It divides the transform into two pieces of size $N/2$ at each step, which limites itself to values of $N = 2^p$ for $p \in \mathbb{N}$. This is not a problem in general since the number of sample points N can usually be chosen freely. This decomposition is called the radix-2 case, and for other factorizations of N we call them the mixed-radix cases or split-radix.

The radix-2 decimation-in-time (DIT) FFT divides a DFT of size N into two interleaved DFTs of size $N/2$ with each recursive stage.
The DFT is defined, as we have previously seen, by the formula

$$\tilde{f}_k = \frac{1}{N} \sum_{j=0}^{N-1} f_j \, e^{-\frac{2\pi i}{N} j \, k}.$$

The radix-2 DIT first computes the DFTs of the even-indexed inputs ($f_{2m} = f_0, f_2, \ldots$ $\ldots, f_{N-2}$) and of the odd-indexed inputs ($f_{2m+1} = f_1, f_3, \ldots, f_{N-1}$), and then combines those two results to produce the DFT of the whole sequence. The algorithm rearranges the DFT of the function f_j into a sum over the even-numbered indices $j = 2m$ and a sum over the odd-numbered indices $j = 2m + 1$.

$$
\begin{aligned}
\tilde{f}_k &= \frac{1}{2}\left(\frac{1}{N/2}\sum_{m=0}^{N/2-1} f_{2m}\, e^{-\frac{2\pi i}{N}(2m)\,k}\right) + \frac{1}{2}\left(\frac{1}{N/2}\sum_{m=0}^{N/2-1} f_{2m+1}\, e^{-\frac{2\pi i}{N}(2m+1)\,k}\right) \\
&= \frac{1}{2}\left(\frac{1}{N/2}\sum_{m=0}^{N/2-1} f_{2m}\, e^{-\frac{2\pi i}{N/2}m\,k}\right) + \frac{1}{2}\, e^{-\frac{2\pi i}{N}k}\left(\frac{1}{N/2}\sum_{m=0}^{N/2-1} f_{2m+1}\, e^{-\frac{2\pi i}{N/2}m\,k}\right) \\
&= \frac{1}{2}E_k + \frac{1}{2}\, e^{-\frac{2\pi i}{N}k}O_k.
\end{aligned}
$$

It is clear that the sums within the last two parentheses are the DFT of the even-indexed part f_{2m} and the DFT of odd-indexed part f_{2m+1} of the function f_j. We can denote the DFT of the even-indexed part f_{2m} by E_k and the DFT of the odd-indexed part by O_k and simplify the resulting expression.

Taking advantage of the periodicity of the DFT, we know that $E_{k+\frac{N}{2}} = E_k$ and $O_{k+\frac{N}{2}} = O_k$ if $k < N/2$. Thus, we can rewrite the previous equation as

$$
\tilde{f}_k = \begin{cases} \frac{1}{2}E_k + \frac{1}{2}\, e^{-\frac{2\pi i}{N}k}\, O_k\,, & for \quad 0 \leq k < N/2 \\ \frac{1}{2}E_{k-N/2} + \frac{1}{2}\, e^{-\frac{2\pi i}{N}k}\, O_{k-N/2}\,, & for \quad N/2 \leq k < N\,. \end{cases}
$$

Noticing that

$$
e^{-\frac{2\pi i}{N}(k+N/2)} = e^{-\frac{2\pi i}{N}-\pi i} = e^{-\pi i}e^{-\frac{2\pi i}{N}k} = -e^{-\frac{2\pi i}{N}k}
$$

we can express $\tilde{f}_k$ as

$$
\begin{aligned}
\tilde{f}_k &= \frac{1}{2}E_k + \frac{1}{2}\, e^{-\frac{2\pi i}{N}k}\, O_k \qquad for \quad 0 \leq k < N/2\,, \\
\tilde{f}_{k+N/2} &= \frac{1}{2}E_k - \frac{1}{2}\, e^{-\frac{2\pi i}{N}k}\, O_k \quad for \quad 0 \leq k < N/2\,.
\end{aligned}
$$

Applying this method recursively, splitting into two half-size DFTs, gives a final output of a combination of E_k and $e^{-\frac{2\pi i}{N}k}\, O_k$, which is a very simple size-2 DFT. This procedure can reduce the overall runtime of the DFT, which is $O(N^2)$, to $O(N \log N)$, and moreover, increase the precision of the final results.

Notice that, even though we have explained the Cooley-Tukey algorithm to transform grid points into Fourier coefficients, the algorithm works as well for the inverse process. The only difference in the procedure is the disappearance of the $1/N$ factor and the change of sign of the exponent of the complex exponential, given that the formula for the IDFT, as we stated previously, is

$$
f_j = \sum_{k=0}^{N-1} \tilde{f}_k\, e^{\frac{2\pi i}{N}jk}.
$$

Thus, the factor 1/2 preceeding the sums also disappears, leaving us the formula

$$f_k = \sum_{m=0}^{N/2-1} \tilde{f}_{2m}\, e^{\frac{2\pi i}{N}(2m)\,k} + \sum_{m=0}^{N/2-1} \tilde{f}_{2m+1}\, e^{\frac{2\pi i}{N}(2m+1)\,k}.$$

Manipulating these terms in the same way we previously did, we obtain

$$f_k = \sum_{m=0}^{N/2-1} \tilde{f}_{2m}\, e^{\frac{2\pi i}{N/2} m\,k} + e^{\frac{2\pi i}{N} k} \sum_{m=0}^{N/2-1} \tilde{f}_{2m+1}\, e^{\frac{2\pi i}{N/2} m\,k} = \widetilde{E}_k + e^{\frac{2\pi i}{N} k}\, \widetilde{O}_k.$$

Again, $\widetilde{E}_{k+\frac{N}{2}} = \widetilde{E}_k$ and $\widetilde{E}_{k+\frac{N}{2}} = \widetilde{O}_k$ for $k < N/2$. We can now express f_k as

$$f_k = \begin{cases} \widetilde{E}_k + e^{\frac{2\pi i}{N} k}\, \widetilde{O}_k\,, & for \quad 0 \le k < N/2 \\ \widetilde{E}_{k-N/2} + e^{\frac{2\pi i}{N} k}\, \widetilde{O}_{k-N/2}\,, & for \quad N/2 \le k < N\,. \end{cases}$$

This time we have

$$e^{\frac{2\pi i}{N}(k+N/2)} = e^{\frac{2\pi i}{N} + \pi i} = e^{\pi i} e^{\frac{2\pi i}{N} k} = -e^{\frac{2\pi i}{N} k}$$

Which finally gives us

$$\begin{aligned} f_k &= \widetilde{E}_k + e^{\frac{2\pi i}{N} k}\, \widetilde{O}_k \qquad && for \quad 0 \le k < N/2\,, \\ f_{k+N/2} &= \widetilde{E}_k - e^{\frac{2\pi i}{N} k}\, \widetilde{O}_k && for \quad N/2 \le k < N\,. \end{aligned}$$

Chapter 4

Computer Assisted Proof

In this section we will introduce the procedure for which invariant tori are validated with a computer. Such validation consists of a computer program in C code that receives a torus (among other inputs) and checks whether it fulfills the conditions of the validation Theorem 2.11. If so, the computer gives a green light for the affirmaton of the existence of an invariant torus close to the approximately invariant one given in our input. During this validation process, the computer will have to handle error bounds (such as the invariance error bound or the reducibility error bound), which will possibly lead to the manipulation of very small numbers. In order to keep precision, we will use interval arithmetic so our bounds are precisely enclosed in a small range interval. This forces us to work with multi-precision and interval arithmetic packages, which are handled quite differently than regular double precision numbers.

The key point of the validation is the correspondence between Fourier coefficients and grid points of our torus. The ability to easily move from one space to the other using Fourier transforms (such as the FFT) will allow us to perform long or complicated calculations in a matter of just a multiplication by a constant (as in the case of the rotation) or other very simple and fast operations. Although this sounds very appealing, there is always a drawback. When performing certain operations on Fourier space, the correspondence with the grid points might be broken. Those situations may arise both when operating with matrices and when operating with vectors, and that is when we can make use of the results found in the dedicated Fourier chapter. In such cases one will have to proceed with care, handling properly the errors committed in those situations using high precision calculations.

In addition, a brief section on interval arithmetic is presented, so the reader can understand what is happening in the validation code when dealing with intervals.
Of course, an outline of the programming procedure is given at the end of the chapter so one can grasp the idea that the code is following.

4.1 Computation of Error Bounds

Given that we are working with quasi-periodically forced systems on complex environments, it is not straight-forward to make the computer perform some mathematical tasks, such as the calculation of the rotated torus on a complex environment of the grid. For such manipulations, Fourier transforms are commonly used, so it is easier to perform, following the same example, the rotation on the torus between the Fourier coefficients obtained from the points on the grid than applying the rotation directly on the grid points. Clearly there is a correspondence between grid points and Fourier coefficients given by the Fourier transform, but such relation can be broken when applying operations such matrix inversions or matrix products. In those cases, one has to proceed with caution, calculating the error produced in terms of Fourier transformations and taking them into account when bounding errors.
And that is precisely what this section is all about, finding computable expressions for our one-dimensional objects so the invariance and reducibility errors can be properly bounded by a computer.

Remark 4.1. Keep in mind that under the same Finsler norm, the errors produced are the same even if changing the θ support point of our torus. This means that in order to simplify calculations, for instance, we will take the invariance error as

$$E(\theta) = P_2(\theta+\omega)(F(K_0(\theta),\theta) - K_0(\theta+\omega))$$

instead of

$$E(\theta) = P_2(\theta)(F(K_0(\theta-\omega),\theta-\omega) - K_0(\theta))\,.$$

4.1.1 The Invariance Error Bound

Keeping the same notation as in the validation theorem for our analytic map F and our analytic approximately invariant torus K_0, we can write the error produced in the invariance equation as

$$E(\theta) = P_2(\theta+\omega)(F(K_0(\theta),\theta) - K_0(\theta+\omega))\,.$$

Notice that P_2 will be one of our inputs, provided therefore in the shape of a matrix of periodic functions, that is, truncated Fourier series, the type of object with which we will usually operate.

$$P_2(\theta) = \begin{pmatrix} \sum\limits_{k=-\left[\frac{N-1}{2}\right]}^{\left[\frac{N-1}{2}\right]} p_{1,1,k,2}\, e^{2\pi i k\theta} & \cdots & \sum\limits_{k=-\left[\frac{N-1}{2}\right]}^{\left[\frac{N-1}{2}\right]} p_{1,n,k,2}\, e^{2\pi i k\theta} \\ \vdots & \ddots & \vdots \\ \sum\limits_{k=-\left[\frac{N-1}{2}\right]}^{\left[\frac{N-1}{2}\right]} p_{n,1,k,2}\, e^{2\pi i k\theta} & \cdots & \sum\limits_{k=-\left[\frac{N-1}{2}\right]}^{\left[\frac{N-1}{2}\right]} p_{n,n,k,2}\, e^{2\pi i k\theta} \end{pmatrix}$$

$$= \sum_{k=-\left[\frac{N-1}{2}\right]}^{\left[\frac{N-1}{2}\right]} \begin{pmatrix} p_{1,1,k,2} & \cdots & p_{1,n,k,2} \\ \vdots & \ddots & \vdots \\ p_{n,1,k,2} & \cdots & p_{n,n,k,2} \end{pmatrix} e^{2\pi i k\theta} .$$

Since our theorem's main input object is the approximately invariant torus, we will take it also as a finite sum, and in case we pick N even, the Nyquist term will already be set to 0.

Remark 4.2. It is true that we have previously said that we have to take into account the error committed when we break the correspondence between grid points and Fourier coefficients, which is precisely what happens when we set the Nyquist term to 0. However, when doing so, we are just claiming that the object to be validated is the torus given in Fourier space with the Nyquist term set to 0. That means, that the previous torus evaluated on the grid is no longer our main object of study, and therefore no correspondence is broken.

Such torus will have the form

$$K_0(\theta) = \sum_{k=-\left[\frac{N}{2}\right]}^{\left[\frac{N-1}{2}\right]} \widetilde{K}_{0,k}\, e^{2\pi i k\theta} = \sum_{k=-\left[\frac{N-1}{2}\right]}^{\left[\frac{N-1}{2}\right]} \widetilde{K}_{0,k}\, e^{2\pi i k\theta} .$$

This expression is very useful since we can now easily obtain an analogous expression for $K_0(\theta+\omega)$,

$$K_0(\theta+\omega) = \sum_{k=-\left[\frac{N-1}{2}\right]}^{\left[\frac{N-1}{2}\right]} (\widetilde{K}_{0,k}\, e^{2\pi i k\omega})\, e^{2\pi i k\theta} .$$

We should keep in mind that our main goal in this section is to find a computable value for the error bound of the invariance equation, which will lead us at some point to manipulate the function $F(K_0(\theta),\theta)$ and its norm. Since the Fourier series of $\varphi(\theta) = F(K_0(\theta),\theta)$ is an infinite sum, we would like to approximate it by a finite sum

$$\widetilde{\varphi}(\theta) = \sum_{k=-\left[\frac{N}{2}\right]}^{\left[\frac{N-1}{2}\right]} \widetilde{\varphi}_k\, e^{2\pi i k\theta} .$$

Then, we want to obtain a rigorous bound of $\|\varphi(\theta) - \tilde{\varphi}(\theta)\|_\rho$.

Recalling now the invariance error equation, by taking norms, separating the $P_2(\theta+\omega)$ term and adding and subtracting $\widetilde{\varphi}(\theta)$ we see that,

$$\|E(\theta)\|_\rho \leq \|P_2(\theta+\omega)\|_\rho \left(\|F(K_0(\theta),\theta)-\widetilde{\varphi}(\theta)\|_\rho + \|\widetilde{\varphi}(\theta)-K_0(\theta+\omega)\|_\rho\right) .$$

For a given $\hat{\rho} > \rho$ such that $\{(K_0(\theta),\theta) \mid \theta \in \bar{\mathbb{T}}_{\hat{\rho}}\} \subset \mathcal{U}$, where $\mathcal{U}$ is the domain of F, using Theorem 3.6 we have

$$\|F(K_0(\theta),\theta)-\widetilde{\varphi}(\theta)\|_\rho = \|\varphi(\theta)-\widetilde{\varphi}(\theta)\|_\rho \leq C_N(\rho,\hat{\rho})\,\|\varphi\|_{\hat{\rho}} \leq C_N(\rho,\hat{\rho})\,\|F(K_0(\theta),\theta)\|_{\hat{\rho}} .$$

We have then left to calculate the second term of the sum, which follows

$$\begin{aligned}\|\widetilde{\varphi}(\theta)-K_0(\theta+\omega)\|_\rho &= \left\| \sum_{k=-\left[\frac{N}{2}\right]}^{\left[\frac{N-1}{2}\right]} \left(\widetilde{\varphi}_k - \widetilde{K}_{0,k}\, e^{2\pi i k\omega}\right) e^{2\pi i k\theta} \right\|_\rho \\ &\leq \left\| \sum_{k=-\left[\frac{N}{2}\right]}^{\left[\frac{N-1}{2}\right]} \left(\widetilde{\varphi}_k - \widetilde{K}_{0,k}\, e^{2\pi i k\omega}\right) e^{2\pi i k\theta} \right\|_{F,\rho} \leq \tilde{\varepsilon} .\end{aligned}$$

Then we find

$$\|P_2(\theta+\omega)(F(K_0(\theta),\theta)-K_0(\theta+\omega))\|_\rho \leq \|P_2(\theta+\omega)\|_\rho \left(C_N(\rho,\hat{\rho})\,\|F(K_0(\theta),\theta)\|_{\hat{\rho}} + \tilde{\varepsilon}\right) \leq \varepsilon ,$$

where ε is the invariance bound in Theorem 2.11, and $C_N(\rho,\hat{\rho})$, even though it depends on the system, is very small.

4.1.2 The Reducibility Error Bound

The next bound to be computed is the reducibility error bound, where the reducibility error is given by

$$E_{red}(\theta) = P_2(\theta+\omega)M_0(\theta)P_1(\theta) - \Lambda(\theta) ,$$

with $M_0(\theta) = \mathrm{D}F(K_0(\theta),\theta)$. Clearly, there are more objects in this equation than in the previous one, so we must know first how are we going to deal with each one of them.
Our first inputs will be, then, the matrix valued maps $P_1, P_2 : \bar{\mathbb{T}}_\rho \to M_n(\mathbb{C})$. Since we already showed P_2, we can directly say that P_1 will have the same form

$$P_1(\theta) = \sum_{k=-\left[\frac{N-1}{2}\right]}^{\left[\frac{N-1}{2}\right]} \begin{pmatrix} p_{1,1,k,1} & \cdots & p_{1,n,k,1} \\ \vdots & \ddots & \vdots \\ p_{n,1,k,1} & \cdots & p_{n,n,k,1} \end{pmatrix} e^{2\pi i k\theta} .$$

Next we have our analytic block-diagonal matrix-valued map

$$\Lambda(\theta) = \begin{pmatrix} \Lambda^s(\theta) & 0 \\ 0 & \Lambda^u(\theta) \end{pmatrix}$$

where $\Lambda^s : \bar{\mathbb{T}}_\rho \to M_{n_s}(\mathbb{C})$ and $\Lambda^u : \bar{\mathbb{T}}_\rho \to M_{n_u}(\mathbb{C})$, with $n = n_s + n_u$ into which we want to reduce our system. Again, this can be expressed as

$$\Lambda(\theta) = \sum_{k=-\left[\frac{N-1}{2}\right]}^{\left[\frac{N-1}{2}\right]} \begin{pmatrix} \lambda_{1,1,k} & \cdots & \lambda_{1,n_S,k} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \lambda_{n_S,1,k} & \cdots & \lambda_{n_S,n_S,k} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \lambda_{n_S+1,n_S+1,k} & \cdots & \lambda_{n_S+1,n,k} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \lambda_{n,n_S+1,k} & \cdots & \lambda_{n,n,k} \end{pmatrix} e^{2\pi i k\theta} .$$

All in all, we can think as these inputs as matrices of Fourier coefficients that can be turned into points on the grid by an inverse DFT.

We need now to approximate $M_0(\theta)$ with a DFT approximation with a sampling of N points over the regular grid, where we have our N fixed to a even number. Thus we will have

$$\widetilde{M_0}(\theta) = \sum_{k=-\left[\frac{N-1}{2}\right]}^{\left[\frac{N-1}{2}\right]} \begin{pmatrix} \widetilde{m}_{1,1,k} & \cdots & \widetilde{m}_{1,n,k} \\ \vdots & \ddots & \vdots \\ \widetilde{m}_{n,1,k} & \cdots & \widetilde{m}_{n,n,k} \end{pmatrix} e^{2\pi i k\theta} .$$

Recall that the validation theorem also talks about a constant λ, saying that there must be a λ such that $\|\Lambda^s\|_\rho \leq \lambda < 1$ and $\|(\Lambda^u)^{-1}\|_\rho \leq \lambda < 1$. In order to verify such condition we will need to define the Fourier norm of a matrix. There are several ways of doing so, such as taking the maximum over θ of the $\|\cdot\|_\infty$ norm of the matrix, but we are taking a different one. Let A be an $n \times n$ matrix depending on θ, then

$$\|A\|_{F,\rho} = \max_{1\leq i\leq n} \sum_{j=1}^{n} \|a_{ij}\|_{F,\rho} .$$

This norm still satisfies that $\|A(\theta)\|_\rho \leq \|A(\theta)\|_{F,\rho}$.

Now that we already have the necessary tools for bounding, we have to check if we can find a value $\lambda < 1$ such that $\|\Lambda^s\|_\rho \leq \|\Lambda^s\|_{F,\rho} \leq \lambda$. If this value exists, we have to check the second hypothesis, which is $\|(\Lambda^u)^{-1}\|_\rho \leq \lambda$. The calculation of this norm is not as direct as the previous one. The fact that $(\Lambda^u)^{-1}$ is the inverse of a matrix of Fourier series breaks the correspondence between grid points and Fourier coefficients, hence, we will have to proceed differently.
Notice that

$$\|(\Lambda^u)^{-1}\|_\rho \leq \|(\Lambda^u)^{-1} - \widetilde{(\Lambda^u)^{-1}}\|_\rho + \|\widetilde{(\Lambda^u)^{-1}}\|_\rho \leq \|(\Lambda^u)^{-1} - \widetilde{(\Lambda^u)^{-1}}\|_\rho + \|\widetilde{(\Lambda^u)^{-1}}\|_{F,\rho} .$$

We shall take the first term of the sum apart in order to apply Corollary 3.9, which handles the error while applying a DFT upon inverted matrices as long as the function entries of our matrix can be analytically extended to a complex strip of width $\hat{\rho}$, $\mathbb{T}_{\hat{\rho}}$, which holds true for our case since we are working with analytic functions.

$$\|(\Lambda^u)^{-1} - \widetilde{(\Lambda^u)^{-1}}\|_\rho \leq \frac{\|\widetilde{(\Lambda^u)^{-1}}\|_{\hat{\rho}} \, \|E_{finv}(\theta)\|_\rho}{1 - \|E_{finv}(\theta)\|_\rho}$$

where $E_{finv}(\theta) = Id_{n_U} - \Lambda^u(\theta) \, \widetilde{(\Lambda^u(\theta))^{-1}}$ as used in Corollary 3.9, which also gave us a very useful result, that claimed

$$\|E_{finv}(\theta)\|_\rho \leq C_N(\rho, \hat{\rho}) \, \|\Lambda^u\|_{\hat{\rho}} \, \|\widetilde{(\Lambda^u)^{-1}}\|_{\hat{\rho}} \, .$$

Now we can finally write

$$\begin{aligned} \|(\Lambda^u)^{-1} - \widetilde{(\Lambda^u)^{-1}}\|_\rho &\leq \frac{\|\widetilde{(\Lambda^u)^{-1}}\|_{\hat{\rho}} \, C_N(\rho, \hat{\rho}) \, \|\Lambda^u\|_{\hat{\rho}} \, \|\widetilde{(\Lambda^u)^{-1}}\|_{\hat{\rho}}}{1 - C_N(\rho, \hat{\rho}) \, \|\Lambda^u\|_{\hat{\rho}} \, \|\widetilde{(\Lambda^u)^{-1}}\|_{\hat{\rho}}} \\ &\leq \frac{C_N(\rho, \hat{\rho}) \, \|\Lambda^u\|_{F,\hat{\rho}} \, \|\widetilde{(\Lambda^u)^{-1}}\|^2_{F,\hat{\rho}}}{1 - C_N(\rho, \hat{\rho}) \, \|\Lambda^u\|_{F,\hat{\rho}} \, \|\widetilde{(\Lambda^u)^{-1}}\|_{F,\hat{\rho}}} \, . \end{aligned}$$

The last inequality holds due to the fact that Λ^u is a matrix of Fourier series, which means that there is no error produced while turning back to the points of the grid and forth again to the Fourier series. However, as we have said before, this is not true for $(\Lambda^u)^{-1}$ given that the inversion of the matrix breaks the direct and errorless correspondence between grid points and DFT coefficients.

Once we have expressed the desired norm in computable terms, is time now to check if $\|(\Lambda^u)^{-1}\|_\rho \leq \lambda$, that is, if

$$\|(\Lambda^u)^{-1}\|_\rho \leq \frac{C_N(\rho, \hat{\rho}) \, \|\Lambda^u\|_{F,\hat{\rho}} \, \|\widetilde{(\Lambda^u)^{-1}}\|^2_{F,\hat{\rho}}}{1 - C_N(\rho, \hat{\rho}) \, \|\Lambda^u\|_{F,\hat{\rho}} \, \|\widetilde{(\Lambda^u)^{-1}}\|_{F,\hat{\rho}}} + \|\widetilde{(\Lambda^u)^{-1}}\|_{F,\hat{\rho}} \leq \lambda \, .$$

In case this condition is not satisfied with the first λ we have picked, it may be interesting to play around with the λ value and try to find another $\lambda' < 1$ such that satisfies both conditions. Or we can simply increase a bit the value of λ until the condition is satisfied.

Once the issue is settled and we have a suitable λ, the next step is to find the bound for the error. Recall that

$$E_{red}(\theta) = P_2(\theta + \omega) M_0(\theta) P_1(\theta) - \Lambda(\theta) \, .$$

In order to calculate a suitable bound for $\|E_{red}(\theta)\|$ we will have to manipulate some terms so we can apply Theorem 3.6 and Corollary 3.8.

$$\begin{aligned}\|E_{red}(\theta)\|_\rho &= \|P_2(\theta+\omega)M_0(\theta)P_1(\theta) - \Lambda(\theta)\|_\rho \\ &\leq \|P_2(\theta+\omega)M_0(\theta)P_1(\theta) - \widetilde{P_2(\theta+\omega)M_0(\theta)P_1(\theta)}\|_\rho \\ &\quad + \|\widetilde{P_2(\theta+\omega)M_0(\theta)P_1(\theta)} - \Lambda(\theta)\|_\rho \\ &\leq C_N(\rho,\hat\rho)\|P_2(\theta+\omega)\|_{\hat\rho}\|M_0(\theta)\|_{\hat\rho}\|P_1(\theta)\|_{\hat\rho} + \|\widetilde{P_2(\theta+\omega)M_0(\theta)P_1(\theta)} - \Lambda(\theta)\|_\rho\,,\end{aligned}$$

where the last term remains the same since it is the norm of the difference of two matrices of trigonometric polynomials (given that $\Lambda(\theta)$ is an input), which is a matrix of trigonometric polynomials. Furthermore, using the Fourier norm inequality we obtain

$$\|E_{red}(\theta)\|_\rho \leq C_N(\rho,\hat\rho)\|P_2(\theta+\omega)\|_{F,\hat\rho}\|M_0(\theta)\|_{\hat\rho}\|P_1(\theta)\|_{F,\hat\rho} + \|\widetilde{P_2(\theta+\omega)M_0(\theta)P_1(\theta)} - \Lambda(\theta)\|_{F,\rho}.$$

Remark 4.3. In the example we will present in the following section, Λ will be constant.

4.1.3 The Invertibility Error Bound

Lastly, we seek a bound for the invertibility error. The invertibility error is the error produced when treating $P_2(\theta)$ as the inverse of $P_1(\theta)$, that is

$$E_{inv}(\theta) = P_2(\theta)P_1(\theta) - Id\,.$$

By simply taking norms and applying the same procedures as before we obtain

$$\begin{aligned}\|E_{inv}(\theta)\|_\rho &\leq \|P_2(\theta)P_1(\theta) - \widetilde{P_2(\theta)P_1(\theta)}\|_\rho + \|\widetilde{P_2(\theta)P_1(\theta)} - Id\|_\rho \\ &\leq C_N(\rho,\hat\rho)\;\|P_2(\theta)\|_{\hat\rho}\;\|P_1(\theta)\|_{\hat\rho} + \|\widetilde{P_2(\theta)P_1(\theta)} - Id\|_\rho \\ &\leq C_N(\rho,\hat\rho)\;\|P_2(\theta)\|_{F,\hat\rho}\;\|P_1(\theta)\|_{F,\hat\rho} + \|\widetilde{P_2(\theta)P_1(\theta)} - Id\|_{F,\rho}\,.\end{aligned}$$

4.1.4 Norm of a Bilinear Form

The last bound to be computed is the one related to the second differential of F, which is b. Recall from Theorem 2.11, that we had the following condition:

For all points (x,θ) in the strip

$$\bar D_\rho(K_0,r) = \{(x,\theta)\in\mathbb{C}^n\times\bar{\mathbb{T}}^d_\rho \mid x = K_0(\theta)+P_1(\theta)\xi,\ \xi\in\mathbb{C}^n,\ |\xi|_\theta\leq R\}\,,$$

the bilinear maps over the rotation ω

$$B(x,\theta) = P_2(\theta+\omega)\mathrm{D}_x^2F(x,\theta)[P_1(\theta)\cdot, P_1(\theta)\cdot]$$

satisfy $\|B(x,\theta)\| \leq b$ as a norm of a bilinear form.

It is natural, then, that we seek a computable way to find that b. For that we can simply start by

$$\|B(x,\theta)\| \leq \|P_2(\theta+\omega)\| \, \|\mathrm{D}_x^2 F(x,\theta)\| \, \|P_1(\theta)\|^2$$

Notice that the only expression for which we don't yet have a computable expression is the second differential of F. Since it is only a specific case of a bilinear form, let's define the norm of a more general bilinear form that we will call $H : \mathbb{C}^n \times \mathbb{C}^n \to \mathbb{C}^n$ such that it has m components. That is, we have

$$H = \begin{pmatrix} (H_{ij}^1) \\ \vdots \\ (H_{ij}^m) \end{pmatrix}_{i,j=1,\ldots,n} , \qquad v_s = \begin{pmatrix} v_s^1 \\ \vdots \\ v_s^n \end{pmatrix} \qquad s = 1, 2$$

$$H^k(v_1, v_2) = \sum_{i,j=1}^{n} H_{ij}^k \, v_1^i \, v_2^j \, .$$

With this, we can define

$$\begin{aligned} \|H\|_\infty &= \max_{v_1,v_2 \in \mathbb{C}^n \setminus \{0\}} \frac{|H(v_1,v_2)|_\infty}{|v_1|_\infty |v_2|_\infty} = \max_{v_1,v_2 \in \mathbb{C}^n \setminus \{0\}} \max_{k=1,\ldots,m} \frac{|H^k(v_1,v_2)|}{|v_1|_\infty |v_2|_\infty} \\ &\leq \max_{v_1,v_2 \in \mathbb{C}^n \setminus \{0\}} \frac{\max_{k=1,\ldots,m} \sum_{i,j=1}^n |H_{ij}^k| \, |v_1^i| \, |v_2^j|}{|v_1|_\infty |v_2|_\infty} \leq \max_{k=1,\ldots,m} \sum_{i,j=1}^{n} |H_{ij}^k| \, , \end{aligned}$$

where in the last inequality we have used that $|v_s| \leq |v_s|_\infty$ for $s = 1, 2$.

With such result, we can state that the norm of $B(x,\theta)$ can be bounded by

$$\|B(x,\theta)\| \leq \|P_2(\theta+\omega)\| \max_{(x,\theta) \in \bar{D}(K_0,R)} \|\mathrm{D}_x^2 F(x,\theta)\|_\infty \, \|P_1(\theta)\|^2 \, .$$

4.2 Intervalar Arithmetics

It is known that, although computers are great computing machines, they are not flawless. One of the most notable weaknesses they have is the incapability of representing certain numbers such as irrational numbers. Obviously, a computer only has a finite amount of memory to store floating-point numbers, which means that irrational numbers can only be represented up to a certain decimal, a truncation point (and not only irrationals, but also infinite decimal rationals). The error produced when truncating can be problematic in high precision calculations, such as in validated numerics, and that is why interval arithmetic is used. Instead of computing approximations of functions, the aim is to compute enclosures of functions. That is, in validated numerics one has to provide rigorous intervals for the coefficients of the approximations. Hence, the width of an enclosure gives a rigorous measurement of the quality of

the computation, so the first step to accomplish a validating program is to perform rigorous computations with intervals.
In rigorous computations, real numbers are substituted by intervals whose extrema are computer representable real numbers. In particular, when implementing interval operations in a computer, the result of an operation with intervals is an interval that includes the result. We must also be careful with this when retrieving the interval extrema after a calculation. The default rounding mode is usually set to round to the nearest floating-point value, but this is not the best method since it can leave our value of interest out of the interval when rounding. It is of more convenience to round the lower extrema towards $-\infty$ and the upper extrema towards $+\infty$ so the interval is just enlarged a bit (see [4, 8]). The package we will use for intervalar arithmetics is called MPFI, and will be implemented in C code. Let us formalize the expression of intervals a bit more.

From now on, we will denote intervals with boldface notation, such as $\mathbf{a} = [\underline{a}, \bar{a}]$ where $\underline{a}$ and $\bar{a}$ are called the lower and upper endpoints of $\mathbf{a}$, respectively. The set of real intervals is denoted by $\mathbb{IR}$, in which one can define arithmetic operations. We emphasize that, although real interval addition and multiplication are both associative and commutative, they fail to satisfy distributive law. Let's dive then into the definitions of the basic arithmetic operations between intervals.

4.2.1 Basic Operations

Without going much into detail, we present here the form of basic operations between intervals, the ones that will be used in our computer program by the interval arithmetic package [8].

- Addition: $\mathbf{x} + \mathbf{y} = [\underline{x}, \bar{x}] + [\underline{y}, \bar{y}] = [\underline{x} + \underline{y}, \bar{x} + \bar{y}]$.
- Negation: $-\mathbf{x} = -[\underline{x}, \bar{x}] = [-\bar{x}, -\underline{x}]$.
- Subtraction: $\mathbf{x} - \mathbf{y} = [\underline{x}, \bar{x}] - [\underline{y}, \bar{y}] = [\underline{x}, \bar{x}] + [-\bar{y}, -\underline{y}] = [\underline{x} - \bar{y}, \bar{x} - \underline{y}]$.
- Multiplication: $\mathbf{x} \cdot \mathbf{y} = [\underline{x}, \bar{x}] \cdot [\underline{y}, \bar{y}] = [\min\{\underline{x}\underline{y}, \underline{x}\bar{y}, \bar{x}\underline{y}, \bar{x}\bar{y}\}, \max\{\underline{x}\underline{y}, \underline{x}\bar{y}, \bar{x}\underline{y}, \bar{x}\bar{y}\}] = [\underline{xy}, \overline{xy}]$.
- Reciprocal: $1/\mathbf{x} = 1/[\underline{x}, \bar{x}] = [1/\bar{x}, 1/\underline{x}]$ if $\underline{x} > 0$ or $\bar{x} < 0$. Then we also have the cases $1/[\underline{x}, 0] \to [-\infty, 1/\underline{x}]$ and $1/[0, \bar{x}] \to [1/\bar{x}, \infty]$.
- Division: $\mathbf{x}/\mathbf{y} = [\underline{x}, \bar{x}]/[\underline{y}, \bar{y}] = [\underline{x}, \bar{x}] \cdot 1/[\underline{y}, \bar{y}] = [\underline{x}/\bar{y}, \bar{x}/\underline{y}]$.
- Powers:

$$\mathbf{x}^n = [\underline{x}, \bar{x}]^n = [\underline{x}^n, \bar{x}^n], \text{ if } n = 2k+1 \text{ for } k = 0, 1, \ldots$$

$$\mathbf{x}^n = [\underline{x}, \bar{x}]^n = \begin{cases} [\underline{x}^n, \bar{x}^n], \ \underline{x} \geq 0, \text{ if } n = 2k, \text{ for } k = 1, 2, \ldots \\ [\bar{x}^n, \underline{x}^n], \bar{x} < 0\ , \\ [0, \max\{\underline{x}^n, \bar{x}^n\}]\ , \text{ otherwise }. \end{cases}$$

- Absolute value:

$$|\mathbf{x}| = \begin{cases} [\min\{|\underline{x}|, |\bar{x}|\}, \max\{|\underline{x}|, |\bar{x}|\}] \text{ if } \underline{x}\bar{x} \geq 0\ , \\ [0, \max\{|\underline{x}|, |\bar{x}|\}] \text{ if } \underline{x}\bar{x} < 0\ . \end{cases}$$

Besides arithmetic operations, it is also useful to define equality and inequality operators.

- Equality: $\mathbf{x} = \mathbf{y} \iff [\underline{x}, \bar{x}] = [\underline{y}, \bar{y}] \iff (\underline{x} = \underline{y}) \wedge (\bar{x} = \bar{y})$.
- Inequality: $\mathbf{x} \leq \mathbf{y} \iff [\underline{x}, \bar{x}] \leq [\underline{y}, \bar{y}] \iff (\underline{x} \leq \underline{y}) \wedge (\bar{x} \leq \bar{y})$.
- Distance: $\mathrm{dist}(\mathbf{x}, \mathbf{y}) = \max\{|\underline{x} - \underline{y}|, |\bar{x} - \bar{y}|\}$.
- Maximum: $\max\{\mathbf{x}, \mathbf{y}\} = \max\{\bar{x}, \bar{y}\}$.
- Minimum: $\min\{\mathbf{x}, \mathbf{y}\} = \min\{\underline{x}, \underline{y}\}$.

These are the basic manipulations between intervals that are performed when using a intervalar arithmetic package, such as the one we are using, the MPFI, and this is how we will have to deal with the expressions from the validation theorem, by manipulating intervals around the values we know instead of using actual numbers. The explanation for the programming procedure awaits in the following section.

4.3 Computer Validation

Now that we have the required tools, we can proceed and validate a torus. For starters, we will have to get an approximately invariant torus out of somewhere. For that, we will read the output file generated by the code presented in [11]. In such code a reducibility method algorithm is implemented in C in order to calculate a torus.

Since the reducibility method (explained as well in [11]) also updates P_1 and Λ at each iteration, we will read from file K_0, P_1 and Λ and turn it to Fourier space via FFT. After this, we just have left to set the Nyquist term of every object to 0 before having the initial data fully prepared. Notice that in such data there is no P_2, so we will have to go back to grid space with P_1, invert it, and return to Fourier space. That should give us a good approximation for an inverse. The value of ω in [11] was taken as the golden ratio $\omega = \frac{\sqrt{5}-1}{2}$ (Diophantine irrational number) and so we will take it this time again, but enclosed within an interval.

Keep in mind that we are working in the complex field, but we also want to calculate with intervals instead of numbers. This means that we will have to create a new structure in our program, *complexi*, that represents complex intervals, that is, objects of the form $[\underline{x}, \bar{x}] \times [\underline{y}, \bar{y}]$, where $[\underline{x}, \bar{x}]$ is the real part interval and $[\underline{y}, \bar{y}]$ is the imaginary part interval. With this, all new functions have to be created so basic operations between complex intervals are covered. For this, we will use the MPFI package to, firstly set all intervals to work with a ~ 30 digit precision, and then create all the functions for complex intervals using the package's own interval operations functions applied to the real and imaginary parts. With the basic operations between complex intervals we can construct more intricate functions, such as the much needed FFT or other vectors and matrix operations.

Once all functions are created, the results found in Section 4.1 are applied using complex intervals so the constants $\lambda, \varepsilon, \sigma$ and τ from Theorem 2.11 are found (of course, in the form of a real interval, since they are norm bounds). Notice that we have to evaluate a ρ-norm of the non-truncated object $F(K_0(\theta), \theta)$ and its differential. For that, we will have to evaluate each object (let's take $F(K_0(\theta), \theta)$ as the example) over a complex neighborhood of our grid. Such new extended domain is made of the complex boxes $C_j = \{\theta_j + \varphi \mid |\mathrm{Re}\, \varphi| \leq \frac{1}{2N},\ |\mathrm{Im}\, \varphi| \leq \hat{\rho}\}$ (notice that the choice of ρ or $\hat{\rho}$ will depend on the context). First we will have to calculate the image of such boxes through K before applying F. Thus, we first need to calculate $K_0(\theta + \varphi)$. Notice that this is no more than rotating the torus as we have done before, but this time the rotation is complex.

$$K_0(\theta + \varphi) = \sum_{k=-\left[\frac{N-1}{2}\right]}^{\left[\frac{N-1}{2}\right]} (\widetilde{K}_{0,k}\, e^{2\pi i k \varphi})\, e^{2\pi i k \theta} = \sum_{k=-\left[\frac{N-1}{2}\right]}^{\left[\frac{N-1}{2}\right]} \left(\frac{\widetilde{K}_{0,k}}{e^{2\pi k \mathrm{Im} \varphi}}\, e^{2\pi i k \mathrm{Re} \varphi} \right) e^{2\pi i k \theta} .$$

So we just have to change the Fourier coefficients and rotate by a factor of $\mathrm{Re}\, \varphi$. With this new torus, we can calculate $F(K(C_j), C_j)$ for each j and its differential. We just have to find its supremum norm afterwards.

The procedure for calculating b is quite analogous. The only more problematic object is the norm of the second differential. Recall that the domain in which $\|B(z, \theta)\| \leq b$ (adapted to our case, where $z = (x, y)$) is

$$\bar{D}_\rho(K_0, R) = \{(z, \theta) \in \mathbb{C}^n \times \bar{\mathbb{T}}^d_\rho \mid z = K_0(\theta) + P_1(\theta)\xi,\ \xi \in \mathbb{C}^n,\ |\xi|_\theta \leq R\} .$$

Hence, our goal is to evaluate each of our second differential matrices (and picking the largest one) with z's that belong to the space created by the sum of an extended torus and an extended $P_1\xi$. For that, we will need to extend both the torus and P_1. After performing the same operations as before, we go back to grid space with both $K_0(C_j)$ and $P_1(C_j)\xi$ for each j to add them together. Then we evaluate them on the norm of the second differential (taking the maximum between both components) and take the supremum for each C_j. The calculation of the remaining constants follows naturally.

4.3.1 Study Case: The Standard Map

Our case of study is the perturbed Standard Map since it is the one with which the approximate torus is computed. Thus, $\hat{F}$ is $(F, f) : \mathbb{C}^2 \times \mathbb{T}_\rho \to \mathbb{C}^2 \times \mathbb{T}_\rho$ given by

$$\begin{cases} f(\theta) = \theta + \omega \\ F^x(x, y) = x + y - \frac{\kappa}{2\pi} \sin(2\pi x) - \varepsilon \sin(2\pi\theta) \\ F^y(x, y) = y - \frac{\kappa}{2\pi} \sin(2\pi x) - \varepsilon \sin(2\pi\theta) \end{cases}$$

with differential matrix

$$\mathrm{D}_{x,y}F(x, y) = \begin{pmatrix} 1 - \kappa \cos(2\pi x) & 1 \\ -\kappa \cos(2\pi x) & 1 \end{pmatrix}$$

and second differential matrices for components x and y

$$\mathrm{D}^2 F^x(x, y) = \begin{pmatrix} 2\pi\kappa \sin(2\pi x) & 0 \\ 0 & 0 \end{pmatrix} \quad \mathrm{D}^2 F^y(x, y) = \begin{pmatrix} 2\pi\kappa \sin(2\pi x) & 0 \\ 0 & 0 \end{pmatrix} .$$

With this map, the approximate torus is calculated using the reducibility method within a continuation method for ε. Applying the previous explanation to this specific map will yield the necessary constants and error bounds for the validation (see the Annex for details on the implementation of the previous explanation to this case).

4.3.2 Results

In this section, some output examples of validations will be displayed. We have taken different inputs for different values of ε and checked the conditions for the torus to be validated.

We will start with the first case $\varepsilon = 0.1$ (which is almost a planer torus) and play around with ρ and $\hat{\rho}$ to see what happen to the errors and constants. We can start by setting $\rho = 0$ so we first look at the real torus with $R = 10^{-4}$. For the calculation of Fourier norms we will use $\hat{\rho} = 10^{-2}$, but for the inflation of the torus we will use $\hat{\rho} = 5 \cdot 10^{-3}$.

```
CN(rho, rhohat): [6.8047018919335281312076456046925e-13,6.8047018919335281312076456070531e-13]

Invariance error: [1.5909644176201588172860064471273e-11,1.5932268362185730186487664094488e-11]

Lambda: [3.3828548368011231753495616835876e-1,3.3828548368011613854985332359308e-1]

Reducibility Error: [3.0953037229975405447470059225473e-9,3.0958269249002620014168739252433e-9]

Inversion Error: [9.2463408143891988645614886642009e-10,9.2463408143895368910319512333124e-10]

b: [2.8017225229974841450476962477150,3.4314984707078682189822147117411]

lambda + sigma + tau < 1

h: [1.0179917753413927022556095782971e-10,1.2485907231844659912024893932707e-10]
```

```
h < 1/2

r0: [1.9602613295325132697310121913807e-11,2.9531315149394923545195111500449e-11]

r0 < R

mu: [2.2475075849175180504485006805087e-9,2.2654382229911630953803933481896e-9]

mu < 1/(1+sqrt(2))


Congratulations, there exists a hyperbolic invariant torus with invariant subbundles.
Moreover, it is unique within a radius 9.9999999999999999999999999999998e-5
 and it is contained within a radius 2.9531315149394923545195111500448e-11.
 What’s more, the distance between approximately invariant bundles
and the invariant bundles is smaller than 2.2654382332555838495193443527494e-9
```

Let’s see what happens if we take a complex torus with $\rho = 10^{-3}$.

```
CN(rho, rhohat): [1.7151174562071982359681427214174e-11,1.7151174562071982359681427219777e-11]

Invariance error: [1.2671751643804756352142247008045e-10,1.2728776894796857823594169011216e-10]

Lambda: [3.3828548368568411773038311949549e-1,3.3828548368568793874528040059926e-1]

Reducibility Error: [7.7218585454551975545008788694923e-8,7.7231772699429858729316817325368e-8]

Inversion Error: [2.3305297996373874253806384903207e-8,2.3305297996374717600595203447535e-8]

b: [2.8019370675805860925791246439299,3.4317663589204338906579305969056]

lambda + sigma + tau < 1

h: [8.1087486197811468693993435736664e-10,9.9761551537244109710919947589108e-10]

h < 1/2

r0: [1.5565271442923392613575933794970e-10,2.3666031400294644837325377618902e-10]

r0 < R

mu: [5.5981071546925876047236822389787e-8,5.6129754182266561634778270625012e-8]

mu < 1/(1+sqrt(2))


Congratulations, there exists a hyperbolic invariant torus with invariant subbundles.
Moreover, it is unique within a radius 9.9999999999999999999999999999998e-5
 and it is contained within a radius 2.3666031400294644837325377618901e-10.
 What’s more, the distance between approximately invariant bundles
and the invariant bundles is smaller than 5.6129760483366231795665949214173e-8
```

By looking at $C_N(\rho, \hat{\rho})$ we can see the estimates are worse, and of course, ρ plays a crucial role in that since it gives the band for which the torus is real-analytic. Let's see what happens if we decrease the $\hat{\rho}$ used for the norms to $2 \cdot 10^{-3}$, for instance.

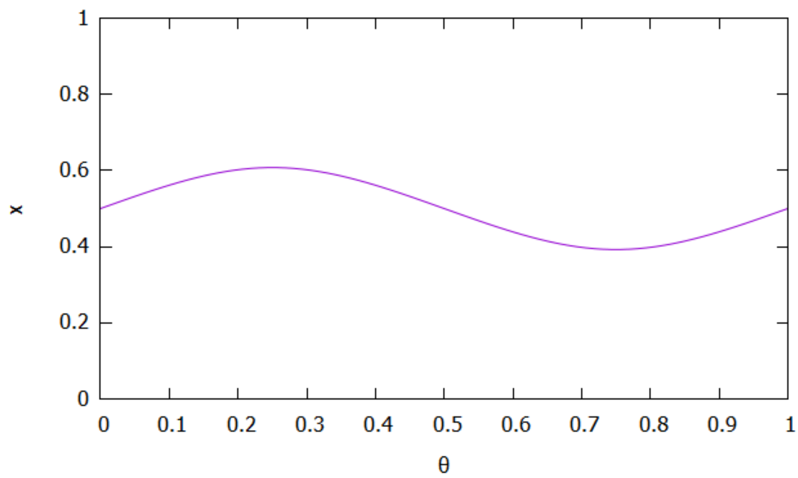
```
CN(rho, rhohat): [1.700925059305839987672363932619e1,1.700925059305839987672363933056e1]

Invariance error: [1.200744597603107810318169828125e1,1.257297971292069895532034024874e1]

Lambda: [3.382854836799999986851616995409e-1,3.382854836799999986851616995409e-1]

Reducibility Error: [1.130550318150456188050686661724e2,1.130744201415352278522171512063e2]

Inversion Error: [3.426422530815323063934624035217e1,3.426422530815323063934750613223e1]

b: [2.8019370675805860925791246439299,3.4317663589204338906579305969056]

lambda + sigma + tau >= 1, condition not satisfied

h: [1.5638150541687520970518216456548e-3,2.0060747890028793391902878200682e-3]

h < 1/2

r0: [-1.1008592908211105868724739470982e-1,-6.3865611346287163627016135577732e-2]

r0 < R

mu: [-2.4596440744294486693543550197822,-2.4571379585268243467905628052118]

mu < 1/(1+sqrt(2))

The initial torus does not meet the requirements
to ensure the existence of an invariant torus
```

Notice the big change in $C_N(\rho, \hat{\rho})$, which ultimately leads to the impossibility of validating the torus. This means that the values $\rho, \hat{\rho}$ and also N are related to each other. A poor configuration of them can lead to big exponents in complex exponentials, which translates into large values of $C_N(\rho, \hat{\rho})$. Let's revert the value of the $\hat{\rho}$ we just changed and increase the value of the $\hat{\rho}$ for which we inflate the torus. Set $\hat{\rho} = 10^{-2}$.

```
CN(rho, rhohat): [1.7151174562071982359681427214174e-11,1.7151174562071982359681427219777e-11]

Invariance error: [1.1460988119269239890219488624462e-10,3.7366228296279814862444928125141e155]

Lambda: [3.3828548368568411773038311949549e-1,3.3828548368568793874528040059926e-1]

Reducibility Error: [2.3627902796587507103658356522390e-8,2.2981990896026450621867537082922e159]

Inversion Error: [2.3305297996373874253806384903207e-8,2.3305297996374717600595203447535e-8]

b: [0,1.8125957313665597697531662867201e168]

lambda + sigma + tau >= 1, condition not satisfied
```

```
h: [-@Inf@,@Inf@]

h >= 1/2, condition not satisfied

r0: [@NaN@,@NaN@]

r0 >= R, condition not satisfied

mu: [@NaN@,@NaN@]

mu >= 1/(1+sqrt(2)), condition not satisfied

The initial torus does not meet the requirements
to ensure the existence of an invariant torus
```

The change in this case is even more drastic, we see that solutions explode by just tweaking $\hat{\rho}$ a little bit. It is important then to play around and select the best values of ρ and $\hat{\rho}$ so everything works properly.
Now we will show the bounds and constants for a couple more values of ε with the same $\rho = 10^{-3}$, $\hat{\rho} = 5 \cdot 10^{-3}$ for the inflation and $\hat{\rho} = 10^{-2}$ for the norms values. Starting for instance with $\varepsilon = 0.5$. The x component of the initial torus has the following shape (taken from [11]):

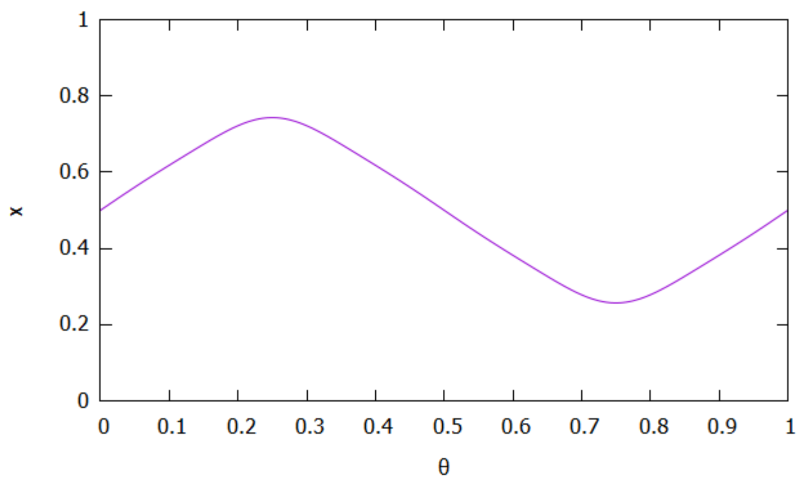



```
CN(rho, rhohat): [1.7151174562071982359681427214174e-11,1.7151174562071982359681427219777e-11]

Invariance error: [1.3364537996428360704009295355594e-10,1.3665348908480223800232028358809e-10]

Lambda: [3.5717463727666573959231070015372e-1,3.5717463727666860535348366102287e-1]

Reducibility Error: [7.5676212897829297173314739964981e-8,7.6011799392034107526664262791188e-8]

Inversion Error: [2.2885536863654524298845311645552e-8,2.2885536863663332134927968855888e-8]
```

```
b: [1.530380596619337265666681754768 2e1,1.8102967423659252483371854366236e1]
```

```
lambda + sigma + tau < 1

h: [4.9495706400868533523120604721764e-9,5.9866603745667573088995375987655e-9]

h < 1/2

r0: [1.7188735150489477615221158007997e-10,2.5712534272931502317776166704568e-10]

r0 < R

mu: [5.8291138190188813977146725006795e-8,5.9257244200849084012035154639138e-8]

mu < 1/(1+sqrt(2))


Congratulations, there exists a hyperbolic invariant torus with invariant subbundles.
Moreover, it is unique within a radius 9.9999999999999999999999999999998e-5
 and it is contained within a radius 2.5712534272931502317776166704567e-10.
 What’s more, the distance between approximately invariant bundles
and the invariant bundles is smaller than 5.9257251223692313033193228552948e-8
```

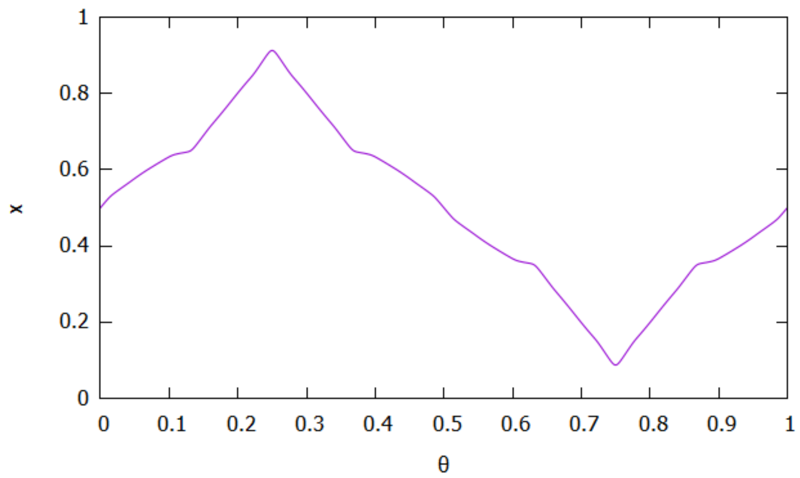
The validation still performs successfully. Let’s increase it a bit more, say $\varepsilon = 1$.




```
CN(rho, rhohat): [1.7151174562071982359681427214174e-11,1.7151174562071982359681427219777e-11]

Invariance error: [1.5226628748035801030337401028769e-10,1.6225121315164687442984533494377e-10]

Lambda: [4.4694956099061944525103200030983e-1,4.4694956099062804253455088253309e-1]

Reducibility Error: [7.0560283822576446595354775950299e-8,7.2199585063103034758875268336808e-8]

Inversion Error: [2.1552180291964324020285335997142e-8,2.1552180292020165461634090105779e-8]
```

```
b: [5.5354007416605775096064972160613e1,5.9500127532316678343831363521980e1]

lambda + sigma + tau < 1

h: [2.7556463974720913612903758798558e-8,3.1562871974132590898752909138025e-8]

h < 1/2

r0: [2.4037321413516475689098011366700e-10,3.3602865763938612939838946786470e-10]

r0 < R

mu: [7.3773242054655672534409686721587e-8,7.8424295237820443203941404235227e-8]

mu < 1/(1+sqrt(2))


Congratulations, there exists a hyperbolic invariant torus with invariant subbundles.
Moreover, it is unique within a radius 9.9999999999999999999999999999998e-5
 and it is contained within a radius 3.3602865763938612939838946786469e-10.
 What's more, the distance between approximately invariant bundles
and the invariant bundles is smaller than 7.8424307538563504332985962585107e-8
```

Notice how the torus is beginning to fractalize. This happens because ε is approaching a critical value in which the hyperbolicity property drops, and therefore it is harder to validate the torus. A deeper explanation on this phenomenon can be found in [11] or [4]. We can approach a bit more that critical value by taking $\varepsilon = 1.23$.




```
CN(rho, rhohat): [4.5730019875188309775879928534326e-24,4.5730019875188309775879928562778e-24]

Invariance error: [9.9512038569959792929174098340899e-9,7.2042284186602077935925951692173e204]

Lambda: [6.4652901822691745257909562140192e-1,2.1142393717449225605528340862315]
```

```
Reducibility Error: [2.4118312001358992063980495542665e11,3.0654258154845546997938811529493e239]

Inversion Error: [2.4118312001361417813578648336979e11,2.4118312001378599546280602547596e11]

b: [0,2.8294106485487984763006470336043e231]

lambda + sigma + tau >= 1, condition not satisfied

h: [0,8.7605152205044994410718512354246e412]

h >= 1/2, condition not satisfied

r0: [@NaN@,@NaN@]

r0 >= R, condition not satisfied

mu: [@NaN@,@NaN@]

mu >= 1/(1+sqrt(2)), condition not satisfied

The initial torus does not meet the requirements
to ensure the existence of an invariant torus
```

As we said before, the closer ε gets to the critical value $\varepsilon \sim 1.2342$, the harder it gets to validate the torus as seen in this last example. Even though ρ and $\hat{\rho}$ can be adjusted to obtain better results, it is still hard to find the sweet spot to validate the torus.
One can go even a bit further and try validate tori within a continuation method for ω or ε, or try figure out how the constants ρ, $\hat{\rho}$ and N depend on each other so one can choose optimally their values for a correct validation.

Appendix

Whiskers

In this appendix we want to show the bases of a possible future work encouraged by the validation process taken to term in the previous chapters. It is a topic that has been discussed during the research and writing processes but that was left as a possible future project should the first validation work succeed. Such topic is the study of the invariant manifolds known as whiskers.

When it comes to the study of quasi-periodically forced maps, there are two main interesting objects of study. One of them has been already addressed, and it is the existence of invariant tori under the quasi-periodic map. This has already been proven in the previous chapters through the validation theorem. The second one is the existence of asymptotic manifolds attached to our invariant torus. These manifolds, as we have already said, are usually called whiskers.

Since the proof of existence of such manifolds can be very complicated, we will present here just the main definitions and a computation algorithm for the easiest case, the one-dimensional and constant dynamics case. Keep in mind that our quasi-periodically forced map remains the same as before, where the dynamics on the torus is given by an ergodic rotation $\omega \in \mathbb{R}^d$. That is, $k \cdot \omega \notin \mathbb{Z}$, $\forall k \in \mathbb{Z}^d \backslash \{0\}$.

Equations for Whiskers

In order to study whiskers of rank m attached to a torus, we seek maps $W : \mathbb{C}^m \times \mathbb{T}^d_\rho \to \mathbb{C}^n$ and $\Lambda : \mathbb{C}^m \times \mathbb{T}^d_\rho \to \mathbb{C}^m$ in such a way that

$$F(W(\eta,\theta),\theta) = W(\Lambda(\eta,\theta),\theta+\omega)\ , \tag{4.1}$$

$$\Lambda(0,\theta) = 0\ . \tag{4.2}$$

Notice that equation (4.1) implies that the graph of W

$$\mathcal{W} = \{(W(\eta,\theta),\theta) \mid \eta \in \mathbb{C}^m,\ \theta \in \mathbb{T}^d_\rho\}$$

is an $m + d$ invariant manifold under $\hat{F} = (F, f)$. Recall that $f(\theta) = \theta + \omega$. It is useful to think of $\mathcal{W}$ as a d-parameter family of m-dimensional manifolds

$$\mathcal{W}_\theta = \{(W(\eta, \theta), \theta) \mid \eta \in \mathbb{C}^m\} \,,$$

with $\theta \in \mathbb{T}^d_\rho$. Notice also that the leaves $\mathcal{W}_\theta$ are not invariant, since $\hat{F}(\mathcal{W}_\theta) = \mathcal{W}_{\theta+\omega}$.

We note that Λ is part of the unknowns that need to be computed, and corresponds to finding a representation of the dynamics on the invariant manifold. Given a point η belonging to a leaf $\mathcal{W}_\theta$, Λ gives the dynamical displacement of such point on the same leaf.

Remark 4.4. Bear in mind that by choosing $K(\theta) = W(0, \theta)$ we obtain a K which satisfies the invariance equation, so the manifold $\mathcal{W}$ contains an invariant torus $\mathcal{K}$.

By taking derivatives on (4.1) with respect to η, and evaluating them at $\eta = 0$, we obtain the equation for the linearization of the whisker:

$$\mathrm{D}_x F(W(0, \theta), \theta)\mathrm{D}_\eta W(0, \theta) = \mathrm{D}_\eta W(0, \theta + \omega)\mathrm{D}_\eta \Lambda(0, \theta) \,.$$

Hence, we obtain that the vector space based at $K(\theta) = W(0, \theta)$ spanned by $W_1(\theta) = \mathrm{D}_\eta W(0, \theta)$ is mapped by $\mathrm{D}_x F(W(0, \theta), \theta)$ into the corresponding vector space based at $K(\theta + \omega) = W(0, \theta + \omega)$ spanned by $W_1(\theta + \omega) = \mathrm{D}_\eta W(0, \theta + \omega)$ through the linear map $\Lambda_1(\theta) = \mathrm{D}_\eta \Lambda(0, \theta)$ [5].

Summing up, we obtain that W_1 is an invariant subbundle of the cocycle. There may be, of course, several bundles invariant under linearization. For each of these invariant bundles, we can try to find an invariant manifold (whisker) tangent to it.

One-dimensional Setting and Reducible Case

As we previously said, the matter of whiskers can get very complicated, so we will simplify our work gradually. For starters, we will work with rank-1 whiskers. That will be our first step towards simplification. In that case we would be looking for a manifold $W : \mathbb{C} \times \mathbb{T}^d_\rho \to \mathbb{C}^n$ such that the invariance equation

$$F(W(s, \theta), \theta) = W(\lambda(\theta)s, \theta + \omega)$$

is satisfied, where $s = \eta$ is a one-dimensional parameter and $\lambda : \mathbb{T}^d_\rho \to \mathbb{C}$ a map such that $\exists\, \lambda_s < 1$, $C > 0$ which satisfy

$$|\lambda(\theta + (k-1)\omega) \cdots \lambda(\theta + \omega)\lambda(\theta)| \leq C \cdot \lambda_s^k \,. \tag{4.3}$$

The second step for simplicity is to assume that the internal dynamics on the whisker is given by a multiplication by a constant. As said in section 1.2.3, in this case, the system is reducible. We will also assume along this section that $\omega \in \mathbb{R}^d$ is Diophantine. The definition of a Diophantine number is given as follows.

Definition 4.5. *Given $\gamma > 0$ and $\tau \geq d$, we say that $\omega \in \mathbb{R}^d$ is a (γ, τ)-Diophantine vector of frequencies if*

$$|k \cdot \omega - m| \geq \frac{\gamma}{|k|_1^\tau}\,, \quad \forall k \in \mathbb{Z}^d \backslash \{0\},\, m \in \mathbb{Z},$$

where $|k|_1 = \sum_{i=1}^{n} |k_i|$.

This basically means that ω cannot be quickly approximated by rationals.

For our desired reduction, we have to express our $\lambda(\theta)$ as a constant in our frame, to get so, we look for a function $p : \mathbb{T}_\rho^d \to \mathbb{C}\backslash\{0\}$. Specifically, we will take it as a positive function when applied to real torus.

$$p(\theta+\omega)^{-1}\lambda(\theta)p(\theta) = \lambda_0 \longrightarrow \lambda(\theta)p(\theta) = p(\theta+\omega)\lambda_0\,.$$

Considering the possibility of $\lambda(\theta)$ being negative, we will take its absolute value when taking logarithms on both sides and $sign(\lambda(\theta)) = \pm 1$ as the sign of $\lambda(\theta)$.

$$\begin{aligned} \log|\lambda(\theta)| + \log p(\theta) &= \log p(\theta+\omega) + \log|\lambda_0| \\ -\log p(\theta) + \log p(\theta+\omega) &= \log|\lambda(\theta)| - \log|\lambda_0|\,. \end{aligned}$$

Taking $\xi(\theta) = \log p(\theta)$, $\eta(\theta) = \log|\lambda(\theta)|$ and $\eta_0 = \log|\lambda_0|$

$$\xi(\theta+\omega) - \xi(\theta) = \eta(\theta) - \eta_0\,. \tag{4.4}$$

Let's express $\xi(\theta)$ and $\eta(\theta)$ in terms of its Fourier series for a better manipulation of the expressions.

$$\begin{aligned} \xi(\theta) &= \sum_{k\in\mathbb{Z}} \hat{\xi}_k\, e^{2\pi i k\theta} \\ \eta(\theta) &= \sum_{k\in\mathbb{Z}} \hat{\eta}_k\, e^{2\pi i k\theta}\,. \end{aligned}$$

So, rewriting equation (4.4) in terms of Fourier series, we obtain

$$\sum_{k\in\mathbb{Z}} \left(\hat{\xi}_k\, e^{2\pi i k(\theta+\omega)} - \hat{\xi}_k\, e^{2\pi i k\theta}\right) = \sum_{k\neq 0} \hat{\xi}_k \left(e^{2\pi i k\omega} - 1\right)\, e^{2\pi i k\theta} = \left(\sum_{k\neq 0} \hat{\eta}_k\, e^{2\pi i k\theta}\right) + \hat{\eta}_0 - \eta_0\,.$$

Equating Fourier coefficients we get

$$\begin{aligned} k \neq 0\,, &\quad \hat{\xi}_k = \frac{\hat{\eta}_k}{e^{2\pi i k\omega} - 1} \\ k = 0\,, &\quad \hat{\eta}_0 = \eta_0\,. \end{aligned}$$

Notice that, thanks to the fact that ω is Diophantine ("very irrational") the denominator of the coefficients in the $k \neq 0$ case, will tend slower to 0, and therefore, the coefficients themselves will tend to infinity slower. With those expressions we can say now that $p(\theta) = \exp \xi(\theta)$ and

$$\eta_0 = \hat{\eta}_0 = \int_{\theta \in \mathbb{T}^d_\rho} \log|\lambda(\theta)| \, \mathrm{d}\theta \longrightarrow |\lambda_0| = \exp \int_{\theta \in \mathbb{T}^d_\rho} \log|\lambda(\theta)| \, \mathrm{d}\theta \, .$$

And therefore, $\lambda_0 = sign(\lambda(\theta)) \cdot |\lambda_0|$.

Computation of the Manifold

In this section we will specify an algorithm for the computation of whiskers of rank 1, which are parametrized by the map $W(s, \theta)$. Such map satisfies the invariance equation, assuming that the invariant bundles are of rank 1 and that ω is Diophantine. For the first calculations we will assume the linear dynamics are uniform on the bundles and given by a map $\lambda : \mathbb{T}^d_\rho \to \mathbb{C}$. Nonetheless, assuming the dynamics are reduced to multiplication by a constant can simplify the calculations if the linear dynamics is hyperbolic and if the previous section's conditions are fulfilled.
All in all, we are looking for a manifold $W : \mathbb{C} \times \mathbb{T}^d_\rho \to \mathbb{C}^n$ and a map $\lambda : \mathbb{T}^d_\rho \to \mathbb{C}$ satisfying condition (4.3), such that the invariance equation

$$F(W(s, \theta), \theta) = W(\lambda(\theta)s, \theta + \omega)$$

holds, with $W_0(\theta) = W(0, \theta) = K(\theta)$, that is, on the invariant torus. To do that, we will express $W(s, \theta)$ in Fourier-Taylor series form, such that

$$W(s, \theta) = \sum_{k=0}^{\infty} W_k(\theta) s^k \, , \tag{4.5}$$

where $W_k(\theta)$ are the Taylor coefficients of the series, which are also periodic functions (Fourier series) with respect to θ. Notice that we already know that $W_0(\theta) = K(\theta)$. By differentiating the invariance equation with respect to s and taking $s = 0$, we have

$$\mathrm{D}_x F(K(\theta), \theta) \mathrm{D}_s W(0, \theta) = \mathrm{D}_s W(0, \theta + \omega) \lambda(\theta) \, .$$

From here we get our second Taylor coefficient $W_1(\theta) = \mathrm{D}_s W(0, \theta)$, which is a frame of an invariant bundle, turning the previous equation into

$$\mathrm{D}_x F(K(\theta), \theta) W_1(\theta) = W_1(\theta + \omega) \lambda(\theta) \, .$$

At this point we will assume that we have calculated $W_{<k}(s, \theta) = \sum_{r<k} W_r(\theta) s^r$ and we want to calculate $W_k(\theta)$. Substituting expression (4.5) into the invariance equation we have

$$F(W_{<k}(s, \theta) + W_k(\theta)s^k + \cdots, \theta) = W_{<k}(\lambda(\theta)s, \theta + \omega) + W_k(\theta + \omega)\lambda(\theta)^k s^k + \cdots$$

Taylor expanding the left hand side of the equation we have

$$\begin{aligned} F(W_{<k}(s,\theta) + W_k(\theta)s^k + \cdots, \theta) &= F(W_{<k}(s,\theta),\theta) + \mathrm{D}_x F(W_{<k}(s,\theta),\theta)(W_k(\theta)s^k + \cdots) + \cdots \\ &= [F(W_{<k}(s,\theta),\theta)]_{<k} + [F(W_{<k}(s,\theta),\theta)]_k \, s^k + \cdots \\ &\quad + \mathrm{D}_x F(K(\theta),\theta)\, W_k(\theta)s^k + \cdots \end{aligned}$$

Recalling that $\mathrm{D}_x F(K(\theta),\theta) = M(\theta)$ and equating terms of the same order, we obtain

$$M(\theta)W_k(\theta)s^k - W_k(\theta+\omega)\lambda(\theta)^k s^k = -\left[F(W_{<k}(s,\theta),\theta)\right]_k s^k \, .$$

Thus we obtain the cohomolgy equation

$$M(\theta)W_k(\theta) - W_k(\theta+\omega)\lambda(\theta)^k = -\left[F(W_{<k}(s,\theta),\theta)\right]_k \, , \quad k \geq 2 \, .$$

Keep in mind that, for this equation to be solvable, the condition for

$$\lambda_0 = \exp \int_{\theta \in \mathbb{T}^d_\rho} \log|\lambda(\theta)| \, \mathrm{d}\theta$$

to $\lambda_0^l \notin Spec(\mathcal{M}_\omega)$ for $l \geq 2$ must be satisfied (so $\mathcal{M}_\omega - \lambda_0^l I$ is invertible), where $\mathcal{M}_\omega$ is the transfer operator defined in Section 1.2.2. With that final expression we are now able to find the k-th term of the expansion, allowing us to fully calculate the object.
Notice that in order to prove the existence of $W(s,\theta)$, we should prove first that indeed the fact that $\lambda_0^l \notin Spec(\mathcal{M}_\omega)$ for $l \geq 2$ ensures the existence of solution of the cohomology equation. And secondly, we should prove that the series defining $W(s,\theta)$ converges.
Such proof would require a lot of preparation work and, as said, it goes beyond the scope of this project. However, it is a very interesting topic worth considering for a future work, both of a theoretical nature and of a more validation focused approach.

Conclusions

Certainly, skew-product systems are not unexplored territory although there is still a lot to learn. There are even several ways of proving the validation theorem even though we just have stated one. Nevertheless, although the approach that has been taken here has already been used for validations in KAM theory, the implementation to this very specific case can draw a path for other different and better computer assisted proofs to come. In the same way, regardless of the commonness and familiarity with Fourier transforms, the validation process using intervalar arithmetics and multi-precision can teach a lot about scientific coding that it might not be possible to learn somewhere else. The precision and care with which every computation has to be performed can be overwhelming at times but also very rewarding professional-wise.

As it has been already stated, this is just the beginning for what computer validations can become, by using its tools in other areas such as PDE's. But without leaving the topic of skew-product systems, we have also seen that we can move forward and see, for instance, how can whiskers be validated using similar procedures. As always in science and of course in mathematics, there is still a lot to progress and learn.

Annex

```
#include <stdio.h>
#include <math.h>
#include <complex.h>
#include <stdlib.h>
#include <mpfi.h>
#include <mpfi_io.h>

typedef struct real {
  mpfi_t real;
}real;

typedef struct complexi {
  mpfi_t real;
  mpfi_t imag;
}complexi;

real PI, DPI, one, two;
int prec = 100;
double rhod = 1.e-3, hatrhod = 5.e-3;

void comp_print (complexi x) {
  /* Prints complex intervals */
  mpfi_out_str(stdout, 10, 0, x.real);
  printf(" x ");
  mpfi_out_str(stdout, 10, 0, x.imag);
  printf("\n\n");
}

void real_print (real x) {
  /* Prints real intervals */
  mpfi_out_str(stdout, 10, 0, x.real);
  printf("\n\n");
}

complexi comp_init_c (complex x) {
  /* Initializes a complex interval around a complex number */
  complexi z;
  mpfi_init2(z.real, prec);
  mpfi_init2(z.imag, prec);
  mpfi_set_d(z.real, creal(x));
  mpfi_set_d(z.imag, cimag(x));
```

```

  return z;
}

complexi comp_set_c (complex x) {
  /* Sets a complex interval around a complex number */
  complexi z;

  z = comp_init_c(0);

  mpfi_set_d(z.real, creal(x));
  mpfi_set_d(z.imag, cimag(x));

  return z;
}

void comp_clear (complexi x) {
  /* Clears complex intervals */
  mpfi_clear(x.real);
  mpfi_clear(x.imag);
}

void allocm (complexi *m[2][2], unsigned N) {
  /* Allocates a matrix of complex intervals */
    int i, j;
    for(i=0; i<2; i++)
        for(j=0; j<2; j++)
            m[i][j] = (complexi *) malloc(N*sizeof(complexi));
}

void allocv (complexi *v[2], unsigned N) {
  /* Allocates a vector of complex intervals */
    int i;
    for(i=0; i<2; i++)
        v[i] = (complexi *) malloc(N*sizeof(complexi));
}

void allocv_d (double *v[2], unsigned N) {
  /* Allocates a vector of real numbers */
        int i;
        for(i=0; i<2; i++)
                v[i] = (double *) malloc(N*sizeof(double));
}

void allocv_real (real *v[2], unsigned N) {
        /* Allocates a vector of real numbers */
        int i;
        for(i=0; i<2; i++)
                v[i] = (real *) malloc(N*sizeof(real));
}


```

```
void allocm_d (double *m[2][2], unsigned N) {
        /* Allocates a matrix of complex intervals */
        int i, j;
        for(i=0; i<2; i++)
                for(j=0; j<2; j++)
                        m[i][j] = (double *) malloc(N*sizeof(double));
}

void allocv_c (complex *v[2], unsigned N) {
  /* Allocates a vector of complex numbers */
        int i;
        for(i=0; i<2; i++)
                v[i] = (complex *) malloc(N*sizeof(complex));
}

void freev (complexi *v[2]) {
  /* Frees the memory occupied by a vector */
    int i;
    for(i=0; i<2; i++)
          free(v[i]);
}

void freem (complexi *m[2][2]) {
  /* Frees the memory occupied by a matrix */
    int i, j;
    for(i=0; i<2; i++)
        for(j=0; j<2; j++)
            free(m[i][j]);
}

void comp_init_m (complexi *m[2][2], unsigned N) {
  /* Initializes a matrix of complex intervals */
  int i, j, k;
  for(k=0; k<N; k++)
    for(i=0; i<2; i++)
      for(j=0; j<2; j++)
        m[i][j][k] = comp_init_c(0);
}

void comp_init_v (complexi *v[2], unsigned N) {
  /* Initializes a vector of complex intervals */
        int i, k;
        for(k=0; k<N; k++)
                for(i=0; i<2; i++)
                    v[i][k] = comp_init_c(0);
}

void real_init_v (real *v[2], unsigned N) {
        /* Initializes a vector of complex intervals */
        int i, k;
        for(k=0; k<N; k++)
                for(i=0; i<2; i++) {
```

```
      mpfi_init2(v[i][k].real, prec);
      mpfi_set_d(v[i][k].real, 0);
    }
}

complexi comp_add (complexi x, complexi y) {
  /* Sum of complex intervals */
  complexi z;

  z = comp_init_c(0);

  mpfi_add(z.real, x.real, y.real);
  mpfi_add(z.imag, x.imag, y.imag);

  return z;
}

real real_add (real x, real y) {
  /* Sum of real intervals */
  real sum;

  mpfi_init2(sum.real, prec);

  mpfi_add(sum.real, x.real, y.real);

  return sum;
}

complexi comp_sub (complexi x, complexi y) {
  /* Substraction of complex intervals */
        complexi z;

        z = comp_init_c(0);

  mpfi_sub(z.real, x.real, y.real);
        mpfi_sub(z.imag, x.imag, y.imag);

  return z;
}

real real_sub (real x, real y) {
  /* Substraction of real intervals */
  real sub;

  mpfi_init2(sub.real, prec);

  mpfi_sub(sub.real, x.real, y.real);

  return sub;
}

complexi comp_mul (complexi x, complexi y) {
```

```
  /* Product of complex intervals */
  complexi z, r;

  z = comp_init_c(0);
        r = comp_init_c(0);

        mpfi_mul(z.real, x.real, y.real);
  mpfi_mul(z.imag, x.imag, y.imag);
  mpfi_sub(r.real, z.real, z.imag);

        mpfi_mul(z.real, x.real, y.imag);
        mpfi_mul(z.imag, x.imag, y.real);
        mpfi_add(r.imag, z.real, z.imag);

  return r;
}

real real_mul (real x, real y) {
  /* Multiplies real intervals */
  real mul;

  mpfi_init2(mul.real, prec);

  mpfi_mul(mul.real, x.real, y.real);

  return mul;
}

real real_sc_mul (double a, real x) {
  /* Multiplies a real interval by a real scalar */
  real mul;

  mpfi_init2(mul.real, prec);

  mpfi_mul_d(mul.real, x.real, a);

  return mul;
}

real real_sc_div (double a, real x) {
  /* Divides a real interval by a real scalar */
  real div;

  mpfi_init2(div.real, prec);

  mpfi_d_div(div.real, a, x.real);

  return div;
}

complexi comp_div (complexi x, complexi y) {
  /* Division of complex intervals */
```

```
        complexi z1, r;
    real z2;

    z1 = comp_init_c(0);
    r = comp_init_c(0);
    mpfi_init2(z2.real, prec);

    /*Real numerator*/
        mpfi_mul(z1.real, x.real, y.real);
        mpfi_mul(z1.imag, x.imag, y.imag);
        mpfi_add(r.real, z1.real, z1.imag);

    /*Denominator*/
    mpfi_mul(z1.real, y.real, y.real);
    mpfi_mul(z1.imag, y.imag, y.imag);
    mpfi_add(z2.real, z1.real, z1.imag);

    /*Division of real part*/
    mpfi_div(r.real, r.real, z2.real);

    /*Imaginary numerator*/
        mpfi_mul(z1.real, x.real, y.imag);
        mpfi_mul(z1.imag, x.imag, y.real);
        mpfi_sub(r.imag, z1.imag, z1.real);

    /*Division of imagaginary part*/
    mpfi_div(r.imag, r.imag, z2.real);

        return r;
}

real real_div (real x, real y) {
    /* Divides real intervals */
    real div;

    mpfi_init2(div.real, prec);

    mpfi_div(div.real, x.real, y.real);

    return div;
}

real real_sqrt (real x) {
    /* Calculates the square root of a real interval */
    real sqrt;

    mpfi_init2(sqrt.real, prec);
    mpfi_sqrt(sqrt.real, x.real);

    return sqrt;
}
```

```c
complexi comp_reali_mul (real a, complexi x) {
  /* Multiplies a real interval by a complex interval as a scalar */
  complexi z;

  z = comp_init_c(0);

  mpfi_mul(z.real, a.real, x.real);
  mpfi_mul(z.imag, a.real, x.imag);

  return z;
}

complexi comp_sc_mul (double a, complexi x) {
  /* Multiplies a complex interval by a real scalar */
  complexi z;
  real y;

  z = comp_init_c(0);

  mpfi_init2(y.real, prec);
  mpfi_set_d(y.real, a);

  mpfi_mul(z.real, y.real, x.real);
  mpfi_mul(z.imag, y.real, x.imag);

  mpfi_clear(y.real);

  return z;
}

complexi comp_abs (complexi x) {
  /* Computes the absolute value of a complex interval interval-wise */
  /* Caution: this is not the modulus of a complex interval */
  complexi abs;

  abs = comp_init_c(0);

  mpfi_abs(abs.real, x.real);
  mpfi_abs(abs.imag, x.imag);

  return abs;
}

real real_abs (real x) {
  /* Calculates the absolute value of a real interval */
  real abs;

  mpfi_init2(abs.real, prec);

  mpfi_abs(abs.real, x.real);

  return abs;
```

```
}

real comp_mod (complexi z) {
  /* Calculates the modulus of a complex interval */
  real mod;

  mpfi_init2(mod.real, prec);

  mpfi_hypot(mod.real, z.real, z.imag);

  return mod;
}

complexi comp_sin (complexi x) {
  /* Computes complex sine */
        complexi z;
        real a, b;

        z = comp_init_c(0);

        mpfi_init2(a.real, prec);
        mpfi_init2(b.real, prec);

        mpfi_sin(a.real, x.real);
        mpfi_cosh(b.real, x.imag);
        mpfi_mul(z.real, a.real, b.real);

        mpfi_cos(a.real, x.real);
        mpfi_sinh(b.real, x.imag);
        mpfi_mul(z.imag, a.real, b.real);

  mpfi_clear(a.real);
  mpfi_clear(b.real);

        return z;
}

complexi comp_cos (complexi x) {
  /* Computes complex cosine */
  complexi z;
        real a, b, one;

  mpfi_init2(one.real, prec);
  mpfi_set_d(one.real, -1);
        z = comp_init_c(0);

        mpfi_init2(a.real, prec);
        mpfi_init2(b.real, prec);

        mpfi_cos(a.real, x.real);
        mpfi_cosh(b.real, x.imag);
        mpfi_mul(z.real, a.real, b.real);
```

```

        mpfi_sin(a.real, x.real);
        mpfi_sinh(b.real, x.imag);
  mpfi_mul(b.real, one.real, b.real);
        mpfi_mul(z.imag, a.real, b.real);

        mpfi_clear(a.real);
        mpfi_clear(b.real);
  mpfi_clear(one.real);

        return z;
}

real real_sin (real x) {
  /* Calculates the sine of a real interval */
  real sine;

  mpfi_init2(sine.real, prec);

  mpfi_sin(sine.real, x.real);

  return sine;
}

real real_cos (real x) {
  /* Calculates the cosine of a real interval */
  real cosine;

  mpfi_init2(cosine.real, prec);

  mpfi_cos(cosine.real, x.real);

  return cosine;
}

void matrixmult (complexi *z[2][2], complexi *x[2][2], complexi *y[2][2],
    int N) {
  /* Multiplies two matrices */
    int i, j, k, l;
    complexi p[2][2];
  for(i=0; i<2; i++)
    for(j=0; j<2; j++)
      p[i][j] = comp_init_c(0);

    for(k=0; k<N; k++) {
        for(i=0; i<2; i++) {
            for(j=0; j<2; j++) {
                                p[i][j] = comp_set_c(0);
                                    for(l=0; l<2; l++){
                                      p[i][j] = comp_add(p[i][j], comp_mul(x[i
                                          ][l][k], y[l][j][k]));
                                    }
```

```
                }
            }
            for(i= 0; i<2; i++)
                for(j= 0; j<2; j++)
                                        z[i][j][k] = p[i][j];
        }
}

void matrixvecmul (complexi *r[2], complexi *x[2][2], complexi *v[2], int N) {
    /* Multiplies a matrix by a vector */
        int i, j, k;
        complexi p[2];
            for(i=0; i<2; i++)
                  p[i] = comp_init_c(0);

        for(k=0; k<N; k++) {
            for(i=0; i<2; i++) {
                p[i] = comp_set_c(0);
                for(j=0; j<2; j++) {
                                        p[i] = comp_add(p[i], comp_mul(x[i][j][k], v[j][k]));
                }
            }
            for(i=0; i<2; i++) {
                r[i][k] = p[i];
            }
        }
}

void inverse (complexi *inv[2][2], complexi *a[2][2], int N) {
            /* Inverts a matrix */
        int k;
        complexi det, adj[2][2];

    det = comp_init_c(0);
    adj[0][0] = comp_init_c(0);
    adj[0][1] = comp_init_c(0);
    adj[1][0] = comp_init_c(0);
    adj[1][1] = comp_init_c(0);

        for(k=0; k<N; k++) {
        det = comp_sub(comp_mul(a[0][0][k], a[1][1][k]), comp_mul(a[0][1][k], a[1][0][k]));
            adj[0][0] = a[1][1][k];
            adj[1][1] = a[0][0][k];
            adj[0][1] = comp_sc_mul(-1, a[1][0][k]);
            adj[1][0] = comp_sc_mul(-1, a[0][1][k]);
            inv[0][0][k] = comp_div(adj[0][0], det);
            inv[1][1][k] = comp_div(adj[1][1], det);
            inv[1][0][k] = comp_div(adj[0][1], det);
            inv[0][1][k] = comp_div(adj[1][0], det);
```

```
    }
}

complexi comp_exp (real x) {
  /* Calculates the complex exponential */
  complexi compexp;

  compexp = comp_init_c(0);

  mpfi_cos(compexp.real, x.real);
  mpfi_sin(compexp.imag, x.real);

  return compexp;
}

real real_exp (real x) {
  /* Calculates the real exponential */
  real exp;

  mpfi_init2(exp.real, prec);

  mpfi_exp(exp.real, x.real);

  return exp;
}

void dft (complexi *coef, complexi *grid, int N) {
  /* Discrete Fourier Transform */
    int j, k;
    complexi sum, compexp;
  compexp = comp_init_c(0);
    for(k=0; k<N; k++){
    sum = comp_init_c(0.);
        for(j=0; j<N; j++) {
      compexp = comp_exp(real_sc_mul(-2.0*k*j/N, PI)); /* Note: -2*k*j/N is perfectly represented by the computer, but the product by PI is not */
            sum = comp_add(sum, comp_mul(grid[j], compexp));
        }
        coef[k] = comp_sc_mul(1./N, sum); /* Note: 1/N is perfectly represented by the computer */
    comp_clear(sum);
    }
  comp_clear(compexp);
}

void idft (complexi *grid, complexi *coef, int N) {
    /* Inverse Discrete Fourier Transform */
  int j, k;
    complexi sum, compexp;
  compexp = comp_init_c(0);
    for(k=0; k<N; k++){
```

```
            sum = comp_init_c(0.);
            for(j=N/2; j<N; j++){
        compexp = comp_exp(real_sc_mul(2.0*k*j/N, PI));
        sum = comp_add(sum, comp_mul(coef[j], compexp));
        compexp = comp_exp(real_sc_mul(2.0*k*(N-1-j)/N, PI));
        sum = comp_add(sum, comp_mul(coef[N-1-j], compexp));
          }
          grid[k] = sum;
      comp_clear(sum);
      }
    comp_clear(compexp);
}

void separate (complexi *a, int n) {
          /* Copies all even elements to lower-half of a[]
          and all odd elements to upper-half of a[] */
      complexi b[n/2];
      int i;
      for(i=0; i<n/2; i++) {
      b[i] = comp_init_c(0);
      b[i] = a[i*2+1];
    }
      for(i=0; i<n/2; i++)
          a[i] = a[i*2];
      for(i=0; i<n/2; i++) {
          a[i+n/2] = b[i];
    }
}

void _fft (complexi *X, int N) {
    /* Fast Fourier Transform */
          int k;
          complexi e, o, w;

    e = comp_init_c(0);
    o = comp_init_c(0);
    w = comp_init_c(0);

      if(N<2){

      }else{
          separate(X, N);
          _fft(X, N/2);
          _fft(X+N/2, N/2);
          for(k=0; k<N/2; k++) {
              e = X[k];
              o = X[k+N/2];
        w = comp_exp(real_sc_mul((-2.0*k)/N, PI));
        X[k] = comp_add(comp_reali_mul(real_div(one, two), e), comp_mul(w,
            comp_reali_mul(real_div(one, two), o)));
        X[k+N/2] = comp_sub(comp_reali_mul(real_div(one, two), e), comp_mul(w
            , comp_reali_mul(real_div(one, two), o)));
```

```
        }
    }
}

void fft (complexi *coef, complexi *grid, int N) {
  /* Copies vectors and applies Fast Fourier Transform */
    int k;
    for (k=0; k<N; k++){
          coef[k] = grid[k];
    }
    _fft(coef, N);
}

void _ifft (complexi *X, int N){
  /* Inverse Fast Fourier Transform */
        int k;
        complexi e, o, w;

  e = comp_init_c(0);
        o = comp_init_c(0);
        w = comp_init_c(0);

    if(N<2){

  }else{
        separate(X, N);
        _ifft(X, N/2);
        _ifft(X+N/2, N/2);
        for(k=0; k<N/2; k++) {
            e = X[k];
            o = X[k+N/2];
      w = comp_exp(real_sc_mul((2.0*k)/N, PI));
            X[k] = comp_add(e, comp_mul(w, o));
      X[k+N/2] = comp_sub(e, comp_mul(w, o));
        }
    }
}

void ifft (complexi *grid, complexi *coef, int N){
  /* Copies vectors and applies inverse Fast Fourier Transform */
    int k;
    for(k=0; k<N; k++){
          grid[k] = coef[k];
    }
    _ifft(grid, N);
}

real real_sup (real x, real y) {
  /* Finds the supremum of two real intervals */
        real sup1, sup2;
  mpfr_t sup1r, sup2r;
  int n;
```

```

  mpfr_init2(sup1r, prec);
  mpfr_init2(sup2r, prec);

        mpfi_init2(sup1.real, prec);
        mpfi_init2(sup2.real, prec);

        sup1 = real_abs(x);
        sup2 = real_abs(y);

  mpfi_get_right(sup1r, sup1.real);
  mpfi_get_right(sup2r, sup2.real);

  n = mpfr_cmp(sup1r, sup2r);

        if(n < 0) {
                return sup2;
        } else {
                return sup1;
        }
}

void F (complexi *FK[2], complexi *K[2], real b, real e,  int N) {
        /* Calculates the imagage through the Standard Map */
    int j;
  real sin;
  mpfi_init2(sin.real, prec);
  complexi sine;
  sine = comp_init_c(0);
    for(j=0; j<N; j++){
    sin = real_mul(e, real_sin(real_sc_mul((1.*j)/N, DPI)));
    mpfi_set(sine.real, sin.real);
    FK[1][j] = comp_sub(K[1][j], comp_add(comp_reali_mul(real_div(b, DPI),
        comp_sin(comp_reali_mul(DPI, K[0][j]))), sine));
                FK[0][j] = comp_add(K[0][j], FK[1][j]);
    }
}

void Fbox (complexi *FK[2], complexi *K[2], real b, real e,  int N) {
        /* Calculates the imagage through the Standard Map */
        int j;
  complexi sine, theta[N];

  for(j=0; j<N; j++) {
    theta[j] = comp_init_c(0);
    mpfi_interv_d(theta[j].real, (1.*j)/N-1./(2*N), (1.*j)/N+1./(2*N));
    mpfi_interv_d(theta[j].imag, -hatrhod, hatrhod);
  }

        sine = comp_init_c(0);

        for(j=0; j<N; j++){
```

```
                sine = comp_reali_mul(e, comp_sin(comp_reali_mul(DPI, theta[j])));
                FK[1][j] = comp_sub(K[1][j], comp_add(comp_reali_mul(real_div(b, DPI), comp_sin(comp_reali_mul(DPI, K[0][j]))), sine));
                FK[0][j] = comp_add(K[0][j], FK[1][j]);
        }
}


void difmatrix(complexi *dif[2][2], complexi *K[2], real b, int N){
        /* Evaluates the differential matrix of the Standard Map over K */
    int k;
    for(k=0; k<N; k++){
        dif[0][0][k] = comp_sub(comp_init_c(1), comp_reali_mul(b, comp_cos(comp_reali_mul(DPI, K[0][k]))));
        dif[0][1][k] = comp_init_c(1);
        dif[1][1][k] = comp_init_c(1);
        dif[1][0][k] = comp_reali_mul(real_sc_mul(-1, b), comp_cos(comp_reali_mul(real_sc_mul(2.0, PI), K[0][k])));
    }
}

real diff2norm (complexi *K[2], real b, int N){
        /* Evaluates the second differential matrix of the Standard Map over K */
        int k;
  real norm, sup;
  mpfi_init2(norm.real, prec);
  mpfi_init2(sup.real, prec);
  mpfi_set_d(sup.real, 0);
        for(k=0; k<N; k++){
                norm = comp_mod(comp_reali_mul(real_mul(DPI, b), comp_sin(comp_reali_mul(DPI, K[0][k]))));
                sup = real_sup(sup, norm);
        }
  return sup;
}

real mu (real delta, int N) {
  real a, b;
  mpfi_init2(a.real, prec);
  mpfi_init2(b.real, prec);
  a = real_sc_mul(2.0, real_exp(real_mul(PI, delta)));
  b = real_add(real_exp(real_sc_mul(2.0, real_mul(PI, delta))), one);
  if(N%2 == 0) {
    return one;
  } else {
    return real_div(a, b);
  }
}
```

```
real CN (real rho, real hat, int N) {
  /* Calculation of C_N Fourier error bound */
  real S1, S2, T, a, b, c;

  mpfi_init(S1.real);
  mpfi_init(S2.real);
  mpfi_init(T.real);
  mpfi_init(a.real);
  mpfi_init(b.real);
  mpfi_init(c.real);

  a = real_div(real_exp(real_sc_mul(-2.0*N, real_mul(PI, hat))), real_sub(
      one, real_exp(real_sc_mul(-2.0*N, real_mul(PI, hat)))));
  b = real_div(real_add(real_exp(real_sc_mul(-2.0, real_mul(PI, real_add(
      hat, rho)))), one), real_sub(real_exp(real_sc_mul(-2.0, real_mul(PI,
      real_add(hat, rho)))), one));
  c = real_sub(one, real_mul(mu(real_sub(real_sc_mul(-1, hat), rho), N),
      real_exp(real_sc_mul(N, real_mul(PI, real_add(hat, rho))))));
  S1 = real_mul(a, b);
  S1 = real_mul(S1, c);

  b = real_div(real_add(real_exp(real_sc_mul(2.0, real_mul(PI, real_sub(hat
      , rho)))), one), real_sub(real_exp(real_sc_mul(2.0, real_mul(PI,
      real_sub(hat, rho)))), one));
  c = real_sub(one, real_mul(mu(real_sub(real_sc_mul(1, hat), rho), N),
      real_exp(real_sc_mul(-N, real_mul(PI, real_sub(hat, rho))))));
  S2 = real_mul(a, b);
  S2 = real_mul(S2, c);

  c = real_mul(mu(real_sub(real_sc_mul(1, hat), rho), N), real_exp(
      real_sc_mul(-N, real_mul(PI, real_sub(hat, rho)))));
  T = real_mul(b, c);

  return real_add(S1, real_add(S2, T));
}

void comp_add_v (complexi *s, complexi *x, complexi *y, int N) {
  /* Sum of vectors */
    int k;
    for(k=0; k<N; k++)
        s[k]= comp_add(x[k], y[k]);
}

void comp_sub_v (complexi *r, complexi *x, complexi *y, int N) {
  /* Substraction of vectors */
        int k;
        for(k=0; k<N; k++)
                r[k]= comp_sub(x[k], y[k]);
}

void comp_mul_v (complexi *m, complexi *x, complexi *y, int N) {
  /* Product of vectors component-wise */
```

```
        int k;
        for(k=0; k<N; k++)
                m[k]= comp_mul(x[k], y[k]);
}

void fourierrot(complexi *xrot, complexi *x, real om, int N){
        /* Rotates the Fourier coefficients */
    int k;
  complexi compexp;
  compexp = comp_init_c(0);
    for(k=0; k<N/2; k++) {
    compexp = comp_exp(real_sc_mul(k, real_mul(DPI, om)));
    xrot[k] = comp_mul(x[k], compexp);
  }
    for(k=N/2; k<N; k++) {
    compexp = comp_exp(real_sc_mul((k-N), real_mul(DPI, om)));
    xrot[k] = comp_mul(x[k], compexp);
  }
}

void fourierrot_c_m (complexi *frotx[2][2], complexi *fx[2][2], complexi
    phi, int N) {
  /* Rotates a matrix by a complex factor */
        real aux;
  complexi *mat[2][2];
        int i;

  allocm(mat, N);

  comp_init_m(mat, N);

        mpfi_init2(aux.real, prec);

        mpfi_mul(aux.real, DPI.real, phi.imag);

        for(i=0; i<N; i++) {
                if(i < N/2) {
                        mat[0][0][i] = comp_reali_mul(real_div(one,
                            real_exp(real_sc_mul(i, aux))), fx[0][0][i]);
                        mat[0][1][i] = comp_reali_mul(real_div(one,
                            real_exp(real_sc_mul(i, aux))), fx[0][1][i]);
                        mat[1][0][i] = comp_reali_mul(real_div(one,
                            real_exp(real_sc_mul(i, aux))), fx[1][0][i]);
                        mat[1][1][i] = comp_reali_mul(real_div(one,
                            real_exp(real_sc_mul(i, aux))), fx[1][1][i]);
                } else {
                        mat[0][0][i] = comp_reali_mul(real_div(one,
                            real_exp(real_sc_mul((i-N), aux))), fx[0][0][i
                            ]);
                        mat[0][1][i] = comp_reali_mul(real_div(one,
                            real_exp(real_sc_mul((i-N), aux))), fx[0][1][i
                            ]);
```

```
                mat[1][0][i] = comp_reali_mul(real_div(one, real_exp(real_sc_mul((i-N), aux))), fx[1][0][i]);
                mat[1][1][i] = comp_reali_mul(real_div(one, real_exp(real_sc_mul((i-N), aux))), fx[1][1][i]);

            }
        }

        mpfi_set(aux.real, phi.real);

        fourierrot(frotx[0][0], mat[0][0], aux, N);
        fourierrot(frotx[0][1], mat[0][1], aux, N);
        fourierrot(frotx[1][0], mat[1][0], aux, N);
        fourierrot(frotx[1][1], mat[1][1], aux, N);
}

void fourierrot_c_v (complexi *frotx[2], complexi *fx[2], complexi phi, int N) {
  /* Rotates a vector by a complex factor */
        real aux;
  complexi *vec[2];
        int i;

  allocv(vec, N);

  comp_init_v(vec, N);

        mpfi_init2(aux.real, prec);

        mpfi_mul(aux.real, DPI.real, phi.imag);

        for(i=0; i<N; i++) {
                if(i < N/2) {
                        vec[0][i] = comp_reali_mul(real_div(one, real_exp(real_sc_mul(i, aux))), fx[0][i]);
                        vec[1][i] = comp_reali_mul(real_div(one, real_exp(real_sc_mul(i, aux))), fx[1][i]);
                } else {
                        vec[0][i] = comp_reali_mul(real_div(one, real_exp(real_sc_mul((i-N), aux))), fx[0][i]);
                        vec[1][i] = comp_reali_mul(real_div(one, real_exp(real_sc_mul((i-N), aux))), fx[1][i]);

                }
        }

        mpfi_set(aux.real, phi.real);

        fourierrot(frotx[0], vec[0], aux, N);
        fourierrot(frotx[1], vec[1], aux, N);
```

```

}


void matrixgf (complexi *coef[2][2], complexi *grid[2][2], int N) {
        /* Transforms an array of matrices evaluated over a grid into
            matrices of Fourier coefficients */
    fft(coef[0][0], grid[0][0], N);
    fft(coef[0][1], grid[0][1], N);
    fft(coef[1][0], grid[1][0], N);
    fft(coef[1][1], grid[1][1], N);
}

void matrixfg (complexi *grid[2][2], complexi *coef[2][2], int N) {
        /* Transforms an array of matrices evaluated over a grid into
            matrices of Fourier coefficients */
        ifft(grid[0][0], coef[0][0], N);
        ifft(grid[0][1], coef[0][1], N);
        ifft(grid[1][0], coef[1][0], N);
        ifft(grid[1][1], coef[1][1], N);
}

real fournorm (complexi *coef, real rho, int N) {
  real sum;
  int i;

  mpfi_init2(sum.real, prec);
  mpfi_set_d(sum.real, 0);

  for(i=0; i<N; i++) {
    if(i<N/2){
      sum = real_add(sum, real_mul(real_exp(real_sc_mul(2.0l*fabs(i),
          real_mul(PI, rho))), comp_mod(coef[i])));
    } else {
      sum = real_add(sum, real_mul(real_exp(real_sc_mul(2.0l*fabs(i-N),
          real_mul(PI, rho))), comp_mod(coef[i])));
    }
  }

  return sum;
}

real fournorm_v (complexi *x[2], real rho, int N) {
  real a, b;
  mpfi_init2(a.real, prec);
  mpfi_init2(b.real, prec);
  a = real_abs(fournorm(x[0], rho, N));
  b = real_abs(fournorm(x[1], rho, N));

        if(a.real <= b.real) {
                return b;
        } else {
```

```
                return a;
        }
}

real fournorm_m (complexi *x[2][2], real rho, int N) {
  real sum1, sum2;

  mpfi_init2(sum1.real, prec);
  mpfi_init2(sum2.real, prec);

  sum1 = real_add(fournorm(x[0][0], rho, N), fournorm(x[0][1], rho, N));
    sum2 = real_add(fournorm(x[1][0], rho, N), fournorm(x[1][1], rho, N));

  return real_sup(sum1, sum2);
}

real supnorm (complexi *x, int N) {
  int i, n;
  mpfr_t supr, modr;
  real sup, mod;

  mpfr_init2(supr, prec);
        mpfr_init2(modr, prec);

  mpfi_init2(sup.real, prec);
  mpfi_set_d(sup.real, 0);
  mpfi_init2(mod.real, prec);

  for(i=0; i<N; i++) {
    mod = comp_mod(x[i]);

    mpfi_get_right(supr, sup.real);
          mpfi_get_right(modr, mod.real);

    n = mpfr_cmp(supr, modr);
    if(n < 0) {
      mpfi_set(sup.real, mod.real);
    }
  }
  return sup;
}

real supnorm_v (complexi *x[2], int N) {
  real sup1, sup2;
  int n;
  mpfr_t sup1r, sup2r;

  mpfi_init2(sup1.real, prec);
  mpfi_init2(sup2.real, prec);
  mpfr_init2(sup1r, prec);
  mpfr_init2(sup2r, prec);

```

```
  sup1 = supnorm(x[0], N);
  sup2 = supnorm(x[1], N);

  mpfi_get_right(sup1r, sup1.real);
  mpfi_get_right(sup2r, sup2.real);

  n = mpfr_cmp(sup1r, sup2r);

  if(n < 0) {
    return sup2;
  } else {
    return sup1;
  }
}

real supnorm_m (complexi *x[2][2], int N) {
  real sum1, sum2;

  mpfi_init2(sum1.real, prec);
        mpfi_init2(sum2.real, prec);

        sum1 = real_add(supnorm(x[0][0], N), supnorm(x[0][1], N));
        sum2 = real_add(supnorm(x[1][0], N), supnorm(x[1][1], N));

  return real_sup(sum1, sum2);
}

real comp_sup (complexi x, complexi y) {
  real sup1, sup2;

  mpfi_init2(sup1.real, prec);
  mpfi_init2(sup2.real, prec);

  sup1 = comp_mod(x);
  sup2 = comp_mod(y);

  if(sup1.real <= sup2.real) {
    return sup2;
  } else {
    return sup1;
  }
}

int main(){
  complexi *K[2], *diff[2][2], *fdiff[2][2], *frotdiff[2][2], *FK[2], *fFK
      [2], *fK[2], *frotK[2], *P1[2][2], *fP1[2][2], *frotP1[2][2], *fP2
      [2][2], *frotP2[2][2], *P2[2][2],
     *Lam[2][2], *fLam[2][2], *vec[2], *auxc[2], *fcopy[2][2], *Id[2][2], *
         fI[2][2], *xi[2], phi;
  int i, n, N, valh, valr0, valmu, vall;
  double *Kc[2], lam0, lam1, *P0[2][2], R = 1.e-4;
  FILE *torus;
```

```
real b, e, om, rho, rhohat, inverr, rederr, finverr, fP1norm, fP2norm,
    fdiffnorm, tildeeps, sum[2], *aux[2],
      lambda, lams, lamu, lamuinv, lambdap, term1, term2, term3, five,
          kappa, hateps, beta, h, r0, r1, mu, Rinterv, hatlambda, alpha;
mpfr_t bound1, bound2;

/* Initializes 1, 2 and 2*PI */
mpfi_init2(one.real, prec);
mpfi_set_str(one.real, "1", 10);

mpfi_init2(two.real, prec);
mpfi_set_str(two.real, "2", 10);

      mpfi_init2(PI.real, prec);
      mpfi_atan(PI.real, one.real);

      mpfi_mul(PI.real, PI.real, two.real);
      mpfi_mul(PI.real, PI.real, two.real);

mpfi_init2(DPI.real, prec);
mpfi_mul(DPI.real, PI.real, two.real);

/* Reads N */
  torus = fopen("K0.100000.txt", "r");
  if (!torus) {
        puts("File Error");
  }

fscanf(torus, "%d", &N);
fclose(torus);

/* Allocates vectors */
allocv(K, N);
allocv(vec, N);
allocv(auxc, N);
allocv(fK, N);
allocv(frotK, N);
allocv(xi, N);
allocv(fFK, N);
allocv_real(aux, N);
allocv_d(Kc, N);
allocm_d(P0, N);
allocv(FK, N);
allocm(P1, N);
allocm(fP1, N);
allocm(fP2, N);
allocm(frotP2, N);
allocm(frotP1, N);
allocm(P2, N);
allocm(diff, N);
allocm(fdiff, N);
allocm(frotdiff, N);
```

```
  allocm(Lam, N);
  allocm(fLam, N);
  allocm(fcopy, N);
  allocm(Id, N);
  allocm(fI, N);

  /* Reads grid input and initializes complex intervals */
  torus = fopen("K0.100000.txt", "r");
  for(i=0; i<N; i++) {
    fscanf(torus, "%*d %*f %lf %lf %lf %lf %lf %lf %lf %lf %*f %*f", &Kc
        [0][i], &Kc[1][i], &P0[0][0][i], &P0[0][1][i], &P0[1][0][i], &P0
        [1][1][i], &lam0, &lam1);

    K[0][i] = comp_init_c(Kc[0][i]);
    K[1][i] = comp_init_c(Kc[1][i]);

    P1[0][0][i] = comp_init_c(P0[0][0][i]);
        P1[0][1][i] = comp_init_c(P0[0][1][i]);
        P1[1][0][i] = comp_init_c(P0[1][0][i]);
        P1[1][1][i] = comp_init_c(P0[1][1][i]);

    Lam[0][0][i] = comp_init_c(lam0);
      Lam[1][1][i] = comp_init_c(lam1);
    Lam[0][1][i] = comp_init_c(0);
    Lam[1][0][i] = comp_init_c(0);

    Id[0][0][i] = comp_init_c(1);
                Id[1][1][i] = comp_init_c(1);
                Id[0][1][i] = comp_init_c(0);
                Id[1][0][i] = comp_init_c(0);

  }

  /* Initialize complex and real intervals */
  comp_init_v(FK, N);
  comp_init_v(fK, N);
  comp_init_v(vec, N);
  comp_init_v(auxc, N);
  comp_init_v(frotK, N);
  comp_init_v(fFK, N);
  real_init_v(aux, N);
  comp_init_m(fP1, N);
  comp_init_m(P2, N);
  comp_init_m(frotP2, N);
  comp_init_m(frotP1, N);
  comp_init_m(fP2, N);
  comp_init_m(diff, N);
  comp_init_m(fdiff, N);
  comp_init_m(frotdiff, N);
  comp_init_m(fLam, N);
  comp_init_m(fcopy, N);
  comp_init_m(fI, N);
```

```
comp_init_v(xi, N);
phi = comp_init_c(0);

mpfi_init2(b.real, prec);

mpfi_init2(kappa.real, prec);
mpfi_set_str(kappa.real, "1.3", 10);

mpfi_init2(e.real, prec);
      mpfi_set_str(e.real, "0.10", 10);

mpfi_init2(five.real, prec);
mpfi_set_str(five.real, "5", 10);
mpfi_init2(om.real, prec);
om = real_div(real_sub(real_sqrt(five), one), two);

mpfi_init2(rho.real, prec);
      mpfi_set_str(rho.real, "1.e-3", 10);

mpfi_init2(rhohat.real, prec);
      mpfi_set_str(rhohat.real, "1.e-2", 10);

mpfi_init2(sum[0].real, prec);
mpfi_set_d(sum[0].real, 0);

mpfi_init2(sum[1].real, prec);
mpfi_set_d(sum[1].real, 0);

mpfi_init2(inverr.real, prec);
mpfi_init2(rederr.real, prec);
mpfi_init2(finverr.real, prec);
mpfi_init2(fP1norm.real, prec);
mpfi_init2(fP2norm.real, prec);
mpfi_init2(fdiffnorm.real, prec);
mpfi_init2(tildeeps.real, prec);
mpfi_init2(lams.real, prec);
mpfi_init2(lamu.real, prec);
mpfi_init2(lambdap.real, prec);
mpfi_init2(lamuinv.real, prec);
mpfi_init2(lambda.real, prec);
mpfi_init2(hateps.real, prec);
mpfi_init2(beta.real, prec);
mpfi_init2(h.real, prec);
mpfi_init2(r0.real, prec);
mpfi_init2(r1.real, prec);
mpfi_init2(mu.real, prec);
mpfi_init2(term1.real, prec);
mpfi_init2(term2.real, prec);
mpfi_init2(term3.real, prec);
mpfi_init2(Rinterv.real, prec);
mpfi_set_str(Rinterv.real, "1.e-4", 10);
mpfi_init2(hatlambda.real, prec);
```

```
mpfi_init2(alpha.real, prec);

mpfr_init2(bound1, prec);
mpfr_init2(bound2, prec);


/* Invariance Error */

/* The inputs of the algorithm are in Fourier space, but given our input
    is in grid space
 * we will first transform them into Fourier series */

/* Since the eigenvalues are swapped, we change them back */
for(i=0; i<N; i++) {
  fcopy[0][0][i] = Lam[0][0][i];
  Lam[0][0][i] = Lam[1][1][i];
  Lam[1][1][i] = fcopy[0][0][i];
}

matrixgf(fP1, P1, N);
matrixgf(fLam, Lam, N);

fft(fK[0], K[0], N);
      fft(fK[1], K[1], N);

/* We must set the Nyquist term to 0 */
fP1[0][0][N/2] = comp_set_c(0);
fP1[0][1][N/2] = comp_set_c(0);
fP1[1][0][N/2] = comp_set_c(0);
fP1[1][1][N/2] = comp_set_c(0);

fLam[0][0][N/2] = comp_set_c(0);
fLam[0][1][N/2] = comp_set_c(0);
fLam[1][0][N/2] = comp_set_c(0);
fLam[1][1][N/2] = comp_set_c(0);

fK[0][N/2] = comp_set_c(0);
fK[1][N/2] = comp_set_c(0);


/* We will take as the first P2 the inverse of P1, for that we will have
    to come back
 * to grid space and invert, since P2 is the approximate inverse by
     hypothesis, we
 * won't mind the inversion error produced by Fourier transforming */
matrixfg(P1, fP1, N);
inverse(P2, P1, N);
matrixgf(fP2, P2, N);

fP2[0][0][N/2] = comp_set_c(0);
      fP2[0][1][N/2] = comp_set_c(0);
      fP2[1][0][N/2] = comp_set_c(0);
```

```
        fP2[1][1][N/2] = comp_set_c(0);

    /* Rotate K and P2 in Fourier space */
    fourierrot(frotK[0], fK[0], om, N);
    fourierrot(frotK[1], fK[1], om, N);

    fourierrot(frotP2[0][0], fP2[0][0], om, N);
    fourierrot(frotP2[0][1], fP2[0][1], om, N);
    fourierrot(frotP2[1][0], fP2[1][0], om, N);
    fourierrot(frotP2[1][1], fP2[1][1], om, N);

    /* In order to operate on the grid, we must use our Fourier objects and
       transform them */
    ifft(K[0], fK[0], N);
    ifft(K[1], fK[1], N);

    F(FK, K, kappa, e, N);

    /* Save non-inflated K for next step */
    difmatrix(diff, K, kappa, N);

    fft(fFK[0], FK[0], N);
        fft(fFK[1], FK[1], N);

    for(i=0; i<N; i++) {
      auxc[0][i] = comp_sub(fFK[0][i], frotK[0][i]);
      auxc[1][i] = comp_sub(fFK[1][i], frotK[1][i]);
    }

    tildeeps = fournorm_v(auxc, rho, N);

    fP2norm = fournorm_m(frotP2, rho, N);

    /* Inflate K so we can calculate the norm of FK */
    mpfi_interv_d(phi.real, (double) -1/(2*N), (double) 1/(2*N));
        mpfi_interv_d(phi.imag, -hatrhod, hatrhod);

    fourierrot_c_v(frotK, fK, phi, N);

    ifft(auxc[0], frotK[0], N);
    ifft(auxc[1], frotK[1], N);

    /* Calculate the image through the standard map using complex boxes */
        Fbox(FK, auxc, kappa, e, N);

    term1 = supnorm_v(FK, N);

    inverr = real_add(real_mul(CN(rho, rhohat, N), term1), tildeeps);

    inverr = real_mul(fP2norm, inverr);

    printf("CN(rho, rhohat): ");
```

```
  real_print(CN(rho, rhohat, N));

  printf("Invariance error: ");
  real_print(inverr);

  /* Pick lambda so the norm of the stable block of Lamda and the norm of
     the inverse of the unstable
   * block are bounded by such lambda */
  for(i=0; i<N; i++) {
    fcopy[0][0][i] = fLam[0][0][i];
    fcopy[0][1][i] = fLam[0][1][i];
    fcopy[1][0][i] = fLam[1][0][i];
    fcopy[1][1][i] = fLam[1][1][i];
  }

  lams = fournorm(fLam[0][0], rho, N);
  lamu = fournorm(fLam[1][1], rhohat, N);
  matrixfg(Lam, fLam, N);
  inverse(fcopy, Lam, N);
  matrixgf(fcopy, fcopy, N);
  lamuinv = fournorm(fcopy[1][1], rhohat, N);
  lambda = lams;

  lambdap = real_add(real_div(real_mul(CN(rho, rhohat, N), real_mul(lamu,
     real_mul(lamuinv, lamuinv))), real_sub(one, real_mul(CN(rho, rhohat,
     N), real_mul(lamu, lamuinv)))),lamuinv);

  mpfi_get_right(bound1, lambdap.real);
  mpfi_get_right(bound2, lambda.real);
  n = mpfr_cmp(bound1, bound2);
  if(n > 0) {
    lambda = lambdap;
  }

  printf("Lambda: ");
  real_print(lambda);

  /* Reducibility error */
  for(i=0; i<N; i++) {
              fcopy[0][0][i] = Lam[0][0][i];
              Lam[0][0][i] = Lam[1][1][i];
              Lam[1][1][i] = fcopy[0][0][i];
        }

  /* Move to grid space to multiply the differential and P1 and come back
     to Fourier space */
  matrixgf(fLam, Lam, N);
        matrixfg(P2, frotP2, N);
        matrixmult(fcopy, diff, P1, N);
        matrixmult(fcopy, P2, fcopy, N);
        matrixgf(fcopy, fcopy, N);

```

```
        for(i=0; i<N; i++) {
                fcopy[0][0][i] = comp_sub(fcopy[0][0][i], fLam[0][0][i]);
                fcopy[0][1][i] = comp_sub(fcopy[0][1][i], fLam[0][1][i]);
                fcopy[1][0][i] = comp_sub(fcopy[1][0][i], fLam[1][0][i]);
                fcopy[1][1][i] = comp_sub(fcopy[1][1][i], fLam[1][1][i]);
        }

        term1 = fournorm_m(fcopy, rho, N);

  /* Calculate differential with inflated torus with hatrho */
  difmatrix(diff, auxc, kappa, N);

  fP2norm = fournorm_m(frotP2, rhohat, N);
  fP1norm = fournorm_m(fP1, rhohat, N);
  fdiffnorm = supnorm_m(diff, N);

  term2 = real_mul(CN(rho, rhohat, N), real_mul(fP2norm, real_mul(fdiffnorm
      , fP1norm)));

  rederr = real_add(term1, term2);

  printf("Reducibility Error: ");
  real_print(rederr);

  /* Inversion Error */
  inverse(P2, P1, N);
  matrixmult(fcopy, P2, P1, N);
  matrixgf(fcopy, fcopy, N);
  matrixgf(fI, Id, N);

  for(i=0; i<N; i++) {
                fcopy[0][0][i] = comp_sub(fcopy[0][0][i], fI[0][0][i]);
                fcopy[1][1][i] = comp_sub(fcopy[1][1][i], fI[1][1][i]);
        }

  term1 = real_mul(CN(rho, rhohat, N), real_mul(fournorm_m(fP2, rhohat, N),
       fournorm_m(fP1, rhohat, N)));

  term2 = fournorm_m(fcopy, rho, N);

  finverr = real_add(term1, term2);

  printf("Inversion Error: ");
  real_print(finverr);

  /* Norm of B */
  for(i=0; i<N; i++) {
    mpfi_interv_d(xi[0][i].real, -R, R);
    mpfi_interv_d(xi[0][i].imag, -R, R);
    mpfi_interv_d(xi[1][i].real, -R, R);
    mpfi_interv_d(xi[1][i].imag, -R, R);
  }
```

```c

/* Extend each theta by 1/2N the real part, and rho the imaginary part */
mpfi_interv_d(phi.real, (double) -1/(2*N), (double) 1/(2*N));
mpfi_interv_d(phi.imag, -rhod, rhod);

fourierrot_c_v(frotK, fK, phi, N);

fourierrot_c_m(frotP1, fP1, phi, N);

/* Calculate z */
matrixfg(fcopy, frotP1, N);
matrixvecmul(vec, fcopy, xi, N);
comp_add_v(auxc[0], vec[0], auxc[0], N);
    comp_add_v(auxc[1], vec[1], auxc[1], N);

/* Calculate the norm of the second differential */
term2 = diff2norm(auxc, kappa, N);

matrixgf(fP2, P2, N);
fourierrot(frotP2[0][0], fP2[0][0], om, N);
      fourierrot(frotP2[0][1], fP2[0][1], om, N);
      fourierrot(frotP2[1][0], fP2[1][0], om, N);
      fourierrot(frotP2[1][1], fP2[1][1], om, N);
fP2norm = fournorm_m(frotP2, rho, N);

matrixgf(fP1, P1, N);
fP1norm = fournorm_m(fP1, rho, N);

b = real_mul(fP2norm, real_mul(term2, real_mul(fP1norm, fP1norm)));
printf("b: ");
real_print(b);

/* Calculation of constants */
term1 = real_add(lambda, real_add(rederr, finverr));
mpfi_get_right(bound1, term1.real);
      mpfi_get_left(bound2, one.real);
      n = mpfr_cmp(bound1, bound2);
if(n < 0) {
  printf("lambda + sigma + tau < 1\n\n");
  vall = 1;
} else {
  printf("lambda + sigma + tau >= 1, condition not satisfied\n\n");
}

hateps = real_div(inverr, real_sub(one, real_add(lambda, real_add(rederr,
     finverr))));

beta = real_div(b, real_sub(one, real_add(lambda, real_add(rederr,
    finverr))));

h = real_mul(hateps, beta);

```

```
  printf("h: ");
  real_print(h);

  mpfi_get_right(bound1, h.real);
  term1 = real_div(one, two);
  mpfi_get_left(bound2, term1.real);
  n = mpfr_cmp(bound1, bound2);
  if(n < 0) {
    printf("h < 1/2\n\n");
    valh = 1;
  } else {
    printf("h >= 1/2, condition not satisfied\n\n");
  }

  r0 = real_mul(real_div(real_sub(one, real_sqrt(real_sub(one, real_mul(two, h)))), h), hateps);

  printf("r0: ");
  real_print(r0);
  mpfi_get_right(bound1, r0.real);
  mpfi_get_left(bound2, term1.real);
  n = mpfr_cmp(bound1, bound2);
        if(n < 0) {
                printf("r0 < R\n\n");
                valr0 = 1;
        } else {
    printf("r0 >= R, condition not satisfied\n\n");
  }

  hatlambda = supnorm_m(Lam, N);
  mu = real_mul(real_div(lambda, real_sub(one, real_mul(lambda, lambda))), real_mul(real_div(one, real_sub(one, finverr)), real_add(real_mul(b, r0), real_add(rederr, real_mul(hatlambda, finverr)))));
  printf("mu: ");
  real_print(mu);

  mpfi_get_right(bound1, mu.real);
  term1 = real_div(one, real_add(one, real_sqrt(two)));
        mpfi_get_left(bound2, term1.real);
        n = mpfr_cmp(bound1, bound2);
        if(n < 0) {
                printf("mu < 1/(1+sqrt(2))\n\n");
                valmu = 1;
        } else {
    printf("mu >= 1/(1+sqrt(2)), condition not satisfied\n\n");
  }

  if((vall == 1) && (valh ==1) && (valr0 == 1) && (valmu == 1)) {
    printf("\nCongratulations, there exists a hyperbolic invariant torus with invariant subbundles. \nMoreover, it is unique within a radius ");
```

```
    r1 = real_mul(real_div(real_add(one, real_sqrt(real_sub(one, real_mul(two, h)))), h), hateps);
    mpfi_get_left(bound1, r1.real);
    mpfi_get_left(bound2, Rinterv.real);
    n = mpfr_cmp(bound1, bound2);
    if(n <  0) {
      mpfr_out_str(stdout, 10, 0, bound1, MPFR_RNDD);
    } else {
      mpfr_out_str(stdout, 10, 0, bound2, MPFR_RNDD);
    }
    printf("\n and it is contained within a radius ");
    mpfi_get_right(bound1, r0.real);
    mpfr_out_str(stdout, 10, 0, bound1, MPFR_RNDD);
    printf(".\n What's more, the distance between approximately invariant bundles \nand the invariant bundles is smaller than ");

    term1 = real_sub(one, real_mul(two, mu));

    term2 = real_sqrt(real_sub(real_sub(one, real_mul(real_mul(two, two), mu)), real_mul(real_mul(two, two), real_mul(mu, mu))));

    term3 = real_mul(two, mu);

    alpha = real_div(term3, real_add(term1, term2));

    mpfi_get_right(bound1, alpha.real);

    mpfr_out_str(stdout, 10, 0, bound1, MPFR_RNDD);

    printf("\n");
  } else {
    printf("The initial torus does not meet the requirements \nto ensure the existence of an invariant torus\n");
  }

  freev(K);
        freev(vec);
        freev(auxc);
        freev(fK);
        freev(frotK);
        freev(xi);
        freev(fFK);
  freev(FK);
  freem(P1);
        freem(fP1);
        freem(fP2);
        freem(frotP2);
        freem(frotP1);
        freem(P2);
        freem(diff);
        freem(fdiff);
        freem(frotdiff);
```

```
        freem(Lam);
        freem(fLam);
        freem(fcopy);
        freem(Id);
        freem(fI);

  return 0;
}
```

Bibliography

[1] Marta Canadell and Àlex Haro. Computation of quasiperiodic normally hyperbolic invariant tori: rigorous results. *Journal of Nonlinear Science*, 27(6):1869–1904, 2017.

[2] Rafael De La Llave and Rafael Obaya. Regularity of the composition operator in spaces of hölder functions. *Discrete and Continuous Dynamical Systems*, 5:157–184, 1999.

[3] J-Ll Figueras, Alex Haro, and Alejandro Luque. Rigorous computer-assisted application of kam theory: a modern approach. *Foundations of Computational Mathematics*, 17(5):1123–1193, 2017.

[4] Alex Haro, Marta Canadell, Jordi-Lluis Figueras, Alejandro Luque, and Josep-Maria Mondelo. The parameterization method for invariant manifolds. *Appl. Math. Sci*, 195, 2016.

[5] Alex Haro and Rafael de la Llave. A parameterization method for the computation of invariant tori and their whiskers in quasi-periodic maps: numerical algorithms. *Discrete and Continuous Dynamical Systems Series B*, 6(6):1261, 2006.

[6] Dale Husemoller. *Fibre bundles*, volume 5. Springer, 1966.

[7] Jake VanderPlas. Understanding the FFT Algorithm. `https://jakevdp.github.io/blog/2013/08/28/understanding-the-fft/`. [Accessed November 26, 2019].

[8] Ronald T Kneusel. *Numbers and computers*, volume 2. Springer, 2017.

[9] Rowland, Todd. "Fiber Bundle." From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/FiberBundle.html`. [Accessed October 5, 2019].

[10] Rowland, Todd. "Vector Bundle." From MathWorld–A Wolfram Web Resource, created by Eric W. Weisstein. `http://mathworld.wolfram.com/FiberBundle.html`. [Accessed October 5, 2019].

[11] Eric Sandín Vidal. Quasi-periodic solutions in quasi-periodic systems via fourier transforms. *http://hdl.handle.net/2445/127600*, 2018.

[12] Silly rabbit at the English Wikipedia. Bundle section. `https://en.wikipedia.org/wiki/File:Bundle_section.svg`. [Accessed October 5, 2019].

[13] Dirk Werner. *Funktionalanalysis*. Springer, 2006.